





KENDE MÁRIA–NAGY ISTVÁN

# **ORACLE példatár**

## **SQL, PL/SQL**

PANEM

Copyright © 2005 Panem Könyvkiadó

ISBN 963 545 436 8

Lektor: Juhász István

Műszaki szerkesztő: Papp Gyula

Tördelte: Pipaszó Bt.

Borítóterv: Édelkraut Róbert

A kiadásért felel a Panem Könyvkiadó Kft. ügyvezetője, 2005

[panem@panem.hu](mailto:panem@panem.hu)

[www.panem.hu](http://www.panem.hu)

A könyv a kiadó és a szerzők gondos munkájának eredménye. Ennek ellenére előfordulhatnak benne hibák, melyeknek következményeiért sem a szerzők, sem a kiadó nem vállalnak felelősséget.

Minden jog fenntartva. Jelen könyvet, illetve annak részeit tilos reprodukálni, adatrögzítő rendszerben tárolni, bármilyen formában vagy eszközzel – elektronikus úton vagy más módon – közzölni a kiadó engedélye nélkül.

# Tartalom

Előszó .....	15
<b>I. rész. FELADATGYŰJTEMÉNY .....</b>	<b>17</b>
1. fejezet. Egyszerű lekérdezések.....	19
2. fejezet. Egytáblás csoportosító lekérdezések .....	49
3. fejezet. Többtáblás lekérdezések, allekérdezések.....	56
4. fejezet. Interaktív környezet .....	71
5. fejezet. Adattáblák létrehozása, módosítása, tranzakciók, megszorítások.....	90
6. fejezet. Nézettáblák, felső-N analízis, ROWNUM .....	111
7. fejezet. Összetett SQL-feladatsorok .....	120
8. fejezet. Változóhasználat, vezérlési szerkezetek a PL/SQL-ben .....	142
9. fejezet. Hivatkozási és összetett adattípusok, kurzor, ROWID .....	159
10. fejezet. Kivételkezelés, alprogramok, triggerek.....	176
11. fejezet. Adatbázis-adminisztrátori ismeretek.....	204
12. fejezet. Kapcsolat a külvilággal .....	235
13. fejezet. Részletező csoportosítások és analitikus függvények az SQL-ben .....	253
14. fejezet. A DEMO vállalat (üzleti alkalmazások) .....	297
<b>II. rész. MEGOLDÁSCYŰJTEMÉNY .....</b>	<b>301</b>
1. fejezet. Egyszerű lekérdezések.....	303
2. fejezet. Egytáblás csoportosító lekérdezések .....	321
3. fejezet. Többtáblás lekérdezések, allekérdezések.....	336
4. fejezet. Interaktív környezet .....	373
5. fejezet. Adattáblák létrehozása, módosítása, tranzakciók, megszorítások.....	396
6. fejezet. Nézettáblák, felső-N analízis, ROWNUM .....	444
7. fejezet. Összetett SQL-feladatsorok .....	491
8. fejezet. Változóhasználat, vezérlési szerkezetek a PL/SQL-ben .....	547
9. fejezet. Hivatkozási és összetett adattípusok, kurzor, ROWID .....	556
10. fejezet. Kivételkezelés, alprogramok, triggerek.....	616

11. fejezet. Adatbázis-adminisztrátori ismeretek.....	644
12. fejezet. Kapcsolat a külvilággal .....	654
13. fejezet. Részletező csoportosítások és analitikus függvények az SQL-ben .....	655
14. fejezet. A DEMO vállalat (üzleti alkalmazások) .....	656
<b>MELLÉKLETEK .....</b>	<b>691</b>
1. melléklet. A Personal Oracle telepítése.....	693
2. melléklet. Az Oracle alaptáblái.....	694
3. melléklet. Bevezetés az SQL*Plus-környezet és az SQL nyelv használatába .....	698
4. melléklet. Elveszett függőségek megvalósítása triggerrel .....	709
5. melléklet. Gyakran használatos adatszótárnézetek.....	720
<b>A CD-mellékletről.....</b>	<b>728</b>
<b>Irodalomjegyzék .....</b>	<b>730</b>
<b>Tárgymutató .....</b>	<b>733</b>

# Tartalomjegyzék

Előszó .....	15
<b>I. rész</b>	
<b>FELADATGYŰJTEMÉNY</b> .....	17
Bevezetés.....	17
Néhány megjegyzés az elméleti összefoglalókhöz .....	17
Az utasítások szintaktikus jelölése .....	18
<b>1. fejezet. Egyszerű lekérdezések</b> .....	19
Elméleti összefoglaló .....	19
A SELECT utasítás .....	19
SzelekciósLista .....	19
TáblaLista .....	20
LogikaiOszlopkifejezés .....	20
CsoportosítóOszlopkifejezés-lista.....	20
RendezőOszlopkifejezés-lista .....	21
Megjegyzések a SELECT utasításhoz .....	21
Speciális függvények .....	23
NVL függvény .....	23
DECODE függvény.....	24
CASE kifejezés .....	24
Karakterkezelő függvények .....	26
Dátumok és számok formázott megjelenítése.....	35
Rendszerdátum és -idő formátumának beállítása .....	37
Dátum és idő megjelenítése és használata .....	37
Számok megjelenítése .....	45
Feladatok .....	46
<b>2. fejezet. Egytáblás csoportosító lekérdezések</b> .....	49
Elméleti összefoglaló .....	49
Csoportképzés.....	49
Csoportfüggvények .....	50
Feladatok .....	53

<b>3. fejezet. Többtáblás lekérdezések, allekérdezések.....</b>	56
Elméleti összefoglaló .....	56
Táblák összekapcsolása.....	56
Táblák összekapcsolása a FROM és WHERE utasításrészben .....	56
Táblák összekapcsolása a FROM utasításrészben a JOIN segítségével.....	57
Hierarchikus adatszerkezet megjelenítése.....	62
Allekérdezések .....	66
Allekérdezések a FROM utasításrészben .....	66
Allekérdezések a WHERE utasításrészben.....	66
Allekérdezések a szelekciós listában .....	67
Feladatok .....	67
Összetett feladatok.....	69
<b>4. fejezet. Interaktív környezet.....</b>	71
Elméleti összefoglaló .....	71
Felhasználói (helyettesítő) változók használata.....	71
A felhasználói változók hivatkozási jelei (& és &&) .....	71
DEFINE utasítás .....	72
ACCEPT utasítás .....	72
PROMPT utasítás .....	73
UNDEFINE utasítás .....	73
Az SQL*Plus-környezet beállítása .....	74
A környezet lekérdezése.....	74
A környezet beállítása.....	74
Az SQL*Plus-utasítások lekérdezése (HELP) .....	75
Az SQL*Plus formázási utasításai .....	78
Szkript programok felhasználóvezérelt hívása .....	85
Feladatok .....	87
<b>5. fejezet. Adattáblák létrehozása, módosítása, tranzakciók, megszorítások .....</b>	90
Elméleti összefoglaló .....	90
DML-utasítások.....	90
Beszúrás.....	90
Módosítás .....	91
Törlés.....	91
Adatbázis-tranzakciók.....	91
DDL-utasítások .....	92
Tábla létrehozása (közvetlen létrehozás) .....	92
Tábla létrehozása allekérdezéssel (származtatott létrehozás) .....	92
Tábla módosítása .....	93
Integritási megszorítások.....	94
Feladatok .....	107

<b>6. fejezet. Nézettáblák, felső-N analízis, ROWNUM.....</b>	111
Elméleti összefoglaló .....	111
Nézettábla létrehozása és törlése .....	111
Inline nézet .....	113
Felső-N analízis .....	113
Feladatok .....	116
<b>7. fejezet. Összetett SQL-feladatsorok .....</b>	120
Elméleti összefoglaló .....	120
Mintafeladatsor .....	120
Mintafeladatsor és megoldása .....	121
Feladatok .....	135
A. feladatsor .....	135
B. feladatsor .....	136
C. feladatsor .....	137
D. feladatsor .....	137
E. feladatsor.....	138
F. feladatsor .....	138
G. feladatsor.....	139
H. feladatsor.....	139
I. feladatsor .....	140
J. feladatsor.....	141
<b>8. fejezet. Változóhasználat, vezérlési szerkezetek a PL/SQL-ben.....</b>	142
Elméleti összefoglaló .....	142
PL/SQL blokk .....	142
Deklarációs szegmens .....	143
Végrehajtható szegmens.....	143
Kivételkezelő szegmens .....	143
A blokkok szerkezete .....	143
Változók használata .....	144
I. Az SQL*Plus felhasználói változói.....	144
II. Az SQL*Plus környezeti változói.....	144
III. A PL/SQL változói .....	145
A változóhasználat szemléltetése .....	145
Vezérlési szerkezetek, utasítások .....	149
A SELECT utasítás a PL/SQL-ben .....	149
Feltételes utasítás.....	149
Ciklusutasítások.....	150
Adatok kiírása képernyőre .....	151
Véletlen számok generálása .....	155
Feladatok .....	157

<b>9. fejezet. Hivatkozási és összetett adattípusok, kurzor, ROWID .....</b>	159
Elméleti összefoglaló .....	159
Hivatkozási adattípus.....	159
Rekord adattípus .....	160
Gyűjtőtábla típus .....	160
Kurzorok .....	161
Explicit kurzor .....	161
Kurzorhasználat FOR-ciklusban (rejtett kurzorok) .....	162
Kurzorattribútumok .....	163
ROWID .....	164
FOR UPDATE záradék.....	164
Feladatok .....	168
Összetett (párosítási) feladatok .....	170
<b>10. fejezet. Kivételkezelés, alprogramok, triggerek.....</b>	176
Elméleti összefoglaló .....	176
Kivételkezelés (EXCEPTION) .....	176
A kivételek fajtái.....	176
Alprogramok .....	177
Paraméterek adattípusai .....	178
Tárolt alprogramok .....	181
Tárolt alprogramok létrehozása.....	181
Tárolt alprogramok törlése.....	182
Tárolt alprogramok hibakezelése .....	182
Tárolt alprogramok lekérdezése.....	183
Tárolt eljárás futtatása.....	184
Triggerek.....	186
Triggerindító események.....	186
A triggerek alkalmazása .....	186
A triggerek felépítése és lekérdezése.....	186
A tárolt alprogramok és triggerek lekérdezése .....	187
A triggerek jellemzői.....	187
A triggerek szintaktikus alakja .....	187
A triggerek működésének jellegzetességei.....	188
A korrelációs nevek használata .....	189
Trigger törlése .....	189
Azonos eseményre tüzelő triggerek.....	189
A triggerek letiltása és engedélyezése.....	190
A triggerelő esemény meghatározása .....	191
Feladatok .....	202
<b>11. fejezet. Adatbázis-adminisztrátori ismeretek.....</b>	204
Elméleti összefoglaló .....	204
Jogosultság (jog, privilegium).....	204

Rendszerjogosultságok.....	205
Objektumkezelési jogosultságok .....	206
Jogosultságok lekérdezése.....	207
Táblaterület .....	207
Táblaterület létrehozása .....	208
Táblaterületek lekérdezése .....	208
Tábla létrehozása adott táblaterületen .....	208
Táblaterület módosítása.....	209
Táblaterület törlése.....	209
Felhasználó (USER).....	209
Felhasználó létrehozása .....	210
Felhasználó törlése .....	211
Felhasználó kapcsolódása az adatbázishoz.....	211
Szerepkör (ROLE) .....	211
Szerepkör létrehozása .....	211
Szerepkör törlése .....	211
Példák.....	212
Feladatok .....	233
<b>12. fejezet. Kapcsolat a külvilággal .....</b>	<b>235</b>
Elméleti összefoglaló .....	235
Programkapcsolat – külső programok futtatása .....	235
Kapcsolat az operációs rendszerrel .....	235
Tárolt eljárások hívása az SQL*Plus-környezetben .....	240
Adatkapcsolat – mentés, betöltés .....	240
Teljes adatbázis-terület és egyes táblaterületek archiválása fájlba .....	240
Táblák exportálása, importálása Oracle-eszközökkel .....	243
Tábla tartalmának exportálása fájlba .....	244
Tábla tartalmának betöltése Wordbe .....	247
Tábla tartalmának betöltése Excelbe .....	248
<b>13. fejezet. Részletező csoportosítások és analitikus függvények az SQL-ben.....</b>	<b>253</b>
Elméleti összefoglaló .....	253
Az SQL új csoportképzési eszközei .....	253
A ROLLUP operátor .....	253
A CUBE operátor .....	257
A GROUPING függvény.....	260
A GROUPING_ID függvény.....	263
A GROUPING SETS függvény.....	265
Az SQL analitikus függvényei .....	266
Az analitikus függvények általános felépítése.....	267
Rang függvények .....	269
Statisztikai függvények.....	282
Hisztogramfüggvények .....	292

<b>14. fejezet. A DEMO vállalat (üzleti alkalmazások).....</b>	297
A DEMO vállalat felépítése .....	297
Feladatok .....	298
<b>II. rész</b>	
<b>MEGOLDÁSGYŰJTEMÉNY.....</b>	301
Bevezetés.....	301
<b>1. fejezet. Egyszerű lekérdezések .....</b>	303
Feladatok és megoldások.....	303
<b>2. fejezet. Egytáblás csoportosító lekérdezések .....</b>	321
Feladatok és megoldások.....	321
<b>3. fejezet. Többtáblás lekérdezések, allekérdezések.....</b>	336
Feladatok és megoldások.....	336
Összetett feladatok és megoldások .....	353
<b>4. fejezet. Interaktív környezet .....</b>	373
Feladatok és megoldások.....	373
<b>5. fejezet. Adattáblák létrehozása, módosítása, tranzakciók, megszorítások .....</b>	396
Feladatok és megoldások.....	396
<b>6. fejezet. Nézettáblák, felső-N analízis, ROWNUM.....</b>	444
Feladatok és megoldások.....	444
<b>7. fejezet. Összetett SQL-feladatsorok .....</b>	491
Feladatok és megoldások.....	491
A. feladatsor (megoldások).....	491
B. feladatsor (megoldások) .....	507
C. feladatsor (megoldások) .....	515
D. feladatsor (megoldások) .....	521
H. feladatsor (megoldások) .....	531
I. feladatsor (megoldások) .....	540
J. feladatsor (megoldások) .....	544
<b>8. fejezet. Változóhasználat, vezérlési szerkezetek a PL/SQL-ben.....</b>	547
Feladatok és megoldások.....	547
<b>9. fejezet. Hivatkozási és összetett adattípusok, kurzor, ROWID.....</b>	556
Feladatok és megoldások.....	556
Összetett (párosítási) feladatok és megoldások .....	591

<b>10. fejezet. Kivételkezelés, alprogramok, triggerek.....</b>	616
Feladatok és megoldások.....	616
<b>11. fejezet. Adatbázis-adminisztrátori ismeretek.....</b>	644
Feladatok és megoldások.....	644
<b>12. fejezet. Kapcsolat a külvilággal .....</b>	654
<b>13. fejezet. Részletező csoportosítások és analitikus függvények az SQL-ben.....</b>	655
<b>14. fejezet. A DEMO vállalat (üzleti alkalmazások).....</b>	656
Feladatok és megoldások.....	656
<b>MELLÉKLETEK .....</b>	691
<b>1. melléklet. A Personal Oracle telepítése .....</b>	693
<b>2. melléklet. Az Oracle alaptáblái .....</b>	694
Az alaptáblák és értelmezésük .....	694
Az emp tábla tartalma.....	694
Az emp tábla értelmezése.....	695
Az emp tábla szerkezete .....	695
A dept tábla tartalma .....	696
A dept tábla értelmezése .....	696
A dept tábla szerkezete .....	696
A salgrade tábla tartalma.....	696
A salgrade tábla értelmezése.....	697
A salgrade tábla szerkezete.....	697
<b>3. melléklet. Bevezetés az SQL*Plus-környezet és az SQL nyelv használatába .....</b>	698
Belépés az SQL*Plus-környezetbe .....	698
Felhasználói táblák lekérdezése .....	700
Felhasználói katalógus és adatszótár.....	700
Felhasználói adattábla lekérdezése.....	701
Felhasználóváltás .....	701
Szkript programok írása, futtatása .....	702
Szövegszerkesztés az SQL*Plus-környezetben.....	702
A login szkript létrehozása.....	702
A login szkript futtatása .....	703
Általános szkript program létrehozása és futtatása .....	703
Kiíratás szkript programból .....	703
Megjegyzések használata.....	703
A sorfolytató jel használata .....	704

Változók használata.....	705
Az SQL*Plus editorának lecserélése .....	705
Tábla létrehozása, feltöltése, törlése, dátumformák .....	706
<b>4. melléklet. Elvezett függőségek megvalósítása triggerrel .....</b>	<b>709</b>
I. A probléma felvetése (elméleti háttér) .....	709
II. A probléma megoldása triggerrel .....	712
<b>5. melléklet. Gyakran használatos adatszótárnézetek .....</b>	<b>720</b>
Táblaszerkezet lekérdezése .....	720
Csatlakozás az adatbázishoz különböző felhasználóként .....	721
Objektumok tulajdonságainak lekérdezése .....	721
Az adatszótár és általános nézeteinek lekérdezése .....	721
A felhasználói objektumok hasznos (szűkített) lekérdezései .....	722
Megszorítások lekérdezése .....	722
PL/SQL-objektumok lekérdezése .....	723
A felhasználók és környezetük lekérdezése .....	723
Az összes felhasználó lekérdezése .....	723
Felhasználók rendszeradatainak lekérdezése .....	724
Felhasználó saját rendszeradatainak lekérdezése .....	724
Felhasználók és táblaterületeik lekérdezése .....	724
Felhasználó saját tábláinak és táblaterületeinek lekérdezése .....	725
A táblaterületek és azok állapotának lekérdezése .....	725
A táblaterületek tárolási helyének lekérdezése .....	725
A felhasználó számára elérhető táblaterületek listája .....	725
Felhasználói táblák és azok táblaterületeinek lekérdezése .....	726
A felhasználók és jogosultságaik lekérdezése .....	726
A Sys rendszeradminisztrátor jogosultságainak lekérdezése .....	726
A System felhasználó jogosultságainak lekérdezése .....	726
Felhasználó rendszerjogosultságainak lekérdezése .....	726
Felhasználó objektumkezelési jogosultságainak lekérdezése .....	726
<b>A CD-mellékletről.....</b>	<b>728</b>
<b>Irodalomjegyzék.....</b>	<b>730</b>
<b>Tárgymutató .....</b>	<b>733</b>

# Előszó

Ez a példatár segítséget kíván nyújtani az egyetemek és főiskolák informatikus hallgatói számára az Adatbázis-kezelés című tárgy gyakorlati anyagának önálló elsajátításához, ám az alapokon túlnyúló professzionális eszközök bemutatása révén a szakirányú képzések és a speciális Oracle- (SQL, PL/SQL) tanfolyamok résztvevői is haszonnal forgathatják.

A példatár első hét fejezete az Oracle SQL\*Plus-környezetében bemutatja az SQL nyelv (DQL-, DML- és DDL-eszközeinek) használatát az alapvető ismeretektől és módszerektől, a többtáblás és csoportosító lekérdezéseken, a táblák létrehozásán és módosításán keresztül, a nézetek segítségével történő összetettebb feladatok megoldásáig.

A következő három fejezet az Oracle magasszintű, professzionális alkalmazások létrehozására szolgáló programozási nyelvét, a PL/SQL-t mutatja be a nyelvi alapoktól az összetett és hivatkozási típusok használatán keresztül a kurzorok, tárolt eljárások és triggerek témaöréig. Jelentőségét az adja, hogy a hagyományosan deklaratív (és emiatt rendkívül tömör) SQL nyelvet a legújabb szabványbővítések éppen a PL/SQL-eszközök beépítésével viszik procedurális irányba, vagyis úgy tűnik, hogy a nem is távoli jövő a teljes alkalmazásfejlesztésre képes SQL nyelvé.

A 11. fejezet az SQL jogosultságkezeléssel kapcsolatos (DCL-eszközeinek) használatát mutatja be, ezzel az adatbázis-adminisztrátori ismeretek területére kalauzolva az olvasót. A 12. fejezet foglalkozik a gyakorlati szakemberek számára igen fontos adatmentés, adatvisszatöltés, archiválás, valamint saját alkalmazások Oracle-rendszerben való futtatásának témaörével.

Az utolsó két fejezet az Oracle legújabb, a hierarchikus struktúrákkal kapcsolatos elemzések (például vállalatok pénzügyi, gazdasági számításainak) elvégzésére szolgáló eszközeiből (részletező csoportosítások, analitikus függvények), és ezeknek egy fiktív vállalatra vonatkozó alkalmazásából nyújt ízelítőt.

A példatár első tizenegy fejezete egy (akár felsőoktatásbeli, akár tanfolyami) képzés tematikus menetrendjénél, kiegészítő tankönyvének is tekinthető. (Heti egy gyakorlati foglalkozással, és egy vagy két felnémű zárt helyi dolgozattal számolva ez éppen egy szemesztert jelent.)

A gyakorlati képzés az Oracle adatbázis-kezelő rendszer SQL\*Plus-környezetének használatán alapszik, ezért feltételezzük, hogy az olvasó számára elérhető számítógépen az már telepítve van. (A felsőoktatás oktatóinak és hallgatóinak az otthoni gépre való telepítést az Oracle Magyarország az Academic Licence által biztosítja.)

A példatár a Feladatgyűjteményből, a Megoldásgyűjteményből, számos mellékletből, irodalomjegyzékből, az egyes példákra mutató részletes címszójegyzékből (tárgymutatóból) áll. A könyvhöz tartozó CD-melléklet a példák megoldásait futtatható formában tartalmazza.

A Feladatgyűjtemény minden fejezete az adott tématerület rövid elméleti összefoglalójával kezdődik. A Megoldásgyűjteményben az egyes feladatokhoz gyakran több megoldást is adunk, elemezve az egyes változatok előnyeit, hátrányait, sőt az oktatási tapasztalatok alapján esetenként egy-egy tipikus hibás megoldást is bemutatunk, megadva a hiba helyét és okát.

A közölt példák és feladatok mindegyike az Oracle-rendszer hagyományos alaptábláit használja, melyek tartalmát és felépítését a mellékletben adjuk meg. Használatuk előnye, hogy az Oracle-rendszer telepítésével minden olvasó egységes feladatkörnyezetbe kerül, és mindenki is láthat az egyszerűbb lekérdezéseknek, nem szükséges a fogalmilag bonyolultabb adattábla létrehozásával kezdenie. (E táblákat generáló szkript programokat feltettük a CD-mellékletre, az esetlegesen „sérült” táblák visszaállításához.)

Bár a példatár az elméleti összefoglalók révén önmagában is használható, ajánlott tankönyvként rendszeresen hivatkozik Kende Mária, Kotsis Domokos és Nagy István *Adatbázis-kezelés az Oracle-rendszerben* című könyvére (Panem Könyvkiadó, 2002).

A példaanyag összeválogatásánál gondoltunk az Oracle használatában már jártas olvasókra is. Mind az SQL-lel, mind a PL/SQL-lel foglalkozó fejezetekben helyeztünk el olyan feladatokat, melyeknek megoldása még a gyakorlott szakembereket is próbára teheti.

Azok számára pedig, akik e példatáron keresztül szeretnének megismerni az Oracle-rendszerrel, a mellékletben megadjuk, hogy

- miként célszerű telepíteni az Oracle-rendszer fejlesztői (egygépes telepítésű) Personal Oracle-változatát,
- mi az Oracle-alaptáblák tartalma és az értelmezése, továbbá
- miként kell a parancsnyelvű SQL\*Plus-környezetbe belépni, és azt, valamint az alapvető SQL-utasításokat használni.

Végül köszönetet szeretnénk mondani mindenkit, akik nélkül ez a könyv sohase készült volna el.

Hálás köszönetünk Kotsis Domokosnak, akihez mindig fordulhattunk baráti vagy szakmai tanácsért, és aki a nélkülözhetetlen szakmai környezetet biztosította. Köszönet illeti Czinkóczki Lászlót, ainek az inspiráló szakmai-baráti beszélgetéseken túl közvetlenül köszönhetjük a 13. fejezetet, mivel ennek anyagát tőle tanultuk. Köszönetünk Juhász Istvánnak, immár második alkalommal lektorunknak, aki nemcsak alapos vizsgálója volt vaskos munkánknak, hanem ötleteivel, javaslataival szemléletet is adott. Nem szeretnénk megfeledkezni azokról a kollégáinkról, barátainkról: Belle Zsuzsannáról, Csink Lászlóról, Kiss Mariannról, Légrádi Gáborról, Losonczi Ilonáról, Radnai Tamásról, Sántáné Tóth Editről és mindenkitől, akiknek segítsége, biztatása és javaslatai „beépültek” a könyvbe.

Bár a felsorolás végére maradtak, tán előre kellett volna venni – hallgatóinkat, a Budapesti Műszaki Főiskola Műszaki Informatikus és Gazdasági Informatikus hallgatóit, akik nélkül e példatár egész biztosan nem készült volna el. Bár a feladatok létrejöttének körülményei (zárthelyi és vizsgadolgozatok) talán nem a legkellemesebb emlékeket idézik bennük, ám az elmúlt 10-12 szemeszter során általuk kitalált szellemes megoldások csakúgy, mint az elkövetett hibák nagyon sok ötletet adtak, és a könyv szempontjából talán nem mellékes bizonysságot, nevezetesen, hogy e példatár tananyaga (az első tíz fejezet) egy szemeszter alatt megtanulható...

# I. RÉSZ

## FELADATGYŰJTEMÉNY

### Bevezetés

A feladatok az Oracle-rendszer alaptábláira (emp, dept, és salgrade) vonatkoznak, melyeknek tartalmát, valamint az angol oszlopnevek általunk hivatkozott magyar megnevezéseit a 2. melléklet, az ezeket generáló szkript programokat pedig a CD-melléklet tartalmazza. A kezdő olvasók számára javasoljuk a feladatok megoldása előtt a 3. melléklet alapos átanulmányozását.

A feladatokban általában nem adjuk meg, hogy melyik adattáblát kell használni, a megfelelő adattábla kijelölése is része a megoldásnak. A feladatok szövegében sokszor az sincs pontosan megadva, hogy mit és hogyan kell például listázni (esetleg csak annyi szerepel, hogy „listázza azon dolgozókat...”, vagy „listázza azon dolgozók adatait...”). Ezekben az esetekben a feladat része annak megállapítása, hogy mit célszerű, illetve, hogy mit lehet listázni. Ilyenkor gondolunk arra, hogy egy listának a feltételek ellenőrzése érdekében azokat az adatokat is érdemes tartalmaznia, amelyek a feladatban voltak megfogalmazva.

Az elméleti összefoglalókban bemutatott mintapéldák SQL\*Plus szkript programjai és a feladatmegoldások szkript programjai fejezetek szerint csoportosítva megtalálhatóak a CD-mellékleten. Lásd a CD-mellékletről szóló részt a könyv végén.

Végül megjegyezzük, hogy a 11. fejezet kivételével a feladatokat legegyszerűbben Scott felhasználóként oldhatjuk meg. Ehhez az SQL\*Plus-környezetbe való belépéskor a scott felhasználónév és a tiger jelszó megadása, vagy bármilyen felhasználóként az SQL\*Plus-környezetbe bejutva, ott a CONNECT scott/tiger SQL\*Plus-utasítás kiadása szükséges.

### Néhány megjegyzés az elméleti összefoglalóhoz

Az adatbázis-kezelés az elmúlt ötven év alatt terebélyes tudománnyá és gyakorlati technológiák gyűjteményévé nőtte ki magát. E példatár célja, hogy korlátozott idő (egy szemeszter – körülbelül három hónap) alatt lehetővé tegye olyan gyakorlati képességek elsajátítását, melyek révén az olvasó közvetlen módon bekapcsolódhat például egy vállalat controlling- vagy adatbázis-adminisztrációs munkájába.

E cél megvalósításához korlátozni kellett a példatárban tárgyalt ismeretek körét. Bár mind az SQL, mind a PL/SQL nyelv utasításai igen bőségesen paraméterezhetők (szépen mutatva azokat a változatos igényeket, melyeknek kielégítésére alkalmasak), ma már a gyakorlott szakemberek sem birtokolják napi készség szintjén az összes paraméterek, opciók használatát.

A fentiekre tekintettel az egyes témaköörök (fejezetek) elején található elméleti összefoglalók célja alapvetően csupán annyi, hogy a bemutatott elméleti ismeretekkel, példákkal közvetlen (és lehetőség szerint teljes) segítséget adjanak a példatárban szereplő feladatok megoldásához. Nem céluk referenciaiként szolgálni, nem kívánják tehát helyettesíteni sem a kézikönyveket, sem az egy-egy tématerületet részletesen feldolgozó szakkönyveket. Ennek megfelelően az egyes utasítások bemutatása, szintaktikai megadása sem teljes. Az érdeklődők számára azonban minden esetben hivatkozni fogunk az adott területhez tartozó részletesebb ismereteket bemutató (alapvetően magyar nyelvű) szakirodalomra, utalva a példatár irodalomjegyzékére.

A szakirodalmi hivatkozások során amikor azt mondjuk, hogy „a témakörről lásd [...], [...] és [...]”, akkor – ha nem is adjuk meg –, értelemszerűen hozzájondoljuk az Oracle rendkívül részletes dokumentációját (amely a telepítőcsomag részeként egy önálló CD, irodalomjegyzékünkben – az ez idő szérint legutolsó – [26]-ként szerepel), valamint az internet korlátlan adattárát (melyből néhány hasznosnak tűnőt azért felsoroltunk: [27]–[60]).

## Az utasítások szintaktikus jelölése

- Az SQL-, SQL\*Plus- és PL/SQL-utasítások alapszavait (terminális szavait) Courier típusú nagybetűk jelzik (pl. SELECT).
- Az utasítások magyarázó szövegeit (nemterminálisait) dőlt betűk jelzik (pl. OszlopNév).
- Az egyéb, utasításbeli (terminális) szimbólumok szintén Courier típusú szimbólumok (pl. PROCEDURE(változónév IN típus) alakban a zárójelek).
- Az alternatívákat egy függőleges vonal ( | ) választja el (pl. ASC | DESC ).
- Az opcionális elemek szögletes zárójelben szerepelnek (pl. [DISTINCT]).
- A kötelezően megadandó alternatívák listáját kapcsos zárójelek határolják (pl. {érték} | \* ), ezek közül tehát az egyiknek szerepelnie kell az utasításban).
- Az iteráció jelölésére a három pont (...) szolgál (pl. oszlop [, oszlop...]).

## 1. FEJEZET

# Egyeszerű 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 – adatvézérlő 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 "MásodlagosOszlophév"**.

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 allekérdezésekkel álló logikai kifejezés, mely tartalmazhat:

- logikai műveletjeleket  
(AND, OR, NOT),
- hasonlító műveletjeleket  
(>, >=, <, <=, =, <>, !=),
- intervallumvizsgálatot  
(oszlopkifejezés BETWEEN AlsóHatár AND FelsőHatár),
- alsztringvizsgálatot  
(oszlopkifejezés LIKE %alsztring%),
- allekérdezésre vonatkozó halmafvizsgálatot  
(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 jelöli 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 ki-számított (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 minden 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 utasításrész, 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 utasításrészben 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 mezőszélességeben 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 mindenkorral ügyelni kell, hogy az összehasonlított, és a visszaadott értékek azonos típusúak legyenek, illetve ez utóbbit lehet NULL érték is. Az összehasonlításnál ügyeljünk arra, hogy ha a vizsgált oszlopkifejezés esetleges NULL értékére szeretnénk feltételezni, 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-függvényvel 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 kvázigráfikus megjelenítésére (lásd 1.6. és 1.7. példa: „Grafikus” megjelenítés).

A karakterkezelő függvények bemenő és kimenő paramétere karakter, karaktersorozat (kivételek 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 <i>m</i> -edik pozíciójától kezdődően <i>n</i> karaktert ad vissza ( <i>n</i> hiánya esetén az összeset).
LENGTH({oszlop   kifejezés})	A karakterlánc hosszát adja vissza.
INSTR({oszlop   kifejezés}, minta [, m [, n]])	A <i>mintá</i> karaktersorozatnak az első paraméterként megadott karaktersorozatbeli pozícióját adja vissza. Ha a visszatérő érték nulla, akkor nem talált. Az opcionális paraméterek jelentése: <i>m</i> : kezdőpozíció, <i>n</i> : <i>n</i> -edik előfordulás:
LPAD({oszlop   kifejezés}, <i>n</i> [, 'kitöltő'])	Jobbra igazítja a karakterláncot, és balról kiegészíti a <i>kitöltő</i> karakterrel oly módon, hogy a hossza éppen <i>n</i> legyen. A <i>kitöltő</i> karakter hiánya esetén szóköz karaktert használ.
RPAD({oszlop   kifejezés}, <i>n</i> [, 'kitöltő'])	Balra igazítja a karakterláncot, és jobbról kiegészíti a <i>kitöltő</i> karakterrel oly módon, hogy a hossza éppen <i>n</i> legyen. A <i>kitöltő</i> 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
-------

**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 oszlopból 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 oszlopértéket 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 szóközzel, 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	:##
ALLÉN	:##
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 függvénytel, dátumból vagy számból formázott karakterláncot pedig a TO\_CHAR függvénytel 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 fel-sorolás 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.
SSSS	Az éjfél óta eltelt másodpercek száma.
/,-:._szóköz	A dátumelemek között használható elválasztó karakterek (a <code>TO_CHAR</code> elhelyezi, a <code>TO_DATE</code> 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 helyértéken vezető nullával).
99999	5 értékes számjegy, vezető nullák nem jelennek meg.
09999	5 értékes számjegy, ugyanennyi vezető nullaival együtt.
0999.999	4 egészjegy, vezető nullaival, 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	
<hr/>	
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	
<hr/>	
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	
<hr/>	

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 cél-szerű 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 mindenkor 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 értékét törthónapként (azaz pontos érték-ként) 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 ( $24*60*60 = 86400$ ).

## 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

## Számok megjelenítése

Az alábbi példában bemutatjuk a TO\_CHAR konverziós függvény használatát tizedesjegyeket is tartalmazó számok megjelenítésére.

### 1.14. példa

Határozza meg az aranymetszés értékét három tizedesjegyig a Fibonacci-számSOR hetedik és nyolcadik eleméből.

#### A módszer

Az aranymetszés értéke közelítően: Aranymetszés  $\approx F_n / F_{n+1}$ , ahol  $F_n$  az  $n$ -edik Fibonacci-szám. (Ez a képlet annál jobban közelíti az aranymetszés értékét, minél nagyobb az  $n$  értéke.) A Fibonacci-sorozat elemei: 0, 1, 1, 2, 3, 5, 8, 13 stb., és  $F_n = F_{n-1} + F_{n-2}$ , tehát a hetedik és a nyolcadik Fibonacci-szám:  $F_7 = 8$ , és  $F_8 = 13$ .

A számolást a legkényelmesebben az Oracle dual nevű munkatábláján végezhetjük. Ez egyetlen oszlopból és egyetlen sorból áll.

#### 1. megoldás

```
SELECT 8/13    AS "Aranymetszés"
      FROM dual;
```

#### Eredmény

```
Aranymetszés
-----
.615384615
```

#### 2. megoldás (Formázott kiíratással: legyen kiíratva három tizedes, vezető nulla, és a fejléckiíratáshoz elegendő karakterhely)

```
SELECT TO_CHAR(8/13,'9999990.999')   AS "Aranymetszés"
      FROM dual;
```

#### Eredmény

```
Aranymetszés
-----
0.615
```

#### 3. megoldás (Formázott kiíratással: legyen kiíratva három tizedes, az első négy helyiértéken vezető nulla, és a fejléckiíratáshoz elegendő karakterhely)

```
SELECT TO_CHAR(8/13,'9990000.999')   AS "Aranymetszés"
      FROM dual;
```

**Eredmény**

Aranymetszés

0000.615

# Feladatok

**1.1. feladat**

Listázza ki a 20-as részleg dolgozóinak nevét, belépési idejét, foglalkozását a nevek szerint csökkenően rendezve. (F01-01.sql)

**1.2. feladat**

Készítsen két listát, melyek a dolgozók adatait tartalmazzák. Az egyiket a fizetés szerint növekvően, a másikat a fizetés szerint csökkenően rendezze.

**1.3. feladat**

Listázza ki a dolgozók nevét, fizetését, jövedelmét a jövedelmük szerint csökkenően rendezve. (A jövedelem a fizetés és a jutalék összege.)

**1.4. feladat**

Listázza ki a dolgozók nevét, részlegüket, jövedelmüket és az adójukat (a jövedelmük 20%-a), az adójuk szerint csökkenően, a nevük szerint növekvő módon rendezve.

**1.5. feladat**

Írassa ki azon alkalmazottak nevét, munkakörét és fizetését, akiknek fizetése nincs az 1500–2850 USD tartományban. A lista fejléce legyen Név, Munkakör, Fizetés.

**1.6. feladat**

Írassa ki azon dolgozók nevét, munkakörét, fizetését, jutalékát és részlegazonosítóját, akik 1000 USD-nál többet keresnek, és 1981. március 1. és szeptember 30. között léptek be a vállalathoz.

**1.7. feladat**

Írassa ki a jutalékkal rendelkező alkalmazottak nevét, jutalékát, főnökének azonosítóját. Legyen a lista rendezett a főnök azonosítója és az alkalmazottak neve szerint.

**1.8. feladat**

Írassa ki azon alkalmazottak azonosítóját, nevét, foglalkozását, fizetését és jutalékát, akiknek jutaléka meghaladja a fizetésük 50%-át.

**1.9. feladat**

Írja ki azon dolgozók nevét, foglalkozását, fizetését és belépési dátumát, akik 1981-ben léptek be a vállalathoz. A lista a belépési dátum szerint legyen rendezve.

**1.10. feladat**

Listázza ki azon alkalmazottak nevét, foglalkozását és jövedelmét, akiknek a névében két L betű szerepel, továbbá vagy a 30-as részlegben dolgozik, vagy a főnökének azonosítója 7782.

**1.11. feladat**

Listázza ki részlegazonosító szerint rendezve a clerk és a salesman munkakörű dolgozók éves fizetését a részleg szerint rendezve.

**1.12. feladat**

Listázza ki az összes dolgozót oly módon, hogy azoknál, akik nem kapnak jutalékot, az a szöveg jelenjen meg, hogy „Nincs jutalék”. A lista fejléce legyen azonosító, belépési dátum, név, foglalkozás, jutalék.

**1.13. feladat**

Listázza ki a man karaktersorozatot tartalmazó munkakörben dolgozók nevét és munkakörét, a munkakör és a név szerint rendezve.

**1.14. feladat**

Listázza ki foglalkozás szerint csoportosítva azon dolgozók nevét, foglalkozását, jövedelmét és részlegét, akiknek jövedelme kisebb 2500 USD-nál, valamint 1981 és 1982 között léptek be. A keletkezett lista elsődlegesen a foglalkozás, másodlagosan a dolgozó neve szerint legyen rendezve.

**1.15. feladat**

Listázza ki azoknak az alkalmazottaknak a nevét, éves fizetését és a munkában eltöltött hónap-jainak számát, akik 1981.07.01. előtt léptek be a vállalathoz. A lista a hónapok száma szerint csökkenően legyen rendezve.

**1.16. feladat**

Listázza ki a C és az M betűvel kezdődő foglalkozású alkalmazottak nevét (nevüket nagybetűvel kezdve és kisbetűvel folytatva), valamint nevük hosszát. Rendezze a listát a foglalkozás szerint.

**1.17. feladat**

A belépési dátum napjai szerint csoportosítva listázza ki azon dolgozók azonosítóját, nevét, jövedelmét, munkába állásuk napját, részlegét, akiknek jövedelme 1300 és 5500 USD közötti érték. A keletkezett lista elsődlegesen a napok sorszáma szerint, másodlagosan a dolgozó neve szerint legyen rendezve. A hétfő napja legyen a vasárnap.

**1.18. feladat**

A vállaltnál hűségjutalmat adnak, és ehhez szükséges azon dolgozók azonosítója, neve, fizetése és munkában eltöltött éve, akik legalább 15 éve álltak munkába. Rendezze a listát a munkában eltöltött évek szerint csökkenően, valamint az azonosító szerint növekvően.

**1.19. feladat**

Listázza ki a dolgozók nevét, munkakörét, fizetését és a fizetési kategóriáját, mely 1000 USD alatt 1, 2000 USD alatt 2 stb., ez utóbbi szerint csökkenően rendezve. (A fizetéstartomány 1..6000 USD.)

**1.20. feladat**

Listázza ki a dolgozók nevét, azonosítóját és beosztását a név szerint rendezve. Egy dolgozó beosztása Főnök, ha van beosztottja, egyébként NULL érték.

## 2. FEJEZET

# Egytáblás csoportosító lekérdezések

## Elméleti összefoglaló

### Csoportképzés

Adattáblák feldolgozásánál gyakran van szükség arra, hogy az attribútumok (oszlopok) értékei (a mezőértékek) szerint a tábla sorait csoportosítsuk. Gondoljunk egy főiskola diákjainak az évfolyamuk, valamint a tankörszámuk alapján történő csoportosítására, amely például a tankörök tanulmányi átlagának meghatározásához nélkülözhetetlen.

A csoportosítást is tartalmazó SELECT utasítás általános felépítését már bemutattuk az 1. fejezetben. A csoportosítás szempontjából a következőképpen osztályozhatjuk egy csoportosításra kijelölt tábla oszlopait. Csoportképzőnek nevezzük azokat az oszlopfejezéseket (illetve oszlopokat), amelyek szerint elvégezzük a csoportosítást. (Itt az oszlopfejezések oszlopokra vonatkozó egysoros függvényekből, illetve műveletekből álló kifejezésekkel jelölnek.) Ezen oszlopfejezések vésszövel elválasztott listája áll a GROUP BY utasításrészben, és ez a lista vagy ennek egy része szerepelhet a szelekciós listában. Az e kifejezésekben nem szereplő (téhát nem-csoportképző) oszlopok viszont nem állhatnak, vagy csak csoportfüggvények paramétereiként állhatnak a szelekciós listában. Az alábbi utasításvázlat az elmondottakat szemlélteti:

```
SELECT ... , X+Y, ... , f(U), g(V)
  FROM ...
 WHERE ...
 GROUP BY ... , X+Y
 HAVING X+Y < 53;
```

ahol X és Y csoportképző attribútumok, U és V pedig nem-csoportképző attribútumok.

Mint a fenti példából is látható, csoportosítás esetén lehetőség van akár a csoportba kerülő sorok szűrésére (a WHERE utasításrésszel), akár a listába kerülő csoportok szűrésére (a HAVING utasításrésszel.)

## Csoportfüggvények

Csoportképzés esetén alkalmazhatjuk az úgynevezett többsoros, azaz csoportfüggvényeket, melyek a sorok GROUP BY utasításrésszel kijelölt csoportjára, illetve ennek hiányában a tábla összes sorára vonatkoznak.

- **AVG( [DISTINCT] oszlopkifejezés )**  
az oszlopkifejezés átlagát adja meg a csoportra vonatkozóan,
- **STDDEV( [DISTINCT] oszlopkifejezés )**  
az oszlopkifejezés szórását (az átlagtól való eltérés értékét) adja meg a csoportra vonatkozóan,
- **VARIANCE( [DISTINCT] oszlopkifejezés )**  
az oszlopkifejezés szórásnégyzetét (a varianciát) adja meg a csoportra vonatkozóan,
- **SUM( [DISTINCT] oszlopkifejezés )**  
az oszlopkifejezés összegét adja meg a csoportra vonatkozóan,
- **MIN( [DISTINCT] oszlopkifejezés )**  
megadja az oszlopkifejezés legkisebb értékét a csoportban,
- **MAX( [DISTINCT] oszlopkifejezés )**  
megadja az oszlopkifejezés legnagyobb értékét a csoportban,
- **COUNT( [ \* | DISTINCT oszlopkifejezés | oszlopkifejezés ] )**  
\* használata esetén megadja a csoport elemeinek (sorainak) számát,  
DISTINCT oszlopkifejezés használata esetén megadja a csoport oszlopkifejezésre vonatkozóan különböző, de nem NULL értékű elemeinek (sorainak) számát,  
oszlopkifejezés használata esetén megadja a csoport oszlopkifejezésre vonatkozó nem NULL értékű elemeinek (sorainak) számát.



### Megjegyzés

- A DISTINCT használata esetén a csoportfüggvény értékének meghatározásában csak egyszer veszi figyelembe az azonos sorokat.
- A csoportfüggvények egymásba ágyazhatók.

### 2.1. példa

Listázza ki a dolgozók részlegenkénti átlagfizetését és ennek szórását egész értékre kerekítve.

### Megoldás

```
SELECT deptno          AS részleg,
       ROUND(AVG(sal),0)    AS átlag,
       ROUND(STDDEV(Sal),0) AS szórás
  FROM emp
 GROUP BY deptno;
```

**Eredmény**

RÉSZLEG	ÁTLAG	SZÓRÁS
10	2917	1894
20	2175	1123
30	1567	668

**2.2. példa**

Listázza a 2000 USD-nál nagyobb átlagjövedelmű részlegeket a jövedelem szerint rendezve.

**Megoldás**

```
SELECT deptno AS részlegazonosító,
       ROUND(AVG(sal+NVL(comm,0))) AS átlagjövedelem
    FROM emp
   GROUP BY deptno
  HAVING ROUND(AVG(sal+NVL(comm,0))) > 2000
 ORDER BY átlagjövedelem ASC;
```

**Eredmény****RÉSZLEGASONOSÍTÓ ÁTLAGJÖVEDELEM**

20	2175
10	2917

**Megjegyzés**

A feladatok megfogalmazása a gyakorlatban általában nem pontos, de a „józan ész” alapján azért ki lehet találni, hogy mire is van szüksége a felhasználónak. Tekintsük a fenti példát. Itt a „Listázza a ... részlegeket” kifejezés jelentése: „Listázza a ... részlegeknek a feladat szempontjából lényeges adatait”, vagyis az azonosítóját és az átlagjövedelmét, illetve a „rendezve” kifejezés jelentése egyaránt lehet „növekvően rendezve” vagy „csökkenően rendezve”, ám az első a gyakoribb. A feladatot természetesen már e megfontolások szellemében oldottuk meg.

**2.3. példa (Csoportfüggvények egymásba ágyazása)**

Listázza a részlegenkénti legnagyobb fizetéseket, a legkisebb, a legnagyobb és az átlagos részlegenkénti legnagyobb fizetéseket, a részlegenkénti létszámokat és a legkisebb, a legnagyobb és az átlagos részlegenkénti létszámokat.

**Megoldás**

(Az alábbiakban szereplő PROMPT SQL\*Plus-utasítás segítségével lehet megadott szöveget a képernyőre kiírni. Részletesen lásd 4. fejezet.)

PROMPT Részlegenkénti legnagyobb fizetések:

```
SELECT deptno, MAX(sal)
      FROM emp
     GROUP BY deptno;
```

PROMPT A legkisebb, a legnagyobb és az átlagos

részlegenkénti legnagyobb fizetések:

```
SELECT MIN(MAX(sal)), MAX(MAX(sal)), ROUND(AVG(MAX(sal)))
      FROM emp
     GROUP BY deptno;
```

PROMPT Részlegenkénti létszámok:

```
SELECT deptno, COUNT(*)
      FROM emp
     GROUP BY deptno;
```

PROMPT A legkisebb, a legnagyobb és az átlagos-  
részlegenkénti létszámok:

```
SELECT MIN(COUNT(*)), MAX(COUNT(*)), ROUND(AVG(COUNT(*)), 2)
      FROM emp
     GROUP BY deptno;
```

### Eredmény

Részlegenkénti legnagyobb fizetések:

DEPTNO	MAX(SAL)
10	5000
20	3000
30	2850

A legkisebb, a legnagyobb és az átlagos  
részlegenkénti legnagyobb fizetések:

```
MIN(MAX(SAL)) MAX(MAX(SAL)) ROUND(AVG(MAX(SAL)))
```

2850	5000	3617
------	------	------

Részlegenkénti létszámok:

DEPTNO	COUNT(*)
10	3
20	5
30	6

A legkisebb, a legnagyobb és az átlagos részlegenkénti létszámok:

MIN(COUNT(*))	MAX(COUNT(*))	ROUND(AVG(COUNT(*)), 2)
3	6	4.67

## Feladatok

### 2.1. feladat

Listázza ki munkakörönként az átlagfizetéseket két tizedesre kerekítve. Rendezze átlagfizetések szerint csökkenően.

### 2.2. feladat

Listázza ki csökkenően rendezve a főnökök átlagfizetését egész értékre kerekítve. (Főnök az a dolgozó, akinek azonosítója szerepel az mgr oszlopban.)

### 2.3. feladat

Listázza ki részlegenként a legnagyobb és legkisebb havi jövedelmeket.

### 2.4. feladat

Listázza ki a legalább egy dolgozójú részlegeket a dolgozószám szerint csökkenően rendezve.

### 2.5. feladat

Listázza ki a főnökök azonosítóit, valamint azt, hogy hány beosztottjuk van. Rendezze a listát a beosztottak száma szerint csökkenően. Ha valakinek nincs főnöke, ahhoz írjon valamilyen megjegyzést ( tulajdonos vagy elnök stb.).

### 2.6. feladat

Listázza ki az azonosítójuk hárommal való oszthatósága alapján a dolgozók átlagjövedelmét, a dolgozók számát és legkisebb fizetését.

### 2.7. feladat

Listázza ki a 2000 USD-nál nagyobb átlagjövedelmeket egész értékre kerekítve a foglalkozás szerint csoportosítva. A lista a foglalkozás szerint legyen rendezett.

### 2.8. feladat

Listázza ki azokat a részlegeket, ahol a fizetésátlag nagyobb 1500 USD-nál. Rendezze fizetés-átlag szerint csökkenően.

### 2.9. feladat

Listázza ki foglalkozásonként a legnagyobb jövedelmeket, jövedelem szerint rendezve.

**2.10. feladat**

Listázza ki, hogy az egyes foglalkozási csoportokon belül hányan dolgoznak. A lista a létszám szerint legyen rendezett.

**2.11. feladat**

Listázza ki a főnökök azonosítóit és a főnökökhöz tartozó beosztottak számát, ez utóbbi adat szerint rendezze.

**2.12. feladat**

Listázza ki azon foglalkozások átlagjövedelmét, amelyek nevében a man alsztring megtalálható. A listát rendezze az átlagjövedelem szerint csökkenő sorrendben.

**2.13. feladat**

Listázza ki rendezve azon foglalkozási csoportok átlagfizetését, ahol kettő vagy ennél több alkalmazott dolgozik.

**2.14. feladat**

Írjon utasítást azon részlegek azonosítójának, dolgozói számának és azok legnagyobb és legkisebb jövedelmének lekérdezésére, ahol a részlegszám páros. A listát a részlegazonosító szerint rendezze.

**2.15. feladat**

Listázza ki az azonosító paritása szerint csoportosítva a dolgozókat. Hagyja ki azokat a dolgozókat, akik 1981 után léptek be a vállalathoz. A listát elsődlegesen paritás szerint, másodlagosan a dolgozó neve szerint rendezze.

**2.16. feladat**

Számítsa ki az átlagos jutalékot.

**2.17. feladat**

Készítsen listát a páros és a páratlan azonosítójú dolgozók számáról.

**2.18. feladat**

Listázza ki a dolgozók számát fizetési kategóriák szerint. (A fizetési kategóriákat vagy ön definiálja, vagy vegye a salgrade táblából.)

**2.19. feladat**

Listázza ki főnökönként (mgr) a főnökhöz tartozó legkisebb dolgozói fizetéseket. Hagyja ki azon dolgozók fizetését, akiknek nincs főnökük, valamint azokat a csoportokat, ahol a legkisebb fizetés nagyobb 2000 USD-nál. Rendezze a listát a legkisebb fizetések szerint növekvően.

**2.20. feladat**

Listázza ki főnökönként (mgr) a főnökhöz tartozó dolgozói átlagfizetéseket. Hagyja ki azon dolgozók fizetését, akiknek nincs főnökük, valamint azokat a csoportokat, ahol az átlagfizetés nagyobb 3000 USD-nál. Rendezze a listát az átlagfizetések szerint csökkenően.

**2.21. feladat**

Listázza ki főnökönként a főnökhöz tartozó dolgozók jövedelme közül a legnagyobbat. Hagyja ki a listakészítésből azon dolgozókat, akiknek nincs jutalékuk, valamint azokat a (legnagyobb) jövedelemeket, melyek nagyobbak 3500 USD-nál. Rendezze a listát a legnagyobb jövedelem szerint csökkenően.

**2.22. feladat**

Listázza ki részlegenként az egy tizedesre kerekített átlagfizetéseket. Hagyja ki az átlag meghatározásából az 1981. január 1-je előtt belépett dolgozókat, valamint azon részlegek átlagfizetését, melyekben a legkisebb fizetés kisebb 1000 USD-nál. Rendezze a listát az átlagfizetések szerint növekvően.

**2.23. feladat**

Listázza ki munkakörönként a dolgozók számát és az egész értékre kerekített átlagfizetésüket numerikusan és grafikusan is. Ez utóbbit csillag (\*) karakterek sorozataként balra igazítva jeleznítsse meg oly módon, hogy e sorozatban 200 US dolláronként egy csillag karakter álljon. Rendezze a listát az átlagfizetések szerint csökkenően.

**2.24. feladat**

Listázza ki főnökönként a legrégebb óta munkaviszonyban álló dolgozóknak a mai napig munkában töltött éveinek számát numerikusan és grafikusan is. Ez utóbbit kettőskereszt (#) karakterek sorozataként balra igazítva jelenítse meg oly módon, hogy ebben a sorozatban 5 évenként egy kettőskereszt karakter álljon. Rendezze a listát az évek száma szerint növekvően.

### 3. FEJEZET

# Többtáblás lekérdezések, allekérdezések

## Elméleti összefoglaló

### Táblák összekapcsolása

#### Táblák összekapcsolása a FROM és WHERE utasításrészben

A SELECT utasítás egyszerűsített alakja (általános alakját és értelmezését lásd 1. fejezet és [16]):

```
SELECT SzelekciósLista
      FROM tábla1 [MásodlagosTáblanév1],
           tábla2 [MásodlagosTáblanév2]
           [...]
           [WHERE LogikaiFeltétel]
           [GROUP BY ...]
           [HAVING ...]
           [ORDER BY ...];
```

ahol:

- a SzelekciósLista MinősítettOszlopneveket tartalmaz minden olyan oszlop esetén, amely azonos névvel szerepel több összekapcsolandó táblában vagy inline nézet szelekciós listájában (lásd később) szerepel, ahol a MinősítettOszlopnev alakja:
  - [tulajdonos].[táblanév].[oszlopnev], vagy
  - [tulajdonos].[MásodlagosTáblanév].[oszlopnev](a szelekciós lista kezdődhet a DISTINCT kulcsszóval, ha a többszörös sormegjelenítést ki akarjuk szűrni),
- a tábla fizikai, logikai (nézet-) tábla vagy allekérdezés (azaz inline nézet), ahol az inline nézetek esetén a másodlagos táblanév használata célszerű,
- a MásodlagosTáblanév egy összefüggő (szóközt nem tartalmazó) karaktersorozat, amely a tábla tartalmára utal,

- a LogikaiFeltétel egy tábla-összekapcsoló és táblaszűrő feltételeket tartalmazó logikai kifejezés.

### Megjegyzés

Az összekapcsolandó táblák között lehetnek azonosak (azaz egy táblát önmagával is összekapcsolhatunk), ekkor másodlagos táblanevek használata kötelező. Ilyen eset fordul elő például a logikailag hierarchikus adatszerkezetű adattáblák korrelált lekérdezése esetén (lásd az alábbiakban).

## Táblák összekapcsolása a FROM utasításrészben a JOIN segítségével

Az SQL92 szabvány és ennek alapján az Oracle 9i lehetővé teszi az összekapcsolást megvalósító JOIN utasításrész használatát. Ez már megengedi a szokásos (úgynevezett belső) tábla-összekapcsoláson kívül az olyan eredménysort létrehozó összekapcsolást is, ahol az egyik tábla valamely attribútumértékének megfelelő sor nem szerepel a másik táblában (ezt nevezik külső tábla-összekapcsolásnak). Korábban az Oracle a szabványtól eltérő külső összekapcsoló (+) operátor alkalmazta az ilyen esetekre. Más (például SQL Server) adatbázis-kezelő rendszerekben a JOIN kulcsszót szintén az alábbi értelemben használják.

A JOIN összekapcsolást használó SELECT utasítás egyszerűsített szintaktikája általános alakját és értelmezését lásd [20] vagy [22]):

```
SELECT SzelekciósLista
  FROM tábla1
    [ {INNER | LEFT | RIGHT | FULL}
      JOIN tábla2 { ON TáblaösszekötőFeltétel |
                    USING( oszlop1 [, oszlop2, ...] ) } ]
    ...
    [WHERE ...]
    [GROUP BY ...]
    [HAVING ...]
    [ORDER BY ...];
```

ahol a korábban nem ismertetett kulcsszavak jelentése a következő:

- INNER esetén az eredménylistában csak olyan sorok összekapcsolása szerepelhet, amelyben az összekapcsolási feltételnek megfelelő sorok oszlopértéke mindegyik táblában létezik (beli összekapcsolás),
- LEFT esetén e kulcsszótól balra álló tábla (például *tábla1*) sorai akkor is szerepelnek a listában, ha az összekapcsolási feltétel szerinti oszlop a jobboldali táblában (például *tábla2*) nem rendelkezik megfelelő értékkel, ekkor ez utóbbi attribútumainak helyén nem jelenik meg érték (külső összekapcsolás),
- RIGHT az előzőhez hasonló, csak a feltételeknek fordított irányban kell teljesülniük (ez is külső összekapcsolás),
- FULL esetén minden írásban engedélyezett a hiányos listasor (ez is külső összekapcsolás),

- az ON kulcsszó mellett álló Táblaösszekötő Feltétel egy, a jobb és bal oldali táblát összekapcsoló logikai kifejezés,
- ha a jobb és a bal oldali táblát egyenlőséggel összekapcsoló oszlopok nevei páronként megegyeznek, akkor elegendő ezen oszlopok neveit felsorolni a USING kulcsszó után.



### Megjegyzés

- Az összekapsolandó táblák között ekkor is lehetnek azonosak.
- A JOIN használata esetén a WHERE utasításrészben csak táblaszűrő feltételeket célszerű alkalmazni (hiszen a táblaösszekötő feltételek már szerepelnek az ON, illetve a USING mellett). Sőt, ha a hatékony lekérdezés érdekében a táblanevek helyén táblaszűréseket tartalmazó allekérdezéseket vagy ilyeneket tartalmazó logikai (más néven nézet-) táblákat alkalmazunk, akkor a WHERE utasításrész teljesen el is maradhat.

## 3.1. példa

Hozza létre az emp tábla felépítésével azonos emp1 táblát, majd a 30-as deptno attribútumértékeket változtassa 50-re. Ezek után listázza ki az emp1 tábla dolgozóit és azok munkahelyét.

### Megoldás

#### 1. lépés

Először létrehozzuk az emp tábla felépítésével azonos emp1 táblát, majd módosítjuk annak tartalmát.

```
DROP TABLE emp1;

CREATE TABLE emp1
AS SELECT * FROM emp;

SELECT * FROM emp1;
```

#### 2. lépés

Ezután a 30-as deptno attribútumértékeket változtassuk 50-re (részletesen lásd az UPDATE adatmódosító utasítás ismertetésénél). Ezt tulajdonképpen csak azért tesszük, hogy legyen olyan deptno érték, amely a dept táblában nem szerepel.

```
UPDATE emp1
SET deptno = 50
WHERE deptno = 30;

SELECT * FROM emp1;
SELECT * FROM dept;
```

Ezek után az emp1 tábla:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	50
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	50
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	50
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		50
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	50
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		50
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

Idézzük fel a dept táblát:

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

### 3A. lépés (Tábla-összekapcsolás a JOIN segítségével [belso összekapcsolás])

```
SELECT ename, deptno, loc
  FROM emp1
    INNER JOIN dept USING(deptno);
```

ENAME	DEPTNO	LOC
SMITH	20	DALLAS
JONES	20	DALLAS
CLARK	10	NEW YORK
SCOTT	20	DALLAS
KING	10	NEW YORK
ADAMS	20	DALLAS
FORD	20	DALLAS
MILLER	10	NEW YORK

8 sor kijelölve.

Ebben a listában csak azok szerepelnek, akiknek deptno értéke mindenkorban előfordul.

**3B. lépés (Tábla-összekapcsolás a JOIN segítségével [külső összekapcsolás a LEFT kulcs-szóval])**

```
SELECT ename, deptno, loc
  FROM emp1
    LEFT JOIN dept USING(deptno);
```

ENAME	DEPTNO	LOC
MILLER	10	NEW YORK
KING	10	NEW YORK
CLARK	10	NEW YORK
FORD	20	DALLAS
ADAMS	20	DALLAS
SCOTT	20	DALLAS
JONES	20	DALLAS
SMITH	20	DALLAS
JAMES	50	
TURNER	50	
BLAKE	50	
MARTIN	50	
WARD	50	
ALLEN	50	

14 sor kijelölve.

Ebben a listában már minden dolgozó szerepelnek, akik a bal oldali (emp1) táblában szerepelnek, bár a deptno attribútum értékeinek nincs minden esetben megfelelő értéke a jobb oldali (dept) táblában (ez az 50-es deptno).

**3C. lépés (Tábla-összekapcsolás a JOIN segítségével [külső összekapcsolás a RIGHT kulcs-szóval])**

```
SELECT ename, deptno, loc
  FROM emp1
    RIGHT JOIN dept USING (deptno);
```

ENAME	DEPTNO	LOC
SMITH	20	DALLAS
JONES	20	DALLAS
CLARK	10	NEW YORK
SCOTT	20	DALLAS
KING	10	NEW YORK

ADAMS	20 DALLAS
FORD	20 DALLAS
MILLER	10 NEW YORK
	30 CHICAGO
	40 BOSTON

10 sor kijelölve.

Ebben a listában a jobb oldali tábla (dept) minden sora (vagyis minden részleg azonosítója) szerepel akkor is, ha a bal oldali táblában (emp1) ennek nincs megfelelője (azaz akkor is, ha azon a részlegben nincs is dolgozó). Ebben az esetben nincs olyan dolgozó az emp1 táblában, aki a 30-as vagy a 40-es részlegben dolgozik, és a bal oldali (emp1) tábla minden sora nem is jelent meg.

### 3D. lépés (Tábla-összekapcsolás a JOIN segítségével [külső összekapcsolás a FULL kulcsszóval])

```
SELECT ename, deptno, loc
  FROM emp1
    FULL JOIN dept USING(deptno);
```

ENAME	DEPTNO	LOC
MILLER	10	NEW YORK
KING	10	NEW YORK
CLARK	10	NEW YORK
FORD	20	DALLAS
ADAMS	20	DALLAS
SCOTT	20	DALLAS
JONES	20	DALLAS
SMITH	20	DALLAS
JAMES	50	
TURNER	50	
BLAKE	50	
MARTIN	50	
WARD	50	
ALLEN	50	
	30	CHICAGO
	40	BOSTON

16 sor kijelölve.

A keletkezett listában minden tábla összes sora szerepel.

## Hierarchikus adatszerkezet megjelenítése

Az adattáblák logikai szerkezetében gyakran fordul elő hierarchikus (faszerkezetű) kapcsolat az adatok között (például főnök-beosztott, családfa stb.). Ilyen esetekben az egyik tipikus lekérdezési feladat e hierarchikus kapcsolat megjelenítése. Az alábbiakban két módszert is bemutatunk.

Az egyik (a korrelált lekérdezés) során az egyetlen adattáblára két különböző néven hivatkozunk, és így a hierarchikus kapcsolatot e „két” tábla között hagyományos többtáblás lekérdezésként térképezzük fel.

A másik módszer az SQL nyelv kifejezetten a hierarchikus adatszerkezetek szemléletes megjelenítésére szolgáló nyelvi eszközének, a SELECT utasításba épített CONNECT BY utasításrésznek a használatán alapul.

### KORRELÁLT LEKÉRDEZÉS

#### 3.2. példa

Listázzuk ki az emp tábla alapján a főnökkhöz tartozó beosztottak nevét és részlegük azonosítóját célszerű módon.

##### 1. megoldás

A feladat megoldható az emp tábla önmagára való hivatkozásával, vagyis az alábbi, logikailag többtáblás (korrelált) lekérdezéssel:

```
SELECT főnök.ename AS "Főnök neve",
       dolgozó.ename AS "Dolgozó neve",
       dolgozó.deptno AS "Részlegazonosító"
  FROM emp dolgozó,
       emp főnök
 WHERE főnök.empno = dolgozó.mgr
 ORDER BY főnök.ename,
          dolgozó.ename;
```

Főnök neve Dolgozó ne Részlegazonosító

BLAKE	ALLEN	30
BLAKE	JAMES	30
BLAKE	MARTIN	30
BLAKE	TURNER	30
BLAKE	WARD	30
CLARK	MILLER	10
FORD	SMITH	20
JONES	FORD	20
JONES	SCOTT	20
KING	BLAKE	30
KING	CLARK	10

KING	JONES	20
SCOTT	ADAMS	20

13 sor kijelölve.

A fenti listában csak 13 sor szerepel, mivel King-nek nincs főnöke.

### **HIERARCHIALEKÉRDEZÉS (CONNECT BY)**

Az előző pontban elkezdett feladat megoldása előtt bemutatjuk a CONNECT BY utasításrész használatát ( pontos ismertetést lásd [12]):

```
SELECT SzelekciósLista
  FROM tábla
  [WHERE LogikaiFeltétel]
  CONNECT BY { PRIOR kifejezés = kifejezés |
                kifejezés = PRIOR kifejezés }
  [START WITH kifejezés = kifejezés]
  [ORDER BY ...];
```

ahol a korábban nem ismertetett kulcsszavak jelentése:

- a SzelekciósLista tartalmazhatja a LEVEL pszeudooszlopot (belől változót), amely megadja, hogy az aktuálisan listázott elem a hierarchia mely szintjén található,
- a START WITH utasításrészben adjuk meg, hogy a hierarchikus szerkezet feldolgozása hol kezdődjön,
- a WHERE utasításrészben megadott logikai feltétel csak konkrét sorok kiszűrésére alkalmas, leszármazottaik és őseik (szülőcsúcstól függően) kiszűrésére nem,
- a CONNECT BY utasításrészben megadott feltételt használjuk a sorok és leszármazottainak kiszűrésére,
- a PRIOR meghatározza a bejárás szerinti szülőcsúcsot (függetlenül attól, hogy az egyenlőségjel jobb vagy bal oldalán áll).

### **A 3.2. példa megoldásának folytatása**

#### **2. megoldás**

Tekintsük ezután az előző feladat megoldását a CONNECT BY utasításrész használatával:

```
SELECT LEVEL AS szint,
       ename AS dolgozó,
       deptno AS részleg
  FROM emp
 CONNECT BY mgr = PRIOR empno
 START WITH UPPER(ename) = 'KING'
 ORDER BY deptno;
```

SZINT DOLGOZÓ	RÉSZLEG
1 KING	10
2 CLARK	10
3 MILLER	10
2 JONES	20
3 SCOTT	20
4 ADAMS	20
3 FORD	20
4 SMITH	20
2 BLAKE	30
3 ALLEN	30
3 WARD	30
3 MARTIN	30
3 TURNER	30
3 JAMES	30

14 sor kijelölve.

A fenti listában most már 14 sor szerepel, mivel a King is a hierarchikus szervezet tagja.

### 3. megoldás

Ha az előző példabeli listához hasonló eredményt szeretnénk látni, akkor a

```
SELECT LEVEL AS szint,
       ename AS dolgozó,
       deptno AS részleg
  FROM emp
 WHERE mgr IS NOT NULL
 CONNECT BY mgr = PRIOR empno
 START WITH UPPER(ename) = 'KING'
 ORDER BY deptno;
```

utasítást használva kapjuk:

SZINT DOLGOZÓ	RÉSZLEG
2 CLARK	10
3 MILLER	10
2 JONES	20
3 SCOTT	20
4 ADAMS	20
3 FORD	20
4 SMITH	20

2 BLAKE	30
3 ALLEN	30
3 WARD	30
3 MARTIN	30
3 TURNER	30
3 JAMES	30

13 sor kijelölve.

#### 4. megoldás

Ha a hierarchikus szerkezet egy ágát az ellenkező irányból kívánjuk bejárni, például Smith-től kezdődően, akkor a

```
SELECT LEVEL AS szint,
       ename AS dolgozó,
       deptno AS részleg
  FROM emp
CONNECT BY PRIOR mgr = empno
START WITH UPPER(ename) = 'SMITH'
ORDER BY deptno;
```

vagy

```
SELECT LEVEL AS szint,
       ename AS dolgozó,
       deptno AS részleg
  FROM emp
CONNECT BY empno = PRIOR mgr
START WITH UPPER(ename) = 'SMITH'
ORDER BY deptno;
```

utasítással kapott lista:

SZINT	DOLGOZÓ	RÉSZLEG
4	KING	10
1	SMITH	20
2	FORD	20
3	JONES	20

## Allekérdezések

Az Oracle-ban lehetőség van arra, hogy bizonyos SQL-utasításokon belül is használjunk SELECT utasításokat. Ezeket allekérdezéseknek nevezzük (általános ismertetésüket lásd [16]). A belső SELECT, vagyis az allekérdezés eredménye nem jelenik meg a listában, de eredménysorokat vagy oszlopkifejezéseket ad át a külső SQL-utasítás (például egy külső SELECT) számára. Az allekérdezések több szinten egymásba ágyazhatók, és a kiértékelés legbelülről halad kifelé. Ez azt jelenti, hogy először a legbelső SELECT utasítást értékeli ki az adatbázis motorja, és onnan halad kifelé.

Allekérdezések előfordulhatnak a FROM és a WHERE utasításrészeken, valamint a szelekciós listákban.

### Allekérdezések a FROM utasításrészben

A FROM utasításrészben szereplő allekérdezést (melyet inline nézetnek is nevezünk) minden másodlagos táblánévvel kell ellátni, ekkor az allekérdezés szelekciós listájának elemeire a külső SELECT szelekciós listájában minősített névvel hivatkozhatunk. Hasonló módon hivatkozhatunk rá ugyanezen szint WHERE utasításrészének logikai kifejezéseiben is.

### Allekérdezések a WHERE utasításrészben

A WHERE utasításrész logikai kifejezésében a szokásos műveleteken és relációkon kívül szerepelhetnek az IN, ANY, ALL, EXISTS logikai operátorok is:

- **oszlopkifejezés [NOT] IN halmaz**  
logikai kifejezés értéke TRUE, ha az oszlopkifejezés értéke szerepel (NOT esetén nem szerepel) a halmaz elemei között,
- **oszlopkifejezés relációjel ANY halmaz**  
ahol a relációjel az =, <, >, <=, >=, <>, != valamelyike lehet (az utolsó kettő jelentése ugyanaz; nem egyenlő), továbbá e logikai kifejezés értéke TRUE, ha a halmaz elemei között van legalább egy, amely a kijelölt relációnak megfelel,
- **oszlopkifejezés relációjel ALL halmaz**  
ahol a relációjel az =, <, >, <=, >=, <>, != valamelyike lehet, továbbá e logikai kifejezés értéke TRUE, ha a halmaz elemeinek mindegyike megfelel a kijelölt relációnak,
- **[NOT] EXISTS halmaz**  
logikai kifejezés értéke TRUE, ha a halmaz legalább egy elemet tartalmaz (NOT esetén egyetlen elemet sem tartalmaz, azaz üres a halmaz),

ahol:

*halmaz*: { (érték1 [, érték2]...) | (allekérdezés) | halmaz halmazművelet halmaz}

melyben a halmazművelet az UNION (halmazegyesítés), az INTERSECT (halmazmetszet) és a MINUS (halmazkülönbség) valamelyike lehet, továbbá az allekérdezés az IN, ANY, ALL esetén csak egyetlen oszlopot tartalmazhat (ekkor tehát az allekérdezés által visszaadott sorok egyetlen oszlop értékei).

A WHERE utasításrészben szereplő összehasonlító relációk (=, <, >, <=, >=, <>, !=) esetén a fentiekhez hasonló módon a relációjelek bal oldalán minden oszlop kifejezés állhat, míg a jobb oldalon az oszlop kifejezésen kívül állhat allekérdezés is [a „(” és „)” zárójelek között], feltéve hogy ennek értékkészlete összhangban van az adott relációval. Ez egyszerűen azt jelenti, hogy az allekérdezés csupán egyetlen értéket (egyetlen sort és egyetlen oszlopot) adhat vissza (az ilyen lekérdezéseket nevezzük *egyértékű allekérdezéseknek*), másrészt azt jelenti, hogy ennek az egyetlen értéknek kompatibilis típusúnak kell lennie a relációjel bal oldalán álló oszlop kifejezéssel.

Egyébként általános szabály, hogy ahol érték állhat (például a LIKE vagy a BETWEEN esetén), ott egyértékű allekérdezés is állhat (természetesen zárójelben).

A WHERE utasításrészben szereplő allekérdezések jellegzetessége, hogy azokat nem lehet ellátni másodlagos táblánál, ezért nem is hivatkozhatunk rá sem további feltételekben, sem például egy külső SELECT szelekciós listájában.

## Allekérdezések a szelekciós listában

A szelekciós listában is állhat allekérdezés (ez természetesen csak egyértékű lehet), melyre azonban még utasításon belül sem lehet hivatkozni.

# Feladatok

### 3.1. feladat

Listázza ki azon dolgozók nevét és részlegük nevét, akiknek nevében az A betű szerepel.

### 3.2. feladat

Listázza ki a Dallas-i telephely minden dolgozójának nevét, munkakörét, fizetését és részlegének azonosítóját.

### 3.3. feladat

Listázza ki a clerk munkakörű dolgozókat foglalkoztató részlegek azonosítóját, nevét és telephelyét. A lista a részlegnév szerint rendezve.

### 3.4. feladat

Listázza ki a Dallas-ban és Chicago-ban dolgozók nevét, munkakörét és telephelyét. A lista telephely szerint legyen rendezett.

### 3.5. feladat

Listázza ki az egyes részlegek nevét, telephelyük címét, dolgozóik átlagfizetését a részlegnevek szerint rendezve.

**3.6. feladat**

Listázza ki a 20-as és a 30-as részleg legnagyobb fizetésű dolgozóinak azonosítóját, nevét, foglalkozását, jutalékát és belépési dátumát.

**3.7. feladat**

Listázza ki minden részleg legkisebb jövedelmű dolgozójának azonosítóját, nevét, foglalkozását, jutalékát és belépési dátumát.

**3.8. feladat**

Listázza ki azon részlegek nevét és telephelyét, ahol a dolgozók átlagjövedelme kisebb mint 2200 USD.

**3.9. feladat**

Írjon olyan lekérdezést, ami megadja az összes, jutalékkal rendelkező alkalmazott nevét, részlegének nevét és helyét.

**3.10. feladat**

Listázza ki a dolgozók nevét és azonosítóját a főnökük (mgr) nevével és azonosítójával együtt úgy, hogy akinek nincs főnöke, annak a NULL érték helyére a Legfőbb karaktersorozatot írja.

**3.11. feladat**

Listázza ki a NEW YORK telephely minden dolgozójának nevét, azonosítóját, jövedelmét és főnökének nevét, telephelyét.

**3.12. feladat**

Listázza ki minden alkalmazott nevét, részlegének nevét és fizetését, akiknek fizetése meggyezik valamelyik Dallas-ban dolgozó alkalmazottával. Legyen a lista fejléce név, részleg neve, fizetés, és a lista legyen a fizetés és a részleg neve szerint rendezett.

**3.13. feladat**

Listázza ki azokat a dolgozókat, akiknek neve hasonlít egy munkakör nevéhez.

**3.14. feladat**

Listázza ki azoknak a főnököknek az azonosítóját, akik nem menedzser foglalkozásúak. A lista a főnök azonosítója (mgr) szerint legyen rendezett.

**3.15. feladat**

Hány olyan főnök van, aki nem menedzser foglalkozású?

**3.16. feladat**

Listázza ki a főnökeik szerint csoportosítva a legkisebb jövedelmű dolgozókat. Hagya ki azokat a dolgozókat, akiknek nincs főnökük, valamint azokat a csoportokat, ahol a legkisebb jövedelem nagyobb 3000 USD-nál. Rendezze a listát a legkisebb jövedelmek szerint növekvően.

**3.17. feladat**

Listázza ki minden olyan dolgozó azonosítóját és nevét, akik olyan részlegben dolgoznak, melyen található nevében T betűt tartalmazó dolgozó. Legyen a lista fejléce azonosító, név, részleg helye, és a lista legyen a részleg helye és a név szerint rendezett.

**3.18. feladat**

Listázza ki a főnökeik (mgr) szerint csoportosítva azokat a dolgozókat, akiknek fizetése e csoportosítás szerint a legkisebb, de nagyobb 1000 USD-nál. A lista a fizetés növekvő értéke szerint legyen rendezett. Legyen a lista fejléce: főnök kódja, dolgozónév, fizetés.

**3.19. feladat**

Listázza ki azon főnököknél (mgr) a legkisebb és legnagyobb fizetéseket, melyeknél a legkisebb fizetések 3000 USD-nál alacsonyabbak. A listát a legkisebb fizetés szerint rendezze, a fejléc pedig legyen Főnök kódja, Legkisebb fizetés és Legnagyobb fizetés.

**3.20. feladat**

Listázza ki a fizetés szerint csökkenő sorba rendezve az eladók (salesman) és a hivatalnokok (clerk) főnökeinek nevét és fizetését, a saját nevét, munkakörét, fizetését, valamint a saját fizetés/főnök fizetése arányukat elsődlegesen a főnök neve szerint, másodlagosan a fizetés aránya szerint rendezve.

# Összetett feladatok

**3.21. feladat**

Listázza ki a Chicago-i telephelyű főnök nevét, azonosítóját, munkakörét, fizetését, beosztottainak átlagfizetését és annak szórását és varianciáját.

**3.22. feladat**

Listázza ki a 2000 és 4000 USD közötti fizetésű főnökök nevét, fizetését, telephelyét és beosztottainak átlagfizetését a főnök neve szerint rendezve.

**3.23. feladat**

Listázza ki minden részleg legkisebb jövedelmű dolgozójának azonosítóját, nevét, foglalkozását, részlegének azonosítóját, telephelyét és munkában eltöltött éveinek számát. Legyen a lista a munkában töltött évek szerint rendezve.

**3.24. feladat**

Listázza ki az egyes részlegek telephelyének nevét, a részleg dolgozóinak egész értékre kerekített átlagjövedelmét, valamint az itt dolgozók főnökeinek nevét, fizetését és telephelyét az átlagjövedelem szerint rendezve, és a részlegadatokat ismétlésmentesen megjelenítve.

**3.25. feladat**

Listázza ki minden dolgozó nevét, foglalkozását, telephelyét, valamint jövedelmük és a részlegük átlagjövedelme közti különbséget, akiknél a munkakörük átlagjövedelme kisebb az összes dolgozó átlagjövedelménél. A listát rendezze telephely szerint.

**3.26. feladat**

Listázza ki azoknak a dolgozóknak a nevét, munkakörét, jövedelmét, telephelyét és a munkakörük átlagjövedelmét, akiknek jövedelme a munkakörük átlagjövedelménél kisebb. A lista a dolgozók neve szerint legyen rendezve.

**3.27. feladat**

Listázza ki minden dolgozó nevét, foglalkozását, részlegük nevét, valamint részlegük átlagjövedelme és saját jövedelmük közti különbséget, akiknek a munkaköri átlagjövedelme kisebb az összes dolgozó átlagjövedelménél. A listát a részleg neve szerint rendezze.

**3.28. feladat**

Listázza ki minden dolgozó nevét, munkakörét, telephelyét, fizetését, valamint a saját telephelyén a munkakörének átlagfizetését és telephelyének átlagfizetését. A listát munkakör és telephely szerint növekvő módon rendezze.

**3.29. feladat**

Listázza ki minden dolgozó nevét, munkakörét, telephelyét, fizetését, valamint fizetését a saját telephelyén a munkaköre átlagfizetésének százalékában, és a fizetését a saját telephelye átlagfizetésének százalékában (a százalékértékeket egészrész kírásával), a munkakör és a telephely szerint növekvő módon rendezze.

**3.30. feladat**

Listázza ki munkakörönként azokat a dolgozókat és telephelyüket, akiknek fizetése több a munkakörük átlagfizetésénél.

**3.31. feladat**

Listázza ki a legkisebb átlagfizetésű részleg dolgozóit, ennek a részlegnek a nevét, a részleg átlagfizetését és telephelyét.

**3.32. feladat**

Listázza ki a legkisebb létszámú foglalkozási csoport dolgozóinak nevét és foglalkozását. (Természetesen az „elnök” munkakör figyelmen kívül hagyásával.)

# Interaktív környezet

## Elméleti összefoglaló

Különböző feladatok megoldásához az SQL\*Plus-környezetben változókat hozhatunk létre. E változók segítségével a felhasználó adatokat adhat meg egy SQL- vagy SQL\*Plus-utasítás (vagy mint később látni fogjuk akár egy PL/SQL program) számára, és íly módon interaktív szkript programokat készíthet. Ezek az értékek az úgynevezett felhasználói (más néven helyettesítő, vagy makró) változók segítségével eltárolhatók az SQL\*Plus-környezetben, majd az onnan való kilépéskor törlődnek. Kezelésük az & és az && jelekkel, valamint az ACCEPT, a DEFINE és az UNDEFINE utasításokkal történik.

## Felhasználói (helyettesítő) változók használata

### A felhasználói változók hivatkozási jelei (& és &&)

Az SQL\*Plus-környezetben felhasználói változókat jelölhetünk és definiálhatunk, és karakter-sorozat (CHAR típusú) értékkel láthatunk el &változó és &&változó módon.

Ha korábban e változók még nem kaptak értéket, akkor az &, illetve az && jelek adatbekérést is kezdeményeznek, azonban míg az &változó módon bekért érték nem kerül eltárolásra az SQL\*Plus-környezetben, addig az &&változó módon bekért értékét az SQL\*Plus-környezet megőrzi mindaddig, amíg ki nem lépünk az SQL\*Plus-környezetből, vagy egy UNDEFINE utasítással ki nem töröljük (lásd alább). Tehát az && jellel egy felhasználói változó definiálható, melynek értéke csak egy DEFINE vagy ACCEPT utasítással változtatható meg (lásd alább).

A már értékkel ellátott felhasználói változóra bármely SQL- vagy SQL\*Plus-utasításban hivatkozhatunk. A hivatkozás eredménye egyszerű behelyettesítés (ezért nevezik helyettesítő változónak is). Tehát, ha egy utasításban egy értékkel rendelkező helyettesítő változóra hivatkozunk, akkor az utasítás előbb felveszi a behelyettesítést követő alakot, és csak azután kerül végrehajtásra. Ennek megfelelően, bár egy & vagy && módon bekért helyettesítő változó belső tárolása karakter-sorozat formában (CHAR típusként) történik, használatánál ugyanazok a szabályok, mintha a tartalma közvetlenül konstansként lett volna beírva.

Például, ha karakteres konstansként vagy dátumkonstansként szerepel, akkor a hivatkozás módja '&változó', vagyis a hivatkozást aposztrófok (') közé kell tenni (mint például az &név hivatkozást a

```
SELECT * FROM emp WHERE ename = '&név';
```

utasításban), míg numerikus konstansként vagy függvénynévként elég közvetlenül szerepelnie &változó módon (mint például a

```
SELECT * FROM emp WHERE sal<&minfiz AND hiredate<&mainap;
```

utasításban, feltéve, hogy a minfiz változóba egy számot, a mainap nevű változóba pedig a sysdate karakter sorozatot írtuk).

További alkalmazási szabály, hogy ha a változó által tartalmazott értéket közvetlenül követné valamilyen karakter sorozat (lásd például alább a PROMPT utasításnál), akkor a hivatkozás (&változó) és az azt követő karakter sorozat közé egy pont (.) karaktert kell tenni.

Az értékkel ellátott (tárolt) felhasználói változóra kiadott &&változó módon történő hivatkozás a tárolt értéket használja fel.

## DEFINE utasítás

Felhasználói változók definiálására szolgál. Az utasítás alakja:

DEFINE változó = érték	Létrehoz egy CHAR típusú felhasználói változót, és egy értéket rendel hozzá.
DEFINE változó	Megjeleníti a változót, az értékét és az adattípusát.
DEFINE	Megjeleníti az összes felhasználói változót, értékeivel és adattípusával együtt.



### Megjegyzés

Egy DEFINE segítségével létrehozott felhasználói változó az értékét mindaddig megőrzi, amíg ki nem lépünk az SQL\*Plus-környezetből vagy egy UNDEFINE utasítással ki nem töröljük. A változó értéke csak egy újabb DEFINE, vagy ACCEPT utasítással változtatható meg, és arra &változó módon hivatkozhatunk.

## ACCEPT utasítás

Az ACCEPT utasítás definiál egy felhasználói változót, küld egy üzenetet a felhasználónak, majd a felhasználó válaszát eltárolja az általa definiált változóban (a felhasználói választ el is tudja rejteni, így alkalmas jelszó bevitelére). Az utasítás alakja:

```
ACCEPT változó [adattípus] [FORMAT 'formátummaszk'] [PROMPT 'üzenet'] [HIDE]
```

ahol:

- változó a változó neve (ha még nem létezik ez a változó, akkor létrehozza),
- adattípus NUMBER: esetleg mínusz előjelet és/vagy tizedespontot tartalmazó szám,
- CHAR: tetszőleges karakter sorozat (maximum 240 bájt),
- DATE: érvényes dátumot tartalmazó karakterlánc,

- **FORMAT 'formátummaszk'** kijelöli a használandó formátummaszkot (lásd 1. fejezet: *Dátumok és számok formázott megjelenítése c. pont*),
- **PROMPT 'üzenet'** a felhasználó adatmegadása előtt megjeleníti az üzenetet, elrejti a felhasználó által megadott adatot (jelszó).
- **HIDE**



### Megjegyzés

- Az ACCEPT utasítás az SQL\*Plus fejlesztésének egy későbbi szakaszában keletkezett, ez már (ellenértben az && vagy a DEFINE segítségével definiált változókkal) nem csupán karakteres típust rendelhet az általa definiált változóhoz. Mivel e változók belső reprezentációja már lehet NUMBER vagy DATE is, ezért egy &változó hivatkozás használatánál ezt figyelembe kell venni.
- Egy ACCEPT segítségével létrehozott felhasználói változó az értékét mindaddig megőrzi, amíg ki nem lépünk az SQL\*Plus-környezetből, vagy egy UNDEFINE utasítással ki nem töröljük. A változó értéke csak egy újabb DEFINE vagy ACCEPT utasítással változtatható meg, és arra &változó módon hivatkozhatunk.

## PROMPT utasítás

Egy szkript programból való üzenet kiírására a PROMPT utasítás önmagában is kiadható a

**PROMPT [szöveg]**

alakban, ahol a megadott szöveg megjelenik a felhasználó képernyőjén, illetve ennek hiánya esetén egy üres sor (azaz egy sora mellé).



### Megjegyzés

- Figyeljünk fel arra, hogy ebben az esetben a kiírandó szöveget nem határolja aposztróf ('') karakter.
- A PROMPT utasításban egy felhasználói változóra &változó módon hivatkozhatunk (lásd feljebb A felhasználói változók hivatkozási jelei c. pontban leírtakat).
- Ha a szövegen szerepel egy hivatkozás (&változó), akkor azt egy pont (.) karakterek kell lezárnia (például a PROMPT Ez a(z) &index.-ik utasítás hatására a 8-as számot tartalmazó index változó esetén a kiírt szöveg: Ez a(z) 8-ik).

## UNDEFINE utasítás

Az ACCEPT, a DEFINE vagy az && előtaggal definiált SQL\*Plus felhasználói változók mindaddig megőrzik értéküket, míg egy UNDEFINE utasítással nem töröljük őket, vagy ki nem lépünk az SQL\*Plus-környezetből. Az utasítás alakja:

**UNDEFINE változó**

## 4.1. példa

Írjon szkript programot, mely a felhasználó által megadott lekérdező utasítást futtatja.

### Megoldás

```
-- Futtató.sql szkript program eleje
SET numwidth 6
ACCEPT utasítás PROMPT 'Adjon meg egy lekérdező utasítást: '
SELECT * FROM (&utasítás);
SET numwidth 10
-- Futtató.sql szkript program vége
```

### Eredmény

Adjon meg egy lekérdező utasítást: select \* from emp where deptno=10  
régi 1: SELECT \* FROM (&utasítás)  
új 1: SELECT \* FROM (select \* from emp where deptno=10)

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

## Az SQL\*Plus-környezet beállítása

### A környezet lekérdezése

Az SQL\*Plus-környezet lekérdezése a

SHOW ALL

utasítással, egy rendszerváltozó lekérdezése pedig a

SHOW Rendszerváltozó

segítségével történik.

### A környezet beállítása

Az SQL\*Plus-környezet beállítása a

SET Rendszerváltozó Érték

utasítással, vagy az Opciók / Környezet menüpontban történhet.

## Az SQL\*Plus-utasítások lekérdezése (HELP)

Az SQL\*Plus környezetbeállító utasításainak lekérdezése a

`HELP SQL*Plus_utasítás`

utasítással történhet.

### 4.2. példa

Az alábbiakban bemutatjuk az SQL\*Plus környezetbeállító utasításainak használatát.

#### 1. A HELP használata

`SQL> HELP`

`HELP`

----

Accesses this command line help system. Enter HELP INDEX for a list of topics.

In iSQL\*Plus, click the Help button to display iSQL\*Plus help.

`HELP [topic]`

#### 2. Az SQL\*Plus-utasítások lekérdezése.

`SQL> HELP INDEX`

Enter Help [topic] for help.

@	COPY	PAUSE	SHUTDOWN
@@	DEFINE	PRINT	SPOOL
/	DEL	PROMPT	SQLPLUS
ACCEPT	DESCRIBE	QUIT	START
APPEND	DISCONNECT	RECOVER	STARTUP
ARCHIVE_LOG	EDIT	REMARK	STORE
ATTRIBUTE	EXECUTE	REPFOOTER	TIMING
BREAK	EXIT	REPHEADER	TTITLE
BTITLE	GET	RESERVED WORDS (SQL)	UNDEFINE
CHANGE	HELP	RESERVED WORDS (PL/SQL)	VARIABLE
CLEAR	HOST	RUN	WHENEVER OSERROR
COLUMN	INPUT	SAVE	WHENEVER SQLERROR
COMPUTE	LIST	SET	
CONNECT	PASSWORD	SHOW	

### 3. A SET utasítások lekérdezése

SQL> HELP SET

SET

---

Sets a system variable to alter the SQL\*Plus environment settings for your current session, for example:

- display width for data
- turn on HTML formatting
- enabling or disabling printing of column headings
- number of lines per page

In iSQL\*Plus, you can also use the System Variables screen to set system variables.

SET system\_variable value

where system\_variable and value represent one of the following clauses:

APP[ING] {OFF ON text}	*NEWP[AGE] {1 n NONE}
ARRA[Y] {SIZE} {15 n}	NULL text
AUTO[COMMIT] {OFF ON IMMEDIATE n}	NUMF[ORMAT] format
AUTOP[RINT] {OFF ON}	NUM[WIDTH] {10 n}
AUTORECOVERY {ON OFF}	PAGES[IZE] {24 n}
AUTOT[RACE] {OFF ON TRACE[ONLY]}	*PAU[SE] {OFF ON text}
[EXP[LAIN]] [STATISTICS]	RECSEP {WRAPPED}
BLO[CKTERMINATOR] {. c}	EA[CH] OFF}
CMDS[EP] {; c OFF ON}	RECSEPCHAR {_ c}
COLSEP {_ text}	SERVEROUT[PUT] {OFF ON}
COM[PATIBILITY] {V7 V8 NATIVE}	[SIZE n] [FORMAT]
CON[CAT] {. c OFF ON}	{WRAPPED}
COPY[COMMIT] {0 n}	WORD_WWRAPPED]
COPYTYPECHECK {OFF ON}	TRUECATED})]
DEF[INE] {& c OFF ON}	*SHIFT[INOUT] {VIS[IBLE]
DESCRIBE {DEPTH {1 n ALL}}	INVIS[IBLE])
[LINENUM {ON OFF}] [INDENT {ON OFF}]	*SHOW[MODE] {OFF ON}
ECHO {OFF ON}	*SQLBL[ANKLINES] {ON OFF}
*EDITF[ILE] file_name[.ext]	SQLC[ASE] {MIX[ED]
EMB[EDDED] {OFF ON}	LO[WER]   UP[PER])
ESC[APE] {\\ c OFF ON}	*SQLCO[NTINUE] {>   text}
FEED[BACK] {6 n OFF ON}	*SQLN[UMBER] {OFF ON}
FLAGGER {OFF ENTRY INTERMED[IATE] FULL}	SQLPLUSCOMPAT[IBILITY] {x.y[.z]}
*FLU[SH] {OFF ON}	*SQLPRE[FIX] {# c}

HEA[DING] {OFF ON}	*SQLP[ROMPT] {SQL> text}
HEADS[EP] {   c OFF ON}	SQLT[ERMINATOR]
INSTANCE [instance_path LOCAL]	{; c OFF ON}
LIN[E SIZE] {80 n} ({150 n} iSQL*Plus)	*SUF[FIX] {SQL text}
LOBOF[FSET] {n 1}	TAB {OFF ON}
LOGSOURCE [pathname]	TERM[OUT] {OFF ON}
LONG {80 n}	TI[ME] {OFF ON}
LONGC[HUNKSIZE] {80 n}	TIMI[NG] {OFF ON}
MARK[UP] HTML {ON OFF}	TRIM[OUT] {OFF ON}
[HEAD text] [BODY text] [TABLE text]	TRIMS[POOL] {ON OFF}
[ENTMAP {ON OFF}]	UND[ERLINE] {- c ON OFF}
[SPOOL {ON OFF}]	VER[IFY] {OFF ON}
[PRE[FORMAT] {ON OFF}]	WRA[P] {OFF ON}

An asterisk (\*) indicates the SET option is not supported in iSQL\*Plus.

#### 4. A numerikus mezőszélesség beállítása és hatása

```
SQL> SHOW numwidth
numwidth 10
```

```
SQL> SELECT empno, sal
  2   FROM emp;
```

EMPNO	SAL
7369	800
7499	1600

...

14 sor kijelölve.

Állítsuk át a kiírandó számjegyek számát 5-re

```
SQL> SET numwidth 5
SQL> SELECT empno, sal
  2   FROM emp;
SQL> SET numwidth 10
```

EMPNO	SAL
7369	800
7499	1600

...

14 sor kijelölve.

Megjegyezzük, hogy alapértelmezésben, ha a lista hat, vagy annál több sorból áll, akkor kiírja a listázott sorok számát. Ez a feedback rendszerváltoztató módosítható; ki- és bekapcsolható (SET feedback {6|n|OFF|ON} – lásd feljebb).

## Az SQL\*Plus formázási utasításai

Az SQL\*Plus-környezet lehetőséget ad arra, hogy az eredménylisták megjelenését beállítsuk. A formázási utasítások egy része az eredménylista egészére vonatkozik (fejléc, lábléc stb.), a másik része az egyes (karakteres, numerikus és dátum típusú) oszlopok szélességét és megjelenését állítja be. Lehetőség van a formázási utasítások letiltására, újra engedélyezésére és törlésére is.

### **LISTA SZINTŰ FORMÁZÁSOK**

TTITLE ["szöveg"   OFF   ON]	Minden oldal tetején megjelenő fejléc megadása, letiltása, engedélyezése.
BTITLE ["szöveg"   OFF   ON]	Minden oldal alján megjelentő lábléc megadása, letiltása, engedélyezése.
BREAK [ON {oszlopnév   oszlopkifejezés} [ON {oszlopnév   oszlopkifejezés}]...]	Kiszűri az ismétlődő értékek megjelenítését az egymást követő sorokban, és sortöréssel tagolja az adatsorokat.

### **OSZLOP SZINTŰ FORMÁZÁSOK**

A COLUMN utasítás az oszlopok és az oszlopfejlécek megjelenési formáját szabályozza a paraméterek megfelelően. A paraméterek megadhatók egymás után vesszővel elválasztva, vagy külön COLUMN utasításokkal is. A COLUMN utasítás szintaktikája:

```
COLUMN {oszlopnév | oszlopkifejezés}
  [FORMAT 'formátummaszk']
  [HEADING "szöveg"]
  [JUSTIFY {LEFT | CENTER | RIGHT}]
```

illetve

```
COLUMN oszlopnév {PRINT | NOPRINT | OFF | ON | CLEAR}
```

ahol:

FORMAT 'formátummaszk'	Megadja az oszlopadatok megjelenítési formáját. Karakteres adatok esetén az <i>Aszám</i> formátum (pl. A15) azt jelenti, hogy az adat számára hány karakternyi helyet engedélyezünk, numerikus és dátum típusú adatok esetén a korábban ismertetett formátummaszk elemei használhatók (lásd 1. fejezet: A dátumok és számok formázott megjelenítése c. rész).
------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**HEADING "szöveg"****JUSTIFY**

{LEFT | CENTER | RIGHT}

NOPRINT, illetve PRINT

OFF, illetve ON

CLEAR

Egy oszlop fejlécének beállítására szolgál. Alapértelmezés az oszlopnév. Ha ezen változtatni akarunk, meg kell adni a szöveget. Ha a szövegen üres vagy speciális karakter van, aposztrófok közé kell tenni. Ha a szöveg tördelőkaraktert (|) tartalmaz, akkor annak helyén a szöveget tördelve jeleníti meg.

Az *oszlopfejléc* igazítása. Alapértelmezés: számoknál jobbra igazítás, minden más típusnál balra igazítás.

Letiltja (elrejti), illetve engedélyezi az oszlop megjelenítését.

Kikapcsolja, illetve bekapcsolja az oszlophoz rendelt formázást.

A megadott oszlop formátumának törlése.

**Megjegyzés**

- Ügyeljünk arra, hogy a formázó utasításokat minden egyetlen sorba írjuk, illetve, ha több sorba írjuk, akkor az egyes sorok végére sorfolytató jelet (-) tegyünk (ugyanis ezek SQL\*Plus-utasítások, melyeknél ez követelmény – lásd 3. melléklet).
- A formázott listázások után állítsuk vissza a környezeti beállításokat, mert különben a későbbi listázásoknál is megjelenhetnek a beállított formátumok.
- Sajnálatos módon a COLUMN utasítás az Unicode karakterkészlet esetén hibásan működik még a 9.2. verzióban is! (Telepítéskor ezért az úgynevezett default karakterkészletet célszerű választani. Lásd 1. melléklet.)

**FORMÁZÁSOK TÖRLÉSE**

CLEAR {BREAKS | COLUMNS | SCREEN}

ahol:

- a CLEAR BREAKS utasítással törölhető a BREAK utasítás hatása,
- a CLEAR COLUMNS utasítással az összes oszlopformázás törölhető,
- a CLEAR SCREEN utasítással törölhető az SQL\*Plus-környezet képernyője.

**4.3. példa**

Állítsuk át a fizetés értéket a helyi pénznemre. Listázzuk ki a 30-as részlegben dolgozók nevét, fizetését és jutalékát.

**Megoldás**

```
SET feedback OFF
COLUMN sal FORMAT L99999
SELECT ename, sal, comm FROM emp WHERE deptno=30;

COLUMN sal OFF
SELECT ename, sal, comm FROM emp WHERE deptno=30;
COLUMN sal ON
```

```
COLUMN sal NOPRINT
SELECT ename, sal, comm FROM emp WHERE deptno=30;
COLUMN sal PRINT
```

```
COLUMN sal CLEAR
SET feedback ON
```

**Eredmény**

ENAME	SAL	COMM
BLAKE	Ft2850	
MARTIN	Ft1250	1400
ALLEN	Ft1600	300
TURNER	Ft1500	0
JAMES	Ft950	
WARD	Ft1250	500

ENAME	SAL	COMM
BLAKE	2850	
MARTIN	1250	1400
ALLEN	1600	300
TURNER	1500	0
JAMES	950	
WARD	1250	500

ENAME	COMM
BLAKE	
MARTIN	1400
ALLEN	300
TURNER	0
JAMES	
WARD	500

**4.4. példa**

Legyen az ename listázandó oszlop felirata "A dolgozók neve", de a neve már új sorban legyen (sortörés). Igazítsuk középre.

**Megoldás**

```
COLUMN ename HEADING "A dolgozók|neve" JUSTIFY CENTER PRINT
SET numwidth 5
SELECT * FROM emp ORDER BY ename;
SET numwidth 10
```

**Eredmény**

A dolgozók

EMPNO	neve	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
....	....	....	....	....	....	....	...

*Most igazítsuk jobbra:*

```
COLUMN ename HEADING "A dolgozók|neve" JUSTIFY RIGHT
SET numwidth 5
SELECT * FROM emp ORDER BY ename;
SET numwidth 10
```

**Eredmény**

A dolgozók

EMPNO	neve	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
....	....	....	....	....	....	....	...

**Tiltsuk le az empno és hiredate oszlop kiírását.**

```
COLUMN empno NOPRINT
COLUMN hiredate NOPRINT
SET numwidth 5
SELECT * FROM emp ORDER BY ename;
SET numwidth 10
```

**Eredmény**

A dolgozók

neve	JOB	MGR	SAL	COMM	DEPTNO
ADAMS	CLERK	7788	1100		20
ALLEN	SALESMAN	7698	1600	300	30
....	....	....	....	....	...

**Megjegyzés**

Sose feledkezzünk meg a formázási és rendszerbeállítások visszaállításáról:

```
COLUMN empno OFF
COLUMN hiredate CLEAR
CLEAR BREAKS
TTITLE OFF
```

```
BTITLE OFF
COLUMN ename CLEAR
vagy
CLEAR COLUMNS
```

## 4.5. példa

Listázza ki az emp táblából a felhasználó által megadott foglalkozású dolgozókat.

### Megoldás

```
SET linesize 50
BREAK ON job
TTITLE "Lista a foglalkozás szerinti dolgozók fizetéséről"
COLUMN ename FORMAT A15 HEADING "A dolgozó|neve" JUSTIFY LEFT
COLUMN sal FORMAT 999999999999 HEADING "A dolgozó|havi jövedelme neve"
JUSTIFY CENTER

SELECT job,
       ename,
       sal
  FROM emp
 WHERE UPPER(job) = UPPER('&foglalkozás');
```

### Eredmény

```
Adja meg a(z) foglalkozás értékét: clerk
régi 5: WHERE UPPER(job) = UPPER('&foglalkozás')
új 5: WHERE UPPER(job) = UPPER('clerk')
```

K. Okt 05	lap	1
Lista a foglalkozás szerinti dolgozók fizetéséről		
	A dolgozó neve	A dolgozó havi jövedelme neve
JOB		
CLERK	SMITH	800
	ADAMS	1100
	JAMES	950
	MILLER	1300

*Ne felejtsük el visszaállítani a beállításokat:*

```
SET linesize 400
CLEAR BREAKS
TTITLE OFF
```

```
COLUMN ename OFF
COLUMN sal OFF
CLEAR COLUMNS
```

## 4.6. példa

Listázza ki az emp táblából a felhasználó által megadott foglalkozású dolgozók főnökeinek nevét.

### Megoldás

```
SET verify OFF

SELECT dolgozó.ename AS dolgozónév,
       dolgozó.job   AS dolgozóFoglalkozás,
       dolgozó.sal    AS dolgozóFizetés,
       főnök.ename   AS főnökNév
  FROM emp főnök,
       (SELECT *
        FROM emp
       WHERE UPPER(job) = UPPER('&foglalkozás')) dolgozó
 WHERE dolgozó.mgr = főnök.empno;

SET verify ON
```

### Eredmény

Adja meg a(z) foglalkozás értékét: clerk

#### DOLGOZÓNÉV DOLGOZÓFO DOLGOZÓFIZETÉS FŐNÖKNÉV

SMITH	CLERK	800	FORD
ADAMS	CLERK	1100	SCOTT
JAMES	CLERK	950	BLAKE
MILLER	CLERK	1300	CLARK

## 4.7. példa

Listázza ki a felhasználó által megadott főnök beosztottjainak hierarchiaszintjét, azonosítóját, nevét, a főnökének kódját és részlegét.

### Megoldás

```
SELECT LEVEL   AS "Beosztási szint",
       empno    AS azonosító,
       ename    AS neve,
       mgr     AS főnök_kódja,
       deptno  AS részleg
```

```
FROM emp
CONNECT BY mgr = PRIOR empno
START WITH UPPER(ename) = UPPER('&név')
ORDER BY deptno;
```

### Eredmény

Adja meg a(z) név értékét: Jones

Beosztási szint	AZONOSÍTÓ NEVE	FŐNÖK_KÓDJA	RÉSZLEG
1	7566 JONES	7839	20
2	7788 SCOTT	7566	20
3	7876 ADAMS	7788	20
2	7902 FORD	7566	20
3	7369 SMITH	7902	20

### 4.8. példa

Listázza ki a felhasználó által megadott oszlop szerinti, valamint a felhasználó által megadott belépési időintervallumba eső dolgozók átlagfizetését.

### Megoldás

```
ACCEPT oszlop PROMPT 'Az oszlop neve: '
ACCEPT dátum1 PROMPT 'Kezdő dátum (éééé.hh.nn): '
ACCEPT dátum2 PROMPT 'Záró dátum (éééé.hh.nn): '
SELECT &oszlop,
       ROUND(AVG(sal)) AS "Átlagfizetés"
FROM emp
WHERE hiredate BETWEEN TO_DATE('&dátum1', 'YYYY.MM.DD')
                     AND TO_DATE('&dátum2', 'YYYY.MM.DD')
GROUP BY &oszlop;
```

### Eredmény

Az oszlop neve: deptno  
 Kezdő dátum (éééé.hh.nn): 1980.01.01  
 Záró dátum (éééé.hh.nn): 1990.01.01  
 régi 1: SELECT &oszlop,  
 új 1: SELECT deptno,  
 régi 4: WHERE hiredate BETWEEN TO\_DATE('&dátum1', 'YYYY.MM.DD')  
 új 4: WHERE hiredate BETWEEN TO\_DATE('1980.01.01', 'YYYY.MM.DD')  
 régi 5: AND TO\_DATE('&dátum2', 'YYYY.MM.DD')  
 új 5: AND TO\_DATE('1990.01.01', 'YYYY.MM.DD')  
 régi 6: GROUP BY &oszlop  
 új 6: GROUP BY deptno

## DEPTNO Átlagfizetés

DEPTNO	Átlagfizetés
10	2917
20	2175
30	1567

**Szkript programok felhasználóvezérelt hívása**

Bár az SQL\*Plus környezetének szkript programjai nem összetett feladatok megoldására készültek (erre a magasszintű programozási nyelvek szolgálnak; PL/SQL, Delphi, Java, C# stb.), azért van lehetőség különböző hívási szerkezetek létrehozására. Az alábbi példában ezt mutatjuk be. Ennek során fel fogjuk használni a jelen fejezetben bemutatott felhasználói változókat és az interaktivitást lehetővé tevő ACCEPT utasítást.

**4.9. példa**

Írjon egy aa.sql nevű szkript programot, mely felkínálja a felhasználónak a választást a b és a c módszer között (előbbi az ab.sql, utóbbi az ac.sql szkript program „futtatja”), majd a választól függően meghívja (a vele azonos könyvtárban lévő) megfelelő szkript programot. Ha a felhasználó válasza hibás, akkor írja ki, hogy „A válasz hibás!”, és kínálja fel újra a választást. Helyes válaszok esetén az egyes meghívott szkript programok írják ki a nevüköt, majd ezekből visszatérve az aa.sql búcsúzzon el a felhasználótól, és lépj ki az SQL\*Plus-környezetbe.

**Megoldás**

(A példabeli feladatot az alábbi öt szkript program segítségével oldottuk meg. Mindegyiket a megadott nevű fájlba kell tenni, és azzal a fájlnévvel meghívni!)

```
-- Az aa.sql szkript program eleje
PROMPT Ez az aa.sql szkript program:
DEFINE OK = 'NEM'
ACCEPT válasz PROMPT 'Melyik módszert választja (b, c): '
PROMPT
@@a&válasz
@@a&OK
-- Az aa.sql szkript program vége

-- Az ab.sql szkript program eleje
PROMPT Ez az ab.sql szkript program:
DEFINE OK = 'IGEN'
PROMPT Most az a&válasz..sql szkript program fut (ab)
-- további utasítások
PROMPT
-- Az ab.sql szkript program vége
```

```
-- Az ac.sql szkript program eleje
PROMPT Ez az ac.sql szkript program:
DEFINE OK = 'IGEN'
PROMPT Most az a&válasz..sql szkript program fut (ac)
-- további utasítások
PROMPT
-- Az ac.sql szkript program vége

-- Az aNEM.sql szkript program eleje
PROMPT
PROMPT Ez az OK = 'NEM' válaszhoz tartozó aNEM.sql szkript program:
PROMPT A(z) &válasz válasz hibás!
PROMPT
@@aa
-- Az aNEM.sql szkript program vége

-- Az aIGEN.sql szkript program eleje
PROMPT Ez az OK = 'IGEN' válaszhoz tartozó aIGEN.sql szkript program:
PROMPT Viszontlátásra!
-- Az aIGEN.sql szkript program vége
```

### Eredmény

SQL> @c:\próba\aa

Ez az aa.sql szkript program:

Melyik programot választja (b, c): x

SP2-0310: nem lehet megnyitni a(z) "ax.sql" fájlt

Ez az aNEM.sql szkript program:

A(z) x válasz hibás!

Ez az aa.sql szkript program:

Melyik programot választja (b, c): b

Ez az ab.sql szkript program:

Most az ab.sql szkript program fut (ab)

Ez az aIGEN.sql szkript program:

Viszontlátásra!

**Megjegyzés**

- A @@xxx utasítás révén hívható meg (útvonal megadása nélkül) az aktuális könyvtárban (jelen esetben a c:\próba könyvtárban) levő xxx.sql szkript program.
- Az & jelrel beillesztett környezeti változót egy közvetlen utána álló szövegtől egy pont(.) segítségével lehet elkülöníteni. Ha magára a pontra külön is szükségünk van (mint az előző példában a hivatkozott szkript program fájlnevében), akkor azt duplán kell kitennünk (lásd a&válasz..sql).
- A PROMPT utasítás használatával természetesen a fentieknél látványosabb menü is kialakítható.

# Feladatok

## 4.1. feladat

Mit csinál az alábbi szkript program?

```
SET verify OFF
```

```
SELECT * FROM &tabla1;
SELECT * FROM &&tabla2;
```

```
ACCEPT tabla3 PROMPT "A tábla neve: "
SELECT * FROM &tabla3;
```

```
DEFINE tabla4="salgrade"
SELECT * FROM &tabla4;
```

```
DEFINE
```

```
ACCEPT abc PROMPT "Táblanevek törlése: (Üsd le az Enter billentyűt...)"
UNDEFINE tabla2
UNDEFINE tabla3
UNDEFINE tabla4
UNDEFINE abc
```

```
DEFINE
```

```
SET verify ON
```

## 4.2. feladat

Listázza ki rendezve azon foglalkozási csoportok dolgozónak nevét, amelyekben a felhasználó által megadott számnál többen tartoznak. A lista fejléce legyen "Név", "Munkakör", "Munkaköri létszám".

**4.3. feladat**

Listázza ki ismétlésmentesen azon részlegek telephelyét, a részlegek tagjainak nevét és jövedelmüket, ahol az átlagjövedelem meghaladja a felhasználó által megadott értéket. A lista fejléce: Telephely, Dolgozónév, jövedelem.

**4.4. feladat**

Listázza ki ismétlésmentesen a felhasználó által megadott jövedelmi tartományba eső dolgozókat. Legyen a lista fejléce: TELEPHELY, DOLGOZÓ, MUNKAKÖR, JÖVEDELEM. A lista a telephely és a dolgozók neve szerint növekvően legyen rendezett.

**4.5. feladat**

Listázza ki ismétlésmentesen a felhasználó által megadott azonosítójú főnök (mgr) beosztottjait. Legyen a lista TELEPHELY, DOLGOZÓ, MUNKAKÖR, JÖVEDELEM, FŐNÖK\_KÓD. A lista a telephely és a dolgozók neve szerint növekvően legyen rendezett.

**4.6. feladat**

Listázza ki a felhasználó által megadott időszakban belépett dolgozókat foglalkoztató részlegeket. Legyen a lista RÉSZLEGNÉV, RÉSZLEGASONOSÍTÓ, KEZDŐDÁTUM, VÉGDÁTUM. A lista a részlegnév szerint legyen növekvő sorba rendezett.

**4.7. feladat**

Listázza ki ismétlésmentesen a felhasználó által megadott betű tartalmazó telephelyen dolgozókat. Legyen a lista fejléce: TELEPHELY, DOLGOZÓ, MUNKAKÖR, JÖVEDELEM. A lista a telephely és a dolgozók neve szerint növekvően legyen rendezett.

**4.8. feladat**

Listázza ki formázottan az emp táblából a felhasználó által megadott foglalkozású dolgozókat.

**4.9. feladat**

Listázza ki formázottan az emp táblából a felhasználó által megadott foglalkozású dolgozók főnökeinek nevét.

**4.10. feladat**

Listázza ki formázottan a felhasználó által megadott oszlop szerinti, valamint a felhasználó által megadott belépési dátum intervallumba eső dolgozók átlagfizetését.

**4.11. feladat**

Írjon szkript programot, amely formázottan listázza az emp és dept táblából a felhasználó által megadott munkakörű dolgozók telephelyét, részlegük legkisebb jövedelme és a saját jövedelme közti különbséget, valamint főnökének nevét. A lista fejléce legyen: dolgozó neve, telephelye, munkaköre, jövedelme, legkisebb jövedelem, különbség, főnök neve.

**4.12. feladat**

Írjon szkript programot, amely formázottan listázza az emp és dept táblából a felhasználó által megadott nevű főnök beosztottainak számát, telephelyét, átlagfizetését, a főnök és beosztottak átlagjövedelme közötti különbséget. A lista fejléce legyen: Főnök neve, Telephelye, Fizetése, Beosztottainak száma, átlagjövedelmük, jövedelem különbség.

**4.13. feladat**

Írjon szkript programot, amely formázottan (fej- és láblécssel stb.), valamint ismétlésmentesen listázza a dolgozók nevét, fizetését, részlegük telephelyét, annak összfizetését, valamint az abban dolgozó főnökök (mgr) nevét és fizetését a főnök neve és a dolgozók neve szerint rendezve.

## 5. FEJEZET

# Adattáblák létrehozása, módosítása, tranzakciók, megszorítások

## Elméleti összefoglaló

### DML-utasítások

A DML (Data Manipulation Language) az SQL nyelv igen fontos része. Gyakran kerül sor olyan feladatokra, mint például, „Módosítsuk a minimálbér összegét 6000 Ft-al, emeljük vagy csökkentsük az adott betéti kamatokat, töröljük azon dolgozók adatait, akik már nyugdíjba mentek, egy belépő új dolgozó adatait rögzítsük az alkalmazottak listájába stb.”

Az adattáblák tartalmát módosító (a fentiekhez hasonló) feladatok az alábbi módon csoporthatók:

- új sor beszúrása a táblába,
- létező sor egyes mezőinek megváltoztatása,
- létező sor törlése a táblából.

Mindegyik DML-utasítás megváltoztatja az adattábla tartalmát. E változásokat tranzakcióknak nevezzük. Az Oracle a tranzakciókat figyeli, és az elemi tranzakciókat megőrzi annak érdekében, hogy azok esetleg visszatéríthetők legyenek, vagyis ha egy változtatás nem megfelelő, akkor az eredeti adatai visszaállíthatók legyenek.

### Beszúrás

```
INSERT INTO táblanév [(BeszúrandóOszlopok)]  
VALUES (BeszúrandóÉrtékek);
```

ahol:

- *táblanév* a megváltoztatandó tábla neve,
- *BeszúrandóOszlopok* akkor szükséges az oszlopokat felsorolni (vesszővel elválasztva), ha nem adunk a tábla minden oszlopának értéket,
- *BeszúrandóÉrtékek* lehetnek allekérdezés eredményei is (csak ha egyetlen értéket adnak vissza), valamint helyettesítő (SQL\*Plus-) változók is (az elválasztójel itt is vessző). Hiányzó oszlopfelsorolás esetén minden oszlopnak értéket kell adni.

## Módosítás

`UPDATE táblanév`

`SET MódosítandóOszlop = ÚjÉrték  
[, MódosítandóOszlop = ÚjÉrték]`

`[WHERE LogikaiKifejezés];`

ahol:

- `MódosítandóOszlop` az az oszlop, amelynek egy vagy több értéke fog megváltozni,
- `ÚjÉrték` az oszlop értékének megfelelően, mindig csak egy érték lehet (ez az érték lehet egy bonyolultabb allekérdezés eredménye is),
- `LogikaiKifejezés` az ebben megfogalmazott feltételeknek megfelelő sorok mindenekben megtörténik a kijelölt adatmódosítás.

## Törlés

`DELETE [FROM] táblanév`

`[WHERE logikai kifejezés];`

ahol a `LogikaiKifejezés`-ben megfogalmazott feltételeknek megfelelő sorok mindenike törlésre kerül. (A `FROM` kulcsszó az utasítás tartalmának változtatása nélkül elhagyható.)

## Adatbázis-tranzakciók

Egy tranzakció DML-utasítások (elemi tranzakciók) sorozata. A tranzakció a COMMIT utasítással véglegesíthető. Amíg a változtatások nincsenek lezártva a COMMIT utasítással, addig e változtatások a ROLLBACK utasítással visszavonhatók (az adattábla adatai visszaállíthatók). A visszavonás (visszagörgetés, érvénytelenítés) hatása attól függ, hogy elhelyeztünk-e úgynevezett *mentési pontot* (SAVEPOINT). Ha igen, akkor addig tart, ha nem, akkor pedig a kiinduló állapotig (illetve az utolsó COMMIT utasítással véglegesített állapotig).

`SAVEPOINT MentésipontNeve;`

`ROLLBACK [TO [SAVEPOINT] MentésipontNeve];`

`COMMIT;`



### Megjegyzés

- A DDL-utasítások hatásai nem állíthatók vissza (lásd alább).
- Ha egy adattábla valamely (DML-utasítással végrehajtott) módosítását egy nézet-táblán keresztül hajtottuk végre, akkor a nézettábla törlése után az adat-visszaállítás nem hajtható végre! (Ezért, bár közvetlenül nem tárolnak adatot, mégis célszerű körültekintően eljárni a nézettáblák törlésekor. Lásd 6. fejezet.)
- A jogosultsággal kapcsolatos részletes ismeretek a 11. fejezetben, továbbá a [12], [16] és [25] irodalomban találhatók.

## DDL-utasítások

A táblák létrehozását, szerkezetének módosítását, oszlopok beszúrását, oszloptípusok módosítását, törlését, átnevezését végző SQL-utasításokat a DDL (Data Definition Language) nyelv tartalmazza. Megjegyezzük, hogy a DDL-utasításokat a táblákon kívül egyéb adatbázis-objektumokra (nézettábla, index, szinonima stb.) is használjuk.

### Tábla létrehozása (közvetlen létrehozás)

Az utasítás alakja:

```
CREATE TABLE [felhasználó .]táblanév
  (oszlopnév adattípus [DEFAULT kifejezés] [oszlopmegszorítások]
  [, oszlopnév adattípus [DEFAULT kifejezés] [oszlopmegszorítások]...]
  [, táblamegszorítások]);
```

ahol az *adattípus* (többek között) a következők valamelyike lehet:

NUMBER[(m [, t] )]	változó hosszúságú numerikus adat, ahol <i>m</i> a mezőszélesség, mely az esetleges előjelet és tizedespontot is tartalmazza, <i>t</i> ebből a tizedesek száma,
CHAR( <i>n</i> )	kötött hosszúságú karaktersorozat, vagyis a definícióban megadottnál rövidebb karaktersorozatokat balra tömöríti és jobbról szóköz karakterekkel tölti fel, az <i>n</i> maximális értéke 2000, alapértelmezett értéke 1,
VARCHAR2( <i>n</i> )	változó hosszúságú karaktersorozat, vagyis a definícióban megadottnál rövidebb karaktersorozatokat a tényleges hosszúságuk szerint kezeli (például jeleníti meg), az <i>n</i> maximális értéke 4000,
DATE	dátumok tárolására alkalmas típus,

továbbá az *oszlopmegszorítások* helyén egy vagy több, szóközzel elválasztott *oszlopmegszorítás*, a *táblamegszorítások* helyén pedig egy vagy több, vesszővel elválasztott *táblamegszorítás* állhat.



#### Megjegyzés

- A továbbiakban általában nem jelöljük a táblanév előtt a felhasználóra való hivatkozás lehetőségét, de azért az természetesen minden esetben lehetséges.
- Az *oszlopmegszorítás* és a *táblamegszorítás* fogalmát lásd később (jelen fejezet *Intégritási megszorítások* c. pontja).
- A további alkalmazható adattípusokat lásd [12], [16] és [26].

### Tábla létrehozása allekérdezéssel (származtatott létrehozás)

```
CREATE TABLE táblanév
AS allekérdezés;
```



### Megjegyzés

Ezzel az utasítással készíthetünk egy tábláról másolatot. Például az emp táblából egy emp1 nevű táblát a

```
CREATE TABLE emp1
AS SELECT * FROM emp;
```

utasítással hozhatunk létre. Ügyeljünk arra, hogy az így létrehozott tábla csak adattartalmában azonos az eredetivel, annak megszorításai közül azonban *nem mindegyiket* tartalmazza (lásd később).

## Tábla módosítása

### ÚJ OSZLOP HOZZÁADÁSA

```
ALTER TABLE táblanév
ADD (oszlopnév adattípus [DEFAULT kifejezés] [oszlopmegszorítások]);
```

### OSZLOP MÓDOSÍTÁSA

```
ALTER TABLE táblanév
MODIFY (oszlopnév [adattípus] [DEFAULT kifejezés] [oszlopmegszorítások]);
```

### OSZLOP TÖRLÉSE (ELDOBÁSA)

```
ALTER TABLE táblanév
DROP COLUMN oszlopnév [CASCADE CONSTRAINTS];
```



### Megjegyzés

A CASCADE CONSTRAINTS opció esetén az oszlophoz rendelt esetleges (PRIMARY KEY, FOREIGN KEY) kulcmegszorítások ellenére megtörténik a törlés (lásd később).

### OSZLOP ÁTNEVEZÉSE

```
ALTER TABLE táblanév
RENAME COLUMN RégiOszlopnév TO ÚjOszlopnév;
```



### Megjegyzés

Csak az Oracle 9i verziójától alkalmazható az oszlop átnevezése.

## TÁBLA TÖRLÉSE (ELDOBÁSA)

DROP TABLE táblanév;

## TÁBLA ÁTNEVEZÉSE

RENAME RégiTáblanév TO ÚjTáblanév;

## Integritási megszorítások

Az integritási megszorítások az adattáblára vonatkozó olyan szabályok, melyek a tábla oszlopaira vonatkoznak (függőségek). Ezek biztosítják, hogy a DML-utasítások a szabályok megsértése esetén ne hajtódjanak végre.

Az integritási megszorítások a tábladefiníció részét alkotják, de utólag is megadhatók. Utólag módosíthatók, törölhetők, és lehetőség van arra is, hogy ideiglenesen letiltsuk, illetve újra engedélyezzük.

Az integritási megszorítások, mint a nevük is mutatja, az adatbázisok integritásának (hiteleségének, megbízhatóságának) legfontosabb eszközei. Általában csak egyetlen táblára vonatkoznak (kivétel a FOREIGN KEY, azaz idegen kulcs és a REFERENCES, azaz hivatkozási megszorítás), és jellemzően a tábla minden sorára érvényes szabályok megfogalmazására alkalmasak csupán. A gyakorlatban előfordul, akár az adatbázis tervezésekor (a normalizálás során), akár már használatba vett adatbázisra vonatkozó alkalmi felhasználói igények esetén, hogy szükségesnek mutatkozik több táblára, vagy csak a tábla néhány sorára vonatkozó szabályok érvényesítése. Az ilyen feladatok megoldására használjuk (egyebek mellett) a triggereket (lásd 10. fejezet).

## MEGSZORÍTÁSOK DEFINÍCIÓJA

### Oszlopmegszorítás

A tábladefinícióban, az oszlopdefiníció részeként szerepel, de utólag is megadható. Segítségével szabályt definiálhatunk az adott oszlopra.

### Táblamegszorítás

A tábladefiníció részeként, az oszlopdefiníció után következik, de utólag is megadható. Segítségével szabályt definiálhatunk bármely oszlopra vagy oszlopok kombinációjára.

### Az oszlop- és táblamegszorítások megadása

[CONSTRAINT MegszorításNév] MegszorításTípus



#### Megjegyzés

Ha egy megszorításnak nem adunk nevet, akkor az Oracle-rendszer automatikusan generál számára egy SYS\_C00xxxx alakú nevet, ahol xxxx egy sorszám.

## A megszorítások típusai

- **NULL**      *NULL megszorítás.* Az adott oszlop tartalmazhat NULL értéket. Ez az alapértelmezés. Csak oszlopmegszorításban szerepelhet.

*Megadása:*

NULL

- **NOT NULL**      *NOT NULL megszorítás.* Az adott oszlop nem tartalmazhat NULL értéket. Csak oszlopmegszorításban szerepelhet.

*Megadása:*

NOT NULL

- **UNIQUE**      *Egyedi kulcs megszorítás.* Az oszlop minden sorában különböző értéknek kell lennie. Tartalmazhat NULL értéket is.

*Megadása táblamegszorításként:*

UNIQUE

(*vonatkozó\_oszlopnév* [, *vonatkozó\_oszlopnév*]...)

ahol a *vonatkozó\_oszlopnév* az adott tábla oszlopát jelöli, és több oszlop kijelölése esetén az egyediség csupán a megadott oszlopokból álló vektor értékeire, és nem az egyes oszlopok értékeire vonatkozik.

*Megadása oszlopmegszorításként:*

UNIQUE

- **PRIMARY KEY**      *Elsődleges kulcs megszorítás.* Az elsődleges kulcs biztosítja, hogy minden sor egyértelműen azonosítható legyen, így minden sorban különböző értékű, és nem tartalmazhat NULL értéket.

*Megadása táblamegszorításként:*

PRIMARY KEY

(*vonatkozó\_oszlopnév* [, *vonatkozó\_oszlopnév*]...)

ahol több oszlop kijelölése adja az összetett kulcsot (az egyediségre vonatkozóan lásd a fenti megjegyzést).

*Megadása oszlopmegszorításként:*

PRIMARY KEY

- **REFERENCES**      *Hivatkozási megszorítás.* Egyaránt vonatkozhat saját vagy más táblára. Az általa jelölt oszlop értékeinek szerepelnie kell a hivatkozott tábla hivatkozott (elsődleges vagy egyedi kulcs) oszlopának értékei között. Önállóan csak oszlopmegszorításban szerepelhet, egyébként az idegen kulcs megszorítás részeként.

*Megadása az idegen kulcs megszorítás részeként:*

(lásd alább az idegen kulcs megszorításnál),

*Megadása önálló megszorításként:*

REFERENCES *hivatkozott\_táblanév*

(*hivatkozott\_oszlopnév*) [ON DELETE CASCADE]

ahol az ON DELETE CASCADE megadása esetén a hivatkozási integritás biztosítása érdekében a megadott oszlop értékének törlése esetén a törlés továbbgyűrűzhet a hivatkozott oszlop vonatkozó értékére is.

- **FOREIGN KEY**      *Idegen kulcs (külső vagy másodlagos kulcs) megszorítás.*

Egy oszlopot, illetve oszlopok egy halmazát jelöli ki idegen kulcs kapcsolatként a hivatkozott tábla hivatkozott (nem feltétlenül azonos nevű, elsődleges vagy egyedi kulcs) osz-

lopára, illetve oszlopaira vonatkozóan. Az idegen kulcs megszorítás (speciális esetként) használható saját táblára is. Ekkor „idegen kulcsként” természetesen a saját elsődleges kulcsra kell hivatkozni (lásd 5.3. példa). Csak táblamegszorításban szerepelhet.

Megadása:

```
FOREIGN KEY hivatkozott_táblanév
  (vonatkozó_oszlopnév [, vonatkozó_oszlopnév]...)
  REFERENCES hivatkozott_táblanév
  (hivatkozott_oszlopnév [, hivatkozott_oszlopnév]...)
  [ON DELETE CASCADE]
```

ahol az ON DELETE CASCADE megadása esetén a hivatkozási integritás biztosítása érdekében egy saját (vonatkozó) oszlop értékének törlése esetén a törlés továbbgyűrűzhet a külső (hivatkozott) oszlopok megfelelő értékeire is.

- CHECK Feltételes megszorítás. Oszlop- és táblamegszorításban egyaránt szerepelhet.

Megadása:

```
CHECK (feltétel)
```

ahol a feltétel tetszőleges, az adott tábla oszlopaira vonatkozó logikai kifejezés lehet, és annak igaz (TRUE) értéke esetén engedélyezi az adatbevitelt, illetve módosítást. Ügyelni kell arra, hogy a megadott logikai kifejezés a teljes táblára érvényes legyen.



**Megjegyzés**  
Az adattábláknak a

CREATE TABLE táblanév AS allekérdezés;

utasítással való másolása esetén a megszorítások közül csak a NOT NULL másolódik át, a megszorítások nem.

## MEGSZORÍTÁS HOZZÁADÁSA

```
ALTER TABLE táblanév
ADD Oszlopmegszorítások;
```

## MEGSZORÍTÁS TÖRLÉSE

```
ALTER TABLE táblanév
DROP CONSTRAINT MegszorításNév] [CASCADE];
```

ahol a CASCADE opció megadása esetén minden olyan megszorítás törlődik, mely függ a törölt megszorítástól.

## **MEGSZORÍTÁSOK LEKÉRDEZÉSE**

A megszorítások adatai az adatszótár nézeteiből kaphatók meg. A megszorítások nevét és típusát a user\_constraints, a megszorítások nevét és a hozzájuk tartozó oszlopok nevét pedig a user\_cons\_columns adatszótár nézetből kérdezhetjük le.

*A lekérdezések általános alakja:*

```
SELECT constraint_name, constraint_type
  FROM user_constraints
 WHERE table_name = 'táblanév';
```

illetve

```
SELECT constraint_name, column_name
  FROM user_cons_columns
 WHERE table_name = 'táblanév';
```

## **MEGSZORÍTÁS FELFÜGGESZTÉSE (LETILTÁSA)**

Egy megszorítást ideiglenesen felfüggeszthetünk a DISABLE utasításrésszel. A CASCADE opció használata esetén egy másik táblára ható (tovagyűrűző) integritási megszorítást is felfüggeszthetünk.

*A megszorítás letiltásának általános alakja*

```
ALTER TABLE táblanév
  DISABLE CONSTRAINT MegszorításNév [CASCADE];
```

ahol a CASCADE opció megadása esetén minden olyan megszorítás felfüggesztésre kerül, mely függ a felfüggesztett megszorítástól.

## **MEGSZORÍTÁS ENGEDÉLYEZÉSE**

A tábla definíójában megadott, de felfüggesztett megszorításokat engedélyezhetjük az ENABLE utasításrésszel. (Egyedi vagy elsődleges kulcs engedélyezésekor újragenerálódik az egyedi index a táblához.)

*A megszorítás engedélyezésének általános alakja*

```
ALTER TABLE táblanév
  ENABLE CONSTRAINT MegszorításNév;
```

## 5.1. példa

Hozzuk létre az emp és a dept táblát a megfelelő megszorításokkal. (Lásd a DEMO-1.sql szkript programot a CD-mellékleten.)

### Megoldás

```

DROP TABLE EMP;
DROP TABLE DEPT;

ALTER SESSION SET NLS_DATE_LANGUAGE = ENGLISH;
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';

CREATE TABLE DEPT
  (DEPTNO          NUMBER(2) NOT NULL,
   DNAME           VARCHAR2(14),
   LOC             VARCHAR2(13),
   CONSTRAINT DEPT_PRIMARY_KEY PRIMARY KEY (DEPTNO));

INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');
INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');
INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');

CREATE TABLE EMP
  (EMPNO          NUMBER(4) NOT NULL,
   ENAME           VARCHAR2(10),
   JOB             VARCHAR2(9),
   MGR             NUMBER(4) CONSTRAINT EMP_SELF_KEY REFERENCES EMP (EMPNO),
   HIREDATE        DATE,
   SAL              NUMBER(7,2),
   COMM             NUMBER(7,2),
   DEPTNO          NUMBER(2) NOT NULL,
   CONSTRAINT EMP_FOREIGN_KEY FOREIGN KEY (DEPTNO) REFERENCES DEPT (DEPTNO),
   CONSTRAINT EMP_PRIMARY_KEY PRIMARY KEY (EMPNO));

INSERT INTO EMP VALUES (7839,'KING','PRESIDENT',NULL,'17-NOV-1981',5000,NULL,10);
INSERT INTO EMP VALUES (7698,'BLAKE','MANAGER',7839,'1-MAY-1981',2850,NULL,30);
INSERT INTO EMP VALUES (7782,'CLARK','MANAGER',7839,'9-JUN-1981',2450,NULL,10);
INSERT INTO EMP VALUES (7566,'JONES','MANAGER',7839,'2-APR-1981',2975,NULL,20);
INSERT INTO EMP VALUES (7654,'MARTIN','SALESMAN',7698,'28-SEP-1981',1250,1400,30);
INSERT INTO EMP VALUES (7499,'ALLEN','SALESMAN',7698,'20-FEB-1981',1600,300,30);
INSERT INTO EMP VALUES (7844,'TURNER','SALESMAN',7698,'8-SEP-1981',1500,0,30);
INSERT INTO EMP VALUES (7900,'JAMES','CLERK',7698,'3-DEC-1981',950,NULL,30);
INSERT INTO EMP VALUES (7521,'WARD','SALESMAN',7698,'22-FEB-1981',1250,500,30);

```

```
INSERT INTO EMP VALUES (7902,'FORD','ANALYST',7566,'3-DEC-1981',3000,NULL,20);
INSERT INTO EMP VALUES (7369,'SMITH','CLERK',7902,'17-DEC-1980',800,NULL,20);
INSERT INTO EMP VALUES (7788,'SCOTT','ANALYST',7566,'09-DEC-1982',3000,NULL,20);
INSERT INTO EMP VALUES (7876,'ADAMS','CLERK',7788,'12-JAN-1983',1100,NULL,20);
INSERT INTO EMP VALUES (7934,'MILLER','CLERK',7782,'23-JAN-1982',1300,NULL,10);
```

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

```
SET numwidth 5
SELECT * FROM EMP;
SELECT * FROM DEPT;
SET numwidth 10
```

### Eredmény

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON



#### Megjegyzés

A fenti szkript programban az emp táblát létrehozó utasításban megfogalmazott, a dept táblára vonatkozó idegen kulcs megszorítás miatt először az emp táblát kell törlőlni, csak azután törölhető a dept tábla.

## 5.2. példa

Próbálunk meg törölni egy főnököt. Miért nem sikerül?

### Megoldás

```
DELETE FROM emp
WHERE UPPER(ename) = 'BLAKE';
```

### Eredmény

```
DELETE FROM emp
```

\*

Hiba a(z) 1. sorban:

ORA-02292: intégritás megszorítás (SCOTT.EMP\_SELF\_KEY) megsértés -  
gyermek rekord található meg



#### Megjegyzés

- Ha kissé tört magyarsággal is, de a fenti hibaüzenettel azt üzeni a rendszer, hogy az EMP\_SELF\_KEY nevű, az emp tábla mgr oszlopáról az empno oszlopára mutató hivatkozási típusú megszorítás miatt a kijelölt rekord nem törölhető (Blake ugyanis egy főnök, mivel az azonosítója szerepel az mgr értékek között).
- Ahhoz, hogy a fenti törlési utasítás mégiscsak végrehajtható legyen, előtte le kellene tiltani az mgr oszlop fölötti hivatkozási megszorítást az
 

```
ALTER TABLE emp
        DISABLE CONSTRAINT EMP_SELF_KEY;
```

 utasítással, majd a törlés után természetesen újra engedélyeznünk kellene e megszorítást az
 

```
ALTER TABLE emp
        ENABLE CONSTRAINT EMP_SELF_KEY;
```

 utasítással. Ezt azonban mégse célszerű megtennünk, mivel sérülne az adattábla integritása, hiszen minden dolgozók mgr értéke, akiknek Blake volt a főnökük, most egy nem létező empno értékre mutatna. Tegyük hozzá, hogy éppen emiatt a fenti megszorítás engedélyező utasítást nem is lehetne sikeresen lefuttatni.
- Ha már mindenkorban törölni szeretnénk Blake-et, akkor ennek a korrekt módja az, hogy először a Blake beosztottjait átirányítjuk egy másik főnökhöz. Ezután már az EMP\_SELF\_KEY megszorítás érvényben hagyásával is elvégezhetjük a törlést.

## 5.3. példa

Hozzunk létre az emp és a dept táblákkal azonos adattartalmú, és célszerű megszorításokkal kiegészített emp1 és a dept1 táblákat. (Lásd a DEMO-10.sql, DEMO-11.sql és DEMO-12.sql, szkript programokat a CD-mellékleten.)

### 1. megoldás (Közvetlen létrehozás adatmegadással)

A „legegyszerűbb” megoldás az emp és dept táblákat generáló szkript „átjavítása” empl és dept1 nevekre (lásd DEMO-10.sql), illetve ennek kiegészítése néhány további megszorítással (lásd DEMO-11.sql).

### 2. megoldás (Származtatott létrehozás az emp és dept táblákból)

Egy táblamásolat előállításához az előző megoldás természetesen nem célszerű. A származtatott létrehozás esetén azonban (amint ezt feljebb már említettük) csak a NOT NULL megszorítások öröklődnek, a többi már nekünk kell külön megadni. Az alábbiakban ezt mutatjuk be, az eredeti megszorításokat néhány továbbival kiegészítve (lásd DEMO-12.sql).

```
-- 01. Az empl és dept1 táblák törlése
-- és származtatott létrehozása az emp és dept táblákból
DROP TABLE empl;
DROP TABLE dept1;
CREATE TABLE empl
    AS SELECT * FROM emp;
CREATE TABLE dept1
    AS SELECT * FROM dept;

-- 02. Az empl és dept1 táblák elsődleges kulcsainak kijelölése
ALTER TABLE empl
    ADD CONSTRAINT c_empno PRIMARY KEY (empno);
ALTER TABLE dept1
    ADD CONSTRAINT c_deptno_dept PRIMARY KEY (deptno);

-- 03. Az empl tábla idegen kulcsának kijelölése
ALTER TABLE empl
    ADD CONSTRAINT c_deptno_emp FOREIGN KEY (deptno) REFERENCES dept1
(deptno);

-- 04. A meglévő mgr oszlop hivatkozási megszorításának megadása
-- az empno kulcsra vonatkozó "idegen kulcs" megszorítással
ALTER TABLE empl
    ADD CONSTRAINT c_mgr FOREIGN KEY (mgr) REFERENCES empl (empno);

-- 05.1. Egy új mgr1 oszlop hozzáadása az empno-ra vonatkozó
-- hivatkozási megszorítással
-- ALTER TABLE empl
--     ADD CONSTRAINT c_mgr1 mgr1 REFERENCES empl (empno);

-- 05.2. Egy új mgr1 oszlop hozzáadása, majd kiegészítése
-- az empno-ra vonatkozó hivatkozási megszorítással
-- (funkcionálisan azonos a 05.1.-gyel)
```

```
ALTER TABLE emp1
  ADD (mgr1 NUMBER(4));
ALTER TABLE emp1
  ADD CONSTRAINT c_mgr1 FOREIGN KEY (mgr1) REFERENCES emp1 (empno);

-- 06. Egy új mgr2 oszlop hozzáadása típusmegadással, és
-- empno-ra vonatkozó hivatkozási megszorítással
ALTER TABLE emp1
  ADD (mgr2 NUMBER(4)) CONSTRAINT c_mgr2 REFERENCES emp1 (empno);

-- 07. A meglévő sal oszlop kiegészítése NOT NULL megszorítással
ALTER TABLE emp1
  MODIFY (sal CONSTRAINT c_sal1 NOT NULL);

-- 08. A meglévő ename oszlop kiegészítése UNIQUE megszorításokkal
ALTER TABLE emp1
  ADD CONSTRAINT c_ename UNIQUE (ename);

-- 09. A meglévő sal oszlop kiegészítése CHECK megszorításokkal
ALTER TABLE emp1
  ADD CONSTRAINT c_sal21 CHECK (sal <= 9999);
ALTER TABLE emp1
  ADD CONSTRAINT c_sal22 CHECK (sal >= 500);

-- 10. A sal oszlop NOT NULL megszorításának törlése
ALTER TABLE emp1
  DROP CONSTRAINT c_sal1;

-- 11. Egy új empnol oszlop hozzáadása típusmegadással, és
-- feltételes megszorítással
ALTER TABLE emp1
  ADD (empnol NUMBER(4)) CONSTRAINT c_empnol CHECK (empnol < 9999);

-- 12. Adatfeltöltés
UPDATE emp1
  SET mgr1 = mgr,
      mgr2 = mgr,
      empnol = empno;

-- 13. A meglévő empnol oszlop kiegészítése NOT NULL megszorítással
-- (Ezt a 12. Adatfeltöltés nélkül nem lehetne kiadni)
ALTER TABLE emp1
  MODIFY (empnol CONSTRAINT c_empnol2 NOT NULL);
```

```
-- 14. Az emp1 és deptl struktúrájának lekérdezése
SET linesize 30
DESC emp1
DESC deptl
SET linesize 400

-- 15. Az emp1 és deptl táblák listázása
SET numwidth 5
SELECT * FROM emp1;
SELECT * FROM deptl;
SET numwidth 10

-- 16. Az emp1 és deptl táblák megszorításainak lekérdezése
SET feedback OFF
BREAK ON "TáblaNeve"
SELECT SUBSTR(LOWER(user_constraints.table_name),1,10) "TáblaNeve",
       RPAD(user_cons_columns.column_name,12)          "OszlopNeve",
       SUBSTR(user_constraints.constraint_name,1,20)    "MegszorításNeve",
       RPAD(user_constraints.constraint_type,6)         "Típusa"
  FROM user_constraints, user_cons_columns
 WHERE (user_constraints.constraint_name =
        user_cons_columns.constraint_name) AND
       (LOWER(user_constraints.table_name) IN ('empl','deptl'))
 ORDER BY "TáblaNeve" DESC, "OszlopNeve" ASC;
PROMPT
CLEAR BREAKS
SET feedback 6
```

## Eredmény

A tábla módosítva.

A tábla módosítva.

14 sor módosítva.

A tábla módosítva.

Név	Üres?	Típus
EMPNO		NOT NULL NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NOT NULL NUMBER(2)
MGR1		NUMBER(4)
MGR2		NUMBER(4)
EMPN01		NOT NULL NUMBER(4)

Név	Üres?	Típus
DEPTNO		NOT NULL NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	MGR1	MGR2	EMPN01
7839	KING	PRESIDENT		81-NOV-17	5000		10			7839
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	7839	7839	7698
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10	7839	7839	7782
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20	7839	7839	7566
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	7698	7698	7654
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	7698	7698	7499
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	7698	7698	7844
7900	JAMES	CLERK	7698	81-DEC-03	950		30	7698	7698	7900
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30	7698	7698	7521
7902	FORD	ANALYST	7566	81-DEC-03	3000		20	7566	7566	7902
7369	SMITH	CLERK	7902	80-DEC-17	800		20	7902	7902	7369
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20	7566	7566	7788
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20	7788	7788	7876
7934	MILLER	CLERK	7782	82-JAN-23	1300		10	7782	7782	7934

14 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

TáblaNeve	OszlopNeve	MegszorításNeve	Típusa
emp1	DEPTNO	C_DEPTNO_EMP	R
	DEPTNO	SYS_C003759	C
	EMPNO	C_EMPNO	P
	EMPNO	SYS_C003758	C
	EMPN01	C_EMPN01	C
	EMPN01	C_EMPN02	C
	ENAME	C_ENAME	U
	MGR	C_MGR	R
	MGR1	C_MGR1	R
	MGR2	C_MGR2	R
	SAL	C_SAL21	C
	SAL	C_SAL22	C
dept1	DEPTNO	C_DEPTNO_DEPT	P
	DEPTNO	SYS_C003760	C



### Megjegyzés

A fenti táblázat három SYS\_C00xxxx nevű megszorítása még az eredeti emp és dept táblából származtatott létrehozás során öröklődött NOT NULL megszorítások.

## 5.4. példa

Írjon szkript programot, amely létrehoz egy dolgozó táblát az emp táblából, és ellátja olyan megszorítással, amely nem engedi, hogy egy salesman munkakörű dolgozó 1100 USD-nál kevesebb, vagy 2000 USD-nál magasabb havi alapbért kapjon.

### 1. lépés (A dolgozó tábla származtatott létrehozása az emp táblából)

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
```

### Eredmény

A tábla eldobva.  
A tábla létrejött.

## 2. lépés (Hibás megszorítás)

```
ALTER TABLE dolgozó
    ADD CONSTRAINT c_munkakör
        CHECK ( UPPER(job) = 'SALESMAN' AND
                sal BETWEEN 1100 AND 2000 );
```

### Eredmény

```
ADD CONSTRAINT c_munkakör
*
```

Hiba a(z) 2. sorban:

ORA-02293: (SCOTT.C\_MUNKAKÖR) nem engedélyezhető - az ellenőrzési megszorítások megsértése



### Megjegyzés

Ez így sajnos nem jó. Miért? Azért, mert a táblában már vannak adatok, és ezekkel ez a megszorítás ellentmondásban van. Utólag csak olyan megszorítást adhatunk a táblához, amely teljes mértékben összhangban van a táblában már szereplő adatokkal.

## 3. lépés (Helyes megszorítás)

A helyes CHECK megszorítás gondolatmenete: egy dolgozó vagy salesman, és korlátok között van a fizetése, vagy nem salesman, és akkor nincs korlátozva a fizetése. Ebbe belefér a KING az 5000 USD fizetésével éppúgy, mint a clerk munkakörű dolgozók a 800, illetve a 950 USD fizetéssel.

```
ALTER TABLE dolgozó
    ADD CONSTRAINT c_munkakör
        CHECK ( (UPPER(job) = 'SALESMAN' AND
                  sal BETWEEN 1100 AND 2000) OR
                  (UPPER(job) != 'SALESMAN') );
```

### Eredmény

A tábla módosítva.

## 4. lépés (A megszorítás lekérdezése)

```
SELECT SUBSTR(LOWER(user_constraints.table_name),1,10) "TáblaNeve",
       RPAD(user_cons_columns.column_name,12)          "OszlopNeve",
       SUBSTR(user_constraints.constraint_name,1,20)     "MegszorításNeve",
       RPAD(user_constraints.constraint_type,6)          "Típusa"
  FROM user_constraints, user_cons_columns
 WHERE (user_constraints.constraint_name =
           user_cons_columns.constraint_name) AND
       (LOWER(user_constraints.table_name) = 'dolgozó')
 ORDER BY "OszlopNeve";
```

## Eredmény

TáblaNeve	OszlopNeve	MegszorításNeve	Típusa
dolgozó	DEPTNO	SYS_C003780	C
dolgozó	EMPNO	SYS_C003779	C
dolgozó	JOB	C_MUNKAKÖR	C
dolgozó	SAL	C_MUNKAKÖR	C



### Megjegyzés

- A fenti táblázat két SYS\_C00xxxx nevű megszorítása itt is az eredeti emp táblából származtatott létrehozás során öröklődött NOT NULL megszorítások.
- Figyeljünk fel arra, hogy az általunk megadott c\_munkakör nevű, összetett CHECK megszorítás két megszorításként jelenik meg az adatszótárban. Ennek oka az, hogy valóban két oszlopra adtunk meg korlátozást.

## Feladatok

Az alábbi feladatok egymásra épülnek, ezért célszerű az adott sorrendben megoldani őket, különben nem minden értelmezhető a feladat és látható az eredmény.

### 5.1. feladat

Hozzon létre az emp táblából egy dolgozó nevű táblát, és nevezze át annak empno oszlopát azonosító névre. Kérdezze le az új tábla szerkezetét.

### 5.2. feladat

Hozzon létre egy dolgozó nevű táblát az emp tábla azon dolgozóiból, akiknek 500 USD-nál több a jövedelmük, és 1980. január 1-je után léptek be. (Megjegyezzük, hogy az 5.2–5.8. feladatok összefüggő feladatsort alkotnak, ezért értelmezésük, megoldásuk és a megoldások végrehajtása csak ebben a sorrendben lehetséges!)

### 5.3. feladat

Nevezze át a dolgozó nevű tábla oszlopneveit értelemszerűen a következőkre: AZONOSÍTÓ, NÉV, MUNKAKÖR, FŐNÖK\_ID, BÉLÉPÉS, FIZETÉS, JUTALÉK, RÉSZLEG\_ID.

### 5.4. feladat

Bővítsse a dolgozó táblát a lakkely oszloppal.

### 5.5. feladat

Tölts fel a dolgozó tábla lakkely oszlopát a következőképpen:

A. A New York-i dolgozók lakkelye is New York, de King a Long Beach-en lakik.

- B.: A Dallas-i clerk munkakörük Garland-ban, a többiek Dallas-ban laknak.
- C.: A Chicago-ban dolgozók Boston-ban, a Boston-ban dolgozók pedig Chicago-ban laknak, kivéve azokat a Chicago-i dolgozókat, akiknek Blake a főnöke, mert ők Indianapolisban laknak, feltéve, hogy nem clerk munkakörük, mert akkor sehol sem laknak.
- D.: Listázza ki a dolgozók azonosítóját, nevét, munkakörét, főnökének nevét, a telephelyét és a lakhelyét a dolgozók neve szerint rendezve.

### 5.6. feladat

Hozzon létre egy részleg táblát a dept táblából. Bővítsé a dolgozó táblát Kovács-csal, akinek a foglalkozása főnök, 9999 az azonosítója, neki nincs főnöke, de a King nevű alkalmazott a beosztottja, 2001. január 5-én lépett be a céghoz, 9999 USD a fizetése, ugyanennyi a jutaléka, a 99-es azonosítójú Főnökség a munkahelye Budapest-en, és Soroksár-on lakik. Ehhez a bővítéshez tegyen mentési pontot.

### 5.7. feladat

Bővítsé a dolgozó táblát a következő alkalmazottal:

- Neve: Kelemen;  
 Azonosítója: 1111;  
 Munkaköre: megegyezik a legkisebb fizetésű dolgozó munkakörével,  
 Főnöke: megegyezik a Dallas-i Ford főnökével,  
 Belépése: az aktuális dátum,  
 Fizetése: a legnagyobb, 5000 USD-nál kisebb munkaköri átlagfizetés,  
 Jutaléka: 111 USD,  
 Részlege: megegyezik a Budapest-i részleg számával,  
 Lakóhelye: Soroksár.

A bővítéshez tegyen mentési pontot.

### 5.8. feladat

Bővítsé a dolgozó táblát két dolgozával. Az adatok egyéni legyenek. Listázza ki a táblát. Érvénytelenítse az eddigi bővítéseket, majd végegesítse az adattáblát.

### 5.9. feladat

- A.: Írjon szkript programot, amely (eldob és) létrehoz egy dolgozó és egy részleg táblát az emp és dept táblákból, majd az alábbi megszorításokkal látja el. Megkísérel minden feltételhez egy-egy rekordot felvinni a dolgozó és részleg táblába, amelyek közül egy megfelel, egy pedig nem felel meg e megszorításoknak.
- B.: Legyen ename egyedi kulcs a dolgozó táblában, és kérdezze le e megszorítást.
- C.: Törölje a dolgozó táblából a névre vonatkozó egyedi megszorítást.
- D.: Legyen az empno oszlop elsődleges kulcs a dolgozó táblában.
- E.: Legyen a deptno elsődleges kulcs a részleg, és idegen kulcs a dolgozó táblában.
- F.: Egy részleg csak New York, Dallas, Chicago vagy Boston valamelyikében lehet.
- G.: Egy manager nem kaphat 2000 USD-nál kevesebb, és 3000 USD-nál több fizetést.
- H.: Egy clerk nem kaphat 500 USD-nál kevesebb és 1500 USD-nál több fizetést.
- I.: Az új dolgozó belépési dátuma 1978 és a legkésőbbi dátum közé essen.
- J.: Az új dolgozó belépési dátuma 1981. február 19. és 2004. október 25. közé essen.

### **5.10. feladat**

Hozza létre az emp táblával azonos dolgozó táblát. Növelje meg ebben a részlegük átlagfizetésénél alacsonyabb fizetésű dolgozók havi bérét saját fizetésük 20%-ával.

### **5.11. feladat**

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát. Ebben törölje a legjobban fizetett Dallas-i dolgozót.

### **5.12. feladat**

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát. Bővíts ezt egy jövedelem névű oszloppal, melyben összegzi a fizetést és a jutalékot. Ezután listázza e tábla tartalmát elsődlegesen a jövedelem szerint csökkenően, másodlagosan pedig a dolgozók neve szerint növekvő módon rendezve.

### **5.13. feladat**

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát, majd minden olyan dolgozónak, akinek legalább egy beosztottja van, egyszeri 1000 USD jutaléket ad (comm), végül listázza ki a dolgozók tábláját elsődlegesen a jutalék szerint csökkenően, másodlagosan pedig a dolgozók neve szerint növekvő módon rendezve.

### **5.14. feladat**

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát, majd megnöveli a főnökök (mgr) fizetését 10%-kal, végül listázza ki őket (csak a főnököket) oly módon, hogy elsődlegesen a foglalkozásuk szerint csökkenően, másodsorban a nevük szerint növekvően legyenek rendezve.

### **5.15. feladat**

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát, majd megnöveli a Chicago-i és a New York-i dolgozók havi fizetését a kereskedők jutalékának átlagával. A megoldáshoz használjon nézettáblát. Listázza ki a módosított táblát.

### **5.16. feladat**

Írjon szkript programot, amely új adattáblát hoz létre az emp és a dept táblákból dolgozó és részleg néven. (E táblák létrehozása előtt győződjön meg róla, hogy ezeken nem létezik-e már másik objektum, és ha igen, akkor törölje azt.) Lássa el e táblákat az alábbi megszorításokkal:

- Legyen az empno egyedi attribútum a dolgozó táblában.
- Legyen a deptno elsődleges kulcs a részleg táblában, idegen kulcs a dolgozó táblában oly módon, hogy egy részleg törlése esetén törlődjenek e részleg dolgozói is. E megszorítás helyes működésének ellenőrzéseként töröljön egy részleget, listázza ki ennek eredményét, majd vonja vissza a törlést, és ellenőrizze a visszavonás hatását.
- Egy új dolgozó csak a szkript programban megadott, 700 és 7000 USD értéktartománybeli fizetést kaphasson. E megszorítás helyes működésének ellenőrzéseként kíséreljen meg egy megfelelő és egy hibás rekordot felvinni. Listázással ellenőrizze ezeknek a műveleteknek a végrehajtását.

A fentiek után állítsa vissza, majd véglegesítse az adattáblákat.

**5.17. feladat**

Írjon szkript programot, mely minden dolgozónak, akinek legalább egy olyan munkatársa van, aki ugyanazon telephelyen, úgyanolyan munkakörben dolgozik, egyszeri 500 USD jutaléket ad (comm), majd pedig listázza a dolgozók tábláját elsődlegesen a jutalék szerint csökkenően, másodlagosan pedig a dolgozók neve szerint növekvő módon rendezve.

**5.18. feladat**

Írjon szkript programot, mely megnöveli minden dolgozónak fizetését (sal) egy, a felhasználó által megadott értékkel, akiknek nincs, vagy nulla (0) a jutaléku, foglalkozási csoportjukban egynél többen vannak, valamint Dallas-ban vagy Chicago-ban dolgoznak. A szkript program végül listázza a dolgozók tábláját elsődlegesen a fizetés szerint csökkenően, másodlagosan pedig a dolgozók neve szerint növekvő módon rendezve.

## 6. FEJEZET

# Nézettáblák, felső-N analízis, ROWNUM

## Elméleti összefoglaló

A nézettábla (vagy röviden csak nézet) olyan objektum, amelyet a rendszer bár táblaként kezel, mégsem tárol fizikai adatokat. A nézettábla egy (esetenként több adat- vagy más nézettáblára hivatkozó) SELECT utasítást tárol, és amikor a nézettáblára történik hivatkozás, minden a benne tárolt SELECT utasítás hajtódik végre. Legfontosabb előnyei:

- eltárolhatók benne a gyakori lekérdezések,
- bonyolult lekérdezések építhetők fel egyszerűbb nézettáblák használatával,
- nézetekkel korlátozni lehet az eredeti adattáblákhoz való hozzáférést (mivel csak a nézetben szereplő oszlopokhoz lehet hozzáférni).

## Nézettábla létrehozása és törlése

### A nézettábla létrehozása

```
CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW NézetNév  
      [(MásodlagosNév [, MásodlagosNév]...)]  
      AS allekérdezés  
      [ WITH READ ONLY | WITH CHECK OPTION];
```

ahol:

- |                   |                                                                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| ▪ OR REPLACE      | ha már van ilyen nevű nézet, akkor újra létrehozza,                                                                                       |
| ▪ FORCE           | a nézetet mindenképpen létrehozza, akkor is, ha a benne hivatkozott fizikai tábla nem létezik,                                            |
| ▪ NOFORCE         | a nézetet csak akkor hozza létre, ha a lekérdezett táblák minden egyike létezik (ez az alapértelmezés),                                   |
| ▪ NézetNév        | a felhasználó által megadott név,                                                                                                         |
| ▪ MásodlagosNév   | hivatkozási nevek zárójelben (a nézetben megjelenő másodlagos nevek számának, és az allekérdezés kimenő oszlopai számának egyeznie kell), |
| ▪ AS allekérdezés | a nézettábla „tartalmát” előállító SELECT utasítás,                                                                                       |

- WITH READ ONLY a nézeten csak lekérdezéseket hajthatunk végre, adatmanipulációt (DML-utasításokat) nem,
- WITH CHECK OPTION használatával lehetőségünk van a hivatkozási integritás ellenőrzésére is. (Az utasításrész nem engedi, hogy nézeten keresztül végrehajtott INSERT, UPDATE utasítások olyan sorokat hozzanak létre, amelyeket a nézet nem választhat ki.)

#### A nézettábla törlése

DROP VIEW NézetNév;



#### Megjegyzés

- Az allekérdezésben az oszlopoknak másodlagos név adható, de célszerű az idézőjel használatát kerülni, mivel ekkor az is azonosító karakternek minősül (vagyis idézőjel használata esetén az erre való hivatkozásnak is tartalmaznia kell az idézőjelet).
- Az allekérdezés lehet egyszerű vagy összetett, több egymásba ágyazott lekérdezési blokkból is állhat. Az allekérdezés WHERE, illetve HAVING utasításrésze választja ki azokat a sorokat, amelyek a nézetben szerepelni fognak.
- Ha a WITH READ ONLY utasításrész nem szerepel, akkor a nézettáblán keresztül módosíthatók az eredeti adattáblabeli értékek.
- Egy nézetre vonatkozó UPDATE, DELETE és INSERT utasítás akkor működik (azaz akkor továbbítja az adatmódosítást az általa hivatkozott táblába), ha egyetlen táblára épül, és *nem tartalmaz GROUP BY, DISTINCT utasításrészt, csoportfüggvényeket, nem hivatkozik pseudooszlopra, és nem tartalmaz kifejezést*.
- A nézettáblán keresztül történő adatbevitelt *ellenőrzött vagy alséma szerinti ellenőrzött* adatbevitelnek nevezük, ha teljesülnek az iménti feltételek (azaz engedélyezett és lehetséges az adatbevitel nézettáblán keresztül). Ekkor a nézettábla séma (oszlopneveinek halmaza) a hivatkozott adattábla *alsémája*, ha a hivatkozott adattábla sémájának valamely részhalmaza. Az ilyen adatbevitel révén megakadályozható a *jogosulatlan adatmódosítás* nézettáblán keresztül, vagyis a nézettábla felhasználója csak olyan oszlopok adatait „látja” (tudja lekérdezni) és tudja beinni, illetve módosítani, amelyek szerepelnek az alsémájában. Ha a WITH CHECK OPTION kulcskifejezést is használjuk, akkor a WHERE utasításrészben megadott *kiválasztási* feltételek az adatbevitel, illetve módosítás esetén *ellenőrzési* feltételekké válnak.
- A nézettábla sok tekintetben úgy viselkedik, mint a hagyományos programozásban az eljárás. Nagyon fontos, hogy hatékony lekérdezést tartalmazzon, mivel a nézettáblát tipikusan összetett lekérdezések „allekérdezéseként” alkalmazzuk (bár – mint említettük – ellenőrzött adatbevitelre is jól használható). A hatékonyság iránti igény miatt a nézettáblába általában nem teszünk rendezést. Ez alól egyetlen kivétel a felső-N analízis (lásd alább).

- A létrehozott nézettáblákat a feladatmegoldások végén „illik” törölni, mivel az Oracle azokat megőrzi. (A gyakorlati alkalmazások során a „beépített” nézettáblákat természetesen nem töröljük!)
- Ha nézettáblán keresztül hajtunk végre DML-utasítást (adatbevitelt, -módosítást vagy -törölést), és később (például a feladatmegoldás végén) törölni szeretnénk a létrehozott nézeteket, és vissza szeretnénk állítani az adattábla korábbi értékeit, akkor ügyeljünk arra, hogy először állítsuk vissza az adatokat (ROLLBACK), és csak utána törljük a nézeteket. Fordított esetben ugyanis nem lesz sikeres az adat-visszaállítás!

## Inline nézet

A másodlagos névvel ellátott allekérdezést nevezzük inline nézettel. Ez az allekérdezés a FROM utasításrészben található. (Ezzel az allekérdezések témaköre foglalkozik, lásd 3. fejezet.)

## Felső-N analízis

Gyakran van szükség valamilyen szempont szerint rendezett sorok közül az első vagy az utolsó valahány kiválasztására. Például egy tábla valamely oszlopának legnagyobb vagy legkisebb értékére szerint keressük az első vagy utolsó  $n$  sort. (Ilyenkor tulajdonképpen egy úgynevezett toplistát készítünk.) Az SQL-ben ezt a feladatot megoldhatjuk a

```
SELECT { [oszloplista , ROWNUM] | [oszloplista] }
  FROM [táblalista ,] allekérdezés
 WHERE ROWNUM <= n;
```

utasítással, ahol:

- allekérdezés a szükséges rendezést tartalmazó allekérdezés,
- ROWNUM pszeudooszlop, mely a kiválasztásra kerülő sorokhoz hozzárendeli a kiválasztásuk sorszámát,
- $n$  a kiválasztott sorok száma.

### 6.1. példa (A ROWNUM használata)

Listázza ki az ábécé szerinti első három dolgozót.

#### 1. megoldás (Hibás!)

```
SELECT *
  FROM emp
 WHERE ROWNUM = 3;
```

#### Eredmény

nincsenek kijelölve sorok



### Megjegyzés

A ROWNUM nem sorszám, hanem a lekérdezés eredményének sorrendjét adja meg. Ezért egyenlőség nem állhat a WHERE feltételben (csak a <, <=, >, vagy >= relációjelek valamelyike).

### 2. megoldás (Hibás!)

```
SELECT ROWNUM AS sorszám,
       ename   AS név
  FROM emp
 WHERE ROWNUM <= 3;
```

#### Eredmény

SORSZÁM	NÉV
1	KING
2	BLAKE
3	CLARK



### Megjegyzés

- Ez már működik, az eredmény három név, de nincs rendezve névsor szerint.
- Rendezzük névsor szerint az emp táblát, és listázzuk annak első 3 sorát. Ekkor azonban előbb kellene állnia az ORDER BY ename utasításrésznek, mint a WHERE ROWNUM <= 3 utasításrésznek, ezt pedig a SELECT utasítás szintaktikája nem engedi meg.
- A másik lehetőség az, hogy a szintaktikai követelményeknek eleget teszünk. Nézzük!

### 3. megoldás (Hibás!)

```
SELECT ROWNUM AS sorszám,
       ename   AS név
  FROM emp
 WHERE ROWNUM <= 3
 ORDER BY ename;
```

#### Eredmény

SORSZÁM	NÉV
2	BLAKE
3	CLARK
1	KING



### Megjegyzés

- Ez a lekérdezés csak a tárolás szerinti első három sort rendezi, mely természetesen megint csak hibás.

- Azt az ellentmondást, hogy előbb kellene rendezni, és csak utána kiválasztani az első hármat, csak allekérdezéssel lehet feloldani. Ezt mutatja az alábbi, most már jó megoldás.

#### 4. Jó megoldás

```
SELECT ROWNUM AS sorszám,
       ename AS név
  FROM (SELECT ename
           FROM emp
          ORDER BY ename)
 WHERE ROWNUM <= 3;
```

#### Eredmény

SORSZÁM	NÉV
1	ADAMS
2	ALLEN
3	BLAKE



#### Megjegyzés

A megoldás tehát a következő: az adattáblát egy allekérdezésben rendezzük. Ennek eredményét a külső lekérdezés e rendezés sorrendjében kapja meg. Végül pedig a ROWNUM pszeudooszlopra vonatkozó feltétellel a generálás sorrendje szerinti első három elemet listázzuk. Az alábbi szkript programmal megadhatjuk, hogy az ábécé szerint rendezett dolgozók közül az első hányat kívánjuk listázni.

#### 5. megoldás (Szkript program nézettáblával)

```
ACCEPT darab PROMPT 'Hány dolgozót kíván listázni az ábécé elejéről: '
```

```
CREATE OR REPLACE VIEW DolgozóLista
  (Sorszám, Név, Munkakör, Fizetés)
```

```
AS
```

```
  SELECT ROWNUM, al.* 
    FROM (SELECT ename, job, sal
           FROM emp
          ORDER BY ename) al
   WHERE ROWNUM <= &darab;
```

```
SELECT * FROM DolgozóLista;
```

```
DROP VIEW DolgozóLista;
```

```
UNDEFINE darab
```

## Eredmény

Hány dolgozót kíván listázni az ábécé elejéről: 4

régi 8: WHERE ROWNUM <= &darab

új 8: WHERE ROWNUM <= 4

A nézet létrejött.

SORSZÁM	NÉV	MUNKAKÖR	FIZETÉS
1	ADAMS	CLERK	1100
2	ALLEN	SALESMAN	1600
3	BLAKE	MANAGER	2850
4	CLARK	MANAGER	2450

# Feladatok

A feladatokat minden szkript állományba írjuk és abból futtassuk. Egy tábla vagy nézettábla létrehozása előtt minden ellenőrizzük, hogy létezik-e az SQL\*Plus-környezetünkben ezen a néven valamelyen objektum, és ha igen, akkor adjuk ki a rá vonatkozó törlési utasítást.

### 6.1. feladat

Hozza létre az emp és dept adattáblából a dolgozó és részleg nevű adattáblákat, és ezekre hivatkozóan a Dolgozónézet és Részlegnézet nevű nézettáblákat, majd listázza ez utóbbitakat.

### 6.2. feladat

Hozza létre az emp tábla 500 és 1500 USD közötti jövedelemmel rendelkező dolgozónak nevéből, munkaköréből és jövedelméből a Szegény Dolgozók nézettáblát, jövedelem szerint növekvően rendezve listázza ki, majd törölje.

### 6.3. feladat

Listázza ki a négy legjobban és a négy legrosszabbul kereső dolgozó azonosítóját, nevét, foglalkozását, fizetését és telephelyét az emp és dept táblák alapján.

### 6.4. feladat

Futtassa le a 6.1. feladatban létrehozott szkript programot, és ellenőrizze a megfelelő adatszótárbeli nézetekből (user\_objects, user\_catalog, user\_views), hogy létezik-e valóban a dolgozó és a részleg adattábla, valamint a Dolgozónézet és a Részlegnézet nézettábla.

### 6.5. feladat (Adatbevitel nézettáblán keresztül)

Hozza létre (ha még nem létezik) a dolgozó táblát az emp táblából, és végezzen egy nézettáblán keresztül történő, úgynevezett ellenőrzött adatbevitelt.

A nézettábla neve legyen ūrlap, és a sémája (oszlopneveinek halmaza) a dolgozó tábla minden oszlopnevét tartalmazza a jutalék (comm) kivételével.

A kiválasztási feltételek, és egyben ellenőrzési feltételek legyenek a következők:

- csak hivatalnok (clerk) legyen felvihető,
- a főnökkódja csak a 7788 és a 7902 valamelyike lehessen,
- fizetése 800 és 1200 USD között legyen,
- a részlegazonosítója vagy a 10, vagy a 20 valamelyike lehessen.

Ezekután bővítsse a dolgozó adattáblát az ūrlap nézettáblán keresztül az ABSOLON nevű dolgozóval (a többi érték az engedélyezett tartományon belül legyen tetszőleges). Végül törölje az ūrlap nézettáblát.

#### **6.6. feladat**

Hozza létre a négy legrosszabb dolgozó ázonosítóját, nevét, foglalkozását, fizetését és telephelyét tartalmazó nézettáblát az emp és a dolgozó táblára vonatkozóan (lásd 6.3. feladat), majd listázza ki őket. Indokolja az eredményt.

#### **6.7. feladat**

Hozza létre úgy a 6.5. feladatbeli ūrlap nézettáblát, hogy rajta keresztül csak lekérdezni lehessen a dolgozó tábla adatait, de módosítani vagy bővíteni már ne.

#### **6.8. feladat**

Listázza ki a dolgozó tábla két legmagasabb átlagjövedelmű olyan munkaköri csoportjának nevét, amelyekben a dolgozóknak van főnökük.

#### **6.9. feladat**

Listázza ki a dolgozó tábla azon dolgozónak nevét, munkakörét és fizetését, aikik a legmagasabb átlagjövedelmű olyan munkaköri csoportban dolgoznak, amelyben minden dolgozónak van főnöke.

#### **6.10. feladat**

Írjon egy olyan nézettáblát létrehozó és listázó szkript programot a dolgozó táblára vonatkozóan, amely azokat a Chicago-i dolgozókat tartalmazza, akiknek főnöke 1980 és 1981 között lépett be. A nézettábla tartalmát elsőlegesen a munkakör, másodlagosan a név szerint rendezve listázza. Ennek során kerülje a munkakör ismétlődéseit.

#### **6.11. feladat**

Készítsen az emp tábla alapján egy olyan nézettáblát, amely meghatározza az egyes fizetési osztályokban a dolgozók számát, valamint a legalacsonyabb és legmagasabb fizetéseket, ahol a fizetési osztály 1000 USD alatt 1, 2000 USD alatt 2 stb. Ezután listázza ki ezt a nézettáblát a fizetési osztályok szerint csökkenően rendezve.

#### **6.12. feladat**

Készítsen az emp tábla alapján egy Dolgozó1 nézettáblát, amelynek oszlopai a dolgozók neve, munkaköre, fizetése és fizetési osztályának sorszáma, és amely a FizetésOsztály nézettábla

(lásd az előző feladatot) segítségével besorolja a dolgozókat a fizetésük szerint. Listázza ki ezt a nézettáblát elsődlegesen a fizetési osztály, másodlagosan a dolgozók neve szerint rendezve.

### 6.13. feladat

Írjon szkript programot, amely a dolgozó táblából töri a legjobban fizetett Dallas-i dolgozót. A törlés sikereségéről listázással győződjön meg, majd állítsa vissza az eredeti táblatartalmat. A megoldáshoz használjon nézettáblát, melyet a feladatmegoldás után töröljön.

### 6.14. feladat

Írjon szkript programot, amely a dolgozó táblában megnöveli Blake beosztottjainak fizetését az átlagfizetésük 15%-ával. Az adatmódosítás sikereségéről listázással győződjön meg, majd állítsa vissza az eredeti táblatartalmat. A megoldáshoz használjon nézettáblát, melyet a feladatmegoldás után töröljön.

### 6.15. feladat

Írjon nézettáblát használó szkript programot, mely a felhasználó által megadott értékkel megnöveli a dolgozó táblában minden fizetését, akiknek nincs, vagy nulla a jutalékuk, foglalkozási csoportjukban egynél többen vannak, és Dallas-ban vagy Chicago-ban dolgoznak. A szkript program ezután listázza ki a dolgozók tábláját elsődlegesen a fizetés szerint csökkenően, másodlagosan pedig a dolgozók neve szerint növekvő módon rendezve, majd állítsa vissza az eredeti táblatartalmat, végül törölje a létrehozott nézettáblát.

### 6.16. feladat

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos, dolgozó nevű táblát, és bővítsé ki azt egy Prémium nevű oszloppal. Az egyes dolgozók Prémium értéke legyen a saját telephelyük átlagjutalékának 10%-a. Használjon nézettáblát.

### 6.17. feladat

Írjon szkript programot, amely létrehoz az emp táblából egy dolgozó táblát. Növelje meg ebben 10%-kal a legkisebb átlagfizetésű részleg dolgozóinak fizetését, és listázza ki e részleg nevét, az itt dolgozók nevét, valamint azok régi és új fizetését. A megoldáshoz használjon nézettáblát.

### 6.18. feladat

Írjon szkript programot, amely létrehoz az emp táblából egy dolgozó táblát. Növelje meg ebben 20%-kal a három legrosszabb fizetett dolgozó fizetését az eredeti fizetés oszlopban, és listázza ki a dolgozó táblát. A megoldáshoz használjon nézettáblát.

### 6.19. feladat

Írjon szkript programot, mely minden részlegben megnöveli a részleg-átlagfizetésnél alacsonyabb jövedelmű dolgozók fizetését, az ottani legnagyobb és legkisebb jövedelem közötti különbség 10%-ával. A szkript program végül listázzon is elsődlegesen a részlegazonosító szerint növekvően, másodlagosan pedig a jövedelem szerint csökkenően rendezve.

**6.20. feladat**

Írjon szkript programot, mely foglalkozásonként megnöveli az azon belül legalacsonyabb fizetésű dolgozók fizetését, az összes dolgozó átlagfizetése és foglalkozási csoportjuk átlagfizetése közötti különbség 20%-ával, ha ez a különbség pozitív érték. A szkript program végül listázzon is elsődlegesen a foglalkozás szerint növekvően, másodlagosan pedig a jövedelem szerint csökkenően rendezve.

**6.21. feladat**

Írjon szkript programot, mely egyszeri 999 USD jutalékot ad (comm) minden olyan dolgozónak, akinek legalább kettő közvetlen beosztottja van, majd pedig listázza a dolgozók tábláját elsődlegesen a jutalék szerint csökkenően, másodlagosan pedig a dolgozók neve szerint növekvő módon rendezve.

**6.22. feladat**

Írjon szkript programot, mely minden részlegben megnöveli a két legalacsonyabb fizetésű dolgozó fizetését, a részleg legnagyobb és legkisebb fizetése közötti különbség 15%-ával. A szkript program végül listázzon is elsődlegesen a részlegazonosító szerint növekvően, másodlagosan pedig a fizetés szerint csökkenően rendezve.

**6.23. feladat**

Írjon szkript programot, mely minden (nem president) munkakörben megnöveli a két legkisebb fizetésű dolgozó fizetését, a munkakör legnagyobb és legkisebb fizetése közötti különbség 5%-ával. A szkript program végül listázzon is elsődlegesen a munkakör szerint növekvően, másodlagosan pedig a fizetés szerint csökkenően rendezve.

## 7. FEJEZET

# Összetett SQL-feladatsorok

Az alábbi feladatsorok abból a célból készültek, hogy az olvasó ellenőrizesse az eddig tanultakat. minden feladatsorozat közel azonos nehézségű, azok egyes részfeladatai között előfordulnak egyformák is.

Mivel e feladatsorok egyes feladatai igen hasonlók a korábbi fejezetek példáihoz, feladataihoz, ezért a megoldásrészben csak azok megoldásait mutatjuk be, melyek vagy bonyolultságukban, vagy a megoldásukhoz szükséges ötletekben valami újat tartalmaznak.

## Elméleti összefoglaló

E fejezet új elméleti ismeretre nem hivatkozik, ezért „elméletként” az alábbiakban egy mintafeladatsort mutatunk be annak megoldásával együtt.

## Mintafeladatsor

### M1. feladat

Írjon szkript programot, amely új adattáblát hoz létre az emp és a dept táblákból dolgozó és részleg néven. (E táblák létrehozása előtt győződjön meg arról, hogy ezeken nem létezik-e már másik objektum, és ha igen, akkor törölje azt.) Lássa el a táblákat az alábbi megszorításokkal:

- Legyen az empno egyedi attribútum a dolgozó táblában.
- Legyen a deptno elsődleges kulcs a részleg táblában, idegen kulcs a dolgozó táblában oly módon, hogy egy részleg törlése esetén törlődjönek e részleg dolgozói is. E megszorítás helyes működésének ellenőrzéseként töröljön egy részleget, listázza ki ennek eredményét, majd vonja vissza a törlést, és ellenőrizze a visszavonás hatását.
- Egy új dolgozó csak a szkript programban megadott értéktartománybeli fizetést kaphasson. Ellenőrzésként kíséreljen meg egy megfelelő és egy nem megfelelő rekordot felvinni.

Listázással győződjön meg e műveletek helyes végrehajtásáról.

A fentiek után állítsa vissza az adattáblákat, és ellenőrizze a visszaállítás hatását. A továbbiakban ezeket a táblákat használja.

## M2. feladat

Listázza a dolgozók nevét, jövedelmét és telephelyét, továbbá a főnökük nevét, fizetését és telephelyét elsődlegesen a főnök neve szerint csökkenően, másodlagosan a dolgozó neve szerint növekvően rendezve. (Amely dolgozónak nincs főnöke, annál a főnök neveként a "Nincs főnök" szöveget írja ki.)

## M3. feladat

Írjon szkript programot, amely listázza az egyes részlegek telephelyét, nevét, az ezekben dolgozók átlagjövedelmét (a kiíratás e három szerint legyen ismétlésmentes), főnökeik nevét, fizetését, valamint telephelyét, elsődlegesen a részleg telephelye szerint, másodlagosan pedig a főnökök neve szerint rendezve. A megoldáshoz használjon nézettáblákat, és ezeket a szkript program végén törölje.

## M4. feladat

Végezzen ellenőrzött, azaz nézettáblán keresztül történő adatfelvitelt a dolgozó táblába.

- Ennek érdekében hozzon létre egy Úrlap nevű nézettáblát, mely megakadályozza a jutalék (comm) mezőnek való értékadást, de minden más adat bevitelét engedélyezi a következő feltételekkel: a beviendő dolgozó
  - munkaköre csak hivatalnok (clerk) vagy eladó (salesman) lehet,
  - a főnöke csak Blake, Ford vagy Scott valamelyike lehet,
  - a fizetése 1000 és 2000 USD között legyen, továbbá
  - a 10-es, a 20-as és a 30-as részlegek valamelyikébe kerüljön.
- Készítsen egy olyan szkript programot, amely a felhasználótól bekért adatok alapján az iménti nézettáblán keresztül viszi be az adatokat a dolgozó táblába oly módon, hogy
  - a dolgozó azonosítóját (empno) nem kéri be, hanem automatikusan előállítja az addigi legnagyobb azonosító eggyel megnövelt értékeként, és
  - a belépési dátumot (hiredate) sem kéri be, hanem automatikusan előállítja az aznapi dátumot.
- E szkript program segítségével vigye be legalább egy új dolgozó adatait.
- Listázza a dolgozó táblát.
- Végül állítsa vissza az eredeti táblatartalmakat, és törölje a feladat megoldásakor létrehozott nézettáblákat.

## Mintafeladatsor és megoldása

### M1. feladat

Írjon szkript programot, amely új adattáblát hoz létre az emp és a dept táblákból dolgozó és részleg néven. (E táblák létrehozása előtt győződjön meg arról, hogy ezeken a neveken nem létezik-e már másik objektum, és ha igen, akkor törölje azt.) Lássa el a táblákat az alábbi megszorításokkal:

- Legyen az empno egyedi attribútum a dolgozó táblában.
- Legyen a deptno elsődleges kulcs a részleg táblában, idégen kulcs a dolgozó táblában oly módon, hogy egy részleg törlése esetén törlődjenek e részleg dolgozói is. E megszorítás helyes működésének ellenőrzéseként töröljön egy részleget, listázza ki ennek eredményét, majd vonja vissza a törlést, és ellenőrizze a visszavonás hatását.
- Egy új dolgozó csak a szkript programban megadott értéktartománybeli fizetést kaphasson. Ellenőrzésként kíséreljen meg egy megfelelő és egy nem megfelelő rekordot felvinni. Listázással győződjön meg e műveletek helyes végrehajtásáról.

A fentiek után állítsa vissza az adattáblákat, és ellenőrizze a visszaállítás hatását. A továbbiakban ezeket a táblákat használja.

### Megoldás

#### 1. lépés (A user\_catalog lekérdezése)

```
SELECT * FROM user_catalog;
```

#### Részeredmény

TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
EMP	TABLE
RÉSZLEG	VIEW
SALGRADE	TABLE

#### 2. lépés (A létrehozandó nevű objektum törlése)

```
DROP VIEW részleg;
```

```
SELECT * FROM user_catalog;
```

#### Részeredmény

A nézet eldobva.

TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
EMP	TABLE
SALGRADE	TABLE

#### 3. lépés (A feladatot megoldó szkript program elkészítése)

```
-- Munkatáblák törlése
```

```
DROP TABLE dolgozó;
```

```
DROP TABLE részleg;
```

```
-- Munkatáblák létrehozása
CREATE TABLE dolgozó
    AS SELECT * FROM emp;
CREATE TABLE részleg
    AS SELECT * FROM dept;

COLUMN empno FORMAT 9999
COLUMN ename FORMAT A7
COLUMN mgr    FORMAT 9999
COLUMN sal    FORMAT 9999
COLUMN comm   FORMAT 9999

SELECT * FROM dolgozó;
SELECT * FROM részleg;

-- Egyedi kulcs megszorítás definiálása
ALTER TABLE dolgozó
    ADD CONSTRAINT kulcs1
        UNIQUE (empno);

-- Elsődleges kulcs megszorítás definiálása
ALTER TABLE részleg
    ADD CONSTRAINT kulcs2
        PRIMARY KEY (deptno);

-- Idegen kulcs megszorítás definiálása
ALTER TABLE dolgozó
    ADD CONSTRAINT kulcs3
        FOREIGN KEY (deptno) REFERENCES részleg (deptno)
            ON DELETE CASCADE;
PROMPT Idegen kulcs megszorítás ellenőrzése törléssel:
DELETE FROM részleg
    WHERE deptno = 30;

SELECT * FROM dolgozó;
SELECT * FROM részleg;

PROMPT Visszaállítás:
ROLLBACK;

SELECT * FROM dolgozó;
SELECT * FROM részleg;

-- Fizetési korlátok beállítása
```

```

DEFINE minfiz = 700
DEFINE maxfiz = 7000

-- Check megszorítás definiálása
ALTER TABLE dolgozó
ADD CONSTRAINT fizetés1
    CHECK (sal BETWEEN &minfiz AND &maxfiz);

-- Check megszorítás ellenőrzése adatbevitellel
INSERT INTO dolgozó
VALUES(8888, 'Morgó', 'törpe', 7839, '04-OKT-10', 350, NULL, 30);

-- Check megszorítás ellenőrzése adatbevitellel
INSERT INTO dolgozó
VALUES(8889, 'Tudor', 'törpe', 7839, '04-OKT-10', 3350, NULL, 30);

SELECT * FROM dolgozó;

-- Visszaállítás
ROLLBACK;
-- Végegesítés
-- (Ez a ROLLBACK általi teljes visszaállítás után már felesleges)
-- COMMIT;
SELECT * FROM dolgozó;
CLEAR COLUMNS

```

### Eredmény

A tábla eldobva.  
A tábla eldobva.  
A tábla létrejött.  
A tábla létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20

7369	SMITH	CLERK	7902	80-DEC-17	800	20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000	20
7876	ADAMS	CLERK	7788	83-JAN-12	1100	20
7934	MILLER	CLERK	7782	82-JAN-23	1300	10

14 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 sor kijelölve.

A tábla módosítva.

A tábla módosítva.

A tábla módosítva.

Idegen kulcs megszorítás ellenőrzése törléssel:

1 sor törölve.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

8 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
40	OPERATIONS	BOSTON

3 sor kijelölve.

Visszaállítás:

A visszaállítás befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 sor kijelölve.

régi 3: CHECK (sal BETWEEN &minfiz AND &maxfiz)  
 új 3: CHECK (sal BETWEEN 700 AND 7000)

A tábla módosítva.

INSERT INTO dolgozó

\*

Hiba a(z) 1. sorban:

ORA-02290: ellenőrző megszorítás (SCOTT.FIZETÉS1) megsértése

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
8889	Tudor	törpe	7839	04-OKT-10	3350		30

15 sor kijelölve.

A visszaállítás befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

## M2. feladat

Listázza ki a dolgozók nevét, jövedelmét és telephelyét, továbbá a főnökük nevét, fizetését és telephelyét elsődlegesen a főnök neve szerint csökkenően, másodlagosan a dolgozó neve szerint növekvően rendezve. (Amely dolgozónak nincs főnöke, annál a főnök neveként a „Nincs főnök” szöveget írja ki.)

### 1. megoldás (Külső összekapcsolással)

```

SELECT Ddolgozó.ename          AS "DolgozóNeve",
       Ddolgozó.sal + NVL(Ddolgozó.comm,0) AS "jövedelme",
       Drészleg.loc           AS "telephelye",
       NVL(Fdolgozó.ename, 'Nincs főnök')   AS "FőnökNeve",
       Fdolgozó.sal           AS "fizetése",
       Frészleg.loc           AS "telephelye"

FROM dolgozó  Ddolgozó,
     részleg  Drészleg,
     dolgozó  Fdolgozó,
     részleg  Frészleg

WHERE
    Ddolgozó.deptno = Drészleg.deptno      AND
    Ddolgozó.mgr   = Fdolgozó.empno (+) AND
    Fdolgozó.deptno = Frészleg.deptno (+)

ORDER BY Fdolgozó.ename DESC,
         Ddolgozó.ename ASC;
  
```

### 2. megoldás (Összekapcsolással a JOIN segítségével – SQL92)

```

SELECT Ddolgozó.ename          AS "DolgozóNeve",
       Ddolgozó.sal + NVL(Ddolgozó.comm,0) AS "jövedelme",
       Drészleg.loc           AS "telephelye",
       NVL(Fdolgozó.ename, 'Nincs főnök')   AS "FőnökNeve",
       Fdolgozó.sal           AS "fizetése",
       Frészleg.loc           AS "telephelye"

FROM dolgozó  Ddolgozó
INNER JOIN
     részleg  Drészleg USING(deptno)
LEFT JOIN
     dolgozó  Fdolgozó ON (Ddolgozó.mgr = Fdolgozó.empno)
LEFT JOIN
     részleg  Frészleg ON (Fdolgozó.deptno = Frészleg.deptno)

ORDER BY Fdolgozó.ename DESC,
         Ddolgozó.ename ASC;
  
```

**Eredmény (mindkét megoldás esetén)**

DolgozóNev	jövedelme telephelye	FőnökNeve	fizetése telephelye
KING	5000 NEW YORK	Nincs főnök	
ADAMS	1100 DALLAS	SCOTT	3000 DALLAS
BLAKE	2850 CHICAGO	KING	5000 NEW YORK
CLARK	2450 NEW YORK	KING	5000 NEW YORK
JONES	2975 DALLAS	KING	5000 NEW YORK
FORD	3000 DALLAS	JONES	2975 DALLAS
SCOTT	3000 DALLAS	JONES	2975 DALLAS
SMITH	800 DALLAS	FORD	3000 DALLAS
MILLER	1300 NEW YORK	CLARK	2450 NEW YORK
ALLEN	1900 CHICAGO	BLAKE	2850 CHICAGO
JAMES	950 CHICAGO	BLAKE	2850 CHICAGO
MARTIN	2650 CHICAGO	BLAKE	2850 CHICAGO
TURNER	1500 CHICAGO	BLAKE	2850 CHICAGO
WARD	1750 CHICAGO	BLAKE	2850 CHICAGO

14 sor kijelölve.

**M3. feladat**

Írjon szkript programot, amely listázza az egyes részlegek telephelyét, nevét, az ezekben dolgozók átlagjövedelmét (a kiíratás e három szerint legyen ismétlésmentes), főnökeik nevét, fizetését, valamint telephelyét elsődlegesen a részleg telephelye szerint, másodlagosan pedig a főnökök neve szerint rendezve. A megoldáshoz használjon nézettáblákat, és ezeket a szkript program végén törölje.

**Megoldás****1. lépés (Az allekérdezések megtervezése)**

Szint	A lekérdezés neve	A hivatkozott tábla, nézet, vagy allekérdezés	Az eredmény oszlopai, másodlagos nevei
1.	---	Részleg1, Részleg2, Részleg3	Részleg1.loc AS "Részleg telephelye", Részleg1.dname AS "neve", Részleg2.átlagjöv AS "átlagjövedelem", Részleg3.ename AS "Főnök neve", Részleg3.sal AS "fizetése", Részleg3.loc AS "telephelye"
2.	Részleg1	részleg	loc, deptno, dname
2.	Részleg2	dolgozó	deptno, ROUND(AVG(sa1+NVL(comm, 0))) AS átlagjöv

2.	Részleg3	dolgozó D, dolgozó F, részleg FR	D.deptno, Eename, Esal, FR.loc
----	----------	----------------------------------------	--------------------------------

Készítsük el minden három allekérdezést nézetként. E módszer előnye, hogy az egyszer megírt és letesztelt allekérdezések többször felhasználhatók, ám számunkra ezúttal a legfőbb előnye a könnyű ellenőrzés.

## 2. lépés (Részleg1 elkészítése)

```
CREATE OR REPLACE VIEW Részleg1
```

```
AS
```

```
SELECT deptno, loc, dname  
FROM részleg;
```

```
SELECT * FROM Részleg1;
```

### Részeredmény

A nézet létrejött.

DEPTNO	LOC	DNAME
10	NEW YORK	ACCOUNTING
20	DALLAS	RESEARCH
30	CHICAGO	SALES
40	BOSTON	OPERATIONS



### Megjegyzés

Ez egy nagyon egyszerű nézet, létrehozását elkerülhetők volna az elsődleges (1. szintű) lekérdezésben való közvetlen beírással.

## 3. lépés (Részleg2 elkészítése)

```
CREATE OR REPLACE VIEW Részleg2
```

```
AS
```

```
SELECT deptno,  
       ROUND(AVG(sal+NVL(comm,0))) AS átlagjöv  
FROM dolgozó  
GROUP BY deptno;
```

```
SELECT * FROM Részleg2;
```

### Részeredmény

A nézet létrejött.

DEPTNO	ÁTLAGJÖV
10	2917
20	2175
30	1933

DEPTNO	ÁTLAGJÖV
10	2917
20	2175
30	1933

#### 4. lépés (Részleg3 elkészítése)

```
CREATE OR REPLACE VIEW Részleg3
```

```
AS
```

```
SELECT DISTINCT
    D.deptno,
    F.ename,
    F.sal,
    FR.loc
FROM dolgozó D
    INNER JOIN
        dolgozó F ON (D.mgr = F.empno)
    INNER JOIN
        részleg FR ON (F.deptno = FR.deptno);
```

```
SELECT * FROM Részleg3;
```

#### Részeredmény

A nézet létrejött.

DEPTNO	ENAME	SAL	LOC
10	CLARK	2450	NEW YORK
10	KING	5000	NEW YORK
20	FORD	3000	DALLAS
20	JONES	2975	DALLAS
20	KING	5000	NEW YORK
20	SCOTT	3000	DALLAS
30	BLAKE	2850	CHICAGO
30	KING	5000	NEW YORK

8 sor kijelölve.



#### Megjegyzés

Ezúttal nem volt szükség a főnökkkel nem rendelkező elnök (president) speciális kezelésére (LEFT JOIN), mivel csak a tényleges főnököket kellett listázni.

## 5. lépés (A főlekérdezés elkészítése – a feladat megoldása, a nézettáblák törlése)

```
BREAK ON "RészlegHelye".ON "Neve" ON "ÁtlJövedelem"
COLUMN "Neve"           FORMAT A11
COLUMN "Fizetése"        FORMAT 9999
COLUMN "Telephelye"       FORMAT A10
```

```
SELECT Részleg1.loc      AS "RészlegHelye",
       Részleg1.dname     AS "Neve",
       Részleg2.áttagjöv   AS "ÁtlJövedelem",
       Részleg3.ename     AS "Főnökneve",
       Részleg3.sal       AS "Fizetése",
       Részleg3.loc       AS "Telephelye"
  FROM Részleg1
    INNER JOIN
      Részleg2  USING(deptno)
    INNER JOIN
      Részleg3  USING(deptno)
 ORDER BY Részleg1.loc,
          Részleg3.ename;
```

```
CLEAR BREAKS
CLEAR COLUMNS
DROP VIEW Részleg1;
DROP VIEW Részleg2;
DROP VIEW Részleg3;
```

### Eredmény

RészlegHelye	Neve	ÁtlJövedelem	Főnökneve	Fizetése	Telephelye
CHICAGO	SALES	1933	BLAKE	2850	CHICAGO
			KING	5000	NEW YORK
DALLAS	RESEARCH	2175	FORD	3000	DALLAS
			JONES	2975	DALLAS
			KING	5000	NEW YORK
NEW YORK	ACCOUNTING	2917	SCOTT	3000	DALLAS
			CLARK	2450	NEW YORK
			KING	5000	NEW YORK

8 sor kijelölve.



### Megjegyzés

A feladat teljes megoldását a fenti lépések programrészleteiből álló szkript program adja.

## M4. feladat

Végezzen ellenőrzött, azaz nézettáblán keresztül történő adatfelvitelt a dolgozó táblába.

- Ennek érdekében hozzon létre egy Űrlap nevű nézettáblát, mely megakadályozza a jutalék (comm) mezőnek való értékadást, de minden más adat bevitelét engedélyezi a következő feltételekkel: a beviendő dolgozó
  - munkaköre csak hivatalnok (clerk) vagy eladó (salesman) lehet,
  - a főnöke csak Blake, Ford vagy Scott valamelyike lehet,
  - a fizetése 1000 és 2000 USD között legyen, továbbá
  - a 10-es, a 20-as és a 30-as részlegek valamelyikébe kerüljön.
- Készítsen olyan szkript programot, mely a felhasználótól bekért adatok alapján az iménti nézettáblán keresztül viszi be az adatokat a dolgozó táblába oly módon, hogy
  - a dolgozó azonosítóját (empno) nem kéri be, hanem automatikusan előállítja az addigi legnagyobb azonosító eggyel megnövelt értékeként, és
  - a belépési dátumot (hiredate) sem kéri be, hanem automatikusan előállítja az aznapi dátumot.
- E szkript program segítségével vigye be legalább egy új dolgozó adatait.
- Listázza ki a dolgozó táblát.
- Végül állítsa vissza az eredeti táblatartalmakat, és törölje a feladat megoldásakor létrehozott nézettáblákat.

### Megoldás

#### 1. lépés (Az Űrlap nézettábla létrehozása)

```
CREATE OR REPLACE VIEW Űrlap
  (empno, ename, job, mgr, hiredate, sal, deptno)
AS
  SELECT empno, ename, job, mgr, hiredate, sal, deptno
    FROM dolgozó
   WHERE
     UPPER(job) IN ('CLERK', 'SALESMAN') AND
     mgr IN (SELECT empno
              FROM dolgozó
             WHERE UPPER(ename) IN ('BLAKE', 'FORD', 'SCOTT')) AND
     sal BETWEEN 1000 AND 2000 AND
     deptno IN (10,20,30)
  WITH CHECK OPTION;
```

```
SELECT *
  FROM Űrlap
 ORDER BY ename;
```

#### Részeredmény

A nézet létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7876	ADAMS	CLERK	7788	83-JAN-12	1100	20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	30
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	30

## 2. lépés (Adatbeviteli szkript program létrehozása)

```
SET verify OFF
ACCEPT új_ename    CHAR   PROMPT "A dolgozó neve: "
ACCEPT új_job      CHAR   PROMPT "munkaköre: "
ACCEPT új_Fneve    CHAR   PROMPT "főnöke: "
ACCEPT új_sal      NUMBER PROMPT "fizetése: "
ACCEPT új_deptno   NUMBER PROMPT "részlegazonosítója: "
```

```
INSERT INTO
  Ürlap (empno, ename, job, mgr, hiredate, sal, deptno)
VALUES ((SELECT MAX(empno)+1 FROM Dolgozó),
        UPPER('&új_ename'),
        UPPER('&új_job'),
        (SELECT empno
         FROM dolgozó
         WHERE UPPER(ename) = UPPER('&új_Fneve')),
        sysdate,
        &új_sal,
        &új_deptno);
```

```
SET numwidth 5
SELECT * FROM Dolgozó;
SET numwidth 10
ROLLBACK;
DROP VIEW Ürlap;
SET verify ON
```

### 2.1. eredmény (Hibás adatbevitel)

```
A dolgozó neve: Kuka
munkaköre: clerk
főnöke: King
fizetése: 1200
részlegazonosítója: 10
Ürlap (empno, ename, job, mgr, hiredate, sal, deptno)
*
Ürlap (empno, ename, job, mgr, hiredate, sal, deptno)
```

Hiba a(z) 2. sorban:

ORA-01402: a WITH CHECK OPTION-nal létrehozott nézet WHERE feltétele nem teljesül

## 2.2. eredmény (Sikeres adatbevitel)

A dolgozó neve: Kuka  
 munkaköre: clerk  
 főnöke: Scott  
 fizetése: 1200  
 részlegazonosítója: 10

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	3196		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2796		10
7566	JONES	MANAGER	7839	81-ÁPR-02	3321		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
7935	KUKA	CLERK	7788	04-NOV-27	1200		10

15 sor kijelölve.

A visszaállítás befejeződött.

# Feladatok

## A. feladatsor

### A1. feladat

Az emp és dept táblákból hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbevitellel.

telő kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok e táblákra vonatkoznak, a megoldások során alkalmazzon nézetáblákat is!

#### A2. feladat

Részlegenkénti csoportosításban és ismétlésmentes módon listázza a dolgozók nevét, fizetését, részlegük telephelyét és részlegük összfizetését, ez utóbbi szerint rendezve.

#### A3. feladat

Vegyen fel egy új dolgozót a legkisebb összfizetésű részlegbe olyan fizetéssel, hogy ez a részleg már éppen ne a legkisebb összfizetésű legyen. Listázással ellenőrizze a felvétel helyességét, végül állítsa vissza az eredeti táblatartalmakat.

#### A4. feladat

Listázza ki formázottan (fej- és lábléccel stb.) a felhasználó által megadott telephelyű részleg dolgozónak nevét, fizetését, részlegének telephelyét és annak összfizetését, valamint az abban dolgozó főnökök (mgr) nevét és fizetését, a főnök neve és a dolgozók neve szerint rendezve.

### B. feladatsor

#### B1. feladat

Az emp és dept táblákból hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbeviteli kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok e táblákra vonatkoznak, a megoldások során alkalmazzon nézetáblákat is!

#### B2. feladat

Munkakörönkénti csoportosításban és ismétlésmentes módon listázza ki a dolgozók nevét, munkakörét, telephelyét, fizetését és a munkakörében dolgozók összfizetését, ez utóbbi szerint rendezve.

#### B3. feladat

Vegyen fel egy új dolgozót a legkisebb összfizetésű munkakörbe olyan fizetéssel, hogy az már éppen ne a legkisebb összfizetésű legyen. Listázással ellenőrizze a felvétel helyességét, végül állítsa vissza az eredeti táblatartalmakat.

#### B4. feladat

Listázza ki formázottan (fej- és lábléccel stb.) a felhasználó által megadott munkakörű dolgozók nevét, munkakörét, telephelyét, fizetését, a munkakörében dolgozók összfizetését, továbbá a főnökének (mgr) nevét, fizetését és telephelyét a dolgozók neve szerint rendezve.

## C. feladatsor

### C1. feladat

Az emp és dept táblákból hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbeviteli kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok e táblákra vonatkoznak, a megoldások során alkalmazzon nézettáblákat is!

### C2. feladat

Részlegenkénti csoportosításban és ismétlésmentes módon listázza a dolgozók nevét, fizetését, részlegük telephelyét és ennek összfizetését, ez utóbbi szerint rendezve.

### C3. feladat

Növelje meg minden dolgozó fizetését a legkisebb összfizetésű részlegben annyival, hogy e részleg már éppen ne a legkisebb összfizetésű legyen. Listázással ellenőrizze a növelés helyességét, végül állítsa vissza az eredeti táblatartalmakat.

### C4. feladat

Listázza ki formázottan (fej- és lábleccel stb.) a felhasználó által megadott telephelyű részleg dolgozónak nevét, fizetését, részlegének telephelyét és annak összfizetését, valamint az abban dolgozó főnökök (mgr) nevét és fizetését, a dolgozók neve szerint rendezve.

## D. feladatsor

### D1. feladat

Az emp és dept táblákból hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbeviteli kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok ezekre a táblákra vonatkoznak, a megoldások során alkalmazzon nézettáblákat is!

### D2. feladat

Munkakörönkénti csoportosításban és ismétlésmentes módon listázza ki a főnökök (mgr) nevét, munkakörét, telephelyét, közvetlen beosztottainak összfizetését, ez utóbbi szerint rendezve.

### D3. feladat

Vegyen fel egy új dolgozót a legkisebb összfizetésű közvetlen beosztottakkal rendelkező főnök-höz olyan fizetéssel, hogy az már éppen ne a legkisebb összfizetésű legyen. Listázással ellenőrizze a folyétel helyességét, végül állítsa vissza az eredeti táblatartalmakat.

### D4. feladat

Listázza ki formázottan (fej- és lábleccel stb.) a felhasználó által megadott munkakörű főnökök nevét, munkakörét, telephelyét, valamint közvetlen beosztottainak nevét, fizetését és telephelyét a főnökök neve szerint rendezve.

## E. feladatsor

### E1. feladat

Az emp és dept táblákból hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbeviteli kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok ezekre a táblákra vonatkoznak, a megoldások során alkalmazzon nézettáblákat is!

### E2. feladat

Munkakörönkénti csoportosításban és ismétlésmentes módon listázza a dolgozók nevét, munkakörét, telephelyét, fizetését és a munkakörében dolgozók összfizetését, ez utóbbi szerint rendezve.

### E3. feladat

Növelje meg a legkisebb összfizetésű munkakörben dolgozók mindegyikének a fizetését annyival, hogy e munkakör már éppen ne a legkisebb összfizetésű legyen. Listázással ellenőrizze a növelés helyességét, végül állítsa vissza az eredeti táblatartalmakat.

### E4. feladat

Listázza ki formázottan (fej- és lábléccel stb.) a felhasználó által megadott munkakörű dolgozók nevét, munkakörét, telephelyét, fizetését, a munkakörében dolgozók összfizetését, továbbá a főnökének (mgr) nevét, fizetését és telephelyét a dolgozók neve szerint rendezve.

## F. feladatsor

### F1. feladat

Az emp és dept táblákból hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbeviteli kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok ezekre a táblákra vonatkoznak, a megoldások során alkalmazzon nézettáblákat is!

### F2. feladat

Munkakörönkénti csoportosításban és ismétlésmentes módon listázza a főnökök (mgr) nevét, munkakörét, telephelyét, közvetlen beosztottainak összfizetését ez utóbbi szerint rendezve.

### F3. feladat

Növelje meg a legkisebb összfizetésű közvetlen beosztottakkal rendelkező főnök ezen beosztottai mindegyikének a fizetését annyival, hogy már éppen ne ők legyenek a legkisebb összfizetésű közvetlen beosztottak. Listázással ellenőrizze a növelés helyességét, végül állítsa vissza az eredeti táblatartalmakat.

#### F4. feladat

Listázza ki formázottan (fej- és lábléccel stb.) a felhasználó által megadott munkakörű főnökök nevét, munkakörét, telephelyét, továbbá közvetlen beosztottainak nevét, fizetését és telephelyét, a főnökök neve szerint rendezve.

### G. feladatsor

#### G1. feladat

Az emp és dept táblából hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbeviteli kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok ezekre a táblákra vonatkoznak.

#### G2. feladat

Készítsen egy Intézetek nevű nézettáblát, mely megadja az igazgatók nevét, azonosítóját, valamint a közvetlen és közvetett beosztottai nevét és azonosítóját, ahol igazgatóknak nevezük az elnök (president) közvetlen beosztottait (vagyis az Intézeteket az igazgatóik azonosítják). Listázza e nézettáblát az igazgatók neve szerint rendezve.

#### G3. feladat

Vegyen fel egy új dolgozót a legkisebb összfizetésű Intézetbe olyan fizetéssel, hogy ez az Intézet már éppen ne a legkisebb összfizetésű legyen. Az adatfelvitelt az Intézetek nézettáblán keresztül végezze, melyet ellenőrzésképpen végül listázzon is ki.

#### G4. feladat

Listázza ki formázottan (fej- és lábléccel stb.) és ismétlésmentesen a felhasználó által megadott igazgató beosztottainak nevét, fizetését, részlegének telephelyét és annak összfizetését, valamint az abban dolgozó főnökök (mgr) nevét és fizetését, a dolgozók neve szerint rendezve. Használja az Intézetek nézettáblát.

### H. feladatsor

#### H1. feladat

Az emp és dept táblából hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbeviteli kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok ezekre a táblákra vonatkoznak.

#### H2. feladat

Készítsen egy Intézetek nevű nézettáblát, mely megadja az igazgatók nevét, azonosítóját, valamint a közvetlen és közvetett beosztottai nevét, azonosítóját és fizetését, ahol igazgatóknak ne-

vezzük az elnök (president) közvetlen beosztottait (vagyis az Intézeteket az igazgatóik azonosítják). Listázza ki e nézettáblát az igazgatók neve szerint rendezve.

### H3. feladat

Növelje meg az egynél több beosztott foglalkoztató, legkisebb összfizetésű Intézet beosztott dolgozói mindegyikének fizetését annyival, hogy ez az Intézet már éppen ne a legkisebb összfizetésű legyen. Az adatmódosítást az Intézetek nézettáblán keresztül végezze, melyet ellenőrzésképpen végül listázzon is ki.

### H4. feladat

Listázza ki formázottan (fej- és láblécssel stb.) és ismétlésmentesen a felhasználó által megadott igazgató nevét és fizetését, valamint a beosztottainak nevét, fizetését, részlegük telephelyét és annak összfizetését, valamint az abban dolgozó főnökök (mgr) nevét és fizetését, a dolgozók neve szerint rendezve. Használja az Intézetek nézettáblát.

## I. feladatsor

### I1. feladat

Az emp és dept táblákból hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbeviteli kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok ezekre a táblákra vonatkoznak.

### I2. feladat

Készítsen egy NagyFőnökök nevű nézettáblát, mely megadja az elnök (president) közvetlen beosztottainak adatait, majd listázza a nagyfőnökök neve szerint rendezve.

### I3. feladat

Vegyen fel egy új főnököt, akinek főnöke az elnök (president), fizetése a nagyfőnökök átlagos fizetése, részlege pedig a legnagyobb fizetésű nagyfőnökével egyező, a többi adat tetszőleges. Az adatfelvitelt a NagyFőnökök nézettáblán keresztül végezze, melyet ellenőrzésképpen végül listázzon is ki.

### I4. feladat

Listázza ki formázottan (fej- és láblécssel stb.) és ismétlésmentesen a dolgozók neve szerint rendezve a felhasználó által megadott nevű nagyfőnök nevét, fizetését, közvetlen beosztottainak nevét, fizetését, munkakörét és részlegének nevét. Használja a NagyFőnökök nézettáblát.

## J. feladatsor

### J1. feladat

Az emp és dept táblákból hozzon létre dolgozó és részleg nevű táblákat. A létrehozott táblákat lássa el összesen három megszorítással, ellenőrizze ezeket sikeres és sikertelen adatbeviteli kísérletekkel, majd állítsa vissza az eredeti táblatartalmakat. A további feladatok ezekre a táblákra vonatkoznak.

### J2. feladat

Készítsen egy NagyFőnökök nevű nézettáblát, mely megadja az elnök (president) közvetlen beosztottainak adatait, majd listázza a nagyfőnökök neve szerint rendezve.

### J3. feladat

Növelje meg a nagyfőnökök mindegyikének fizetését a közvetlen beosztottai átlagfizetésének 10%-ával. Az adatmódosítást a NagyFőnökök nézettáblán keresztül végezze, melyet ellenőrzésképpen végül listázzon is ki.

### J4. feladat

Listázza ki formázottan (fej- és lábléccel stb.) és ismétlésmentesen a felhasználó által megadott nevű nagyfőnök nevét, jövedelmét, közvetlen beosztottainak nevét, jövedelmét, munkakörét és részlegének nevét. Használja a NagyFőnökök nézettáblát.

## 8. FEJEZET

# Változóhasználat, vezérlési szerkezetek a PL/SQL-ben

## Elméleti összefoglaló

A következő három fejezet az Oracle magasszintű, professzionális alkalmazások létrehozására szolgáló programozási nyelvét, a PL/SQL-t mutatja be a nyelvi alapoktól az összetett és hivatkozási típusok használatán keresztül a kurzorok, tárolt eljárások és triggerek téma köréig. Különös jelentőséget ad a PL/SQL nyelvnek az a tény, hogy a legújabb szabványbővítmények éppen a PL/SQL-eszközök beépítésével viszik procedurális, illetve objektumorientált irányba a hagyományosan deklaratív SQL nyelvet, vagyis úgy tűnik, hogy a nem is távoli jövő a teljes alkalmazásfejlesztésre képes SQL nyelvé [20].

A PL/SQL az Oracle által kifejlesztett magasszintű programnyelv, a hagyományos SQL nyelv procedurális kiterjesztése. Megtalálhatók benne a korszerű programozási nyelvek jellemzői, így többek között a kivételkezelés és az objektumhasználat. A PL/SQL programok írásával tehát egysíthatók a deklaratív SQL nyelv tömörsegégtől egy procedurális nyelv algoritmusleírásra, teljes alkalmazások kifejlesztésre való képességével.

Az Oracle a PL/SQL programokat külön „motorral”, a PL/SQL fordítóprogrammal dolgozza fel, és ez független az SQL-utasításokat feldolgozó „SQL-motortól”. Ennek hatására növekszik a teljesítmény. A „külön motor” kiszűri az SQL-utasításokat és továbbítja az SQL-motor felé. A PL/SQL a szerveroldalon fut, a fordítóprogram része a PL/SQL programokat egy köztes (pseudo) kódba fordítja, így futtatáskor nem kell lexikális elemzést végezni, és a futtató programja futás előtt még optimalizálja is ezt a kódot. (A PL/SQL nyelv részletesebb ismertetését lásd [15].)

## PL/SQL blokk

Az SQL\*Plus-környezetben a PL/SQL nyelvű programozás alapfogalma a (névtelen) blokk, ahol a blokk leginkább a hagyományos program fogalmának felel meg. Egy PL/SQL blokk sokszor csak egy egyszerű utasítássorozat, amelyet esetleg egy SELECT utasítás vesz körül valamilyen interaktív környezettel annak érdekében, hogy a felhasználó azt az aktuális igényeinek megfelelően tudja paraméterezni. Más esetben azonban egy összetett vezérlési szerkezet, amely egy bonyolult táblaalgoritmust valósít meg. A különböző igényekhez különböző eszközöket biztosít a PL/SQL.

Egy PL/SQL blokkot az SQL\*Plus- (mint gazdanyelvi) környezetben egy SQL\*Plus szkript programba ágyazva futtatunk. A beágyazás módja a következő: a bevezető (általában adatbeérést is tartalmazó) SQL\*Plus- és SQL-utasítások után következik a PL/SQL blokk, melyet a / jel zár le. Ezt további SQL\*Plus- és SQL-utasítások, valamint PL/SQL blokkok követhetnek.

Egy blokk három, funkcionálisan jól elkülöníthető részből, úgynevezett szegmensből állhat: a deklarációs, a végrehajtható és a kivételkezelő részből. A deklarációs rész és a kivételkezelő rész használata opcionális, míg a végrehajtható rész használata kötelező.

## Deklarációs szegmens

A deklarációs szegmenst a DECLARE kulcsszó vezeti be. Ez tartalmazza a blokkban felhasznált összes változót, kurzort és a felhasználó által definiált kivételeket.

## Végrehajtható szegmens

A BEGIN és az EXCEPTION, vagy annak hiányában az END kulcsszavak között találhatók SQL-utasítások, PL/SQL-utasítások, alprogramhívások, vezérlési szerkezetek. A végrehajtható szegmensbe újabb PL/SQL blokkok ágyazhatók be.

Megjegyezzük, hogy az SQL-utasítások egy része csak módosított szintaktikával használható a PL/SQL blokkban.

## Kivételkezelő szegmens

Az EXCEPTION kulcsszó vezeti be. A végrehajtandó részben felmerülő hibák, és abnormális állapotok esetén végrehajtandó utasításokat határozza meg. Helye az END utasítás előtt van. (Részletesen lásd 10. fejezet.)

## A blokkok szerkezete

A blokk felépítésének általános alakja:

```
[DECLARE
  deklarációk]
BEGIN
  végrehajtható utasítások
[EXCEPTION
  kivételkezelés]
END;
/
```



### Megjegyzés

Sose felejtkezzünk el a PL/SQL blokkot lezáró / jel használatáról!

## Változók használata

Az SQL\*Plus-környezetben történő PL/SQL programozás során háromféle változót használhatunk:

- az SQL\*Plus felhasználói (helyettesítő) változóit,
- az SQL\*Plus környezeti (gazdakörnyezeti) változóit,
- a PL/SQL (belső) változóit.

Az alábbiakban röviden áttekintjük ezeket, illetve egymással való kapcsolatukat.

### I. Az SQL\*Plus felhasználói változói

Egy PL/SQL blokk részére az őt futtató SQL\*Plus-környezetből adatot bevinni a (4. fejezetben már részletesen ismertetett) felhasználói (más néven helyettesítő) változókkal lehetséges. (E változók értékének beolvasása a PL/SQL blokkon belül &változó módon törénhet). Értéküket megőrzik a PL/SQL blokkból való visszatérés után is.

### II. Az SQL\*Plus környezeti változói

Egy PL/SQL blokkból adatokat kivinni az őt futtató környezetükbe az úgynevezett környezeti (gazdakörnyezeti vagy hozzárendelt) változókkal lehetséges. E változók az SQL\*Plus-környezetben a

`VARIABLE változónév adattípus`

utasítás segítségével hozhatók létre, ahol az `adattípus` a következők valamelyike lehet:

NUMBER	esetleges negatív előjelet és/vagy tizedespontot tartalmazó szám,
CHAR( <i>n</i> )	kötött hosszúságú karaktersorozat, vagyis a definícióban megadottnál rövidebb karaktersorozatokat balra tömöríti és jobbról szóköz karakterekkel tölti fel,
VARCHAR2( <i>n</i> )	az <i>n</i> maximális mérete 2000, alapértelmezett értéke 1, változó hosszúságú karaktersorozat, vagyis a definícióban megadottnál rövidebb karaktersorozatokat a tényleges hosszúságuk szerint kezeli (például jeleníti meg), az <i>n</i> maximális mérete 4000.

E változók a PL/SQL blokkban kaphatnak értéket értékadó utasítással. (Ott a PL/SQL belső változótól való megkülönböztetés érdekében a nevük előre egy kettőspont (:) karaktert kell tenni.) A környezeti változó az SQL\*Plus-környezetből való kilépésig őrzi meg az értékét (akkor törlődik). Az Oracle által javasolt karakter típus a VARCHAR2(*n*).

Környezeti változó tulajdonságainak lekérdezése:

`VARIABLE [változónév]`

Környezeti változó értékének lekérdezése (csak az az SQL\*Plus-környezetben lehetséges):

```
PRINT [változónév]
```

### III. A PL/SQL változói

A PL/SQL nyelv igen sokféle változó használatát biztosítja. A PL/SQL-változókat (az úgynevezett belső változókat) deklarációs utasítással a PL/SQL blokkban hozzuk létre, és az abból való viszszalépéskor automatikusan megsemmisülnek. (A változók létrehozására, hatáskörére és az alkalmazható adattípusokra vonatkozó részletes ismertetést lásd [15], [16] és [26].)

A belső változók deklarációja a deklarációs szegmensben történik a DECLARE kulcsszót követően:

```
DECLARE
    változónév adattípus;
    [változónév adattípus;]
    ...

```

ahol az egyes változók deklációjában (az elemi deklarációkban) szereplő adattípusok lehetnek egyszerű és összetett típusok.

A PL/SQL-ben használható egyszerű típusok egyrészt az 5. fejezetben a Tábla létrehozása c. pontban bemutatott NUMBER[(*m* [, *t*])], CHAR[*(n*)], VARCHAR2(*n*) és DATE típusok, másrészről a további szám, illetve karakteres típusok (például INTEGER, REAL, FLOAT, BINARY\_INTEGER, illetve STRING, LONG stb.), a nagy méretű (például multimédiás) adatok tárolására szolgáló típusok (LOB, BLOB, CLOB) és a logikai BOOLEAN típus (mely a szokásos TRUE és FALSE értéken kívül felveheti a NULL értéket is).

A PL/SQL összetett adattípusaival (hivatkozási típusok, rekord, gyűjtőtábla, kurzor stb.) a következő fejezetben foglalkozunk.

### A változóhasználat szemléltetése

#### SQL\*Plus-szkript

- I. Értékadással létrehozzuk  
és lekérdezhetjük
- II. Bevezetjük

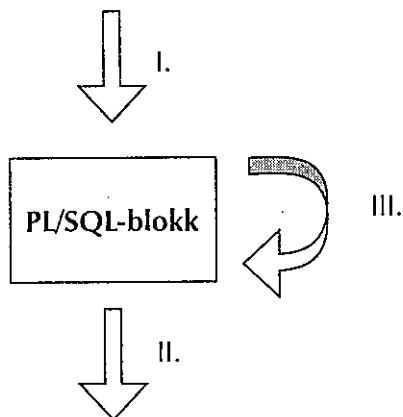
#### PL/SQL blokk

- I. Lekérdezhetjük
- II. Értéket kaphat, értéket adhat
- III. Bevezetjük, értéket kaphat, értéket adhat

#### SQL\*Plus szkript (folytatás)

- I. Lekérdezhető
- II. Lekérdezhető
- III. Nem elérhető

Vázlatosan:



### 8.1. példa

Írjon PL/SQL blokkot tartalmazó SQL\*Plus szkript programot, mely a felhasználótól bekér egy egész számot. Ha ez a szám nagyobb 100-nál, akkor a PL/SQL blokkban, egyébként pedig a PL/SQL blokkot követő gazdanyelvi (SQL\*Plus-) környezetben írassa ki.

#### Megoldás

```

-- A DBMS_OUTPUT.PUT_LINE eljárással történő
-- PL/SQL kiíratás engedélyezése:
SET serveroutput ON
-- Felhasználó által megadott változó definiálása
-- és értékének bekérése:
ACCEPT valtozo_I PROMPT "Kérem adjon meg egy egész számot: "
-- Hozzárendelt (gazdakörnyezeti) változó deklarálása:
VARIABLE valtozo_II NUMBER
-- A PL/SQL-blokk deklarációs szegmense:
-- Változó deklarálása:
DECLARE
    valtozo_III NUMBER;
-- A PL/SQL-blokk végrehajtási szegmense:
BEGIN
-- Értékkadás:
    valtozo_III := &valtozo_I;
-- Feltételes utasítás:
    IF valtozo_III > 100
    THEN
        -- PL/SQL futás közbeni kiíratás
        DBMS_OUTPUT.PUT_LINE('A megadott szám: '|| valtozo_III);
    ELSE
        -- Hozzárendelt változó felveszi a PL/SQL-változó értékét
        :valtozo_II := valtozo_III;
    END IF;
  
```

```
-- A PL/SQL-blokk vége
```

```
END;
```

```
-- A PL/SQL-blokk futtatása:
```

```
/
```

PROMPT --> A hozzárendelt változó kiíratása a gazdakörnyezetben:

```
PRINT valtozo_II
```

PROMPT --> A hozzárendelt változók lekérdezése a gazdakörnyezetben:

```
-- (Ezeket nem kell törölni,
```

```
-- mivel az SQL*Plus-ból való kilépéskor törlődnek)
```

```
VARIABLE
```

PROMPT --> A gazdakörnyezeti és helyettesítő változók lekérdezése:

```
-- (Ezek is törlődnek az SQL*Plus-ból való kilépéskor,
```

```
-- de ezek törölhetők is UNDEFINE utasítással)
```

```
DEFINE
```

PROMPT --> A helyettesítő változók törlése:

```
UNDEFINE valtozo_I
```

PROMPT --> A gazdakörnyezeti és helyettesítő változók lekérdezése:

```
DEFINE
```

## 1. futtatás után

Kérem adjon meg egy egész számot: 555

```
régi 6: valtozo_III := &valtozo_I;
```

```
új 6: valtozo_III := 555;
```

```
A megadott szám: 555
```

A PL/SQL eljárás sikeresen befejeződött.

--> A hozzárendelt változó kiíratása a gazdakörnyezetben:

```
VALTOZO_II
```

```
-----
```

--> A hozzárendelt változók lekérdezése a gazdakörnyezetben:

```
változó valtozo_ii
```

```
adattípus NUMBER
```

--> A gazdakörnyezeti és helyettesítő változók lekérdezése:

```
DEFINE _CONNECT_IDENTIFIER = "myora" (CHAR)
```

```
DEFINE _SQLPLUS_RELEASE = "902000100" (CHAR)
```

```

DEFINE _EDITOR          = "Notepad" (CHAR)
DEFINE _O_VERSION       = "Personal Oracle9i Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production" (CHAR)
DEFINE _O_RELEASE        = "902000100" (CHAR)
DEFINE _RC               = "1" (CHAR)
DEFINE VALTOZO_I         = "555" (CHAR)

--> A helyettesítő változók törlése:
--> A gazdakörnyezeti és helyettesítő változók lekérdezése:
DEFINE _CONNECT_IDENTIFIER = "myora" (CHAR)
DEFINE _SQLPLUS_RELEASE   = "902000100" (CHAR)
DEFINE _EDITOR            = "Notepad" (CHAR)
DEFINE _O_VERSION          = "Personal Oracle9i Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production" (CHAR)
DEFINE _O_RELEASE          = "902000100" (CHAR)
DEFINE _RC                 = "1" (CHAR)

```

## 2. futtatás után

Kérem adjon meg egy egész számot: 33  
régi 6: valtozo\_III := &valtozo\_I;  
Új 6: valtozo\_III := 33;

A PL/SQL eljárás sikeresen befejeződött.

--> A hozzárendelt változó kiíratása a gazdakörnyezetben:

VALTOZO\_II

-----

33

--> A hozzárendelt változók lekérdezése a gazdakörnyezetben:

változó valtozo\_ii

adattípus NUMBER

--> A gazdakörnyezeti és helyettesítő változók lekérdezése:

DEFINE \_CONNECT\_IDENTIFIER = "myora" (CHAR)

DEFINE \_SQLPLUS\_RELEASE = "902000100" (CHAR)

DEFINE \_EDITOR = "Notepad" (CHAR)

DEFINE \_O\_VERSION = "Personal Oracle9i Release 9.2.0.1.0 - Production

With the Partitioning, OLAP and Oracle Data Mining options

JServer Release 9.2.0.1.0 - Production" (CHAR)

DEFINE \_O\_RELEASE = "902000100" (CHAR)

DEFINE \_RC = "1" (CHAR)

DEFINE VALTOZO\_I = "33" (CHAR)

```
--> A helyettesítő változók törlése:  

--> A gazdakörnyezeti és helyettesítő változók lekérdezése:  

DEFINE _CONNECT_IDENTIFIER = "myora" (CHAR)  

DEFINE _SQLPLUS_RELEASE = "902000100" (CHAR)  

DEFINE _EDITOR = "Notepad" (CHAR)  

DEFINE _O_VERSION = "Personal Oracle9i Release 9.2.0.1.0 - Production  

With the Partitioning, OLAP and Oracle Data Mining options  

JServer Release 9.2.0.1.0 - Production" (CHAR)  

DEFINE _O_RELEASE = "902000100" (CHAR)  

DEFINE _RC = "1" (CHAR)
```

## Vezérlési szerkezetek, utasítások

### A SELECT utasítás a PL/SQL-ben

Egy adattábla egyes sorait PL/SQL-ben a SELECT utasítás alábbi alakjával kérdezhetjük le (v. ö. az 1. fejezetben bemutatott alakkal):

```
SELECT SzelekciósLista  

  INTO {ElemiVáltozó [, ElemiVáltozó]... | RekordVáltozó}  

  FROM TáblaLista  

  [WHERE LogikaiOszlopkifejezés]  

  [GROUP BY CsoportosítóOszlopkifejezés-lista]  

  [HAVING LogikaiOszlopkifejezés]  

  [ORDER BY RendezőOszlopkifejezés-lista];
```

ahol az elemi és a rekordváltozók a PL/SQL-ben deklarált változók. A PL/SQL-ben a SELECT utasítás WHERE logikai kifejezését úgy kell megadni, hogy a lekérdezés egyszerre csak egy sort adhasson vissza. A szelekciós listában csak azok az oszlopok, és azok is csak olyan sorrendben szerepelhetnek, melyekhez rendelt elemi változók az INTO kulcsszó után állnak, minthogy sorrend szerinti összerendeléssel veszik fel az oszlopértékeket. (Értelemszerűen hasonló követelmény áll fenn a rekord típusú változó adatmezőivel kapcsolatban is.)

### Feltételes utasítás

Az utasítás általános alakja:

```
IF feltétel THEN  

  utasítások  

[ELSIF feltétel THEN  

  utasítások]  

...  

[ELSE  

  utasítások]  

END IF;
```

ahol:

- **feltétel** logikai változó vagy kifejezés, TRUE, FALSE és NULL értéke lehet,
- **utasítások** egy vagy több PL/SQL- vagy SQL-utasítás. Ezek is tartalmazhatnak további feltételes utasításokat,
- **THEN** a **feltétel** teljesülése esetén a végrehajtás ága,
- **ELSIF** ha az első feltétel hamis, akkor a vezérlés ezekre az ágakra adódik, ahol további feltételek adhatók,
- **ELSE** ha egyetlen feltétel sem teljesül, akkor a vezérlés erre az ágra adódik,
- **END IF** a feltételes utasítás lezárása.

## Ciklusutasítások

### A LOOP-CIKLUS

A LOOP-ciklus (vagy más néven egyszerű ciklus) a ciklusmag (a LOOP és az END LOOP közötti rész) utasításait feltétel nélkül hajtja végre. Kilépés az EXIT vagy az EXIT WHEN utasításokkal történhet. (E kilépő utasításrészek a ciklusmag tetszőleges helyén állhatnak, akár egy feltételes utasítás valamelyik ágán is.)

```

LOOP
  utasítás ;
  [utasítás ;]

  ...
  [EXIT ;]
  [EXIT WHEN feltétel ;]
END LOOP;

```

### A FOR-CIKLUS

A FOR-ciklus ciklusmagja egy LOOP-ciklust tartalmaz, és ez a ciklusmag meghatározott számú feldolgozására használatos. A ciklusváltozója (melynek hatóköre maga a ciklus, azon kívül nincs is értelmezve) sorra felveszi az adathalmazának értékeit, miközben minden ciklusváltozó értékhez végrehajtja a ciklusmag utasításait. A FOR-ciklus rendezett adathalmazát alapértelmezésben a legkisebb és legnagyobb értékek megadásával jelöljük ki. E halmaznak ekkor egyesével növekvő egész számokból kell állnia. (A következő fejezetben látni fogjuk, hogy a FOR-ciklus általános esetben tetszőleges rendezett halmazon is működhet.) Az alapértelmezett FOR-ciklusban a rendezett adathalmaz elemeinek bejárása egyesével növekvő, illetve a REVERSE kulcsszó használata esetén egyesével csökkenő.

```

FOR ciklusváltozó IN [REVERSE] alsóhatár .. felsőhatár
  LOOP-ciklus

```

A ciklusváltozó deklarációja automatikus (nem deklarálható külön), és csak a cikluson belül van értéke. Az index név nem lehet ciklusváltozó, mert ez az Oracle-ben kulcsszó. Megjegyezzük,

hogy a későbbiekben a FOR-ciklusnak bemutatjuk azt az alakját, ahol az IN kulcsszó után egy rendezett halmaz (explicit vagy rejtett kurzor) is állhat (lásd 9. fejezet).

### A WHILE-CIKLUS

A WHILE-ciklus egy előtesztelő ciklus. A ciklusmagjában lévő LOOP-ciklust addig hajtja végre, amíg a ciklusfejben lévő feltétel teljesül.

*WHILE feltétel*

LOOP-ciklus



#### Megjegyzés

Az EXIT utasítás használata szempontjából az egyszerű LOOP-ciklus és a FOR-, illetve WHILE-ciklusok között az a lényeges különbség, hogy míg a LOOP-ciklusból csak ennek segítségével lehet kijutni (különben végtelen ciklussá válna), addig a másik két ciklus esetén ez csak kiegészítő lehetőség.

## Adatok kiírása képernyőre

Az úgynevezett DBMS-csomagok (melyek az objektumorientált programozás objektumaihoz hasonlítanak) a PL/SQL-alkalmazások fejlesztését segítik. A specifikációs részük egy interfész biztosít, amelyen keresztül hozzáférhetünk a csomag eszközeihez, a törzsrészük pedig tartalmazza a futtatható kódot (részletesen lásd [15]).

Az Oracle beépített csomagai közé tartozik a DBMS\_OUTPUT csomag, mely adatok képernyőre való kiíratását végzi. Egy PL/SQL programon belül egy (konstansokból, változókból, függvényekből és műveletekből álló) kifejezés képernyőre történő kiírása e csomag PUT\_LINE eljárásával (módotlanul) lehetséges a

```
DBMS_OUTPUT.PUT_LINE[(kifejezés)];
```

utasítás segítségével, ahol a kifejezés egy karaktersorozat típusú kifejezés, mely tetszőleges SQL karakteres függvényt tartalmazhat. (Ha a kifejezés hiányzik, akkor a képernyőn egy sora melles történik.) Ez az eljárás automatikus (karaktersorozattá történő) konverziót végez az adatokon a kiíratás érdekében. Több elem kiíratása esetén a szokásos || jelet, vagy a CONCAT függvényt használhatjuk. Ciklusból (jellemzően listászerűen, egymás alá) történő kiíratáskor az oszlopszerű formátum érdekében célszerű az RPAD és az LPAD függvényeket alkalmazni.

A DBMS\_OUTPUT csomag képernyőre való kiíratását az SQL\*Plus-ban engedélyeztetni kell, mely a

```
SET serveroutput ON
```

utasítással tehető meg.

## 8.2. példa

Kérjen be két egész számot és döntse el, hogy az összegük páros-e vagy páratlan. Az eredményt futás közben (a PL/SQL blokkban) írja ki.

### 1. megoldás

```
SET serveroutput ON
DECLARE
    C NUMBER;
BEGIN
    C := &A +&B;
    DBMS_OUTPUT.PUT_LINE(C);
    IF MOD(C,2) = 1
    THEN
        DBMS_OUTPUT.PUT_LINE(C || ' PÁRATLAN');
    ELSE
        DBMS_OUTPUT.PUT_LINE(C || ' PÁROS');
    END IF;
END;
/
```

### Eredmény

```
Adja meg a(z) a értékét: 34
Adja meg a(z) b értékét: 23
régi 4:   C := &A +&B;
új  4:   C := 34 +23;
57
57  PÁRATLAN
```

A PL/SQL eljárás sikeresen befejeződött.

### 2. megoldás

```
SET serveroutput ON
ACCEPT A NUMBER PROMPT'Egyik: '
ACCEPT B NUMBER PROMPT 'Másik: '

DECLARE
    C NUMBER;
    CA NUMBER;
    CB NUMBER;
BEGIN
    CA := &A;
    CB := &B;
    C := CA + CB;
```

```

IF MOD(C,2) = 1
THEN
    DBMS_OUTPUT.PUT_LINE(CA||'+'||CB||'='||C||', PÁRATLAN');
ELSE
    DBMS_OUTPUT.PUT_LINE(CA||'+'||CB||'='||C||', PÁROS');
END IF;
END;
/

```

**Eredmény**

Egyik: 222

Másik: 444

régi 6: CA := &A;

új 6: CA := 222;

régi 7: CB := &B;

új 7: CB := 444;

222+444=666, PÁROS

A PL/SQL eljárás sikeresen befejeződött.

**3. megoldás**

SET serveroutput ON

```

VARIABLE C          NUMBER
VARIABLE PARITAS  VARCHAR2(10)

```

ACCEPT A NUMBER PROMPT 'Egyik: '

ACCEPT B PROMPT 'Másik: '

BEGIN

:C := &A + TO\_NUMBER(&B);

IF MOD(:C,2) = 1

THEN

:PARITAS := 'PÁRATLAN';

ELSE

:PARITAS := 'PÁROS';

END IF;

DBMS\_OUTPUT.PUT\_LINE('Az összeg: '||:C||', '|| :PARITAS);

END;

/

**Eredmény**

Egyik: 555

Másik: 111

régi 2: :C := &A + TO\_NUMBER(&B);

új 2: :C := 555 + TO\_NUMBER(111);

Az összeg: 666. PÁROS

A PL/SQL eljárás sikeresen befejeződött.

**8.3. példa**

Írjon PL/SQL programot, amely kiírja az emp tábla sorainak számát és az átlagfizetést.

**Megoldás**

```
SET serveroutput ON
VARIABLE sorok_száma NUMBER
VARIABLE átlag NUMBER

DECLARE
    v_létszám NUMBER;
    v_átlag NUMBER(7,2);
BEGIN
    SELECT COUNT(*), ROUND(AVG(sal))
        INTO v_létszám, v_átlag
        FROM emp;
    DBMS_OUTPUT.PUT_LINE ('létszám: ||v_létszám||'
                           ' átlagfizetés: ||v_átlag||');
    :sorok_száma := v_létszám;
    :átlag := v_átlag;
END;
/
PRINT sorok_száma
PRINT átlag
```

**Eredmény**

létszám: 14 átlagfizetés: 2073

A PL/SQL eljárás sikeresen befejeződött.

SOROK\_SZÁMA

-----

ÁTLAG

---

2073

## Véletlen számok generálása

Az Oracle beépített csomagjai közé tartozik a DBMS\_RANDOM csomag is, mely pszeudovéletlen számokat állít elő az Oracle véletlenszám-generátorából.

A véletlenszám-generátor függvény alakja

DBMS\_RANDOM.VALUE(*AlsóHatár*, *FelsőHatár*)

ahol az *AlsóHatár* és a *FelsőHatár* egyaránt Binary\_Integer típusú számok.

### 8.4. példa

Állítsunk elő egy 0 és 10 közötti véletlen számot.

#### Megoldás

```
SELECT DBMS_RANDOM.VALUE(0,10)
      FROM dual;
```

#### Eredmény

```
DBMS_RANDOM.VALUE(0,10)
-----
2.06535198
```

### 8.5. példa

Állítsunk elő egy 50 és 150 közötti véletlen számot.

#### Megoldás

```
SELECT DBMS_RANDOM.VALUE(50,150)
      FROM dual;
```

#### 1. futási eredmény

```
DBMS_RANDOM.VALUE(50,150)
-----
107.463214
```

#### 2. futási eredmény

```
DBMS_RANDOM.VALUE(50,150)
-----
138.737076
```

## 8.6. példa

Állítsunk elő egy -50 és +50 közötti véletlen számot.

### Megoldás

```
SELECT DBMS_RANDOM.VALUE(-50,+50)
  FROM dual;
```

### Eredmény

```
DBMS_RANDOM.VALUE(-50,+50)
```

```
-----  
-23.502705
```

## 8.7. példa

Karácsonyi nyereményjátékot tartanak az emp tábla vállalatánál. minden dolgozó 0 és 100 USD közötti jutalmat nyerhet, melyet mindenki hozzá is adnak a jutalékához. Készítse el a játék programját.

### 1. megoldás

```
SELECT azonosító, név, fizetés, jutalék, nyeremény,
       NVL(jutalék,0)+nyeremény AS "ÚjJutalék"
  FROM (SELECT empno AS azonosító,
               ename AS név,
               sal   AS fizetés,
               comm  AS jutalék,
               ROUND(DBMS_RANDOM.VALUE(0,100)) AS nyeremény
      FROM emp);
```

### 2. megoldás

```
SELECT empno AS azonosító,
       ename AS név,
       sal   AS fizetés,
       comm  AS jutalék,
       nyeremény,
       NVL(comm,0)+ nyeremény AS "ÚjJutalék"
  FROM emp,
       (SELECT ROUND(DBMS_RANDOM.VALUE(0,100)) AS nyeremény
      FROM dual);
```

### Eredmény (mindkét esetben hasonló)

AZONOSÍTÓ NÉV	FIZETÉS	JUTALÉK	NYEREMÉNY	ÚjJutalék
7839 KING	5000		4	4

7698	BLAKE	2850		78	78
7782	CLARK	2450		87	87
7566	JONES	2975		25	25
7654	MARTIN	1250	1400	48	1448
7499	ALLEN	1600	300	54	354
7844	TURNER	1500	0	57	57
7900	JAMES	950		8	8
7521	WARD	1250	500	79	579
7902	FORD	3000		27	27
7369	SMITH	800		77	77
7788	SCOTT	3000		6	6
7876	ADAMS	1100		2	2
7934	MILLER	1300		38	38

14 sor kijelölve.

## Feladatok

### 8.1. feladat

Határozza meg egy PL/SQL program segítségével a felhasználó által megadott telephelyen dolgozók bérösszegét.

### 8.2. feladat

Írjon PL/SQL programot, amely meghatározza, hogy a felhasználó által megadott nevű és korú személy hány év múlva lesz, illetve hány éve már nagykorú. A kiírást valósítja meg a PL/SQL blokkon belül és az SQL\*Plus-környezetben is.

### 8.3. feladat

Állítsa elő a felhasználó által megadott darabszámig a Fibonacci-sorozat elemeit (0, 1, 1, 2, 3, 5, ...). A megoldáshoz LOOP-ciklust használjon.

### 8.4. feladat

Írjon PL/SQL programot, amely a felhasználó által megadott telephelyen kiszámítja a legnagyobb fizetési különbséget. A feladatot oldja meg SQL és PL/SQL nyelven.

### 8.5. feladat

Írjon PL/SQL programot, amely bekér két egész számot, és kiírja a legnagyobb közös osztókat. (Például 8 és 12 esetén: 4.) Használjon WHILE-ciklust.

**8.6. feladat**

Írjon SQL\*Plus szkript programot, mely egy PL/SQL blokkban kiszámítja a felhasználó által megadott A számtól a szintén felhasználó által megadott B számig a páratlan számok négyzetösszegét, és ezt az SQL\*Plus gazdakörnyezetben írja ki. Használjon FOR-ciklust.

**8.7. feladat**

Írjon szkript programot, amely előállítja, és egymást követő sorokba kiírja a felhasználó által megadott két egész szám összes közös osztóját. Például 18 és 24 esetén: 1, 2, 3, 6.

**8.8. feladat**

Írjon szkript programot, mely előállítja és kiírja a felhasználó által megadott két egész szám közül legalább az egyikkel osztható első 11 különböző számot. Például 3 és 4 esetén: 3, 4, 6, 8, 9, 12, 15, 16, 18, 20, 21.

**8.9. feladat**

Írjon szkript programot, mely előállítja és kiírja a felhasználó által megadott két egész szám legkisebb közös többszörösét. Például 8 és 12 esetén: 24.

## 9. FEJEZET

# Hivatkozási és összetett adattípusok, kurzor, ROWID

## Elméleti összefoglaló

A hivatkozási (más néven relatív, vagy kapcsolt) adattípusok valamely korábban már deklarált változóra, vagy valamely adattábla valamely oszlopára, vagy valamely adattáblára (ezáltal annak összes oszlopára) hivatkoznak. A hivatkozott objektum jellegétől függően a hivatkozási adattípus lehet egyszerű vagy rekord típusú. (Például adattáblára való hivatkozás esetén nyilvánvalóan rekord típust kapunk.)

A hivatkozási adattípusok révén olyan feldolgozóprogramok írhatók, melyek az adattáblák szerkezetének kisebb változásaival szemben érzéketlenek, és amelyekben nagyobb változások (például numerikusról karakteresre való áttérés) esetén is jól kézbentarthatók a szükséges módosítások.

A PL/SQL (jelen példatárban használt) összetett adattípusai a rekord, a gyűjtőtábla (PL/SQL-tábla) és a kurzor. Az összetett típusok (bár lehetnek közvetlen típusmegadásúak is) általában hivatkozási típusok. Használatukkal igen tömör és hatékony programok írhatók. A kurzort (mely tulajdonképpen egy lekérdezés) felhasználhatjuk (egyéb utasítások használata mellett) lekérdezések egymást követő sorainak egyenkénti feldolgozására, sőt egy FOR-ciklus adathalmazaként is. Ennek a feladatkörnek talán legösszetettebb esetei az úgynvezett párosítási feladatok (lásd jelen fejezet: Összetett feladatok). Ezzel a téma körrel részletesen a [15] és [16] foglalkozik.

## Hivatkozási adattípus

A hivatkozási típusú változók deklarálásának módja:

```
DECLARE
    változó1    táblanév.oszlopnév%TYPE;
    változó2    változó1%TYPE;
    változó3    táblanév%ROWTYPE;
```

ahol az elemi típusú változó1 típusa azonos a táblanév nevű tábla oszlopnév nevű oszlopának típusával, a változó2 típusa azonos a változó1 típusával, és a rekord típusú változó3 egyes me-

zőire minősített nevekkel lehet hivatkozni. (Ha például a táblanév nevű táblában volt egy kor nevű oszlop, akkor a változó3 nevű változónak lesz egy kor nevű mezője, melyre változó3.kor néven hivatkozhatunk.)

## Rekord adattípus

Rekord adattípus és ilyen típusú változó deklarálása:

```
DECLARE
    TYPE rekordtípusnév IS RECORD
        (oszlopnév1     típus1
         [, oszlopnév2     típus2]... );
    változónév {rekordtípusnév | tábla%ROWTYPE};
```

ahol:

- oszlopnév a rekord egy mezőjének neve,
- típusn {típusnév | változó%TYPE | tábla.oszlop%TYPE}.

## Gyűjtőtábla típus

A gyűjtőtábla (vagy más néven PL/SQL-tábla) típusú változók adat- vagy nézettáblák lekérdezett sorainak a memoriában való listaszerű tárolására alkalmasak.

```
DECLARE
    TYPE típusnév IS TABLE OF
        {oszloptípus | változó%TYPE | tábla.oszlop%TYPE | tábla%ROWTYPE}
        [NOT NULL]
        INDEX BY BINARY_INTEGER;
    változónév típusnév;
```

ahol:

- típusnév a gyűjtőtábla típus neve,
- oszloptípus standard PL/SQL-adattípus, de lehet rekord is,
- változó%TYPE hivatkozás korábban deklarált változóra,
- tábla.oszlop%TYPE hivatkozás egy tábla oszlopának típusára,
- NOT NULL hivatkozás egy táblára,
- INDEX BY BINARY\_INTEGER megszorítás megadása,
- az utasításrész egyetlen funkciója a felhasználó emlékeztetése arra, hogy az INDEX egész típusú.

## Kurzorok

Az eddigi eszközeinkkel nem volt lehetőségünk arra, hogy egy adattábla (nézettábla) megadott feltételeknek eleget tevő sorait egyenként feldolgozzuk. Az SQL UPDATE utasítása segítségével ugyanis a benne szereplő WHERE feltételnek megfelelő sorokon csak ugyanazt a módosítást lehet végrehajtani. Gyakran azonban arra van szükség, hogy a megadott feltételnek eleget tevő sorok feldolgozása különböző lehessen, mivel az egymást követő elemi feldolgozások között kölcsönhatás van (lásd például a párosítási feladatokat a jelen fejezet összetett feladatai között).

A PL/SQL már megismert SELECT utasítása ugyan már csak egyetlen sort ad eredményként (SELECT ... INTO ...), ám erről nekünk kellett gondoskodni kiválasztó feltételek (jellemzően a WHERE feltétel) megfelelő beállításával. Azt a célt tehát, hogy ugyanazt a feltételt kielégítő sorokat egyenként dolgozzuk fel, ez sem elégít ki.

Az PL/SQL azonban rendelkezik egy igen sokoldalúan használható eszközzel is, a kurzorra, melynek segítségével lehetővé válik adattáblák (nézettáblák) soronkénti feldolgozása. Ennek során az Oracle egy munkaterületet hoz létre, egy memóriaterületet foglal le, ahol a feldolgozandó (a kurzorhoz tartozó SELECT utasítással lekérdezett) adatokat, vagyis e SELECT eredménytábláját átmenetileg tárolja. (Ez a SELECT utasítás azonban már egy hagyományos SQL SELECT, amely természetesen általában több sort ad vissza.) Ehhez a lefoglalt munkaterülethez (a kurzorterülethez, vagy más néven az aktív halmazhoz) azonosítót rendelhetünk (ez a cursor neve), és a későbbiek során e névvel hivatkozhatunk rá. (Ezt nevezzük *explicit* kurzornak). A cursor neve tehát egy szimbolikus mutató, amely ennek az átmeneti tárolóhelynek (eredménytáblának) a kezdetére mutat. Ha ezt a mutatót (egy ciklus segítségével) végigmozgatjuk az átmeneti eredménytábla minden egyes során, akkor e sorokat egyenként feldolgozhatjuk. Nagyon fontos, hogy egy kurzorterület megnyitása után a kurzorterületen lévő adatok nem frissülnek (egészen a következő megnyitásig), és azok csak feldolgozásra használhatók, azon keresztül az adattábla adatai nem változtathatók meg. Az Oracle-ben kétféle cursor létezik:

- az *implicit* és
- az *explicit* cursor.

A PL/SQL nyelv minden, PL/SQL blokkban kiadott DML (INSERT, DELETE, UPDATE) és explicit kurzorral nem rendelkező DQL- (SELECT) utasításhoz létrehoz egy úgynevezett *implicit cursor*. Az ekkor keletkező átmeneti eredménytáblának (kurzorterületnek) nincs neve, azt az Oracle-rendszer automatikusan kezeli. A létrehozott implicit cursorra SQL előnevű függvényekkel (úgynevezett cursorattribútumokkal, lásd alább) hivatkozhatunk.

A programozó saját maga is deklarálhat cursor-t, ezt *explicit cursor*nak nevezzük, és az OPEN megnyitó, a FETCH léptető, és a CLOSE lezáró utasításokkal vezérelhetjük (lásd alább).

## Explicit cursor

A PL/SQL lehetővé teszi adattáblák soronkénti feldolgozását az úgynevezett cursor adattípus segítségével. A cursor használatakor (mint feljebb már említettük) az Oracle egy munkaterületet hoz létre, ahol a feldolgozandó (a kurzorhoz tartozó SELECT utasítással lekérdezett) adatokat átmenetileg tárolja. Az explicit cursor deklarálásának módja:

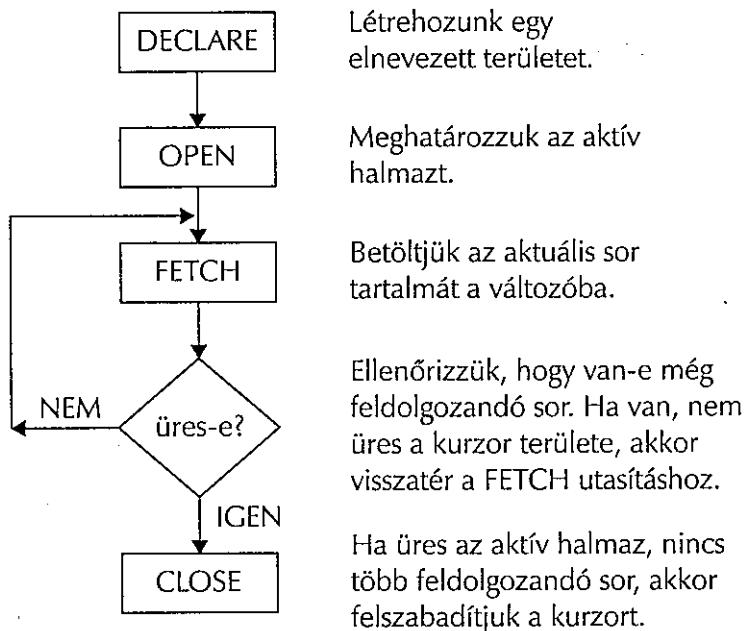
```

DECLARE
CURSOR kurzornév IS
SELECT-utasítás;

```

ahol a *SELECT*-utasítás SQL szintaktikájú (így természetesen nem tartalmaz *INTO* utasításrészet!).

Explicit cursor használata esetén először meg kell nyitni a kurzort, ezután történik a cursor sorainak feldolgozása egyenként, majd a cursorterületet be kell zárni. Ennek vázlatát mutatja be az alábbi ábra:



## Kurzorhasználat FOR-ciklusban (rejtett kurzorok)

A kurzorok használatát jelentősen leegyszerűsítik a kurzort használó FOR-ciklusok. Lehetőség van ugyanis arra (miként ezt korábban már jeleztük), hogy a FOR-ciklus rendezett adathalmazaként egy kurzort adjunk meg. Ekkor a ciklusváltozó típusa (hivatkozási típusként) automatikusan felveszi a cursor által meghatározott rekordszerkezetet, és értékeként minden ciklusban felveszi a cursor egymást követő sorait. Ebben az esetben a cursor megnyitása, sorainak betölése, egyesével való léptetése, és az összes sor feldolgozása utáni bezárása *implicit* módon, tehát felhasználói *OPEN*, *FETCH* és *CLOSE* utasítás kiadása nélkül történik. Ezt a feldolgozási módszert az *explicit cursor rejtett használatának* is nevezzük.

Tovább is egyszerűsíthetjük azonban a programot azzal, hogy egyáltalán nem deklarálunk kurzort, hanem a cursorbeli lekérdezést közvetlenül adjuk meg a FOR-ciklus rendezett adathalmazaként. Ekkor *rejtett cursorról* beszélhetünk. (A rejtett cursor tulajdonképpen egy speciális *implicit cursor*.)

A kurzorhasználat minden két fenti esetben:

```

FOR ciklusváltozó IN kurzor
LOOP
  utasítás;
  [utasítás;]
  ...
END LOOP;

```

Ekkor a *kurzor* helyén állhat egy deklarált (explicit) kurzor neve, de állhat egy lekérdezés (azaz rejtett kurzor) is. Emlékeztetünk arra, hogy a *ciklusváltozót* nem kell külön dekláralni, mert az minden esetben implicit módon történik, és érvényességi köre kizárolag a ciklusra terjed ki.

## Kurzorattribútumok

### A kurzorattribútumok használata

Az Oracle (mint már jeleztük) minden PL/SQL-ben kiadott SELECT, INSERT, DELETE és UPDATE utasításhoz *implicit kurzort* rendel. Ezeket a név nélküli kurzorokat az Oracle automatikusan kezeli, tulajdonságaiknak lekérdezése az e műveletekhez rendelt memóriaváltozókkal, az SQL előtagú úgynévezett kurzorattribútumokkal

*SQL%Kurzorattribútum*

alakban történhet (például *SQL%FOUND*).

*Explicit kurzrok* esetén a fenti kurzorattribútumokat

*KurzorNeve%Kurzorattribútum*

módon használhatjuk. Segítségükkel ellenőrizhetjük, hogy mi történt a kurzor utolsó használata során. A kurzorattribútumok tipikus alkalmazása a kurzorciklusból való kilépés feltételének figyelése (az EXIT WHEN utasításrészben).

### A PL/SQL kurzorattribútumai

- **%FOUND** visszaadott értéke TRUE, ha a legutóbbi SQL-utasítás legalább egy sort megvizsgált vagy feldolgozott,
- **%NOTFOUND** ha nincs több feldolgozandó sor, akkor a visszaadott értéke TRUE,
- **%ROWCOUNT** a megvizsgált, feldolgozott sorok számát adja vissza,
- **%ISOPEN** a kurzor megnyitását ellenőrzi; ha a kurzor nincs megnyitva, akkor értéke FALSE (implicit kurzor esetén mindig FALSE).



### Megjegyzés

A kurzorattribútumok tulajdonképpen úgy viselkednek, mint a paraméter nélküli függvények, ezért rájuk a *kurzorfüggvény* kifejezés is használatos.

## ROWID

A ROWID egy 18 karakteres pszeudooszlop, egy fizikai adattáblasor logikai címét tartalmazza. A logikai cím a tábla létrehozásakor jön létre, és adatbázisszinten egyedi. A SELECT utasítás segítségével lekérdezhető. Tipikus alkalmazása a tábla sorainak azonosítása például egy (rejtett) kurzort használó FOR-ciklus esetén.

## FOR UPDATE záradék

Szintaktikája:

FOR UPDATE [OF oszlop] [NOWAIT]

és az SQL UPDATE utasításban

CURRENT OF *kurzor*

Ha a PL/SQL-ben sorok értékeit módosítani vagy törölni szeretnénk, akkor szükséges a kurzorból a FOR UPDATE záradékot használni. A FOR UPDATE azonosítja azokat a sorokat, amelyeket módosítani, illetve törölni fogunk, valamint zárolja az eredményhalmaz minden egyes sorát. Különösen akkor hasznos, ha a módosítás a sorok már létező értékein alapul. Biztosnak kell lennünk abban, hogy módosítás előtt más felhasználó nem változtatja meg ezeket az értékeket.

A NOWAIT opcionális kulcsszóval üzenünk a rendszernek, hogy ne várakozzon a zárolt sorokra, hanem végezzen el más feladatot addig, majd próbálkozzon újra.

Amikor a kurzort megnyitjuk, minden sor zárolt lesz, nem csak a FETCH által lehozott sor. A sorok zárolása megszűnik a visszakerülés és a vélegesítés hatására. Ezért egy FOR UPDATE-tel ellátott kurzorból a COMMIT utasítás kiadása után nem lehet egy következő sort lehívni.

Amikor egy kurzorból többtáblás lekérdezés szerepel, a FOR UPDATE záradék a táblák nem mindegyikére vonatkozik. A zárolandó sorokat itt oszlopmegadással kell elérni, azaz a FOR UPDATE OF oszlop utasítással. Ez azonosítja az oszlopnéven keresztül a tábla zárolandó sorát. Tekintsük például az alábbi lekérdezést tartalmazó kurzort:

```
SELECT ename, sal, loc
  FROM emp, dept
 WHERE emp.deptno = dept.deptno
 FOR UPDATE OF sal;
```

Ez azt jelenti, hogy azt a sort (annak a táblának a sorait) zároljuk, amelyikben a sal oszlop szerepel, tehát az emp tábla zárolt lesz, míg a dept tábla nem.

Módosítás esetén a FETCH által utoljára lehívott, azaz aktuális sor azonosítása a

CURRENT OF *kurzor*

záradékkal történik, amely viszont csak a FOR UPDATE utasítással együtt használható.

## 9.1. példa

Növelje meg a hivatalnokok (clerk) fizetését a saját fizetésük 20%-ával.

### 1. Rossz megoldás (PL/SQL-ben)

```

DROP TABLE dolgozó;

CREATE TABLE dolgozó
AS SELECT * FROM emp;

SET serveroutput ON

DECLARE
  CURSOR egydolgozó IS
    SELECT empno, ename, sal
      FROM dolgozó
     WHERE UPPER(job) = 'CLERK';

  azonosító  dolgozó.empno%TYPE;
  név        dolgozó.ename%TYPE;
  fizetés    dolgozó.sal%TYPE;

BEGIN
  OPEN egydolgozó;
  LOOP
    FETCH egydolgozó
      INTO azonosító, név, fizetés;
    EXIT WHEN egydolgozó%NOTFOUND;
    fizetés := fizetés * 1.2;
    UPDATE dolgozó
      SET sal = fizetés
        WHERE UPPER(job) = 'CLERK';
    DBMS_OUTPUT.PUT_LINE(név || ' ' || fizetés);
  END LOOP;
  CLOSE egydolgozó;
END;
/

```

```

SET numwidth 6
SELECT * FROM dolgozó;
SET numwidth 10

```

**Eredmény**

A tábla eldobva.  
A tábla létrejött.

SMITH	960
ADAMS	1320
JAMES	1140
MILLER	1560

A PL/SQL eljárás sikeresen befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	1560		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANÁLYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1560		20
7900	JAMES	CLERK	7698	81-DEC-03	1560		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1560		10

**Megjegyzés**

Az eredmény csak Miller-nél jó. Több lett Smith, Adams és James fizetése, mint amit kapniuk kellett volna, mert az UPDATE utasítás minden esetben minden clerk fizetését megemelte. Az utolsó lehozott sor Miller-é, akinek az új fizetése 1560. A módosítandó egy-egy sor nincs azonosítva.

## 2. Jó megoldás (CURRENT OF záradékkal azonosítjuk az utolsónak lehívott sort, és minden csak azt módosítjuk)

DROP TABLE dolgozó;

```
CREATE TABLE dolgozó
AS SELECT * FROM emp;
```

```
SET serveroutput ON
```

```

DECLARE
    CURSOR egydolgozó IS
        SELECT empno, ename, sal
        FROM dolgozó
        WHERE UPPER(job) = 'CLERK'
        FOR UPDATE NOWAIT;

    azonosító  dolgozó.empno%TYPE;
    név        dolgozó.ename%TYPE;
    fizetés   dolgozó.sal%TYPE;

BEGIN
    OPEN egydolgozó;
    LOOP
        FETCH egydolgozó
        INTO azonosító, név, fizetés;
        EXIT WHEN egydolgozó%NOTFOUND;
        fizetés := fizetés * 1.2;

        UPDATE dolgozó
        SET sal = fizetés
        WHERE CURRENT OF egydolgozó;

        DBMS_OUTPUT.PUT_LINE( név || ' ' || fizetés);
    END LOOP;
    CLOSE egydolgozó;
END;
/

```

SET numwidth 6  
SELECT \* FROM dolgozó;  
SET numwidth 10

**Eredmény**

A tábla eldobva.  
A tábla létrejött.

SMITH	960
ADAMS	1320
JAMES	1140
MILLER	1560

A PL/SQL eljárás sikeresen befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	960		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1320		20
7900	JAMES	CLERK	7698	81-DEC-03	1140		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1560		10

14 sor kijelölve.

## Feladatok

### 9.1. feladat

Határozza meg egy PL/SQL program segítségével a felhasználó által megadott telephelyen dolgozók létszámát és bérösszegét. (Lásd a 8.1. feladatot is.)

### 9.2. feladat

Írjon PL/SQL programot, amely meghatározza a 7698 azonosítójú dolgozó nevét gyűjtőtábla használatával.

### 9.3. feladat

Írjon PL/SQL programot, amely létrehoz az emp táblából egy dolgozó táblát, és megnöveli a felhasználó által megadott foglalkozású dolgozók fizetését 1000 USD-ral.

### 9.4. feladat

Hozza létre a dolgozó táblát az emp táblából, és bővítsse azt egy sorszám oszloppal. Ezt töltse fel 1-től kiindulva egyesével növekvő értékkel minden dolgozó esetén a dolgozók nevének ábécé sorrendje szerint.

### 9.5. feladat

Írjon PL/SQL programot, amely létrehoz az emp táblából egy dolgozó táblát, és megnöveli a felhasználó által megadott százalékértékkel minden, az átlagfizetésnél alacsonyabb fizetéssel rendelkező dolgozó fizetését.

### 9.6. feladat

Írjon PL/SQL blokkot tartalmazó szkript programot, amely (eldob és) létrehoz egy dolgozó táblát az emp táblából, és ebben foglalkozásonként megnöveli a legkisebb fizetésű dolgozók bérét a foglalkozási csoportjukban legnagyobb fizetés és az ugyanitt számított átlagfizetés különbségének 20%-ával.

### 9.7. feladat

Mivel annál több a munka, minél kevesebben végzik, ezért kapjon minden dolgozó egyszeri prémiumot, amely fizetésének  $n$ -ed része, ahol  $n$  a vele azonos munkakörben dolgozók száma (őt magát is beleértve). Az emp táblából létrehozott dolgozó táblát használja.

### 9.8. feladat

Írjon PL/SQL programot, amely az emp táblából létrehozott dolgozó táblában megnöveli azoknak a főnököknek (mgr) a fizetését, akiknek egynél több beosztottjuk van. A növelés mértéke: annyiszor  $N$  USD, ahány beosztottjuk van. Az  $N$  értékét a felhasználó adja meg. A fizetés csökkenő értéke szerint rendezve listázza a főnököket.

### 9.9. feladat

Írjon PL/SQL programot, amely az emp táblából létrehozott dolgozó táblában megnöveli azoknak a dolgozóknak a fizetését, akiknek az azonosítója páros, és fizetésük kisebb, mint e csoport átlagfizetése. A növelés e fizetéskülönbség 20%-a.

### 9.10. feladat

Írjon egy olyan PL/SQL blokkot tartalmazó SQL\*Plus szkript programot, amely azon foglalkozási csoportok dolgozóiit, amelyekben az összesített jövedelem nem haladja meg a felhasználó által megadott értéket, egyszeri, a jutalékot növelő jutalomban részesíti. A jutalom összege azon foglalkozási csoportok átlagjövedelmének 1%-a, amelyeknek összes jövélme meghaladja a felhasználó által megadott előző értéket. A szkript program végül listázza ki a dolgozó tábla személyeit a nevük szerint csökkenő sorrendben.

### 9.11. feladat

Írjon egy olyan PL/SQL blokkot tartalmazó SQL\*Plus szkript programot, amely (eldob és) létrehoz az emp táblából egy dolgozó táblát, és amely elvégzi a kötelező bérrelmést a felhasználó által megadott évben oly módon, hogy a főnökök (akiknek tehát legalább egy beosztottjuk van) fizetését annak 20%-ával növeli a 3-mal osztható években, a beosztottakét pedig a fizetésük 10%-ával növeli az 5-tel osztható években. A szkript program végül listázza az alkalmazottak dolgozó tábláját a fizetésük szerint csökkenő sorrendben.

### 9.12. feladat

(Az emp táblából hozzon létre egy dolgozó táblát).

Egy PL/SQL blokkot tartalmazó szkript programmal segítsen az elnöknek (president), aki a karácsony alkalmából jutalmazni akarja dolgozót (önmagát nem!) az alábbi módon:

- minden dolgozója kapja meg a saját fizetése 5%-át jutalomként,

- kapja meg az eredeti fizetése további 10%-át
  - egy *beosztott* (akinek tehát nincs beosztottja), ha az aktuális évben 2-vel osztható a vállalatnál töltött éveinek száma,
  - egy *főnök* (akinek van legalább egy közvetlen beosztottja), ha az aktuális évben 4-gyel osztható a vállalatnál töltött éveinek száma.

A jutalmazást a jutalék (comm) növelésével oldja meg. A jutalmazott dolgozókat névsor szerint rendezett és formázott módon listázza is.

### 9.13. feladat

(Az emp táblából hozzon létre egy dolgozó táblát).

Egy PL/SQL blokkot tartalmazó szkript programmal segítsen az elnöknek (president), aki a kárcsóny alkalmából jutalmazni akarja dolgozóit (önmagát nem!) az alábbi módon:

- minden *beosztott* (akinek nincs további beosztottja) kapja meg a saját fizetése 10%-át jutalomként,
- minden *főnök* (akinek van legalább egy beosztottja) kapja meg fizetése
  - 8%-át, ha kettőnél több közvetlen beosztottja van, és
  - 4%-át, ha legfeljebb kettő közvetlen beosztottja van.

A jutalmazást a jutalék (comm) növelésével oldja meg. A jutalmazott dolgozókat névsor szerint rendezett és formázott módon listázza is.

### 9.14. feladat

(Az emp táblából hozzon létre egy dolgozó táblát).

Készítse el a dolgozók helyettesítési rendjét a következőképpen:

- a dolgozó táblát bővítsé egy helyettes oszloppal, melybe majd a helyettesítő dolgozó neve kerül,
- két dolgozó helyettesítheti egymást, ha azonos a főnökük, vagy a munkakörük,
- önmagát senki sem helyettesítheti, de előfordulhat, hogy valakinek nincs helyettese,
- minden dolgozónak legfeljebb egy helyettese lehet,
- ha az X dolgozónak Y a helyettese, akkor az Y helyettese X.

Listázza a teljes dolgozó táblát a dolgozók neve szerint rendezett és formázott módon.

## Összetett (párosítási) feladatok

### 9.15. feladat

A RAPID RALLY autóversenyre benevezett versenyzők adatbázissémája:

*versenyzők( név, feladatkör, nemzetiség, minősítés, partner ),*

ahol az attribútumok adattípusai:

- név, nemzetiség, partner: karakterSOROZATOK,
- feladatkör: (navigátor, pilóta),
- minősítés: 1..1000,

- a partner inicializált értéke: '---' vagy NULL.
1. Hozza létre a fenti táblát, és töltse fel az alábbi feladat értelmes megoldásához elegendő adattal.
  2. Készítse PL/SQL programot, amely minden versenyzőhöz megkísérel az alábbi feltételeknek megfelelő partnert keresni:
    - a partnerek nemzetisége egyezzen meg,
    - a partnerek minősítése legyen 10%-on belül,
    - a feladatkörök illeszkedjenek,
    - senkinek nem lehet egynél több partnere,
    - akinek nem jut partner, azt kizárák a versenyből.

### 9.16. feladat

A FŐBUHA vállalat dolgozóinak adatbázissémája:

*dolgozó(név, üzlet, munkakör, fizetés, gyakorlat, partner)*

ahol az attribútumok adattípusai:

- név, partner: karaktersorozatok,
- üzlet: a dolgozót foglalkoztató üzlet neve (TVBolt, PCBolt),
- munkakör: (eladó, szerelő),
- fizetés: pozitív egész szám,
- gyakorlat: a dolgozó ledolgozott éveinek száma (1 és 40 közötti érték),
- a partner inicializált értéke: '---' vagy NULL.

1. Hozza létre a fenti táblát az alábbi megszorításokkal. Adatfelviteli kísérletekkel tesztelje a megszorításokat, és töltse fel a táblát legalább 9 sorral.
  - Legyen a név elsőleges kulcs a *dolgozó* táblában,
  - Egy eladó nem kaphat 50 000 Ft-nál kevesebb, és 120 000 Ft-nál több, egy szerelő pedig 80 000 Ft-nál kevesebb, és 180 000 Ft-nál több fizetést.
2. Növelje meg a 10 évnél régebb óta dolgozók fizetését a munkakörük átlagának 10%-ával, de legfeljebb az adott munkakörhöz tartozó fizetéstartomány felső értékére, és listázza az összes dolgozó minden adatát név szerint rendezve. Ügyeljen arra, hogy a növelési tevékenység során az eredeti fizetésértékek alapján számoljon!
3. Készítse egy SQL\*Plus programba ágyazott PL/SQL blokkot, amely a felhasználó által megadott üzlet minden, helyettesítővel még nem rendelkező dolgozójának partner mezőjébe beírja egy olyan dolgozónak a nevét,
  - akinek gyakorlati ideje legfeljebb 2 ével tér el a helyettesítendő dolgozó gyakorlati idejétől,
  - helyettesként még senkihez sincs párosítva (önmagához nem párosítható),
  - a helyettesítendő dolgozóval azonos munkakörben dolgozik,
  - ugyanazon üzlet dolgozója,
  - a „partner” tulajdonság *szimmetrikus*, tehát ha egy X személy partnere az Y, akkor annak partnere az X.

### 9.17. feladat

A BEPAKOL vállalat konténereibe legfeljebb két tárgy lehető. A tárgyak adatbázissémája:

*tárgyak(név, megrendelő, típus, súly, sürgősségi, másik)*

ahol az attribútumok adattípusai:

- név, másik: karakterszorozatok,
- megrendelő: a tárgyat megrendelő vállalkozók neve (Tóth, Kovács),
- típus: a tárgyak típusa (Monitor, TV),
- súly: a tárgyak súlya (pozitív egész szám),
- sürgősségi: a leszállítás sürgőssége (1 és 10 közötti érték, 10: a legsürgősebb),
- másik: az adott tárggyal egy konténerbe helyezhető másik tárgy neve, inicializált értéke: '---' vagy NULL.

1. Hozza létre a fenti táblát az alábbi megszorításokkal. A megszorításokat tesztelje adatfelviteli kísérletekkel, és töltse fel a táblát legalább 9 sorral.
  - Legyen a megrendelő, a név és a sürgősségi együttesen elsődleges kulcs,
  - Egy Monitor súlya nem lehet 3 kp-nál kevesebb, és 13 kp-nál több, egy TV-é pedig 5 kp-nál kevesebb, és 25 kp-nál több.
2. Növelje meg a 10 kp-nál nehezebb tárgyak sürgősségi értékét a megrendelőjükhez tartozó tárgyak átlagos sürgősségi értékének 8%-ával, de legfeljebb a határértékre, és listázza az összes tárgy minden adatát név szerint rendezve. Ügyeljen arra, hogy a növelési tevékenység során az eredeti sürgősségi értékek alapján számoljon!
3. Készítsen egy SQL\*Plus programba ágyazott PL/SQL blokkot, amely a felhasználó által megadott megrendelő minden, az együttszállításra még nem társított tárgyának a „másik” mezőjébe beírja egy olyan tárgynak a nevét,
  - amelynek sürgősségi értéke legfeljebb 3-mal tért el a társítandó tárgy sürgősségi értékétől,
  - amely még nincs egy tárgyhoz sem társítva (önmagához nem társítható),
  - a hivatalos tárggyal azonos típusú,
  - ugyanazon megrendelő tulajdona,
  - a „társ” tulajdonság szimmetrikus, tehát ha egy X tárgy társa az Y, akkor annak társa az X.

### 9.18. feladat

A FEKETE KIRÁLYNŐ sakkkaversenyre benevezett versenyzők adatbázissémája:

*versenyzők(név, nem, nemzetiség, élőpontszám, partner)*,

ahol az attribútumok adattípusai:

- név, nemzetiség, partner: karakterszorozatok,
- nem: (férfi, nő),
- élőpontszám: 1..10000,
- a partner inicializált értéke: '---', vagy NULL.

1. Hozza létre a fenti táblát, és töltse fel a 2. részfeladat értelmes megoldásához elég adattal.
2. Készítsen PL/SQL programot, mely minden versenyzőhöz megkísérel az alábbi feltételeknek megfelelő partnert keresni:

- a partnerek nemzetisége egyezzen meg,
- a partnerek minősítései legyenek 20%-on belül,
- egymással csak azonos neműek játszhatnak,
- senkinek nem lehet egynél több partnere,
- akinek nem jut partner, azt kizárták a versenyből.

### 9.19. feladat

A PÁRKERESŐ szolgálat ügyfeleinek adatbázissémája:

*személyek(név, nem, szül\_dátum, szándék, párjelölt)*

ahol az attribútumok adattípusai:

- név, párjelölt: karaktersorozatok,
- nem: (férfi, nő),
- szül\_dátum: születési dátum, ahol egy női ügyfél nem lehet 16 évnél, egy férfi ügyfél pedig 18 évnél fiatalabb,
- szándék: az ügyfél házasodási szándékának mértéke (0 .. 10, ahol 0: nem akar házasodni, .. 10: nagyon akar házasodni),
- a párjelölt inicializált értéke: '---' vagy NULL.

1. Hozza létre e táblát a fenti megszorításokkal. Adatfelviteli kísérletekkel tesztelje a megszorításokat, és töltse fel a táblát legalább 9 sorral.
2. Növelje meg a 30 év alatti ügyfelek szándékértékét, az azonos korosztályú ügyfelek átlagos szándékértéke 20%-ának, a 30 év feletti ügyfelek szándékértékét pedig az azonos korosztályú ügyfelek átlagos szándékértéke 40%-ának egy tizedesre kerekített értékével, de legfeljebb a határértékre, és listázza az összes ügyfél minden adatát név szerint rendezve. Ügyeljen arra, hogy a növelési tevékenység során az eredeti szándékértékei alapján számoljon!
3. Készítsen egy SQL\*Plus programba ágyazott PL/SQL blokkot, amely az ügyfeleket úgy párosítja, hogy az ügyfelek párjelölt mezőjébe beírja a kiválasztott másik ügyfél nevét, ahol a kiválasztási szempontok az alábbiak:

  - minden ügyfél csak ellenkező nemű, legfeljebb 10 év korkülönbségű ügyféllel párosítható,
  - a szándékértéke legfeljebb 3-mal térhet el a másik hasonló értékétől,
  - csak monogám párosítások engedhetők meg,
  - a „párosítás” tulajdonság szimmetrikus, tehát ha egy X személy párja az Y, akkor annak párja az X.

### 9.20. feladat

A kétágyas körtermekkel rendelkező ÚRIKÓRHÁZ Betegelosztó Irodáján a betegnyilvántartó adatbázissémája:

*betegek(név, nem, szül\_dátum, osztály, állapot, szobatárs)*

ahol az attribútumok adattípusai:

- név, szobatárs: karaktersorozatok,
- nem: (férfi, nő),

- szül\_dátum: születési dátum,
- osztály: (Sebészeti, Belgyógyászati),
- állapot: a beteg állapotának súlyossága (sebészeti beteg esetén 3 és 10, belgyógyászati beteg esetén 1 és 8 közötti érték),
- a szobatárs inicializált értéke: '---' vagy NULL.

1. Hozza létre e táblát a fenti megszorításokkal. Adatfelviteli kísérletekkel tesztelje a megszorításokat, és töltse fel a táblát legalább 9 sorral.
2. Növelje meg a 60 évnél idősebb férfiak állapotértékét a velük azonos osztályon fekvő férfiak átlagos állapotértékének 10%-ával, de legfeljebb a megengedett tartomány felső értékére, és listázza az összes beteg minden adatát elsődlegesen az osztály, másodlagosan a név szerint rendezve. Ügyeljen arra, hogy a növelési tevékenység során az eredeti állapotértékek alapján számoljon!
3. Készítsen egy SQL\*Plus programba ágyazott PL/SQL blokkot, amely a felhasználó által megadott osztály minden, szobatárssal még nem rendelkező betegének szobatárs mezőjébe beírja egy olyan beteg nevét, akire az alábbi feltételek mindegyike teljesül, azaz
  - vele azonos nemű,
  - vele azonos osztályon fekszik,
  - állapotértéke legfeljebb 3-mal tér el,
  - szobatársként még senkihez sincs párosítva (önmagához nem párosítható),
  - a „párosítás” tulajdonság szimmetrikus, tehát ha egy X személy párja az Y, akkor annak párja az X.

### 9.21. feladat

A SZEGÉNYKÓRHÁZ orvosait nyilvántartó adatbázissémája:

*orvosok<név, szakképesítés, gyakorlat, fizetés, beosztás, helyettes>*

ahol az attribútumok adattípusai:

- név, helyettes: karaktersorozatok,
- szakképesítés: (szemész, sebész),
- gyakorlat: az orvos ledolgozott éveinek száma (1 és 40 közötti érték),
- fizetés: az orvos fizetése,
- beosztás: a beosztás neve (segédorvos 1-5, alorvos 2-8, adjunktus 3-15, főorvos 10-40 év gyakorlat esetén),
- a helyettes inicializált értéke: '---' vagy NULL.

1. Hozza létre e táblát a fenti megszorításokkal. Adatfelviteli kísérletekkel tesztelje a megszorításokat, és töltse fel a táblát legalább 9 sorral.
2. Növelje meg minden 15 évnél régebb óta dolgozó fizetését a vele azonos szakképesítésűek átlagának 15%-ával, de legfeljebb a vele azonos beosztású, legmagasabb fizetésű orvos fizetésére, és listázza az orvosok minden adatát elsődlegesen a szakképesítés, másodlagosan pedig a név szerint rendezve. Ügyeljen arra, hogy a növelési tevékenység során az eredeti fizetések alapján számoljon!

3. Készítsen egy SQL\*Plus programba ágyazott PL/SQL blokkot, amely minden, a felhasználó által megadott szakképesítésű, helyettesessel még nem rendelkező orvos helyettes mezőjébe beírja egy olyan orvos nevét, akire az alábbi feltételek mindegyike teljesül, azaz

- vele azonos szakképesítésű,
- gyakorlati ideje legfeljebb 5 évvvel tér el,
- beosztása a listában legfeljebb a saját beosztásának szomszédja,
- helyettesként még senkihez sincs párosítva (önmagához nem párosítható),
- a „párosítás” tulajdonság *szimmetrikus*, tehát ha egy X személy párja az Y, akkor annak párja az X.

## 9.22. feladat

A LERENDEZLEK rendezvényszervező iroda előadóművészeket nyilvántartó adatbázisának séma-jája:

*művész(név, műfaj, díj, ismertség, helyettes)*

ahol az attribútumok adattípusai:

- név, helyettes: karaktersorozatok,
- műfaj: (zenész, bohóc, politikus),
- díj: a művész tiszteletdíja (zenész esetén 10 000-20 000 Ft, bohóc esetén 10 000-100 000 Ft, politikus esetén 50 000-500 000 Ft),
- ismertség: a művész ismertségének mértéke (1 és 100 közötti érték),
- a helyettes inicializált értéke: '---' vagy NULL.

1. Hozza létre e táblát a fenti megszorításokkal. Adatfelviteli kísérletekkel tesztelje a megszorításokat, és töltse fel a táblát legalább 9 sorral.
2. Növelje meg a 70-es értéknél ismertebb művészek tiszteletdíját a vele azonos műfajúak átlagának 8%-ával, de legfeljebb a műfajuk szerinti felső korlátra, és listázza őket elsődlegesen a műfaj, másodlagosan pedig a név szerint rendezve. Ügyeljen arra, hogy a növelési tevékenység során az eredeti tiszteletdíjértékek alapján számoljon!
3. Készítsen egy SQL\*Plus programba ágyazott PL/SQL blokkot, amely a felhasználó által megadott műfajú minden, helyettessel még nem rendelkező művésznek a helyettes mezőjébe beírja egy olyan művész nevét, akire az alábbi feltételek mindegyike teljesül, azaz
  - vele azonos műfajú,
  - tiszteletdíja legfeljebb 20%-kal tér el,
  - ismertségértéke legfeljebb 20-szal tér el,
  - helyettesként még senkihez sincs párosítva (önmagához nem párosítható).
  - a „párosítás” tulajdonság *szimmetrikus*, tehát ha egy X személy párja az Y, akkor annak párja az X.

## 10. FEJEZET

# Kivételkezelés, alprogramok, triggerek

## Elméleti összefoglaló

### Kivételkezelés (EXCEPTION)

A PL/SQL-ben a kivétel egy azonosítóval ellátott esemény, amelynek hatására a blokk végrehajtása félbeszakad. A kivételkezelő rész a blokk azon része, amelyre kivétel esetén a vezérlés adódik. (Vázlatosan már utaltunk rá a 8. fejezetben.) A kivételkezelő résszel együtt a teljes blokk alakja:

```
DECLARE  
...  
BEGIN  
...  
EXCEPTION  
...  
END;
```

### A kivételek fajtái

#### *A FELHASZNÁLÓ ÁLTAL DEFINIÁLT KIVÉTELEK*

A kivételt deklarálni kell (EXCEPTION típussal), majd a RAISE utasítás kiváltja az adott nevű kivételt. Ekkor a vezérlés a blokk kivételkezelő részére ugrik, ahol a hibához tartozó utasításrész végrehajtódik („lekezeli a hibát”), melynek során tipikusan felhasználói üzenet is kiíródik.

*Alakja:*

```
RAISE [kivételnév];
```

*Tipikus használata:*

```
DECLARE  
  Kivételnév EXCEPTION;  
...
```

```

BEGIN
...
  IF LogikaiKifejezés
  THEN
    RAISE KivéteLnév;
...
END IF

EXCEPTION
  WHEN KivéteLnév
  THEN utasítások;
END;

```

A felhasználó által definiált kivételeket társíthatjuk a rendszerkivételekhez az alprogram (lásd alább) deklaráló részében:

pragma EXCEPTION\_INIT (*kivéteLnév, hibakód*);

ahol:

- *kivéteLnév* egy előzőleg deklarált kivétel,
- *hibakód* egy már létező Oracle-hiba sorszáma.

A pragma a fordítónak szóló direktíva (a futtatható kódba nem kerülő utasítás).

### **A PL/SQL ELŐREDEFINIÁLT AUTOMATIKUS RENDSZERKIVÉTELEI**

Ezeket a futtatórendszer váltja ki, és a STANDARD csomagban találhatók, melyet a PL/SQL használ.

#### **Példa (részlet)**

```

EXCEPTION
  WHEN No_Data_Found -- a SELECT INTO nem ad vissza sorokat
  THEN üzenet;
  WHEN Too_Many_Rows -- a SELECT INTO több sorral tér vissza
  THEN üzenet;
  WHEN OTHERS          -- minden más meg nem nevezett hiba esetén
  THEN üzenet;          -- SQLCODE és SQLEERRM hibaüzenettel és kóddal
END;                      -- tér vissza

```

## **Alprogramok**

A névvel ellátott és paraméterezhető blokkot alprogramnak nevezzük. Az alprogramok lehetnek eljárások és függvények.

### Az eljárás szintaktikája

```
PROCEDURE név [(paraméter [, paraméter]...)]
IS
  [lokális deklarációk (a DECLARE kulcsszó nélkül)]
BEGIN
  végrehajtható utasítások
[EXCEPTION
  kivételkezelés]
END [név];
```

### A függvény szintaktikája

```
FUNCTION név [(paraméter [, paraméter]...)]
RETURN adattípus
IS
  [lokális deklarációk (a DECLARE kulcsszó nélkül)]
BEGIN
  végrehajtható utasítások
  RETURN visszatérési érték
[EXCEPTION
  kivételkezelés]
END [név];
```

### A paraméter szintaktikája (mindkét esetben):

paraméter\_név [IN | OUT | IN OUT] adattípus [{DEFAULT | :=} kifejezés]

- IN bemenő paramétert jelöl (az értéke nem változtatható meg az alprogramban, ez az alapértelmezés, ezért az IN kulcsszó elhagyható),
- OUT kimenő paramétert jelöl (az alprogramban csak kaphat, de nem adhat át értéket),
- IN OUT be- és kimenő paramétert jelöl (az alprogramban kaphat és adhat is értéket),
- kifejezés kezdőérték megadására szolgál (csak az IN, vagy az IN OUT paraméterek esetén használható).

### Paraméterek adattípusai

Az alprogramokban a már megismert PL/SQL-típusok használhatók azzal a megkötéssel, hogy nem tartalmazhatnak korlátozásokat (például a hosszságra vagy a pontosságra vonatkozóan). Ennek megfelelően használhatók a NUMBER, az INTEGER, a REAL, a CHAR, a VARCHAR2 stb. (így, ebben a formában!). Természetesen használhatók a hivatkozott (%TYPE, %ROWTYPE) típusok is. (Részletesen lásd [15] és [16].)

### Példa (részlet)

```
PROCEDURE Bolygók(sugár      IN      INTEGER,
                  térfogat OUT      REAL,
                  sorrend IN OUT INTEGER)
```



#### Megjegyzés

A függvényeknél lehetőleg ne használjunk OUT és IN OUT paramétereket, mert mélékhatást okozhatnak. A függvény használatának célja egy visszatérési érték meghatározása.

## 10.1. példa

Mivel közeledik az év vége, ezért néhány dolgozó jutalmat kap a tulajdonostól. A jutalom a dolgozó fizetésének 10%-a, feltéve, hogy a fizetése 3000 USD alatt van, egyébként csak egy „Boldog Karácsonyt!” üdvözletet kap. Írjon erre egy PL/SQL programot, amely eljárást és kivételkezelést is tartalmaz. A jutalmat az emp táblából létrehozott dolgozó tábla jutalék (comm) értékéhez adja hozzá.

### Megoldás

```
SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

-- JutalmazóProgram
ACCEPT neve PROMPT 'Adja meg a jutalmazandó dolgozó nevét:'

-- A főprogram deklarációs szegmense
DECLARE
    v_neve      dolgozó.ename%TYPE;
    v_jutalma   dolgozó.sal%TYPE;
    NincsJutalom EXCEPTION;

-- Az alprogram deklaráció
PROCEDURE Jutalmazás (p_neve      IN      dolgozó.ename%TYPE,
                      p_jutalma   OUT     dolgozó.sal%TYPE)
IS
    -- Lokális deklaráció
    p_fizetése dolgozó.sal%TYPE;
BEGIN
    SELECT sal
    INTO p_fizetése
    FROM dolgozó
    WHERE UPPER(ename) = UPPER(p_neve);
```

```

IF p_fizetése >= 3000
THEN
    RAISE NincsJutalom;
ELSE
    p_jutalma := 0.1 * p_fizetése;
    DBMS_OUTPUT.PUT_LINE('>>');
    DBMS_OUTPUT.PUT_LINE('>> ' || p_neve || ' jutalma: ' ||
                           p_jutalma || ' USD');
END IF;

EXCEPTION
    WHEN NO_DATA_FOUND
        THEN
            DBMS_OUTPUT.PUT_LINE('>>');
            DBMS_OUTPUT.PUT_LINE('>> NINCS ilyen nevű dolgozó...');

    WHEN NincsJutalom
        THEN
            DBMS_OUTPUT.PUT_LINE('>>');
            DBMS_OUTPUT.PUT_LINE('>> Boldog Karácsonyt Kedves ' ||
                           p_neve || '!');
END Jutalmazás;
-- Az alprogram deklarációjának vége

-- Főprogram blokkja
BEGIN
    v_neve := '&neve';
    Jutalmazás(v_neve, v_jutalma);
    UPDATE dolgozó
        SET comm = NVL(comm,0) + v_jutalma
        WHERE UPPER(ename) = UPPER(v_neve);
END; -- JutalmazóProgram
/

```

-- PROMPT Listázás:

```

SELECT ename      AS "Név",
       job        AS "Munkakör",
       sal        AS "Fizetés",
       NVL(comm,0) AS "Jutalék+Jutalom"
  FROM dolgozó
 WHERE UPPER(ename) = UPPER('&neve');

```

### 1. futtatás eredménye

A tábla eldobva.  
A tábla létrejött.

Adja meg a jutalmazandó dolgozó nevét: Smith  
 >>  
 >> Smith jutalma: 80 USD

A PL/SQL eljárás sikeresen befejeződött.

Név	Munkakör	Fizetés	Jutalék+Jatalom
SMITH	CLERK	800	80

## 2. futtatás eredménye

Adja meg a jutalmazandó dolgozó nevét: Mikulás  
 >>  
 >> NINCS ilyen nevű dolgozó...

A PL/SQL eljárás sikeresen befejeződött.

nincsenek kijelölve sorok

## 3. futtatás eredménye

Adja meg a jutalmazandó dolgozó nevét: Ford  
 >>  
 >> Boldog Karácsonyt Kedves Ford!

A PL/SQL eljárás sikeresen befejeződött.

Név	Munkakör	Fizetés	Jutalék+Jatalom
FORD	ANALYST	3000	0

# Tárolt alprogramok

Az alprogramok eltárolhatók, ezzel lehetőség van újrafelhasználásra, saját vagy más felhasználó számára (ha erre van jog).

Megjegyezzük, hogy a SHOW ERRORS SQL\*Plus-utasítás segítségével listázhatjuk az alprogramok fordításánál bekövetkezett hibákat.

## Tárolt alprogramok létrehozása

```
CREATE OR REPLACE PROCEDURE eljárásnév [(paraméter [, paraméter]...)]
{IS | AS}
[lokális deklarációk (a DECLARE kulcsszó nélkül)]
blokk
```

```
CREATE OR REPLACE FUNCTION függvénynév [(paraméter [, paraméter]...)]
RETURN adattípus
{IS | AS}
[lokális deklarációk (a DECLARE kulcsszó nélkül)]
blokk
```

## Tárolt alprogramok törlése

```
DROP PROCEDURE eljárásnév;
DROP FUNCTION függvénynév;
```

## Tárolt alprogramok hibakezelése

A RAISE\_APPLICATION\_ERROR eljárás a tárolt alprogramban a felhasználó által definiált, és előredefiniált kivételek üzenetét segíti megjeleníteni. Használatával megtudhatjuk, hogy az alprogramok közül melyikben történt a kivétel, és azt mi okozta. Ez az eljárás bárhonnan hívható. Szintaktikája:

```
RAISE_APPLICATION_ERROR(hibakód, üzenet)
```

ahol:

- a hibakód értékének engedélyezett tartománya: -20000 ... -20999,
- az üzenet egy karaktersorozat, és rövidebb 2048 bajtnál.



### Megjegyzés

A RAISE\_APPLICATION\_ERROR eljárás felfüggeszti az őt tartalmazó tárolt alprogramban a DBMS\_OUTPUT.PUT\_LINE kiírató eljárások hatását. Az ezzel történő kiíratások zárolását, vagyis e zárolt memóriaterületről a szöveg képernyőn való megjelenését csak egy névtelen blokkban történő kiíratás szabadítja fel. (Lásd még a 10.1. feladat megoldásánál tett megjegyzést.)

### Példa (részlet)

```
hiba_1 EXCEPTION;
hiba_2 EXCEPTION;
BEGIN
  ...
  IF NOT alprogfeltéte1(aktuális paraméter lista)
  THEN
    RAISE hiba_1;
  END IF;
  IF NOT alprogfeltéte2(aktuális paraméter lista)
  THEN
    RAISE hiba_2;
  END IF;
```

```
-- A kivételkezelő rész
EXCEPTION
  WHEN hiba_1 THEN
    RAISE_APPLICATION_ERROR(-20101, 'Kivétel az 1.alprogramban!');
  WHEN hiba_2 THEN
    RAISE_APPLICATION_ERROR(-20102, 'Kivétel a 2.alprogramban!');
END;
```

## Tárolt alprogramok lekérdezése

A már eltárolt alprogramok (és triggerek, lásd alább) lekérdezhetők az adatszótár nézeteiből a user\_source nézettábla segítségével, melynek szerkezete:

```
SET linesize 65
DESC user_source
SET linesize 400
```

### Eredmény

Név	Üres?	Típus
NAME		VARCHAR2(30)
TYPE		VARCHAR2(12)
LINE		NUMBER
TEXT		VARCHAR2(4000)

## 10.2. példa

Készítse egy tárolt függvényt, mely TRUE értékkel tér vissza, ha az ÚjFizetés bemenő paraméter értéke nagyobb a RégiFizetés bemenő paraméternél, egyébként FALSE értékkel. A tárolt függvény forráskódját kérdezze le.

### Megoldás

#### 1. lépés (A tárolt függvény elkészítése)

```
CREATE OR REPLACE FUNCTION
  Fizetés (ÚjFizetés IN NUMBER,
            RégiFizetés IN NUMBER)
RETURN BOOLEAN IS
BEGIN
  IF ÚjFizetés > RégiFizetés
  THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
```

```

    END IF;
END;
/
SHOW ERRORS

```

**Eredmény**

A függvény létrejött.

Nincsenek hibák.

**2. lépés (A forráskód lekérdezése)**

```

COLUMN text FORMAT A50
COLUMN line FORMAT 9999
SELECT line, text
  FROM user_source
 WHERE UPPER(type)='FUNCTION' AND
       UPPER(name)='FIZETÉS';
CLEAR COLUMNS

```

**Eredmény**

LINE TEXT

---

```

1 FUNCTION
2   Fizetés (ÚjFizetés      IN NUMBER,
3             RégiFizetés   IN NUMBER)
4 RETURN BOOLEAN IS
5 BEGIN
6   IF ÚjFizetés > RégiFizetés
7   THEN
8     RETURN TRUE;
9   ELSE
10    RETURN FALSE;
11  END IF;
12 END;

```

12 sor kijelölve.

**Tárolt eljárás futtatása**

A már eltárolt alprogramokat jellemzően SQL\*Plus szkript programbeli PL/SQL blokkból való hívással szoktuk meghívni, azonban lehetőség van arra is, hogy egy tárolt eljárást közvetlenül az SQL\*Plus-környezetből hívunk meg az

`EXECUTE [felhasználó.]TároltEljárásNeve`

SQL\*Plus-utasítás segítségével, ahol a felhasználó megadása elmaradhat.

### 10.3. példa

Készítsen egy tárolt eljárást, mely a paramétereként adott két egész számot összeadja, és az eredményt kiírja a képernyőre. A kiíratás előtt írjon ki egy üres sort, és az eredménysor tabulálással kezdődjön. Az eljárást futtassa le közvetlenül az SQL\*Plus-környezetből.

#### Megoldás

```

CREATE OR REPLACE PROCEDURE
    Összead (Egyik IN NUMBER,
              Másik IN NUMBER)
IS
    Eredmény NUMBER(3);
    Kiírás VARCHAR2(20);
BEGIN
    Eredmény := Egyik + Másik;
    Kiírás := CHR(10)          || /* soronelés (LF) */
              CHR(9)           || /* tabulálás (TAB) */
              TO_CHAR(Egyik)   || ' + ' ||
              TO_CHAR(Másik)   || ' = ' ||
              TO_CHAR(Eredmény);
    DBMS_OUTPUT.PUT_LINE(Kiírás);
END;
/
SHOW ERRORS

```

```

SET serveroutput ON
EXECUTE Összead(3,4)

```

#### Eredmény

Az eljárás létrejött.

Nincsenek hibák.

$$3 + 4 = 7$$

A PL/SQL eljárás sikeresen befejeződött.



#### Megjegyzés

Figyeljünk fel a vezérlőkarakterek használatára!

## Triggerek

A trigger olyan PL/SQL nyelven írt tárolt program, amelyet valamilyen esemény indít el (aktivizál, tüzel). A triggerek a hozzájuk rendelt esemény hatására automatikusan aktivizálódnak, a felhasználó azt befolyásolni nem tudja. (Emiatt a triggerek és az általuk hívott eljárások nem tartalmazhatnak COMMIT, ROLLBACK és SAVEPOINT utasításokat.) Egy trigger működése más triggereket is elindíthat. Az ilyen láncolt kapcsolatokat a triggerek tervezésekor figyelembe kell venni. (A triggerekről bővebben lásd [15].)

### Triggerindító események

A triggert indító események lehetnek:

- a DELETE, INSERT, UPDATE DML-utasítások,
- a CREATE, ALTER, DROP DDL-utasítások,
- rendszeresemények, mint például az adatbázis indítása, leállítása és hibaüzenetek,
- felhasználói események, mint például egy felhasználó be- vagy kilépése.

A továbbiakban mi a DML-utasításokhoz (tehát törléshez, beszúráshoz és módosításhoz) kapcsolódó triggerekkel fogunk foglalkozni.

### A triggerek alkalmazása

A triggerek alkalmazásának elsődleges célja az adatbázis integritásának (a benne tárolt adattáblák hitelességének és összhangjának) védelme, az adatbázisban történt változások feljegyzése. Ennek értelmében triggereket használhatunk:

- felhasználói előírások érvényesítésére (például többtáblás függőségek megvalósítására – lásd 4. melléklet),
- érvénytelen tranzakciók megelőzésére,
- származtatott (kiszámított) oszlopértékek automatikus generálására,
- adattáblák szinkronizálására (például elsődleges kulcs–idegen kulcs kapcsolathoz),
- események (törles, beszúrás, módosítás) naplózására.

### A triggerek felépítése és lekérdezése

A trigger felépítésének általános alakja:

- A trigger feje
- A trigger törzse

ahol a trigger törzse egy névtelen PL/SQL blokk, a trigger feje (description) pedig tartalmazza a trigger jellemzőit. Az adatbázisban tárolt trigger fejrészei a

```
SELECT description
  FROM user_triggers;
```

utasítással, míg a trigger teljes forráskódja a fent már bemutatott módon a

```
SELECT line, text
  FROM user_source
 WHERE UPPER(type)='TRIGGER' AND
       UPPER(name)='TriggerNeve';
```

utasítással kérdezhető le, ahol *TriggerNeve* a lekérdezendő trigger neve.

## A tárolt alprogramok és triggerek lekérdezése

Mivel a tárolt alprogramok és triggerek (hasonlóan az adat- és nézettáblákhoz) olyan objektumok, melyeket az Oracle-rendszer tárol (és így az SQL\*Plus-környezetből való kilépés után is megőrződnek), ezért célszerű időnként lekérdezni őket. Ez a

```
SELECT object_type, object_name
  FROM user_objects
 WHERE object_type IN ('PROCEDURE', 'FUNCTION', 'TRIGGER');
```

utasítással tehető meg.

## A triggerek jellemzői

- A trigger időzítése (BEFORE, AFTER, INSTEAD OF).
- A trigger által figyelt esemény (DELETE, INSERT, UPDATE).
- A trigger típusa (utasításszintű, sorszintű).
- A trigger által felügyelt (a triggerhez rendelt) objektum (adattábla, nézettábla).

## A triggerek szintaktikus alakja

```
CREATE [OR REPLACE] TRIGGER triggernév
  [BEFORE | AFTER | INSTEAD OF]
  triggerelő_esemény [OR triggerelő_esemény]...
  ON {adattábla | nézettábla}
  [FOR EACH ROW
    [WHEN feltétel]]
  [DECLARE
    lokális változó deklarációja]
  BEGIN
    utasítások
    [kivételkezelés]
  END;
```

ahol:

- a REPLACE opciót akkor használjuk, ha a trigger már létezik,
- a *triggerelő esemény* egy DML-utasításhoz kapcsolódik, szintaktikus alakja:  
`{DELETE | INSERT | UPDATE [OF oszlop]}`,
- az OF kulcsszóval megadhatjuk, hogy melyik oszlop módosítására legyen érzékeny a trigger (ha ez hiányzik, akkor bármely oszlop módosítására érzékeny),
- a trigger törzse tartalmazhat kivételkezelést,
- a FOR EACH ROW opció teszi a triggert sorszintűvé,
- a sorszintű trigger minden olyan soron végrehajtódik, melyet a triggert kiváltó DML-utasítás érint,
- az INSTEAD OF triggerek (az Oracle 8-as verziójától léteznek) csak sorszintűek lehetnek, ezért esetükben a FOR EACH ROW opció el is hagyható,
- a BEFORE és az AFTER triggerek csak adattáblára, míg az INSTEAD OF triggerek csak nézettáblára definiálhatók (melyet ezért előzetesen létre kell hozni a módosítandó adattáblára vonatkozóan),
- az INSTEAD OF trigger mindig az őt elindító DML-művelet helyett aktivizálódik, ezért (ellettében a BEFORE és AFTER triggerekkel) a triggerben elő kell írni a megfelelő DML-műveletet (e megoldás járulékos előnye, hogy egy esetlegesen előírt törlési művelet helyett végrehajthatunk például egy módosítást),
- az utasításszintű (tehát FOR EACH ROW opciót nem tartalmazó BEFORE és AFTER) triggerek az általuk felügyelt táblára vonatkozó triggerelő esemény hatására minden aktivizálódnak (lefutnak), akkor is, ha az esemény egyetlen sort sem kezel,
- a sorszintű triggerekben, valamint az e triggerekből hívott tárolt alprogramokban hivatkohatunk a módosított sorok eredeti és új értékeire az OLD és a NEW korrelációs nevek segítségével (lásd alább),
- a WHEN utáni feltétel a sorszintű triggerek működését az e feltétel által kijelölt sorokra korlátozza, ahol a feltételben az OLD és a NEW korrelációs nevekkel minősített adatokra kell hivatkogni.

## A triggerek működésének jellegzetességei

- Ha egy triggerben, vagy benne meghívott tárolt alprogramban szerepel a RAISE\_APPLICATION\_ERROR eljárás (és ez meg is kapja a vezérlést), akkor a triggert kiváltó utasítás nem kerül végrehajtásra, illetve automatikusan visszagördül (mintha egy ROLLBACK utasítást adtunk volna ki, lásd még a 10.4. példa megjegyzését).
- A RAISE\_APPLICATION\_ERROR eljárás további hatása, hogy felfüggeszti a DBMS\_OUTPUT.PUT.PUT\_LINE kiírató eljárás hatását mind a triggerben, mind a belőle hívott tárolt eljárásokban. A DBMS\_OUTPUT.PUT\_LINE eljárással történő kiíratások zárolását, vagyis e zárolt memóriaterületről a szöveg képernyőn való megjelenését csak egy névtelen blokkban történő kiíratás szabadítja fel. (Lásd még a Tárolt alprogramok hibakezelése c. pontot jelen fejezet korábbi részében, valamint a 10.1. feladat megoldásánál tett megjegyzést).
- Egy adattábla elsődleges- és idegenkulcs-megszorításokkal ellátott oszlopainak értékeit nem lehet triggerből megváltoztatni, illetve ilyen megszorításokat tartalmazó táblán csak

olyan sorok törölhetők triggerrel, melyeket nem védenek e megszorítások. (Például az emp táblában nem törölhetők a főnökök, lásd 10.5. példa.)

- Egy (DML-utasítás által) módosulásra kijelölt tábla BEFORE vagy AFTER triggerből nem kérdezhető le, és nem módosítható, kivéve, ha a triggerelő esemény egyetlen sor beszúrása. (Az INSTEAD OF triggerre ilyen korlátozás nincs.)

## A korrelációs nevek használata

Gyakran előfordul, hogy egy sorszintű triggerben szükségünk volna a triggerelő esemény (utasítás) valamelyik adatparaméterére, illetve a triggerhez rendelt (a trigger által felügyelt) táblának a triggerelő esemény által kijelölt sora valamelyik adatmezőjére. Az ilyen igények kielégítésére szolgálnak az OLD és NEW korrelációs nevek, melyeket minősítő nevekként kell használni a triggerhez rendelt adattábla, vagy nézettábla oszlopaira például :OLD.ename, illetve :NEW.ename módon. A korrelációs nevek jelentése az egyes DML-utasításoknál az alábbi:

- DELETE műveletnél az utasításbeli adatra :OLD értékként kell hivatkozni, a :NEW értéke NULL,
- INSERT műveletnél az utasításbeli adatra :NEW értékként kell hivatkozni, az :OLD értéke NULL,
- UPDATE műveletnél az utasításbeli adatra :NEW értékként, a tárolt adatra pedig :OLD értékként kell hivatkozni.

## Trigger törlése

A triggert a CREATE OR REPLACE előírás ellenére célszerű a létrehozása előtt törölni, mivel nem történik meg a régi változat felülírása, ha az például másik táblára vonatkozott.

A trigger törlése a

```
DROP TRIGGER TriggerNeve;
```

utasítással lehető meg.

Ügyeljünk arra, hogy az adattáblára definiált BEFORE és AFTER triggerek az adattáblájuk törlésekor automatikusan töröldnek. Ezzel ellentétben az INSTEAD OF triggerek a hozzájuk rendelt nézettábla, illetve az abban hivatkozott adattábla törlésekor nem töröldnek.

## Azonos eseményre tüzelő triggerek

Az azonos eseményre tüzelő triggerek közül a legrégebbi aktivizálódik, ezért ügyelni kell arra, hogy az azonos fejrészű (description) triggerök közül a régebbieket töröljük. (Ehhez persze ismernünk kell, hogy milyen triggerek léteznek, amihez pedig le kell kérdezni a triggereket, lásd korábban Triggerok felépítése és lekérdezése és A tárolt alprogramok és triggerek lekérdezése c. pontok.)

## A triggerek letiltása és engedélyezése

A már tárolt (létrehozott) triggereket letilthatjuk és engedélyezhetjük. Az engedélyezett trigger a hozzá rendelt esemény hatására aktiválódik, a letiltott pedig nem. Létrehozásukkor a triggerek engedélyezett állapotban vannak, esetenként azonban szükség lehet (például bizonyos triggerekkel megalósított megszorítások átmeneti felfüggesztése érdekében) a triggerek letiltására (lásd 10.5. példa).

### *Trigger letiltása, illetve engedélyezése*

```
ALTER TRIGGER TriggerNeve [DISABLE | ENABLE];
```

### *Egy tábla összes triggerének letiltása, illetve engedélyezése*

```
ALTER TRIGGER TriggerNeve [DISABLE | ENABLE] ALL TRIGGERS;
```

ahol a DISABLE jelenti a letiltást és az ENABLE az engedélyezést.

## 10.4. példa

Hozzunk létre egy olyan utasításszintű BEFORE triggert, amely megakadályozza a munkaidőn kívüli adatmanipulációkat az emp táblán.

### Megoldás

```
CREATE OR REPLACE TRIGGER NeNyúljHozzá
BEFORE DELETE OR INSERT OR UPDATE ON emp
BEGIN
  IF TO_CHAR(sysdate, 'HH24:MI') NOT BETWEEN '08:00' AND '16:30'
  THEN
    RAISE_APPLICATION_ERROR(-20111,
                           'Csak munkaidőben szabad az adatbázis!');
  END IF;
END;
/
SHOW ERRORS
```

### Eredmény

A trigger létrejött.

Nincsenek hibák.

### Ellenőrzés

```
INSERT INTO emp
VALUES (1234,'KISS','CLERK',1111,'99-MÁJ-20',1200,NULL,10);
```

## Az ellenőrzés eredménye

```
INSERT INTO emp
```

```
*
```

Hiba a(z) 1. sorban:

ORA-20111: Csak munkaidőben szabad az adatbázis!

ORA-06512: a(z) "SCOTT.NENYÚLJHOZZÁ", helyen a(z) 4. sornál

ORA-04088: hiba a(z) 'SCOTT.NENYÚLJHOZZÁ' trigger futása közben

## A triggerrelő esemény meghatározása

Ha egy trigger több műveletet is elindíthat, akkor a feldolgozásnál szükség lehet az indító művelet meghatározására. Erre a célról szolgálnak a trigger törzsében, vagy az onnan hívott tárolt alprogramban egyaránt használható DELETING, INSERTING és UPDATING függvények.

### 10.5. példa

Oldjuk meg az előző feladatot oly módon, hogy azt is kiíratjuk, milyen műveletet kíséreltek meg végrehajtani munkaidőn kívül.

#### Megoldás

```
CREATE OR REPLACE TRIGGER NeNyúljHozzá
BEFORE DELETE OR INSERT OR UPDATE ON emp
BEGIN
    IF TO_CHAR(sysdate, 'HH24:MI') NOT BETWEEN '08:00' AND '16:30'
    THEN
        IF DELETING THEN
            RAISE_APPLICATION_ERROR(-20211,
                'Csak munkaidőben szabad adatot törölni!');
        ELSIF INSERTING THEN
            RAISE_APPLICATION_ERROR(-20212,
                'Csak munkaidőben szabad adatot bevinni!');
        ELSE
            RAISE_APPLICATION_ERROR(-20213,
                'Csak munkaidőben szabad adatot módosítani!');
        END IF;
    END IF;
END;
/
SHOW ERRORS
```

#### Eredmény

A trigger létrejött.

Nincsenek hibák.

### **Ellenőrzés**

```
DELETE FROM emp
WHERE UPPER(ename) = UPPER('Smith');
```

### **Az ellenőrzés eredménye**

```
DELETE FROM emp
```

\*

Hiba a(z) 1. sorban:

ORA-20211: Csak munkaidőben szabad adatot törölni!

ORA-06512: a(z) "SCOTT.NENYÜLJHOZZÁ", helyen a(z) 5. sornál

ORA-04088: hiba a(z) 'SCOTT.NENYÜLJHOZZÁ' trigger futása közben

## **10.6. példa**

Fogalmazzuk át a 10.4. példát a következőképpen:

Hozzunk létre olyan triggereket, melyek segítségével megakadályozzuk (letiltjuk) a munkaidőn kívüli beavatkozásokat az emp táblán, továbbá egy Feljegyzés nevű adattábla feljegyezzük a munkaidőn kívül megkísérelt, és a munkaidőben sikeresen elvégzett (engedélyezett) beavatkozások időpontját, jellegét ("Törés", "Beszúrás", "Módosítás") és minősítését ("Letiltott", "Engedélyezett").

### **Megoldás**

#### **1. lépés (A Feljegyzés adattábla létrehozása)**

```
DROP TABLE Feljegyzés;
CREATE TABLE Feljegyzés
(DátumIdő      VARCHAR2(22),
 Beavatkozás   VARCHAR2(12),
 Jelleg        VARCHAR2(13));
```

### **Eredmény**

```
DROP TABLE Feljegyzés
```

\*

Hiba a(z) 1. sorban:

ORA-00942: a tábla vagy a nézet nem létezik

A tábla létrejött.



#### **Megjegyzés**

A fenti "Hiba..." valójában nem hiba, egyszerűen csak azt jelzi, hogy a tábla még nem létezett.

#### **2. lépés (Segéd-nézettábla létrehozása)**

```
CREATE OR REPLACE VIEW Dolgozónézet
AS SELECT * FROM emp;
```

**Eredmény**

A nézet létrejött.

**3. lépés (A tiltott beavatkozást figyelő és feljegyző trigger törlése és létrehozása)**

DROP TRIGGER NeNyúljHozzá;

```

CREATE OR REPLACE TRIGGER NeNyúljHozzá
INSTEAD OF DELETE OR INSERT OR UPDATE ON DolgozóNézet
DECLARE
    TörlésPróba      EXCEPTION;
    BeszúrásPróba   EXCEPTION;
    MódosításPróba  EXCEPTION;
    xempno          emp.empno%TYPE;
    xename          emp.ename%TYPE;
    xjob            emp.job%TYPE;
    xmgr             emp.mgr%TYPE;
    xhiredate       emp.hiredate%TYPE;
    xsal             emp.sal%TYPE;
    xcomm            emp.comm%TYPE;
    xdeptno         emp.deptno%TYPE;
BEGIN
    DBMS_OUTPUT.PUT_LINE('==> A NeNyúljHozzá trigger aktiv...');

    IF TO_CHAR(sysdate, 'HH24:MI') NOT BETWEEN '08:00' AND '16:30'
    THEN
        IF DELETING THEN
            INSERT INTO Feljegyzés
                VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
                        'Törlés', 'Letiltott');
            RAISE TörlésPróba;
        ELSIF INSERTING THEN
            INSERT INTO Feljegyzés
                VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
                        'Beszúrás', 'Letiltott');
            RAISE BeszúrásPróba;
        ELSE
            INSERT INTO Feljegyzés
                VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
                        'Módosítás', 'Letiltott');
            RAISE MódosításPróba;
        END IF;
    ELSE
        IF DELETING THEN
            DELETE FROM emp
            WHERE UPPER(ename) = UPPER(:OLD.ename); -- VIGYÁZAT: NEM :NEW!!!
        END IF;
    END IF;
END;

```

```

ELSIF INSERTING THEN
  INSERT INTO emp
    VALUES (:NEW.empno,      :NEW.ename, :NEW.job,   :NEW.mgr,
           :NEW.hiredate, :NEW.sal,   :NEW.comm,  :NEW.deptno);
ELSIF UPDATING THEN
  IF :NEW.empno     IS NULL
    THEN xempno    := :OLD.empno;
    ELSE xempno    := :NEW.empno;    END IF;
  IF :NEW.ename     IS NULL
    THEN xename     := :OLD.ename;
    ELSE xename     := :NEW.ename;    END IF;
  IF :NEW.job       IS NULL
    THEN xjob       := :OLD.job;
    ELSE xjob       := :NEW.job;    END IF;
  IF :NEW.mgr       IS NULL
    THEN xmgr       := :OLD.mgr;
    ELSE xmgr       := :NEW.mgr;    END IF;
  IF :NEW.hiredate IS NULL
    THEN xhiredate  := :OLD.hiredate;
    ELSE xhiredate  := :NEW.hiredate; END IF;
  IF :NEW.sal       IS NULL
    THEN xsal       := :OLD.sal;
    ELSE xsal       := :NEW.sal;    END IF;
  IF :NEW.comm      IS NULL
    THEN xcomm      := :OLD.comm;
    ELSE xcomm      := :NEW.comm;    END IF;
  IF :NEW.deptno    IS NULL
    THEN xdeptno   := :OLD.deptno;
    ELSE xdeptno   := :NEW.deptno;  END IF;
  UPDATE emp
    SET empno      = xempno,
        ename       = xename, -- Mivel névvel azonosítunk, ez nem aktív!
        job        = xjob,
        mgr        = xmgr,
        hiredate   = xhiredate,
        sal         = xsal,
        comm        = xcomm,
        deptno     = xdeptno
      WHERE UPPER(ename) = UPPER(xename); -- Névvel azonosítunk!
    END IF;
  END IF;
EXCEPTION
  WHEN TörlésPróba THEN
    DBMS_OUTPUT.PUT_LINE('>> Csak munkaidőben szabad adatot törölni!');


```

```

WHEN BeszúrásPróba THEN
    DBMS_OUTPUT.PUT_LINE('>> Csak munkaidőben szabad adatot bevinni!');
WHEN MódosításPróba THEN
    DBMS_OUTPUT.PUT_LINE('>> Csak munkaidőben szabad adatot módosítani!');
END;
/
SHOW ERRORS

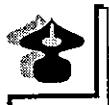
```

### Eredmény

```

DROP TRIGGER NeNyúljHozzá
*
Hiba a(z) 1. sorban:
ORA-04080: 'NENYÚLJHOZZÁ' trigger nem létezik
A trigger létrejött.
Nincsenek hibák.

```



#### Megjegyzés

- A fenti „Hiba...” ezúttal sem hiba (lásd az előző megjegyzést).
- Mivel INSTEAD OF triggert használtunk, ezért gondoskodni kellett a triggert indító DML-utasítások helyettesítéséről a triggerben. Szintén a trigger INSTEAD OF jellegréből következik, hogy egyszer csak a trigger, valamint az egyes DML-utasításokat is a munkatáblán értelmezett nézettáblára kell definiálni.
- Ezúttal a hibaüzenetet nem a RAISE\_APPLICATION\_ERROR függvénytel, hanem a DBMS\_OUTPUT.PUT\_LINE függvénytel írottuk ki. Ennek az oka, hogy a RAISE\_APPLICATION\_ERROR függvény minden, a trigger tüzelését követő DML-műveletet visszagörödít, és nem csupán azt, amelyikre a trigger érzékeny. Ennek az lett volna az eredménye, hogy a Feljegyzés táblába történő feljegyzéseket is kitörölte volna, márpédig e megoldás célja éppen a feljegyzések archiválása volt. Megjegyezzük, hogy látszólag a DBMS\_OUTPUT.PUT\_LINE függvény alkalmas volt a hibaüzenet kiíratására, ám ez nem akadályozza meg az INSTEAD OF trigger által kiváltott eredeti DML-utasítás üzenetének kiíratását (melyet viszont a RAISE\_APPLICATION\_ERROR függvény megakadályoz). Például a trigger által megakadályozott törlés esetén kiírta volna az „1 sor törölve.” üzenetet, amelyik persze hamis, hiszen a trigger helyes működése megakadályozta a törlést. Annak érdekében, hogy elkerüljük e hamis hibaüzenet kiírását, kiadtuk a SET feedback OFF utasítást.

### 4. lépés (Az engedélyezett beavatkozást utólag feljegyző trigger törlése és létrehozása)

```
DROP TRIGGER Feljegyez;
```

```

CREATE OR REPLACE TRIGGER Feljegyez
AFTER DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('==> A Feljegyez trigger aktív...');

```

```

IF DELETING THEN
  INSERT INTO Feljegyzés
    VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
            'Törlés', 'Engedélyezett');
  DBMS_OUTPUT.PUT_LINE('>> Sikeres adattörlés!');

ELSIF INSERTING THEN
  INSERT INTO Feljegyzés
    VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
            'Beszúrás', 'Engedélyezett');
  DBMS_OUTPUT.PUT_LINE('>> Sikeres adatbeszúrás!');

ELSE -- (ekvivalens a ELSIF UPDATING THEN utasításrésszel)
  INSERT INTO Feljegyzés
    VALUES (TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'),
            'Módosítás', 'Engedélyezett');
  DBMS_OUTPUT.PUT_LINE('>> Sikeres adatmódosítás!');

END IF;
END;
/
SHOW ERRORS

```

### Eredmény

```

DROP TRIGGER Feljegyez
*
Hiba a(z) 1. sorban:
ORA-04080: 'FELJEGYEZ' trigger nem létezik
A trigger létrejött.
Nincsenek hibák.

```

### 5. lépés (Ellenőrzés)

```

--tesztelő program eleje
SET serveroutput ON
SET feedback OFF

```

PROMPT Törlési próba:

```

DELETE FROM DolgozóNézet
  WHERE UPPER(ename) = UPPER('Smith');

```

PROMPT Beszúrási próba:

```

INSERT INTO DolgozóNézet
  VALUES (1234,'KISS','CLERK',7499,'99-MÁJ-20',1200,NULL,10);

```

PROMPT Módosítási próba:

```

UPDATE DolgozóNézet
  SET sal = sal + 1000
  WHERE UPPER(ename) = UPPER('Blake');

```

SET feedback 5

PROMPT A beavatkozások listázása:

SELECT \* FROM Feljegyzés;

PROMPT A módosított emp tábla listázása:

SET numwidth 5

SELECT \* FROM emp;

SET numwidth 10

--tesztelő program vége

### Az ellenőrzés eredménye (az engedélyezett időszakban)

Törlési próba:

==> A NeNyúljHozzá trigger aktív...

==> A Feljegyez trigger aktív...

>> Sikeres adattörölés!

Beszúrási próba:

==> A NeNyúljHozzá trigger aktív...

==> A Feljegyez trigger aktív...

>> Sikeres adatbeszúrás!

Módosítási próba:

==> A NeNyúljHozzá trigger aktív...

==> A Feljegyez trigger aktív...

>> Sikeres adatmódosítás!

### Az ellenőrzés eredménye (a tiltott időszakban)

Törlési próba:

==> A NeNyúljHozzá trigger aktív...

>> Csak munkaidőben szabad adatot törölni!

Beszúrási próba:

==> A NeNyúljHozzá trigger aktív...

>> Csak munkaidőben szabad adatot bevinni!

Módosítási próba:

==> A NeNyúljHozzá trigger aktív...

>> Csak munkaidőben szabad adatot módosítani!

A beavatkozások listázása:

DÁTUMIDŐ	BEAVATKOZÁS	JELLEG
2004.12.24, 14:05:07	Törlés	Engedélyezett
2004.12.24, 14:05:07	Beszúrás	Engedélyezett
2004.12.24, 14:05:08	Módosítás	Engedélyezett
2004.12.24, 18:09:51	Törlés	Letiltott
2004.12.24, 18:09:51	Beszúrás	Letiltott
2004.12.24, 18:09:51	Módosítás	Letiltott

6 sor kijelölve.

A módosított emp tábla listázása:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	3850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
1234	KISS	CLERK	7499	99-MÁJ-20	1200		10

14 sor kijelölve.



**Megjegyzés**

- Szegény Smith törlöve...
- Figyeljünk fel arra, hogy a fenti Feljegyez nevű, az emp táblára definiált AFTER trigger korrekt módon működött a hozzá rendelt DML-utasításokra, bár az azokat tartalmazó tesztelőutasítások nem közvetlenül az emp táblán, hanem közvetett módon, a Dolgozónézet nevű nézettáblán keresztül végezték a beavatkozásokat. Tehát a Feljegyez trigger tulajdonképpen nem a tesztelőutasítások, hanem a megfelelő (az emp táblára ható) DML-utasításokat tartalmazó NeNyúljHozzá trigger indította el!
- A fentiek tesztelésénél ügyeljünk arra, hogy az emp tábla újragenerálásakor (törlésekor) a Feljegyez trigger törlődik, ezért azt szintén újra kell generálni. (Nem mondhatunk futtatást, hiszen az csak a triggerrelő esemény hatására következik be.)
- Hasonlóképpen, ha a tesztelés során töröljük a Dolgozónézet nézettáblát, akkor az erre hivatkozó NeNyúljHozzá trigger törlődik.

**6. lépés (Takarítás: adat-visszaállítás, létrehozott triggerek, adattáblák és nézetek törlése)**

ROLLBACK;

DROP TRIGGER NeNyúljHozzá;

DROP TRIGGER Feljegyez;

DROP TABLE Feljegyzés;

DROP VIEW Dolgozónézet;

```
SET numwidth 5
SELECT * FROM emp;
SET numwidth 10
```

### Eredmény

A visszaállítás befejeződött.  
 A trigger eldobva.  
 A trigger eldobva.  
 A tábla eldobva.  
 A nézet eldobva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.



#### Megjegyzés

Fontos a fenti törlési sorrend betartása. Ha a NeNyúljHozzá trigger törlése előtt töröljük a Dolgozónézet nézettáblát, akkor az erre hivatkozó trigger automatikus törlődése miatt azt már nem tudjuk törölni. Ez még nem olyan nagy baj. Ennél súlyosabb az, ha az adat-visszaállítás (ROLLBACK) előtt töröljük azt a nézettáblát (Dolgozónézet), melyen keresztül a beavatkozások történtek. Ekkor tudniüllik nem lehet az adat-visszaállítást elvégezni.

### 10.7. példa

Írjon triggeret, mely megakadályozza az elnökre (president) vonatkozó törlő, beszúró és adatmódosító utasítások működését.

### Megoldás (BEFORE-WHEN trigger alkalmazása)

```

CREATE OR REPLACE TRIGGER KivéveAzElnök
BEFORE DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
WHEN (UPPER(OLD.job) = 'PRESIDENT' OR
      UPPER(NEW.job) = 'PRESIDENT')
BEGIN
  IF DELETING THEN
    RAISE_APPLICATION_ERROR(-20001,'>> Az elnököt nem lehet törölni...');
  ELSIF INSERTING THEN
    RAISE_APPLICATION_ERROR(-20002,'>> Elnököt nem lehet beszúrni...');
  ELSIF UPDATING THEN
    RAISE_APPLICATION_ERROR(-20003,'>> Elnök adatok nem
módosíthatók...');
  END IF;
END;
/
SHOW ERRORS

```

### Eredmény

A trigger létrejött.

Nincsenek hibák.

### Ellenőrzés

```

-- A tesztelő szkript program eleje
SET serveroutput ON
PROMPT Hivatkozási megszorítás felfüggesztése:
ALTER TABLE emp
  DISABLE CONSTRAINT EMP_SELF_KEY;

UPDATE emp
  SET sal = sal - 555
 WHERE deptno = 10;

INSERT INTO emp
  VALUES (1234,'KISS','president',NULL,sysdate,6000,NULL,10);

DELETE FROM emp
 WHERE deptno = 10;

PROMPT Hivatkozási megszorítás engedélyezése:
ALTER TABLE emp
  ENABLE CONSTRAINT EMP_SELF_KEY;
SET numwidth 5

```

```
SELECT * FROM emp;
SET numwidth 10
-- A tesztelő szkript program vége
```

**Eredmény**

Hivatkozási megszorítás felfüggesztése:

A tábla módosítva.

```
WHERE deptno = 10
```

```
*
```

Hiba a(z) 3. sorban:

ORA-20003: >> Elnök adatok nem módosíthatók...

ORA-06512: a(z) "SCOTT.KIVÉVEAZELNÖK", helyen a(z) 7. sornál

ORA-04088: hiba a(z) 'SCOTT.KIVÉVEAZELNÖK' trigger futása közben

```
VALUES (1234,'KISS','president',NULL,sysdate,6000,NULL,10)
```

```
*
```

Hiba a(z) 2. sorban:

ORA-20002: >> Elnököt nem lehet beszúrni...

ORA-06512: a(z) "SCOTT.KIVÉVEAZELNÖK", helyen a(z) 5. sornál

ORA-04088: hiba a(z) 'SCOTT.KIVÉVEAZELNÖK' trigger futása közben

```
WHERE deptno = 10
```

```
*
```

Hiba a(z) 2. sorban:

ORA-20001: >> Az elnököt nem lehet törölni...

ORA-06512: a(z) "SCOTT.KIVÉVEAZELNÖK", helyen a(z) 3. sornál

ORA-04088: hiba a(z) 'SCOTT.KIVÉVEAZELNÖK' trigger futása közben

Hivatkozási megszorítás engedélyezése:

A tábla módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20

7369	SMITH	CLERK	7902	80-DEC-17	800	20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000	20
7876	ADAMS	CLERK	7788	83-JAN-12	1100	20
7934	MILLER	CLERK	7782	82-JAN-23	1300	10

14 sor kijelölve.



### Megjegyzés

- Ahhoz, hogy a fenti törlési utasítás esetén a trigger hatását be tudjuk mutatni, előtte törölnünk kellett az mgr oszlop fölötti hivatkozási megszorítást a következő utasítással:

```
ALTER TABLE emp
    DISABLE CONSTRAINT EMP_SELF_KEY;
```

A tesztelés után természetesen vissza kellett állítanunk e megszorítást az alábbi utasítással:

```
ALTER TABLE emp
    ENABLE CONSTRAINT EMP_SELF_KEY;
```

- Feltehető a kérdés, hogyan lehetne a KivéveAzElnök trigger tárolása esetén mégis csak módosítani az elnök adatait. Ez a trigger működésének felfüggesztésével végezhető el, az

```
ALTER TRIGGER KivéveAzElnök DISABLE;
```

utasítással azok révén, akiknek erre jogosultságuk van, majd a megfelelő beavatkozás után az

```
ALTER TRIGGER KivéveAzElnök ENABLE;
```

utasítással újra zárolhatók az elnök adatai (lásd Triggerek letiltása és engedélyezése c. pont).

## Feladatok

### 10.1. feladat

Írjon triggert (és ellenőrizze is működését), amely az alábbi feltételek teljesülése esetén engedélyezi az adatbevitelt, míg e feltételek nem teljesülése esetén az adatbevitelt megtagadja. minden feltétel ellenőrzését külön alprogrammal valósítsa meg. A trigger tartalmazzon kivételkezelést, és a hibaüzenetek is ebben szerepeljenek. A feltételek az alábbiak:

- egy dolgozóhoz csak már tárolt dolgozó azonosítója adható meg főnök-kódként,
- csak olyan munkakör adható meg, amely már szerepel,
- új dolgozó az azonos munkakörű társainál csak alacsonyabb fizetést kaphat.

### 10.2. feladat

Írjon triggert (és ellenőrizze is működését), amely az alábbi feltételek teljesülése esetén engedélyezi az adatbevitelt, míg e feltételek nem teljesülése esetén az adatbevitelt megtagadja. Min-

den feltétel ellenőrzését külön alprogrammal valósítsa meg, és ezek tartalmazzák a kivételekezelést, valamint a hibaüzenet kiírását is. A feltételek az alábbiak:

- egy dolgozóhoz csak már tárolt dolgozó azonosítója adható meg főnök-kódként,
- csak olyan részlegazonosító adható meg, amely már szerepel,
- új dolgozó az azonos részlegbeli társainál csak alacsonyabb fizetést kaphat.

### 10.3. feladat

Írjon triggeret (és ellenőrizze is működését), amely megakadályozza, hogy a felhasználó olyan dolgozót

- töröljön,
- beszűrjon, vagy
- lásson el fizetésemeléssel,

akinek havi bérre eléri vagy meghaladja részlegének átlagfizetését.

### 10.4. feladat

Írjon triggeret (és ellenőrizze is működését), amely az alábbi feltételek teljesülése esetén engedélyezi az adatbevitelt, míg e feltételek nem teljesülése esetén az adatbevitelt megtagadja. minden feltétel ellenőrzését külön alprogrammal valósítsa meg. A trigger tartalmazzon kivételekezelést, és a hibaüzenet ebben szerepeljen. A fizetésemelésre vonatkozó feltételek az alábbiak:

- az elnök (president) soha nem kaphat fizetésemelést,
- akinek fizetése eléri a 3000 USD-t, az csak a kijelölt fizetésemelés felét kapja,
- akinek egynél több közvetlen beosztottja van, az a kijelölt fizetésemelés kétszeresét kapja, feltéve, hogy fizetése nem éri el a 3000 USD-t.

### 10.5. feladat

Készítsen triggeret (és ellenőrizze is működését), amely megakadályozza olyan új dolgozó felvételét, akinek

- *nincs* kijelölt főnöke (mgr), vagy a kijelölt főnöke még nem főnök,
- a kijelölt főnöke már legalább két közvetlen beosztottal rendelkezik,
- akár a kijelölt munkaköre, akár a kijelölt részlege még nem létezik,
- a megígyert fizetése a leendő részlege átlagfizetésénél nagyobb,
- a megígyert fizetése a leendő munkaköre legkisebb fizetésénél kisebb.

A fenti feltételek mindegyikét különálló tárolt alprogrammal ellenőrizze.

# Adatbázis-adminisztrátori ismeretek

## Elméleti összefoglaló

### Jogosultság (jog, privilégium)

A korszerű adatbázis-kezelő rendszerek a felhasználói jogok összetett rendszerével (jogosultságokkal, más szóval privilégiummal) rendelkeznek a különböző adatbázis-hozzáférések és beavatkozások elvégzéséhez (bizonyos SQL-utasítások kiadásához), vagyis csak olyan tevékenységeket tudnak elvégezni, amelyhez jogosultsággal rendelkeznek. A jogosultságok kezelését az adatvezérlő nyelv végzi (Data Control Language – DCL), mely a szabványos SQL nyelv része.

A jogosultságok használatának célja tehát az adatbázisok védelme, a beavatkozások korlátozása. Eszerint léteznek

- a rendszerszintű védelmet biztosító jogosultságok (amelyek az egyes rendszerkomponensek elérését korlátozzák),
- az adatbázis védelmét biztosító jogosultságok (amelyek az adatbázishoz való hozzáférést korlátozzák),
- az objetumkezelési jogosultságok (amelyek az objektumok tartalmához való hozzáférést korlátozzák), és
- a továbbadási jogosultságok (melyek a jogosultságok más felhasználónak vagy úgynevet szerepkörnek való átadását és visszavonását korlátozzák – a szerepkör fogalmát lásd később).

Az SQL-utasítások elvégzésének korlátozása szerint beszélhetünk *rendszerjogosultságokról* és *objektumkezelési jogosultságokról*.

A jogosultságok legáltalánosabb tulajdonosa a DBA (Data Base Administrator), vagyis az adatbázis-adminisztrátor (az Oracle-rendszerben a Sys felhasználó). Az egyes felhasználók alapértelmezésben tőle kapják a jogosultságokat, amelyeket azután (ha van erre vonatkozó jogosultságuk!) tovább is adhatnak.

Általános szabályként megjegyezzük, hogy egy felhasználó, ha rendelkezik valamely objektumtípus (például tábla, nézet, tárolt eljárás) létrehozásának jogával, akkor minden arra vonatkozó (módosító és törlési) joggal is rendelkezik.

E fejezetben az adatbázis-adminisztrátori ismereteknek csupán a vázlatos bemutatása található. Ennek megfelelően az egyes utasítások (szintaktikai) megadása nem teljes, csupán arra szolgálnak, hogy szemléltessék a lehetőségeket. A jogosultsággal kapcsolatos részletes ismereteket lásd [12], [25], és [26].

## Rendszerjogosultságok

A rendszerjogosultságok biztosítják az adatbázis védelmét azáltal, hogy korlátozzák a lemezrúlethez és az adatbázishoz való hozzáférést (adatbázis-objektumok létrehozását, törlését). E jogosultságok a felhasználó által kiadható SQL-utasítások közül azokat korlátozzák, melyek révén objektumok hozhatók létre, törölhetők, illetve megváltoztatható a szerkezetük. E korlátozás érinti tehát például a CREATE, ALTER és DROP utasításokat. Ezek közül néhány:

CREATE SESSION	kapcsolódás az adatbázishoz (ez alapvetően szükséges minden felhasználó számára),
CREATE TABLE	tábla létrehozása a felhasználó saját tulajdonában,
ALTER ANY TABLE	az adatbázisban bármely tábla szerkezetének megváltoztatása,
DROP ANY TABLE	az adatbázis bármely táblájának törlése,
CREATE PROCEDURE	tárolt eljárás, függvény vagy csomag létrehozása a felhasználó saját tulajdonában,
ALTER ANY TRIGGER	az adatbázisban bármely trigger megváltoztatása,
CREATE USÉR	felhasználó létrehozása,
DROP USER	felhasználó törlése.

Végül is elmondhatjuk, hogy megfelelő rendszerjogosultságok nélkül egy felhasználó egyáltalán nem tudná használni a rendszert.

E tárgykörben a legfontosabb utasítás a GRANT, illetve a REVOKE, amelyek segítségével a jogosultságok átadhatók (továbbadhatók), illetve visszavonhatók (törölhetők).

### RENSZERJOGOSULTSÁGOK ADÁSA (GRANT)

A rendszerjogosultságot adó utasítás általános alakja

```
GRANT {rendszerjogosultság | szerepkör}
      [, {rendszerjogosultság | szerepkör}]...
TO {felhasználó | szerepkör | PUBLIC}
      [, {felhasználó | szerepkör}]...
[WITH ADMIN OPTION];
```

ahol:

rendszerjogosultság	a megadni kívánt rendszerjogosultság,
felhasználó	a jogosultságot kapó felhasználó azonosítója (részletesen lásd kézőbb),
szerepkör	lásd később,
T0	a felhasználó vagy szerepkör, aki a jogosultságot kapja,

PUBLIC minden (jelenlegi és jövőbeni) felhasználót kijelöl,  
 WITH ADMIN OPTION megadja a felhasználónak a jogosultság továbbadásának jogát.

### **RENSZERJOGOSULTSÁGOK VISSZAVONÁSA (REVOKE)**

A rendszerjogosultságot visszavonó utasítás általános alakja

```
REVOKE {rendszerjogosultság | szerepkör}
[, {rendszerjogosultság | szerepkör}]...
FROM {felhasználó | szerepkör | PUBLIC}
[, {felhasználó | szerepkör}]...;
```

ahol az egyes fogalmak jelentését lásd a GRANT utasításnál.

## **Objektumkezelési jogosultságok**

Az objektumkezelési jogosultság olyan SQL-műveletek végrehajtását korlátozza, melyek megnevezett adatbázis-objektumok (például táblák, nézetek, tárolt eljárások) használatára, lekérdezére és módosítására vonatkoznak. minden objektumhoz a jogosultságok meghatározott csoportja tartozik.

Egy objektum tulajdonosa a saját objektumaihoz minden jogosultsággal rendelkezik, és ezekre vonatkozóan adhat bármely jogosultságot más (vagy akár az összes) felhasználó számára.

Objektumkezelési jogosultságok többek között az alábbiak lehetnek:

- adott tábla, vagy nézet adatainak (sor, oszlop) megtekintése (SELECT),
- új sorok hozzáadása adott táblához (INSERT),
- adott tábla sorainak módosítása (UPDATE),
- sorok törlése (DELETE) adott táblából,
- adott tárolt program végrehajtása (EXECUTE),
- adott objektum újraredefiniálása (ALTER),
- adott könyvtár állományainak megtekintése (READ),
- adott táblára hivatkozó megszorítás létrehozása (REFERENCE),
- index létrehozása adott táblára (INDEX).

### **OBJEKTUMKEZELÉSI JOGOSULTSÁG ADÁSA (GRANT)**

```
GRANT {ObjektumkezelésiJog | ALL} [(oszlopok)]
ON objektum
TO {felhasználó | szerepkör | PUBLIC}
[, {felhasználó | szerepkör}]...
[WITH GRANT OPTION];
```

ahol:

ObjektumkezelésiJog	a megadni kívánt objektumkezelési jogosultság,
ALL	az összes objektumkezelési jogosultság,

<i>oszlopok</i>	azoknak a tábláknak vagy nézeteknek (vesszővel elválasztott) oszlopai, amelyre a jogosultság vonatkozik,
<i>ON objektum</i>	az objektum, amelyre a jogosultság vonatkozik,
<i>T0</i>	a felhasználó vagy szerepkör, aki a jogosultságot kapja,
<i>felhasználó</i>	a jogosultságot kapó felhasználó neve (részletesen lásd később), lásd később,
<i>szerepkör</i>	
<i>PUBLIC</i>	minden (jelenlegi és jövőbeni) felhasználót kijelöl,
<i>WITH GRANT OPTION</i>	magadja a felhasználónak a jogosultság továbbadásának jogát (szerepkör esetén nem megengedett, mivel a szerepkör jogosultságait a hozzárendelt felhasználók automatikusan megkapják).

### ***OBJEKTUMKEZELÉSI JOGOSULTSÁG VISSZAVONÁSA (REVOKE)***

```
REVOKE {Objektumkezelésijog | ALL}
  ON objektum
  FROM {felhasználó | szerepkör | PUBLIC}
  [, {felhasználó | szerepkör}]...;
```

ahol a fogalmak jelentése a fentiekkel azonos.

### **Jogosultságok lekérdezése**

A jogosultságokat is nyilvántartja a rendszer az adatszótár nézeteiben. Ezekből lekérdezhetők a szerepkörök és felhasználók jogosultságai és azok tulajdonságai.

*A jogosultságra vonatkozó néhány nézet*

<i>role_sys_privs</i>	szerepköröknek adott rendszerjogosultságok,
<i>role_tab_privs</i>	szerepköröknek adott objektumkezelési jogosultságok,
<i>user_sys_privs</i>	a felhasználó számára elérhető jogosultságok,
<i>user_role_privs</i>	a felhasználóhoz rendelt szerepkörök felsorolása,
<i>user_tab_privs_made</i>	a felhasználó által a saját objektumára más felhasználónak adott objektumkezelési jogosultságok,
<i>user_tab_privs_recd</i>	a felhasználó részére adott objektumkezelési jogosultságok,
<i>user_col_privs_made</i>	a felhasználó által a saját objektumainak (tábla, nézet) oszlopaira más felhasználónak adott objektumkezelési jogosultságok,
<i>user_col_privs_recd</i>	a felhasználó részére adott objektumoszlopokra (tábla, nézet) objektumkezelési jogosultságok.

## **Táblaterület**

Az Oracle objektumai (táblák adatai, a jogosultsági adatok, tárolt programok) táblaterületeken tárolódnak. A táblaterületek olyan fizikai fájlok, amelynek belső szerkezetét az Oracle alakítja

ki és értelmezi (tehát az operációs rendszer számára ezek szerkezettel nem rendelkező adatfájlok). Az Oracle a rendszeradatok számára a SYSTEM nevű táblaterületet használja, a felhasználók pedig vagy az általuk létrehozott területeket, vagy az Oracle által előkészített valamelyik táblaterületet (például a users táblaterületet), amelyeket az Oracle már megfelelő indexeléssel és optimalizálásokkal látott el.

## Táblaterület létrehozása

Táblaterület a CREATE TABLESPACE jogosultság birtokában a következő utasítással hozható létre:

```
CREATE TABLESPACE táblaterület
  DATAFILE 'állománynév'
  [SIZE méret [K | M] [REUSE] ];
```

ahol:

táblaterület	a létrehozandó táblaterület neve,
'állománynév'	a táblaterületet tároló fizikai fájl neve útvonalmegadással együtt,
SIZE méret	a táblaterület mérete (egész szám), amely a K és az M hiánya esetén a méret bájtból értendő (megadása kötelező, ha az adott fizikai fájl még nincs lefoglalva),
K	a táblaterület mérete kilobájtból,
M	a táblaterület mérete megabájtból,
REUSE	táblaterület létezése esetén annak törlését és létrehozását eredményezi.

## Táblaterületek lekérdezése

A táblaterületeket is nyilvántartja a rendszer az adatszótár nézeteiben. Ezekből lekérdezhetők a táblaterületek tulajdonosai, helyük, méretük és egyéb tulajdonságaik.

### A táblaterületre vonatkozó néhány nézet

dba_tablespaces	az adatbázis táblaterületeinek nézete,
user_tablespaces	felhasználói táblaterület nézete,
dba_data_files	táblaterületek tárolásának nézete.

## Tábla létrehozása adott táblaterületen

Tábla a CREATE TABLE vagy a CREATE ANY TABLE jogosultság birtokában hozható létre. Alapértelmezésben egy felhasználó által létrehozott tábla a felhasználóhoz rendelt táblaterületen történik (lásd alább Felhasználó létrehozása c. pont), de ha a felhasználó rendelkezik a CREATE ANY TABLE jogosultsággal, akkor más táblaterületen is létrehozhat táblát. Az adott táblaterületen történő táblalétrehozási utasítás vázlatos alakja:

```
CREATE TABLE táblanév
  ...
  TABLESPACE táblaterület;
```

ahol zárójelben (...) a táblalétrehozó utasítás korábban már bemutatott része szerepel (lásd 5. fejezet).

## Táblaterület módosítása

Ha egy táblaterület tartalmát szeretnénk átmozgatni egy másik tárterületre (például adatmennős céljából), akkor először offline állapotba kell hozni, elvégezni a fájlmásolást, majd a további használat érdekében ismét online állapotba kell hozni. A táblaterület módosítása az ALTER TABLESPACE jogosultság birtokában lehetséges az

```
ALTER TABLESPACE TáblaterületNeve {ONLINE | OFFLINE};
```

utasítás segítségével.

## Táblaterület törlése

Egy táblaterület törlése annak offline állapotba állítása után lehetséges (lásd előző pont) a DROP TABLESPACE jogosultság birtokában a

```
DROP TABLESPACE TáblaterületNeve
  [INCLUDING CONTENTS];
```

utasítás segítségével, ahol az INCLUDING CONTENTS opció lehetővé teszi olyan adatterület törlését is, amelyen adatok vannak. Enélkül csak üres táblaterület törölhető.

## Felhasználó (USER)

A felhasználót elsősorban az ő létrehozó CREATE USER utasítás (lásd alább), valamint a számrara különböző jogosultságokat adó GRANT, illetve azokat visszavonó REVOKE utasítások együttesen definiálják (lásd feljebb).

Ennek értelmében a felhasználó rendelkezik felhasználói névvel és belépési jelszóval, amellyel csatlakozni tud az adatbázishoz. Hozzáférési lehetősége van egy vagy több táblaterülethez, amelyeken SQL-utasításokat futtathat. A létrehozásához kapcsolódó bizonyos tulajdon-ságait maga is megváltoztathatja (például a jelszavát vagy a hozzárendelt táblaterület méretét). Végül különböző jogosultságokkal is rendelkezik, melyek révén ténylegesen használatba is tudja venni az adatbázist. A legáltalánosabb jogosultságokkal rendelkező adatbázis-adminisztrátor (a Sys) maga is egy felhasználó.

A felhasználók (melyek lehetnek személyek, intézmények, de akár termelési eredményeket archiváló gyártó berendezések, vagy az internetről bejelentkező alkalmazói programok stb.) az

általuk birtokolt jogosultságok szerint is megkülönböztethetők, csoportosíthatók. Ha a jogosultságok valamely halmazát névvel látjuk el, akkor szerepkörről beszélünk (lásd alább). A felhasználókhöz szerepkörök (egy felhasználóhoz akár több is) rendelhetők. A szerepkörökön keresztül a felhasználók részére mind a jogosultságok odaadása (GRANT), mind a visszavonása (REVOKE) egyszerűbbé válik.

A fentieken túl egy felhasználó (amennyiben van rá jogosultsága) a működése során létrehozhat új felhasználókat, szerepköröket, adattáblákat és egyéb adatbázis-objektumokat, amelyeknek ő a tulajdonosa. Ezek fölött korlátlan jogosultsággal rendelkezik (lekérdezheti a tulajdonságaikat, módosíthatja, törölheti őket, és e jogosultságokat másra is átruházzatja).

Az adatbázis-adminisztrátor kivételével minden felhasználó törölhető. Egy felhasználót mindenkorban törölhet az a felhasználó, aki létrehozta, de másnak is lehet erre jogosultsága.

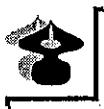
## Felhasználó létrehozása

Felhasználó a CREATE USER jogosultság birtokában hozható létre a következő utasítással:

```
CREATE USER felhasználó
  IDENTIFIED BY jelszó
  [ DEFAULT TABLESPACE Táblaterület
    [ TEMPORARY TABLESPACE IdeiglenesTáblaterület ]
    [ QUOTA {méret [K | M] | UNLIMITED} ON Táblaterület ]
    [ QUOTA {méret [K | M] | UNLIMITED} ON IdeiglenesTáblaterület ]];
```

ahol a még be nem mutatott fogalmak:

<i>felhasználó</i>	a létrehozandó felhasználó neve,
<i>jelszó</i>	kapcsolódási jelszava az adatbázishoz,
<i>Táblaterület</i>	az általa létrehozandó objektumok tárolási helye,
<i>IdeiglenesTáblaterület</i>	az általa végzendő műveletek részeredményeinek ideiglenes tárolási helye,
<i>QUOTA</i>	az engedélyezett tárterület nagyságának korlátozása a megadott táblaterületre vonatkozóan (UNLIMITED esetén nincs korlátozás).



### Megjegyzés

- Ha a felhasználó létrehozásakor nem adunk meg (nem rendelünk hozzá) táblaterületet, akkor automatikusan a SYSTEM táblaterületet kapja. Ezt azonban csak az UNLIMITED TABLESPACE jogosultság birtokában használhatja, e nélkül nem hozhat létre táblát akkor sem, ha megkaptá a CREATE TABLE jogosultságot.
- A rendszer hatékony működése érdekében ne használjuk a SYSTEM táblaterületet felhasználói célokra. Ez azt jelenti, hogy lehetőség szerint soha ne adjunk felhasználónak UNLIMITED TABLESPACE jogosultságot (szerepkörnek nem is lehet adni).
- Ha egy felhasználó létrehozásakor el szeretnénk kerülni saját táblaterület definíálását, akkor használjuk a users táblaterületet (lásd Táblaterület c. pont, és többek között 11.9. példa).

## Felhasználó törlése

Felhasználó a DROP USER jogosultság birtokában törölhető a

```
DROP USER felhasználó [CASCADE];
```

utasítással, ahol a CASCADE megadása esetén törlődnek a felhasználó által létrehozott objektumok is. Ha ilyenek léteznek, akkor a CASCADE opció használata kötelező.

## Felhasználó kapcsolódása az adatbázishoz

Egy felhasználó a

```
CONNECT felhasználó / jelszó
```

SQL\*Plus-utasítással kapcsolódhat az adatbázishoz (lásd 11.1. példa.), illetve a

```
DISCONNECT
```

SQL\*Plus-utasítással kapcsolódhat le róla. (Ez utóbbinak lényegesen kisebb a jelentősége, mivel az SQL\*Plus-környezetből való kilépéskor a lekapcsolódás automatikusan megtörténik.)

## Szerepkör (ROLE)

Szerepkörnek nevezük a jogosultságok olyan halmazát, amelyet névvel látunk el. A szerepkörmök segítségével lehetőség van a felhasználók csoportosítására, jogosultságaik egyszerű kezelésére. Például egyetlen hozzárendeléssel megadhatjuk, illetve visszavonhatjuk a benne szereplő összes jogosultságot a szerepkörhöz tartozó összes felhasználónak. A szerepkörök számtalan előnye mellett hátrány, hogy egy szerepkör nem lehet objektum tulajdonosa. Így például szerepkörbeli jogokkal nem lehet az egyes szerepkörbeli felhasználók által létrehozott objektumokat további jogosultságok adása nélkül a többi (ugyanahhoz a szerepkörhöz tartozó) felhasználó számára automatikusan hozzáférhetővé tenni. (A szerepkör tehát egy olyan fiktív felhasználó, amely csak jogosultságokkal rendelkezik, objektummal nem.)

## Szerepkör létrehozása

```
CREATE ROLE szerepkör
[IDENTIFIED BY jelszó];
```

## Szerepkör törlése

```
DROP ROLE szerepkör;
```

# Példák

## 11.1. példa

Kapcsolódjon az adatbázishoz Sys, System, majd Scott felhasználóként.

### Megoldás

```
CONNECT sys/rendszerelő AS SYSDBA
CONNECT system/gazda
CONNECT scott/tiger
```

### Eredmény

Kapcsolódva.  
Kapcsolódva.  
Kapcsolódva.



#### Megjegyzés

A fenti utasítások feltételezik, hogy az Oracle telepítésekor a Sys felhasználóhoz a rendszerelő jelszót, a System felhasználóhoz pedig a gazda jelszót rendeltük hozzá. A Scott jelszava hagyományosan a tiger.

## 11.2. példa

Írjon szkript programot, amely lekérdezi Sys vagy System felhasználóként az adatbázishoz hozzáférő felhasználók listáját.

### Megoldás

```
CONNECT sys/rendszerelő AS SYSDBA
```

vagy

```
CONNECT system/gazda
```

majd

```
SELECT Username, Default_Tablespace
  FROM dba_users
 ORDER BY Username;
```

### Eredmény

USERNAME	DEFAULT_TABLESPACE
ANONYMOUS	XDB

CTXSYS	DRSYS
.....	.....
SCOTT	SYSTEM
SH	EXAMPLE
SYS	SYSTEM
SYSTEM	SYSTEM
.....	.....

30 sor kijelölve.

### 11.3. példa

Írjon szkript programot, amely lekérdezi Sys vagy System felhasználóként a felhasználók számára elérhető táblaterületeket.

#### Megoldás

```

SET linesize 60
DESC user_tablespaces
SET linesize 400

COLUMN TABLESPACE_NAME FORMAT A15
SELECT TABLESPACE_NAME, STATUS, CONTENTS, LOGGING, FORCE_LOGGING,
       EXTENT_MANAGEMENT, ALLOCATION_TYPE, SEGMENT_SPACE_MANAGEMENT
  FROM user_tablespaces;
CLEAR COLUMNS

```

#### Eredmény

Név	Üres?	Típus
TABLESPACE_NAME	NOT NULL	VARCHAR2(30)
BLOCK_SIZE	NOT NULL	NUMBER
INITIAL_EXTENT		NUMBER
NEXT_EXTENT		NUMBER
MIN_EXTENTS	NOT NULL	NUMBER
MAX_EXTENTS		NUMBER
PCT_INCREASE		NUMBER
MIN_EXTLEN		NUMBER
STATUS		VARCHAR2(9)
CONTENTS		VARCHAR2(9)
LOGGING		VARCHAR2(9)
FORCE_LOGGING		VARCHAR2(3)
EXTENT_MANAGEMENT		VARCHAR2(10)
ALLOCATION_TYPE		VARCHAR2(9)
SEGMENT_SPACE_MANAGEMENT		VARCHAR2(6)

TABLESPACE_NAME	STATUS	CONTENTS	LOGGING	FOR EXTENT_MAN	ALLOCATIO	SEGMENT
SYSTEM	ONLINE	PERMANENT	LOGGING	NO	LOCAL	SYSTEM
UNDOTBS1	ONLINE	UNDO	LOGGING	NO	LOCAL	SYSTEM
TEMP	ONLINE	TEMPORARY	Nologging	NO	LOCAL	UNIFORM
CWMLITE	ONLINE	PERMANENT	LOGGING	NO	LOCAL	SYSTEM
DRSYS	ONLINE	PERMANENT	LOGGING	NO	LOCAL	SYSTEM
EXAMPLE	ONLINE	PERMANENT	LOGGING	NO	LOCAL	SYSTEM
INDX	ONLINE	PERMANENT	LOGGING	NO	LOCAL	SYSTEM
ODM	ONLINE	PERMANENT	LOGGING	NO	LOCAL	SYSTEM
TOOLS	ONLINE	PERMANENT	LOGGING	NO	LOCAL	SYSTEM
USERS	ONLINE	PERMANENT	LOGGING	NO	LOCAL	SYSTEM
XDB	ONLINE	PERMANENT	LOGGING	NO	LOCAL	SYSTEM

11 sor kijelölve.

## 11.4. példa

Kérdezze le Sys vagy System felhasználóként a táblaterületeket és azok állapotát.

### Megoldás

```
SELECT TABLESPACE_NAME, STATUS
  FROM dba_tablespaces;
```

### Eredmény

TABLESPACE_NAME	STATUS
SYSTEM	ONLINE
UNDOTBS1	ONLINE
TEMP	ONLINE
CWMLITE	ONLINE
DRSYS	ONLINE
EXAMPLE	ONLINE
INDX	ONLINE
ODM	ONLINE
TOOLS	ONLINE
USERS	ONLINE
XDB	ONLINE

11 sor kijelölve.

## 11.5. példa

Írjon szkript programot, amely lekérdezi Sys vagy System felhasználóként a táblaterületek tárolási helyét.

### Megoldás

```
COLUMN TABLESPACE_NAME    FORMAT A15
COLUMN FILE_NAME          FORMAT A50

SELECT TABLESPACE_NAME, FILE_NAME
  FROM dba_data_files;
CLEAR COLUMNS
```

### Eredmény

TABLESPACE_NAME	FILE_NAME
SYSTEM	C:\ORACLE\ORADATA\SAJTORA\SYSTEM01.DBF
UNDOTBS1	C:\ORACLE\ORADATA\SAJTORA\UNDOTBS01.DBF
CWMLITE	C:\ORACLE\ORADATA\SAJTORA\CWMLITE01.DBF
DRSYS	C:\ORACLE\ORADATA\SAJTORA\DRSYS01.DBF
EXAMPLE	C:\ORACLE\ORADATA\SAJTORA\EXAMPLE01.DBF
INDX	C:\ORACLE\ORADATA\SAJTORA\indx01.DBF
ODM	C:\ORACLE\ORADATA\SAJTORA\ODM01.DBF
TOOLS	C:\ORACLE\ORADATA\SAJTORA\TOOLS01.DBF
USERS	C:\ORACLE\ORADATA\SAJTORA\USERS01.DBF
XDB	C:\ORACLE\ORADATA\SAJTORA\xdb01.DBF

10 sor kijelölve.

## 11.6. példa

Írjon szkript programot, amely lekérdezi a Sys adatbázis-adminisztrátor jogosultságait.

### Megoldás

```
CONNECT sys/rendszerGazda AS SYSDBA

SET pagesize 150
SELECT *
  FROM user_sys_privs;
SET pagesize 40
```

### Eredmény

Kapcsolódva.

USERNAME	PRIVILEGE	ADM
...	.....	..
SYS	DROP USER	NO
SYS	RESUMABLE	NO
SYS	ALTER USER	NO
...	.....	..
SYS	CREATE ANY EVALUATION CONTEXT	YES
SYS	EXECUTE ANY EVALUATION CONTEXT	YES

139 sor kijelölve.

## 11.7. példa

Írjon szkript programot, mely lekérdezi a System és a Scott felhasználók jogosultságait.

### Megoldás

```

CONNECT system/gazda
PROMPT A System felhasználó jogosultságai:
SELECT *
  FROM user_sys_privs;

CONNECT scott/tiger
PROMPT A Scott felhasználó jogosultságai:
SELECT *
  FROM user_sys_privs;

```

### Eredmény

Kapcsolódva.

A System felhasználó jogosultságai:

USERNAME	PRIVILEGE	ADM
SYSTEM	UNLIMITED TABLESPACE	YES

Kapcsolódva.

A Scott felhasználó jogosultságai:

USERNAME	PRIVILEGE	ADM
SCOTT	UNLIMITED TABLESPACE	NO

## 11.8. példa

Ebben a példában szemléltetjük a felhasználó létrehozásával és jogosultságok átadásával kapcsolatos SQL-utasítások használatát. Ennek során létre fogunk hozni egy meglehetősen tág jo-

gosultsági körrel rendelkező Tudor nevű felhasználót a System felhasználó által. Ehhez először létrehoznunk egy táblaterületet, melyen a Tudor táblákat tud előállítani. (Megjegyezzük, hogy létrehozhatnánk a Tudor felhasználót a users táblaterülethez rendelve, mely egy már létező, megfelelő indexeléssel és optimalizálásokkal ellátott terület.) A továbbiakban először Tudor felhasználóként létrehozzuk a Szundi felhasználót (korlátozottabb jogosultságokkal) a users táblaterületen, végül pedig bemutatjuk a jogosultságok visszavonását, és azok továbbgyűrűzését.

### Megoldás

#### 0. lépés (A System felhasználó kapcsolódása az adatbázishoz)

```
CONNECT system/gazda
```

### Eredmény

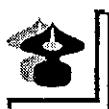
Kapcsolódva.

#### 1. lépés (Felhasználói táblaterület létrehozása)

```
CREATE TABLESPACE könyvtár
  DATAFILE 'C:\TUDOR\KONYVEK1.dat'
    SIZE 10 M REUSE;
```

### Eredmény

A táblaterület létrejött.



#### Megjegyzés

- A fenti utasítás kiadásakor feltételeztük, hogy a TUDOR alkönyvtár már létezett a C: meghajtón. Hatására a C:\TUDOR alkönyvtárban létrejött egy KONYVEK1.DAT nevű 10 MByte méretű fájl.
- E létrehozott táblaterület megszüntetése a
 

```
DROP TABLESPACE könyvtár;
```

 utasítással lehetséges.

#### 2. lépés (A táblaterületek és azok állapotának lekérdezése)

```
SELECT TABLESPACE_NAME, STATUS
  FROM dba_tablespaces;
```

### Eredmény

TABLESPACE_NAME	STATUS
SYSTEM	ONLINE
UNDOTBS1	ONLINE
TEMP	ONLINE
CWMLITE	ONLINE
DRSYS	ONLINE
EXAMPLE	ONLINE
INDX	ONLINE

ODM	ONLINE
TOOLS	ONLINE
USERS	ONLINE
XDB	ONLINE
KÖNYVTÁR	ONLINE

12 sor kijelölve.

### 3. lépés (Felhasználó létrehozása)

```
CREATE USER tudor
  IDENTIFIED BY tudor123
  DEFAULT TABLESPACE könyvtár
  TEMPORARY TABLESPACE temp
  QUOTA UNLIMITED ON könyvtár;
```

#### Eredmény

A felhasználó létrejött.



#### Megjegyzés

A Tudor felhasználóhoz a fenti utasítással hozzárendeltük általános használatra a könyvtár táblaterületet, a különböző (például rendező stb.) utasításokhoz szükséges ideiglenes tárhelyként a temp táblaterületet, és lehetőséget adtunk arra is, hogy szükség esetén a könyvtár táblaterület korlátlanul növekedhessék (QUOTA...).

### 4. lépés (Felhasználó kapcsolódása az adatbázishoz)

```
CONNECT tudor/tudor123
```

#### Eredmény

ERROR:

ORA-01045: user TUDOR lacks CREATE SESSION privilege; logon denied



#### Megjegyzés

- A fenti hibaüzenet oka, hogy a létrehozott felhasználó még semmilyen jogosultsággal nem rendelkezett. Az adatbázishoz való hozzáférés a CREATE SESSION jogosultság birtoklásával lehetséges.
- A csatlakozási hiba mellékkövetkezménye, hogy megszakadt a kapcsolat az adatbázissal, ezért azt újra létre kell hozni, azaz a System felhasználónak ismét kapcsolódnia kell az adatbázishoz.

### 5. lépés (Rendszerjogosultságok adása a felhasználónak)

```
CONNECT system/gazda
```

```
GRANT CREATE SESSION,
```

```
CREATE TABLESPACE, CREATE USER, CREATE ANY TABLE, CREATE ANY VIEW,
```

```
ALTER TABLESPACE, ALTER USER, ALTER ANY TABLE,
DROP TABLESPACE, DROP USER, DROP ANY TABLE, DROP ANY VIEW
TO tudor
WITH ADMIN OPTION;
```

**Eredmény**

Kapcsolódva.

Az engedélyezés sikeresen befejeződött.

**Megjegyzés**

A fentiekben a Tudor az adatbázishoz való hozzáférésen kívül megkapta a rendszerjogosultságok egy jelentős részét, és azok továbbadási jogát (WITH ADMIN OPTION).

**6. lépés (Felhasználó kapcsolódása az adatbázishoz)**

```
CONNECT tudor/tudor123
```

**Eredmény**

Kapcsolódva.

**7. lépés (Felhasználó jelszavának megváltoztatása)**

```
ALTER USER tudor
IDENTIFIED BY törpilla;
```

**Eredmény**

A felhasználó módosítva.

**Megjegyzés**

Ezután a Tudor már beléphet a törpilla jelszóval:

```
CONNECT tudor/törpilla
```

**Eredmény**

Kapcsolódva.

**8. lépés (Felhasználó rendszerjogosultságainak lekérdezése)**

```
COLUMN USERNAME FORMAT A15
COLUMN PRIVILEGE FORMAT A20
SELECT *
  FROM user_sys_privs;
CLEAR COLUMNS
```

**Eredmény**

USERNAME	PRIVILEGE	ADM
TUDOR	DROP USER	YES
TUDOR	ALTER USER	YES
TUDOR	CREATE USER	YES
TUDOR	DROP ANY VIEW	YES
TUDOR	CREATE SESSION	YES
TUDOR	DROP ANY TABLE	YES
TUDOR	ALTER ANY TABLE	YES
TUDOR	CREATE ANY VIEW	YES
TUDOR	DROP TABLESPACE	YES
TUDOR	ALTER TABLESPACE	YES
TUDOR	CREATE ANY TABLE	YES
TUDOR	CREATE TABLESPACE	YES

12 sor kijelölve.

**9. lépés (Felhasználó számára elérhető táblaterületek lista)**

```
COLUMN TABLESPACE_NAME FORMAT A15
SELECT TABLESPACE_NAME, STATUS, CONTENTS,
       EXTENT_MANAGEMENT, ALLOCATION_TYPE, SEGMENT_SPACE_MANAGEMENT
  FROM user tablespaces;
CLEAR COLUMNS
```

**Eredmény**

TABLESPACE_NAME	STATUS	CONTENTS	EXTENT_MAN	ALLOCATIO	SEGME
KÖNYVTÁR	ONLINE	PERMANENT	LOCAL	SYSTEM	MANUAL

**Megjegyzés**

Ha a táblaterület tartalmát szeretnénk átmozgatni egy másik tárterületre (például adatmentés céljából), akkor először offline állapotba kell hozni, elvégezni a fájlmásolást, majd a további használat érdekében ismét online állapotba kell hozni. Ennek módja:

```
ALTER TABLESPACE TáblaterületNeve {ONLINE | OFFLINE};
```

**10. lépés (Saját tábla létrehozása a táblaterületen)**

CREATE TABLE könyv

```
( cím          VARCHAR2(20),
  dátum        DATE,
  példányszám  NUMBER );
```

**Megjegyzés**

A könyv táblát létrehozhattuk volna az alábbi módon is:

```
CREATE TABLE könyv
  ( cím          VARCHAR2(20),
    dátum        DATE,
    példányszám  NUMBER )
TABLESPACE könyvtár;
```

**Eredmény**

A tábla létrejött.

**Megjegyzés**

Mivel a Tudor most már rendelkezik saját objektummal (a könyv adattáblával), ezért a törlése már nem történhetne a

```
DROP USER tudor;
```

utasítással, csak a

```
DROP USER tudor CASCADE;
```

utasítással, amelynek eredményeként a Tudor felhasználón kívül minden általa létrehozott objektum is törlődik az adatbázisból.

### **11. lépés (Saját tábla feltöltése és lekérdezése)**

```
INSERT INTO könyv ( cím, dátum, példányszám )
  VALUES ('Oracle referencia', '2001-JAN-01', 2000);
INSERT INTO könyv ( cím, dátum, példányszám )
  VALUES ('Oracle használat', '2002-MÁR-15', 3000);
INSERT INTO könyv ( cím, dátum, példányszám )
  VALUES ('Oracle példatár', sysdate, 1500);
```

```
SELECT * FROM könyv;
```

**Eredmény**

- 1 sor létrejött.
- 1 sor létrejött.
- 1 sor létrejött.

CÍM	DÁTUM	PÉLDÁNYSZÁM
Oracle referencia	01-JAN-01	2000
Oracle használat	02-MÁR-15	3000
Oracle példatár	05-JAN-07	1500

**Megjegyzés**

Egy felhasználónak, ha rendelkezik tábla-létrehozási jogosultsággal, akkor az általa létrehozott (saját) tábla használatához már minden jogosultsággal (így adatbeviteli és lekérdezési, azaz INSERT és SELECT objektumkezelési jogosultsággal is) automatikusan rendelkezik.

**12. lépés (A Tudor dept táblájának lekérdezése a System által)**

CONNECT system/gazda

SELECT \* FROM tudor.könyv;

**Eredmény**

CÍM	DÁTUM	PÉLDÁNYSZÁM
Oracle referencia	01-JAN-01	2000
Oracle használat	02-MÁR-15	3000
Oracle példatár	05-JAN-07	1500

**Megjegyzés**

A System, mint az adott adatbázisterület összes rendszer- és objektumkezelési jogosultságának tulajdonosa, képes a Tudor objektumainak kezelésére.

**13. lépés (A Scott dept táblájának lekérdezése a Tudor által)**

SELECT \* FROM scott.dept;

**Eredmény**

SELECT \* FROM scott.dept  
\*

Hiba a(z) 1. sorban:

ORA-01031: nincs megfelelő jogosultsága

**Megjegyzés**

A Tudor még nem kapott jogosultságot a Scott tábláinak lekérdezésére, mivel ez nem a saját táblája (v. ö. az előző megjegyzéssel).

**14. lépés (Objektumkezelési jogosultságok adása Tudornak)**

CONNECT scott/tiger

vagy

CONNECT system/gazda

továbbá

GRANT SELECT, INSERT, UPDATE, DELETE  
ON scott.dept  
TO tudor  
WITH GRANT OPTION;

## Eredmény

Kapcsolódva.

Az engedélyezés sikeresen befejeződött.



### Megjegyzés

- A fentiekben a lekérdezésen kívül adatbeviteli, módosítási és törlési jogosultságot, valamint mindeneknek továbbadására is kapott jogosultságot Tudor a Scott dept táblájához. E jogosultságokat a Tudor nem csupán Scott-tól, vagyis a dept tábla tulajdonosától, hanem a System-től is kaphatta, mivel a System az adatbázison belül az összes rendszer- és objektumkezelési jogosultsággal rendelkezik.
- Figyeljünk fel arra, hogy az imént kiadott GRANT utasítás után változatlanul érvényben vannak a korábbi GRANT utasítással kapott jogosultságok (amiről könnyen meggyőződhetünk a user\_sys\_privs nézettábla lekérdezésével, lásd 8. lépés, illetve alább). Általánosan elmondhatjuk, hogy az egymást követő GRANT utasítások nem írják felül a régebbieket, hanem csupán bővítenek az adott felhasználó jogosultságainak halmozatát. A korábbi jogosultságok visszavonása (törölése) kizárálag a REVOKE utasítással történhet.

## 15. lépés (A Tudor rendszerjogosultságainak lekérdezése)

```
CONNECT tudor/törpilla
COLUMN USERNAME FORMAT A15
COLUMN PRIVILEGE FORMAT A20
SELECT *
  FROM user_sys_privs;
CLEAR COLUMNS
```

## Eredmény

Kapcsolódva.

USERNAME	PRIVILEGE	ADM
TUDOR	DROP USER	YES
TUDOR	ALTER USER	YES
TUDOR	CREATE USER	YES
TUDOR	DROP ANY VIEW	YES
TUDOR	CREATE SESSION	YES
TUDOR	DROP ANY TABLE	YES
TUDOR	ALTER ANY TABLE	YES
TUDOR	CREATE ANY VIEW	YES
TUDOR	DROP TABLESPACE	YES
TUDOR	ALTER TABLESPACE	YES
TUDOR	CREATE ANY TABLE	YES
TUDOR	CREATE TABLESPACE	YES

12 sor kijelölve.

### 16. lépés (A Tudor objektumkezelési jogosultságainak lekérdezése)

```

CONNECT tudor/törpilla
SET linesize 60
DESC user_tab_privs_recd
SET linesize 400

COLUMN OWNER      FORMAT A8
COLUMN TABLE_NAME FORMAT A10
COLUMN GRANTOR    FORMAT A8
COLUMN PRIVILEGE  FORMAT A9
COLUMN GRANTABLE   FORMAT A9
COLUMN HIERARCHY  FORMAT A9
SELECT *
  FROM user_tab_privs_recd;
CLEAR COLUMNS

```

#### Eredmény

Kapcsolódva.

Név	Üres?	Típus
OWNER	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
GRANTOR	NOT NULL	VARCHAR2(30)
PRIVILEGE	NOT NULL	VARCHAR2(40)
GRANTABLE		VARCHAR2(3)
HIERARCHY		VARCHAR2(3)

OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY
SCOTT	DEPT	SCOTT	INSERT	YES	NO
SCOTT	DEPT	SCOTT	SELECT	YES	NO
SCOTT	DEPT	SCOTT	UPDATE	YES	NO
SCOTT	DEPT	SCOTT	DELETE	YES	NO

### 17. lépés (A Scott dept táblájának lekérdezése)

```

CONNECT tudor/törpilla
SELECT * FROM scott.dept;

```

#### Eredmény

Kapcsolódva.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK

20 RESEARCH	DALLAS
30 SALES	CHICAGO
40 OPERATIONS	BOSTON

### 18. lépés (Új felhasználó létrehozása a Tudor által)

Egy szkript program keretében hozzon létre a Tudor egy felhasználót (Szundi), lássa el néhány rendszerjogosultsággal, majd a Szundi (az új felhasználó) hozzon létre egy saját adattáblát, és töltse fel adattal, ezután pedig a Tudor adja tovább a Szundinak a Scott-tól a dept táblára kapott objektumkezelési jogosultságokat. Ennek ellenőrzéseképpen a Szundi kérdezze le a dept táblát.

#### Megoldás

```

CONNECT system/gazda
DROP USER szundi CASCADE;

CONNECT tudor/törpilla
CREATE USER szundi
  IDENTIFIED BY szundi123
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp
  QUOTA UNLIMITED ON users;

GRANT CREATE SESSION,
   CREATE USER, CREATE ANY TABLE, CREATE ANY VIEW,
   DROP USER
   TO szundi;

CONNECT szundi/szundi123
ALTER USER szundi
  IDENTIFIED BY hófehérke;
CONNECT szundi/hófehérke;

COLUMN USERNAME FORMAT A15
COLUMN PRIVILEGE FORMAT A20
SELECT *
  FROM user_sys_privs;
CLEAR COLUMNS

COLUMN TABLESPACE_NAME FORMAT A15
SELECT TABLESPACE_NAME, STATUS, CONTENTS,
   EXTENT_MANAGEMENT, ALLOCATION_TYPE, SEGMENT_SPACE_MANAGEMENT
  FROM user tablespaces;
CLEAR COLUMNS

```

```
CREATE TABLE könyv2
( cím          VARCHAR2(30),
  dátum        DATE,
  példányszám  NUMBER );
```

```
INSERT INTO könyv2 ( cím, dátum, példányszám )
  VALUES ('Hófehérke és a hét törpe', '1872-FEB-28', 18000);
INSERT INTO könyv2 ( cím, dátum, példányszám )
  VALUES ('Hófehérke és a két törpe', '2000-DEC-31', 300);
INSERT INTO könyv2 ( cím, dátum, példányszám )
  VALUES ('Hófehérke és a Szundi', sysdate, 1);
```

```
SELECT * FROM könyv2;
```

```
CONNECT tudor/törpilla
GRANT SELECT, INSERT, UPDATE, DELETE
  ON scott.dept
  TO szundi
  WITH GRANT OPTION;
```

```
CONNECT szundi/hófehérke
COLUMN USERNAME FORMAT A15
COLUMN PRIVILEGE FORMAT A20
SELECT *
  FROM user_sys_privs;
CLEAR COLUMNS
```

```
COLUMN OWNER      FORMAT A8
COLUMN TABLE_NAME FORMAT A10
COLUMN GRANTOR    FORMAT A8
COLUMN PRIVILEGE  FORMAT A9
COLUMN GRANTABLE   FORMAT A9
COLUMN HIERARCHY   FORMAT A9
SELECT *
  FROM user_tab_privs_recd;
CLEAR COLUMNS
```

```
SELECT * FROM scott.dept;
```

### Eredmény

Kapcsolódva.  
A felhasználó eldobva.

Kapcsolódva.

A felhasználó létrejött.

Az engedélyezés sikeresen befejeződött.

Kapcsolódva.

A felhasználó módosítva.

Kapcsolódva.

USERNAME	PRIVILEGE	ADM
SZUNDI	DROP USER	NO
SZUNDI	CREATE USER	NO
SZUNDI	CREATE SESSION	NO
SZUNDI	CREATE ANY VIEW	NO
SZUNDI	CREATE ANY TABLE	NO

TABLESPACE_NAME	STATUS	CONTENTS	EXTENT_MAN	ALLOCATIO	SEGMENT
USERS	ONLINE	PERMANENT	LOCAL	SYSTEM	AUTO

A tábla létrejött.

1 sor létrejött.

1 sor létrejött.

1 sor létrejött.

CÍM	DÁTUM	PÉLDÁNYSZÁM
Hófehérke és a hét törpe	72-FEB-28	18000
Hófehérke és a két törpe	00-DEC-31	300
Hófehérke és a Szundi	05-JAN-08	1

Kapcsolódva.

Az engedélyezés sikeresen befejeződött.

Kapcsolódva.

USERNAME	PRIVILEGE	ADM
SZUNDI	DROP USER	NO
SZUNDI	CREATE USER	NO
SZUNDI	CREATE SESSION	NO
SZUNDI	CREATE ANY VIEW	NO
SZUNDI	CREATE ANY TABLE	NO

OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY
SCOTT	DEPT	TUDOR	DELETE	YES	NO
SCOTT	DEPT	TUDOR	INSERT	YES	NO
SCOTT	DEPT	TUDOR	SELECT	YES	NO
SCOTT	DEPT	TUDOR	UPDATE	YES	NO
	DEPTNO	DNAME	LOC		
	10	ACCOUNTING	NEW YORK		
	20	RESEARCH	DALLAS		
	30	SALES	CHICAGO		
	40	OPERATIONS	BOSTON		

### 19. lépés (Objektumkezelési jogosultság visszavonása)

```

CONNECT tudor/törpilla
SELECT * FROM scott.dept;
CONNECT szundi/hófehérke
SELECT * FROM scott.dept;

CONNECT system/gazda
REVOKE SELECT
  ON scott.dept
  FROM tudor;

```

```

CONNECT tudor/törpilla
SELECT * FROM scott.dept;
CONNECT szundi/hófehérke
SELECT * FROM scott.dept;

```

### Eredmény

Kapcsolódva.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Kapcsolódva.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS

30 SALES	CHICAGO
40 OPERATIONS	BOSTON

Kapcsolódva.

A visszavonás sikeresen befejeződött.

Kapcsolódva.

```
SELECT * FROM scott.dept
```

\*

Hiba a(z) 1. sorban:

ORA-01031: nincs megfelelő jogosultsága

Kapcsolódva.

```
SELECT * FROM scott.dept
```

\*

Hiba a(z) 1. sorban:

ORA-01031: nincs megfelelő jogosultsága



### Megjegyzés

Láthatóan a Tudortól visszavont objektumkezelési jogosultság (SELECT jogosultság a Scott dept táblájára) után Tudor már nem tudta lekérdezni a Scott dept tábláját, de Szundi sem, tehát az objektumkezelési jogosultság visszavonása továbbgyűrűzik.

## 20. lépés (Rendszerjogosultság visszavonása)

```
CONNECT system/gazda
GRANT CREATE SESSION,
      CREATE TABLESPACE, CREATE USER, CREATE ANY TABLE, CREATE ANY VIEW,
      ALTER TABLESPACE, ALTER USER, ALTER ANY TABLE,
      DROP TABLESPACE,   DROP USER,   DROP ANY TABLE,   DROP ANY VIEW
TO tudor
WITH ADMIN OPTION;
```

```
CONNECT tudor/törpilla
```

```
GRANT CREATE SESSION,
      CREATE USER, CREATE ANY TABLE, CREATE ANY VIEW,
      DROP USER
TO szundi;
```

```
CONNECT tudor/törpilla
```

PROMPT Az aa1 tábla létrehozása:

```
CREATE TABLE aa1 (száml NUMBER);
```

```
CONNECT szundi/hófehérke
```

PROMPT Az aa2 tábla létrehozása:

```
CREATE TABLE aa2 (szám2 NUMBER);
```

```
CONNECT system/gazda
REVOKE CREATE ANY TABLE
  FROM tudor;
-- REVOKE CREATE ANY TABLE
--  FROM szundi;
```

```
CONNECT tudor/törpilla
PROMPT Az aa3 tábla létrehozása:
CREATE TABLE aa3 (szám3 NUMBER);
CONNECT szundi/hófehérke
PROMPT Az aa4 tábla létrehozása:
CREATE TABLE aa4 (szám4 NUMBER);
```

### Eredmény

Kapcsolódva.  
Az engedélyezés sikeresen befejeződött.

Kapcsolódva.  
Az engedélyezés sikeresen befejeződött.

Kapcsolódva.  
Az aal tábla létrehozása:  
A tábla létrejött.

Kapcsolódva.  
Az aa2 tábla létrehozása:  
A tábla létrejött.

Kapcsolódva.  
A visszavonás sikeresen befejeződött.

Kapcsolódva.  
Az aa3 tábla létrehozása:
CREATE TABLE aa3 (szám3 NUMBER)
\*
Hiba a(z) 1. sorban:
ORA-01031: nincs megfelelő jogosultsága

Kapcsolódva.  
Az aa4 tábla létrehozása:  
A tábla létrejött.

**Megjegyzés**

Láthatóan a Tudortól visszavont tábla-létrehozási rendszerjogosultság visszavonása után Tudor már nem tudott létrehozni táblát (aa3), ám Szundi továbbra is tudott táblát létrehozni (aa4), vagyis a *rendszerjogosultságok visszavonása nem gyűrűzik tovább*. Természetesen egy külön REVOKE utasítással Szunditól is visszavonhatjuk e jogosultságot (lásd Megjegyzés a fenti szkript programban).

**21. lépés (Felhasználó törlése, ez SEM gyűrűzik tovább!)**

```
CONNECT system/gazda
SELECT Username, Default_Tablespace
  FROM dba_users
 WHERE DEFAULT_TABLESPACE IN ('KÖNYVTÁR', 'USERS');

DROP USER tudor CASCADE;

SELECT Username, Default_Tablespace
  FROM dba_users
 WHERE DEFAULT_TABLESPACE IN ('KÖNYVTÁR', 'USERS');
```

**Eredmény**

Kapcsolódva.

USERNAME	DEFAULT_TABLESPACE
SZUNDI	USERS
TUDOR	KÖNYVTÁR

A felhasználó eldobva.

USERNAME	DEFAULT_TABLESPACE
SZUNDI	USERS

**Megjegyzés**

Láthatóan a Tudor törlése után a Tudor által létrehozott Szundi megmaradt, tehát a felhasználó törlése sem gyűrűzik tovább.

**11.9. példa**

A System felhasználó hozza létre a Varázsló szerepkört a tábla-létrehozási jogosultsággal, hozza létre a Hőfehérke és a GonoszMosztoha felhasználót, és rendelje hozzájuk a Varázsló szerepkört. Végül a Hőfehérke felhasználóként hozza létre a herceg táblát, töltse fel néhány adattal, majd kérdezze le a tartalmát.

**Megoldás**

```
CONNECT system/gazda
```

```
-- DROP TABLE hófehérke.herceg;  
-- DROP USER hófehérke;  
-- DROP USER gonoszmostoha;  
-- DROP ROLE varázsló;
```

```
CREATE USER hófehérke  
IDENTIFIED BY titok  
DEFAULT TABLESPACE users  
TEMPORARY TABLESPACE temp  
QUOTA UNLIMITED ON users;
```

```
CREATE USER gonoszmostoha  
IDENTIFIED BY méreg  
DEFAULT TABLESPACE users  
TEMPORARY TABLESPACE temp  
QUOTA UNLIMITED ON users;
```

```
CREATE ROLE varázsló;
```

```
GRANT CREATE SESSION, CREATE TABLE  
TO varázsló;
```

```
GRANT varázsló  
TO hófehérke, gonoszmostoha;
```

```
CONNECT hófehérke/titok  
CREATE TABLE herceg  
(Neve      VARCHAR2(20),  
 Születés   DATE,  
 Vagyona   NUMBER);
```

```
INSERT INTO herceg  
VALUES ('Szőke Alfonz', TO_DATE('1974-09-20', 'YYYY-MM-DD'), 1000);  
INSERT INTO herceg  
VALUES ('Randa Lajos', TO_DATE('1931-09-11', 'YYYY-MM-DD'), 1000000);  
INSERT INTO herceg  
VALUES ('Ifjú Sándor', TO_DATE('1985-11-22', 'YYYY-MM-DD'), 1);
```

```
SELECT *  
FROM herceg  
ORDER BY Vagyona DESC;
```

### Eredmény

Kapcsolódva.

A felhasználó létrejött.

A felhasználó létrejött.

A szerepkör létrejött.

Az engedélyezés sikeresen befejeződött.

Az engedélyezés sikeresen befejeződött.

Kapcsolódva.

A tábla létrejött.

1 sor létrejött.

1 sor létrejött.

1 sor létrejött.

NEVE	SZÜLETÉS	VAGYONA
Randa Lajos	31-SZE-11	1000000
Szőke Alfonz	74-SZE-20	1000
Ifjú Sándor	85-NOV-22	1



### Megjegyzés

A fenti szkript program ismételt futtatása esetén már szükséges az objektumok és felhasználók eldobása.

## Feladatok

### 11.1. feladat

Hozzon létre Scott felhasználóként az emp táblából egy vele azonos emp1 nevű táblát, valamint System felhasználóként egy szerepkört és egy ehhez rendelt felhasználót. Adjон a szerepkörnek olyan jogokat, hogy képes legyen az emp1 tábla szerkezetét módosítani, abban az adatokat lekérdezni, módosítani és törölni. Az új felhasználóként ellenőrizze, hogy e jogátadási tevékenységek mindegyike sikeres volt-e.

### 11.2. feladat

A System felhasználóként törölje a 11.1. feladatban létrehozott felhasználót és szerepkört.

### 11.3. feladat

Hozzon létre egy felhasználót System felhasználóként táblaterület hozzárendelése nélkül. Az új felhasználó neve a saját vezetéknévének értelemszerű rövidítése, jelszava pedig dolgozat legyen. Változtassa meg ezt a jelszót a saját utónevére. Adjon e felhasználónak az adatbázishoz

hozzáférési és nézet-létrehozási jogokat, valamint a scott.dolgozó táblájához lekérdezési jogosultágot, amit a Scott hozzon létre előbb.

#### **11.4. feladat**

A System felhasználó vonja vissza a 11.3. feladatban adott jogosultságokat, és törölje a létrehozott felhasználót.

#### **11.5. feladat**

Hozzon létre System felhasználóként a saját nevével egy felhasználót, amely hozzáfér a users táblaterülethez. Adjon az új felhasználónak alapvető rendszerjogosultságokat, majd változtassa meg a jelszavát. Ezután hozzon létre egy raktár táblát a users táblaterületen, töltse fel néhány adattal, majd listázza.

#### **11.6. feladat**

A System felhasználó vonja vissza a 11.5. feladatban adott jogosultságokat, és törölje a létrehozott felhasználót.

#### **11.7. feladat**

Készítsen szkript programot, mely először a Scott felhasználóként az emp táblából létrehozza a dolgozó táblát. Ezután System felhasználóként hozza létre a Szervező szerepkört, és lássa el tábla-létrehozási (rendszer)jogosultsággal, valamint a scott.dolgozó táblán lekérdezési és módosítási (objektumkezelési) jogosultságokkal. Hozzon létre a saját nevének felhasználót, majd rendelje hozzá a Szervező szerepkörhöz. Ellenőrizze az új felhasználóként a kapott rendszerjogosultságot egy tábla létrehozásával. Ezt töltse fel néhány adattal, majd kérdezze le. A kapott objektumkezelési jogosultságokat a scott.dolgozó tábla módosításával és lekérdezéssel ellenőrizze.

#### **11.8. feladat**

A System felhasználóként törölje a 11.7. feladatban létrehozott felhasználót és szerepkört.

## 12. FEJEZET

# Kapcsolat a külvilággal

## Elméleti összefoglaló

Egy adatbázis-kezelő program általában egy zárt világ, és különösen így van ez a biztonsági problémák elkerülésére kényesen ügyelő kliens-szerver rendszereknél, amilyen az Oracle is. Tegyük ehhez még hozzá, hogy e világ az SQL nyelvnek a procedurális PL/SQL nyelvvel való kiengészítésével algoritmikusan is teljessé vált. (Ez egyébként az SQL szabványának továbbfejlesztési iranya is, amint erre a bevezetőben már utaltunk.) Mégis előfordulhat, hogy időnként igényünk támad valamelyik másik világunkba való átjárásra.

Az átjárási igénynek két fő oka lehet. Az egyik az, hogy néha szeretnénk a fejlesztőkörnyezetből való kilépés nélkül valamilyen külső (operációs rendszerbeli, vagy akár saját fejlesztésű) programot futtatni. A másik ok az adatok archiválásával, exportálásával, importálásával, külső eszközökkel való feldolgozásával kapcsolatos.

Szükség lehet időnként arra, hogy a teljes adatbázist, egy vagy több munkaterületet, vagy néhány adattáblát archiváljunk. Felmerülhet az igénye annak, hogy egy adat- vagy nézettábla tartalmát átvigyük egy másik alkalmazásba, például a Microsoft Wordbe dokumentálás, vagy a Microsoft Excelbe későbbi feldolgozás (adatelemzés, grafikakészítés) érdekében. Végül pedig akár a normál napi munka részeként rendszeres igény lehet (különösen a több adatbázis-kezelő rendszert működtető adattárházak használatakor) bizonyos lekérdezések eredményeit (például nézettáblák tartalmát) más adatbázis-kezelő rendszernek átadni. Az alábbiakban e feladatok megoldására mutatunk be néhány módszert. (Az e területen is hasznos dinamikus SQL tárgyalása már túlmutat e könyv téma-körén. Lásd [15], [20], [22] és [26].)

## Programkapcsolat – külső programok futtatása

### Kapcsolat az operációs rendszerrel

Az operációs rendszer bármely parancssori környezetben (DOS) futó (tehát nem grafikus) programja meghívható a

HOST *ProgramNeve*

SQL-utasítással. Ha nem kívánjuk a programfutásokat látni (mert csak a végeredmény érdekel), akkor annak megjelenítését a

```
SET termout OFF
```

utasítással letilthatjuk.

Az alábbi példák a HOST utasítás néhány jellegzetes alkalmazását mutatják be.

## **A DOS-KÖRNYEZET BEHÍVÁSA SQL\*PLUS-KÖRNYEZETBŐL**

### **12.1. példa**

Hívja meg az operációs rendszer parancssori (DOS-) környezetét az SQL\*Plus-környezetből. (Visszalépés onnan az EXIT parancsal lehetséges.) Itt kérdezze le az aktuális könyvtár tartalmát laponkénti várakozással, azaz adja ki a

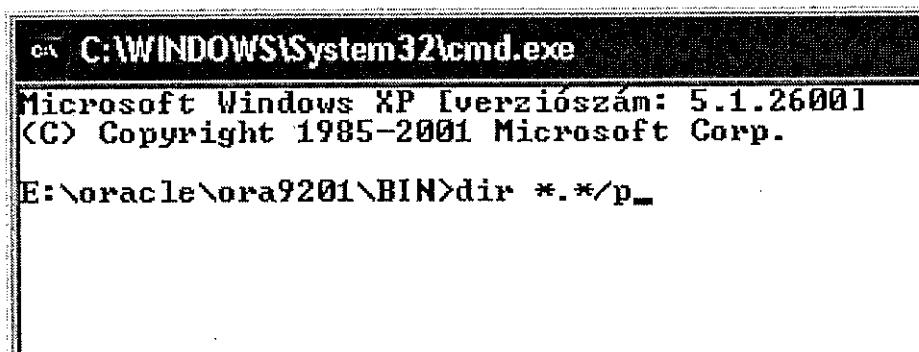
```
dir *.* /p
```

DOS-utasítást.

#### **Megoldás**

```
HOST cmd
```

A szükséges parancs kiadását a 12.1. ábra mutatja.



```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows XP [verziószám: 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

E:\oracle\ora9201\BIN>dir *.* /p
```

12.1. ábra. DOS-utasítás kiadása parancssori környezetben

#### **Eredmény**

(Lásd 12.2. ábra.)



#### **Megjegyzés**

Figyeljünk fel az ablakból leolvasható információkra! Látható, hogy az utasítást futtató gépen az Oracle-rendszer az E: jelű meghajtóra lett telepítve, az Oracle telepítési neve ora9201, és a HOST utasítás számára a lokális környezet ennek BIN nevű alkönyvtára. Egyébként ebben a könyvtárban vannak az adatbázis-kezelő rendszer programjai, ide tettük a minden betöltéskor lefutó login.sql szkript programot, és ha az ed próba utasítással végezzük a szerkesztést, akkor a keletkező próba.sql szkript program is ide kerül.

```
C:\WINDOWS\system32\cmd.exe - dir *.* /p
A meghajtóban <E> lévő kötet Logana.
A kötet sorozatszáma: 589C-288A

E:\oracle\ora9201\BIN tartalma:

2005.01.16. 15:05 <DIR>
2005.01.16. 15:05 <DIR> ..
2005.01.16. 02:02 542 aa.sql
2005.01.16. 12:26 520 ab.sql
2005.01.16. 17:53 548 ac.sql
2002.02.13. 08:23 257 024 agent.exe
2002.04.26. 17:29 33 040 agentctl.exe
2002.04.26. 17:29 28 944 agntsrvc.exe
2002.02.13. 08:23 254 464 agntsvc.exe
2002.04.26. 17:29 20 752 agntutil.exe
2002.04.28. 11:07 28 944 agtctl.exe
2005.01.16. 14:28 0 A_DIR.TXT
2002.04.26. 21:20 1 218 396 bulkmodify.exe
2002.02.08. 21:26 108 458 cmmmap000.bin
2002.02.13. 08:23 1 954 config
2002.04.27. 23:37 86 136 COREJAVA.DLL
2002.04.26. 20:00 114 960 csscan.exe
2002.04.26. 15:35 98 576 cthxhx.exe
2002.04.26. 15:35 459 024 ctxkbtc.exe
A folytatáshoz nyomjon meg egy billentyűt .. .
```

12.2. ábra. Az aktuális könyvtár tartalmának első lapja

**DOS-PARANCS KIADÁSA SQL\*PLUS-KÖRNYEZETBEN****12.2. példa**

Oldja meg az előbbi feladatot oly módon, hogy az előbbi könytárlekérdező (DOS-) utasítást SQL\*Plus-környezetben adja ki.

**Megoldás**

```
HOST dir *.* /p
```

**Eredmény**

```
C:\WINDOWS\system32\cmd.exe
A meghajtóban <E> lévő kötet Logana.
A kötet sorozatszáma: 589C-288A

E:\oracle\ora9201\BIN tartalma:

2005.01.16. 15:05 <DIR>
2005.01.16. 15:05 <DIR> ..
2005.01.16. 02:02 542 aa.sql
2005.01.16. 12:26 520 ab.sql
2005.01.16. 17:53 548 ac.sql
2002.02.13. 08:23 257 024 agent.exe
2002.04.26. 17:29 33 040 agentctl.exe
2002.04.26. 17:29 28 944 agntsrvc.exe
2002.02.13. 08:23 254 464 agntsvc.exe
2002.04.26. 17:29 20 752 agntutil.exe
2002.04.28. 11:07 28 944 agtctl.exe
2005.01.16. 14:28 0 A_DIR.TXT
2002.04.26. 21:20 1 218 396 bulkmodify.exe
2002.02.08. 21:26 108 458 cmmmap000.bin
2002.02.13. 08:23 1 954 config
2002.04.27. 23:37 86 136 COREJAVA.DLL
2002.04.26. 20:00 114 960 csscan.exe
2002.04.26. 15:35 98 576 cthxhx.exe
2002.04.26. 15:35 459 024 ctxkbtc.exe
A folytatáshoz nyomjon meg egy billentyűt .. .
```

12.3. ábra. Az aktuális könyvtár tartalmának első lapja

**DOS PARANCSKÖTEG (BATCH PROGRAM) FUTTATÁSA****12.3. példa**

Írjon SQL\*Plus szkript programot, mely az alábbiakat végzi el:

- törli a C:\MUNKA nevű munkakönyvtár backup fájljait,
- lekérdezi annak tartalmát egy munkafájlba,
- a munkafájl tartalmát konvertálja a Windows betűkészletére,
- rendezzi a munkafájl tartalmát a fájlnevek szerint,
- a rendezés eredményét az A\_DIR.TXT fájlba tölti, és végül
- törli a fentiek elvégzése során keletkezett munkafájlokat.

**Megoldás**

```
SET termout OFF
HOST del c:\munka\*.bak
HOST del c:\munka\*.*~*
HOST del c:\munka\*.*{*
HOST del c:\munka\*.*.wbk*
HOST dir c:\munka\*.* > c:\munka\AAA
HOST c:\munka\konvert.exe DW c:\munka\AAA c:\munka\BBB
HOST type c:\munka\BBB | sort.exe /+38 > c:\munka\A_DIR.TXT
HOST del c:\munka\AAA
HOST del c:\munka\BBB
SET termout ON
```

**Eredmény**

A C:\A\_DIR.TXT fájl tartalma:

c:\munka tartalma:

A kötet sorozatszáma: 7009-371E

A meghajtóban (C) lévő kötet ANALOG.

2 könyvtár 12'561'625'088 bájt szabad		
2005.01.16.	18:54	<DIR>
2005.01.16.	18:54	<DIR>
		..
	9 fájl	54'832 bájt
2005.01.16.	18:54	0 AAA
2005.01.16.	01:50	6'434 emp.txt
2005.01.16.	02:17	14'848 emp.xls
2002.06.22.	00:01	85 ER.BAT
2005.01.16.	14:18	440 Host.sql
2002.06.06.	23:20	8'032 Konvert.EXE
2005.01.16.	14:10	86 login.sql
2005.01.08.	00:29	59 próba.sql
2001.06.21.	00:00	24'848 Sort.exe



### Megjegyzés

A fentiekben az egymást követő (HOST-beli) utasításokat parancssori (DOS-) környezetben egy úgynévezett batch programmal futtathattuk volna.

A felhasznált „külső” programok közül a del, a dir és a type a Windows (DOS magjának) úgynévezett belső parancsai, a sort.exe szintén az operációs rendszeré, annak egy külső parancsfájlja, a konvert.exe pedig egy saját fejlesztésű program. Ez utóbbi használatában a DW paraméter jelzi, hogy itt egy DOS-Windows irányú karakterkonvertálás történik az AAA munkafájlból a BBB munkafájlba.

## KÜLSŐ SZÖVEGSZERKESZTŐ HASZNÁLATA

### 12.4. példa

Az SQL\*Plus-környezetből hívja meg a Windows Write.exe nevű szövegszerkesztő programját, és annak segítségével szerkessze a próba.sql nevű szkriptfájlt.

#### Megoldás

```
HOST write c:\munka\próba.sql
```

#### Eredmény

```

SET termout OFF
HOST del c:\munka\*.bak
HOST del c:\munka\*.*~
HOST del c:\munka\*.*~
HOST del c:\munka\*.wbk~
HOST dir c:\munka\*.* > c:\munka\AAA
HOST c:\munka\konvert.exe DW c:\munka\AAA
HOST type c:\munka\AAA | sort && /128 <

```

12.4. ábra. A Write szövegszerkesztő használat közben



### Megjegyzés

Amint azt a 3. melléklet Az SQL\*Plus editorának lecserélése c. pontjában bemutattuk, ha erre a szövegszerkesztő programra szeretnénk lecserélni az SQL\*Plus eredeti szövegszerkesztőjét, akkor ezt a

```
DEFINE_EDITOR="write.exe"
```

SQL\*Plus-utasítással tehetjük meg.

## Tárolt eljárások hívása az SQL\*Plus-környezetben

A tárolt eljárások SQL\*Plus-környezetbeli közvetlen használata annyiban tartozik témaünk tárgyköréhez, hogy ezeket (valamint a tárolt függvényeket) jellemzően egy (SQL\*Plus-szkríptbe ágyazott) PL/SQL blokkból, illetve tárolt alprogramokból szoktuk meghívni. E lehetőség (vagyis az EXECUTE utasítás) ismertetése (mintapéldával szemléltetve) a 10. fejezet Tárolt eljárás futtatása c. pontjában található.

## Adatkapcsolat – mentés, betöltés

Amint e fejezet bevezetőjében említettük, többféle célja is lehet az adatkapcsolati igénynek. Az előző fejezetben részleteztük a táblaterületek archiválását, amihez a leggyakrabban szükséges a táblák adatának mentése. Amennyiben az adatokat nemrég készítettük, vagy a táblák nagyon kevés változásra számíthatnak, akkor a teljes adatbázis területén elvégzett SELECT parancsokkal könnyen elérhetők lesznek a felhasználók számára.

### Teljes adatbázis-terület és egyes táblaterületek archiválása fájlba

Az 1. mellékletből tudhatjuk, hogy az Oracle E: meghajtóra telepített adatbázisai az

```
e:\oracle\oradata
```

könyvtárba kerülnek az általunk megadott néven. E névről könnyen meggyőződhetünk, ha belépünk a saját \oracle\oradata alkönyvtárunkba. Az általunk használt gépen ez a név a my ora920.

Az adatbázisnév maga is egy alkönyvtár, és ha lekérdezzük a méretét, láthatjuk, hogy elég nagy. A mi frissen telepített gépünkön például ez az érték kb. 1,3 GByte, pedig jószerint csak a rendszer saját adatait és a demonstrációs táblák adatait tartalmazza. (Ez a nagy terület persze gyakorlatilag „üres”, az Oracle csak a megfelelő működési hatékonyság miatt foglalta le, és készítette elő a saját belső adatformátumára. Meglehetősen nagy igénybevételű használat mellett is csak lassan nőne...) Ennek teljes archiválása a mai technikák mellett nem okozna problémát (egy DVD-re bőven rafér), ám az Oracle ennél kifinomultabb archiválási megoldásokat tartalmaz.

Nézzük ennek az adatbázis-alkönyvtárnak a tartalmát. Ha lefuttatjuk a

```
-- A táblaterületek tárolási helyének lekérdezése
CONNECT sys/rendszerelő AS SYSDBA
-- vagy
CONNECT system/gazda
COLUMN tablespace_name    FORMAT A15
COLUMN file_name          FORMAT A50
SELECT tablespace_name, file_name
      FROM dba_data_files;
CLEAR COLUMNS
```

szkript programot (mely az adatbázis táblaterületeit listázza, lásd 11. fejezet vagy 5. melléklet: A táblaterületek tárolási helyének lekérdezése című pont), akkor a

TABLESPACE_NAME	FILE_NAME
SYSTEM	E:\ORACLE\ORADATA\MYORA920\SYSTEM01.DBF
UNDOTBS1	E:\ORACLE\ORADATA\MYORA920\UNDOTBS01.DBF
CWMLITE	E:\ORACLE\ORADATA\MYORA920\CWMLITE01.DBF
DRSYS	E:\ORACLE\ORADATA\MYORA920\DRSYS01.DBF
EXAMPLE	E:\ORACLE\ORADATA\MYORA920\EXAMPLE01.DBF
INDX	E:\ORACLE\ORADATA\MYORA920\indx01.DBF
ODM	E:\ORACLE\ORADATA\MYORA920\ODM01.DBF
TOOLS	E:\ORACLE\ORADATA\MYORA920\TOOLS01.DBF
USERS	E:\ORACLE\ORADATA\MYORA920\USERS01.DBF
XDB	E:\ORACLE\ORADATA\MYORA920\XDB01.DBF
KÖNYVTÁR	E:\TUDOR\KONYVEK1.DAT

futási eredményt összevetve a (mi esetünkben) e:\oracle\oradata\myora920 könyvtár tartalmával, azt láthatjuk, hogy az adatbázis által elfoglalt terület jelentős részét e táblaterületek (melyekhez már egy-egy fájl tartozik) együttesen foglalják el.

Az archiválás szempontjából egy felhasználónak a legfontosabbak a saját adatait tartalmazó táblaterületek. Kérdés tehát, hogy a fenti táblaterületek közül melyek a „fontosak”.

## 12.5. példa

Tekintsük a 11. fejezet 11.8. példáját, és vizsgáljuk meg, hogy az abban létrehozott Tudor felhasználó számára mely táblaterületek archiválása szükséges (az ott szereplő C: meghajtó helyett ezúttal az E: meghajtót használva).

### Megoldás

A hivatkozott példa megoldása során a Tudor felhasználó számára létrehozott e:\tudor alkönyvtárban létrejött egy könyvtár nevű táblaterület a KONYVEK1.DAT nevű fájlban. Ennek méretét egyébként System felhasználóként mi adtuk meg a

```
CREATE TABLESPACE könyvtár
  DATAFILE 'E:\TUDOR\KONYVEK1.dat'
  SIZE 10 M REUSE;
```

utasítással, majd a Tudor felhasználót e táblaterülethez rendeltük hozzá a

```
CREATE USER tudor
  IDENTIFIED BY tudor123
  DEFAULT TABLESPACE könyvtár
  TEMPORARY TABLESPACE temp
  QUOTA UNLIMITED ON könyvtár;
```

utasítással. Nézzük meg most, hogy milyen táblaterülete is van Tudornak! Futtassuk le a

```
-- Felhasználó saját tábláinak és táblaterületeinek lekérdezése
CONNECT tudor/törpilla
COLUMN table_name      FORMAT A10
COLUMN tablespace_name FORMAT A15
SELECT table_name, tablespace_name
       FROM user_all_tables;
CLEAR COLUMNS
```

szkript programot (lásd 5. melléklet). Eredményként azt kaptuk, amit vártunk:

TABLE_NAME	TABLESPACE_NAME
KÖNYV	KÖNYVTÁR

A könyvtár nevű táblaterületről pedig A táblaterületek tárolási helyének lekérdezése feliratú, imént lefuttatott szkript programból tudjuk (no meg azért is, mert mi hoztuk létre!), hogy a E:\TUDOR\KONYVEK1.DAT fájlból helyezkedik el. Ez a fájl tehát biztos, hogy fontos Tudornak. (A 11. fejezetben azt mondta, hogy e területhez Tudornak rendszerszintű jogosultsága van.)

A fent említett mintapéldában azonban Tudor objektumszintű jogosultságot is kapott a Scott felhasználótól, mégpedig annak dept táblájára a

```
CONNECT scott/tiger
GRANT SELECT, INSERT, UPDATE, DELETE
      ON scott.dept
      TO tudor
      WITH GRANT OPTION;
```

utasításokkal. Ennek eredményéről

```
CONNECT tudor/törpilla
COLUMN OWNER      FORMAT A8
COLUMN TABLE_NAME FORMAT A10
COLUMN GRANTOR    FORMAT A8
COLUMN PRIVILEGE  FORMAT A9
COLUMN GRANTABLE  FORMAT A9
COLUMN HIERARCHY FORMAT A9
SELECT *
   FROM user_tab_privs_recd;
CLEAR COLUMNS
```

módon győződhetünk meg, aminek eredménye:

OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY
SCOTT	DEPT	SCOTT	DELETE	YES	NO
SCOTT	DEPT	SCOTT	INSERT	YES	NO
SCOTT	DEPT	SCOTT	SELECT	YES	NO
SCOTT	DEPT	SCOTT	UPDATE	YES	NO

Mivel a dept tábla is fontos Tudornak, nézzük meg, hol is van ez a dept tábla! Futtassuk le az iménti szkript programot, de most a Scottra vonatkozóan:

```
-- Felhasználó saját tábláinak és táblaterületeinek lekérdezése
CONNECT scott/tiger
COLUMN table_name      FORMAT A10
COLUMN tablespace_name FORMAT A15
SELECT table_name, tablespace_name
  FROM user_all_tables;
CLEAR COLUMNS
```

szkript programot. Ennek eredménye

TABLE_NAME	TABLESPACE_NAME
BONUS	SYSTEM
DEPT	SYSTEM
EMP	SYSTEM
SALGRADE	SYSTEM

ahol a SYSTEM táblaterületről a fejtegetésünk elején lefuttatott szkript programból már tudjuk, hogy az E:\ORACLE\ORADATA\MYORA920\SYSTEM01.DBF fájlban van. Ez tehát a másik fájl, amelyik Tudornak fontos. Ennek mérete (a mi géünkön) kb. 420 MByte, amihez, ha hozzávesszük a könyvtár táblaterületet tartalmazó E:\TUDOR\KONYVEK1.DAT fájl 10 MByte méretét, még mindig lényegesen kisebb archiválási feladatot kapunk, mint a teljes adatbázis esetén.

Mindez persze nem az igazi Oracle-megoldás!

## Táblák exportálása, importálása Oracle-eszközökkel

Az Oracle egy adattáblát mint komplex objektumot (a hozzá tartozó megszorításokkal, jogosultságokkal stb.) képes menteni az exportáló programjával (exp.exe), illetve visszatölteni az exportált fájlok ból az importáló programjával (imp.exe). Mindkét program a bin alkönyvtárban található.

Az exportáló és importáló programok interaktív futtatása (a fentiek alapján érhető módon) egyaránt történhet az operációs rendszer parancssorából (az exp, vagy az imp utasítással), vagy az SQL\*Plus-környezetből (a HOST exp, illetve a HOST imp utasítással), de lehetőség van a paraméteres futtatásukra is. Az alábbiakban bemutatjuk a paraméteres táblaexportálást egy min-

tapéldán keresztül. Az importálás is hasonló módon történik, ám erre most nem térünk ki. (Az exportálásról és importálásról részletesen lásd [22].)

Megjegyezzük még, hogy a fenti importáláson kívül az Oracle további lehetőséget is biztosít adattáblák feltöltésére. Az SQL\*Loader az Oracle egy segédprogramja (a bin alkönyvtárban található sqlldr.exe néven. Ismertetése megtalálható a CD-mellékleten lévő sqlldr.hlp szövegfájlban). Segítségével szekvenciális fájból tudunk igen hatékonyan adattáblát feltölteni. (Vázlatos ismertetését lásd [22], részletes bemutatását lásd [12].)

## 12.6. példa

Exportálja a Scott felhasználó emp, dept és salgrade tábláit a C:\MENTETT alkönyvtár scott.dat fájljába az SQL\*Plus-környezetből.

### Megoldás

```
HOST exp scott/tiger      -
       CONSTRAINTS=Y   -
       GRANTS=Y        -
       INDEXES=Y       -
       ROWS=Y          -
       TABLES=(emp,dept,salgrade) -
       FILE=c:\mentett\scott.dat
```



### Megjegyzés

A fenti utasításban az egyes opciók jelentése:

- scott/tiger a felhasználó neve/jelszava,
- CONSTRAINTS=Y a kimentendő táblák *megszorításait* is mentse ki,
- GRANTS=Y a kimentendő táblák *jogosultságait* is mentse ki,
- INDEXES=Y a kimentendő táblák *indexdefiníciót* is mentse ki,
- ROWS=Y a kimentendő táblák *adatait* is mentse ki,
- TABLES=... a kimentendő táblák nevei,
- FILE=... a kimentés eredményét tartalmazó fájl neve.

Megjegyezzük még, hogy az utasításban szereplő „-” jelek az SQL\*Plus szabályos, úgynevet sorfolytató jelei arra az esetre, ha egyetlen sorba nem férne ki egy utasítás (lásd 3. melléklet).

## Tábla tartalmának exportálása fájlba

### A SPOOL UTASÍTÁS

A SPOOL SQL\*Plus-utasítás elindítja vagy leállítja az SQL\*Plus képernyőkimenet fájlba vagy nyomtatóra való küldését. Segítségével naplózni lehet az SQL\*Plus-környezetben lezajló beavatkozásokat, vagy adat- és nézettáblák tartalma exportálható megfelelő SELECT utasítás alkalmazásával.

**Az utasítás szintaktikája**

**SP00L [Fájl/Név[.kiterjesztés] | OFF | OUT]**

ahol:

- **Fájl/Név** az SQL\*Plus-kimenet tárolására szolgáló fájl neve,
- **kiterjesztés** a megadott fájl kiterjesztése,
- **OFF** bezárja a fájlt, leállítja a további kiírást,
- **OUT** leállítja a kiírást, az SQL\*Plus-kimenetet a nyomtatóra küldi.

**Megjegyzés**

- Az opciók nélküli SP00L utasítás az aktuális (még nem lezárt) fájl teljes (útvonal-megadással kiegészített) nevét írja ki a képernyőre.
- A kiterjesztés nélküli állománynév alapértelmezés szerinti kiterjesztése .LST.
- Az SQL\*Plus-kimenet képernyőn való megjelenése a SP00L utasítás kiadása előtt a SET termout OFF utasítással letiltható.

## 12.7. példa

Írjon szkript programot, mely az emp tábla adatait a c:\munka\emp.txt szövegfájlba menti. Az elmentett fájlban az egyes oszlopok adatai között tabulátor karakterek álljanak.

### **Megoldás**

-- Az emp tábla exportálása fájlba

```
rem SET termout OFF
SET newpage 0
SET space 0
SET linesize 400
SET pagesize .0
SET echo OFF
SET feedback OFF
SET heading OFF
SET numwidth 6
CLEAR COLUMNS
CLEAR BREAKS
```

SP00L c:\munka\emp.txt

SELECT empno	CHR(9)
ename	CHR(9)
job	CHR(9)
mgr	CHR(9)
TO_CHAR(hiredate,'YYYY-MON-DD')	CHR(9)
sal	CHR(9)
comm	CHR(9)

```

deptno AS „Az emp táblázat”
FROM emp;
SPOOL
SPOOL OFF

SET space 1
SET pagesize 40
SET feedback 6
SET numwidth 10
rem SET termout ON

```

### Eredmény

Az emp táblázat

7369	SMITH	CLERK	7902	1980-DEC-17	800	20	
7499	ALLEN	SALESMAN		7698	1981-FEB-20	1600	300
7521	WARD	SALESMAN		7698	1981-FEB-22	1250	500
7566	JONES	MANAGER	7839	1981-ÁPR-02	2975	20	
7654	MARTIN	SALESMAN		7698	1981-SZE-28	1250	1400
7698	BLAKE	MANAGER	7839	1981-MÁJ-01	2850	30	
7782	CLARK	MANAGER	7839	1981-JÚN-09	2450	10	
7788	SCOTT	ANALYST	7566	1987-ÁPR-19	3000	20	
7839	KING	PRESIDENT			1981-NOV-17	5000	10
7844	TURNER	SALESMAN		7698	1981-SZE-08	1500	0
7876	ADAMS	CLERK	7788	1987-MÁJ-23	1100	20	
7900	JAMES	CLERK	7698	1981-DEC-03	950	30	
7902	FORD	ANALYST	7566	1981-DEC-03	3000	20	
7934	MILLER	CLERK	7782	1982-JAN-23	1300	10	



### Megjegyzés

A fenti szkript programban az SQL\*Plus rendszerváltozóinak értékeit oly módon állítottuk be, hogy a táblatartalom listázási képét semmiképpen ne torzíthassák.

A rendszerváltozók beállított értékeinek jelentése:

- SET newpage 0      ne dobjon üres lapot a legelején,
- SET space 0        nincs üres sor lapváltásnál,
- SET linesize 400    a sorméret elegendően nagy legyen,
- SET pagesize 0     az írás folyamatos legyen (ne legyen lapozás),
- SET echo OFF      ne írja ki az utasításokat,
- SET feedback OFF   ne írjon ki semmilyen visszajelzést,
- SET heading OFF   törölje az esetleges fejlécet,
- SET numwidth 6     a numerikus adatmezők hossza legyen 6,
- CLEAR COLUMNS    törölje az esetleges oszlopdefiníciókat,
- CLEAR BREAKS     semelyik oszlop értékváltására ne reagáljon.

Megjegyezzük, hogy a tabulátor karakter (kódja 9) hatására a keletkezett szöveg kissé kau-tikusnak tűnhet, ám a Microsoft Office alkalmazásokba való betöltéshez ez igen alkalmasnak fog bizonyulni.

## MÁSOLÁS VÁGÓLAPRA

Egy SELECT utasítás kimeneti listáját az SQL\*Plus-környezetben kijelölés után vágólapra másolhatjuk, majd onnan beilleszthetjük bármilyen szövegfájlba. Ez dokumentálás céljára megfelel, de például sem a Word táblázatkezelése, sem az Excel nem tud vele mit kezdeni.

## Tábla tartalmának betöltése Wordbe

Egy adat- vagy nézettábla Word-táblázatba való betöltésének előfeltétele, hogy táblaadatokat tartalmazó szövegfájlban az egyes oszlopértékek között valamelyen jól meghatározható karakter álljon. E karakter többféle is lehet, ám ha figyelembe vesszük, hogy a táblák szöveges adatokat is tartalmazhatnak, akkor célszerű olyat választani, amelynek kicsi az előfordulási valószínűsége, a tabulátor karakter éppen ilyen.

### 12.8. példa

A c:\munka\emp.txt szövegfájlba mentett emp tábla adatait töltse be egy Microsoft Word dokumentumba.

#### Megoldás

##### 1. lépés (Táblatartalom exportálása szövegfájlba)

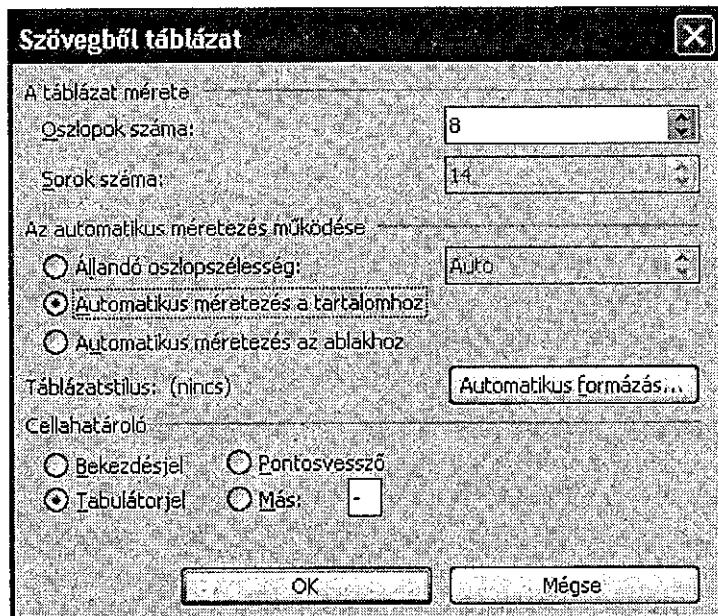
A fentiekre tekintettel abból indulunk ki, hogy a kiválasztott tábla adatait (a SP00L utasítás segítségével) már egy szövegfájlba exportáltuk.

##### 2. lépés (A szövegfájl bemásolása a dokumentumba)

Akár vágólapról való átmásolással akár beszúrással illesszük be a szövegfájl tartalmát a Word-dокументumunkba, majd jelöljük ki az adatokat (az oszlopfejlécet ne!).

##### 3. lépés (A szöveg táblázattá való konvertálása)

Ezek után a Táblázat menüpont Konvertálás almenüjének Szöveg Táblázattá... pontjára kattintunk. A megjelenő ablakban ellenőrizzük, illetve állítsuk be az Oszlopok száma értéket (esetünkben ez most 8), és a Cellahatároló jelet (*Tabulátorjelek*), valamint válasszuk ki az Automatikus méretezés a tartalomhoz beállítást.



12.5. ábra. A táblázattá konvertálás beállítása

Az OK gombra való kattintás után már meg is jelenik a táblázat (melyet kiegészítésképpen – ha szükséges – még állíthatunk fejlécettel):

7369	SMITH	CLERK	7902	1980-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	1981-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	1981-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	1981-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	1981-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	1981-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	1987-ÁPR-19	3000		20
7839	KING	PRESIDENT		1981-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	1981-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	1987-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	1981-DEC-03	950		30
7902	FORD	ANALYST	7566	1981-DEC-03	3000		20
7934	MILLER	CLERK	7782	1982-JAN-23	1300		10

Megjegyezzük, hogy ez a táblázat már felhasználható egyéb táblázatos alkalmazásokban, például Excelben. (Az átvitelhez alkalmazhatunk ismét vágólapot.)

## Tábla tartalmának betöltése Excelbe

Excel esetén a teendők java része megegyezik az előzőben leírtakkal, az eltérések inkább a két program jellegének különbségéből adódnak. A Word-táblázatok, bár lényegesen többet

tudnak annál, mint ami egy egyszerű dokumentáláshoz szükséges (használhatunk bizonyos oszlopfüggvényeket, rendezhetjük a sorokat, sőt táblázatként átvihetjük más alkalmazásokba – például az Excelbe is), mégis természetesen szerényebb képességűek, mint egy igazi Excel-táblázat, ami érthető is, hiszen az Excelnek ez a „hivatása”.

Az Excel-táblázatok legfontosabb jellegzetessége, hogy oszlopaik különböző típusúak lehetnek, és a típusuktól függő, rendkívül nagyszámú függvény alkalmazható rajtuk. Különböző összetett kiértékeléseket végezhetünk rajtuk (érzékenységvizsgálatok, „mi lenne, ha...?” típusú feldolgozások, sőt még néhány adatbázis-művelet is), és természetesen – az Excel egyik erősségeként – látványos grafikákat (oszlop-, vonal-, kördiagramokat stb.) is készíthetünk.

## 12.9. példa

A c:\munka\emp.txt szövegfájlba mentett emp tábla adatait töltse be egy Microsoft Excel munkatáblába.

### Megoldás

#### 1. lépés (Táblatartalom exportálása szövegfájlba)

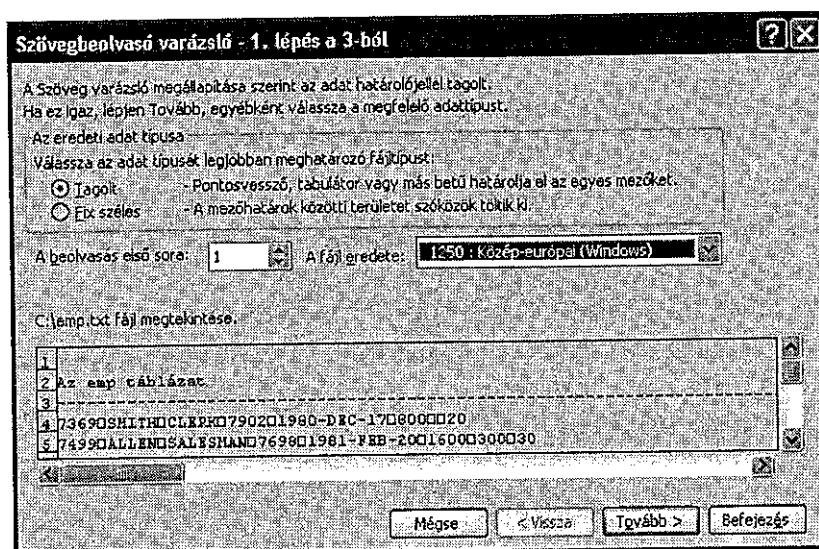
A táblaadatoknak az Oracle-ból az Excelbe való betöltésekor is hasznos, ha a táblaadatokat tartalmazó szövegfájlból tabuláló jelet (CHR(9)) alkalmazunk az egyes oszlopértékek között, ezért az 1. lépés megegyezik a Wordnél említettel. (Tehát a kiválasztott tábla adatait a SP00L utasítás segítségével egy szövegfájlba exportáljuk.)

#### 2. lépés (A szövegfájl betöltése az Excelbe)

Itt már lényeges az eltérés a Wordnél bemutatottól, a létrejött szövegfájlt ugyanis nem bemsoljuk, hanem betöljtük az Excelbe (úgy, ahogy van, az oszlop-fejléccel együtt). Ennek módja: a Fájl menüpont Megnyitás almenütét kijelöljük, majd a beugró fájlablakban a Fájltípus legördülő menüben kiválasztjuk a Szövegfájlok (\*.prn; \*.txt; \*.csv) menüpontot, végül az 1. lépében létrehozott szövegfájlt megkeressük a megfelelő alkönyvtárban, és a nevére duplán kattintunk. Így bejutunk a Szövegbeolvasó varázslóba.

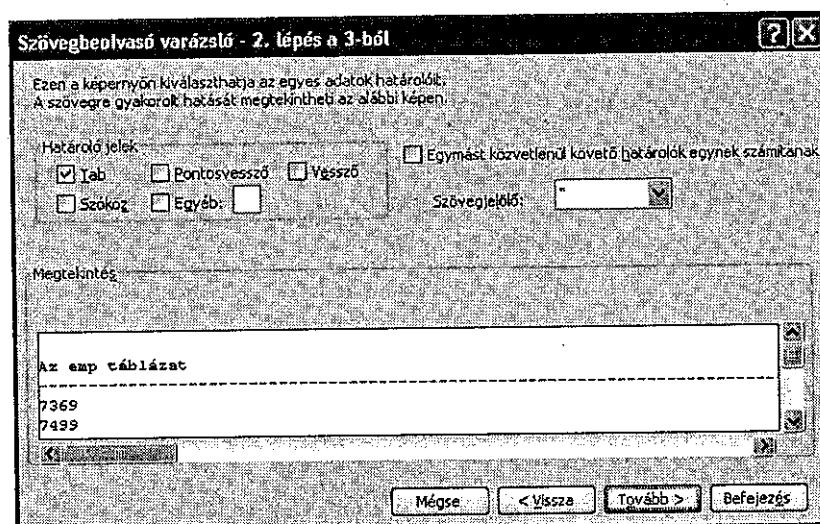
#### 3. lépés (A szöveg táblázattá való konvertálása)

A Szövegbeolvasó varázsló első ablakában (12.6. ábra) Az eredeti adat típusaként Tagolt beállítást, A fájl eredete legördülő menüpontban pedig az 1250 : Közép-európai (Windows) beállítást választjuk ki, majd a Tovább gombra kattintunk:



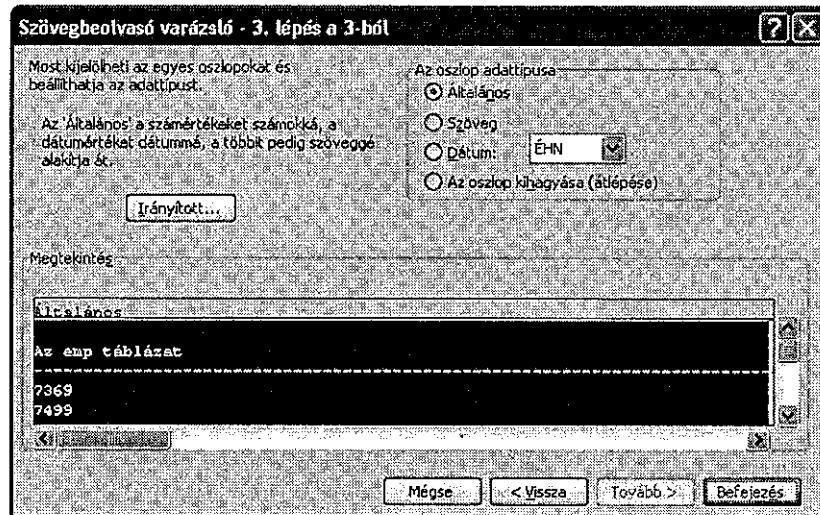
12.6. ábra. A Szövegbeolvasó varázsló első ablaka

A második ablakban (12.7. ábra) a *Hatóroló jelek* közül a *Tab* jelölőnégyzetet választjuk ki, és továbbléünk (*Tovább*):



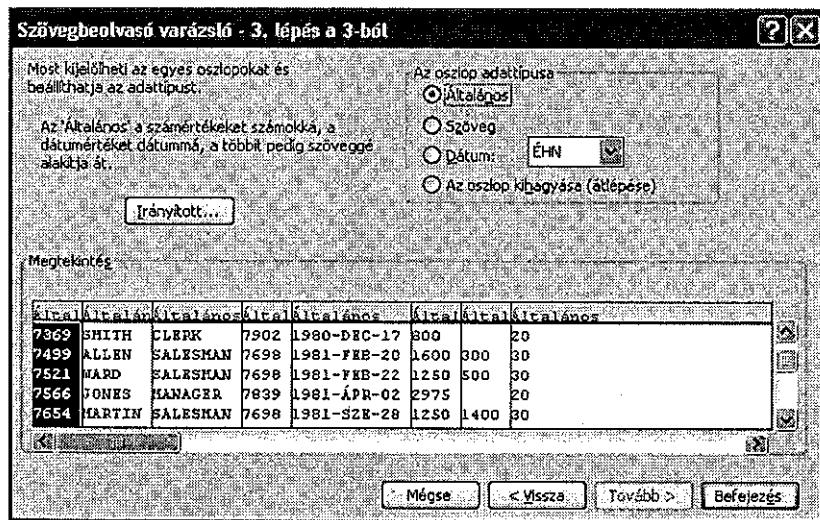
12.7. ábra. A Szövegbeolvasó varázsló második ablaka

A harmadik ablakban (12.8. ábra) állíthatjuk be az egyes oszlopok típusát.



12.8. ábra. A Szövegbeolvasó varázsló harmadik ablaka

Ha a Megtekintés részablakbeli függőleges gördítősávot egy kicsit lejjebb húzzuk, akkor láthatjuk (12.9. ábra), hogy minden oszlop felett az Általános felirat áll, melyről az ablakbeli tájékoztatásból megtudhatjuk, hogy az egyes oszlopok tartalma alapján automatikus szám-, dátum- vagy szövegtípus-konverziót eredményez.



12.9. ábra. A Szövegbeolvasó varázsló harmadik ablaka a táblázat oszlopaival

Mivel számunkra ez éppen megfelelő, ezért a Befejezés gombra kattintunk, amelynek nyomán már megjelenik egy igazi, oszloptípusokkal rendelkező Excel-táblázat (12.10. ábra), majd az oszlopszélességek igazítása után megkapjuk a végleges alakot (12.11. ábra).

#### 4. lépés (A táblázat mentése)

Mivel az Excelbe a betöltött szövegfájlt az elvégzett konvertálás után Excel-munkatáblaként kívánjuk elmenteni, igen fontos a mentési folyamat megfelelő elvégzése.

Először válasszuk ki a Fájl menüpont Mentés másként almenüjét, majd a beugró fájlablakban a Fájltípus legördülő menüben ezúttal a Microsoft Office Excel munkafüzet (\*.xls) menüpontot választjuk ki. Ezután a kívánt néven a kívánt alkonyvtárba elmentjük a táblázatunkat (Mentés).

The screenshot shows the Microsoft Excel interface with a window titled 'Microsoft Excel - emp.txt'. The file contains the following data:

	A	B	C	D	E	F	G	H	
1									
2	Az emp táblázat								
3	-----								
4	7369	SMITH	CLERK	7902	1980-DEC-17	800	20		
5	7499	ALLEN	SALESMAN	7698	1981-FEB-20	1600	300	30	
6	7521	WARD	SALESMAN	7698	1981-FEB-22	1250	500	30	
7	7566	JONES	MANAGER	7839	1981-APR-02	2975		20	
8	7654	MARTIN	SALESMAN	7698	1981-SZE-28	1250	1400	30	
9	7698	BLAKE	MANAGER	7839	1981-MÁJ-01	2850		30	
10	7782	CLARK	MANAGER	7839	1981-JÚN-09	2450		10	
11	7788	SCOTT	ANALYST	7566	1987-APR-19	3000		20	
12	7839	KING	PRESIDENT		1981-NOV-17	5000		10	
13	7844	TURNER	SALESMAN	7698	1981-SZE-08	1500	0	30	
14	7876	ADAMS	CLERK	7788	1987-MÁJ-23	1100		20	
15	7900	JAMES	CLERK	7698	1981-DEC-03	950		30	
16	7902	FORD	ANALYST	7566	1981-DEC-03	3000		20	
17	7934	MILLER	CLERK	7782	1982-JAN-23	1300		10	
18									

12.10. ábra. A konvertált, de még „nyers” Excel-táblázat

The screenshot shows the Microsoft Excel interface with a window titled 'Microsoft Excel - emp.txt'. The file contains the same data as in Figure 12.10, but with adjusted column widths. The data is as follows:

	A	B	C	D	E	F	G	H	
1									
2	Az emp táblázat								
3	-----								
4	7369	SMITH	CLERK	7902	1980-DEC-17	800	20		
5	7499	ALLEN	SALESMAN	7698	1981-FEB-20	1600	300	30	
6	7521	WARD	SALESMAN	7698	1981-FEB-22	1250	500	30	
7	7566	JONES	MANAGER	7839	1981-APR-02	2975		20	
8	7654	MARTIN	SALESMAN	7698	1981-SZE-28	1250	1400	30	
9	7698	BLAKE	MANAGER	7839	1981-MÁJ-01	2850		30	
10	7782	CLARK	MANAGER	7839	1981-JÚN-09	2450		10	
11	7788	SCOTT	ANALYST	7566	1987-APR-19	3000		20	
12	7839	KING	PRESIDENT		1981-NOV-17	5000		10	
13	7844	TURNER	SALESMAN	7698	1981-SZE-08	1500	0	30	
14	7876	ADAMS	CLERK	7788	1987-MÁJ-23	1100		20	
15	7900	JAMES	CLERK	7698	1981-DEC-03	950		30	
16	7902	FORD	ANALYST	7566	1981-DEC-03	3000		20	
17	7934	MILLER	CLERK	7782	1982-JAN-23	1300		10	
18									

12.11. ábra. Az oszlopszélesség igazítása utáni végleges Excel-táblázat

# Részletező csoportosítások és analitikus függvények az SQL-ben

## Elméleti összefoglaló

Gyakran van szükség különböző részletezettséggű csoportosítások szerinti tulajdonságok lekérdezésére, „mi lenne, ha...” típusú kérdések megválaszolására, vagy rangsorok előállítására. E problémákat elemző jellegűeknek nevezhetnénk, nem csupán az elemzőfüggvények miatt, hanem a vizsgált adathalmaz belső szerkezetét kifinomult módon firtató jellegük miatt is. E fejezetben tehát ilyen elemző feladatokra mutatunk be eszközöket.

Miután ez a terület a magyar nyelvű szakirodalomban újdonságnak számít, ezért az elméleti összefoglalóban sok és részletes példát mutatunk be. Ezekhez hasonló feladatsort viszont már nem lett volna célszerű készíteni. Hasznosabbnak gondoltuk, hogy inkább egy „valódi” vállalati környezetben, a következő fejezetben bemutatásra kerülő DEMO vállalatra vonatkozóan fogalmazunk meg olyan feladatokat, melyek megoldásához e fejezet eszközeinek hatékonysága nyilvánvalóvá válik.

## Az SQL új csoportképzési eszközei

### A ROLLUP operátor

A ROLLUP operátor segítségével csoportfüggvényeket alkalmazhatunk oszlopok sorozatán egyre finomodó részcsoporthozzával. A ROLLUP operátort a GROUP BY utasításrészben használjuk a következő szintaktika szerint:

GROUP BY ROLLUP (oszlop1 [, oszlop2]...)

A szelekciós listában kijelölt csoportműveletet a ROLLUP kulcsszó utáni, zárójelben lévő oszlopok szerint végzi a megadott oszlopsorrend szerint jobbról balra (lásd az alábbi példát). Az ezen oszlopokban (nem eredeti táblaadatként) NULL értéket tartalmazó sorok adják a csoportműveletet részértékeit, illetve a végértékét a csoportművelet oszlopában. E megjelenítés eredményeként a függőleges irányban eddig homogén (azonos jelentésű) listák, a ROLLUP operátor következtében ebben az irányban is strukturálttá váltak. (A csoportosítások esetén felmerülő problémákkal kapcsolatban lásd 14.8. feladat.)

### 13.1. példa

Listázzuk ki részlegenként, és ezen belül foglalkozásonként az összfizetéseket, továbbá a részlegenkénti összfizetéseket, végül a vállalati összfizetést, és ugyanezen csoportosításban a létszámokat is:

#### 1. megoldás (Egyszerű csoportosítással)

```
SELECT deptno, job, SUM(sal), COUNT(*) AS Létszám
  FROM emp
 GROUP BY deptno, job;
```

```
SELECT deptno, SUM(sal), COUNT(*) AS Létszám
  FROM emp
 GROUP BY deptno;
```

```
SELECT SUM(sal), COUNT(*) AS Létszám
  FROM emp;
```

#### Eredmény

DEPTNO	JOB	SUM(SAL)	LÉTSZÁM
10	CLERK	1300	1
10	MANAGER	2450	1
10	PRESIDENT	5000	1
20	CLERK	1900	2
20	ANALYST	6000	2
20	MANAGER	2975	1
30	CLERK	950	1
30	MANAGER	2850	1
30	SALESMAN	5600	4

DEPTNO	SUM(SAL)	LÉTSZÁM
10	8750	3
20	10875	5
30	9400	6

SUM(SAL)	LÉTSZÁM
29025	14

#### 2. megoldás (Részcsoporthálózattal)

```
SELECT deptno, job, SUM(sal), COUNT(*) AS Létszám
  FROM emp
 GROUP BY ROLLUP (deptno, job);
```

**Eredmény**

DEPTNO	JOB	SUM(SAL)	LÉTSZÁM
10	CLERK	1300	1
10	MANAGER	2450	1
10	PRESIDENT	5000	1
10		8750	3
20	CLERK	1900	2
20	ANALYST	6000	2
20	MANAGER	2975	1
20		10875	5
30	CLERK	950	1
30	MANAGER	2850	1
30	SALESMAN	5600	4
30		9400	6
		29025	14

13 sor kijelölve.

**Megjegyzés**

Látható tehát, hogy a ROLLUP operátor nélküli GROUP BY utasításrész használatakor az eredeti feladat megoldásához három utasításra volt szükség, mivel így csupán egyszerű csoportösszegeket tudunk képezni a megadott oszlopok szerinti csoportosításban.

A ROLLUP operátort használva már összesített részösszegeket is kapunk a paraméteroszlopok szerint jobbról balra haladva, míg végül az utolsó sorban megkapjuk az összesített részösszeget, vagyis a végösszeget. Ilyen módon az eredeti feladat megoldásához egyetlen utasítás elegendő volt.

**13.2. példa**

Listázza ki munkakörönként és azon belül részlegenként az összfizetést, az átlagfizetést, valamint a csoportlétszámot.

**Megoldás**

```

SELECT job           AS munkakör,
       deptno        AS részleg,
       SUM(sal)       AS részösszeg,
       ROUND(AVG(sal),0) AS átlag,
       COUNT(*)       AS létszám
  FROM emp
 GROUP BY ROLLUP (job, deptno);

```

**Eredmény**

MUNKAKÖR	RÉSZLEG	RÉSZÖSSZEG	ÁTLAG	LÉTSZÁM
CLERK	10	1300	1300	1
CLERK	20	1900	950	2
CLERK	30	950	950	1
CLERK		4150	1038	4
ANALYST	20	6000	3000	2
ANALYST		6000	3000	2
MANAGER	10	2450	2450	1
MANAGER	20	2975	2975	1
MANAGER	30	2850	2850	1
MANAGER		8275	2758	3
SALESMAN	30	5600	1400	4
SALESMAN		5600	1400	4
PRESIDENT	10	5000	5000	1
PRESIDENT		5000	5000	1
		29025	2073	14

15 sor kijelölve.

**13.3. példa**

Oldja meg az előző példa feladatát oly módon, hogy a lista összesítéseket tartalmazó sorába megjelenjen az összesítés értelmezése.

**1. megoldás (CASE kifejezéssel)**

```

SELECT
    NVL(job,'MunkakörÖsszes') AS Munkakör,
    CASE
        WHEN job is NULL
        THEN NULL
        ELSE
            SUBSTR(NVL(TO_CHAR(deptno),'RészlegÖsszes'),1,15)
    END AS részleg,
    SUM(sal) AS részösszeg,
    ROUND(AVG(sal)) AS átlag,
    COUNT(*) AS létszám
FROM emp
GROUP BY ROLLUP (job, deptno);

```

**2. megoldás (DECODE függvényvel)**

```

SELECT
    NVL(job,'MunkakörÖsszes') AS Munkakör,

```

```

    DECODE(NVL(job,'X'),
           'X', NULL,
           SUBSTR(NVL(TO_CHAR(deptno), 'RészlegÖsszes'),1,15))
                           AS részleg,
    SUM(sal)                  AS részösszeg,
    ROUND(AVG(sal))          AS átlag,
    COUNT(*)                  AS létszám
  FROM emp
 GROUP BY ROLLUP (job, deptno);

```

**Eredmény (mindkét esetben)**

MUNKAKÖR	RÉSZLEG	RÉSZÖSSZEG	ÁTLAG	LÉTSZÁM
CLERK	10	1300	1300	1
CLERK	20	1900	950	2
CLERK	30	950	950	1
CLERK	RészlegÖsszes	4150	1038	4
ANALYST	20	6000	3000	2
ANALYST	RészlegÖsszes	6000	3000	2
MANAGER	10	2450	2450	1
MANAGER	20	2975	2975	1
MANAGER	30	2850	2850	1
MANAGER	RészlegÖsszes	8275	2758	3
SALESMAN	30	5600	1400	4
SALESMAN	RészlegÖsszes	5600	1400	4
PRESIDENT	10	5000	5000	1
PRESIDENT	RészlegÖsszes	5000	5000	1
MunkakörÖsszes		29025	2073	14

15 sor kijelölve.

**A CUBE operátor**

A CUBE operátor hasonló a ROLLUP operátorhoz, de segítségével a sorok csoportjaival valamely oszlop(ok)ra vonatkozó összes csoportosítási kombináció szerinti részösszege meghatározható.

GROUP BY CUBE (*oszlop1 [, oszlop2]...*)

A részösszegképzést a zárójelben lévő oszlopok szerint végzi a megadott oszlopsorrend szerint jobbról balra, de az összes kiválasztási kombinációt előállítva (lásd a következő példát). Az üres attribútumú sorok adják a részösszegeket, illetve a végösszeget.

### 13.4. példa

Oldja meg a 13.2. példa feladatát oly módon, hogy a deptno és a job oszlopokra vonatkozó összes felbontási kombináció részösszegét megjelenítse. Tehát képezzen részösszeget

- munkakörönként és azon belül részlegenként (ez a Részösszeg11) (ezzel ekvivalens, ezért külön nem elvégzendő: részlegenként és azon belül munkakörönként),
- munkakörönként (ez a Részösszeg10),
- részlegenként (ez a Részösszeg01),
- teljes táblára (ez a Részösszeg00, amely egyben a Végösszeg).

#### 1. megoldás (A GROUP BY utasításrész használatával)

```
COLUMN deptno          FORMAT 99
```

PROMPT Részösszeg11:

```
SELECT deptno, job, SUM(sal), COUNT(*) AS Létszám
      FROM emp
     GROUP BY deptno, job;
```

PROMPT Részösszeg10:

```
SELECT job, SUM(sal), COUNT(*) AS Létszám
      FROM emp
     GROUP BY job;
```

PROMPT Részösszeg01:

```
SELECT deptno, SUM(sal), COUNT(*) AS Létszám
      FROM emp
     GROUP BY deptno;
```

PROMPT Részösszeg00 (Végösszeg):

```
SELECT SUM(sal), COUNT(*) AS Létszám
      FROM emp;
```

CLEAR COLUMNS

#### Eredmény

Részösszeg11:

DEPTNO	JOB	SUM(SAL)	LÉTSZÁM
10	CLERK	1300	1
10	MANAGER	2450	1
10	PRESIDENT	5000	1
20	CLERK	1900	2
20	ANALYST	6000	2
20	MANAGER	2975	1

30 CLERK	950	1
30 MANAGER	2850	1
30 SALESMAN	5600	4

Részösszeg10:

JOB	SUM(SAL)	LÉTSZÁM
ANALYST	6000	2
CLERK	4150	4
MANAGER	8275	3
PRESIDENT	5000	1
SALESMAN	5600	4

Részösszeg01:

DEPTNO	SUM(SAL)	LÉTSZÁM
10	8750	3
20	10875	5
30	9400	6

Részösszeg00 (Végösszeg):

SUM(SAL)	LÉTSZÁM
29025	14

**2. megoldás (A GROUP BY utasításrész és a CUBE operátor használatával)**

```
SELECT deptno, job, SUM(sal), COUNT(*) AS Létszám
  FROM emp
 GROUP BY CUBE (deptno, job);
```

**Eredmény**

DEPTNO	JOB	SUM(SAL)	LÉTSZÁM	
		29025	14	⇐ Részösszeg00
	CLERK	4150	4	⇐ Részösszeg10
	ANALYST	6000	2	⇐ Részösszeg10
	MANAGER	8275	3	⇐ Részösszeg10
	SALESMAN	5600	4	⇐ Részösszeg10
	PRESIDENT	5000	1	⇐ Részösszeg10
10		8750	3	⇐ Részösszeg01
10	CLERK	1300	1	⇐ Részösszeg11
10	MANAGER	2450	1	⇐ Részösszeg11
10	PRESIDENT	5000	1	⇐ Részösszeg11
20		10875	5	⇐ Részösszeg01

20 CLERK	1900	2	← Részösszeg11
20 ANALYST	6000	2	← Részösszeg11
20 MANAGER	2975	1	← Részösszeg11
30	9400	6	← Részösszeg01
30 CLERK	950	1	← Részösszeg11
30 MANAGER	2850	1	← Részösszeg11
30 SALESMAN	5600	4	← Részösszeg11

18 sor kijelölve.



### Megjegyzés

- Tehát a CUBE operátor minden csoportosítási kombinációt előállít, míg a ROLLUP operátor csak az adott sorrendhez tartozókat. Ha a fenti példában a CUBE helyett ROLLUP operátort használtunk volna, akkor nem kaptuk volna meg a részösszeg10 értékeket (illetve az ezeket tartalmazó sorokat).
- Mivel ilyen módon igen nagy eredménytábla keletkezik (melyben szereplő összes csoportosítási kombinációra az esetek jelentős részében nem tartunk igényt), ezért a CUBE operátort viszonylag ritkán használjuk.

## A GROUPING függvény

A GROUPING indikátorfüggvény megmutatja, hogy a lista egy sorában „ki a felelős” a paraméterek közül. Használható a ROLLUP és a CUBE operátorokkal együtt oly módon, hogy 0 értéket rendel ahhoz az oszlophoz, amelyik szerint csoportképzés történt (vagyis amelyik csoportképző oszlop). Természetesen a csoportosítás vonatkozhat több oszlopra is.

### 13.5. példa

Listálja a részlegenkénti foglalkozásonkénti összfizetéseket, a részlegenkénti összfizetéseket, munkakörönkénti összfizetéseket, valamint a vállalati összfizetést, úgy, hogy minden sorban 0 értéket rendeljen ahhoz az oszlophoz, amelyik szerint ott történt csoportképzés, és 1 értéket pedig ahhoz, amelyik szerint ott nem történt csoportképzés.

#### 1. megoldás (A GROUPING függvény használatával)

```
COLUMN „Részleg” FORMAT 99
SELECT deptno AS "Részleg",
       job      AS "Foglalkozás",
       SUM(sal) AS "Részösszeg",
       GROUPING(deptno) AS "RészlegInd",
       GROUPING(job)   AS "FoglalkozásInd"
FROM emp
GROUP BY CUBE (deptno, job);
CLEAR COLUMNS
```

**Eredmény**

	Részleg	Foglalkoz	Részösszeg	RészlegInd	FoglalkozásInd
		29025		1	1
	CLERK	4150		1	0
	ANALYST	6000		1	0
	MANAGER	8275		1	0
	SALESMAN	5600		1	0
	PRESIDENT	5000		1	0
10		8750		0	1
10	CLERK	1300		0	0
10	MANAGER	2450		0	0
10	PRESIDENT	5000		0	0
20		10875		0	1
20	CLERK	1900		0	0
20	ANALYST	6000		0	0
20	MANAGER	2975		0	0
30		9400		0	1
30	CLERK	950		0	0
30	MANAGER	2850		0	0
30	SALESMAN	5600		0	0

18 sor kijelölve.

**2. megoldás (A GROUPING függvény és a CASE kifejezés használatával)**

```
COLUMN „Részleg” FORMAT 99
SELECT deptno AS "Részleg",
       job      AS "Foglalkozás",
       SUM(sal) AS "Részösszeg",
       CASE
          WHEN GROUPING(deptno)=1 THEN 'nem csoportosít'
          ELSE 'csoportosít'
       END      AS "RészlegInd",
       CASE
          WHEN GROUPING(job)=1 THEN 'nem csoportosít'
          ELSE 'csoportosít'
       END      AS "FoglalkozásInd"
FROM emp
GROUP BY CUBE (deptno,job);
CLEAR COLUMNS
```

**Eredmény**

Részleg	Foglalkoz	Részösszeg	RészlegInd	FoglalkozásInd
		29025	nem csoportosít	nem csoportosít
CLERK		4150	nem csoportosít	csoportosít
ANALYST		6000	nem csoportosít	csoportosít
MANAGER		8275	nem csoportosít	csoportosít
SALESMAN		5600	nem csoportosít	csoportosít
PRESIDENT		5000	nem csoportosít	csoportosít
10		8750	csoportosít	nem csoportosít
10 CLERK		1300	csoportosít	csoportosít
10 MANAGER		2450	csoportosít	csoportosít
10 PRESIDENT		5000	csoportosít	csoportosít
20		10875	csoportosít	nem csoportosít
20 CLERK		1900	csoportosít	csoportosít
20 ANALYST		6000	csoportosít	csoportosít
20 MANAGER		2975	csoportosít	csoportosít
30		9400	csoportosít	nem csoportosít
30 CLERK		950	csoportosít	csoportosít
30 MANAGER		2850	csoportosít	csoportosít
30 SALESMAN		5600	csoportosít	csoportosít

18 sor kijelölve.

**Ellenőrzés**

```
SELECT deptno, SUM(sal)
  FROM emp
 GROUP BY deptno;
```

**Eredmény**

DEPTNO	SUM(SAL)
10	8750
20	10875
30	9400

**Megjegyzés**

- Az ellenőrzés mintapéldájában a deptno oszlop szerint csoportosított részösszegeket kaptuk meg, melyeket tartalmazó feladatmegoldásbeli sorokban ezért a RészlegInd értéke 0, illetve csoportosít.
- Mivel a GROUPING függvény paramétere csak egyetlen oszlop, így több oszlop esetén használata nehézkes, ezért ilyenkor inkább az alább bemutatásra kerülő GROUPING\_ID indikátorfüggvényt alkalmazzuk a csoportosítás kifejezésére.
- Megjegyezzük, hogy a fenti feladat a CASE kifejezés helyett a DECODE függvénnyel is megoldható lett volna.



## A GROUPING\_ID függvény

A GROUPING\_ID indikátorfüggvény értéke egy pozitív egész szám, amely összhangban van a GROUPING függvénnel, és szintén minden sorhoz társítva van. Ennek a függvénynek egy adott sorbeli értéke megadja, hogy a ROLLUP vagy CUBE operátor paraméteroszlopai közül melyik szerint nem történt a csoportképzés. Ennek meghatározása úgy történik, hogy a paraméteroszlopok közül a legjobboldalihoz tartozik 2-nek a 0-adik hatványa, attól balra az 1-ső hatványa stb. Így egy adott sorban e függvény értéke a csoportképzésben aktuálisan részt nem vevő paraméteroszlopokhoz tartozó értékek összege. Például CUBE (A, B, C) esetén, ha egy sor a B oszlop szerinti csoportosítás eredménye, akkor a GROUPING\_ID függvény értéke 5, mivel a C oszlop miatt  $2^0 = 1$ , az A oszlop pedig  $2^2 = 4$ . Az így keletkező függvényértékekkel oly módon lehet rendezni az eredménytábla sorait (lásd az alábbi példa megjegyzését), hogy a legtöbb oszlop szerinti csoportosítás (a legfinomabb felbontás) eredményei az eredménytábla elejére, míg a csoportosítás nélküli eredmények (például a végösszeg) az eredménytábla végére kerüljenek.

### 13.6. példa

Listázza ki a részlegenkénti foglalkozásonkénti összfizetéseket, a részlegenkénti összfizetéseket, a munkakörönkénti összfizetéseket, valamint a vállalati összfizetést, úgy, hogy egy indikátoroszlop minden sorban jelezze, hogy mely oszlop szerint történt a csoportosítás.

#### 1. megoldás

```
COLUMN „Részleg” FORMAT 99
SELECT deptno AS "Részleg",
       job      AS "Foglalkozás",
       SUM(sal) AS "Részösszeg",
       GROUPING_ID(deptno, job) AS "Indikátor"
  FROM emp
 GROUP BY CUBE(deptno, job);
CLEAR COLUMNS
```

#### Eredmény

Részleg	Foglalkoz	Részösszeg	Indikátor
	29025		3
CLERK	4150		2
ANALYST	6000		2
MANAGER	8275		2 (***)
SALESMAN	5600		2
PRESIDENT	5000		2
10	8750		1
10 CLERK	1300		0
10 MANAGER	2450		0
10 PRESIDENT	5000		0

20	10875	1
20 CLERK	1900	0 (*)
20 ANALYST	6000	0
20 MANAGER	2975	0
30	9400	1 (**)
30 CLERK	950	0
30 MANAGER	2850	0
30 SALESMAN	5600	0

18 sor kijelölve.



### Megjegyzés

Egy oszlophoz rendelt 0 (helyiértéken figyelembe véve) itt is a csoportosítást jelzi. Ennek megfelelően a GROUPING\_ID(deptno, job) függvény az értelmezése szerint  $0+0=0$  értéket ad, ha mind a deptno, mind a job szerint,  $0+1=1$  értéket, ha csak a deptno szerint és  $2+0=2$  értéket, ha csak a job szerint történt csoportképzés. Ennek megfelelően a fenti lista (\*) sora a GROUP BY deptno, job, a (\*\*) sora a GROUP BY deptno, míg a (\*\*\*) sora a GROUP BY job csoportképzés eredményét adja a részösszeképzésben.

## 2. megoldás (Tekintsük a fenti listát az indikátorfüggvény szerint rendezve)

```
COLUMN "Részleg"           FORMAT 99
SELECT deptno               AS "Részleg",
       job                   AS "Foglalkozás",
       SUM(sal)              AS "Részösszeg",
       GROUPING_ID(deptno, job) AS "Indikátor"
FROM emp
GROUP BY CUBE (deptno,job)
ORDER BY "Indikátor";
CLEAR COLUMNS
```

### Eredmény

Részleg	Foglalkoz	Részösszeg	Indikátor
10 CLERK	1300	0	
10 MANAGER	2450	0	
10 PRESIDENT	5000	0	
20 CLERK	1900	0	
30 CLERK	950	0	
30 SALESMAN	5600	0	
30 MANAGER	2850	0	
20 MANAGER	2975	0	
20 ANALYST	6000	0	
10	8750	1	
20	10875	1	

30	9400	1
CLERK	4150	2
ANALYST	6000	2
MANAGER	8275	2
PRESIDENT	5000	2
SALESMAN	5600	2
	29025	3

18 sor kijelölve.



### Megjegyzés

Az eredmény ilyen módon rendezve áttekinthető képet ad a különböző szintű csoportok részösszegeiről. Az 1. megoldásbeli (rendezés nélküli) lista előnye viszont, hogy a hierarchiaszintek egymás közötti kapcsolatát mutatja szemléletesen.

## A GROUPING SETS függvény

A GROUPING SETS a GROUP BY utasításrész további kiterjesztése. Segítségével több csoportot képezhetünk egy lekérdezésben. Ilyenkor a GROUPING SETS paramétereként megadott oszlop-halmazokra (és csak azokra) végzi el a csoportosítást.

### 13.7. példa

Képezzen részösszegeket a fizetés (sal) oszlopon az alábbi csoportosításokban:

- mgr, deptno, job,
- deptno, job és
- mgr.

### Megoldás

```
SELECT mgr, deptno, job, SUM(sal)
  FROM emp
 GROUP BY
    GROUPING SETS((mgr, deptno, job),
                  (deptno, job),
                  (mgr));
```

### Eredmény

MGR	DEPTNO	JOB	SUM(SAL)	
7782	10	CLERK	1300	⇐ (mgr, deptno, job)
7839	10	MANAGER	2450	⇐ (mgr, deptno, job)
	10	PRESIDENT	5000	⇐ (mgr, deptno, job)
7788	20	CLERK	1100	⇐ (mgr, deptno, job)
7902	20	CLERK	800	⇐ (mgr, deptno, job)

7566	20 ANALYST	6000	$\Leftarrow$ (mgr, deptno, job)
7839	20 MANAGER	2975	$\Leftarrow$ (mgr, deptno, job)
7698	30 CLERK	950	$\Leftarrow$ (mgr, deptno, job)
7839	30 MANAGER	2850	$\Leftarrow$ (mgr, deptno, job)
7698	30 SALESMAN	5600	$\Leftarrow$ (mgr, deptno, job)
	10 CLERK	1300	$\Leftarrow$ (deptno, job)
	10 MANAGER	2450	$\Leftarrow$ (deptno, job)
	10 PRESIDENT	5000	$\Leftarrow$ (deptno, job)
	20 CLERK	1900	$\Leftarrow$ (deptno, job)
	20 ANALYST	6000	$\Leftarrow$ (deptno, job)
	20 MANAGER	2975	$\Leftarrow$ (deptno, job)
	30 CLERK	950	$\Leftarrow$ (deptno, job)
	30 MANAGER	2850	$\Leftarrow$ (deptno, job)
	30 SALESMAN	5600	$\Leftarrow$ (deptno, job)
7566		6000	$\Leftarrow$ (mgr)
7698		6550	$\Leftarrow$ (mgr)
7782		1300	$\Leftarrow$ (mgr)
7788		1100	$\Leftarrow$ (mgr)
7839		8275	$\Leftarrow$ (mgr)
7902		800	$\Leftarrow$ (mgr)
		5000	$\Leftarrow$ (mgr)

26 sor kijelölve.



#### Megjegyzés

Láthatóan ez a legáttekinthetőbb módja a csoportfüggvények részértéke meghatározásának.

## Az SQL analitikus függvényei

Az analitikus függvények mind az elérhető funkciókban, mind az adattáblák szerkezetének kezelésében új szemléletet jelentenek, új függvénycsaládot alkotnak. Egyszerűt a gazdaságban széles körben alkalmazott kiértékelő függvények, statisztikai számítások széles skáláját tartalmazzák, másrészről pedig például az adott tulajdonság szerint csoportosított sorok által alkotott táblatartományok belsejében mozgó „ablakon” végzett részösszegképzés új lehetőségeket nyitott az adatfeldolgozás előtt. Ezek a függvények jelenleg még nincsenek a standard SQL nyelvben, de ezeké a jövő, igénylik a felhasználók, ezért szabványosítás alatt állnak.

A bemutatásra kerülő analitikus függvényeket három csoportba osztva ismertetjük. Ezek a csoportok a *rang*, a *statisztikai* és a *hisztogram* függvények csoportjai. Egyes függvények definíció szerint ugyan nem analitikusak, ám a többihez hasonló funkciójuk, vagy egyes analitikus függvényeket a használat során jól kiegészítő jellegük miatt mégis itt ismertetjük őket. Ezekre a tulajdonságukra bemutatásuk során külön ki fogunk térni.

## Az analitikus függvények általános felépítése

*Megadás*

```
analitikus_függvény_neve( [ paraméter ] ) OVER ( analitikus_záradék )
```

*Értelmezés (munkatábla, paraméter)*

Az analitikus függvények egy SELECT utasításban kizárolag a szelekciós listában szerepelhetnek (allekérdezésként természetesen már bárhol). Ennek megfelelően azon a táblaterületen működnek, melyeket a SELECT utasítás egyéb utasításrészei (FROM, WHERE, GROUP BY, HAVING) a számára rendelkezésre bocsátanak. Ezt a táblaterületet az analitikus függvények *munkatáblájának* nevezzük.

Egy analitikus függvény paramétere tetszőleges oszlopkifejezés lehet (hasonlóan a csoportfüggvényhez), mely egymással kompatibilis oszlopneveket, konstansokat, műveleteket és sorfüggvényeket tartalmazhat.

A pontosabb értelmezést az egyes részek bemutatásánál adjuk meg. (Részletes leírás az Oracle-rendszer Help funkciója segítségével kapható (magyar nyelven vázlatos ismertetést lásd [15].)



### Megjegyzés

Az analitikus függvények bizonyos értelemben a csoportfüggvények kiterjesztésének tekinthetők, bár elég jelentős fogalmi módosulással. Tény minden esetben, hogy ha egy analitikus függvény neve szerepel a csoportfüggvények között, akkor az (legalábbis szintaktikailag) előállítható az OVER opció elhagyásával.

## AZ ANALITIKUS ZÁRADÉK

*Megadás*

```
[ [ partíció-tag ] [ rendező-tag [ ablak-tag ] ] ]
```

*Értelmezés (munkaterület)*

Az analitikus záradék az analitikus függvény munkatáblájának munkaterületét jelöli ki. Értelmezését az alábbiakban tagonként adjuk meg. A szintaktikai leírásból láthatóan az analitikus záradék teljesen el is maradhat. Ekkor a függvény értelmezési tartománya a teljes munkatábla.



### Megjegyzés

Az analitikus záradék hiánya egy üres, azaz OVER() alakú OVER opciót eredményez, mely nem azonos az OVER opció hiányával. Ez utóbbi – mint feljebb már jelezük – (formailag legalábbis) csoportfüggvényé alakítaná az analitikus függvényt.

### A partíció-tag

*Megadás*

```
PARTITION BY oszlopkifejezések_listája
```

### **Értelmezés (partíció)**

A partíció-tag segítségével a munkatábla munkaterületén olyan összefüggő táblatartományokat (azaz partíciókat) lehet képezni, ahol egy partícion belül az oszlopkifejezések értéke azonos.

Ez egrészt egy rendezést jelent az oszlopkifejezés szerint. Ha az oszlopkifejezés-lista több-elemű, akkor az abban megadott sorrend szerint történik elsődleges, másodlagos stb. rendezés.

Másrészt egy-egy partíció egymást követő sorai jelölik ki az analitikus függvény egy-egy értelmezési tartományát, vagyis az analitikus függvény minden partícion újra és újra kiértékelésre kerül.

Megjegyezzük, hogy a partícióképzés láthatóan nem azonos a GROUP BY utasításrész által végzett csoportképzéssel. A csoportfüggvények minden csoporthoz csak egyetlen értéket rendelnek, míg az analitikus függvények a partíciók minden sorához rendelnek értéket. A csoportok elemi sorait csak a csoportképző oszlopok értékei és a nem csoportképző oszlopokon kiértékeltek csoportfüggvények értékei reprezentálják, maguk az elemi sorok, valamint a további oszlopok „eltűnnek”.

### **A rendező-tag**

Megadás

ORDER BY *oszlopkifejezések\_listája* [ ASC | DESC ]

### **Értelmezés (partíciókon belüli rendezés)**

A rendező-tag minden egyes partícion belül rendez a benne megadott oszlopkifejezések szerint. Sokszor azonos az analitikus függvény paramére és a rendező-tagban megadott oszlop. Ilyenkor egy partícion belüli rendezés és az analitikus függvényérték (például kumulált összeg) jól látható kapcsolatban áll egymással, más esetben ez a kapcsolat rejttebb (lásd 13.15. és 13.16. példa).

Egyes analitikus függvényeknél (például a rang függvények analitikus alakjánál) hiányzik a paraméter megadása. Ekkor a függvény paramétere tulajdonképpen nem más, mint a rendező-tagban megadott oszlop (lásd 13.11. példa).

### **A ablak-tag**

Az ablak-tag kijelöli minden partícion belül azt az (úgynevezett aktuális sorhoz rögzített méretű, és a partíció megadott résztartományán folyamatosan mozgó) összefüggő táblatartományt, melynek sorain történik az analitikus függvény által kijelölt műveletvégzés (például kumulált összegzés – lásd 13.17. példa).

Az ablak-tag használata esetén az analitikus függvény lényegében egy olyan csoportfüggvény, melynek csoportképző attribútuma a paraméterében megadott oszlopkifejezés, hatóköre az ablak. A függvényértéket az aktuális sorhoz rendeli.

Az ablaktartomány kijelölése lehet fizikai (ROWS), illetve logikai (RANGE).

### **A fizikai tartományt kijelölő ablak-tag megadása**

ROWS BETWEEN *kifejezés1* PRECEDING AND *kifejezés2* FOLLOWING

ahol *kifejezés1* megadja azt, hogy az ablak a partícióján belül az aktuális sor (CURRENT ROW) előtt hányszorral kezdődjön, a *kifejezés2* pedig azt, hogy utána hányszor tartson.

A logikai tartományt kijelölő ablak-tag megadása

RANGE BETWEEN [UNBOUNDED PRECEDING | CURRENT ROW] AND  
[CURRENT ROW | UNBOUNDED FOLLOWING]

vagy

RANGE [UNBOUNDED PRECEDING | CURRENT ROW]

ahol az egyes kulcskifejezések jelentése:

- UNBOUNDED PRECEDING: a partíció első sorától kezdődően,
- UNBOUNDED FOLLOWING: a partíció utolsó soráig,
- CURRENT ROW: a partíció aktuális sorától, illetve soráig.

## Rang függvények

### AZ ANALITIKUS RANG FÜGGVÉNYEK

Az analitikus rang függvények az OVER opciójával ellátott RANK, DENSE\_RANK és PERCENT\_RANK függvények. Jellegzetességük, hogy nem tartalmaznak ablak-tagot, és paraméterlistájuk üres. Tulajdonképpen a rendező tagban szereplő oszlopkifejezés a paraméterük, aszerint végeznék rangsorolást a kijelölt partíciókban. Hiányzó partíció-tag esetén a rangsorolás az egész táblára vonatkozik.

A RANK függvény az 1-től a partíció méretéig terjedő számtartományban sorszámozza (azaz látja el „rang”-gal) a partíció elemeit alkotó sorokat oly módon, hogy az azonos szinten levőknek ugyanazt a rangot adja, a következő szinten álló pedig annyival nagyobb rangsorbeli számot kap, ahány egyforma rangértékű talált az előző szinten. (Tekintsük a 13.8. példát. Martin és Ward rangja egyaránt 4, de a rangsorban a következő, Miller rangja már 6. Az 5-ös rangérték tehát kímaradt.) Szemléletesen szólva, e függvény esetén sor  $n$  rangértéke azt jelenti, hogy  $(n-1)$  darab sor előzi meg a rangsorban.

A DENSE\_RANK függvény működése hasonlít a RANK függvényhez, ám ez “tömöríti” a sorszámozást. Az előző példánál maradva, ez a függvény már kiosztja az 5-ös rangértéket. Szemléletesen, a függvény esetén az  $n$ -edik rangú sor a minősítés szempontjából az  $n$ -edik, bár lehet, hogy  $(n-1)$ -nél több sor előzi meg a rangsorban.

Mind a RANK, mind a DENSE\_RANK függvény esetén megállapítható, hogy az általuk visszaadott rangszámok elvileg másat jelentenek, mint a korábban már megismert ROWNUM függvény (pszeudooszlop). Ez utóbbi egyáltalán nem tudott mit kezdeni az azonos minősítésű sorokkal, gyakorlatilag a tárolási sorrendtől függően adott az egyiknek alacsonyabb, a másiknak magasabb sorszámot.

A PERCENT\_RANK függvény a rangértékeket a  $[0..1]$  zárt tartománybeli értékként adja meg a következőképpen: ha egy sor rangja a RANK függvény szerint  $n$ , és a partíció (mely tartalmazza)  $k$  sorból áll, akkor a rangja  $(n-1)/(k-1)$  a PERCENT\_RANK függvény szerint. Láthatóan a legkisebb kiosztott rangérték e függvény szerint a 0, ám a legnagyobb nem feltétlenül az 1.

*Az analitikus rang függvények használatának szintaktikája:*

```
RANK() OVER ([PartíciósTag]
    ORDER BY oszlopkifejezés1 [DESC | ASC]
    [, oszlopkifejezés2 [DESC | ASC] ...])
```

```
DENSE_RANK() OVER ([PartíciósTag]
    ORDER BY oszlopkifejezés1 [DESC | ASC]
    [, oszlopkifejezés2 [DESC | ASC] ...])
```

```
PERCENT_RANK() OVER ([PartíciósTag]
    ORDER BY oszlopkifejezés1 [DESC | ASC]
    [, oszlopkifejezés2 [DESC | ASC] ...])
```

## **AZ AGGREGÁLÓ RANG FÜGGVÉNYEK**

Az OVER opció helyett a WITHIN GROUP opcióval ellátott rang függvényeket aggregáló rang függvényeknek nevezzük. Valójában nem analitikus függvények, de a működésük mögött az analitikus rang függvények állnak. Ezek paraméterlistája már nem üres.

Egy aggregáló rang függvény a paramétereként megadott konstans kifejezés értékének helyét keresi meg az egyes partíciókban (vagy az egész munkatáblában) annak feltételezésével, hogy e konstans érték maga is az egyes partíciók (vagy az egész munkatábla) eleme. (Pontosabban azt feltételezve, hogy e konstans az egyes partíciókat alkotó sorokra vonatkozóan a rendező-tag oszlopkifejezése értéktartományának eleme.) Látható tehát, hogy e függvény a „*mi lenne, ha...*” típusú kérdések megválaszolására alkalmas, és működése során felhasználja az analitikus rangfüggvényt.

Használatuk jellegzetessége, hogy mivel nem analitikus függvények, ezért az analitikus függvények partícióstruktúráját a GROUP BY csoportosító utasításrésszel kell „pótolni”. Jól szemléltetik mindezt a 13.9. és 13.10. példa, ahol az első, megadott partíciókon végez rangsorolást, az utóbbi pedig azt adja meg, hogy az egyes partíciókon belül egy hipotetikus érték milyen rangsorú lenne. A partíciókat itt egy GROUP BY utasításrész adja meg.

További jellegzetességek, hogy a fentiekből következően az aggregáló rang függvények már csak azért sem tartalmazhatnak ablak-tagot, mert nem analitikusak.

Megjegyezzük, hogy a konstans paraméterkifejezések és a rendező-tagban lévő oszlopkifejezések sorrendjének és típusának nyilván egyeznie kell.

*Az aggregáló rang függvények használatának szintaktikája*

```
RANK(kifejezés1 [, kifejezés2]...) WITHIN GROUP
    (ORDER BY oszlopkifejezés1 [DESC | ASC]
    [, oszlopkifejezés2 [DESC | ASC] ...])
```

```
DENSE_RANK(kifejezés1 [, kifejezés2]...) WITHIN GROUP
    (ORDER BY oszlopkifejezés1 [DESC | ASC]
    [, oszlopkifejezés2 [DESC | ASC] ...])
```

```
PERCENT_RANK(kifejezés1 [, kifejezés2]...) WITHIN GROUP
  (ORDER BY oszlopkifejezés1 [DESC | ASC]
   [, oszlopkifejezés2 [DESC | ASC] ]...)
```

## 13.8. példa

Készítsen szkript programot, mely előállítja a dolgozók fizetési rangsorát, valamint megállapítja, hogy egy, a felhasználó által megadott fizetési érték hányadik lenne a feltételezett rangsorban.

### Megoldás

```
-- Szkript program az analitikus és az aggregáló
-- rang függvények szemléltetésére
COLUMN ename          FORMAT A7
COLUMN sal            FORMAT 9999
COLUMN "SzázalékosRangsor" FORMAT 0.99

-- Analitikus
SELECT ename,
       sal,
       RANK() OVER (ORDER BY sal DESC)           AS "NormálRangsor",
       DENSE_RANK() OVER (ORDER BY sal DESC)      AS "TömörítettRangsor",
       PERCENT_RANK() OVER (ORDER BY sal DESC)    AS "SzázalékosRangsor"
  FROM emp;

-- Aggregáló
ACCEPT fiz PROMPT 'Fizetés: '
SELECT UPPER('Hányadik?')           AS "Kérdés",
       RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
                           AS "NormálRangsor",
       DENSE_RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
                           AS "TömörítettRangsor",
       PERCENT_RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
                           AS "SzázalékosRangsor"
  FROM emp;
CLEAR COLUMNS
```

### Eredmény

ENAME	SAL	NormálRangsor	TömörítettRangsor	SzázalékosRangsor
KING	5000	1	1	0.00
FORD	3000	2	2	0.08
SCOTT	3000	2	2	0.08
JONES	2975	4	3	0.23
BLAKE	2850	5	4	0.31

CLARK	2450	6	5	0.38
ALLEN	1600	7	6	0.46
TURNER	1500	8	7	0.54
MILLER	1300	9	8	0.62
MARTIN	1250	10	9	0.69
WARD	1250	10	9	0.69
ADAMS	1100	12	10	0.85
JAMES	950	13	11	0.92
SMITH	800	14	12	1.00

14 sor kijelölve.

Fizetés: 4000

régi 2:	RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
új 2:	RANK(4000) WITHIN GROUP (ORDER BY sal DESC)
régi 4:	DENSE_RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
új 4:	DENSE_RANK(4000) WITHIN GROUP (ORDER BY sal DESC)
régi 6:	PERCENT_RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
új 6:	PERCENT_RANK(4000) WITHIN GROUP (ORDER BY sal DESC)

Kérdés NormálRangsor TömörítettRangsor SzázalékosRangsor

HÁNYADIK? 2 2 0.07



### Megjegyzés.

A „hányadik?” kérdésre adott válaszok megadják az eredménytábla egyes rangsorai-  
ban (oszlopaiban), hogy a felhasználó által megadott fizetésű dolgozó adatainak az  
emp táblába való feltételezett beillesztése után e dolgozó az új rangsorokban hányadik lenne.

## 13.9. példa

Készítse el a dolgozók részlegenkénti rangsorát elsődlegesen a fizetés, másodlagosan a munka-  
kör szerint rendezve.

### Megoldás

COLUMN ename	FORMAT A7
COLUMN sal	FORMAT 9999
COLUMN deptno	FORMAT 99

```

SELECT deptno, ename, sal, job,
       RANK() OVER (PARTITION BY deptno
                     ORDER BY sal DESC, job) AS "NormálRangsor",
       DENSE_RANK() OVER (PARTITION BY deptno
                     ORDER BY sal DESC, job) AS "TömörRangsor"
FROM emp;
    
```

CLEAR COLUMNS

**Eredmény**

DEPTNO	ENAME	SAL	JOB	NormálRangsor	TömörRangsor
10	KING	5000	PRESIDENT	1	1
10	CLARK	2450	MANAGER	2	2
10	MILLER	1300	CLERK	3	3
20	FORD	3000	ANALYST	1	1
20	SCOTT	3000	ANALYST	1	1
20	JONES	2975	MANAGER	3	2
20	ADAMS	1100	CLERK	4	3
20	SMITH	800	CLERK	5	4
30	BLAKE	2850	MANAGER	1	1
30	ALLEN	1600	SALESMAN	2	2
30	TURNER	1500	SALESMAN	3	3
30	MARTIN	1250	SALESMAN	4	4
30	WARD	1250	SALESMAN	4	4
30	JAMES	950	CLERK	6	5

14 sor kijelölve.

**13.10. példa**

Állapítsa meg, hogy a felhasználó által megadott fizetésérték hányadik lenne a fizetések részlegenkénti feltételezett rangsorában.

**Megoldás**

ACCEPT fiz PROMPT 'Fizetés:'

```
COLUMN SUM(sal)          FORMAT 99999
COLUMN deptno            FORMAT 99
```

```
SELECT &fiz                  AS "MegadottFizetés",
       deptno,
       SUM(sal),
       RANK(&fiz) WITHIN GROUP(ORDER BY sal)      AS "NormálRangsor",
       DENSE_RANK(&fiz) WITHIN GROUP(ORDER BY sal) AS "TömörRangsor"
  FROM emp
 GROUP BY deptno;
```

CLEAR COLUMNS

**Eredmény**

Fizetés:1500 AS  
 régi 1: SELECT &fiz AS  
 "MegadottFizetés",  
 új 1: SELECT 1500 AS  
 "MegadottFizetés",  
 régi 4: RANK(&fiz) WITHIN GROUP(ORDER BY sal) AS  
 "NormálRangsor",  
 új 4: RANK(1500) WITHIN GROUP(ORDER BY sal) AS  
 "NormálRangsor",  
 régi 5: DENSE\_RANK(&fiz) WITHIN GROUP(ORDER BY sal) AS  
 "TömörRangsor"  
 új 5: DENSE\_RANK(1500) WITHIN GROUP(ORDER BY sal) AS  
 "TömörRangsor"

MegadottFizetés DEPTNO SUM(SAL) NormálRangsor TömörRangsor

1500	10	8750	2	2
1500	20	10875	3	3
1500	30	9400	4	3

**Megjegyzés**

- Az eredménytábla rangsor oszlopaiban szereplő értékek azt adják meg, hogy a felhasználó által megadott fizetésű dolgozó adatainak az emp táblába való feltételezett beillesztése után ez a dolgozó az egyes részlegek új rangsoraiban hányadik lenne.
- E feladat megoldásának helyességét az előző példa eredménytáblája alapján ellenőrizhetjük.

**13.11. példa**

Határozza meg a Dallas-i és a New York-i dolgozók fizetési sorrendjét.

**1. megoldás**

```
BREAK ON deptno  ON loc
SELECT
  e.deptno,
  loc,
  ename,
  sal,
  DENSE_RANK() OVER(PARTITION BY e.deptno ORDER BY sal DESC)
    AS sorrend
FROM emp e,
     dept d
```

```

WHERE e.deptno = d.deptno AND
      UPPER(loc) IN ('DALLAS', 'NEW YORK')
ORDER BY d.deptno, ename;
CLEAR BREAK

```

## 2. megoldás

```

BREAK ON deptno  ON loc
SELECT
    e.deptno,
    loc,
    ename,
    sal,
    DENSE_RANK() OVER(PARTITION BY e.deptno, ename ORDER BY sal DESC)
        AS sorrend
FROM emp e,
     dept d
WHERE e.deptno = d.deptno AND
      UPPER(loc) IN ('DALLAS', 'NEW YORK');
CLEAR BREAK

```

### Eredmény (mindkét megoldás esetén)

DEPTNO	LOC	ENAME	SAL	SORREND
10	NEW YORK	CLARK	2450	2
		KING	5000	1
		MILLER	1300	3
20	DALLAS	ADAMS	1100	3
		FORD	3000	1
		JONES	2975	2
		SCOTT	3000	1
		SMITH	800	4

8 sor kijelölve.

## 13.12. példa

Határozza meg előbb részlegenként és azon belül munkakörönként, majd munkakörönként és azon belül részlegenként a dolgozók fizetési sorrendjét.

### Megoldás

```

BREAK ON deptno  ON job
SELECT
    deptno,
    job,

```

```

ename,
sal,
RANK() OVER (PARTITION BY deptno, job ORDER BY sal DESC)
          AS NormálRangsor,
DENSE_RANK() OVER (PARTITION BY deptno, job ORDER BY sal DESC)
          AS TömörRangsor
FROM emp;

BREAK ON job  ON deptno
SELECT
job,
deptno,
ename,
sal,
RANK() OVER (PARTITION BY job, deptno ORDER BY sal DESC)
          AS NormálRangsor,
DENSE_RANK() OVER (PARTITION BY job, deptno ORDER BY sal DESC)
          AS TömörRangsor
FROM emp;
CLEAR BREAK

```

**Eredmény**

DEPTNO	JOB	ENAME	SAL	NORMÁLRANGSOR	TÖMÖRRANGSOR
10	CLERK	MILLER	1300	1	1
	MANAGER	CLARK	2450	1	1
	PRESIDENT	KING	5000	1	1
20	ANALYST	FORD	3000	1	1
		SCOTT	3000	1	1
	CLERK	ADAMS	1100	1	1
		SMITH	800	2	2
	MANAGER	JONES	2975	1	1
30	CLERK	JAMES	950	1	1
	MANAGER	BLAKE	2850	1	1
	SALESMAN	ALLEN	1600	1	1
		TURNER	1500	2	2
		MARTIN	1250	3	3
		WARD	1250	3	3

14 sor kijelölve.

JOB	DEPTNO	ENAME	SAL	NORMÁLRANGSOR	TÖMÖRRANGSOR
ANALYST	20	FORD	3000	1	1
		SCOTT	3000	1	1

CLERK	10 MILLER	1300	1	1
	20 ADAMS	1100	1	1
	SMITH	800	2	2
	30 JAMES	950	1	1
MANAGER	10 CLARK	2450	1	1
	20 JONES	2975	1	1
	30 BLAKE	2850	1	1
PRESIDENT	10 KING	5000	1	1
SALESMAN	30 ALLEN	1600	1	1
	TURNER	1500	2	2
	MARTIN	1250	3	3
	WARD	1250	3	3

14 sor kijelölve.



### Megjegyzés

Láthatóan a kétféle particionálás (éppen azért, mert azonos részletezettségű felosztást eredményezett) az egyes dolgozókat ugyanazokkal hozta közös rangsorba és minden két particionálásban azonos ranggal.

### 13.13. példa

Határozza meg előbb részlegenként és azon belül munkakörönként, majd csak részlegenként a dolgozók fizetési sorrendjét. A lista legyen elsődlegesen a részlegazonosító, másodlagosan a munkakör, harmadlagosan a dolgozók neve szerint rendezve.

#### Megoldás

```

BREAK ON deptno  ON job
SELECT
    deptno,
    job,
    ename,
    sal,
    RANK() OVER (PARTITION BY deptno, job ORDER BY sal DESC)
                  AS NormálRangsor,
    DENSE_RANK() OVER (PARTITION BY deptno, job ORDER BY sal DESC)
                  AS TömörRangsor
FROM emp
ORDER BY deptno, job, ename;

SELECT
    deptno,
    job,
    ename,
```

```

ename,
sal,
RANK() OVER (PARTITION BY deptno ORDER BY sal DESC)
          AS NormálRangsor,
DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal DESC)
          AS TömörRangsor
FROM emp
ORDER BY deptno, job, ename;
CLEAR BREAK

```

**Eredmény**

DEPTNO	JOB	ENAME	SAL	NORMÁLRANGSOR	TÖMÖRRANGSOR
10	CLERK	MILLER	1300	1	1
	MANAGER	CLARK	2450	1	1
	PRESIDENT	KING	5000	1	1
20	ANALYST	FORD	3000	1	1
		SCOTT	3000	1	1
	CLERK	ADAMS	1100	1	1
		SMITH	800	2	2
	MANAGER	JONES	2975	1	1
30	CLERK	JAMES	950	1	1
	MANAGER	BLAKE	2850	1	1
	SALESMAN	ALLEN	1600	1	1
		MARTIN	1250	3	3
		TURNER	1500	2	2
		WARD	1250	3	3

14 sor kijelölve.

DEPTNO	JOB	ENAME	SAL	NORMÁLRANGSOR	TÖMÖRRANGSOR
10	CLERK	MILLER	1300	3	3
	MANAGER	CLARK	2450	2	2
	PRESIDENT	KING	5000	1	1
20	ANALYST	FORD	3000	1	1
		SCOTT	3000	1	1
	CLERK	ADAMS	1100	4	3
		SMITH	800	5	4
	MANAGER	JONES	2975	3	2
30	CLERK	JAMES	950	6	5
	MANAGER	BLAKE	2850	1	1
	SALESMAN	ALLEN	1600	2	2
		MARTIN	1250	4	4

TURNER	1500	3	3
WARD	1250	4	4

14 sor kijelölve.



### Megjegyzés

- Tekintsük először a rangsorértékeket. Látható, hogy a két lista – bár felépítésük azonos, mégis különböző rangsorértékeket rendel az egyes dolgozókhoz. Ennek oka a particionálás különböző részletezettségében keresendő. A gyakorlat számára feltehetően a második lista a használhatóbb. A tanulság az, hogy analitikus függvények használatánál adott részletezettségű lista előállításához elegendő kisebb részletezettségű particionálást alkalmazni, mivel előfordulhat, hogy a túlzott részletezettségű particionálás már semmitmondó lesz.
- Figyeljünk fel a rendezésre is. Mindkét listát azonos rendező-utasításrésszel láttuk el, ennek megfelelően azonos sorrendű sorokat kaptunk (itt most csak az első négy oszlopot tekintjük). Ha azonban összehasonlítjuk e példa első listáját az előző példa első listájával, akkor azt láthatjuk, hogy azok között csak a részleg-munkakör partíciókon belüli nevek sorrendjében van különbség, noha az első példa megoldásában nem is alkalmaztunk rendező utasításrészt. Ennek oka az (amint az analitikus függvények partíció-tagjának ismertetésénél már utaltunk rá), hogy a particionáló oszlopkejezések szerint már történik egy rendezés. Feltehető a kérdés, vajon nem lett volna-e egyszerűbb, ha a név szerinti rendezést az első listát generáló utasításban nem a rendező utasításrészbe tesszük, hanem az analitikus függvény rendező-tagjába ORDER BY ename, sal..., vagy ORDER BY sal, ename... módon. Csak hogy akkor a rangsor meghatározása már nem csupán a fizetésre, hanem a név–fizetés, vagy a fizetés–név párosra történt volna!

## 13.14. példa

Állítsa elő a dolgozók fizetésének részlegenkénti, majd a teljes vállalatra vonatkozó halmozott összegeit. (Az ilyen összegképzést kumulált összegnek is nevezik.)

### Megoldás

COLUMN ename	FORMAT A7
COLUMN sal	FORMAT 9999
COLUMN deptno	FORMAT 99

PROMPT Részlegösszegek kiíratása dolgozónként:

```

SELECT ename, emp.deptno, sal, RészlegÖsszeg
  FROM emp,
       (SELECT deptno,
                SUM(sal) RészlegÖsszeg
      FROM emp
     GROUP BY deptno) Részleg
  
```

```
WHERE emp.deptno = Részleg.deptno
ORDER BY deptno, sal;
```

PROMPT Részlegösszegek kiíratása gyűjtősortba:

```
SELECT deptno, ename, SUM(sal)
FROM emp
GROUP BY ROLLUP (deptno, ename)
ORDER by deptno, SUM(sal);
```

PROMPT Halmozott részlegösszegek kiíratása:

```
SELECT deptno,
ename,
sal,
RANK() OVER (PARTITION BY deptno ORDER BY sal ASC)
AS "NormálRangsor",
DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal ASC)
AS "TömörRangsor",
SUM(SAL) OVER (PARTITION BY deptno ORDER BY sal ASC)
AS "RészlegreHalmozottÖsszeg"
FROM emp;
```

PROMPT Halmozott összegek kiíratása a teljes vállalatra:

```
SELECT ename,
sal,
RANK() OVER (ORDER BY sal ASC)
AS "NormálRangsor",
DENSE_RANK() OVER (ORDER BY sal ASC)
AS "TömörRangsor",
SUM(SAL) OVER (ORDER BY sal ASC)
AS "HalmozottÖsszeg"
FROM emp;
```

CLEAR COLUMNS

### Eredmény

Részlegösszegek kiíratása dolgozónként:

ENAME	DEPTNO	SAL	RÉSZLEGÖSSZEG
MILLER	10	1300	8750
CLARK	10	2450	8750
KING	10	5000	8750
SMITH	20	800	10875
ADAMS	20	1100	10875
JONES	20	2975	10875

SCOTT	20	3000	10875
FORD	20	3000	10875
JAMES	30	950	9400
MARTIN	30	1250	9400
WARD	30	1250	9400
TURNER	30	1500	9400
ALLEN	30	1600	9400
BLAKE	30	2850	9400

14 sor kijelölve.

Részlegösszegek kiíratása gyűjtősorba:

DEPTNO ENAME SUM(SAL)

10 MILLER	1300
10 CLARK	2450
10 KING	5000
10	8750
20 SMITH	800
20 ADAMS	1100
20 JONES	2975
20 FORD	3000
20 SCOTT	3000
20	10875
30 JAMES	950
30 WARD	1250
30 MARTIN	1250
30 TURNER	1500
30 ALLEN	1600
30 BLAKE	2850
30	9400
	29025

18 sor kijelölve.

Halmozott részlegösszegek kiíratása:

DEPTNO ENAME SAL NormálRangsor TömörRangsor RészlegreHalmozottÖsszeg

10 MILLER	1300	1	1	1300
10 CLARK	2450	2	2	3750
10 KING	5000	3	3	8750
20 SMITH	800	1	1	800
20 ADAMS	1100	2	2	1900
20 JONES	2975	3	3	4875

20	FORD	3000	4	4	10875
20	SCOTT	3000	4	4	10875
30	JAMES	950	1	1	950
30	MARTIN	1250	2	2	3450
30	WARD	1250	2	2	3450
30	TURNER	1500	4	3	4950
30	ALLEN	1600	5	4	6550
30	BLAKE	2850	6	5	9400

14 sor kijelölve.

Halmozott összegek kiíratása a teljes vállalatra:

ENAME	SAL	NormálRangsor	TömörRangsor	HalmozottÖsszeg
-------	-----	---------------	--------------	-----------------

SMITH	800	1	1	800
JAMES	950	2	2	1750
ADAMS	1100	3	3	2850
MARTIN	1250	4	4	5350
WARD	1250	4	4	5350
MILLER	1300	6	5	6650
TURNER	1500	7	6	8150
ALLEN	1600	8	7	9750
CLARK	2450	9	8	12200
BLAKE	2850	10	9	15050
JONES	2975	11	10	18025
FORD	3000	12	11	24025
SCOTT	3000	12	11	24025
KING	5000	14	12	29025

14 sor kijelölve.

## Statisztikai függvények

A hagyományos statisztikai függvényeknek (SUM, AVG, STDDEV, VARIANCE, MIN, MAX és COUNT) szintén létezik analitikus alakja, melyeket a következő szintaktika szerint használhatunk:

függvénynév( [ DISTINCT | ALL ] paraméter ) OVER ( analitikus\_záradék )

ahol láthatóan újat az analitikus függvények eddigi definíciójához képest a DISTINCT és az ALL opciók megjelenése jelent. E kulcsszavak a korábbi (szelekciós listabeli, illetve a csoportfüggvényeknél említett) jelentésükhez hasonló módon azt eredményezik, hogy a függvény csak a munkatábla különböző, illetve minden paraméterértékét veszik figyelembe a függvényérték generálásakor (az ALL az alapértelmezett opció).

### 13.15. példa

Határozza meg a fizetésük szerint növekvően rendezett dolgozók fizetésének folyamatos halmozott összegét, az első sortól az utolsóig. (Ezt kumulált összegnek is nevezzük.)

#### Megoldás

```
SELECT empno, ename, sal,
       SUM(sal) OVER
          (ORDER BY sal
             RANGE UNBOUNDED PRECEDING)
             AS "Kumulált",
       SUM(sal) OVER
          (ORDER BY sal
             RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
             AS "Ugyanaz"
FROM emp;
```

#### Eredmény

EMPNO	ENAME	SAL	Kumulált	Ugyanaz
7369	SMITH	800	800	800
7900	JAMES	950	1750	1750
7876	ADAMS	1100	2850	2850
7654	MARTIN	1250	5350	5350
7521	WARD	1250	5350	5350
7934	MILLER	1300	6650	6650
7844	TURNER	1500	8150	8150
7499	ALLEN	1600	9750	9750
7782	CLARK	2450	12200	12200
7698	BLAKE	2850	15050	15050
7566	JONES	2975	18025	18025
7902	FORD	3000	24025	24025
7788	SCOTT	3000	24025	24025
7839	KING	5000	29025	29025

14 sor kijelölve.



#### Megjegyzés

- Az eredmény első sorában az első dolgozó fizetése (mint halmozott összeg), a másodikban az első és a második fizetés összege és így tovább. Az utolsó sorban tehát az összes dolgozó összfizetése található.
- Az azonos értékeket egyszerre adja össze, ha közvetlenül egymás mellett állnak (lásd WARD és MARTIN sorait).

### 13.16. példa

Határozza meg a névsor szerint növekvően rendezett dolgozók fizetéseinek részlegükön belüli halmozott (kumulált) összegét.

#### Megoldás

```
SELECT deptno, ename, sal,
       SUM(sal) OVER
          (PARTITION BY deptno ORDER BY ename
           RANGE UNBOUNDED PRECEDING)
          AS "KumuláltÖsszeg",
       SUM(sal) OVER
          (PARTITION BY deptno ORDER BY ename
           RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
          AS "Ugyanaz"
FROM emp;
```

#### Eredmény

DEPTNO	ENAME	SAL	KumuláltÖsszeg	Ugyanaz
10	CLARK	2450	2450	2450
10	KING	5000	7450	7450
10	MILLER	1300	8750	8750
20	ADAMS	1100	1100	1100
20	FORD	3000	4100	4100
20	JONES	2975	7075	7075
20	SCOTT	3000	10075	10075
20	SMITH	800	10875	10875
30	ALLEN	1600	1600	1600
30	BLAKE	2850	4450	4450
30	JAMES	950	5400	5400
30	MARTIN	1250	6650	6650
30	TURNER	1500	8150	8150
30	WARD	1250	9400	9400

14 sor kijelölve.

### 13.17. példa

Határozza meg a fizetésük szerint rendezett dolgozókhhoz az olyan mozgó „ablakbeli” halmozott összeget, mely végigmegy e rendezett tábla teljes tartományán, és az aktuális sor előtti két sort és az aktuális utáni egy sort tartalmazza (mozgó ablakban képzett összeg). A lista tartalmazza a kumulált összeget is a dolgozók fenti sorrendjében.

**Megoldás**

```

SELECT ename, sal,
       SUM(sal) OVER
          (ORDER BY sal ASC
           ROWS BETWEEN 2 PRECEDING AND 1 FOLLOWING)
           AS "AblakÖsszeg",
       SUM(sal) OVER
          (ORDER BY sal ASC)
           AS "KumuláltÖsszeg"
FROM emp;

```

**Eredmény**

ENAME	SAL	AblakÖsszeg	KumuláltÖsszeg
SMITH	800	1750	800
JAMES	950	2850	1750
ADAMS	1100	4100	2850
MARTIN	1250	4550	5350
WARD	1250	4900	5350
MILLER	1300	5300	6650
TURNER	1500	5650	8150
ALLEN	1600	6850	9750
CLARK	2450	8400	12200
BLAKE	2850	9875	15050
JONES	2975	11275	18025
FORD	3000	11825	24025
SCOTT	3000	13975	24025
KING	5000	11000	29025

14 sor kijelölve.

**Megjegyzés**

Tekintsük aktuális sorként (CURRENT ROW) például a Jones sorát, fizetése 2975. Az ablakot úgy adtuk meg a lekérdezésben, hogy az aktuális sor előtt kettő, utána pedig egy sor tartozzon bele. Az aktuális sor előtt lévő két fizetés; Clark-é 2450 és Blake-é 2850, valamint az utána következő, ez Ford-é, mely 3000. Összeadva e fizetésértékeket ( $2975 + 2450 + 2850 + 3000 = 11275$ ), valóban az aktuális sor AblakÖsszeg-beli értéket kaptuk.

**RATIO\_TO\_REPORT FÜGGVÉNY**

A RATIO\_TO\_REPORT egy speciális statisztikai függvény, kiszámítja az érték és a csoportösszeg közötti arányt. A csoport itt is lehet az egész tábla. Használata a következő szintaktika szerint történhet:

RATIO\_TO\_REPORT( *kifejezés* ) OVER ( *PartícióTag* )

ahol tehát az analitikus záradékból csak a partíció-tag szerepelhet.

### 13.18. példa

Listázza ki a dolgozók azonosítóját, nevét, fizetését, valamint azt az értéket, mely megadja, mi-  
lyen arányban részesülnek az összfizetésből.

#### 1. megoldás

```
SELECT empno, ename, sal,
       RATIO_TO_REPORT(sal) OVER()
          AS részesedés
  FROM emp ORDER BY sal desc;
```

#### Eredmény

EMPNO	ENAME	SAL	RÉSZESEDÉS
7839	KING	5000	.172265289
7788	SCOTT	3000	.103359173
7902	FORD	3000	.103359173
7566	JONES	2975	.102497847
7698	BLAKE	2850	.098191214
7782	CLARK	2450	.084409991
7499	ALLEN	1600	.055124892
7844	TURNER	1500	.051679587
7934	MILLER	1300	.044788975
7521	WARD	1250	.043066322
7654	MARTIN	1250	.043066322
7876	ADAMS	1100	.037898363
7900	JAMES	950	.032730405
7369	SMITH	800	.027562446

#### 2. megoldás (Listázás formátum megadásával)

```
SELECT empno, ename, sal,
       TO_CHAR(RATIO_TO_REPORT(sal) OVER(), '9990.999')
          AS részarány
  FROM emp
 ORDER BY sal DESC;
```

#### Eredmény

EMPNO	ENAME	SAL	RÉSZARÁNY
7839	KING	5000	0.172
7788	SCOTT	3000	0.103

7902	FORD	3000	0.103
7566	JONES	2975	0.102
7698	BLAKE	2850	0.098
7782	CLARK	2450	0.084
7499	ALLEN	1600	0.055
7844	TURNER	1500	0.052
7934	MILLER	1300	0.045
7521	WARD	1250	0.043
7654	MARTIN	1250	0.043
7876	ADAMS	1100	0.038
7900	JAMES	950	0.033
7369	SMITH	800	0.028

### 13.19. példa

Listázza ki a dolgozók munkakörét, nevét, fizetését, valamint azt az értéket, mely megadja, milyen arányban részesülnek munkakörük összfizetéséből.

#### Megoldás

```
SELECT job, ename, sal,
       TO_CHAR(RATIO_TO_REPORT(sal) OVER
              (PARTITION BY job), '99999990.999')
              AS "munkakörarány"
FROM emp;
```

#### Eredmény

JOB	ENAME	SAL	munkakörarány
ANALYST	SCOTT	3000	0.500
ANALYST	FORD	3000	0.500
CLERK	SMITH	800	0.193
CLERK	ADAMS	1100	0.265
CLERK	MILLER	1300	0.313
CLERK	JAMES	950	0.229
MANAGER	JONES	2975	0.360
MANAGER	CLARK	2450	0.296
MANAGER	BLAKE	2850	0.344
PRESIDENT	KING	5000	1.000
SALESMAN	ALLEN	1600	0.286
SALESMAN	MARTIN	1250	0.223
SALESMAN	TURNER	1500	0.268
SALESMAN	WARD	1250	0.223

14 sor kijelölve.

## SPECIÁLIS SZÉLSŐÉRTÉK-FÜGGVÉNYEK

A FIRST\_VALUE, illetve LAST\_VALUE szélsőértékek meghatározására alkalmas analitikus függvények. Tulajdonképpen speciális statisztikai függvényeknek tekinthetjük őket, melyek képesek

- nem feltétlenül numerikus értékeken is szélsőértékek meghatározására (erre még a MIN és MAX is képes), továbbá
- valamely oszlop szerint rendezett ablaktartomány valamely (esetleg másik) oszlopa első és utolsó elemének meghatározására.

Ezeket a függvényeket a következő szintaktika szerint használhatjuk:

*függvénynév( kifejezés ) OVER ( analitikus utasításrész )*

Ez a két analitikus függvény nem ágyazható be más analitikus függvények kifejezés részébe.

### 13.20. példa

Listázza ki dolgozónként a nevüket, a fizetésüket, a részlegazonosítójukat, valamint részlegük legkisebb és legnagyobb fizetését, elsődlegesen a részlegazonosító, másodlagosan a fizetés szerint rendezve.

#### 1. megoldás (A korábbi ismereteink alapján, inline nézettel)

```
SELECT ename, sal, al.*  
  FROM emp,  
        (SELECT deptno,  
              MIN(sal) AS legkisebb,  
              MAX(sal) AS legnagyobb  
        FROM emp  
        GROUP BY deptno) al  
 WHERE emp.deptno = al.deptno  
 ORDER BY al.deptno, sal;
```

#### 2. megoldás (A FIRST\_VALUE és LAST\_VALUE függvényekkel)

```
SELECT ename, sal, deptno,  
      FIRST_VALUE(sal) OVER  
        (PARTITION BY deptno  
        ORDER BY sal ASC  
        RANGE BETWEEN UNBOUNDED PRECEDING AND  
                  UNBOUNDED FOLLOWING)  
          AS legkisebb,  
      LAST_VALUE(sal) OVER  
        (PARTITION BY deptno  
        ORDER BY sal ASC  
        RANGE BETWEEN UNBOUNDED PRECEDING AND
```

```

        UNBOUNDED FOLLOWING)
        AS legnagyobb
FROM emp;

```

**Eredmény (mindkét esetben)**

ENAME	SAL	DEPTNO	LEGKISEBB	LEGNAGYOBBA
MILLER	1300	10	1300	5000
CLARK	2450	10	1300	5000
KING	5000	10	1300	5000
SMITH	800	20	800	3000
ADAMS	1100	20	800	3000
JONES	2975	20	800	3000
SCOTT	3000	20	800	3000
FORD	3000	20	800	3000
JAMES	950	30	950	2850
WARD	1250	30	950	2850
MARTIN	1250	30	950	2850
TURNER	1500	30	950	2850
ALLEN	1600	30	950	2850
BLAKE	2850	30	950	2850

14 sor kijelölve.

**Megjegyzés**

Ha a fenti megoldásban fordított irányban rendezünk (ASC helyett DESC kulcsszót írva), akkor természetesen a FIRST\_VALUE, illetve a LAST\_VALUE függvények a legnagyobb, illetve legkisebb értékeket adják.

**13.21. példa**

Listázza ki dolgozónként a nevüket, a fizetésüket, a részlegazonosítójukat, valamint részlegüknek az ábécé szerinti első és utolsó dolgozója nevét elsődlegesen a részlegazonosító, másodlagosan a név szerint rendezve.

**1. megoldás (A korábbi ismereteink alapján, inline nézettel)**

```

SELECT ename, sal, al.*
  FROM emp,
       (SELECT deptno,
               MIN(ename) AS legkisebb,
               MAX(ename) AS legnagyobb
        FROM emp
       GROUP BY deptno) al
 WHERE emp.deptno = al.deptno
 ORDER BY al.deptno, ename;

```

## 2. megoldás (A FIRST\_VALUE és LAST\_VALUE függvényekkel)

```

SELECT ename, sal, deptno,
       FIRST_VALUE(ename) OVER
        (PARTITION BY deptno
         ORDER BY ename ASC
         RANGE BETWEEN UNBOUNDED PRECEDING AND
                  UNBOUNDED FOLLOWING)
          AS Első,
       LAST_VALUE(ename) OVER
        (PARTITION BY deptno
         ORDER BY ename ASC
         RANGE BETWEEN UNBOUNDED PRECEDING AND
                  UNBOUNDED FOLLOWING)
          AS Utolsó
FROM emp;

```

### Eredmény (mindkét esetben)

ENAME	SAL	DEPTNO	ELŐSŐ	UTOLSÓ
CLARK	2450	10	CLARK	MILLER
KING	5000	10	CLARK	MILLER
MILLER	1300	10	CLARK	MILLER
ADAMS	1100	20	ADAMS	SMITH
FORD	3000	20	ADAMS	SMITH
JONES	2975	20	ADAMS	SMITH
SCOTT	3000	20	ADAMS	SMITH
SMITH	800	20	ADAMS	SMITH
ALLEN	1600	30	ALLEN	WARD
BLAKE	2850	30	ALLEN	WARD
JAMES	950	30	ALLEN	WARD
MARTIN	1250	30	ALLEN	WARD
TURNER	1500	30	ALLEN	WARD
WARD	1250	30	ALLEN	WARD

14 sor kijelölve.

## 13.22. példa

Listázza ki dolgozónként a nevüket, a fizetésüket, a részlegazonosítójukat, valamint részlegüknek az ábécé szerinti első és utolsó dolgozója fizetését elsődlegesen a részlegazonosító, másodlagosan a név szerint rendezve.

## Megoldás

```

SELECT ename, sal, deptno,
       FIRST_VALUE(sal) OVER
        (PARTITION BY deptno
         ORDER BY ename ASC
         RANGE BETWEEN UNBOUNDED PRECEDING AND
                  UNBOUNDED FOLLOWING)
          AS Első,
       LAST_VALUE(sal) OVER
        (PARTITION BY deptno
         ORDER BY ename ASC
         RANGE BETWEEN UNBOUNDED PRECEDING AND
                  UNBOUNDED FOLLOWING)
          AS Utolsó
FROM emp;

```

## Eredmény

ENAME	SAL	DEPTNO	ELSŐ	UTOLSÓ
CLARK	2450	10	2450	1300
KING	5000	10	2450	1300
MILLER	1300	10	2450	1300
ADAMS	1100	20	1100	800
FORD	3000	20	1100	800
JONES	2975	20	1100	800
SCOTT	3000	20	1100	800
SMITH	800	20	1100	800
ALLEN	1600	30	1600	1250
BLAKE	2850	30	1600	1250
JAMES	950	30	1600	1250
MARTIN	1250	30	1600	1250
TURNER	1500	30	1600	1250
WARD	1250	30	1600	1250

14 sor kijelölve.



### Megjegyzés

Ennek a feladatnak a hagyományos eszközökkel való megoldása már igen bonyolult lett volna.

## Hisztogramfüggvények

### WIDTH\_BUCKET FÜGGVÉNY

E függvény segítségével „azonos szélességű” hisztogram készíthető, azaz meghatározza, hogy azonos méretű értéktartományokban az oszlopkifejezés értékei hogyan oszlanak meg (vagyis egy-egy érték melyik tartományba kerül). Használatának szintaktikája:

`WIDTH_BUCKET(oszlopkifejezés, alsóhatár, felsőhatár, felosztások száma)`

ahol a függvény minden sorhoz meghatározza annak az értéktartománynak a sorszámát, melybe e sor oszlopkifejezése tartozik. Az egyes értéktartományok az oszlopkifejezés alsó- és felsőhatárai között azonos méretűek, sorszámuk értékük szerint növekvő. Számukat a felosztások száma határozza meg. Ha az oszlopkifejezés értéktartománya belesik az alsó- és felsőhatárok közötti tartományba, akkor e szám a felosztások száma, de létezhet nulladik és (felosztások száma + 1)-edik tartomány is, melyekkel az alul- és túlcordulások kezelhetők.

Megjegyezzük, hogy e függvény tulajdonképpen nem is analitikus függvény (hiányzik az OVER opció), hanem csoportfüggvény. A hisztogramfüggvények körében bemutatott másik függvény, az NTILE (lásd alább) azonban már az, hasonló funkciójuk miatt célszerű őket együtt tárgyalni.

### 13.23. példa

Listálza ki a dolgozók nevét, fizetését és a fizetési kategóriáját, ahol ez utóbbit úgy kapja meg, hogy három egyenlő részre osztja a 800..5001 értéktartományt.

#### Megoldás

```
SELECT ename, sal,
       WIDTH_BUCKET(sal,800,5001,3) AS Kategória
  FROM emp
 ORDER BY Kategória;
```

#### Eredmény

ENAME	SAL	KATEGÓRIA
MARTIN	1250	1
ALLEN	1600	1
JAMES	950	1
SMITH	800	1
ADAMS	1100	1
WARD	1250	1
TURNER	1500	1
MILLER	1300	1
BLAKE	2850	2
CLARK	2450	2

JONES	2975	2
SCOTT	3000	2
FORD	3000	2
KING	5000	3

14 sor kijelölve.

### 13.24. példa

Oldja meg az előző feladatot, ám most az 1000..4500 tartomány 3 egyenlő részre osztásával.

#### Megoldás

```
SELECT ename, sal,
       WIDTH_BUCKET(sal,1000,4500,3) AS Kategória
  FROM emp
 ORDER BY Kategória;
```

#### Eredmény

ENAME	SAL	KATEGÓRIA
SMITH	800	0
JAMES	950	0
ALLEN	1600	1
WARD	1250	1
ADAMS	1100	1
TURNER	1500	1
MARTIN	1250	1
MILLER	1300	1
JONES	2975	2
BLAKE	2850	2
CLARK	2450	2
FORD	3000	2
SCOTT	3000	2
KING	5000	4

14 sor kijelölve.



#### Megjegyzés

Láthatóan a Smith és a James lefelé, King pedig felfelé „lög ki” a kijelölt (1000..4500) tartományból.

## NTILE FÜGGVÉNY

Az NTILE függvény a WIDTH\_BUCKET függvényhez hasonlóan hisztogram készítésére használható, ám ez a hisztogram „azonos magasságú, különböző szélességű” (vagyis a különböző szélességű értéktartományok azonos számú sort tartalmaznak). Használata a következő szintaktika szerint történhet:

`NTILE(kifejezés) OVER ( [PartícióTag] RendezőTag )`

ahol a függvény hatására két esemény következik be. Egyszer történik egy rendezés a rendező tag szerint, másrészről az így rendezett lista sorait a kifejezés értékének megfelelő számú (lehetőleg azonos mennyiségű sort tartalmazó) résztartományra bontja, és ezeket 1-től kezdődő sorszámmal látja el. A függvény minden sorhoz hozzárendeli az ezt a sort tartalmazó résztartomány sorszámát.

### 13.25. példa

Listázza ki a fizetésük szerint rendezett dolgozók nevét, fizetését és fizetési kategóriáját, ahol ez utóbbi az összes dolgozó fizetési tartományának három részre osztásával úgy kapja meg, hogy minden részbe közel azonos számú dolgozó kerüljön. Az egyes kategóriák sorszáma a fizetések szerint legyen növekvő.

#### Megoldás

```
SELECT ename, sal,
       NTILE(3) OVER (ORDER BY sal) AS Kategória
    FROM emp;
```

#### Eredmény

ENAME	SAL	KATEGÓRIA
SMITH	800	1
JAMES	950	1
ADAMS	1100	1
MARTIN	1250	1
WARD	1250	1
MILLER	1300	2
TURNER	1500	2
ALLEN	1600	2
CLARK	2450	2
BLAKE	2850	2
JONES	2975	3
FORD	3000	3
SCOTT	3000	3
KING	5000	3

14 sor kijelölve.

### 13.26. példa

Listázza ki a fizetésük szerint az előző feladatbeli módon kategorizált dolgozók nevét, részleg-azonosítóját, fizetését és fizetési kategóriáját elsődlegesen a részlegazonosító, másodlagosan pedig a kategória szerint rendezve.

#### Megoldás

```
SELECT *
  FROM (SELECT ename, deptno, sal,
               NTILE(3) OVER (ORDER BY sal) AS Kategória
      FROM emp)
 ORDER BY deptno, Kategória;
```

#### Eredmény

ENAME	DEPTNO	SAL	KATEGÓRIA
MILLER	10	1300	2
CLARK	10	2450	2
KING	10	5000	3
SMITH	20	800	1
ADAMS	20	1100	1
JONES	20	2975	3
FORD	20	3000	3
SCOTT	20	3000	3
JAMES	30	950	1
MARTIN	30	1250	1
WARD	30	1250	1
TURNER	30	1500	2
BLAKE	30	2850	2
ALLEN	30	1600	2

14 sor kijelölve.

### 13.27. példa

Kategorizáljuk a dolgozókat a 13.25. példabelihez hasonló módon, de most a kategorizálási tartomány ne az összes, hanem csak az azonos munkakörű dolgozók fizetése által legyen kijelölve. A lista a munkakört, a nevet, a fizetést és a kategóriát tartalmazza.

#### Megoldás

```
SELECT job, ename, sal,
       NTILE(3) OVER (PARTITION BY job ORDER BY sal) AS Kategória
  FROM emp;
```

**Eredmény**

JOB	ENAME	SAL	KATEGÓRIA
ANALYST	FORD	3000	1
ANALYST	SCOTT	3000	2
CLERK	SMITH	800	1
CLERK	JAMES	950	1
CLERK	ADAMS	1100	2
CLERK	MILLER	1300	3
MANAGER	CLARK	2450	1
MANAGER	BLAKE	2850	2
MANAGER	JONES	2975	3
PRESIDENT	KING	5000	1
SALESMAN	MARTIN	1250	1
SALESMAN	WARD	1250	1
SALESMAN	TURNER	1500	2
SALESMAN	ALLEN	1600	3

14 sor kijelölve.

**Megjegyzés**

A megadott három kategóriához minden munkakörben három csoportot hozott létre, kivéve azokat a munkaköröket, melyekben nem volt legalább három dolgozó.

## 14. FEJEZET

# A DEMO vállalat (üzleti alkalmazások)

Jelen fejezet DEMO vállalatának összetett szerkezetén keresztül lehetőség nyílik a korábbi fejezetek gyakorlati alkalmazására. Talán figyelemre méltó, hogy ez a nem is olyan egyszerű felépítésű vállalat a mindannyiunk számára jól ismert, 14 soros emp tábla adatain jött létre...

## A DEMO vállalat felépítése

- A DEMO vállalat az emp, dept és salgrade táblák által reprezentált szervezet, melyben a vállalati struktúrát az emp tábla alapján a következőképpen értelmezzük.
- Egy a dolgozónak közvetlen főnöke a b dolgozó, ha a b dolgozó azonosítója (empno) az a dolgozó főnökazonosítója (mgr). Ekkor természetesen egyúttal az a dolgozó közvetlen beosztottja a b dolgozónak.
- Főosztályvezetőknek nevezzük az elnök (president) azon közvetlen beosztottjait, akiknek van beosztottja.
- Osztályvezetőknek nevezzük azokat a dolgozókat, akiknek közvetlen főnöke főosztályvezető, és van beosztottja.
- Főosztálynak nevezzük a dolgozók azon halmazát, akiknek ugyanaz a főosztályvezetője. A főosztályvezető is hozzátartozik a saját főosztályához. Egy főosztályt a főosztályvezetője nevével azonosítunk.
- Osztálynak nevezzük a dolgozók azon halmazát, akiknek ugyanaz az osztályvezetőjük. Az osztályvezető is hozzátartozik a saját osztályához. Egy osztályt az osztályvezetője nevével azonosítunk.
- Részlegnek nevezzük a dolgozók azon halmazát, akiknek ugyanaz a részlegazonosítója.
- A fenti struktúrát oly módon értelmezzük, hogy a részlegek, a munkakörök és az osztályok egymást kölcsönösen átfedő halmazok. Például egy részleg több osztályról is tartalmazhat dolgozókat, egy osztályon több részlegről is lehetnek dolgozók, és azonos munkakörű dolgozók különböző részlegekben is dolgozhatnak. A továbbiakban még akkor is ezekkel a feltételezésekkel fogunk dolgozni, ha ezek közül egyik vagy másik az emp tábla kisszámú adatából nem következik.

# Feladatok

## 14.1. feladat

Listázza ki „gazdaságosan” (a szükséges mezőszélességekkel) az emp és a dept táblákat.

## 14.2. feladat

Ábrázolja grafikusan a DEMO vállalat dolgozóit a vállalati hierarchiában elfoglalt helyük szerint. Az egyes dolgozók neve mellett tüntesse fel a besorolását is (elnök, főosztályvezető, osztályvezető, főosztályi közvetlen beosztott, beosztott).

## 14.3. feladat

Állítsa elő a főosztályvezetők, az osztályvezetők, a beosztottak, a főosztályi közvetlen beosztottak, az osztályok és a főosztályok nézettábláit. Állítson elő egy olyan kibővített osztály nézettáblát is, amely az osztályok dolgozóin kívül tartalmazza a főosztályi közvetlen beosztottakat a főosztályvezetőjükkel (mint pszeudo-osztályvezetőjükkel) együtt.

## 14.4. feladat

Adja meg főosztályvezetőnként a főosztályok dolgozónak számát és összfizetését, ez utóbbi szerint rendezve.

## 14.5. feladat

Listázza ki az osztály-dolgozók fizetéseit főosztályonkénti részösszegképzéssel.

## 14.6. feladat

Listázza ki a főosztályi közvetlen beosztottak fizeteseit főosztályonkénti részösszegképzéssel.

## 14.7. feladat

Listázza ki a dolgozók fizeteseit főosztályonkénti részösszegképzéssel.

## 14.8. feladat

Listázza ki főosztályvezetőnként rendezve a főosztályok dolgozónak fizetését osztályonként csoportosítva, osztályonkénti és főosztályonkénti részösszeg-meghatározással kiegészítve.

## 14.9. feladat

Listázza ki főosztályvezetőnként rendezve a főosztályok dolgozónak fizetését munkakörönként csoportosítva, munkakörönként ( minden főosztályon belül) képezve a részösszegeket.

## 14.10. feladat

Listázza ki részlegenként (a részlegek neve szerint rendezve) a dolgozók fizetését munkakörönként csoportosítva, munkakörönként ( minden részlegen belül) képezve a részösszegeket.

**14.11. feladat**

Listázza ki osztályvezetőnként a dolgozói számát, összfizetését, a főosztályvezető nevét és telephelyét, elsődlegesen az osztályvezetők, másodlagosan a főosztályvezetők neve szerint rendezve. Alkalmazzon ismétlésmentes kiíratásokat.

**14.12. feladat**

Listázza ki osztályonként (az osztályok vezetőinek neve szerint rendezve) a dolgozók nevét, munkakörét, fizetését, osztályának átlagfizetését és fizetésbeállási értékét (amely a saját fizetés és az osztály átlagos fizetésének a hányadosa). Alkalmazzon ismétlésmentes kiíratásokat.

**14.13. feladat**

Listázza ki osztályonként (az osztályok vezetőinek neve szerint rendezve) a dolgozók nevét, munkakörét, fizetését, osztályának összfizetését, és a fizetésrészeseidéi értékét (amely a saját fizetés és az osztály összfizetésének a hányadosa). Alkalmazzon ismétlésmentes kiíratásokat.

**14.14. feladat**

Írjon szkript programot, amely

- A. főosztályonként és azon belül osztályonként rendezett módon csoportosítva listázza a dolgozók nevét, munkakörét, fizetését, fizetésének az osztálya összfizetéséből való részesedését, valamint elsődlegesen a fizetésének az osztály összfizetéséből való részesedése szerint csökkenően, másodlagosan a munkaköre szerint növekvő módon az osztályabeli rangsorát (alkalmazzon ismétlésmentes kiíratásokat);
- B. kimutatást készít arról, hogy a felhasználó által megadott fizetésű új dolgozó a fenti rangsorban hányadik helyre kerülne az egyes osztályokon;
- C. kimutatást készít arról, hogy a felhasználó által megadott fizetésű új dolgozó az egyes osztályokon mekkora bérbeszütséget okozna belépésekével, ahol a bérbeszütség az új dolgozó fizetésének és az egyes osztályok átlagfizetésének előjeles különbsége.

Az osztályok és a főosztályok azonosítására azok vezetőinek nevét használja.

**14.15. feladat**

Adja meg főosztályonként (azok vezetőinek neve szerint rendezve) a legmagasabb, a legalacsonyabb és átlagos fizetéseket, valamint azok szórásait, az összfizetéseket és az átlagos fizetésbeállási értékeket osztályonként, majd főosztályonként csoportosítva.



## **II. RÉSZ**

# **MEGOLDÁSGYŰJTEMÉNY**

### **Bevezetés**

Az egyes feladatok megoldásai előtt a feladatok szövegét a könnyebb kezelhetőség érdekében megismételtük. A feladatok egy részéhez több megoldást is adunk, elemezve az egyes változatok előnyeit, hátrányait, sőt az oktatási tapasztalatok alapján esetenként egy-egy tipikus hibás megoldást is bemutatunk, megadva a hiba helyét és okát. Néhány olyan feladat esetén azonban elhagytuk a megoldást, mely elég összetett ahhoz, hogy önálló gyakorláshoz érdemesnek látszott a példatárban szerepeltetni, ám a korábbiak alapján, azokhoz hasonló módszerek felhasználásával megoldható.

Megjegyezzük, hogy a feladatmegoldások SQL\*Plus szkript programjai fejezetek szerint csoportosítva megtalálhatók a CD-mellékleten (lásd A CD-mellékletről c. részt a könyv végén).



## 1. FEJEZET

# Egyszerű lekérdezések

## Feladatok és megoldások

### 1.1. feladat

Listázza ki a 20-as részleg dolgozóinak nevét, belépési idejét, foglalkozását a nevek szerint csökkenően rendezve. (F01-01.sql)

#### Megoldás

```
COLUMN "A dolgozó neve" FORMAT A14  
COLUMN "belépési dátuma" FORMAT A15  
COLUMN "foglalkozása" FORMAT A12
```

```
SELECT ename AS "A dolgozó neve",  
       hiredate AS "belépési dátuma",  
       job      AS "foglalkozása"  
  FROM emp  
 WHERE deptno = 20  
 ORDER BY ename DESC;
```

```
CLEAR COLUMNS
```

#### Eredmény

A dolgozó neve belépési dátuma foglalkozása

SMITH	80-DEC-17	CLERK
SCOTT	87-ÁPR-19	ANALYST
JONES	81-ÁPR-02	MANAGER
FORD	81-DEC-03	ANALYST
ADAMS	87-MÁJ-23	CLERK

## 1.2. feladat

Készítse két listát, melyek a dolgozók adatait tartalmazzák. Az egyiket a fizetés szerint növekvően, a másikat a fizetés szerint csökkenően rendezze.

### Megoldás (1. lista)

```
SELECT *
  FROM EMP
 ORDER BY sal; --(vagy ORDER BY sal ASC;)
```

### Eredmény

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10

14 sor kijelölve.

### Megoldás (2. lista)

```
SELECT *
  FROM EMP
 ORDER BY sal DESC;
```

### Eredmény

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10

7499 ALLEN	SALESMAN	7698 81-FEB-20	1600	300	30
7844 TURNER	SALESMAN	7698 81-SZE-08	1500	0	30
7934 MILLER	CLERK	7782 82-JAN-23	1300		10
7521 WARD	SALESMAN	7698 81-FEB-22	1250	500	30
7654 MARTIN	SALESMAN	7698 81-SZE-28	1250	1400	30
7876 ADAMS	CLERK	7788 87-MÁJ-23	1100		20
7900 JAMES	CLERK	7698 81-DEC-03	950		30
7369 SMITH	CLERK	7902 80-DEC-17	800		20

14 sor kijelölve.

### 1.3. feladat

Listázza ki a dolgozók nevét, fizetését, jövedelmét a jövedelmük szerint csökkenően rendezve.  
(A jövedelem a fizetés és a jutalék összege.)

#### 1. megoldás (Hibás megoldás)

```
SELECT ename      AS név,
       sal        AS fizetés,
       sal+comm   AS jövedelem
  FROM emp
 ORDER BY sal+comm DESC;
```

#### Eredmény

NÉV	FIZETÉS	JÖVEDELEM
SMITH	800	
JONES	2975	
CLARK	2450	
BLAKE	2850	
SCOTT	3000	
KING	5000	
JAMES	950	
MILLER	1300	
FORD	3000	
ADAMS	1100	
MARTIN	1250	2650
ALLEN	1600	1900
WARD	1250	1750
TURNER	1500	1500

14 sor kijelölve.

**Megjegyzés**

A havi jövedelem az összes pénzbeli juttatás, amit a dolgozó kap a vállalattól. Két pénz jellegű oszlop van az emp táblában egyik a sal (havi fizetés), a másik a comm (jutalmak). Hogy megkapjuk a havi jövedelmet, a két értéket össze kell adnunk. Nézzük meg az eredményt. A jövedelem oszlopban hiányzik a dolgozók többségének havi jövedelme, holott mindenkinek van fizetése. E jelenség oka, hogy a comm mező a dolgozók egy részénél NULL értékű.

Figyelembe kell vennünk, hogy a NULL érték nem nulla, hanem egy jelölőérték, mely kiíratáskor üres mezőként jelenik meg, és amelyre az a számítási szabály vonatkozik, hogy egy NULL értéket tartalmazó kifejezés értéke NULL érték. A kifejezésekben való szerepeltetése az NVL függvényen keresztül történhet, amelyben megadható az, hogy adott esetben milyen „valódi” értékként kezeljük. Nyilvánvaló például, hogy összeadásnál nullával, szorzásnál eggyel, sztring összefűzésnél pedig üres sztringgel célszerű helyettesíteni.

**2. megoldás (Jó megoldás)**

```
SELECT ename          AS név,
       sal           AS fizetés,
       sal+NVL(comm,0) AS jövedelem
  FROM emp
 ORDER BY jövedelem DESC;
```

**Eredmény**

NÉV	FIZETÉS	JÖVEDELEM
KING	5000	5000
SCOTT	3000	3000
FORD	3000	3000
JONES	2975	2975
BLAKE	2850	2850
MARTIN	1250	2650
CLARK	2450	2450
ALLEN	1600	1900
WARD	1250	1750
TURNER	1500	1500
MILLER	1300	1300
ADAMS	1100	1100
JAMES	950	950
SMITH	800	800

14 sor kijelölve.

## 1.4. feladat

Listázza ki a dolgozók nevét, részlegüket, jövedelmüket és az adójukat (a jövedelmük 20%-a), az adójuk szerint csökkenően, a nevük szerint pedig növekvő módon rendezve.

### Megoldás

```
SELECT ename          AS Név,
       deptno         AS Részleg,
       sal+NVL(comm,0) AS Jövedelem,
       (sal+NVL(comm,0)) * 0.20 AS "ADÓ"
  FROM emp
 ORDER BY "ADÓ" DESC,
          Név ASC;
```

### Eredmény

NÉV	RÉSZLEG	JÖVEDELEM	ADÓ
KING	10	5000	1000
FORD	20	3000	600
SCOTT	20	3000	600
JONES	20	2975	595
BLAKE	30	2850	570
MARTIN	30	2650	530
CLARK	10	2450	490
ALLEN	30	1900	380
WARD	30	1750	350
TURNER	30	1500	300
MILLER	10	1300	260
ADAMS	20	1100	220
JAMES	30	950	190
SMITH	20	800	160

14 sor kijelölve.

## 1.5. feladat

Írassa ki azon alkalmazottak nevét, munkakörét és fizetését, akiknek fizetése nincs az 1500-2850 USD tartományban. A lista fejléce legyen Név, Munkakör, Fizetés.

### Megoldás

```
SELECT ename AS "Név",
       job   AS "Munkakör",
       sal   AS "Fizetés"
  FROM emp
 WHERE sal NOT BETWEEN 1500 AND 2850;
```

**Eredmény**

Név	Munkakör	Fizetés
SMITH	CLERK	800
WARD	SALESMAN	1250
JONES	MANAGER	2975
MARTIN	SALESMAN	1250
SCOTT	ANALYST	3000
KING	PRESIDENT	5000
ADAMS	CLERK	1100
JAMES	CLERK	950
FORD	ANALYST	3000
MILLER	CLERK	1300

10 sor kijelölve.

**1.6. feladat**

Írassa ki azon dolgozók nevét, munkakörét, fizetését, jutalékát és részlegazonosítóját, akik 1000 USD-nál többet keresnek, és 1981. március 1. és szeptember 30. között léptek be a vállalathoz.

**Megoldás**

```

SELECT ename      AS név,
       job        AS munkakör,
       sal        AS fizetés,
       comm       AS jutalék,
       deptno    AS részleg,
       hiredate   AS belépés
  FROM emp
 WHERE sal > 1000 AND
       hiredate BETWEEN '81-MÁR-01' AND '81-SZE-30';

```

**Eredmény**

NÉV	MUNKAKÖR	FIZETÉS	JUTALÉK	RÉSZLEG BELÉPÉS
JONES	MANAGER	2975		20 81-ÁPR-02
MARTIN	SALESMAN	1250	1400	30 81-SZE-28
BLAKE	MANAGER	2850		30 81-MÁJ-01
CLARK	MANAGER	2450		10 81-JÚN-09
TURNER	SALESMAN	1500	0	30 81-SZE-08

**Megjegyzés**

Problémát okozhat a dátum intervallum értékének megadása a BETWEEN ... AND utasításrészben. A dátumot függvényként tartja nyilván a rendszer, tehát március 1. ki-sebb, mint szeptember 30. Ügyeljünk arra, hogy a dátumot aposztrófok között adjuk meg, hasonlóan a karaktersorozathoz.

**1.7. feladat**

Írassa ki a jutalékkal rendelkező alkalmazottak nevét, jutalékát, főnökének azonosítóját. Legyen a lista rendezett a főnök azonosítója és az alkalmazottak neve szerint.

**Megoldás**

```
SELECT ename AS név,
       comm AS jutalék,
      mgr AS "főnök azonosító"
  FROM emp
 WHERE comm IS NOT NULL
 ORDER BY "főnök azonosító", név;
```

**Eredmény**

NÉV	JUTALÉK	főnök azonosító
ALLEN	300	7698
MARTIN	1400	7698
TURNER	0	7698
WARD	500	7698

**1.8. feladat**

Írassa ki azon alkalmazottak azonosítóját, nevét, foglalkozását, fizetését és jutalékát, akiknek jutaléka meghaladja a fizetésük 50%-át.

**Megoldás**

```
SELECT empno AS azonosító,
       ename AS név,
       job AS foglalkozás,
       sal AS fizetés,
      comm AS jutalék
  FROM emp
 WHERE comm > 0.5*sal
 ORDER BY név;
```

**Eredmény**

AZONOSÍTÓ NÉV	FOGLALKOZ.	FIZETÉS	JUTALÉK
7654 MARTIN	SALESMAN	1250	1400

**1.9. feladat**

Írja ki azon dolgozók nevét, foglalkozását, fizetését és belépési dátumát, akik 1981-ben léptek be a vállalathoz. A lista a belépési dátum szerint legyen rendezve.

**Megoldás**

```

SELECT empno      AS azonosító,
       ename       AS név,
       job        AS foglalkozás,
       comm        AS jutalék,
       hiredate   AS belépés
  FROM emp
 WHERE TO_CHAR(hiredate, 'YYYY') = '1981'
 ORDER BY hiredate;

```

**Eredmény**

AZONOSÍTÓ NÉV	FOGLALKOZ.	JUTALÉK	BELÉPÉS
7499 ALLEN	SALESMAN	300	81-FEB-20
7521 WARD	SALESMAN	500	81-FEB-22
7566 JONES	MANAGER		81-ÁPR-02
7698 BLAKE	MANAGER		81-MÁJ-01
7782 CLARK	MANAGER		81-JÚN-09
7844 TURNER	SALESMAN	0	81-SZE-08
7654 MARTIN	SALESMAN	1400	81-SZE-28
7839 KING	PRESIDENT		81-NOV-17
7900 JAMES	CLERK		81-DEC-03
7902 FORD	ANALYST		81-DEC-03

10 sor kijelölve..

**Megjegyzés ..**

Problémát okozhat az 1981-es év összehasonlítása a hiredate oszlop értékeivel. Ilyenkor ugyanis nem használható a teljes dátum, csak az a része, melyet a feladat igényel. A megoldás lényege a megfelelő dátumformátum alkalmazása a hiredate oszlopra. Ekkor az oszloban levő dátumból csak az év részt hasonlítjuk össze az igényelt évszám karakteres alakjával.

## 1.10. feladat

Listázza ki azon alkalmazottak nevét, foglalkozását és jövedelmét, akiknek a nevében két L betű szerepel, továbbá vagy a 30-as részlegben dolgozik, vagy a főnökének azonosítója 7782.

### Megoldás

```
SELECT ename          AS név,
       job            AS foglalkozás,
       sal+NVL(comm,0) AS jövedelem,
       mgr           AS főnök
  FROM emp
 WHERE UPPER(ename) LIKE '%L%L%' AND
       (deptno = 30   OR
        mgr    = 7782);
```

### Eredmény

NÉV	FOGLALKOZ	JÖVEDELEM	FŐNÖK
ALLEN	SALESMAN	1900	7698
MILLER	CLERK	1300	7782



### Megjegyzés

Összetett logikai kifejezések használatakor célszerű az elsődlegesen elvégzendő műveletet (operandusaival együtt) zárójelbe tenni. Ilyen módon függetleníthetjük magunkat a nehezen megjegyezhető precedenciaszabályoktól, és egyben áttekinthetőbb alakhoz is jutunk.

## 1.11. feladat

Listázza ki részlegazonosító szerint rendezve a clerk és a salesman munkakörű dolgozók éves fizetését a részleg szerint rendezve.

### Megoldás

```
SELECT ename      AS Név,
       sal*12    AS "Éves fizetés",
       deptno    AS Részleg
  FROM emp
 WHERE UPPER(job) IN ('CLERK', 'SALESMAN')
 ORDER BY deptno;
```

### Eredmény

NÉV	Éves fizetés	RÉSZLEG
MILLER	15600	10

SMITH	9600	20
ADAMS	13200	20
ALLEN	19200	30
WARD	15000	30
JAMES	11400	30
MARTIN	15000	30
TURNER	18000	30

8 sor kijelölve.

## 1.12. feladat

Listázza ki az összes dolgozót oly módon, hogy azoknál, akik nem kapnak jutalékot, az a szöveg jelenjen meg, hogy Nincs jutalék. A lista fejléce legyen azonosító, belépési dátum, név, foglalkozás, jutalék.

### Megoldás

```
SELECT emplno AS azonosító,
       hiredate AS "belépési dátum",
        ename AS név,
        job AS foglalkozás,
NVL(TO_CHAR(comm), 'Nincs jutalék') AS jutalék
  FROM emp;
```

### Eredmény

AZONOSÍTÓ	belépési	NÉV	FOGLALKOZ	JUTALÉK
7369	80-DEC-17	SMITH	CLERK	Nincs jutalék
7499	81-FEB-20	ALLEN	SALESMAN	300
7521	81-FEB-22	WARD	SALESMAN	500
7566	81-ÁPR-02	JONES	MANAGER	Nincs jutalék
7654	81-SZE-28	MARTIN	SALESMAN	1400
7698	81-MÁJ-01	BLAKE	MANAGER	Nincs jutalék
7782	81-JÚN-09	CLARK	MANAGER	Nincs jutalék
7788	87-ÁPR-19	SCOTT	ANALYST	Nincs jutalék
7839	81-NOV-17	KING	PRESIDENT	Nincs jutalék
7844	81-SZE-08	TURNER	SALESMAN	0
7876	87-MÁJ-23	ADAMS	CLERK	Nincs jutalék
7900	81-DEC-03	JAMES	CLERK	Nincs jutalék
7902	81-DEC-03	FORD	ANALYST	Nincs jutalék
7934	82-JAN-23	MILLER	CLERK	Nincs jutalék

14 sor kijelölve.

**Megjegyzés**

Figyeljünk fel arra, hogy a feladat szerint a comm oszlop NULL értékű mezőiben karaktersorozatot kell listázni, amely nem kompatibilis a comm oszlop adattípusával. Ennek értelmében az NVL függvény használata során a comm mező számértékét konvertáljuk szövegsorozattá, helyettesítő értékként pedig adjuk meg a kiírandó szöveget.

### 1.13. feladat

Listázza ki a man karaktersorozatot tartalmazó munkakörben dolgozók nevét és munkakörét, a munkakör és a név szerint rendezve.

**Megoldás**

```
SELECT ename, job
  FROM emp
 WHERE UPPER(job) LIKE '%MAN%'
 ORDER BY job, ename;
```

**Eredmény**

ENAME	JOB
BLAKE	MANAGER
CLARK	MANAGER
JONES	MANAGER
ALLEN	SALESMAN
MARTIN	SALESMAN
TURNER	SALESMAN
WARD	SALESMAN

7 sor kijelölve.

**Megjegyzés**

Az SQL a karaktersorozatok összehasonlításánál megkülönbözteti a kis- és nagybetűket. Ezért célszerű az összehasonlító feltétel oszlopnév oldalán a karaktersorozatot nagybetűssé átalakítani (ugyanis nem minden tudhatjuk, hogy a lekérdezendő adattáblában kis- vagy nagybetűvel írták-e a szövegadatokat).

### 1.14. feladat

Listázza ki foglalkozás szerint csoportosítva azon dolgozók nevét, foglalkozását, jövedelmét és részlegét, akiknek jövedelme kisebb 2500 USD-nál, valamint 1981 és 1982 között léptek be. A keletkezett lista elsődlegesen a foglalkozás, másodlagosan a dolgozó neve szerint rendezve.

**Megoldás**

```

SELECT ename      AS név,
       job        AS foglalkozás,
       sal+NVL(comm,0) AS jövedelem,
       deptno     AS részleg
  FROM emp
 WHERE sal+NVL(comm,0) < 2500 AND
       TO_CHAR(hiredate,'YYYY') BETWEEN '1981' AND '1982'
 ORDER BY job, ename;

```

**Eredmény**

NÉV	FOGLALKOZ	JÖVEDELEM	RÉSZLEG
JAMES	CLERK	950	30
MILLER	CLERK	1300	10
CLARK	MANAGER	2450	10
ALLEN	SALESMAN	1900	30
TURNER	SALESMAN	1500	30
WARD	SALESMAN	1750	30

6 sor kijelölve.

**1.15. feladat**

Listázza ki azoknak az alkalmazottaknak a nevét, éves fizetését és a munkában eltöltött hónap-jainak számát, akik 1981.07.01. előtt léptek be a vállalathoz. A lista a hónapok száma szerint csökkenően legyen rendezve.

**Megoldás**

```

SELECT ename      AS "Név",
       sal*12    AS "Éves fizetés",
       ROUND( MONTHS_BETWEEN(sysdate, hiredate) )
                  AS "Munkában töltött hónapok száma"
  FROM emp
 WHERE TO_CHAR(hiredate, 'YYYY.MM.DD.') < '1981.07.01.'
 ORDER BY ROUND(MONTHS_BETWEEN(sysdate, hiredate)) DESC;
 -- vagy
 -- ORDER BY "Munkában töltött hónapok száma" DESC;

```

**Eredmény**

Név	Éves fizetés	Munkában töltött hónapok száma
SMITH	9600	287
ALLEN	19200	285

WARD	15000	285
JONES	35700	283
BLAKE	34200	282
CLARK	29400	281

6 sor kijelölve.



#### Megjegyzés

Figyeljünk arra, hogy a dátum kezelése a feladatnak megfelelő legyen.

### 1.16. feladat

Listázza ki a C és az M betűvel kezdődő foglalkozású alkalmazottak nevét (nevüket nagybetűvel kezdve és kisbetűvel folytatva), valamint nevük hosszát. Rendezze a listát a foglalkozás szerint.

#### Megoldás

```
SELECT INITCAP(ename) AS "Név",
       INITCAP(job ) AS "Foglalkozás",
       LENGTH(ename) AS "Név hossza"
  FROM emp
 WHERE UPPER(job) LIKE 'C%' OR
       UPPER(job) LIKE 'M%'
 ORDER BY INITCAP(job);
```

#### Eredmény

Név	Foglalkoz	Név hossza
Smith	Clerk	5
Adams	Clerk	5
Miller	Clerk	6
James	Clerk	5
Jones	Manager	5
Blake	Manager	5
Clark	Manager	5

7 sor kijelölve.



#### Megjegyzés

E feladatnál különösen ügyeljünk a kis- és nagybetűk használatára. Az igényelt kiírás megvalósítása (nagybetűvel kezdődik és kisbetűvel folytatódik) az INITCAP függvényenél lehetséges, a karaktersorozat hossza pedig a LENGTH egysoros függvénnel kérdezhető le.

## 1.17. feladat

A belépési dátum napjai szerint csoportosítva listázza ki azon dolgozók azonosítóját, nevét, jövedelmét, munkába állásuk napját, részlegét, akiknek jövedelme 1300 és 5500 USD közötti érték. A keletkezett lista elsődlegesen a napok sorszáma szerint, másodlagosan a dolgozó neve szerint legyen rendezve. A hét első napja legyen a vasárnap.

### Megoldás

```

SELECT empno          AS azonosító,
       ename           AS név,
       sal + NVL(comm,0) AS jövedelem,
       TO_CHAR(hiredate,'DAY') AS "KezdőNap",
       deptno          AS részleg
  FROM emp
 WHERE sal + NVL(comm,0) BETWEEN 1300 AND 5500
 ORDER BY TO_CHAR(hiredate+1,'d'),
          ename ;

```

### Eredmény

AZONOSÍTÓ NÉV	JÖVEDELEM	KezdőNap	RÉSZLEG
7788 SCOTT	3000	VASÁRNAP	20
7521 WARD	1750	VASÁRNAP	30
7654 MARTIN	2650	HÉTFŐ	30
7782 CLARK	2450	KEDD	10
7839 KING	5000	KEDD	10
7844 TURNER	1500	KEDD	30
7902 FORD	3000	CSÜTÖRTÖK	20
7566 JONES	2975	CSÜTÖRTÖK	20
7499 ALLEN	1900	PÉNTEK	30
7698 BLAKE	2850	PÉNTEK	30
7934 MILLER	1300	SZOMBAT	10

11 sor kijelölve.



### Megjegyzés

- A belépési napok kiíratási formátuma a 'DAY' karakterSOROZAT beírásától függ. Így 'DAY' esetén a napok neveit csupa nagybetűvel írja ki, 'Day' esetén a napok nevében csak a kezdőbetű nagy, 'day' esetén pedig napok nevében minden betű kicsi. Hasonló szabály vonatkozik a többi karakteres dátumrészre is (például 'MON', 'MONTH' stb.).
- A napok sorszáma 1–7 ig (hétfőtől kezdve) a 'd' formátum. Ezért a napok nevei szerint történő rendezés ennek segítségével történhet.
- Ha a hét kezdőnapjaként nem a hétfőt kívánjuk listázni, akkor a dátum értéket kell az eltolás mértékével megnövelni vagy csökkenteni (lásd a példát).

## 1.18. feladat

A vállalatról hűségjutalmat adnak, és ehhez szükséges azon dolgozók azonosítója, neve, fizetése és munkában eltöltött éve, akik legalább 15 éve álltak munkába. Rendezze a listát a munkában eltöltött évek szerint csökkenően, valamint az azonosító szerint növekvően.

### 1. megoldás (Dátumfüggvényekkel)

```
SELECT ROUND(MONTHS_BETWEEN(sysdate,hiredate)/12) AS év,
       empno                           AS azonosító,
       ename                            AS név,
       sal                             AS fizetés
  FROM emp
 WHERE ROUND(MONTHS_BETWEEN(sysdate,hiredate)/12) > 15
 ORDER BY év DESC, empno;
```

### 2. megoldás (Aritmetikai műveletekkel)

```
SELECT ROUND((sysdate-hiredate)/365) AS év,
       empno                           AS azonosító,
       ename                            AS név,
       sal                             AS fizetés
  FROM emp
 WHERE ROUND((sysdate-hiredate)/365) > 15
 ORDER BY év DESC, empno;
```

### Eredmény (mindkét megoldás esetén)

ÉV	AZONOSÍTÓ	NÉV	FIZETÉS
24	7369	SMITH	800
24	7499	ALLEN	1600
24	7521	WARD	1250
23	7566	JONES	2975
23	7654	MARTIN	1250
23	7698	BLAKE	2850
23	7782	CLARK	2450
23	7839	KING	5000
23	7844	TURNER	1500
23	7900	JAMES	950
23	7902	FORD	3000
23	7934	MILLER	1300
17	7788	SCOTT	3000
17	7876	ADAMS	1100

14 sor kijelölve.

## 1.19. feladat

Listázza ki a dolgozók nevét, munkakörét, fizetését és a fizetési kategóriáját, mely 1000 USD alatt 1, 2000 USD alatt 2 stb., ez utóbbi szerint csökkenően rendezve. (A fizetéstartomány 1..6000 USD.)

### Megoldás

```

SELECT ename           AS "Neve",
       job            AS "Munkakör",
       sal            AS "Fizetés",
       CASE
         WHEN sal < 1000 THEN 1
         WHEN sal < 2000 THEN 2
         WHEN sal < 3000 THEN 3
         WHEN sal < 4000 THEN 4
         WHEN sal < 5000 THEN 5
         ELSE 6
       END           AS "FizetésiKategória"
  FROM emp
 ORDER BY "FizetésiKategória" DESC;

```

### Eredmény

Neve	Munkakör	Fizetés	FizetésiKategória
KING	PRESIDENT	5000	6
FORD	ANALYST	3000	4
SCOTT	ANALYST	3000	4
BLAKE	MANAGER	2850	3
CLARK	MANAGER	2450	3
JONES	MANAGER	2975	3
MARTIN	SALESMAN	1250	2
MILLER	CLERK	1300	2
ADAMS	CLERK	1100	2
WARD	SALESMAN	1250	2
ALLEN	SALESMAN	1600	2
TURNER	SALESMAN	1500	2
JAMES	CLERK	950	1
SMITH	CLERK	800	1

14 sor kijelölve.

## 1.20. feladat

Listázza ki a dolgozók nevét, azonosítóját és beosztását a név szerint rendezve. Egy dolgozó beosztása Főnök, ha van beosztottja, egyébként NULL érték.

### Megoldás

```
COLUMN "Beosztás"           FORMAT A9
SELECT ename                AS "Neve",
       empno                 AS "Azonosítója",
       CASE
          WHEN empno IN (SELECT mgr FROM emp)
          THEN 'Főnök'
          ELSE NULL
       END                   AS "Beosztás"
FROM emp
ORDER BY ename;
CLEAR COLUMNS
```

### Eredmény

Neve	Azonosítója	Beosztás
ADAMS	7876	
ALLEN	7499	
BLAKE	7698	Főnök
CLARK	7782	Főnök
FORD	7902	Főnök
JAMES	7900	
JONES	7566	Főnök
KING	7839	Főnök
MARTIN	7654	
MILLER	7934	
SCOTT	7788	Főnök
SMITH	7369	
TURNER	7844	
WARD	7521	

14 sor kijelölve.



### Megjegyzés

A kiíratás-konvertálási feladatokban sok esetben egyaránt használható a CASE kifejezés és a DECODE függvény. A fenti feladat azonban abba a típusba tartozik, amely csak a CASE kifejezés használatával oldható meg. Próbálkozzunk!

Figyelembe véve az elméleti összefoglalóban bemutatott szintaktikai alakokat, a fenti megoldás

```
CASE
    WHEN empno IN (SELECT mgr FROM emp)
        THEN 'Főnök'
    ELSE NULL
END           AS "Beosztás"
```

utasításrész a

```
DECODE(empno IN (SELECT mgr FROM emp),
       TRUE,  'Főnök',
       NULL   AS "Beosztás"
```

utasításrésszel volna helyettesíthető, azonban ez mégse működik. Mi lehet ennek az oka?

A DECODE függvény első paraméterként egy olyan oszlopkifejezést igényel, mely egy egysoros függvény (egy matematikai kifejezés). Itt azonban egy allekérdezést tartalmazó logikai kifejezés áll, mely ugyan jól használható egy SELECT utasítás WHERE utasításrészében, itt azonban nem.

## 2. FEJEZET

# Egytáblás csoportosító lekérdezések

## Feladatok és megoldások

### 2.1. feladat

Listázza ki munkakörönként az átlagfizetéseket két tizedesre kerekítve. Rendezze átlagfizetések szerint csökkenően.

#### Megoldás

```
SELECT job AS munkakör,  
       ROUND(AVG(sal),2) AS "Munkaköri Átlagfizetés"  
  FROM emp  
 GROUP BY job  
 ORDER BY "Munkaköri Átlagfizetés" DESC;
```

#### Eredmény

MUNKAKÖR	Munkaköri Átlagfizetés
PRESIDENT	5000
ANALYST	3000
MANAGER	2758.33
SALESMAN	1400
CLERK	1037.5

### 2.2. feladat

Listázza ki csökkenően rendezve a főnökök átlagfizetését egész értékre kerekítve. (Főnök az a dolgozó, akinek azonosítója szerepel az mgr oszlopban.)

#### Megoldás

```
SELECT mgr AS "Főnök",  
       ROUND(AVG(sal),0) AS "FőnökÁtlag"  
  FROM emp
```

```
GROUP BY mgr
ORDER BY "FőnökÁtlag" DESC;
```

**Eredmény**

Főnök FőnökÁtlag

	5000
7566	3000
7839	2758
7698	1310
7782	1300
7788	1100
7902	800

7 sor kijelölve.

**2.3. feladat**

Listázza ki részlegenként a legnagyobb és legkisebb havi jövedelmeket.

**Megoldás**

```
SELECT deptno          AS "részleg",
       MIN(sal+NVL(comm,0)) AS "legkisebb jövedelem",
       MAX(sal+NVL(comm,0)) AS "legnagyobb jövedelem"
  FROM emp
 GROUP BY deptno;
```

**Eredmény**

részleg legkisebb jövedelem legnagyobb jövedelem

10	1300	5000
20	800	3000
30	950	2850

**2.4. feladat**

Listázza ki a legalább egy dolgozójú részlegeket a dolgozószám szerint csökkenően rendezve.

**Megoldás**

```
SELECT COUNT(*)    AS "DolgozóSzám",
       deptno      AS "RészlegSzám"
  FROM emp
 GROUP BY deptno
 HAVING COUNT(*) > = 1
 ORDER BY "DolgozóSzám";
```

**Eredmény**

DolgozóSzám	RészlegSzám
3	10
5	20
6	30

**2.5. feladat**

Listázza ki ki a főnökök azonosítóit, valamint azt, hogy hány beosztottjuk van. Rendezze a listát a beosztottak száma szerint csökkenően. Ha valakinek nincs főnöke, ahoz írjon valamilyen megjegyzést (tulajdonos vagy elnök stb.).

**Megoldás**

```
SELECT NVL(TO_CHAR(mgr), 'elnök') AS "FőnökAzonosító",
       COUNT(*)                      AS "Beosztottak száma"
  FROM emp
 GROUP BY mgr
 ORDER BY "Beosztottak száma" DESC;
```

**Eredmény**

FőnökAzonosító	Beosztottak száma
7698	5
7839	3
7566	2
7782	1
7788	1
7902	1
elnök	1

7 sor kijelölve.

**2.6. feladat**

Listázza ki az azonosítójuk hárommal való oszthatósága alapján a dolgozók átlagjövedelmét, a dolgozók számát és legkisebb fizetését.

**Megoldás**

```
SELECT ROUND(MOD(empno,3))          AS "osztható-e",
       ROUND(AVG(sal+NVL(comm,0)))  AS "ÁtlagJövedelem",
       COUNT(*)                     AS létszám,
       MIN(sal)                     AS fizetés
  FROM emp
 GROUP BY ROUND(MOD(empno,3));
```

**Eredmény**

osztható-e	Átlagjövedelem	LÉTSZÁM	FIZETÉS
0	3004	7	1250
1	1375	4	800
2	1567	3	1300

**2.7. feladat**

Listázza ki a 2000 USD-nál nagyobb átlagjövedelmeket egész értékre kerekítve a foglalkozás szerint csoportosítva. A lista a foglalkozás szerint legyen rendezett.

**Megoldás**

```
SELECT job AS foglalkozás,
       ROUND(AVG(sal+NVL(comm,0)),0) AS átlagjövedelem
  FROM emp
 GROUP BY job
 HAVING ROUND(AVG(sal+NVL(comm,0)),0) > 2000
 ORDER BY job;
```

**Eredmény**

## FOGLALKOZ ÁTLAGJÖVEDELEM

ANALYST	3000
MANAGER	2758
PRESIDENT	5000

**2.8. feladat**

Listázza ki azokat a részlegeket, ahol a fizetésátlag nagyobb 1500 USD-nál. Rendezze fizetésátlag szerint csökkenően.

**Megoldás**

```
SELECT deptno AS részleg,
       ROUND(AVG(sal)) AS fizetésátlag
  FROM emp
 GROUP BY deptno
 HAVING ROUND(AVG(sal)) > 1500
 ORDER BY fizetésátlag DESC;
```

**Eredmény**

## RÉSZLEG FIZETÉSÁTLAG

10	2917
----	------

20	2175
30	1567

## 2.9. feladat

Listázza ki foglalkozásonként a legnagyobb jövedelmeket, jövedelem szerint rendezve.

### Megoldás

```
SELECT job AS foglalkozás,
       MAX(sal+ NVL(comm,0)) AS "Legnagyobb jövedelem"
    FROM emp
   GROUP BY job
  ORDER BY "Legnagyobb jövedelem";
```

### Eredmény

FOGLALKOZ Legnagyobb jövedelem

CLERK	1300
SALESMAN	2650
MANAGER	2975
ANALYST	3000
PRESIDENT	5000

## 2.10. feladat

Listázza ki, hogy az egyes foglalkozási csoportokon belül hányan dolgoznak. A lista a létszám szerint legyen rendezett.

### Megoldás

```
SELECT COUNT(*) AS létszám,
       job AS foglalkozás
    FROM emp
   GROUP BY job
  ORDER BY létszám;
```

### Eredmény

LÉTSZÁM FOGLALKOZ

1	PRESIDENT
2	ANALYST
3	MANAGER
4	CLERK
4	SALESMAN

## 2.11. feladat

Listázza ki a főnökök azonosítóit és a főnökökhöz tartozó beosztottak számát, ez utóbbi adat szerint rendezve.

### Megoldás

```
SELECT mgr      AS "Főnök",
       COUNT(*) AS "BeosztottakSzáma"
  FROM emp
 GROUP BY mgr
 HAVING mgr IS NOT NULL
 ORDER BY "BeosztottakSzáma":
```

### Eredmény

Főnök	BeosztottakSzáma
7782	1
7788	1
7902	1
7566	2
7839	3
7698	5

6 sor kijelölve.

## 2.12. feladat

Listázza ki azon foglalkozások átlagjövedelmét, amelyek nevében a man alsztring megtalálható. A listát rendezze az átlagjövedelem szerint csökkenő sorrendben.

### Megoldás

```
SELECT job                      AS munkakör,
       ROUND(AVG(sal+NVL(comm,0))) AS ÁtlagJövedelem
  FROM emp
 GROUP BY job
 HAVING UPPER(job) LIKE ('%MAN%')
 ORDER BY ÁtlagJövedelem;
```

### Eredmény

MUNKAKÖR	ÁTLAGJÖVEDELEM
SALESMAN	1950
MANAGER	2758

## 2.13. feladat

Listázza ki rendezve azon foglalkozási csoportok átlagfizetését, ahol kettő vagy ennél több alkalmazott dolgozik.

### Megoldás

```
SELECT job AS foglalkozás,
       ROUND(AVG(sal)) AS Átlagfizetés,
       COUNT(*) AS létszám
  FROM emp
 GROUP BY job
 HAVING COUNT(*) >= 2
 ORDER BY Átlagfizetés;
```

### Eredmény

FOGLALKOZ	ÁTLAGFIZETÉS	LÉTSZÁM
CLERK	1038	4
SALESMAN	1400	4
MANAGER	2758	3
ANALYST	3000	2

## 2.14. feladat

Írjon utasítást azon részlegek azonosítójának, dolgozói számának és azok legnagyobb és legkisebb jövedelmének lekérdezésére, ahol a részlegszám páros. A listát a részlegazonosító szerint rendezze.

### Megoldás

```
SELECT deptno AS "Részleg",
       COUNT(*) AS "DolgozókSzáma",
       MAX(sal+NVL(comm,0)) AS "Legnagyobb",
       MIN(sal+NVL(comm,0)) AS "Legkisebb"
  FROM emp
 GROUP BY deptno
 HAVING MOD(deptno,2) = 0
 ORDER BY deptno;
```

### Eredmény

Részleg	DolgozókSzáma	Legnagyobb	Legkisebb
10	3	5000	1300
20	5	3000	800
30	6	2850	950

## 2.15. feladat

Listázza ki az azonosító paritása szerint csoportosítva a dolgozókat. Hagya ki azokat a dolgozókat, akik 1981 után léptek be a vállalathoz. Rendezze a listát elsődlegesen paritás szerint, másodlagosan a dolgozó neve szerint.

### Megoldás

```
SELECT MOD(empno, 2) AS párosság,
       empno          AS azonosító,
       ename          AS név,
       TO_CHAR(hiredate, 'YYYY') AS belépés
  FROM emp
 WHERE TO_CHAR(hiredate, 'YYYY') < '1982'
 ORDER BY párosság, név;
```

### Eredmény

PÁROSSÁG	AZONOSÍTÓ NÉV	BELÉ
0	7698 BLAKE	1981
0	7782 CLARK	1981
0	7902 FORD	1981
0	7900 JAMES	1981
0	7566 JONES	1981
0	7654 MARTIN	1981
0	7844 TURNER	1981
1	7499 ALLEN	1981
1	7839 KING	1981
1	7369 SMITH	1980
1	7521 WARD	1981

11 sor kijelölve.

## 2.16. feladat

Számítsa ki az átlagos jutalékot.

### 1. megoldás (Hibás)

```
SELECT ROUND(AVG(NVL(comm,0))) AS "ÁtlagJutalék"
  FROM emp;
```

### Eredmény

ÁtlagJutalék
--------------

**Megjegyzés**

Bár a fenti megoldás numerikusan helyes, mégis feltételezhető, hogy a felhasználónak nem erre az adatra volt szüksége. Gondolunk csak bele, hogy miként alakulna ez az érték abban az esetben, ha még további 20 dolgozó lenne a vállalatnál olyan beosztásban, amely nem jár jutalékkal! A felhasználó feltehetően arra kívánta, hogy a jutalékkal rendelkezőket tekintve mekkora az átlagjutalék. Ekkor azonban az átlag meghatározásából ki kell hagyni a jutalékkal nem rendelkezőket.

**2. megoldás (Jó megoldás)**

```
SELECT ROUND(AVG(comm)) AS "ÁtlagJutalék"
FROM emp
WHERE comm IS NOT NULL;
```

**3. megoldás (Jó megoldás másként)**

```
SELECT ROUND(AVG(comm)) AS "ÁtlagJutalék"
FROM emp;
```

**Eredmény (mindkét esetben)**

ÁtlagJutalék

-----  
550

**2.17. feladat**

Készítsen listát a páros és a páratlan azonosítójú dolgozók számáról.

**Megoldás**

```
SELECT MOD(empno,2) AS Paritás,
       COUNT(*)      AS Létszám
  FROM emp
 GROUP BY MOD(empno,2);
```

**Eredmény**

PARITÁS	LÉTSZÁM
0	10
1	4

**2.18. feladat**

Listázza ki a dolgozók számát fizetési kategóriák szerint. (A fizetési kategóriákat vagy ön definiálja, vagy vegye a salgrade táblából.)

## 1. megoldás (Saját fizetési kategória definícióval)

Definiáljuk 1000 USD-onként a fizetési kategóriákat:

Kategória	Fizetések
1	0-999
2	1000-1999
3	2000-2999
4	3000-3999
5	4000-4999
6	5000-5999

```
SELECT TRUNC(sal/1000)+1 AS Kategória,
       COUNT(*)           AS "Dolgozók Száma"
  FROM emp
 GROUP BY TRUNC(sal/1000)+1;
```

### Eredmény

KATEGÓRIA	Dolgozók Száma
1	2
2	6
3	3
4	2
6	1

KATEGÓRIA	Dolgozók Száma
1	2
2	6
3	3
4	2
6	1

#### Megjegyzés

- A fenti fizetési kategóriák esetén a TRUNC függvénnyel egyszerűen előállítható az eredmény. Emlékeztetünk arra, hogy ezt a feladatot a CASE kifejezés segítségével oldottuk meg az 1.19. feladatban.
- Ha a salgrade táblát választjuk, akkor a megoldás csak a két tábla összekapcsolásával kapható meg (erről csak a következő fejezetben lesz szó). Alább azért megadjuk ezt a megoldást is. Ekkor az eredmény természetesen eltérő, mivel mások a fizetési kategóriák.

## 2. megoldás (A salgrade tábla használatával)

```
SELECT GRADE    AS kategória,
       COUNT(*) AS "Dolgozók Száma"
  FROM emp, salgrade
 WHERE sal BETWEEN losal AND hisal
 GROUP BY grade;
```

**Eredmény**

KATEGÓRIA Dolgozók Száma	
1	3
2	3
3	2
4	5
5	1

1	3
2	3
3	2
4	5
5	1

**2.19. feladat**

Listázza ki főnökönként (mgr) a főnökhöz tartozó legkisebb dolgozói fizetéseket. Hagyja ki azon dolgozók fizetését, akiknek nincs főnökük, valamint azokat a csoportokat, ahol a legkisebb fizetés nagyobb 2000 USD-nál. Rendezze a listát a legkisebb fizetések szerint növekvően.

**Megoldás**

```
SELECT mgr      AS "Főnök",
       MIN(sal) AS "Legkisebb fizetés"
  FROM emp
 WHERE mgr IS NOT NULL
 GROUP BY mgr
 HAVING MIN(sal)< 2000
 ORDER BY "Legkisebb fizetés" ASC;
```

**Eredmény**

Főnök	Legkisebb fizetés
7902	800
7698	950
7788	1100
7782	1300

7902	800
7698	950
7788	1100
7782	1300

**2.20. feladat**

Listázza ki főnökönként (mgr) a főnökhöz tartozó dolgozói átlagfizetéseket. Hagyja ki azon dolgozók fizetését, akiknek nincs főnökük, valamint azokat a csoportokat, ahol az átlagfizetés nagyobb 3000 USD-nál. Rendezze a listát az átlagfizetések szerint csökkenően.

**Megoldás**

```
SELECT mgr      AS "Főnök",
       ROUND(AVG(sal)) AS "Átlagfizetés"
  FROM emp
 WHERE mgr IS NOT NULL
 GROUP BY mgr
```

```
HAVING ROUND(AVG(sal)) < 3000
ORDER BY ROUND(AVG(sal)) DESC;
```

**Eredmény**

Főnök Átlagfizetés

7839	2758
7698	1310
7782	1300
7788	1100
7902	800

**2.21. feladat**

Listázza ki főnökönként a főnökhöz tartozó dolgozók jövedelme közül a legnagyobbat. Hagyja ki a listákészítésből azon dolgozókat, akiknek nincs jutalékuk, valamint azokat a (legnagyobb) jövedelemeket, melyek nagyobbak 3500 USD-nál. Rendezze a listát a legnagyobb jövedelem szerint csökkenően.

**Megoldás**

```
SELECT mgr AS "FőnökAzon",
       MAX(sal+NVL(comm,0)) AS "Legnagyobb jövedelem"
  FROM emp
 WHERE comm IS NOT NULL
 GROUP BY mgr
 HAVING (MAX(sal+NVL(comm,0))) <= 3500
 ORDER BY "Legnagyobb jövedelem" DESC;
```

**Eredmény**

FőnökAzon Legnagyobb jövedelem

7698	2650
------	------

**2.22. feladat**

Listázza ki részlegenként az egy tizedesre kerekített átlagfizetéseket. Hagyja ki az átlag meghatározásából az 1981. január 1-je előtt belépett dolgozókat, valamint azon részlegek átlagfizetését, melyekben a legkisebb fizetés kisebb 1000 USD-nál. Rendezze a listát az átlagfizetések szerint növekvően.

**Megoldás**

```
SELECT deptno AS "Részleg",
       (ROUND(AVG(sal),1)) AS "ÁtlagfizetésMin"
  FROM emp
```

```

WHERE hiredate > '81-JAN-01'
GROUP BY deptno
HAVING MIN(sal) >=1000
ORDER BY "ÁtlagfizetésMin";

```

**Eredmény**

Részleg ÁtlagfizetésMin

20	2518.8
10	2916.7

**2.23. feladat**

Listázza ki munkakörönként a dolgozók számát és az egész értékre kerekített átlagfizetésüket numerikusan és grafikusan is. Ez utóbbit csillag (\*) karakterek sorozataként balra igazítva jelemezze meg oly módon, hogy e sorozatban 200 US dolláronként egy csillag karakter álljon. Rengetesse a listát az átlagfizetések szerint csökkenően.

**1. megoldás**

```
SET linesize 70
```

```

SELECT LPAD(COUNT(*),5,' ') || RPAD(' ',10) ||
       ROUND(AVG(SAL))||RPAD (' ',ROUND(AVG(sal))/200+1, '*') AS
      "Dolgozók száma, Átlagfizetés "
      FROM emp
      GROUP BY job
      ORDER BY ROUND(AVG(SAL));

```

```
SET linesize 400
```

**Eredmény**

Dolgozók száma, Átlagfizetés

4	1038 *****
4	1400 *****
3	2758*****
2	3000 *****
1	5000 *****

**Megjegyzés**

Az LPAD és RPAD karakterkezelő függvények minden esetben egy teljes sorra vonatkoznak. Ezért az ebben listázandó oszlopfejezéset össze kell fűzni (az || operátorral), és az oszlopfejezések között a megfelelő értelmezéshez szükséges üres karaktereket szintén a két karakterkezelő függvénnyel lehet beiktatni. A kitöltendő oszloból egy karak-

ter az oszlop nevét szimbolizáló üres karakter, ezért a végén még 1-et hozzá kell adnunk. Ha visszövel választjuk el az oszlopkifejezéseket, akkor új sorban fogja listázni ezeket. Azonban ha a SUBSTR alsztringkezelő függvényt használjuk arra, hogy egy teljes sorból kivonjuk a listsort, akkor ezután már listázhatók az oszlopkifejezések a szokásos módon. Listázzuk ki a foglalkozást is.

## 2. megoldás (Kibővített feladat megoldása)

```
SET linesize 70
```

```
SELECT SUBSTR(LPAD(COUNT(*),5,' ') || RPAD(' ',10) ||
    ROUND(AVG(SAL))||RPAD (' ',ROUND(AVG(sal))/200+1, '#'),0,50) AS
    "Dolgozók száma Átlagfizetés ", job AS munkakör
FROM emp
GROUP BY job
ORDER BY ROUND(AVG(SAL));
```

```
SET linesize 400
```

### Eredmény

Dolgozók száma	Átlagfizetés	MUNKAKÖR
4	1038 #####	CLERK
4	1400 #####	SALESMAN
3	2758 #####	MANAGER
2	3000 #####	ANALYST
1	5000 #####	PRESIDENT

## 2.24. feladat

Listázza ki főnökönként a legrégebbi óta munkaviszonyban álló dolgozóknak a mai napig munkában töltött éveinek számát numerikusan és grafikusan is. Ez utóbbit kettőskereszt (#) karakterek sorozataként balra igazítva jelenítse meg oly módon, hogy ebben a sorozatban 5 évenként egy kettőskereszt karakter álljon. Rendezze a listát az évek száma szerint növekvően.

### Megoldás

```
SET linesize 30
```

```
SELECT NVL(TO_CHAR(mgr),'      ')||RPAD(' ',5)|||
    Max(hiredate)|| RPAD(' ',5)|||
    RPAD(' ',
        MAX(ROUND(MONTHS_BETWEEN(sysdate,hiredate)/12))/5+1,'#')
    AS "Főnök      Legrégebben dolgozó"
FROM emp
GROUP BY mgr
```

```
ORDER BY MAX(ROUND(MONTHS_BETWEEN(sysdate,hiredate)/12)) ASC;
```

```
SET linesize 400
```

### Eredmény

Főnök	Legrégebben dolgozó
7788	87-MÁJ-23 #####
7566	87-ÁPR-19 #####
	81-NOV-17 #####
7782	82-JAN-23 #####
7698	81-DEC-03 #####
7839	81-JÚN-09 #####
7902	80-DEC-17 #####



#### Megjegyzés

Az a dolgozó van a vállalatnál legrégebben, akinek a munkában itt eltöltött éveinek száma a legnagyobb. Ez könnyen kiszámítható az aktuális dátumértéket megadó sysdate rendszerváltozó, valamint a MONTHS\_BETWEEN dátumkezelő függvény segítségével. Ezt az eredményt (hónapok számát) osztani kell 12-vel és megkapjuk az évek számát. Megoldható azonban a feladat dátumkezelő függvény nélkül is. Ekkor a mai nap dátumából (sysdate) levonjuk a belépési dátumot (hiredate) és az eredményt osztjuk egy év napjainak számával (365). A kapott évek számának legnagyobb értékeit listázzuk a sorrendnek megfelelően. (Gondolkodjon el azon, hogy miként lehetne figyelembe venni a szökőéveket.)

### 3. FEJEZET

# Többtáblás lekérdezések, allekérdezések

## Feladatok és megoldások

### 3.1. feladat

Listázza ki azon dolgozók nevét és részlegük nevét, akiknek nevében az A betű szerepel.

#### Megoldás

```
SELECT ename AS név,  
       dname AS részlegnév  
  FROM emp dolg, dept d  
 WHERE ename LIKE '%A%' AND  
       dolg.deptno = d.deptno;
```

#### Eredmény

NÉV	RÉSZLEGNÉV
ALLEN	SALES
WARD	SALES
MARTIN	SALES
BLAKE	SALES
CLARK	ACCOUNTING
ADAMS	RESEARCH
JAMES	SALES

7 sor kijelölve.

### 3.2. feladat

Listázza ki a Dallas-i telephely minden dolgozójának nevét, munkakörét, fizetését és részlegének azonosítóját.

**Megoldás**

```
SELECT ename, job, sal, dept.deptno
  FROM emp, dept
 WHERE emp.deptno = dept.deptno AND
       UPPER(dept.loc) = 'DALLAS';
```

**Eredmény**

ENAME	JOB	SAL	DEPTNO
SMITH	CLERK	800	20
JONES	MANAGER	2975	20
SCOTT	ANALYST	3000	20
ADAMS	CLERK	1100	20
FORD	ANALYST	3000	20

**3.3. feladat**

Listázza ki a clerk munkakörű dolgozókat foglalkoztató részlegek azonosítóját, nevét és telephelyét. A lista a részlegnév szerint legyen rendezve.

**1. Megoldás**

```
SELECT DISTINCT dept.deptno AS részleg,
                  dname      AS részlegnév,
                  loc        AS telephely
  FROM emp, dept
 WHERE UPPER(job) = 'CLERK' AND
       emp.deptno = dept.deptno
 ORDER BY részlegnév;
```

**2. megoldás (allekérdezés a FROM utasításrészben)**

```
SELECT dept.deptno AS részleg,
      dname      AS részlegnév,
      loc        AS telephely
  FROM dept,
       (SELECT deptno
        FROM emp
       WHERE UPPER(job) = 'CLERK'
       GROUP BY deptno) al
 WHERE al.deptno = dept.deptno
 ORDER BY részlegnév;
```

**3. megoldás (allekérdezés a WHERE utasításrészben)**

```
SELECT dept.deptno AS részleg,
      dname      AS részlegnév,
```

```

    loc          AS telephely
  FROM dept
 WHERE deptno IN
      (SELECT deptno
        FROM emp
       WHERE UPPER(job) = 'CLERK'
      GROUP BY deptno)
 ORDER BY részlegnév;

```

**Eredmény (mindhárom megoldás esetén)**

RÉSZLEG	RÉSZLEGNÉV	TELEPHELY
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO

**3.4. feladat**

Listázza ki a Dallas-ban és Chicago-ban dolgozók nevét, munkakörét és telephelyét. A lista telephely szerint legyen rendezett.

**1. megoldás**

```

SELECT ename, job, loc
  FROM emp, dept
 WHERE UPPER(loc) IN ('DALLAS', 'CHICAGO') AND
       emp.deptno = dept.deptno
 ORDER BY loc;

```

**2. megoldás**

```

SELECT ename, job, loc
  FROM emp, dept
 WHERE emp.deptno IN
      (SELECT deptno
        FROM dept
       WHERE UPPER(loc) = 'DALLAS' OR
             UPPER(loc) = 'CHICAGO')
      AND dept.deptno = emp.deptno
 ORDER BY loc;

```

**Eredmény (mindkét megoldás esetén)**

ENAME	JOB	LOC
ALLEN	SALESMAN	CHICAGO
WARD	SALESMAN	CHICAGO

MARTIN	SALESMAN	CHICAGO
BLAKE	MANAGER	CHICAGO
TURNER	SALESMAN	CHICAGO
JAMES	CLERK	CHICAGO
SMITH	CLERK	DALLAS
JONES	MANAGER	DALLAS
ADAMS	CLERK	DALLAS
FORD	ANALYST	DALLAS
SCOTT	ANALYST	DALLAS

11 sor kijelölve.

### 3.5. feladat

Listázza ki az egyes részlegek nevét, telephelyük címét, dolgozóik átlagfizetését a részlegnevek szerint rendezve.

#### 1. megoldás (Egyeszerű, de kéttáblás)

```

SELECT d.dname           AS "RészlegNév",
       d.loc            AS "Telephely",
       ROUND(AVG(e.sal),0) AS "RészlegÁtlag"
  FROM emp e, dept d
 WHERE e.deptno = d.deptno
 GROUP BY d.dname, d.loc
 ORDER BY d.dname;

```

#### 2. megoldás (FROM-beli allekérdezéssel )

```

SELECT dname      AS "RészlegNév",
       loc        AS "Telephely",
       at átlag   AS "RészlegÁtlag"
  FROM dept,
       (SELECT ROUND(AVG(sal),0) AS átlag,
              deptno
     FROM emp
    GROUP BY deptno) at
 WHERE dept.deptno = at.deptno
 ORDER BY dname;

```

#### Eredmény (mindkét megoldás esetén)

RészlegNév	Telephely	RészlegÁtlag
ACCOUNTING	NEW YORK	2917
RESEARCH	DALLAS	2175
SALES	CHICAGO	1567

### 3.6. feladat

Listázza ki a 20-as és a 30-as részleg legnagyobb fizetésű dolgozóinak azonosítóját, nevét, foglalkozását, jutalékát és belépési dátumát.

#### 1. megoldás (Allekérdezés a WHERE logikai kifejezés jobb oldalán)

```

SELECT empno      AS azonosító,
       ename       AS név,
       job        AS foglalkozás,
       sal         AS fizetés,
       comm        AS jutalék,
       hiredate    AS belépés
  FROM emp e1
 WHERE sal = (SELECT MAX(sal)
                FROM emp e2
               WHERE e1.deptno = e2.deptno
                 GROUP BY deptno
                HAVING deptno IN (20, 30));

```

#### 2. megoldás (Allekérdezés a FROM utasításrészben – inline nézet)

```

SELECT empno      AS azonosító,
       ename       AS név,
       job        AS foglalkozás,
       legnagyobb.MaxFiz AS fizetés,
       comm        AS jutalék,
       hiredate    AS belépés
  FROM emp,
       (SELECT MAX(sal) AS MaxFiz
          FROM emp
         GROUP BY deptno
        HAVING deptno IN (20, 30)) legnagyobb
 WHERE sal = legnagyobb.MaxFiz;

```

#### Eredmény (mindkét megoldás esetén)

AZONOSÍTÓ NÉV	FOGLALKOZ	FIZETÉS	JUTALÉK BELÉPÉS
7698 BLAKE	MANAGER	2850	81-MÁJ-01
7788 SCOTT	ANALYST	3000	87-ÁPR-19
7902 FORD	ANALYST	3000	81-DEC-03

### 3.7. feladat

Listázza ki minden részleg legkisebb jövedelmű dolgozójának azonosítóját, nevét, foglalkozását, jutalékát és belépési dátumát.

**Megoldás**

```

SELECT empno          AS azonosító,
       ename           AS név,
       job             AS foglalkozás,
       comm            AS jutalék,
       hiredate        AS belépés,
       min_fiz.legkisebb AS kicsifiz
  FROM emp,
       (SELECT MIN(sal+NVL(comm,0)) AS legkisebb,
              deptno
     FROM emp
    GROUP BY deptno) min_fiz
 WHERE (sal+NVL(comm,0)) = min_fiz.legkisebb AND
       emp.deptno = min_fiz.deptno;

```

**Eredmény**

AZONOSÍTÓ NÉV	FOGLALKOZ	JUTALÉK	BELÉPÉS	KICSIFIZ
7934 MILLER	CLERK		82-JAN-23	1300
7369 SMITH	CLERK		80-DEC-17	800
7900 JAMES	CLERK		81-DEC-03	950

**Megjegyzés**

A jövedelemegyezés nem elegendő feltétel, mert lehet egy másik részlegén is ugyanilyen fizetésű dolgozó. Így az allekérdezésben szükség van a deptno egyezések vizsgálatára is.

**3.8. feladat**

Listázza ki azon részlegek nevét és telephelyét, ahol a dolgozók átlagjövedelme kisebb, mint 2200 USD.

**Megoldás****1. lépés (Minden csoportnak kiszámoljuk az átlagfizetését)**

```

SELECT d.dname          AS részlegnév,
       d.loc            AS telephely,
       ROUND(AVG(e.sal+NVL(comm,0)),0) AS részlegátlag
  FROM emp e, dept d
 WHERE e.deptno = d.deptno
 GROUP BY d.dname,d.loc;

```

**Eredmény**

RÉSZLEGNÉV	TELEPHELY	RÉSZLEGÁTLAG
ACCOUNTING	NEW YORK	2917
RESEARCH	DALLAS	2175
SALES	CHICAGO	1933

**2. lépés (Megadjuk a csoportfeltételt – HAVING-ben nem lehet másodlagos név!)****2A. Allekérdezéssel**

```

SELECT dname      AS részlegnév,
       loc        AS telephely,
       at.atlag   AS részlegátlag
  FROM dept,
       (SELECT ROUND(AVG(sal+NVL(comm,0))) AS atlag,
              deptno
     FROM emp
    GROUP BY deptno
   HAVING ROUND(AVG(sal+NVL(comm,0))) < 2200) at
 WHERE dept.deptno = at.deptno;

```

**Eredmény**

RÉSZLEGNÉV	TELEPHELY	RÉSZLEGÁTLAG
RESEARCH	DALLAS	2175
SALES	CHICAGO	1933

**2B. Allekérdezés nélkül**

```

SELECT dname          AS részlegnév,
       loc            AS telephely,
       ROUND(AVG(sal+NVL(comm,0))) AS részlegátlag
  FROM emp e, dept d
 WHERE e.deptno = d.deptno
 GROUP BY dname, loc
 HAVING ROUND(AVG(sal+NVL(comm,0))) < 2200;

```

**Eredmény**

RÉSZLEGNÉV	TELEPHELY	RÉSZLEGÁTLAG
SALES	CHICAGO	1933
RESEARCH	DALLAS	2175

**Megjegyzés**

Allekérdezéssel hatékonyabb megoldást kapunk, mivel kisebb táblákat kell feldolgozni.

### 3.9. feladat

Írjon olyan lekérdezést, ami megadja az összes, jutalékkal rendelkező alkalmazott nevét, részlegének nevét és helyét.

#### Megoldás

```
SELECT emp.ename,dept.dname, dept.loc
  FROM emp, dept
 WHERE emp.comm IS NOT NULL
   AND dept.deptno = emp.deptno;
```

#### Eredmény

ENAME	DNAME	LOC
MARTIN	SALES	CHICAGO
ALLEN	SALES	CHICAGO
TURNER	SALES	CHICAGO
WARD	SALES	CHICAGO

### 3.10. feladat

Listázza ki a dolgozók nevét és azonosítóját a főnökük (mgr) nevével és azonosítójával együtt úgy, hogy akinek nincs főnöke, annak a NULL érték helyére a Legfőbb karaktersorozatot írja.

#### 1. megoldás (Külső összekapcsolással)

```
SELECT dolgozó.ename          AS "DolgozóNeve",
       dolgozó.empno         AS "DolgozóAzonosítója",
       főnök.ename           AS "FőnökNeve",
       NVL(TO_CHAR(főnök.empno), 'Legfőbb') AS "FőnökAzonosítója"
  FROM emp dolgozó,
       emp főnök
 WHERE dolgozó.mgr = főnök.empno (+)
 ORDER BY főnök.ename;
```

#### Eredmény

DolgozóNeve	DolgozóAzonosítója	FőnökNeve	FőnökAzonosítója
ALLEN	7499	BLAKE	7698
WARD	7521	BLAKE	7698
MARTIN	7654	BLAKE	7698
JAMES	7900	BLAKE	7698
TURNER	7844	BLAKE	7698
MILLER	7934	CLARK	7782
SMITH	7369	FORD	7902

SCOTT	7788 JONES	7566
FORD	7902 JONES	7566
JONES	7566 KING	7839
BLAKE	7698 KING	7839
CLARK	7782 KING	7839
ADAMS	7876 SCOTT	7788
KING	7839	Legfőbb

14 sor kijelölve.

## 2. megoldás (JOIN használatával)

```

SELECT dolgozó.ename          AS "DolgozóNeve",
       dolgozó.empno        AS "DolgozóAzonosítója",
       főnök.ename          AS "FőnökNeve",
       NVL(TO_CHAR(főnök.empno), 'Legfőbb') AS "FőnökAzonosítója"
  FROM emp dolgozó
    LEFT JOIN emp főnök ON dolgozó.mgr = főnök.empno
 ORDER BY főnök.ename;

```

### Eredmény

DolgozóNev	DolgozóAzonosítója	FőnökNeve	FőnökAzonosítója
JAMES	7900	BLAKE	7698
TURNER	7844	BLAKE	7698
MARTIN	7654	BLAKE	7698
WARD	7521	BLAKE	7698
ALLEN	7499	BLAKE	7698
MILLER	7934	CLARK	7782
SMITH	7369	FORD	7902
FORD	7902	JONES	7566
SCOTT	7788	JONES	7566
CLARK	7782	KING	7839
JONES	7566	KING	7839
BLAKE	7698	KING	7839
ADAMS	7876	SCOTT	7788
KING	7839		Legfőbb

14 sor kijelölve.



### Megjegyzés (Teljes és részleges rendezés)

A két megoldás listája közötti eltérést az okozza, hogy a rendezési előírás csak részleges volt, azaz nem írt elő az összes rekordra vonatkozó rendezést. Így megjelent a két megoldás kiszámítási módszere közötti különbség. Ha az ORDER BY "FőnökNeve" utasításrész helyett az ORDER BY "FőnökNeve", "DolgozóNeve" utasításrészt alkalmazzuk, akkor

a két megoldás azonos listát adott volna, amelyik pedig gyakorlati szempontból is megfelelőbb lett volna. Célszerű tehát a megoldásainkban teljes rendezést előírni.

### 3.11. feladat

Listázza ki a NEW YORK telephely minden dolgozójának nevét, azonosítóját, jövedelmét és főnökének nevét, telephelyét.

#### 1. megoldás (Külső összekapcsolással)

```
SELECT dolgozó.ename          AS név,
       dolgozó.empno         AS azonosító,
       dolgozó.sal+NVL(dolgozó.comm,0) AS jövedelem,
       főnök.ename           AS FőnökNeve,
       loc                   AS Telephely
  FROM emp dolgozó,
       emp főnök,
       dept
 WHERE dolgozó.mgr = főnök.empno(+)
   AND dolgozó.deptno = dept.deptno
   AND UPPER(loc) = 'NEW YORK' ;
```

#### 2. megoldás (JOIN összekapcsolással)

```
SELECT dolgozó.ename          AS név,
       dolgozó.empno         AS azonosító,
       dolgozó.sal+NVL(dolgozó.comm,0) AS jövedelem,
       főnök.ename           AS FőnökNeve,
       loc                   AS Telephely
  FROM emp dolgozó
  LEFT JOIN emp főnök ON dolgozó.mgr = főnök.empno
  INNER JOIN dept ON dolgozó.deptno = dept.deptno
 WHERE UPPER(loc) = 'NEW YORK';
```

#### Eredmény (mindkét megoldás esetén)

NÉV	AZONOSÍTÓ	JÖVEDELEM	FŐNÖKNEVE	TELEPHELY
CLARK	7782	2450	KING	NEW YORK
KING	7839	5000		NEW YORK
MILLER	7934	1300	CLARK	NEW YORK

### 3.12. feladat

Listázza ki minden alkalmazott nevét, részlegének nevét és fizetését, akiknek fizetése meggyezik valamelyik Dallas-ban dolgozó alkalmazottával. Legyen a lista fejléce név, részleg neve, fizetés, és a lista legyen a fizetés és a részleg neve szerint rendezett.

**Megoldás**

```

SELECT e.ename AS név,
       d.dname AS "Részleg neve",
       e.sal   AS fizetés
  FROM emp e,
       dept d
 WHERE (d.deptno = e.deptno) AND
       sal IN
          (SELECT sal
             FROM emp e,
                  dept d
            WHERE UPPER(loc) = 'DALLAS' AND
                  d.deptno = e.deptno)
 ORDER BY sal, dname;

```

**Eredmény**

NÉV	Részleg neve	FIZETÉS
SMITH	RESEARCH	800
ADAMS	RESEARCH	1100
JONES	RESEARCH	2975
SCOTT	RESEARCH	3000
FORD	RESEARCH	3000

**3.13. feladat**

Listázza ki azokat a dolgozókat, akiknek neve hasonlítható egy munkakör nevéhez.

**Megoldás**

```

SELECT DISTINCT
           ename          AS Név,
           similar.Munkakör AS Hasonló
      FROM emp,
           (SELECT job AS Munkakör
              FROM emp) similar
     WHERE SOUNDEX(UPPER(ename)) =
           SOUNDEX(UPPER(similar.Munkakör));

```

**Eredmény**

NÉV	HASONLÓ
CLARK	CLERK



### Megjegyzés

A feladat könnyen megoldható a SOUNDEX karakteres függvény használatával. Ez a függvény a *hangzás alapján* (nyilván egy megfelelő belső algoritmus szerint) előállít egy értéket. Az azonos karaktersorozathoz azonos értéket állít elő. Ez a függvény felhasználható a hangzás alapú hasonlósági vizsgálatokhoz. Az összehasonlítható karakterláncok kezdőbetűjének minden esetben azonosnak kell lennie.

## 3.14. feladat

Listázza ki azoknak a főnököknek az azonosítóját, akik nem menedzser foglalkozásúak. A lista a főnök azonosítója (mgr) szerint legyen rendezett.

### 1. megoldás (Allekérdezéssel)

```
SELECT főnök.mgr, dolgozó.job
  FROM emp dolgozó,
       (SELECT mgr
        FROM emp
        GROUP BY mgr) főnök
 WHERE dolgozó.empno = főnök.mgr AND
       UPPER(dolgozó.job) != 'MANAGER';
```

### Eredmény

MGR	JOB
7788	ANALYST
7839	PRESIDENT
7902	ANALYST

-----  
 7788 ANALYST  
 7839 PRESIDENT  
 7902 ANALYST

### 2. megoldás (Halmazművevekkel – INTERSECT)

```
SELECT mgr
  FROM emp
  GROUP BY mgr
INTERSECT
SELECT empno
  FROM emp
 WHERE UPPER(job) != 'MANAGER';
```

### Eredmény

MGR
7788
7839
7902

-----  
 7788  
 7839  
 7902

### 3.15. feladat

Hány olyan főnök van, aki nem menedzser foglalkozású?

#### Megoldás

```
SELECT COUNT(*) AS "Nem manager főnökök száma"
  FROM emp dolgozó,
       (SELECT mgr
        FROM emp
        GROUP BY mgr) főnök
 WHERE dolgozó.empno = főnök.mgr AND
       UPPER(dolgozó.job) != 'MANAGER';
```

#### Eredmény

Nem manager főnökök száma

-----  
3

### 3.16. feladat

Listázza ki a főnökeik szerint csoportosítva a legkisebb jövedelmű dolgozókat. Hagya ki azokat a dolgozókat, akiknek nincs főnökük, valamint azokat a csoportokat, ahol a legkisebb jövedelem nagyobb 3000 USD-nál. Rendezze a listát a legkisebb jövedelmek szerint növekvően.

#### Megoldás

##### 1. lépés (A szükséges allekérdezés)

```
SELECT mgr                               AS Főnöke,
       MIN(sal+NVL(comm,0))   AS Jövedelme
  FROM emp
 WHERE mgr IS NOT NULL
 GROUP BY mgr
 HAVING MIN(sal+NVL(comm,0)) <= 3000
 ORDER BY Jövedelme;
```

#### Eredmény

FŐNÖKE JÖVEDELME

7902	800
7698	950
7788	1100
7782	1300
7839	2450
7566	3000

6 sor kijelölve.

## 2. lépés (A teljes lekérdezés előállítása)

```

SELECT ename      AS "Dolgozó",
       mgr        AS "Főnöke_Kódja",
       Jövedelme   AS "Jövedelme (a kicsi)"
  FROM emp,
       (SELECT mgr                  AS Főnöke,
              MIN(sal+NVL(comm,0)) AS Jövedelme
         FROM emp
        WHERE mgr IS NOT NULL
        GROUP BY mgr
       HAVING MIN(sal+NVL(comm,0)) <= 3000)  kisjövedelműek
 WHERE kisjövedelműek.Főnöke = emp.mgr AND
       kisjövedelműek.Jövedelme = emp.sal+NVL(emp.comm,0)
 ORDER BY Jövedelme;

```

### Eredmény

Dolgozó	Főnöke_Kódja	Jövedelme (a kicsi)
SMITH	7902	800
JAMES	7698	950
ADAMS	7788	1100
MILLER	7782	1300
CLARK	7839	2450
SCOTT	7566	3000
FORD	7566	3000

7 sor kijelölve.



### Megjegyzés

- A teljes lekérdezés azért tartalmaz egyelő több sort, mint az allekérdezés, mivel a SCOTT-nak és a FORD-nak egyaránt minimális a jövedelme.
- A feladat megoldásához a külső lekérdezést kell rendezni. Ekkor azonban az allekérdezés rendezése nyilván elhagyható.

## 3.17. feladat

Listázza ki minden olyan dolgozó azonosítóját és nevét, akik olyan részlegen dolgoznak, melyen található névében T betűt tartalmazó dolgozó. Legyen a lista fejléce azonosító, név, részleg helye, és a lista legyen a részleg helye és a név szerint rendezett.

### Megoldás

```

SELECT empno AS AZONOSÍTÓ,
       ename AS NÉV,
       loc  AS "Részleg helye"

```

```

FROM emp, dept
WHERE emp.deptno IN
    (SELECT deptno
     FROM emp
     WHERE ename LIKE '%T%') AND
     emp.deptno = dept.deptno
ORDER BY loc, ename;

```

**Eredmény**

AZONOSÍTÓ NÉV	Részleg helye
7499 ALLEN	CHICAGO
7698 BLAKE	CHICAGO
7900 JAMES	CHICAGO
7654 MARTIN	CHICAGO
7844 TURNER	CHICAGO
7521 WARD	CHICAGO
7876 ADAMS	DALLAS
7902 FORD	DALLAS
7566 JONES	DALLAS
7788 SCOTT	DALLAS
7369 SMITH	DALLAS

11 sor kijelölve.

**3.18. feladat**

Listázza ki a főnökeik (mgr) szerint csoportosítva azokat a dolgozókat, akiknek fizetése e csoportosítás szerint a legkisebb, de nagyobb 1000 USD-nál. A lista a fizetés növekvő értéke szerint legyen rendezett. Legyen a lista fejléce: főnök kódja, dolgozónév, fizetés.

**Megoldás**

```

SELECT al.mgr AS főnök_kódja,
       ename AS dolgozónév,
       sal   AS fizetés
  FROM emp,
       (SELECT mgr,
              MIN(sal) AS legkisebb
         FROM emp
        GROUP BY mgr
       HAVING MIN(sal) > 1000) al
 WHERE al.mgr = emp.mgr AND
       sal = al.legkisebb
 ORDER BY al.legkisebb;

```

**Eredmény**

FŐNÖK_KÓDJA	DOLGOZÓNÉV	FIZETÉS
7788	ADAMS	1100
7782	MILLER	1300
7839	CLARK	2450
7566	FORD	3000
7566	SCOTT	3000

**3.19. feladat**

Listázza ki azon főnökönél (mgr) a legkisebb és legnagyobb fizetéseket, melyeknél a legkisebb fizetések 3000 USD-nál alacsonyabbak. A listát a legkisebb fizetés szerint rendezze, a fejléc pedig legyen Főnök kódja, Legkisebb fizetés és Legnagyobb fizetés.

**Megoldás**

```
SELECT mgr      AS "Főnök kódja",
       MIN(sal) AS "Legkisebb fizetés",
       MAX(sal) AS "Legnagyobb fizetés"
  FROM emp
 WHERE mgr IS NOT NULL
 GROUP BY mgr
 HAVING MIN(sal) < 3000
 ORDER BY "Legnagyobb fizetés";
```

**Eredmény**

Főnök kódja	Legkisebb fizetés	Legnagyobb fizetés
7902	800	800
7788	1100	1100
7782	1300	1300
7698	950	1600
7839	2450	2975

Főnök kódja	Legkisebb fizetés	Legnagyobb fizetés
7902	800	800
7788	1100	1100
7782	1300	1300
7698	950	1600
7839	2450	2975

**3.20. feladat**

Listázza ki a fizetés szerint csökkenő sorba rendezve az eladók (salesman) és a hivatalnokok (clerk) főnökeinek nevét és fizetését, az egyes dolgozók saját nevét, munkakörét, fizetését, valamint az egyes dolgozók saját fizetése/főnök fizetése arányát, elsődlegesen a főnök neve szerint, másodlagosan a fizetés aránya szerint rendezve.

### 1. megoldás

```

SELECT F.ename      AS "Főnök",
       F.sal        AS "FőnökFiz",
       D.ename      AS "DolgozóNeve",
       D.job        AS "Munkakör",
       D.sal        AS "Fizetés",
       D.sal/F.sal AS "FizArány"

FROM emp D,
     emp F
WHERE D.mgr = F.empno AND
      UPPER(D.job) IN ('SALESMAN', 'CLERK')
ORDER BY F.ename, D.ename;

```

### Eredmény

Főnök	FőnökFiz	DolgozóNeve	Munkakör	Fizetés	FizArány
BLAKE	2850	ALLEN	SALESMAN	1600	.561403509
BLAKE	2850	JAMES	CLERK	950	.333333333
BLAKE	2850	MARTIN	SALESMAN	1250	.438596491
BLAKE	2850	TURNER	SALESMAN	1500	.526315789
BLAKE	2850	WARD	SALESMAN	1250	.438596491
CLARK	2450	MILLER	CLERK	1300	.530612245
FORD	3000	SMITH	CLERK	800	.266666667
SCOTT	3000	ADAMS	CLERK	1100	.366666667

8 sor kijelölve.

### 2. megoldás (Formázott kiírással)

```

SELECT SUBSTR(F.ename,1,5)          AS "Főnök",
       LPAD(F.sal,8)           AS "FőnökFiz",
       RPAD(D.ename,11)         AS "DolgozóNeve",
       SUBSTR(D.job,1,8)        AS "Munkakör",
       LPAD(D.sal,7)           AS "Fizetés",
       TO_CHAR(D.sal/F.sal,'9990.99') AS "FizArány"

FROM emp D,
     emp F
WHERE D.mgr = F.empno AND
      UPPER(D.job) IN ('SALESMAN', 'CLERK')
ORDER BY F.ename, D.ename;

```

**Eredmény**

Főnök	FőnökFiz	DolgozóNeve	Munkakör	Fizetés	FizArány
BLAKE	2850	ALLEN	SALESMAN	1600	0.56
BLAKE	2850	JAMES	CLERK	950	0.33
BLAKE	2850	MARTIN	SALESMAN	1250	0.44
BLAKE	2850	TURNER	SALESMAN	1500	0.53
BLAKE	2850	WARD	SALESMAN	1250	0.44
CLARK	2450	MILLER	CLERK	1300	0.53
FORD	3000	SMITH	CLERK	800	0.27
SCOTT	3000	ADAMS	CLERK	1100	0.37

8 sor kijelölve.

## Összetett feladatok és megoldások

### 3.21. feladat

Listázza ki a Chicago-i telephelyű főnök nevét, azonosítóját, munkakörét, fizetését, beosztott-jainak átlagfizetését és annak szórását és varianciáját.

#### Megoldás

```

SELECT főnök.ename           AS "Főnök neve",
       főnök.empno          AS "azonosítója",
       főnök.job             AS "munkaköre",
       főnök.sal              AS "fizetése",
       ROUND(dolgozó.átlag)   AS "Dolgozói átlag",
       ROUND(dolgozó.szórás)  AS "szórás",
       ROUND(dolgozó.variancia) AS "variancia"
FROM emp főnök,
     dept,
     (SELECT mgr,
            AVG(sal)      AS átlag,
            STDDEV(sal)   AS szórás,
            VARIANCE(sal) AS variancia
     FROM emp
     GROUP BY mgr) dolgozó
WHERE főnök.empno = dolgozó.mgr    AND
      főnök.deptno = dept.deptno    AND
      UPPER(loc) = 'CHICAGO';
  
```

**Eredmény**

Főnök neve azonosítója munkaköre	fizetése Dolgozói átlag	szórás	variancia
BLAKE	7698 MANAGER	2850	1310

**Megjegyzés**

Figyeljünk fel arra, hogy e feladatban az átlagot, a szórást és a varianciát csak allekérdezésben számíthattuk ki.

**3.22. feladat**

Listázza ki a 2000 és 4000 USD közötti fizetésű főnökök nevét, fizetését, telephelyét és beosztottjainak átlagfizetését a főnök neve szerint rendezve.

**Megoldás**

```

SELECT főnök.ename           AS "FőnökNév",
       főnök.sal            AS "FőnökFizetés",
       főnök.job             AS "FőnökFoglalkozás",
       dept.loc              AS "FőnökTelephely",
       ROUND(dolgozó.átlag)  AS "DolgozóiÁtlag"
  FROM emp főnök,
       dept,
       (SELECT AVG(sal) AS átlag,
               mgr
      FROM emp
     GROUP BY mgr) dolgozó
 WHERE (főnök.deptno = dept.deptno)      AND
       (főnök.sal BETWEEN 2000 AND 4000)    AND
       (dolgozó.mgr = főnök.empno)
 ORDER BY főnök.ename;

```

**Eredmény**

FőnökNév	FőnökFizetés	FőnökFoglalkozás	FőnökTelephely	DolgozóiÁtlag
BLAKE	2850	MANAGER	CHICAGO	1310
CLARK	2450	MANAGER	NEW YORK	1300
FORD	3000	ANALYST	DALLAS	800
JONES	2975	MANAGER	DALLAS	3000
SCOTT	3000	ANALYST	DALLAS	1100

### 3.23. feladat

Listázza ki minden részleg legkisebb jövedelmű dolgozójának azonosítóját, nevét, foglalkozását, részlegének azonosítóját, telephelyét és munkában eltöltött éveinek számát. Legyen a lista a munkában töltött évek szerint rendezve.

#### Megoldás

```

SELECT empno AS "Azonosító",
       ename AS "Név",
       job   AS "Foglalkozás",
       dept.deptno AS "Részleg",
       ROUND(MONTHS_BETWEEN(sysdate,hiredate)/12) AS "Évek Száma",
       min_jöv.legkisebb AS "KicsiJöv",
       loc   AS "Telephely"
  FROM emp,
       dept,
       (SELECT MIN(sal+NVL(comm,0)) AS legkisebb,
              deptno
     FROM emp
    GROUP BY deptno) min_jöv
 WHERE (sal+NVL(comm,0) = min_jöv.legkisebb) AND
       (emp.deptno = min_jöv.deptno)          AND
       (emp.deptno = dept.deptno)
 ORDER BY "Évek Száma";

```

#### Eredmény

Azonosító	Név	Foglalkoz	Részleg	Évek Száma	KicsiJöv	Telephely
7934	MILLER	CLERK		10	23	1300 NEW YORK
7900	JAMES	CLERK		30	23	950 CHICAGO
7369	SMITH	CLERK		20	24	800 DALLAS

### 3.24. feladat

Listázza ki az egyes részlegek telephelyének nevét, a részleg dolgozónak egész értékre kerekített átlagjövedelmét, valamint az itt dolgozók főnökeinek nevét, fizetését és telephelyét az átlagjövedelem szerint rendezve, és a részlegadatokat ismétlésmentesen megjelenítve.

#### Megoldás

##### 1. lépés (Listázzuk a részlegek azonosítóját és az itt dolgozók átlagjövedelmét)

```

SELECT deptno,
       ROUND(AVG(sal+NVL(comm,0))) AS ÁtlagJövedelem
  FROM emp
 GROUP BY deptno;

```

**Eredmény**

DEPTNO ÁTLAGJÖVEDELEM

10	2917
20	2175
30	1933

**2. lépés** (Listázzuk ki a részlegek azonosítóját, nevét, és az itt dolgozók átlagjövedelmét.  
Ehhez használjuk fel az előző lekérdezést. Az összekötést a JOIN kulcsszóval végezzük)

**2A. A USING kulcsszó használatával**

```
SELECT deptno      AS RészlegDeptno,
       loc        AS RészlegTelephely,
       ÁtlagJövedelem AS RészlegÁtlag
  FROM dept
    INNER JOIN
      (SELECT emp.deptno,
              ROUND(AVG(sal+NVL(comm,0))) AS ÁtlagJövedelem
     FROM emp
    GROUP BY deptno) USING (deptno);
```

**2B. Az ON kulcsszó használatával**

```
SELECT dept.deptno      AS RészlegDeptno,
       dept.loc        AS RészlegTelephely,
       subl.ÁtlagJövedelem AS RészlegÁtlag
  FROM dept
    INNER JOIN
      (SELECT emp.deptno,
              ROUND(AVG(sal+NVL(comm,0))) AS ÁtlagJövedelem
     FROM emp
    GROUP BY deptno) subl ON (dept.deptno = subl.deptno);
```

**Eredmény (mindkét megoldás esetén)**

RÉSZLEGDEPTNO RÉSZLEGTELEPH RÉSZLEGÁTLAG

10	NEW YORK	2917
20	DALLAS	2175
30	CHICAGO	1933

**Megjegyzés**

Az ON kulcsszó használata esetén az allekérdezést névvel láthattuk el. A USING esetén ezt nem lehetjük, azonban a SELECT lista egyszerűbbé tehető, ugyanis nincs szükség minősítők használatára.

**3. lépés (Listázzuk ki részlegenként az ott dolgozók főnökeit)**

```
SELECT DISTINCT
    deptno,
    mgr
FROM emp
WHERE mgr IS NOT NULL;
```

**Eredmény**

DEPTNO	MGR
10	7782
10	7839
20	7566
20	7788
20	7839
20	7902
30	7698
30	7839

8 sor kijelölve.

**Megjegyzés**

- A DISTINCT kulcsszó használatára azért volt szükség, mivel ez a lekérdezés tulajdonképpen egy olyan projekció, melyben az eredményt megfosztottuk attól az oszloptól (empno), amelyiknek éppen az egyes sorok megkülönböztetése a feladata (kulcs).
- Láthatóan az egy részlegen dolgozóknál több főnök is van, és egy főnök több részlegen dolgozóknak is főnöke. Ennek legfontosabb tanulsága, hogy az „egy főnök beosztottjai” és az „egy részlegen dolgozó beosztottak” egymást kölcsönösen átlapoló felosztások.
- Az IS NOT NULL szűrőfeltételt azért alkalmaztuk, mivel a feladat a tényleges főnökök egyes adatait kérte.

**4. lépés (Az előző lépés lekérdezésének felhasználásával listázzuk ki részlegenként az ott dolgozók főnökeinek nevét, fizetését, részlegét és annak helyét)**

```
SELECT DISTINCT
    dolgozó.deptno AS RészlegDeptno,
    dolgozó.mgr AS RészlegFőnök,
    főnök.ename AS FőnökNév,
    főnök.sal AS FőnökFizetés,
    főnök.deptno AS FőnökDeptno,
    dept.loc AS FőnökTelephely
FROM emp dolgozó
INNER JOIN
```

```

emp főnök ON (dolgozó.mgr IS NOT NULL) AND
(dolgozó.mgr = főnök.empno)
INNER JOIN
dept ON (dept.deptno = főnök.deptno);

```

**Eredmény**

RÉSZLEGDEPTNO	RÉSZLEGFŐNÖK	FŐNÖKNÉV	FŐNÖKFIZETÉS	FŐNÖKDEPTNO	FŐNÖKTELEPHEL
10	7782	CLARK	2450	10	NEW YORK
10	7839	KING	5000	10	NEW YORK
20	7566	JONES	2975	20	DALLAS
20	7788	SCOTT	3000	20	DALLAS
20	7839	KING	5000	10	NEW YORK
20	7902	FORD	3000	20	DALLAS
30	7698	BLAKE	2850	30	CHICAGO
30	7839	KING	5000	10	NEW YORK

8 sor kijelölve.

**Megjegyzés**

- A fenti korrelált lekérdezésben igen fontos a deptno minősítőjének használata. Ha nem különböztettük volna meg a dolgozó.deptno és a főnök.deptno kijelölést, hanem csak az egyiket használtuk volna (azt feltételezve, hogy minden főnök ugyanott van, mint a beosztottjai), akkor hibás eredményt kaptunk volna, hiszen láthatóan a King soraiban a két deptno érték különböző.
- Figyeljünk fel arra, hogy az ON kulcsszó után nem csupán tábla-összekapcsoló feltétel állhat, hanem táblaszűrő feltétel is.

### 5. lépés (Kapcsoljuk össze a fenti lekérdezéseket, és listázzunk ismétlésmentesen a részleghelyre és az átlagjövedelmre vonatkozóan)

BREAK ON "Részleghely" ON "átlagjövedelem"

```

SELECT Részleg1.RészlegTelephely AS "Részleghely",
       Részleg1.RészlegÁtlag      AS "átlagjövedelem",
       Főnök1.FőnökNév          AS "Főnök",
       Főnök1.FőnökFizetés       AS "fizetése",
       Főnök1.FőnökTelephely     AS "telephelye"
  FROM (SELECT dept.deptno      AS RészlegDeptno,
              dept.loc        AS RészlegTelephely,
              sub1.ÁtlagJövedelem AS RészlegÁtlag
             FROM dept
           INNER JOIN
             (SELECT emp.deptno,
                    ROUND(AVG(sal+NVL(comm,0))) AS ÁtlagJövedelem
                   
```

```

        FROM emp
        GROUP BY deptno) sub1
            ON (dept.deptno = sub1.deptno)) Részleg1
    INNER JOIN
        (SELECT DISTINCT
            dolgozó.deptno AS RészlegDeptno,
            dolgozó.mgr AS RészlegFőnök,
            főnök.ename AS FőnökNév,
            főnök.sal AS FőnökFizetés,
            főnök.deptno AS FőnökDeptno,
            dept.loc AS FőnökTelephely
        FROM emp dolgozó
        INNER JOIN
            emp főnök ON (dolgozó.mgr IS NOT NULL) AND
                (dolgozó.mgr = főnök.empno)
        INNER JOIN
            dept ON (dept.deptno = főnök.deptno)) Főnök1
        ON (Részleg1.RészlegDeptno = Főnök1.RészlegDeptno)
    ORDER BY "átlagjövedelem";

```

CLEAR BREAKS

### Végeredmény

Részleghely	átlagjövedelem	Főnök	fizetése	telephely
CHICAGO	1933	BLAKE	2850	CHICAGO
		KING	5000	NEW YORK
DALLAS	2175	JONES	2975	DALLAS
		SCOTT	3000	DALLAS
		KING	5000	NEW YORK
NEW YORK	2917	FORD	3000	DALLAS
		CLARK	2450	NEW YORK
		KING	5000	NEW YORK

8 sor kijelölve.

### Megjegyzés

- Érdemes a megoldást újra végiggondolni abból a szempontból, hogy miként cél-szerű egy összetett feladatot részfeladatokra bontani.
- Figyeljünk fel arra, hogy még egy ilyen összetett lekérdezés esetén is elmaradhat a WHERE utasításrész, ha a JOIN tábla-összekapcsolásoknál a kapcsoló és szűrő feltételeket megfelelően használjuk. Az ilyen módon felépített lekérdezés nem csupán áttekinthetőbb (hiszen minden feltétel azon táblánév mellett van, amelyre vonatkozik), hanem lényegesen hatékonyabb is. Ennek az az oka, hogy az egyes allekér-

dezések kisebb eredménylistát adnak vissza, így a teljes eredménylista felépítéséhez kevesebb művelet szükséges.

- Megjegyezzük, hogy a fenti feladat egyszerűbben megoldható, ha az egyes allekérdezéseket úgynevezett nézettáblákba helyezzük (lásd később), mivel ezek önmagukban lekérdezhetők (tesztelhetők), tárolhatók, és a teljes feladatot megoldó lekérdezésben elegendő csupán hivatkozni rájuk, nem kell a bennük szereplő allekérdezéseket megismételni.

### 3.25. feladat

Listázza ki minden dolgozó nevét, foglalkozását, telephelyét, valamint jövedelmük és a részlegük átlagjövedelme közti különbséget, akiknél a munkakörük átlagjövedelme kisebb az összes dolgozó átlagjövedelménél. A listát rendezze telephely szerint.

#### Megoldás

##### 1. lépés (Az összes dolgozó átlagának meghatározása)

```
SELECT AVG(sal+NVL(comm,0)) összesátlag
      FROM emp;
```

#### Eredmény

ÖSSZESÁTLAG
-----
2230.35714

##### 2. lépés (Az átlagjövedelem meghatározása munkakörönként)

```
SELECT job,
       AVG(sal+NVL(comm,0)) munkaköriátlag
      FROM emp
     GROUP BY job
    HAVING AVG(sal+NVL(comm,0)) <           -- munkaköriátlag
          (SELECT AVG(sal+NVL(comm,0))           -- összesátlag
            FROM emp);
```

#### Eredmény

JOB	MUNKAKÖRIÁTLAG
-----	-----
CLERK	1037.5
SALESMAN	1950

##### 3. lépés (A részleg átlagának kiszámítása)

```
SELECT deptno,
       AVG(sal+NVL(comm,0)) AS részlegátlag
      FROM emp
     GROUP BY deptno;
```

**Eredmény**

DEPTNO RÉSZLEGÁTLAG

DEPTNO	RÉSZLEGÁTLAG
10	2916.66667
20	2175
30	1933.33333

**4. lépés (A végső megoldás)**

```

SELECT emp.ename           AS "NÉV",
       emp.job            AS "munkakör",
       dept.loc           AS "telephely",
       részlegáttag -      ROUND(emp.sal+NVL(emp.comm,0)) AS "különbözet"
FROM emp,
     dept,
     (SELECT deptno,
             ROUND(AVG(sal+NVL(comm,0))) ) AS részlegáttag
     FROM emp
     GROUP BY deptno) részleg,
     (SELECT job
        FROM emp
        GROUP BY job
        HAVING AVG(sal+NVL(comm,0)) <          -- munkaköri
               (SELECT AVG(sal+NVL(comm,0)) -- összes átlag
                  FROM emp)) alulfizetett
WHERE emp.job = alulfizetett.job AND
      emp.deptno = részleg.deptno AND
      emp.deptno = dept.deptno
ORDER BY dept.loc;
  
```

**Eredmény**

NÉV	munkakör	telephely	különbözet
JAMES	CLERK	CHICAGO	983
ALLEN	SALESMAN	CHICAGO	33
TURNER	SALESMAN	CHICAGO	433
MARTIN	SALESMAN	CHICAGO	-717
WARD	SALESMAN	CHICAGO	183
SMITH	CLERK	DALLAS	1375
ADAMS	CLERK	DALLAS	1075
MILLER	CLERK	NEW YORK	1617

8 sor kijelölve.

### 3.26. feladat

Listázza ki azoknak a dolgozóknak a nevét, munkakörét, jövedelmét, telephelyét és a munkakörük átlagjövedelmét, akiknek a jövedelme a munkakörük átlagjövedelménél kisebb. A lista a dolgozók neve szerint legyen rendezve.

#### Megoldás

##### 1. lépés (Munkaköri átlagjövedelem számolása)

```
SELECT job,
       ROUND(AVG(sal+NVL(comm,0))) AS átlagjövedelem
    FROM emp
   GROUP BY job;
```

#### Eredmény

JOB	ÁTLAGJÖVEDELEM
ANALYST	3000
CLERK	1038
MANAGER	2758
PRESIDENT	5000
SALESMAN	1950

##### 2. lépés (Kapcsoljuk össze a dept-tel, hogy a telephelyet is kiírhassuk, és rendezzük név szerint)

```
SELECT e.ename          AS név,
       e.job            AS munkakör,
       e.sal+NVL(e.comm,0) AS jövedelem,
       d.loc            AS telephely,
       belso.átlagjövedelem AS "munkaköri átlagjövedelem"
    FROM emp e,dept d,
         (SELECT job, ROUND(AVG(sal+NVL(comm,0))) AS átlagjövedelem
          FROM emp
         GROUP BY job) belso
   WHERE e.sal+NVL(e.comm,0) < belso.átlagjövedelem AND
         e.job = belso.job AND
         e.deptno = d.deptno
  ORDER BY név;
```

#### Eredmény

NÉV	MUNKAKÖR	JÖVEDELEM	TELEPHELY	munkaköri átlagjövedelem
ALLEN	SALESMAN	1900	CHICAGO	1950
CLARK	MANAGER	2450	NEW YORK	2758
JAMES	CLERK	950	CHICAGO	1038

SMITH	CLERK	800 DALLAS	1038
TURNER	SALESMAN	1500 CHICAGO	1950
WARD	SALESMAN	1750 CHICAGO	1950

6 sor kijelölve.

### 3.27. feladat

Listázza ki minden dolgozó nevét, foglalkozását, részlegük nevét, valamint részlegük átlagjövedelme és saját jövedelmük közti különbséget, akiknek a munkaköri átlagjövedelme kisebb az összes dolgozó átlagjövedelménél. A listát a részleg neve szerint rendezze.

#### Megoldás

```

SELECT emp.ename          AS "NÉV",
       emp.job           AS "munkakör",
       dept.dname        AS "részlegnév",
       részlegáttag -
           emp.sal+NVL(emp.comm,0) AS "különbözet"
  FROM emp,
       dept,
       (SELECT deptno, ROUND(AVG(sal+NVL(comm,0))) ) AS részlegáttag
      FROM emp
     GROUP BY deptno) részleg,
       (SELECT job
         FROM emp
        GROUP BY job
       HAVING AVG(sal+NVL(comm,0)) <           -- munkaköri
              (SELECT AVG(sal+NVL(comm,0))           -- összes átlag
                FROM emp)) alulfizetett
 WHERE emp.job = alulfizetett.job AND
       emp.deptno = részleg.deptno AND
       emp.deptno = dept.deptno
 ORDER BY dept.dname;

```

#### Eredmény

NÉV	munkakör	részlegnév	különbözet
MILLER	CLERK	ACCOUNTING	1617
SMITH	CLERK	RESEARCH	1375
ADAMS	CLERK	RESEARCH	1075
JAMES	CLERK	SALES	983
ALLEN	SALESMAN	SALES	633
MARTIN	SALESMAN	SALES	2083

TURNER	SALESMAN	SALES	433
WARD	SALESMAN	SALES	1183

8 sor kijelölve.

### 3.28. feladat

Listázza ki minden dolgozó nevét, munkakörét, telephelyét, fizetését, valamint a saját telephelyén a munkakörének átlagfizetését és telephelyének átlagfizetését. A listát munkakör és telephely szerint növekvő módon rendezze.

#### Megoldás

```

SELECT emp.ename           AS "Név",
       emp.job            AS "munkakör",
       dept.loc           AS "telephely",
       emp.sal            AS "fizetés",
       sLocJob.loc_job_sal AS "TelephelyMunkakörÁtlag",
       sLoc.loc_sal        AS "TelephelyÁtlag"
  FROM emp,
       dept,
       (SELECT deptno,
              job,
              ROUND(AVG(sal)) loc_job_sal
         FROM emp
        GROUP BY deptno, job) sLocJob,
       (SELECT deptno,
              ROUND(AVG(sal)) loc_sal
         FROM emp
        GROUP BY deptno) sLoc
 WHERE emp.deptno = dept.deptno      AND
       emp.deptno = sLocJob.deptno    AND
       emp.deptno = sLoc.deptno      AND
       emp.job = sLocJob.job
 ORDER BY emp.job ASC, dept.loc ASC;

```

#### Eredmény

Név	munkakör	telephely	fizetés	TelephelyMunkakörÁtlag	TelephelyÁtlag
FORD	ANALYST	DALLAS	3000	3000	2175
SCOTT	ANALYST	DALLAS	3000	3000	2175
JAMES	CLERK	CHICAGO	950	950	1567
SMITH	CLERK	DALLAS	800	950	2175
ADAMS	CLERK	DALLAS	1100	950	2175
MILLER	CLERK	NEW YORK	1300	1300	2917

BLAKE	MANAGER	CHICAGO	2850	2850	1567
JONES	MANAGER	DALLAS	2975	2975	2175
CLARK	MANAGER	NEW YORK	2450	2450	2917
KING	PRESIDENT	NEW YORK	5000	5000	2917
ALLEN	SALESMAN	CHICAGO	1600	1400	1567
MARTIN	SALESMAN	CHICAGO	1250	1400	1567
TURNER	SALESMAN	CHICAGO	1500	1400	1567
WARD	SALESMAN	CHICAGO	1250	1400	1567

14 sor kijelölve.

### 3.29. feladat

Listázza ki minden dolgozó nevét, munkakörét, telephelyét, fizetését, valamint fizetését a saját telephelyén a munkaköre átlagfizetésének százalékában, és a fizetését a saját telephelye átlagfizetésének százalékában (a százalékértékeket egészsként kiíratva), a munkakör és a telephely szerint növekvő módon rendezve.

#### Megoldás

```

SELECT emp.ename AS név,
       emp.job AS munkakör,
       dept.loc AS telephely,
       emp.sal AS fizetés,
       ROUND((emp.sal/sLocJob.loc_job_sal)*100) AS relatív_fiz1,
       ROUND((emp.sal/sLoc.loc_sal)*100)         AS relatív_fiz2
  FROM emp,
       dept,
       (SELECT deptno,
              job,
              ROUND(AVG(sal)) loc_job_sal
         FROM emp
        GROUP BY deptno, job) sLocJob,
       (SELECT deptno,
              ROUND(AVG(sal)) loc_sal
         FROM emp
        GROUP BY deptno) sLoc
 WHERE emp.deptno=dept.deptno      AND
       emp.deptno=sLocJob.deptno    AND
       emp.deptno=sLoc.deptno      AND
       emp.job=sLocJob.job
 ORDER BY emp.job ASC,
          dept.loc ASC;

```

**Eredmény**

NÉV	MUNKAKÖR	TELEPHELY	FIZETÉS	RELATÍV_FIZ1	RELATÍV_FIZ2
FORD	ANALYST	DALLAS	3000	100	138
SCOTT	ANALYST	DALLAS	3000	100	138
JAMES	CLERK	CHICAGO	950	100	61
SMITH	CLERK	DALLAS	800	84	37
ADAMS	CLERK	DALLAS	1100	116	51
MILLER	CLERK	NEW YORK	1300	100	45
BLAKE	MANAGER	CHICAGO	2850	100	182
JONES	MANAGER	DALLAS	2975	100	137
CLARK	MANAGER	NEW YORK	2450	100	84
KING	PRESIDENT	NEW YORK	5000	100	171
ALLEN	SALESMAN	CHICAGO	1600	114	102
MARTIN	SALESMAN	CHICAGO	1250	89	80
TURNER	SALESMAN	CHICAGO	1500	107	96
WARD	SALESMAN	CHICAGO	1250	89	80

14 sor kijelölve.

### 3.30. feladat

Listázza ki munkakörönként azokat a dolgozókat és telephelyüket, akiknek fizetése több a munkakörük átlagfizetésénél.

#### Megoldás

##### 1. megoldás (FROM-beli allekérdezéssel – inline nézettel)

```

SELECT ename           AS "Név",
       loc            AS "Telephely",
       sal            AS "Fizetés",
       emp.job        AS "Munkakör",
       munkakör.Átlagfizetés AS "MunkaköriÁtlag"
  FROM emp,
       dept,
       (SELECT job,
               ROUND(AVG(sal)) Átlagfizetés
      FROM emp
      GROUP BY job) munkakör
 WHERE (emp.job = munkakör.job)          AND
       (emp.sal > munkakör.Átlagfizetés) AND
       (emp.deptno = dept.deptno)
 ORDER BY "Munkakör", "Fizetés";

```

**Eredmény**

Név	Telephely	Fizetés	Munkakör	Munkaköri Átlag
ADAMS	DALLAS	1100	CLERK	1038
MILLER	NEW YORK	1300	CLERK	1038
BLAKE	CHICAGO	2850	MANAGER	2758
JONES	DALLAS	2975	MANAGER	2758
TURNER	CHICAGO	1500	SALESMAN	1400
ALLEN	CHICAGO	1600	SALESMAN	1400

6 sor kijelölve.

**2. megoldás (WHERE-beli allekérdezéssel)**

```

SELECT ename AS "Név",
       loc   AS "Telephely",
       sal   AS "Fizetés",
       job   AS "Munkakör"
  FROM emp e,
       dept d
 WHERE sal >
       (SELECT ROUND(AVG(sal))
        FROM emp
        GROUP BY job
        HAVING job = e.job) AND
       e.deptno = d.deptno
 ORDER BY "Munkakör",
          "Fizetés";
    
```

**Eredmény**

Név	Telephely	Fizetés	Munkakör
ADAMS	DALLAS	1100	CLERK
MILLER	NEW YORK	1300	CLERK
BLAKE	CHICAGO	2850	MANAGER
JONES	DALLAS	2975	MANAGER
TURNER	CHICAGO	1500	SALESMAN
ALLEN	CHICAGO	1600	SALESMAN

6 sor kijelölve.

**Megjegyzés**

A 2. megoldásnál nem tudjuk listázni közvetlenül az átlagfizetést, mert a WHERE feltételeben lévő allekérdezés nem nevezhető el, és ezért a külső szelekciós listában nem hivatkozhatunk rá. A megoldást azért közöttük, mert a feladatban nem volt megadva, hogy az átlagfizetést is listázzuk ki (célszerű listázni).

### 3.31. feladat

Listázza ki a legkisebb átlagfizetésű részleg dolgozóit, ennek a részlegnek a nevét, a részleg átlagfizetését és telephelyét.

#### 1. megoldás (FROM-beli allekérdezésekkel)

```

SELECT ename      AS "Név",
       sal        AS "Fizetés",
       dname      AS "Részlegnév",
       minfiz    AS "RészlegMinFizetés"
  FROM emp,
       dept,
       (SELECT deptno,
              ROUND(AVG(sal)) AS atlagfiz
             FROM emp
            GROUP BY deptno)  részleg,
       (SELECT MIN(atlagfiz) minfiz
          FROM
          (SELECT deptno,
                 ROUND(AVG(sal)) AS atlagfiz
                FROM emp
               GROUP BY deptno))  minrészleg
 WHERE
   részleg.atlagfiz = minrészleg.minfiz AND
   dept.deptno = emp.deptno           AND
   emp.deptno = részleg.deptno;

```

#### Eredmény

Név	Fizetés	Részlegnév	RészlegMinFizetés
ALLEN	1600	SALES	1567
BLAKE	2850	SALES	1567
MARTIN	1250	SALES	1567
JAMES	950	SALES	1567
TURNER	1500	SALES	1567
WARD	1250	SALES	1567

6 sor kijelölve.

#### 2. megoldás (Egymásba ágyazott allekérdezésekkel – lépésekkel táblaredukció)

##### 2.1. lépés (Részlegátlag)

```

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

```

**Eredmény**

```
DEPTNO ROUND(AVG(SAL))
-----
```

10	2917
20	2175
30	1567

**2.2. lépés (Legkisebb részlegátlag)**

```
SELECT MIN(rátlag) AS minátlag
FROM
(SELECT deptno AS részleg,
ROUND(AVG(sal)) AS rátlag
FROM emp
GROUP BY deptno);
```

**Eredmény**

```
MINÁTLAG
```

1567
------

**2.3. lépés (Melyik részlegnek az átlaga ez?)**

```
SELECT deptno AS részleg,
ROUND(AVG(sal)) AS rátlag
FROM emp
GROUP BY deptno
HAVING ROUND(AVG(sal)) =
(SELECT MIN(rátlag) AS minátlag
FROM (SELECT deptno AS részleg,
ROUND(AVG(sal)) AS rátlag
FROM emp
GROUP BY deptno));
```

**Eredmény**

```
RÉSZLEG RÁTLAG
```

30	1567
----	------

**2.4. lépés (Külső lekérdezésbe ágyazva az előző lépés eredményét)**

```
SELECT ename AS név,
sal AS fizetés,
dname AS telephely,
részlegátlag.rátlag AS minfiz
FROM emp,
dept,
```

```

        (SELECT deptno          AS részleg,
               ROUND(AVG(sal)) AS rátlag
          FROM emp
         GROUP BY deptno
        HAVING ROUND(AVG(sal)) =
              (SELECT MIN(rátlag)  AS minátlag
                 FROM (SELECT deptno          AS részleg,
                               ROUND(AVG(sal)) AS rátlag
                          FROM emp
                         GROUP BY deptno))) részlegátlag
   WHERE részleg = emp.deptno AND
     emp.deptno = dept.deptno;

```

**Végeredmény**

NÉV	FIZETÉS TELEPHELY	MINFIZ
ALLEN	1600 SALES	1567
BLAKE	2850 SALES	1567
MARTIN	1250 SALES	1567
JAMES	950 SALES	1567
TURNER	1500 SALES	1567
WARD	1250 SALES	1567

6 sor kijelölve.

**3.32. feladat**

Listálja ki a legkisebb létszámú foglalkozási csoport dolgozóinak nevét és foglalkozását. (Természetesen az „elnök” munkakör figyelmen kívül hagyásával.)

**Megoldás (FROM-beli allekérdezésekkel)****1. lépés (A dolgozók száma foglalkozásonként)**

```

SELECT COUNT(ename),
       job
  FROM emp
 WHERE UPPER(job) <> 'PRESIDENT'
 GROUP BY job;

```

**Eredmény**

COUNT(ENAME) JOB
------------------

2 ANALYST
4 CLERK
3 MANAGER
4 SALESMAN

**2. lépés (A legkisebb létszám)**

```
SELECT MIN(Létszám) AS KicsiLétszám
  FROM (SELECT COUNT(ename) AS Létszám
          FROM emp
         WHERE UPPER(job) <> 'PRESIDENT'
        GROUP BY job);
```

**Eredmény**

KICSILÉTSZÁM
2

**3. lépés (Legkisebb létszámú foglalkozási csoport foglalkozásának neve)**

```
SELECT job
  FROM emp
 GROUP BY job
 HAVING COUNT(job) =
    (SELECT MIN(Létszám)
      FROM (SELECT COUNT(ename) AS Létszám
              FROM emp
             WHERE UPPER(job) <> 'PRESIDENT'
            GROUP BY JOB));
```

**Eredmény**

JOB
ANALYST

**4. lépés (Azon dolgozók neve és foglalkozása, akik ebben a csoportban dolgoznak)**

```
SELECT ename,
       emp.job
  FROM emp,
       (SELECT job
          FROM emp
         GROUP BY job
        HAVING COUNT(job) =
           (SELECT MIN(Létszám)
             FROM (SELECT COUNT(ename) AS Létszám
                     FROM emp
                    WHERE UPPER(job) <> 'PRESIDENT'
                   GROUP BY job))) legkisebb_foglalkozás
 WHERE emp.job = legkisebb_foglalkozás.job;
```

**Végeredmény**

ENAME	JOB
SCOTT	ANALYST
FORD	ANALYST

## 4. FEJEZET

# Interaktív környezet

## Feladatok és megoldások

### 4.1. feladat

Mit csinál az alábbi szkript program?

```
set verify off

SELECT * FROM &tabla1;
SELECT * FROM &&tabla2;

ACCEPT tabla3 PROMPT "A tábla neve: "
SELECT * FROM &tabla3;

DEFINE tabla4="salgrade"
SELECT * FROM &tabla4;

DEFINE

ACCEPT abc PROMPT "Táblanevek törlése: (Üsd le az Enter billentyűt...)"
UNDEFINE tabla2
UNDEFINE tabla3
UNDEFINE tabla4
UNDEFINE abc

DEFINE

set verify on
```



#### Megjegyzés

A futási listát sem tesszük bele, hiszen a felhasználó által megadott értékektől függően minden futáskor más és más lesz. Egy biztos, a tabla1 nem került bele az SQL\*Plus

változói közé, viszont ebben a környezetben megőrződött a tabla2, tabla3 és tabla4 változó.

A fenti szkript program részlete egyébként bekérte a tabla1, tabla2, tabla3 változókat, de a tabla4-et nem, mivel az a DEFINE utasítással közvetlenül bekerült a változók közé.

## 4.2. feladat

Listázza ki rendezve azon foglalkozási csoportok dolgozónak nevét, amelyekben a felhasználó által megadott számnál többen tartoznak. A lista fejléce legyen "Név", "Munkakör", "Munkaköri létszám".

### Megoldás

#### 1. lépés (A foglalkozásonkénti létszám lekérdezése)

```
SELECT job, COUNT(ename) AS létszám
  FROM emp
 GROUP BY job
 HAVING COUNT(ename) > &szam;
```



#### Megjegyzés

Ha például a felhasználó 2-t ad meg, akkor három olyan foglalkozási csoport van, amelyiknek a létszáma kettőnél nagyobb.

### Eredmény

```
Adja meg a(z) szam értékét: 2
régi 4: HAVING COUNT(ename) > &szam
új 4: HAVING COUNT(ename) > 2
```

JOB	LÉTSZÁM
CLERK	4
MANAGER	3
SALESMAN	4

Ezután azokat a dolgozókat kell listáznunk, akiknek a munkaköre a fentiek valamelyike. Tehát a most előállított lekérdezéshez elő kell állítani egy, ezt allekérdezésként használó külső lekérdezést. Ezeket a job oszlop fogja összekapcsolni.

#### 2. lépés (A teljes feladat megoldása)

```
ACCEPT szám PROMPT "Minimális létszám: "
```

```
BREAK ON "Munkakör" ON "Munkaköri létszám"
```

```
SELECT dolgozó.ename AS "Név",
      dolgozó.job   AS "Munkakör",
```

```

al.létszám      AS "Munkaköri létszám"
FROM emp dolgozó,
  (SELECT job,
          COUNT(ename) AS létszám
   FROM emp
   GROUP BY job
   HAVING COUNT(ename) > &szám) al
WHERE dolgozó.job = al.job;

```

CLEAR BREAKS

### Eredmény

Minimális létszám: 2

régi 9:	HAVING COUNT(ename) > &szám) al
új 9:	HAVING COUNT(ename) > 2) al

Név	Munkakör	Munkaköri létszám
SMITH	CLERK	4
ADAMS		
MILLER		
JAMES		
JONES	MANAGER	3
CLARK		
BLAKE		
ALLEN	SALESMAN	4
MARTIN		
TURNER		
WARD		

11 sor kijelölve.



### Megjegyzés

- A listát úgy formáztuk a BREAK SQL\*Plus környezeti parancs segítségével, hogy az egymás után következő azonos munkakörököt és létszámokat ne listázza. Ezt *ismétlésmentes listázásnak* nevezzük. Így a lekérdezés áttekinthetőbb. Ennek használatánál ügyelni kell arra, hogy a paramétereként megadott oszlopnév pontosan azonos legyen a megjelenítendő oszlopnévvel!
- Már korábban utaltunk rá, hogy az SQL\*Plus-környezetbe ne közvetlenül írjuk be az utasításokat, hanem szkript programon keresztül. Ennek legfőbb oka az, hogy így az SQL\*Plus-környezet saját editorával tudunk szerkeszteni, tesztelni, próbálkozni. Ráadásul így több utasításból álló, akár SQL\*Plus környezeti utasításokat is tartalmazó utasítássorozat is könnyen javítható, próbálható.

### 4.3. feladat

Listázza ki ismétlésmentesen azon részlegek telephelyét, a részlegek tagjainak nevét és jövedelmüket, ahol az átlagjövedelem meghaladja a felhasználó által megadott értéket. A lista fejléce: Telephely, Dolgozónév, jövedelem.

#### Megoldás

##### 1. lépés (A részlegenkénti átlag)

```
SELECT deptno,
       ROUND(AVG(sal+NVL(comm,0)))
  FROM emp
 GROUP BY deptno;
```

#### Eredmény

	DEPTNO	ROUND(AVG(SAL+NVL(COMM,0)))
	10	2917
	20	2175
	30	1933

##### 2. lépés (Ezen részlegek tagjainak nevét szükséges a különböző SELECT-ben azonosítani, és kiírni az ehhez a részleghez tartozó telephelyet)

```
ACCEPT szám PROMPT "Részleg átlagjövedelem minimuma: "
BREAK ON "Telephely" ON "RészlegÁtlagMin"
SELECT loc           AS "Telephely",
       &szám        AS "RészlegÁtlagMin",
       ename         AS "Dolgozónév",
       sal+NVL(comm,0) AS "jövedelem"
  FROM emp,
       dept,
       (SELECT deptno,
               ROUND(AVG(sal+NVL(comm,0)))
          FROM emp
         GROUP BY deptno
        HAVING AVG(sal+NVL(comm,0)) > &szám) belso
 WHERE emp.deptno = belso.deptno AND
       emp.deptno = dept.deptno;
CLEAR BREAKS
```

#### Eredmény

Részleg átlagjövedelem minimuma: 2000		
régi 2:	&szám	AS "RészlegÁtlagMin",
új 2:	2000	AS "RészlegÁtlagMin",
régi 11:	HAVING AVG(sal+NVL(comm,0)) > &szám) belso	

új 11: HAVING AVG(sal+NVL(comm,0)) > 2000) belső

Telephely	Részleg	Átlag	Min	Dolgozó	Név	jövedelem
NEW YORK		2000	CLARK			2450
			KING			5000
			MILLER			1300
DALLAS		2000	SMITH			800
			ADAMS			1100
			FORD			3000
			SCOTT			3000
			JONES			2975

8 sor kijelölve.

#### 4.4. feladat

Listázza ki ismétlésmentesen a felhasználó által megadott jövedelmi tartományba eső dolgozókat. Legyen a lista fejléce: TELEPHELY, DOLGOZÓ, MUNKAKÖR, JÖVEDELEM. A lista a telephely és a dolgozók neve szerint növekvően legyen rendezett.

#### Megoldás

```

BREAK ON TELEPHELY
SELECT loc          AS TELEPHELY,
       ename         AS DOLGOZÓ,
       job           AS MUNKAKÖR,
       sal+NVL(comm,0) AS JÖVEDELEM
  FROM emp,
       dept
 WHERE emp.deptno = dept.deptno AND
       sal+NVL(comm,0) BETWEEN &minjöv AND &maxjöv
 ORDER BY loc,
          ename;
CLEAR BREAKS

```

#### Eredmény

Adja meg a(z) minjöv értékét: 1000

Adja meg a(z) maxjöv értékét: 2500

régi 8: sal+NVL(comm,0) BETWEEN &minjöv AND &maxjöv

új 8: sal+NVL(comm,0) BETWEEN 1000 AND 2500

TELEPHELY	DOLGOZÓ	MUNKAKÖR	JÖVEDELEM
CHICAGO	ALLEN	SALESMAN	1900

	TURNER	SALESMAN	1500
	WARD	SALESMAN	1750
DALLAS	ADAMS	CLERK	1100
NEW YORK	CLARK	MANAGER	2450
	MILLER	CLERK	1300

6 sor kijelölve.

#### 4.5. feladat

Listázza ki ismétlésmentesen a felhasználó által megadott azonosítójú főnök (mgr) beosztja-jit. Legyen a lista fejléce TELEPHELY, DOLGOZÓ, JÖVEDELEM, FŐNÖK\_KÓD. A lista a telephely és a dolgozók neve szerint növekvően legyen rendezett.

#### Megoldás

```

BREAK ON TELEPHELY
SELECT loc          AS TELEPHELY,
       ename        AS DOLGOZÓ,
       sal+NVL(comm,0) AS JÖVEDELEM,
       mgr         AS FŐNÖK_KÓD
  FROM emp,
       dept
 WHERE emp.deptno = dept.deptno AND
       mgr = &főnökkód
 ORDER BY loc,
          ename;
CLEAR BREAKS

```

#### Eredmény

Adja meg a(z) főnökkód értékét: 7698

régi 8: mgr = &főnökkód

új 8: mgr = 7698

TELEPHELY	DOLGOZÓ	JÖVEDELEM	FŐNÖK_KÓD
CHICAGO	ALLEN	1900	7698
	JAMES	950	7698
	MARTIN	2650	7698
	TURNER	1500	7698
	WARD	1750	7698

## 4.6. feladat

Listázza ki a felhasználó által megadott időszakban belépett dolgozókat foglalkoztató részlegeket. Legyen a lista fejléce RÉSZLEGNÉV, RÉSZLEGASONOSÍTÓ, KEZDŐDÁTUM, VÉGDÁTUM. A lista a részlegnév szerint növekvő sorba legyen rendezett.

### Megoldás

#### 1. lépés (A megadott időszakban belépett dolgozók)

##### 1A. megoldás

```
SELECT ename, deptno, hiredate
  FROM emp
 WHERE hiredate BETWEEN '&kezd' AND '&vég';
```

##### Eredmény

Adja meg a(z) kezd értékét: 81-JAN-10

Adja meg a(z) vég értékét: 82-JAN-10

régi 3: WHERE hiredate BETWEEN '&kezd' AND '&vég'

új 3: WHERE hiredate BETWEEN '81-JAN-10' AND '82-JAN-10'

nincsenek kijelölve sorok



##### Megjegyzés

Dátumforma nélküli lekérdezésekben ne használunk hiányos dátumformát (amelyik az évszázadot nem tartalmazza), mivel az nem használható az előző évszázadi dátumokra, ha a belső dátumforma az évszázadot is tartalmazza. (Lásd 1. fejezet.)

##### 1B. megoldás

```
SELECT ename, deptno, hiredate
  FROM emp
 WHERE hiredate BETWEEN TO_DATE('&kezd', 'YYYY.MM.DD') AND
                      TO_DATE('&vég', 'YYYY.MM.DD');
```

##### Eredmény

Adja meg a(z) kezd értékét: 1981.01.10

régi 3: WHERE hiredate BETWEEN TO\_DATE('&kezd', 'YYYY.MM.DD') AND

új 3: WHERE hiredate BETWEEN TO\_DATE('1981.01.10', 'YYYY.MM.DD') AND

Adja meg a(z) vég értékét: 1982.01.10

régi 4: TO\_DATE('&vég', 'YYYY.MM.DD')

új 4: TO\_DATE('1982.01.10', 'YYYY.MM.DD')

ENAME	DEPTNO	HIREDATE
ALLEN	30	81-FEB-20

WARD	30 81-FEB-22
JONES	20 81-ÁPR-02
MARTIN	30 81-SZE-28
BLAKE	30 81-MÁJ-01
CLARK	10 81-JÚN-09
KING	10 81-NOV-17
TURNER	30 81-SZE-08
JAMES	30 81-DEC-03
FORD	20 81-DEC-03

10 sor kijelölve.



### Megjegyzés

Ez a módszer már megfelelő eredményt ad, mert a belső dátumformától független a dátumbevitel módja, ám nehézkes, mivel a felhasználó számára semmilyen utalást nem ad a megfelelő adatbevitelhez (azaz ismerni kellene a szkript program által igényelt dátumformát).

### 1C. megoldás

```
ACCEPT kezd PROMPT 'Kezdő dátum (YYYY.MM.DD) : '
```

```
ACCEPT vég PROMPT ' Vég dátum (YYYY.MM.DD) : '
```

```
SELECT ename, deptno, hiredate
  FROM emp
 WHERE hiredate BETWEEN TO_DATE('&kezd','YYYY.MM.DD') AND
                      TO_DATE('&vég','YYYY.MM.DD');
```

```
UNDEFINE kezd
```

```
UNDEFINE vég
```

### Eredmény

Kezdő dátum (YYYY.MM.DD) : 1981.01.10

Vég dátum (YYYY.MM.DD) : 1982.01.10

régi 3: WHERE hiredate BETWEEN TO\_DATE('&kezd','YYYY.MM.DD') AND

új 3: WHERE hiredate BETWEEN TO\_DATE('1981.01.10','YYYY.MM.DD') AND

régi 4: TO\_DATE('&vég','YYYY.MM.DD')

új 4: TO\_DATE('1982.01.10','YYYY.MM.DD')

ENAME	DEPTNO	HIREDATE
ALLEN	30	81-FEB-20
WARD	30	81-FEB-22
JONES	20	81-ÁPR-02
MARTIN	30	81-SZE-28

BLAKE	30 81-MÁJ-01
CLARK	10 81-JÚN-09
KING	10 81-NOV-17
TURNER	30 81-SZE-08
JAMES	30 81-DEC-03
FORD	20 81-DEC-03

10 sor kijelölve.



### Megjegyzés

Ez már nem csupán jó megoldás, de felhasználóbarát is, mivel az ACCEPT SQL\*Plus-utasítás révén lehetőséget ad a felhasználó tájékoztatására a megfelelő dátumformátumról. Egy hiányossága azért ennek a módszernek is van, ugyanis azzal, hogy a dátumot karakteres változóba tölti (ha az ACCEPT utasítás változója után nem áll típusmegjelölés, akkor ez az alapértelmezés), fennáll a veszélye annak, hogy a felhasználó mégiscsak elhibázza a dátumformát, és erre semmi sem figyelmezteti (ha csak azt nem tekintjük annak, hogy üres lista tűnt kap eredményként...). Ne felejtkezzünk el a szkript program végén a munkaváltozók törléséről (UNDEFINE).

### 1D. megoldás

```

ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY.MM.DD';
ACCEPT kezd DATE PROMPT "Kezdő dátum (YYYY.MM.DD) : "
ACCEPT vég DATE PROMPT " Vég dátum (YYYY.MM.DD) : "

SELECT TO_DATE('&kezd') FROM dual;
SELECT TO_DATE('&vég') FROM dual;

SELECT ename, deptno, hiredate
  FROM emp
 WHERE hiredate BETWEEN TO_DATE('&kezd','YYYY.MM.DD') AND
                      TO_DATE('&vég','YYYY.MM.DD');

ALTER SESSION SET NLS_DATE_FORMAT = 'YY-MON-DD';
UNDEFINE kezd
UNDEFINE vég

```

### Eredmény

A munkamenet módosítva.

```

Kezdő dátum (YYYY.MM.DD) : 1981.01.10
Vég dátum (YYYY.MM.DD) : 1982.01.10
régi   1: SELECT TO_DATE('&kezd') FROM dual
új    1: SELECT TO_DATE('1981.01.10') FROM dual

```

```
TO_DATE('1
-----
```

```
1981.01.10
```

```
régi 1: SELECT TO_DATE('&vég') FROM dual
új 1: SELECT TO_DATE('1982.01.10') FROM dual
```

```
TO_DATE('1
-----
```

```
1982.01.10
```

```
régi 3: WHERE hiredate BETWEEN TO_DATE('&kezd','YYYY.MM.DD') AND
új 3: WHERE hiredate BETWEEN TO_DATE('1981.01.10','YYYY.MM.DD') AND
régi 4:                      TO_DATE('&vég','YYYY.MM.DD')
új 4:                      TO_DATE('1982.01.10','YYYY.MM.DD')
```

ENAME	DEPTNO	HIREDATE
KING	10	1981.11.17
BLAKE	30	1981.05.01
CLARK	10	1981.06.09
JONES	20	1981.04.02
MARTIN	30	1981.09.28
ALLEN	30	1981.02.20
TURNER	30	1981.09.08
JAMES	30	1981.12.03
WARD	30	1981.02.22
FORD	20	1981.12.03

10 sor kijelölve.

A munkamenet módosítva.



### Megjegyzés

Ez a legmegfelelőbb megoldás, mivel a nem dátum jellegű adat bevitele ellen közvetlenül véd azáltal, hogy a dátumot DATE típusú adatként kéri be az ACCEPT SQL\*Plus utasítás. (Például nem engedi dátumként bevinni az 19800110 karakterszorozatot.) Automatikusan dátumkonverziót is végez, de ennek használatával vigyázni kell, mert az évszázadot nem tartalmazó dátumbevitel esetén az aktuális évszázaddal egészít ki. (Például 81-01-10 esetén automatikusan a 2081.01.10 alakra konvertál, ám ha az 1981-01-10 vagy az 1981.jan.10 alakot írjuk be, akkor már a helyes 1981.01.10 alakra konvertál.)

Természetesen ekkor gondoskodni kell a megfelelő dátumforma beállításáról (ALTER SESSION SET NLS...) és a visszaállításról, valamint a munkaváltozók törléséről (UNDEFINE).

**2. lépés (A megfelelő részlegek listázása [a feladat teljes megoldása szkript programban])**

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY.MM.DD';
ACCEPT kezd DATE PROMPT "Kezdő dátum (YYYY.MM.DD) : "
ACCEPT vég DATE PROMPT " Vég dátum (YYYY.MM.DD) : "
```

```
SELECT dname      AS részlegnév,
       dept.deptno AS részlegazonosító,
       '&kezd'     AS kezdődátum,
       '&vég'       AS végdátum
  FROM dept,
       (SELECT deptno
        FROM emp
       WHERE hiredate BETWEEN
              TO_DATE('&kezd', 'YYYY.MM.DD') AND
              TO_DATE('&vég', 'YYYY.MM.DD')
       GROUP BY deptno) .tartomány
 WHERE dept.deptno = tartomány.deptno;

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

**Eredmény**

A munkamenet módosítva.

```
Kezdő dátum (YYYY.MM.DD) : 1981.01.10
Vég dátum (YYYY.MM.DD) : 1982.01.10
régi 3:      '&kezd'     AS kezdő_dátum,
új   3:      '1981.01.10'   AS kezdő_dátum,
régi 4:      '&vég'      AS vég_dátum
új   4:      '1982.01.10'   AS vég_dátum
régi  9:           TO_DATE('&kezd', 'YYYY.MM.DD') AND
új   9:           TO_DATE('1981.01.10', 'YYYY.MM.DD') AND
régi 10:          TO_DATE('&vég', 'YYYY.MM.DD')
új  10:          TO_DATE('1982.01.10', 'YYYY.MM.DD')
```

RÉSZLEGNÉV	RÉSZLEGAZONOSÍTÓ	KEZDŐDÁTUM	VÉGDÁTUM
ACCOUNTING	10	1981.01.10	1982.01.10
RESEARCH	20	1981.01.10	1982.01.10
SALES	30	1981.01.10	1982.01.10

A munkamenet módosítva.

## 4.7. feladat

Listázza ki ismétlésmentesen a felhasználó által megadott betűt tartalmazó telephelyen dolgozókat. Legyen a lista fejléce: TELEPHELY, DOLGOZÓ, MUNKAKÖR, JÖVEDELEM. A lista a telephely és a dolgozók neve szerint növekvően legyen rendezett.

### 1. megoldás

```

BREAK ON Telephely
SELECT loc          AS Telephely,
       ename         AS Dolgozó,
       job           AS Munkakör,
       sal+NVL(comm,0) AS Jövedelem
  FROM emp,
       dept
 WHERE UPPER(loc) LIKE UPPER('%&betű%') AND
       emp.deptno = dept.deptno
 ORDER BY Telephely, Dolgozó;
CLEAR BREAKS

```

### Eredmény

Adja meg a(z) betű értékét: a  
régi 7: WHERE UPPER(loc) LIKE UPPER('%&betű%') AND  
új 7: WHERE UPPER(loc) LIKE UPPER('%a%') AND

TELEPHELY	DOLGOZÓ	MUNKAKÖR	JÖVEDELEM
CHICAGO	ALLEN	SALESMAN	1900
	BLAKE	MANAGER	2850
	JAMES	CLERK	950
	MARTIN	SALESMAN	2650
	TURNER	SALESMAN	1500
	WARD	SALESMAN	1750
DALLAS	ADAMS	CLERK	1100
	FORD	ANALYST	3000
	JONES	MANAGER	2975
	SCOTT	ANALYST	3000
	SMITH	CLERK	800

11 sor kijelölve.

### 2. megoldás

```

BREAK ON Telephely
SELECT betűtábla.loc      AS Telephely,
       ename        AS Dolgozó,

```

```

job          AS Munkakör,
sal+NVL(comm,0) AS Jövedelem
FROM emp,
dept,
(SELECT loc
  FROM dept
 WHERE UPPER(loc) LIKE UPPER('%&betű%')) betűtábla
WHERE emp.deptno = dept.deptno AND
betűtábla.loc = dept.loc
ORDER BY Telephely, Dolgozó;
CLEAR BREAKS

```

**Eredmény**

Adja meg a(z) betű értékét: a  
régi 9: WHERE UPPER(loc) LIKE UPPER('%&betű%')) betűtábla  
új 9: WHERE UPPER(loc) LIKE UPPER('%a%')) betűtábla

TELEPHELY	DOLGOZÓ	MUNKAKÖR	JÖVEDELEM
CHICAGO	ALLEN	SALESMAN	1900
	BLAKE	MANAGER	2850
	JAMES	CLERK	950
	MARTIN	SALESMAN	2650
	TURNER	SALESMAN	1500
	WARD	SALESMAN	1750
DALLAS	ADAMS	CLERK	1100
	FORD	ANALYST	3000
	JONES	MANAGER	2975
	SCOTT	ANALYST	3000
	SMITH	CLERK	800

11 sor kijelölve.

**4.8. feladat**

Listázza ki formázottan az emp táblából a felhasználó által megadott foglalkozású dolgozókat.

**Megoldás**

```

SET verify OFF
SET pagesize 15
SET linesize 50
SET feedback OFF
TTITLE 'Dolgozók|listázása'
BTITLE 'Budapesti Műszaki Főiskola'

```

```
COLUMN ename HEADING 'A dolgozó|neve'      FORMAT A15
COLUMN job   HEADING 'A dolgozó|munkaköre'  FORMAT A15
COLUMN dname HEADING 'A dolgozó|részlege'   FORMAT A15
```

ACCEPT munka PROMPT 'Kérem írja be munkakört: '

```
SELECT dolgozó.ename,
       dolgozó.job,
       részleg.dname
  FROM emp dolgozó,
       dept részleg
 WHERE UPPER(dolgozó.job) = UPPER('&munka') AND
       dolgozó.deptno = részleg.deptno
 ORDER BY dolgozó.ename;
```

```
SET verify ON
SET pagesize 40
SET linesize 400
SET feedback ON
TTITLE OFF
BTITLE OFF
CLEAR BREAKS
COLUMN ename CLEAR
COLUMN dname CLEAR
UNDEFINE oszlop
```

### Eredmény

Kérem írja be munkakört: clerk

V. Okt 24

lap 1

#### Dolgozók listázása

A dolgozó neve	A dolgozó munkaköre	A dolgozó részlege
ADAMS	CLERK	RESEARCH
JAMES	CLERK	SALES
MILLER	CLERK	ACCOUNTING
SMITH	CLERK	RESEARCH

## 4.9. feladat

Listázza ki formázottan az emp táblából a felhasználó által megadott foglalkozású dolgozók főnökeinek nevét.

### Megoldás

```

SET verify OFF
SET pagesize 15
SET linesize 50
SET feedback OFF
TTITLE 'Dolgozók főnökeinek listázása'
BTITLE 'Budapesti Műszaki Főiskola'
COLUMN ename HEADING 'A dolgozó|neve'           FORMAT A15
COLUMN dname HEADING 'A dolgozó|részlege'        FORMAT A15

ACCEPT munka PROMPT 'Kérem írja be munkakört: '

SELECT dolgozo.ename || ' főnöke ' || főnök.ename as "Dolgozó      Főnö-
ke"
FROM emp főnök,
      emp dolgozo
WHERE főnök.empno=dolgozo.mgr AND
      dolgozo.job=UPPER('&munka');

SET verify ON
SET pagesize 40
SET linesize 400
TTITLE OFF
BTITLE OFF
CLEAR BREAKS
COLUMN ename CLEAR
COLUMN dname CLEAR

```

### Eredmény

Kérem írja be munkakört: clerk

V. Okt 24

lap 1

Dolgozók főnökeinek listázása

Dolgozó	Főnöke
SMITH	FORD
ADAMS	SCOTT

SMITH	FORD
ADAMS	SCOTT

JAMES főnöke BLAKE  
 MILLER főnöke CLARK

Budapesti Műszaki Főiskola

#### 4.10. feladat

Listázza ki formázottan a felhasználó által megadott oszlop szerinti, valamint a felhasználó által megadott belépési dátum intervallumba eső dolgozók átlagfizetését.

##### Megoldás

```

SET verify OFF
SET pagesize 15
SET linesize 65
SET feedback OFF
TTITLE 'A megadott időben dolgozók bekért oszlop szerinti átlaga'
BTITLE 'Budapesti Műszaki Főiskola'

ACCEPT oszlop PROMPT 'Kérlem írja be a kívánt oszlop nevét: '
ACCEPT dalsó PROMPT 'Kérlem írja be a belépés dátumának alsó határát: '
ACCEPT dfelső PROMPT 'Kérlem írja be a belépés dátumának felső határát: '

SELECT empno          AS azonosító,
      ename           AS név,
      hiredate        AS belépés,
      &oszlop         AS felhasználói_oszlop,
      átlagfizetés
     FROM emp,
          (SELECT ROUND(AVG(sal)) AS átlagfizetés,
          &oszlop AS Kérdéses
          FROM emp
          WHERE hiredate BETWEEN TO_DATE('&dalsó','YYYY.MM.DD.') AND
                TO_DATE('&dfelső','YYYY.MM.DD.')
          GROUP BY &oszlop) átlag
     WHERE &oszlop IN (átlag.kérdéses) AND
           hiredate BETWEEN TO_DATE('&dalsó','YYYY.MM.DD.') AND
                         TO_DATE('&dfelső','YYYY.MM.DD.')
     ORDER BY ename;
```

```

SET verify ON
SET pagesize 40
SET linesize 400
TTITLE OFF
BTITLE OFF
UNDEFINE oszlop
UNDEFINE dalsó
UNDEFINE dfelső

```

**Eredmény**

V. Okt 17. lap 1

A megadott időben dolgozók bekért-oszlop szerinti átlaga

AZONOSÍTÓ NÉV	BELÉPÉS	KÉRDÉSES_OSZLOP	ÁTLAGFIZETÉS
7698 BLAKE	81-MÁJ-01	30	1867
7782 CLARK	81-JÚN-09	10	2450
7566 JONES	81-ÁPR-02	20	2975
7654 MARTIN	81-SZE-28	30	1867
7844 TURNER	81-SZE-08	30	1867

Budapesti Műszaki Főiskola ORACLE oktatói

**4.11. feladat**

Írjon szkript programot, amely formázottan listázza ki az emp és dept táblából a felhasználó által megadott munkakörű dolgozók telephelyét, részlegük legkisebb jövedelme és a saját jövedelme közti különbséget, valamint főnökének nevét. A lista fejléce legyen: dolgozó neve, telephelye, munkaköre, jövedelme, legkisebb jövedelem, különbség, főnök neve.

**1. megoldás**

```

SET numwidth 5
SET linesize 65
SET pagesize 15
SET verify OFF

```

ACCEPT munkakör PROMPT "A listázandó dolgozók munkaköre: "

```

TTITLE 'A feladat megoldása'
BTITLE 'Budapesti Műszaki Főiskola'
COLUMN "dolgozó neve"      HEADING "dolgozó| neve"  FORMAT A7

```

```

COLUMN "telephelye"          FORMAT A10
COLUMN "munkaköre"           HEADING " munka-| kör"   FORMAT A7
COLUMN "legkisebb jövedelem"  HEADING "legkisebb|jövedelem"
COLUMN "főnök neve"          HEADING "főnök|neve"    FORMAT A7
BREAK ON "munkaköre"

SELECT
  emp.ename                  AS "dolgozó neve",
  Részleg.loc                 AS "telephelye",
  emp.job                     AS "munkaköre",
  emp.sal+NVL(emp.comm,0)     AS "jövedelme",
  ROUND(RészlegJövedelem.MinKereset,0)
                                AS "legkisebb jövedelem",
  ((emp.sal+NVL(emp.comm,0)) - ROUND(RészlegJövedelem.MinKereset,0))
                                AS "különbség",
  Főnök.ename                 AS "főnök neve"
FROM
  emp
  INNER JOIN
    (SELECT deptno,
            loc
     FROM dept)  Részleg
  ON emp.deptno = Részleg.deptno
  INNER JOIN
    (SELECT deptno,
            (MIN(sal) + MIN(NVL(comm,0))) AS MinKereset
     FROM emp
     GROUP BY deptno)  RészlegJövedelem
  ON emp.deptno = RészlegJövedelem.deptno
  INNER JOIN
    (SELECT ename,
            empno
     FROM emp)  Főnök
  ON Főnök.empno = emp.mgr
WHERE
  UPPER(emp.job) = UPPER('&munkakör')
ORDER BY emp.ename;

CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF
BTITLE OFF
SET verify ON
SET pagesize 40

```

```
SET linesize 400
UNDEFINE munkakör
```

## 2. megoldás

```
SET numwidth 5
SET linesize 65
SET pagesize 15
SET verify OFF
```

```
ACCEPT munkakör PROMPT "A listázandó dolgozók munkaköre: "
```

```
TTITLE 'A feladat megoldása'
BTITLE 'Budapesti Műszaki Főiskola'
COLUMN dolg FORMAT A7 HEADING 'dolgozó|neve' JUSTIFY CENTER
COLUMN loc FORMAT A10 HEADING 'telephelye' JUSTIFY CENTER
COLUMN job FORMAT A7 HEADING 'munka-| kör' JUSTIFY LEFT
COLUMN jov FORMAT 99999999 HEADING 'jövedelme' JUSTIFY CENTER
COLUMN kisjov FORMAT 99999999 HEADING 'legkisebb|jövedelem' -
JUSTIFY CENTER
COLUMN kul FORMAT 99999999 HEADING 'különbség' JUSTIFY CENTER
COLUMN fonok FORMAT A7 HEADING 'főnök|neve' JUSTIFY LEFT
BREAK ON job
```

```
SELECT e.ename AS dolg,
       d.loc,
       e.job,
       e.sal+NVL(e.comm,0) AS jov,
       al.kisjov AS kisjov,
       (e.sal+NVL(e.comm,0))-al.kisjov AS kul,
       f.ename AS fonok
  FROM dept d,
       emp e,
       emp f,
       (SELECT emp.deptno,
                  MIN(emp.sal)+MIN(NVL(emp.comm,0)) AS kisjov
             FROM emp,
                  dept
            WHERE emp.deptno = dept.deptno
            GROUP BY emp.deptno) al
 WHERE (UPPER(e.job) = UPPER('&munkakör')) AND
       (d.deptno = e.deptno) AND
       (e.mgr = f.empno) AND
       (al.deptno = e.deptno)
 ORDER BY dolg;
```

```
CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF
BTITLE OFF
SET pagesize 40
SET linesize 400
SET verify ON
UNDEFINE munkakör
```

**Eredmény (mindkét esetben)**

A listázandó dolgozók munkaköre: manager

H. Nov 29

lap 1

**A feladat megoldása**

dolgozó neve	telephelye	munka- kör	legkisebb jövedelme	jövedelem	különbség	főnök neve
BLAKE	CHICAGO	MANAGER	2850	950	1900	KING
CLARK	NEW YORK		2450	1300	1150	KING
JONES	DALLAS		2975	800	2175	KING

Budapesti Műszaki Főiskola

**4.12. feladat**

Írjon szkript programot, amely formázottan listázza ki az emp és dept táblából a felhasználó által megadott nevű főnök beosztottainak számát, telephelyét, átlagfizetését, a főnök és beosztottak átlagjövedelme közötti különbséget. A lista fejléce legyen: Főnök neve, Telephelye, Fizetése, Beosztottainak száma, átlagjövedelmük, jövedelem különbség.

**Megoldás**

```
SET feedback OFF
SET linesize 70
SET pagesize 12
TTITLE 'A 4.12-es interaktív feladat megoldása'
BTITLE 'Budapesti Műszaki Főiskola'
COLUMN atl FORMAT 99999 HEADING "átlagjö-|vedelmük"
COLUMN "Jk" FORMAT 99999999 HEADING "jövedelem|különbség"
COLUMN szam FORMAT 999999999 HEADING "Beosztottainak| száma"
```

```

SELECT f.ename          AS "Főnök neve",
       d.loc            AS "Telephelye",
       f.sal             AS "Fizetése",
       dolg.atl,
       (f.sal-NVL(f.comm,0))-dolg.atl   AS "Jk",
       dolg.szam
  FROM emp f,
       dept d,
       (SELECT mgr,
              ROUND(AVG(sal+NVL(comm,0))) AS atl,
              COUNT(*)                  AS szam
         FROM emp
        GROUP BY mgr)  dolg
 WHERE dolg.mgr        = f.empno      AND
       f.deptno        = d.deptno      AND
       UPPER(f.ename)  = UPPER('&név');

COLUMN atl CLEAR
COLUMN "JK" CLEAR
COLUMN szam CLEAR
TTITLE OFF
BTITLE OFF
SET pagesize 40
SET linesize 400
SET verify ON
SET feedback ON

```

**Eredmény**

Adja meg a(z) név értékét: Blake  
régi 16: UPPER(f.ename) = UPPER('&név')  
új 16: UPPER(f.ename) = UPPER('Blake')

Sze Nov 10

lap 1

A 4.12-es interaktív feladat megoldása

Főnök neve	Telephelye	átlagjö- jövedelem	Beosztottainak	
		Fizetése	vedelmük különbség	száma
BLAKE	CHICAGO	2850	1750	1100

### 4.13. feladat

Írjon szkript programot, amely formázottan (fej- és lábléccel stb.), valamint ismétlésmentesen listázza ki a dolgozók nevét, fizetését, részlegük telephelyét, annak összfizetését, valamint az abban dolgozó főnökök (mgr) nevét és fizetését a főnök neve és a dolgozók neve szerint rendezve.

#### Megoldás

```

SET verify OFF
SET feedback OFF
SET linesize 70
SET pagesize 26
TTITLE "A feladat megoldása|formázott és ismétlésmentes kiíratással"
BTITLE "Készítették:[a Budapesti Műszaki Főiskola|adatbázis oktatói"

COLUMN DNeve      HEADING 'Dolgozó|neve'
COLUMN Dfiz        HEADING 'Dolgozó|fizetése' JUSTIFY RIGHT
COLUMN Rosszes    HEADING 'Részleg|összfizetése' JUSTIFY RIGHT
COLUMN loc         HEADING 'Részleg|helye'                   FORMAT A10
COLUMN FNeve       HEADING 'Főnök|neve'                  FORMAT A7
COLUMN FFiz        HEADING 'Főnök|fizetése' JUSTIFY RIGHT FORMAT 9999
BREAK ON rosszes ON loc ON FNeve ON FFiz

SELECT DNeve, Dfiz, rosszes, adatok.loc, FNeve, FFiz
  FROM
    (SELECT dolg.ename   AS DNeve,
            dolg.sal     AS Dfiz,
            főnök.ename  AS FNeve,
            főnök.sal    AS FFiz,
            loc
       FROM emp dolg,
             emp főnök,
             dept d
      WHERE dolg.mgr = főnök.empno AND
            főnök.deptno = d.deptno) adatok,
    (SELECT SUM(sal)  AS Rosszes,
            loc
       FROM emp,
             dept
      WHERE emp.deptno = dept.deptno
      GROUP BY loc ) resz1
    WHERE resz1.loc = adatok.loc
  ORDER BY FNeve, DNeve;

```

```
CLEAR BREAKS
CLEAR COLUMNS
TTITLE OFF
BTITLE OFF
SET verify ON
SET feedback ON
```

**Eredmény**

Sze Nov 10

lap 1

A feladat megoldása  
formázott és ismétlésmentes kiíratással

Dolgozó neve	Dolgozó fizetése	Részleg összfizetése	Részleg helye	Főnök neve	Főnök fizetése
ALLEN	1600		9400 CHICAGO	BLAKE	2850
JAMES	950				
MARTIN	1250				
TURNER	1500				
WARD	1250				
MILLER	1300		8750 NEW YORK	CLARK	2450
SMITH	800		10875 DALLAS	FORD	3000
FORD	3000			JONES	2975
SCOTT	3000				
BLAKE	2850		8750 NEW YORK	KING	5000
CLARK	2450				
JONES	2975				
ADAMS	1100		10875 DALLAS	SCOTT	3000

Készítették:  
a Budapesti Műszaki Főiskola  
adatbázis oktatói

## 5. FEJEZET

# Adattáblák létrehozása, módosítása, tranzakciók, megszorítások

## Feladatok és megoldások

Az alábbi feladatok egymásra épülnek, ezért célszerű az adott sorrendben megoldani őket, különben nem minden részletet lehet elérni.

### 5.1. feladat

Hozzon létre az emp táblából egy dolgozó nevű táblát, és nevezze át annak empno oszlopát azonosító névre. Kérdezze le az új tábla szerkezetét.

#### Megoldás

```
DROP TABLE dolgozó;

CREATE TABLE dolgozó
AS
SELECT * FROM emp;

ALTER TABLE dolgozó
RENAME COLUMN empno TO azonosító;

SET linesize 60
DESC dolgozó
SET linesize 400;
```

#### Eredmény

- A tábla eldobva.
- A tábla létrejött.
- A tábla módosítva.

Név	Üres?	Típus
AZONOSÍTÓ		NOT NULL NUMBER(4)

ENAME	VARCHAR2(10)
JOB	VARCHAR2(9)
MGR	NUMBER(4)
HIREDATE	DATE
SAL	NUMBER(7,2)
COMM	NUMBER(7,2)
DEPTNO	NOT NULL NUMBER(2)

## 5.2. feladat

Hozzon létre egy dolgozó nevű táblát az emp tábla azon dolgozóiból, akiknek 500 USD-nál több a jövedelmük, és 1980. január 1-je után léptek be. (Megjegyezzük, hogy az 5.2–5.8. feladatok összefüggő feladatsort alkotnak, ezért értelmezésük, megoldásuk és a megoldások végrehajtása csak ebben a sorrendben lehetséges!)

### Megoldás

```
DROP TABLE dolgozó;

CREATE TABLE dolgozó
AS
SELECT *
  FROM emp
 WHERE sal+NVL(comm,0) > 500 AND
       hiredate > '1980.jan.1';

SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
```

### Eredmény

A tábla eldobva.  
A tábla létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10

7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

### 5.3. feladat

Nevezze át a dolgozó nevű tábla oszlopneveit értelemszerűen a következőkre: AZONOSÍTÓ, NÉV, MUNKAKÖR, FŐNÖK\_ID, BELÉPÉS, FIZETÉS, JUTALÉK, RÉSZLEG\_ID.

#### **Megoldás**

```
ALTER TABLE dolgozó
```

```
    RENAME COLUMN empno TO azonosító;
```

```
ALTER TABLE dolgozó
```

```
    RENAME COLUMN ename TO név;
```

```
ALTER TABLE dolgozó
```

```
    RENAME COLUMN job TO munkakör;
```

```
ALTER TABLE dolgozó
```

```
    RENAME COLUMN mgr TO főnök_id;
```

```
ALTER TABLE dolgozó
```

```
    RENAME COLUMN hiredate TO belépés;
```

```
ALTER TABLE dolgozó
```

```
    RENAME COLUMN sal TO fizetés;
```

```
ALTER TABLE dolgozó
```

```
    RENAME COLUMN comm TO jutalék;
```

```
ALTER TABLE dolgozó
```

```
    RENAME COLUMN deptno TO részleg_id;
```

```
SET linesize 50
```

```
DESC dolgozó;
```

```
SET linesize 400
```

```
COLUMN "NÉV"          FORMAT A7
```

```
COLUMN "MUNKAKÖR"     FORMAT A10
```

```
SET numwidth 5
```

```
SELECT * FROM dolgozó;
SET numwidth 10
CLEAR COLUMNS
```

**Eredmény**

A tábla módosítva.  
A tábla módosítva.

Név	Üres?	Típus
AZONOSÍTÓ		NOT NULL NUMBER(4)
NÉV		VARCHAR2(10)
MUNKAKÖR		VARCHAR2(9)
FŐNÖK_ID		NUMBER(4)
BELÉPÉS		DATE
FIZETÉS		NUMBER(7,2)
JUTALÉK		NUMBER(7,2)
RÉSZLEG_ID		NOT NULL NUMBER(2)

AZONOSÍTÓ	NÉV	MUNKAKÖR	FŐNÖK_ID	BELÉPÉS	FIZETÉS	JUTALÉK	RÉSZLEG_ID
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

## 5.4. feladat

Bővítsé a dolgozó táblát a lakkely oszloppal.

### Megoldás

```
ALTER TABLE dolgozó
    ADD lakkely VARCHAR2(18);
```

```
COLUMN "NÉV"           FORMAT A7
COLUMN "MUNKAKÖR"      FORMAT A10
COLUMN LAKHELY         FORMAT A8
SET numwidth 5
```

```
SELECT * FROM dolgozó;
```

```
SET numwidth 10
```

```
CLEAR COLUMNS
```

### Eredmény

A tábla módosítva.

AZONOSÍTÓ_NÉV	MUNKAKÖR	FŐNÖK_ID	BELÉPÉS	FIZETÉS	JUTALÉK	RÉSZLEG_ID	LAKHELY
7369 SMITH	CLERK	7902	80-DEC-17	800		20	
7499 ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	
7521 WARD	SALESMAN	7698	81-FEB-22	1250	500	30	
7566 JONES	MANAGER	7839	81-ÁPR-02	2975		20	
7654 MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	
7698 BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	
7782 CLARK	MANAGER	7839	81-JÚN-09	2450		10	
7788 SCOTT	ANALYST	7566	87-ÁPR-19	3000		20	
7839 KING	PRESIDENT		81-NOV-17	5000		10	
7844 TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	
7876 ADAMS	CLERK	7788	87-MÁJ-23	1100		20	
7900 JAMES	CLERK	7698	81-DEC-03	950		30	
7902 FORD	ANALYST	7566	81-DEC-03	3000		20	
7934 MILLER	CLERK	7782	82-JAN-23	1300		10	

14 sor kijelölve.

## 5.5. feladat

Töltsé fel a dolgozó tábla lakkely oszlopát a következőképpen:

- A. A New York-i dolgozók lakkelye is New York, de King a Long Beach-en lakik.

- B. A Dallas-i clerk munkakörűek Garland-ban, a többiek Dallas-ban laknak.
- C. A Chicago-ban dolgozók Boston-ban, a Boston-ban dolgozók pedig Chicago-ban laknak, kivéve azokat a Chicago-i dolgozókat, akiknek Blake a főnöke, mert ők Indianapolisban laknak, feltéve, hogy nem clerk munkakörűek, mert akkor sehol sem laknak.
- D. Listázza ki a dolgozók azonosítóját, nevét, munkakörét, főnökének nevét, a telephelyét és a lakhelyét a dolgozók neve szerint rendezve.

### **Megoldás (A feladatot részfeladatonként oldjuk meg)**

#### **A. részfeladat**

A New York-i dolgozók lakhelye is New York, de King a Long Beach-en lakik.

#### **A. megoldás**

```
UPDATE dolgozó
SET lakhely = 'NEW YORK'
WHERE részleg_id IN (SELECT DISTINCT részleg_id
                      FROM dolgozó,
                           dept
                     WHERE UPPER(loc) = 'NEW YORK' AND
                           deptno = részleg_id)
      AND
      UPPER(név) <> 'KING';
```

```
UPDATE dolgozó
SET lakhely = 'LONG BEACH'
WHERE UPPER(név) = 'KING';
```

#### **Eredmény**

2 sor módosítva.  
1 sor módosítva.



#### **Megjegyzés**

- Két módosító utasítással volt megoldható a részfeladat.
- A karakter típusú adatokkal való azonosítás esetén mindenki használja az UPPER (esetleg a LOWER) függvényt, mivel általában nem tudhatjuk, hogy mi a belső tárolás formája.

#### **B. részfeladat**

A Dallas-i clerk munkakörűek Garland-ban, a többiek Dallas-ban laknak.

#### **B. megoldás**

##### **B1. lépés (Először módosítandó Dallas-ban a clerk munkakörűek lakhelye)**

```
UPDATE dolgozó
SET lakhely ='GARLAND'
WHERE UPPER(munkakör) = 'CLERK' AND
```

```

részleg_id = (SELECT DISTINCT részleg_id      -- (!)
               FROM dolgozó,
                    dept
              WHERE UPPER(loc) = 'DALLAS' AND
                    deptno = részleg_id);

```

**Eredmény**

2 sor módosítva.

**B2. lépés (Másodszor azok lakhelyét állítjuk be, akik Dallas-ban dolgoznak, nem clerk munkakörűek, ők Dallas-ban laknak)**

```

UPDATE dolgozó
  SET lakhely = 'DALLAS'
 WHERE UPPER(munkakör) <> 'CLERK' AND
       részleg_id IN (SELECT részleg_id      -- (!)
                      FROM dolgozó,
                           dept
                     WHERE UPPER(loc) = 'DALLAS' AND
                           deptno = részleg_id);

```

**Eredmény**

3 sor módosítva.

**C. részfeladat**

A Chicago-ban dolgozók Boston-ban, a Boston-ban dolgozók pedig Chicago-ban laknak, kivéve azon Chicago-i dolgozókat, akiknek Blake a főnökük, mert azok Indianapolis-ban laknak, feltéve, hogy nem clerk munkakörűek, mert akkor sehol sem laknak.

**C. megoldás****C1. lépés (Azon Chicago-i dolgozóknak, akiknek főnökük Blake, a lakhelyük Indianapolis)**

```

UPDATE dolgozó
  SET lakhely = 'INDIANAPOLIS'
 WHERE részleg_id = (SELECT DISTINCT d1.részleg_id
                      FROM dolgozó d1,
                           dept d,
                           dolgozó d2
                     WHERE UPPER(loc) = 'CHICAGO'        AND
                           d1.főnök_id = d2.azonosító AND
                           d1.részleg_id = d.deptno    AND
                           UPPER(d2.név) = 'BLAKE')
          AND
          UPPER(munkakör) <> 'CLERK' AND
          UPPER(név) <> 'BLAKE';

```

**Eredmény**

4 sor módosítva.

**C2. lépés (Azon Chicago-i dolgozóknak, akiknek főnökük nem Blake, a lakhelyük Boston, a Boston-i dolgozók lakhelye pedig Chicago)**

UPDATE dolgozó

```
SET lakhely = 'BOSTON'
WHERE név IN (SELECT d1.név
                FROM dolgozó d1,
                     dept d,
                     dolgozó d2
               WHERE UPPER(loc) = 'CHICAGO'      AND
                     d1.főnök_id = d2.azonosító AND
                     d1.részleg_id = d2.deptno    AND
                     UPPER(d2.név) <> 'BLAKE');
```

UPDATE dolgozó

```
SET lakhely = 'CHICAGO'
WHERE részleg_id = (SELECT deptno
                      FROM dept
                     WHERE UPPER(loc) = 'BOSTON');
```

**Eredmény**

1 sor módosítva.

0 sor módosítva.

**Megjegyzés**

Boston-i telephely ugyan létezik, de mivel ott egyenlőre nem dolgozik senki, így az erre vonatkozó módosító utasítás nem tud sorokat módosítani.

**D. részfeladat**

Listázzuk ki a dolgozók azonosítóját, nevét, munkakörét, főnökének nevét, a dolgozó telephelyét és a lakhelyét a dolgozók neve szerint rendezve.

**D. megoldás**

```
SELECT d1.azonosító,
       d1.név,
       d1.munkakör,
       d2.név AS főnök_neve,
       d1.loc AS telephely,
       d1.lakhely
  FROM dolgozó d1,
       dept d,
       dolgozó d2
```

```

WHERE d1.főnök_id = d2.azonosító (+) AND
      d1.részleg_id = d.deptno
ORDER BY d1.név;

```

**Eredmény**

AZONOSÍTÓ NÉV	MUNKAKÖR	FŐNÖK_Neve	TELEPHELY	LAKHELY
7876 ADAMS	CLERK	SCOTT	DALLAS	GARLAND
7499 ALLEN	SALESMAN	BLAKE	CHICAGO	INDIANAPOLIS
7698 BLAKE	MANAGER	KING	CHICAGO	BOSTON
7782 CLARK	MANAGER	KING	NEW YORK	NEW YORK
7902 FORD	ANALYST	JONES	DALLAS	DALLAS
7900 JAMES	CLERK	BLAKE	CHICAGO	
7566 JONES	MANAGER	KING	DALLAS	DALLAS
7839 KING	PRESIDENT		NEW YORK	LONG BEACH
7654 MARTIN	SALESMAN	BLAKE	CHICAGO	INDIANAPOLIS
7934 MILLER	CLERK	CLARK	NEW YORK	NEW YORK
7788 SCOTT	ANALYST	JONES	DALLAS	DALLAS
7369 SMITH	CLERK	FORD	DALLAS	GARLAND
7844 TURNER	SALESMAN	BLAKE	CHICAGO	INDIANAPOLIS
7521 WARD	SALESMAN	BLAKE	CHICAGO	INDIANAPOLIS

14 sor kijelölve.

**5.6. feladat**

Hozzon létre egy részleg táblát a dept táblából. Bővítse a dolgozó táblát Kovács-csal, aki-nek a foglalkozása főnök, 9999 az azonosítója, neki nincs főnöke, de a King nevű alkalmazott a beosztottja, 2001. január 5-én lépett be a céghoz, 9999 USD a fizetése, ugyanennyi a jutá-leka, a 99-es azonosítójú Főnökség a munkahelye Budapest-en, és Soroksár-on lakik. Ehhez a bővítéshez tegyen mentési pontot.

**Megoldás****1. lépés (Készítsük el a dept tábla másolatát részleg néven)**

```
DROP TABLE részleg;
```

```

CREATE TABLE részleg
AS SELECT * FROM dept;

```

```
SELECT * FROM részleg;
```

**Eredmény**

A tábla eldobva.

A tábla létrejött.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

## 2. lépés (Bővítsük a részleg táblát az új részleg nevével, a telephelyével, és az azonosítójával)

```
INSERT INTO részleg
VALUES(99, 'FŐNÖKSÉG', 'BUDAPEST');
```

```
SELECT * FROM részleg;
```

### Eredmény

1 sor létrejött.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
99	FŐNÖKSÉG	BUDAPEST



### Megjegyzés

Mivel az eredeti dept táblában a deptno elsődleges kulcs, és idegen kulcs az emp táblában, ezért először a részleg táblába kell felvenni az új részlegazonosítót, majd ezután ebbe a részlegbe már fel lehet vonni dolgozót.

## 3. lépés (Bővítsük a dolgozó táblát Kovács adataival)

```
INSERT INTO dolgozó
(AZONOSÍTÓ, NÉV, MUNKAKÖR, FŐNÖK_ID,
BELÉPÉS,
FIZETÉS, JUTALÉK,
RÉSZLEG_ID, LAKHELY)
VALUES
(9999, 'KOVÁCS', 'FŐNÖK', NULL,
TO_DATE('2001.jánuár 5.', 'YYYY.month fmDD.'),
9999, 9999,
99, 'SOROKSÁR');
```

### Eredmény

1 sor létrejött.

**4. lépés (Mivel Kovács lett King főnöke, ezért King főnökazonosítóját módosítani kell)**

```

UPDATE dolgozó
SET főnök_id = 9999
WHERE UPPER(név) = 'KING';

SET numwidth 5
COLUMN "NÉV"          FORMAT A7
COLUMN "MUNKAKÖR"      FORMAT A10
COLUMN LAKHELY         FORMAT A12
COLUMN "FŐNÖK_ID"       FORMAT 9999
COLUMN "FIZETÉS"        FORMAT 9999
COLUMN "JUTALÉK"        FORMAT 9999

SELECT * FROM dolgozó;
CLEAR COLUMNS
SET numwidth 10

```

**Eredmény**

1 sor módosítva.

AZONOSÍTÓ	NÉV	MUNKAKÖR	FŐNÖK_ID	BELÉPÉS	FIZETÉS	JUTALÉK	RÉSZLEG_ID	LAKHELY
7369	SMITH	CLERK	7902	80-DEC-17	800		20	GARLAND
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	INDIANAPOLIS
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30	INDIANAPOLIS
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20	DALLAS
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	INDIANAPOLIS
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	BOSTON
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10	NEW YORK
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20	DALLAS
7839	KING	PRESIDENT	9999	81-NOV-17	5000		10	LONG BEACH
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	INDIANAPOLIS
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20	GARLAND
7900	JAMES	CLERK	7698	81-DEC-03	950		30	
7902	FORD	ANALYST	7566	81-DEC-03	3000		20	DALLAS
7934	MILLER	CLERK	7782	82-JAN-23	1300		10	NEW YORK
9999	KOVÁCS	FŐNÖK		01-JAN-05	9999	9999	99	SOROKSÁR

15 sor kijelölve.

**5. lépés (Helyezzük el a mentési pontot)**

SAVEPOINT Első;

**Eredmény**

A mentési pont létrejött.

**Megjegyzés**

Az új dolgozónak minden adatát fel kell venni. Ezek közül különösen a dátum bevitelére kell ügyelni. A megadott dátum formátuma: 'YYYY. month fmDD.'. Az év négy karakter, majd pont, utána a hónap neve teljes alakban és kisbetűvel írva, majd a napok száma vezető nullák nélkül. Ehhez a módosításhoz tettünk mentési pontot, melynek neve Első.

**5.7. feladat**

Bővítsse a dolgozó táblát a következő alkalmazottal:

Neve: Kelemen;

Azonosítója: 1111;

Munkaköre: megegyezik a legkisebb fizetésű dolgozó munkakörével,

Főnöke: megegyezik a Dallas-i Ford főnökével,

Belépése: az aktuális dátum,

Fizetése: a legnagyobb, 5000 USD-nál kisebb munkaköri átlagfizetés,

Jutaléka: 111 USD,

Részlege: megegyezik a Budapest-i részleg számával,

Lakóhelye: Soroksár.

A bővítéshez tegyen mentési pontot.

**Megoldás****1. lépés (Először a legkisebb fizetésű alkalmazott fizetésére van szükségünk)**

```
SELECT MIN(fizetés) AS MinFiz
FROM dolgozó;
```

**Eredmény**

```
MINFIZ
```

```
-----
```

```
800
```

**2. lépés (Ahhoz, hogy egy és csak egy munkakör értéket kapjuk meg, még egy külső le-kérdezésbe kell ezt a lekérdezést ágyazní)****2A. megoldás (FROM-beli allekérdezéssel – inline nézettel)**

```
SELECT munkakör
FROM dolgozó,
     (SELECT MIN(fizetés) AS MinFiz
      FROM dolgozó) MinFizDolg
WHERE dolgozó.fizetés = MinFizDolg.MinFiz;
```

**2B. megoldás (WHERE-beli allekérdezéssel – ez a gyorsabb!)**

```
SELECT munkakör
      FROM dolgozó
     WHERE fizetés = (SELECT MIN(fizetés) AS MinFiz
                        FROM dolgozó);
```

**Eredmény**

MUNKAKÖR

---

CLERK

**3. lépés (A Dallas-i Ford főnökének meghatározása)**

```
SELECT főnök_id
      FROM dolgozó,
           részleg
     WHERE UPPER(név) = 'FORD'                               AND
           dolgozó.részleg_id = részleg.deptno    AND
           UPPER(loc) = 'DALLAS';
```

**Eredmény**

FŐNÖK\_ID

---

7566

**4. lépés (Belépési dátuma: az aktuális dátum)**

Dátumként a Sysdate rendszerváltozót kell megadni.

**5. lépés (Fizetése: a munkaköri fizetések legnagyobbikával egyenlő, de kisebb mint 5000 USD)**

```
SELECT ROUND(MAX(Átlag))
      FROM (SELECT ROUND(AVG(fizetés)) AS átlag
              FROM dolgozó
             GROUP BY munkakör)
     WHERE átlag < 5000;
```

**Eredmény**

ROUND(MAX(ÁTLAG))

---

3000

**6. lépés (Jutaléka: 111 USD)**

Jutalékként a 111 értéket kell megadni.

**7. lépés (Részlege: megegyezik a Budapest-i részleg számával)**

A Budapest-i részleg száma:

```
SELECT deptno
      FROM részleg
     WHERE UPPER(loc) = 'BUDAPEST';
```

**Eredmény**

DEPTNO
99

**8. lépés (Lakóhelye: Soroksár)**

Lakóhelyként 'Soroksár'-t kell megadni.

**9. lépés (A beszúrandó sor)**

```
INSERT INTO dolgozó
(AZONOSÍTÓ, NÉV,
MUNKAKÖR,
FŐNÖK_ID,
BELÉPÉS,
FIZETÉS,
JUTALÉK,
RÉSZLEG_ID,
LAKHELY)
VALUES
(1111, 'KELEMEN',
(SELECT munkakör
   FROM dolgozó
  WHERE fizetés = (SELECT MIN(fizetés) AS MinFiz
                    FROM dolgozó)),
(SELECT főnök_id
   FROM dolgozó,
        részleg
  WHERE UPPER(név) = 'FORD'      AND
        UPPER(loc) = 'DALLAS'    AND
        dolgozó.részleg_id = részleg.deptno),
SYSDATE,
(SELECT MAX(Átlag)
   FROM (SELECT ROUND(ROUND(AVG(fizetés))) AS átlag
         FROM dolgozó
        GROUP BY munkakör)
  WHERE átlag < 5000),
111;
```

```
(SELECT deptno
   FROM részleg
 WHERE UPPER(loc) = 'BUDAPEST'),
'SOROKSÁR');

COLUMN "NÉV"      FORMAT A7
COLUMN "MUNKAKÖR" FORMAT A10
COLUMN LAKHELY    FORMAT A12
COLUMN "FŐNÖK_ID" FORMAT 9999
COLUMN "FIZETÉS"  FORMAT 9999
COLUMN "JUTALÉK"  FORMAT 9999
```

```
SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
CLEAR COLUMNS
```

### Eredmény

1 sor létrejött.

AZONOSÍTÓ NÉV	MUNKAKÖR	FŐNÖK_ID	BELÉPÉS	FIZETÉS	JUTALÉK	RÉSZLEG_ID	LAKHELY
7369 SMITH	CLERK	7902	80-DEC-17	800		20	GARLAND
7499 ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	INDIANAPOLIS
7521 WARD	SALESMAN	7698	81-FEB-22	1250	500	30	INDIANAPOLIS
7566 JONES	MANAGER	7839	81-ÁPR-02	2975		20	DALLAS
7654 MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	INDIANAPOLIS
7698 BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	BOSTON
7782 CLARK	MANAGER	7839	81-JÚN-09	2450		10	NEW YORK
7788 SCOTT	ANALYST	7566	87-ÁPR-19	3000		20	DALLAS
7839 KING	PRESIDENT	9999	81-NOV-17	5000		10	LONG BEACH
7844 TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	INDIANAPOLIS
7876 ADAMS	CLERK	7788	87-MÁJ-23	1100		20	GARLAND
7900 JAMES	CLERK	7698	81-DEC-03	950		30	
7902 FORD	ANALYST	7566	81-DEC-03	3000		20	DALLAS
7934 MILLER	CLERK	7782	82-JAN-23	1300		10	NEW YORK
1111 KELEMEN	CLERK	7566	04-NOV-29	3000	111		SOROKSÁR
9999 KOVÁCS	FŐNÖK		01-JAN-05	9999	9999	99	SOROKSÁR

16 sor kijelölve.

### 10. lépés (Mentési pont)

SAVEPOINT Második;

## Eredmény

A mentési pont létrejött.

## 5.8. feladat

Bővítsse a dolgozó táblát két dolgozával. Az adatok egyéni legyenek. Listázza ki a táblát. Érvénytelenítse az eddigi bővítéseket, majd véglegesítse az adattáblát.

### Megoldás

```

INSERT INTO dolgozó
    VALUES(3030,'SAS','PÉK', 4444,'04-OKT-09',3333,333,99, 'BUDAPEST');

INSERT INTO dolgozó
    VALUES(2020,'PAPP', 'MÉRNÖK',3210,'04-OKT-09',4343,NULL,99, 'TATA');

SET numwidth 5
COLUMN "NÉV"          FORMAT A7
COLUMN "MUNKAKÖR"      FORMAT A10
COLUMN LAKHELY        FORMAT A12
COLUMN "FŐNÖK_ID"      FORMAT 9999
COLUMN "FIZETÉS"       FORMAT 9999
COLUMN "JUTALÉK"       FORMAT 9999
SELECT * FROM dolgozó;

ROLLBACK TO Második;
SELECT * FROM dolgozó;

ROLLBACK TO Első;
SELECT * FROM dolgozó;

ROLLBACK;
COMMIT;
SELECT * FROM dolgozó;
SET numwidth 10
CLEAR COLUMNS

```

## Eredmény

1 sor létrejött.

1 sor létrejött.

AZONOSÍTÓ	NÉV	MUNKAKÖR	FŐNÖK_ID	BELÉPÉS	FIZETÉS	JUTALÉK	RÉSZLEG_ID	LAKHELY
7369	SMITH	CLERK	7902	80-DEC-17	800		20	GARLAND
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	INDIANAPOLIS

7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30	INDIANAPOLIS
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20	DALLAS
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	INDIANAPOLIS
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	BOSTON
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10	NEW YORK
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20	DALLAS
7839	KING	PRESIDENT	9999	81-NOV-17	5000		10	LONG BEACH
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	INDIANAPOLIS
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20	GARLAND
7900	JAMES	CLERK	7698	81-DEC-03	950		30	
7902	FORD	ANALYST	7566	81-DEC-03	3000		20	DALLAS
7934	MILLER	CLERK	7782	82-JAN-23	1300		10	NEW YORK
1111	KELEMEN	CLERK	7566	04-NOV-29	3000	111	99	SOROKSÁR
9999	KOVÁCS	FŐNÖK		01-JAN-05	9999	9999	99	SOROKSÁR
3030	SAS	PÉK	4444	04-OKT-09	3333	333	99	BUDAPEST
2020	PAPP	MÉRNÖK	3210	04-OKT-09	4343		99	TATA

18 sor kijelölve.

A visszaállítás befejeződött.

AZONOSÍTÓ NÉV	MUNKAKÖR	FŐNÖK_ID	BELÉPÉS	FIZETÉS	JUTALEK	RÉSZLEG_ID	LAKHELY	
7369	SMITH	CLERK	7902	80-DEC-17	800		20	GARLAND
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	INDIANAPOLIS
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30	INDIANAPOLIS
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20	DALLAS
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	INDIANAPOLIS
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	BOSTON
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10	NEW YORK
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20	DALLAS
7839	KING	PRESIDENT	9999	81-NOV-17	5000		10	LONG BEACH
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	INDIANAPOLIS
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20	GARLAND
7900	JAMES	CLERK	7698	81-DEC-03	950		30	
7902	FORD	ANALYST	7566	81-DEC-03	3000		20	DALLAS
7934	MILLER	CLERK	7782	82-JAN-23	1300		10	NEW YORK
1111	KELEMEN	CLERK	7566	04-NOV-29	3000	111		SOROKSÁR
9999	KOVÁCS	FŐNÖK		01-JAN-05	9999	9999	99	SOROKSÁR

16 sor kijelölve.

A visszaállítás befejeződött.

AZONOSÍTÓ NÉV	MUNKAKÖR	FŐNÖK_ID	BELÉPÉS	FIZETÉS	JUTALÉK	RÉSZLEG_ID	LAKHELY
7369 SMITH	CLERK	7902	80-DEC-17	800		20	GARLAND
7499 ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	INDIANAPOLIS
7521 WARD	SALESMAN	7698	81-FEB-22	1250	500	30	INDIANAPOLIS
7566 JONES	MANAGER	7839	81-ÁPR-02	2975		20	DALLAS
7654 MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	INDIANAPOLIS
7698 BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	BOSTON
7782 CLARK	MANAGER	7839	81-JÚN-09	2450		10	NEW YORK
7788 SCOTT	ANALYST	7566	87-ÁPR-19	3000		20	DALLAS
7839 KING	PRESIDENT	9999	81-NOV-17	5000		10	LONG BEACH
7844 TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	INDIANAPOLIS
7876 ADAMS	CLERK	7788	87-MÁJ-23	1100		20	GARLAND
7900 JAMES	CLERK	7698	81-DEC-03	950		30	
7902 FORD	ANALYST	7566	81-DEC-03	3000		20	DALLAS
7934 MILLER	CLERK	7782	82-JAN-23	1300		10	NEW YORK
9999 KOVÁCS	FŐNÖK		01-JAN-05	9999	9999	99	SOROKSÁR

15 sor kijelölve.

A visszaállítás befejeződött.

A jóváhagyás befejeződött.

AZONOSÍTÓ NÉV	MUNKAKÖR	FŐNÖK_ID	BELÉPÉS	FIZETÉS	JUTALÉK	RÉSZLEG_ID	LAKHELY
7369 SMITH	CLERK	7902	80-DEC-17	800		20	GARLAND
7499 ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	INDIANAPOLIS
7521 WARD	SALESMAN	7698	81-FEB-22	1250	500	30	INDIANAPOLIS
7566 JONES	MANAGER	7839	81-ÁPR-02	2975		20	DALLAS
7654 MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	INDIANAPOLIS
7698 BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	BOSTON
7782 CLARK	MANAGER	7839	81-JÚN-09	2450		10	NEW YORK
7788 SCOTT	ANALYST	7566	87-ÁPR-19	3000		20	DALLAS
7839 KING	PRESIDENT	9999	81-NOV-17	5000		10	LONG BEACH
7844 TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	INDIANAPOLIS
7876 ADAMS	CLERK	7788	87-MÁJ-23	1100		20	GARLAND
7900 JAMES	CLERK	7698	81-DEC-03	950		30	
7902 FORD	ANALYST	7566	81-DEC-03	3000		20	DALLAS
7934 MILLER	CLERK	7782	82-JAN-23	1300		10	NEW YORK

14 sor kijelölve.

## 5.9. feladat

- A. Írjon szkript programot, amely (eldob és) létrehoz egy dolgozó és egy részleg táblát az emp és dept táblákból, majd az alábbi megszorításokkal látja el. Megkísérél minden feltételhez egy-egy rekordot felvinni a dolgozó és részleg táblába, amelyek közül egy megfelel, egy pedig nem felel meg e megszorításoknak.
- B. Legyen ename egyedi kulcs a dolgozó táblában, és kérdezze le e megszorítást.
- C. Törölje a dolgozó táblából a névre vonatkozó egyedi megszorítást.
- D. Legyen az empno oszlop elsődleges kulcs a dolgozó táblában.
- E. Legyen a deptno elsődleges kulcs a részleg, és idegen kulcs a dolgozó táblában.
- F. Egy részleg csak New York, Dallas, Chicago vagy Boston valamelyikében lehet.
- G. Egy manager nem kaphat 2000 USD-nál kevesebb, és 3000 USD-nál több fizetést.
- H. Egy clerk nem kaphat 500 USD-nál kevesebb és 1500 USD-nál több fizetést.
- I. Az új dolgozó belépési dátuma 1978 és a legkésőbbi dátum közé essen.
- J. Az új dolgozó belépési dátuma 1981. február 19. és 2004. október 25. közé essen.

**Megoldás (A feladatot részfeladatonként oldjuk meg)**

### A. részfeladat

Írjon szkript programot, amely (eldob és) létrehoz egy dolgozó és egy részleg táblát az emp és dept táblákból, majd az alábbi megszorításokkal látja el. Megkísérél minden feltételhez egy-egy rekordot felvinni a dolgozó és részleg táblába, amelyek közül egy megfelel, egy pedig nem felel meg e megszorításoknak.

### A. megoldás

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS
SELECT * FROM emp;

DROP TABLE részleg;
CREATE TABLE részleg
AS
SELECT * FROM dept;
```

### Eredmény

- A tábla eldobva.
- A tábla létrejött.
- A tábla eldobva.
- A tábla létrejött.

### B. részfeladat

Legyen ename egyedi kulcs a dolgozó táblában, és kérdezze le e megszorítást.

**B. megoldás**

```
ALTER TABLE dolgozó
  DROP CONSTRAINT egyedil;
```

```
ALTER TABLE dolgozó
  ADD CONSTRAINT egyedil UNIQUE(ename);
```

```
INSERT INTO dolgozó
  VALUES(1010, 'JONES', 'TANÁR', 3210, '04-OKT-09', 3333, 333, 40);
INSERT INTO dolgozó
  VALUES(2020, 'PAPP', 'MÉRNÖK', 3210, '04-OKT-09', 4343, NULL, 40);
SET numwidth 5
SELECT * FROM dolgozó;
```

```
BREAK ON "Tábla tulajdonosa" ON "Neve"
COLUMN "Tábla tulajdonosa" HEADING " Tábla tulajdonosa" FORMAT A12
COLUMN "Neve" FORMAT A8
COLUMN "Típusa" FORMAT A6
COLUMN "Leírása" FORMAT A20
SELECT SUBSTR(owner,1,20) AS "Tábla tulajdonosa",
       SUBSTR(table_name,1,12) AS "Neve",
       constraint_name AS "Megszorítás Neve",
       CONCAT(constraint_type, ' ') AS "Típusa",
       search_condition AS "Leírása"
  FROM user_constraints
 WHERE UPPER(table_name) = 'DOLGOZÓ';
CLEAR BREAKS
CLEAR COLUMNS
SET numwidth 10
```

**Eredmény**

A tábla módosítva.

A tábla módosítva.

```
INSERT INTO dolgozó
*
Hiba a(z) 1. sorban:
ORA-00001: a(z) (SCOTT.EGYEDI1) egyediségre vonatkozó megszorítás nem teljesül
1 sor létrejött.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20

7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
2020	PAPP	MÉRNÖK	3210	04-OKT-09	4343		40

15 sor kijelölve..

Tábla	tulajdonosa	Neve	Megszorítás	Neve	Típusa	Leírása
	SCOTT	DOLGOZÓ	EGYEDI1		U	



### Megjegyzés

Az adatszótárbeli nézet lekérdezése után is láthatjuk, hogy létrejött az egyedil nevű egyedi megszorítás (U a UNIQUE rövidítése). A megszorítás akkor nem jön létre, ha olyan feltételt szabunk, amely nem felel meg a táblában már meglévő adatoknak.

### C. részfeladat

Törölje a dolgozó táblából a névre vonatkozó egyedi megszorítást.

### C. megoldás

```
ALTER TABLE dolgozó
  DROP CONSTRAINT egyedi1;
```

### Eredmény

A tábla módosítva.

### D. részfeladat

Legyen az empno oszlop elsőleges kulcs a dolgozó táblában.

### D. megoldás

```
ALTER TABLE dolgozó
  DROP CONSTRAINT kulcs1;
```

```

ALTER TABLE dolgozó
  ADD CONSTRAINT kulcs1 PRIMARY KEY (empno);

INSERT INTO dolgozó
  VALUES(7788,'KISS','TANÁR',4141,'04-OKT-09',3333,333,40);

BREAK ON "Tábla tulajdonosa" ON "Neve"
COLUMN "Tábla tulajdonosa" HEADING " Tábla tulajdonosa" FORMAT A12
COLUMN "Neve" FORMAT A8
COLUMN "Típusa" FORMAT A6
COLUMN "Leírása" FORMAT A14
SELECT SUBSTR(owner,1,20) AS "Tábla tulajdonosa",
       SUBSTR(table_name,1,12) AS "Neve",
       constraint_name AS "Megszorítás Neve",
       CONCAT(constraint_type, ' ') AS "Típusa",
       search_condition AS "Leírása"
  FROM user_constraints
 WHERE UPPER(table_name) = 'DOLGOZÓ';
SET numwidth 5
SELECT * FROM dolgozó;
CLEAR COLUMNS
CLEAR BREAKS
SET numwidth 10

```

### Eredmény

A tábla módosítva.

```

INSERT INTO dolgozó
  VALUES(5555,'NAGY','MÉRNÖK',8765,'04-OKT-09',3434,NULL,40);

2  VALUES(7788,'KISS','TANÁR',4141,'04-OKT-09',3333,333,40);
INSERT INTO dolgozó
*
Hiba a(z) 1. sorban:
ORA-00001: a(z) (SCOTT.KULCS1) egyediségre vonatkozó megszorítás nem teljesül
1 sor létrejött.

```

Tábla	tulajdonosa	Neve	Megszorítás	Neve	Típusa	Leírása
SCOTT	DOLGOZÓ	EGYEDI1	KULCS1		U	P

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
2020	PAPP	MÉRNÖK	3210	04-OKT-09	4343		40
5555	NAGY	MÉRNÖK	8765	04-OKT-09	3434		40

16 sor kijelölve.

### E. részfeladat

Legyen a deptno elsődleges kulcs a részleg, és idegen kulcs a dolgozó táblában.

### E. megoldás

```
ALTER TABLE részleg
  DROP CONSTRAINT C2;
ALTER TABLE dolgozó
  DROP CONSTRAINT C3;
```

```
ALTER TABLE részleg
  ADD CONSTRAINT C2 PRIMARY KEY (deptno);
ALTER TABLE dolgozó
  ADD CONSTRAINT C3 FOREIGN KEY (deptno)
    REFERENCES részleg(deptno) ON DELETE CASCADE;
```

```
BREAK ON "Tábla tulajdonosa" ON "Neve"
COLUMN "Tábla tulajdonosa" HEADING " Tábla tulajdonosa" FORMAT A12
COLUMN "Neve" FORMAT A8
COLUMN "Típusa" FORMAT A6
COLUMN "Leírása" FORMAT A14
SELECT SUBSTR(owner,1,20) AS "Tábla tulajdonosa",
       SUBSTR(table_name,1,12) AS "Neve",
       constraint_name AS "Megszorítás Neve",
```

```

    CONCAT(constraint_type, '      ') AS "Típusa",
    search_condition           AS "Leírása"
  FROM user_constraints
 WHERE UPPER(table_name) = 'DOLGOZÓ';

SELECT * FROM dolgozó;

SELECT SUBSTR(table_name,1,12)          AS "Tábla Neve",
       constraint_name            AS "Megszorítás Neve",
       CONCAT(constraint_type, '      ') AS "Típusa",
       search_condition           AS "Leírása"
  FROM user_constraints
 WHERE UPPER(table_name) = 'RÉSZLEG';

CLEAR COLUMNS
CLEAR BREAKS

INSERT INTO dolgozó
  VALUES(7777,'KISS','TANÁR',4141,'04-OKT-09',3333,333,70);
INSERT INTO részleg
  VALUES(70,'TANSZÉK',NULL);
INSERT INTO dolgozó
  VALUES(5566,'TUDOR','MÉRNÖK',8765,'04-OKT-09',3434,NULL,70);
SET numwidth 5
SELECT * FROM dolgozó;
SELECT * FROM részleg;
SET numwidth 10

```

### Eredmény

A tábla módosítva.

A tábla módosítva.

A tábla módosítva.

A tábla módosítva.

#### Tábla

tulajdonosa	Tábla Neve	Megszorítás Neve	Típusa	Leírása
SCOTT	DOLGOZÓ	C3	R	
		EGYEDI1	U	
		KULCSI	P	

#### Tábla Neve      Megszorítás Neve      Típusa      Leírása

RÉSZLEG	C2	P
---------	----	---

INSERT INTO dolgozó

\*

Hiba a(z) 1. sorban:

ORA-02291: integritás megszorítás (SCOTT.C3) megsértés - a szülő kulcs nem található meg

1 sor létrejött.

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
2020	PAPP	MÉRNÖK	3210	04-OKT-09	4343		40
5555	NAGY	MÉRNÖK	8765	04-OKT-09	3434		40
5566	TUDOR	MÉRNÖK	8765	04-OKT-09	3434		70

17 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
70	TANSZÉK	

#### F. részfeladat

Egy részleg csak New York, Dallas, Chicago vagy Boston valamelyikében lehet.

**F. megoldás**

```
ALTER TABLE részleg
  DROP CONSTRAINT C4;
```

```
ALTER TABLE részleg
  ADD CONSTRAINT C4 CHECK (UPPER(loc) IN ('NEW YORK', 'DALLAS',
                                             'CHICAGO', 'BOSTON'));
```

```
INSERT INTO részleg
  VALUES(50, 'SZERVÍZ', 'BUDAPEST');
INSERT INTO részleg
  VALUES(50, 'SZERVÍZ', 'DALLAS');
```

```
SELECT * FROM részleg;
```

**Eredmény**

A tábla módosítva.

A tábla módosítva.

```
INSERT INTO részleg
```

```
*
```

Hiba a(z) 1. sorban:

ORA-02290: ellenőrző megszorítás (SCOTT.C4) megsértése

1 sor létrejött.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
70	TANSZÉK	
50	SZERVÍZ	DALLAS

6 sor kijelölve.

**Megjegyzés**

A megszorítást definíáló utasításban nem lehet allekérdezés.

**G. részfeladat**

Egy manager nem kaphat 2000 USD-nál kevesebb és 3000 USD-nál több fizetést.

### G. megoldás

```

ALTER TABLE dolgozó
    DROP CONSTRAINT C5;

ALTER TABLE dolgozó
    ADD CONSTRAINT C5 CHECK
        ((job = 'MANAGER' AND sal BETWEEN 2000 AND 3000) OR
         (job != 'MANAGER'));

INSERT INTO dolgozó
    VALUES(1010,'JÓNÁS','MANAGER',3210,'04-OKT-09',1888,333,70);
INSERT INTO dolgozó
    VALUES(1234,'VIDOR','MANAGER',3210,'04-OKT-10',2300,NULL,40);
SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10

```

### Eredmény

A tábla módosítva.

A tábla módosítva.

INSERT INTO dolgozó

\*

Hiba a(z) 1. sorban:

ORA-02290: ellenőrző megszorítás (SCOTT.C5) megsértése

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
2020	PAPP	MÉRNÖK	3210	04-OKT-09	4343		40

5555	NAGY	MÉRNÖK	8765	04-OKT-09	3434	40
5566	TUDOR	MÉRNÖK	8765	04-OKT-09	3434	70
1234	VIDOR	MANAGER	3210	04-OKT-10	2300	40

18 sor kijelölve.



### Megjegyzés

Figyeljünk fel a C5 CHECK-megszorítás feltételrészében szereplő kifejezésre. Egy CHECK-megszorítás kifejezésének minden a teljes táblára kell értelmezve lennie, ezért volt szükség az OR (job != 'MANAGER') részkifejezésre.

## H. részfeladat

Egy clerk nem kaphat 500 USD-nál kevesebb, és 1500 USD-nál több fizetést.

### H. megoldás

```

ALTER TABLE dolgozó
  DROP CONSTRAINT C6;
ALTER TABLE dolgozó
  ADD CONSTRAINT C6 CHECK
    ((job = 'CLERK' AND sal BETWEEN 500 AND 1500) OR
     (job != 'CLERK'));

INSERT INTO dolgozó
  VALUES(1010,'MÁTÉ','CLERK', 3210,'04-OKT-09',3333,333,70);
INSERT INTO dolgozó
  VALUES(2345,'SZUNDI','CLERK',3210,'04-OKT-10',1400,NULL,70);
SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10

```

### Eredmény

A tábla módosítva.

A tábla módosítva.

```
INSERT INTO dolgozó
```

\*

Hiba a(z) 1. sorban:

ORA-02290: ellenőrző megszorítás (SCOTT.C6) megsértése

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30

7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
2020	PAPP	MÉRNÖK	3210	04-OKT-09	4343		40
5555	NAGY	MÉRNÖK	8765	04-OKT-09	3434		40
5566	TUDOR	MÉRNÖK	8765	04-OKT-09	3434		70
1234	VIDOR	MANAGER	3210	04-OKT-10	2300		40
2345	SZUNDI	CLERK	3210	04-OKT-10	1400		70

19 sor kijelölve.

### I. részfeladat

Az új dolgozó belépési dátuma 1978 és a legkésőbbi dátum közé essen.

#### I. megoldás

```
ALTER TABLE dolgozó
```

```
    DROP CONSTRAINT idő1;
```

```
ALTER TABLE dolgozó
```

```
    DROP CONSTRAINT idő2;
```

```
ALTER TABLE dolgozó
```

```
    ADD CONSTRAINT idő1 CHECK
```

```
        (TO_CHAR(hiredate, 'YYYY') BETWEEN '1978' AND '2004');
```

```
ALTER TABLE dolgozó
```

```
    ADD CONSTRAINT idő2 CHECK
```

```
        (TO_NUMBER(TO_CHAR(hiredate, 'YYYY')) BETWEEN 1978 AND 2004);
```

```
INSERT INTO dolgozó
```

```
    VALUES(2222,'PÉTER','CLERK',3210,'05-OKT-09',1000,333,70);
```

```
INSERT INTO dolgozó
```

```
    VALUES(3480,'HAPCI','MÉRNÖK',3210,'1995-OKT-10',2500,NULL,70);
```

```
SET numwidth 5
```

```
SELECT * FROM dolgozó;
```

```
SET numwidth 10
```

## Eredmény

A tábla módosítva.

A tábla módosítva.

A tábla módosítva.

A tábla módosítva.

INSERT INTO dolgozó

\*

Hiba a(z) 1. sorban:

ORA-02290: ellenőrző megszorítás (SCOTT.IDÖ2) megsértése

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
2020	PAPP	MÉRNÖK	3210	04-OKT-09	4343		40
5555	NAGY	MÉRNÖK	8765	04-OKT-09	3434		40
5566	TUDOR	MÉRNÖK	8765	04-OKT-09	3434		70
1234	VIDOR	MANAGER	3210	04-OKT-10	2300		40
2345	SZUNDI	CLERK	3210	04-OKT-10	1400		70
3480	HAPCI	MÉRNÖK	3210	95-OKT-10	2500		70

20 sor kijelölve.



### Megjegyzés

A feladat megoldásához vegyük figyelembe a következőket:

- A dolgozó tábla hiredate oszlopa dátum típusú.
- A felső határ nem lehet kisebb, mint a táblabeli értékek.
- A dátumra vonatkozó megszorításba nem tehető a sysdate rendszerdátum lekérdezés, csak dátumkonstans.

Végül tekintsük át a karakterszorozatokra és dátumokra vonatkozó konverziós függvényeket:

```
TO_CHAR({szám | dátum} [, formátummaszk])
TO_NUMBER(karaktersorozat [, formátummaszk])
TO_DATE(karaktersorozat [, formátummaszk])
```

### J. részfeladat

Az új dolgozó belépési dátuma 1981. február 19. és 2004. október 25. közé essen.

### J. megoldás (Dobjuk el a korábbi idő1 és idő2 megszorítást)

```
ALTER TABLE dolgozó
    DROP CONSTRAINT idő1;
ALTER TABLE dolgozó
    DROP CONSTRAINT idő2;

ALTER TABLE dolgozó
    DROP CONSTRAINT idő3;
ALTER TABLE dolgozó
    ADD CONSTRAINT idő3 CHECK
        (TO_CHAR(hiredate, 'YYYY.month.DD.')
         BETWEEN
             TO_CHAR('1978.február.02.')
         AND
             TO_CHAR('2004.október.25.'));

INSERT INTO dolgozó
    VALUES(2002,'TAMÁS','CLERK', 3210,
           TO_DATE(sysdate,'YYYY.month.DD.'),1500,333,40);
INSERT INTO dolgozó
    VALUES(2990,'SZENDE','MÉRNÖK',3210,
           TO_DATE('1995.október.09','YYYY.month.DD.'),2500,NULL,70);
SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
```

### Eredmény

A tábla módosítva.  
A tábla módosítva.  
A tábla módosítva.  
A tábla módosítva.

```
INSERT INTO dolgozó
*
```

Hiba a(z) 1. sorban:

ORA-02290: ellenőrző megszorítás (SCOTT.ID03) megsértése

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
2020	PAPP	MÉRNÖK	3210	04-OKT-09	4343		40
5555	NAGY	MÉRNÖK	8765	04-OKT-09	3434		40
5566	TUDOR	MÉRNÖK	8765	04-OKT-09	3434		70
1234	VIDOR	MANAGER	3210	04-OKT-10	2300		40
2345	SZUNDI	CLERK	3210	04-OKT-10	1400		70
3480	HAPCI	MÉRNÖK	3210	95-OKT-10	2500		70
2990	SZENDE	MÉRNÖK	3210	95-OKT-09	2500		70

21 sor kijelölve.

## 5.10. feladat

Hozza létre az emp táblával azonos dolgozó táblát. Növelje meg ebben a részlegük átlagfizeté- sénél alacsonyabb fizetésű dolgozók havi bérét saját fizetésük 20%-ával.

### Megoldás

#### 1. lépés (Részlegenként mennyi az átlagfizetés, és kiknek van ennél kevesebb?)

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
```

```
SELECT empno,
       ename,
       sal,
```

```

        ROUND(rész.átlag)
FROM dolgozó,
     (SELECT AVG(sal) AS átlag,
              deptno
       FROM dolgozó
      GROUP BY deptno) rész
 WHERE dolgozó.deptno = rész.deptno AND
       sal < rész.átlag;

```

**Eredmény**

A tábla eldobva.  
A tábla létrejött.

EMPNO	ENAME	SAL	ÁTLAG
7782	CLARK	2450	2917
7934	MILLER	1300	2917
7876	ADAMS	1100	2175
7369	SMITH	800	2175
7654	MARTIN	1250	1567
7844	TURNER	1500	1567
7900	JAMES	950	1567
7521	WARD	1250	1567

8 sor kijelölve.

**2. lépés (Ennek a nyolc dolgozónak a fizetését módosítsuk a saját fizetésük 20%-ával)**

```

UPDATE dolgozó
    SET sal = 1.2*sal
 WHERE empno IN (SELECT empno
                  FROM dolgozó,
                       (SELECT AVG(sal) AS átlag,
                               deptno
                     FROM dolgozó
                    GROUP BY deptno) rész
                  WHERE dolgozó.deptno = rész.deptno AND
                        sal < rész.átlag);

SET numwidth 6
SELECT * FROM dolgozó;
SET numwidth 10

```

**Eredmény**

8 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	960		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1500	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1500	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2940		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1800	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1320		20
7900	JAMES	CLERK	7698	81-DEC-03	1140		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1560		10

14 sor kijelölve.

## 5.11. feladat

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát. Ebben törölje a legjobban fizetett Dallas-i dolgozót.

### Megoldás

#### 1. lépés (Mennyi a legnagyobb fizetés Dallas-ban?)

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
```

```
DROP TABLE részleg;
CREATE TABLE részleg
AS SELECT * FROM dept;
```

```
SELECT MAX(sal) AS MaxFiz,
       loc
  FROM dolgozó,
       részleg
 WHERE dolgozó.deptno = részleg.deptno
 GROUP BY loc
 HAVING UPPER(loc) = 'DALLAS';
```

**Eredmény**

MAXFIZ LOC

-----  
3000 DALLAS**2. lépés (Ki a legnagyobb fizetésű Dallas-ban? Többen is lehetnek!)**

```
SELECT empno
  FROM dolgozó
 WHERE sal = (SELECT MAX(sal)
                FROM dolgozó,
                     részleg
               WHERE dolgozó.deptno = részleg.deptno
                 GROUP BY loc
                HAVING UPPER(loc) = 'DALLAS');
```

**Eredmény**

EMPNO

-----  
7902  
7788**3. lépés (A Dallas-i legnagyobb fizetésűek törlése)**

```
DELETE dolgozó
 WHERE empno IN
       (SELECT empno
          FROM dolgozó
         WHERE sal = (SELECT MAX(sal)
                        FROM dolgozó,
                             részleg
                       WHERE dolgozó.deptno = részleg.deptno
                         GROUP BY loc
                        HAVING UPPER(loc) = 'DALLAS'));
```

```
SET numwidth 6
SELECT * FROM dolgozó;
SET numwidth 10
```

**Eredmény**

2 sor törölve.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30

7566 JONES	MANAGER	7839	81-ÁPR-02	2975	20
7654 MARTIN	SALESMAN	7698	81-SZE-28	1250	1400
7698 BLAKE	MANAGER	7839	81-MÁJ-01	2850	30
7782 CLARK	MANAGER	7839	81-JÚN-09	2450	10
7839 KING	PRESIDENT		81-NOV-17	5000	10
7844 TURNER	SALESMAN	7698	81-SZE-08	1500	0
7876 ADAMS	CLERK	7788	87-MÁJ-23	1100	20
7900 JAMES	CLERK	7698	81-DEC-03	950	30
7934 MILLER	CLERK	7782	82-JAN-23	1300	10

12 sor kijelölve.

## 5.12. feladat

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát. Bővíts ezt egy jövedelem nevű oszloppal, melyben összegzi a fizetést és a jutaléket. Ezután listázza ki e tábla tartalmát elsődlegesen a jövedelem szerint csökkenően, másodlagosan pedig a dolgozók neve szerint növekvő módon rendezve.

### Megoldás

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

ALTER TABLE dolgozó
ADD jövedelem NUMBER(7);

UPDATE dolgozó
SET jövedelem = sal+NVL(comm,0);

SET numwidth 5
SELECT *
FROM dolgozó
ORDER BY jövedelem DESC,
ename ASC;
SET numwidth 10
```

### Eredmény

A tábla eldobva.  
A tábla létrejött.  
A tábla módosítva.  
14 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	JÖVEDELEM
7839	KING	PRESIDENT		81-NOV-17	5000		10	5000
7902	FORD	ANALYST	7566	81-DEC-03	3000		20	3000
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20	3000
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20	2975
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	2850
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	2650
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10	2450
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	1900
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30	1750
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	1500
7934	MILLER	CLERK	7782	82-JAN-23	1300		10	1300
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20	1100
7900	JAMES	CLERK	7698	81-DEC-03	950		30	950
7369	SMITH	CLERK	7902	80-DEC-17	800		20	800

14 sor kijelölve.

### 5.13. feladat

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát, majd minden olyan dolgozónak, akinek legalább egy beosztottja van, egyszeri 1000 USD jutaléket ad (comm), végül listázza ki a dolgozók tábláját elsődlegesen a jutalék szerint csökkenően, másodlagosan pedig a dolgozók neve szerint növekvő módon rendezve.

#### Megoldás

##### 1. lépés (Azon főnökök kiválasztása, akiknek legalább egy beosztottjuk van)

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
```

```
SELECT dolg.mgr AS azonosító
  FROM dolgozó dolg,
       dolgozó főnök
 WHERE dolg.mgr = főnök.empno
 GROUP BY dolg.mgr
 HAVING COUNT(dolg.job) >= 1;
```

#### Eredmény

AZONOSÍTÓ

```
-----
7566
7698
```

7782  
7788  
7839  
7902

6 sor kijelölve.

## 2. lépés (Módosítás)

UPDATE dolgozó

```
SET comm = NVL(comm,0) + 1000
WHERE empno IN (SELECT dolg.mgr
                  FROM dolgozó dolg,
                       dolgozó főnök
                 WHERE dolg.mgr = főnök.empno
               GROUP BY dolg.mgr
              HAVING COUNT(dolg.job) >= 1);
```

```
SET numwidth 6
SELECT *
  FROM dolgozó
 ORDER BY comm DESC, ename;
SET numwidth 10
```

## Eredmény

6 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100	20	
7900	JAMES	CLERK	7698	81-DEC-03	950	30	
7934	MILLER	CLERK	7782	82-JAN-23	1300	10	
7369	SMITH	CLERK	7902	80-DEC-17	800	20	
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850	1000	30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450	1000	10
7902	FORD	ANALYST	7566	81-DEC-03	3000	1000	20
7566	JONES	MANAGER	7839	81-ÁPR-02	2975	1000	20
7839	KING	PRESIDENT		81-NOV-17	5000	1000	10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000	1000	20
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30

14 sor kijelölve.

## 5.14. feladat

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát, majd megnöveli a főnökök (mgr) fizetését 10%-kal, végül listázza ki őket (csak a főnököket) oly módon, hogy elsődlegesen a foglalkozásuk szerint csökkenően, másodsorban a nevük szerint növekvően legyenek rendezve.

### Megoldás

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

UPDATE dolgozó
SET sal = ROUND(sal*1.1)
WHERE empno IN (SELECT mgr FROM dolgozó);

SET numwidth 6
SELECT *
FROM dolgozó
WHERE empno IN (SELECT mgr FROM dolgozó)
ORDER BY job DESC, ename ASC;
SET numwidth 10
```

### Eredmény

A tábla eldobva.  
A tábla létrejött.

6 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5500		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	3135		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2695		10
7566	JONES	MANAGER	7839	81-ÁPR-02	3273		20
7902	FORD	ANALYST	7566	81-DEC-03	3300		20
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3300		20

6 sor kijelölve.

## 5.15. feladat

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos dolgozó nevű táblát, majd megnöveli a Chicago-i és a New York-i dolgozók havi fizetését a kereskedők jutalékának átlagával. A megoldáshoz használjon nézettáblát. Listázza ki a módosított táblát.

### Megoldás

```

DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

SELECT empno AS azonosító,
       ename,
       sal,
       loc
  FROM dolgozó,
       dept
 WHERE dolgozó.deptno = dept.deptno AND
       UPPER(loc) IN ('CHICAGO','NEW YORK');

SELECT 'SALESMAN'           AS munkakör,
       ROUND(AVG(comm))   AS átlag
  FROM dolgozó
 WHERE UPPER(job) = 'SALESMAN';

UPDATE dolgozó
   SET sal = sal + (SELECT ROUND(AVG(comm))
                      FROM dolgozó
                     WHERE UPPER(job) = 'SALESMAN')
 WHERE dolgozó.empno IN
       (SELECT empno
          FROM dolgozó,
               dept
         WHERE dolgozó.deptno = dept.deptno AND
               UPPER(loc) IN ('CHICAGO','NEW YORK'));

SET numwidth 6
SELECT * FROM dolgozó;
SET numwidth 10

```

### Eredmény

A tábla eldobva.  
A tábla létrejött.

AZONOSÍTÓ ENAME	SAL LOC
7839 KING	5000 NEW YORK
7698 BLAKE	2850 CHICAGO
7782 CLARK	2450 NEW YORK
7654 MARTIN	1250 CHICAGO
7499 ALLEN	1600 CHICAGO
7844 TURNER	1500 CHICAGO
7900 JAMES	950 CHICAGO
7521 WARD	1250 CHICAGO
7934 MILLER	1300 NEW YORK

9 sor kijelölve.

MUNKAKÖR	ÁTLAG
----------	-------

SALESMAN	550
----------	-----

9 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	2150	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1800	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1800	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	3400		30
7782	CLARK	MANAGER	7839	81-JÚN-09	3000		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5550		10
7844	TURNER	SALESMAN	7698	81-SZE-08	2050	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	1500		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1850		10

14 sor kijelölve.

## 5.16. feladat

Írjon szkript programot, amely új adattáblát hoz létre az emp és a dept táblákból dolgozó és részleg néven. (E táblák létrehozása előtt győződjön meg róla, hogy ezeken a neveken nem létezik-e már másik objektum, és ha igen, akkor törölje azt.) Lássa el e táblákat az alábbi megszorításokkal:

- Legyen az empno egyedi attribútum a dolgozó táblában.
- Legyen a deptno elsődleges kulcs a részleg táblában, idegen kulcs a dolgozó táblában oly módon, hogy egy részleg törlése esetén törlődjenek e részleg dolgozói is. E megszorítás helyes működésének ellenőrzéseként töröljön egy részleget, listázza ki ennek eredményét, majd vonja vissza a törlést, és ellenőrizze a visszavonás hatását.
- Egy új dolgozó csak a szkript programban megadott, 700 és 7000 USD értéktartománybeli fizetést kaphasson. E megszorítás helyes működésének ellenőrzéseként kíséreljen meg egy megfelelő és egy hibás rekordot felvinni. Listázással ellenőrizze ezeknek a műveletek a végrehajtását.

A fentiek után állítsa vissza, majd véglegesítse az áadtáblákat.

### Megoldás

#### 1. lépés (A user\_catalog lekérdezése)

```
SELECT * FROM user_catalog;
```

### Eredmény

TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
DOLGOZÓ	VIEW
EMP	TABLE
SALGRADE	TABLE

#### 2. lépés (A létrehozandó nevű objektum törlése)

```
DROP VIEW dolgozó;
```

```
SELECT * FROM user_catalog;
```

### Eredmény

A nézet eldobva.

TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
EMP	TABLE
SALGRADE	TABLE

**3. lépés (A feladatot megoldó szkript program elkészítése)**

```
-- Munkatáblák törlése
DROP TABLE dolgozó;
DROP TABLE részleg;

-- Munkatáblák létrehozása
CREATE TABLE dolgozó
AS SELECT * FROM emp;

CREATE TABLE részleg
AS SELECT * FROM dept;

SET numwidth 6
SELECT * FROM dolgozó;
SELECT * FROM részleg;

-- Egyedi kulcs megszorítás definiálása
ALTER TABLE dolgozó
ADD CONSTRAINT kulcs1
    UNIQUE (empno);

-- Elsődleges kulcs megszorítás definiálása
ALTER TABLE részleg
ADD CONSTRAINT kulcs2
    PRIMARY KEY (deptno);

-- Idegen kulcs megszorítás definiálása
ALTER TABLE dolgozó
ADD CONSTRAINT kulcs3
    FOREIGN KEY (deptno) REFERENCES részleg (deptno)
        ON DELETE CASCADE;

-- Idegen kulcs megszorítás ellenőrzése törléssel
DELETE FROM részleg
WHERE deptno = 30;

SELECT * FROM dolgozó;
SELECT * FROM részleg;

-- Visszaállítás
ROLLBACK;

SELECT * FROM dolgozó;
SELECT * FROM részleg;
```

```
-- Fizetési korlátok beállítása
DEFINE minfiz = 700
DEFINE maxfiz = 7000

-- Check megszorítás definiálása
ALTER TABLE dolgozó
  ADD CONSTRAINT fizetési
    CHECK (sal BETWEEN &minfiz AND &maxfiz);

-- Check megszorítás ellenőrzése adatbevitellel
INSERT INTO dolgozó
  VALUES(8888, 'Morgó', 'törpe', 7839, '04-OKT-10', 350, NULL, 30);

-- Check megszorítás ellenőrzése adatbevitellel
INSERT INTO dolgozó
  VALUES(8889, 'Tudor', 'törpe', 7839, '04-OKT-10', 3350, NULL, 30);

SELECT * FROM dolgozó;

-- Visszaállítás
ROLLBACK;
-- Végegesítés
COMMIT;

SELECT constraint_name AS megszorításnév,
       constraint_type AS Típus
  FROM user_constraints
 WHERE table_name = 'DOLGOZÓ';

SELECT constraint_name AS megszorításnév,
       constraint_type AS Típus
  FROM user_constraints
 WHERE table_name = 'RÉSZLEG';
SELECT * FROM dolgozó;
SET numwidth 10
```

**Eredmény**

```
DROP TABLE dolgozó
*
Hiba a(z) 1. sorban:
ORA-00942: a tábla vagy a nézet nem létezik

DROP TABLE részleg
*
```

Hiba a(z) 1. sorban:

ORA-00942: a tábla vagy a nézet nem létezik

A tábla létrejött.

A tábla létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

A tábla módosítva.

A tábla módosítva.

A tábla módosítva.

1 sor törlése.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20

7839	KING	PRESIDENT	81-NOV-17	5000	10
7876	ADAMS	CLERK	7788 87-MÁJ-23	1100	20
7902	FORD	ANALYST	7566 81-DEC-03	3000	20
7934	MILLER	CLERK	7782 82-JAN-23	1300	10

8 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
40	OPERATIONS	BOSTON

A visszaállítás befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

régi 3: CHECK (sal BETWEEN &minfiz AND &maxfiz)  
új 3: CHECK (sal BETWEEN 700 AND 7000)

A tábla módosítva.

INSERT INTO dolgozó

\*

Hiba a(z) 1. sorban:

ORA-02290: ellenőrző megszorítás (SCOTT.FIZETÉS1) megsértése

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
8889	Tudor	törpe	7839	04-OKT-10	3350		30

15 sor kijelölve.

A visszaállítás befejeződött.

A jóváhagyás befejeződött.

MEGSZORÍTÁSNÉV	T
KULCS1	U
KULCS3	R
FIZETÉS1	C

MEGSZORÍTÁSNÉV	T
KULCS2	P

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7788	SCOTT	ANALYST	7566	87-ÁPR-19	3000		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7876	ADAMS	CLERK	7788	87-MÁJ-23	1100		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

## 6. FEJEZET

# Nézettáblák, felső-N analízis, ROWNUM

## Feladatok és megoldások

A feladatokat minden szkript állományba írjuk, és abból futtassuk. Egy tábla vagy nézettábla létrehozása előtt minden ellenőrizzük, hogy létezik-e az SQL\*Plus-környezetünkben ezen a néven valamelyen objektum, és ha igen, akkor adjuk ki a rá vonatkozó törlési utasítást.

### 6.1. feladat

Hozza létre az emp és dept adattáblákból a dolgozó és részleg nevű adattáblákat, és ezekre hivatkozóan a Dolgozónézet és Részlegnézet nevű nézettáblákat, majd listázza ki ez utóbbitakat.

#### Megoldás

##### 1. lépés (A user\_catalog lekérdezése)

```
SELECT * FROM user_catalog;
```

#### Eredmény

TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
DOLGOZÓ	VIEW
EMP	TABLE
RÉSZLEG	VIEW
SALGRADE	TABLE

6 sor kijelölve.

##### 2. lépés (A létrehozandó nevű objektum törlése)

```
DROP VIEW dolgozó;  
DROP VIEW részleg;  
SELECT * FROM user_catalog;
```

**Eredmény**

A nézet eldobva.

A nézet eldobva.

TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
EMP	TABLE
SALGRADE	TABLE

**3. lépés (A feladatot megoldó szkript program elkészítése)**

```
CREATE TABLE dolgozó
    AS SELECT * FROM emp;
CREATE TABLE részleg
    AS SELECT * FROM dept;
CREATE OR REPLACE VIEW Dolgozónézet
    AS SELECT * FROM dolgozó;
CREATE OR REPLACE VIEW Részlegnézet
    AS SELECT * FROM részleg;
```

**Eredmény**

A tábla létrejött.

A tábla létrejött.

A nézet létrejött.

A nézet létrejött.

**4. lépés (Ellenőrzés, nézettáblák törlése)**

```
SET numwidth 5
SELECT * FROM Dolgozónézet;
SELECT * FROM Részlegnézet;
SET numwidth 10
```

**Eredmény**

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30

7521 WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902 FORD	ANALYST	7566	81-DEC-03	3000		20
7369 SMITH	CLERK	7902	80-DEC-17	800		20
7788 SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876 ADAMS	CLERK	7788	83-JÁN-12	1100		20
7934 MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

## 6.2. feladat

Hozza létre az emp tábla 500 és 1500 USD közötti jövedelemmel rendelkező dolgozóinak nevétől, munkaköréből és jövedelméből a SzegényDolgozók nézettáblát, jövedelem szerint növekvően rendezve listázza ki, majd törölje.

### Megoldás

```

CREATE OR REPLACE VIEW SzegényDolgozók
  (Név, Munkakör, Jövedelem)
AS
  SELECT ename,
         job,
         sal+NVL(comm,0) AS jöv
    FROM emp
   WHERE sal+NVL(comm,0) BETWEEN 500 AND 1500
   ORDER BY jöv;
SELECT * FROM SzegényDolgozók;
DROP VIEW SzegényDolgozók;
  
```

### Eredmény

A nézet létrejött.

NÉV	MUNKAKÖR	JÖVEDELEM
SMITH	CLERK	800
JAMES	CLERK	950
ADAMS	CLERK	1100

MILLER	CLERK	1300
TURNER	SALESMAN	1500

A nézet eldobva.

### 6.3. feladat

Listázza ki a négy legjobban és a négy legrosszabbul kereső dolgozó azonosítóját, nevét, foglalkozását, fizetését és telephelyét az emp és dept táblák alapján.

#### 1. megoldás (Inline nézettel)

SET feedback OFF

PROMPT A legjobban keresők:

```
SELECT el.empno          AS "Azonosító",
       el.ename           AS "Név",
       el.job             AS "Foglalkozás",
       el.sal+NVL(el.comm,0) AS "Jövedelem",
       ROWNUM            AS "Sorszám",
       loc               AS "Telephely"
  FROM dept,
       (SELECT *
        FROM emp
        ORDER BY sal+NVL(comm,0) DESC) el
 WHERE (ROWNUM <= 4)    AND
       (el.deptno = dept.deptno);
```

PROMPT A legrosszabbul keresők:

```
SELECT el.empno          AS "Azonosító",
       el.ename           AS "Név",
       el.job             AS "Foglalkozás",
       el.sal+NVL(el.comm,0) AS "Jövedelem",
       ROWNUM            AS "Sorszám",
       loc               AS "Telephely"
  FROM dept,
       (SELECT *
        FROM emp
        ORDER BY sal+NVL(comm,0) ASC) el
 WHERE (ROWNUM <= 4)    AND
       (el.deptno = dept.deptno);
SET feedback ON
```

**Eredmény**

A legjobban keresők:

Azonosító Név	Foglalkoz	Jövedelem	Sorszám	Telephely
7839 KING	PRESIDENT	5000	1	NEW YORK
7902 FORD	ANALYST	3000	2	DALLAS
7788 SCOTT	ANALYST	3000	3	DALLAS
7566 JONES	MANAGER	2975	4	DALLAS

A legrosszabbul keresők:

Azonosító Név	Foglalkoz	Jövedelem	Sorszám	Telephely
7369 SMITH	CLERK	800	1	DALLAS
7900 JAMES	CLERK	950	2	CHICAGO
7876 ADAMS	CLERK	1100	3	DALLAS
7934 MILLER	CLERK	1300	4	NEW YORK

**2. megoldás (Nézettáblákkal)**

CREATE OR REPLACE VIEW JólKeresők

AS

```

SELECT el.empno          AS "Azonosító",
       el.ename          AS "Név",
       el.job            AS "Foglalkozás",
       el.sal+NVL(el.comm,0) AS "Jövedelem",
       ROWNUM           AS "Sorszám",
       loc              AS "Telephely"
  FROM dept,
  (SELECT *
   FROM emp
   ORDER BY sal+NVL(comm,0) DESC) el
 WHERE (ROWNUM <= 4) AND
       (el.deptno = dept.deptno);

```

CREATE OR REPLACE VIEW RosszulKeresők

AS

```

SELECT el.empno          AS "Azonosító",
       el.ename          AS "Név",
       el.job            AS "Foglalkozás",
       el.sal+NVL(el.comm,0) AS "Jövedelem",
       ROWNUM           AS "Sorszám",
       loc              AS "Telephely"
  FROM dept,
  (SELECT *
   FROM emp

```

```

        ORDER BY sal+NVL(comm,0) ASC) el
WHERE (ROWNUM <= 4) AND
      (el.deptno = dept.deptno);

SET feedback OFF
PROMPT A legjobban keresők:
SELECT * FROM JólKeresők;
PROMPT A legrosszabbul keresők:
SELECT * FROM RosszulKeresők;
DROP VIEW JólKeresők;
DROP VIEW RosszulKeresők;
SET feedback ON

```

### Eredmény

A nézet létrejött.

A nézet létrejött.

A legjobban keresők:

Azonosító	Név	Foglalkoz	Jövedelem	Sorszám	Telephely
7839	KING	PRESIDENT	5000	1	NEW YORK
7902	FORD	ANALYST	3000	2	DALLAS
7788	SCOTT	ANALYST	3000	3	DALLAS
7566	JONES	MANAGER	2975	4	DALLAS

A legrosszabbul keresők:

Azonosító	Név	Foglalkoz	Jövedelem	Sorszám	Telephely
7369	SMITH	CLERK	800	1	DALLAS
7900	JAMES	CLERK	950	2	CHICAGO
7876	ADAMS	CLERK	1100	3	DALLAS
7934	MILLER	CLERK	1300	4	NEW YORK

## 6.4. feladat

Futtassa le a 6.1. feladatban létrehozott szkript programot, és ellenőrizze a megfelelő adatszótárbeli nézetekből (user\_objects, user\_catalog, user\_views), hogy létezik-e valóban a dolgozó és a részleg adattábla, valamint a DolgozóNézet és a RészlegNézet nézettábla.

### 1. megoldás (user\_objects)

```

SET linesize 200
SET pagesize 60
SELECT * FROM user_objects;

```

**Eredmény**

OBJECT\_NAME

OBJECT_TYPE	CREATED	LAST_DDL_TIMESTAMP	STATUS	T	G	S
BONUS						
TABLE	04-OKT-31	04-OKT-31 2004-10-31:17:24:08	VALID	N	N	N
DEPT						
TABLE	04-OKT-31	04-OKT-31 2004-10-31:17:24:05	VALID	N	N	N
DEPT_PRIMARY_KEY						
INDEX	04-OKT-31	04-OKT-31 2004-10-31:17:24:05	VALID	N	N	N
DOLGOZÓ						
TABLE	04-NOV-27	04-NOV-27 2004-11-27:13:34:44	VALID	N	N	N
DOLGOZÓNÉZET						
VIEW	04-NOV-27	04-NOV-27 2004-11-27:13:55:11	VALID	N	N	N
EMP						
TABLE	04-NOV-26	04-NOV-26 2004-11-26:08:19:57	VALID	N	N	N
EMP_PRIMARY_KEY						
INDEX	04-NOV-26	04-NOV-26 2004-11-26:08:19:57	VALID	N	N	N
RÉSZLEG						
TABLE	04-NOV-27	04-NOV-27 2004-11-27:13:34:45	VALID	N	N	N
RÉSZLEGNÉZET						
VIEW	04-NOV-27	04-NOV-27 2004-11-27:13:55:12	VALID	N	N	N
SALGRADE						
TABLE	04-OKT-31	04-OKT-31 2004-10-31:17:24:08	VALID	N	N	N
TESZT						
FUNCTION	04-JÚN-21	04-JÚN-21 2004-06-21:18:58:43	INVALID	N	N	N
TIIT						
FUNCTION	04-MÁJ-06	04-MÁJ-06 2004-05-06:20:20:51	INVALID	N	N	N

12 sor kijelölve.

**2. megoldás (user\_catalog)**

```
SELECT * FROM user_catalog;
```

**Eredmény**

TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
DOLGOZÓ	TABLE
DOLGOZÓNÉZET	VIEW
EMP	TABLE
RÉSZLEG	TABLE
RÉSZLEGNÉZET	VIEW
SALGRADE	TABLE

8 sor kijelölve.

**3. megoldás (user\_views)**

```
SET linesize 68
SELECT SUBSTR(view_name,1,12) AS "NézetNév",
       LPAD(text_length,5)   AS "Hossz",
       text                  AS "Szöveg"
FROM user_views;
SET linesize 400
```

**Eredmény**

NézetNév	Hossz
Szöveg	

DOLGOZÓNÉZET	80
--------------	----

```
SELECT "EMPNO", "ENAME", "JOB", "MGR", "HIREDATE", "SAL", "COMM", "DEPTNO"
FROM dolgozó
```

RÉSZLEGNÉZET	42
--------------	----

```
SELECT "DEPTNO", "DNAME", "LOC" FROM részleg
```

**6.5. feladat (Adatbevitel nézettáblán keresztül)**

Hozza létre (ha még nem létezik) a dolgozó táblát az emp táblából, és végezzen egy nézettáblán keresztül történő, úgynevezett ellenőrzött adatbevitelt.

A nézettábla neve legyen ūrlap, és a sémaja (oszlopneveinek halmaza) a dolgozó tábla minden oszlopnevét tartalmazza a jutalék (comm) kivételével.

A kiválasztási feltételek, és egyben ellenőrzési feltételek legyenek a következők:

- csak hivatalnok (clerk) legyen felvihető,
- a főnökkódja csak a 7788 és a 7902 valamelyike lehessen,
- fizetése 800 és 1200 USD között legyen,
- a részlegazonosítója vagy a 10, vagy a 20 valamelyike lehessen.

Ezekután bővítsse a dolgozó adattáblát az űrlap nézettáblán keresztül az ABSOLON nevű dolgozóval (a többi érték az engedélyezett tartományon belül legyen tetszőleges). Végül törölje az űrlap nézettáblát.

### **Előkészület a megoldáshoz**

#### **E1. lépés (Az SQL\*Plus-környezet vizsgálata)**

```
SELECT * FROM user_catalog;
```

#### **Eredmény**

TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
DOLGOZÓ	TABLE
DOLGOZÓNÉZET	VIEW
EMP	TABLE
RÉSZLEG	TABLE
RÉSZLEGNÉZET	VIEW
SALGRADE	TABLE

8 sor kijelölve.

#### **E2. lépés (A félösleges objektumok törlése)**

```
DROP VIEW Dolgozónézet;
```

```
DROP VIEW Részlegnézet;
```

#### **Eredmény**

A nézet eldobva.

A nézet eldobva.

#### **Megoldás**

#### **1. lépés (Mivel már létezett, így nem hozzuk létre a dolgozó adattáblát)**

----

#### **2. lépés (Az űrlap nézettábla létrehozása)**

```
CREATE OR REPLACE VIEW űrlap
```

```
AS
```

```
SELECT empno, ename, job, mgr, hiredate, sal, deptno
      FROM dolgozó
```

```

WHERE
  UPPER(job) = 'CLERK'      AND
  mgr IN (7788, 7902)       AND
  sal BETWEEN 800 AND 1200   AND
  deptno IN (10, 20)
WITH CHECK OPTION;

SELECT *
  FROM űrlap
 ORDER BY ename;

```

### Eredmény

A nézet létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7876	ADAMS	CLERK	7788	83-JAN-12	1100	20
7369	SMITH	CLERK	7902	80-DEC-17	800	20

### 3. lépés (Adatfelviteli kísérletek)

PROMPT Hibás munkakör megadás:

```

INSERT INTO űrlap
VALUES(7777,'ABSOLON','SALESMAN',7788,sysdate,900,10);

```

PROMPT Hibás főnökkód megadás:

```

INSERT INTO űrlap
VALUES(7777,'ABSOLON','CLERK',7698,sysdate,900,10);

```

PROMPT Hibás fizetés megadás:

```

INSERT INTO űrlap
VALUES(7777,'ABSOLON','CLERK',7788,sysdate,500,10);

```

PROMPT Hibás részlegazonosító megadás:

```

INSERT INTO űrlap
VALUES(7777,'ABSOLON','CLERK',7788,sysdate,900,30);

```

PROMPT Tiltott adatmegadás a comm adatmezőnek:

```

INSERT INTO űrlap
VALUES(7777,'ABSOLON','CLERK',7788,sysdate,900,NULL,10);

```

PROMPT Helyes adatmegadás:

```

INSERT INTO űrlap
VALUES(7777,'ABSOLON','CLERK',7788,sysdate,900,10);

```

PROMPT Ellenőrző lekérdezés a "dolgozó" táblából:  
SELECT \* FROM dolgozó;

### Eredmény

Hibás munkakör megadás:

INSERT INTO űrlap  
\*

Hiba a(z) 1. sorban:

ORA-01402: a WITH CHECK OPTION-nal létrehozott nézet WHERE feltétele  
nem teljesül

Hibás főnökkód megadás:

INSERT INTO űrlap  
\*

Hiba a(z) 1. sorban:

ORA-01402: a WITH CHECK OPTION-nal létrehozott nézet WHERE feltétele  
nem teljesül

Hibás fizetés megadás:

INSERT INTO űrlap  
\*

Hiba a(z) 1. sorban:

ORA-01402: a WITH CHECK OPTION-nal létrehozott nézet WHERE feltétele  
nem teljesül

Hibás részlegazonosító megadás:

INSERT INTO űrlap  
\*

Hiba a(z) 1. sorban:

ORA-01402: a WITH CHECK OPTION-nal létrehozott nézet WHERE feltétele  
nem teljesül

Tiltott adatmegadás a comm adatmezőnek:

INSERT INTO űrlap  
\*

Hiba a(z) 1. sorban:

ORA-00913: túl sok az érték

Helyes adatmegadás:

1 sor létrejött.

Ellenőrző lekérdezés a "dolgozó" táblából:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
7777	ABSOLON	CLERK	7788	04-NOV-26	900		10

15 sor kijelölve.

#### 4. lépés (Az űrlap nézettábla törlése)

DROP VIEW űrlap;

#### Eredmény

A nézet eldobva.

### 6.6. feladat

Hozza létre a négy legrosszabbul kereső dolgozó azonosítóját, nevét, foglalkozását, fizetését és telephelyét tartalmazó nézettáblát az emp és a dolgozó táblára vonatkozóan (lásd 6.3. feladat), majd listázza ki őket. Indokolja az eredményt.

#### Előkészületek a megoldáshoz

-- Munkatáblák törlése

DROP TABLE dolgozó;

DROP TABLE részleg;

-- Munkatáblák létrehozása

CREATE TABLE dolgozó

AS SELECT \* FROM emp;

CREATE TABLE részleg

AS SELECT \* FROM dept;

### **Megoldás**

```
CREATE OR REPLACE VIEW RosszulKeresők1
AS
  SELECT e1.empno          AS "Azonosító",
         e1.ename           AS "Név",
         e1.job              AS "Foglalkozás",
         e1.sal+NVL(e1.comm,0) AS "Jövedelem",
         ROWNUM             AS "Sorszám",
         loc                AS "Telephely"
    FROM dept,
         (SELECT *
            FROM emp
           ORDER BY sal+NVL(comm,0) ASC) e1
   WHERE (ROWNUM <= 4) AND
        (e1.deptno = dept.deptno);
```

```
CREATE OR REPLACE VIEW RosszulKeresők2
AS
  SELECT e1.empno          AS "Azonosító",
         e1.ename           AS "Név",
         e1.job              AS "Foglalkozás",
         e1.sal+NVL(e1.comm,0) AS "Jövedelem",
         ROWNUM             AS "Sorszám",
         loc                AS "Telephely"
    FROM dept,
         (SELECT *
            FROM dolgozó
           ORDER BY sal+NVL(comm,0) ASC) e1
   WHERE (ROWNUM <= 4) AND
        (e1.deptno = dept.deptno);
```

PROMPT A legrosszabbul keresők az emp táblában:

```
SELECT * FROM RosszulKeresők1;
```

PROMPT A legrosszabbul keresők a dolgozó táblában:

```
SELECT * FROM RosszulKeresők2;
```

### **Eredmény**

A nézet létrejött.

A nézet létrejött.

A legrosszabbul keresők az emp táblában:

Azonosító	Név	Foglalkoz	Jövedelem	Sorszám	Telephely
7369	SMITH	CLERK	800	1	DALLAS

7900 JAMES	CLERK	950	2 CHICAGO
7876 ADAMS	CLERK	1100	3 DALLAS
7934 MILLER	CLERK	1300	4 NEW YORK

A legrosszabbul keresők a dolgozó táblában:

Azonosító Név	Foglalkoz	Jövedelem	Sorszám	Telephely
7369 SMITH	CLERK	800	1	DALLAS
7777 ABSOLON	CLERK	900	2	NEW YORK
7900 JAMES	CLERK	950	3	CHICAGO
7876 ADAMS	CLERK	1100	4	DALLAS

### **Indoklás**

Mivel a 6.5. feladatban az űrlap nézettáblán keresztül kibővítettük a dolgozó táblát az 1100 USD fizetésű Absolon-nal, így e táblában ő bekerült a legrosszabbul keresők közé. E bővítés azonban természetesen nem érintette az emp táblát, ezért abban a 6.3. feladatbeli állapot maradt.

## **6.7. feladat**

Hozza létre úgy a 6.5. feladatbeli űrlap nézettáblát, hogy rajta keresztül csak lekérdezni lehessen a dolgozó tábla adatait, de módosítani vagy bővíteni már ne.

### **Előkészületek a megoldáshoz**

-- Munkatáblák törlése

DROP TABLE dolgozó;

DROP TABLE részleg;

-- Munkatáblák létrehozása

CREATE TABLE dolgozó

AS SELECT \* FROM emp;

CREATE TABLE részleg

AS SELECT \* FROM dept;

### **Megoldás**

CREATE OR REPLACE VIEW űrlap

AS

SELECT empno, ename, job, mgr, hiredate, sal, deptno

FROM dolgozó

WHERE

UPPER(job) = 'CLERK' AND

mgr IN (7788, 7902) AND

sal BETWEEN 800 AND 1200 AND

deptno IN (10, 20)

WITH READ ONLY;

PROMPT Lekérdezés az "Ürlap" nézettáblán keresztül:

```
SELECT *
  FROM Ürlap
 ORDER BY ename;
```

PROMPT Helyes adatmegadás:

```
INSERT INTO Ürlap
VALUES(7778, 'ABSOLON1', 'CLERK', 7788, sysdate, 950, 10);
```

```
DROP VIEW Ürlap;
```

### Eredmény

A nézet létrejött.

Lekérdezés az "Ürlap" nézettáblán keresztül:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7777	ABSONLON	CLERK	7788	04-NOV-27	900	10
7876	ADAMS	CLERK	7788	83-JAN-12	1100	20
7369	SMITH	CLERK	7902	80-DEC-17	800	20

3 sor kijelölve.

Helyes adatmegadás:

```
INSERT INTO Ürlap
```

\*

Hiba a(z) 1. sorban:

ORA-01733: virtuális (képzett) oszlop használata itt nem megengedett

A nézet eldobva.

## 6.8. feladat

Listázza ki a dolgozó tábla két legmagasabb átlagjövedelmű olyan munkaköri csoportjának nevét, amelyekben a dolgozóknak van főnökük.

### Előkészületek a megoldáshoz

```
-- Munkatáblák törlése
DROP TABLE dolgozó;
DROP TABLE részleg;
```

```
-- Munkatáblák létrehozása
CREATE TABLE dolgozó
```

```
AS SELECT * FROM emp;
CREATE TABLE részleg
AS SELECT * FROM dept;
```

**Megoldás**

```
CREATE OR REPLACE VIEW ÁtlJövNézet
AS
SELECT ROUND(AVG(sal+NVL(comm,0))) AS ÁtlJöv,
       job                               AS munkakör
  FROM dolgozó
 WHERE mgr IS NOT NULL
 GROUP BY job
 ORDER BY ÁtlJöv DESC;

SELECT ÁtlJövNézet.*,
       ROWNUM AS sorrend
  FROM ÁtlJövNézet
 WHERE ROWNUM <= 2;

DROP VIEW ÁtlJövNézet;
```

**Eredmény**

A nézet létrejött.

ÁTLJÖV MUNKAKÖR	SORREND
3000 ANALYST	1
2758 MANAGER	2

A nézet eldobva.

**6.9. feladat**

Listázza ki a dolgozó tábla azon dolgozónak nevét, munkakörét és fizetését, akik a legmagasabb átlagjövedelmű olyan munkaköri csoportban dolgoznak, amelyben minden dolgozónak van főnöke.

**Előkészületek a megoldáshoz**

- Munkatáblák törlése

```
DROP TABLE dolgozó;
DROP TABLE részleg;
```

- Munkatáblák létrehozása

```
CREATE TABLE dolgozó
```

```

    AS SELECT * FROM emp;
CREATE TABLE részleg
    AS SELECT * FROM dept;
```

### 1. megoldás (Nézettáblával és MAX függvényel)

```

CREATE OR REPLACE VIEW ÁtlJövNézet
AS
SELECT ROUND(AVG(sal+NVL(comm,0))) AS ÁtlJöv,
       job                      AS munkakör
  FROM dolgozó
 WHERE mgr IS NOT NULL
 GROUP BY job
 ORDER BY ÁtlJöv DESC;

SET numwidth 5
SELECT ename, job, sal
  FROM dolgozó,
       ÁtlJövNézet,
       (SELECT MAX(ÁtlJöv) AS MaxÁtlJöv
        FROM ÁtlJövNézet)
 WHERE ÁtlJöv = MaxÁtlJöv AND
       job = munkakör;
SET numwidth 10
DROP VIEW ÁtlJövNézet;
```

### 2. megoldás (Nézettáblával és felső-N analízissel)

```

CREATE OR REPLACE VIEW ÁtlJövNézet
AS
SELECT ROUND(AVG(sal+NVL(comm,0))) AS ÁtlJöv,
       job                      AS munkakör
  FROM dolgozó
 WHERE mgr IS NOT NULL
 GROUP BY job
 ORDER BY ÁtlJöv DESC;

SET numwidth 5
SELECT ename, job, sal
  FROM dolgozó,
       (SELECT munkakör,
              ROWNUM
        FROM ÁtlJövNézet
       WHERE ROWNUM <= 1)
 WHERE job = munkakör;
SET numwidth 10
DROP VIEW ÁtlJövNézet;
```

**Eredmény (mindkét megoldás esetén)**

A nézet létrejött.

ENAME	JOB	SAL
FORD	ANALYST	3000
SCOTT	ANALYST	3000

A nézet eldobva.

**6.10. feladat**

Írjon egy olyan nézettáblát létrehozó és listázó szkript programot a dolgozó táblára vonatkozóan, amely azokat a Chicago-i dolgozókat tartalmazza, akiknek főnöke 1980 és 1981 között lépett be. A nézettábla tartalmát elsődlegesen a munkakör, másodlagosan a név szerint rendezve listázza. Ennek során kerülje a munkakör ismétlődéseit.

**Előkészületek a megoldáshoz**

-- Munkatáblák törlése

DROP TABLE dolgozó;

DROP TABLE részleg;

-- Munkatáblák létrehozása

CREATE TABLE dolgozó

AS SELECT \* FROM emp;

CREATE TABLE részleg

AS SELECT \* FROM dept;

**Megoldás**

-- Nézettábla létrehozása:

CREATE OR REPLACE VIEW RégiFőnökökDolgozói

AS

SELECT D.\*,

LPAD(TO\_CHAR(F.hiredate,'YYYY'),7) AS FőnökÉv

FROM dolgozó D,

részleg R,

dolgozó F

WHERE D.mgr = F.empno AND

D.deptno = R.deptno AND

UPPER(R.loc) = 'CHICAGO' AND

TO\_CHAR(F.hiredate,'YYYY') BETWEEN '1980' AND '1981';

-- Lekérdezés

BREAK ON job

```

SET numwidth 5
SELECT *
  FROM RégiFőnökökDolgozói
 ORDER BY job, ename;
SET numwidth 10
CLEAR BREAKS

DROP VIEW RégiFőnökökDolgozói;

```

### Eredmény

A tábla eldobva.  
A tábla eldobva.  
A tábla létrejött.  
A tábla létrejött.  
A nézet létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	FŐNÖKÉV
7900	JAMES	CLERK	7698	81-DEC-03	950		30	1981
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	1981
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	1981
7654	MARTIN		7698	81-SZE-28	1250	1400	30	1981
7844	TURNER		7698	81-SZE-08	1500	0	30	1981
7521	WARD		7698	81-FEB-22	1250	500	30	1981

6 sor kijelölve.

A nézet eldobva.

## 6.11. feladat

Készítsen az emp tábla alapján egy olyan nézettáblát, amely meghatározza az egyes fizetési osztályokban a dolgozók számát, valamint a legalacsonyabb és legmagasabb fizetéseket, ahol a fizetési osztály 1000 USD alatt 1, 2000 USD alatt 2 stb. Ezután listázza ki ezt a nézettáblát a fizetési osztályok szerint csökkenően rendezve.

### 1. megoldás (Matematikai kifejezéssel)

```

CREATE OR REPLACE VIEW FizetésOsztály
AS
  SELECT COUNT(empno)      AS létszám,
         MIN(sal)          AS legkisebb,
         MAX(sal)          AS legnagyobb,
         TRUNC(sal/1000)+1 AS osztály
    FROM emp
   GROUP BY TRUNC(sal/1000)+1;

```

```
SELECT *
  FROM FizetésOsztály
 ORDER BY osztály DESC;
```

```
DROP VIEW FizetésOsztály;
```

## 2. megoldás (Allekérdezésben CASE kifejezéssel)

```
CREATE OR REPLACE VIEW FizetésOsztály
AS
  SELECT COUNT(osztály) AS létszám,
         MIN(sal)      AS legkisebb,
         MAX(sal)      AS legnagyobb,
         osztály
    FROM emp,
         (SELECT empno,
                 CASE
                   WHEN sal BETWEEN 0 AND 999 THEN 1
                   WHEN sal BETWEEN 1000 AND 1999 THEN 2
                   WHEN sal BETWEEN 2000 AND 2999 THEN 3
                   WHEN sal BETWEEN 3000 AND 3999 THEN 4
                   WHEN sal BETWEEN 4000 AND 4999 THEN 5
                   ELSE 6
                 END AS osztály
      FROM emp) Osztályozás
   WHERE emp.empno = Osztályozás.empno
 GROUP BY osztály;
```

```
SELECT *
  FROM FizetésOsztály
 ORDER BY osztály DESC;
```

```
DROP VIEW FizetésOsztály;
```

## Eredmény (mindkét megoldás esetén)

A nézet létrejött.

LÉTSZÁM	LEGKISEBB	LEGNAGYOBBA	OSZTÁLY
1	5000	5000	6
2	3000	3000	4
3	2450	2975	3
6	1100	1600	2
2	800	950	1

A nézet eldobva.

### 3. megoldás (Osztályozó tábla használatával)

#### 3.1. lépés (Az Osztályozó tábla létrehozása és feltöltése)

```
DROP TABLE Osztályozó;
```

```
CREATE TABLE Osztályozó
  (alsó      NUMBER(4),
   felső     NUMBER(4),
   osztály   Number(2));
```

```
INSERT INTO Osztályozó
  VALUES( 0, 999,1);
```

```
INSERT INTO Osztályozó
  VALUES(1000,1999,2);
```

```
INSERT INTO Osztályozó
  VALUES(2000,2999,3);
```

```
INSERT INTO Osztályozó
  VALUES(3000,3999,4);
```

```
INSERT INTO Osztályozó
  VALUES(4000,4999,5);
```

```
INSERT INTO Osztályozó
  VALUES(5000,5999,6);
```

```
SELECT *
  FROM Osztályozó
 ORDER BY osztály DESC;
```

#### Eredmény

ALÓSÓ	FELSŐ	OSZTÁLY
5000	5999	6
4000	4999	5
3000	3999	4
2000	2999	3
1000	1999	2
0	999	1

6 sor kijelölve.

#### 3.2. lépés (A FizetésOsztály nézettábla létrehozása az Osztályozó tábla alapján)

```
CREATE OR REPLACE VIEW FizetésOsztály
```

```
AS
```

```
  SELECT COUNT(empno)          AS létszám,
        MIN(sal)                AS legkisebb,
        MAX(sal)                AS legnagyobb,
        osztály
```

```
FROM emp, Osztályozó
WHERE sal BETWEEN alsó AND felső
GROUP BY osztály;
```

```
SELECT *
FROM FizetésOsztály
ORDER BY osztály DESC;

DROP TABLE Osztályozó;
DROP VIEW FizetésOsztály;
```

### Eredmény

A nézet létrejött.

LÉTSZÁM	LEGKISEBB	LEGNAGYOBBA	OSZTÁLY
1	5000	5000	6
2	3000	3000	4
3	2450	2975	3
6	1100	1600	2
2	800	950	1

A tábla eldobva.

A nézet eldobva.



### Megjegyzés

Bár az 1. megoldás a legegyszerűbb, mégis a 2., illetve a 3. a használhatóbb, hiszen azokban tetszőleges fizetési tartományok jelölhetők ki (azaz olyanok is, melyeket nem tudnánk matematikai kifejezéssel leírni). Azt, hogy létrehozzunk-e egy fizikai adattáblát egy osztályozási feladatra, kizárolag gyakorlati szempontok döntik el (például hogy milyen gyakran használják, fontos-e, hogy többen ugyanazt a táblát használják, csak megfelelő jogosítványokkal rendelkező személyek tartják-e karban a táblát stb.).

## 6.12. feladat

Készítsen az emp tábla alapján egy Dolgozó1 nézettáblát, amelynek oszlopai a dolgozók neve, munkaköre, fizetése és fizetési osztályának sorszáma, és amely a FizetésOsztály nézettábla (lásd előző feladat) segítségével besorolja a dolgozókat a fizetésük szerint. Listázza ki ezt a nézettáblát elsődlegesen a fizetési osztály, másodlagosan a dolgozók neve szerint rendezve.

### Megoldás

```
CREATE OR REPLACE VIEW FizetésOsztály
AS
SELECT COUNT(osztály) AS létszám,
       MIN(sal)      AS legkisebb,
```

```

MAX(sal)          AS legnagyobb,
osztály
FROM emp,
(SELECT empno,
CASE
WHEN sal BETWEEN 0 AND 999 THEN 1
WHEN sal BETWEEN 1000 AND 1999 THEN 2
WHEN sal BETWEEN 2000 AND 2999 THEN 3
WHEN sal BETWEEN 3000 AND 3999 THEN 4
WHEN sal BETWEEN 4000 AND 4999 THEN 5
ELSE 6
END AS osztály
FROM emp) Osztályozás
WHERE emp.empno = Osztályozás.empno
GROUP BY osztály;

```

```

CREATE OR REPLACE VIEW Dolgozó1
AS
SELECT ename           AS név,
       job            AS munkakör,
       sal            AS fizetés,
       osztály
FROM emp, FizetésOsztály
WHERE sal BETWEEN FizetésOsztály.legkisebb AND
      FizetésOsztály.legnagyobb;

```

```

SELECT *
FROM Dolgozó1
ORDER BY osztály, név;

```

```

DROP VIEW Dolgozó1;
DROP VIEW FizetésOsztály;

```

### Eredmény

A nézet létrejött.

A nézet létrejött.

NÉV	MUNKAKÖR	FIZETÉS	OSZTÁLY
JAMES	CLERK	950	1
SMITH	CLERK	800	1
ADAMS	CLERK	1100	2
ALLEN	SALESMAN	1600	2
MARTIN	SALESMAN	1250	2
MILLER	CLERK	1300	2

TURNER	SALESMAN	1500	2
WARD	SALESMAN	1250	2
BLAKE	MANAGER	2850	3
CLARK	MANAGER	2450	3
JONES	MANAGER	2975	3
FORD	ANALYST	3000	4
SCOTT	ANALYST	3000	4
KING	PRESIDENT	5000	6

14 sor kijelölve.

A nézet eldobva.

A nézet eldobva.

### 6.13. feladat

Írjon szkript programot, amely a dolgozó táblából törli a legjobban fizetett Dallas-i dolgozót. A törlés sikerességéről listázással győződjön meg, majd állítsa vissza az eredeti táblatartalmat. A megoldáshoz használjon nézettáblát, melyet a feladatmegoldás után töröljön.

#### Előkészületek a megoldáshoz

-- Munkatáblák törlése

DROP TABLE dolgozó;

DROP TABLE részleg;

-- Munkatáblák létrehozása

CREATE TABLE dolgozó

AS SELECT \* FROM emp;

CREATE TABLE részleg

AS SELECT \* FROM dept;

#### Megoldás

##### 1. lépés (A legjobb fizetés Dallas-ban – nézettábla)

###### 1.1. változat

```
CREATE OR REPLACE VIEW Dolgozónézet
AS.
```

```
SELECT dolgozó.deptno AS Razonosító,
       MAX(sal)      AS MaxFizetés
  FROM dolgozó,
       részleg
 WHERE dolgozó.deptno = részleg.deptno AND
       UPPER(loc) = 'DALLAS'
 GROUP BY dolgozó.deptno;
SELECT * FROM Dolgozónézet;
```

## 1.2. változat

```
CREATE OR REPLACE VIEW Dolgozónézet
AS
  SELECT dolgozó.deptno AS Razonosító,
         MAX(sal)      AS MaxFizetés
    FROM dolgozó,
         részleg
   WHERE dolgozó.deptno = részleg.deptno
 GROUP BY dolgozó.deptno
 HAVING dolgozó.deptno = (SELECT deptno
                            FROM részleg
                           WHERE UPPER(loc) = 'DALLAS');
SELECT * FROM Dolgozónézet;
```

### Eredmény (mindkét változat esetén)

RAZONOSÍTÓ MAXFIZETÉS

20	3000



#### Megjegyzés

- A legjobb fizetéssel megegyezhet más városokban dolgozók fizetése is, ezért csak a részlegazonosítóval történő szűkítés után szabad törölni.
- Az 1.2. változat a hatékonyabb, hiszen a Dallas szerinti szűrést kisebb adattáblán végzi.

## 2. lépés (Kit, illetve kiket érint a törlés?)

Bár a feladatnak nem része, ellenőrzésképpen mégsem árt tudni, hogy kit vagy kiket érint a törlés.

```
SELECT ename, sal
  FROM dolgozó
 WHERE sal = (SELECT MaxFizetés
                  FROM Dolgozónézet);
```

### Eredmény

ENAME	SAL
FORD	3000
SCOTT	3000

## 3. lépés (A legjobb fizetésű dolgozók törlése)

```
-- Törlés
DELETE dolgozó
 WHERE empno IN (SELECT empno
```

```

        FROM dolgozó
        WHERE sal IN (SELECT MaxFizetés
                        FROM Dolgozónézet));
-- Listázás
SET numwidth 5
SELECT *
    FROM dolgozó;
SET numwidth 10

-- Visszaállítás, és nézettábla törlése
ROLLBACK;
DROP VIEW Dolgozónézet;

```

**Eredmény**

2 sor törölve.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

12 sor kijelölve.

A visszaállítás befejeződött.

A nézet eldobva.

**Megjegyzés**

Láthatóan a két 3000 USD fizetésű Dallas-i dolgozó (Ford és Scott) hiányzik.

## 6.14. feladat

Írjon szkript programot, amely a dolgozó táblában megnöveli Blake beosztottjainak fizetését az átlagfizetésük 15%-ával. Az adatmódosítás sikereségéről listázással győződjön meg, majd állítsa vissza az eredeti táblatartalmat. A megoldáshoz használjon nézettáblát, melyet a feladat-megoldás után töröljön.

### Előkészületek a megoldáshoz

```
-- Munkatáblák törlése
DROP TABLE dolgozó;
DROP TABLE részleg;

-- Munkatáblák létrehozása
CREATE TABLE dolgozó
AS SELECT * FROM emp;
CREATE TABLE részleg
AS SELECT * FROM dept;
```

### Megoldás

#### 1. lépés (A beosztottak nézettábla létrehozása)

```
CREATE OR REPLACE VIEW beosztottak
AS
SELECT D.empno,
       D.sal
  FROM dolgozó D,
       dolgozó F
 WHERE D.mgr = F.empno AND
       UPPER(F.ename) = 'BLAKE';
SELECT * FROM beosztottak;
```

### Eredmény

EMPNO	SAL
7654	1250
7499	1600
7900	950
7521	1250
7844	1500

#### 2. lépés (A dolgozó tábla módosítása a nézettábla alapján)

```
UPDATE dolgozó
SET sal = sal + (SELECT ROUND(0.15 * AVG(sal))
                  FROM beosztottak)
```

```

WHERE empno IN (SELECT empno
                  FROM beosztottak);
SELECT * FROM beosztottak;

```

**Eredmény**

5 sor módosítva.

EMPNO	SAL
7654	1447
7499	1797
7900	1147
7521	1447
7844	1697

**3. lépés (Adat-visszaállítás, nézettábla törlése)**

ROLLBACK;

DROP VIEW beosztottak;

**Eredmény**

A visszaállítás befejeződött.

A nézet eldobva.

**Megjegyzés**

Figyeljünk fel arra, hogy az UPDATE utasítás egy olyan nézettábla alapján módosítja a dolgozó tábla saját oszlopát, mely maga is a dolgozó tábla saját oszlopára hivatkozik. Ha körbehibatkozás lenne, akkor a dolgozó tábla feldolgozása során az egymást követő dolgozók fizetésnövekménye más és más volna. Ám láthatóan a fizetésnövekmény minden érintett dolgozó esetén ugyanaz. Ebből már látható, hogy az UPDATE utasítás feldolgozásakor átmeneti tárolás történik (zárolva ezzel a hivatkozott tábla adatait), és az eredeti tábla adatainak módosítása csak az UPDATE utasítás teljes feldolgozása után történik meg.

**6.15. feladat**

Írjon nézettáblát használó szkript programot, mely a felhasználó által megadott értékkel megnöveli a dolgozó táblában minden fizetését, aiknek nincs vagy nulla a jutalékuk, foglalkozási csoportjukban egynél többen vannak, és Dallas-ban vagy Chicago-ban dolgoznak. A szkript program ezután listázza ki a dolgozók tábláját elsődlegesen a fizetés szerint csökkenően, másodlagosan pedig a dolgozók neve szerint növekvő módon rendezve, majd állítsa vissza az eredeti táblatartalmat, végül törölje a létrehozott nézettáblát.

### Megoldás

```
ACCEPT növekmény PROMPT 'A fizetésemelés mértéke: ';
```

```
-- Munkatáblák törlése
```

```
DROP TABLE dolgozó;
```

```
DROP TABLE részleg;
```

```
-- Munkatáblák létrehozása
```

```
CREATE TABLE dolgozó
```

```
AS SELECT * FROM emp;
```

```
CREATE TABLE részleg
```

```
AS SELECT * FROM dept;
```

```
CREATE OR REPLACE VIEW MunkakörNézet
```

```
AS
```

```
SELECT job,
       COUNT(*) AS létszám
  FROM dolgozó
 GROUP BY job;
```

```
SELECT * FROM MunkakörNézet;
```

```
CREATE OR REPLACE VIEW Kiválasztottak
```

```
AS
```

```
SELECT empno
```

```
  FROM dolgozó      D,
```

```
        részleg      R,
```

```
        MunkakörNézet MK
```

```
 WHERE
```

```
    NVL(comm,0) = 0          AND
```

```
    MK.létszám > 1          AND
```

```
    D.job = MK.job          AND
```

```
    UPPER(loc) IN ('DALLAS','CHICAGO') AND
```

```
    D.deptno = R.deptno;
```

```
SELECT * FROM Kiválasztottak;
```

```
UPDATE dolgozó
```

```
  SET sal = sal + &növekmény
```

```
 WHERE empno IN (SELECT empno
```

```
                  FROM Kiválasztottak);
```

```
SET numwidth 5
```

```
SELECT *
```

```
  FROM dolgozó
```

```
 ORDER BY sal DESC,
```

```
           ename ASC;
```

```
SET numwidth 10
```

```
ROLLBACK;
DROP VIEW MunkakörNézet;
DROP VIEW Kiválasztottak;
```

**Eredmény**

A fizetésemelés mértéke: 1001

A tábla eldobva.  
A tábla eldobva.  
A tábla létrejött.  
A tábla létrejött.  
A nézet létrejött.

JOB	LÉTSZÁM
ANALYST	2
CLERK	4
MANAGER	3
PRESIDENT	1
SALESMAN	4

A nézet létrejött.

EMPNO
7902
7788
7369
7566
7876
7900
7844
7698

8 sor kijelölve.

régi 2: SET sal = sal + &növekmény  
új 2: SET sal = sal + 1001

8 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10

7902	FORD	ANALYST	7566	81-DEC-03	4001	20	
7788	SCOTT	ANALYST	7566	82-DEC-09	4001	20	
7566	JONES	MANAGER	7839	81-ÁPR-02	3976	20	
7698	BLAKE	MANAGER	7839	81-MÁJ-01	3851	30	
7844	TURNER	SALESMAN	7698	81-SZE-08	2501	0	30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450	10	
7876	ADAMS	CLERK	7788	83-JAN-12	2101	20	
7900	JAMES	CLERK	7698	81-DEC-03	1951	30	
7369	SMITH	CLERK	7902	80-DEC-17	1801	20	
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7934	MILLER	CLERK	7782	82-JAN-23	1300	10	
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30

14 sor kijelölve.

A visszaállítás befejeződött.

A nézet eldobva.

A nézet eldobva.

## 6.16. feladat

Írjon szkript programot, amely (eldob és) létrehoz az emp táblából egy vele azonos, dolgozó nevű táblát, és bővítsé ki azt egy Prémium nevű oszloppal. Az egyes dolgozók Prémium értéke legyen a saját telephelyük átlagjutalékának 10%-a. Használjon nézettáblát.

### Megoldás

```
DROP TABLE dolgozó;
```

```
CREATE TABLE dolgozó
AS SELECT * FROM emp;
```

```
ALTER TABLE dolgozó
ADD (Prémium NUMBER(4));
```

```
CREATE OR REPLACE VIEW PrémiumNézet
AS
SELECT deptno          AS Részleg,
       ROUND(AVG(NVL(comm,0))) AS Érték
  FROM dolgozó
 GROUP BY deptno;
```

PROMPT A Prémium nézettábla tartalma:  
SELECT \* FROM PrémiumNézet;

```
UPDATE dolgozó
```

```

SET prémium =
  (SELECT Érték
   FROM PrémiumNézet
   WHERE Részleg = deptno);

SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
DROP VIEW PrémiumNézet;

```

**Eredmény**

- A tábla eldobva.  
A tábla létrejött.  
A tábla módosítva.  
A nézet létrejött.

A Prémium nézettábla tartalma:

RÉSZLEG	ÉRTÉK
10	0
20	0
30	367

14 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	PRÉMIUM
7839	KING	PRESIDENT		81-NOV-17	5000		10	0
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	367
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10	0
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20	0
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	367
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	367
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	367
7900	JAMES	CLERK	7698	81-DEC-03	950		30	367
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30	367
7902	FORD	ANALYST	7566	81-DEC-03	3000		20	0
7369	SMITH	CLERK	7902	80-DEC-17	800		20	0
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20	0
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20	0
7934	MILLER	CLERK	7782	82-JAN-23	1300		10	0

14 sor kijelölve.

A nézet eldobva.

## 6.17. feladat

Írjon szkript programot, amely létrehoz az emp táblából egy dolgozó táblát. Növelje meg ebben 10%-kal a legkisebb átlagfizetésű részleg dolgozónak fizetését, és listázza ki e részleg nevét, az itt dolgozók nevét, valamint azok régi és új fizetését. A megoldáshoz használjon nézettáblát.

### Megoldás

#### 1. lépés (A munkatáblák törlése és létrehozása)

```
DROP TABLE dolgozó;
DROP TABLE részleg;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
CREATE TABLE részleg
AS SELECT * FROM dept;
```

### Eredmény

- A tábla eldobva.
- A tábla eldobva.
- A tábla létrejött.
- A tábla létrejött.

#### 2. lépés (A dolgozó tábla bővítése új oszloppal)

```
ALTER TABLE dolgozó
ADD újfizetés NUMBER(4);
```

### Eredmény

- A tábla módosítva.



#### Megjegyzés

Kérdezzük le a dolgozó tábla sal oszlopának adattípusát, és aszerint állítsuk be az újfizetés oszlop adattípusát.

#### 3. lépés (A dolgozó tábla sal oszlopának átnevezése)

```
ALTER TABLE dolgozó
RENAME COLUMN sal TO régifizetés;
```

### Eredmény

- A tábla módosítva.

#### 4. lépés (A részlegenkénti átlagfizetés nézettáblájának létrehozása)

```
CREATE OR REPLACE VIEW RészlegÁtlagFiz
AS
```

```

SELECT deptno,
       ROUND(AVG(régifizetés)) AS ÁtlagFiz
  FROM dolgozó
 GROUP BY deptno;
SELECT * FROM RészlegÁtlagFiz;

```

**Eredmény**

A nézet létrejött.

DEPTNO	ÁTLAGFIZ
10	2917
20	2175
30	1567

**5. lépés (A legkisebb átlagszámot megadó részleg dolgozóinak nézettáblája)**

```

CREATE OR REPLACE VIEW SzegényDolgozók
AS
SELECT empno,
       régifizetés
  FROM dolgozó,
       RészlegÁtlagFiz,
       (SELECT MIN(ÁtlagFiz) AS MinÁtlag
        FROM RészlegÁtlagFiz) RészlegMinÁtlag
 WHERE
       ÁtlagFiz = MinÁtlag AND
       dolgozó.deptno = RészlegÁtlagFiz.deptno;
SELECT * FROM SzegényDolgozók;

```

**Eredmény**

A nézet létrejött.

EMPNO	RÉGIFIZETÉS
7698	2850
7654	1250
7499	1600
7844	1500
7900	950
7521	1250

6 sor kijelölve.

## 6. lépés (A legkisebb átlagfizetésű részlegen a dolgozók fizetésének módosítása, listázás)

UPDATE dolgozó

SET újfizetés =

```
(SELECT ROUND(régifizetés * 1.1)
   FROM SzegényDolgozók
 WHERE dolgozó.empno = SzegényDolgozók.empno);
```



### Megjegyzés

Ha ezzel az utasítással módosítanánk, akkor a nem SzegényDolgozók nézetbeli dolgozók újfizetés értékei „eltűnnének”, hiszen esetükben a fenti allekérdezés NULL értéket ad. E NULL értéket tehát az eredeti fizetésértékkel kell helyettesíteni az NVL függvény felhasználásával. Az adatmódosító utasítás tehát helyesen:

UPDATE dolgozó

SET újfizetés =

```
NVL((SELECT ROUND(régifizetés * 1.1)
      FROM SzegényDolgozók
     WHERE dolgozó.empno = SzegényDolgozók.empno),
      régifizetés));
```

SET numwidth 5

SELECT \* FROM dolgozó;

SET numwidth 10

### Eredmény

14 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	RÉGIFIZETÉS	COMM	DEPTNO	ÚJFIZETÉS
7839	KING	PRESIDENT		81-NOV-17	5000		10	5000
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	3135
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10	2450
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20	2975
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	1375
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	1760
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	1650
7900	JAMES	CLERK	7698	81-DEC-03	950		30	1045
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30	1375
7902	FORD	ANALYST	7566	81-DEC-03	3000		20	3000
7369	SMITH	CLERK	7902	80-DEC-17	800		20	800
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20	3000
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20	1100
7934	MILLER	CLERK	7782	82-JAN-23	1300		10	1300

14 sor kijelölve.

**7. lépés (A legkisebb átlagfizetésű részleg nevének, az itt dolgozók nevének, valamint azok régi és új fizetésének listázása)**

**7.1. változat**

```
BREAK ON dname
SELECT dolgozó.ename,
       dolgozó.régifizetés,
       dolgozó.újfizetés,
       részleg.dname
  FROM dolgozó,
       SzegényDolgozók,
       részleg
 WHERE részleg.deptno = dolgozó.deptno AND
       dolgozó.empno = SzegényDolgozók.empno
 ORDER BY ename;
CLEAR BREAKS
```

**7.2. változat**

```
BREAK ON dname
SELECT ename,
       régifizetés,
       újfizetés,
       dname
  FROM dolgozó,
       részleg
 WHERE részleg.deptno = dolgozó.deptno AND
       NOT újfizetés = régifizetés
 ORDER BY ename;
CLEAR BREAKS
```

**Eredmény (mindkét változat esetén)**

ENAME	RÉGIFIZETÉS	ÚJFIZETÉS	DNAME
ALLEN	1600	1760	SALES
BLAKE	2850	3135	
JAMES	950	1045	
MARTIN	1250	1375	
TURNER	1500	1650	
WARD	1250	1375	

6 sor kijelölve.

**8. lépés (Nézettáblák törlése)**

```
DROP VIEW RészlegÁtlagFiz;
DROP VIEW SzegényDolgozók;
```

**Eredmény**

A nézet eldobva.

A nézet eldobva.

**6.18. feladat**

Írjon szkript programot, amely létrehoz az emp táblából egy dolgozó táblát. Növelje meg ebben 20%-kal a három legrosszabb fizetett dolgozó fizetését az eredeti fizetés oszlopban, és listázza ki a dolgozó táblát. A megoldáshoz használjon nézettáblát.

**Előkészületek a megoldáshoz**

-- Munkatábla törlése és létrehozása

DROP TABLE dolgozó;

CREATE TABLE dolgozó

AS SELECT \* FROM emp;

**1. megoldás (Nézettáblával, UPDATE – SET-beli allekérdezéssel)****1.1. lépés (A három legszegényebb nézetének létrehozása)**

CREATE OR REPLACE VIEW szegény

AS

```
SELECT empno,
       ROWNUM AS sorszám
  FROM (SELECT empno
         FROM dolgozó
        ORDER BY sal)
 WHERE ROWNUM <= 3;
```

SELECT \* FROM szegény;

**Eredmény**

A nézet létrejött.

EMPNO	SORSZÁM
7369	1
7900	2
7876	3

**1.2. lépés (Növeljük meg ezeknek és csak ezeknek a fizetését – adatmódosítás, listázás)**

UPDATE dolgozó

```
SET sal = (SELECT sal*1.2
            FROM szegény
           WHERE dolgozó.empno = szegény.empno);
```

**Megjegyzés**

Az előző feladat 6. lépésében az újfizetés értékre vonatkozó adatmódosításnál tett megjegyzés értelmében a fenti utasítás hibás. A helyes alak:

```
UPDATE dolgozó
SET sal = NVL((SELECT ROUND(sal*1.2)
                 FROM szegény
                WHERE dolgozó.empno = szegény.empno),
               sal);

SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
DROP VIEW szegény;
```

**Eredmény**

14 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	1140		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	960		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1320		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

A nézet eldobva.

**2. megoldás (Nézettáblával, UPDATE – WHERE-beli allekérdezéssel)****2.1. lépés (A három legszegényebb nézetének létrehozása)**

```
CREATE OR REPLACE VIEW szegény
```

```
AS
```

```
SELECT empno
      FROM (SELECT empno
              FROM dolgozó
             ORDER BY sal
        LIMIT 3)
```

```

        ORDER BY sal)
WHERE ROWNUM <= 3;
SELECT * FROM szegény;

```

**Eredmény**

A nézet létrejött.

EMPNO
7369
7900
7876

**2.2. lépés (Növeljük meg ezeknek és csak ezeknek a fizetését – adatmódosítás, listázás)**

```

UPDATE dolgozó
SET sal = ROUND(sal*1.2)
WHERE empno IN
      (SELECT empno FROM szegény);
-- SET numwidth 5
-- SELECT * FROM dolgozó;
-- SET numwidth 10
DROP VIEW szegény;

```

**Eredmény**

3 sor módosítva.

(Az 1. megoldás eredménytábláját kapjuk)

A nézet eldobva.

**3. megoldás (Nézettábla nélkül, UPDATE – WHERE-beli összetett allekérdezéssel)**

```

UPDATE dolgozó
SET sal = ROUND(sal*1.2)
WHERE empno IN (SELECT empno
                 FROM (SELECT empno
                        FROM emp
                        ORDER BY sal)
                 WHERE ROWNUM <= 3);

```

**Eredmény**

3 sor módosítva.

(Az 1. megoldás eredménytábláját kapjuk)

## 6.19. feladat

Írjon szkript programot, mely minden részlegben megnöveli a részleg-átlagfizetésnél alacsonyabb jövedelmű dolgozók fizetését, az ottani legnagyobb és legkisebb jövedelem közötti különbség 10%-ával. A szkript program végül listázzon is elsődlegesen a részlegazonosító szerint növekvően, másodlagosan pedig a jövedelem szerint csökkenően rendezve.

### Előkészületek a megoldáshoz

```
-- Munkatábla törlése és létrehozása
```

```
DROP TABLE dolgozó;
```

```
CREATE TABLE dolgozó
```

```
AS SELECT * FROM emp;
```

### Megoldás

#### 1. lépés (Nézet létrehozása a részlegbeli jövedelmek átlagának, maximumának és minimumának meghatározására)

```
CREATE OR REPLACE VIEW RészlegNézet
```

```
AS
```

```
SELECT deptno,
       ROUND(AVG(sal+NVL(comm,0))) AS ÁtlagJöv,
       MIN(sal+NVL(comm,0))        AS MinJöv,
       MAX(sal+NVL(comm,0))        AS MaxJöv
  FROM dolgozó
 GROUP BY deptno;
SELECT * FROM RészlegNézet;
```

### Eredmény

A nézet létrejött.

DEPTNO	ÁTLAGJÖV	MINJÖV	MAXJÖV
10	2917	1300	5000
20	2251	960	3000
30	1965	1140	2850

#### 2. lépés (A részlegátlagnál kisebb jövedelmű dolgozók azonosítása)

```
CREATE OR REPLACE VIEW SzegényDolgozó
```

```
AS
```

```
SELECT empno,
       dolgozó.deptno
  FROM dolgozó,
       RészlegNézet
 WHERE sal+NVL(comm,0) < ÁtlagJöv AND
       dolgozó.deptno = RészlegNézet.deptno;
SELECT * FROM SzegényDolgozó;
```

**Eredmény**

A nézet létrejött.

EMPNO	DEPTNO
7782	10
7934	10
7876	20
7369	20
7499	30
7844	30
7900	30
7521	30

8 sor kijelölve.

**3. lépés (A keresetnövelés meghatározása)**

```
CREATE OR REPLACE VIEW Növekmény
AS
SELECT empno,
       0.1 * (RészlegNézet.MaxJöv -
               RészlegNézet.MinJöv) AS Érték
  FROM SzegényDolgozó,
       RészlegNézet
 WHERE SzegényDolgozó.deptno = RészlegNézet.deptno;
SELECT * FROM Növekmény;
```

**Eredmény**

A nézet létrejött.

EMPNO	ÉRTÉK
7782	370
7934	370
7876	204
7369	204
7499	171
7844	171
7900	171
7521	171

8 sor kijelölve.

#### 4. lépés (A kiválasztott személyek módosítása)

##### 4.1. változat

```
UPDATE dolgozó
SET sal = sal + NVL((SELECT Érték
                      FROM Növekmény
                     WHERE dolgozó.empno = Növekmény.empno),
                     0);
```

##### Eredmény

14 sor módosítva.

##### 4.2. változat

```
UPDATE dolgozó
SET sal = sal + (SELECT Érték
                   FROM Növekmény
                  WHERE dolgozó.empno = Növekmény.empno)
WHERE empno IN (SELECT empno FROM SzegényDolgozó);
```

##### Eredmény

8 sor módosítva.



##### Megjegyzés

- Nyilván a 4.2. változat a hatékonyabb, hiszen kevesebb adaton kell végrehajtani a módosítást.
- A két változat kipróbálása között természetesen ki kellett adni a ROLLBACK utasítást.

#### 5. lépés (Végeredmény listázása, adat-visszaállítás, nézetek eldobása)

```
SET numwidth 5
SELECT *
  .FROM dolgozó
 ORDER BY deptno ASC,
          sal+NVL(comm,0) DESC;
SET numwidth 10
```

ROLLBACK;

DROP VIEW RészlegNézet;

DROP VIEW SzegényDolgozó;

DROP VIEW Növekmény;

##### Eredmény

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7782	CLARK	MANAGER	7839	81-JÚN-09	2820		10

7934 MILLER	CLERK	7782	82-JAN-23	1670	10
7902 FORD	ANALYST	7566	81-DEC-03	3000	20
7788 SCOTT	ANALYST	7566	82-DEC-09	3000	20
7566 JONES	MANAGER	7839	81-ÁPR-02	2975	20
7876 ADAMS	CLERK	7788	83-JAN-12	1320	20
7369 SMITH	CLERK	7902	80-DEC-17	1020	20
7698 BLAKE	MANAGER	7839	81-MÁJ-01	2850	30
7654 MARTIN	SALESMAN	7698	81-SZE-28	1250	1400
7499 ALLEN	SALESMAN	7698	81-FEB-20	1790	300
7521 WARD	SALESMAN	7698	81-FEB-22	1440	500
7844 TURNER	SALESMAN	7698	81-SZE-08	1690	0
7900 JAMES	CLERK	7698	81-DEC-03	1140	30

14 sor kijelölve.

A visszaállítás befejeződött.

A nézet eldobva.

A nézet eldobva.

A nézet eldobva.

## 6.20. feladat

Írjon szkript programot, mely foglalkozásonként megnöveli az azon belül legalacsonyabb fizetésű dolgozók fizetését, az összes dolgozó átlagfizetése és foglalkozási csoportjuk átlagfizetése közötti különbség 20%-ával, ha ez a különbség pozitív érték. A szkript program végül listázzon is elsődlegesen a foglalkozás szerint növekvően, másodlagosan pedig a jövedelem szerint csökkenően rendezve.

### Előkészületek a megoldáshoz

```
-- Munkatábla törlése és létrehozása
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
```

### Megoldás

#### 1. lépés (Nézet létrehozása a munkakörbeli fizetések minimumának és átlagának meghatározására)

```
CREATE OR REPLACE VIEW MunkakörNézet
AS
SELECT job,
       MIN(sal)          AS MinFiz,
       ROUND(AVG(sal))   AS ÁtlagFiz
  FROM dolgozó
 GROUP BY job;
SELECT * FROM MunkakörNézet;
```

**Eredmény**

A nézet létrejött.

JOB	MINFIZ	ÁTLAGFIZ
ANALYST	3000	3000
CLERK	800	1038
MANAGER	2450	2758
PRESIDENT	5000	5000
SALESMAN	1250	1400

**2. lépés (A munkakör szerinti legkisebb fizetésű dolgozó azonosítása)**

```
CREATE OR REPLACE VIEW SzegényDolgozó
AS
```

```
SELECT empno,
       MK.job
  FROM dolgozó      D,
       MunkakörNézet  MK
 WHERE D.sal = MK.MinFiz;
SELECT * FROM SzegényDolgozó;
```

**Eredmény**

A nézet létrejött.

EMPNO	JOB
7369	CLERK
7654	SALESMAN
7521	SALESMAN
7782	MANAGER
7902	ANALYST
7788	ANALYST
7839	PRESIDENT

7 sor kijelölve.

**3. lépés (A dolgozói összes átlag meghatározása)**

```
CREATE OR REPLACE VIEW ÖsszÁtlagNézet
AS
```

```
SELECT ROUND(AVG(sal)) AS ÖsszÁtlag
  FROM dolgozó;
SELECT * FROM ÖsszÁtlagNézet;
```

**Eredmény**

A nézet létrejött.

ÖSSZÁTLAG

-----  
2073

**4. lépés (A fizetésemelés meghatározása)**

```
CREATE OR REPLACE VIEW NövekményNézet
AS
SELECT empno,
       ROUND(0.2 * (ÖsszÁtlag - ÁtlagFiz)) AS Növekmény
  FROM SzegényDolgozó SZ,
       MunkakörNézet MK,
       ÖsszÁtlagNézet ÖÁ
 WHERE SZ.job = MK.job;
SELECT * FROM NövekményNézet;
```

**Eredmény**

A nézet létrejött.

EMPNO NÖVEKMÉNY

7369	207
7654	135
7521	135
7782	-137
7902	-185
7788	-185
7839	-585

7 sor kijelölve.

**5. lépés (A fizetések módosítása pozitív növekmény esetén)****5.1. változat**

```
UPDATE dolgozó
   SET sal = sal + NVL((SELECT Növekmény
                          FROM NövekményNézet
                         WHERE dolgozó.empno = NövekményNézet.empno
AND
                           Növekmény > 0),
                      0)
 WHERE empno IN (SELECT empno
                  FROM SzegényDolgozó);
```

**Eredmény**

7 sor módosítva.

**5.2. változat**

UPDATE dolgozó

```

SET sal = sal + (SELECT Növekmény
                   FROM NövekményNézet
                   WHERE dolgozó.empno = NövekményNézet.empno)
WHERE empno IN (SELECT SZ.empno
                  FROM SzegényDolgozó SZ,
                      NövekményNézet NN
                  WHERE SZ.empno = NN.empno AND
                      Növekmény > 0);

```

**Eredmény**

3 sor módosítva.

**Megjegyzés**

Természetesen ezúttal is a WHERE utasításrészben szűrő megoldás (5.2. változat) a hatékonyabb, hiszen ebben az esetben kevesebb adatot kell módosítani.

**6. lépés (A végeredmény listázása, adat-visszaállítás, a létrehozott nézetek törlése)**

SET numwidth 5

SELECT \*

FROM dolgozó

ORDER BY job ASC,

sal+NVL(comm,0) DESC;

SET numwidth 10

ROLLBACK;

DROP VIEW SzegényDolgozó;

DROP VIEW MunkakörNézet;

DROP VIEW ÖsszÁtlagNézet;

DROP VIEW NövekményNézet;

**Eredmény**

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7369	SMITH	CLERK	7902	80-DEC-17	1007		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30

7566 JONES	MANAGER	7839	81-ÁPR-02	2975	20
7698 BLAKE	MANAGER	7839	81-MÁJ-01	2850	30
7782 CLARK	MANAGER	7839	81-JÚN-09	2450	10
7839 KING	PRESIDENT		81-NOV-17	5000	10
7654 MARTIN	SALESMAN	7698	81-SZE-28	1385	1400
7499 ALLEN	SALESMAN	7698	81-FEB-20	1600	300
7521 WARD	SALESMAN	7698	81-FEB-22	1385	500
7844 TURNER	SALESMAN	7698	81-SZE-08	1500	0

14 sor kijelölve.

A visszaállítás befejeződött.

A nézet eldobva.

A nézet eldobva.

A nézet eldobva.

A nézet eldobva.

# Összetett SQL-feladatsorok

## Feladatok és megoldások

Mivel e feladatsorok egyes feladatai igen hasonlóak a korábbi fejezetek példáihoz, feladataihoz, ezért a megoldásrészben csak azok megoldásait mutatjuk be, melyek vagy bonyolultságukban, vagy a megoldásukhoz szükséges ötletekben valami újat tartalmaznak.

### A. feladatsor (megoldások)

**A1. feladat** (Részleges megoldás – tábla létrehozása. Ezt az összes többi feladatsor megoldása előtt meg kell tenni!)

**(A1.) Megoldás** (Az emp és dept táblákból a dolgozó és részleg táblák létrehozása)

-- Munkatáblák törlése

```
DROP TABLE dolgozó;
```

```
DROP TABLE részleg;
```

-- Munkatáblák létrehozása

```
CREATE TABLE dolgozó
```

```
AS SELECT * FROM emp;
```

```
CREATE TABLE részleg
```

```
AS SELECT * FROM dept;
```

#### Eredmény

A tábla eldobva.

A tábla eldobva.

A tábla létrejött.

A tábla létrejött.

## A2. feladat

Részlegenkénti csoportosításban és ismétlésmentes módon listázza ki a dolgozók nevét, fizetését, részlegük telephelyét és részlegük összfizetését, ez utóbbi szerint rendezve.

### (A2.) 1. megoldás

#### (A2.) 1.1. lépés (Nézetábla létrehozása, mely tartalmazza a részlegenkénti összfizetést)

```
CREATE OR REPLACE VIEW ReszlegInfo
AS
SELECT deptno      AS ReszlegKod,
       SUM(sal)     AS Osszfizetes,
       COUNT(ename)  AS DolgozoSzam
  FROM dolgozó
 GROUP BY deptno;

SELECT * FROM ReszlegInfo;
```

### Eredmény

A nézet létrejött.

	RESZLEGKOD	OSSZFIZETES	DOLGOZOSZAM
10	8750	3	
20	10875	5	
30	9400	6	

#### (A2.) 1.2. lépés (A feladat megoldása)

```
BREAK ON "RészlegHely"  ON "RészlegSzám"  ON "RészlegÖsszfizetése"
```

```
SELECT e.ename          AS "Dolgozónév",
       d.loc            AS "RészlegHely",
       d.deptno         AS "RészlegSzám",
       n.osszfizetes   AS "RészlegÖsszfizetése",
       e.sal            AS "Dolgozó fizetése"
  FROM dolgozó e,
       részleg d,
       ReszlegInfo n
 WHERE e.deptno = d.deptno      AND
       e.deptno = n.ReszlegKod
 ORDER BY n.Osszfizetes;
```

```
CLEAR BREAKS
DROP VIEW ReszlegInfo;
```

**Eredmény**

DolgozóNév	RészlegHely	RészlegSzám	RészlegÖsszfizetése	Dolgozó fizetése
KING	NEW YORK	10	8750	5000
CLARK				2450
MILLER				1300
BLAKE	CHICAGO	30	9400	2850
MARTIN				1250
ALLEN				1600
TURNER				1500
WARD				1250
JAMES				950
JONES	DALLAS	20	10875	2975
SCOTT				3000
ADAMS				1100
SMITH				800
FORD				3000

14 sor kijelölve.

breaks törölve  
A nézet eldobva.

**(A2.) 2. megoldás****(A2.) 2.1. lépés (A telephelyenkénti összfizetés nézettáblájának létrehozása)**

```
CREATE OR REPLACE VIEW teleposszes
AS
SELECT loc,
       SUM(sal) AS sumsal
  FROM dolgozó
     JOIN részleg USING (deptno)
 GROUP BY loc;
```

```
SELECT * FROM teleposszes;
```

**Eredmény**

A nézet létrejött.

LOC	SUMSAL
CHICAGO	9400
DALLAS	10875
NEW YORK	8750

**(A2.) 2.2. lépés (A feladat szerinti nézetábla létrehozása)**

```

CREATE OR REPLACE VIEW feladat2
  (dnev, dfiz, telephely, osszesfiz)
AS
  SELECT ename,
         sal,
         loc,
         sumsal AS összes
    FROM dolgozó
   JOIN
     részleg USING(deptno)
   JOIN
     teleposszes USING(loc)
 ORDER BY összes;

```

SELECT \* FROM feladat2;

**Eredmény**

A nézet létrejött.

DNEV	DFIZ TELEPHELY	OSSZESFIZ
KING	5000 NEW YORK	8750
CLARK	2450 NEW YORK	8750
MILLER	1300 NEW YORK	8750
BLAKE	2850 CHICAGO	9400
MARTIN	1250 CHICAGO	9400
ALLEN	1600 CHICAGO	9400
TURNER	1500 CHICAGO	9400
WARD	1250 CHICAGO	9400
JAMES	950 CHICAGO	9400
JONES	2975 DALLAS	10875
SCOTT	3000 DALLAS	10875
ADAMS	1100 DALLAS	10875
SMITH	800 DALLAS	10875
FORD	3000 DALLAS	10875

14 sor kijelölve.

**(A2.) 2.3. lépés (Formázott kiíratás, nézetek törlése, formázások visszaállítása)**

```

SET PAGESIZE 27
SET LINESIZE 60
TTITLE 'Az A feladatsor megoldása|2. feladat'
BTITLE 'Készítették:|a Budapesti Műszaki Főiskola|Oracle oktatói'

```

```

COLUMN dnev      HEADING 'Név'          JUSTIFY CENTER FORMAT A10
COLUMN dfiz      HEADING 'Fizetés'       -
COLUMN telephely HEADING 'Telephely|cime' JUSTIFY CENTER FORMAT A10
COLUMN osszesfiz HEADING 'Telephelyi|összfizetés' JUSTIFY RIGHT FORMAT $99,999
BREAK ON telephely ON osszesfiz

SELECT * FROM feladat2;

DROP VIEW teleposszes;
DROP VIEW feladat2;
CLEAR BREAKS
CLEAR COLUMNS
TTITLE OFF
BTITLE OFF
SET linesize 300
SET pagesize 40

```

**Eredmény**

Cs. Nov 18

lap 1

Az A feladatsor megoldása  
2. feladat

Név	Fizetés	Telephely	Telephelyi címe	Összfizetés
KING	\$5,000	NEW YORK		\$8,750
CLARK	\$2,450			
MILLER	\$1,300			
BLAKE	\$2,850	CHICAGO		\$9,400
MARTIN	\$1,250			
ALLEN	\$1,600			
TURNER	\$1,500			
WARD,	\$1,250			
JAMES	\$950			
JONES	\$2,975	DALLAS		\$10,875
SCOTT	\$3,000			
ADAMS	\$1,100			
SMITH	\$800			
FORD	\$3,000			

Készítették:  
a Budapesti Műszaki Főiskola  
Oracle oktatói

14 sor kijelölve.

A nézet eldobva.

A nézet eldobva.



### Megjegyzés

- Itt a vessző (,) karakter az ezres csoportosítás jele, és nem a tizedesvessző.
- Az SQL\*Plus-utasítások esetén a sorvégi „-” a folytatás jel, vagyis azt jelzi, hogy az utasítás a következő sorban folytatódik.
- Az SQL\*Plus formázó utasításai (COLUMN, BREAK ON stb.) „nem szeretik” az ékezes betűket az oszlopnevekben (a 9.2. és korábbi verziókban)!

## A3. feladat

Vegyen fel egy új dolgozót a legkisebb összfizetésű részlegbe olyan fizetéssel, hogy ez a részleg már éppen ne a legkisebb összfizetésű legyen. Listázással ellenőrizze a felvétel helyességét, végül állítsa vissza az eredeti táblatartalmakat.

### (A3.) 1. megoldás

#### (A3.) 1.1. lépés (Nézet a részlegösszeg és részleglétszám meghatározására)

PROMPT A részlegösszeg és részleglétszám meghatározása:

```
CREATE OR REPLACE VIEW ReszlegSzint
AS
SELECT deptno      AS RészlegKod,
       Osszfiz     AS Osszfizetes,
       ROWNUM      AS Szint,
       létszám
FROM (SELECT deptno,
             SUM(sal)   AS Osszfiz,
             COUNT(*)   AS létszám
        FROM dolgozó
       GROUP BY deptno
      ORDER BY Osszfiz);
```

SELECT \* FROM ReszlegSzint;

### Eredmény

A részlegösszeg és részleglétszám meghatározása:

A nézet létrejött.

RESZLEGKOD	OSSZFIZETES	SZINT	LÉTSZÁM
10	8750	1	3
30	9400	2	6
20	10875	3	5

**(A3.) 1.2. lépés (Nézet a második és első legkisebb összfizetés különbségére)**

PROMPT Mennyi az eltérés a második és az első között:

CREATE OR REPLACE VIEW Különbség

AS

```

SELECT ossz2,
       ossz1,
       ossz2 - ossz1      AS elteres
  FROM (SELECT osszfizetes AS ossz2
         FROM ReszlegSzint
        WHERE szint = 2),
       (SELECT osszfizetes AS ossz1
         FROM ReszlegSzint
        WHERE szint = 1);
    
```

SELECT \* FROM különbség;

**Eredmény**

Mennyi az eltérés a második és az első között:

A nézet létrejött.

OSSZ2	OSSZ1	ELTERES
9400	8750	650

**(A3.) 1.3. lépés (Az új dolgozó beléptetése, akinek fizetése az összfizetés-különbség + 1 lesz)**

PROMPT Az új dolgozó beléptetése

INSERT INTO dolgozó

```

(empno, ename, job,
 mgr,
 hiredate,
 sal,
 comm,
 deptno)
    
```

```

VALUES (4444, 'Ujbelépő', 'TITKÁR',
        (SELECT empno
         FROM dolgozó
        WHERE UPPER(job) = 'PRESIDENT'),
    
```

```

        sysdate,
        (SELECT elteres + 1
         FROM különbség),
        NULL,
        (SELECT deptno
         FROM dolgozó
         WHERE UPPER(job) = 'PRESIDENT') );

```

SELECT \* FROM dolgozó;

### Eredmény

Az új dolgozó beléptetése  
1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
4444	Ujbelépő	TITKÁR	7839	04-NOV-18	651		10

15 sor kijelölve.

### (A3.) 1.4. lépés (Ellenőrzés, nézettáblák törlése)

```

SELECT * FROM ReszlegSzint;
DROP VIEW ReszlegSzint;
DROP VIEW Különbség;

```

### Eredmény

RESZLEGKOD	OSSZFIZETES	SZINT	LÉTSZÁM
30	9400	1	6
10	9401	2	4
20	10875	3	5

A nézet eldobva.  
A nézet eldobva.

**(A3.) 2. megoldás****(A3.) 2.1. lépés (Az összfizetések nézete)**

```
CREATE OR REPLACE VIEW részlegnézet1
AS
SELECT reszl.deptno      AS Részlegszám,
       reszl.loc        AS Telephely,
       altabla.osszfiz  AS Összfizetés
  FROM részleg reszl,
       (SELECT SUM(sal)  AS osszfiz,
              deptno    AS részleg
             FROM dolgozó
            GROUP BY deptno)  altabla
 WHERE (reszl.deptno = altabla.részleg);
```

```
SELECT * FROM részlegnézet1;
```

**Eredmény**

A nézet létrejött.

RÉSZLEGSZÁM TELEPHELY	ÖSSZFIZETÉS
10 NEW YORK	8750
20 DALLAS	10875
30 CHICAGO	9400

**(A3.) 2.2. lépés (A legkisebb és a második legkisebb fizetésű részleg közötti különbség)**

```
SELECT (SUM(al.fiz) - MIN(al.fiz)) - MIN(al.fiz)  AS kulonbseg
  FROM (SELECT összfizetés  AS fiz
         FROM részlegnézet1
        ORDER BY összfizetés)  al
 WHERE ROWNUM <= 2;
```

**Eredmény**

KULONBSEG
650

**(A3.) 2.3. lépés (Új dolgozó felvitele)**

```
INSERT INTO dolgozó
VALUES(9999,
       'ujDolgozó',
```

```

'ujMunka',
7698,
sysdate,
(SELECT (SUM(al.fiz) - MIN(al.fiz)) -
    MIN(al.fiz) AS kulonbseg
  FROM (SELECT összfizetés AS fiz
        FROM részlegnézet1
        ORDER BY összfizetés) al
 WHERE ROWNUM <= 2)+1,
NULL,
(SELECT részlegszám
  FROM részlegnézet1
 WHERE összfizetés = (SELECT MIN(összfizetés)
                       FROM részlegnézet1));
SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10

```

### Eredmény

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
9999	ujDolgozó	ujMunka	7698	04-NOV-18	651		10

15 sor kijelölve.

### (A3.) 2.4. lépés (Ellenőrzés, táblatartalom visszaállítása, nézettábla törlése)

```

SELECT * FROM részlegnézet1;
ROLLBACK;
DROP VIEW részlegnézet1;

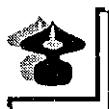
```

**Eredmény**

RÉSZLEGSZÁM TELEPHELY	ÖSSZFIZETÉS
10 NEW YORK	9401
20 DALLAS	10875
30 CHICAGO	9400

A visszaállítás befejeződött.

A nézet eldobva.

**Megjegyzés**

- Az új dolgozónak a fenti megoldásokban meghatározott 651 USD fizetése az az összeg, mely a feladat követelményeit *minimális fizetésértékkal* kielégíti. A tényleges fizetés ennél lehet több, de nem akármennyivel, hiszen a követelmények között az is szerepelt, hogy az új dolgozó felvételével annak részlege „éppen ne a legkisebb összfizetésű legyen”.
- Fölvethető a kérdés, hogy helyes-e a fenti megoldás abban az esetben, ha a két legkisebb összfizetésű részleg összfizetése azonos. Ekkor ugyan a „legkisebb” összfizetésű részlegbe felvéve az új dolgozót, az összfizetés már éppen nem a legkisebb lesz (ebben az értelemben tehát a megoldás megfelelő), csak hát a két azonos összfizetésű részleg közül mi alapján döntöttünk. Kétségtelen, hogy a döntés esetleges volt (például a tárolási sorrend szerint történt), mégse a megoldást hibázattassuk. A fenti probléma elkerülhető lett volna a feladat más megfogalmazásával. Például „Vegyen fel egy új dolgozót a legkisebb összfizetésű részlegek valamelyikébe...”, vagy „Vegyen fel egy-egy új dolgozót a legkisebb összfizetésű részlegek mindegyikébe...”.

**A4. feladat**

Listázza ki formázottan (fej- és lábléccel stb.) a felhasználó által megadott telephelyű részleg dolgozónak nevét, fizetését, részlegének telephelyét és annak összfizetését, valamint az abban dolgozó főnökök (mgr) nevét és fizetését, a főnök neve és a dolgozók neve szerint rendezve.

**(A4.) 1. megoldás****(A4.) 1.1. lépés (Dolgozók és főnökök adatai részlegenként – ez inline nézet lesz)**

```

SELECT loc          AS Hely,
       dolg.ename   AS DNev,
       dolg.sal     AS DFiz,
       főnök.ename  AS FNev,
       főnök.sal    AS FFiz
  FROM dolgozó dolg,
       dolgozó főnök,
       dept d
 WHERE főnök.empno (+) = dolg.mgr      AND
       dolg.deptno = d.deptno;

```

**Eredmény**

HELY	DNEV	DFIZ	FNEW	FFIZ
DALLAS	FORD	3000	JONES	2975
DALLAS	SCOTT	3000	JONES	2975
CHICAGO	MARTIN	1250	BLAKE	2850
CHICAGO	ALLEN	1600	BLAKE	2850
CHICAGO	JAMES	950	BLAKE	2850
CHICAGO	WARD	1250	BLAKE	2850
CHICAGO	TURNER	1500	BLAKE	2850
NEW YORK	MILLER	1300	CLARK	2450
DALLAS	ADAMS	1100	SCOTT	3000
CHICAGO	BLAKE	2850	KING	5000
NEW YORK	CLARK	2450	KING	5000
DALLAS	JONES	2975	KING	5000
DALLAS	SMITH	800	FORD	3000
NEW YORK	KING	5000		

14 sor kijelölve.

**(A4.) 1.2. lépés (Részlegenkénti összfizetés – ez is inline nézet lesz)**

```
SELECT SUM(sal) AS ROsszes,
       loc      AS Hely
  FROM dolgozó,
       részleg
 WHERE dolgozó.deptno = részleg.deptno
 GROUP BY loc;
```

**Eredmény**

ROSSZES HELY

9400 CHICAGO
10875 DALLAS
8750 NEW YORK

**(A4.) 1.3. lépés (A két inline nézet összekapcsolása és formázás)**

```
ACCEPT telephely PROMPT 'Telephely: '
SET verify OFF
SET feedback OFF
SET linesize 70
SET pagesize 18
TTITLE "A mintapélda megoldása|formázott és ismétlésmentes kiiratással"
BTITLE "Készítették:|a Budapesti Műszaki Főiskola|adatbázis oktatói"
```

```

COLUMN DNeve      HEADING 'Dolgozó|neve'
COLUMN Dfiz       HEADING 'Dolgozó|fizetése'      JUSTIFY RIGHT
COLUMN ROsszes   HEADING 'Részleg|összfizetése' JUSTIFY RIGHT
COLUMN Hely       HEADING 'Részleg|helye'           FORMAT A10
COLUMN FNeve     HEADING 'Főnök|neve'              FORMAT A7
COLUMN FFiz       HEADING 'Főnök|fizetése'        JUSTIFY RIGHT FORMAT 9999
BREAK ON ROsszes ON Hely ON FNeve ON FFiz

SELECT DNeve, Dfiz, ROsszes, adatok.Hely, FNeve, FFiz
  FROM
    (SELECT loc          AS Hely,
            dolg.ename  AS DNeve,
            dolg.sal    AS Dfiz,
            főnök.ename AS FNeve,
            főnök.sal   AS FFiz
     FROM dolgozó dolg,
          dolgozó főnök,
          dept d
    WHERE főnök.empno (+) = dolg.mgr      AND
          dolg.deptno = d.deptno)      adatok,
    (SELECT SUM(sal)   AS ROsszes,
            loc         AS Hely
     FROM dolgozó,
          részleg
    WHERE dolgozó.deptno = részleg.deptno
    GROUP BY loc)                      resz1
  WHERE resz1.Hely = adatok.Hely AND
        UPPER(resz1.Hely) = UPPER('&telephely')
  ORDER BY FNeve, DNeve;

CLEAR BREAKS
CLEAR COLUMNS
TTITLE OFF
BTITLE OFF
SET verify ON
SET feedback ON

```

**Eredmény**

Telephely: Chicago

Cs. Nov 18

lap 1

A mintapélda megoldása  
formázott és ismétlésmentes kiíratással

Dolgozó neve	Dolgozó fizetése	Részleg összfizetése	Részleg helye	Főnök neve	Főnök fizetése
ALLEN	1600		9400 CHICAGO	BLAKE	2850
JAMES	950				
MARTIN	1250				
TURNER	1500				
WARD	1250				
BLAKE	2850			KING	5000

Készítették:  
a Budapesti Műszaki Főiskola  
adatbázis oktatói

#### (A4.) 2. megoldás

##### (A4.) 2.1. lépés (A teleposszes nézettábla létrehozása)

```
CREATE OR REPLACE VIEW teleposszes
AS
  SELECT loc,
         SUM(sal) AS sumsal
    FROM dolgozó
   JOIN
        részleg USING(deptno)
  GROUP BY loc;
SELECT * FROM teleposszes;
```

#### Eredmény

A nézet létrejött.

LOC	SUMSAL
CHICAGO	9400
DALLAS	10875
NEW YORK	8750

3 sor kijelölve.

##### (A4.) 2.2. lépés (A megoldás nézettábla létrehozása)

ACCEPT hely CHAR PROMPT "Részleg helye: "

```
CREATE OR REPLACE VIEW megoldas
(dnev, dfiz, telephely, szumfiz, fnev, ffiz)
AS
  SELECT d.ename,
```

```

d.sal,
loc,
sumsal,
f.ename,
f.sal
FROM dolgozó d
RIGHT JOIN
    dolgozó f ON (d.mgr = f.empno)
INNER JOIN
    részleg ON (d.deptno = részleg.deptno)
LEFT JOIN
    teleposzes USING(loc)
WHERE UPPER(loc) = UPPER('&hely')
ORDER BY f.ename, d.ename;
SELECT * FROM megoldas;

```

**Eredmény**

Részleg helye: Chicago  
régi 17: WHERE UPPER(loc) = UPPER('&hely')  
új 17: WHERE UPPER(loc) = UPPER('Chicago')

A nézet létrejött.

DNEV	DFIZ TELEPHELY	SZUMFIZ FNEV	FFIZ
ALLEN	1600 CHICAGO	9400 BLAKE	2850
JAMES	950 CHICAGO	9400 BLAKE	2850
MARTIN	1250 CHICAGO	9400 BLAKE	2850
TURNER	1500 CHICAGO	9400 BLAKE	2850
WARD	1250 CHICAGO	9400 BLAKE	2850
BLAKE	2850 CHICAGO	9400 KING	5000

6 sor kijelölve.

**(A4.) 2.3. lépés (Az eredmény lekérdezése formázottan)**

```

SET pagesize 20
SET linesize 70
BREAK ON telephely ON szumfiz ON fnev
TTITLE 'A feladatsor megoldása| |A4. feladat'
BTITLE 'Készítettek:|A Budapesti Műszaki Főiskola|Oracle oktatói'
COLUMN dnev      HEADING 'Név'          JUSTIFY CENTER FORMAT A10
COLUMN dfiz      HEADING 'Fizetés'       JUSTIFY RIGHT  FORMAT $99,999
COLUMN deptnev   HEADING 'Telephely|címé' JUSTIFY CENTER FORMAT A10
COLUMN szumfiz   HEADING 'Telephely|összfizetés' JUSTIFY RIGHT FORMAT $99,999

```

COLUMN fnev HEADING 'Név' JUSTIFY CENTER FORMAT A10  
 COLUMN ffiz HEADING 'Fizetés' JUSTIFY RIGHT FORMAT \$99,999

SELECT \* FROM megoldas  
 ORDER BY fnev, dnev;

CLEAR BREAKS  
 CLEAR COLUMNS  
 TTITLE OFF  
 BTITLE OFF  
 SET linesize 300  
 SET pagesize 40  
 DROP VIEW megoldas;  
 DROP VIEW teleposszes;

### Eredmény

P. Nov 19

lap 1

A feladatsor megoldása

A4. feladat

Név	Fizetés	TELEPHELY	Telephely		
			Összfizetés	Név	Fizetés
ALLEN	\$1,600	CHICAGO	\$9,400	BLAKE	\$2,850
JAMES	\$950				\$2,850
MARTIN	\$1,250				\$2,850
TURNER	\$1,500				\$2,850
WARD	\$1,250				\$2,850
BLAKE	\$2,850			KING	\$5,000

Készítették:  
 A Budapesti Műszaki Főiskola  
 Oracle oktatói

6 sor kijelölve.

A nézet eldobva.  
 A nézet eldobva.

## B. feladatsor (megoldások)

### B2. feladat

Munkakörönkénti csoportosításban és ismétlésmentes módon listázza ki a dolgozók nevét, telephelyét, fizetését, munkakörét és a munkakörében dolgozók összfizetését, ez utóbbi szerint rendezve.

#### Előkészület a megoldáshoz (A munkatáblák törlése és létrehozása)

```
DROP TABLE dolgozó;
DROP TABLE részleg;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
CREATE TABLE részleg
AS SELECT * FROM dept;
```

#### (B2.) 1. megoldás (Nézettábla nélkül)

```
BREAK ON munkakor ON mfizetes
SELECT ename      AS név,
       loc        AS telephely,
       sal        AS fizetés,
       dolgozó.job AS munkakor,
       sumsal     AS mfizetes
  FROM dolgozó,
       (SELECT SUM(sal)  AS sumsal,
              job
             FROM dolgozó
            GROUP BY job) al,
       részleg
 WHERE dolgozó.job = al.job AND
       dolgozó.deptno = részleg.deptno
 ORDER BY mfizetes;
CLEAR BREAKS
```

#### (B2.) 2. megoldás (Nézettáblával)

```
CREATE OR REPLACE VIEW SumSalView
AS
SELECT job,
       SUM(sal)  AS sumsal
  FROM dolgozó
 GROUP BY job;
```

```
BREAK ON munkakor ON mfizetes
SELECT ename      AS név,
```

```

loc          AS telephely,
sal          AS fizetés,
job          AS munkakor,
sumsal      AS mfizetés

FROM részleg
    INNER JOIN
        dolgozó USING(deptno)
    LEFT JOIN
        SumSalView USING(job)
ORDER BY mfizetés;

CLEAR BREAKS
DROP VIEW SumSalView;

```

#### Eredmény (mindkét megoldás esetén)

NÉV	TELEPHELY	FIZETÉS	MUNKAKOR	MFIZETÉS
MILLER	NEW YORK	1300	CLERK	4150
ADAMS	DALLAS	1100		
SMITH	DALLAS	800		
JAMES	CHICAGO	950		
KING	NEW YORK	5000	PRESIDENT	5000
WARD	CHICAGO	1250	SALESMAN	5600
TURNER	CHICAGO	1500		
MARTIN	CHICAGO	1250		
ALLEN	CHICAGO	1600		
SCOTT	DALLAS	3000	ANALYST	6000
FORD	DALLAS	3000		
JONES	DALLAS	2975	MANAGER	8275
CLARK	NEW YORK	2450		
BLAKE	CHICAGO	2850		

14 sor kijelölve.

### B3. feladat

Vegyen fel egy új dolgozót a legkisebb összfizetésű munkakörbe olyan fizetéssel, hogy az már éppen ne a legkisebb összfizetésű legyen. Listázással ellenőrizze a felvétel helyességét, végül állítsa vissza az eredeti táblatartalmakat.

#### (B3.) Megoldás

##### (B3.) 1. lépés (Első nézet: a legkisebb és a második legkisebb összfizetés munkakörönként)

```

CREATE OR REPLACE VIEW KétLegkisebb
    (Munkakör, ÖsszFiz, Sorszám)
AS

```

```

SELECT al.* , ROWNUM
FROM (SELECT job,
            SUM(sal) AS ÖsszFiz
      FROM dolgozó
     GROUP BY job
    ORDER BY ÖsszFiz) al
 WHERE ROWNUM <= 2;
SELECT * FROM KétLegkisebb;

```

**Eredmény**

MUNKAKÖR	ÖSSZFIZ	SORSZÁM
CLERK	4150	1
PRESIDENT	5000	2

**(B3.) 2. lépés (Második nézet: a legkisebb összfizetés munkakörönként)**

```

CREATE OR REPLACE VIEW Kisebb
  (Munkakör, KisebbÖsszFiz, Sorszám)
AS
SELECT *
  FROM KétLegkisebb
 WHERE Sorszám = 1;
SELECT * FROM Kisebb;

```

**Eredmény**

A nézet létrejött.

MUNKAKÖR	KISEBBÖSSZFIZ	SORSZÁM
CLERK	4150	1

**(B3.) 3. lépés (Harmadik nézet: a két legkisebb közül a nagyobb)**

```

CREATE OR REPLACE VIEW Nagyobb
  (Munkakör, NagyobbÖsszFiz, Sorszám)
AS
SELECT *
  FROM KétLegkisebb
 WHERE Sorszám = 2;
SELECT * FROM Nagyobb;

```

**Eredmény**

A nézet létrejött.

MUNKAKÖR	NAGYOBBOSSZFIZ	SORSZÁM
PRESIDENT	5000	2

**(B3.) 4. lépés (Negyedik nézet: a két legkisebb különbsége)**

```

CREATE VIEW Különbség
(Érték)
AS
SELECT NagyobbÖsszFiz - KisebbÖsszFiz
  FROM Kisebb, Nagyobb;
SELECT * FROM Különbség;

```

**Eredmény**

A nézet létrejött.

ÉRTÉK

850

**(B3.) 5. lépés (Új dolgozó felvétele)**

```

INSERT INTO dolgozó
(empno, ename, job, mgr, hiredate, sal, comm, deptno)
VALUES((SELECT MAX(empno) + 5
        FROM dolgozó),           -- a legnagyobb empno-nál legyen öttel több
       'TUDOR',
       (SELECT Munkakör
        FROM Kisebb),           -- a legkisebb összfizetésű munkakör
       1111,
       sysdate,
       (SELECT Érték + 1
        FROM Különbség),       -- a különbséghez adjunk egyet
       NULL,
       40);

```

```

SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10

```

```

ROLLBACK;
DROP VIEW KétLegkisebb;
DROP VIEW Kisebb;
DROP VIEW Nagyobb;
DROP VIEW Különbség;

```

**Eredmény**

1 sor létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
7939	TUDOR	CLERK	1111	04-NOV-20	851		40

15 sor kijelölve.

A visszaállítás befejeződött.

A nézet eldobva.

A nézet eldobva.

A nézet eldobva.

A nézet eldobva.

**B4. feladat**

Listázza ki formázottan (fej- és lábléccel stb.) a felhasználó által megadott munkakörű dolgozók nevét, munkakörét, telephelyét, fizetését, a munkakörében dolgozók összfizetését, továbbá a főnökének (mgr) nevét, fizetését és telephelyét a dolgozók neve szerint rendezve.

**(B4.) 1. megoldás**

ACCEPT munkakör PROMPT 'Kérem adja meg a munkakört: '

SET linesize 80

SET pagesize 16

SET verify OFF

BREAK ON d\_job ON sumsal

TTITLE 'A &amp;munkakör munkakörű dolgozó adatai'

```

BTITLE 'Az Oracle példatár|megoldása'
COLUMN d_ename    HEADING 'Dolgozó|neve'          FORMAT A7 JUSTIFY LEFT
COLUMN d_job      HEADING 'munkaköre'           FORMAT A9 JUSTIFY CENTER
COLUMN dr_loc     HEADING 'telephelye'          FORMAT A10 JUSTIFY CENTER
COLUMN d_sal      HEADING 'fizetése'            FORMAT $0999 JUSTIFY CENTER
COLUMN sumsal     HEADING 'Munkaköri|összfizetés' FORMAT $9999 JUSTIFY CENTER
CENTER
COLUMN f_ename    HEADING 'Főnök|neve'          FORMAT A8 JUSTIFY LEFT
COLUMN f_sal      HEADING 'fizetése'            FORMAT $0999 JUSTIFY CENTER
COLUMN fr_loc     HEADING 'telephelye'          FORMAT A10 JUSTIFY CENTER

SELECT d.ename    AS d_ename,
       d.job      AS d_job,
       dr.loc     AS dr_loc,
       d.sal      AS d_sal,
       sumsal,
       f.ename    AS f_ename,
       f.sal      AS f_sal,
       fr.loc     AS fr_loc
  FROM részleg     dr,
       dolgozó     d,
       (SELECT SUM(sal)  AS sumsal,
              job
        FROM dolgozó
        GROUP BY job) s,
       dolgozó     f,
       részleg     fr
 WHERE UPPER(d.job) = UPPER('&munkakör') AND
       d.deptno = dr.deptno          AND
       d.job = s.job                AND
       d.mgr = f.empno (+)          AND
       f.deptno = fr.deptno
 ORDER BY d.ename;

CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF
BTITLE OFF
SET verify ON
SET linesize 300
SET pagesize 40

```

**(B4.) 2. megoldás (Nézettáblával és JOIN összekapcsolással)****(B4.) 2.1. lépés (Nézettábla létrehozása)**

```
CREATE OR REPLACE VIEW SumSalView
  (job, sumsal)
AS
  SELECT job,
         SUM(sal)
    FROM dolgozo
   GROUP BY job;
SELECT * FROM SumSalView;
```

**Eredmény**

A nézet létrejött.

JOB	SUMSAL
ANALYST	6000
CLERK	4150
MANAGER	8275
PRESIDENT	5000
SALESMAN	5600

**(B4.) 2.2. lépés (Listakészítés)**

ACCEPT munkakör PROMPT 'Kérlek adja meg a munkakört:

```
SET linesize 80
SET pagesize 16
SET verify OFF

BREAK ON job ON sumsal
TTITLE 'A &munkakör munkakörű|dolgozó adatai'
BTITLE 'Az Oracle példatár|megoldása'
COLUMN d_ename    HEADING 'Dolgozó|neve'          FORMAT A7 JUSTIFY LEFT
COLUMN job        HEADING 'munkaköre'             FORMAT A9 JUSTIFY CENTER
COLUMN dr_loc     HEADING 'telephelye'            FORMAT A10 JUSTIFY CENTER
COLUMN d_sal      HEADING 'fizetése'              FORMAT $0999 JUSTIFY CENTER
COLUMN sumsal     HEADING 'Munkaköri|összfizetés' FORMAT $9999 JUSTIFY CENTER
COLUMN f_ename    HEADING 'Főnök|neve'            FORMAT A8 JUSTIFY LEFT
COLUMN f_sal      HEADING 'fizetése'              FORMAT $0999 JUSTIFY CENTER
COLUMN fr_loc     HEADING 'telephelye'            FORMAT A10 JUSTIFY CENTER

SELECT d_ename AS d_ename,
       job,
```

```

dr.loc    AS dr_loc,
d.sal     AS d_sal,
sumsal,
f.ename   AS f_ename,
f.sal     AS f_sal,
fr.loc    AS fr_loc
FROM részleg dr
INNER JOIN
  dolgozó d USING(deptno)
LEFT JOIN
  SumSalView USING(job)
LEFT JOIN
  dolgozó f ON (d.mgr = f.empno)
INNER JOIN
  részleg fr ON (f.deptno=fr.deptno)
WHERE UPPER(job) = UPPER('&munkakör')
ORDER BY d.ename;

```

```

CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF
BTITLE OFF
SET verify ON
SET linesize 300
SET pagesize 40
DROP VIEW SumSalView;

```

### A listázás eredménye (mindkét megoldás esetén)

Kérem adja meg a munkakört: clerk

Szo Nov 20

lap 1

A clerk munkakörű  
dolgozó adatai

Dolgozó neve	munkaköre	telephelye	fizetése	Munkaköri összfizetés	Főnök neve	fizetése	telephelye
ADAMS	CLERK	DALLAS	\$1100	\$4150	SCOTT	\$3000	DALLAS
JAMES		CHICAGO	\$0950		BLAKE	\$2850	CHICAGO
MILLER		NEW YORK	\$1300		CLARK	\$2450	NEW YORK
SMITH		DALLAS	\$0800		FORD	\$3000	DALLAS

## C. feladatsor (megoldások)

### C3. feladat

Növelje meg minden dolgozó fizetését a legkisebb összfizetésű részlegben annyival, hogy e részleg már éppen ne a legkisebb összfizetésű legyen. Listázással ellenőrizze a növelés helyeségét, végül állítsa vissza az eredeti táblatartalmakat.

#### Előkészület a megoldáshoz (A munkatáblák törlése és létrehozása)

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
```

PROMPT dolgozó:

```
SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
```

#### Eredmény

A tábla eldobva.  
A tábla létrejött.

dolgozó:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

14 sor kijelölve.

**(C3.) 1. megoldás****(C3.) 1.1. lépés (Összfizetés részlegenként)**

```
SELECT deptno,
       SUM(sal),
       COUNT(*)
  FROM dolgozó
 GROUP BY deptno
 ORDER BY SUM(sal);
```

**Eredmény**

DEPTNO	SUM(SAL)	COUNT(*)
10	8750	3
30	9400	6
20	10875	5

**(C3.) 1.2. lépés (Összfizetés, létszám részlegenként – a két legutolsóból nézet)**

```
CREATE OR REPLACE VIEW SzegényRészleg
  (deptno, RészlegÖsszeg, RészlegLétszám)
AS
SELECT *
  FROM (SELECT deptno,
               SUM(sal),
               COUNT(*)
      FROM dolgozó
     GROUP BY deptno
    ORDER BY SUM(sal))
 WHERE ROWNUM <= 2;
PROMPT SzegényRészleg:
SELECT * FROM SzegényRészleg;
```

**Eredmény**

A nézet létrejött.

SzegényRészleg:

DEPTNO	RÉSZLEGÖSSZEG	RÉSZLEGLÉTSZÁM
10	8750	3
30	9400	6

**(C3.) 1.3. lépés (Legkisebb összfizetés, létszám, részleg – az utolsóból nézet)**

```
CREATE OR REPLACE VIEW Legssegényebb
  (deptno, RészlegÖsszeg, RészlegLétszám)
AS
```

```
SELECT *  
  FROM (SELECT deptno,  
            SUM(sal),  
            COUNT(*)  
           FROM dolgozó  
          GROUP BY deptno  
          ORDER BY SUM(sal))  
 WHERE ROWNUM <= 1;  
PROMPT Legszegényebb:  
SELECT * FROM Legszegényebb;
```

## Eredmény

A nézet létrejött.

Legszegényebb:

DEPTNO RÉSZLEGÖSSZEG RÉSZLEGLÉTSZÁM

10 8750 3

#### (C3.) 1.4. lépés (A növekmény meghatározása)

CREATE OR REPLACE VIEW Növekmény

(Frétek)

AS

```
SELECT ROUND(Különbség / Legszegényebb.RészlegLétszám) + 1
    FROM (SELECT MAX(RészlegÖsszeg) - MIN(RészlegÖsszeg) AS Különbség
          FROM SzegényRészleg),
```

Legszegényebb;

### PROMPT Növekmény:

```
SELECT * FROM Növekmény;
```

## Eredmény

A nézet létrejött.

Növekmény:

218

(C3.) 1.5. lépés (Feladat: az előírt fizetések emelése)

### Hibás megoldás

UPDATE dolgozó

SET sal = sal + Növekmény.Érték

```
WHERE deptno = Legszegebb.deptno;
```

**Eredmény**

```
WHERE deptno = Legszegényebb.deptno
```

```
*
```

Hiba a(z) 3. sorban:

ORA-00904: "LEGSZEGÉNYEBB"."DEPTNO": érvénytelen azonosító

**Jó megoldás:**

UPDATE dolgozó

```
SET sal = sal + (SELECT Érték FROM Növekmény)
```

```
WHERE deptno = (SELECT deptno FROM Legszegényebb);
```

PROMPT dolgozó:

SET numwidth 5

SELECT \* FROM dolgozó;

SET numwidth 10

**Eredmény**

3 sor módosítva.

dolgozó:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5218		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2668		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1518		10

14 sor kijelölve.

**(C3.) 1.6. lépés (Ellenőrzés, táblaértékek visszaállítása, nézettáblák törlése)**

PROMPT SzegényRészleg:

SELECT \* FROM SzegényRészleg;

ROLLBACK;

```
DROP VIEW SzegényRészleg;
DROP VIEW Legszegényebb;
DROP VIEW Növekmény;
```

**Eredmény**

DEPTNO	RÉSZLEGÖSSZEG	RÉSZLEGLÉTSZÁM
--------	---------------	----------------

30	9400	6
10	9404	3

A visszaállítás befejeződött.

A nézet eldobva.

A nézet eldobva.

A nézet eldobva.

**(C3.) 2. megoldás****(C3.) 2.1. lépés (A két legalacsonyabb részlegenkénti összfizetés nézettábla létrehozása)**

```
DROP VIEW SzegényRészleg;
```

```
CREATE OR REPLACE VIEW SzegényRészleg
(Sorszám, Deptno, RészlegÖsszeg, RészlegLétszám)
AS
SELECT ROWNUM,
       a1.*
  FROM (SELECT deptno,
               SUM(sal),
               COUNT(sal)
          FROM dolgozó
         GROUP BY deptno
        ORDER BY SUM(sal)) a1
 WHERE ROWNUM <= 2;
```

PROMPT SzegényRészleg:

```
SELECT * FROM SzegényRészleg;
```

**Eredmény**

A nézet eldobva.

A nézet létrejött.

SzegényRészleg:

SORSZÁM	DEPTNO	RÉSZLEGÖSSZEG	RÉSZLEGLÉTSZÁM
---------	--------	---------------	----------------

1	10	8750	3
2.	30	9400	6

**(C3.) 2.2. lépés (A növekmény meghatározása)**

```

CREATE OR REPLACE VIEW Növekmény
    (Érték)
AS
    SELECT ROUND((RészlegÖsszeg2 - RészlegÖsszeg1) / Létszáml) + 1
    FROM (SELECT RészlegÖsszeg AS RészlegÖsszeg2
            FROM SzegényRészleg
           WHERE Sorszám = 2),
         (SELECT RészlegÖsszeg AS RészlegÖsszeg1,
              RészlegLétszám AS Létszáml
            FROM SzegényRészleg
           WHERE Sorszám = 1);

```

PROMPT Növekmény:

```
SELECT * FROM Növekmény;
```

**Eredmény**

A nézet létrejött.

Növekmény:

```
ÉRTÉK
```

```
-----
```

```
218
```

**(C3.) 2.3. lépés (Feladat: az előírt fizetések emelése)**

```

UPDATE dolgozó
    SET sal = sal + (SELECT Érték FROM Növekmény)
    WHERE deptno = (SELECT deptno
                      FROM SzegényRészleg
                     WHERE Sorszám = 1);

```

PROMPT dolgozó:

```
SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
```

**Eredmény**

3 sor módosítva.

dolgozó:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5218		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30

7782	CLARK	MANAGER	7839	81-JÚN-09	2668	10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975	20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0
7900	JAMES	CLERK	7698	81-DEC-03	950	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500
7902	FORD	ANALYST	7566	81-DEC-03	3000	20
7369	SMITH	CLERK	7902	80-DEC-17	800	20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000	20
7876	ADAMS	CLERK	7788	83-JAN-12	1100	20
7934	MILLER	CLERK	7782	82-JAN-23	1518	10

14 sor kijelölve.

#### (C3.) 2.4. lépés (Ellenőrzés, táblaértékek visszaállítása, nézettábla törlése)

PROMPT SzegényRészleg:

SELECT \* FROM SzegényRészleg;

ROLLBACK;

DROP VIEW SzegényRészleg;

DROP VIEW Növekmény;

#### Eredmény

SORSZÁM	DEPTNO	RÉSZLEGÖSSZEG	RÉSZLEGLÉTSZÁM
1	30	9400	6
2	10	9404	3

A visszaállítás befejeződött.

A nézet eldobva..

A nézet eldobva..

## D. feladatsor (megoldások)

### D2. feladat

Munkakörönkénti csoportosításban és ismétlésmentes módon listázza ki a főnökök (mgr) nevét, munkakörét, telephelyét, közvetlen beosztottainak összfizetését, ez utóbbi szerint rendezve.

#### Előkészület a megoldáshoz (A munkatáblák törlése és létrehozása)

DROP TABLE dolgozó;

DROP TABLE részleg;

CREATE TABLE dolgozó

```

AS SELECT * FROM emp;
CREATE TABLE részleg
AS SELECT * FROM dept;

```

### (D2.) 1. megoldás (Rossz hatékonyságú)

```

BREAK ON Fmunkakor
SELECT d.mgr      AS FőnökAzonosító,
       f.ename    AS Fnév,
       f.job     AS Fmunkakor,
       fr.loc    AS Ftelephely,
       SUM(d.sal) AS BeosztottÖsszFiz
FROM dolgozó d,
     dolgozó f,
     részleg fr
WHERE d.mgr = f.empno AND
      f.deptno = fr.deptno
GROUP BY d.mgr, f.ename, f.job, fr.loc
ORDER BY Fmunkakor,BeosztottÖsszFiz;
CLEAR BREAKS

```

### (D2.) 2. megoldás

```

BREAK ON Fmunkakor
SELECT d.mgr      AS FőnökAzonosító,
       f.ename    AS Fnév,
       f.job     AS Fmunkakor,
       fr.loc    AS Ftelephely,
       BeosztottÖsszFiz
FROM dolgozó f,
     részleg fr,
     (SELECT mgr,
             SUM(sal)   AS BeosztottÖsszFiz
      FROM dolgozó
      GROUP BY mgr)  d
WHERE d.mgr = f.empno AND
      f.deptno= fr.deptno
ORDER BY Fmunkakor,BeosztottÖsszFiz;
CLEAR BREAKS

```

### Eredmény (mindkét megoldás esetén)

FŐNÖKAZONOSÍTÓ FNÉV	FMUNKAKOR	FTELEPHELY	BEOSZTOTTÖSSZFIZ.
7902 FORD	ANALYST	DALLAS	800
7788 SCOTT		DALLAS	1100
7782 CLARK	MANAGER	NEW YORK	1300

7566 JONES	DALLAS	6000
7698 BLAKE	CHICAGO	6550
7839 KING	PRESIDENT NEW YORK	8275

6. sor kijelölve.

breaks törölve



### Megjegyzés

- Az 1. megoldás GROUP BY utasításrészében logikailag egyetlen oszlop szerint történik a csoportosítás, bár ténylegesen négy oszlop szerepel. Ennek az az oka, hogy e négy oszlop által képviselt tulajdonság közül az empno egy egyedi azonosító, mely miatt a keletkező csoportok mindegyike csupán *egyetlen* sorból áll! Emiatt a további csoportosító tulajdonságok a keletkező felbontást már nem tudják finomítani. (A GROUP BY utasításrészben az oszlopok számának növelése általában finomítja a felbontást, azaz csökkenti egy-egy csoport elemeinek számát. Erre példa egy GROUP BY deptno kifejezés helyett a GROUP BY deptno, job kifejezés használata.)
- A két megoldás közül mindenkihez a 2. a megfelelőbb, két ok miatt is. Egyszerűtől a feladat megoldható allekérdezéssel, akkor általában az a hatékonyabb. (Ennek oka roppant egyszerű, kisebb táblával kell műveletet végezni.) A másik ok a csoportosítás elvégzésének módjában keresendő. Ha több szempont szerint kell csoportosítani, akkor ezt az (egyébként jelentős számítási erőforrást lekövü) műveletet az összes csoportosítási tulajdonság (oszlop) szerint el kell végezni akkor is, ha a felbontás finomsága nem változik.
- A fentiek összefoglalásaként azt mondhatjuk, hogy
  - soha ne végezzünk egyedi azonosítók szerinti csoportosítást,
  - lehetőség szerint használunk allekérdezéseket.

## D3. feladat

Vegyen fel egy új dolgozót a legkisebb összfizetésű közvetlen beosztottakkal rendelkező főnök-höz olyan fizetéssel, hogy az már éppen ne a legkisebb összfizetésű legyen. Listázással ellenőrizze a felvétel helyességét, végül állítsa vissza az eredeti táblatartalmakat.

### (D3.) Megoldás

#### (D3.) 1. lépés (Főnökönként a beosztottak összfizetése – nézet)

```
CREATE OR REPLACE VIEW FőnökNézet
(FőnökAzonosító, BeosztottÖsszfiz)
AS
SELECT mgr,
       SUM(sal)
  FROM dolgozó
 WHERE mgr IS NOT NULL
 GROUP BY mgr
```

```
ORDER BY SUM(sal);
SELECT * FROM FőnökNézet;
```

**Eredmény**

A nézet létrejött.

**FŐNÖKAZONOSÍTÓ BEOSZTOTTÖSSZFIZ**

7902	800
7788	1100
7782	1300
7566	6000
7698	6550
7839	8275

6 sor kijelölve.

**(D3.) 2. lépés (A két legkisebb összfizetésű beosztottakkal rendelkező főnök – nézet)**

```
CREATE OR REPLACE VIEW Legkisebb2Nézet
  (FőnökAzonosító, BeosztottÖsszfiz, Sorszám)
AS
  SELECT FőnökNézet.*,
         ROWNUM
    FROM FőnökNézet
   WHERE ROWNUM <= 2;
SELECT * FROM Legkisebb2Nézet;
```

**Eredmény**

A nézet létrejött.

FŐNÖKAZONOSÍTÓ BEOSZTOTTÖSSZFIZ		SORSZÁM
7902	800	1
7788	1100	2

**(D3.) 3. lépés (Az új dolgozó adatai)**

Az új dolgozó fizetése a második és az első legkisebb összfizetésű csoport összfizetésének különbsége +1, főnöke a legkisebb összfizetésű csoport főnöke, réslege tetszőleges, legyen 10.

```
INSERT INTO dolgozó
  (empno, ename, job,
   mgr,
   hiredate,
   sal,
   comm, deptno)
```

```

VALUES (2222, 'Gábor', 'Diák',
       (SELECT FőnökAzonosító
        FROM Legkisebb2Nézet
        WHERE Sorszám = 1),
       sysdate,
       (SELECT BeosztottÖsszfiz
        FROM Legkisebb2Nézet
        WHERE Sorszám = 2) -
       (SELECT BeosztottÖsszfiz
        FROM Legkisebb2Nézet
        WHERE Sorszám = 1) + 1,
       NULL, 10);

```

**Eredmény**

1 sor létrejött.

**(D3.) 4. lépés (Ellenőrzés, táblaértek visszaállítása, nézetek törlése)**

```
SELECT * FROM Legkisebb2Nézet;
```

```

SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10

ROLLBACK;
DROP VIEW FőnökNézet;
DROP VIEW Legkisebb2Nézet;

```

**Eredmény**

FŐNÖKAZONOSÍTÓ BEOSZTOTTÖSSZFIZ	SORSZÁM	
7788	1100	1
7902	1101	2

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	7839	81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FÉB-22	1250	500	30

7902 FORD	ANALYST	7566	81-DEC-03	3000	20
7369 SMITH	CLERK	7902	80-DEC-17	800	20
7788 SCOTT	ANALYST	7566	82-DEC-09	3000	20
7876 ADAMS	CLERK	7788	83-JAN-12	1100	20
7934 MILLER	CLERK	7782	82-JAN-23	1300	10
2222 Gábor	Diák	7902	04-NOV-21	301	10

15 sor kijelölve.

A visszaállítás befejeződött.

A nézet eldobva.

A nézet eldobva.

## D4. feladat

Listázza ki formázottan (fej- és lábléccel stb.) a felhasználó által megadott munkakörű főnökök nevét, munkakörét, telephelyét, valamint közvetlen beosztottainak nevét, fizetését és telephelyét a főnökök neve szerint rendezve.

### (D4.) Megoldás

#### (D4.) 1. lépés (Főnökök nézettáblája)

```

CREATE OR REPLACE VIEW FőnökNézet
  (FőnökNév, FőnökMunkakör, FőnökTelephely, FőnökAzonosító)
AS
  SELECT ename,
         job,
         loc,
         empno
    FROM dolgozó,
         részleg
   WHERE empno IN (SELECT mgr
                     FROM dolgozó) AND
         dolgozó.deptno = részleg.deptno;
  SELECT * FROM FőnökNézet;

```

### Eredmény

FŐNÖKNÉV	FŐNÖKMUNK	FŐNÖKTELEPHEL	FŐNÖKAZONOSÍTÓ
JONES	MANAGER	DALLAS	7566
BLAKE	MANAGER	CHICAGO	7698
CLARK	MANAGER	NEW YORK	7782
SCOTT	ANALYST	DALLAS	7788
KING	PRESIDENT	NEW YORK	7839
FORD	ANALYST	DALLAS	7902

6 sor kijelölve.

**Megjegyzés**

Itt nem volt szükség a főnökazonosítót visszaadó allekérdezésben kiszűrni a NULL értéket, mivel az empno IN... kifejezéssel halmazban való előfordulást vizsgáltunk.

**(D4.) 2. lépés (Közvetlen beosztottak nézettáblája)**

```
CREATE OR REPLACE VIEW BeosztottNézet
  (Dolgozónév, DolgozóFizetés, DolgozóTelephely, FőnökAzonosító)
AS
  SELECT ename,
         sal,
         loc,
         mgr
    FROM dolgozó,
         részleg
   WHERE mgr IN (SELECT mgr
                  FROM dolgozó) AND
         dolgozó.deptno = részleg.deptno;
  SELECT * FROM BeosztottNézet;
```

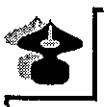
**Eredmény**

A nézet létrejött.

**DOLGOZÓNÉV DOLGOZÓFIZETÉS DOLGOZÓTELEPH FŐNÖKAZONOSÍTÓ**

SCOTT	3000	DALLAS	7566
FORD	3000	DALLAS	7566
MARTIN	1250	CHICAGO	7698
TURNER	1500	CHICAGO	7698
WARD	1250	CHICAGO	7698
JAMES	950	CHICAGO	7698
ALLEN	1600	CHICAGO	7698
MILLER	1300	NEW YORK	7782
ADAMS	1100	DALLAS	7788
CLARK	2450	NEW YORK	7839
BLAKE	2850	CHICAGO	7839
JONES	2975	DALLAS	7839
SMITH	800	DALLAS	7902

13 sor kijelölve.

**Megjegyzés**

Beosztottként King nem szerepel, mivel neki nincs főnöke.

**(D4.) 3. lépés (A formázott kiíratás listázási magja: a feladat által igényelt oszlopok formázás nélküli listázása)**

```

ACCEPT Fmunkakör PROMPT 'Kérem a főnök munkakörét: '
SELECT FőnökNév          AS Fneve,
       FőnökMunkakör    AS Ffogl,
       FőnökTelephely   AS Fhely,
       DolgozóNév       AS Dneve,
       DolgozóFizetés   AS Dfiz,
       DolgozóTelephely AS Dhely
  FROM FőnökNézet,
       BeosztottNézet
 WHERE FőnökNézet.FőnökAzonosító = BeosztottNézet.FőnökAzonosító AND
       UPPER(FőnökNézet.FőnökMunkakör) = UPPER('&Fmunkakör')
 ORDER BY Fneve;

```

**(D4.) 3.\* lépés (A formázott kiíratás listázási magja – nézettáblák használata nélkül)**

```

ACCEPT Fmunkakör PRÓMPT 'Kérem a főnök munkakörét: '
SELECT f.ename           AS Fneve,
       f.job             AS Ffogl,
       fr.loc            AS Fhely,
       d.ename           AS Dneve,
       d.sal             AS Dfiz,
       dr.loc            AS Dhely
  FROM dolgozó d,
       dolgozó f,
       részleg dr,
       részleg fr
 WHERE UPPER(f.job) = UPPER('&Fmunkakör') AND
       d.mgr      = f.empno          AND
       f.deptno   = fr.deptno        AND
       d.deptno   = dr.deptno
 ORDER BY Fneve;

```

**Eredmény (mindkét listázási mag esetén)**

Kérem a főnök munkakörét: manager

régi 10:	UPPER(FőnökNézet.FőnökMunkakör) = UPPER('&Fmunkakör')
új 10:	UPPER(FőnökNézet.FőnökMunkakör) = UPPER('manager')

FNEVE	FFOGL	FHELY	DNEVE	DFIZ	DHELY
BLAKE	MANAGER	CHICAGO	MARTIN	1250	CHICAGO
BLAKE	MANAGER	CHICAGO	WARD	1250	CHICAGO
BLAKE	MANAGER	CHICAGO	ALLEN	1600	CHICAGO
BLAKE	MANAGER	CHICAGO	JAMES	950	CHICAGO

BLAKE	MANAGER	CHICAGO	TURNER	1500 CHICAGO
CLARK	MANAGER	NEW YORK	MILLER	1300 NEW YORK
JONES	MANAGER	DALLAS	SCOTT	3000 DALLAS
JONES	MANAGER	DALLAS	FORD	3000 DALLAS

8 sor kijelölve.



### Megjegyzés

- A formázott kiíratás listázási magjának másodlagos oszlopneveit célszerű úgy megtervezni, hogy egyszerű rövidek legyenek (hiszen úgyse ezek kerülnek a fejlécre), másrészt NE tartalmazzanak ékezetes betűket, mivel az SQL\*Plus formázási utasításai (de csak ezek!) „nem szeretik” az ékezetes betűket (egyszerűen nem hajtódnak végre!).
- Ebben a listában főnökként nem szerepel Ford, King és Scott, mivel ők nem „manager”-ek.
- A fenti két listázási mag közül a nézettáblákat nem használó megoldás látszik megfelelőbbnek elsősorban az egyszerűsége miatt. A látszat azonban csalóka. Egyszerű megfelelő nézettáblák használata révén könnyebben tudunk egy összetett feladatot megoldani, mint ha „egy lépéssben” próbálkozunk vele. Ezenkívül a gyakorlat számára nem kevésbé lényeges szempont az is, hogy a nézettáblákban szereplő alkérdezések nem csupán „didaktikusabbak”, hanem lényegesen hatékonyabbak is az egyetlen alkérdezés WHERE utasításrészében történő összekapcsolási feltételekkel való kiválasztásnál.

### (D4.) 4. lépés (A megoldás formázott kiírása, nézettáblák törlése)

ACCEPT Fmunkakör PROMPT 'Kérem a főnök munkakörét: '

```
SET linesize 70
SET pagesize 22
SET verify OFF
SET feedback OFF
```

```
BREAK ON Dhely  ON Fneve  ON Ffogl  ON Fhely
TTITLE "A &Fmunkakör munkakörű főnökök|adatbázisa"
BTITLE "Készítették:|az Oracle példatár|szerzői"
COLUMN Fneve HEADING "Főnök|neve"      FORMAT A8   JUSTIFY LEFT
COLUMN Ffogl  HEADING "munkaköre"        FORMAT A10  JUSTIFY CENTER
COLUMN Fhely  HEADING "telephelye"       FORMAT A10  JUSTIFY CENTER
COLUMN Dneve  HEADING "Beosztott|neve"    FORMAT A10  JUSTIFY LEFT
COLUMN Dfiz   HEADING "fizetése"         FORMAT $0999 JUSTIFY RIGHT
COLUMN Dhely  HEADING "telephelye"       FORMAT A10  JUSTIFY CENTER
```

```
SELECT FőnökNév          AS Fneve,
       FőnökMunkakör    AS Ffogl,
```

```

FőnökTelephely AS Phely,
DolgozóNév AS Dneve,
DolgozóFizetés AS Dfiz,
DolgozóTelephely AS Dhely
FROM FőnökNézet,
BeosztottNézet
WHERE FőnökNézet.FőnökAzonosító = BeosztottNézet.FőnökAzonosító AND
      UPPER(FőnökNézet.FőnökMunkakör) = UPPER('&Fmunkakör')
ORDER BY Fneve;

DROP VIEW FőnökNézet;
DROP VIEW BeosztottNézet;
CLEAR BREAKS
CLEAR COLUMNS
TTITLE OFF
BTITLE OFF
SET linesize 400
SET pagesize 40
SET verify ON
SET feedback ON

```

### Eredmény

Kérem a főnök munkakörét: manager

V. Nov 21

lap 1

#### A manager munkakörű főnökök adatbázisa

Főnök neve	munkaköre	telephelye	Beosztott neve	fizetése	telephelye
BLAKE	MANAGER	CHICAGO	MARTIN	\$1250	CHICAGO
			WARD	\$1250	
			ALLEN	\$1600	
			JAMES	\$0950	
			TURNER	\$1500	
CLARK	MANAGER	NEW YORK	MILLER	\$1300	NEW YORK
JONES	MANAGER	DALLAS	SCOTT	\$3000	DALLAS
			FORD	\$3000	

Készítették:  
az Oracle példatár  
szerzői

## H. feladatsor (megoldások)

### H2. feladat

Készítsen egy Intézetek nevű nézettáblát, mely megadja az igazgatók nevét, azonosítóját, valamint a közvetlen és közvetett beosztottai nevét, azonosítóját és fizetését, ahol igazgatóknak nevezük az elnök (president) közvetlen beosztottait (vagyis az Intézeteket az igazgatóik azonosítják). Listázza ki e nézettáblát az igazgatók neve szerint rendezve.

#### Előkészület a megoldáshoz (A munkatáblák törlése és létrehozása)

```
DROP TABLE dolgozó;
DROP TABLE részleg;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
CREATE TABLE részleg
AS SELECT * FROM dept;
```

```
SET numwidth 5;
```

#### (H2.) Megoldás

PROMPT Az igazgatók:

```
CREATE OR REPLACE VIEW Igazgató
AS
SELECT ename,
       empno,
       sal
  FROM dolgozó
 WHERE mgr = (SELECT empno
               FROM dolgozó
              WHERE UPPER(job) = 'PRESIDENT');
SELECT *
  FROM Igazgató
 ORDER BY ename;
```

PROMPT Az igazgatók közvetlen beosztottai:

```
CREATE OR REPLACE VIEW KözvetlenBeo
AS
SELECT IG.empno AS IGempno,
       IG.ename AS IGename,
       D.empno,
       D.ename,
       D.sal
  FROM dolgozó D,
       Igazgató IG
```

```

    WHERE D.mgr IN IG.empno;
SELECT *
  FROM KözvetlenBeo
 ORDER BY IGename, ename;

```

PROMPT Az igazgatók közvetett beosztottai:

```

CREATE OR REPLACE VIEW KözvetettBeo
AS

```

```

SELECT IG.empno AS IGempno,
       IG.ename AS IGename,
       D.empno,
       D.ename,
       D.sal
  FROM dolgozó      D,
       Igazgató      IG,
       KözvetlenBeo KB
 WHERE D.mgr IN KB.empno AND
       KB.IGempno = IG.empno;

```

```

SELECT *
  FROM KözvetettBeo
 ORDER BY IGename, ename;

```

PROMPT Az Intézetek igazgatói és beosztottaik:

-- 1.megoldás

```

CREATE OR REPLACE VIEW Intézetek
AS

```

```

SELECT IG.ename AS IGnév,
       IG.empno AS IGazonosító,
       IG.sal   AS IGfiz,
       B.ename  AS DolgNév,
       B.empno  AS DolgAzonosító,
       B.sal    AS DolgFiz
  FROM Igazgató  IG
 INNER JOIN
 ((SELECT * FROM KözvetlenBeo)
 UNION
 (SELECT * FROM KözvetettBeo)) B
 ON B.IGempno = IG.empno;

```

```

SELECT *
  FROM Intézetek
 ORDER BY IGnév, DolgNév;

```

```

PROMPT Az Intézetek igazgatói és beosztottaik:
-- 2.megoldás
CREATE OR REPLACE VIEW Intézetek
AS
  SELECT IG.ename    AS IGnév,
         IG.empno   AS IGazonosító,
         IG.sal     AS IGfiz,
         B.ename    AS DolgNév,
         B.empno    AS DolgAzonosító,
         B.sal      AS DolgFiz
    FROM Igazgató      IG,
         Dolgozó       B
   WHERE B.empno IN (SELECT empno
                      FROM KözvetlenBeo  KnB
                     WHERE KnB.IGempno = IG.empno)  OR
         B.empno IN (SELECT empno
                      FROM KözvetettBeo  KtB
                     WHERE KtB.IGempno = IG.empno);

SELECT *
  FROM Intézetek
 ORDER BY IGnév, DolgNév;

```

### Eredmény

Az igazgatók:

A nézet létrejött.

ENAME	EMPNO	SAL
BLAKE	7698	2850
CLARK	7782	2450
JONES	7566	2975

Az igazgatók közvetlen beosztottai:

A nézet létrejött.

IGEMPNO	IGENAME	EMPNO	ENAME	SAL
7698	BLAKE	7499	ALLEN	1600
7698	BLAKE	7900	JAMES	950
7698	BLAKE	7654	MARTIN	1250
7698	BLAKE	7844	TURNER	1500
7698	BLAKE	7521	WARD	1250
7782	CLARK	7934	MILLER	1300
7566	JONES	7902	FORD	3000
7566	JONES	7788	SCOTT	3000

8 sor kijelölve.

Az igazgatók közvetett beosztottai:

A nézet létrejött.

IGEMPNO	IGENAME	EMPNO	ENAME	SAL
7566	JONES	7876	ADAMS	1100
7566	JONES	7369	SMITH	800

Az Intézetek igazgatói és beosztottaik:

A nézet létrejött.

IGNÉV	IGAZONOSÍTÓ	IGFIZ	DOLGNÉV	DOLGAZONOSÍTÓ	DOLGFIZ
BLAKE	7698	2850	ALLEN	7499	1600
BLAKE	7698	2850	JAMES	7900	950
BLAKE	7698	2850	MARTIN	7654	1250
BLAKE	7698	2850	TURNER	7844	1500
BLAKE	7698	2850	WARD	7521	1250
CLARK	7782	2450	MILLER	7934	1300
JONES	7566	2975	ADAMS	7876	1100
JONES	7566	2975	FORD	7902	3000
JONES	7566	2975	SCOTT	7788	3000
JONES	7566	2975	SMITH	7369	800

10 sor kijelölve.

### H3. feladat

Növelje meg az egynél több beosztottat foglalkoztató, legkisebb összfizetésű Intézet beosztott dolgozói mindegyikének fizetését annyival, hogy ez az Intézet már éppen ne a legkisebb összfizetésű legyen. Az adatmódosítást az Intézetek nézettáblán keresztül végezze, melyet ellenőrzésképpen végül listázzon ki.

#### (H3.) Megoldás

PROMPT Az Intézetek összfizetései és beosztottainak létszámai:

CREATE OR REPLACE VIEW IntézetFizLétszám

AS

```

SELECT IGazonosító,
       SUM(DolgFiz)   AS ÖsszFiz,
       COUNT(*)        AS DolgLétszám
  FROM Intézetek
 GROUP BY IGazonosító;

```

```

SELECT *
  FROM IntézetFizLétszám
 ORDER BY IGazonosító;

```

PROMPT A két legkisebb összfizetésű,  
PROMPT egynél több beosztottat foglalkoztató Intézet:  
CREATE OR REPLACE VIEW SzegényIntézet  
AS  
SELECT al.\*,  
 ROWNUM AS Sorszám  
 FROM (SELECT \*  
 FROM IntézetFizLétszám  
 ORDER BY ÖsszFiz) al  
 WHERE DolgLétszám >1 AND  
 ROWNUM <= 2;  
SELECT \*  
 FROM SzegényIntézet  
 ORDER BY Sorszám;

PROMPT Fizetésemelés:

UPDATE dolgozó  
 SET sal = sal + 1 +  
 ROUND(((SELECT ÖsszFiz  
 FROM SzegényIntézet  
 WHERE Sorszám = 2) -  
 (SELECT ÖsszFiz  
 FROM SzegényIntézet  
 WHERE Sorszám = 1)) /  
 (SELECT DolgLétszám  
 FROM SzegényIntézet  
 WHERE Sorszám = 1))  
 WHERE empno IN (SELECT DolgAzonosító  
 FROM Intézetek,  
 SzegényIntézet  
 WHERE Intézetek.IGazonosító =  
 SzegényIntézet.IGazonosító AND  
 Sorszám = 1);

PROMPT A két legkisebb összfizetésű,  
PROMPT egynél több beosztottat foglalkoztató Intézet:  
SELECT \*  
 FROM SzegényIntézet  
 ORDER BY Sorszám;

PROMPT Az Intézetek (ellenőrzés):

SELECT \*  
 FROM Intézetek  
 ORDER BY IGNév, DolgNév;

**Eredmény**

Az Intézetek összfizetései és beosztottainak létszámai:

A nézet létrejött.

**IGAZONOSÍTÓ ÖSSZFIZ DOLGLÉTSZÁM**

7566	7900	4
7698	6550	5
7782	1300	1

A két legkisebb összfizetésű,

egynél több beosztottat foglalkoztató Intézet:

A nézet létrejött.

**IGAZONOSÍTÓ ÖSSZFIZ DOLGLÉTSZÁM SORSZÁM**

7698	6550	5	1
7566	7900	4	2

Fizetésemelés:

5 sor módosítva.

A két legkisebb összfizetésű,

egynél több beosztottat foglalkoztató Intézet:

**IGAZONOSÍTÓ ÖSSZFIZ DOLGLÉTSZÁM SORSZÁM**

7566	7900	4	1
7698	7905	5	2

Az Intézetek (ellenőrzés):

IGNÉV	IGAZONOSÍTÓ	IGFIZ	DOLGNÉV	DOLGAZONOSÍTÓ	DOLGFIZ
BLAKE	7698	2850	ALLEN	7499	1871
BLAKE	7698	2850	JAMES	7900	1221
BLAKE	7698	2850	MARTIN	7654	1521
BLAKE	7698	2850	TURNER	7844	1771
BLAKE	7698	2850	WARD	7521	1521
CLARK	7782	2450	MILLER	7934	1300
JONES	7566	2975	ADAMS	7876	1100
JONES	7566	2975	FORD	7902	3000
JONES	7566	2975	SCOTT	7788	3000
JONES	7566	2975	SMITH	7369	800

10 sor kijelölve.

**Megjegyzés**

Az adatmódosítást nem lehet az Intézetek nézettáblán keresztül végezni, mivel ez egy többtáblás nézet. Nem is lehetne egytáblás, mivel soraiban kifejezésekkel származó oszlopokat is tartalmaz (az igazgatók nevét és azonosítóját).

## H4. feladat

Listázza ki formázottan (fej- és lábléccel stb.) és ismétlésmentesen a felhasználó által megadott igazgató nevét és fizetését, valamint a beosztottainak nevét, fizetését, részlegük telephelyét és annak összfizetését, valamint az abban dolgozó főnökök (mgr) nevét és fizetését, a dolgozók neve szerint rendezve. Használja az Intézetek nézettáblát.

### (H4.) Megoldás

#### (H4.) 1. lépés (A formázott kiíratás listázási magja: a feladat által igényelt oszlopok formázás nélküli listázása)

```
SET verify OFF
ACCEPT IgazgatóNeve PROMPT "Az igazgató neve: "
```

```
SELECT IGnév           AS IG_neve,
       IGfiz            AS IG_fiz,
       DolgNév          AS D_neve,
       DolgFiz          AS D_fiz,
       loc              AS R_hely,
       összfiz          AS R_fiz
  FROM Intézetek
 INNER JOIN
       dolgozó  DD
      ON (DD.empno = DolgAzonosító)
 INNER JOIN
       (SELECT D.deptno,
               loc,
               SUM(sal)  AS összfiz
      FROM dolgozó  D,
           részleg  R
     WHERE D.deptno = R.deptno
     GROUP BY D.deptno, loc)  RR
      ON (DD.deptno = RR.deptno)
 WHERE UPPER(IGnév) = UPPER('&IgazgatóNeve');
```

### Eredmény

Az igazgató neve: Blake

IG_NEVE	IG_FIZ	D_NEVE	D_FIZ	R_HELY	R_FIZ
BLAKE	2850	ALLEN	1871	CHICAGO	10755
BLAKE	2850	WARD	1521	CHICAGO	10755
BLAKE	2850	MARTIN	1521	CHICAGO	10755
BLAKE	2850	TURNER	1771	CHICAGO	10755
BLAKE	2850	JAMES	1221	CHICAGO	10755

## (H4.) 2. lépés (A megoldás formázott kiírása, a nézettáblák törlése)

```

SET numwidth 10
SET verify OFF
ACCEPT IgazgatóNeve PROMPT "Az igazgató neve: "

SET linesize 65
SET pagesize 17
TTITLE ON
BTITLE ON
TTITLE '&IgazgatóNeve beosztottainak listázása'
BTITLE 'Készítették: | E könyv szerzői'
COLUMN IG_neve FORMAT A8          HEADING 'Igazgató|neve'      -
                JUSTIFY LEFT
COLUMN IG_fiz  FORMAT $99,999    HEADING 'Igazgató|fizetése'   -
                JUSTIFY RIGHT
COLUMN D_neve  FORMAT A8          HEADING 'Dolgozó|neve'     -
                JUSTIFY LEFT
COLUMN D_fiz   FORMAT $99,999    HEADING 'Dolgozó|fizetése'   -
                JUSTIFY RIGHT
COLUMN R_hely   FORMAT A15        HEADING 'Dolgozó|telephelye' -
                JUSTIFY LEFT
COLUMN R_fiz   FORMAT $99,999    HEADING 'Részleg|összfizetése' -
                JUSTIFY RIGHT
BREAK ON IG_neve  ON IG_fiz  ON R_hely

SELECT IGnév           AS IG_neve,
       IGfiz            AS IG_fiz,
       DolgNév          AS D_neve,
       DolgFiz          AS D_fiz,
       loc              AS R_hely,
       összfiz          AS R_fiz
FROM Intézetek
INNER JOIN
       dolgozó  DD
ON (DD.empno = DolgAzonosító)
INNER JOIN

```

```

        (SELECT D.deptno,
               loc,
               SUM(sal) AS összfiz
        FROM dolgozó D,
             részleg R
       WHERE D.deptno = R.deptno
     GROUP BY D.deptno, loc) RR
    ON (DD.deptno = RR.deptno)
 WHERE UPPER(IGNÉV) = UPPER('&IgazgatóNeve');

```

```

TTITLE OFF
BTITLE OFF
CLEAR BREAKS
CLEAR COLUMNS
SET verify ON
UNDEFINE IgazgatóNeve

```

```

DROP VIEW Igazgató;
DROP VIEW KözvetlenBeo;
DROP VIEW KözvetettBeo;
DROP VIEW Igazgató;
DROP VIEW Intézetek;
DROP VIEW IntézetFizLétszám;
DROP VIEW SzegényIntézet;

```

### Eredmény

Az igazgató neve: Blake

V. Dec 05

lap 1

Blake beosztottainak  
listázása

Igazgató neve	Igazgató fizetése	Dolgozó neve	Dolgozó fizetése	Részleg telephelye	Részleg összfizetése
BLAKE	\$2,850	ALLEN	\$1,871	CHICAGO	\$10,755
		WARD	\$1,521		\$10,755
		MARTIN	\$1,521		\$10,755
		TURNER	\$1,771		\$10,755
		JAMES	\$1,221		\$10,755

Készítették:  
E könyv szerzői

A nézet eldobva.  
 A nézet eldobva.

## I. feladatsor (megoldások)

### I2. feladat

Készítsen egy NagyFőnökök nevű nézettáblát, mely megadja az elnök (president) közvetlen beosztottainak adatait, majd listázza ki a nagyfőnökök neve szerint rendezve.

#### Előkészület a megoldáshoz (A munkatáblák törlése és létrehozása)

```
DROP TABLE dolgozó;
DROP TABLE részleg;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
CREATE TABLE részleg
AS SELECT * FROM dept;
```

#### (I2.) Megoldás

```
CREATE OR REPLACE VIEW NagyFőnökök
AS
  SELECT *
    FROM dolgozó
   WHERE mgr = (SELECT empno
                  FROM dolgozó
                 WHERE UPPER(job) = 'PRESIDENT');

SET numwidth 5
SELECT *
   FROM NagyFőnökök
  ORDER BY ename;
SET numwidth 10
```

#### Eredmény

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20

## I3. feladat

Vegyen fel egy új főnököt, akinek főnöke az elnök (president), fizetése a nagyfőnökök átlagos fizetése, részlege pedig a legnagyobb fizetésű nagyfőnökével egyező, a többi adat tetszőleges. Az adatfelvitelt a NagyFőnökök nézettáblán keresztül végezze, melyet végül listázzon ki.

### (I3.) Megoldás

PROMPT Új főnök adatfelvitele:

```
INSERT INTO NagyFőnökök
VALUES(7777,
      'OKOSKA',
      'MANAGER',
      (SELECT empno
       FROM NagyFőnökök
       WHERE UPPER(job) = 'PRESIDENT'),
      sysdate,
      (SELECT AVG(sal)
       FROM NagyFőnökök),
      NULL,
      (SELECT deptno
       FROM NagyFőnökök
       WHERE sal = (SELECT MAX(sal)
                     FROM NagyFőnökök)));
```

PROMPT Ellenőrző lekérdezés:

```
SET numwidth 5
SELECT *
  FROM NagyFőnökök
 ORDER BY ename;
SET numwidth 10
```

### Eredmény

Új főnök adatfelvitele:  
1 sor létrejött.

Ellenőrző lekérdezés:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7777	OKOSKA	MANAGER	7839	04-NOV-26	2758		20

## 14. feladat

Listázza ki formázottan (fej- és lábléccel stb.) és ismétlésmentesen a dolgozók neve szerint rendezve a felhasználó által megadott nevű nagyfőnök nevét, fizetését, közvetlen beosztottainak nevét, fizetését, munkakörét és részlegének nevét. Használja a NagyFőnökök nézettáblát.

### (14.) Megoldás

#### (14.) 1. lépés (A formázatlan lista)

```

ACCEPT nagyfőnök PROMPT 'A NagyFőnök neve:
SELECT NF.ename      AS NF_ename,
       NF.sal        AS NF_sal,
       D.ename      AS D_ename,
       D.job        AS D_job,
       D.sal        AS D_sal,
       R.dname      AS R_dname
  FROM dolgozó      D,
       részleg      R,
       NagyFőnökök NF
 WHERE D.deptno =R.deptno    AND
       D.mgr IN NF.empno     AND
       UPPER(NF.ename) = UPPER('&nagyfőnök');
UNDEFINE nagyfőnök

```

#### Eredmény

```

A NagyFőnök neve: Jones
régi 12:          UPPER(NF.ename) = UPPER('&nagyfőnök')
új 12:            UPPER(NF.ename) = UPPER('Jones')

```

NF_ENAME	NF_SAL	D_ENAME	D_JOB	D_SAL	R_DNAME
JONES	2975	FORD	ANALYST	3000	RESEARCH
JONES	2975	SCOTT	ANALYST	3000	RESEARCH

#### (14.) 2. lépés (A formázott lista)

```
ACCEPT nagyfőnök PROMPT 'A NagyFőnök neve:
```

```

SET verify OFF
SET linesize 65
SET pagesize 17
SET feedback OFF
TTITLE '&nagyfőnök beosztottainak|listája'
BTITLE 'Készítették|az Oracle példatár szerzői'
COLUMN NF_ename HEADING 'A nagyfőnök|neve'           FORMAT A11
COLUMN NF_sal   HEADING 'fizetése'                   FORMAT 9999

```

```

COLUMN D_ename  HEADING 'A közvetlen|beosztott |neve'  FORMAT A12
COLUMN D_job    HEADING 'munkaköre'                  FORMAT A9
COLUMN D_sal    HEADING 'fizetése'                   FORMAT 9999
COLUMN R_dname  HEADING 'részlegének|neve'          FORMAT A12
BREAK ON NF_ename  ON NF_sal

SELECT NF.ename   AS NF_ename,
       NF.sal     AS NF_sal,
       D.ename    AS D_ename,
       D.job     AS D_job,
       D.sal     AS D_sal,
       R.dname   AS R_dname
  FROM dolgozó      D,
       részleg     R,
       NagyFőnökök NF
 WHERE D.deptno =R.deptno    AND
       D.mgr IN NF.empno      AND
       UPPER(NF.ename) = UPPER('&nagyfőnök');

SET verify ON
SET linesize 400
SET pagesize 40
SET feedback ON
TTITLE OFF
BTITLE OFF
CLEAR BREAKS
CLEAR COLUMNS
UNDEFINE nagyfőnök

```

**Eredmény**

A NagyFőnök neve: Blake

P. Nov 26

lap 1

Blake beosztottainak  
listája

A nagyfőnök neve	A közvetlen fizetése	beosztott neve	munkaköre	részlegének fizetése	neve
BLAKE	2850	MARTIN	SALESMAN	1250	SALES
		ALLEN	SALESMAN	1600	SALES
		JAMES	CLERK	950	SALES
		WARD	SALESMAN	1250	SALES
		TURNER	SALESMAN	1500	SALES

Készítették  
az Oracle példatár szerzői

## J. feladatsor (megoldások)

### J3. feladat

Növelje meg a nagyfőnökök mindegyikének fizetését a közvetlen beosztottai átlagfizetésének 10%-ával. Az adatmódosítást a NagyFőnökök nézettáblán keresztül végezze. Végül listázza ki a NagyFőnökök nézettáblát a nevek szerint rendezve.

#### Előkészület a megoldáshoz (A munkatáblák törlése és létrehozása)

```
DROP TABLE dolgozó;
DROP TABLE részleg;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
CREATE TABLE részleg
AS SELECT * FROM dept;
```

#### (J3.) Megoldás

```
CREATE OR REPLACE VIEW NagyFőnökök
AS
SELECT *
FROM dolgozó
WHERE mgr = (SELECT empno
              FROM dolgozó
              WHERE UPPER(job) = 'PRESIDENT');
```

```
CREATE OR REPLACE VIEW Növekmény
AS
SELECT NF.empno          AS Mgr_ID,
       NF.ename           AS Mgr_ename,
       0.1 * AVG(KB.sal) AS Mgr_Inc
  FROM dolgozó      KB,
       NagyFőnökök  NF
 WHERE KB.mgr = NF.empno
 GROUP BY NF.empno, NF.ename;
```

```
PROMPT ELLENŐRZÉS_1:
SET numwidth 5
PROMPT A NagyFőnökök (az eredeti fizetéssel):
SELECT *
  FROM NagyFőnökök
 ORDER BY ename;
```

```

--  

PROMPT Közvetlen beosztottak:  

SELECT NF.ename AS NF_ename,  

       KB.ename AS KB_ename,  

       KB.sal AS KB_sal  

  FROM dolgozó KB,  

       NagyFőnökök NF  

 WHERE KB.mgr = NF.empno  

 ORDER BY NF.ename;  

--  

PROMPT A NagyFőnökök fizetésnövekményei:  

SELECT *  

  FROM Növekmény  

 ORDER BY Mgr_ename;  

PROMPT A NagyFőnökök fizetésnövelése:  

UPDATE NagyFőnökök  

  SET sal = sal + (SELECT Mgr_Inc  

                      FROM Növekmény  

                     WHERE NagyFőnökök.empno = Növekmény.Mgr_ID);  

PROMPT ELLENŐRZÉS_2:  

PROMPT A NagyFőnökök (az új fizetéssel):  

SELECT *  

  FROM NagyFőnökök  

 ORDER BY ename;  

SET numwidth 10  

DROP VIEW NagyFőnökök;  

DROP VIEW Növekmény;

```

**Eredmény**

- A tábla eldobva.
- A tábla létrejött.
- A nézet létrejött.
- A nézet létrejött.

A NagyFőnökök (az eredeti fizetéssel):

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20

## ELLENŐRZÉS\_1:

Közvetlen beosztottak:

NF_ENAME	KB_ENAME	KB_SAL
BLAKE	MARTIN	1250
BLAKE	ALLEN	1600
BLAKE	JAMES	950
BLAKE	WARD	1250
BLAKE	TURNER	1500
CLARK	MILLER	1300
JONES	FORD	3000
JONES	SCOTT	3000

8 sor kijelölve.

A NagyFőnökök fizetésnövekményei:

MGR_ID	MGR_ENAME	MGR_INC
--------	-----------	---------

7698	BLAKE	131
7782	CLARK	130
7566	JONES	300

A NagyFőnökök fizetésnövelése:

3 sor módosítva.

## ELLENŐRZÉS\_2:

A NagyFőnökök (az új fizetéssel):

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2981		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2580		10
7566	JONES	MANAGER	7839	81-ÁPR-02	3275		20

A nézet eldobva.

A nézet eldobva.

## 8. FEJEZET

# Változóhasználat, vezérlési szerkezetek a PL/SQL-ben

## Feladatok és megoldások

### 8.1. feladat

Határozza meg egy PL/SQL program segítségével a felhasználó által megadott telephelyen dolgozók bérösszegét.

#### Megoldás

```
SET serveroutput ON
```

```
ACCEPT részleg PROMPT 'A részleg neve:
```

```
DECLARE
    v_össz_bér    NUMBER;
BEGIN
    SELECT SUM(sal)
    INTO v_össz_bér
    FROM emp, dept
    WHERE UPPER(loc) = UPPER('&részleg') AND
          emp.deptno = dept.deptno;
    DBMS_OUTPUT.PUT_LINE('A(z) '||'&részleg'||'
                           ' részlegen dolgozók havi bér összesen: '|||
                           v_össz_bér);
END;
/
```

#### Eredmény

```
A részleg neve: Dallas
régi 7:      WHERE UPPER(loc) = UPPER('&részleg') AND
új 7:      WHERE UPPER(loc) = UPPER('Dallas') AND
régi 10:     DBMS_OUTPUT.PUT_LINE('A(z) '||'&részleg'|||
új 10:     DBMS_OUTPUT.PUT_LINE('A(z) '||'Dallas'|||
```

A(z) Dallas részlegen dolgozók havi béré összesen: 10875

A PL/SQL eljárás sikeresen befejeződött.



### Megjegyzés

E feladat további megoldásaira a következő fejezetben visszatérünk (lásd a 9.1. feladat megoldását is).

## 8.2. feladat

Írjon PL/SQL programot, amely meghatározza, hogy a felhasználó által megadott nevű és korú személy hány év múlva lesz, illetve hány éve már nagykorú. A kiírást valósítsa meg a PL/SQL blokkon belül, és az SQL\*Plus-környezetben is.

### Megoldás

```
-- Az utasítások kiíratásának engedélyezése/letiltása
Rem SET echo ON
SET echo OFF
Rem A kiíratás engedélyezése:
SET serveroutput ON
Rem Az SQL*Plus ellenőrzések letiltása
SET verify OFF

Rem SQL*Plus környezeti változók:
VARIABLE Delta1      NUMBER
VARIABLE NeveKora    VARCHAR2(30)

Rem SQL*Plus helyettesítő változók:
ACCEPT Neve  PROMPT 'Neve: '
ACCEPT Kora   PROMPT 'Kora: '

Rem Az alábbi sortól kezdve a szkript egy PL/SQL-blokk,
Rem melyben SQL*Plus utasítások már nem adhatók ki.
DECLARE          -- PL/SQL-blokkváltozók deklarálása
  NagyNeve    VARCHAR2(20);
  SzamKora    NUMBER;
  Delta2      NUMBER;

BEGIN
  --:NeveKora := '&Neve (' || '&Kora' || ')';
  :NeveKora := '&Neve (&Kora)';
  NagyNeve := UPPER(:NeveKora);
  SzamKora := TO_NUMBER('&Kora');
  :Delta1 := 18 - SzamKora;
```

```

Delta2 := SzamKora - 18;

DBMS_OUTPUT.PUT_LINE('-----');
IF SzamKora < 18
THEN
  DBMS_OUTPUT.PUT_LINE(NagyNeve || ' nagykorú lesz '|| :Delta1 ||
                       ' év múlva');
ELSE
  DBMS_OUTPUT.PUT_LINE(NagyNeve || ' már '|| Delta2 ||
                       ' éve nagykorú');
END IF;
DBMS_OUTPUT.PUT_LINE('-----');
END;
/

```

Rem Innentől ismét SQL\*Plus környezetben vagyunk

```

Print Delta1
Print NeveKora

```

### 1. futás eredménye

```

Neve: Kelemen
Kora: 45
-----
```

```
KELEMEN (45) már 27 éve nagykorú
```

```
-----  
A PL/SQL eljárás sikeresen befejeződött.
```

```
DELTA1
```

```
-----  
-27
```

```
NEVEKORA
```

```
-----  
Kelemen (45)
```

### 2. futás eredménye

```

Neve: Molnár
Kora: 13
-----
```

```
MOLNÁR (13) nagykorú lesz 5 év múlva
```

```
-----  
A PL/SQL eljárás sikeresen befejeződött.
```

DELTA1

---

5

NEVEKORA

---

Molnár (13)

### 8.3. feladat

Állítsa elő a felhasználó által megadott darabszámig a Fibonacci-sorozat elemeit (0, 1, 1, 2, 3, 5,...). A megoldáshoz LOOP-ciklust használjon.

#### Megoldás

```
SET serveroutput ON
```

```
SET echo OFF
```

```
SET verify OFF
```

```
ACCEPT darabszám PROMPT "Kérem az előállítandó darabszámot: "
```

```
DECLARE
```

```
első NUMBER;
```

```
második NUMBER;
```

```
új NUMBER;
```

```
darab NUMBER;
```

```
számlál NUMBER;
```

```
BEGIN
```

```
darab := &darabszám;
```

```
első := 0;
```

```
második := 1;
```

```
számlál := 2;
```

```
DBMS_OUTPUT.PUT_LINE('A(z)' || LPAD(1,3) || '. elem:' ||  
LPAD(első,6));
```

```
DBMS_OUTPUT.PUT_LINE('A(z)' || LPAD(2,3) || '. elem:' ||  
LPAD(második,6));
```

```
LOOP
```

```
új := első + második;
```

```
számlál := számlál + 1;
```

```
DBMS_OUTPUT.PUT_LINE('A(z)' || LPAD(számlál,3) || '. elem:' ||  
LPAD(új,6));
```

```
EXIT WHEN számlál = darab;
```

```
első := második;
```

```
második := új;
```

```
END LOOP;
```

```
end;
/
```

### Eredmény

Kérem az előállítandó darabszámot: 11

```
A(z) 1. elem:    0
A(z) 2. elem:    1
A(z) 3. elem:    1
A(z) 4. elem:    2
A(z) 5. elem:    3
A(z) 6. elem:    5
A(z) 7. elem:    8
A(z) 8. elem:   13
A(z) 9. elem:   21
A(z) 10. elem:  34
A(z) 11. elem: 55
```

A PL/SQL eljárás sikeresen befejeződött.

## 8.4. feladat

Írjon PL/SQL programot, amely a felhasználó által megadott telephelyen kiszámítja a legnagyobb fizetési különbséget. A feladatot oldja meg SQL és PL/SQL nyelven.

### 1. megoldás (SQL)

```
SET verify OFF
ACCEPT Telephely PROMPT "Kérem a telephely nevét: "
SELECT loc          AS "Telephely",
       MIN(sal)      AS "MinFiz",
       MAX(sal)      AS "MaxFiz",
       MAX(sal)-MIN(sal) AS "MaxFizKülönbség"
  FROM emp, dept
 WHERE emp.deptno = dept.deptno AND
       UPPER(loc) = UPPER('&Telephely')
 GROUP BY emp.deptno, loc;
```

### Eredmény

Kérem a telephely nevét: New York

Telephely	MinFiz	MaxFiz	MaxFizKülönbség
-----	-----	-----	-----
NEW YORK	1300	5000	3700

## 2. megoldás (PL/SQL)

```

SET serveroutput ON
SET verify OFF
ACCEPT Telephely PROMPT 'Adja meg a telephely nevét: '
Rem Hozzárendelt változó:
VARIABLE gdiff NUMBER

Rem Itt kezdődik a PL/SQL-program
BEGIN
    SELECT MAX(sal)-MIN(sal)
        INTO :gdiff
        FROM emp
        GROUP BY deptno
        HAVING deptno = (SELECT deptno
                            FROM dept
                            WHERE UPPER(loc) = UPPER('&Telephely'));
END;
/
PRINT gdiff

```

### Eredmény

Adja meg a telephely nevét: New York

A PL/SQL eljárás sikeresen befejeződött.

GDIFF

-----

3700

## 8.5. feladat

Írjon PL/SQL programot, amely bekér két egész számot, és kiírja a legnagyobb közös osztókat. (Például 8 és 12 esetén: 4.) Használjon WHILE-ciklust.

### Megoldás

SET serveroutput ON

SET echo OFF

SET verify OFF

ACCEPT aa PROMPT 'Adja meg az első számot: '

ACCEPT bb PROMPT 'Adja meg a második számot: '

VARIABLE lnszko NUMBER

```

DECLARE
    a          NUMBER;
    b          NUMBER;
    c          NUMBER;
    LépésSzám NUMBER;
BEGIN
    a := TO_NUMBER(&aa);
    b := TO_NUMBER(&bb);
    LépésSzám := 1;
-- Euklideszi algoritmus:
    WHILE (a<>b)
        LOOP
            IF a<b
                THEN
                    c:=a;
                    a:=b;
                    b:=c;
                END IF;
                a:=a-b;
                DBMS_OUTPUT.PUT_LINE(LépésSzám || '. lépés: '||a);
                LépésSzám := LépésSzám + 1;
            END LOOP;
            :lnko:=a;
            DBMS_OUTPUT.PUT_LINE('A legnagyobb közös osztó: '||a);
        END;
    /
PROMPT A legnagyobb közös osztó:
PRINT lnko

```

**1. futtatás eredménye**

Adja meg az első számot: 72

Adja meg a második számot: 12

1. lépés: 60

2. lépés: 48

3. lépés: 36

4. lépés: 24

5. lépés: 12

A legnagyobb közös osztó: 12

A PL/SQL eljárás sikeresen befejeződött.

A legnagyobb közös osztó:

LNKO

## 2. futtatás eredménye

Adja meg az első számot: 12

Adja meg a második számot: 60

1. lépés: 48

2. lépés: 36

3. lépés: 24

4. lépés: 12

A legnagyobb közös osztó: 12

A PL/SQL eljárás sikeresen befejeződött.

A legnagyobb közös osztó:

LNKO

-----  
12

## 8.6. feladat

Írjon SQL\*Plus szkript programot, mely egy PL/SQL blokkban kiszámítja a felhasználó által megadott A számtól a szintén felhasználó által megadott B számig a páratlan számok négyzetösszegét, és ezt az SQL\*Plus gazdakörnyezetben írja ki. Használjon FOR-ciklust.

### Megoldás

```
SET serveroutput ON
```

```
SET verify OFF
```

```
ACCEPT Aszám PROMPT "Kérem az egyik számot: "
```

```
ACCEPT Bszám PROMPT "Kérem a másik számot: "
```

```
VARIABLE négyzetösszeg NUMBER
```

```
DECLARE
```

```
    v_Aszám NUMBER;
```

```
    v_Bszám NUMBER;
```

```
    segéd NUMBER;
```

```
    szumma NUMBER;
```

```
BEGIN
```

```
    v_Aszám := &Aszám;
```

```
    v_Bszám := &Bszám;
```

```
    szumma := 0;
```

```
    IF v_Aszám > v_Bszám
```

```
    THEN
```

```
        segéd := v_Aszám;
```

```
        v_Aszám := v_Bszám;
```

```
        v_Bszám := segéd;
```

```
    END IF;
```

```
FOR ciklusváltozó IN v_Aszám .. v_Bszám
LOOP
  IF MOD(ciklusváltozó, 2) != 0
  THEN
    segéd := POWER(ciklusváltozó,2);
    DBMS_OUTPUT.PUT_LINE('A(z) '|| ciklusváltozó ||
      ' négyzete: '|| segéd);
    szumma := szumma + segéd;
  END IF;

  :négyzetösszeg := szumma;
END LOOP;
END;
/
```

PROMPT A megadott tartomány páratlan számainak négyzetösszege:  
PRINT négyzetösszeg

### Eredmény

```
Kérem az egyik számot: 2
Kérem a másik számot: 10
A(z) 3 négyzete: 9
A(z) 5 négyzete: 25
A(z) 7 négyzete: 49
A(z) 9 négyzete: 81
```

A PL/SQL eljárás sikeresen befejeződött.

A megadott tartomány páratlan számainak négyzetösszege:

NÉGYZETÖSSZEG

-----  
164

## 9. FEJEZET

# Hivatkozási és összetett adattípusok, kurzor, ROWID

## Feladatok és megoldások

### 9.1. feladat

Határozza meg egy PL/SQL program segítségével a felhasználó által megadott telephelyen dolgozók létszámát és bérösszegét. (Lásd a 8.1. feladatot is.)

#### 1. megoldás (Közvetlen összegzéssel)

```
SET serveroutput ON
ACCEPT részleg PROMPT 'A részleg neve:'

DECLARE
    összfiz    emp.sal%TYPE;    -- Hivatkozási típus deklarálása
    létszám   NUMBER;
BEGIN
    SELECT SUM(sal), COUNT(sal)
        INTO összfiz, létszám
        FROM emp, dept
        WHERE UPPER(loc) = UPPER('&részleg') AND
              emp.deptno = dept.deptno;
    DBMS_OUTPUT.PUT_LINE('A ' || '&részleg' || '-i létszám: ' || létszám);
    DBMS_OUTPUT.PUT_LINE('A ' || '&részleg' || '-i összfizetés: ' || összfiz);
END;
/
```

#### Eredmény

```
A részleg neve: Dallas
régi  8:      WHERE UPPER(loc) = UPPER('&részleg') AND
új   8:      WHERE UPPER(loc) = UPPER('Dallas') AND
régi 10:     DBMS_OUTPUT.PUT_LINE('A ' || '&részleg' || '-i létszám:
```

```
'|| létszám);
új 10:  DBMS_OUTPUT.PUT_LINE('A ' || 'Dallas'||'-i létszám: ' ||
létszám);
régi 11:  DBMS_OUTPUT.PUT_LINE('A ' || '&részleg'||'-i összfizetés: ' ||
'|| összfiz);
új 11:  DBMS_OUTPUT.PUT_LINE('A ' || 'Dallas'||'-i összfizetés: ' ||
összfiz);
A Dallas-i létszám:      5
A Dallas-i összfizetés: 10875
```

A PL/SQL eljárás sikeresen befejeződött.

## 2. megoldás (Tagonkénti összesítéssel és nézettáblával)

```
SET serveroutput ON
ACCEPT részleg PROMPT 'A részleg neve: '
-- Nézet létrehozása a dolgozói sorszám meghatározásához
CREATE OR REPLACE VIEW RészlegLista
AS
SELECT ROWNUM      AS sorszám,
       ename,
       sal,
       loc
  FROM emp, dept
 WHERE UPPER(loc) = UPPER('&részleg') AND
       emp.deptno = dept.deptno;
SELECT * FROM RészlegLista;

DECLARE
  név      RészlegLista.ename%TYPE;
  fizetés  RészlegLista.sal%TYPE;
  létszám NUMBER;
  lindex   létszám%TYPE;
  részleg   RészlegLista.loc%TYPE;
  összfiz  fizetés%TYPE;
BEGIN
  SELECT COUNT(*), loc
    INTO létszám, részleg
   FROM RészlegLista
  GROUP BY loc;
  DBMS_OUTPUT.PUT_LINE('A ' || részleg ||'-i létszám: ' || létszám);
-- CIKLUS --
  lindex := 1;
  összfiz := 0;
  WHILE lindex <= létszám
```

```

LOOP
    SELECT ename, sal
        INTO név, fizetés
        FROM RészlegLista
        WHERE sorszám = lindex;
    DBMS_OUTPUT.PUT_LINE(név ||' fizetése: '|| fizetés );
    lindex := lindex + 1;
    összfiz := összfiz + fizetés;
END LOOP;
DBMS_OUTPUT.PUT_LINE('A ' || részleg ||'-i összfizetés: '|| összfiz);
END;
/

```

### Eredmény

A részleg neve: Dallas  
régi 8: WHERE UPPER(loc) = UPPER('&részleg') AND  
új 8: WHERE UPPER(loc) = UPPER('Dallas') AND  
A nézet létrejött.

SORSZÁM	ENAME	SAL	LOC
1	JONES	2975	DALLAS
2	FORD	3000	DALLAS
3	SMITH	800	DALLAS
4	SCOTT	3000	DALLAS
5	ADAMS	1100	DALLAS

A DALLAS-i létszám: 5  
JONES fizetése: 2975  
FORD fizetése: 3000  
SMITH fizetése: 800  
SCOTT fizetése: 3000  
ADAMS fizetése: 1100  
A DALLAS-i összfizetés: 10875

A PL/SQL eljárás sikeresen befejeződött.

### 3. megoldás (Tagonkénti összesítéssel és rejtett kurzorral)

```

SET serveroutput ON
ACCEPT részleg PROMPT 'A részleg neve: '

```

```

DECLARE
    név      RészlegLista.ename%TYPE;
    fizetés  RészlegLista.sal%TYPE;

```

```

létszám      NUMBER;
részleg      RészlegLista.loc%TYPE;
összfiz     fizetés%TYPE;

BEGIN
  SELECT COUNT(*), loc
    INTO létszám, részleg
   FROM emp, dept
  WHERE UPPER(loc) = UPPER('&részleg') AND
        emp.deptno = dept.deptno
 GROUP BY loc;
  DBMS_OUTPUT.PUT_LINE('A ' || részleg || '-i létszám: '|| létszám);
-- CIKLUS --
  összfiz := 0;
FOR lindex IN (SELECT ROWNUM      AS sorszám,
                     ename,
                     sal,
                     loc
                  FROM emp, dept
                 WHERE UPPER(loc) = UPPER('&részleg') AND
                       emp.deptno = dept.deptno)
LOOP
  név      := lindex.ename;
  fizetés := lindex.sal;
  DBMS_OUTPUT.PUT_LINE(név ||' fizetése: '|| fizetés );
  összfiz := összfiz + fizetés;
END LOOP;
  DBMS_OUTPUT.PUT_LINE('A ' || részleg || '-i összfizetés: '|| összfiz);
END;
/

```

**Eredmény**

A részleg neve: Dallas

régi 11: WHERE UPPER(loc) = UPPER('&részleg') AND

új 11: WHERE UPPER(loc) = UPPER('Dallas') AND

régi 22: WHERE UPPER(loc) = UPPER('&részleg') AND

új 22: WHERE UPPER(loc) = UPPER('Dallas') AND

A DALLAS-i létszám: 5

JONES fizetése: 2975

FORD fizetése: 3000

SMITH fizetése: 800

SCOTT fizetése: 3000

ADAMS fizetése: 1100

A DALLAS-i összfizetés: 10875

A PL/SQL eljárás sikeresen befejeződött.



### Megjegyzés

A szűken értelmezett feladatra nyilván az 1. megoldás a legmegfelelőbb, mivel az a legegyszerűbb.

Ha a dolgozói fizetések külön-külön is szükségesek (és főleg, ha azokkal valamilyen egyéni fel-dolgozást is el kell végezni, például sorszámozni kell, mint a 9.4. feladatban), akkor már a 2. vagy 3. megoldásra van szükség. E két utóbbi közül első látásra a rejtett kurzort használó 3. megoldás csak azért tűnik szobbnek, mert egyszerűbb. Alaposabban megvizsgálva a működésüket, azt lát-hatjuk, hogy még a 3. megoldásban a dolgozói lista (a rejtett kurzor) rendezett halmazán csak egy-szer meg végig a feldolgozás, addig a 2. megoldás listáján (RészlegLista) minden ciklusban.

Tanulságként elmondhatjuk, hogy kurzorhasználat nélkül is meg tudunk oldani egy felada-tot, ám csupán lényegesen rosszabb hatékonysággal.

## 9.2. feladat

Írjon PL/SQL programot, amely meghatározza a 7698 azonosítójú dolgozó nevét gyűjtőtábla használatával.

### Megoldás

```

SET serveroutput ON
DECLARE
    TYPE dolg_tabla_tipus IS TABLE OF emp%ROWTYPE
        INDEX BY BINARY_INTEGER;
    dolgozo dolg_tabla_tipus;
BEGIN
    SELECT *
        INTO dolgozo(1)
        FROM emp
        WHERE empno = 7698;

    IF dolgozo.EXISTS(1)
    THEN
        DBMS_OUTPUT.PUT_LINE(dolgozo(1).ename);
    END IF;
END;
/

```

### Eredmény

BLAKE

A PL/SQL eljárás sikeresen befejeződött.

## 9.3. feladat

Írjon PL/SQL programot, amely létrehoz az emp táblából egy dolgozó táblát, és megnöveli a felhasználó által megadott foglalkozású dolgozók fizetését 1000 USD-ra!

## Megoldás

```

SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

-- Itt kezdődik a PL/SQL-blokk
DECLARE
    v_azonosito    dolgozó.empno%TYPE;
    v_nev          dolgozó.ename%TYPE;
    v_munka        dolgozó.job%TYPE;
    v_fiz          dolgozó.sal%TYPE;

-- kurzor deklarációja
CURSOR dolg_kurzor IS
    SELECT empno, ename, sal
    FROM dolgozó
    WHERE UPPER(job) = UPPER('&foglalkozás');

BEGIN
    -- kurzor megnyitása
    OPEN dolg_kurzor;
    LOOP
        -- Az aktuális rekord beolvasása
        FETCH dolg_kurzor
            INTO v_azonosito, v_nev, v_fiz;
        -- Ha a kurzorterületen elfogytak a sorok, lépj ki a ciklusból
        EXIT WHEN dolg_kurzor %NOTFOUND;
        -- aktuális rekord feldolgozása
        v_fiz := v_fiz + 1000;

        UPDATE dolgozó
            SET sal = v_fiz
            WHERE empno = v_azonosito;

        DBMS_OUTPUT.PUT_LINE(v_azonosito||' '||v_nev||': ' ||
                             v_fiz||' USD');

    END LOOP;
    -- kurzor lezárása
    CLOSE dolg_kurzor;
END;
/
-- Itt fejeződik be a PL/SQL blokk
-- Ez itt mégint SQL*Plus környezet

```

```
-- A számok kiírandó oszlopszélességének beállítása
SET numwidth 5
SELECT * FROM dolgozó;
-- A számok alapértelmezett oszlopszélességének visszaállítása
SET numwidth 10
```

### Eredmény

A tábla eldobva.  
A tábla létrejött.  
Adja meg a(z) foglalkozás értékét: clerk  
7900 JAMES: 1950 USD  
7369 SMITH: 1800 USD  
7876 ADAMS: 2100 USD  
7934 MILLER: 2300 USD

A PL/SQL eljárás sikeresen befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	1950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	1800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	2100		20
7934	MILLER	CLERK	7782	82-JAN-23	2300		10

14 sor kijelölve.

### 9.4. feladat

Hozza létre a dolgozó táblát az emp táblából, és bővítsé azt egy sorszám oszloppal. Ezt töltse fel 1-től kiindulva egyesével növekvő értékkel minden dolgozó esetén a dolgozók nevének ábécé sorrendje szerint.

## 1. megoldás (Explicit kurzorra)

```

DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
ALTER TABLE dolgozó
ADD (sorszám NUMBER(2));

DECLARE
    -- Deklarálja a sorszám oszlophoz a v_sorszám változőt,
    -- és ennek kezdőértéke legyen egy
    v_sorszám dolgozó.sorszám%TYPE := 1;
    -- A kurzor a tábla rendezett sorait tartalmazza
CURSOR dolg_kurzor IS
    SELECT *
        FROM dolgozó
        ORDER BY ename;

BEGIN
    FOR drekord IN dolg_kurzor
    LOOP
        UPDATE dolgozó
            SET sorszám = v_sorszám
            WHERE empno = drekord.empno;
        v_sorszám := v_sorszám + 1;
    END LOOP;
END;
/
SET numwidth 5
SELECT *
    FROM dolgozó
    ORDER BY ename;
SET numwidth 10

```

## 2. megoldás (Rejtett kurzorra)

```

DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
ALTER TABLE dolgozó
ADD (sorszám NUMBER(2));

DECLARE
    v_sorszám dolgozó.sorszám%TYPE := 1;
BEGIN

```

```

FOR drekord IN (SELECT *
                 FROM dolgozó
                 ORDER BY ename)
LOOP
  UPDATE dolgozó
    SET sorszám = v_sorszám
   WHERE empno = drekord.empno;
  v_sorszám := v_sorszám + 1;
END LOOP;
END;
/
SET numwidth 5
SELECT *
  FROM dolgozó
  ORDER BY ename;
SET numwidth 10

```

### 3. megoldás (Explicit kurzorra és CURRENT OF hivatkozással)

```

DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
ALTER TABLE dolgozó
  ADD (sorszám NUMBER(2));

DECLARE
  v_sorszám dolgozó.sorszám%TYPE := 1;
  CURSOR dolg_kurzor IS
    SELECT *
      FROM dolgozó
      ORDER BY ename
      FOR UPDATE OF sorszám NOWAIT;

BEGIN
  FOR drekord IN dolg_kurzor
  LOOP
    UPDATE dolgozó
      SET sorszám = v_sorszám
     WHERE CURRENT OF dolg_kurzor;
    v_sorszám := v_sorszám + 1;
  END LOOP;
END;
/

```

```

SET numwidth 5
SELECT *
  FROM dolgozó
 ORDER BY ename;
SET numwidth 10

```

### Eredmény (mindhárom megoldás esetén)

- A tábla eldobva.
- A tábla létrejött.
- A tábla módosítva.
- A PL/SQL eljárás sikeresen befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	SORSZÁM
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20	1
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30	2
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30	3
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10	4
7902	FORD	ANALYST	7566	81-DEC-03	3000		20	5
7900	JAMES	CLERK	7698	81-DEC-03	950		30	6
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20	7
7839	KING	PRESIDENT		81-NOV-17	5000		10	8
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30	9
7934	MILLER	CLERK	7782	82-JAN-23	1300		10	10
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20	11
7369	SMITH	CLERK	7902	80-DEC-17	800		20	12
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30	13
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30	14

14 sor kijelölve.

## 9.5. feladat

Írjon PL/SQL programot, amely létrehoz az emp táblából egy dolgozó táblát, és megnöveli a felhasználó által megadott százalékértékkel minden, az átlagfizetésnél alacsonyabb fizetéssel rendelkező dolgozó fizetését.

### 1. megoldás (Explicit kurzorra)

```

SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

```

```

COLUMN empno FORMAT 9999
COLUMN comm FORMAT 9999
COLUMN deptno FORMAT 9999

ACCEPT növekmény PROMPT 'A növekmény százalékértéke: '

-- Az allekérdezés tesztelése (ebből lesz a kurzor)
SELECT empno,
       ename,
       sal,
       ROUND(al.átlag - dolgozó.sal) AS különbség
FROM dolgozó,
     (SELECT ROUND(AVG(sal)) AS átlag
      FROM dolgozó) al
WHERE al.átlag - dolgozó.sal > 0;

-- A PL/SQL-blokk
DECLARE
    CURSOR egydolgozó IS
        SELECT empno,
               ename,
               sal,
               ROUND(al.átlag - dolgozó.sal)
        FROM dolgozó,
             (SELECT ROUND(AVG(sal)) AS átlag
              FROM dolgozó) al
        WHERE al.átlag - dolgozó.sal > 0
        FOR UPDATE OF sal NOWAIT;

    v_azon      dolgozó.empno%TYPE;
    v_ename     dolgozó.ename%TYPE;
    v_különbség dolgozó.sal%TYPE;
    v_sal       dolgozó.sal%TYPE;
    v_százalék  NUMBER(2);

BEGIN
    v_százalék := &növekmény;
    OPEN egydolgozó;
    LOOP
        FETCH egydolgozó
        INTO v_azon,
             v_ename,
             v_sal,
             v_különbség;

```

```

    EXIT WHEN egydolgozó %NOTFOUND;
    v_sal := v_sal + ROUND(v_sal * v_százalék/100);
    UPDATE dolgozó
        SET sal = v_sal
        WHERE CURRENT OF egydolgozó;

    DBMS_OUTPUT.PUT_LINE('aktuális sor ' ||
                           v_azon||' '||v_ename||' '||v_sal);
END LOOP;
CLOSE egydolgozó;
COMMIT;
END;
/

```

CLEAR COLUMNS

### Eredmény

A tábla eldobva.  
A tábla létrejött.

A növekmény százalékértéke: 25

EMPNO	ENAME	SAL	KÜLÖNBSÉG
7654	MARTIN	1250	823
7499	ALLEN	1600	473
7844	TURNER	1500	573
7900	JAMES	950	1123
7521	WARD	1250	823
7369	SMITH	800	1273
7876	ADAMS	1100	973
7934	MILLER	1300	773

8 sor kijelölve.

```

régi 20:  v_százalék := &növekmény;
új 20:  v_százalék := 25;
aktuális sor 7654 MARTIN 1563
aktuális sor 7499 ALLEN 2000
aktuális sor 7844 TURNER 1875
aktuális sor 7900 JAMES 1188
aktuális sor 7521 WARD 1563
aktuális sor 7369 SMITH 1000
aktuális sor 7876 ADAMS 1375

```

aktuális sor 7934 MILLER 1625

A PL/SQL eljárás sikeresen befejeződött.

## 2. megoldás (Rejtett kurzorra)

```
SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
COLUMN empno FORMAT 9999
COLUMN comm FORMAT 9999
COLUMN deptno FORMAT 9999
SET verify OFF
```

ACCEPT növekmény PROMPT 'A növekmény százalékértéke: '

```
-- Az allekérdezés tesztelése
SELECT empno,
       ename,
       sal,
       ROUND(al.átlag - dolgozó.sal) AS különbség
  FROM dolgozó,
       (SELECT ROUND(AVG(sal)) AS átlag
        FROM dolgozó) al
 WHERE al.átlag - dolgozó.sal > 0;
```

```
-- A PL/SQL program
DECLARE
  v_százalék NUMBER(2);

BEGIN
  v_százalék := &növekmény;
  FOR DolgIndex IN
    (SELECT empno
     FROM dolgozó,
          (SELECT ROUND(AVG(sal)) AS átlag
           FROM dolgozó) al
     WHERE al.átlag - dolgozó.sal > 0)
  LOOP
    UPDATE dolgozó
      SET sal = sal + ROUND(sal * v_százalék/100)
      WHERE empno = DolgIndex.empno;
    DBMS_OUTPUT.PUT_LINE('aktuális sor ' || DolgIndex.empno);
  END LOOP;
```

```

    COMMIT;
END;
/

SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
SET verify ON
CLEAR COLUMNS
UNDEFINE növekmény

```

**Eredmény**

A tábla eldobva.  
A tábla létrejött.

A növekmény százalékértéke: 25

EMPNO	ENAME	SAL	KÜLÖNBSÉG
7654	MARTIN	1250	823
7499	ALLEN	1600	473
7844	TURNER	1500	573
7900	JAMES	950	1123
7521	WARD	1250	823
7369	SMITH	800	1273
7876	ADAMS	1100	973
7934	MILLER	1300	773

8 sor kijelölve.

aktuális sor 7654  
aktuális sor 7499  
aktuális sor 7844  
aktuális sor 7900  
aktuális sor 7521  
aktuális sor 7369  
aktuális sor 7876  
aktuális sor 7934

A PL/SQL eljárás sikeresen befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10

7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1563	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	2000	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1875	0	30
7900	JAMES	CLERK	7698	81-DEC-03	1188		30
7521	WARD	SALESMAN	7698	81-FEB-22	1563	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	1000		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1375		20
7934	MILLER	CLERK	7782	82-JAN-23	1625		10

14 sor kijelölve.

## 9.6. feladat

Írjon PL/SQL blokkot tartalmazó szkript programot, amely (eldob és) létrehoz egy dolgozó táblát az emp táblából, és ebben foglalkozásonként megnöveli a legkisebb fizetésű dolgozók bérét a foglalkozási csoportjukban legnagyobb fizetés és az ugyanitt számított átlagfizetés különbségének 20%-ával.

### Megoldás (Rejtett kurzorral)

#### 1. lépés (A cursor lekérdezése)

```
SELECT MAX(sal)      AS nagyfiz,
       MIN(sal)      AS kisfiz,
       ROUND(AVG(sal)) AS átlagfiz,
       job          AS foglalkozás
  FROM dolgozó
 GROUP BY JOB;
```

#### 2. lépés (A dolgozók fizetésének egyesével történő módosítása)

```
SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
SET numwidth 5
SELECT * FROM dolgozó
 ORDER BY job, sal;

BEGIN
  FOR foglIndex IN
    (SELECT MAX(sal)      AS nagyfiz,
```

```

        MIN(sal)          AS kisfiz,
        ROUND(AVG(sal))  AS átlagfiz,
        job               AS foglalkozás
      FROM dolgozó
      GROUP BY job)

LOOP
  UPDATE dolgozó
    SET sal = sal +
      ROUND((foglIndex.nagyfiz -
      foglIndex.átlagfiz)*0.2)
  WHERE sal = foglIndex.kisfiz AND
    job = foglIndex.foglalkozás;
END LOOP;
END;
/

```

SELECT \* FROM dolgozó  
 ORDER BY job, sal;  
 SET numwidth 10

**Eredmény**

- A tábla eldobva.
- A tábla létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30

14 sor kijelölve.

A PL/SQL eljárás sikeresen befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	852		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
7782	CLARK	MANAGER	7839	81-JÚN-09	2493		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7654	MARTIN	SALESMAN	7698	81-SZE-28	1290	1400	30
7521	WARD	SALESMAN	7698	81-FEB-22	1290	500	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30

14 sor kijelölve.

## 9.7. feladat

Mivel annál több a munka, minél kevesebben végzik, ezért kapjon minden dolgozó egyszeri prémiumot, amely fizetésének  $n$ -ed része, ahol  $n$  a vele azonos munkakörben dolgozók száma (őt magát is beleértve). Az emp táblából létrehozott dolgozó táblát használja.

### 1. megoldás (Gyűjtőtáblával)

```

SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
COLUMN empno FORMAT 9999
COLUMN comm FORMAT 9999
COLUMN deptno FORMAT 9999

DECLARE
CURSOR MunkaKurzor IS
SELECT job, COUNT(job)
FROM dolgozó
GROUP BY job;
TYPE T_munkakör_rekord IS RECORD
(munkakör dolgozó.job%TYPE,
tétszám NUMBER(2));
TYPE T_munkakör_tábla IS TABLE OF T_munkakör_rekord
INDEX BY BINARY_INTEGER;

```

```

CURSOR DolgozóKurzor IS
  SELECT *
    FROM dolgozó;
TYPE T_prémium_rekord IS RECORD
  (azonosító  dolgozó.empno%TYPE,
   név        dolgozó.ename%TYPE,
   munkakör   dolgozó.job%TYPE,
   fizetés    dolgozó.sal%TYPE,
   létszám   NUMBER(2),
   prémium    dolgozó.sal%TYPE);
TYPE T_prémium_tábla IS TABLE OF T_prémium_rekord
  INDEX BY BINARY_INTEGER;
Munka      T_munkakör_tábla;
MIndex     BINARY_INTEGER;
Prémium   T_prémium_tábla;
PIndex     BINARY_INTEGER;
Drekord   dolgozó%ROWTYPE;

BEGIN
  -- A Munkakör tábla feltöltése
  MIndex := 1;
  OPEN MunkaKurzor;
  LOOP
    FETCH MunkaKurzor
      INTO Munka(MIndex);
    EXIT WHEN MunkaKurzor %NOTFOUND;

    MIndex := MIndex + 1;
  END LOOP; -- Munka
  CLOSE MunkaKurzor;

  -- A Munkakör tábla kiíratása
  DBMS_OUTPUT.PUT_LINE('A MUNKAKÖR TÁBLA:');
  DBMS_OUTPUT.PUT_LINE('Munkakör Létszám');
  FOR MIndex IN Munka.FIRST..Munka.LAST
    LOOP
      DBMS_OUTPUT.PUT_LINE(Munka(MIndex).munkakör||' '||
                           Munka(MIndex).létszám);
    END LOOP; -- Munka
  DBMS_OUTPUT.PUT_LINE('A munkakörök száma: '|||
                        Munka.COUNT);

  -- A Prémium tábla feltöltése
  PIndex := 1;
  OPEN DolgozóKurzor;

```

```

LOOP
  FETCH DolgozóKurzor
    INTO Drekord;
  EXIT WHEN DolgozóKurzor %NOTFOUND;

  Prémium(PIndex).azonosító := Drekord.empno;
  Prémium(PIndex).név      := Drekord.ename;
  Prémium(PIndex).munkakör := Drekord.job;
  Prémium(PIndex).fizetés  := Drekord.sal;
  FOR MIndex IN Munka.FIRST..Munka.LAST
    LOOP
      IF Munka(MIndex).munkakör = Prémium(PIndex).munkakör
      THEN
        Prémium(PIndex).létszám := Munka(MIndex).létszám;
        Prémium(PIndex).prémium :=
          ROUND(Prémium(PIndex).fizetés / Prémium(PIndex).létszám);
        EXIT;
      END IF;
    END LOOP; -- Munka
  PIndex := PIndex + 1;
END LOOP; -- Prémium
CLOSE DolgozóKurzor;

-- A Prémium tábla kiíratása
DBMS_OUTPUT.PUT_LINE('----');
DBMS_OUTPUT.PUT_LINE('A PRÉMIUM TÁBLA:');
DBMS_OUTPUT.PUT_LINE('Azonosító Név Munkakör' || |
                      'Fizetés Létszám Prémium');

PIndex := Prémium.FIRST;
WHILE PIndex IS NOT NULL
  LOOP
    DBMS_OUTPUT.PUT_LINE(Prémium(PIndex).azonosító||' '|||
                          Prémium(PIndex).név      ||' '|||
                          Prémium(PIndex).munkakör ||' '|||
                          Prémium(PIndex).fizetés  ||' '|||
                          Prémium(PIndex).létszám ||' '|||
                          Prémium(PIndex).prémium);

    PIndex := Prémium.NEXT(PIndex);
  END LOOP; -- Prémium
END;
/

SELECT * FROM dolgozó;
CLEAR COLUMNS

```

**Eredmény**

A tábla eldobva.

A tábla létrejött.

A MUNKAKÖR TÁBLA:

Munkakör Létszám

ANALYST 2

CLERK 4

MANAGER 3

PRESIDENT 1

SALESMAN 4

A munkakörök száma: 5

----

A PRÉMIUM TÁBLA:

Azonosító	Név	Munkakör	Fizetés	Létszám	Prémium
7839	KING	PRESIDENT	5000	1	5000
7698	BLAKE	MANAGER	2850	3	950
7782	CLARK	MANAGER	2450	3	817
7566	JONES	MANAGER	2975	3	992
7654	MARTIN	SALESMAN	1250	4	313
7499	ALLEN	SALESMAN	1600	4	400
7844	TURNER	SALESMAN	1500	4	375
7900	JAMES	CLERK	950	4	238
7521	WARD	SALESMAN	1250	4	313
7902	FORD	ANALYST	3000	2	1500
7369	SMITH	CLERK	800	4	200
7788	SCOTT	ANALYST	3000	2	1500
7876	ADAMS	CLERK	1100	4	275
7934	MILLER	CLERK	1300	4	325

```

BEGIN
  -- A fejléc kiíratása
  DBMS_OUTPUT.PUT_LINE('NÉV      |||'
                       'Munkakör    |||'
                       'Fizetés     |||'
                       'Létszám     |||'
                       'Prémium');

  DBMS_OUTPUT.PUT_LINE('=====');

  -- A külső ciklus (végigmegy az összes dolgozó rekordján)
  FOR egydolgozó IN (SELECT * FROM dolgozó)
  LOOP
    -- A belső ciklus (végigmegy az egyes munkakörökön)
    FOR munkakör IN (SELECT job,
                           COUNT(job) AS létszám
                         FROM dolgozó
                         GROUP BY job)
    LOOP
      -- Ha a külső ciklus aktuális dolgozójának munkaköre
      -- megegyezik a belső ciklus aktuális munkakörével,
      IF egydolgozó.job = munkakör.job
      THEN
        -- akkor növeljük meg a dolgozó jutalékát a prémiumával
        UPDATE dolgozó
          SET comm = ROUND(NVL(comm,0)+ sal/munkakör.létszám)
          WHERE empno = egydolgozó.empno;
        -- Kiíratás
        DBMS_OUTPUT.PUT_LINE(egydolgozó.ename ||| '|||'
                            egydolgozó.sal ||| '|||'
                            munkakör.létszám ||| '|||'
                            ROUND(egydolgozó.sal/munkakör.létszám));
        -- Az egyezés és módosítás után kiugrás a belső ciklusból
        -- ugrás a következő dolgozó rekordjára
        EXIT;
      END IF;
    -- A belső (munkakör) ciklus bezárása
    END LOOP;
  -- A külső (egydolgozó) ciklus bezárása
  END LOOP;
END;
/
SELECT * FROM dolgozó;
CLEAR COLUMNS

```

## Eredmény

A tábla eldobva.

A tábla létrejött.

NÉV	Munkakör	Fizetés	Létszám	Prémium
KING	5000	1	5000	
BLAKE	2850	3	950	
CLARK	2450	3	817	
JONES	2975	3	992	
MARTIN	1250	4	313	
ALLEN	1600	4	400	
TURNER	1500	4	375	
JAMES	950	4	238	
WARD	1250	4	313	
FORD	3000	2	1500	
SMITH	800	4	200	
SCOTT	3000	2	1500	
ADAMS	1100	4	275	
MILLER	1300	4	325	

NÉV	Munkakör	Fizetés	Létszám	Prémium
KING	5000	1	5000	
BLAKE	2850	3	950	
CLARK	2450	3	817	
JONES	2975	3	992	
MARTIN	1250	4	313	
ALLEN	1600	4	400	
TURNER	1500	4	375	
JAMES	950	4	238	
WARD	1250	4	313	
FORD	3000	2	1500	
SMITH	800	4	200	
SCOTT	3000	2	1500	
ADAMS	1100	4	275	
MILLER	1300	4	325	

A PL/SQL eljárás sikeresen befejeződött.

A dolgozó tábla listája a harmadik megoldás után található.

### 3. megoldás (Rejtett kurzorral és ROWID-val)

```

SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
COLUMN empno FORMAT 9999
COLUMN comm FORMAT 9999
COLUMN deptno FORMAT 9999

BEGIN
    -- A fejléc kiíratása
    DBMS_OUTPUT.PUT_LINE('NÉV      ||')
    DBMS_OUTPUT.PUT_LINE('Munkakör ||')
    DBMS_OUTPUT.PUT_LINE('Fizetés  ||')
    DBMS_OUTPUT.PUT_LINE('Létszám ||')
    DBMS_OUTPUT.PUT_LINE('Prémium');

    DBMS_OUTPUT.PUT_LINE('=====');
    -- A külső ciklus (végigmegy az összes dolgozó rekordján)

```

```

FOR egydolgozó IN (SELECT ename, empno, job, sal, ROWID
                     FROM dolgozó)
LOOP
  -- A belső ciklus (végigmegy az egyes munkakörökön)
  FOR munkakör IN (SELECT job,
                        COUNT(job) AS létszám
                        FROM dolgozó
                        GROUP BY job)
  LOOP
    -- Ha a külső ciklus aktuális dolgozójának munkaköre
    -- megegyezik a belső ciklus aktuális munkakörével,
    IF egydolgozó.job = munkakör.job
    THEN
      -- akkor növeljük meg a dolgozó jutalékát a prémiumával
      UPDATE dolgozó
        SET comm = ROUND(NVL(comm,0)+ sal/munkakör.létszám)
        WHERE ROWID = egydolgozó.ROWID;
      -- Megjegyezzük, hogy itt a ROWID logikailag ekvivalens
      -- az "empno = egydolgozó.empno" feltételellet.
      -- Kiíratás
      DBMS_OUTPUT.PUT_LINE(egydolgozó.ename || ' ' ||
                           egydolgozó.sal || ' ' ||
                           munkakör.létszám || ' ' ||
                           ROUND(egydolgozó.sal/munkakör.létszám));
      -- Az egyezés és módosítás után kiugrás a belső ciklusból
      -- ugrás a következő dolgozó rekordjára
      EXIT;
    END IF;
    -- A belső (munkakör) ciklus bezárása
  END LOOP;
  -- A külső (egydolgozó) ciklus bezárása
END LOOP;
END;
/
SELECT * FROM dolgozó;
CLEAR COLUMNS

```

### Eredmény

A tábla eldobva.  
A tábla létrejött.

NÉV Munkakör Fizetés Létszám Prémium

KING	5000	1	5000	
BLAKE	2850	3	950	
CLARK	2450	3	817	
JONES	2975	3	992	
MARTIN	1250	4	313	
ALLEN	1600	4	400	
TURNER	1500	4	375	
JAMES	950	4	238	
WARD	1250	4	313	
FORD	3000	2	1500	
SMITH	800	4	200	
SCOTT	3000	2	1500	
ADAMS	1100	4	275	
MILLER	1300	4	325	

A PL/SQL eljárás sikeresen befejeződött.

#### A dolgozó tábla minden sor megoldás után

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000	5000	10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850	950	30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450	817	10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975	992	20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1713	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	700	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	375	30
7900	JAMES	CLERK	7698	81-DEC-03	950	238	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	813	30
7902	FORD	ANALYST	7566	81-DEC-03	3000	1500	20
7369	SMITH	CLERK	7902	80-DEC-17	800	200	20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000	1500	20
7876	ADAMS	CLERK	7788	83-JAN-12	1100	275	20
7934	MILLER	CLERK	7782	82-JAN-23	1300	325	10

14 sor kijelölve.

## 9.8. feladat

Írjon PL/SQL programot, amely az emp táblából létrehozott dolgozó táblában megnöveli azoknak a főnököknek (mgr) a fizetését, akiknek egynél több beosztottjuk van. A növelés mértéke: annyiszor  $N$  USD, ahány beosztottjuk van. Az  $N$  értékét a felhasználó adja meg. A fizetés csökkenő értéke szerint rendezve listázza a főnököket.

### Megoldás

```

SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
SET numwidth 5
PROMPT Főnökök :
SELECT *
  FROM dolgozó
 WHERE empno IN
      (SELECT mgr FROM dolgozó)
 ORDER BY sal DESC;
ACCEPT N PROMPT 'A növelési tényező (1,2,...) :'

DECLARE
  N dolgozó.sal%TYPE;

BEGIN
  FOR főnIndex IN (SELECT COUNT(dolg.job) AS létszám,
                      dolg.mgr      AS azonosító
                     FROM dolgozó dolg,
                          dolgozó főnök
                    WHERE dolg.mgr = főnök.empno
                      GROUP BY dolg.mgr
                      HAVING COUNT(dolg.job) > 1)
  LOOP
    UPDATE dolgozó
      SET sal = sal + &N*főnIndex.létszám
        WHERE dolgozó.empno = főnIndex.azonosító AND
              főnIndex.létszám > 1;
    DBMS_OUTPUT.PUT_LINE(főnIndex.létszám||' ' ||
                           főnIndex.azonosító);
  END LOOP;
END;
/
PROMPT Főnökök :
```

```

SELECT *
  FROM dolgozó
 WHERE empno IN
       (SELECT mgr FROM dolgozó)
 ORDER BY sal DESC;
SET numwidth 10

```

**Eredmény**

A tábla eldobva.

A tábla létrejött.

Főnökök :

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10

6 sor kijelölve.

A növelési tényező (1,2,...) : 50

régi 14: SET sal = sal + &N\*főnIndex.létszám

új 14: SET sal = sal + 50\*főnIndex.létszám

2 7566

5 7698

3 7839

A PL/SQL eljárás sikeresen befejeződött.

Főnökök :

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5150		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	3100		30
7566	JONES	MANAGER	7839	81-ÁPR-02	3075		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10

6 sor kijelölve.

## 9.9. feladat

Írjon PL/SQL programot, amely az emp táblából létrehozott dolgozó táblában megnöveli azoknak a dolgozóknak a fizetését, akiknek az azonosítója páros, és fizetésük kisebb, mint e csoport átlagfizetése. A növelés e fizetéskülönbség 20%-a.

### 1. megoldás (CURSOR, SELECT és FOR)

```

SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

DECLARE
    CURSOR párosak IS
        SELECT empno, sal
        FROM dolgozó
        WHERE MOD(empno,2) = 0;

    v_áttag dolgozó.sal%TYPE;

BEGIN
    SELECT AVG(sal)
    INTO v_áttag
    FROM dolgozó,
        (SELECT empno
        FROM dolgozó
        WHERE MOD(empno,2) = 0) pdolgozó
    WHERE dolgozó.empno = pdolgozó.empno;
    DBMS_OUTPUT.PUT_LINE('Párosak átlagfizetése: '||TO_CHAR(v_áttag));
    --Egyenként végignézi a dolgozókat
    FOR egyenként IN
        (SELECT empno, sal
        FROM dolgozó)
    LOOP
        -- A párosak ciklusa
        FOR pkurzor IN párosak
        LOOP
            IF (egyenként.empno = pkurzor.empno) AND
                (v_áttag - egyenként.sal > 0)
            THEN
                UPDATE dolgozó
                SET sal = sal + ROUND((v_áttag - sal) * 0.2)
                WHERE dolgozó.empno = pkurzor.empno;
                DBMS_OUTPUT.PUT_LINE(TO_CHAR(egyenként.empno)||' ' ||
                                      TO_CHAR(v_áttag - egyenként.sal));
            END IF;
        END LOOP;
    END LOOP;
END;

```

```

    END IF;
    END LOOP;
END LOOP;
END;
/

SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10

```

### Eredmény

A tábla eldobva.  
A tábla létrejött.

Párosak átlagfizetése: 2037.5

```

7654 787.5
7844 537.5
7900 1087.5
7876 937.5
7934 737.5

```

A PL/SQL eljárás sikeresen befejeződött.

### 2. megoldás (FOR és allekérdezés a FOR-ciklusban)

```

SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
COLUMN empno FORMAT 9999
COLUMN comm FORMAT 9999
COLUMN deptno FORMAT 9999

BEGIN
FOR pdolgozó IN
  (SELECT páros.pátlag AS pátlagok,      -- hozzuk ki az átlagot
   dolgozó.empno AS pazon,              -- az empno-t azonosítás céljából
   dolgozó.sal AS fizetés            -- és a fizetést az átlaghoz való hasonlítás miatt
  FROM dolgozó,
       (SELECT AVG(sal) AS pátlag        -- az átlag kiszámítása
        FROM dolgozó
        WHERE MOD(empno,2) = 0) páros  -- ha a maradék 0, akkor páros
  WHERE dolgozó.empno IN
        (SELECT empno                  -- párosak azonosítójával egyenlők
         FROM dolgozó
         WHERE MOD(empno,2) = 0))

```

```

LOOP
  IF (pdolgozó.pátlagok > pdolgozó.fizetés)
  THEN
    DBMS_OUTPUT.PUT_LINE(pdolgozó.pazon);
    UPDATE dolgozó
      SET sal = sal +
        ROUND((pdolgozó.pátlagok - sal)*0.2)
      WHERE (dolgozó.empno = pdolgozó.pazon); -- ha az azonosító megegyezik
  END IF;
END LOOP;
END;

```

```

SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
CLEAR COLUMNS

```

### Eredmény

A tábla eldobva.  
A tábla létrejött.

7654  
7844  
7876  
7900  
7934

A PL/SQL eljárás sikeresen befejeződött.  
A tábla eldobva.  
A tábla létrejött.

Párosak átlagfizetése: 2037.5  
7654 787.5  
7844 537.5  
7900 1087.5  
7876 937.5  
7934 737.5

A PL/SQL eljárás sikeresen befejeződött.

### 3. megoldás (Allekérdezés a FOR-ciklusban mint rejtett kurzor)

#### 3.1. lépés (Állítsuk elő a lekérdezést és ellenőrizzük)

```
DROP TABLE dolgozó;
```

```
CREATE TABLE dolgozó
```

```
AS / SELECT * FROM emp;
```

```
COLUMN empno FORMAT 9999
```

```
COLUMN comm FORMAT 9999
```

```
COLUMN deptno FORMAT 9999
```

```
SELECT empno, sal, al.átlag
  FROM dolgozó,
       (SELECT AVG(sal) AS átlag
        FROM dolgozó
        GROUP BY MOD(empno,2)
        HAVING MOD(empno,2) = 0) al
 WHERE sal < al.átlag AND
       (MOD(empno,2) = 0);
```

#### Eredmény

Tábla eldobva.

Tábla létrejött.

EMPNO	SAL	ÁTLAG
7654	1250	2037.5
7844	1500	2037.5
7876	1100	2037.5
7900	950	2037.5
7934	1300	2037.5

#### 3.2. lépés (A PL/SQL program)

```
DROP TABLE dolgozó;
```

```
CREATE TABLE dolgozó
```

```
AS / SELECT * FROM emp;
```

```
SET numwidth 5
```

```
SELECT * FROM dolgozó;
```

```
SET numwidth 10
```

```
BEGIN
```

```
FOR pdolgozó IN
      (SELECT empno AS pazon,
              al.átlag AS pátlagok
```

```

        FROM dolgozó,
        (SELECT AVG(sal) AS átlag
         FROM dolgozó
         GROUP BY MOD(empno,2)
         HAVING MOD(empno,2) = 0) al
        WHERE sal < al.átlag AND
              (MOD(empno,2) = 0))          -- paritás szerint csoportosítva .
        LOOP                                -- de csak a páros kell
        UPDATE dolgozó                      -- fizetés kisebb az átlagnál
        SET sal = sal +                     -- de csak a páros empnoknál
              ROUND((pdolgozó.pátlagok - sal)*0.2) -- ciklus indítása
        WHERE (dolgozó.empno = pdolgozó.pazon); -- növeljük ezek fizetését
        END LOOP;
    END;
/
SET numwidth 5
SELECT * FROM dolgozó;
SET numwidth 10
CLEAR COLUMNS

```

### Eredmény

A tábla eldobva.  
A tábla létrejött.  
A PL/SQL eljárás sikeresen befejeződött.

### A dolgozó tábla mindegyik megoldás után

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1408	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1608	0	30
7900	JAMES	CLERK	7698	81-DEC-03	1168		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1288		20
7934	MILLER	CLERK	7782	82-JAN-23	1448		10

14 sor kijelölve.

## 9.10. feladat

Írjon egy olyan PL/SQL blokkot tartalmazó SQL\*Plus szkript programot, amely azon foglalkozási csoportok dolgozóit, amelyekben az összesített jövedelem nem haladja meg a felhasználó által megadott értéket, egyszeri, a jutalékot növelő jatalomban részesíti. A jatalom összege azon foglalkozási csoportok átlagjövedelmének 1%-a, amelyeknek összes jövelme meghaladja a felhasználó által megadott előző értéket. A szkript program végül listázza ki a dolgozó tábla személyeit a nevük szerint csökkenő sorrendben.

### Megoldás

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
COLUMN empno FORMAT 9999
COLUMN deptno FORMAT 9999
COLUMN mgr FORMAT 9999
SET verify OFF
```

#### 1. lépés (A megfelelő csoportok nézete)

```
CREATE OR REPLACE VIEW megfelelők
AS
SELECT job AS foglalkozás,
       SUM(sal+NVL(comm,0)) AS összesjöv
  FROM dolgozó
 GROUP BY job
 HAVING SUM(sal+NVL(comm,0)) <= &érték;
SELECT * FROM megfelelők;
```

#### 2. lépés (Nem megfelelő foglalkozási csoportok átlagának 1%-a)

```
SELECT AVG(összesjöv) * 0.01
  FROM dolgozó,
       (SELECT job AS foglalkozás,
              SUM(sal+NVL(comm,0)) AS összesjöv
     FROM dolgozó
    GROUP BY job
   HAVING SUM(sal+NVL(comm,0)) >= & érték);
```

#### 3. lépés (PL/SQL blokk)

```
BEGIN
  UPDATE dolgozó
    SET comm = sal +NVL(comm,0)
      + (SELECT AVG(összesjöv)* 0.01
        FROM (SELECT job,
                     SUM(sal+NVL(comm,0))AS összesjöv
```

```

        FROM dolgozó
        GROUP BY job
        HAVING SUM(sal+NVL(comm,0)) >= &érték))
WHERE job IN (SELECT foglalkozás
                FROM megfelelők);
END;
/
SET numwidth 5
SELECT * FROM dolgozó
    ORDER BY ename DESC;
SET numwidth 10
CLEAR COLUMNS
SET verify ON

```

### Eredmény

A fizetés összértéke: 6666  
A tábla eldobva.  
A tábla létrejött.  
régi 7: HAVING SUM(sal+NVL(comm,0)) <= & érték  
új 7: HAVING SUM(sal+NVL(comm,0)) <= 6666  
A nézet létrejött.

### FOGLALKOZ ÖSSZESJÖV

```

-----
ANALYST      6000
CLERK        4150
PRESIDENT    5000
-----
```

### AVG(ÖSSZESJÖV)\*0.01

```

-----
         93.87532
```

A PL/SQL eljárás sikeresen befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7369	SMITH	CLERK	7902	80-DEC-17	800	880.38	20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000	3080.38	20
7934	MILLER	CLERK	7782	82-JAN-23	1300	1380.38	10
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7839	KING	PRESIDENT		81-NOV-17	5000	5080.38	10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20

7900	JAMES	CLERK	7698	81-DEC-03	950	1030.38	30
7902	FORD	ANALYST	7566	81-DEC-03	3000	3080.38	20
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7876	ADAMS	CLERK	7788	83-JAN-12	1100	1180.38	20

14 sor kijelölve.

## 9.11. feladat

Írjon egy olyan PL/SQL blokkot tartalmazó SQL\*Plus szkript programot, amely (eldob és) létrehoz az emp táblából egy dolgozó táblát, és amely elvégzi a kötelező bérémelést a felhasználó által megadott évben oly módon, hogy a főnökök (akiknek tehát legalább egy beosztottjuk van) fizetését annak 20%-ával növeli a 3-mal osztható években, a beosztottakét pedig a fizetésük 10%-ával növeli az 5-tel osztható években. A szkript program végül listázza ki az alkalmazottak dolgozó tábláját a fizetésük szerint csökkenő sorrendben.

### Megoldás

```

ACCEPT DátumÉv PROMPT 'Az adatmódosítás éve: ';
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
COLUMN empno FORMAT 9999
COLUMN comm FORMAT 9999
COLUMN deptno FORMAT 9999

DECLARE
    v_AdatÉv      NUMBER(4);
    v_FőnökÉv     NUMBER(4) := 3;
    v_DolgÉv       NUMBER(4) := 5;
    v_FőnökSzázalék NUMBER(3,1) := 1.2;
    v_DolgSzázalék NUMBER(3,1) := 1.1;

BEGIN
    v_AdatÉv := &DátumÉv;
    FOR dIndex IN (SELECT empno FROM dolgozó)
    LOOP
        IF MOD(v_AdatÉv, v_FőnökÉv) = 0 THEN          -- Főnök ág
            UPDATE dolgozó
            SET sal = sal * v_FőnökSzázalék
            WHERE empno = dIndex.empno AND
                  empno IN (SELECT NVL(mgr,0)
                             FROM dolgozó);
        END IF;
    END IF;

```

```

IF MOD(v_AdatÉv, v_DolgÉv) = 0 THEN          -- Beosztott ág
    UPDATE dolgozó
        SET sal = sal * v_DolgSzázalék
        WHERE empno = dIndex.empno AND
              empno NOT IN (SELECT NVL(mgr,0)
                               FROM dolgozó);
    END IF;
END LOOP;
END;
/
SET numwidth 5
SELECT * FROM dolgozó
    ORDER BY sal DESC;
SET numwidth 10
CLEAR COLUMNS

```

### Eredmény

Az adatmódosítás éve: 1995

A tábla eldobva.

A tábla létrejött.

régi 9: v\_AdatÉv := &DátumÉv;

új 9: v\_AdatÉv := 1995;

A PL/SQL eljárás sikeresen befejeződött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	6000		10
7902	FORD	ANALYST	7566	81-DEC-03	3600		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3600		20
7566	JONES	MANAGER	7839	81-ÁPR-02	3570		20
7698	BLAKE	MANAGER	7839	81-MÁJ-01	3420		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2940		10
7499	ALLEN	SALESMAN	7698	81-FEB-20	1760	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1650	0	30
7934	MILLER	CLERK	7782	82-JAN-23	1430		10
7654	MARTIN	SALESMAN	7698	81-SZE-28	1375	1400	30
7521	WARD	SALESMAN	7698	81-FEB-22	1375	500	30
7876	ADAMS	CLERK	7788	83-JAN-12	1210		20
7900	JAMES	CLERK	7698	81-DEC-03	1045		30
7369	SMITH	CLERK	7902	80-DEC-17	880		20

14 sor kijelölve.

# Összetett (párosítási) feladatok és megoldások

## 9.15. feladat

A RAPID RALLY autóversenyre benevezett versenyzők adatbázissémája:

*versenyzők( név, feladatkör, nemzetiség, minősítés, partner ),*

ahol az attribútumok adattípusai:

- név, nemzetiség, partner: karakterSORozatOK,
- feladatkör: (navigátor, pilóta),
- minősítés: 1..1000,
- a partner inicializált értéke: '---' vagy NULL.

1. Hozza létre a fenti táblát, és töltse fel az alábbi feladat értelmes megoldásához elegendő adattal.
2. Készítsen PL/SQL programot, amely minden versenyzőhöz megkísérel az alábbi feltételeknek megfelelő partnert keresni:
  - a partnerek nemzetisége egyezzen meg,
  - a partnerek minősítése legyen 10%-on belül,
  - a feladatkörök illeszkedjenek,
  - senkinek nem lehet egynél több partnere,
  - akinek nem jut partner, azt kizárták a versenyből.

**Megoldásvázlat (A fentihez hasonló, úgynevezett partnerfeladatok logikája explicit kurzrok használata esetén)**

- Deklaráció
 

```
Kurzor1 (ROWID is szerepeljen lekérdezett oszlopként)
Kurzor2 (ROWID is szerepeljen lekérdezett oszlopként)
```
- BEGIN
 

```
LOOP1
Kurzor1 OPEN
FETCH lekérdezés rekord1-be
Kilépés, ha nincs több
```
- SELECT az eredeti táblából, ahol a ROWID megegyezik, megnézzük a partner mezőt,
 

```
hogy üres-e
```
- LOOP2
 

```
LOOP2
Kurzor2 OPEN
FETCH rekord2-be
Kilépés, ha nincs több
```
- SELECT a partner rekord lekérdezése, azonosítás a ROWID alapján, hogy a partner mezo üres-e

IF feltételek

önmaga ne legyen  
 partner mező legyen üres  
 egyéb feltételek is teljesüljenek  
 (például azonos csapat, minősítési követelmények,  
 feladatkörök stb.)

Módosítás1:

UPDATE, SET partnert beírom a kereső partner mezejébe  
 (WHERE CURRENT OF)

Módosítás2:

UPDATE, SET a megtaláltba beírom a keresőt mint partner  
 (WHERE CURRENT OF)

```
END LOOP1      Belső ciklus lezárása
Kurzor1 CLOSE
END LOOP2      Külső ciklus lezárása
Kurzor2 CLOSE
COMMIT
■ END
```

## 1. megoldás (Explicit kurzorokkal)

### 1.1. lépés (Az adattábla létrehozása)

```
DROP TABLE rally;
```

```
CREATE TABLE rally
( nev          VARCHAR2(25),
  csapat        VARCHAR2(20)  CONSTRAINT csapat_notnull NOT NULL,
  feladatkör    VARCHAR2(9)   CONSTRAINT feladatkör_notnull NOT NULL,
  minősítés     NUMBER(3)    CONSTRAINT minősítés_notnull NOT NULL,
  partner        VARCHAR2(25),
  CONSTRAINT nev_primarykey
    PRIMARY KEY (nev),
  CONSTRAINT feladatkör_check
    CHECK (UPPER(feladatkör)='PILOT' OR
           UPPER(feladatkör)='CO-PILOT'),
  CONSTRAINT minősítés_check
    CHECK (minősítés BETWEEN 1 AND 100) );
COLUMN nev        FORMAT A20
COLUMN csapat     FORMAT A8
COLUMN minősítés FORMAT 9999

INSERT INTO rally
VALUES ('Balint'      , 'HUN'      , 'PILOT'      , 50,      '---');
INSERT INTO rally
VALUES ('Marton'     , 'GBR'      , 'PILOT'      , 72,      '---');
```

```

INSERT INTO rally
    VALUES ('Gabor'      , 'NED'     , 'PILOT'   , 60,   '---');
INSERT INTO rally
    VALUES ('Sandor'    , 'HUN'     , 'PILOT'   , 67,   '---');
INSERT INTO rally
    VALUES ('Suveg'     , 'FRA'     , 'PILOT'   , 57,   '---');
INSERT INTO rally
    VALUES ('Greber'    , 'HUN'     , 'PILOT'   , 80,   '---');
INSERT INTO rally
    VALUES ('Mika'      , 'PL'      , 'CO-PILOT', 55,   '---');
INSERT INTO rally
    VALUES ('Montoya'   , 'PL'      , 'CO-PILOT', 76,   '---');
INSERT INTO rally
    VALUES ('Panis'     , 'GBR'     , 'CO-PILOT', 67,   '---');
INSERT INTO rally
    VALUES ('Kertesz'   , 'HUN'     , 'CO-PILOT', 55,   '---');
INSERT INTO rally
    VALUES ('Otto'      , 'SUBARU', 'CO-PILOT', 67,   '---');
INSERT INTO rally
    VALUES ('Lelkes'    , 'SKODA'   , 'CO-PILOT', 88,   '---');

SELECT * FROM rally;

```

### Eredmény

A tábla eldobva.  
A tábla létrejött.  
1 sor létrejött.  
  
1 sor létrejött.

NEV	CSAPAT	FELADATKÖ	MINŐSITÉS	PARTNER
Balint	HUN	PILOT	50	---
Marton	GBR	PILOT	72	---
Gabor	NED	PILOT	60	---
Sandor	HUN	PILOT	67	---
Suveg	FRA	PILOT	57	---
Greber	HUN	PILOT	80	---
Mika	PL	CO-PILOT	55	---
Montoya	PL	CO-PILOT	76	---
Panis	GBR	CO-PILOT	67	---
Kertesz	HUN	CO-PILOT	55	---
Otto	SUBARU	CO-PILOT	67	---
Lelkes	SKODA	CO-PILOT	88	---

12 sor kijelölve.

### 1.2. lépés (Partnerkeresés)

```

SET serveroutput ON
SET verify ON

DECLARE
    v_pilota_partner      rally.partner%TYPE;
    v_part_partner        rally.partner%TYPE;

CURSOR pilota_kurzor IS
    SELECT nev, csapat, feladatkör, minősítés, ROWID sorazonosito
    FROM rally
    WHERE partner='---'
    FOR UPDATE OF minősítés NOWAIT;
pilota_rekord      pilota_kurzor%ROWTYPE;

CURSOR partner_kurzor IS
    SELECT nev, csapat, feladatkör, minősítés, ROWID sorazonosito
    FROM rally
    WHERE partner = '---'
    FOR UPDATE OF minősítés NOWAIT;
partner_rekord     partner_kurzor%ROWTYPE;

BEGIN
    OPEN pilota_kurzor;
    LOOP
        FETCH pilota_kurzor
        INTO pilota_rekord;
        EXIT WHEN pilota_kurzor%NOTFOUND;
        SELECT partner
            INTO v_pilota_partner
            FROM rally
            WHERE ROWID = pilota_rekord.sorazonosito;
        IF v_pilota_partner = '---'
        THEN
            OPEN partner_kurzor;
            LOOP
                FETCH partner_kurzor
                INTO partner_rekord;
                EXIT WHEN partner_kurzor%NOTFOUND;
                SELECT partner
                    INTO v_part_partner
                    FROM rally
                    WHERE ROWID = partner_rekord.sorazonosito;
            END LOOP;
        END IF;
    END LOOP;
END;

```

```

    IF (partner_rekord.sorazonosito <> pilota_rekord.sorazonosito)
        AND
        (pilota_rekord.minősítés) BETWEEN
            (partner_rekord.minősítés * 0.9) AND
            (partner_rekord.minősítés * 1.1)
        AND
        (partner_rekord.minősítés) BETWEEN
            (pilota_rekord.minősítés * 0.9) AND
            (pilota_rekord. minősítés * 1.1)
        AND
        (v_part_partner = '---')
        AND
        (partner_rekord.feladatkör <> pilota_rekord.feladatkör)
        AND
        (partner_rekord.csapat = pilota_rekord.csapat)
THEN
    DBMS_OUTPUT.PUT_LINE(pilota_rekord.nev||' - '||

                           partner_rekord.nev);
    UPDATE rally
        SET partner = partner_rekord.nev
        WHERE CURRENT OF pilota_kurzor;
    UPDATE rally
        SET partner = pilota_rekord.nev
        WHERE CURRENT OF partner_kurzor;
    EXIT;
END IF;
END LOOP;
CLOSE partner_kurzor;
END IF;
END LOOP;
CLOSE pilota_kurzor;
COMMIT;
END;
/
SELECT * FROM rally;
CLEAR COLUMNS;

```

**Eredmény**

Balint - Kertesz  
 Marton - Panis

A PL/SQL eljárás sikeresen befejeződött.

NEV	CSAPAT	FELADATKÖ	MINŐSITÉS	PARTNER
Balint	HUN	PILOT	50	Kertesz
Marton	GBR	PILOT	72	Panis
Gabor	NED	PILOT	60	---
Sandor	HUN	PILOT	67	---
Suveg	FRA	PILOT	57	---
Greber	HUN	PILOT	80	---
Mika	PL	CO-PILOT	55	---
Montoya	PL	CO-PILOT	76	---
Panis	GBR	CO-PILOT	67	Marton
Kertesz	HUN	CO-PILOT	55	Balint
Otto	SUBARU	CO-PILOT	67	---
Lelkes	SKODA	CO-PILOT	88	---

12 sor kijelölve.

## 2. megoldás (Rejtett kurzorokkal)

### 2.1. lépés. (Az adattábla létrehozása)

DROP TABLE versenyzők;

```
CREATE TABLE versenyzők
( nev          VARCHAR2(20),
  nemzetiség   VARCHAR2(20),
  partner       VARCHAR2(20),
  feladatkör    VARCHAR2(20)
    CHECK (feladatkör IN ('PILOTA', 'NAVIGATOR')),
  minősítés     NUMBER
    CHECK (minősítés>=1 AND minősítés<=1000) );
```

```
INSERT INTO versenyzők VALUES('PILOTA1', 'MAGYAR', NULL, 'PILOTA', 150);
INSERT INTO versenyzők VALUES('PILOTA2', 'MAGYAR', NULL, 'PILOTA', 170);
INSERT INTO versenyzők VALUES('PILOTA3', 'MAGYAR', NULL, 'PILOTA', 190);
INSERT INTO versenyzők VALUES('PILOTA4', 'MAGYAR', NULL, 'PILOTA', 200);
INSERT INTO versenyzők VALUES('PILOTA5', 'MAGYAR', NULL, 'PILOTA', 185);
INSERT INTO versenyzők VALUES('PILOTA6', 'MAGYAR', NULL, 'PILOTA', 150);
INSERT INTO versenyzők VALUES('PILOTA7', 'MAGYAR', NULL, 'PILOTA', 140);
INSERT INTO versenyzők VALUES('PILOTA8', 'MAGYAR', NULL, 'PILOTA', 110);
INSERT INTO versenyzők VALUES('PILOTA9', 'MAGYAR', NULL, 'PILOTA', 120);
INSERT INTO versenyzők VALUES('PILOTA11', 'OLASZ', NULL, 'PILOTA', 150);
INSERT INTO versenyzők VALUES('PILOTA12', 'OLASZ', NULL, 'PILOTA', 160);
INSERT INTO versenyzők VALUES('PILOTA13', 'OLASZ', NULL, 'PILOTA', 170);
INSERT INTO versenyzők VALUES('PILOTA14', 'OLASZ', NULL, 'PILOTA', 180);
INSERT INTO versenyzők VALUES('PILOTA15', 'OLASZ', NULL, 'PILOTA', 190);
```

```
INSERT INTO versenyzők VALUES('PILOTA16','OLASZ',NULL,'PILOTA',200);
INSERT INTO versenyzők VALUES('PILOTA17','OLASZ',NULL,'PILOTA',210);
INSERT INTO versenyzők VALUES('PILOTA18','OLASZ',NULL,'PILOTA',220);
INSERT INTO versenyzők VALUES('PILOTA19','OLASZ',NULL,'PILOTA',230);
INSERT INTO versenyzők VALUES('NAVIGATOR1','MAGYAR',NULL,'NAVIGATOR',155);
INSERT INTO versenyzők VALUES('NAVIGATOR2','MAGYAR',NULL,'NAVIGATOR',171);
INSERT INTO versenyzők VALUES('NAVIGATOR3','MAGYAR',NULL,'NAVIGATOR',190);
INSERT INTO versenyzők VALUES('NAVIGATOR4','MAGYAR',NULL,'NAVIGATOR',200);
INSERT INTO versenyzők VALUES('NAVIGATOR5','MAGYAR',NULL,'NAVIGATOR',185);
INSERT INTO versenyzők VALUES('NAVIGATOR6','MAGYAR',NULL,'NAVIGATOR',100);
INSERT INTO versenyzők VALUES('NAVIGATOR7','MAGYAR',NULL,'NAVIGATOR',100);
INSERT INTO versenyzők VALUES('NAVIGATOR8','MAGYAR',NULL,'NAVIGATOR',110);
INSERT INTO versenyzők VALUES('NAVIGATOR9','MAGYAR',NULL,'NAVIGATOR',129);
INSERT INTO versenyzők VALUES('NAVIGATOR11','OLASZ',NULL,'NAVIGATOR',50);
INSERT INTO versenyzők VALUES('NAVIGATOR12','OLASZ',NULL,'NAVIGATOR',161);
INSERT INTO versenyzők VALUES('NAVIGATOR13','OLASZ',NULL,'NAVIGATOR',10);
INSERT INTO versenyzők VALUES('NAVIGATOR14','OLASZ',NULL,'NAVIGATOR',380);
INSERT INTO versenyzők VALUES('NAVIGATOR15','OLASZ',NULL,'NAVIGATOR',190);
INSERT INTO versenyzők VALUES('NAVIGATOR16','OLASZ',NULL,'NAVIGATOR',200);
INSERT INTO versenyzők VALUES('NAVIGATOR17','OLASZ',NULL,'NAVIGATOR',210);
INSERT INTO versenyzők VALUES('NAVIGATOR18','OLASZ',NULL,'NAVIGATOR',260);
INSERT INTO versenyzők VALUES('NAVIGATOR19','OLASZ',NULL,'NAVIGATOR',230);
SET pagesize 40
COLUMN nev          FORMAT A14
COLUMN nemzet       FORMAT A8
COLUMN minősítés    FORMAT 9999
COLUMN partner      FORMAT A14
COLUMN nemzetiség   FORMAT A 10
SELECT nev,
       nemzetiség AS nemzet,
       partner,
       feladatkör AS feladat,
       minősítés
  FROM versenyzők;
```

### Eredmény

- A tábla eldobva.
- A tábla létrejött.
- 1 sor létrejött.
- ...
- 1 sor létrejött.

NEV	NEMZET	PARTNER	FELADAT	MINŐSITÉS
PILOTA1	MAGYAR		PILOTA	150
PILOTA2	MAGYAR		PILOTA	170
PILOTA3	MAGYAR		PILOTA	190
PILOTA4	MAGYAR		PILOTA	200
PILOTA5	MAGYAR		PILOTA	185
PILOTA6	MAGYAR		PILOTA	150
PILOTA7	MAGYAR		PILOTA	140
PILOTA8	MAGYAR		PILOTA	110
PILOTA9	MAGYAR		PILOTA	120
PILOTA11	OLASZ		PILOTA	150
PILOTA12	OLASZ		PILOTA	160
PILOTA13	OLASZ		PILOTA	170
PILOTA14	OLASZ		PILOTA	180
PILOTA15	OLASZ		PILOTA	190
PILOTA16	OLASZ		PILOTA	200
PILOTA17	OLASZ		PILOTA	210
PILOTA18	OLASZ		PILOTA	220
PILOTA19	OLASZ		PILOTA	230
NAVIGATOR1	MAGYAR		NAVIGATOR	155
NAVIGATOR2	MAGYAR		NAVIGATOR	171
NAVIGATOR3	MAGYAR		NAVIGATOR	190
NAVIGATOR4	MAGYAR		NAVIGATOR	200
NAVIGATOR5	MAGYAR		NAVIGATOR	185
NAVIGATOR6	MAGYAR		NAVIGATOR	100
NAVIGATOR7	MAGYAR		NAVIGATOR	100
NAVIGATOR8	MAGYAR		NAVIGATOR	110
NAVIGATOR9	MAGYAR		NAVIGATOR	129
NAVIGATOR11	OLASZ		NAVIGATOR	50
NAVIGATOR12	OLASZ		NAVIGATOR	161
NAVIGATOR13	OLASZ		NAVIGATOR	10
NAVIGATOR14	OLASZ		NAVIGATOR	380
NAVIGATOR15	OLASZ		NAVIGATOR	190
NAVIGATOR16	OLASZ		NAVIGATOR	200
NAVIGATOR17	OLASZ		NAVIGATOR	210
NAVIGATOR18	OLASZ		NAVIGATOR	260
NAVIGATOR19	OLASZ		NAVIGATOR	230

36 sor kijelölve.

## 2.2. Lépés (Partnerkeresés)

SET serveroutput ON

SET verify ON

```

BEGIN
    FOR i IN (SELECT *
               FROM versenyzők
               WHERE (partner IS NULL) AND
                     (feladatkör = 'PILOTA'))
    LOOP
        FOR j IN (SELECT *
                   FROM versenyzők
                   WHERE (partner IS NULL) AND
                         (feladatkör='NAVIGATOR'))
        LOOP
            IF (i.nemzetiség=j.nemzetiség) AND
                (ABS(i.minősítés-j.minősítés) <=
                 (i.minősítés+j.minősítés)/20)
            THEN
                UPDATE versenyzők
                SET partner = i.nev
                WHERE nev = j.nev;

                UPDATE versenyzők
                SET partner = j.nev
                WHERE nev = i.nev;
                EXIT;
            END IF;
        END LOOP; -- j ciklus
    END LOOP; -- i ciklus

    FOR i IN (SELECT *
               FROM versenyzők
               WHERE (partner IS NULL))
    LOOP
        UPDATE versenyzők
        SET partner = 'KIZARVA'
        WHERE nev = i.nev;
    END LOOP;
END;
/

SELECT nev,
       nemzetiség AS nemzet,
       partner,
       feladatkör AS feladat,
       minősítés
  FROM versenyzők;
CLEAR COLUMNS

```

**Eredmény**

A PL/SQL eljárás sikeresen befejeződött.

NEV	NEMZET	PARTNER	FELADAT	MINŐSITÉS
PILOTA1	MAGYAR	NAVIGATOR1	PILOTA	150
PILOTA2	MAGYAR	NAVIGATOR2	PILOTA	170
PILOTA3	MAGYAR	NAVIGATOR3	PILOTA	190
PILOTA4	MAGYAR	NAVIGATOR4	PILOTA	200
PILOTA5	MAGYAR	NAVIGATOR5	PILOTA	185
PILOTA6	MAGYAR	KIZARVA	PILOTA	150
PILOTA7	MAGYAR	NAVIGATOR9	PILOTA	140
PILOTA8	MAGYAR	NAVIGATOR6	PILOTA	110
PILOTA9	MAGYAR	NAVIGATOR8	PILOTA	120
PILOTA11	OLASZ	NAVIGATOR12	PILOTA	150
PILOTA12	OLASZ	KIZARVA	PILOTA	160
PILOTA13	OLASZ	KIZARVA	PILOTA	170
PILOTA14	OLASZ	NAVIGATOR15	PILOTA	180
PILOTA15	OLASZ	NAVIGATOR16	PILOTA	190
PILOTA16	OLASZ	NAVIGATOR17	PILOTA	200
PILOTA17	OLASZ	NAVIGATOR19	PILOTA	210
PILOTA18	OLASZ	KIZARVA	PILOTA	220
PILOTA19	OLASZ	KIZARVA	PILOTA	230
NAVIGATOR1	MAGYAR	PILOTA1	NAVIGATOR	155
NAVIGATOR2	MAGYAR	PILOTA2	NAVIGATOR	171
NAVIGATOR3	MAGYAR	PILOTA3	NAVIGATOR	190
NAVIGATOR4	MAGYAR	PILOTA4	NAVIGATOR	200
NAVIGATOR5	MAGYAR	PILOTA5	NAVIGATOR	185
NAVIGATOR6	MAGYAR	PILOTA8	NAVIGATOR	100
NAVIGATOR7	MAGYAR	KIZARVA	NAVIGATOR	100
NAVIGATOR8	MAGYAR	PILOTA9	NAVIGATOR	110
NAVIGATOR9	MAGYAR	PILOTA7	NAVIGATOR	129
NAVIGATOR11	OLASZ	KIZARVA	NAVIGATOR	50
NAVIGATOR12	OLASZ	PILOTA11	NAVIGATOR	161
NAVIGATOR13	OLASZ	KIZARVA	NAVIGATOR	10
NAVIGATOR14	OLASZ	KIZARVA	NAVIGATOR	380
NAVIGATOR15	OLASZ	PILOTA14	NAVIGATOR	190
NAVIGATOR16	OLASZ	PILOTA15	NAVIGATOR	200
NAVIGATOR17	OLASZ	PILOTA16	NAVIGATOR	210
NAVIGATOR18	OLASZ	KIZARVA	NAVIGATOR	260
NAVIGATOR19	OLASZ	PILOTA17	NAVIGATOR	230

### 3. megoldás (Gyűjtőtáblával)

#### 3.1. lépés (Az adattábla létrehozása)

```
DROP TABLE versenyzők;
```

```
CREATE TABLE versenyzők
  (nev          VARCHAR2(20)    NOT NULL,
   feladatkör   VARCHAR2(10)    NOT NULL,
   nemzetiség   VARCHAR2(10)    NOT NULL,
   minősítés    NUMBER        NOT NULL,
   partner      VARCHAR2(20),
   CONSTRAINT feladatkör_check_kulcs
     CHECK (feladatkör IN ('NAVIGÁTOR','PILÓTA')),
   CONSTRAINT minősítés_check_kulcs
     CHECK (minősítés BETWEEN 1 AND 1000) );
```

```
INSERT INTO versenyzők
  VALUES('Kiss Péter','PILÓTA','magyar',700,null);
INSERT INTO versenyzők
  VALUES('Nagy Ferenc','NAVIGÁTOR','magyar',893,null);
INSERT INTO versenyzők
  VALUES('Hans Göbel','PILÓTA','német',230,null);
INSERT into versenyzők
  VALUES('Horváth József','PILÓTA','magyar',841,null);
INSERT INTO versenyzők
  VALUES('Bill Christal','NAVIGÁTOR','angol',902,null);
INSERT INTO versenyzők
  VALUES('Philip Daemon','PILÓTA','angol',913,null);
INSERT into versenyzők
  VALUES('Christian Görsch','PILÓTA','német',520,null);
INSERT INTO versenyzők
  VALUES('Jan Peterson','NAVIGÁTOR','német',313,null);
INSERT into versenyzők
  VALUES('Matt Foely','PILÓTA','angol',896,null);

SELECT * FROM versenyzők;
```

#### Eredmény

A tábla eldobva.

A tábla létrejött.

1 sor létrejött.

...

1 sor létrejött.

NEV	FELADATKÖR	NEMZETISÉG	MINŐSITÉS	PARTNER
Kiss Péter	PILÓTA	magyar	700	
Nagy Ferenc	NAVIGÁTOR	magyar	893	
Hans Göbel	PILÓTA	német	230	
Horváth József	PILÓTA	magyar	841	
Bill Christal	NAVIGÁTOR	angol	902	
Philip Daemon	PILÓTA	angol	913	
Christian Görsch	PILÓTA	német	520	
Jan Peterson	NAVIGÁTOR	német	313	
Matt Foely	PILÓTA	angol	896	

9 sor kijelölve.

### 3.2. lépés (Partnerkeresés)

SET serveroutput ON

SET verify ON

DECLARE

```
v_nev          versenyzők.nev%TYPE;
v_feladatkör  versenyzők.feladatkör%TYPE;
v_nemzetiség   versenyzők.nemzetiség%TYPE;
v_minősítés    versenyzők.minősítés%TYPE;
v_partner      versenyzők.partner%TYPE;
```

CURSOR versenyzők\_kurzor

IS

```
SELECT nev, feladatkör, nemzetiség, minősítés, partner
  FROM versenyzők
 FOR UPDATE OF partner NOWAIT;
```

```
TYPE versenyzők_kurzor_tipus IS TABLE OF
  versenyzők_kurzor%ROWTYPE
  INDEX BY BINARY_INTEGER;
```

```
versenyzők_tabla    versenyzők_kurzor_tipus;
sorszám           NUMBER :=1;
```

BEGIN

OPEN versenyzők\_kurzor;

LOOP

```
  FETCH versenyzők_kurzor
    INTO versenyzők_tabla(sorszám);
  EXIT WHEN versenyzők_kurzor%NOTFOUND;
```

```

    sorszam := sorszam + 1;
END LOOP;
CLOSE versenyzők_kurzor;

FOR i IN 1..versenyzők_tabla.COUNT
LOOP
    FOR j IN 1..versenyzők_tabla.COUNT
    LOOP
        IF i<>j AND
            versenyzők_tabla(j).nemzetiség =
            versenyzők_tabla(i).nemzetiség      AND
            versenyzők_tabla(j).minősítés >
            versenyzők_tabla(i).minősítés - 100  AND
            versenyzők_tabla(j).minősítés <
            versenyzők_tabla(i).minősítés + 100  AND
            versenyzők_tabla(j).feladatkör <>
            versenyzők_tabla(i).feladatkör      AND
            versenyzők_tabla(i).partner IS NULL  AND
            versenyzők_tabla(j).partner IS NULL
        THEN
            versenyzők_tabla(i).partner := versenyzők_tabla(j).nev;
            versenyzők_tabla(j).partner := versenyzők_tabla(i).nev;
        END IF;
    END LOOP;
END LOOP;

sorszam := 1;

OPEN versenyzők_kurzor;
LOOP
    FETCH versenyzők_kurzor
        INTO v_nev, v_feladatkör, v_nemzetiség, v_minősítés, v_partner;
    EXIT WHEN versenyzők_kurzor%NOTFOUND;

    IF versenyzők_tabla(sorszam).partner IS NULL
    THEN
        v_partner := 'Kizárvva';
    ELSE
        v_partner := versenyzők_tabla(sorszam).partner;
    END IF;

    UPDATE versenyzők
        SET partner = v_partner
        WHERE CURRENT OF versenyzők_kurzor;

```

```

    sorszam := sorszam + 1;
END LOOP;
CLOSE versenyzők_kurzor;
END;
/

```

SELECT \* FROM versenyzők;

### Eredmény

A PL/SQL eljárás sikeresen befejeződött.

NEV	FELADATKÖR	NEMZETISÉG	MINŐSITÉS	PARTNER
Kiss Péter	PILÓTA	magyar	700	Kizárvá
Nagy Ferenc	NAVIGÁTOR	magyar	893	Horváth József
Hans Göbel	PILÓTA	német	230	Jan Peterson
Horváth József	PILÓTA	magyar	841	Nagy Ferenc
Bill Christal	NAVIGÁTOR	angol	902	Philip Daemon
Philip Daemon	PILÓTA	angol	913	Bill Christal
Christian Görsch	PILÓTA	német	520	Kizárvá
Jan Peterson	NAVIGÁTOR	német	313	Hans Göbel
Matt Foely	PILÓTA	angol	896	Kizárvá

9 sor kijelölve.

### 9.16. feladat

A FÓBUHA vállalat dolgozóinak adatbázissémája:

*dolgozó* (név, üzlet, munkakör, fizetés, gyakorlat, partner)

ahol az attribútumok adattípusai:

- név, partner: karakterSORozatok,
- üzlet: a dolgozót foglalkoztató üzlet neve (TVBolt, PCBolt),
- munkakör: (eladó, szerelő),
- fizetés: pozitív egész szám,
- gyakorlat: a dolgozó ledolgozott éveinek száma (1 és 40 közötti érték),
- a partner inicializált értéke: '---' vagy NULL.

1. Hozza létre a fenti táblát az alábbi megszorításokkal. Adatfelviteli kísérletekkel tesztelje a megszorításokat, és töltse fel a táblát legalább 9 sorral.
  - Legyen a név elsődleges kulcs a *dolgozó* táblában,
  - Egy eladó nem kaphat 50 000 Ft-nál kevesebb és 120 000 Ft-nál több, egy szerelő pedig 80 000 Ft-nál kevesebb és 180 000 Ft-nál több fizetést.

2. Növelje meg a 10 évnél régebb óta dolgozók fizetését a munkakörük átlagának 10%-ával, de legfeljebb az adott munkakörhöz tartozó fizetéstartomány felső értékére, és listázza ki az összes dolgozó minden adatát név szerint rendezve. Ügyeljen arra, hogy a növelési tevékenység során az eredeti fizetésértékek alapján számoljon!
3. Készítsen egy SQL\*Plus programba ágyazott PL/SQL blokkot, amely a felhasználó által megadott üzlet minden, helyettesítővel még nem rendelkező dolgozójának partner mezőjébe beírja egy olyan dolgozónak a nevét,
  - akinek gyakorlati ideje legfeljebb 2 évvel tért el a helyettesítendő dolgozó gyakorlati idejétől,
  - helyettesként még senkihez sincs párosítva (önmagához nem párosítható),
  - a helyettesítendő dolgozóval azonos munkakörben dolgozik,
  - ugyanazon üzlet dolgozója,
  - a „partner” tulajdonság szimmetrikus, tehát ha egy X személy partnere az Y, akkor annak partnere az X.

### Megoldás

```
DROP TABLE dolgozó;
```

```
-- tábla létrehozása megszorítással
```

```
CREATE TABLE dolgozó (
név          VARCHAR2(20) PRIMARY KEY,
üzlet        VARCHAR2(10),
munkakör     VARCHAR2(30),
fizetés      NUMBER(6),
gyakorlat    NUMBER(2),
partner      VARCHAR2(20),
CONSTRAINT fizetés_check CHECK (fizetés BETWEEN 50000 and 120000
                                  AND LOWER(munkakör) = 'eladó'
                                  OR fizetés BETWEEN 80000 and 180000
                                  AND LOWER(munkakör) = 'szerelő'));
```

```
SET linesize 60
```

```
DESC dolgozó
```

```
SET linesize 200
```

```
-- Tábla feltöltése
```

```
INSERT INTO dolgozó
```

```
VALUES('Kis','PCBOLT','eladó',110000,12,null);
```

```
INSERT INTO dolgozó
```

```
VALUES('Nagy','TVBOLT','szerelő',90000,5,null);
```

```
INSERT INTO dolgozó
```

```
VALUES('Sas','TVBOLT','eladó',110000,28,null);
```

```
INSERT INTO dolgozó
```

```
VALUES('Kelemen','PCBOLT','szerelő',160000,21,null);
```

```
INSERT INTO dolgozó
```

```

VALUES('Magyar','PCBOLT','eladó',90000,30,null);
INSERT INTO dolgozó
    VALUES('Tóth','TVBOLT','szerelő',120000,7,null);
INSERT INTO dolgozó
    VALUES('Németh','TVBOLT','szerelő',170000,22,null);
INSERT INTO dolgozó
    VALUES('Varga','PCBOLT','szerelő',130000,19,null);
INSERT INTO dolgozó
    VALUES('Gál','TVBOLT','szerelő',170000,27,null);
INSERT INTO dolgozó
    VALUES('Pék','PCBOLT','eladó',100000,11,null);

COLUMN munkakor FORMAT A10
COLUMN partner FORMAT A10
COLUMN nev      FORMAT A10

SELECT nev        AS nev,
       Üzlet,
       munkakör   AS munkakor,
       fizetés,
       gyakorlat,
       partner
  FROM dolgozó
 ORDER BY nev;

-- A dolgozó fizetésének emelése a munkaköri átlagának 10%-val
SET serveroutput ON
DECLARE
v_név      dolgozó.név%Type;
v_üzlet    dolgozó.üzlet%Type;
v_munkakör dolgozó.munkakör%Type;
v_fizetés  dolgozó.fizetés%Type;
v_gyakorlat dolgozó.gyakorlat%Type;
v_átlag    dolgozó.fizetés%Type;
v_emej    dolgozó.fizetés%Type;

CURSOR dolgozo10 IS
  SELECT név,üzlet, munka.munkakör,fizetés, gyakorlat, munka.átlag
    FROM dolgozó,
         (SELECT munkakör,AVG(fizetés) as átlag
          FROM dolgozó
          GROUP BY munkakör) munka
   WHERE dolgozó.munkakör = munka.munkakör AND
        gyakorlat > 10;

```

```

BEGIN
OPEN dolgozo10;
LOOP
  FETCH dolgozo10
    INTO v_név,v_üzlet, v_munkakör, v_fizetés,v_gyakorlat, v_átlag;
  EXIT WHEN dolgozo10%NOTFOUND;

-- fizetésemelés beállítása
  v_emel := v_fizetés + v_átlag*0.1;

  IF v_munkakör = 'eladó'
  THEN
    IF v_emel > 120000
    THEN
      v_emel := 120000;
      DBMS_OUTPUT.PUT_LINE(v_név||v_munkakör|| v_emel || 'nagyobb');
    END IF;
  ELSIF
    IF v_emel >180000
    THEN
      v_emel := 180000;
      DBMS_OUTPUT.PUT_LINE(v_név||v_munkakör|| v_emel || 'nagyobb');
    END IF;
  END IF;
END IF;

-- módosítás
  UPDATE dolgozó
    SET fizetés = v_emel
    WHERE név = v_név;

  END LOOP;
CLOSE dolgozo10;
END;
/
-- Partner-keresés
UPDATE dolgozó
  SET partner = '***';

ACCEPT g_üzlet Prompt 'Kérem az üzlet nevét: '

DECLARE
  v_üzlet      dolgozó.üzlet%TYPE;
  v_gyakorlat  dolgozó.gyakorlat%TYPE;
  v_dolgozo    dolgozó.partner%TYPE;
  v_helyettesit dolgozó.partner%TYPE;

```

```

CURSOR dolgozo (p_üzlet dolgozó.üzlet%TYPE) IS
  SELECT név, üzlet, munkakör, gyakorlat, ROWID AS rekordazonosito
    FROM dolgozó
   WHERE (üzlet = p_üzlet) AND
         (partner = '***')
  FOR UPDATE NOWAIT;
dolgozorekord dolgozo%ROWTYPE;

CURSOR Helyettes (p_üzlet dolgozó.üzlet%TYPE ) IS
  SELECT név, üzlet, munkakör, gyakorlat, ROWID AS rekordazonosito
    FROM dolgozó
   WHERE (üzlet = p_üzlet) AND
         (partner = '***')
  FOR UPDATE NOWAIT;
helyettesrekord helylettes%ROWTYPE;

BEGIN
  v_üzlet := UPPER('&g_üzlet');
  OPEN dolgozo(v_üzlet);
  LOOP
    FETCH dolgozo
      INTO dolgozorekord;
    EXIT WHEN dolgozo%NOTFOUND;

    SELECT partner
      INTO v_dolgozo
      FROM dolgozó
     WHERE ROWID = dolgozorekord.rekordazonosito;

    IF v_dolgozo = '***'
    THEN
      OPEN Helyettes(v_üzlet);
      LOOP
        FETCH helyettes
          INTO helyettesrekord;
      EXIT WHEN helyettes%NOTFOUND;

      SELECT partner
        INTO v_helyettesit
        FROM dolgozó
       WHERE ROWID = helyettesrekord.rekordazonosito;

      IF dolgozorekord.rekordazonosito <>
         helyettesrekord.rekordazonosito)
        AND

```

```

        (ABS(dolgozorekord.gyakorlat - helyettesrekord.gyakorlat))
          <= 2
        (dolgozorekord.munkakör = helyettesrekord.munkakör) AND
        (v_helyettesit='***')

THEN
  UPDATE dolgozó
    SET partner = helyettesrekord.név
    WHERE CURRENT OF dolgozo;

  UPDATE dolgozó
    SET partner = dolgozorekord.név
    WHERE CURRENT OF helylettes;
  EXIT;
END IF ;
END LOOP;
CLOSE helylettes;
END IF ;
END LOOP;
CLOSE dolgozo;
COMMIT;
END;
/

COLUMN munkakor FORMAT A10
COLUMN partner FORMAT A10
COLUMN nev FORMAT A10

SELECT nev AS nev,
       Üzlet,
       munkakör AS munkakor,
       fizetés,
       gyakorlat,
       partner
  FROM dolgozó;
CLEAR COLUMNS

```

### Eredmény

A tábla eldobva.  
A tábla létrejött.

Név	Üres?	Típus
NÉV	NOT NULL	VARCHAR2(20)
ÜZLET		VARCHAR2(10)
MUNKAKÖR		VARCHAR2(30)

FIZETÉS	NUMBER(6)
GYAKORLAT	NUMBER(2)
PARTNER	VARCHAR2(20)

1 sor létrejött.

:::::

1 sor létrejött.

NEV	ÜZLET	MUNKAKOR	FIZETÉS	GYAKORLAT	PARTNER
Kis	PCBOLT	eladó	120000	12	***
Nagy	TVBOLT	szerelő	90000	5	***
Sas	TVBOLT	eladó	120000	28	***
Kelemen	PCBOLT	szerelő	174000	21	***
Magyar	PCBOLT	eladó	100250	30	***
Tóth	TVBOLT	szerelő	120000	7	***
Németh	TVBOLT	szerelő	180000	22	***
Varga	PCBOLT	szerelő	144000	19	***
Gál	TVBOLT	szerelő	180000	27	***
Pék	PCBOLT	eladó	110250	11	***

10 sor kijelölve.

10 sor módosítva.

Kérem az üzlet névét: PCBOLT

régi 24: v\_üzlet := UPPER('&g\_üzlet');

új 24: v\_üzlet := UPPER('PCBOLT');

A PL/SQL eljárás sikeresen befejeződött.

10 sor kijelölve.

NEV	ÜZLET	MUNKAKOR	FIZETÉS	GYAKORLAT	PARTNER
Kis	PCBOLT	eladó	120000	12	Pék
Nagy	TVBOLT	szerelő	90000	5	***
Sas	TVBOLT	eladó	120000	28	***
Kelemen	PCBOLT	szerelő	174000	21	Varga
Magyar	PCBOLT	eladó	100250	30	***
Tóth	TVBOLT	szerelő	120000	7	***
Németh	TVBOLT	szerelő	180000	22	***
Varga	PCBOLT	szerelő	144000	19	Kelemen
Gál	TVBOLT	szerelő	180000	27	***
Pék	PCBOLT	eladó	110250	11	Kis

10 sor kijelölve.

## 9.17. feladat

A BEPAKOL vállalat konténereibe legfeljebb két tárgy lehető. A tárgyak adatbázissémája:

*tárgyak* {név, megrendelő, típus, súly, sürgősségi érték, másik}

ahol az attribútumok adattípusai:

- név, másik: karaktersorozatok,
- megrendelő: a tárgyat megrendelő vállalkozók neve (Tóth, Kovács),
- típus: a tárgyak típusa (Monitor, TV),
- súly: a tárgyak súlya (pozitív egész szám),
- sürgősségi érték: a leszállítás sürgőssége (1 és 10 közötti érték, 10: a legsürgősebb),
- másik: az adott tárggyal egy konténerbe helyezhető másik tárgy neve,
- inicializált értéke: '---' vagy NULL.

1. Hozza létre a fenti táblát az alábbi megszorításokkal. A megszorításokat tesztelje adatfelviteli kísérletekkel, és töltse fel a táblát legalább 9 sorral.
  - Legyen a megrendelő, a név és a sürgősségi értékben elsődleges kulcs,
  - Egy Monitor súlya nem lehet 3 kp-nál kevesebb, és 13 kp-nál több, egy TV-é pedig 5 kp-nál kevesebb, és 25 kp-nál több.
2. Növelje meg a 10 kp-nál nehezebb tárgyak sürgősségi értékét a megrendelőjükhoz tartozó tárgyak átlagos sürgősségi értékének 8%-ával, de legfeljebb a határértékre, és listázza az összes tárgy minden adatát név szerint rendezve. Ügyeljen arra, hogy a növelési tevékenység során az eredeti sürgősségeket alapján számoljon!
3. Készítsen egy SQL\*Plus programba ágyazott PL/SQL blokkot, amely a felhasználó által megadott megrendelő minden, az együttszállításra még nem társított tárgyának a „másik” mezőjébe beírja egy olyan tárgynak a nevét,
  - amelynek sürgősségi értéke legfeljebb 3-mal tér el a társítandó tárgy sürgősségi értékétől,
  - amely még nincs egy tárgyhoz sem társítva (önmagához nem társítható),
  - a hivatkozási tárggyal azonos típusú,
  - ugyanazon megrendelő tulajdona,
  - a „társ” tulajdonság szimmetrikus, tehát ha egy X tárgy társa az Y, akkor annak társa az X.

### Megoldás

```
SET serveroutput ON
```

```
DROP TABLE tárgyak;
```

```
-- tábla létrehozása megszorítással
CREATE TABLE tárgyak (
  név          VARCHAR2(20),
  megrendelő   VARCHAR2(10),
  típus        VARCHAR2(8),
  súly         NUMBER(4),
  sürgősségi  NUMBER(4,2),    -- Itt kell figyelni, az átlag valós lesz
```

```
másik          VARCHAR2(20),
CONSTRAINT első PRIMARY KEY (név,megrendelő,sürgősségi),
CONSTRAINT súly_check CHECK (súly BETWEEN 3 AND 13 AND
                           típus = 'monitor'
OR súly BETWEEN 5 AND 25 AND típus = 'TV'));
```

```
SET linesize 60
```

```
DESC tárgyak;
```

```
SET linesize 300
```

```
-- Tábla feltöltése
```

```
INSERT INTO tárgyak
VALUES('LG','Kovács','monitor',12,7,null);
INSERT INTO tárgyak
VALUES('Tisza','Tóth','TV',6,5,null);
INSERT INTO tárgyak
VALUES('Kuka','Tóth','monitor',11,2,null);
INSERT INTO tárgyak
VALUES('Duna','Kovács','TV',11,3,null);
INSERT INTO tárgyak
VALUES('Lap','Kovács','monitor',11,10,null);
INSERT INTO tárgyak
VALUES('Mátra','Tóth','monitor',8,8,null);
INSERT INTO tárgyak
VALUES('Orion','Tóth','TV',16,6,null);
INSERT INTO tárgyak
VALUES('GE','Kovács','TV',20,4,null);
INSERT INTO tárgyak
VALUES('Vidi','Kovács','monitor',17,5,null);
INSERT INTO tárgyak
VALUES('Kékes','Tóth','PCBOLT',7,5,null);
INSERT INTO tárgyak
VALUES('Kékes','Tóth','TV',7,5,null);
SELECT * from tárgyak;
```

```
-- A dolgozó fizetésének emelése a munkaköri átlagának 10%-val
```

```
SET serveroutput ON
```

```
DECLARE
```

```
    v_atlag    tárgyak.sürgősségi%TYPE;
    v_emel    tárgyak.sürgősségi%TYPE;
```

```
CURSOR tárgy10 IS
```

```
    SELECT név,tárgyak.megrendelő,típus,súly,
          sürgősségi, rendel.átlag as rendelátlag
```

```

    FROM tárgyak,
        (SELECT megrendelő, AVG(sürgősségi) AS átlag
         FROM tárgyak
         GROUP BY megrendelő) rendel
    WHERE tárgyak.megrendelő = rendel.megrendelő AND
          súly > 10;
    tárgyrekord TÁRGY10%ROWTYPE;           --kurzornak megfelelő rekord

BEGIN
    OPEN TÁRGY10;
    LOOP
        FETCH TÁRGY10
        INTO tárgyrekord;
        EXIT WHEN TÁRGY10%NOTFOUND;

        V_ATLAG := TÁRGYREKORD.RENDELÁTLAG*0.08;
-- DBMS_OUTPUT.PUT_LINE('atlag '||V_ATLAG);
        V_EMEL := TÁRGYREKORD.SÜRGŐSSÉG + V_ATLAG;
        DBMS_OUTPUT.PUT_LINE(TÁRGYREKORD.NÉV||' '|| V_EMEL );
        IF V_EMEL > 10
        THEN V_EMEL := 10;
        END IF;

        UPDATE TÁRGYAK
            SET SÜRGŐSSÉG = V_EMEL
            WHERE NÉV = TÁRGYREKORD.NÉV;

    END LOOP;
    CLOSE TÁRGY10;
END;
/

```

SELECT \* FROM TÁRGYAK  
ORDER BY NÉV;

**Eredmény**

A tábla eldobva.  
A tábla létrejött.

Név	Üres?	Típus
NÉV	NOT NULL	VARCHAR2(20)
MEGRENDELŐ	NOT NULL	VARCHAR2(10)
TÍPUS		VARCHAR2(8)

SÚLY	NUMBER(4)
SÜRGÖSSÉG	NOT NULL NUMBER(4,2)
MÁSIK	VARCHAR2(20)

1 sor létrejött.

:::::::

1 sor létrejött.

INSERT INTO tárgyak

\*

Hiba a(z) 1. sorban:

ORA-02290: ellenőrző megszorítás (SCOTT.SÚLY\_CHECK) megsértése

INSERT INTO tárgyak

\*

Hiba a(z) 1. sorban:

ORA-02290: ellenőrző megszorítás (SCOTT.SÚLY\_CHECK) megsértése

1 sor létrejött.

NÉV	MEGRENDELŐ	TÍPUS	SÚLY	SÜRGÖSSÉG	MÁSIK
LG	Kovács	monitor	12		7
Tisza	Tóth	TV	6		5
Kuka	Tóth	monitor	11		2
Duna	Kovács	TV	11		3
Lap	Kovács	monitor	11		10
Mátra	Tóth	monitor	8		8
Orion	Tóth	TV	16		6
GE	Kovács	TV	20		4
Kékes	Tóth	TV	7		5

9 sor kijelölve.

Kuka 2.42

Orion 6.42

LG 7.48

Duna 3.48

Lap 10.48

GE 4.48

A PL/SQL eljárás sikeresen befejeződött.

NÉV	MEGREDELŐ	TÍPUS	SÚLY	SÜRGŐSSÉG	MÁSIK
Duna	Kovács	TV	11	3.48	
GE	Kovács	TV	20	4.48	
Kékes	Tóth	TV	7	5	
Kuka	Tóth	monitor	11	2.42	
Lap	Kovács	monitor	11	10	
LG	Kovács	monitor	12	7.48	
Mátra	Tóth	monitor	8	8	
Orion	Tóth	TV	16	6.42	
Tisza	Tóth	TV	6	5	

9 sor kijelölve.

A feladatsor harmadik részét önálló feldolgozásra javasoljuk, az előző két példa partner jelleű feladatrészei alapján.

## 10. FEJEZET

# Kivételkezelés, alprogramok, triggerek

## Feladatok és megoldások

### 10.1. feladat

Írjon triggert (és ellenőrizze is működését), amely az alábbi feltételek teljesülése esetén engedélyezi az adatbevitelt, míg e feltételek nem teljesülése esetén az adatbevitelt megtagadja. minden feltétel ellenőrzését külön alprogrammal valósítsa meg. A trigger tartalmazzon kivételkezelést, és a hibaüzenetek is ebben szerepeljenek. A feltételek az alábbiak:

- egy dolgozóhoz csak már tárolt dolgozó azonosítója adható meg főnök-kódként,
- csak olyan munkakör adható meg, amely már szerepel,
- új dolgozó az azonos munkakörű társainál csak alacsonyabb fizetést kaphat.

### Előkészület a megoldáshoz

PRÖMPT A munkatábla törlése és létrehozása:

DROP TABLE dolgozó;

CREATE TABLE dolgozó

AS SELECT \* FROM emp;

PROMPT Eljárások, függvények és triggerek lekérdezése:

SELECT object\_type, object\_name

FROM user\_objects

WHERE object\_type IN ('PROCEDURE', 'FUNCTION', 'TRIGGER');

### Eredmény

A munkatábla törlése és létrehozása:

A tábla eldobva.

A tábla létrejött.

Eljárások, függvények és triggerek lekérdezése:

nincsenek kijelölve sorok

## Megoldás

### 1. lépés (A megadott azonosítóhoz tartozik-e már valaki)

```

CREATE OR REPLACE FUNCTION Feltétel_mgr(x_mgr IN NUMBER)
RETURN BOOLEAN
IS
    dolgozószám  INTEGER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> A főnök-kód vizsgálata:');
    SELECT COUNT(*)
        INTO dolgozószám
       FROM dolgozó
      WHERE empno = x_mgr;
    IF dolgozószám = 0
    THEN
        RETURN FALSE;
    ELSE
        RETURN TRUE;
    END IF;
END;
/
SHOW ERRORS

```

## Eredmény

A függvény létrejött.

Nincsenek hibák.

### 2. lépés (Létezik-e a megadott munkakör)

```

CREATE OR REPLACE FUNCTION Feltétel_job(x_job IN CHAR)
RETURN BOOLEAN
IS
    dolgozószám  INTEGER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> A munkakör vizsgálata:');
    SELECT COUNT(*)
        INTO dolgozószám
       FROM dolgozó
      WHERE UPPER(job) = UPPER(x_job);
    IF dolgozószám = 0
    THEN
        RETURN FALSE;
    ELSE
        RETURN TRUE;
    END IF;
END;

```

```
/  
SHOW ERRORS
```

### Eredmény

A függvény létrejött.

Nincsenek hibák.

### 3. lépés (Az új dolgozó fizetése valóban alacsonyabb-e)

```
CREATE OR REPLACE FUNCTION Feltétel_sal(x_job IN CHAR,  
                                         x_sal IN NUMBER)  
RETURN BOOLEAN  
IS  
    dolgozószám INTEGER;  
BEGIN  
    DBMS_OUTPUT.PUT_LINE('=> A fizetés vizsgálata:');  
    -- Ha az azonos munkakörük között van kisebb fizetésű,  
    -- akkor a feltétel nem teljesül.  
    -- (Új dolgozó csak alacsonyabb fizetésű lehet).  
    SELECT COUNT(*)  
        INTO dolgozószám  
        FROM dolgozó  
        WHERE UPPER(job) = UPPER(x_job) AND  
              sal <= x_sal;  
    DBMS_OUTPUT.PUT_LINE('=> Kisebb fizetésűek a'||  
                         x_job||' munkakörben: '|| dolgozószám);  
    IF dolgozószám = 0  
    THEN  
        RETURN TRUE;  
    ELSE  
        RETURN FALSE;  
    END IF;  
END;  
/  
SHOW ERRORS
```

### Eredmény

A függvény létrejött.

Nincsenek hibák.

### 4. lépés (A trigger törlése és létrehozása)

```
DROP TRIGGER Ellenőr;
```

```
CREATE OR REPLACE TRIGGER Ellenőr  
BEFORE INSERT ON dolgozó
```

```

FOR EACH ROW
DECLARE
    hiba_mgr EXCEPTION;
    hiba_job EXCEPTION;
    hiba_sal EXCEPTION;
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger aktiv...');

    IF NOT Feltétel_mgr(:new.mgr)
    THEN
        RAISE hiba_mgr;
    END IF;
    IF NOT Feltétel_job(:new.job)
    THEN
        RAISE hiba_job;
    END IF;
    IF NOT Feltétel_sal(:new.job, :new.sal)
    THEN
        RAISE hiba_sal;
    END IF;

    EXCEPTION
        WHEN hiba_mgr THEN
            RAISE_APPLICATION_ERROR(-20101,'>> Sérült a "főnök-kód" feltétel!');
        WHEN hiba_job THEN
            RAISE_APPLICATION_ERROR(-20102,'>> Sérült a "munkakör" feltétel!');
        WHEN hiba_sal THEN
            RAISE_APPLICATION_ERROR(-20103,'>> Sérült a "fizetés" feltétel!');
END;
/
SHOW ERRORS

```

### Eredmény

A trigger eldobva.  
A trigger létrejött.  
Nincsenek hibák.

### 5. lépés (A trigger működésének ellenőrzése)

```

SET serveroutput ON
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger tesztelése...');

END;
/

```

PROMPT A főnök-kód hibás:

```
INSERT INTO dolgozó
VALUES (1234,'KISS','CLERK',1111,'99-MÁJ-20',750,NULL,10);
```

PROMPT A foglalkozás hibás:

```
INSERT INTO dolgozó
VALUES (1234,'KISS','Tanár',7499,'99-MÁJ-20',750,NULL,10);
```

PROMPT A fizetés hibás:

```
INSERT INTO dolgozó
VALUES (1234,'KISS','CLERK',7499,'99-MÁJ-20',1000,NULL,10);
```

PROMPT Helyes adatmegadás:

```
INSERT INTO dolgozó
VALUES (1234,'KISS','CLERK',7499,'99-MÁJ-20',750,NULL,10);
```

BEGIN

```
    DBMS_OUTPUT.PUT_LINE('=> A trigger tesztelése befejezve...');
```

END;

/

### Eredmény

=> Az "Ellenőr" trigger tesztelése...

A PL/SQL eljárás sikeresen befejeződött.

A főnök-kód hibás:

```
INSERT INTO dolgozó
*
```

Hiba a(z) 1. sorban:

ORA-20101: >> Sérült a "főnök-kód" feltétel!

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 22. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

A foglalkozás hibás:

```
INSERT INTO dolgozó
*
```

Hiba a(z) 1. sorban:

ORA-20102: >> Sérült a "munkakör" feltétel!

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 24. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

A fizetés hibás:

```
INSERT INTO dolgozó
```

\*

Hiba a(z) 1. sorban:

ORA-20103: >> Sérült a "fizetés" feltétel!

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 26. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

Helyes adatmegadás:

=> Az "Ellenőr" trigger aktív...

=> A főnök-kód vizsgálata:

=> Az "Ellenőr" trigger aktív...

=> A főnök-kód vizsgálata:

=> A munkakör vizsgálata:

=> Az "Ellenőr" trigger aktív...

=> A főnök-kód vizsgálata:

=> A munkakör vizsgálata:

=> A fizetés vizsgálata:

=> Kisebb fizetésűek a CLERK munkakörben: 2

=> Az "Ellenőr" trigger aktív...

=> A főnök-kód vizsgálata:

=> A munkakör vizsgálata:

=> A fizetés vizsgálata:

=> Kisebb fizetésűek a CLERK munkakörben: 0

1 sor létrejött.

=> A trigger tesztelése befejezve...

A PL/SQL eljárás sikeresen befejeződött.

## 6. Lépés (A módosított dolgozó tábla listázása)

PROMPT A módosított dolgozó tábla:

SET numwidth 5

SELECT \*

FROM dolgozó

ORDER BY ename;

SET numwidth 10

## Eredmény

A módosított dolgozó tábla:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30

7782 CLARK	MANAGER	7839	81-JÚN-09	2450	10
7902 FORD	ANALYST	7566	81-DEC-03	3000	20
7900 JAMES	CLERK	7698	81-DEC-03	950	30
7566 JONES	MANAGER	7839	81-ÁPR-02	2975	20
7839 KING	PRESIDENT		81-NOV-17	5000	10
1234 KISS	CLERK	7499	99-MÁJ-20	750	10
7654 MARTIN	SALESMAN	7698	81-SZE-28	1250	1400
7934 MILLER	CLERK	7782	82-JAN-23	1300	10
7788 SCOTT	ANALYST	7566	82-DEC-09	3000	20
7369 SMITH	CLERK	7902	80-DEC-17	800	20
7844 TURNER	SALESMAN	7698	81-SZE-08	1500	0
7521 WARD	SALESMAN	7698	81-FEB-22	1250	500
					30

15 sor kijelölve.

## 7. lépés (A létrehozott objektumok törlése)

```
DROP FUNCTION Feltétel_mgr;
DROP FUNCTION Feltétel_job;
DROP FUNCTION Feltétel_sal;
DROP TRIGGER Ellenőr;
```

### Eredmény

A függvény eldobva.  
 A függvény eldobva.  
 A függvény eldobva.  
 A trigger eldobva.



#### Megjegyzés

- A triggeret azért célszerű a létrehozása előtt törölni, mert a CREATE OR REPLACE előírás ellenére nem történik meg a régi változat felülírása, ha az például másik adattáblára vonatkozott.
- Figyeljünk fel a kiíratások sorrendjére! A RAISE\_APPLICATION\_ERROR eljárás a DBMS\_OUTPUT.PUT\_LINE eljárással történő kiíratásokat zárolja, és e zárolt memóriaterületről (spool) a szöveget csak egy névtelen blokkban történő kiíratás szabádítja fel, és engedi a képernyőn megjelenni. Ennek az a célja, hogy a trigger működése minél hatékonyabb legyen.
- A tesztelések során ügyeljünk arra, hogy az adattáblára definiált (BEFORE és AFTER) triggerek az adattábla törlésekor maguk is törlődnek. (A tárolt eljárások nem!)

## 10.2. feladat

Írjon triggeret (és ellenőrizze is működését), amely az alábbi feltételek teljesülése esetén engedélyezi az adatbevitelt, míg e feltételek nem teljesülése esetén az adatbevitelt megtagadja. minden feltétel ellenőrzését külön alprogrammal valósítsa meg, és ezek tartalmazzák a kivételkezelést, valamint a hibaüzenet kiírását is. A feltételek az alábbiak:

- egy dolgozóhoz csak már tárolt dolgozó azonosítója adható meg főnök-kódként,
- csak olyan részlegazonosító adható meg, amely már szerepel,
- új dolgozó az azonos részlegbeli társainál csak alacsonyabb fizetést kaphat.

### Előkészület a megoldáshoz

PROMPT A munkatáblák törlése és létrehozása:

```
DROP TABLE dolgozo;
```

```
DROP TABLE részleg;
```

```
CREATE TABLE dolgozo
```

```
AS SELECT * FROM emp;
```

```
CREATE TABLE részleg
```

```
AS SELECT * FROM dept;
```

PROMPT Eljárások, függvények és triggerek lekérdezése:

```
SELECT object_type, object_name
```

```
FROM user_objects
```

```
WHERE object_type IN ('PROCEDURE', 'FUNCTION', 'trigger');
```

### Eredmény

A munkatábla törlése és létrehozása:

A tábla eldobva.

A tábla eldobva.

A tábla létrejött.

A tábla létrejött.

Eljárások, függvények és triggerek lekérdezése:

nincsenek kijelölve sorok

### Megoldás

#### 1. lépés (A főnök-kód ellenőrzése)

```
CREATE OR REPLACE PROCEDURE Feltétel_mgr(x_mgr IN NUMBER)
```

```
AS
```

```
    dolgozoszám INTEGER;
```

```
BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('=> A főnök-kód vizsgálata:');
```

```
    SELECT COUNT(*)
```

```
        INTO dolgozoszám
```

```
        FROM dolgozó
```

```

WHERE empno = x_mgr;
IF dolgozószám = 0
THEN
    RAISE_APPLICATION_ERROR(-20201,'>> Sérült a "főnök-kód" feltétel!');
END IF;
END;
/
SHOW ERRORS

```

### Eredmény

A függvény létrejött.

Nincsenek hibák.

### 2. lépés (A részlegazonosító ellenőrzése)

```

CREATE OR REPLACE PROCEDURE Feltétel_deptno(x_deptno IN NUMBER)
AS
    dolgozószám INTEGER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> A részlegazonosító vizsgálata:');
    SELECT COUNT(*)
        INTO dolgozószám
        FROM dolgozó
        WHERE deptno = x_deptno;
    IF dolgozószám = 0
    THEN
        RAISE_APPLICATION_ERROR(-20202,'>> Sérült a "részleg" feltétel!');
    END IF;
END;
/
SHOW ERRORS

```

### Eredmény

A függvény létrejött.

Nincsenek hibák.

### 3. lépés (A fizetés ellenőrzése)

```

CREATE OR REPLACE PROCEDURE Feltétel_sal(x_deptno IN NUMBER,
                                         x_sal     IN NUMBER)
AS
    minfiz  dolgozó.sal%TYPE;
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> A fizetés vizsgálata:');
    -- Ha a részlegben a legkisebb fizetés nem nagyobb,
    -- akkor a feltétel nem teljesül.

```

```
-- (Új dolgozó csak alacsonyabb fizetésű lehet),
SELECT MIN(sal)
  INTO minfiz
  FROM dolgozó
 GROUP BY deptno
 HAVING deptno = x_deptno;
DBMS_OUTPUT.PUT_LINE('=> A legalacsonyabb fizetés a '|||
                      x_deptno||' jelű részlegben: '|| minfiz);
IF minfiz <= x_sal
THEN
  RAISE_APPLICATION_ERROR(-20203,'>> Sérült a "fizetés" feltétele!');

END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20204,'>> Sérült a "részleg" feltétele!');

END;
/
SHOW ERRORS
```

**Eredmény**

A függvény létrejött.

Nincsenek hibák.

**4. lépés (A trigger törlése és létrehozása)**

```
DROP TRIGGER Ellenőr;
```

```
CREATE OR REPLACE TRIGGER Ellenőr
BEFORE INSERT ON dolgozó
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger aktiv...');

  Feltétel_mgr(:NEW.mgr);
  Feltétel_deptno(:NEW.deptno);
  Feltétel_sal(:NEW.deptno, :NEW.sal);

END;
/
SHOW ERRORS
```

**Eredmény**

A trigger eldobva.

A trigger létrejött.

Nincsenek hibák.

### 5. lépés (A trigger működésének ellenőrzése)

```
SET serveroutput ON
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger tesztelése...');
END;
/
```

PROMPT A főnök-kód hibás:

```
INSERT INTO dolgozó
    VALUES (1234,'KISS','CLERK',1111,'99-MÁJ-20',1200,NULL,10);
```

PROMPT A részlegazonosító hibás:

```
INSERT INTO dolgozó
    VALUES (1234,'KISS','CLERK',7499,'99-MÁJ-20',1200,NULL,15);
```

PROMPT A fizetés hibás:

```
INSERT INTO dolgozó
    VALUES (1234,'KISS','CLERK',7499,'99-MÁJ-20',1500,NULL,10);
```

PROMPT Helyes adatmegadás:

```
INSERT INTO dolgozó
    VALUES (1234,'KISS','CLERK',7499,'99-MÁJ-20',1200,NULL,10);
```

BEGIN

```
    DBMS_OUTPUT.PUT_LINE('=> A trigger tesztelése befejezve...');
```

END;

/

### Eredmény

=> Az "Ellenőr" trigger tesztelése...

A PL/SQL eljárás sikeresen befejeződött.

A főnök-kód hibás:

```
INSERT INTO dolgozó
    *
```

Hiba a(z) 1. sorban:

ORA-20201: >> Sérült a "főnök-kód" feltétele!

ORA-06512: a(z) "SCOTT.FELTÉTEL\_MGR", helyen a(z) 12. sornál

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 3. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

A részlegazonosító hibás:

```
INSERT INTO dolgozó
    *
```

Hiba a(z) 1. sorban:

ORA-20202: >> Sérült a "részleg" feltétel!  
 ORA-06512: a(z) "SCOTT.FELTÉTEL\_DEPTNO", helyen a(z) 12. sornál  
 ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 4. sornál  
 ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

A fizetés hibás:

INSERT INTO dolgozó

\*

Hiba a(z) 1. sorban:

ORA-20203: >> Sérült a "fizetés" feltétel!  
 ORA-06512: a(z) "SCOTT.FELTÉTEL\_SAL", helyen a(z) 19. sornál  
 ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 5. sornál  
 ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

Helyes adatmegadás:

=> Az "Ellenőr" trigger aktív...  
 => A főnök-kód vizsgálata:  
 => Az "Ellenőr" trigger aktív...  
 => A főnök-kód vizsgálata:  
 => A részlegazonosító vizsgálata:  
 => Az "Ellenőr" trigger aktív...  
 => A főnök-kód vizsgálata:  
 => A részlegazonosító vizsgálata:  
 => A fizetés vizsgálata:  
 => A legalacsonyabb fizetés a 10 jelű részlegben: 1300  
 => Az "Ellenőr" trigger aktív...  
 => A főnök-kód vizsgálata:  
 => A részlegazonosító vizsgálata:  
 => A fizetés vizsgálata:  
 => A legalacsonyabb fizetés a 10 jelű részlegben: 1300

1 sor létrejött.

=> A trigger tesztelése befejezve...

A PL/SQL eljárás sikeresen befejeződött.

## 6. lépés (A módosított dolgozó tábla listázása)

PROMPT A módosított dolgozó tábla:  
 SET numwidth 5  
 SELECT \*  
 FROM dolgozó  
 ORDER BY ename;  
 SET numwidth 10

**Eredmény**

A módosított dolgozó tábla:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
1234	KISS	CLERK	7499	99-MÁJ-20	1200		10
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7934	MILLER	CLERK	7782	82-JAN-23	1300		10
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30

15 sor kijelölve.

**7. lépés (A létrehozott objektumok törlése)**

```
DROP PROCEDURE Feltétel_mgr;
DROP PROCEDURE Feltétel_deptno;
DROP PROCEDURE Feltétel_sal;
DROP TRIGGER Ellenőr;
```

**Eredmény**

- A függvény eldobva.
- A függvény eldobva.
- A függvény eldobva.
- A trigger eldobva.

**Megjegyzés**

Az előző feladathoz képest a lényeges különbség az, hogy a hiba kezelését és kiírását ezúttal a tárolt függvények végezték, ami által a trigger egyszerűbb lett. Az egyszerűbb trigger hatékonyabb végrehajtást is jelent, ezért a két módszer közül az utóbbi a célszerűbb.

### 10.3. feladat

Írjon triggeret (és ellenőrizze is működését), amely megakadályozza, hogy a felhasználó olyan dolgozót

- törljön,
- beszúrjon, vagy
- lásson el fizetésemeléssel,

akinek havi béré előre eléri vagy meghaladja részlegének átlagfizetését.

#### Előkészület a megoldáshoz

PROMPT A munkatáblák törlése és létrehozása:

```
DROP TABLE dolgozo;
DROP TABLE részleg;
CREATE TABLE dolgozo
AS SELECT * FROM emp;
CREATE TABLE részleg
AS SELECT * FROM dept;
```

PROMPT Eljárások, függvények és triggerek lekérdezése:

```
SELECT object_type, object_name
FROM user_objects
WHERE object_type IN ('PROCEDURE', 'FUNCTION', 'TRIGGER');
```

#### Eredmény

A munkatábla törlése és létrehozása:

```
A tábla eldobva.
A tábla eldobva.
A tábla létrejött.
A tábla létrejött.
```

Eljárások, függvények és triggerek lekérdezése:

nincsenek kijelölve sorok

#### 1. megoldás (Segéd-adattáblával, BEFORE triggerrel)

##### 1.1. lépés (Segédtábla létrehozása)

```
DROP TABLE Segéd;
CREATE TABLE Segéd
AS SELECT * FROM dolgozo;
```

#### Eredmény

```
A tábla eldobva.
A tábla létrejött.
```

**1.2.1. lépés (RészlegÁtlag tárolt függvény létrehozása – munkatáblára. Használható INSERT művelethez)**

```
CREATE OR REPLACE FUNCTION RészlegÁtlag_d(RészlegID NUMBER)
RETURN NUMBER
IS
    v_átlag NUMBER(7,2);

BEGIN
    DBMS_OUTPUT.PUT_LINE('=> RészlegÁtlag számítás - Munkatáblára');
    SELECT AVG(sal)
        INTO v_átlag
        FROM dolgozó
        GROUP BY deptno
        HAVING deptno = RészlegID;
    RETURN v_átlag;
END;
/
SHOW ERRORS
```

**Eredmény**

A függvény létrejött.

Nincsenek hibák.

**1.2.2. lépés (RészlegÁtlag tárolt függvény létrehozása – segédtáblára. Használható DELETE, INSERT és UPDATE műveletekhez)**

```
CREATE OR REPLACE FUNCTION RészlegÁtlag_s(RészlegID NUMBER)
RETURN NUMBER
IS
    v_átlag NUMBER(7,2);

BEGIN
    DBMS_OUTPUT.PUT_LINE('=> RészlegÁtlag számítás - Segédtáblára');
    SELECT AVG(sal)
        INTO v_átlag
        FROM Segéd
        GROUP BY deptno
        HAVING deptno = RészlegID;
    RETURN v_átlag;
END;
/
SHOW ERRORS
```

**Eredmény**

A függvény létrejött.

Nincsenek hibák.

**1.3. lépés (A BEFORE trigger létrehozása)**

```
DROP TRIGGER Ellenőr;
```

```
CREATE OR REPLACE TRIGGER Ellenőr  
BEFORE DELETE OR INSERT OR UPDATE ON dolgozó  
FOR EACH ROW
```

```
DECLARE  
    v_átlag      NUMBER;  
    TörlésHiba   EXCEPTION;  
    BeszúrásHiba EXCEPTION;  
    EmelésHiba   EXCEPTION;  
  
BEGIN  
    DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger aktív...');  
    IF DELETING THEN  
        IF :OLD.sal >= RészlegÁtlag_s(:OLD.deptno)  
        THEN  
            RAISE TörlésHiba;  
        END IF;  
    ELSIF INSERTING THEN  
        IF :NEW.sal >= RészlegÁtlag_d(:NEW.deptno)  
        THEN  
            RAISE BeszúrásHiba;  
        END IF;  
    ELSIF UPDATING THEN  
        IF :OLD.sal >= RészlegÁtlag_s(:OLD.deptno)  
        THEN  
            RAISE EmelésHiba;  
        END IF;  
    END IF;  
EXCEPTION  
    WHEN TörlésHiba    THEN  
        RAISE_APPLICATION_ERROR(-20001,'>> Nem lehet törölni...');  
    WHEN BeszúrásHiba THEN  
        RAISE_APPLICATION_ERROR(-20002,'>> Nem lehet beszúrni...');  
    WHEN EmelésHiba   THEN  
        RAISE_APPLICATION_ERROR(-20003,'>> Nem lehet emelni...');  
END;  
/  
SHOW ERRORS
```

## Eredmény

A trigger eldobva.  
 A trigger létrejött.  
 Nincsenek hibák.

### 1.4. lépés (Ellenőrzés a törléshez)

```
-- Tesztelő program eleje
SET serveroutput ON
SET verify OFF
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger tesztelése...');

END;
/
-- Tesztelő program magjának eleje
ACCEPT Név      PROMPT "A dolgozó neve: "
DELETE FROM dolgozó
  WHERE UPPER(ename) = UPPER('&Név');
-- Tesztelő program magjának vége

SET verify ON
UNDEFINE Név
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> A trigger tesztelése befejezve...');

END;
/
-- Tesztelő program vége
```

### 1.4. 1. futási eredmény (A teljes tesztelő program futtatása esetén)

=> Az "Ellenőr" trigger tesztelése...
 A PL/SQL eljárás sikeresen befejeződött.

A dolgozó neve: Blake

DELETE FROM dolgozó

\*

Hiba a(z) 1. sorban:

ORA-20001: >> Nem lehet törölni...

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 26. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

=> Az "Ellenőr" trigger aktív...

=> RészlegÁtlag számítás - Segédtáblára

=> A trigger tesztelése befejezvé...

A PL/SQL eljárás sikeresen befejeződött.

#### 1.4. 2. futási eredmény (A tesztelő program magjának futtatása esetén)

A dolgozó neve: Smith  
 => Az "Ellenőr" trigger aktív...

1. sor törölve.

#### 1.5. lépés (Ellenőrzés a beszúráshoz)

```
-- Tesztelő program eleje
SET serveroutput ON
SET verify OFF
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger tesztelése...');

END;
/
-- Tesztelő program magjának eleje
ACCEPT Név      PROMPT "    A dolgozó neve: "
ACCEPT Azonosító PROMPT "    azonosítója: "
ACCEPT Fizetés   PROMPT "    fizetése: "
ACCEPT Részleg   PROMPT "részlegazonosítója: "
INSERT INTO dolgozó (empno, ename, sal, deptno)
  VALUES (&Azonosító, '&Név', &Fizetés, &Részleg);
-- Tesztelő program magjának vége

SET verify ON
UNDEFINE Név
UNDEFINE Azonosító
UNDEFINE Fizetés
UNDEFINE Részleg
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> A trigger tesztelése befejezve...');

END;
/
-- Tesztelő program vége
```

#### 1.5. 1. futási eredmény (A teljes tesztelő program futtatása esetén)

=> Az "Ellenőr" trigger tesztelése...
 A PL/SQL eljárás sikeresen befejeződött.

A dolgozó neve: Szundi
 azonosítója: 1111
 fizetése: 3500
 részlegazonosítója: 30
 INSERT INTO dolgozó (empno, ename, sal, deptno)

\*

Hiba a(z) 1. sorban:

ORA-20002: >> Nem lehet beszúrni...

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 28. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

=> Az "Ellenőr" trigger aktív...

=> RészlegÁtlag számítás - Munkatáblára

=> A trigger tesztelése befejezve...

A PL/SQL eljárás sikeresen befejeződött.

### 1.5. 2. futási eredmény (A tesztelő program magjának futtatása esetén)

A dolgozó neve: Vidor

azonosítója: 2222

fizetése: 1500

részlegazonosítója: 10

=> Az "Ellenőr" trigger aktív...

1 sor létrejött.

### 1.6. lépés (Ellenőrzés a módosításhoz)

```
-- Tesztelő program eleje
SET serveroutput ON
SET verify OFF
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger tesztelése...');

END;
/


-- Tesztelő program magjának eleje
ACCEPT Név      PROMPT "A dolgozó neve: "
ACCEPT Fizetés  PROMPT "Fizetésemelése: "
UPDATE dolgozó
    SET sal = sal + &Fizetés
    WHERE UPPER(ename) = UPPER('&Név');
-- Tesztelő program magjának vége

SET verify ON
UNDEFINE Név
UNDEFINE Fizetés
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> A trigger tesztelése befejezve...');

END;
/
-- Tesztelő program vége
```

**1.6. 1. futási eredmény (A teljes tesztelő program futtatása esetén)**

=> Az "Ellenőr" trigger tesztelése...

A PL/SQL eljárás sikeresen befejeződött.

A dolgozó neve: Blake

Fizetésemelése: 111

UPDATE dolgozó

\*

Hiba a(z) 1. sorban:

ORA-20003: >> Nem lehet emelni...

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 30. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

=> Az "Ellenőr" trigger aktív...

=> RészlegÁtlag számítás - Segédtáblára

=> A trigger tesztelése befejezve...

A PL/SQL eljárás sikeresen befejeződött.

**1.6. 2. futási eredmény (A tesztelő program magjának futtatása esetén)**

A dolgozó neve: Adams

Fizetésemelése: 111

=> Az "Ellenőr" trigger aktív...

1 sor módosítva.

**1.7. lépés (A munkatábla listázása)**

SET numwidth 5

SELECT \*

FROM dolgozó

ORDER BY ename;

SET numwidth 10

**Eredmény**

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	87-MÁJ-23	1211		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7839	KING	PRESIDENT		81-NOV-17	5000		10
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30

7934 MILLER	CLERK	7782	82-JAN-23	1300	10
7788 SCOTT	ANALYST	7566	87-ÁPR-19	3000	20
7844 TURNER	SALESMAN	7698	81-SZE-08	1500	0
2222 Vidor				1500	10
7521 WARD	SALESMAN	7698	81-FEB-22	1250	500

14 sor kijelölve.

### 1.8. lépés (A létrehozott objektumok törlése)

```
DROP FUNCTION RészlegÁtlag_d;
DROP FUNCTION RészlegÁtlag_s;
DROP TRIGGER Ellenőr;
DROP TABLE Segéd;
```

#### Eredmény

- A függvény eldobva.
- A függvény eldobva.
- A trigger eldobva.
- A tábla eldobva.



#### Megjegyzés

A DELETE és UPDATE DML-utasítások esetén a dolgozó munkatáblára létrehozott BEFORE trigger számára ez a tábla nem látható, így a triggerből meghívott Részleg Átlag függvény számára sem. Ha mégis megpróbálnánk ezt a függvényt a dolgozó munkatáblára kiszámítani (azaz törléskor és módosításkor a RészlegÁtlag\_d függvényt használnánk a RészlegÁtlag\_s függvény helyett), akkor az alábbi üzenetet kapnánk például törlés esetén:

A dolgozó neve: Blake

```
DELETE FROM dolgozó
```

\*

Hiba a(z) 1. sorban:

ORA-04091: SCOTT.DOLGOZÓ tábla változtatás alatt áll, trigger/funkció számára nem látható

ORA-06512: a(z) "SCOTT.RÉSZLEGÁTLAG\_D", helyen a(z) 8. sornál

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 9. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

Annak érdekében, hogy a DELETE és UPDATE utasítások esetén a megfelelő feltételeket még-iscsak tudjuk ellenőrizni, létre kellett hoznunk egy segédtáblát (segéd). E megoldás azonban egyszerűbb leírta a hatékonyságot (hiszen adattábla-másolással járt), másrészt az egymást követő DML-műveletek (törles, beszúrás, módosítás) hatását nem is tudja követni (csak állandó új-ragenerálással...).

Az INSERT utasítás esetén (a DELETE és az UPDATE utasításokkal ellentétben) a BEFORE trigger megfelelően működött a dolgozó munkatáblára vonatkozóan is (vagyis nem volt szükség segédtábla létrehozására).

Figyeljünk fel a kijelölt és a tárolt adat triggerbeli használatára az egyes DML-műveleteknél (erről lásd jelen fejezet A korrelációs nevek használata c. pont).

## 2. megoldás (Segéd-nézettáblával, INSTEAD OF triggerrel)

### 2.1. lépés (Segédnézet létrehozása az INSTEAD OF triggerhez)

```
CREATE OR REPLACE VIEW DolgozóNézet
AS SELECT * FROM dolgozó;
```

#### Eredmény

A nézet létrejött.

### 2.2. lépés (RészlegÁtlag tárolt függvény létrehozása – munkatáblára)

```
CREATE OR REPLACE FUNCTION RészlegÁtlag(RészlegID NUMBER)
RETURN NUMBER
IS
    v_áttag NUMBER(7,2);

BEGIN
    SELECT AVG(sal)
    INTO v_áttag
    FROM dolgozó
    GROUP BY deptno
    HAVING deptno = RészlegID;
    RETURN v_áttag;
END;
/
SHOW ERRORS
```

#### Eredmény

A függvény létrejött.

Nincsenek hibák.

### 2.3. lépés (Az INSTEAD OF trigger létrehozása)

```
DROP TRIGGER Ellenőr;
```

```
CREATE OR REPLACE TRIGGER Ellenőr
INSTEAD OF DELETE OR INSERT OR UPDATE ON DolgozóNézet
FOR EACH ROW
DECLARE
    v_áttag      NUMBER;
    TörlesHiba  EXCEPTION;
```

```
BeszúrásHiba    EXCEPTION;
EmelésHiba      EXCEPTION;

BEGIN
  DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger aktív...');

  IF DELETING THEN
    IF :OLD.sal >= RészlegÁtlag(:OLD.deptno)
    THEN
      RAISE TörlésHiba;
    ELSE
      DBMS_OUTPUT.PUT_LINE('>> Lehet törölni...');
      DELETE FROM dolgozó
        WHERE UPPER(ename) = UPPER(:OLD.ename); -- VIGYÁZAT: NEM :NEW!!!
    END IF;
  ELSIF INSERTING THEN
    IF :NEW.sal >= RészlegÁtlag(:NEW.deptno)
    THEN
      RAISE BeszúrásHiba;
    ELSE
      DBMS_OUTPUT.PUT_LINE('>> Lehet beszúrni...');
      INSERT INTO dolgozó (empno, ename, sal, deptno)
        VALUES (:NEW.empno, :NEW.ename, :NEW.sal, :NEW.deptno);
    END IF;
  ELSIF UPDATING THEN
    IF :OLD.sal >= RészlegÁtlag(:OLD.deptno)
    THEN
      RAISE EmelésHiba;
    ELSE
      DBMS_OUTPUT.PUT_LINE('>> Lehet emelni...');
      UPDATE dolgozó
        SET sal = :NEW.sal
        WHERE UPPER(ename) = UPPER(:NEW.ename);
    END IF;
  END IF;
EXCEPTION
  WHEN TörlésHiba    THEN
    RAISE_APPLICATION_ERROR(-20011,'>> Nem lehet törölni...');

  WHEN BeszúrásHiba THEN
    RAISE_APPLICATION_ERROR(-20012,'>> Nem lehet beszúrni...');

  WHEN EmelésHiba    THEN
    RAISE_APPLICATION_ERROR(-20013,'>> Nem lehet emelni...');

END;
/
SHOW ERRORS
```

**Eredmény**

A trigger eldobva.  
 A trigger létrejött.  
 Nincsenek hibák.

**2.4. lépés (Ellenőrzés a törléshez)**

```
-- Tesztelő program eleje
SET serveroutput ON
SET verify OFF
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger tesztelése...');

END;
/
-- Tesztelő program magjának eleje
ACCEPT Név      PROMPT "A dolgozó neve: "
DELETE FROM dolgozóNézet
  WHERE UPPER(ename) = UPPER('&Név');
-- Tesztelő program magjának vége

SET verify ON
UNDEFINE Név
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> A trigger tesztelése befejezve...');

END;
/
-- Tesztelő program vége
```

**2.4. 1. futási eredmény (A teljes tesztelő program futtatása esetén)**

=> Az "Ellenőr" trigger tesztelése...
 A PL/SQL eljárás sikeresen befejeződött.

A dolgozó neve: Blake
 DELETE FROM dolgozóNézet

\*

Hiba a(z) 1. sorban:

ORA-20011: >> Nem lehet törölni...

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 39. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

=> Az "Ellenőr" trigger aktív...

=> A trigger tesztelése befejezve...

A PL/SQL eljárás sikeresen befejeződött.

## 2.4. 2. futási eredmény (A teljes tesztelő program futtatása esetén)

=> Az "Ellenőr" trigger tesztelése...  
 A PL/SQL eljárás sikeresen befejeződött.

A dolgozó neve: Smith

=> Az "Ellenőr" trigger aktív...

>> Lehet törölni...

1 sor törölve.

=> A trigger tesztelése befejezve...

A PL/SQL eljárás sikeresen befejeződött.

## 2.5. lépés (Ellenőrzés a beszúráshoz)

```
-- Tesztelő program eleje
SET serveroutput ON
SET verify OFF
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger tesztelése...');

END;
/

-- Tesztelő program magjának eleje
ACCEPT Név      PROMPT "      A dolgozó neve: "
ACCEPT Azonosító PROMPT "      azonosítója: "
ACCEPT Fizetés   PROMPT "      fizetése: "
ACCEPT Részleg   PROMPT "részlegazonosítója: "
INSERT INTO DolgozóNézet (empno, ename, sal, deptno)
  VALUES (&Azonosító, '&Név', &Fizetés, &Részleg);
-- Tesztelő program magjának vége

SET verify ON
UNDEFINE Név
UNDEFINE Azonosító
UNDEFINE Fizetés
UNDEFINE Részleg
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> A trigger tesztelése befejezve...');

END;
/
-- Tesztelő program vége
```

## 2.5. 1. futási eredmény (A teljes tesztelő program futtatása esetén)

=> Az "Ellenőr" trigger tesztelése...  
 A PL/SQL eljárás sikeresen befejeződött.

A dolgozó neve: Szundi  
 azonosítója: 1111  
 fizetése: 3500  
 részlegazonosítója: 30  
 INSERT INTO DolgozóNézet (empno, ename, sal, deptno)  
 \*  
 Hiba a(z) 1. sorban:  
 ORA-20012: >> Nem lehet beszúrni...  
 ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 41. sornál  
 ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben  
 => Az "Ellenőr" trigger aktív...  
 => A trigger tesztelése befejezve...  
 A PL/SQL eljárás sikeresen befejeződött.

## 2.5. 2. futási eredmény (A tesztelő program magjának futtatása esetén)

A dolgozó neve: Vidor  
 azonosítója: 2222  
 fizetése: 1500  
 részlegazonosítója: 10  
 => Az "Ellenőr" trigger aktív...  
 >> Lehet beszúrni...

1 sor létrejött.

## 2.6. lépés (Ellenőrzés a módosításhoz)

```
-- Tesztelő program eleje
SET serveroutput ON
SET verify OFF
BEGIN
  DBMS_OUTPUT.PUT_LINE('=> Az "Ellenőr" trigger tesztelése...');
END;
/

-- Tesztelő program magjának eleje
ACCEPT Név      PROMPT "A dolgozó neve: "
ACCEPT Fizetés PROMPT "Fizetésemelése: "
UPDATE DolgozóNézet
  SET sal = sal + &Fizetés
  WHERE UPPER(ename) = UPPER('&Név');
-- Tesztelő program magjának vége

SET verify ON
UNDEFINE Név
```

```

UNDEFINE Fizetés
BEGIN
    DBMS_OUTPUT.PUT_LINE('=> A trigger tesztelése befejezve...');

END;
/
-- Tesztelő program vége

```

### 2.6. 1. futási eredmény (A teljes tesztelő program futtatása esetén)

=> Az "Ellenőr" trigger tesztelése...  
 A PL/SQL eljárás sikeresen befejeződött.

A dolgozó neve: Blake

Fizetésemelése: 111

UPDATE DolgozóNézet

\*

Hiba a(z) 1. sorban:

ORA-20013: >> Nem lehet emelni...

ORA-06512: a(z) "SCOTT.ELLENŐR", helyen a(z) 43. sornál

ORA-04088: hiba a(z) 'SCOTT.ELLENŐR' trigger futása közben

=> Az "Ellenőr" trigger aktív...

=> A trigger tesztelése befejezve...

A PL/SQL eljárás sikeresen befejeződött.

### 2.6. 2. futási eredmény (A tesztelő program magjának futtatása esetén)

A dolgozó neve: Adams

Fizetésemelése: 111

=> Az "Ellenőr" trigger aktív...

>> Lehet emelni...

1 sor módosítva.

### 2.7. lépés (A munkatábla listázása)

```

SET numwidth 5
SELECT *
  FROM dolgozó
 ORDER BY ename;
SET numwidth 10

```

#### Eredmény

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	87-MÁJ-23	1211		20
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30

7782 CLARK	MANAGER	7839	81-JÚN-09	2450	10
7902 FORD	ANALYST	7566	81-DEC-03	3000	20
7900 JAMES	CLERK	7698	81-DEC-03	950	30
7566 JONES	MANAGER	7839	81-ÁPR-02	2975	20
7839 KING	PRESIDENT		81-NOV-17	5000	10
7654 MARTIN	SALESMAN	7698	81-SZE-28	1250	1400
7934 MILLER	CLERK	7782	82-JAN-23	1300	10
7788 SCOTT	ANALYST	7566	87-ÁPR-19	3000	20
7844 TURNER	SALESMAN	7698	81-SZE-08	1500	0
2222 Vidor				1500	10
7521 WARD	SALESMAN	7698	81-FEB-22	1250	500

14 sor kijelölve.

## 2.8. lépés (A létrehozott objektumok törlése)

DROP FUNCTION RészlegÁtlag;

DROP TRIGGER Ellenőr;

DROP VIEW DolgozóNézet;

### Eredmény

A függvény eldobva.

A trigger eldobva.

A nézet eldobva.



### Megjegyzés

Ez a feladat hagyományos triggerrel csak nehézkesen oldható meg (lásd 1. megoldás és az ahhoz fűzött megjegyzés).

Hatókony megoldást biztosít az Oracle 8-as verziójában bevezetett INSTEAD OF trigger használata. E trigger azonban nem definiálható közvetlenül a dolgozó munkatáblára, mivel ekkor az alábbi hibaüzenetet kapjuk:

INSTEAD OF UPDATE ON dolgozó

\*

Hiba a(z) 2. sorban:

ORA-25002: táblákhöz nem lehet INSTEAD OF triggereket létrehozni

Az INSTEAD OF triggert csak nézettáblára lehet definálni, ezért először létre kellett hozni a módosítandó dolgozó munkatáblára vonatkozó segédnézetet (DolgozóNézet). Így a dolgozó táblán már hatékonyan elvégezhető volt az összes DML-művelet (hiszen a nézettábla létrehozás nem jár adattöbbszörözéssel).

Mivel az INSTEAD OF trigger mindenkor az őt elindító DML-művelet helyett aktivizálódik, ezért (ellenértében a BEFORE és AFTER triggerekkel) a triggerben elő kell írni a megfelelő DML-műveletet. E megoldás járulékos előnye, hogy egy esetlegesen előírt törlési művelet helyett végrehajthatunk például egy módosítást.

## 11. FEJEZET

# Adatbázis-adminisztrátori ismeretek

## Feladatok és megoldások

### 11.1. feladat

Hozzon létre Scott felhasználóként az emp táblából egy vele azonos emp1 nevű táblát, valamint System felhasználóként egy szerepkört és egy ehhez rendelt felhasználót. Adjon a szerépkörnek olyan jogokat, hogy képes legyen az emp1 tábla szerkezetét módosítani, abban az adatokat lekérdezni, módosítani és törölni. Az új felhasználóként ellenőrizze, hogy ezen jogátadási tevékenységek mindegyike sikeres volt-e.

#### Megoldás

```
CONNECT scott/tiger
```

```
-- Az emp1 tábla létrehozása
-- (a DEMO-10.sql szkript program futtatása):
SET feedback OFF
@ DEMO-10
SET feedback 6

-- SELECT * FROM emp1;
```

```
CONNECT system/gazda
```

```
PROMPT A Tanár szerepkör létrehozása:
DROP ROLE tanár;
CREATE ROLE tanár;
```

```
PROMPT Adatbázishoz való kapcsolodási jog megadása a tanár szerepkörnek:
GRANT CREATE SESSION TO tanár;
```

```
PROMPT Objektumkezelései jogok adása a tanár szerepkörnek:
```

```
GRANT SELECT, UPDATE, DELETE  
  ON scott.emp1  
  TO tanár;
```

PROMPT A TóthBéla nevű felhasználó létrehozása:

```
DROP USER TóthBéla;  
CREATE USER TóthBéla  
  IDENTIFIED BY macsek;
```

PROMPT A TóthBéla nevű felhasználó hozzárendelése a tanár szerepkörhöz:  
GRANT tanár TO TóthBéla;

PROMPT TóthBéla kapcsolódása az adatbázishoz:  
CONNECT TóthBéla/macsek

```
SET numwidth 6  
SELECT *  
  FROM scott.emp1  
 WHERE UPPER(job) = 'SALESMAN';
```

```
UPDATE scott.emp1  
  SET job = 'kereskedő'  
 WHERE UPPER(job) = 'SALESMAN';
```

```
DELETE FROM scott.emp1  
 WHERE UPPER(ename) = 'TURNER';
```

```
SELECT *  
  FROM scott.emp1  
 WHERE job = 'kereskedő';  
SET numwidth 10
```

### Eredmény

Kapcsolódva.  
Kapcsolódva.

A Tanár szerepkör létrehozása:  
A szerepkör eldobva.  
A szerepkör létrejött.

Adatbázishoz való kapcsolodási jog megadása a tanár szerepkörnek:  
Az engedélyezés sikeresen befejeződött.

Objektumkezelési jogok adása a tanár szerepkörnek:  
Az engedélyezés sikeresen befejeződött.

A TóthBéla nevű felhasználó létrehozása:  
A felhasználó eldobva.  
A felhasználó létrejött.

A TóthBéla nevű felhasználó hozzárendelése a tanár szerepkörhöz:  
Az engedélyezés sikeresen befejeződött.

TóthBéla kapcsolódása az adatbázishoz:

Kapcsolódva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30

4 sor módosítva.

1 sor törölve.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7654	MARTIN	kereskedő	7698	81-SZE-28	1250	1400	30
7499	ALLEN	kereskedő	7698	81-FEB-20	1600	300	30
7521	WARD	kereskedő	7698	81-FEB-22	1250	500	30

## 11.2. feladat

A System felhasználóként törölje a 11.1. feladatban létrehozott felhasználót és szerepkört.

### Megoldás

CONNECT system/gazda

DROP USER TóthBéla;  
DROP ROLE tanár;

### Eredmény

Kapcsolódva.  
A felhasználó eldobva.  
A szerepkör eldobva.

### 11.3. feladat

Hozzon létre egy felhasználót System felhasználóként táblaterület hozzárendelése nélkül. Az új felhasználó neve a saját vezetéknévénak értelemszerű rövidítése, jelszava pedig dolgozat legyen. Változtassa meg ezt a jelszót a saját utónevére. Adjon e felhasználónak az adatbázishoz hozzáférési és nézet-létrehozási jogokat, valamint a scott.dolgozó táblájához lekérdezési jogosultágot, amit a Scott hozzon létre előbb.

#### Megoldás

PROMPT Tábla létrehozása:

```
CONNECT scott/tiger
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;
```

PROMPT Felhasználó létrehozása:

```
CONNECT system/gazda
DROP USER KissPéter CASCADE;
CREATE USER KissPéter
IDENTIFIED BY dolgozat;
```

PROMPT Rendszerjogosultság adása az új felhasználónak:

```
GRANT CREATE SESSION,
CREATE VIEW
TO KissPéter;
```

PROMPT Lekérdezési jog adása a Scott dolgozó nevű táblájára:

```
GRANT SELECT
ON scott.dolgozó
TO KissPéter;
```

PROMPT Saját jelszó módosítás:

```
CONNECT KissPéter/dolgozat
ALTER USER KissPéter
IDENTIFIED BY Péter;
```

PROMPT Lekérdezési jog ellenőrzése:

```
SET numwidth 6
SELECT *
FROM scott.dolgozó;
```

PROMPT Nézet létrehozási jog ellenőrzése:

```
CREATE OR REPLACE VIEW Dolgozónézet
```

AS

```
SELECT *
  FROM scott.dolgozó
 WHERE deptno = 20;
SELECT * FROM Dolgozónézet;
SET numwidth 10
```

### Eredmény

Tábla létrehozása:

Kapcsolódva.

A tábla eldobva.

A tábla létrejött.

Felhasználó létrehozása:

Kapcsolódva.

```
DROP USER KissPéter CASCADE
```

\*

Hiba a(z) 1. sorban:

ORA-01918: a(z) 'KISSPÉTER' felhasználó nem létezik

A felhasználó létrejött.

Rendszerjogosultság adása az új felhasználónak:

Az engedélyezés sikeresen befejeződött.

Lekérdezési jog adása a Scott dolgozó nevű táblájára:

Az engedélyezés sikeresen befejeződött.

Saját jelszó módosítás:

Kapcsolódva.

A felhasználó módosítva.

Lekérdezési jog ellenőrzése:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20

7369	SMITH	CLERK	7902	80-DEC-17	800	20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000	20
7876	ADAMS	CLERK	7788	83-JAN-12	1100	20
7934	MILLER	CLERK	7782	82-JAN-23	1300	10

14 sor kijelölve.

Nézet létrehozási jog ellenőrzése:

A nézet létrejött.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20



### Megjegyzés

A KissPéter felhasználóhoz nem rendeltünk táblaterületet, és ilyen módon a SYSTEM táblaterületre került. Figyeljünk fel arra, hogy bár a SYSTEM táblaterülethez nem kapott UNLIMITED TABLESPACE jogosultságot (és így a CREATE TABLE jogosultság birtokában sem tudott volna ott adattáblát létrehozni), mégiscsak létre tudta hozni a Dolgozónézet nézettáblát, és le is tudta azt kérdezni.

## 11.4. feladat

A System felhasználó vonja vissza a 11.3. feladatban adott jogosultságokat, és törölje a létrehozott felhasználót.

### Megoldás

```
CONNECT system/gazda
```

```
REVOKE SELECT
  ON scott.dolgozó
  FROM KissPéter;
```

```
REVOKE CREATE SESSION,
  CREATE VIEW
  FROM KissPéter;
```

```
DROP VIEW KissPéter.Dolgozónézet;
DROP USER KissPéter;
```

## Eredmény

Kapcsolódva.  
 A visszavonás sikeresen befejeződött.  
 A visszavonás sikeresen befejeződött.  
 A nézet eldobva.  
 A felhasználó eldobva.



### Megjegyzés

Ezúttal azért nem volt szükség a KissPéter felhasználó törlésénél a CASCADE opció használatára, mivel előzetesen kitöröltük a KissPéter tulajdonában lévő objektumokat (a Dolgozónézet nézetet).

## 11.7. feladat

Készítsen szkript programot, mely először a Scott felhasználóként az emp táblából létrehozza a dolgozó táblát. Ezután System felhasználóként hozza létre a Szervező szerepkört, és lássa el tábla-létrehozási (rendszer)jogosultsággal, valamint a scott.dolgozó táblán lekérdezési és módosítási (objektumkezelési) jogosultságokkal. Hozzon létre a saját nevén egy felhasználót, majd rendelje hozzá a Szervező szerepkörhöz. Ellenőrizze az új felhasználóként a kapott rendszerjogosultságot egy tábla létrehozásával. Ezt töltse fel néhány adattal, majd kérdezze le. A kapott objektumkezelési jogosultságokat a scott.dolgozó tábla módosításával és lekérdezéssel ellenőrizze.

### Megoldás

```

CONNECT scott/tiger
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

CONNECT system/gazda

DROP ROLE Szervező;
DROP USER Kovács CASCADE;

CREATE ROLE Szervező;

PROMPT Rendszerjogosultság adása a szerepkörnek:
GRANT CREATE SESSION,
      CREATE TABLE
      TO Szervező;

PROMPT objektumkezelési jogok adása a szerepkörnek:
GRANT SELECT, UPDATE
      ON scott.dolgozó
      TO Szervező;
```

PROMPT Kovács felhasználó létrehozása:

```
CREATE USER Kovács  
IDENTIFIED BY Elemér  
DEFAULT TABLESPACE users  
TEMPORARY TABLESPACE temp  
QUOTA UNLIMITED ON users;
```

PROMPT Kovács felhasználó a Szervező szerepkörhöz rendelése:

```
GRANT Szervező  
TO Kovács;
```

PROMPT Az új felhasználó kapcsolódása az adatbázishoz:

```
CONNECT Kovács/Elemér
```

```
SET numwidth 6
```

PROMPT A lekérdezési jogosultság ellenőrzése:

```
SELECT *  
FROM scott.dolgozó  
WHERE UPPER(job) = 'ANALYST';
```

PROMPT A módosítási jogosultság ellenőrzése:

```
UPDATE scott.dolgozó  
SET sal = sal + 777  
WHERE UPPER(ename)= 'FORD';  
SELECT *  
FROM scott.dolgozó  
WHERE UPPER(job) = 'ANALYST';
```

PROMPT A tábla-létrehozási jogosultság ellenőrzése:

```
CREATE TABLE próba  
(név VARCHAR2(10),  
 ár NUMBER(5));  
INSERT INTO próba  
VALUES('cipó',162);  
INSERT INTO próba  
VALUES('zsemle',14);  
INSERT INTO próba  
VALUES('perek',103);  
SELECT * FROM próba;  
SET numwidth 10
```

### Eredmény

Kapcsolódva.  
A tábla eldobva.

A tábla létrejött.

Kapcsolódva.

A szerepkör eldobva.

A felhasználó eldobva.

A szerepkör létrejött.

Rendszerjogosultság adása a szerepkörnek:

Az engedélyezés sikeresen befejeződött.

objektumkezelési jogok adása a szerepkörnek:

Az engedélyezés sikeresen befejeződött.

Kovács felhasználó létrehozása:

A felhasználó létrejött.

Kovács felhasználó a Szervező szerepkörhöz rendelése:

Az engedélyezés sikeresen befejeződött.

Az új felhasználó kapcsolódása az adatbázishoz:

Kapcsolódva.

A lekérdezési jogosultság ellenőrzése:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20

A módosítási jogosultság ellenőrzése:

1 sor módosítva.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7902	FORD	ANALYST	7566	81-DEC-03	3777		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20

A tábla-létrehozási jogosultság ellenőrzése:

A tábla létrejött.

1 sor létrejött.

1 sor létrejött.

1 sor létrejött.

NÉV ÁR

cipő 162

zsemle	14
perek	103

## 11.8. feladat

A System felhasználóként törölje a 11.7. feladatban létrehozott felhasználót és szerepkört.

### Megoldás

```
CONNECT system/gazda
```

```
DROP USER Kovács CASCADE;  
DROP ROLE Szervező;
```

### Eredmény

Kapcsolódva.

A felhasználó eldobva.

A szerepkör eldobva.



### Megjegyzés

Ezúttal a CASCADE opciót használtuk a felhasználó törléséhez, melynek révén törlődött a felhasználó által létrehozott Próba adattábla is.

## 12. FEJEZET

# Kapcsolat a külvilággal

E fejezetre vonatkozóan nincsenek feladatok, csak mintapéldák, melyek pedig az I. részben, a Feladatgyűjteményben szerepeltek.

## 13. FEJEZET

# Részletező csoportosítások és analitikus függvények az SQL-ben

E fejezetre vonatkozóan nincsenek feladatok, csak mintapéldák, melyek pedig az I. részben, a Feladatgyűjteményben szerepeltek.

## 14. FEJEZET

# A DEMO vállalat (üzleti alkalmazások)

## Feladatok és megoldások

### 14.1. feladat

Listázza ki „gazdaságosan” (a szükséges mezőszélességekkel) az emp és a dept táblákat.

#### Megoldás (A DEMO vállalat részlegei és dolgozói)

```
COLUMN empno          FORMAT 9999
COLUMN ename          FORMAT A7
COLUMN job            FORMAT A11
COLUMN mgr            FORMAT 9999
COLUMN sal             FORMAT 9999
COLUMN comm            FORMAT 9999
COLUMN deptno          FORMAT 99
COLUMN dname          FORMAT A11
COLUMN loc             FORMAT A8
SELECT * FROM dept;
SELECT * FROM emp;
CLEAR COLUMNS
```

#### Eredmény

DEPTNO	DNAME	LOC
--------	-------	-----

10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
-------	-------	-----	-----	----------	-----	------	--------

7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30

7782	CLARK	MANAGER	7839	81-JÚN-09	2450	10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975	20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0
7900	JAMES	CLERK	7698	81-DEC-03	950	30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500
7902	FORD	ANALYST	7566	81-DEC-03	3000	20
7369	SMITH	CLERK	7902	80-DEC-17	800	20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000	20
7876	ADAMS	CLERK	7788	83-JAN-12	1100	20
7934	MILLER	CLERK	7782	82-JAN-23	1300	10

14 sor kijelölve.

## 14.2. feladat

Ábrázolja grafikusan a DEMO vállalat dolgozóit a vállalati hierarchiában elfoglalt helyük szerint. Az egyes dolgozók neve mellett tüntesse fel a besorolását is (elnök, főosztályvezető, osztályvezető, főosztályi közvetlen beosztott, beosztott).

### Megoldás (A DEMO vállalat struktúrája)

#### 1. lépés (Hierarchiaszintek)

```
SELECT LEVEL AS "HierarchiaSzint",
       empno AS "FőnökAzon",
       ename AS "FőnökNév"
  FROM emp
 CONNECT BY mgr = PRIOR empno
 START WITH UPPER(job) = 'PRESIDENT';
```

#### Eredmény

HierarchiaSzint	FőnökAzon	FőnökNév
1	7839	KING
2	7698	BLAKE
3	7654	MARTIN
3	7499	ALLEN
3	7844	TURNER
3	7900	JAMES
3	7521	WARD
2	7782	CLARK
3	7934	MILLER
2	7566	JONES
3	7902	FORD

```

4      7369 SMITH
3      7788 SCOTT
4      7876 ADAMS

```

14 sor kijelölve.

## 2. lépés (Grafikus ábrázolás)

```

SELECT LPAD(' ',LEVEL*2) ||
       RPAD(LEVEL,10)    ||
       RPAD(empno,10)    ||
       RPAD(ename,10)     AS "Szint    azonosító    név"
FROM emp
CONNECT BY mgr = PRIOR empno
START WITH UPPER(job) = 'PRESIDENT';

```

## Eredmény

Szint	azonosító	név
1	7839	KING
2	7698	BLAKE
3	7654	MARTIN
3	7499	ALLEN
3	7844	TURNER
3	7900	JAMES
3	7521	WARD
2	7782	CLARK
3	7934	MILLER
2	7566	JONES
3	7902	FORD
4	7369	SMITH
3	7788	SCOTT
4	7876	ADAMS

14 sor kijelölve.

## 3. lépés (A besorolás jelzéssel kiegészítve)

### Megoldás

```

SET linesize 80
SELECT LPAD(' ',LEVEL*2) ||
       RPAD(LEVEL,8)  ||
       RPAD(ename,10)  ||
CASE
  WHEN LEVEL=1 THEN 'Elnök'

```

```

WHEN LEVEL=2 THEN 'Főosztályvezető'
WHEN LEVEL=3 THEN
  CASE
    WHEN 0 =
      (SELECT COUNT(*)
       FROM emp
       WHERE emp.empno = e.empno AND
             emp.empno IN
           (SELECT mgr
            FROM emp))
    THEN 'Főosztályi közvetlen beosztott'
    ELSE 'Osztályvezető'
  END
ELSE 'Beosztott'
END
AS "Szint-----Név-----Besorolás"
FROM emp e
CONNECT BY mgr = PRIOR empno
START WITH UPPER(ename) = 'KING';
SET linesize 200

```

**Eredmény**

Szint-----Név-----Besorolás

---

1	KING	Elnök
2	BLAKE	Főosztályvezető
3	MARTIN	Főosztályi közvetlen beosztott
3	ALLEN	Főosztályi közvetlen beosztott
3	TURNER	Főosztályi közvetlen beosztott
3	JAMES	Főosztályi közvetlen beosztott
3	WARD	Főosztályi közvetlen beosztott
2	CLARK	Főosztályvezető
3	MILLER	Főosztályi közvetlen beosztott
2	JONES	Főosztályvezető
3	FORD	Osztályvezető
4	SMITH	Beosztott
3	SCOTT	Osztályvezető
4	ADAMS	Beosztott

14 sor kijelölve.

### 14.3. feladat

Állítsa elő a főosztályvezetők, az osztályvezetők, a beosztottak, a főosztályi közvetlen beosztottak, az osztályok és a főosztályok nézettábláit. Állítson elő egy olyan kibővített osztály nézettáblát is, amely az osztályok dolgozóin kívül tartalmazza a főosztályi közvetlen beosztottakat a főosztályvezetőjükkel (mint pszeudo-osztályvezetőjükkel) együtt.

#### Megoldás

##### A főosztályvezető-nézet

```
PROMPT A főosztályvezető-nézet:
CREATE OR REPLACE VIEW Főosztályvezető
AS
SELECT d.empno,
       d.ename
  FROM emp d,
       emp f
 WHERE d.mgr = f.empno          AND
       UPPER(f.job) = 'PRESIDENT' AND
       d.empno IN (SELECT mgr
                    FROM emp);

SELECT *
  FROM Főosztályvezető
 ORDER BY ename;
```

#### Eredmény

A főosztályvezető-nézet:

A nézet létrejött.

EMPNO	ENAME
7698	BLAKE
7782	CLARK
7566	JONES

##### Az osztályvezető-nézet

```
PROMPT Az osztályvezető-nézet:
CREATE OR REPLACE VIEW Osztályvezető
AS
SELECT empno,
       ename
  FROM emp
 WHERE mgr IN (SELECT empno
                  FROM Főosztályvezető) AND
```

```
empno IN (SELECT mgr
           FROM emp);
```

vagy másként létrehozva:

PROMPT Az osztályvezető-nézet:  
CREATE OR REPLACE VIEW Osztályvezető  
AS

```
SELECT emp.empno,
       emp.ename
  FROM emp, Főosztályvezető
 WHERE emp.mgr IN Főosztályvezető.empno AND
       emp.empno IN (SELECT mgr
                      FROM emp);
```

```
SELECT *
  FROM Osztályvezető
 ORDER BY ename;
```

### Eredmény (mindkét esetben)

Az osztályvezető-nézet:  
A nézet létrejött.

EMPNO	ENAME
7902	FORD
7788	SCOTT

### A beosztottak (főosztályon vagy osztályon) nézete

PROMPT A beosztottak (főosztályon vagy osztályon) nézete:

CREATE OR REPLACE VIEW Beosztott  
AS

```
SELECT empno,
       ename
  FROM emp
 WHERE empno NOT IN (SELECT mgr
                      FROM emp
                     WHERE mgr IS NOT NULL);
```

```
SELECT *
  FROM Beosztott
 ORDER BY ename;
```

**Eredmény**

A beosztottak (főosztályon vagy osztályon) nézete:  
 A nézet létrejött.

EMPNO ENAME

---

7876 ADAMS  
 7499 ALLEN  
 7900 JAMES  
 7654 MARTIN  
 7934 MILLER  
 7369 SMITH  
 7844 TURNER  
 7521 WARD

8 sor kijelölve.

**A főosztályi közvetlen beosztottak nézete**

PROMPT A főosztályi közvetlen beosztottak nézete:

CREATE OR REPLACE VIEW FBeosztott

AS

```
SELECT Főosztályvezető.empno AS FOVempno,
       Főosztályvezető.ename AS FOVename,
       emp.empno,
       emp.ename
  FROM emp, Beosztott, Főosztályvezető
 WHERE emp.mgr IN Főosztályvezető.empno AND
       emp.empno IN Beosztott.empno;
```

```
SELECT *
  FROM FBeosztott
 ORDER BY ename;
```

**Eredmény**

A főosztályi közvetlen beosztottak nézete:  
 A nézet létrejött.

FOVEMPNO FOVENAME	EMPNO ENAME
7698 BLAKE	7499 ALLEN
7698 BLAKE	7900 JAMES
7698 BLAKE	7654 MARTIN
7782 CLARK	7934 MILLER

7698 BLAKE	7844 TURNER
7698 BLAKE	7521 WARD

6 sor kijelölve.

### 1. ellenőrzés (A Blake főosztályvezető közvetlen beosztottjai)

```
SELECT *
  FROM FBeosztott
 WHERE UPPER(FOVename) = UPPER('Blake')
 ORDER BY ename;
```

#### Eredmény

FOVEMPNO FOVENAME	EMPNO ENAME
7698 BLAKE	7499 ALLEN
7698 BLAKE	7900 JAMES
7698 BLAKE	7654 MARTIN
7698 BLAKE	7844 TURNER
7698 BLAKE	7521 WARD

### 2. ellenőrzés (Jones főosztályvezető közvetlen beosztottjai)

```
SELECT *
  FROM FBeosztott
 WHERE UPPER(FOVename) = UPPER('Jones')
 ORDER BY ename;
```

#### Eredmény

nincsenek kijelölve sorok

#### Az osztály-nézet

```
PROMPT Az osztály-nézet:
CREATE OR REPLACE VIEW Osztály
AS
  SELECT Osztályvezető.empno AS OVempno,
         Osztályvezető.ename AS OVename,
         emp.empno,
         emp.ename
    FROM emp,
         Osztályvezető
   WHERE emp.mgr IN Osztályvezető.empno OR
         emp.empno IN Osztályvezető.empno;
```

```
SELECT *
  FROM Osztály
 ORDER BY OVename, ename;
```

**Eredmény**

Az osztály-nézet:

A nézet létrejött.

OVEMPNO	OVENAME	EMPNO	ENAME
7902	FORD	7902	FORD
7902	FORD	7369	SMITH
7788	SCOTT	7876	ADAMS
7788	SCOTT	7788	SCOTT

**Ellenőrzés (Scott osztályán dolgozók)**

```
SELECT *
  FROM Osztály
 WHERE UPPER(OVename) = UPPER('Scott')
 ORDER BY ename;
```

**Eredmény**

OVEMPNO	OVENAME	EMPNO	ENAME
7788	SCOTT	7876	ADAMS
7788	SCOTT	7788	SCOTT

**A főosztály-nézet**

PROMPT A főosztály-nézet:

```
CREATE OR REPLACE VIEW Főosztály
AS
  SELECT Főosztályvezető.empno AS FOVempno,
         Főosztályvezető.ename AS FOVename,
         emp.empno,
         emp.ename
    FROM emp,
         Főosztályvezető
   WHERE emp.mgr IN Főosztályvezető.empno
     OR
     emp.empno IN Főosztályvezető.empno
     OR
     emp.mgr IN (SELECT Osztályvezető.empno
                  FROM emp,
                       Osztályvezető
                 WHERE emp.empno = Osztályvezető.empno AND
                       emp.mgr = Főosztályvezető.empno)
     OR
     emp.empno IN (SELECT Osztályvezető.empno
                  FROM emp,
                       Osztályvezető
                 WHERE emp.empno = Osztályvezető.empno AND
```

```

        emp.mgr = Főosztályvezető.empno)
ORDER BY FOVename;

SELECT *
FROM Főosztály
ORDER BY FOVename, ename;

```

**Eredmény**

A főosztály-nézet:

A nézet létrejött.

FOVEMPNO	FOVENAME	EMPNO	ENAME
7698	BLAKE	7499	ALLEN
7698	BLAKE	7698	BLAKE
7698	BLAKE	7900	JAMES
7698	BLAKE	7654	MARTIN
7698	BLAKE	7844	TURNER
7698	BLAKE	7521	WARD
7782	CLARK	7782	CLARK
7782	CLARK	7934	MILLER
7566	JONES	7876	ADAMS
7566	JONES	7902	FORD
7566	JONES	7566	JONES
7566	JONES	7788	SCOTT
7566	JONES	7369	SMITH

13 sor kijelölve.

**Ellenőrzés (Blake főosztályán dolgozók)**

```

SELECT *
FROM Főosztály
WHERE UPPER(FOVename) = UPPER('Blake')
ORDER BY ename;

```

**Eredmény**

FOVEMPNO	FOVENAME	EMPNO	ENAME
7698	BLAKE	7499	ALLEN
7698	BLAKE	7698	BLAKE
7698	BLAKE	7900	JAMES
7698	BLAKE	7654	MARTIN
7698	BLAKE	7844	TURNER
7698	BLAKE	7521	WARD

6 sor kijelölve.

### A kibővített osztály-nézet

PROMPT A kibővített osztály-nézet:

```
CREATE OR REPLACE VIEW Osztály1
AS
SELECT Osztályvezető.empno AS OVempno,
       Osztályvezető.ename AS OVename,
       emp.empno,
       emp.ename
  FROM emp,
       Osztályvezető
 WHERE emp.mgr IN Osztályvezető.empno OR
       emp.empno IN Osztályvezető.empno
UNION
SELECT Főosztályvezető.empno AS OVempno,
       Főosztályvezető.ename AS OVename,
       emp.empno,
       emp.ename
  FROM emp,
       Főosztályvezető
 WHERE emp.empno IN Főosztályvezető.empno OR
       (emp.mgr IN Főosztályvezető.empno AND
        emp.empno NOT IN (SELECT mgr
                           FROM emp
                           WHERE mgr IS NOT NULL));
SELECT *
  FROM Osztály1
 ORDER BY OVename, ename;
```

### Eredmény

A kibővített osztály-nézet:  
A nézet létrejött.

OVEMPNO	OVENAME	EMPNO	ENAME
7698	BLAKE	7499	ALLEN
7698	BLAKE	7698	BLAKE
7698	BLAKE	7900	JAMES
7698	BLAKE	7654	MARTIN
7698	BLAKE	7844	TURNER
7698	BLAKE	7521	WARD
7782	CLARK	7782	CLARK

7782 CLARK	7934 MILLER
7902 FORD	7902 FORD
7902 FORD	7369 SMITH
7566 JONES	7566 JONES
7788 SCOTT	7876 ADAMS
7788 SCOTT	7788 SCOTT

13 sor kijelölve.

### Ellenőrzés (Blake osztályán dolgozók)

```
SELECT *
  FROM Osztaly1
 WHERE UPPER(OVename) = UPPER('Blake')
 ORDER BY ename;
```

### Eredmény

OVEMPNO OVENAME	EMPNO ENAME
7698 BLAKE	7499 ALLEN
7698 BLAKE	7698 BLAKE
7698 BLAKE	7900 JAMES
7698 BLAKE	7654 MARTIN
7698 BLAKE	7844 TURNER
7698 BLAKE	7521 WARD



### Megjegyzés

A kibővített osztály-nézet alkalmazására lásd 14.8. feladat.

## 14.4. feladat

Adja meg főosztályvezetőnként a főosztályok dolgozóinak számát és összfizetését, ez utóbbi szerint rendezve.

### Megoldás

PROMPT Főosztályi dolgozók száma és összfizetése:

```
SELECT FOVename AS "FOVneve",
       COUNT(*) AS "Folétszám",
       SUM(sal) AS "Főösszfizetés"
  FROM emp, főosztály
 WHERE emp.empno = főosztály.empno
 GROUP BY FOVename
 ORDER BY "Főösszfizetés";
```

**Eredmény**

Főosztályi dolgozók száma és összfizetése:

FOVneve	FOLÉTSZÁM	FÖÖSSZFIZETÉS
CLARK	2	3750
BLAKE	6	9400
JONES	5	10875

**14.5. feladat**

Listázza az osztály-dolgozók fizetéseit főosztályonkénti részösszegképzéssel.

**Megoldás**

PROMPT Osztály-dolgozók fizetései osztályonkénti és

PROMPT főosztályonkénti részösszegképzéssel:

```
SELECT FOVename      AS "FOVneve",
       OVename        AS "OVneve",
       emp.ename      AS "Dolgozónév",
       SUM(sal)        AS "Fizetés"
  FROM emp, Osztály, Főosztály
 WHERE (emp.empno IN Főosztály.empno AND
       emp.empno IN Osztály.empno)
 GROUP BY ROLLUP (FOVename, OVename, emp.ename);
```

**Eredmény**

Osztály-dolgozók fizetései osztályonkénti és

főosztályonkénti részösszegképzéssel:

FOVneve	OVneve	Dolgozónév	Fizetés
JONES	FORD	FORD	3000
JONES	FORD	SMITH	800
JONES	FORD		3800
JONES	SCOTT	ADAMS	1100
JONES	SCOTT	SCOTT	3000
JONES	SCOTT		4100
JONES			7900
			7900

8 sor kijelölve.

## 14.6. feladat

Listázza ki a főosztályi közvetlen beosztottak fizetéseit főosztályonkénti részösszegképzéssel.

### Megoldás

PROMPT Főosztályi közvetlen beosztottak fizetései-  
részösszegképzéssel.

PROMPT (NEM-osztály-dolgozók fizetései ...)

```
SELECT FOVename      AS "FOVneve",
       emp.ename     AS "Dolgozónév",
       SUM(sal)       AS "Fizetés"
  FROM emp, Főosztály
 WHERE emp.empno IN Főosztály.empno AND
       emp.empno IN ((SELECT empno
                      FROM Főosztály)
                  MINUS
                  (SELECT empno
                     FROM Osztály))
 GROUP BY ROLLUP (FOVename, emp.ename);
```

### Eredmény

Főosztályi közvetlen beosztottak fizetései részösszegképzéssel.  
(NEM-osztály-dolgozók fizetései ...)

FOVneve	Dolgozónév	Fizetés
BLAKE	WARD	1250
BLAKE	ALLEN	1600
BLAKE	BLAKE	2850
BLAKE	JAMES	950
BLAKE	MARTIN	1250
BLAKE	TURNER	1500
BLAKE		9400
CLARK	CLARK	2450
CLARK	MILLER	1300
CLARK		3750
JONES	JONES	2975
JONES		2975
		16125

13 sor kijelölve.

## 14.7. feladat

Listázza ki a dolgozók fizetéseit főosztályonkénti részösszegképzéssel.

### Megoldás

PROMPT A dolgozói fizetések főosztályonkénti részösszegképzéssel:

```
SELECT FOVename      AS "FOVneve",
       emp.ename     AS "Dolgozónév",
       SUM(sal)       AS "Fizetés"
  FROM emp, Főosztály
 WHERE emp.empno IN Főosztály.empno
 GROUP BY ROLLUP (FOVename, emp.ename);
```

### Eredmény

A dolgozói fizetések főosztályonkénti részösszegképzéssel:

FOVneve	Dolgozónév	Fizetés
BLAKE	WARD	1250
BLAKE	ALLEN	1600
BLAKE	BLAKE	2850
BLAKE	JAMES	950
BLAKE	MARTIN	1250
BLAKE	TURNER	1500
BLAKE		9400
CLARK	CLARK	2450
CLARK	MILLER	1300
CLARK		3750
JONES	FORD	3000
JONES	ADAMS	1100
JONES	JONES	2975
JONES	SCOTT	3000
JONES	SMITH	800
JONES		10875
		24025

17 sor kijelölve.

## 14.8. feladat

Listázza ki főosztályvezetőnként rendezve a főosztályok dolgozóinak fizetését osztályonként csoportosítva, osztályonkénti és főosztályonkénti részösszeg-meghatározással kiegészítve.

## 1. Hibás megoldás (Egyeszerű csoportosítással)

PROMPT Dolgozók fizetései részösszegképzéssel:

```

SELECT F.FOVename          AS "FOVneve",
       O.OVename           AS "OVneve",
       E.ename             AS "Dolgozónév",
       sal                AS "DolgozóFiz",
       OS.Osumsal          AS "OsztFiz",
       FS.FOsumsal         AS "FőOsztFiz"
  FROM emp                  E,
       Főosztály          F,
       Osztály            O,
       (SELECT OVempno,
              SUM(sal)  AS Osumsal
             FROM emp, Osztály
            WHERE emp.empno IN Osztály.empno
            GROUP BY OVempno) OS,
       (SELECT FOVempno,
              SUM(sal)  AS FOsumsal
             FROM emp, Főosztály
            WHERE emp.empno IN Főosztály.empno
            GROUP BY FOVempno) FS
 WHERE E.empno IN F.empno    AND
       E.empno IN O.empno    AND
       O.OVempno IN OS.OVempno AND
       F.FOVempno IN FS.FOVempno
 ORDER BY "FOVneve", "OVneve", "Dolgozónév";

```

### Eredmény

Dolgozók fizetései részösszegképzéssel:

FOVneve	OVneve	Dolgozónév	DolgozóFiz	OsztFiz	FőOsztFiz
JONES	FORD	FORD	3000	3800	10875
JONES	FORD	SMITH	800	3800	10875
JONES	SCOTT	ADAMS	1100	4100	10875
JONES	SCOTT	SCOTT	3000	4100	10875

## 2. Hibás megoldás (Részcsoporthálózattal)

PROMPT Dolgozók fizetései részösszegképzéssel:

```

SELECT F.FOVename          AS "FOVneve",
       O.OVename           AS "OVneve",
       E.ename             AS "Dolgozónév",
       SUM(sal)            AS "Fizetés"
  FROM emp                  E,
       Főosztály          F,
       Osztály            O;

```

```

Osztály          0,
Főosztály        F
WHERE E.empno IN F.empno AND
      E.empno IN O.empno
GROUP BY ROLLUP (F.FOVename, O.OVename, E.ename);

```

**Eredmény**

Dolgozók fizetései részösszegképzéssel:

FOVneve	OVneve	Dolgozónév	Fizetés
JONES	FORD	FORD	3000
JONES	FORD	SMITH	800
JONES	FORD		3800
JONES	SCOTT	ADAMS	1100
JONES	SCOTT	SCOTT	3000
JONES	SCOTT		4100
JONES			7900
			7900

8 sor kijelölve.

**1. Jó megoldás (Egyszerű csoportosítással)**

PROMPT Dolgozók fizetései részösszegképzéssel:

```

SELECT F.FOVename           AS "FOVneve",
       O.OVename            AS "OVneve",
       E.ename              AS "Dolgozónév",
       sal                  AS "DolgozóFiz",
       OS.Osumsal           AS "OsztFiz",
       FS.FOsumsal          AS "FőOsztFiz"
FROM emp                   E,
     Főosztály          F,
     Osztály1            O,
     (SELECT OVempno,
            SUM(sal)  AS Osumsal
         FROM emp, Osztály1
        WHERE emp.empno IN Osztály1.empno
        GROUP BY OVempno)  OS,
     (SELECT FOVempno,
            SUM(sal)  AS FOsumsal
         FROM emp, Főosztály
        WHERE emp.empno IN Főosztály.empno
        GROUP BY FOVempno) FS
WHERE E.empno IN F.empno    AND
      E.empno IN O.empno

```

```

E.empno IN 0.empno      AND
0.OVempno IN OS.OVempno AND
F.FOVempno IN FS.FOVempno
ORDER BY "FOVneve", "OVneve", "Dolgozónév";

```

**Eredmény**

Dolgozók fizetései részösszegképzéssel:

FOVneve	OVneve	Dolgozónév	DolgozóFiz	OsztFiz	FőosztFiz
BLAKE	BLAKE	ALLEN	1600	9400	9400
BLAKE	BLAKE	BLAKE	2850	9400	9400
BLAKE	BLAKE	JAMES	950	9400	9400
BLAKE	BLAKE	MARTIN	1250	9400	9400
BLAKE	BLAKE	TURNER	1500	9400	9400
BLAKE	BLAKE	WARD	1250	9400	9400
CLARK	CLARK	CLARK	2450	3750	3750
CLARK	CLARK	MILLER	1300	3750	3750
JONES	FORD	FORD	3000	3800	10875
JONES	FORD	SMITH	800	3800	10875
JONES	JONES	JONES	2975	2975	10875
JONES	SCOTT	ADAMS	1100	4100	10875
JONES	SCOTT	SCOTT	3000	4100	10875

13 sor kijelölve.

**2. Jó megoldás (Részcsoporthálózattal)**

PROMPT Dolgozók fizetései részösszegképzéssel:

```

SELECT F.FOVename          AS "FOVneve",
       0.OVename           AS "OVneve",
       E.ename             AS "Dolgozónév",
       SUM(sal)            AS "Fizetés"
  FROM emp                 E,
       Osztály1            O,
       Főosztály           F
 WHERE E.empno IN F.empno  AND
       E.empno IN 0.empno
 GROUP BY ROLLUP (F.FOVename, 0.OVename, E.ename);

```

**Eredmény**

Dolgozók fizetései részösszegképzéssel:

FÖVneve	ÖVneve	DolgozóNév	Fizetés
BLAKE	BLAKE	WARD	1250
BLAKE	BLAKE	ALLEN	1600
BLAKE	BLAKE	BLAKE	2850
BLAKE	BLAKE	JAMES	950
BLAKE	BLAKE	MARTIN	1250
BLAKE	BLAKE	TURNER	1500
BLAKE	BLAKE		9400
BLAKE			9400
CLARK	CLARK	CLARK	2450
CLARK	CLARK	MILLER	1300
CLARK	CLARK		3750
CLARK			3750
JONES	FORD	FORD	3000
JONES	FORD	SMITH	800
JONES	FORD		3800
JONES	JONES	JONES	2975
JONES	JONES		2975
JONES	SCOTT	ADAMS	1100
JONES	SCOTT	SCOTT	3000
JONES	SCOTT		4100
JONES			10875
			24025

22 sor kijelölve.

### 3. Jó megoldás (Részcsoporthézással és értelmező kiíratással)

PROMPT Dolgozók fizetései részösszegképzéssel:

```
SELECT NVL(F.FOVename,'ÖsszFőosztályÖsszes')
      AS "FÖVneve",
CASE
    WHEN F.FOVename IS NULL
        THEN NULL
    ELSE
        NVL(O.ÖVename,'FőosztályÖsszes')
    END
      AS "ÖVneve",
CASE
    WHEN O.ÖVename IS NULL
        THEN NULL
    ELSE
        NVL(E.ename,'OsztályÖsszes')
    END
      AS "DolgozóNév",
SUM(sal)          AS "Fizetés"
```

```

FROM emp          E,
Osztály1        O,
Főosztály       F
WHERE E.empno IN F.empno AND
      E.empno IN O.empno
GROUP BY ROLLUP (F.FOVename, O.OVename, E.ename);

```

**Eredmény**

Dolgozók fizetései részösszegképzéssel:

FOVneve	OVneve	Dolgozónév	Fizetés
BLAKE	BLAKE	WARD	1250
BLAKE	BLAKE	ALLEN	1600
BLAKE	BLAKE	BLAKE	2850
BLAKE	BLAKE	JAMES	950
BLAKE	BLAKE	MARTIN	1250
BLAKE	BLAKE	TURNER	1500
BLAKE	BLAKE	OsztályÖsszes	9400
BLAKE	FőosztályÖsszes		9400
CLARK	CLARK	CLARK	2450
CLARK	CLARK	MILLER	1300
CLARK	CLARK	OsztályÖsszes	3750
CLARK	FőosztályÖsszes		3750
JONES	FORD	FORD	3000
JONES	FORD	SMITH	800
JONES	FORD	OsztályÖsszes	3800
JONES	JONES	JONES	2975
JONES	JONES	OsztályÖsszes	2975
JONES	SCOTT	ADAMS	1100
JONES	SCOTT	SCOTT	3000
JONES	SCOTT	OsztályÖsszes	4100
JONES	FőosztályÖsszes		10875
ÖsszFőosztályÖsszes			24025

22 sor kijelölve.

**4. Jó megoldás (Részcsoporthálózat kiválasztása, értelmező kiíratással, csak a valódi besorolások kiíratásával, ismétlődésmentes kiíratással)**

BREAK ON "FOVneve" ON "OVneve"

PROMPT Dolgozók fizetései részösszegképzéssel:

SELECT NVL(F.FOVename, 'ÖsszFőosztályÖsszes')

AS "FOVneve",

```

CASE
    WHEN F.FOVename IS NULL
        THEN NULL
    ELSE
        CASE
            WHEN O.OVename = F.FOVename
                THEN NULL
            ELSE
                NVL(O.OVename, 'Főösszes')
        END
    END AS "OVneve",
CASE
    WHEN O.OVename IS NULL
        THEN NULL
    ELSE
        CASE
            WHEN O.OVename = F.FOVename
                THEN
                    CASE
                        WHEN
                            ((SELECT COUNT(*)
                                FROM FBeosztott
                                WHERE FOVename = F.FOVename) = 0)
                            THEN
                                NVL(E.ename, 'Fő Közvetlen(csak FOV)')
                            ELSE
                                NVL(E.ename, 'Fő KözvetlenÖsszes')
                        END
                    ELSE
                        NVL(E.ename, 'OsztályÖsszes')
                    END
                END AS "DolgozóNév",
            SUM(sal) AS "Fizetés"
        END
    END
FROM emp E,
     Osztályl O,
     Főosztály F
WHERE E.empno IN F.empno AND
      E.empno IN O.empno
GROUP BY ROLLUP (F.FOVename, O.OVename, E.ename);
CLEAR BREAKS

```

**Eredmény**

Dolgozók fizetései részösszegképzéssel:

FOVneve	OVneve	Dolgozónév	Fizetés
BLAKE		WARD	1250
		ALLEN	1600
		BLAKE	2850
		JAMES	950
		MARTIN	1250
		TURNER	1500
		FőKözvetlenÖsszes	9400
		Főösszes	9400
CLARK		CLARK	2450
		MILLER	1300
		FőKözvetlenÖsszes	3750
		Főösszes	3750
JONES	FORD	FORD	3000
		SMITH	800
		OsztályÖsszes	3800
		JONES	2975
		Fő.Közvetlen(csak FOV)	2975
	SCOTT	ADAMS	1100
		SCOTT	3000
		OsztályÖsszes	4100
		Főösszes	10875
ÖsszFőösszes			24025

22 sor kijelölve.



Megjegyzés (Csoportképzés nézettáblán keresztül, teljes és nem teljes hierarchiák)

A fenti 2., 3. és 4. jó megoldás teljes összesítése nem tartalmazza az elnök (King) fizetését, mivel az összesítést csak főosztályi szinten végeztük!

Vegyük észre, hogy a jó megoldásokhoz szükség volt az Osztály1 kibővített osztálynézetre, mivel az Osztály nézet használata esetén kimaradtak a feldolgozásból minden dolgozók, akik nem voltak tagjai valamelyik osztálynak, hanem közvetlen főosztályvezetői beosztottak (ilyen lehet például egy főosztályvezetői titkárnő).

Ha a hibás megoldások általánosabb okát vizsgáljuk, akkor azt láthatjuk, hogy a hiba amiatt következett be, mert olyan nézettábla valamely oszlop szerint végeztük a csoportosítást, mely nem volt az eredeti tábla minden sorára értelmezve.

A fentiek alapján az egyik tanulság tehát az, hogy legyünk fokozottan óvatosak a nézettáblán keresztül végzett csoportosítások esetén.

A jelen feladatra vonatkozóan pedig megállapíthatjuk, hogy a csoportosításhoz olyan hierarchikus logikai adatszerkezetre van szükség, mely a csoportosítás szintjén teljes, más néven ki-egyenlített. Ez alatt azt értjük, hogy a csoportosítási szempontok (oszlopok) az általunk feldol-

gozni kívánt elemek minden egyikére értelmezett legyen. Vigyázat, ez nem jelenti azt, hogy minden egyik elemre! Gondolunk arra, hogy az elnök (King) a jó megoldásba sem került bele, ám ez nem baj, hiszen a listázást csupán főosztályi szintig akartuk elvégezni. A hibás megoldások esetén azonban az olyan dolgozók se kerültek a megoldásba, akik tagjai valamelyik főosztálynak.

A gyakorlatban sokszor fordul elő az, hogy a feldolgozandó objektumok (adattáblák) logikai adatszerkezete valamely vizsgálandó szempont szerint nem teljes. Az életben ez nem okoz gondot. Nem okoz gondot az, hogy a megyei jogú városok nem tartoznak egy megyéhez se, vagy az, hogy a földszint és az alagsor nem emelet stb. Az automatikus adatfeldolgozás során azonban gondoskodni kell arról, hogy a hierarchia hiányzó szinjeit valamivel pótoljuk. Az előző példák esetén valószínűleg segíti a feldolgozást, ha a megyei jogú városok megyenevénél és megyeszékhelyének önmagát vesszük fel, vagy a földszintet és az alagsort 0-dik, illetve (-1)-edik emeletnek nevezzük. Végül tekintsük a DEMO vállalatot. Itt a közvetlen főosztályi beosztottakat virtuális osztályként (melynek „osztályvezetője” a főosztályvezető) egyszerűen hozzácsaptuk az osztályok halmazához, így kaptuk a kibővített Osztály1 nézetet. Az eredeti Elnök-Főosztály-Osztály nem teljes hierarchia helyett kapott Elnök-Főosztály-Osztály1 hierarchia már teljes, és a kívánt feladat rajta megoldható volt.

## 14.9. feladat

Listázza ki főosztályvezetőnként rendezve a főosztályok dolgozóinak fizetését munkakörönként csoportosítva, munkakörönként (minden főosztályon belül) képezve a részösszegeket.

### Megoldás (Részcsoporthépzéssel)

```

SELECT F.FOVename           AS "FOVneve",
       job                  AS "Munkakör",
       E.ename              AS "Dolgozónév",
       SUM(sal)              AS "Fizetés"
  FROM emp                 E,
       Főosztály            F
 WHERE E.empno IN F.empno
 GROUP BY ROLLUP (F.FOVename, job, E.ename);

```

### Eredmény

FOVneve	Munkakör	Dolgozónév	Fizetés
BLAKE	CLERK	JAMES	950
BLAKE	CLERK		950
BLAKE	MANAGER	BLAKE	2850
BLAKE	MANAGER		2850
BLAKE	SALESMAN	WARD	1250
BLAKE	SALESMAN	ALLEN	1600
BLAKE	SALESMAN	MARTIN	1250
BLAKE	SALESMAN	TURNER	1500
BLAKE	SALESMAN		5600
BLAKE			9400

CLARK	CLERK	MILLER	1300
CLARK	CLERK		1300
CLARK	MANAGER	CLARK	2450
CLARK	MANAGER		2450
CLARK			3750
JONES	CLERK	ADAMS	1100
JONES	CLERK	SMITH	800
JONES	CLERK		1900
JONES	ANALYST	FORD	3000
JONES	ANALYST	SCOTT	3000
JONES	ANALYST		6000
JONES	MANAGER	JONES	2975
JONES	MANAGER		2975
JONES			10875
			24025

25 sor kijelölve.

### 14.10. feladat

Listázza ki részlegenként (a részlegek neve szerint rendezve) a dolgozók fizetését munkakörönként csoportosítva, munkakörönként ( minden részlegen belül) képezve a részösszegeket.

#### Megoldás

```
SELECT deptno          AS "Részleg",
       job            AS "Munkakör",
       ename          AS "Dolgozónév",
       SUM(sal)        AS "Fizetés"
  FROM emp
 GROUP BY ROLLUP (deptno, job, ename);
```

#### Eredmény

Részleg	Munkakör	Dolgozónév	Fizetés
10	CLERK	MILLER	1300
10	CLERK		1300
10	MANAGER	CLARK	2450
10	MANAGER		2450
10	PRESIDENT	KING	5000
10	PRESIDENT		5000
10			8750
20	CLERK	ADAMS	1100
20	CLERK	SMITH	800
20	CLERK		1900
20	ANALYST	FORD	3000

20 ANALYST	SCOTT	3000
20 ANALYST		6000
20 MANAGER	JONES	2975
20 MANAGER		2975
20		10875
30 CLERK	JAMES	950
30 CLERK		950
30 MANAGER	BLAKE	2850
30 MANAGER		2850
30 SALESMAN	WARD	1250
30 SALESMAN	ALLEN	1600
30 SALESMAN	MARTIN	1250
30 SALESMAN	TURNER	1500
30 SALESMAN		5600
30		9400
		29025

27 sor kijelölve.

### 14.11. feladat

Listázza ki osztályvezetőként a dolgozói számát, összfizetését, a főosztályvezető nevét és telephelyét, elsődlegesen az osztályvezetők, másodlagosan a főosztályvezetők neve szerint rendezve. Alkalmazzon ismétlésmentes kiíratásokat.

#### Megoldás

```

SELECT O.0Vename          AS "0Vneve",
       COUNT(E.empno)      AS "Létszám",
       SUM(E.sal)           AS "Fizetés",
       F.FOVename          AS "FOVneve",
       FD.loc               AS "telephelye"
FROM emp                   E,
      Osztaly1            O,
      Fosztaly             F,
      emp                  FE,
      dept                FD
WHERE E.empno IN F.empno   AND
      E.empno IN O.empno   AND
      F.FOVempno = FE.empno AND
      FE.deptno = FD.deptno
GROUP BY O.0Vename, F.FOVename, FD.loc;

```

**Eredmény**

OVneve	Létszám	Fizetés	FOVneve	telephelye
FORD	2	3800	JONES	DALLAS
BLAKE	6	9400	BLAKE	CHICAGO
CLARK	2	3750	CLARK	NEW YORK
JONES	1	2975	JONES	DALLAS
SCOTT	2	4100	JONES	DALLAS

**14.12. feladat**

Listázza ki osztályonként (az osztályok vezetőinek neve szerint rendezve) a dolgozók nevét, munkakörét, fizetését, osztályának átlagfizetését és fizetésbeállási értékét (amely a saját fizetés és az osztály átlagos fizetésének a hányadosa). Alkalmazzon ismétlésmentes kiíratásokat.

**Megoldás**

-- Az osztályszintű átlagfizetés nézete

```
CREATE OR REPLACE VIEW OsztÁtlFiz
```

```
AS
```

```
SELECT OVempno,
       OVename,
       ROUND(AVG(sal)) AS OAVGsal
  FROM emp      E,
       Osztály1  O
 WHERE E.empno = O.empno
 GROUP BY OVempno, OVename;
```

PROMPT Osztályszintű átlagfizetés:

```
SELECT * FROM OsztÁtlFiz;
```

PROMPT A dolgozók osztályszintű fizetésbeállási értéke:

BREAK ON "OVneve" ON "OsztÁtlFiz"

```
SELECT O.OVename          AS "OVneve",
       CONCAT(E.ename, ' ')   AS "DolgozóNeve",
       job                     AS "Munkaköre",
       sal                     AS "Fizetése",
       OAVGsal                AS "OsztÁtlFiz",
       TO_CHAR(sal/OAVGsal, '999990.99') AS "FizBeállás"
  FROM emp      E,
       Osztály1  O,
       OsztÁtlFiz OÁF
 WHERE E.empno IN O.empno     AND
       O.OVempno = OÁF.OVempno
 ORDER BY O.OVename;
```

```
CLEAR BREAKS
DROP VIEW OsztÁtlFiz;
```

### Eredmény

A nézet létrejött.

#### Osztályszintű átlagfizetés:

OVEMPNO	OVENAME	OAVGSAL
7566	JONES	2975
7698	BLAKE	1567
7782	CLARK	1875
7788	SCOTT	2050
7902	FORD	1900

#### A dolgozók osztályszintű fizetésbeállási értéke:

OVneve	DolgozóNeve	Munkaköre	Fizetése	OsztÁtlFiz	FizBeállás
BLAKE	ALLEN	SALESMAN	1600	1567	1.02
	WARD	SALESMAN	1250		0.80
	MARTIN	SALESMAN	1250		0.80
	TURNER	SALESMAN	1500		0.96
	JAMES	CLERK	950		0.61
	BLAKE	MANAGER	2850		1.82
CLARK	CLARK	MANAGER	2450	1875	1.31
	MILLER	CLERK	1300		0.69
FORD	SMITH	CLERK	800	1900	0.42
	FORD	ANALYST	3000		1.58
JONES	JONES	MANAGER	2975	2975	1.00
SCOTT	SCOTT	ANALYST	3000	2050	1.46
	ADAMS	CLERK	1100		0.54

13 sor kijelölve.

breaks törölve

A nézet eldobva.

### 14.13. feladat

Listázza ki osztályonként (az osztályok vezetőinek neve szerint rendezve) a dolgozók nevét, munkakörét, fizetését, osztályának összfizetését és a fizetésrészeseidéi értékét (amely a saját fizetés és az osztály összfizetésének a hányadosa). Alkalmazzon ismétlésmentes kiíratásokat.

### 1. megoldás (Külön nézettábla létrehozásával)

-- Az osztályszintű összfizetés nézete

```
CREATE OR REPLACE VIEW OsztÖsszFiz
```

```
AS
```

```
SELECT OVempno,
       OVename,
       SUM(sal) AS OSUMsal
  FROM emp      E,
       Osztály1  O
 WHERE E.empno = O.empno
 GROUP BY OVempno, OVename;
PROMPT Osztályszintű összfizetés:
SELECT * FROM OsztÖsszFiz;
```

PROMPT A dolgozók osztályszintű fizetésrészeseidési értéke:

```
BREAK ON "OVneve" ON "OsztÖsszFiz"
```

```
SELECT O.OVename                               AS "OVneve",
       CONCAT(E.ename, ' ')                   AS "DolgozóNeve",
       job                                     AS "Munkaköre",
       sal                                      AS "Fizetése",
       OSUMsal                                 AS "OsztÖsszFiz",
       TO_CHAR(sal/OSUMsal, '999999990.99')   AS "FizRészeseidés"
  FROM emp      E,
       Osztály1    O,
       OsztÖsszFiz  OÖF
 WHERE E.empno IN O.empno      AND
       O.OVempno = OÖF.OVempno
 ORDER BY O.OVename;
CLEAR BREAKS
DROP VIEW OsztÖsszFiz;
```

### Eredmény

A nézet létrejött.

Osztályszintű összfizetés:

OVEMPNO	OVENAME	OSUMSAL
7566	JONES	2975
7698	BLAKE	9400
7782	CLARK	3750
7788	SCOTT	4100
7902	FORD	3800

A dolgozók osztályszintű fizetésrészeseidési értéke:

OVneve	DolgozóNeve	Munkaköre	Fizetése	OsztÖsszFiz	FizRészesedés
BLAKE	ALLEN	SALESMAN	1600	9400	0.17
	WARD	SALESMAN	1250		0.13
	MARTIN	SALESMAN	1250		0.13
	TURNER	SALESMAN	1500		0.16
	JAMES	CLERK	950		0.10
	BLAKE	MANAGER	2850		0.30
CLARK	CLARK	MANAGER	2450	3750	0.65
	MILLER	CLERK	1300		0.35
FORD	SMITH	CLERK	800	3800	0.21
	FORD	ANALYST	3000		0.79
JONES	JONES	MANAGER	2975	2975	1.00
SCOTT	SCOTT	ANALYST	3000	4100	0.73
	ADAMS	CLERK	1100		0.27

13 sor kijelölve.

breaks törölve

A nézet eldobva.

## 2. megoldás (A RATIO\_TO\_REPORT függvény használatával)

PROMPT A dolgozók osztályszintű fizetésrészesedési értéke:

BREAK ON "OVneve" ON "OsztÖsszFiz"

```

SELECT SUBSTR(OVname,1,6)                               AS "OVneve",
       RPAD(E.ename,11)                                AS "DolgozóNeve",
       job                                         AS "Munkaköre",
       LPAD(sal,8)                                    AS "Fizetése",
       ROUND(sal / (RATIO_TO_REPORT(sal) OVER
                      (PARTITION BY OVempno)))          AS "OsztÖsszFiz",
       TO_CHAR(RATIO_TO_REPORT(sal) OVER
                      (PARTITION BY OVempno), '999999990.99')   AS "FizRészesedés"
FROM emp          E,
      Osztály1      O
WHERE E.empno IN O.empno
ORDER BY O.OVname;
CLEAR BREAKS

```

## Eredmény

A dolgozók osztályszintű fizetésrészesedési értéke:

OVneve	DolgozóNeve	Munkaköre	Fizetése	OsztÖsszFiz	FizRészesedés
--------	-------------	-----------	----------	-------------	---------------

BLAKE	ALLEN	SALESMAN	1600	9400	0.17
-------	-------	----------	------	------	------

	WARD	SALESMAN	1250	0.13
	MARTIN	SALESMAN	1250	0.13
	TURNER	SALESMAN	1500	0.16
	JAMES	CLERK	950	0.10
	BLAKE	MANAGER	2850	0.30
CLARK	CLARK	MANAGER	2450	0.65
	MILLER	CLERK	1300	0.35
FORD	SMITH	CLERK	800	0.21
	FORD	ANALYST	3000	0.79
JONES	JONES	MANAGER	2975	1.00
SCOTT	SCOTT	ANALYST	3000	0.73
	ADAMS	CLERK	1100	0.27

13 sor kijelölve.



#### Megjegyzés

Figyeljünk fel a 2. megoldásban alkalmazott karakteres függvényekre (SUBSTR, RPAD, LPAD, ROUND, TO\_CHAR) is.

### 14.14. feladat

Írjon szkript programot, amely

- főosztályonként és azon belül osztályonként rendezett módon csoportosítva listázza a dolgozók nevét, munkakörét, fizetését, fizetésének az osztálya összfizetéséből való részesedését, valamint elsődlegesen a fizetésének az osztálya összfizetéséből való részesedése szerint csökkenően, másodlagosan a munkaköre szerint növekvő módon az osztályabeli rangsorát (alkalmazzon ismétlésmentes kiíratásokat),
- kimutatást készít arról, hogy a felhasználó által megadott fizetésű új dolgozó a fenti rangsorban hányadik helyre kerülne az egyes osztályokon,
- kimutatást készít arról, hogy a felhasználó által megadott fizetésű új dolgozó az egyes osztályokon mekkora bérbeszültséget okozna belépéssel, ahol a bérbeszültség az új dolgozó fizetésének és az egyes osztályok átlagfizetésének előjeles különbsége.

Az osztályok és a főosztályok azonosítására azok vezetőinek nevét használja.

#### A1. megoldás

PROMPT A dolgozók osztályszintű fizetési rangsora:

BREAK ON "FOVneve" ON "OVneve"

```
SELECT FO.FOVename AS "FOVneve",
       0.OVename AS "OVneve",
       E.ename AS "Dolgozó",
       job AS "Munkaköre",
       sal AS "Fizetése",
       RATIO_TO_REPORT(sal) OVER
          (PARTITION BY OVempno) AS "FizRésze",
```

```

RANK() OVER (PARTITION BY OVempno
              ORDER BY
                sal DESC,
                job ASC)      AS "NormRang",
DENSE_RANK() OVER (PARTITION BY OVempno
                  ORDER BY
                    sal DESC,
                    job ASC)    AS "TömörRang"
FROM emp          E,
      Osztály1   O,
      FőOsztály FO
WHERE E.empno IN O.empno AND
      O.OVempno IN FO.empno
ORDER BY FO.FOVename, O.OVename, "NormRang";
CLEAR BREAKS

```

### Eredmény

A dolgozók osztályszintű fizetési rangsora:

FOVneve	OVneve	Dolgozó	Munkaköre	Fizetése	FizRésze	NormRang	TömörRang
BLAKE	BLAKE	BLAKE	MANAGER	2850	.303191489	1	1
		ALLEN	SALESMAN	1600	.170212766	2	2
		TURNER	SALESMAN	1500	.159574468	3	3
		WARD	SALESMAN	1250	.132978723	4	4
		MARTIN	SALESMAN	1250	.132978723	4	4
		JAMES	CLERK	950	.10106383	6	5
CLARK	CLARK	CLARK	MANAGER	2450	.653333333	1	1
		MILLER	CLERK	1300	.346666667	2	2
JONES	FORD	FORD	ANALYST	3000	.789473684	1	1
		SMITH	CLERK	800	.210526316	2	2
	JONES	JONES	MANAGER	2975	1	1	1
	SCOTT	SCOTT	ANALYST	3000	.731707317	1	1
		ADAMS	CLERK	1100	.268292683	2	2

13 sor kijelölve.

### A2. megoldás (Formázott kiíratással)

```

PROMPT A dolgozók osztályszintű fizetési rangsora:
BREAK ON "FOVneve"  ON "OVneve"
SELECT RPAD(FO.FOVename,7)           AS "FOVneve",
       RPAD(O.OVename,6)            AS "OVneve",
       RPAD(E.ename,7)             AS "Dolgozó",
       SUBSTR(job,1,9)             AS "Munkaköre",
       LPAD(sal,8)                 AS "Fizetése",

```

```

      TO_CHAR(RATIO_TO_REPORT(sal) OVER
              (PARTITION BY OVempno), '9990.99')
              AS "FizRésze",
      RANK() OVER (PARTITION BY OVempno
                  ORDER BY
                      sal DESC,
                      job ASC)           AS "NormRang",
      DENSE_RANK() OVER (PARTITION BY OVempno
                          ORDER BY
                          sal DESC,
                          job ASC)     AS "TömörRang"
FROM emp          E,
      Osztály1    O,
      FőOsztály   FO
WHERE E.empno IN O.empno      AND
      O.OVempno IN FO.empno
ORDER BY FO.FOVename, O.OVename, "NormRang";
CLEAR BREAKS

```

**Eredmény**

A dolgozók osztályszintű fizetési rangsora:

				FOVneve	OVneve	Dolgozó	Munkaköre	Fizetése	FizRésze	NormRang	TömörRang
BLAKE	BLAKE	BLAKE	MANAGER	2850	0.30					1	1
		ALLEN	SALESMAN	1600	0.17					2	2
		TURNER	SALESMAN	1500	0.16					3	3
		WARD	SALESMAN	1250	0.13					4	4
		MARTIN	SALESMAN	1250	0.13					4	4
		JAMES	CLERK	950	0.10					6	5
CLARK	CLARK	CLARK	MANAGER	2450	0.65					1	1
		MILLER	CLERK	1300	0.35					2	2
JONES	FORD	FORD	ANALYST	3000	0.79					1	1
		SMITH	CLERK	800	0.21					2	2
	JONES	JONES	MANAGER	2975	1.00					1	1
	SCOTT	SCOTT	ANALYST	3000	0.73					1	1
		ADAMS	CLERK	1100	0.27					2	2

13 sor kijelölve.

**B. megoldás**

```

SET verify OFF
ACCEPT újfiz PROMPT 'Az új dolgozó fizetése:
PROMPT Az új dolgozó hipotetikus rangja
PROMPT az osztályszintű fizetési rangsorban:

```

```

BREAK ON "FOVneve" ON "OVneve"
SELECT RPAD(FO.FOVename,7) AS "FOVneve",
       RPAD(O.OVename,6) AS "OVneve",
       RANK(&újfiz) WITHIN GROUP
          (ORDER BY
             sal DESC) AS "NormRang",
       DENSE_RANK(&újfiz) WITHIN GROUP
          (ORDER BY
             sal DESC) AS "TömörRang"
FROM emp E,
     Osztály1 O,
     FőOsztály FO
WHERE E.empno IN O.empno AND
      O.OVempno IN FO.empno
GROUP BY FO.FOVename, O.OVename;

CLEAR BREAKS
UNDEFINE újfiz
SET verify ON

```

### Eredmény

Az új dolgozó fizetése: 1222

Az új dolgozó hipotetikus rangja  
az osztályszintű fizetési rangsorban:

	FOVneve	OVneve	NormRang	TömörRang
BLAKE	BLAKE	6	5	
CLARK	CLARK	3	3	
JONES	FORD	2	2	
	JONES	2	2	
	SCOTT	2	2	

### C. megoldás

-- Az osztályszintű átlagfizetés nézete

CREATE OR REPLACE VIEW OsztÁtlFiz

AS

```

SELECT OVempno,
       OVename,
       ROUND(AVG(sal)) AS OAVGsa]
FROM emp E,
     Osztály1 O
WHERE E.empno = O.empno
GROUP BY OVempno, OVename;

```

PROMPT Osztályszintű átlagfizetés:

```
SELECT *
  FROM OsztÁtlFiz
 ORDER BY OVename;
```

SET verify OFF

ACCEPT újfiz PROMPT 'Az új dolgozó fizetése: '

PROMPT Bérfeszültség osztályonként:

```
SELECT RPAD(OVename,7) AS "OsztVez",
       LPAD(OAVGsal,10) AS "OsztÁtlFiz",
       LPAD(&újfiz,12)   AS "ÚjDolgozóFiz",
       &újfiz - OAVGsal AS "ElterésÁtlagtól"
  FROM OsztÁtlFiz
 ORDER BY "OsztVez";
DROP VIEW OsztÁtlFiz;
UNDEFINE újfiz
SET verify ON
```

### Eredmény

A nézet létrejött.

Osztályszintű átlagfizetés:

OVEMPNO	OVENAME	OAVGSAL
7698	BLAKE	1567
7782	CLARK	1875
7902	FORD	1900
7566	JONES	2975
7788	SCOTT	2050

Az új dolgozó fizetése: 2000

Bérfeszültség osztályonként:

OsztVez OsztÁtlFiz ÚjDolgozóFiz ElterésÁtlagtól

BLAKE	1567	2000	433
CLARK	1875	2000	125
FORD	1900	2000	100
JONES	2975	2000	-975
SCOTT	2050	2000	-50

A nézet eldobva.



# **MELLÉKLETEK**



## 1. MELLÉKLET

# A Personal Oracle telepítése

Az Oracle 9.2. egygépes, fejlesztői munkára szánt Personal Oracle változatának telepítési leírása a CD-melléklet 03\_Mellékletek, Segédletek könyvtárának M1.Melléklet alkönyvtárában 1\_Melléklet(Telepítés).pdf néven található.

## 2. MELLÉKLET

# Az Oracle alaptáblái

A továbbiakban megadjuk a feladatokban hivatkozott mintaadattáblákat (ezeket a továbbiakban alaptábláknak nevezzük), melyek az Oracle-rendszerben scott felhasználóként tiger jelszóval érhetők el. (Az Oracle-rendszerbe való belépés leírását lásd 3. melléklet.) E tábláknak a feladatokban hivatkozott magyar nyelvű oszlopneveit a táblák tartalma után szintén megadjuk. (Megjegyezzük, hogy az alábbiakban bemutatásra kerülő emp és dept táblákat generáló szkript programot az 5.1. példa tartalmazza, lásd 5. fejezet.)

## Az alaptáblák és értelmezésük

### Az emp tábla tartalma

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81-NOV-17	5000		10
7698	BLAKE	MANAGER	7839	81-MÁJ-01	2850		30
7782	CLARK	MANAGER	7839	81-JÚN-09	2450		10
7566	JONES	MANAGER	7839	81-ÁPR-02	2975		20
7654	MARTIN	SALESMAN	7698	81-SZE-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	81-FEB-20	1600	300	30
7844	TURNER	SALESMAN	7698	81-SZE-08	1500	0	30
7900	JAMES	CLERK	7698	81-DEC-03	950		30
7521	WARD	SALESMAN	7698	81-FEB-22	1250	500	30
7902	FORD	ANALYST	7566	81-DEC-03	3000		20
7369	SMITH	CLERK	7902	80-DEC-17	800		20
7788	SCOTT	ANALYST	7566	82-DEC-09	3000		20
7876	ADAMS	CLERK	7788	83-JAN-12	1100		20
7934	MILLER	CLERK	7782	82-JAN-23	1300		10

## Az emp tábla értelmezése

Ez a tábla a feladatokban szereplő vállalat dolgozóinak adatait tartalmazza. Az alábbiakban megadjuk a feladatokban hivatkozott magyar nyelvű oszlopneveket.

empno	dolgozó azonosítója
ename	neve
job	munkaköre
mgr	főnökének azonosítója
hiredate	belépési dátuma
sal	havi fizetése
comm	havi jutaléka
deptno	részlegének azonosítója



### Megjegyzés

- A „dolgozó” megnevezés helyett esetenként az „alkalmazott” szót, a „munkakör” helyett pedig a „foglalkozás” szót fogjuk használni.
- A feladatok szövegében gyakran fogunk hivatkozni a dolgozók jövedelmére, mely a fizetés és a jutalék összegét jelenti.
- A fizetésről feltételezzük, hogy állandó érték, így például az éves fizetés ennek 12-szerese. A jutalék havonta változhat, ezért éves jövedelmet e tábla alapján nem tudunk számolni.
- A feladatokban gyakran hivatkozunk a dolgozók főnökeire is. Ezzel kapcsolatban figyelembe kell venni azt, hogy főnök az a dolgozó, akinek azonosítója szerepel az mgr oszlopan. (Gyakori hiba, hogy a manager foglalkozását veszik főnöknek, ami persze hibás, hiszen King nyilván főnök, de a munkaköre president.)
- Figyeljünk fel arra, hogy King-nek nincs főnöke (mgr értéke NULL).

## Az emp tábla szerkezete

Név	Üres?	Típus
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

## A dept tábla tartalma

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

## A dept tábla értelmezése

Ez a tábla a feladatokban szereplő vállalat részlegeinek adatait tartalmazza. Az alábbiakban megadjuk a feladatokban hivatkozott magyar nyelvű oszlopneveket.

deptno	részleg azonosítója
dname	neve
loc	telephelye



### Megjegyzés

- E tábla deptno oszlopának jelentése megegyezik az emp tábla hasonló nevű oszlopának jelentésével, ezért a két tábla összekapcsolása ezen oszlop segítségével törtenhet.
- Figyeljünk fel arra, hogy a 40-es azonosítójú részlegnek nincs dolgozója (lásd az emp táblát).

## A dept tábla szerkezete

Név	Üres?	Típus
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

## A salgrade tábla tartalma

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

## A salgrade tábla értelmezése

Ez a tábla a feladatokban szereplő vállalat fizetési kategóriáinak adatait tartalmazza. Az alábbiakban megadjuk a feladatokban hivatkozott magyar nyelvű oszlopneveket.

grade fizetési kategória sorszáma

losal alsó határa

hisal felső határa



### Megjegyzés

E tábla losal és hisal értékei az emp tábla sal oszlopaira vonatkoznak.

## A salgrade tábla szerkezete

Név	Üres?	Típus
GRADE		NUMBER
LOSAL		NUMBER
HISAL		NUMBER

### 3. MELLÉKLET

# Bevezetés az SQL\*Plus-környezet és az SQL nyelv használatába

Az alábbiakban rövid bevezetést adunk az Oracle adatbázis-kezelő rendszer fejlesztői (egyégen telepítésű) Personal Oracle változata, az SQL\*Plus-környezet, valamint ebben a környezetben az SQL nyelv használatába. Ennek az anyagrésznek a feldolgozását célszerű számítógépen követni. Előtte azonban javasoljuk, hogy az Olvasó tanulmányozza át a 2. mellékletben az Oracle-rendszer alaptábláinak felépítéséről és értelmezéséről leírtakat.

A továbbiakban bemutatásra kerülő utasítások egyrészt részletesen megtalálhatók a [16]-ban, másrészt e példatár különböző fejezetéiben.

## Belépés az SQL\*Plus-környezetbe

Az SQL\*Plus-környezet egy interaktív parancsnyelvi felület, melynek segítségével SQL-parancsok közvetlenül végrehajthatók, formázott listák készíthetők, több utasításból álló, fájlban tárolható szkript programok szerkeszthetők és futtathatók. E szkript programokban változók definiálhatók, melyek révén interaktívvá tehetők, sőt még az Oracle szerveroldali programozását lehetővé tevő PL/SQL programok is elhelyezhetők benne.

Megjegyezzük, hogy e parancsnyelvi felület a Windows operációs rendszer alatt „windowsosan” viselkedik (létezik ugyanis DOS alatt futó változata is). Tehát például a PgUp, PgDn, ↑, ↓ billentyűkkel hátra-előre lehet mozogni a felületen, a Ctrl-C és Ctrl-V billentyűkombinációkkal a szokásos módon lehet akár e felületen belül, akár kifelé (bármilyen szöveges fájlba vagy fájból) szövegeket másolni (ráadásul igen kellemes, oszlopivágásos módon).

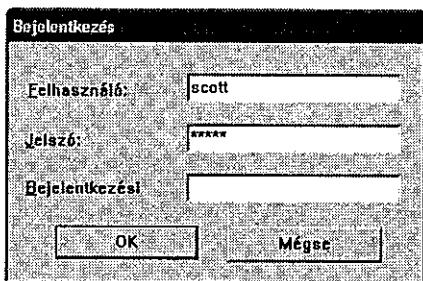
Az SQL\*Plus-környezet tulajdonképpen egy fejlesztői eszköz, melynek segítségével ötletek, megoldások próbálhatók ki annak érdekében, hogy azok később valamilyen alkalmazói rendszer részei legyenek. Előfordul persze az is, hogy egy hirtelen jött alkalmazói feladatot kell megoldani, például egy szépen formázott, jól dokumentált listát készíteni, vagy egy kis interaktív programot készíteni, amely néhány tipikus feladat megoldására alkalmas, mégis könnyen kezelhető stb. Az SQL\*Plus-környezet és szkript programjai erre is remekül megfelelnek. Mind-ebből pedig már következik, hogy az SQL\*Plus-környezet mind az SQL alapszintű tanulásához, mind a professzionális Oracle megléteséhez ideális eszköz.

Nézzük tehát, hogyan lehet belépni az SQL\*Plus-környezetbe! A Personal Oracle telepítése után az SQL\*Plus programot a következőképpen találhatjuk meg:

- egy kattintás a *Start* gombra,
- egy kattintás a *Programok* menüpontra,
- egy kattintás az *Oracle – Orahome 92* mappanévre,
- egy kattintás az *Application Development* mappanévre és már megtaláltuk az *SQL Plus* ikont.

Ezek után vagy kettőt kattintunk erre az ikonra (elindítva ezzel az *SQL\*Plus* programot), vagy (és a magunk részéről inkább ezt javasolnánk) levonszoljuk az Asztalra (desktop) a Ctrl billentyű és az egér bal gombjának egyidejű nyomva tartásával. Amikor később használni akarjuk, elég erre az ikonra kattintani.

Az *SQL\*Plus* program indításakor megjelenik a *Bejelentkezés* ablak. Ennek értelmezéséhez tudni kell, hogy az Oracle professzionális rendszer, ezért csak pontosan tisztázott jogkörű felhasználók léphetnek be, ők is csak felhasználói név és jelszó megadása után. A bejelentkező felület, ahol Personal Oracle esetén csak az első két mező kitöltése szükséges. (Az *SQL\*Plus* program kliens-szerver telepítés esetén is használható, akkor azonban a harmadik mezőt is kell tölni, ezzel most nem foglalkozunk.)



A legtöbb joggal az adatbázis-adminisztrátor rendelkezik. Ő tetszés szerint (az adott számítógép hardverlehetőségein és az operációs rendszer képességeinek korlátain belül) átkonfigurálhatja az adatbázis-kezelő rendszert, új felhasználókat definiálhat, ezeket jogokkal láthatja el, új adatbázis-területeket hozhat létre stb. (Lásd 11. fejezet.) Adatbázis-adminisztrátorként alapértelmezésben a *Sys* felhasználó névvel kellett belépni a *change\_on\_install* jelszóval. A 9.2-es verzióban azonban a telepítőrendszer e jelszó használatát már nem engedi (túl sokan ismerek), és a telepítés során egy új jelszót kell megadni. A *Sys* felhasználónév esetén a bejelentkezési felületen jelszóként megadható a

*Telepítésijelszó as sysdba*

karaktersorozat, ahol a *Telepítésijelszó* a telepítés során a *Sys* felhasználóhoz általunk bevezetett jelszó.

Korlátozottabb lehetőségekkel, de felhasználói és tábladefiníciós jogokkal rendelkezik a *System* felhasználó, akinek az alapértelmezés szerinti *manager* jelszavát szintén a 9.2-es verzió már nem engedélyezi, helyette valami más kell megadni telepítéskor. Ha *system* felhasználóként lépünk be az *SQL\*Plus*-környezetbe, akkor jelszóként a *System* felhasználóhoz általunk bevezetett jelszót kell beírni.

Az Oracle-rendszer leggyakrabban használt mintaadattábláinak tulajdonosa a Scott felhasználó, akinek tiger jelszava szerencsére nem változott az utolsó tíz évben, így ezzel már könnyűszerrel birtokba vehetjük az SQL\*Plus-környezetet.

Ha már valamelyik felhasználóként bejutottunk az SQL\*Plus-környezetbe, akkor bármelyik másik felhasználó környezetébe átléphetünk a

```
CONNECT FelhasználóNév/jelszó [AS {SYSDBA | SYSOPER}]
```

SQL\*Plus-utasítással, ahol az AS SYSDBA, illetve az AS SYSOPER záradékok valamelyikének megadása a Sys felhasználó esetén szükséges.

A sikeres belépés után megjelenik az SQL\*Plus prompt jele (SQL> ). minden SQL- és SQL\*Plus-utasítást ez után kell írni.

Az SQL\*Plus-környezetben (mint már jeleztük) egyaránt kiadhatók SQL-utasítások, és a környezeti beállításokat végző, és a szkript programokat szerkesztő és futtató SQL\*Plus-utasítások. Megjegyezzük, hogy az SQL-utasításokat pontosvesszővel (;) kell lezárni, míg az SQL\*Plus-utasításokat közvetlenül a sorvégiel zárja. Az utasítások használatával kapcsolatos további tudnivaló, hogy az SQL-utasítások tördelhetők több sorba (sőt az áttekinthetőség érdekében célszerű is tördelni), míg az SQL\*Plus-utasításokat egyetlen sorba kell írni.

## Felhasználói táblák lekérdezése

### Felhasználói katalógus és adatszótár

Belépés után kérdezzük le a *felhasználói katalógust* annak érdekében, hogy megtudjuk, a felhasználónak milyen adattáblái, illetve tábla jellegű objektumai vannak. Ezek lekérdezhetők az Oracle-adatbázis nézeteiből. (Az alábbiakban az SQL\*Plus-környezetben kiadott utasítások elő odaírjuk az SQL\*Plus készenléti, azaz prompt jelét, az SQL> karakter sorozatot.)

```
SQL> SELECT * FROM user_catalog;
```

illetve a régebbi verziókban:

```
SQL> SELECT * FROM cat;
```

Ily módon megkapjuk mindeneket a táblákat, nézeteket stb., amelyek az adott felhasználó tulajdonában vannak.

A későbbiek során majd érdekes lehet az egyéb objektumokat is tartalmazó *felhasználói adatszótár*, melyet a

```
SQL> SELECT * FROM user_objects;
```

utasítás segítségével kérdezhetünk le.

## Felhasználói adattábla lekérdezése

Kérdezzük le ezek után a Scott felhasználó tulajdonában levő emp tábla tartalmát:

```
SQL> SELECT * FROM emp;
```

Ha ez lesz az olvasó első lekérdezése, akkor láthatja, hogy a lista több sorba tördelet, nem áttekinthető. Miért? Azért, mert az SQL\*Plus-környezetben a listázandó sor hossza nem megfelelő e tábla listázására. Kérdezzük le az SQL\*Plus-környezet sorhossz beállítását.

```
SQL> SHOW linesize
```

Az eredmény 80. Ez láthatóan kevés ahhoz, hogy az emp tábla minden oszlopa egy sorba kerüljön, azaz egy rekord egy sorban jelenjen meg. Állítsuk be a sorhosszt, mely egy SQL\*Plus-környezeti rendszerváltozó.

```
SQL> SET linesize 400
SQL> SELECT * FROM emp;
```

Az ekkor megjelenő lista már majdnem jó, mert egy rekordot egy sorba ír, függőlegesen azonban kétszer írja a fejlécet, ez pedig zavaró. Tehát kérdezzük le a lapméretet:

```
SQL> SHOW pagesize
```

A kapott érték nyilvánvalóan kevés az emp tábla egy fejléccel való megjelenítéséhez, tehát növeljük meg a lapméret rendszerváltozót:

```
SQL> SET pagesize 40,
```

majd újra kérdezzük le az emp tábla teljes tartalmát:

```
SQL> SELECT * FROM emp;
```

## Felhasználóváltás

Nézzük meg, mi történik, ha újraindítjuk az SQL\*Plus programot, de nem Scott felhasználóként lépünk be, hanem Systemként! Ha megpróbáljuk lekérdezni az emp táblát, nem fog sikerülni. Ennek az az oka, hogy az emp tábla a Scott „tulajdona”. Lehetőségünk van azonban arra, hogy átlépjünk egy másik felhasználó környezetébe (persze csak akkor, ha tudjuk a jelszavát). Adjuk ki az alábbi SQL\*Plus-utasítást

```
SQL> CONNECT scott/tiger
Kapcsolódva.
```

Most lekérdezve az emp táblát, már megjelenik a szokásos lista.

# Szkript programok írása, futtatása

Az iménti lista már megfelelő lesz számunkra (lásd 2. melléklet). Annak érdekében, hogy minden SQL\*Plus-környezetbe való belépéskor ez legyen a munkakörnyezet, létre kell hoznunk egy `login.sql` szkript programot a `c:\oracle\ora92\bin` alkönyvtárban (ha a C: meghajtóra és ora92 néven telepítettük az Oracle-rendszert). Ez a szkript az SQL\*Plus minden indításakor automatikusan lefut, ezáltal inicializálva a környezetet. Az alábbiakban megmutatjuk, hogy miként lehet egyszerűen létrehozni és futtatni egy tetszőleges SQL\*Plus szkript programfájlt.

## Szövegszerkesztés az SQL\*Plus-környezetben

### A `login` szkript létrehozása

Mivel az SQL\*Plus rendelkezik saját szövegszerkesztővel, a legegyszerűbb, ha azt használjuk. Ezt az ed fájlnév SQL\*Plus-utasítással hívhatjuk meg. Az általános gyakorlási célokra válasszunk valamelyen egyszerű programfájlnevet, például abc, vagy aaa, illetve a `login.sql` létrehozása érdekében login. Az

```
SQL> ed login
```

utasítás kiadása után a beugró ablakban megnyíló szövegszerkesztőbe írjuk be például a következő sorokat:

```
SET linesize 400
SET pagesize 40
SELECT * FROM scott.emp;
```

```
login.sql - Jegyzettömb
Fájl Szerkesztés Formátum Nézet Súgó
set linesize 400
set pagesize 40
SELECT * FROM scott.emp;
```

majd zárjuk be a szövegszerkesztőt (például Alt-F4 + Enter parancsal vagy a szokásos grafikus módon). Megjegyezzük, hogy az így szerkesztett szkriptfájl automatikusan megkapja az `sql` kiterjesztést. (Ezúttal egyébként a Scott felhasználó tulajdonában lévő emp táblára `scott.emp` módon hivatkoztunk, amely módon már a System felhasználó is elérheti.)

## A login szkript futtatása

Ha kilépünk az SQL\*Plus-környezetből (az EXIT vagy a QUIT utasítással), majd újra belépünk, akkor már észlelhetjük a login szkriptünk hatását, de a benne szereplő SQL-lekérdezőutasítás révén a közvetlen futtatás is elég látványos. Adjuk ki az

```
SQL> @login
```

utasítást. Ezzel lefuttatjuk a login szkriptet, és megjelenik az emp tábla listája.

## Általános szkript program létrehozása és futtatása

Megjegyezzük, hogy általában „nem illik” egy telepített program rendszerkönyvtáraiba bármit is írni (bár a login szkript kivétel), különösen nem gyakorlási célokra szolgáló mintaprogramokat. Természetesen tetszőleges útvonal is megadható mind az ed szerkesztő, mind az @ futtatóparancshoz. Tehát kiadható például az

```
SQL> ed c:\gyakorlás\abc
```

és az

```
SQL> @c:\gyakorlás\abc
```

utasítás, ezeket viszont sokszor leírni nem éppen kellemes, ezért ha minden ugyanazt a néhány (például aa1, aa2 stb.) programfájlnevet használjuk, akkor a bin alkönyvtárba írás tekinthető akár bocsánatos bűnnek is...

A továbbiakban az utasítások előtt nem fogjuk odaírni az SQL\*Plus prompt jelét.

## Kiíratás szkript programból

Gyakran előfordul, hogy valamilyen értelmező szöveget szeretnénk kiírni egy lista előtt vagy után. Erre szolgál az SQL\*Plus-környezetben a PROMPT utasítás.

Pontos alakja:

```
PROMPT [szöveg]
```

ahol a megadott szöveg megjelenik a felhasználó képernyőjén, illetve ennek hiánya esetén egy üres sor (azaz egy sora meléles).

## Megjegyzések használata

Egy programba az áttekinthetőség és a dokumentálás érdekében megjegyzéseket helyezhetünk el, melyeket az Oracle a futtatható kód előállításakor figyelmen kívül hagy. Ezt háromféleképpen is megtehetjük.

## A REM TÍPUSÚ MEGJEGYZÉS

Csak SQL\*Plus szkriptben használható, PL/SQL nyelvű kódban nem. A sorban nem előzheti meg más, csak szóköz karakter.

rem E megjegyzés előtt csak szóköz állhat a sorban

## A -- TÍPUSÚ MEGJEGYZÉS

Egyaránt használható SQL\*Plus-szkriptben és PL/SQL\_nyelvű kódban. A sorban ezt sem előzheti meg más, csak szóköz karakter.

-- E megjegyzés előtt is csak szóköz lehet

## A /\* \*/ TÍPUSÚ MEGJEGYZÉS

Csak PL/SQL nyelvű kódban használható. A sorban előtte és utána is lehet utasítás, és az ilyen megjegyzés többsoros is lehet. Gyakran alkalmazzuk programok tesztelésénél, amikor egy (akár többsoros) kódrészét szeretnénk átmenetileg kihagyni a futtatásból.

utasítás /\* Ez előtt és után is állhat utasítás,  
és akár többsoros is lehet. \*/

## A sorfolytató jel használata

Az SQL\*Plus utasításai úgynevezett egysoros utasítások, azaz elvben egyetlen sorban el kellene férniük. Mivel a sorok gyakorlatilag tetszőleges hosszúak lehetnek (bár az alkalmazott programszöveg-szerkesztőknek általában van erre felső korlátjuk), ez nem jelent elvi korlátot, ám igen megnehezíti egy hosszabb utasítás áttekintését.

A fenti probléma elkerülése érdekében vezették be a „sorfolytató” jelet, mely egy, a sor végeré írt „-” karakter. Az SQL\*Plus parancsértelmező programja ekkor megvárja a következő sort is, és csak annak beolvasása után (ha ez már nem tartalmaz további sorfolytató jelet) értelmezi az utasítást. A sorfolytató jele(ke)t az SQL\*Plus parancsértelmezője természetesen figyelmen kívül hagyja.

Ennek értelmében a

```
SET linesize 400
SELECT * FROM emp;
```

és a

```
SET -
linesize -
400
```

SELECT \* FROM emp;  
utasítás párok egyenértékűek.

## Változók használata

A már definiált változókat és azok értékét lekérdezhetjük a

DEFINE  
utasítással. Változókat (karakteres típusúakat) definiálhatunk a  
DEFINE változó = érték  
utasítással, és az

ACCEPT változó PROMPT szöveg  
utasítással, mely felhasználói adatbekérésre használható. A definiált változókra hivatkozhatunk az  
&változó  
illetve, ha jobbról közvetlenül szöveg követi, akkor  
&változó.

módon (tehát ponttal a végén), és kiírathatjuk a képernyőre a  
PROMPT &változó

utasítással. Egy definiált változót törölhetünk az  
UNDEFINE változó  
utasítással. (Mindezkről részletesebben szó esik a 4. fejezetben.)

## Az SQL\*Plus editorának lecserélése

Mivel az SQL\*Plus saját szövegszerkesztője meglehetősen egyszerű, előfordulhat, hogy le szeretnénk cserálni egy általunk jobban kedvelt editorra. Tegyük fel, hogy a kedvenc editorunk a Windows write.exe nevű programja. Ekkor ki kell adnunk a

DEFINE\_EDITOR="write.exe"

SQL\*Plus-utasítást, és az ed szerkesztőparancs hatására már a write.exe fog megjelenni. Ha azt szereténk, hogy a továbbiakban minden ez jelenjen meg, akkor a fenti utasítást el kell helyeznünk a login.sql szkriptbe is (az oracle\ora92\bin alkönyvtárba).

## Tábla létrehozása, feltöltése, törlése, dátumformák

Az alábbiakban röviden ismertetjük a táblák létrehozásával kapcsolatos alapvető DDL- (Data Definition Language) utasításokat.

### 1. Tábla létrehozása, adattípusok

Hozzunk létre egy minta nevű adattáblát. Legyen egy név adatmezője, mely változó méretű karakteres típusú, egy szül adatmezője, mely dátum típusú, és egy tan\_át1 adatmezője, mely egy tizedest tartalmazó numerikus típus:

```
CREATE TABLE minta
  (név      VARCHAR2(20),
   szül     DATE,
   tan_át1  NUMBER(3,1));
```

### Eredmény

A tábla létrejött.



#### Megjegyzés

A fenti VARCHAR2(20) definíciós alak egy, legfeljebb 20 karakteres szöveg típusú adat tárolására alkalmas adatmezőt jelöl.

A NUMBER(3,1) definíciós alak tizedesszám formát jelöl, ahol 3 az összes karakterek száma, 1 pedig a tizedesjegyek száma. Mivel egy karakter szükséges a tizedespont számára, így e módon olyan számok ábrázolhatók, melyek egy egész jegyet és egy tizedest tartalmaznak (például 4.6). Ügyeljünk arra, hogy ha negatív érték is lehetséges, akkor még egy karakterhelyre szükség van, ekkor a megfelelő alak tehát: NUMBER(4,1).

### 2. A tábla felépítésének lekérdezése

```
SET linesize 60
DESC minta
SET linesize 400
```

### Eredmény

Név	Üres?	Típus
NÉV		VARCHAR2(20)
SZÜL		DATE
TAN_ÁTL		NUMBER(3,1)

**Megjegyzés**

A sorhossz (length) átállítására azért volt szükség, mert különben nem láttuk volna az eredménysor végét.

**3. A dátum ábrázolása**

Hogyan ábrázolja az Oracle-rendszerünk a dátumot? Kérdezzük le a dual segédtábla segítségével:

```
SELECT sysdate
      FROM dual;
```

**Eredmény**

```
SYSDATE
-----
04-DEC-28
```

Tehát <év (kétféle formátumban) – <hónap neve (3 betűs magyar rövidítéssel) – <a nap száma>. A kétjegyű évszám viszont ma a 20xx-et jelenti. Annak érdekében, hogy az általunk megadott (például múlt századi) dátum biztosan helyes legyen, célszerű az évszázadot is megadni. Egy múlt századi dátum megadása például az '1981-ÁPR-05' alakban történhet.

**4. Tábla feltöltése**

Egy tábla feltöltéséhez a tárolandó adatokat a megfelelő formában kell megadni; a karakter adattípushoz és a dátum adattípushoz például aposztrófok ('') között.

Az imént létrehozott minta nevű adattáblába vigyük be egy rekordot.

```
INSERT INTO minta
VALUES('CSÍNOM PALKÓ','1981-ÁPR-05',2.9);
```

**Eredmény**

1 sor létrejött.

**5. Tábla lekérdezése**

Kérdezzük le a minta nevű adattáblánkat.

```
SELECT *
      FROM minta;
```

**Eredmény**

NÉV	SZÜL	TAN_ÁTL
CSÍNOM PALKÓ	81-ÁPR-05	2.9

## 6. Rendszerdátum formátumának módosítása

Állítsuk át a rendszerdátumot a szokásos magyar formátumra, és kérdezzük le ismét az adattáblánkat.

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

```
SELECT *
  FROM minta;
```

### Eredmény

A munkamenet módosítva.

NÉV	SZÜL	TAN_ÁTL
CSÍNOM PALKÓ	1981.ÁPR.05	2.9

## 7. Tábla törlése

Végül töröljük a minta nevű adattáblánkat.

```
DROP TABLE minta;
```

### Eredmény

A tábla eldobva.

## 8. Rendszerdátum formátumának visszaállítása

Állítsuk vissza a rendszerdátumot az eredeti (évszázadhiányos) formátumra.

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

### Eredmény

A munkamenet módosítva.

#### 4. MELLÉKLET

# Elveszett függőségek megvalósítása triggerrel

## I. A probléma felvetése (elméleti háttér)

Az alábbi problémafelvetés a [16] könyv II. részében, a Boyce–Codd normálformánál bevezetett példa leírásánál található (lásd [16] 224–226. o.). A bemutatott megoldás a felmerült problémák egy lehetséges megoldását szemlélteti, de egyben bemutatja a triggersok és tárolt eljárások használatát is.

Megjegyezzük, hogy az alábbi feladat megoldásához szükséges ismeretanyag *nem* része a példatárnak. A bemutatott példa minden össze a cél szolgálja, hogy rávilágítunk a triggersok egy igen fontos alkalmazására, az adatbázis-tervezés során „elveszett” függőségek, a többtáblás megszorítások megvalósítására. Ha az olvasónak nincs kedve elmélyedni e téma körben, akkor nyugodtan kihagyhatja ezt a mellékletet.

### 4M.1. példa

Legyen egy iskolai nyilvántartás relációsémája a

$\{ \text{Tantárgy}, \text{Tanár}, \text{Diák} \}$

attribútumhalmazon értelmezett

$R(\text{Tantárgy}, \text{Tanár}, \text{Diák})$ .

Állítsuk elő az iskolai nyilvántartás adatbázismodelljét, ha az alábbi ismeretek állnak a rendelkezésünkre:

- i1. Egyetlen attribútum sem határozza meg egyértelműen az összes többit.
- i2. A tantárgy és a tanár együttesen nem határozzák meg a diákokat.
- i3. A diákoknak minden tanáruk csak egy tantárgyat tanít (ebben az iskolában).
- i4. Általában is igaz, hogy minden tanár csak egy tantárgyat tanít (ebben az iskolában).
- i5. A diákoknak minden tantárgyat csak egyetlen tanár tanít (ez elég általános).

## Megoldás

### 1. lépés

Írjuk fel a fenti ismereteket függőségekként. Először azonban – ha triviális is – írjuk fel az

$$f_0 = \{ \text{Tantárgy}, \text{Tanár}, \text{Diák} \} \rightarrow \{ \text{Tantárgy}, \text{Tanár}, \text{Diák} \}$$

egységfüggőséget, mivel ez kifejezi azt, hogy a

$$\{ \text{Tantárgy}, \text{Tanár}, \text{Diák} \}$$

attribútumhalmaz kulcs. A dekompozíciós Armstrong-szabály (lásd [16] 200. o.) értelmében az  $f_0$  függőség egyenértékű módon felbontható az

$$f_{01} = \{ \text{Tantárgy}, \text{Tanár}, \text{Diák} \} \rightarrow \{ \text{Tantárgy} \},$$

$$f_{02} = \{ \text{Tantárgy}, \text{Tanár}, \text{Diák} \} \rightarrow \{ \text{Tanár} \}, \text{ és}$$

$$f_{03} = \{ \text{Tantárgy}, \text{Tanár}, \text{Diák} \} \rightarrow \{ \text{Diák} \}$$

függőségekre. Tekintsük át ezek után az  $i1-i5.$  ismeretekből következő függőségeket:

1.1. Az  $i1$ -ből nem következik függőség, mindössze annyi, hogy egyszerű kulcs nem lesz megfelelő.

1.2. Az  $i2.$  szerint a  $\{ \text{TANTÁRGY}, \text{TANÁR} \}$  összetett kulcsként szintén nem megfelelő.

1.3. Az  $i3$ -ból már következik az alábbi függőség:

$$f_1 = \{ \text{TANÁR}, \text{DIÁK} \} \rightarrow \{ \text{TANTÁRGY} \}.$$

1.4. Az  $i4.$  függőségeként felírva:

$$f_2 = \{ \text{TANÁR} \} \rightarrow \{ \text{TANTÁRGY} \}.$$

1.5. Végül az  $i5.$  is felírható függőségeként:

$$f_3 = \{ \text{TANTÁRGY}, \text{DIÁK} \} \rightarrow \{ \text{TANÁR} \}.$$

Nyilvánvalóan még számos további függőséget felírhatnánk, azonban ezek – hasonlóan az  $f_0$ ,  $f_{01}$ ,  $f_{02}$ ,  $f_{03}$  függőségekhez – triviálisak volnának, vagyis nem tükröznek újabb ismereteket vizsgálatunk tárgyáról.

### 2. lépés

Elemezzük a kapott függőségeket. Láthatóan az  $f_1$  és  $f_2$  függőségek miatt az **R** relációséma nem 2NF alakú, az  $f_2$  és  $f_3$  függőségek miatt pedig nem BCNF alakú.

### 3. lépés (A feladat megoldása)

Normalizálás. Az  $f_1$  és  $f_3$  függőségek alapján a

$$\{ \text{TANÁR}, \text{DIÁK} \}, \text{ és a}$$

$$\{ \text{TANTÁRGY}, \text{DIÁK} \}$$

attribútumhalmazok (alternatív) kulcsai az  $R(TANTÁRGY, TANÁR, Diák)$  relációsémának. Elvégezve (vagy legalábbis megkísérelve) a szokásos (1NF, 2NF, 3NF, BCNF) normalizálási lépések, az alábbi dekompozíciót kapjuk:

$$\varphi[R(Tantárgy, Tanár, Diák)] = \{ R_1(Tantárgy, Tanár), \\ R_2(Tanár, Diák) \},$$

melyben nyilván

$$K\{R_1\} = \{TANÁR\}, \text{ és} \\ K\{R_2\} = \{TANÁR, Diák\}.$$

A fenti  $\varphi$  dekompozíció egyben az iskolai nyilvántartás adatbázismodellje is.



### Megjegyzés

Vizsgáljuk meg a fenti  $\varphi$  dekompozíciót. (Megjegyezzük, hogy a [16]-beli Relációséma veszteségmentes dekompozíciója című pontban bemutatott példa  $R(A)$  relációsémája  $A = \langle TANTÁRGY, Diák, TANÁR \rangle$ , valamint  $\varphi_2 = \varphi$  összerendeléssel a jelen példához hasonlít.) Könnyen belátható, hogy a  $\varphi$  dekompozíció veszteségmentes, és az  $f_2$  függőséget közvetlenül megőrzi ( $R_1$ -ben).

Tekintsük az  $f_1$  függőséget. Bármely konkrét  $\langle TANÁR_0, Diák_0 \rangle$  értékpáros az  $R_2$  relációsémában nyilván csak egyszer fordulhat elő, az  $R_1$ -beli TANÁR attribútumra vonatkozó  $f_2$  függőség miatt pedig e pá�  $TANÁR_0$  értékéhez csak egyetlen  $TANTÁRGY_0$  érték tartozhat, tehát teljesül az  $f_1$  függőség is.

Figyeljünk fel arra, hogy az  $f_1$  függőséget nem közvetlenül őrzi meg a  $\varphi$  dekompozíció. Nem is teheti, hiszen az általa tartalmazott alsémák egyikére sem értelmezhető  $f_1$ . Ezt az  $R_1$  és az  $R_2$  relációsémák együttesen őrzik meg. Azt mondhatjuk tehát, hogy az  $f_1$  függőséget a  $\varphi$  dekompozíció strukturálisan őrzi meg. Ez új jelenség, tárgyalásunk keretei azonban nem engedik meg ennek részletes kifejtését.

Végül tekintsük az  $f_3$  függőséget. Ez nyilvánvalóan nem értelmezhető a  $\varphi$  dekompozíció által tartalmazott alsémákban, de a  $\varphi$  ezt strukturálisan sem őrzi meg, vagyis az  $f_3$  függőség „elvészett”. Mi ennek a következménye? Mint korábban láttuk, a nem függőségőrző dekompozíciók esetén lehetséges „hamis” rekordok felvitele, vagyis bevhetünk az alsémákba olyan rekordokat, melyeket az eredeti relációsémán értelmezett függőségek megtiltanának.

Például, ha a Kiss Dénes diáknak Tóth Pál a matematikatanára, akkor az  $R(TANTÁRGY, TANÁR, Diák)$  relációsémára épülő adattáblába az  $f_3$  függőség nem engedi felvinni a  $\langle "matematika", "Kara Péter", "Kiss Dénes" \rangle$  rekordot, míg e rekord  $\langle "matematika", "Kara Péter" \rangle$ , illetve  $\langle "Kara Péter", "Kiss Dénes" \rangle$  vetületei az  $R_1$ , illetve  $R_2$  alsémák tábláiba minden nehézség nélkül bevhetők. Az pedig, hogy baj történt, csak akkor derül ki, amikor lekérdezzük a diákok adatait és a

Diák	TANTÁRGY	TANÁR
...	...	...
Kiss Dénes	matematika	Tóth Pál
Kiss Dénes	matematika	Kara Péter
...	...	...

listát kapjuk, mely már nyilvánvalóan ellentmondásban van az  $f_3$  függőséggel.

A normalizálás ebben az esetben tehát megsértette az adatbázis integritását, vagyis elveszítette az eredeti modellben megfogalmazott függőségek egy részét. (A függőségeket az Oracle-rendszerben *megszorításoknak* nevezik.)

A gyakorlatban az ilyen esetekben mindig el kell gondolkodnunk azon, hogy vajon nem történt-e „túlnormalizálás”. Tény azonban, hogy az esetleges adatvesztések (melyek az adatbázis-anomáliák révén érhetik az adatbázist – lásd [16]) általában visszariasztják a tervezőket a denormalizálásuktól, és az adatbázismodell integritási sérüléseit inkább más módon korrigálják. (Például az adatbevitelkor aktivizálódó speciális eljárások, az úgynévezett triggerek segítségével többtáblás megszorításként valósítják meg a dekompozíció során „elveszett” megszorításokat.)

## II. A probléma megoldása triggerrel

A fentiekben a probléma lényege az volt, hogy a normalizálás során egyszerűen elveszett egy függőség, ennek hatására pedig hamis rekordok kerülhettek az egyik táblába adatbevitelkor. Azt mondjuk, hogy ilyenkor lényegében két lehetősége van az adatbázis tervezőjének.

Az egyik az, hogy denormalizál, vagyis az érintett relációsémákat visszaviszi egy alacsonyabban normalizált szintre. Ennek azonban az a következménye, hogy kiteszi a rendszerét az adatbázis-anomáliák miatt bekövetkezhető adatvesztésnek.

Egy másik módszer során nem nyúl a táblák szerkezetéhez (a relációsémákhoz). Az elveszett függőséghez tartozó adatbeviteli megszorításokat ekkor az adatbevitelkor aktivizálódó triggerek segítségével valósítja meg. Alábbi példánkban ezt az esetet mutatjuk be.

### 4M.2. példa

Valósítsuk meg triggerrel és tárolt eljárásokkal az ellenőrzött adatbevitelt az előző példára vonatkozóan. Pótoljuk e példabeli normalizálási folyamatban „elveszett” függőséget triggerrel.

#### Megoldás

##### Előzetes megjegyzések az 1. és 2. lépéshöz

Az alábbi 1-Tabla\_R1.sql és a 2-Tabla\_R2.sql programok létrehozzák és néhány mintaadattal feltöltik az R1 és R2 táblákat. (A triggerhasználat egyik jellegzetességeként lehetőség van egy úgynévezett jegyzőkönyvtábla használatára is, a 2-Tabla\_R2.sql program ennek létrehozását is tartalmazza R22 néven.)

##### 1. lépés (Az R1 tábla létrehozása és feltöltése adattal – 1-Tabla\_R1.sql)

```
-- 1-Tabla_R1.sql
DROP TABLE R1;

CREATE TABLE R1
(tanar      VARCHAR2(20)  PRIMARY KEY,
 tantargy   VARCHAR2(10));
```

COMMIT;

```
INSERT INTO R1
    VALUES('Nagy', 'matek');
INSERT INTO R1
    VALUES('Kis', 'fizika');
INSERT INTO R1
    VALUES('Tóth', 'matek');

SELECT * FROM R1;
```

### Eredmény

A tábla eldobva.  
A tábla létrejött.  
A jóváhagyás befejeződött.

1 sor létrejött.  
1 sor létrejött.  
1 sor létrejött.

TANAR	TANTARGY
Nagy	matek
Kis	fizika
Tóth	matek

### 2. lépés (Az R2 tábla létrehozása és feltöltése adattal – 2-Tabla\_R2.sql)

```
-- 2-Tabla_R2.sql
DROP TABLE R2;
DROP TABLE R22;
```

```
CREATE TABLE R2
(tanar    VARCHAR2(20),
diak     VARCHAR2(20),
oraszam  NUMBER(2));
```

```
CREATE TABLE R22 AS
(SELECT * FROM R2);
```

COMMIT;

```
INSERT INTO R2
    VALUES('Nagy', 'Sas', 21);
INSERT INTO R2
```

```

    VALUES('Kis', 'Sas', 17);
INSERT INTO R2
    VALUES('Nagy', 'Szabó', 21);
INSERT INTO R2
    VALUES('Kis', 'Szabó', 17);

SELECT * FROM R2;
SELECT * FROM R22;

```

**Eredmény**

A tábla eldobva.  
A tábla eldobva.  
A tábla létrejött.  
A tábla létrejött.  
A jóváhagyás befejeződött.

1 sor létrejött.  
1 sor létrejött.  
1 sor létrejött.  
1 sor létrejött.

TANAR	DIAK	ORASZAM
Nagy	Sas	21
Kis	Sas	17
Nagy	Szabó	21
Kis	Szabó	17

nincsenek kijelölve sorok

**3. lépés (3-TaroltProg1.sql)**

A 3-TaroltProg1.sql program a szükséges feltételt tartalmazó tárolt eljárást valósítja meg. E program futtatásának hatására a benne foglalt eljárás bekerül a szerver adatbázis-területére, és később az aktualizált paraméterlistájával kiegészített nevének meghívásával bármikor futthatóvá válik.

```

-- 3-TaroltProg1.sql
CREATE OR REPLACE FUNCTION Tilt(tanarnev IN CHAR, diaknev IN CHAR)
RETURN BOOLEAN
IS
    tanarszam INTEGER;

BEGIN
    SELECT COUNT(kozostanar)

```

```

INTO tanarszam
FROM (((SELECT R1.tanar AS kozostanar
          FROM R1
         WHERE (R1.tantargy = (SELECT R1.tantargy
                                FROM R1
                               WHERE R1.tanar = tanarnev)) AND
               (R1.tanar <> tanarnev)))
INTERSECT -- a halmazmetszet művelete
(SELECT R2.tanar as kozostanar
          FROM R2
         WHERE R2.diak = diaknev)));
DBMS_OUTPUT.PUT_Line ('közös tanárok száma =' || tanarszam);

IF tanarszam = 0
THEN RETURN TRUE;
ELSE RETURN FALSE;
END IF;
END;
/

```

### Eredmény

A függvény létrejött.

### 4. lépés (4-Trigger1.sql)

A 4-Trigger1.sql program tartalmazza a triggereljárást. Ez is az adatbázis-területre kerül, ám ennek aktivizálása nem a hagyományos, névvel történő meghívással történik, hanem a benne leírt esemény bekövetkezésének hatására. Jelen esetben az aktivizálódást kiváltó esemény az R2 táblába irányuló rekordbeszúrás INSERT utasításának kiadása. Mint látható, ez a trigger még a triggerelő esemény végrehajtása előtt lefut. (Lehetne olyan triggert is írni, mely a triggerelő esemény végrehajtása után fut le.) A futása során, a korábban eltárolt Tilt nevű tárolt eljárás segítségével megvizsgálja, hogy a beszúráshoz szükséges feltételek fennállnak-e. Ha nem, akkor nem csupán a beírást akadályozza meg, de hibaüzenetet is küld. A sikeres beszúrást nem csupán az R2 táblába, hanem ezzel egyidejűleg a fent említett jegyzőkönyvtáblába is beírja.

```

-- 4-Trigger1.sql
CREATE OR REPLACE TRIGGER Trigi
BEFORE INSERT ON R2
FOR EACH ROW

DECLARE
  Hiba EXCEPTION;

```

```

BEGIN
  IF INSERTING
  THEN
    IF Tilt(:new.Tanar, :new.Diak)
    THEN
      -- Az alábbi INSERT parancsot végrehajtja
      -- az R2-n és az R22 táblákon
      INSERT INTO R22
        VALUES (:new.tanar, :new.diak, :new.oraszam);
    ELSE
      RAISE Hiba;
    END IF;
  END IF;

EXCEPTION
  WHEN Hiba
  THEN
    RAISE_APPLICATION_ERROR(-20001, 'Hiba a Trigi trigger futása során...!');
END;
/

```

### Eredmény

A trigger létrejött.

### 5. lépés (Beírási kísérletek)

Egy sikeres és egy sikertelen beírási kísérletet valósít meg az 5-Bevitel\_R2.sql program, majd e kísérlet eredményét listázza a 6-Listazas.sql szkript program.

```

-- 5-Bevitel_R2.sql
DELETE R22;

INSERT INTO R2
  Values('Molnár', 'Sas', 22);
INSERT INTO R2
  Values('Tóth', 'Sas', 32);

SELECT * FROM R2;
SELECT * FROM R22;

```

### Eredmény

0 sor törölve.

1 sor létrejött.

```
INSERT INTO R2
```

```
*
```

Hiba a(z) 1. sorban:

ORA-20001: Hiba a Trigi trigger futása során...!

ORA-06512: a(z) „SCOTT.TRIGI”, helyen a(z) 21. sornál

ORA-04088: hiba a(z) 'SCOTT.TRIGI' trigger futása közben

TANAR	DIAK	ORASZAM
Nagy	Sas	21
Kis	Sas	17
Nagy	Szabó	21
Kis	Szabó	17
Molnár	Sas	22

TANAR	DIAK	ORASZAM
Molnár	Sas	22

## 6. lépés (A beírási kísérletek eredményének listázása)

```
-- 6-Listazas.sql
SELECT R2.diak, R1.tanar, R1.tantargy
FROM R1, R2
WHERE (R2.diak = 'Sas') AND
      (R1.tantargy = 'fizika') AND
      (R1.tanar = R2.tanar);
```

### Eredmény

DIAK	TANAR	TANTARGY
Sas	Kis	fizika

### Demonstráció

A 7-Tilt.sql program a fenti tárolt eljárás működését szemlélteti SQL\*Plus-környezetben. A 8-Megszoritas1.sql program sikertelen futtatása azt jelzi, hogy az elveszett függőség közvetlen megszorításba (CONSTRAINT ... CHECK) történő írása még tárolt eljárás alkalmazásával sem lehetséges, mivel másik tábla oszlopértékeire való hivatkozás így nem alkalmazható.

### Demonstráció-1:

```
-- 7-Tilt.sql
SET serveroutput ON
```

```
DECLARE
  v_tanar  VARCHAR2(30);
```

```
v_diak      VARCHAR2(30);
v_eredm    BOOLEAN;

FUNCTION Tilt(tanarnev IN CHAR, diaknev IN CHAR)
RETURN  BOOLEAN IS
  tanarszam  INTEGER;
BEGIN
  SELECT COUNT(kozostanar)
    INTO tanarszam
   FROM (((SELECT R1.tanar AS kozostanar
            FROM R1
           WHERE (R1.tantargy = (SELECT R1.tantargy
                                    FROM R1
                                   WHERE R1.tanar = tanarnev)) AND
                 (R1.tanar <> tanarnev))
          INTERSECT
         (SELECT R2.tanar AS kozostanar
          FROM R2
         WHERE R2.diak = diaknev)));
  DBMS_OUTPUT.PUT_Line ('közös tanárok száma =' || tanarszam);
  IF tanarszam = 0
  THEN RETURN TRUE;
  ELSE RETURN FALSE;
  END IF;
END;

BEGIN
  v_tanar := '&tanar';
  v_diak := '&diak';
  v_eredm := Tilt(v_tanar, v_diak);
  DBMS_OUTPUT.PUT_LINE(v_tanar || ' ' || v_diak);
  IF v_eredm
  THEN
    DBMS_OUTPUT.PUT_LINE ('>> Engedélyezett adatbevitel');
  ELSE
    DBMS_OUTPUT.PUT_LINE ('>> Tiltott adatbevitel');
  END IF;
END;
/
-- 7-Tilt.sql vége
```

## Eredmény

SQL> @7-Tilt

```
Adja meg a(z) tanar értékét: Tóth
régi 30: v_tanar := '&tanar';
új 30: v_tanar := 'Tóth';
Adja meg a(z) diak értékét: Sas
régi 31: v_diak := '&diak';
új 31: v_diak := 'Sas';
közös tanárok száma =1
Tóth Sas
>> Tiltott adatbevitel
```

A PL/SQL-eljárás sikeresen befejeződött.

## Demonstráció-2:

-- 8-Megszoritas1.sql

```
ALTER TABLE R2
ADD CHECK (Tilt(tanar, diak)=0);
```

-- 8-Megszoritas1.sql vége

## Eredmény

SQL> @8-Megszoritas1

```
ADD CHECK (Tilt(tanar, diak)=0)
```

\*

Hiba a(z) 2. sorban:

ORA-00904: „TILT”: érvénytelen azonosító



### Megjegyzés

Természetesen a triggerekkel a fenti példában szemléltetett többtáblás adatbeviteli korlátozáson kívül lehetőség van más jellegű felügyeletre is. Általában elmondhatjuk, hogy a triggerek segítségével megfogalmazhatók és futási időben érvényesíthetők adatbeviteli, módosítási és törlési

- többtáblás megszorítások (vagyis amelyek nem funkcionális függősségek), valamint
- táblán kívüli megszorítások (azaz nem függősség típusú – például az adatbevitel, módosítás vagy törlés időpontjára vonatkozó – logikai feltételek).

A triggerek egyik legfontosabb alkalmazása tehát abban nyilvánul meg, hogy egyes matematikailag nagyon nehezen megfogalmazható, az adatintegritást védő feladatokra egyszerű, gyakorlati megoldást kínál.

## 5. MELLÉKLET

# Gyakran használatos adatszótárnézetek

A napi munka során gyakran van szükségünk akár a munkakörnyezet, akár az éppen használt adatbázis-objektumok tulajdonságainak lekérdezésére. Ezeket az információkat az Oracle-rendszer a Sys, illetve a System tulajdonában lévő adatszótárban tárolja, mely különböző adat- és nézettáblákat tartalmaz. Az általános felhasználót érdeklő információk az adatszótár nézet-tábláin keresztül érhetők el.

E nézettáblák lekérdezéséhez általában elegendő a CREATE SESSION jogosultság (lásd 11. fejezet), mellyel minden felhasználó rendelkezik. A DBA\_ kezdetű nézettáblák lekérdezhetősége már korlátozott, de Sys vagy System felhasználóként ezekhez is könnyen hozzáférhetünk.

Megjegyezzük, hogy az alább bemutatásra kerülő nézetek általában már szerepeltek a példatárban. E melléklet célja, hogy ezeket a hasznos eszközököt egy helyen összegyűjtve segítse a minden nap munkát.

Az adatszótárbeli nézettáblák célszerűen csoportosíthatók a felmerülő igények alapján az alábbi módon:

- objektumok tulajdonságainak lekérdezése,
- a felhasználók és környezetük lekérdezése,
- a felhasználók és jogosultságaik lekérdezése.

## Táblaszerkezet lekérdezése

Az adatszótárbeli nézettáblák legfőbb erénye, hogy a teljes adatszótárból elhagyják mindeneket az információkat, amelyekre általában nincs szükségünk, ám még ezek is tartalmazhatnak számunkra éppen felesleges oszlopokat. Fontos ezért, hogy le tudjuk kérdezni a nézettáblák szerkezetét, hiszen a listájuk fejlécében lévő oszlopnevek sokszor csak csonkítva jelennek meg. Az adat- és nézettáblák szerkezetét a

DESC táblanév

SQL\*Plus-utasítással kérdezhetjük le. Ily módon kiválaszthatjuk a számunkra fontos oszlopokat, és csak ezeket listázzuk.

## Csatlakozás az adatbázishoz különböző felhasználóként

Az egyes nézettábla-listázások eredményei jelentős mértékben függnek a felhasználótól, aki a lekérdezéseket kiadja. Egy környezetre vonatkozó lekérdezés, amely valamely átlagos felhasználó esetén néhány sort ad, az a Sys vagy a System esetén akár több száz sort is eredményezhet. Az egyes (például jogosultságra vonatkozó) lekérdezések jelentése is nyilván más a rendszergazda (Sys) esetén, mint a felhasználó esetén.

Ha Sys rendszergazdaként akarunk vizsgálni, akkor

```
CONNECT sys/rendszergazda AS SYSDBA
```

módon, ha a System felhasználóként, akkor

```
CONNECT system/gazda
```

módon, ha a Scott felhasználóként, akkor

```
CONNECT scott/tiger
```

módon kell csatlakoznunk az adatbázishoz. A továbbiakban az átlagos felhasználóként való csatlakozást

```
CONNECT xxx/yyyy
```

módon jelöljük.

## Objektumok tulajdonságainak lekérdezése

### Az adatszótár és általános nézeteinek lekérdezése

-- A TELJES adatszótár lekérdezése:

-- (Ezt Sys-ként, vagy System-ként sose kérdezzük le!)

```
SELECT * FROM dictionary;
```

-- A felhasználói adatszótár „user\_objects” nézettáblája:

-- (Ezt Sys-ként, vagy System-ként sose kérdezzük le!)

```
SELECT * FROM user_objects;
```

-- A felhasználói adatszótár „user\_tables” nézettáblája:

```
SELECT * FROM user_tables;
```

-- A felhasználói adatszótár „user\_views” nézettáblája:

```
SELECT * FROM user_views;
```

## A felhasználói objektumok hasznos (szűkített) lekérdezései

```
-- A felhasználói adatszótár „user_catalog” nézettáblája:  
-- (Táblák és nézetek lekérdezésére igen hasznos)  
SELECT * FROM user_catalog;  
  
-- A felhasználói objektumainak és azok típusának lekérdezése:  
SELECT object_type, object_name  
  FROM user_objects  
 ORDER BY object_name;  
  
-- A felhasználói nézeteinek és azok tartalmának lekérdezése:  
COLUMN view_name FORMAT A15  
COLUMN text      FORMAT A60  
-- A felhasználói adatszótár „user_tables” nézettáblája:  
SELECT view_name, text  
  FROM user_views;  
CLEAR COLUMNS
```

## Megszorítások lekérdezése

```
CONNECT scott/tiger  
  
-- Az emp tábla megszorításainak lekérdezése:  
COLUMN owner          FORMAT A15  
COLUMN constraint_name FORMAT A16  
COLUMN constraint_type FORMAT A20  
COLUMN search_condition FORMAT A20  
SELECT owner, constraint_name, constraint_type, search_condition  
  FROM user_constraints  
 WHERE UPPER(table_name) = 'EMP';  
CLEAR COLUMNS  
  
-- A megszorítások nevének és oszlopainak lekérdezése:  
COLUMN owner          FORMAT A15  
COLUMN constraint_name FORMAT A16  
COLUMN table_name     FORMAT A12  
COLUMN column_name    FORMAT A20  
SELECT owner, constraint_name, table_name, column_name  
  FROM user_cons_columns  
 WHERE UPPER(table_name) = 'EMP';  
CLEAR COLUMNS
```

## PL/SQL-objektumok lekérdezése

```
-- Eljárások, függvények és triggerek lekérdezése:  
COLUMN object_name FORMAT A20  
SELECT object_name, object_type  
  FROM user_objects  
 WHERE object_type IN ('PROCEDURE', 'FUNCTION', 'TRIGGER');  
CLEAR COLUMNS  
  
-- A triggerek lekérdezése:  
COLUMN trigger_name      FORMAT A15  
COLUMN triggering_event   FORMAT A27  
COLUMN description        FORMAT A30  
SELECT trigger_name, triggering_event, description  
  FROM user_triggers;  
CLEAR COLUMNS  
  
-- A "Fizetés" tárolt függvény forráslistájának lekérdezése  
-- (Ezzel lekérdezhető bármely tárolt alprogram, trigger forráskódja)  
COLUMN text  FORMAT A50  
COLUMN line  FORMAT 9999  
SELECT line, text  
  FROM user_source  
 WHERE UPPER(type)='FUNCTION' AND  
       UPPER(name)='FIZETÉS';  
CLEAR COLUMNS
```

## A felhasználók és környezetük lekérdezése

### Az összes felhasználó lekérdezése

```
CONNECT xxx/yyyy  
-- vagy  
CONNECT scott/tiger  
-- vagy  
CONNECT sys/rendszerelő AS SYSDBA  
-- vagy  
CONNECT system/gazda  
SELECT * FROM all_users;
```

## Felhasználók rendszeradatainak lekérdezése

```

CONNECT sys/rendszer_gazda AS SYSDBA
-- vagy
CONNECT system/gazda
COLUMN username          FORMAT A15
COLUMN account_status    FORMAT A16
COLUMN default_tablespace FORMAT A20
COLUMN temporary_tablespace FORMAT A20
SELECT username, account_status, default_tablespace, temporary_tablespace
FROM dba_users;
CLEAR COLUMNS

```

## Felhasználó saját rendszeradatainak lekérdezése

```

CONNECT sys/rendszer_gazda AS SYSDBA
-- vagy
CONNECT system/gazda
-- vagy
CONNECT xxx/yyyy
-- vagy
CONNECT scott/tiger
COLUMN username          FORMAT A15
COLUMN account_status    FORMAT A16
COLUMN default_tablespace FORMAT A20
COLUMN temporary_tablespace FORMAT A20
SELECT username, account_status, default_tablespace, temporary_tablespace
FROM user_users;
CLEAR COLUMNS

```

## Felhasználók és táblaterületeik lekérdezése

```

CONNECT sys/rendszer_gazda AS SYSDBA
-- vagy
CONNECT system/gazda
SELECT username, default_tablespace
FROM dba_users
ORDER BY username;

```

## Felhasználó saját tábláinak és táblaterületeinek lekérdezése

```
CONNECT xxx/yyyy
-- vagy
CONNECT scott/tiger
COLUMN table_name      FORMAT A10
COLUMN tablespace_name FORMAT A15
SELECT table_name, tablespace_name
  FROM user_all_tables;
CLEAR COLUMNS
```

## A táblaterületek és azok állapotának lekérdezése

```
CONNECT sys/rendszerelő AS SYSDBA
-- vagy
CONNECT system/gazda
SELECT tablespace_name, status
  FROM dba_tablespaces;
```

## A táblaterületek tárolási helyének lekérdezése

```
CONNECT sys/rendszerelő AS SYSDBA
-- vagy
CONNECT system/gazda
COLUMN tablespace_name  FORMAT A15
COLUMN file_name        FORMAT A50
SELECT tablespace_name, file_name
  FROM dba_data_files;
CLEAR COLUMNS
```

## A felhasználó számára elérhető táblaterületek listája

```
CONNECT sys/rendszerelő AS SYSDBA
-- vagy
CONNECT system/gazda
-- vagy
CONNECT scott/tiger
COLUMN tablespace_name FORMAT A15
SELECT tablespace_name, status, contents, logging, force_logging,
       extent_management, allocation_type, segment_space_management
  FROM user tablespaces;
CLEAR COLUMNS
```

## Felhasználói táblák és azok táblaterületeinek lekérdezése

```
CONNECT xxx/yyyy
-- vagy
CONNECT scott/tiger
SELECT table_name, tablespace_name
  FROM user_tables;
```

## A felhasználók és jogosultságaik lekérdezése

### A Sys rendszeradminisztrátor jogosultságainak lekérdezése

```
CONNECT sys/rendszerelő AS SYSDBA
SET pagesize 150
SELECT *
  FROM user_sys_privs;
SET pagesize 40
```

### A System felhasználó jogosultságainak lekérdezése

```
CONNECT system/gazda
SELECT *
  FROM user_sys_privs;
```

### Felhasználó rendszerjogosultságainak lekérdezése

```
CONNECT xxx/yyyy
-- vagy
CONNECT scott/tiger
COLUMN username FORMAT A15
COLUMN privilege FORMAT A20
SELECT *
  FROM user_sys_privs;
CLEAR COLUMNS
```

### Felhasználó objektumkezelési jogosultságainak lekérdezése

```
CONNECT xxx/yyyy
-- vagy
CONNECT scott/tiger
COLUMN owner      FORMAT A8
COLUMN table_name FORMAT A10
COLUMN grantor    FORMAT A8
```

```
COLUMN privilege    FORMAT A9
COLUMN grantable   FORMAT A9
COLUMN hierarchy   FORMAT A9
SELECT *
  FROM user_tab_privs_recd;
CLEAR COLUMNS
```

# A CD-mellékletről

A könyvhöz tartozó CD-mellékletben alapvetően a példatár SQL és PL/SQL mintapéldái, feladatainak megoldásai (a Példák és Feladatok megoldásai nevű alkönyvtárban), és az Oracle-rendszer demonstrációs célokra szolgáló alaptábláit generáló szkript programok (az Adattáblák létrehozása nevű alkönyvtárban) találhatók. Ezek a merevlemezre való átmásolás után az SQL\*Plus-környezetben közvetlenül futtathatók, ám futtatásukhoz e környezet természetesen szükséges, vagyis feltételezzük, hogy a számítógépen telepítve van az Oracle legalább 9i változata (célszerűen a 9.2-es verzió).

Az elméleti összefoglalókban bemutatott mintapéldák SQL\*Plus szkript programjai az

A⟨fejezet kétjegyű sorszáma⟩-⟨példa kétjegyű sorszáma⟩.sql

nevű fájlokban, a feladatmegoldások szkript programjai pedig az

F⟨fejezet kétjegyű sorszáma⟩-⟨feladat kétjegyű sorszáma⟩.sql

nevű fájlokban, fejezetek szerint csoportosítva megtalálhatóak a CD-mellékleten. Tehát például az első fejezet első példáját tartalmazó programfájl neve A01-01.sql, az első fejezet első feladatának megoldását tartalmazó programfájl neve pedig F01-01.sql. Ha egy példának vagy feladatnak több megoldását is megadtuk, akkor arra az alul vonás ( ...) karakter után hivatkozunk (például A01-01\_1.sql jelenti az 1.1. példa 1. megoldását).

A CD egyes szkript programjainak legegyszerűbb futtatása, ha vágólapon keresztül bermásoljuk az SQL\*Plus-környezetbe a prompt jel után, ám ez nem minden működik. A legbiztosabb módszer (és ez minden működik), ha az SQL\*Plus-környezetben hozunk létre egy szkriptfájlt (például SQL> ed aa módon az aa.sql fájlt), ebbe másoljuk a megoldást, majd ezt futtattuk (SQL> @aa módon). (Részletesen lásd 3. melléklet.)

A CD-melléklet tartalmazza még a példatár összes mellékletét is pdf formátumban. Ezek olvasásához tehát szükséges az Acrobat Reader program, mely azonban (jogtiszta módon) ingyenesen letölthető az internetről. Ha valamelyik mellékletről egy programrészletet futtatni szeretnénk, akkor a „V” parancs kiadása után (az eszközsorban a T ikonra kattintva), majd az egérrel kijelölve a szükséges szöveget, a vágólapon keresztül bermásolhatjuk egy szkriptfájlba és futtathatjuk.

További információk találhatók a kiadó [www.panem.hu](http://www.panem.hu) honlapján a könyv ismertetésénél, de a [info@logana.com](mailto:info@logana.com) címen lehetőség van a szerzőkkel való közvetlen kapcsolatfelvételre is.

A könyv megírásakor és az egyes mintapéldák kidolgozásánál a szerzők a tőlük telhető legnagyobb gondossággal jártak el, azonban így is előfordulhatnak hibák. Köszönettel vesznek tehát minden észrevételt és javaslatot. Hisznek abban, hogy egy későbbi kiadásban ezek révén lehetőségük lesz még inkább segíteni az olvasókat tanulmányaiukban, munkájukban.

# Irodalomjegyzék

- [1] Gyenes László–Juhos Margit: Az SQL alapjai. Systrade Kft., 1990.
- [2] Juhász István–Almási Béla–Márton Ágnes–Balogh Judit: Oracle 6.0 Referencia Kézikönyv. Piremon, 1992.
- [3] Stolnicki Gyula: SQL kézikönyv. ComputerBooks, 1994.
- [4] Szelezsán János: Adatbázisok. LSI Oktatóközpont, 1997.
- [5] Bhamidipati, K.: SQL programozói referenciaikönyv. Panem, 1999.
- [6] Sunderraman, R.: Oracle programming: A primer. Addison-Wesley, 1998.
- [7] Ullmann, J. D.–Widom, J.: Adatbázis-rendszerök. (Alapvetés) Panem–Prentice-Hall, 1998.
- [8] Ensor, D.–Stevenson, I.: Oracle-tervezés. Kossuth Kiadó, 2000.
- [9] Gajdos Sándor: Adatbázisok. Műegyetemi Kiadó, 2000.
- [10] Abbey, M.–Corey, M. J.–Abramson, I.: Oracle 8i, Kézikönyv kezdőknek. Panem, 2001.
- [11] Adamo, J.: Data Mining for Association Rules and Sequential Patterns. Sequential and Paralell Algorithms. Panem, 2001.
- [12] Loney, K.–Koch, G.: Oracle 8i, Teljes referencia. Panem, 2001.
- [13] Adriaans, P.–Zantinge, D.: Adatbányászat. (Információ Technológia sorozat) Panem, 2002.
- [14] Celko, J.: SQL felsőfokon. Kiskapu Kiadó, 2002.
- [15] Gábor András–Juhász István: PL/SQL-programozás. Alkalmazásfejlesztés Oracle 9i-ben. Panem, 2002.
- [16] Kende Mária–Kotsis Domokos–Nagy István: Adatbázis-kezelés az Oracle-rendszerben. Panem, 2002.
- [17] Roland, F.: Adatbázis rendszerek. (Információ Technológia sorozat) Panem, 2002.
- [18] Celko, J.: SQL fejtörők. Kiskapu Kiadó, 2003.
- [19] Gábor András–Gunda Lénárd–Juhász István–Kollár Lajos–Mohai Gábor–Vágner Anikó: Az Oracle és a web. Haladó Oracle 9i ismeretek. Panem, 2003.
- [20] Gruber, M.: SQL A–Z. Kiskapu Kiadó, 2003.
- [21] Gultzan, P.–Pelzer, T.: SQL teljesítményfokozás. Kiskapu Kiadó, 2003.
- [22] Stolnicki Gyula: SQL programozóknak. Objektumok, relációk a gyakorlatban. ComputerBooks, 2003.
- [23] Bódi Bence: Az SQL példákon keresztül. Jedlik Oktatási Stúdió, 2004.
- [24] Han, J.–Kamber, M.: Adatbányászat. Koncepciók és technikák. Panem, 2004.
- [25] Theriault, M.–Carmichael, R.–Viscusi, J.: Oracle 9i adatbázisok adminisztrációja. Panem, 2004.

- [26] Oracle9i Database Documentation Release 2 (9.2.0.2.0) for Microsoft Windows, Viewable CD.  
Oracle9i Database Release 2 (9.2.0.1.0) CD Pack for Microsoft Windows, Release Date: APR-04.
- [27] <http://otn.oracle.com/software/products/oracle9i/content.html>
- [28] <http://www.oracle.com>
- [29] <http://www.oracle.hu>
- [30] <http://www.oracle.com/technology>
- [31] <http://metalink.oracle.com>
- [32] <http://www.orafaq.com>
- [33] <http://www.mydotweb.com/oracle-dba.html>
- [34] <http://www.sql.org>
- [35] <http://www.prog.hu>
- [36] <http://www.animare.hu>
- [37] <http://www.softwareonline.hu>
- [38] <http://www.kossuth.hu>
- [39] <http://www.oreilly.com>
- [40] <http://oracle.oreilly.com>
- [41] <http://www.kiskapu.hu>
- [42] <http://www.panem.hu>
- [43] <http://www.prentice-hall.com>
- [44] <http://www.celko.com>
- [45] <http://cgi.cs.indiana.edu/~oracle>
- [46] <http://www.ioug.org>
- [47] <http://oracle.ittoolbox.com>
- [48] <http://sirius.cs.ucdavis.edu/teaching/sqltutorial>
- [49] <http://www.usforacle.com>
- [50] <http://www.w3schools.com/sql/default.asp>
- [51] <http://www-db.stanford.edu/~ullman/fcdb/oracle/or-plsql.html>
- [52] <http://www.orafaq.com/faqplsql.htm>
- [53] <http://www.logana.com/ni>
- [54] <http://index.index.hu/cgi-bin/kereses.cgi?L=0&KERESES=oracle&LISTA=1>
- [55] <http://lazarus.elte.hu/~hzsolt/oracle>
- [56] <http://www.oracle.lap.hu>
- [57] <http://www.houg.hu>
- [58] <http://sql.lap.hu>
- [59] <http://www.dbasupport.com>
- [60] <http://webstore.ansi.org/ansidocstore/default.asp>



# Tárgymutató

- lásd kötőjel  
-- (SQL\*Plus megjegyzés jel) 74, 704,  
Példa: 4.1  
!= (nem-egyenlőség jel) 20, 66, Példa: 5.4  
„ (idézőjel) 20, 22, Példa: 1.1  
# (konstans karakterként) 32, Példa: 1.6  
\$9999 (formátummaszk) 37, 495  
% (százalékjel) lásd százalékjel,  
alsztringhatároló jel  
%FOUND 163  
%ISOPEN 163  
%NOTFOUND 163, 602, Példa: 9.1  
%ROWCOUNT 163  
%ROWTYPE 160, 178, 560, 573, 594  
%TYPE 160, 179, Példa: 9.1, 10.1  
& 71, 373, 705, 718, Példa: 4.1, 4.9  
&& 71, 373, 705  
( ) (allekérdezés határoló jele) 67, Példa:  
4.1, 4.6  
\* lásd csillag jel  
. lásd vessző  
. lásd pont  
... (szintaktikai jel) 18, 24  
.LST 245  
/ lásd ferde vonal  
/\* \*/ (PL/SQL megjegyzés jel) 185, 704,  
Példa: 10.3  
: lásd kettőspont  
:= (értékkadás jele) 146, 180, Példa: 8.1,  
10.1  
:NEW 189, Példa: 10.6  
:OLD 189, Példa: 10.6  
; 19, 143, 145, Példa: 1.1, 8.1  
@ 75, 86, 719, Példa: 4.9  
@@ 75, 85, Példa: 4.9

[ ] (szintaktikai jel) 18-20, 27  
\_ lásd aláhúzás jel  
{ } (szintaktikai jel) 18, 27  
| lásd függőleges vonal  
|| (összefűzés jel) 27, 151, Példa: 1.6, 8.1,  
10.3  
. lásd aposztróf  
. ' lásd szóköz  
+ lásd összedás jel, összekapcsolás jel  
< (kisebb jel) 20, 66  
<= (kisebb-egyenlő jel) 20, 66, 102, 113,  
Példa: 6.1  
<> (nem-egyenlőség jel) 20, 66, 370  
= (egyenlőség jel) 20, 66, Példa: 1.1  
> (nagyobb jel) 20, 66, Példa: 2.2, 8.1, 12.3  
>= (nagyobb-egyenlő jel) 20, 66, 102  
0999.999 (formátummaszk) 37, Példa: 1.8  
09999 (formátummaszk) 37, 512  
-1 25, 26, Példa: 1.2  
9999000 (formátummaszk) 37, 513  
99999 (formátummaszk) 37

## A

ablak-tag 267  
ACCEPT 72, 74, 705, Példa: 4.1, 4.8  
adatbázis adminisztrátor 204  
adatformázás 34, Példa: 1.8  
adatkapcsolat 240  
adatszótár nézetek 97, 107, 183, 207-  
208, 416, 449, 700, 720  
adattípus 23, 72, 92-93, 144-145, 159,  
161, 170, 178-179, 182, 313, 556, 591,  
706, lásd még DESC

ADD 93, 96, Példa: 5.3  
 AFTER 187  
 aggregáló rang függvények 270  
 aktuális dátum 37, Példa: 1.9  
 aláhúzás jel (\_)  
     dátumjel 36  
     névbeli karakterként 83, Példa: 4.7  
 ALL 20, 66, 74, 76, 206-207, 282, 723  
 ALL TRIGGERS 190  
 all\_users 723  
 allekérdezés 20, 56, 66  
 alprogram 177  
     tárolt ~ 181  
 alsztringhatároló jel (LIKE % %) 20, 67,  
     311, 313  
 alsztringkeresés *lásd* LIKE  
 ALTER ANY TABLE 205, Példa: 11.8  
 ALTER ANY TRIGGER 205  
 ALTER SESSION 37, 98, 708, Példa: 1.10,  
     5.1  
 ALTER TABLE 93, 96-97, 202, Példa: 5.2  
 ALTER TABLESPACE 209, Példa: 11.8  
 ALTER TRIGGER 190, 202  
 ALTER USER 219, 225, Példa: 11.8  
 analitikus függvények 253, 266  
 analitikus záradék 267  
 AND 20, Példa: 4.8, 13.15  
 ANY 20, 66  
 aposztróf ('') 22, 71, 73, Példa: 1.1  
 aranymetszés 45, Példa: 1.14  
 AS 20, 58, 92, 111, 181, 212, 699-700  
     Példa: 1.1, 3.1, 11.1  
 ASC (növekvően) 21, Példa: 2.2, 13.14  
 átnevezés *lásd* RENAME  
 AVG 50, 282, Példa: 2.1, 2.3

**B**

batch 238  
 BEFORE 187, Példa: 10.4, 10.7  
 BEGIN 143, 178, 187, Példa: 8.1  
 belső összekapcsolás 57  
 beszúrás *lásd* INSERT  
 BETWEEN 20, 67, 268-269, Példa: 4.8,  
     13.15  
 BIN 236  
 BINARY\_INTEGER 145, 572

BLOB 145  
 blokk 142, Példa: 8.1  
 BOOLEAN 145, Példa: 10.2  
 BREAK 75, 78, 374, Példa: 4.5  
 BREAKS 79, Példa: 4.5  
 BTITLE 75, 78, Példa: 4.4

**C, CS**

C 105, 107, Példa: 5.3, 5.4, *lásd még*  
     CHECK, NULL, megszorítás  
 CASCADE 95-97, 211, 418, Példa: 11.8  
 CASE kifejezés 24, 256, 319, Példa: 1.2,  
     13.1  
 cat 700  
 CD-melléklet 15-17, 98, 100, 244, 300,  
     693, 728, Példa: 5.1, 5.3, 12.5  
 CENTER 78, 495, Példa: 4.4  
 CHAR 71-72, 92, 144-145, 178  
 CHECK 96, 423-424, 426, Példa: 5.3  
 CHR 27, 185, Példa: 10.3  
 CHR(10) a sorémelés (LF) kódja 27,  
     185, Példa: 10.3  
 CHR(32) a szóköz (SPACE) kódja 27,  
     185, Példa: 10.3  
 CHR(9) a tabulátor (TAB) kódja 27,  
     185, 245, Példa: 10.3, 12.7  
 ciklus 150  
 ciklusváltozó 150  
 CLEAR 75, 78, Példa: 1.1, 4.3  
 CLEAR BREAKS 79, Példa: 4.4  
 CLEAR COLUMNS 23, 79, Példa: 1.1, 4.4  
 CLEAR SCREEN 79  
 CLOB 145  
 CLOSE 161, Példa: 9.1  
 cmd 236, Példa: 12.1  
 COLUMN 22, 75, 78, 93, Példa: 1.1, 4.3  
 COLUMNS 79, Példa: 1.1  
 COMMIT 91, 164, 186, Példa: 7.M1  
 CONCAT 27, 151  
 CONNECT 17, 211, 700, 721, Példa: 11.1  
 CONNECT BY 63, 658, Példa: 3.2, 4.7, 14.2  
 CONSTRAINT 94, *lásd még* megszorítás  
 COUNT 50, 282, 572, Példa: 2.3  
 COUNT metódus (a TABLE típushoz) 573  
 CREATE ANY TABLE 208, Példa: 11.8  
 CREATE ANY VIEW 218, 225, Példa: 11.8

- CREATE FUNCTION 205  
 CREATE OR REPLACE FUNCTION 182, Példa: 10.2  
 CREATE OR REPLACE PROCEDURE 181, Példa: 10.3  
 CREATE OR REPLACE TRIGGER 187, Példa: 10.4  
 CREATE OR REPLACE VIEW 111, Példa: 6.1  
 CREATE PROCEDURE 205  
 CREATE ROLE 211, 644, Példa: 11.9  
 CREATE SESSION 205, 644, 720, Példa: 11.8  
 CREATE TABLE 58, 92, 205, 706, Példa: 3.1, 5.1, 5.3  
 CREATE TABLESPACE 208, Példa: 11.8, 12.5  
 CREATE TRIGGER 187  
 CREATE USER 205, 210, Példa: 11.8, 12.5  
 CREATE VIEW 111  
 CUBE operátor 257, Példa: 13.4  
 CURRENT OF 164, Példa: 9.1  
 CURRENT ROW 269  
 CURSOR 162, Példa: 9.1  
 csillag jel (\*)  
   oszlojel 18, 20, 21, 50, Példa: 2.3, 3.1, 6.1  
   szorzásjel 44, Példa: 1.13  
   Joker karakter (DOS) Példa: 12.1-12.3  
 csomag 151, 155  
 csoportfüggvény 50, 253  
 csoportképzés, csoportosítás 50, 253  
 csoportképző attribútum 50
- D**
- D (formátummaszk) 36  
 DATAFILE 208, Példa: 11.8  
 DATE 72, 92, 145, Példa: 5.1  
 dátumformátum 35, 37, 309, *lásd még*  
   formátummaszk  
 dátum-idő 43  
 dátumjel (/ , - : ...) 36, Példa: 1.9  
 DAY (formátummaszk) 36  
 DBA 204  
 dba\_data\_files 208, 240, 725, Példa: 11.5  
 dba\_tablespaces 208, 725, Példa: 11.8  
 dba\_users 212, 724, Példa: 11.2, 11.8
- DBMS csomag 151  
 DBMS\_OUTPUT csomag 151  
 DBMS\_OUTPUT.PUT\_LINE 146, 151, 182, 188, Példa: 8.1  
 DBMS\_RANDOM csomag 155  
 DBMS\_RANDOM.VALUE 155 Példa: 8.4  
 DCL 19, 204  
 DD (formátummaszk) 36, Példa: 1.9  
 DDD (formátummaszk) 36  
 DDL utasítás 19, 92  
 DECLARE 143, 178, 187, Példa: 8.1, 10.6  
 DECODE függvény 24, 256, 319-320, Példa: 1.2, 13.1  
 DEFAULT 92, 178, Példa: 11.8  
 default karakterkészlet 26, 79, 1. melléklet  
 DEFAULT TABLESPACE 210, Példa: 11.8  
 DEFINE 71, 75, 76, 705, Példa: 4.9, 8.1  
 DEFINE\_EDITOR 239, 705, Példa: 12.4  
 del 239, Példa: 12.3  
 DELETE 91, 187, 189, Példa: 5.2, 10.4  
 DELETING 191, Példa: 10.5  
 demo programok  
   DEMO-1.sql 98, Példa: 5.1  
   DEMO-10.sql 101, 644 Példa: 5.3  
   DEMO-11.sql 101, Példa: 5.3  
   DEMO-12.sql 101, Példa: 5.3  
 DEMO vállalat 297, 656  
 DEMO vállalat 253, 297  
 DENSE\_RANK függvény 269-270, Példa: 13.8-13.14  
 dept tábla 58, 696, Példa: 3.1, 5.1  
 DESC (csökkenően) 21, 232, 268, Példa: 1.1, 13.8  
 DESCRIBE, DESC 75, 183, 213, 706, 720, Példa: 5.3, 11.3  
 dictionary 721  
 dir 236, 239, Példa: 12.1  
 DISABLE 97, 190, Példa: 5.2  
 DISCONNECT 211  
 DISTINCT 20, 50, 282  
 DML 19, 90  
 DOS 235  
 DQL 19  
 DROP ANY TABLE 205, Példa: 11.8  
 DROP ANY VIEW 219, Példa: 11.8  
 DROP COLUMN 93  
 DROP CONSTRAINT 96  
 DROP FUNCTION 182

DROP PROCEDURE 182  
 DROP ROLE 211, 650, Példa: 11.9  
 DROP TABLE 58, 93, 708, Példa: 3.1, 11.9  
 DROP TABLESPACE 209, Példa: 11.8  
 DROP TRIGGER 189, Példa: 10.6  
 DROP USER 205, 211, 653, Példa: 11.8,  
 11.9  
 DROP VIEW 112, Példa: 6.1  
 dual 29, 38, 45, 155, 707, Példa: 1.3, 1.9,  
 1.14, 8.4  
 DY (formátummaszk) 36

**E, É**

echo 76, 245, Példa: 12.7  
 ed 75, 702  
 egyedi kulcs (UNIQUE) 95, Példa: 5.1-  
 5.4, lásd még megszorítás  
 egyértékű allekérdezés 21, 67  
 elemzőfüggvények 253  
 eljárás 178  
*ellenőrzött adatbevitel* (nézeten keresztül)  
 451  
 előjel (-) 30, Példa: 1.2  
 ELSE 24, 149, Példa: 1.2, 8.1  
 ELSIF 149, Példa: 10.5  
 elsődleges kulcs (PRIMARY KEY) 95,  
 Példa: 5.1-5.4, lásd még megszorítás  
 eltelt idő 43  
 emp tábla 22, 694, Példa: 1.1, 5.1  
 ENABLE 97, 190, Példa: 5.2  
 END 24, 143, 178, 187, Példa: 1.2, 8.1,  
 10.1  
 END IF 149, Példa: 8.1, 10.6  
 END LOOP 150, Példa: 9.1  
 értékkadás jele (:=) 153, 185, Példa:  
 8.1  
 érzékenységvizsgálat 249, lásd még  
 RANK, DENSE\_RANK és PERCENT\_RANK  
 függvény  
 Euklideszi algoritmus: 553  
 Excel (MS-Excel) 235, 248  
 EXCEPTION 143, 176-178, Példa: 10.1,  
 10.6  
 EXCEPTION\_INIT 177  
 EXECUTE 184-185, 206, Példa: 10.3  
 EXISTS 20, 66  
 EXIT 150, Példa: 9.1

exit (kilépés az SQL\*Plus-ból) 75, 703  
 EXIT WHEN 150, 163, Példa: 9.1  
 exp.exe 243, Példa: 12.6  
 explicit kurzor 161, 561, Példa: 9.1  
 exportálás 243  
 ezres csoportosítás jele (,) 37, 495

**F**

FALSE 24  
 feedback 76, Példa: 4.2, 5.3  
 fejléc 22, 45, lásd még COLUMN  
 felhasználói változó 71  
 felső-N analízis (Top-N) 113, 447, Példa:  
 6.1  
 ferde vonal (/)  
 dátumjel 36, Példa: 1.9  
 futtató jel (PL/SQL blokk) 143, Példa: 8.1  
 jelszó elválasztó jel (CONNECT) 17, Példa:  
 11.1  
 osztásjel 31, Példa: 1.6  
 web-cím elválasztó jel 731  
 FETCH 161, 561, Példa: 9.1  
 Fibonacci-számsor 45, Példa: 1.14  
 FIRST metódus (a TABLE típushoz) 573-574  
 FIRST\_VALUE 288, Példa: 13.20  
 fizikai táblanév 20  
 FLOAT 145  
 FOLLOWING 268, Példa: 13.17  
 FOR EACH ROW 187, Példa: 10.7  
 FOR UPDATE 164, Példa: 9.1  
 FORCE 111  
 FOR-ciklus 150, 162  
 FOREIGN KEY 95, Példa: 5.1-5.3, lásd még  
 megszorítás  
 FORMAT 22, 72, 78, Példa: 1.1, 4.3  
 formátummaszk 36-37, 72, Példa: 1.9  
 formázási utasítások 78  
 formázott kiírás 35, 45, 352  
 FROM 19, 56, 66, 206, Példa: 1.1, 5.2, 8.3  
 FULL 57, Példa: 3.1  
 FUNCTION 178  
 futtató jel (/) 143, Példa: 8.1  
 függőleges vonal (|)  
 szintaktikai jel 18, 27, 79, Példa: 4.4,  
 12.3  
 tördelő jel 79, Példa: 4.4  
 függvény 178

**G, GY**

gazdanyelv (Host Language) 143  
 grafikus megjelenítés 30  
 GRANT 205-206, Példa: 11.8  
 GROUP BY 19, 21, 49, 253, Példa: 2.1, 13.1  
 GROUP BY CUBE 257, Példa: 13.4  
 GROUP BY ROLLUP 253, Példa: 13.1  
 GROUPING indikátorfüggvény 260, Példa:  
 13.5  
 GROUPING SETS függvény 265, Példa: 13.7  
 GROUPING\_ID indikátorfüggvény 263,  
 Példa: 13.6  
 gyűjtőtábla típus 160, 560

**H**

halmaz 66  
 halmazművelet 66  
 hasonlító műveletjelek 20  
 hatványozás (POWER) 555  
 HAVING 19, 21, 49, Példa: 2.2  
 heading 77, 245, Példa: 12.7  
 HEADING 22, 78, Példa: 1.1, 4.4  
 HELP 75, Példa: 4.2  
 helyettesítő változó 71  
 helytakarékos listázás 30, Példa: 1.5  
 HH (formátummaszk) 36  
 HH12 (formátummaszk) 36  
 HH24 (formátummaszk) 36, Példa: 1.9  
 hibakód 177, 182  
 HIDE 72  
 hierarchia lekérdezés 63  
 hierarchikus adatszerkezet 62  
 hisztogram függvények 292  
 hivatkozási adattípusok 159  
 HOST 75, 235, Példa: 12.1, 12.2  
 Host Language (gazdanyelv) 143  
 hozzáférési korlátozás 204

**I**

idegen kulcs (FOREIGN KEY) 96, Példa:  
 5.1-5.4, *lásd még megszorítás*  
 IDENTIFIED BY 210-211, Példa: 11.8  
 idézőjel („) 20, 22, Példa: 1.1  
 IF 149

igazítás (JUSTIFY) 78, 495, Példa: 4.4  
 implicit kurzor 161-163  
 importálás 243  
 IN 20, 34, 66, 150, 163, 178, 573-574  
 Példa: 1.8, 10.1  
 IN\_OUT 178  
 INCLUDING CONTENTS 209  
 index 150  
 INDEX 75, 206, Példa: 4.2  
 INDEX BY BINARY\_INTEGER 160  
 inicializált változó *lásd DEFAULT*  
 INITCAP 27, 315  
 inline nézet 56, 113  
 INNER 57, Példa: 3.1  
 inp.exe 243  
 INSERT 90, 187, 189, 707, Példa: 5.1,  
 10.4, 10.6  
 INSERTING 191, Példa: 10.5  
 INSTEAD OF 187, Példa: 10.6  
 INSTR 27, Példa: 1.4  
 INTEGER 145, 178  
 integritási megszorítás 94, *lásd még*  
*megszorítás*  
 interaktív környezet 71  
 INTERSECT 66, 347  
 INTO 90, 149, 161, Példa: 5.1, 8.3, 9.1  
 IS 20, 160, 162, 178, 181, Példa: 9.1,  
 10.1, 10.2, 10.6  
 IS NULL 20, Példa: 10.6

**J**

jelszó elválasztó jel (/) 17, Példa: 11.1  
 jog, jogosultság 204  
 JOIN 57, 344, Példa: 3.1  
 JUSTIFY 78, 495, Példa: 4.4

**K**

K (KByte) 208, 210  
 karakterkezelő függvények 26, 27, Példa:  
 1.3-1.8  
 kettőspont (:)  
 dátumjel 36, Példa: 1.9  
 konstans karakter 32, Példa: 1.6  
 környezeti változó 144, Példa: 8.1

korrelációs név (:NEW, :OLD) 189,  
Példa: 10.6  
kiterjesztés jele fájlban (.) 21, 98, 238, 728,  
Példa: 5.1, 5.3  
kitöltő karakter 27  
kivételez 143, 176  
kivonásjel (-) 40, Példa: 1.11  
konstans oszlop 30  
konverziós függvények 37, 426  
korrelációs név (:NEW, :OLD) 189, Példa:  
10.6  
korrelált lekérdezés 62  
környezeti változó 144, Példa: 8.1  
kötőjel (-)  
    dátumjel 36, 37, Példa: 1.9  
    előjel 30, Példa: 1.2  
    kivonásjel 40, Példa: 1.11  
    sorfolytató jel 79, 244, 495, 704, Példa:  
    12.6  
    szövegkötőjel 15  
kötőjel (-) 36, Példa: 1.10  
kulcs lásd megszorítások  
kumulált összegképzés Példa: 13.14, 13.15  
kurzor 161  
külső összekapcsolás 57, 343  
külső összekapcsolás jele (+) 57, 343

**L**

L999 (formátummaszk) 37  
LAST metódus (a TABLE típushoz) 573-574  
LAST\_VALUE 288, Példa: 13.20  
LEFT 57, 78, Példa: 3.1  
LENGTH 27, 315  
létrehozás lásd még CREATE  
    eljárás ~ 181  
    felhasználó ~ 210  
    függvény ~ 182  
    nézet ~ 111  
    szerepkör ~ 211  
    tábla ~ 92, 706  
    táblaterület ~ 208  
    trigger ~ 187  
LEVEL 63, 658  
LIKE 20, 67, 311, 313  
linesize 77, Példa: 5.3  
lista elválasztó jele (.) 19, 21, Példa: 1.1  
LOB 145

logikai műveletjelek 20  
logikai táblanév 20  
LogikaiOszlopkifejezés 20  
login.sql 236, 702  
LONG 145  
LOOP-ciklus 150, Példa: 9.1  
LOWER 27, 103, 401, Példa: 5.3  
LPAD 27, 151, 333, Példa: 1.7, 1.8

**M**

M (MByte) 208, 210, Példa: 11.8  
maradéképzés lásd MOD  
másodlagos táblanév 20, 56  
MásodlagosOszlopnév 20  
másodperc 43  
MAX 50, 282, Példa: 2.3, 13.20  
megjegyzés jel  
    -- (SQL\*Plus megjegyzés jel) 74, 704,  
    Példa: 4.1  
    REM (SQL\*Plus megjegyzés jel) 704  
    /\* \*/ (PL/SQL megjegyzés jel) 185, 704,  
    Példa: 10.3  
megszorítás (CONSTRAINT) 94, Példa: 5.1-  
5.4  
    egyedi kulcs ~ (UNIQUE) 95, 415  
    elsődleges kulcs ~ (PRIMARY KEY)  
    95, 417-418  
    feltételes ~ (CHECK) 96, 423-424,  
    426  
    hivatkozási ~ (REFERENCES) 95, 418  
    idegen kulcs ~ (FOREIGN KEY) 95,  
    418  
    NULL ~, NOT NULL ~ 95  
    oszlop~ 92, 94  
    tábla~ 92, 94  
    ~ engedélyezése 97  
    ~ felfüggesztése (tiltása) 97  
    ~ hozzáadása 96  
    ~ törlése 97  
MI (formátummaszk) 36, Példa: 1.9  
mi lenne ha... lásd  
    érzékenységvizsgálat  
MIN 50, 282, Példa: 2.3, 13.20  
minősített név jele (.) 21, 56, 92, 159-160,  
    164, 189, 222, Példa: 3.2, 11.8  
MINUS 66  
MM (formátummaszk) 36, Példa: 1.9

MOD 327, 555, Példa: 8.2  
 MODIFY 93, Példa: 5.3  
 módosítás lásd UPDATE  
 MON (formátummaszk) 36  
 mon (formátummaszk) 36  
 MONTH (formátummaszk) 36  
 MONTHS\_BETWEEN 43, Példa: 1.12

**N, NY**

nem-egyenlőség jel (<>, !=) 20, 66, Példa: 5.4  
 NEW 188, Példa: 10.7  
 newpage 76, 245, Példa: 12.7  
 NEXT metódus (a TABLE típushoz) 573-574  
 nézet, nézettábla 111  
 NLS\_DATE\_FORMAT 37, 708, Példa: 1.10, 5.1  
 NLS\_DATE\_LANGUAGE 97, Példa: 5.1  
 NO\_DATA\_FOUND 177, Példa: 10.1  
 NOFORCE 111  
 NOPRINT 78, Példa: 4.3, 4.4  
 NOT 20, Példa: 5.1, 5.3  
 NOWAIT 164, Példa: 9.1  
 NTILE függvény 294, Példa: 13.25  
 NULL 20, 23, 50, 94, 145, 253, 306, Példa: 5.1-5.3, lásd még megszorítás  
 NUMBER 72, 92, 144-145, Példa: 5.1, 8.1, 8.3  
 numwidth 76, Példa: 4.1, 5.3  
 NVL függvény 23, 306, Példa: 1.2, 13.1

**O**

OF 160, 164, 187  
 OFF 76, 245, Példa: 4.3  
 OFFLINE 209, Példa: 11.8  
 OLD 188, Példa: 10.7  
 ON 57, 76, 78, 95, 187, 210, Példa: 4.3, 4.5, 10.4, 11.8  
 ON DELETE CASCADE 95  
 ONLINE 209, Példa: 11.8  
 OPEN 161, Példa: 9.1  
 OR 20, 187, Példa: 5.4, 10.4  
 oradata 240, Példa: 12.5  
 ORDER BY 19, 270-271, Példa: 1.1, 13.8, 13.25

oszlopkifejezés 20, 50  
 oszlopmegszorítás 92, 94  
 osztásjel (/) 31, Példa: 1.6  
 OUT 178, 245, Példa: 10.1  
 OVER 267, Példa: 13.8, 13.18, 13.25

**Ö**

összeadás jel (+) 21, 31, Példa: 1.6, 2.2  
 összekapcsolás (táblák ~a) 56  
 belső (INNER) 57, Példa: 3.1  
 külső (LEFT, RIGHT, FULL) 57, Példa: 3.1  
 külső (+) 57, 343  
 összekapcsolás jel (+) 57, 343  
 összetett adattípus 159

**P**

P 105, Példa: 5.3, lásd még PRIMARY KEY, megszorítás  
 pagesize 76, Példa: 11.6  
 paraméter 178  
 paraméterelválasztó jel (.) 21, 23, 24, Példa: 1.2  
 partíció 268  
 partíció-tag 267  
 PARTITION BY 267, Példa: 13.9, 13.19  
 PERCENT\_RANK függvény 269-270, Példa: 13.8  
 PL/SQL blokk 142, Példa: 8.1  
 PL/SQL blokk futtató jel (/) 143, Példa: 8.1  
 PL/SQL megjegyzés jel /\* \*/ 185, 704, Példa: 10.3  
 PL/SQL nyelv 15, 142  
 pont (.) 72-73, 87  
 dátumjel 36, Példa: 1.9  
 kiterjesztés jele fájlban 21, 98, 238, 728, Példa: 5.1, 5.3  
 minősített név jele 21, 56, 92, 159-160, 164, 189, 222, Példa: 3.2, 11.8  
 tizedes pont 34-35, 37, Példa: 1.8  
 web-cím elválasztó jel 4, 729, 731  
 POWER lásd hatványozás  
 POWER (hatványozás) 555  
 pragma direktíva 177  
 PRECEDING 268, Példa: 13.15

PRIMARY KEY. 95, Példa: 5.1-5.3, lásd még megszorítás  
 PRINT 75, 78, 145, Példa: 4.3, 4.4, 8.1  
 PRIOR 63, 657-658, Példa: 3.2, 4.7  
 privilegium (jogosultság) 204  
 PROCEDURE 178, Példa: 10.1  
 programkapcsolat 235  
 PROMPT 51, 72, 75, 703, Példa: 2.3, 4.1,  
     4.8, 5.3  
 pszeudóoszlop 24, 63  
 PUBLIC 205

**Q**

quit (kilépés az SQL\*Plus-ból) 75, 703  
 QUOTA 210, Példa: 11.8

**R**

R 105, Példa: 5.3, lásd még FOREIGN KEY,  
     REFERENCES, megszorítás  
 RAISE 176, Példa: 10.6  
 RAISE\_APPLICATION\_ERROR 182, 188,  
     Példa: 10.4  
 rang függvények 269, lásd még RANK,  
     DENSE\_RANK és PERCENT\_RANK  
     függvények  
 RANGE 269, Példa: 13.15  
 RANGE BETWEEN 269, Példa: 13.15  
 RANK függvény 269-270, Példa: 13.8-13.14  
 RATIO\_TO\_REPORT függvény 285-286,  
     Példa: 13.18  
 READ 111, 206  
 REAL 145, 178  
 RECORD típus 160  
 REFERENCES 95, Példa: 5.1-5.3, lásd még  
     megtörés  
 rejtett kurzor 162, 558  
 rekord adattípus 160  
 REM (SQL\*Plus megjegyzés jel) 704  
 RENAME 93  
 rendezés (ORDER BY) 19, Példa: 1.1  
 rendező-tag 267  
 rendszerdátum (sysdate) 23, 35, 37, 44,  
     707, Példa: 1.9, 10.4  
 rendszerkivétel 177  
 rendszerszintű védelem 204

rendszer változó 74  
 REPLACE lásd CREATE OR REPLACE  
 részcsoporthálózat 254  
 részletező csoportosítások 253  
 részösszeg 258  
 RETURN 178, 182, Példa: 10.2  
 REUSE 208, Példa: 11.8  
 REVERSE 150  
 REVOKE 206-207, 649, Példa: 11.8  
 RIGHT 57, 78, 495, Példa: 3.1, 4.4  
 ROLE (szerepkör) 211  
 role\_sys\_privs 207  
 role\_tab\_privs 207  
 ROLLBACK 91, 186, Példa: 7.M1, 10.6  
 ROLLUP operátor 253, Példa: 13.1  
 ROUND 31, 42, 50, 321, Példa: 1.6, 1.12  
 ROWID 164  
 ROWNUM (felső-N analízis) 113, 447, Példa:  
     6.1  
 ROWS BETWEEN 268  
 RPAD 27, 151, 333, Példa: 1.6-1.8

**S, SZ**

S9999 (formátummaszk) 37  
 salgrade tábla 696-697  
 save 75  
 SAVEPOINT 91, 186, 406, Példa: 7.M1  
 Scott 212, 694, 700, Példa: 11.1  
 SCREEN 79  
 SELECT utasítás (PL/SQL) 149, Példa: 8.3  
 SELECT utasítás (SQL) 19, 162, 700, Példa:  
     1.1, 3.1  
 serveroutput 76, 151, Példa: 8.1  
 SESSION 37  
 SET 37, 58, 76, 91, 151, 701, Példa: 4.1,  
     4.2, 5.3  
 SHOW 74, 75, 76, 701, Példa: 4.2  
 SHOW ALL 74  
 SHOW ERRORS 181, 190, 617, Példa: 10.2,  
     10.4  
 SIZE 208, Példa: 11.8  
 soraemelés (LF) 27  
 sorfolytató jel (-) 79, 244, 495, 704, Példa:  
     12.6  
 sorfüggvények, egysoros függvények 21, 24  
 sort.exe 239, Példa: 12.3  
 SOUNDEX 346

- space 76, 245, Példa: 12.7  
SPPOOL 75, 244, Példa: 12.7  
SQL nyelv 15, 19, 142, 698  
SQL%FOUND 163  
SQL%ISOPEN 163  
SQL%NOTFOUND 163  
SQL%ROWCOUNT 163  
SQL\*Loader 244  
SQL\*Plus környezet 22, 71, 74, 698  
SQL\*Plus megjegyzés jel (--) 74, 704, Példa: 4.1  
SQL\*Plus megjegyzés jel (REM) 704  
SQL> (az SQL\*Plus prompt jele) 700, Példa: 4.2, 4.9  
SQLCODE 177  
SQLEERRM 177  
sqlldr.exe 244  
SS (formátummaszk) 36, Példa: 1.9  
SSSSS (formátummaszk) 36  
START WITH 63, 658, Példa: 3.2, 4.7  
statisztikai függvényeknek 50, 282  
STDDEV 50, 282, Példa: 2.1  
STRING 145  
SUBSTR 27, 334, Példa: 1.3, 1.5, 1.7  
SUM 50, 282, Példa: 13.14, 13.15  
Sys 212, 699, 720, Példa: 11.1  
sysdate 23, 35, 37, 44, 707, Példa: 1.9, 10.4  
SYSDBA 212, 700, Példa: 11.1  
SYSOPER 700  
System 212, 699, 720, Példa: 11.1  
SYSTEM táblaterület 210  
SYSTEM01.DBF 243, Példa: 12.5  
számformátumok 35  
százalékjel (%) 46, 109  
SzelekciósLista 19, 67  
szélsőérték függvények 288  
szerepkör (ROLE) 207, 211  
szintaktikus jelölés 18  
szkript program 85, 143, 702, Példa: 8.1  
szóköz jel (SPACE) 20, 27, 33, 36  
szorzásjel (\*) 44, Példa: 1.13  
szövegkeresés 27
- T
- tábla exportálása, importálása 243  
táblák összekapcsolása 56
- TáblaLista 20  
táblamegszorítás 92, 94  
táblaterület 207  
TABLE típus (gyűjtőtábla típus) 160, 560, 572  
TABLESPACE 208, Példa: 11.8  
tabulálás (TAB) 27  
tárolt alprogram 181, 617  
  ~ futtatása (EXECUTE) 184, 240  
  ~ lekérdezése 183, Példa: 10.2  
telepítés (Personal Oracle) 693  
temp táblaterület 218, 225, Példa: 11.8  
TEMPORARY TABLESPACE 210, Példa: 11.8  
termout 77, 236, Példa: 12.3  
THEN 24, 149, 177, Példa: 1.2, 8.1, 10.1  
tiger 212, 694  
time 77  
timing 75, 77  
tizedes pont(.) 34-35, 37, Példa: 1.8  
TO 91, 93, 205, Példa: 11.8  
TO\_CHAR 25, 35, 45, Példa: 1.2, 1.9, 1.14, 10.3  
TO\_DATE 35, Példa: 1.10, 4.8  
TO\_NUMBER 37, Példa: 1.11  
Too\_Many\_Rows 177  
többtáblás lekérdezések 56  
törölés lásd még DELETE  
  sor ~ 91  
törölés lásd még DROP  
  eljárás ~ 182  
  felhasználó ~ 211  
  függvény ~ 182  
  megszorítás ~ 96  
  nézet ~ 112  
  szerepkör ~ 211  
  tábla ~ 93  
  táblaterület ~ 209  
  trigger ~ 189, 198  
  tranzakció 91  
  trigger 186, 622, 636, 643, Példa: 10.4-10.7  
  TRUE 24  
  TRUNC 330  
  TTITLE 75, 78, Példa: 4.5  
  tulajdonos 21, 56, 204, Példa: 11.8  
  TYPE 159-160  
  type 238, Példa: 12.3

## U, Ü

U 105, Példa: 5.3, lásd még UNIQUE, megszorítás  
 UNBOUNDED FOLLOWING 269  
 UNBOUNDED PRECEDING 269, Példa: 13.15  
 UNDEFINE 71, 75, Példa: 6.1, 8.1  
 Unicode (UTF8) karakterkészlet 26, 79, 1.  
 melléklet  
 UNION 66  
 UNIQUE 95, Példa: 5.1-5.4, lásd még  
 megszorítás  
 UNLIMITED 210, Példa: 11.8  
 UPDATE 58, 91, 164, 189, 471, Példa: 5.3,  
 9.1, 10.4  
 UPDATING 191, Példa: 10.6  
 UPPER 22, 27, Példa: 1.1  
 USER 216  
 user\_all\_tables 242, 725, Példa: 12.5  
 user\_catalog 437, 700  
 user\_col\_privs\_made 207  
 user\_col\_privs\_recd 207  
 user\_cons\_columns 96, Példa: 5.3  
 user\_constraints 96, 415, 439, 723,  
 Példa: 5.3  
 user\_objects 187, 449, 700  
 user\_role\_privs 207  
 user\_source 183, 723, Példa: 10.2  
 user\_sys\_privs 207, 726, Példa: 11.6,  
 11.8  
 user\_tab\_privs\_made 207  
 user\_tab\_privs\_recd 207, 727, Példa:  
 11.8, 12.5  
 user\_tables 721  
 user tablespaces 208, 213, 725, Példa:  
 11.3, 11.8  
 user\_triggers 186, 723  
 user\_users 724  
 user\_views 449, 721  
 users (táblaterület) 208, 210, 217,  
 225, Példa: 11.8  
 USING 57, Példa: 3.1

## V

vágólap 247  
 változó  
 felhasználói ~ 71, 144, Példa: 8.1

helyettesítő ~ 71, 144, Példa: 8.1  
 gazdakörnyezeti ~ 144, Példa: 8.1  
 belső ~ 144, Példa: 8.1  
 VALUES 90, Példa: 5.1  
 VARCHAR2 92, 144-145, 178, Példa: 5.1  
 VARIABLE 75, 144, Példa: 8.1, 8.3  
 VARIANCE 50, 282  
 véletlen szám generálás 155  
 verify 77, Példa: 4.6  
 vessző (.)  
 dátumjel 36  
 ezres csoportosítás jele 37, 495  
 lista elválasztó jele 19, 21, Példa: 1.1  
 paraméterelválasztó jel 21, 23, 24, Példa:  
 1.2  
 vessző (.) 19, 36, 37, Példa: 1.1  
 vezérlési szerkezetek 547  
 vezető nulla 45  
 VIEW 111

## W

W (formátummaszk) 36  
 web-cím elválasztó jel (.) 4, 729, 731  
 web-cím elválasztó jel (/) 731  
 WHEN 24, 177, 187, Példa: 1.2, 9.1, 10.1,  
 10.7  
 WHEN OTHERS 177  
 WHERE 19, 56, 66, 91, Példa: 1.1  
 WHILE-ciklus 151  
 WIDTH\_BUCKET 292, Példa: 13.23  
 Windows 239  
 WITH ADMIN OPTION 205, Példa: 11.8  
 WITH CHECK OPTION 111, 453  
 WITH GRANT OPTION 206  
 WITH READ ONLY 111, 457  
 WITHIN GROUP 270, Példa: 13.8  
 Word (MS-Word) 235, 247  
 write.exe 239, Példa: 12.4  
 WW (formátummaszk) 36

## Y, Z

YEAR (formátummaszk) 36  
 YYYY (formátummaszk) 36, Példa: 1.9  
 YYYYMMDD (formátummaszk) 39, Példa:  
 1.10, 1.12