



BME Villamosmérnöki és
Informatikai Kar



Adatbázisok oktatási labor

Knowledge and Database Management

Távközlési és Médiainformatikai Tanszék
Adatbázisok Labor

ADATBÁZISOK LABORATÓRIUM

Oktatási segédanyag az Adatbázisok laboratórium tantárgyhoz

Egyetemi belső használatra

17. javított és bővített kiadás

szerkesztette:
Gajdos Sándor

szerzők:
Balázs Zoltán
Erős Levente
Gajdos Sándor
Golda Bence
Győr Ferenc
Hajnács Zoltán
Hunyadi Levente
Kardkovács Zsolt
Kollár Ádám
Mátéfi Gergely
Marton József
Nagy Gábor
Nagypál Gábor
Paksy Patrik
Sallai Tamás
Soproni Péter
Unghváry Ferenc
Varsányi Márton
Veres-Szentkirályi András

2017.

Tartalom

ELŐSZÓ.....	2
ADATBÁZISOK LABOR ADMINISZTRÁCIÓS RENDSZER	3
I. LABOR: AZ ORACLE ADATBÁZIS-KEZELŐ.....	7
II. LABOR: AZ SQL NYELV	26
II. FÜGGELÉK: ADATBÁZIS KÉNYSZEREK AZ ORACLE-BEN	39

Előszó

Ez a segédanyag (a továbbiakban: Segédlet) a BME Villamosmérnöki és Informatikai karán folyó műszaki informatikus alapképzés (BSc) tantervében a 4. szemeszterben szereplő “Adatbázisok laboratórium” c. tantárgyhoz készült. Ez hivatott arra, hogy az “Adatbázisok” c. elméleti tantárgyhoz gyakorlati ismereteket is szolgáltatson. A Segédlet az adatbázisokhoz kapcsolódó, összesen 6¹ db foglalkozás segédanyagául szolgál.

A laboratóriumi foglalkozások keretében természetesen csak arra nyílik lehetőség, hogy az elméletben megtanultaknak egy viszonylag szűk részével találkozzanak a hallgatók: ez a rész a relációs adatbázis-kezelés és alkalmazásfejlesztés egyes elemeit jelenti. Eközben erősen építünk azokra az ismeretekre, amelyeket pl. az Adatbázisok tárgy előadásai során lehet megszerezni a relációs adatmodellel, adatmodellezéssel, relációs sématervezéssel, lekérdező nyelvekkel, tranzakciókezeléssel kapcsolatban. A laboratórium anyagába bekerülő elemek folyamatosan változnak, ahogyan az adatbázis-kezelés súlypontjai is eltolódnak. Ennek megfelelően az 1998-as évtől kezdődően a Java nyelv és az adatbázisok kapcsolatát célzó labor, 2002-től dinamikus weboldalak készítése PHP nyelven jelent meg újdonságként a tematikában, 2003-tól az XML alapú adatkezelés és alkalmazásfejlesztés egyes lehetőségeivel ismertetjük meg a hallgatóságot, 2012-től pedig egy példát mutatunk arra, hogyan lehet SOA elvek mentén működő, kliens-szerver architektúrájú rendszert készíteni relációs adatbázisok tartalmának hozzáférésére/hozzáféréseivel. Célunk elsődlegesen nem készség szintű ismeretek nyújtása – bár az SQL nyelvvel kapcsolatban ezt is megfogalmazzuk –, sokkal inkább a fontosabb, adatbázis-kezeléssel kapcsolatos technológiákkal való gyakorlati ismerkedés lehetőségének biztosítása.

Az adatbázis laboratóriumok az Oracle relációs adatbázis-kezelő rendszer (2015-től kezdődően a 12cR1 verziója) segítségével valósulnak meg. Tekintettel az Oracle szerteágazó lehetőségeire meg sem kíséreltük, hogy kimerítő ismereteket próbáljunk meg átadni a rendszerről a laboratóriumok és a Segédlet erősen korlátozott lehetőségei között. Így a laboratóriumok anyagának összeállítása a kompromisszumkeresések története volt.

Bár a szerzők és a szerkesztő mindent elkövettek, hogy a Segédletben leírtak pontosak, korrektek legyenek, ennek ellenére előfordulhatnak benne hibák. Ebben az esetben a visszajelzéseket köszönettel vesszük – sőt kérjük – a gajdos@db.bme.hu címre küldött e-mail formájában.

Budapest, 2017. február

Gajdos Sándor
Távközlési és Médiainformatikai Tanszék
Adatbázisok Labor

¹ Az ünnepnapok miatt elmaradó foglalkozások miatt nem mindig fér bele egy félév programjába valamennyi.

Adatbázisok Labor Adminisztrációs Rendszer

Hallgatói Kezelési Segédlet

Kérjük, hogy ezt a leírást mindenki figyelmesen olvassa el, mert a félév során a laborral kapcsolatos dokumentumok, információk és jegyek ezen a rendszeren keresztül fognak vándorolni.

1.	RÖVID ÁTTEKINTÉS	3
2.	ELÉRHETŐSÉG	3
3.	BÖNGÉSZŐK	3
4.	BEJELENTKEZÉS	3
5.	HALLGATÓI MODUL	4
6.	JEGYZŐKÖNYVEK	6
7.	JEGYZŐKÖNYV FÁJLOK	6
8.	TULAJDONSÁGOK OLDAL.....	6
9.	KILÉPÉS	6
10.	LEVELEZŐLISTA.....	6

1. Rövid áttekintés

A rendszer elsősorban a félév során felmerülő adminisztrációs feladatok megkönnyítése érdekében készült. A program célja a hallgatók és a laborvezetők, illetve a jegyzőkönyv-értékelők közti kommunikáció megkönnyítése, beleértve a jegyzőkönyvek leadását és az osztályozást is.

2. Elérhetőség

A program webes felületének címe:

<https://db.bme.hu/szglab5/>

A felbukkanó tanúsítvány-ellenőrzések mindegyikét elfogadva lehet a rendszerbe bejelentkezni.

3. Böngészők

Törekedtünk arra, hogy minden böngészővel lehessen használni a rendszert, azonban a biztonságot szem előtt tartva az SSL használatáról nem mondhattunk le. A belépéshez ezért olyan böngészőre van szükség, ami támogatja a http titkosítását (HTTPS).

4. Bejelentkezés

Minden hallgató a saját NEPTUN kódjával (ezt kell a név mezőbe írni) és a hozzá tartozó jelszóval léphet be a rendszerbe. Ha valaki véletlenül a félév során elfelejti a jelszavát, lehetőség van a login képernyőn új jelszó generálására (új jelszó gomb).


A jelszó újrageneráláshoz a rendszernek szüksége van egy élő email-címre, amire elküldheti a generált jelszót.

Ezt az email címet már csak azért is érdemes megadni, mert a félév során ez lesz a kapcsolattartás legfontosabb csatornája. Ennek hiányában a tárgy oktatói, ill. a feladatok javítói nem tudnak a hallgatókkal kapcsolatba lépni, amely – tapasztalatok szerint – szinte mindig a hallgatók érdekeit szolgálja.

Számítógép labor 5. Adatbázisok

Hirdetmények:

Az első hirdetés (Feladó: Lacay Bálint)	2003. dec. 21.
Ide mindenféle szöveget lehet írni.	
Ami eddig itt volt, az most erre található...	
doksi (Feladó: Nepusz Tamás)	2004. jan. 20.
Doksi itt van!	

Login 

Név:

Jelszó:

5. Hallgatói modul

A sikeres bejelentkezést követően egy méréseket listázó oldalra kerülünk, melyen az éppen aktuális mérés van kiválasztva. A többi mérés a táblázat bal oldalán található „X. mérés” linkek, illetve a táblázat felett lévő sor segítségével választhatók ki.

Egy *inaktív mérésnél* – ezek vannak a világosabb/keskenyebb csíkokban – az alábbiakat láthatjuk balról jobbra haladva:

- mérés száma,
- mérés helye és időpontja,
- a mérésre kapott beugró-, jegyzőkönyvjegy,
- a mérésre kapott végleges jegy, illetve
- ha van már feltöltött jegyzőkönyv, akkor a jobb oldalon a rendszer azt is felkínálja letöltésre.

Egy *kiválasztott mérés* a következő információkat tartalmazza felülről lefelé haladva:

- mérés száma
- hely, idő, jegyek – mint fent,
- amennyiben van feltöltött jegyzőkönyv, akkor azt itt is le lehet tölteni,
- illetve abban az esetben, ha a labor időpontja és a jegyzőkönyv leadási határideje között nézzük az oldalt, lehetőségünk van feltöltésre is.

A javítást követően a javítók és a laborvezetők megjegyzései is itt olvashatók.

Számítógép labor 5. Adatbázisok

Golda Bence -- KYU2GG

hallgató modul | [beállítások](#) | [logout](#)



[A1 mérés](#) | [A2 mérés](#) | [A3 mérés](#) | [A4 mérés](#) | [A5 mérés](#) | [A6 mérés](#)

A1 mérés:	[hely: R4A]	[idő: 2003. dec. 15., 17:15:00]	[beugró: 1]	[jegyzőkönyv: 1]	[laborjegy: 1]	jegyzőkönyv: letöltés
A2 mérés (laborvezető: Laczay Bálint)						
hely	idő	beugró	jegyzőkönyv	laborjegy		
R4A	2003. dec. 30., 00:00:00	-	1	-		
jegyzőkönyv: letöltés				Leadás ideje: 2004. jan. 11.		Elkészítél!
A laborvezető még nem írta meg az értékelést.				Jegyzőkönyv értékelése:		
A3 mérés:	[hely: R4A]	[idő: 2004. jan. 12., 17:15:00]	[beugró: -]	[jegyzőkönyv: -]	[laborjegy: -]	jegyzőkönyv: letöltés
A4	[hely: R4A]	[idő: 2004. jan. 25., 17:15:00]	[beugró: 1]	[jegyzőkönyv: 1]	[laborjegy: 1]	jegyzőkönyv: letöltés

Számítógép labor 5. Adatbázisok

Golda Bence -- KYU2GG

hallgató modul | [beállítások](#) | [logout](#)



[A1 mérés](#) | [A2 mérés](#) | [A3 mérés](#) | [A4 mérés](#) | [A5 mérés](#) | [A6 mérés](#)

A1 mérés:	[hely: R4A]	[idő: 2003. dec. 15., 17:15:00]	[beugró: 1]	[jegyzőkönyv: 1]	[laborjegy: 1]	jegyzőkönyv: letöltés
A2 mérés:	[hely: R4A]	[idő: 2003. dec. 30., 00:00:00]	[beugró: -]	[jegyzőkönyv: 1]	[laborjegy: -]	jegyzőkönyv: letöltés
A3 mérés:	[hely: R4A]	[idő: 2004. jan. 12., 17:15:00]	[beugró: -]	[jegyzőkönyv: -]	[laborjegy: -]	jegyzőkönyv: letöltés
A4 mérés (laborvezető: Laczay Bálint)						
hely	idő		beugró	jegyzőkönyv	laborjegy	
R4A	2004. jan. 25., 17:15:00		-	-	-	
jegyzőkönyv: letöltés			Leadás ideje: 2004. jan. 25.			
A5 mérés:	[hely: R4A]	[idő: 2004. feb. 09., 17:15:00]	[beugró: -]	[jegyzőkönyv: -]	[laborjegy: -]	
A6 mérés:	[hely: R4A]	[idő: 2004. mar. 01., 17:15:00]	[beugró: -]	[jegyzőkönyv: -]	[laborjegy: -]	

6. Jegyzőkönyvek

Jegyzőkönyv feltöltésére a labor időpontja és a leadási határidő között van lehetőség. A leadási határidőt a feltöltésre szolgáló mező mellett jelzi a rendszer.

Javításra, jegyzőkönyv újbóli feltöltésére van lehetőség (természetesen a határidő letelte előtt) úgy, hogy az új dokumentummal a régi felülíródik. A javítók az utoljára feltöltött dokumentumot fogják kijavítani.

Amennyiben a határidő pillanatában nincs az aktuális méréshez feltöltve jegyzőkönyv, ennek egy napon belüli *egyszeri* pótlására a rendszer lehetőséget biztosít, természetesen a késést mind a javító, mind a laborvezető látni fogja, és a késés ténye csökkentheti a beadott munka értékét.

7. Jegyzőkönyv fájlok

A feltöltések során minden mérésnél 1, azaz egy darab ZIP (tömörített) archívumot várunk, melynek mérete maximálisan 2MB, azaz kétfélmillió bájt lehet. Különösen az első jegyzőkönyv elkészítésekor kíván külön odafigyelést, hogy a képek mérete a szükségesnél ne legyen nagyobb.

8. Tulajdonságok oldal

A fejléc jobb oldalán található linkek közül a „tulajdonságokra” kattintva, az accounthoz tartozó email-cím, illetve új jelszó megadására van lehetőség.

Jelszóváltoztatás esetén a legfelső sorba a régi, majd az alsó két sorba az új jelszót várja a rendszer. Bárminemű hiba esetén a fejlécben olvasható a visszaadott hibaüzenet.

9. Kilépés

Fontos, hogy ha már nem használjuk a rendszert, a „logout” linkre kattintva lépünk ki, ugyanis ha ezt nem tesszük meg, adataink rossz kézbe kerülhetnek.

Szeretnénk felhívni mindenkinél a figyelmét a következőkre:

1. Az első belépést követően mindenki változtassa meg a jelszavát. Ezt a jelszó generálást követően rövid időn belül – kb. az első mérések idejében – ellenőrizni fogjuk.
2. Ahhoz, hogy az „új jelszó” gomb helyesen működhessen, mindenkitől elvárjuk, hogy egy élő email-címet írjon be a tulajdonságok oldalon. Bármilyen **fontos**, a labor lebonyolítását érintő változásról/eseményről szóló levelet is erre a címre küld a rendszer. Vállaljuk, hogy ezeket az adatokat bizalmasan kezeljük, harmadik félnek át nem adjuk.
3. A rendszerrel kapcsolatos összes lekérdezést és akciót naplózzuk, a betörési, illegális módosítási kísérleteket is szankcionáljuk.
4. A feltöltött fájlokban felfedezett vírusokat 1-2-3 jegy levonásával jutalmazzuk attól függően, hogy mennyire új vírusról van szó.

Bármilyen technikai jellegű probléma esetén a laboradminisztrációs rendszer készítői elérhetők az alábbi email címen:

- Golda Bence (gbence@db.bme.hu)

10. Levelezőlista

A tárggyal és a beadandó feladatokkal kapcsolatos kérdésekben nyújt konzultációs lehetőséget a konzi.adatlabor@db.bme.hu levelezőlista, amelynek a tárgy oktatói is tagjai. A tárgy célja az önálló feladatmegoldás, ezért a levelezőlista elsősorban feladatértelmezési kérdésekben segíthet, feladatmegoldások közzlése nem engedélyezett. A listáról a labor adminisztrációs rendszerben lehet leiratkozni a „Beállítások” menüpont alatt.

I. labor: Az Oracle adatbázis-kezelő

Szerzők: Gajdos Sándor, Kardkovács Zsolt, Győr Ferenc, az Oracle 12c-re átdolgozta
Marton József²

1.	MIÉRT ÉPPEN AZ ORACLE?	7
2.	AZ ORACLE TÖRTÉNETE	8
3.	AZ ORACLE FELÉPÍTÉSE	8
3.1.	Logikai felépítés	9
3.2.	Fizikai felépítés	14
3.3.	Kapcsolat a logikai és fizikai felépítés között	15
3.4.	A rendszer működése	16
3.5.	Az Oracle biztonsága	18
4.	AZ ORACLE ÜZEMELTETÉSE	19
4.1.	Az SQL Developer indítása	19
4.2.	A szerverpéldány beállításai (DBA üzemmód, „Database Configuration”)	20
4.3.	Munkamenetek/Sessions (Reports fül)	21
4.4.	Zárak/Locks (Reports fül)	21
4.5.	Az adatbázis tartalmának kezelése („Schema Manager”)	21
4.6.	Alapvető biztonsági beállítások (DBA üzemmód, Security)	21
4.7.	Fizikai tárolási paraméterek (DBA üzemmód, Storage)	22
5.	NÉHÁNY TOVÁBBI ORACLE TERMÉK	22
6.	FÜGGELÉK	23
6.1.	Az Oracle újabb verzióinak összehasonlítása	23

„I am Sir Oracle,
And when I ope my lips, let no dog bark!”
(Shakespeare: The Merchant of Venice)

1. Miért éppen az Oracle?

Az adatbázis-kezelők piacán széles választékban állnak rendelkezésre a különböző hatékonyságú és technológiájú eszközök. A labor célja, hogy megismertessük a hallgatóval a korszerű adatbázis-kezelés néhány fontosabb elemét, ugyanakkor a lehetőségekhez képest naprakész tudással is felruházzuk őt. Ebből a célból a magyar piacon nagymértékben domináló terméket, az Oracle Database adatbázis-kezelőt (a továbbiakban, röviden: Oracle) mutatjuk be részletesebben. Megemlíjtük – a teljesség igénye nélkül –, hogy az ismertebb és támogatottabb adatbázis-kezelők (továbbá gyártói) a következők: *SQL Server* (relációs, Microsoft), *DB/2*, *Informix* (relációs, IBM), *Ingres IF*³ (relációs, Actian Corporation, GNU GPL), *Sybase Adaptive Server* (relációs, Sybase), *MySQL*⁴ (relációs, Oracle Corporation, GNU GPL), *O2* (objektumorientált, O2 Inc.), *GemStone* (objektumorientált, GemTalk Systems), *ObjectStore* (objektumorientált, ObjectStore Corporation, Versata csoport).

A későbbiekben, az Oracle ill. más adatbázis-kezelők további érdekesebb alkalmazásaival az ez iránt érdeklődők például a Távközlési és Médiainformatikai Tanszéken találkozhatnak.

A laborokon az Oracle 12cR1 verziójú⁵ szerverével valamint Java platformon futó kliens eszközeivel fogunk dolgozni.

² Oracle Database 10g, 11g és 12c verziókra, és az Oracle SQL Developer kliensszközre átdolgozta.

³ A University of Californián kezdődött az Ingres adatbázis-kezelő pályája kutatási projektként, mely az üzleti frontra lépés után több tulajdonosváltás után került a mostani tulajdonosához. Érdekesség, hogy több relációs adatbázis-kezelő rendszer is az Ingres-ből nőtte ki magát, így például a mai PostgreSQL is.

⁴ A MySQL gyártója eredetileg MySQL AB cég, melyet előbb a Sun Microsystems kebelezett be (2008), majd az Oracle Corp.-hoz került a Sun megvásárlásakor (2010).

⁵ Az Oracle adatbázis-kezelő verziószámában az R1 a Release 1, azaz a főverzió 1. kiadás megfelelője.

2. Az Oracle története

A relációs adatbázis-kezelők megjelenésével párhuzamosan, az 1970-es évek elején a CIA egy projektet indított el annak reményében, hogy olyan tudásbázist hozzon létre, amelynek segítségével úgymond „valamennyi kérdésre megkaphatja a választ”. A „kinyilatkoztatás, prófécia” kódnevet adták neki, azaz „Oracle”-re keresztelték el. A projekt ugyan forráshiány miatt abbamaradt, azonban a munkálatokban résztvevő három szakértő: Larry Ellison (a vállalat jelenlegi elnöke), Bob Miner és Ed Oates 1977-ben megalapította a Software Development Laboratories nevű céget, amely 1983 óta viseli a jelenlegi Oracle Corporation nevet. Az alapítás után nem sokkal piacra került az első Oracle nevű adatbázis-kezelő (a CIA volt az első vásárlója a terméknek). Ma az Oracle a legelterjedtebb adatbázis-kezelő rendszer, minden számottevő operációs rendszerre és hardver architektúrára elérhető és telepíthető.

3. Az Oracle felépítése

Az Oracle relációs (valójában ún. objektum-relációs) adatbázis-kezelő rendszer; karbantartására és használatára egyaránt egy szabványos nyelvet, az SQL-t, pontosabban az SQL:2011-et használhatjuk. Az első labor keretében megismerkedünk az Oracle kezelését nagymértékben egyszerűsítő Oracle SQL Developer program menedzsment-szolgáltatásaival. Az SQL lekérdező részeivel (és az SQL Developer ahhoz kapcsolódó felületével) a következő laboron fogunk mélyebben megismerkedni. Mivel a későbbiekben is intenzíven fogjuk használni, ezért a következő laboron az SQL mélyreható ismeretét fogjuk számon kérni.

A teljes Oracle adatbázis-kezelő rendszer fontosabb tulajdonságai a következők:

- Alapvetően kliens-szerver felépítésű.
- Az operációs rendszertől függetlenül lehetővé teszi a többtaszkos, több felhasználós működést, az adatok egyidejű használatát.
- Térben elosztott rendszerként is képes működni.
- A fontosabb hálózati protokollokkal és operációs rendszerekkel együtt tud működni.
- Támogatja a szoftver- és alkalmazásfejlesztés minden egyes szakaszát.
- Képes együttműködni a lényegesebb fordítókkal és fejlesztői környezetekkel.
- Gyakorlatilag tetszőlegesen nagy adatmennyiséget is képes kezelni (különböző hatékonysággal).
- Napi 24 órás rendelkezésre állást, biztonságos működést tud garantálni.
- Magas szinten képes biztosítani az adatok védelmét, integritását, konzisztenciáját.
- Alkalmas összetett struktúrák (objektumok, multimédia adatok, eljárások) tárolására is.
- Fejlett rendszerfelügyelet biztosítható az Oracle Management Server és a hozzá kapcsolódó Agentek segítségével. Ekkor az Enterprise Manager⁶ alkalmazás segítségével egy tetszőleges méretű adatbázis-park adminisztrálása/távfelügyelete válik lehetővé.
- Az Oracle, mint cég megbízható terméktámogatási rendszert nyújt a felhasználóinak.

A *szerver* (server) alatt minden esetben egy *adatbázist* (database) és egy *szerverpéldányt* (instance) értünk. Az adatbázisban tárolódnak a felhasználói és rendszeradatok⁷, míg a szerverpéldány a szolgáltatás futtatásához szükséges folyamatok (és szálak⁸) összessége. Egy szerverszámítógép több adatbázisnak is helyet adhat. A legmagasabb szintű, névvel ellátott

⁶ Az Enterprise Manager többet is nyújt: az operációs rendszer szintű virtualizációtól kezdve egészen az alkalmazásszerverig egy egész cloud-infrastruktúra szoftveres kezelését képes biztosítani.

⁷ A 12c konténer-alapú, ún. multitenant architektúrájában külön adatbázisban, az ún. CDB\$ROOT-ban kapnak helyet a rendszer-szintű adatok és metaadatok. A felhasználói adatbázisban csak a felhasználói adatok és az ezekhez kapcsolódó metaadatok foglalnak helyet.

⁸ A 12c verziótól kezdődően.

tárolási egység tehát az adatbázis, ami ennek megfelelően jóval több az adatok összességénél! A szerverpéldány a szerver működésének, futásának módját határozza meg. Ilyen például a klaszterezés (Oracle RAC), replikációs beállítások, stb.

A 12c verzió nagy újdonsága az ún. multitenant architektúra, ahol a szerverpéldányhoz több⁹ felhasználói adatbázis (pluggable database, PDB) is tartozhat. A PDB-ktől különválasztották a rendszerszintű adatokat és metaadatokat egy konténer-specifikus adatbázisba (neve: CDB\$ROOT), amelyből konténerenként pontosan 1 példány van. Mivel egy szerverpéldány van jelen, az ahhoz tartozó adatbázisok futásának módja közös, és konténerenként csak egyszer kell a szoftver-karbantartási lépéseket is végrehajtani. A 12c konfigurálható a hagyományos felépítésben történő üzemre is (ez az ún. non-CDB üzemmód), de a felhasználói programok futása szempontjából átlátszó, hogy multitenant vagy hagyományos konfigurációban fut-e az adatbázis-kezelő. A továbbiakban, ha az ellenkezőjét külön nem írjuk, az Oracle Database hagyományos felépítését mutatjuk be, hiszen a labor keretében csak ezt van lehetőség megismerni.

Az adatbázis jól szétválasztható egy fizikai és egy logikai egységre, amelyek külön-külön is karbantarthatóak. A két egységet és a közöttük lévő kapcsolatokat, összefüggéseket az alábbiakban mutatjuk be.

3.1. Logikai felépítés

Az adatbázisokat *táblahelyekre* (tablespace, egyes magyar fordításokban táblatér) oszthatjuk fel, amelyek a tárolás logikai egységeit határozzák meg. A táblahely a legnagyobb logikai tárolási egység. A hasonló működésű és karbantartást igénylő adatok kerülnek célszerűen egy táblahelybe. A táblahelyek lehetőséget biztosítanak arra, hogy az adatbázis adatainak elhelyezését közben tarthassuk; elosszuk különböző tárolási egységek között, kvótákat határozhatunk meg az egyes felhasználók számára stb. Minden esetben van legalább egy táblahely, amelyet **system**nek nevezünk. Ennél természetesen lényegesen több táblahelyet is létre lehet hozni, sőt, általában nem szerencsés felhasználói adatokat a system táblatérbe tenni. Egy jellegzetes rendszer általában a következőket foglalja magába:

system: a rendszerről tárolt információkat tartalmazó táblahely (vö. adatszótár, data dictionary),

sysaux: kiegészítő táblahely a *system* mellett, amely a 10g verzióban jelent meg. Az Oracle adatbázis néhány olyan funkcionalitása, amelyek korábban a system, vagy különálló táblahelyekben kaptak helyet, most a sysauxot használják. Ilyen például a LogMiner (az adatbázis naplóállományainak programozói feldolgozását biztosító csomag) vagy az Oracle Data Mining opció (adatbányászat csomag).

rbs: az adatbázison végzett műveletek eredményeit, naplóit tartalmazó táblahely az Oracle 9 előtti verziókban (lásd még később a rollback szegmensekről írtakat). Az újabb verziókban hasonló szerepet tölt be az **undo** tablespace.

temp: mindenféle átmenetileg tárolandó adat számára fenntartott táblahely (pl. egyes lekérdezések részeredményei, rendezések eredményei),

tools: alkalmazások ill. általános eszközök által használt minta-táblahely (például a Form Builder is ezt használja),

users: általános felhasználói minta-táblahely.

Az utóbbi két táblahely létezése nem kötelező.

A táblahelyek összességében rendszer- (system) vagy felhasználói (user) objektumokat, egységeket tartalmaznak. Ezek az objektumok lehetnek *táblák* (table), *nézetek* (view), *számlálók* (sequence), *szinonimák* (synonym), *indexek* (index), *csoportok* (group), klaszterek (cluster), *kapcsolódási pontok* (database link) stb., illetve ezekből képzett olyan összetett

⁹ A 12c verzióban legfeljebb 252 PDB.

struktúrák, mint amilyen például az *adatszótár* (data dictionary), az (Oracle) *séma* (schema), vagy a futtatható (tárolt) objektumok (stored procedures).

Egy objektum csak egyetlen táblahelynek lehet része, az összetett struktúrák azonban átnyúlhatnak több táblahelyen is.

Fő felhasználói objektumok

Tábla (Table): a logikai adattárolás alapegysége, amely sorokból és oszlopokból áll. Megfeleltethető egy relációnak. A táblát egyértelműen azonosíthatjuk a tábla nevével (pl. SCOTT.EMP). Az oszlopokat a nevük, a típusuk és a méretük jellemzi. Egy táblában csak egyféle típusú adatrekordot tárolhatunk. A sorok sorrendje lényegtelen. Egy tábla létrehozásakor meg kell adni a tábla nevét, valamint a tábla oszlopainak nevét, típusát és méretét. Az alábbi ábra egy táblát, és az abban tárolt adatokat mutatja.

Nézet (View): egy vagy több táblából összeszerkesztett adatok megjelenítésére alkalmas felhasználói objektum. Felfogható úgy is, mint egy tárolt lekérdezés; se nem tartalmaz, se nem tárol (fizikailag) adatot, csak származtatja az adatokat azokból a táblákból, amelyeken értelmezték. Az ilyen táblákat hívjuk a nézet alaptábláinak (base table, master table).

Számláló (Sequence): sorfolytonos, egyedi számgenerátor. Általában valamilyen egyedi azonosító létrehozására használjuk. Fontos, hogy értéke nem tranzakció-orientált, tehát pl. egy rollback művelet nem módosítja a számláló értékét.

	Columns						Column names
Rows	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7329	SMITH	CLERK	7902	17-DEC-88	800.00	300.00	20
7499	ALLEN	SALESMAN	7698	20-FEB-88	1600.00	300.00	30
7521	WARD	SALESMAN	7698	22-FEB-88	1250.00	500.00	30
7566	JONES	MANAGER	7839	02-APR-88	2975.00		20

Column not allowing nulls

Column allowing nulls

1. ábra: Tábla (Ábra forrása: Oracle 11gR1 Database Concepts, Fig 5-2: The EMP Table)

Szinonima (Synonym): egy táblára, nézetre vagy számlálóra több név is megadható a szinonimák segítségével. Lehetőségünk van tehát rövidíteni vagy átlátszóvá tenni az egyes objektumok tárolási helyét. Van nyilvános (public) és rejtett (private) szinonima is. A nyilvános szinonima mindenki számára hozzáférhető, míg a rejtett szinonima csak a felhasználók egy meghatározott körének érhető el. A nyilvános szinonima létrehozása és eldobása speciális jogokhoz köthető.

Index (Index): adatokhoz való hozzáférést (általában) gyorsító eszköz – az Oracle-ben alapesetben egy B* fa. Az esetek többségében olyan táblaoszlop(ok)ra érdemes index létrehozni, amelyre gyakran fogalmazunk meg keresési feltételt. Az indexek automatikusan létrejönnek mindazokra az oszlopokra, amelyekre már a tábla megadásakor megköveteljük az egyediséget. Az indexek frissítését a rendszer automatikusan elvégzi. Az index lehet összetett, azaz több mezőből álló index is, ilyenkor érdemes a nagy kardinalitású oszlopokat az attribútumlista elejére venni. Az indexek létrehozása konkrét esetekben (éles, erőforrás-igényes környezetekben) gondos tervezői munkát igényel, gondatlan megválasztása lassíthatja a működést.

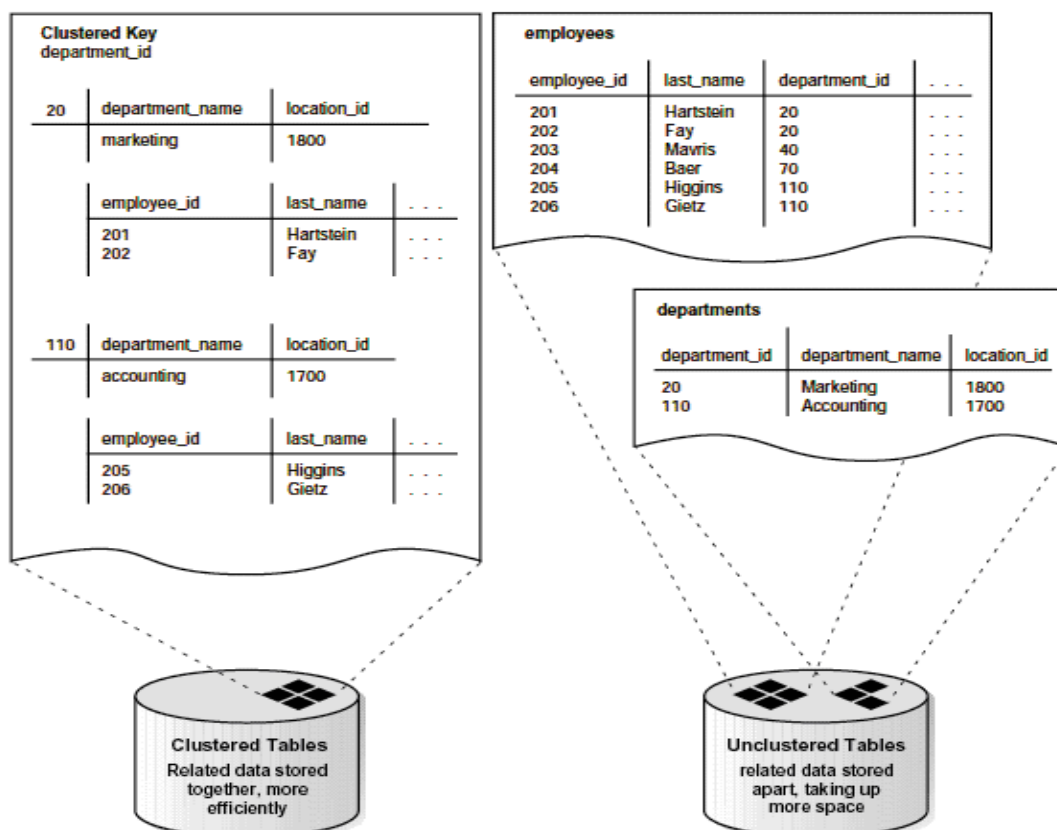
Csoport (Group): Több szervert összefogó struktúra az Oracle 9i-ben. Szerepe a rendszerfelügyelet egyszerűsítése.

Kapcsolódási pont (Database Link): olyan szinonima, amelyen keresztül nem objektumokat, hanem adatbázisokat érhetünk el. Említettük, hogy egy szerverszámítógép több adatbázist is tartalmazhat, sőt lehetnek akár elosztott, azaz több, különböző számítógépen tárolt, azonban adatait tekintve összefüggő adatbázisok is. Ilyen esetekben szükségünk lehet kapcsolódási pontok definiálására.

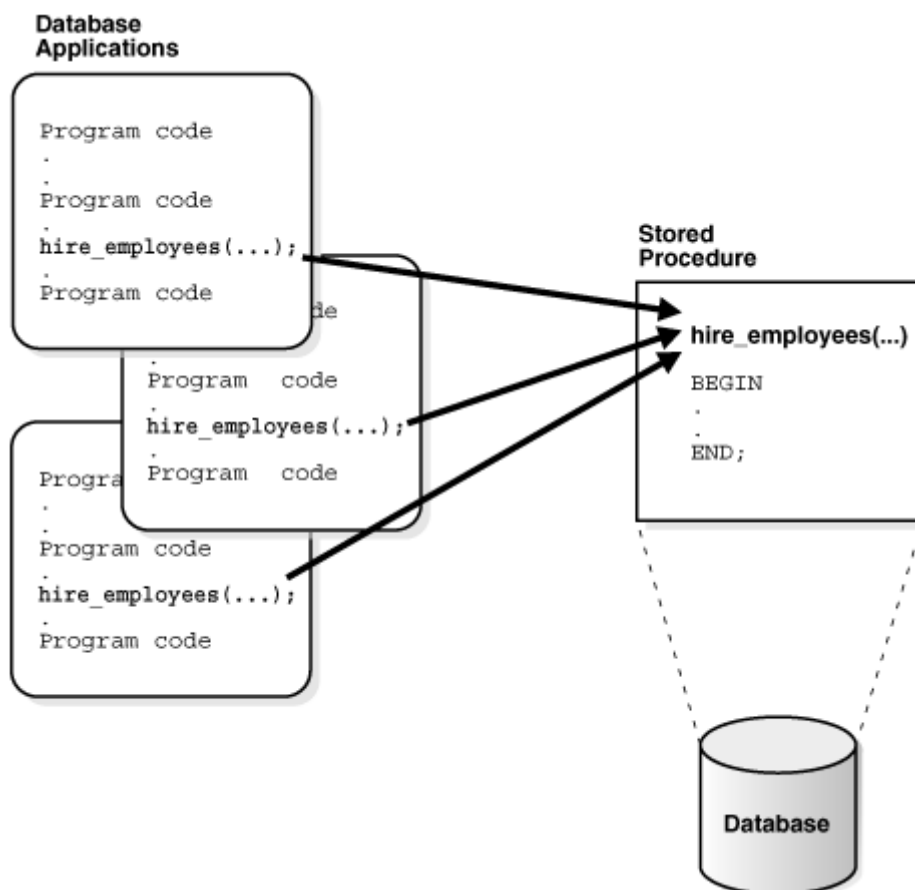
Adatszótár (Data Dictionary): csak olvasható táblák és nézetek gyűjteménye, amelyek a rendszer mindenkor állapotát rögzítik. Ennek megfelelően megtalálható benne, hogy milyen felhasználók vannak a rendszerben, azok mely objektumokhoz férhetnek hozzá; milyen kényszereket kell érvényesíteni az egyes mezőkre; milyen alapértékek vannak beállítva az egyes oszlopokra; mennyi helyet foglalnak az egyes objektumok, mennyi hely van még szabadon; ki, mikor lépett be az adatbázisba és mit módosított vagy nézett meg stb.

Séma (Schema): egy adott felhasználó saját objektumainak összességét nevezzük sémának, vagy a felhasználó sémájának. A felhasználó és sémája között 1-1 értelmű megfeleltetés áll fenn. Az objektumok elérhetőségét a jogosultsági rendszer beállításai korlátozhatják, amelyek a felhasználóra ill. a felhasználók csoportjaira vonatkoznak, így módon az egyes felhasználók számára biztosítható a sémáikon kívüli objektumok elérése is.

Klaszter (Cluster): az azonos kezelési vagy hozzáférési módot igénylő adatokat érdemes egyetlen csoportba, fizikai helyre tenni. Ha az összetartozó adatok fizikailag „közeli” helyeken vannak, akkor adatbehozatalkor a hozzáférési idő jelentősen csökkenhet. Tehát a csoportokba szervezéssel a hatékonyságot lehet növelni. A 2. ábra egy tipikus példát mutat.



2. ábra: Klaszterezett (bal oldal) és klaszterezés nélküli adatszerzés (jobb oldal) (Ábra forrása: Oracle 11gR1 Database Concepts, Fig 5-10: Clustered Table Data)



3. ábra: A tárolt eljárások az adatbázisban tárolódnak és a szervert futtatja őket. (Ábra forrása: Oracle 11gR1 Database Concepts, Fig 25-2: Stored Procedure)

Tárolt eljárások (Stored Procedures, Functions, Packages): az adatbázisban tárolt, és ott futtatható objektumok összessége. Az adatbázis táblahelyeiben lehetőség van (célszerűen a széles körben használt) szervert futó, végrehajtható objektumok (pl. PL/SQL, Java, illetve megfelelő beállítások esetén egyéb forrásnyelvi, ún. programok/programrészletek) tárolására is. Az Oracle rendszer installálásakor számos „gyári” tárolt eljárás kerül telepítésre, amelyek pl. megkönnyítik a rendszer adminisztrálását, fejlesztését (3. ábra).

Az egyes objektumok – ahogyan korábban utaltunk rá – elemi adattípusokból épülnek fel. Az Oracle-ben a következő adattípusokkal fogunk találkozni:

Adattípus	Rövid leírás
CHAR (<i>n</i>)	Állandó méretű karakterfüzér-típus. Állandó, azaz mindig az előre megadott méretű helyet foglalja le számára a rendszer, függetlenül attól, hogy a karakterfüzér kitölti-e a rendelkezésre álló helyet, vagy sem. Amennyiben a beillesztett szöveg nem tölti ki a teljes méretet, úgy az adatbázis-kezelő kiegészíti a végén a megfelelő számú szóközzel. (Ezt lekérdezéskor is figyelembe kell venni!) A típus maximális mérete az Oracle8-as sorozattól kezdve 2000 bájt. Fontos hangsúlyozni, hogy alapértelmezésben az adatbázis illetve a munkamenet beállításától függ, hogy bájtokban vagy karaktorszámokban adjuk meg az adatok lehetséges legnagyobb méretét, hiszen van több bájtos karakterből álló karakterkészlet (pl. az UTF8) is. Ezért javasolt explicit megjelölni a hossz-szemantikát a zárójelben a hosszúságot jelző szám után a „char” ill. „byte” szó megadásával. Ha nem

	adjuk meg az adott típusú mező méretét, akkor az alapértelmezés szerint a rendszer azt 1 bájtának fogja venni. Az adattípusban használt karakterkódolás, és így az ábrázolható karakterek köre az adatbázis karakterkészlet-beállításától (database character set) függ.
VARCHAR2 (n)	Változó hosszúságú karakterfüzér-típus. A CHAR típussal ellentétben ennél a típusnál nem egészíti ki a szöveget szóközökkel a maximális hosszra a rendszer, és általában csak a ténylegesen felhasznált méretet foglalja le az Oracle, így használata sokszor indokolt, jobb hatásfokú. A típus maximális mérete az Oracle8-tól 4000 bájt lehet, 12c-től kezdődően 32767 bájt lehet az adatbázis-kezelő megfelelő konfigurációja esetén.
NCHAR (n) NVARCHAR2 (n)	Az NCHAR és az NVARCHAR2 rendre a CHAR és VARCHAR2 adattípusok Unicode megfelelője, bájtokban vett maximális méretükre ugyanazok a korlátok vonatkoznak. Az ilyen típusú mezőkben Unicode karakterláncok tárolhatók, függetlenül az adatbázis karakterkészlet-beállításától. A típus maximális méretét (n) minden esetben karakterekben kell megadni.
CLOB (LONG)	Nagyméretű szövegek tárolására alkalmas típus. Amennyiben fentieknél nagyobb méretben szeretnénk karakterfüzért tárolni (nem kell megadni felső korlátot), akkor érdemes a megfelelő mezőt CLOB-nak (<i>Character type Large Object</i>) definiálni. A CLOB-nak is van maximális mérete, de ez kellően nagy: elméletileg 4 gibiblokk ¹⁰ is lehet. A korábbi Oracle verziókkal való kompatibilitás miatt megmaradt a LONG típus is, ami a megvalósítását tekintve szintén CLOB adatstruktúra. A kettő azonban nem azonos. Lényegesebb különbségeket kiemelve: a LONG típus 2 gibibájtban ¹¹ limitált; egy sémaobjektum csak egy LONG típust tartalmazhat, míg CLOB-ot tekintve korlátlan sokat; a LONG típus csak soros, míg a CLOB véletlen hozzáférésű is lehet.
NUMBER (p, s)	Tetszőleges szám ábrázolására alkalmas adattípus. Egyaránt használható fix- és lebegőpontos számaábrázolásra. Ábrázolási tartománya azon számok halmaza, amelyek abszolút értéke a $[10^{-130}, 10^{126})$ intervallumban van. A számok esetében kétféle méret is megadható; az első a szám tízes számrendszerbeli helyi értékeinek a számát (p), míg a második a pontosságot (s), azaz a tizedes vessző után álló helyi értékek számát jelenti. Tehát ha egy legfeljebb $\pm 999,99$ -ig terjedő számot, két tizedes pontossággal szeretnénk ábrázolni, akkor azt NUMBER(5, 2) formában kell megadnunk. Egész számokat a NUMBER(p) típussal definiálhatunk, amely ekvivalens a NUMBER(p, 0)-val.
DATE	Dátumok megjelenítésére és tárolására alkalmas típus. Az Oracle valamennyi olyan dátumot képes tárolni, amely i.e. 4713. január 1. és i.sz. 9999. december 31. közé esik. A dátum hét darab mezőből áll: század, év, hónap, nap, óra, perc, másodperc. Számos további származtatott egysége is hozzáférhető, úgymint a hét melyik napja, az év hányadik hete stb. Más időszámítási rendszerek (pl. a pravoszláv, a héber, a kínai, a Julián stb.) is elérhetőek az Oracle-ben, sőt az egyes értékek át is válthatóak egymásra. A dátumkezeléssel kapcsolatban lásd még az Oracle beépített dátumfüggvényeit.
ROWID	Az adatrekordok egyedi logikai és fizikai azonosítója. Minden tábla

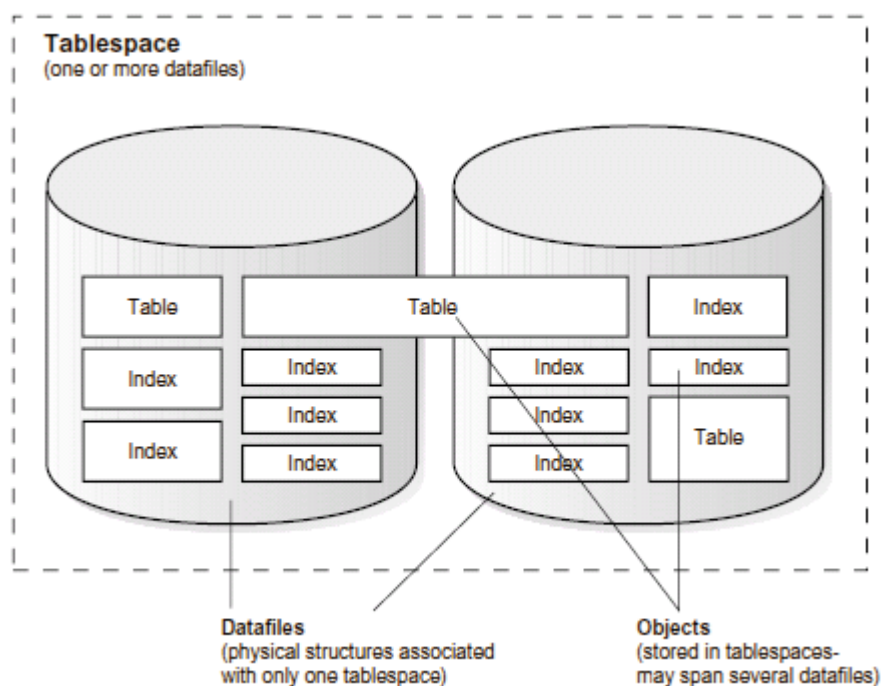
¹⁰ A gibi prefixum általánosságban $2^{30}=1024 \times 1024 \times 1024$ -szorost jelent.

¹¹ 1 gibibájt=1024 mebibájt, míg 1 gigabájt=1000 megabájt. Hasonlóan létezik kibi, mebi...stb. prefixum is.

	rendelkezik egy ROWID nevű segédoszloppal (pseudo column). Az oszlop elemei a tábla sorait egyértelműen azonosítják. Jellemzően hexadecimális kódolásban tartalmazza a sor fizikai elhelyezkedésére és elérésére vonatkozó információkat. Az ilyen kódolású típust nevezzük ROWID típusnak.
UROWID	<p>Az UROWID típus olyan rekordok logikai egyedi azonosítóját tárolja, amelyek fizikai helye más rekordokon végzett műveletektől, vagy az Oracle adatbázis-kezelő hatáskörén kívül eső körülményektől függ. Az ilyen rekordokat tartalmazó táblákban a ROWID nevű segédoszlop UROWID típusú.</p> <p>Az ún. index-szervezésű táblákban (Index-organized Table, IOT) a rekordok az indexek levelében tárolódnak, amelyek új rekordok beszúrásakor/törlésekor, meglévő módosításakor áthelyezésre kerülhetnek más fizikai blokkba. Az index-szervezésű táblák rekordjainak UROWID típusú azonosítója mindaddig változatlan marad, amíg az elsődleges kulcs értéke változatlan.</p> <p>Az Oracle adatbázison kívül tárolt táblák rekordjainak azonosítói szintén UROWID típusúak.</p>

3.2. Fizikai felépítés

Adatállomány (Data file). Az Oracle a táblahelyek adatait egy vagy több adatállományba helyezi el, de egy adatfájl legfeljebb egy táblahelyhez tartozhat – és ennek megfelelően csak egyetlen adatbázishoz (4. ábra). Az állománykezelésnél (lásd Operációs rendszerek, Számítógép architektúrák c. tárgyak) megismert módon, a tárolni kívánt adatok nem feltétlenül azonnal, az utasítás végrehajtásának pillanatában kerülnek be az adatbázisba, jóval hatékonyabb a szakaszos adatkivitel.



4. ábra: A logikai és fizikai tárolás kapcsolata. (Ábra forrása: Oracle 11gR1 Database Concepts, Fig 3-1: Datafiles and Tablespaces)

„Redo-log” állomány (Redo-log file). Egy Oracle adatbázishoz általában kettő vagy több redo-log (pontos fogalmát és szerepét az Adatbázisok c. tárgy részletesen tárgyalja) állomány tartozik. Ezek összességét nevezzük az adatbázis redo-log állományának. Elsődleges feladata, hogy az adatbázison elvégzett műveleteket tárolja egészen az adatkivitel sikeres befejezéséig. Legalább kettő szükséges, így amíg az egyikbe írjuk a redo log buffer tartalmát, addig a másikat további adatbázisfolyamatok (pl. ARCn, ld. később) használhatják. Mivel az adatbiztonság szempontjából igen kritikus az üzemszerű működésük, így az Oracle támogatja a különböző tárterületekre elosztott redo-log állományok (redo-log group több fájlal) használatát.

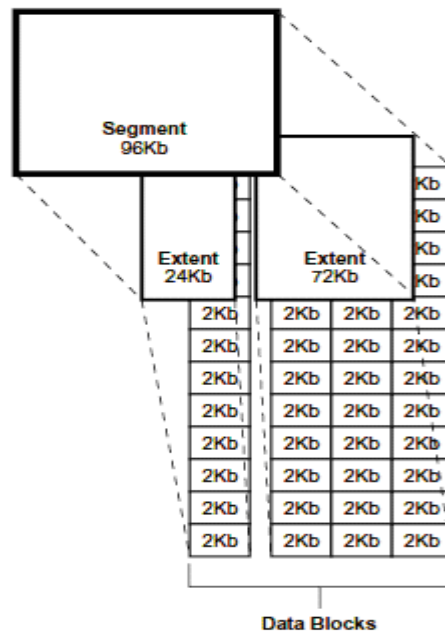
Vezérlési állomány (Control file). Az adatbázis fizikai struktúrájáról tartalmaz információkat. Ilyen információ pl. az adatbázis neve, az adatfájlok neve és fizikai elhelyezkedése, redo-log állományok helye, az adatbázis létrehozásának időpontja stb. (ld. később, az SQL Developer DBA üzemmódjának bemutatásánál). Többnyire egyetlen példány elegendő, de biztonsági okokból ezt is meg lehet többszörözni, el lehet osztani különböző tárterületekre.

3.3. Kapcsolat a logikai és fizikai felépítés között

A fizikai adattárolás logikai egységei három elemből állnak. A legkisebb egység az *adatblokk* (data block, logical block, Oracle block). Az adatblokk általában állandó méretű, összefüggő, az operációs rendszerre vagy magára a (diszken lévő) partícióra jellemző blokkméret (jellemzően 512–4096 bájt) többszörösének megfelelő tárterületet jelent. Az adatblokkok az adatkivitel és az adatbehozatal szempontjából fontosak, hiszen ez az a legkisebb egység, amit az Oracle egy egésként kezel.

Az *extent* (extent) adatblokkok összefüggő halmaza. Az extentek szerepe akkor kerül leginkább előtérbe, amikor egy szegmens – a következő logikai adattároló egység – betelik; ilyenkor az Oracle egy extent méretű hellyel bővíti (egyéb megszorítás hiányában) a használható diszktérületet. Következésképpen egy extent pontosan egy fizikai felépítésre jellemző állományhoz tartozhat.

Ahogy az 5. ábra is mutatja, több logikailag összetartozó extent alkot egy *szegmenst* (segment). Négyféle szegmenst különböztet meg az Oracle, amelyek rendre:



5. ábra: Fizikai adatszervezés az Oracle-ben. (Ábra forrása: Oracle 11gR1 Database Concepts, Fig 2-1: The Relationships Among Segments, Extents, and Data Blocks)

Adatszegment (data segment): minden táblában megtalálható adat egy ilyenben foglal helyet.

Indexszegment (index segment): a különféle indexek hatékony tárolására alkalmas szegment.

Ideiglenes szegment (temporary segment): minden művelet végrehajtásához az Oracle igényelhet egy ideiglenes munkaterületet, amelyet sikeres befejezés után eldob.

Rollback szegment (rollback segment): minden megváltoztatott, de még nem committált érték, elem adatát tárolhatjuk itt. Az újabb Oracle verziókban (9-től felfelé) ez a szegment nem létezik.

Egy szegment több adatállományon is átnyúlhat. A szegmenteket a táblahelyek fizikai megvalósításának tekinthetjük.

3.4. A rendszer működése

Az Oracle indításakor a rendszer lefoglal egy memóriaterületet, valamint elindít számos folyamatot (szálat). Ezek együttese alkot egy Oracle példányt. Minden Oracle adatbázishoz tartozik egy Oracle példány, ami az adatbázis üzemszerű működéséért felelős.

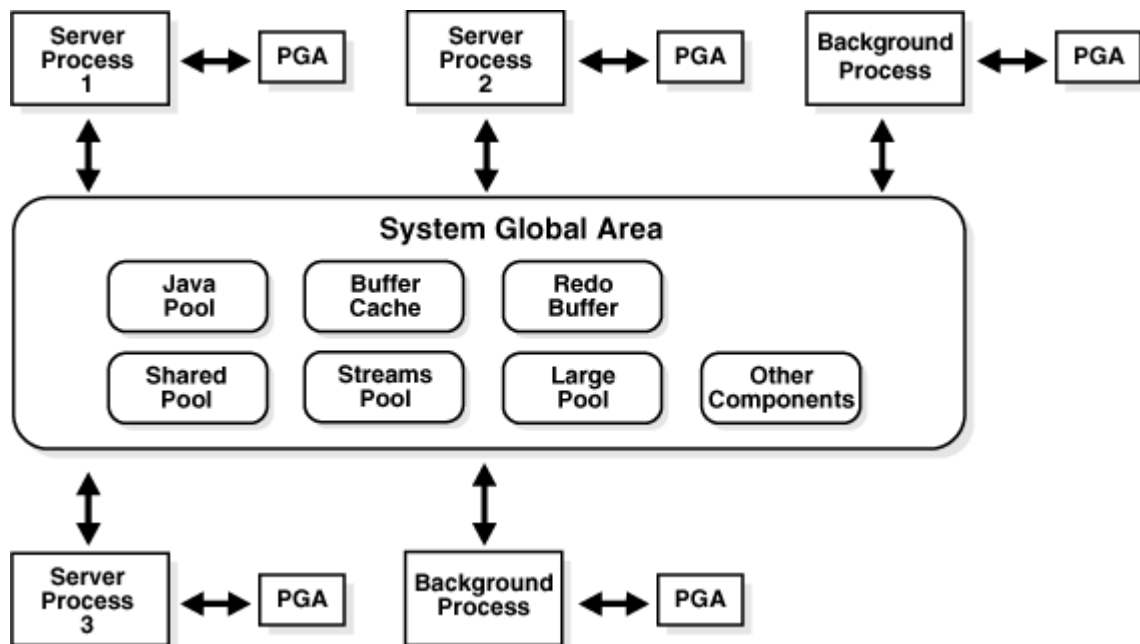
Az indításkor lefoglalt *osztott*, a folyamatok számára elérhető memóriaterület az *SGA* (System Global Area). Az SGA mindazon információkat tartalmazza, amelyek az Oracle vezérléséhez szükségesek, másrészt gyorsítótárként is működik: az utolsó használt blokkokat, egy redo-log puffert, az utolsó használt könyvtárak, állományok adatait, az utolsó végrehajtott utasításokat és azok eredményeit (database buffer cache), az Oracle Java folyamatainak memóriastruktúráit, valamint az adatszótárnak egy részletét is magában foglalja (6. ábra).

Az ábrán látható egyes szerverfolyamatokról a későbbiekben részletesen lesz szó.

Az egyes szerverfolyamatok mindegyikéhez lefoglalásra kerül a *PGA* nevű memóriastruktúra (Program Global Area), amely az adott folyamat állapotát tárolja.

Az Oracle által indított folyamatok részben *rendszerfolyamatok* (server process), részben *háttérfolyamatok* (background process). A rendszerfolyamatok a felhasználót kiszolgáló műveletek: az SQL értelmező és végrehajtó folyamat, az adatkiviteli és -behozatali folyamat a háttértár és az SGA között, valamint a felhasználó számára az eredményeket visszaadó

folyamat. A háttérfolyamatok jóval nagyobb számban vannak, a különböző karbantartási feladatokért, a rendszer hatékonyságának megtartásáért felelősek. A legfontosabbak:



6. ábra: A System Global Area (SGA) által tárolt adatok. (Ábra forrása: Oracle 11gR1 Database Concepts, Fig 8-1: Oracle Database Memory Structures)

Rendszerfelügyelő folyamat (system monitor, SMON): a különböző rendszerhibák utáni helyreállítást végző folyamat. Az Oracle indításakor és befejeződésekor automatikusan elindul. Más esetben, szabályos időközönként „felébresztik”, hogy megnézze, szükség van-e rá. Ilyenkor az ideiglenes szegmensek már nem használt adatait törli.

Folyamat-felügyelő folyamat (process monitor, PMON): míg az SMON a rendszerhibák után, addig a PMON a felhasználókkal kapcsolatban álló szerverfolyamatok hibái után „takarít”. Ha egy ilyen folyamat nem hajtodik teljesen végre, akkor a PMON a felhasználó megfelelő tranzakcióit, zárait és egyéb foglalt erőforrásait felszabadítja.

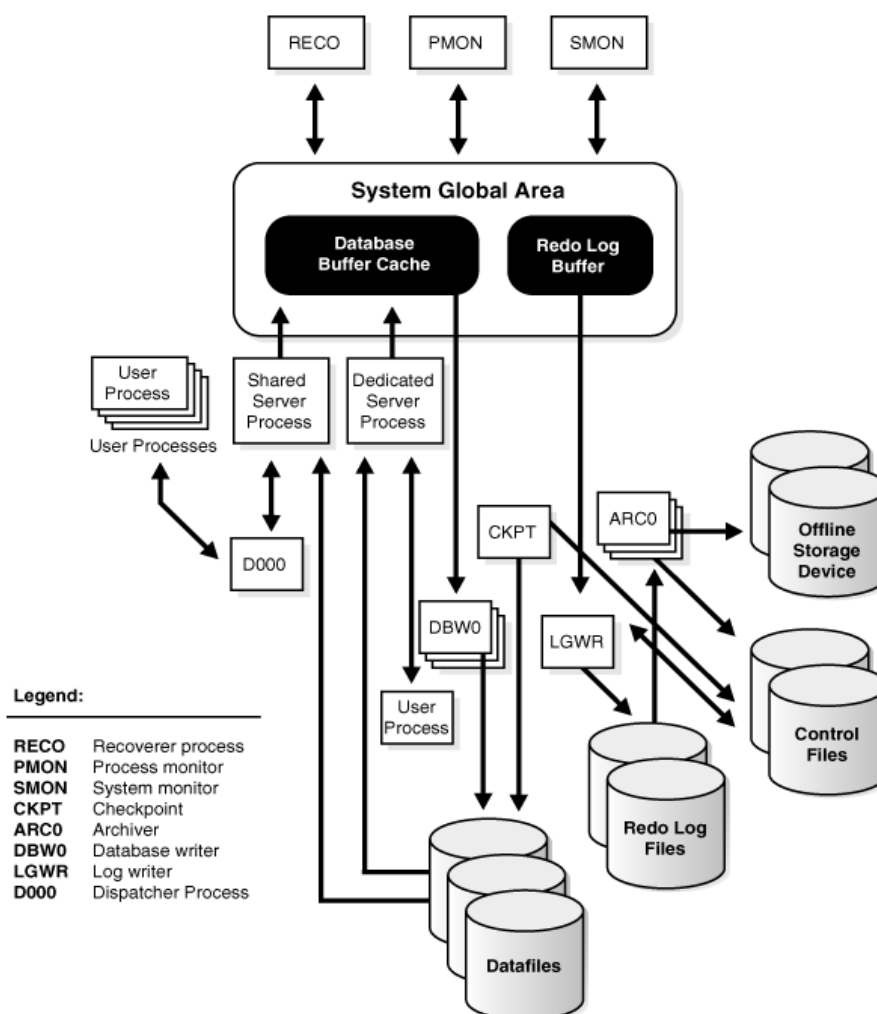
Adatbázis író folyamatok (database writers, DBWn): a szükséges, módosított adatokat írja ki az SGA-ból a háttértárra, a megfelelő adatfájlokba. Legfeljebb 20 ilyen folyamat működhet egyszerre.

Naplókészítő folyamat (log writer, LGWR): a redo-log puffert írja stabil tárba az SGA-ból. Az elvégzett műveleteket az aktív redo-log állományba jegyzi le.

Archívumot készítő folyamat (archiver, ARCn): az aktív, betelt redo-log állományt egy erre a célra kijelölt tárra másolja. A másolat célja biztosítani a rendszerhibák utáni helyreállítást. Legfeljebb 10 ilyen folyamat működhet egyszerre, szükség szerint a LGWR gondoskodik új ARCn folyamatok indításáról.

Zárfolyamatok (lock manager server processes, LMS): Oracle példányok közötti erőforrás-kezelést valósít meg az Oracle Real Application Cluster-ben (kb. Oracle parallel szerver).

Párhuzamosító folyamat (dispatcher, D000): a párhuzamosító feladata, hogy a felhasználói folyamatok között megossza a kisszámú rendszerfolyamatot (többszálúként beállított szerverek esetében (*shared server üzemmód*)). Így ugyanannyi rendszerfolyamattal jóval több felhasználó szolgálható ki. Jellemzően ez a kapcsolódási pont az Oracle szerver és a (kliensen futó) felhasználói folyamatok között. A párhuzamosítóhoz minden esetben SQL*Net (Net8) protokollon keresztül kell csatlakozni.



7. ábra: Az Oracle háttérprogramjai. (Ábra forrása: Oracle 11gR1 Database Concepts, Fig 9-2: Background Processes of a Multiple-Process Oracle Database Instance)

Az egyes folyamatok közötti kapcsolatokat mutatja a 7. ábra.

Összefoglalva: Egy Oracle példány a rendszer és háttérprogramokból, illetve a rendszer által lefoglalt memóriaterületekből (pl. SGA) áll.

Az Oracle és a felhasználói folyamatok (kliensprogramok) mindig Net8 protokollon keresztül kommunikálnak. A Net8 protokoll elfedi a különböző lehetséges hálózatokat és programozói felületeket (viszony, és megjelenítési szintű protokoll). Így a Net8 illeszthető pl. IPX, SPX, IPv4, IPv6, TCP, TCPS hálózatokra egyaránt. Ugyanezen protokoll felelős a nyelvi beállításokért. Azaz: kizárólag a Net8 protokoll *kliensoldali* (felhasználói) beállításaitól függ, hogy az Oracle milyen nyelven írja ki az üzeneteit, a dátumokhoz tartozó mezőneveket, ill. milyen karakterkészlet szerint rendez.

3.5. Az Oracle biztonsága

Az Oracle rendszerhez az Oracle-ben definiált felhasználók csatlakozhatnak. Ezek a felhasználók és a hozzájuk kapcsolódó jogosultságok (első közelítésben) függetlenek az operációs rendszer felhasználóitól, illetve jogosultsági rendszerétől.

Az Oracle jogosultsági rendszere kétlépcsős. Az első csoportba tartoznak a *rendszerjogosultságok* (*system privileges*, pl. a CREATE TABLE a tábla létrehozására, ALTER SESSION a kapcsolat módjának megváltoztatására), amelyekkel a rendszer

egészével kapcsolatos jogokat állíthatjuk be egy-egy felhasználó számára, továbbá a *felhasználói objektumokra vonatkozó jogosultságok (object privileges)*.

Az egyes rendszer- és objektumjogosultságokból összeállítható egy úgynevezett szerep (role). A rendszer tartalmaz néhány előre definiált szerepet; ilyen a DBA (adatbázis rendszergazda, DataBase Administrator), a CONNECT¹² (csatlakozási jog az adatbázis-kezelőhöz) és a további jogosultságokat biztosító RESOURCE „jog” is.

Mivel sokszor kényelmetlen lenne az összes szükséges jogosítványt felsorolni egy-egy feladat elvégzésére, így célszerűbb a jogosítványok mindegyikét tartalmazó szerepet megjelölni. Például ahhoz, hogy az adatbázis lemezterületeit karbantarthassuk, mintegy húszféle rendszerszintű jogosultságra van szükség, azonban a felsorolásuk helyett azt mondjuk, hogy DBA jogokkal kell rendelkezni. A felhasználói adatok és az egyes jogosultságok karbantartásához viszont a CREATE/ALTER/DROP USER jogok is elegendőek.

4. Az Oracle üzemeltetése

Az Oracle a távoli adminisztrációt (is) támogatja különféle eszközök segítségével. Az adatbázis (távoli) karbantartásának központi eleme az Enterprise Manager Grid Control, amely webes felületen az adatbázis-kezelő karbantartásán túl számos infrastrukturális elem (pl. Oracle Virtual Machine Server) felügyeletére is tartalmaz eszközöket. A labor keretében az Enterprise Manager helyett az Oracle SQL Developerbe épített DBA üzemmódot fogjuk megismerni, amely a 3.0 verzióban jelent meg, jelenleg (2015) a 4.0 a stabil kiadás. Az SQL Developer 4.0 Java 1.7-es rendszeren fut az arra alkalmas platformokon.

4.1. Az SQL Developer indítása

Az SQL Developer telepítésére nincs szükség, a letöltött .zip fájl kibontása után az sqldeveloper.exe vagy sqldeveloper.sh fájl futtatásával indítható.

Amennyiben az alkalmazást először indítjuk el, szükség lehet a Java környezet¹³ útvonalának megadására.

Az SQL Developerben három üzemmód-választó fül egyes elemeivel fogunk foglalkozni. A három fül a Connections (definiált adatbázis-kapcsolatok), Reports (az adatainkból vagy az adatbázisról készült különböző kimutatások, „riportok”) és a DBA (egyes adatbázis-adminisztrátori funkciók).

Az adatbázis-kezelőhöz történő csatlakozáshoz a Connections fülön levő zöld + (plusz) ikonra kattintva meg kell adni annak a szervernek az adatait, amelyhez kapcsolódni kívánunk, az alábbiak szerint:

- *Connection name*: tetszőleges név a kapcsolat azonosítására, pl. szglab5
- *Username/Password*: az adatbázis-kezelőhöz történő felhasználói név/jelszó páros. Nem kötelező kitölteni.
- *Save Password*: a jelölőnégyzetet megjelölve a felhasználónév/jelszó párost elmenthetjük (ha a fenti mezőkben megadtuk), így a csatlakozásnál megadásuk már nem lesz szükséges (a HSZK-ban ne jelöljük be a négyzetet).
- Az Oracle fület választva egy Oracle adatbázis-kapcsolat leírását készíthetjük el az alábbiak szerint:
 - *Connection type*: milyen módon csatlakozzunk az adatbázis-kezelőhöz. A Basic üzemmódot választva egyszerűen megadható a szerver címe, portja, és az adatbázis neve (l. a következő vázlatpontokat). Használhatjuk még a TNS

¹² A CONNECT szerep a 10gR1 verzióval bezárólag tipikus felhasználói jogosítványgyűjtemény, amely az adatbázis-kezelőhöz való csatlakozást, néhány típusú objektum létrehozását, használatát teszi lehetővé

¹³ Java Development Kit (JDK) szükséges, Java Runtime Environment (JRE) nem elég.

üzemmódot, ekkor egy ún. TNS-leíró állományra¹⁴ van szükség, és az abban definiált kapcsolatleírók közül választhatunk. Advanced üzemmódban pedig tetszőleges JDBC URL megadására nyílik lehetőség, amelynek érvényessége a felhasználó felelőssége.

- *Role*: meghatározhatjuk, hogy a szervert milyen jogosultságokkal kívánjuk elérni. A mérés során sem a SYSDBA (adatbázis adminisztrátor) sem pedig a (valamivel korlátozottabb jogokat biztosító) SYSOPER szerepet nem fogjuk használni, így a csatlakozás csak „default” (néhány kliensben: „Normal”) módban lehetséges.
- *Hostname*: a szerver DNS-neve, pl. rapid.eik.bme.hu
- *Port*: a szerver port-száma. (Általában 1521 vagy, elsősorban Magyarországon, 1526).
- *SID*: a szerver rendszerazonosítója (System Identifier). Az a név, ahogy az adatbázist az adatbázis-adminisztrátor elnevezi, ez jelen esetben szglab. (Általában nem célszerű 6 karakternél hosszabbra választani.)
- *Service Name*: az adatbázist, mint szolgáltatást azonosítja.
- Amennyiben minden adatot megadtunk, és mentettük a kapcsolatleírót, az meg fog jelenni a képernyő bal oldalán látható fehér területen. Az adatbázis ikonjára kattintva csatlakozhatunk a rendszerhez, felhasználónevünk és jelszavunk begépelése után. (Ezek az adatok az első mérésen kiosztásra kerülnek.)

Amennyiben minden adatot sikeresen adtunk meg, az adatbázis-szervert reprezentáló ikon mellett egy villásdugó jelenik meg, jelezve a sikeres csatlakozást, egyben az ikon melletti „+” jelre kattintva láthatóvá válik az adatbázis objektumait reprezentáló fa-gráf. A munkamenet végeztével kijelentkezni a kapcsolat nevére történő jobb-klikk után a „Disconnect” paranccsal lehetséges (és ajánlott).

A Connections fül mellett található a DBA üzemmódot megtestesítő fül (ha nem látszik, akkor a View menü DBA pontjával hívható elő). Itt a zöld + (plusz) ikonra kattintva engedélyezhetők a kliens DBA üzemmódjához az egyes, már korábban definiált kapcsolatok. A kapcsolat neve melletti „+” jelre kattintva megjelennek a DBA módban elérhető információk és funkciók egy fa-gráf formájában.

A Reports fül az előző kettővel szemben másképp működik: előbb a futtatni kívánt riportot kell kikeresni a megjelenő fa-gráfban, és azután kell megadni, hogy melyik adatbázis-kapcsolaton és (szükség szerint) milyen paraméterekkel fusson.

A továbbiakban a fent említett fa-gráfok egyes elemeit vizsgáljuk meg részletesebben.

4.2. A szerverpéldány beállításai (DBA üzemmód, „Database Configuration”)

Ezen a ponton érhetőek el és változtathatók meg a rendszer működését alapvetően befolyásoló beállítások, az inicializációs paraméterek. A paraméterek megváltoztatása statikus paraméterek esetén a szerverpéldány újraindítását igényli, míg dinamikus paraméterek esetén erre nincs szükség.

Amennyiben egy adatbázison SYSDBA jogosultságunk van (tehát a mi felelősségünk a paraméterek szabályozása) nem célszerű az adatbázis újraindítást erről a felületről elvégezni. (Az adatbázis újraindítása a Connections üzemmódban a kapcsolat nevére történő jobb klikk után a Manage Database pont alatt lehetséges.)

¹⁴ Egy tnsnames.ora fájl, ami a TNS_ADMIN környezeti változó által mutatott könyvtárban (ez az általános megoldás), vagy az SQL Developer Preferences/Database/Advanced/Tnsnames directory helyen van.

4.3. Munkamenetek/Sessions (Reports fül)

A Reports fülön a „Data Dictionary Reports/Database Administration/Sessions” alatt érhetőek el különböző szempontok szerint a munkamenetek adatai: erre a pontra lépve az adatbázissal aktuálisan kapcsolatban lévő, szerveroldalon futó folyamatok megtekintésére és szabályozására nyílik mód (pl. egy megakadt, vagy káros folyamat „kilövésére” a Sessions nevű riportban). Az itt rendelkezésre álló lehetőségek közül a legérdekesebb talán az SQL analízis, ahol egy futó folyamat által kiadott SQL parancsot lehet megtekinteni. Ez a lehetőség a gyakorlatban használható pl. szoftverfejlesztésnél egy kritikus SQL utasítás vagy tárolt program felderítésére és felgyorsítására, a feldolgozási lépések jobb megértésén keresztül.

4.4. Zárak/Locks (Reports fül)

A munkamenetekhez hasonlóan a Reports fülön, a „Data Dictionary Reports/Database Administration/Locks” pont kiválasztásával tekinthetjük meg a rendszerben jelenleg aktív zárat. (Azokat a zárat is, amelyeket pl. az SQL Developer illetve az Oracle belső folyamatai helyeztek el.) A gyakorlatban ez a képernyő a szoftverhibák megkeresését segítheti elő.

Az itt megjelenített információ a V\$LOCK nézetén keresztül kérdezhetőek le SQL felületről.

4.5. Az adatbázis tartalmának kezelése („Schema Manager”)

A Connections fülön, az adatbázis-kapcsolat neve alatt találhatóak a felhasználó saját objektumai típus szerinti bontásban (az „Other Users” alpontban a többi felhasználó objektumai érhetőek el).

Itt nemcsak objektumok egyszerű (varázslók segítségével történő) létrehozására van lehetőség, hanem egyrészt az objektumokhoz kapcsolódó speciális beállítások végezhetőek el (pl. constraintek felvétele, tárolási jellemzők beállítása, triggerek és programok definiálása, amelyet pl. syntax highlighting segít), másrészt pedig az objektumok tartalmának grafikus felületről történő módosítása is lehetséges (pl. egy tábla kitöltése, vagy egy nézet gyors felvétele).

Az Oracle számos, a sémákban elhelyezett objektumok kezelését megkönnyítő eszközt bocsát rendelkezésünkre, általában beépített nézetek formájában. Így pl. az adatbázis-adminisztrátor kikéresheti az összes olyan objektumot, amelynek nevére egy meghatározott karakterlánc illeszkedik.

4.6. Alapvető biztonsági beállítások (DBA üzemmód, Security)

Az adatbázishoz és annak adataihoz való hozzáférés-szabályozás elemei tekinthetőek meg és módosíthatók itt. A megfelelő jogosultságok általában felhasználóhoz kötöttek, így szükséges a felhasználó azonosítása. Erre a célra a méréseken a „hagyományos” felhasználónév-jelszó páros szolgál, de az Oracle lehetőséget biztosít erős titkosításon, illetve nyilvános kulcsú titkosítást használó szoftverarchitektúrán (PKI) keresztül történő azonosításra is (ld. az Oracle Wallet, illetve az Enterprise Security Manager alkalmazást az ún. OCI kliensben). Felhívjuk a figyelmet arra, hogy az Oracle rendszer alapértelmezésben nem használ titkosítást az adatátvitel során (kivéve a jelszavakat), ennek beállítása az adatbázis-adminisztrátor feladata. (Ajánlott az egyszerű SSL alapú titkosítás. Ennek hátránya, hogy megnöveli a kommunikációhoz szükséges sávszélességet, illetve a bejelentkezés időtartamát.)

Elosztott adatbázisrendszerek esetén lehetőség van ehhez idomuló bejelentkezési rendszer megvalósítására (ld. Enterprise Logon Assistant).

A User alpont jobb-klikk Create new parancsával új felhasználót is itt lehet a rendszerhez adni.

A Security ponthoz kapcsolódó fontosabb párbeszédpanel-fülek az alábbiak (nem mindenhol van jelen mindegyik):

- *User*: Itt nyílik lehetőség a felhasználói név és jelszó beállítására, a felhasználó alapértelmezett táblahelyének beállítására, a korábbiakban említett ideiglenes táblahelyek beállítására, illetve a felhasználó engedélyezésére/tiltására.
- *Granted Roles*: a felhasználó szerepeit állíthatjuk be.
- *System privileges*: rendszerszintű privilégiumok szabályozása.
- *Object privileges*: objektumszintű privilégiumok szabályozása.
- *Quotas*: a UNIX rendszerekhez hasonló kvóták beállítása táblahelyenként.
- *Proxy users*: Ez a lehetőség arra szolgál, hogy az egyes speciális feladatok az adatbázist adott felhasználóként el tudják érni.

4.7. Fizikai tárolási paraméterek (DBA üzemmód, Storage)

Az adatbázis fizikai megvalósításának elemeit lehet vele megtekinteni és karbantartani. Alkalmas egyfelől különböző táblahelyek, adatállományok és rollback szegmensek létrehozására, módosítására és szükség esetén ezek törlésére, másfelől a kihasználtságról, a szükséges hely- és tárigényekről kaphatunk részletes információkat. Fontos eleme az ún. High Watermark, amely jelzi az adott objektum létrehozása óta annak valaha előfordult maximális kihasználtságát. Változtatások végrehajtásához DBA jogosultsággal kell rendelkezni.

5. Néhány további Oracle termék

*Pro*nyelvek*: különböző kapcsolódási pontokat, előfordítókat tartalmaz magas szintű programozási nyelvekhez. Létezik – a teljesség igénye nélkül – C, C++, Cobol, Fortran, Pascal, PL/I nyelvekhez.

PL/SQL: Procedurális elemekkel bővített SQL, egyedi az Oracle adatbázis-kezelőre. Tartalmaz számos vezérlési szerkezetet; lehet eljárásokat, függvényeket definiálni, létezik elágazó utasítás (IF), segédváltozókat deklarálhatunk benne stb. Mindezeket tárolhatjuk az Oracle adatbázisban és futtathatjuk akár a szerver, akár a kliensoldalon, az alkalmazásainkból elindítva.

Designer: szintén negyedik generációs sématervező és automatikus kódgeneráló eszköz. Céladatbázis ebben az esetben nem feltétlen Oracle, lehet DB/2, Sybase Adaptive Server, Microsoft SQL Server és bármilyen ODBC kompatibilis adatbázis-kezelő.

Internet File System: Interneten keresztül hozzáférhető szolgáltatások biztosítására alkalmas: távoli adminisztrálást és adatok távoli elérését, SQL utasítások végrehajtását is támogatja Interneten keresztül.

InterMedia: Internetes és vezeték nélküli, nagyrészt multimédia adatokat tartalmazó alkalmazások fejlesztését és kiszolgálását támogató szervizekből álló csomag.

JDeveloper: e-business és internetes tartalomfejlesztő eszköz. Általában magában foglalja az Internet File System és InterMedia csomagokat is. Teljes mértékben Javában írt szoftver, amely tartalmaz egy UML modellezőt és a szoftverfejlesztésekhez számos további segédprogramot.

Reports: az adatbázis-lekérdezések gyors formázására és létrehozására szolgál. Itt is elsősorban az Interneten való megjeleníthetőséget tartották szem előtt, de hozzáférhető XML, CORBA és EJB protokollokon keresztül is.

Warehouse Builder: adattárházak fejlesztésének támogatására kiélezett tervező és fejlesztőrendszer.

Oracle Data Mining option (Oracle Darwin): egyszerűen használható, rejtett összefüggéseket kereső alkalmazás.

6. Függelék

A függelék nem tartozik szorosan a mérés anyagához, nem fogjuk számon kérni a tartalmát. A laborok során nem fogjuk kihasználni az egyes Oracle verziók speciális tulajdonságait. Ahol ezt meg kellett tennünk, ott felhívtuk a figyelmet a változásokra. Mindazonáltal fontosnak tartottuk, hogy a hallgatók érzékeljék a különbségeket, a fejlődési trendeket.

6.1. Az Oracle újabb verzióinak összehasonlítása

Oracle 8

Az Oracle8 az első objektumrelációs adatbázis-kezelő az Oracle sorozatban, amely 1997-ben jelent meg. Leegyszerűsítve ez annyit jelent, hogy az Oracle képes objektumok adatainak és eljárásainak tárolására, megjelent a típus fogalma és lehetőség nyílt a multimédia adatok hatékonyabb kezelésére.

Az adattípusokat illetően a legjelentősebb változás, hogy a LONG helyett a LOB típusokat lehet már használni, ami sokkal dinamikusabb és hatékonyabb. A CHAR és VARCHAR2 maximális mérete is megváltozott, a korábbi 255 ill. 2000 helyett 2000 ill. 4000 lett.

Az alkalmazásfejlesztésekhez kibővítették a JDBC, azaz a Java nyelven keresztüli adathozzáférés képességeit. Az adatbázisból lehetőség van adatbázison kívüli függvényhívásokra is akár HTTP vagy IIOP (egy CORBA szabvány) protokollokon keresztül.

Oracle 8i

Az Oracle-t kifejezetten Internetes alkalmazások támogatására alakították át, megjelenése 1999. A telepítő és a kliensek lényegében Java-alapúak, illetve a beépített Java VM segítségével a szerver maga is képessé vált Java alkalmazások futtatására. Különálló termékként megjelent a WebDB, ami a korábbi Webservert váltotta fel egy sokkal hatékonyabbra. A Webserver PL/SQL segítségével állította elő a HTML oldalak tartalmát. A WebDB alkalmazásban a két dolog felcserélődött, varázslók segítségével rakhatjuk össze a HTML oldalt, a PL/SQL forrás automatikusan generálódik.

A szerver magában foglalja InterMedia csomagot is, amivel multimédia adatok internetes megjelenítését, tárolását, lejátszását és továbbadását is támogatja, de arra is lehetőséget biztosít, hogy különböző lekérdezéseket, riportokat kérjünk le HTML, PDF, Word vagy Excel formában. Mindezek az EDI (Electronic Document Interchange) támogatását szolgálják.

Jelentősen kibővítették a DATE adattípushoz tartozó elemeket, elsősorban a konverziók terén. Például belekerült a fél hónap, 10 nap és a félév fogalma is. De a dátumokat szabadon lehet felüldefiniálni, például az üzleti világ elvárásaink és követelményeinek megfelelően. Az SQL utasítások között megjelent a DROP COLUMN utasítás, azaz lehetőség nyílt oszlopok törlésére. A ROWID típust kiterjesztették UROWID típusná, amely félig logikai kulcs, hiszen a táblák elsődleges kulcsainak kitöltésétől is függ – ezáltal gyorsítva az adathozzáférést.

Oracle 9i

Az adatbázis-kezelő jelentősebb belső átalakításon ment keresztül, amely leginkább az adatbányászat és az adattárházak területeit érintik. A 9i Release1 2001-ben jelent meg. A legfontosabb változás, hogy az Oracle9i SQL-99 kompatibilis lett (korábban csak SQL-92 kompatibilitást biztosítottak).

A korábbi Oracle termékekkel ellentétben, ebben a verzióban már lehetőség van az SGA területének és tartalmának dinamikus, azaz futási időben elvégezhető módosítására – ezek is bekerültek a megfelelő SQL utasítások közé. Ez változat már a különböző blokkméretek kezelésére is lehetőséget biztosít, sőt, akár külső adatbázisbeli felhasználói objektumok elérésére is nyújt interfészt.

Oracle 10g

Az Oracle 10g első kiadása (R1) 2003-ban, a Release 2 2005-ben jelent meg, a nevében a „g” a „Grid computing” kifejezésből származik. A nevével összhangban az elosztott erőforrások és szolgáltatások egy logikai egységként történő kezelésében tartalmaz számos előrelépést, valamint diagnosztikai és tuning-fejlesztéseket. Az elosztott működés nem a 10g újdonsága: az Oracle adatbázis-kezelő már korábban is tartalmazta a Real Application Clusters¹⁵ technológiát.

A 10g verzió számos további újdonsága a logikai és fizikai tárolási rétegek köré csoportosul. A teljesség igénye nélkül megemlítnék néhányat közülük. Az automatikus tárkezelés (Automatic Storage Management, ASM) egy logikai kötetkezelési réteg, amely az adatbázis-kezelő alatti platform tárkezelő mechanizmusaitól független kötetkezelést tesz lehetővé. Míg az Oracle korábbi verzióiban táblahelyet csak azonos platformon futó adatbázisok között lehetett másolni, a 10g-től kezdődően erre különböző platformok esetén is van lehetőség.

Az újonnan megjelent lomtár (recycle bin) a törölt adatbázis-objektumok tárolási helye (amennyiben engedélyezett a szerverpéldány szintjén), ahonnan azok szükség szerint visszaállíthatók. Az objektumok szintje mellett lehetőség van az adatok szintjén is a visszaállításra az ún. flashback technológiával, amely a 10g-ben SQL utasítások szintjére került (korábban egy PL/SQL csomag volt), és lehetőséget biztosít adatbázis és tábla szinten is a visszaállításra, illetőleg a korábbi állapot lekérdezésére.

A rendszer teljesítményanalíziséhez az AWR (Automatic Workload Repository, automatikus terhelés-repozitórium) rendszeres időközönként feljegyzi a fontosabb teljesítmény-paramétereket, amelyet az ADDM (Automatic Database Diagnostics Monitor, automatikus adatbázis-diagnosztikai monitor) komponens dolgoz fel és tesz elérhetővé.

Egy rendszer fejlődése során időnként elkerülhetetlen, hogy a megjelenő új komponensek mellett korábbiak tűnjenek el vagy változzanak meg. Az Oracle 10g lekérdezés-optimalizálója hivatalosan már nem támogatja a szabály-alapú optimalizálást, és ennek megfelelően automatikusan gyűjti és frissíti a költség-alapú optimalizálót segítő objektumstatisztikai adatokat.

Oracle 11g

2007-ben jelent meg az Oracle 11gR1, a Release 2 pedig 2009-ben. Az R1 kiadás a SQL:2003, míg az R2 kiadás az SQL:2008 szabványok kötelező ún. Core részével nagyrészt kompatibilis. Számos apró újítása a hatékonyabb erőforrás-kihasználást célozza meg mind teljesítmény (pl. statisztika-gyűjtés, lekérdezéseredmény cache), mind tárhely (pl. tömörítés az egyedi DML műveletek eredményében is), mind DBA-erőforrások (pl. automatikus memória-tuning, terhelés-profilok rögzítése és visszajátzása: Real Application Testing) tekintetében.

A táblák a 11g-től kezdődően tartalmazhatnak ún. virtuális oszlopokat, amely a nézetekhez hasonlóan teszik lehetővé SQL kifejezésekkel definiált oszlopok megadását a rekord többi mezőjének értéke alapján. Ez a virtuális oszlop a tábla „teljes jogú” oszlopa lekérdezésekkor és indexek építésekor, ill. a tábla ún. particionálásakor.

Oracle 12c

2013-ban jelent meg az Oracle 12cR1, amely már nagyrészt SQL:2011-es szabvány-kompatibilitást nyújt annak kötelező, ún. Core részével. A verziószámában megjelenő „c” a felhő-alapú számítástechnikára utal (cloud-computing). Ehhez kapcsolódó talán

¹⁵ Oracle Real Application Clusters (RAC), a 9i előtti verziókban Oracle Parallel Server (OPS). Elosztott tranzakciófeldolgozási képességeket megvalósító technológia, amelyben több szerverpéldány (instance) kezeli a közös háttértáron elhelyezett adatbázist. A technológia jól skálázható, magas rendelkezésre állású logikai adatbázis-szerveret biztosít.

leglényegesebb újdonság az ún. multitenant architektúra, amely egy konténer adatbázisszerverből, és benne felhasználói adatbázisokból (pluggable database, PDB) áll. Ily módon a szerverpéldány (instance) közös az egy konténerhez tartozó adatbázisok között. Ez az adatbázisok menedzselését könnyíti meg, hiszen az olyan szerverpéldány-szintű beállítások, mint a replikáció, mentés-csoportok, Real Application Cluster (RAC) az összes adatbázisra érvényesek lesznek, és a DBMS szoftverfrissítéseit is csak egyszer kell telepíteni. Az egyes PDB-k könnyen átcsatolhatók a különböző konténerek között. Szintén a 12c újítása, hogy egyes szerverfolyamatok immár többszálú működésre is beállíthatók.

Az architekturális változtatáson túl a fejlesztések egy jelentős hányada azt célozza, hogy minél egyszerűbb legyen más adatbázis-kezelő rendszerekről Oracle Database-re „portolni” az alkalmazásokat. Ilyen SQL-fejlesztések pl. a top-N lekérdezések megfogalmazására szolgáló szintaxis-kiegészítés a select utasításban, vagy a számláló-jellegű, automatikus kitöltésű azonosító mező generálását kényelmesebbé tevő oszlop-beállítás (identity_clause illetve *<sequence>.nextval*, mint alapérték). Egy egzotikusabb példa ugyanebből a körből, hogy a MySQL C nyelvű API-val kompatibilis felületen keresztül közvetlenül elérhető az Oracle Database adatbázis a liboramysql meghajtó segítségével.

További fejlesztések között említjük a select utasítás idősor-jellegű mintaillesztési képességgel történő felruházását (row_pattern_clause), vagy az adaptív lekérdezési tervek készítését. Ennek lényege, hogy a statisztikák alapján kiválasztott illesztési algoritmust bizonyos feltételek mellett futásidőben megváltoztathatja a DBMS, ha a végrehajtás során, a valódi adatok egy részét feldolgozva úgy találja: jobbat is választhat.

II. labor: Az SQL nyelv

Szerzők: Kiss István anyagát kiegészítette Gajdos Sándor, Ungváry Ferenc

1.	AZ SQL TÖRTÉNETE.....	26
2.	A NYELV JELENTŐSÉGE	26
3.	A NYELV DEFINÍCIÓJA	27
4.	TÁBLÁK LÉTREHOZÁSA, TÖRLÉSE.....	27
4.1.	Táblák létrehozása	27
4.2.	Táblák törlése	28
5.	ADATOK BEVITELE, TÖRLÉSE, MÓDOSÍTÁSA.....	28
5.1.	Adatok bevitele.....	28
5.2.	Adatok törlése	29
5.3.	Adatok módosítása.....	29
6.	LEKÉRDEZÉSEK.....	29
6.1.	Vetítés (projection)	30
6.2.	Kizárás (restriction)	30
6.3.	Összekapcsolás (join).....	31
6.4.	Oszlopfüggvények	32
6.5.	Egymásba ágyazott lekérdezések.....	33
6.6.	Csoportosítás.....	34
6.7.	Rendezés.....	34
6.8.	Halmazműveletek.....	34
6.9.	Hierarchikus kapcsolat lekérdezés.....	35
6.10.	Egyéb nem szorosan az SQL nyelvhez tartozó utasítások	35
7.	NÉZETEK	35
8.	INDEXEK.....	36
8.1.	Indexek létrehozása.....	36
8.2.	Indexek törlése	36
9.	JOGOSULTSÁGOK DEFINIÁLÁSA.....	36
10.	TÁBLADEFINIÓK MÓDOSÍTÁSA	37
11.	TRANZAKCIÓK.....	37
12.	PÁRHUZAMOS HOZZÁFÉRÉS SZABÁLYOZÁSA	38
13.	KONZISZTENCIAFELTÉTELEK	38

1. Az SQL története

- 1974–75-ben kezdték a kifejlesztését az IBM-nél, az „eredeti” neve SEQUEL (Structured English QUery Language);
- 1979-től több cég (pl. IBM, ORACLE Corp.) kereskedelmi forgalomban kapható termékeiben;
- 1987-től ANSI szabvány.

2. A nyelv jelentősége

- Szabvány, amelyet jelenleg csaknem minden relációs adatbázis-kezelő alkalmaz (kisebb-nagyobb módosításokkal);
- tömör, felhasználó közeli nyelv, alkalmas hálózatokon adatbázis-kezelő szerver és kliensek közötti kommunikációra;
- nem procedurális programozási nyelv (legalábbis a lekérdezéseknél).

3. A nyelv definíciója

A leírás az ORACLE adatbázis-kezelő SQL dialektusát ismerteti, ez többé-kevésbé megfelel az egyéb termékekben található nyelv variációknak. A nyelv termék- illetve hardverspecifikus elemeit nem, vagy csak futólag ismertetjük.

A nyelv utasításait a következő csoportokra oszthatjuk:

- adatleíró (DDS – Data Definition Statement)
- adاتمódosító (DMS – Data Manipulation Statements)
- lekérdező (Queries)
- adatelérést vezérlő (DCS – Data Control Statements)

A nyelvben – a szöveg literálok kivételével – a kis- és nagybetűket nem különböztetjük meg. A megadott példáknál a könnyebb érthetőség miatt a nyelv alapszavait csupa nagybetűvel, míg a programozó saját neveit kisbetűvel írjuk.

A parancsok több sorba is átnyúlhatnak, a sorokra tördelésnek nincs szemantikai jelentősége. Az SQL parancsokat pontosvessző zárja le.

4. Táblák létrehozása, törlése

4.1. Táblák létrehozása

Új táblát a

```
CREATE TABLE <táblanév>
  (<oszlopnév> <típus> [NOT NULL]
  [, <oszlopnév> <típus> [NOT NULL] , ...]
);
```

paranccsal lehet létrehozni. A lehetséges adattípusok implementációnként változhatnak, általában a következő adattípusok megtalálhatók:

CHAR (n)	max. <i>n</i> karakter hosszú szöveg (karaktersorozat);
LONG	mint a CHAR, de hosszára nincs felső korlát (nagyon nagy);
NUMBER (n)	az előjellel együtt, max <i>n</i> karakter széles egész szám;
DATE	dátum (és általában időpont)

Ha valamely oszlop definíciója tartalmazza a NOT NULL módosítót, a megfelelő mezőben mindig valamilyen legális értéknek kell szerepelnie.

A SCOTT demo adatbázis néhány tábláját (kisebb módosításokkal) a következőképpen lehetne létrehozni:

```
CREATE TABLE customer (
  custid      NUMBER (6) NOT NULL,
  name        CHAR (45),
  address     CHAR (40),
  city        CHAR (30),
  state       CHAR (2),
  zip         CHAR (9),
  area        NUMBER (3),
  phone       CHAR (9),
  repid       NUMBER (4),
  creditlimit NUMBER (9,2),
  comments    LONG);
CREATE TABLE ord (
  ordid      NUMBER (4) NOT NULL,
  orderdate  DATE,
  commplan   CHAR (1),
  custid     NUMBER (6) NOT NULL,
```

```

        shipdate    DATE,
        total       NUMBER (8,2));
CREATE TABLE item (
        ordid       NUMBER (4) NOT NULL,
        itemid      NUMBER (4) NOT NULL,
        prodid      NUMBER (6),
        actualprice  NUMBER (8,2),
        qty         NUMBER (8),
        itemtot      NUMBER (8,2));
CREATE TABLE product (
        prodid      NUMBER (6),
        descrip     CHAR (30),
        partof      NUMBER (6),
        comments    LONG);
CREATE TABLE price (
        prodid      NUMBER (6) NOT NULL,
        stdprice    NUMBER (8,2),
        minprice    NUMBER (8,2),
        startdate   DATE,
        enddate     DATE),
        currency_code CHAR (3));
CREATE TABLE currency (
        currency_code CHAR (3) NOT NULL,
        currency_name CHAR (45),
        is_european CHAR (1));

```

4.2. Táblák törlése

Táblát törölni a

```
DROP TABLE <táblanév>;
```

utasítással lehet.

5. Adatok bevitele, törlése, módosítása

5.1. Adatok bevitele

A `CREATE TABLE` utasítással létrehozott táblák kezdetben üresek. Új sort a következő utasítással lehet felvenni:

```

INSERT INTO <táblanév> [(<oszlopnév> [, <oszlopnév>, ...])]
VALUES (<kif1> [, <kif2> , ...]);

```

Amennyiben nem adjuk meg az oszlopok nevét, akkor a – tábla deklarálásánál megadott sorrendben – minden mezőnek értéket kell adni, ha viszont megadtuk az egyes oszlopok neveit, akkor csak azoknak adunk értéket, mégpedig a felsorolásuk sorrendjében, a többi mező `NULL` értékű lesz.

Az egyes mezőknek `NULL` értéket is adhatunk, ha a deklaráció alapján az adott mezőnek lehet `NULL` értéke.

Figyelem: Amennyiben olyan oszlopnak akarunk `NULL` értéket adni, amelynek nem lehet `NULL` értéke, úgy a parancsvégrehajtás hibával leáll!

Egy ilyen paranccsal egyszerre csak egy sort tudunk felvenni a táblába.

Két új termék bevitele pl. az alábbi SQL utasítással lehetséges:

```

INSERT INTO product (prodid, descrip)
VALUES (111111, 'Gozeke');
INSERT INTO product
VALUES (111112, 'Oracle 6.0', NULL,
        'Relacios adatbazis-kezoelo');

```

5.2. Adatok törlése

Sorokat kitörölni a

```
DELETE FROM <táblanév>
[WHERE <logikai kifejezés>;
```

paranccsal lehet.

Ha a WHERE hiányzik, akkor a tábla valamennyi sorát, egyébként csak a logikai kifejezés által kiválasztott sorokat törli.

Ha az előzőekben felvett adatok közül a 111112 azonosítójú (prodid) törölni akarjuk, akkor ezt a következő módon lehetséges:

```
DELETE FROM product
WHERE prodid = 111112;
```

5.3. Adatok módosítása

Sorokban az egyes mezők értékeit az

```
UPDATE <táblanév>
SET <oszlopnév> = <kifejezés> [, <oszlopnév> = <kifejezés> , ...]
[WHERE <logikai kifejezés>;
```

paranccsal lehet módosítani.

Ha a WHERE hiányzik, akkor a parancs a tábla valamennyi sorában módosít, egyébként csak a logikai kifejezés által kiválasztott sorokban.

Ha a PRODUCT táblában a “Gozeke”-hez megjegyzést akarunk fűzni, akkor ezt a következőképpen lehet megoldani:

```
UPDATE product
SET comments = 'mezogazdasagi gep'
WHERE descrip = 'Gozeke';
```

6. Lekérdezések

A lekérdezések általános szintaxisa a következő:

```
SELECT <jellemzők>
FROM <táblák>
[WHERE <logikai kifejezés>]
[<csoportosítás>]
[<rendezés>;
```

A lekérdezés művelete eredményül egy újabb táblát állít elő – persze lehet, hogy az eredménytáblának csak egy oszlopa és csak egy sora lesz. Az eredménytábla a lekérdezés után megjelenik vagy a tábla felhasználható egy másik parancsba beágyazva (pl. halmazműveletek).

A <jellemzők> definiálják az eredménytábla oszlopait,

a <táblák> adják meg a lekérdezésben résztvevő táblák nevét,

a <logikai kifejezés> segítségével “válogathatunk” az eredmény sorai között,

a <csoportosítás> az eredménytábla sorait rendezi egymás mellé,

a <rendezés> a megjelenő sorok sorrendjét határozza meg.

Nézzük meg, hogy a lekérdezés műveletével hogyan lehet megvalósítani a relációs algebra alapl műveleteit.

6.1. Vetítés (projection)

```
SELECT <jellemzők> FROM <táblanév>;
```

A vetítés művelete egy táblából adott oszlopokat válogat ki. A <jellemzők> között kell felsorolni a kívánt oszlopokat.

Például ha a termékek kódjára és nevére vagyunk kíváncsiak:

```
SELECT prodid, descrip FROM product;
```

minden oszlop kiválasztása:

```
SELECT * FROM product;
```

Ha a <jellemzők>-ben csak a *-ot adjuk meg, akkor az adott tábla minden oszlopát kiválasztja. (Az egész táblát megjeleníti.)

A <jellemzők> közé nemcsak a *FROM* mögött megadott tábla oszlopainak nevét lehet megadni, hanem használhatjuk az SQL beépített műveleteit is: pl. egyszerű aritmetikai kifejezéseket új érték előállítására, oszlopfüggvényeket (lásd később!).

Az áfás ár számítása:

```
SELECT prodid, startdate, 1.25*stdprice FROM price;
```

A lekérdezésben az 1.25*stdprice-nak külön nevet is adhatunk az *AS* kulcsszó segítségével (oszlopszinonima), amelyre az *ORDER BY* (rendezés, ld. 6.7) klózbán, vagy beágyazott lekérdezés esetén a beágyazó kontextusban hivatkozhatunk. Például:

```
SELECT prodid, startdate, 1.25*stdprice AS pricewithtax FROM price;
```

A kiválasztott oszlopokat tartalmazó táblákban lehetnek azonos sorok, ami ellentmond a relációs táblák egyik alapvető követelményének. Ennek ellenére a *SELECT* utasítás nem szűri ki automatikusan az azonos sorokat, mert ez túlságosan időigényes művelet. A programozónak kell tudnia, hogy az előállított táblákban lehetnek-e (zavarnak-e) ilyen sorok. Ha kell, a következőképpen szűrhetjük ki ezeket:

Az összes különböző terméknev:

```
SELECT DISTINCT descrip FROM product;
```

6.2. Szelekció (selection)

A szelekció műveleténél a tábla sorai közül válogatunk a *WHERE* segítségével. A <logikai kifejezés> igaz értékeinek megfelelő sorok kerülnek be az eredménytáblába.

A kifejezések elemi összetevői:

literálok különböző típusú értékekre: számok, szöveg, dátum;

oszlopok nevei;

a fenti elemekből elemi adatközpontokkal képzett kifejezések

számoknál aritmetikai műveletek,
 aritmetikai függvények;

szövegeknél SUBSTR(), INSTR(), UPPER(), LOWER(), SOUNDEX(), ...;

dátumoknál +, -, konverziók;

halmazok pl.: (10, 20, 30);

zárójelek között egy teljes *SELECT* utasítás (egymásba ágyazott lekérdezések).

A fenti műveletekkel képzett adatokból logikai értéket a következő műveletekkel állíthatunk elő:

relációk <, <=, =, !=, >=, >;

intervallumba tartozás BETWEEN ... AND ...;

NULL érték vizsgálat IS NULL, IS NOT NULL;

halmaz eleme IN <halmaz>;
szövegvizsgálat
mintával összevetés ... LIKE <minta>, ahol
 % a tetszőleges, akár nulla hosszúságú karaktersorozat,
 _ a tetszőleges, pontosan egy karakter jelzése.

Végül a logikai értékeket a zárójelezéssel, illetve az AND, OR és NOT műveletekkel lehet tovább kombinálni.

A 2000 dollárnál magasabb árak:

```
SELECT prodid, startdate, stdprice FROM price
WHERE stdprice > 2000;
```

A 2000 dollárnál magasabb áfás árak növekvő sorrendben:

```
SELECT prodid, startdate, 1.25*stdprice AS pricewithtax
FROM price
WHERE 1.25*stdprice > 2000
ORDER BY pricewithtax;
```

Az 1994. március 8.-án érvényes árak:

```
SELECT prodid, stdprice FROM price
WHERE '08-mar-94' BETWEEN startdate AND NVL(enddate, '31-dec-94');
```

Az NVL(<oszlopnév>,<érték>) azt biztosítja, hogy amennyiben enddate értéke NULL lenne, akkor azt 31-dec-94-gyel helyettesíti.

A 2000 dollárnál alacsonyabb árak, ahol nincs minimális ár:

```
SELECT prodid, startdate, stdprice FROM price
WHERE stdprice < 2000 AND minprice IS NULL;
```

6.3. Illesztés (join)

A természetes illesztés műveleténél két vagy több tábla soraiból hozunk össze egy-egy új rekordot akkor, ha a két sor egy-egy mezőjének értéke megegyezik. A SELECT kifejezésben a <táblák>-ban kell megadni az érintett táblák neveit (vesszővel elválasztva), a WHERE mögötti logikai kifejezés definiálja azokat az oszlopokat, amely értékei szerint történik meg az illesztés.

Az egyes termékek neve, árai és az árak érvényességi ideje:

```
SELECT product.descrip, price.*
FROM product, price
WHERE product.prodid=price.prodid;
```

Látható, hogy mindkét felhasznált táblában azonos az illesztést megvalósító oszlop neve, a WHERE-t követő logikai kifejezésben az oszlop neve mellé meg kell adni a tábla nevét is. Hasonló helyzet előfordulhat a SELECT-et követő <jellemzők> között is.

Előfordulhat, hogy az ilyen lekérdezésben egyes sorok nem jelennek meg, mert az adott sorhoz a másik táblában nem található illeszthető sor. Ez lehet nem kívánatos eredmény, azonban az SQL-ben lehetőség van arra is, hogy ezeket a sorokat is illesszük, azaz az összekapcsolás során a másik táblához hozzárendelünk egy üres sort is. Ezt külső illesztés hívjuk.

A módosított példa a következőképpen néz ki:

```
SELECT product.descrip, price.stdprice, price.startdate
FROM product, price
WHERE product.prodid = price.prodid(+);
```


A (+) jelzi azt a táblát, amelyikben ha nincs a másik táblához illeszkedő sor, akkor is kell egy üres mezőket tartalmazó sort hozzávenni a másik táblához.

Külső illesztésnél a (+) jelet ki kell tenni a NULL értékekkel kiegészített tábla WHERE klózban szereplő valamennyi előfordulása után, kivéve, ha az adott feltétel egy újabb külső illesztést ír le. Ilyenkor a „kiegészítendő” táblát nem szabad (+) jellel jelölni. Az alábbi példa egy product -> price -> currency külső illesztés-láncot mutat. Az illesztési feltételekben csak a kiegészítő táblát jelöltük (+) jellel, míg a kiegészítendő tábla jelöletlen, l. az (1) kifejezés bal oldalán. A szűrési feltételekben a külső táblák összes előfordulása jelölendő, l. a (2) kifejezés bal oldalán.

```
SELECT product.descrip, price.stdprice, price.startdate
FROM product, price, currency
WHERE -- illesztési feltételek
      product.prodid = price.prodid(+)
      -- (1) a price táblánál nincs (+)
      AND price.currency_code = currency.currency_code(+)
      -- szűrések
      AND product.partof = 'fülke'
      -- (2) a price táblánál is van (+)
      AND price.minprice(+) > 300
      AND currency.is_european(+) = 'Y'
;
```

Az illesztésnél lehet ugyanarra a táblára többször is hivatkozni.

Az azonos nevű termékek (páronként):

```
SELECT a.descrip, a.prodid, b.prodid
FROM product a, product b
WHERE a.descrip = b.descrip AND a.prodid < b.prodid;
```

A többször használt tábla oszlopainak megkülönböztetésére a táblákat a FROM részben lokális névvel látjuk el. Lokális neveket természetesen különböző táblák esetén is használhatunk.

Az egyesítés mellett egyidejűleg más logikai kifejezéseket is használhatunk.

6.4. Oszlopfüggvények

A lekérdezés eredményeként előálló táblák egyes oszlopaiban lévő értékeken végrehajthatunk a szokásos nyelven ciklussal kifejezhető műveleteket, amelyek egyetlen értéket állítanak elő. Ilyenek:

AVG()	átlag,
SUM()	összeg,
COUNT()	darabszám,
MAX()	maximális érték,
MIN()	minimális érték

A 1994 január 1-től induló árak átlaga:

```
SELECT AVG(stdprice) FROM price WHERE startdate = '01-jan-1994';
```

Hány termék van:

```
SELECT COUNT(*) FROM product;
```

Hány különböző nevű termék van:

```
SELECT COUNT(DISTINCT descrip) FROM product;
```

Átlagos minimális ár:

```
SELECT AVG(NVL(minprice, stdprice)) FROM price;
```

Ha nem oszlopfüggvény eredményéből (és nem konstansból) származó érték is része az eredménytáblának, akkor csoportosítani kell ezen értéklista szerint. A csoportosítás szükséges akkor is, ha a programozó biztos benne: az összes kiválasztott sorban azonosak az értékek.

Például írhatnánk:

```
SELECT COUNT(*), AVG(stdprice) FROM price;
```

de hibás

```
SELECT COUNT(*), descrip FROM product;
```

Az alábbi lekérdezés annak ellenére hibás, hogy a WHERE feltétel garantálja: a startdate értéke minden sorban azonos.

```
SELECT startdate, AVG(stdprice) FROM price
WHERE startdate = '01-jan-94';
```

Ehelyett írhatnánk például:

```
SELECT startdate, AVG(stdprice) FROM price
WHERE startdate = '01-jan-94'
GROUP BY startdate;
```

(a GROUP BY jelentését ld. az 6.6 szakaszban)

6.5. Egymásba ágyazott lekérdezések

A WHERE utasítás mögött állhat egy teljes SELECT utasítás is.

Az 1994. utáni árral rendelkező termékek listája:

```
SELECT prodid, descrip FROM product;
WHERE prodid IN
  (SELECT prodid FROM price
   WHERE startdate >= '01-jan-94');
```

Azaz először kiválasztjuk az árak táblából azon termékazonosítókat, amelyekhez 1994-es dátum tartozik, majd azt vizsgáljuk, hogy az ezekből képzett halmazban található-e az adott termék azonosítója. Mellesleg ugyanezt a listát megkaphatnánk az egyesítés műveletével is:

```
SELECT prodid, product.descrip
FROM product, price
WHERE product.prodid = price.prodid
AND price.startdate >= '01-jan-94';
```

Vegyük észre, hogy a kettő mégsem teljesen ekvivalens: ha egy terméknek az ára 1994-ben megváltozott, akkor az egyesítéssel való lekérdezés minden ár-bejegyzéshez generál egy sort, ellentétben beágyazott lekérdezés fenti formájával.

A beágyazott lekérdezés vagy egyetlen értéket – azért mert egyetlen megfelelő sor egyetlen oszlopát választottuk ki, illetve oszlopfüggvényt használtunk –, vagy több értéket – több sort – állít elő. Az előbbi esetben a SELECT értékét az elemi értékekkel azonos módon használhatjuk. Több érték egy halmazt jelent, tehát a halmazműveleteket használhatjuk. A korábban megismert IN() – eleme – művelet mellett használható az ANY() illetve az ALL() művelet, ahol a kívánt reláció a halmaz legalább egy, illetve valamennyi értékére igaz.

Legmagasabb árú termékek (lehet, hogy több van!):

```
SELECT prodid, stdprice FROM price
WHERE stdprice >= ALL (SELECT stdprice FROM price);
```

illetve ugyanez a példa oszlopfüggvény felhasználásával:

```
SELECT prodid, stdprice FROM price
WHERE stdprice = (SELECT MAX(stdprice) FROM price);
```

6.6. Csoportosítás (grouping)

Az oszlopfüggvények a teljes kiválasztott táblára – minden sorra – lefutnak. Gyakran célszerű lenne a kiválasztott sorokat valamilyen szempont szerint csoportosítani és az oszlopfüggvényeket az egész tábla helyett ezekre a csoportokra alkalmazni.

Legmagasabb ár ugyanazon naptól:

```
SELECT startdate, MAX(stdprice) FROM price  
GROUP BY startdate;
```

Természetesen az oszlopfüggvények használatához hasonlóan a `SELECT <jellemzők>` között csak a csoportosítás alapját képező oszlop neve, illetve a csoportokra alkalmazott oszlopfüggvények szerepelhetnek.

A csoportosítás után az eredményből bizonyos csoportok kihagyhatók.

Maximális árak azonos napon az 1000 és 3000 dollár közötti tartományban:

```
SELECT startdate, MAX(stdprice) AS maxprice FROM price  
GROUP BY startdate  
HAVING MAX(stdprice) BETWEEN 1000 AND 3000;
```

A `HAVING` mögött természetesen csak egy-egy közös jellemzőire vonatkozó értékek – a csoportosítás alapját képező oszlop értéke, vagy oszlopfüggvények eredménye – szerepelhet. Természetesen a csoportosítás előtt azért a `WHERE` feltételek használhatók. Célszerű – gyorsabb – `WHERE` feltételeket alkalmazni mindenhol, ahol csak lehet, és a `HAVING` szerkezetet csak akkor alkalmazni, amikor a teljes csoporttól függő értékeket akarjuk vizsgálni.

6.7. Rendezés (sorting)

Az eddig tárgyalt lekérdezések eredményében a sorok „véletlenszerű” – a programozó által nem megadható – sorrendben szerepeltek. A sorrendet az `ORDER BY` által megadott rendezéssel lehet szabályozni. A rendezés több oszlop értékei szerint is történhet, ilyenkor az először megadott oszlop szerint rendezünk, majd az itt álló azonos értékek esetében használjuk a következőnek megadott oszlop(ok) értékét. Minden egyes oszlop esetében külön meg lehet adni a rendezés „irányát”, amely alapesetben emelkedő, de a `DESC` módosítóval csökkenő rendezés írható elő.

Az 11111-es termék ára (időrendben):

```
SELECT stdprice, startdate FROM price WHERE prodid = 111111  
ORDER BY startdate;
```

Minden termék valaha elért legmagasabb ára csökkenő sorrendben:

```
SELECT prodid, MAX(stdprice) FROM price  
GROUP BY prodid  
ORDER BY MAX(stdprice) DESC;
```

6.8. Halmazműveletek

Az egyes lekérdezések által előállított táblák halmazként is felfoghatók, az SQL nyelv ezen táblák kombinálására tartalmaz halmazműveleteket is. Ilyenek:

UNION	unió,
INTERSECT	metszet,
MINUS	különbség,
IN	tartalmazás

A műveleteket két `SELECT` utasítás közé kell írni. (Lásd korábban az egymásba ágyazott lekérdezéseknél!)

6.9. Hierarchikus kapcsolat lekérdezés

A relációs táblák segítségével le tudunk írni hierarchikus kapcsolatokat a különböző sorok között.

Például a

```
SELECT descrip, prodid, partof
FROM product
CONNECT BY PRIOR prodid = partof
START WITH descrip = 'gozeke';
```

utasítás kiírja a gőzeke alkatrészeit az utolsó kerékig lebontva.

A CONNECT BY klózban megadott feltételt kielégítő rekordpárok a hierarchiában szülő-gyerek viszonyban vannak. A PRIOR kulcsszóval jelölt kifejezés a szülő rekordon értelmezett. Így a példában a PRIOR prodid pl. a fülke azonosítója, míg a fülkeajtó rendszerbe illeszkedését írja le a partof attribútum.

6.10. Egyéb nem szorosan az SQL nyelvhez tartozó utasítások

Nem tartoznak szorosan az SQL nyelvhez, de a legtöbb rendszer tartalmaz olyan utasításokat, amelyekkel a lekérdezések által előállított táblázatok megjelenését – pl.: az oszlopok neveit, szélességét, adatformátumát, illesztését, tördelését stb. – definiálhatjuk. Lásd Server SQL Language Reference Manual helpet!

7. Nézetek

A nézetek olyan virtuális táblák, amelyek a fizikai táblákat felhasználva a tárolt adatok más és más logikai modelljét, csoportosítását tükrözik. Nézetek a

```
CREATE VIEW <nézetnév> [(<oszlopnév> [, <oszlopnév>, ...])]
AS <lekérdezés>;
```

paranccsal készíthetők.

A lekérdezésre az egyedüli megkötés, hogy rendezést nem tartalmazhat. Amennyiben nem adunk meg oszlopneveket, a nézetek oszlopai a SELECT után felsorolt oszlopok neveivel azonosak. Meg kell viszont adni az oszlopneveket, ha a SELECT számított értékeket is előállít.

Például az alábbi nézet megmutatja minden termék aktuális árát:

```
CREATE VIEW prods AS
SELECT product.prodid, product.descrip pdescrip,
       x.stdprice sprice, x.minprice mprice
FROM product, price x
WHERE x.prodid = product.prodid
AND x.startdate >= all (
    SELECT startdate
    FROM price i
    WHERE i.prodid = x.prodid);
```

A nézetek a lekérdezésekben a táblákkal megegyező módon használhatók. Jelentőségük, hogy az adatok más modelljét fejezik ki, felhasználhatók a tárolt információ részeinek elrejtésére, pl. különböző felhasználók más-más nézeteken keresztül szemlélhetik az adatokat. A nézet általában csak olvasható, az adatmódosító műveletekben csak akkor szerepelhet, ha egyetlen táblából keletkezett és nem tartalmaz számított értékeket.

Nézetet törölni a

```
DROP VIEW <nézetnév>;
```

utasítással lehet.

A fenti nézetet az alábbi utasítással törölhetjük:

```
DROP VIEW prods;
```

8. Indexek

Az indexek a táblákban való keresést gyorsítják meg.

8.1. Indexek létrehozása

Egy indexet a

```
CREATE [UNIQUE] INDEX <indexnév>  
ON <táblanév> (<oszlopnév> [, <oszlopnév> , ...]);
```

utasítással lehet létrehozni.

Az indexeket az adatbázis-kezelő a táblák minden módosításnál felfrissíti. Amennyiben valamelyik indexet az `UNIQUE` kulcsszóval definiáltuk, a rendszer biztosítja, hogy az adott oszlopban minden mező egyedi értéket tartalmaz. Lehetséges több oszlopot egybefogó, kombinált indexek létrehozása is. Az indexek a létrehozásuk után a felhasználó számára láthatatlanok, csak éppen bizonyos lekérdezéseket gyorsítanak. Indexeket azokra az oszlopokra érdemes definiálni, amelyek gyakran szerepelnek keresésekben. Index nélkül minden kéréshez be kell olvasni az egész táblát.

Ha a keresési feltételhez van megfelelő index, akkor az adatbázis-kezelő a diszkről csak a valóban szükséges sorokat olvassa be.

Például a

```
SELECT * FROM emp WHERE ename = 'JONES';
```

az `emp` táblában keresés nélkül kiválaszthatja `JONES` rekordját, ha az `ename` oszlopra definiáltunk indexet. Az indexek akkor is gyorsítják a keresést, ha csak a keresési feltétel egy részére vonatkoznak.

8.2. Indexek törlése

Az indexeket a

```
DROP INDEX <indexnév>;
```

paranccsal törölhetjük.

9. Jogosultságok definiálása

Az egyes felhasználók részint az adatbázis-kezelő rendszerrel, részint az egyes objektumaival különböző műveleteket végezhetnek. Ezeknek a megadására szolgálnak a `GRANT` utasítások. A

```
GRANT [DBA | CONNECT | RESOURCES]  
TO <felhasználó> [, <felhasználó> , ...]  
IDENTIFIED BY <jelszó> [, <jelszó> , ...];
```

paranccsal az egyes felhasználóknak az adatbázishoz való hozzáférési jogát szabályozzák. A `DBA` jogosultság az adatbázis adminisztrátorokat (DataBase Administrator) definiálja, akiknek korlátlan jogai vannak az összes adatbázis objektum felett, nemcsak létrehozhatja, módosíthatja, illetve törölheti, de befolyásolhatja az objektumok tárolásával, hozzáféréssel kapcsolatos paramétereket is. A `RESOURCES` jogosultsággal rendelkező felhasználók létrehozhatnak, módosíthatnak, illetve törölhetnek új objektumokat, míg a `CONNECT` jogosultság csak az adatbázis-kezelőbe belépésre jogosít.

Az egyes objektumokhoz – táblák, illetve nézetek – a hozzáférést a

```
GRANT <jogosultság> [, <jogosultság> , ...]  
ON <tábla vagy nézetnév>  
TO <felhasználó>  
[WITH GRANT OPTION];
```

parancs határozza meg. A *<jogosultság>* az objektumon végezhető műveleteket adja meg, lehetséges értékei:

```
ALL,  
SELECT,  
INSERT,  
UPDATE,  
DELETE,  
ALTER,  
INDEX
```

Az utolsó két művelet nézetekre nem alkalmazható. A felhasználó neve helyett PUBLIC is megadható, amely bármelyik felhasználóra vonatkozik. A WITH GRANT OPTION-nel megkapott jogosultságokat a felhasználók tovább is adhatják.

10. Tábladefiníciók módosítása

Már létező táblákat módosítani az

```
ALTER TABLE <táblanév>  
[ADD | MODIFY] <oszlopnév> <típus>;
```

paranccsal lehet, ahol ADD egy új, NULL értékű oszlopot illeszt a táblához, míg MODIFY paranccsal egy létező oszlop típusának tulajdonságait módosítjuk.

Például egy új oszlop felvétele a mytable táblába:

```
ALTER TABLE mytable  
ADD id NUMBER(6);
```

11. Tranzakciók

Az adatbázisok módosítása általában nem történhet meg egyetlen lépésben, hiszen legtöbbször egy módosítás során több táblában tárolt információ is változtatni akarunk, illetve egyszerre több rekordban akarunk módosítani, több rekordot akarunk beilleszteni. Előfordulhat, hogy módosítás közben meggondoljuk magunkat, vagy ami még súlyosabb következményekkel jár, hogy az adatbázis-kezelő leáll. Ilyenkor a tárolt adatok inkonzisztens állapotba kerülhetnének, hiszen egyes módosításokat már elvégeztünk, ehhez szorosan hozzátartozó másokat viszont még nem.

A tranzakció az adatbázis módosításának olyan sorozata, amelyet vagy teljes egészében kell végrehajtani, vagy egyetlen lépését sem. Az adatbázis-kezelők biztosítják, hogy mindig vissza lehessen térni az utolsó teljes egészében végrehajtott tranzakció utáni állapothoz.

Egy folyamatban lévő tranzakciót vagy a COMMIT utasítással zárhatjuk le, amely a korábbi COMMIT óta végrehajtott összes módosítást véglegesíti, vagy a ROLLBACK utasítással törölhetjük a hatásukat, visszatérve a megelőző COMMIT kiadásakor érvényes állapotba.

Beállítható, hogy az SQL műveletek automatikusan COMMIT műveletet hajtsanak végre:

```
SET AUTOCOMMIT [ON | OFF];
```

Az ON állapotban minden SQL utasítás, OFF állapotban az ALTER, CREATE, DROP, GRANT és EXIT utasítások sikeres végrehajtása COMMIT-ot is jelent. (Azaz ezeket nem lehet visszagörgetni, tehát pl. törölt tábla nem állítható vissza!)

A rendszer hardverhiba utáni újrainduláskor, illetve hibás INSERT, UPDATE vagy DELETE parancs hatására automatikusan ROLLBACK-et hajt végre. Érdemes hát biztonságos helyeken COMMIT parancsot kiadni, nehogy egy hibásan kiadott parancs visszavonja a korábbi módosításokat.

12. Párhuzamos hozzáférés szabályozása

Az adatbázis-kezelő rendszereket tipikusan több felhasználó használja, ezzel kapcsolatban újabb problémák merülnek fel, ezért az egyes táblákhoz a párhuzamos hozzáférést külön-külön lehet szabályozni:

```
LOCK TABLE <táblanév> [, <táblanév> , ...]  
IN [SHARE | SHARED UPDATE | EXCLUSIVE] MODE [NOWAIT];
```

A `LOCK` paranccsal egy felhasználó megadhatja, hogy az egyes táblákhoz más felhasználónak milyen egyidejű hozzáférést engedélyez. Az utasítás végrehajtásánál a rendszer ellenőrzi, hogy a `LOCK` utasításban igényelt felhasználási mód kompatibilis-e a táblára érvényben lévő kizárással. Amennyiben megfelelő, az utasítás visszatér és egyéb utasításokat lehet kiadni. Ha az igényelt kizárás nem engedélyezett, az utasítás váratkozik, amíg az érvényes kizárást megszüntetik, ha a parancs nem tartalmazza a `NOWAIT` módosítót. Ebben az esetben a `LOCK` utasítás mindig azonnal visszatér, de visszaadhat hibajelzést is.

A táblához a hozzáférést az első sikeres `LOCK` utasítás definiálja.

Az `EXCLUSIVE` mód kizárólagos hozzáférést biztosít a táblához. A `SHARE` módban megnyitott táblákat mások olvashatják, a `SHARE UPDATE` módban mások módosíthatják is, ilyenkor a kölcsönös kizárást az `ORACLE` soronként biztosítja (automatikusan), tehát nem írhatják ketten egyidejűleg ugyanazt a sort.

13. Konzisztenciafeltételek

A táblák definíciójánál eddig csak azt adhattuk meg, hogy milyen adattípusba tartozó értékeket lehet az egyes oszlopokban használni, illetve mely oszlopokban kell feltétlenül értéknek szerepelnie. Célszerű lenne a táblákhoz olyan feltételeket rendelni, amelyek szigorúbb feltételeket definiálnak az egyes adatokra, amelyeket aztán a rendszer a tábla minden módosításánál ellenőriz (ld. Függelék: Adatbázis kényszerek az Oracle-ben). Ilyenek például:

- az egyes adatok értékkészletének az általános adattípusnál pontosabb definíciója (pl. adott intervallumba tartozás, adott halmazba tartozás, ahol a halmaz lehet egy másik tábla egyik oszlopának értékei);
- az oszlop elsődleges kulcs, azaz a tábla soraiban minden értéke különböző (hasonló hatás elérhető a `UNIQUE` indexszel is);
- az oszlop idegen kulcs, azaz értéke meg kell, hogy egyezzen egy másik tábla elsődleges kulcs oszlopának valamelyik létező elemével.

Amennyiben a táblák módosításánál valamelyik feltételt megsértenénk, a rendszer egy kivételt (exception) generál és lefuttatja a hibához tartozó kiszolgáló utasítást, ha van ilyen.

Az SQL nyelv további elemeiről az online helpben található leírás.

II. Függelék: Adatbázis kényszerek az Oracle-ben

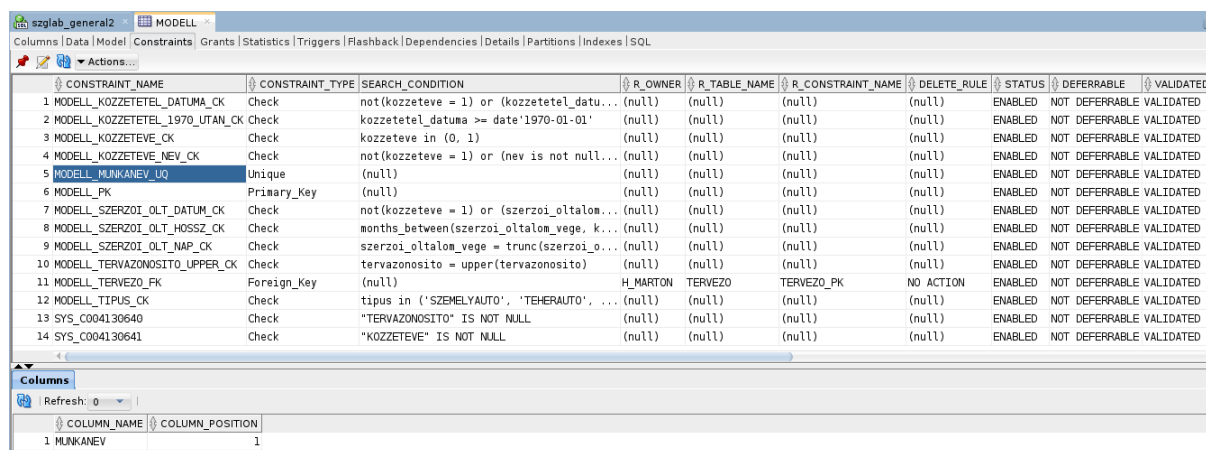
Kiegészítés az I. labor anyagához

1.	CHECK KÉNYSZER.....	40
2.	UNIQUE KÉNYSZER	41
3.	PRIMARY KÉNYSZER	41
4.	FOREIGN KÉNYSZER.....	42
5.	TELJES PÉLDA SQL NYELVEN.....	42

A constraintek, vagyis kényszerek egyszerű előírások, szabályok az adatbázisban található adatokra nézve, melyek elősegítik az ellentmondásmentesség (általában a tágabb értelemben vett “konzisztencia”) fenntartását az adatbázis-szerver szintjén. Sőt, a kényszerek ismeretében az Oracle képes a lekérdezéseket is jobban optimalizálni. Ilyen szabály lehet például egy mező (attribútum) adatainak szintaktikai ellenőrzése (pl. személyi igazolvány szám: két betű, hat szám formátumú (rég, füzet formájú) vagy fordított sorrendben (új, kártya formájú) legyen), de több adat összefüggését is ellenőrizhetjük (az adatok közötti hivatkozások helyességének fenntartása érdekében). A kényszerek tehát igen hasznosak tudnak lenni, ugyanakkor – tapasztalat szerint – egy rendszer módosítása során sok gondot is tudnak okozni.

Egy táblához több kényszer is megadható. Definiálásukra alapvetően két lehetőségünk van: vagy a megfelelő SQL parancsot gépeljük be vagy valamilyen grafikus eszközt használunk a parancs kényelmes összeállítására, így nem kell ismernünk a pontos szintaxist. Az SQL Developerben új tábla „Advanced” módú felvételekor, vagy meglévő tábla módosításakor a Constraints fül alatt találhatóak a kényszerek, és az Actions/Constraints parancsokkal módosíthatók/törölhetők vagy adhatók a táblához újabbak.

Tulajdonképpen az is kényszer, ha egy mező értéke nem lehet NULL, vagyis ha a tábla definíciójában szerepel a NOT NULL kulcsszó. Ezt a tábla szerkesztésénél a mezők megadása során állíthatjuk be, és a constraintek sorában is feltűnik (l. az 1. ábrán).



CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1 MODELL_KOZZETETEL_DATUMA_CK	Check	not(kozzeteve = 1) or (kozzetetele_datu...	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
2 MODELL_KOZZETETEL_1970_UTAN_CK	Check	kozzetetele_datuma >= date'1970-01-01'	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
3 MODELL_KOZZETEVE_CK	Check	kozzeteve in (0, 1)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
4 MODELL_KOZZETEVE_NEV_CK	Check	not(kozzeteve = 1) or (nev is not null...	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
5 MODELL_MUNKANEV_UQ	Unique	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
6 MODELL_PK	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
7 MODELL_SZERZOI_OLT_DATUM_CK	Check	not(kozzeteve = 1) or (szerzoi_oltalom...	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
8 MODELL_SZERZOI_OLT_HOSSZ_CK	Check	months_between(szerzoi_oltalom_vege, k...	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
9 MODELL_SZERZOI_OLT_NAP_CK	Check	szerzoi_oltalom_vege = trunc(szerzoi_o...	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
10 MODELL_TERVAZONOSITO_UPPER_CK	Check	tervazonosito = upper(tervazonosito)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
11 MODELL_TERVEZO_FK	Foreign_Key	(null)	H.MARTON	TERVEZO	TERVEZO_PK	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
12 MODELL_TIPUS_CK	Check	tipus in ('SZEMELYAUTO', 'TEHERAUTO', ...	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
13 SYS_C004130640	Check	"TERVAZONOSITO" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
14 SYS_C004130641	Check	"KOZZETEVE" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED

COLUMN_NAME	COLUMN_POSITION
1 MUNKANEV	1

1. ábra: Táblán definiált kényszerek megjelenítése az SQL Developerben
A teljes példa a függelék végén szerepel

A képernyő alapvetően két részre tagolható. A felső táblázatban jelennek meg a kényszerek alapadatai (név, típus, feltétel, a hivatkozás részletei stb.), az alsó táblázatban pedig az, hogy egy-egy kényszer mely oszlopokra vonatkozik. Az alsó táblázat tartalma mindig a felső táblázat egy-egy sorára vonatkozik.

A felső táblázatban az alábbi fontosabb oszlopok szerepelnek:

Constraint Name – a kényszer neve, amit létrehozáskor megadtunk. Amennyiben nem adtunk meg nevet, a rendszer automatikusan generál nagyjából véletlenszerű karakterláncot (pl. az ábrán látható „not null” constraint neve is így keletkezett). Noha beszédes név megadása nem kötelező mégis célszerű azt használni: ha ugyanis valamilyen későbbi művelet azért nem végrehajtható, mert sértené a kényszeret, akkor a rendszer ezzel a névvel fog hivatkozni a kényszerre. Kellően beszédes név esetén tehát azonnal tudni fogjuk, mi a probléma, anélkül hogy hosszasan keresgelnünk kellene. Érdekes elnevezési konvenciót is használni, pl. a kényszer típusától függő végződéssel ellátni a neveket (pl. primary key: `_PK`, unique: `_UQ`, check: `_CK`, foreign key: `_FK`)

Constraint Type – A kényszer típusa, mely Oracle-ben az alábbi lehet: `UNIQUE`, `PRIMARY_KEY`, `FOREIGN_KEY`, `CHECK`. A kényszer típusa alapvetően megszabja, hogy milyen paramétereket kell megadni létrehozásukkor.

Search Condition – `CHECK` típusú kényszernél használatos. Lásd alább.

R Owner – `FOREIGN` típusú kényszernél használatos. Lásd alább.

R Table Name – `FOREIGN` típusú kényszernél használatos. Lásd alább.

R Constraint Name – `FOREIGN` típusú kényszernél használatos. Lásd alább.

Delete Rule – `FOREIGN` típusú kényszernél használatos. Lásd alább.

Status – `Enabled` v. `Disabled` értéket vehet fel. Ha `Disabled`, akkor ideiglenesen letiltjuk, vagyis a rendszer úgy tekinti, mintha ez a kényszer egyáltalán nem is létezne. Célszerű például ideiglenesen kikapcsolni egy kényszeret, ha nagyon sok, a kényszeret nem sértő adatot importálunk egyszerre – így gyorsabb lesz a feldolgozás.

Deferrable – *Deferrable* v. *Not Deferrable* értéket vehet fel. Amennyiben *Deferrable* értékre állítjuk, úgy a felhasználó (vagy a kapcsolódó alkalmazás) kérheti a kényszer késleltetését (l. `set constraint` parancs). Ennek eredménye, hogy a kényszer megsértése esetén a rendszer nem a kényszeret sértő utasítás után azonnal ad hibajelzést, hanem csak a tranzakció végén, vagyis a `COMMIT` utasítás kiadásakor. Ennek nagy előnye, hogy a tranzakció elemi lépései után még sérülhet ugyan a kényszer (pl. egy rekord módosítása több lépésben, mezőnként), de természetesen a tranzakció végén már nem.

(Initially) Deferred – *Deferred* v. *Immediate* értéket vehet fel. Bár nem jelenik meg ebben a táblázatban, a kényszernek egy olyan fontos tulajdonságát írja le, mely csak akkor értelmezett, ha a *Deferrable* mezőt *Deferrable* értékűre állítottuk. Amennyiben ez a mező is igaz (*deferred*) értékű, akkor a rendszer automatikusan késlelteti a kényszer ellenőrzését a tranzakció végéig, vagyis a felhasználónak nem kell külön kérnie ezt a viselkedést.

Validated – *Validated* v. *Not Validated* értéket vehet fel. Amennyiben értéke *Not Validated*, akkor a rendszer a kényszer *Status=Enabled* állapotában csak az új és módosított rekordokat ellenőrzi. Míg ha értéke *Validated*, a rendszer garantálja, hogy a meglévő rekordok is eleget tesznek a kényszernek.

1. CHECK kényszer

`CHECK` típusú kényszer esetén a *Check Condition* mezőt kell kitölteni. Ide egy logikai kifejezést adhatunk meg, hasonlóan mint a lekérdezések `WHERE` clause-ában. A rendszer rekordok felvételénél és módosításánál ellenőrzi, hogy ez a feltétel teljesül-e, és ha nem, azaz a kényszer sérül, a végrehajtást a rendszer megtagadja. A kifejezésben sajnos nem

használhatunk subquery-eket, vagyis bonyolultabb lekérdezéseket, melyek más táblákra is hivatkozhatnak. A feltétel tehát csakis a kényszerhez tartozó tábla egy vagy néhány mezőjét vizsgálhatja. Nem szerepelhetnek továbbá nem determinisztikus visszatérési értékű függvények hívásai (pl. sysdate) sem a feltételben.

Megjegyzés: az Oracle Database terminológiája szerint a CHECK kényszer teljesül, ha az abban szereplő kifejezés kiértékelés eredménye igaz vagy ismeretlen (UNKNOWN), ami tipikusan NULL értékű mezők miatt fordul elő.

A kényszerekben szerepeltethető kifejezésre néhány példát kiemeltünk a függelék végén közölt demonstrációból.

Rögzített értékkészletű mező („enum”).

tipus in ('SZEMELYAUTO', 'TEHERAUTO', 'VIZI_JARMU', 'LEGI_JARMU')

A fenti megjegyzés értelmében a **tipus** mező a felsorolt 4 értéket veheti fel, vagy üres lehet (NULL értékű).

Rögzített értékkészletű mező, ami nem lehet NULL.

Ezt két lépésben érjük el:

a meződefinícióban: kozzeteve number(1) default 0 **not null**

a feltétel kozzeteve in (0, 1)

A mezőben szereplő betűk nagybetűk legyenek.

tervazonosito = upper(tervazonosito)

A dátum 1970. január 1 vagy későbbi.

kozzetetel_datuma >= date'1970-01-01'

A dátum nap pontosságú (az idő komponense 00:00:00).

szerzoi_oltalom_vege = trunc(szerzoi_oltalom_vege)

Az első dátum max 80 évvel későbbi a másodiknál.

months_between(szerzoi_oltalom_vege, kozzetetel_datuma) <= 80*12

Ha a kozzeteve mezőben 1 szerepel, akkor a kozzetetel_datuma nem üres.

A „ha X akkor Y” implikáció ekvivalens a „nem(X) vagy Y” alakkal, ezért:.

not(kozzeteve = 1) or (kozzetetel_datuma is not null)

2. UNIQUE kényszer

A UNIQUE típusú kényszerrel tulajdonképpen egy kulcsot definiálunk a táblához. Hatása kettős: a kulcsban szereplő mezők értékei által képzett kombinációnak egyedinek kell lennie a táblában. A rendszer alapértelmezésben automatikusan létrehoz egy indexet a megadott mezők alapján az egyediség ellenőrzésére és a keresések gyorsítására. A kulcshoz tartozó mező(ke)t az alsó táblázat Table Columns oszlopában találjuk.

A UNIQUE kényszerben szereplő mezők egyéb tiltás hiányában felvehetnek NULL értéket. Ilyen a függelék végén közölt példában szereplő modell tábla munkanev attribútuma is.

Megjegyzés: egy táblában több olyan rekord is szerepelhet, amely a UNIQUE kényszerben szereplő összes mezőjében a NULL értéket tartalmazza.

3. PRIMARY kényszer

A PRIMARY típusú kényszer az elsődleges kulcsot jelöli. Hasonló a UNIQUE típusú kényszer, de itt valamivel több megkötéssel találkozunk. Egyrészt egy táblára több UNIQUE, de csak egyetlen PRIMARY kényszer létezhet. Másrészt, míg előbbi megengedi a NULL értékeket, a PRIMARY kényszer egyúttal azt is biztosítja, hogy a megadott mezőkben ne lehessen NULL érték. Harmadrészt pedig általában az elsődleges kulcs azonosítja

egyértelműen a rekordot a táblában, ezért idegen kulcsokkal erre hivatkozunk más táblából. Létrehozása ugyanúgy történik mint a UNIQUE kényszernél, és ez is indexet hoz létre a háttérben.

Példa: a függelék végén közölt `modell` tábla `modell_id` attribútumán `modell_pk` néven definiáltunk egy elsődleges kulcs kényszert.

4. FOREIGN kényszer

A FOREIGN típusú kényszer a legérdekesebb és legbonyolultabb kényszer az Oracle repertoárjában. Idegen kulcsot jelöl, tehát az adott táblában bizonyos mezők – egy másik tábla bizonyos mezői alapján – hivatkoznak a másik tábla rekordjaira. A hivatkozott táblát általában szülő, a hivatkozó táblát gyermek táblának szokták hívni. Az idegen kulcshoz tartozó kényszert a gyermek táblában definiáljuk.

Idegen kulcs megadása esetében a Referenced Schema és Referenced Table mezőkben kell megadni, hogy mely séma mely táblájának rekordjaira fogunk hivatkozni az éppen szerkesztett táblából, vagyis itt adjuk meg a szülő táblát (az SQL Developerben ezt érdemes a tábla létrehozásakor azonnal megtenni). A szülő tábla egy Primary vagy Unique kényszerét azonosítva segít az SQL Developer abban, hogy hány oszlopból álló idegen kulcsot kell létrehozni a gyermek táblában.

Ezután az alsó táblázatban adhatjuk meg a hivatkozó és hivatkozott mező(k) páros(ai)t. A gyermek táblában szereplő hivatkozó mezőt a Local Columns, a szülő táblában szereplő hivatkozottat a Referenced Columns oszlopban.

Amennyiben a kényszerhez a Cascade On Delete paramétert bejelöljük, akkor törlési láncot is létrehozunk. Ilyenkor ha egy rekordot kitörölünk a szülő táblában, a rendszer automatikusan törli a gyermek táblában azon rekordokat, melyek az eredetileg törölni kívánt rekordokra hivatkoztak. Másik lehetőség a „set to null” amelynek hatására a szülő táblában történő törlés esetén a gyermek tábla megfelelő bejegyzései null értékre kerülnek beállításra. A „restrict” (vagy a fenti ábra táblázatában: „no action”) alapbeállítás pedig nem enged szülőbejegyzést törölni, amíg van rá hivatkozó rekord a gyermek táblában.

Példa: a függelék végén közölt példában a `modell` tábla `tervezo` attribútuma a `tervezo` tábla `tervezo_id` mezőjére mutat. Ezt írja le a `modell.modell_tervezo_fk` kényszer.

5. Teljes példa SQL nyelven

Az alábbiakban jármű-modelleket és tervezőit leíró táblapár SQL-nyelvű definícióját közöljük a különböző kényszertípusok demonstrálására.

```
create table tervezo (
    tervezo_id number not null
,   nev nvarchar2(200) not null
,   szulettett date
,   constraint tervezo_pk primary key (tervezo_id)
);
comment on table tervezo is 'A tervezők alapadatait tároló tábla.';
```

```

create table modell (
    modell_id number
,   tervezo number
,   tervazonosito nvarchar2(30) not null
,   munkanev nvarchar2(200)
,   tipus nvarchar2(30)
,   kozzeteve number(1) default 0 not null
,   nev nvarchar2(200)
,   kozzetetel_datuma date
,   szerzoi_oltalom_vege date
,   constraint modell_pk primary key (modell_id)
,   constraint modell_tervezo_fk foreign key (tervezo)
        references tervezo (tervezo_id)
,   constraint modell_tervazonosito_upper_ck check
        (tervazonosito = upper(tervazonosito))
,   constraint modell_munkanev_uq unique (munkanev)
,   constraint modell_tipus_ck check
        (tipus in ('SZEMELYAUTO', 'TEHERAUTO', 'VIZI_JARMU', 'LEGI_JARMU'))
,   constraint modell_kozzeteve_ck check (kozzeteve in (0, 1))
-- ha közzétett a modell, akkor nem üres:
--   nev, kozzetetel_datuma, szerzoi_oltalom_vege
,   constraint modell_kozzeteve_nev_ck check
        (not(kozzeteve = 1) or (nev is not null and length(nev)>0))
,   constraint modell_kozzetetel_datuma_ck check
        (not(kozzeteve = 1) or (kozzetetel_datuma is not null))
,   constraint modell_szerzoi_olt_datum_ck check
        (not(kozzeteve = 1) or (szerzoi_oltalom_vege is not null))
,   constraint modell_kozzetetel_1970_utan_ck check
        (kozzetetel_datuma >= date'1970-01-01')
,   constraint modell_szerzoi_olt_nap_ck check
        (szerzoi_oltalom_vege = trunc(szerzoi_oltalom_vege))
,   constraint modell_szerzoi_olt_hossz_ck check
        (months_between(szerzoi_oltalom_vege, kozzetetel_datuma) <= 80*12)
);
comment on table modell is 'Közúti, vízi és légijárművek adatait tároló
tábla.';
comment on column modell.tervazonosito is 'Nagybetűkkel írandó.';
comment on column modell.munkanev is 'A modell egyedi munkaneve (belső
neve). Lehet üres (null), ha még nem választott a tervező.';
comment on column modell.tipus is 'Lehetséges típusok személyautó
(SZEMELYAUTO), tehergépjármű (TEHERAUTO), vízi jármű (VIZI_JARMU) ill. légi
jármű (LEGI_JARMU).';
comment on column modell.kozzeteve is 'A 0, 1 értékek rendre a Hamis, Igaz
logikai értéket kódolják.';
comment on column modell.nev is 'A modell neve. Ha közzétett, akkor nem
lehet üres, tehát sem null, sem a 0-hosszú üres sztring.';
comment on column modell.kozzetetel_datuma is 'A közzététel időpontja. Ha
közzétett, akkor nem lehet üres. Csak 1970-ben vagy később közzétett
modelleket tárolunk.';
comment on column modell.szerzoi_oltalom_vege is 'A szerzői oldatol
lejáratási napja. Ha közzétett, akkor nem lehet üres. A közzététel dátumához
képest 80 éven belüli dátum. Nap pontosságú, és az oltalom a következő
napon szűnik meg.';

```