



Adatbázisok

I. gyakorlat:

Féléves feladat,
adatbázis tervezés,
SQL bevezető feladatok

Tudnivalók

- Dr. Nagy Enikő
nagy.eniko@nik.uni-obuda.hu
BA. 307. szoba
- Segédanyagok:
analog.nik.uni-obuda.hu
- **Moodle!**
elearning.uni-obuda.hu

Gyakorlat követelménye

- 2 db nagy zárthelyi (6. és 13. hét)
- féléves feladat (FF)

Tankönyv:

Kende-Nagy: Oracle példatár
(Panem Könyvkiadó, 2005)

Jegy kialakítása

- Előadás ZH (max. 25 pont) +
Gyakorlati pontszám (max. 50 pont) +
Féléves feladat + védése (max. 25 pont)

Gyakorlati pontszám:

- Gyak. nagyZH-k: 25-25 pont
 - mindkettőt min. 51%-ra kell teljesíteni

Ponthatárok

- Az elégséges jegyhez 51, a közepeshez 63, a jóhoz 74, a jeleshez 85 pontot kell elérni.



FÉLÉVES FELADAT

Követelmény

- **Gyak. vezető által kiosztott témában egyszerű adatbázis**
 - megtervezése
 - létrehozása
 - lekérdezések készítése
- Információk a Moodle-ben
- Részfeladatok beadása Moodle-n keresztül (1-2 Mérföldkő)



EGYSZERŰ SQL LEKÉRDEZÉSEK

Bevezetés

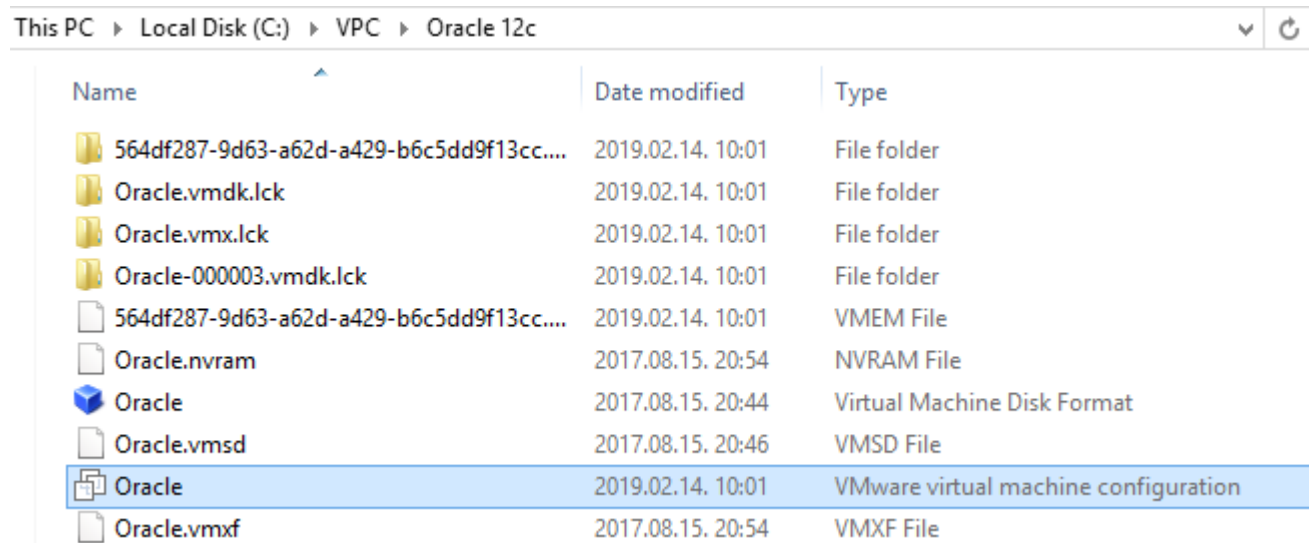
- Mi is az az adatbázis?
 - Első pillantásra: adatok rendezett gyűjteménye.
 - Egyedek (rekord: a táblázat egy sora)
 - Jellemzők (mezők, attribútumok: egy oszlop)

The diagram illustrates the structure of a database table. Above the table, the word "Mezők" (Fields) is centered, with lines connecting it to each of the seven column headers: Customers, FirstName, LastName, StreetAddress, City, State, and ZipCode. To the right of the table, the word "Rekordok" (Records) is positioned, with lines connecting it to each of the six data rows (rows 1010 through 1015).

| Customers | FirstName | LastName | StreetAddress | City | State | ZipCode |
|-----------|-----------|-----------|------------------------|--------------|-------|---------|
| 1010 | Angel | Kennedy | 667 Red River Road | Austin | TX | 78710 |
| 1011 | Alaina | Hallmark | Route 2, Box 203B | Woodinville | WA | 98072 |
| 1012 | Liz | Keyser | 13920 S.E. 40th Street | Bellevue | WA | 98006 |
| 1013 | Rachel | Patterson | 2114 Longview Lane | San Diego | CA | 92199 |
| 1014 | Sam | Abolrous | 611 Alpine Drive | Palm Springs | CA | 92263 |
| 1015 | Darren | Gehring | 2601 Seaview Lane | Chico | CA | 95926 |

Virtuális gép indítása

- Virtuális gép helye: C:\VPC\Oracle 12c
- Virtuális gép indítása: Oracle.vmx
 - VMware virtual machine configuration
- VMware indulásánál: “I copied it” opciót válasszuk



This PC > Local Disk (C:) > VPC > Oracle 12c

| Name | Date modified | Type |
|--|-------------------|--------------------------------------|
| 564df287-9d63-a62d-a429-b6c5dd9f13cc.... | 2019.02.14. 10:01 | File folder |
| Oracle.vmdk.lck | 2019.02.14. 10:01 | File folder |
| Oracle.vmx.lck | 2019.02.14. 10:01 | File folder |
| Oracle-000003.vmdk.lck | 2019.02.14. 10:01 | File folder |
| 564df287-9d63-a62d-a429-b6c5dd9f13cc.... | 2019.02.14. 10:01 | VMEM File |
| Oracle.nvram | 2017.08.15. 20:54 | NVRAM File |
| Oracle | 2017.08.15. 20:44 | Virtual Machine Disk Format |
| Oracle.vmsd | 2017.08.15. 20:46 | VMSD File |
| Oracle | 2019.02.14. 10:01 | VMware virtual machine configuration |
| Oracle.vmx | 2017.08.15. 20:54 | VMXF File |

Szolgáltatás indítása

- Figyeljük meg az IP-címet és másoljuk ki
- Az asztalon található “Tomcat up” parancsikonnal indítsuk el a szolgáltatást
- Böngészőből csatlakozás a szolgáltatáshoz:
IP-cím:8080/apex
Pl.: 192.168.186.128:8080/apex
- Workspace: work1
Username: magyar
Password: Tigris-1
- Otthoni használat:
 - <http://apex.oracle.com>

Ismerkedés a környezettel

- SQL Workshop
 - Object Browser: táblák és egyéb adatbázis-objektumok böngészése
 - SQL Commands: SQL utasítások futtatása
 - SQL Scripts: több utasításból álló "programok" (scriptek) futtatása
- App Builder
 - webalkalmazások készítése és futtatása (nem tanuljuk)

Lekérdezések

- Kérdezzük le az **emp** tábla tartalmát!

```
SELECT * FROM emp;
```

Az EMP tábla - a vállalat dolgozói

- EMPNO: dolgozói azonosítószám (elsődleges kulcs)
- ENAME: név
- JOB: munkakör
- MGR: a dolgozó közvetlen főnökének dolgozói azonosítószáma, ha van az illetőnek főnöke
- HIREDATE: belépés dátuma
- SAL: fizetés
- COMM: jutalék, amennyiben a dolgozó munkaköre szerint kaphat illet
- DEPTNO: részleg azonosítószáma (idegen kulcs a DEPT táblára)

Lekérdezések

- Kérdezzük le a **dept** tábla tartalmát!

```
SELECT * FROM dept;
```

A DEPT tábla - a vállalat részlegei

- DEPTNO: részleg azonosítószáma (elsődleges kulcs)
- DNAME: részleg neve
- LOC: részleg helye (melyik városban található)

Lekérdezések

- Az **emp** táblából csak a dolgozó nevét, fizetését és jutalékát szeretnénk látni.

```
SELECT  ename, sal, comm  
FROM    emp;
```

Lekérdezések

- Nézzük most a nevet, munkakört és a felvétel dátumát!

```
SELECT  ename, job, hiredate  
FROM    emp;
```

Lekérdezések

- Lássuk az azonosítót és a nevet, de az oszlopok elnevezése legyen magyar!

```
SELECT empno AS "Azonosító",  
       ename AS "Név"  
FROM emp;
```

Lekérdezések

- Szeretnénk tudni, ki keres sokat, és milyen munkakörben.

```
SELECT  ename, job, sal  
FROM    emp  
WHERE   sal > 2000;
```

Lekérdezések

- ...és ki kap jutalékot? Mennyit?

```
SELECT  ename, comm  
FROM    emp  
WHERE   comm > 0;
```

Lekérdezések

- Rendezzük a táblázatunkat fizetés szerinti növekvő sorrendbe!

```
SELECT * FROM emp  
ORDER BY sal;
```

Lekérdezések

- ... és csökkenő sorrendbe?

```
SELECT * FROM emp  
ORDER BY sal DESC;
```

Lekérdezések

- Akkor most rendezzünk belépési dátum szerint!

```
SELECT * FROM emp  
ORDER BY hiredate;
```

```
SELECT * FROM emp  
ORDER BY hiredate DESC;
```


Gyakorlás

- Irassuk ki azon dolgozók nevét, munkakörét és fizetését, akiknek a fizetése 1500 USD alatt van. A lista fejléce legyen „Név”, „Munkakör”, „Fizetés”, rendezzen a dolgozók neve szerint növekvő sorrendbe.

Megoldás

```
SELECT  ename AS "Név",  
        job  AS "Munkakör",  
        sal  AS "Fizetés"  
  
FROM    emp  
  
WHERE   sal < 1500  
  
ORDER BY ename;
```

További feltétel kifejezések

- Írassuk ki azon dolgozók nevét, munkakörét és fizetését, akiknek a fizetése **1500 és 2500 USD között** van. A lista fejléce legyen „Név”, „Munkakör”, „Fizetés”, rendezzen a dolgozók neve szerint.

Megoldás

```
SELECT  ename AS "Név",  
        job  AS "Munkakör",  
        sal  AS "Fizetés"  
  
FROM    emp  
  
WHERE   sal BETWEEN 1500 AND  
        2500  
  
ORDER  BY ename;
```

További feltétel kifejezések

- Irassuk ki azon dolgozók nevét, munkakörét és fizetését, akiknek a fizetése **NINCS 1500 és 2500 USD között**. A lista fejléce legyen „Név”, „Munkakör”, „Fizetés”, rendezzen a dolgozók neve szerint.

Megoldás

```
SELECT  ename AS "Név",  
        job  AS "Munkakör",  
        sal  AS "Fizetés"  
  
FROM    emp  
  
WHERE   sal NOT BETWEEN 1500 AND  
        2500  
  
ORDER  BY ename;
```

További feltétel kifejezések

- Írassuk ki a „clerk” munkakörű dolgozók nevét, munkakörét, fizetését. A lista fejléce legyen „Név”, „Munkakör”, „Fizetés”, rendezzen a dolgozók neve szerint.

```
SELECT ename, job, sal FROM emp  
WHERE job LIKE '%clerk%';
```

- Mi történt? Pedig van clerk!!!

Megoldás

- Kis és nagybetűk itt számítanak!

```
SELECT ename, job, sal FROM emp  
WHERE job LIKE '%CLERK%';
```

vagy:

```
SELECT ename, job, sal FROM emp  
WHERE LOWER(job) LIKE '%clerk%';
```


További feltétel kifejezések

- De mi van akkor, ha több munkakörre is kíváncsiak vagyunk? Mondjuk „salesman” és „clerk”...

```
SELECT ename, job, sal FROM emp  
WHERE UPPER(job) IN ('SALESMAN',  
                    'CLERK');
```

Megjegyzés: vagy ... LOWER(job) IN ('salesman',
 'clerk');

Mi az eredmény és miért?

```
SELECT  ename AS Név,  
        sal  AS Fizetés,  
        sal+comm AS Jövedelem  
FROM emp  
ORDER BY sal+comm DESC;
```

Helyes megoldás

```
SELECT  ename AS Név,  
        sal  AS Fizetés,  
        sal+NVL(comm,0) AS Jövedelem  
FROM emp  
ORDER BY Jövedelem DESC;
```

Még egy-két apróság...

- Milyen munkakörök léteznek ennél a cégnél?

```
SELECT job FROM emp;
```

- De nekem elég, ha egy munkakört csak egyszer listáz...

```
SELECT DISTINCT job FROM emp;
```

Még egy-két apróság...

- Ki kap jutalékot? Vigyázat, aki nem kap, annál az érték nem 0, hanem nincs is ott semmi!
- A ...WHERE comm=0; nem fog működni.

```
SELECT ename, comm FROM emp  
WHERE comm IS NOT NULL;
```

Még egy-két apróság...

- Műveletek dátumokkal

```
SELECT ename AS Név, hiredate AS Dátum,  
       EXTRACT(YEAR FROM hiredate) AS BeÉv,  
       EXTRACT(MONTH FROM hiredate) AS BeHó,  
       EXTRACT(DAY FROM hiredate) AS BeNap  
FROM emp  
WHERE hiredate >  
       TO_DATE('05/01/1981','MM/DD/YYYY');
```

Gyakorlás

- Írassuk ki azon dolgozók nevét, jövedelmét (fizetés+jutalék) és részlegazonosítóját, akik a 20 vagy 30 részlegben dolgoznak, és a jövedelmük 1500-nál több. A lista fejléce legyen „Név”, „Jövedelem”, „Részleg”, rendezzen a részlegazonosító szerint növekvő és azon belül név szerint csökkenő sorrendbe.



Adatbázisok

2. gyakorlat:

Csoportosítás

Csoportfüggvények

AVG

- átlag

COUNT

- számosság

SUM

- összegzés

MIN

- minimum

MAX

- maximum

...

Csoportfüggvény használata a teljes táblára

- Példák:

```
SELECT AVG(sal) FROM emp;
```

```
SELECT MAX(comm) FROM emp;
```

```
SELECT MIN(deptno) FROM dept;
```

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

Csoportosítás

GROUP BY

- Csoportosítás valamely attribútum(ok) értékei szerint, pl:

GROUP BY deptno: részlegazonosító szerint

GROUP BY job: munkakör szerint

GROUP BY deptno, job: részleg szerint, azon belül munkakör szerint

➤ Helye: a WHERE után, az ORDER BY előtt

Csoportosítás

- Példa: legnagyobb fizetés részlegenként

```
SELECT deptno, MAX(sal)
```

```
FROM emp
```

```
GROUP BY deptno;
```

Csoportosítás

- A csoportosítás miatt nem szerepelhet bármi a SELECT-ben!
 - nem listázható olyan attribútum, ami szerint nem csoportosítunk
- Szerepelhet: **csoportosító attribútum, csoportfüggvény vagy konstans, illetve az ezekből alkotott kifejezések**

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

Csoportosítás

- A legkisebb jövedelem (sal + comm) munkakörönként...

```
SELECT job Munkakör,  
       MIN(sal+NVL(comm,0)) Legkisebb  
FROM emp  
GROUP BY job;
```

Csoportosítás

- Az átlagfizetés főnökönként...

```
SELECT mgr Főnökazonosító,  
        ROUND(AVG(sal),2) Átlagfizu  
FROM emp  
GROUP BY mgr;
```

- A ROUND függvény második paramétere a tizedeshelyek száma, alapértelmezés a 0 (egészre kerekítés)

Csoportosítás

- Részlegenkénti összfizetés részlegek szerint rendezve...

```
SELECT    deptno Részlegazonosító,  
          SUM(sal) Összfizetés  
  
FROM emp  
  
GROUP BY deptno  
  
ORDER BY deptno;
```


Csoportosítás

- Hányan dolgoznak a különböző munkakörökben 1000 dollárnál nagyobb fizetéssel?

```
SELECT job Munkakör, COUNT(*) Létszám  
FROM emp  
WHERE sal > 1000  
GROUP BY job;
```

- Ebben az esetben a `sal > 1000` szűrés a csoportosítás előtt történik meg!

A COUNT függvényről

- A COUNT a nem-null előfordulásokat számolja, ha paramétert adunk neki:

```
SELECT job Munkakör, COUNT(*) Létszám  
FROM emp GROUP BY job;
```

```
SELECT job Munkakör, COUNT(comm)  
    "Jutalékot kaphat"  
FROM emp GROUP BY job;
```

Önálló feladat

- A főnök szeretné látni **részlegenként**:
 - a legalacsonyabb fizetést,
 - a legmagasabb fizetést,
 - az átlagos fizetést, és
 - a létszámot.

Megoldás

```
SELECT deptno Részleg,  
        MIN(sal) Legkisebb,  
        MAX(sal) Legnagyobb,  
        ROUND(AVG(sal)) Átlag,  
        COUNT(*) Létszám  
  
FROM emp  
  
GROUP BY deptno;
```

Szűrés csoportokra

- HAVING
- A szűrés a **csoportosítás után** történik
- A csoportosítás eredményeképp létrejövő adatokra szűrhetünk
 - tipikusan a csoportfüggvény által előállított értékekre
- Helye a lekérdezésben: a GROUP BY után, az ORDER BY előtt.

Sorrend

- A lekérdezés elemeinek sorrendje tehát:

| | | |
|-----------------|---|----------|
| SELECT | } | kötelező |
| FROM | | |
| WHERE | | |
| GROUP BY | | |
| HAVING | | |
| ORDER BY | | |

Szűrés csoportokra

- Listázzuk a legalább 4 fős részlegeket létszámokkal.

```
SELECT deptno, COUNT(*)
```

```
FROM emp
```

```
GROUP BY deptno
```

```
HAVING COUNT(*) >= 4;
```

Szűrés csoportokra

- Azok a munkakörök, amelyekben a legtöbb kereső dolgozó fizetése 2500 dollár fölött van, a hozzájuk tartozó legnagyobb fizetéssel...

```
SELECT job, MAX(sal)
```

```
FROM emp
```

```
GROUP BY job
```

```
HAVING MAX(sal) > 2500;
```


Szűrés csoportokra

- 2000 USD-nál nagyobb átlagfizetésű részlegek, az átlagfizetés szerint növekvően rendezve...

Megoldás

```
SELECT deptno Részleg,  
        ROUND(AVG(sal)) Átlagfizetés  
FROM emp  
GROUP BY deptno  
HAVING ROUND(AVG(sal)) > 2000  
ORDER BY Átlagfizetés;
```

Szűrés csoportokra

- Listázzuk főnökönként (mgr) a jutalékban nem részesülő dolgozóinak átlagfizetését csökkenő sorrendben, feltéve, hogy ez az érték 1000 USD-nál több.

Megoldás

```
SELECT AVG(sal) Átlagfizetés,  
       mgr Főnök  
FROM emp  
WHERE comm IS NULL AND  
       mgr IS NOT NULL  
GROUP BY mgr  
HAVING AVG(sal) > 1000  
ORDER BY Átlagfizetés DESC;
```



Adatbázisok

3. gyakorlat:

Többléptékes lekérdezések

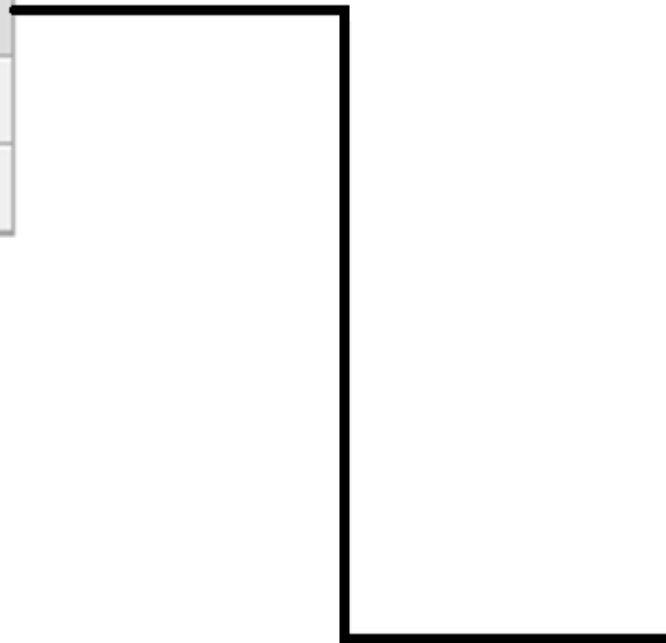
Többléptű lekérdezések

- Cél: a táblák összekapcsolása bizonyos attribútumok megegyezése alapján.
- Példa:
Szeretnénk látni, melyik dolgozó melyik városban dolgozik.
Probléma: a dolgozó neve az emp, a város a dept táblában található!

Táblák összekapcsolása

| DEPT | | |
|--------------------------|--------|-----|
| <input type="checkbox"/> | DEPTNO | 789 |
| <input type="checkbox"/> | DNAME | A |
| <input type="checkbox"/> | LOC | A |

| EMP | | |
|--------------------------|----------|-----|
| <input type="checkbox"/> | EMPNO | 789 |
| <input type="checkbox"/> | ENAME | A |
| <input type="checkbox"/> | JOB | A |
| <input type="checkbox"/> | MGR | 789 |
| <input type="checkbox"/> | HIREDATE | 31 |
| <input type="checkbox"/> | SAL | 789 |
| <input type="checkbox"/> | COMM | 789 |
| <input type="checkbox"/> | DEPTNO | 789 |



Összekapcsolás a FROM és WHERE részben

```
SELECT emp.ename, dept.loc  
FROM emp, dept  
WHERE emp.deptno=dept.deptno;
```


Összekapcsolás a FROM és WHERE részben

- Ha elhagyjuk a feltételt, minden sort minden sorral párosít!
 - DE szintaktikailag nem hibás...

```
SELECT emp.ename, dept.loc  
FROM emp, dept  
ORDER BY emp.ename;
```

Összekapcsolás a FROM és WHERE részben

- Nem kötelező az attribútumoknál a táblanév jelzése, ha csak az egyik táblában van olyan.

```
SELECT ename, loc, emp.deptno  
FROM emp, dept  
WHERE emp.deptno=dept.deptno;
```

Összekapcsolás a FROM és WHERE részben

- A * továbbra is használható, ám ilyenkor mindkét tábla összes oszlopát jelenti.
- A deptno kétszer fog szerepelni!
 - Egyik az emp.deptno, másik a dept.deptno

```
SELECT * FROM emp, dept  
WHERE emp.deptno=dept.deptno;
```

Összekapcsolás a FROM és WHERE részben

- Ha csak az egyik tábla összes oszlopát szeretnénk látni:

```
SELECT ename, sal, dept.*  
FROM emp, dept  
WHERE emp.deptno=dept.deptno;
```

Gyakorlás

Akkor most nézzük a dolgozók

- nevét,
- munkakörét,
- részlege nevét!

```
SELECT ename, job, dname  
FROM emp, dept  
WHERE emp.deptno = dept.deptno;
```

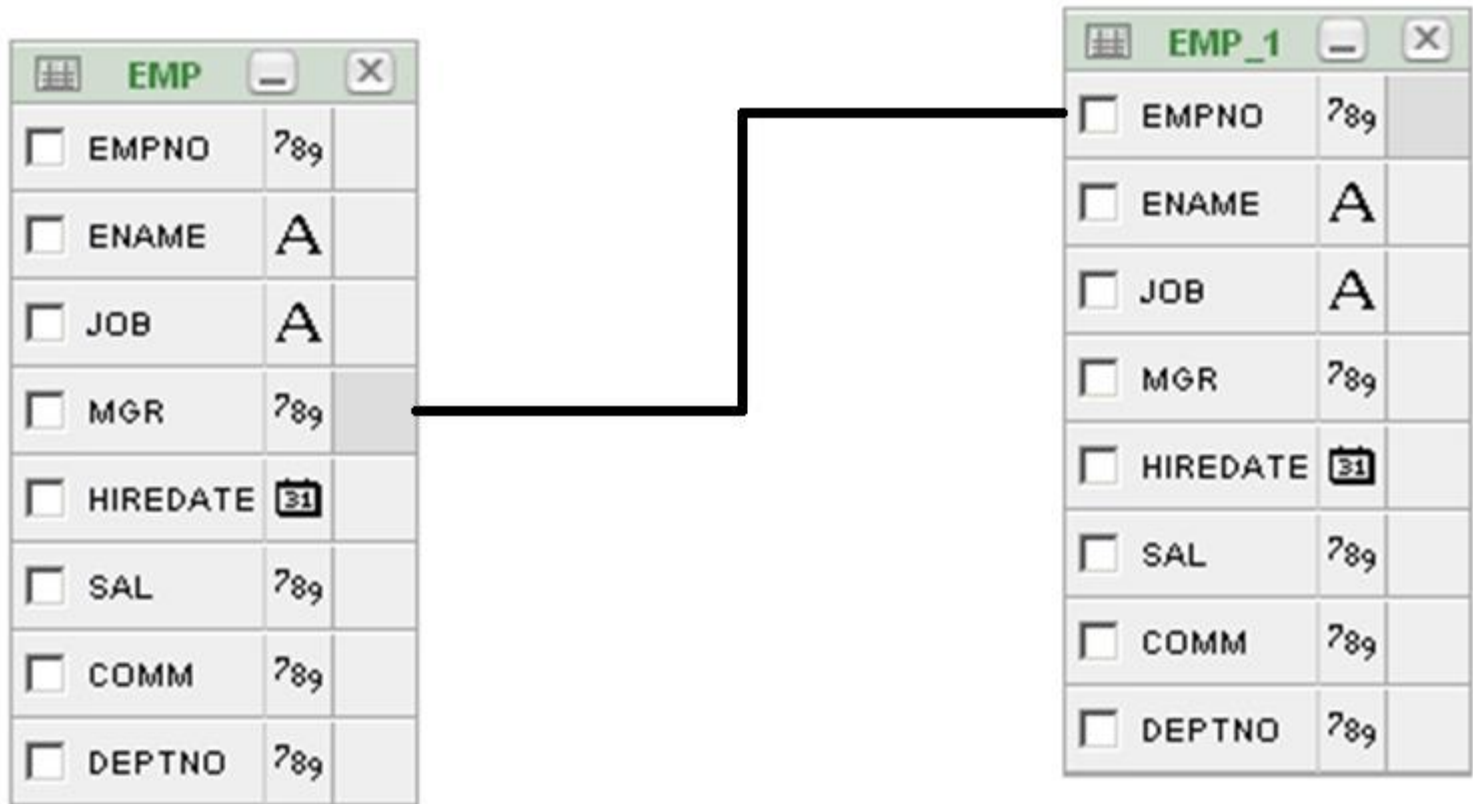
Gyakorlás

Listázzuk az 'ACCOUNTING' részlegben dolgozók

- összes dolgozói adatát,
- és részlegük helyét!

```
SELECT emp.*, loc FROM emp, dept  
WHERE dept.deptno = emp.deptno  
AND dname = 'ACCOUNTING';
```

Táblák összekapcsolása



Példa

- A dolgozók neve mellé kerüljön oda a főnökük neve.

```
SELECT  e1.ename Dolgozó,  
        e2.ename Főnöke  
FROM    emp e1, emp e2  
WHERE   e1.mgr=e2.empno;
```


Példa

- Most a főnök és a beosztott nevén kívül jelenítsük meg kettejük fizetésének a különbségét is!

```
SELECT      beosztott.ename Dolgozó,  
            főnök.ename Főnöke,  
            főnök.sal-beosztott.sal Különbség  
FROM emp főnök, emp beosztott  
WHERE beosztott.mgr=főnök.empno;
```

Gyakorlás

- Listázzuk azon dolgozók nevét, fizetését és főnökük nevét, akiknek a főnöke legalább 3000 dollárt keres.

```
SELECT d.ename Dolgozó, d.sal DolgFizu  
f.ename Főnöke  
FROM emp f, emp d  
WHERE d.mgr=f.empno AND f.sal >= 3000;
```

Gyakorlás

- Listázzuk a főnökök neveit, valamint a beosztottaik számát és átlagfizetését.

```
SELECT f.ename Főnöke,  
COUNT(d.ename) DolgDB,  
ROUND(AVG(d.sal),2)  
DolgFizuÁtlag  
FROM emp f, emp d  
WHERE d.mgr = f.empno  
GROUP BY f.ename;
```

Megoldás

```
SELECT főnök.ename "Főnök neve",  
       COUNT(*) "Beosztottai száma",  
       AVG(beosztott.sal) "Beo. átlagfizu"  
FROM emp főnök, emp beosztott  
WHERE beosztott.mgr=főnök.empno  
GROUP BY főnök.ename;
```

Táblák összekapcsolása

- Másik fajta szintaktika: **JOIN**

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

- Ugyanaz, mint:

```
SELECT ename, loc FROM emp, dept  
WHERE emp.deptno=dept.deptno;
```

Gyakorlás

- Listázzuk a dolgozók nevét, részlegük azonosítóját és nevét.

```
SELECT ename, deptno, dname  
FROM emp INNER JOIN dept  
USING (deptno);
```

- Ilyenkor nem szabad kiírni a táblanevet!

Táblák összekapcsolása

- A * ebben az esetben csak egy deptno oszlopot mutat!

```
SELECT *  
FROM emp INNER JOIN dept  
USING (deptno);
```

Táblák összekapcsolása

- Ha az attribútumnevek nem egyeznek meg, akkor is használható az INNER JOIN:

```
SELECT  e1.ename Beosztott,  
        e2.ename Főnök  
FROM    emp e1 INNER JOIN emp e2  
ON      e1.mgr=e2.empno;
```


Táblák összekapcsolása

- Vigyázat, ON használatakor újra két deptno oszlop lesz:

```
SELECT *
```

```
FROM emp INNER JOIN dept
```

```
ON emp.deptno=dept.deptno;
```

```
SELECT ename, dept.deptno, dname
```

```
FROM emp INNER JOIN dept
```

```
ON emp.deptno=dept.deptno;
```

Összekapcsolás típusok

- INNER JOIN
- OUTER JOIN fajták:
 - LEFT JOIN
 - RIGHT JOIN
 - FULL JOIN
- NATURAL JOIN

INNER vs. OUTER JOIN

INNER JOIN:

- az e1-ből csak azok, akiknek van főnöke;
- az e2-ből csak azok, akiknek van beosztottja

```
SELECT  e1.ename Beosztott,  
        e2.ename Főnök
```

```
FROM emp e1 INNER JOIN emp e2  
ON e1.mgr=e2.empno;
```

INNER vs. OUTER JOIN

-- azokat listázza, akiknek van és azokat is,
-- akinek NINCS főnök (ahol nincs: NULL lesz)
SELECT e1.ename Beosztott, e2.ename Főnök
FROM emp e1 LEFT JOIN emp e2
ON e1.mgr=e2.empno;

-- ez ugyanaz, mint az alatta lévő:
SELECT e1.ename Beosztott, e2.ename Főnök
FROM emp e1 RIGHT JOIN emp e2
ON e1.mgr=e2.empno;

SELECT e1.ename Beosztott, e2.ename Főnök
FROM emp e2 LEFT JOIN emp e1
ON e1.mgr=e2.empno;

-- mindenki: van beosztottja és főnöke
-- nincs főnöke és nincs beosztottja
SELECT e1.ename Beosztott, e2.ename Főnök
FROM emp e2 FULL JOIN emp e1
ON e1.mgr=e2.empno;

INNER vs. OUTER JOIN

Mi az eltérés az előző lekérdezéshez képest?

- ... emp e1 **LEFT JOIN** emp e2 ...
 - Mindenki megjelenik a Beosztott oszlopban, az is, akinek nincs főnöke.
- ... emp e1 **RIGHT JOIN** emp e2 ...
 - Mindenki megjelenik a Főnök oszlopban, az is, akinek nincs beosztottja.
- ... emp e1 **FULL JOIN** emp e2 ...
 - Az előző kettő együtt.

NATURAL JOIN

- Az egyező nevű oszlopok alapján kapcsol össze.

```
SELECT empno, ename, deptno, dname  
FROM emp NATURAL JOIN dept;
```

```
SELECT *  
FROM emp NATURAL JOIN dept;
```

- Ez tehát **NEM JÓ**:
... emp e1 NATURAL JOIN emp e2 ...

Gyakorlás

- Listázzuk azon dolgozók összes dolgozói adatát és részlegük nevét, akik:
 - a Chicago-i részlegben dolgoznak, és
 - 1000 dollárnál többet keresnek VAGY kapnak jutalékot.
- A megoldáshoz használjunk NATURAL JOIN-t.

```
SELECT empno, ename, job, mgr, hiredate, sal, comm, dname  
FROM emp NATURAL JOIN dept  
WHERE loc = 'Chicago' AND sal > 1000 OR comm IS NOT  
NULL;
```

```
NVL(comm,0) != 0  
NVL(comm,0) <> 0
```



Adatbázisok

4. gyakorlat:

DDL utasítások, megszorítások

DDL és DML utasítások

- DDL: Data Definition Language
 - **táblák, nézetek** létrehozása, módosítása, törlése
- DML: Data Manipulation Language
 - **sorok** beszúrása, módosítása, törlése

Nézetek

- View-k
- Logikailag: egy vagy több tábla adatainak egy részhalmaza.
- Gyakorlatilag: egy lekérdezést „mentünk és úgy használjuk, mintha tábla lenne”.

Nézet létrehozása

```
CREATE VIEW empv2 AS  
  SELECT empno, ename, job  
  FROM emp WHERE deptno = 10;
```

Nézet létrehozása

- Hozzunk létre egy nézetet, amely tartalmazza a részlegazonosítót és a részlegenkénti átlagfizetéseket.

```
CREATE VIEW átlagok AS  
SELECT deptno részleg,  
       round(avg(sal)) átlagfizetés  
FROM emp GROUP BY deptno;
```

Gyakorlás

- Hozzunk létre egy nézetet, amely listázza a dolgozók nevét, jövedelmét (fizetés + jutalék), munkakörét, belépésük évét, valamint részlegük nevét.

```
CREATE VIEW adatok AS
SELECT ename név,
sal + NVL(comm,0) jövedelem, job munkakör, hiredate belépés, dname
részleg
FROM emp INNER JOIN dept
ON emp.deptno = dept.deptno;

SELECT * FROM adatok;
```

Nézet törlése

`DROP VIEW empv2;`

`DROP VIEW átlagok;`

- Az adatok megmaradnak!

Tábla létrehozása

- Egyszerűbb eset: tábla létrehozása egy lekérdezés eredménye alapján:

```
CREATE TABLE emp2 AS  
SELECT * FROM emp;
```

- Mire emlékeztet ez a szintaktika?
- Itt valóban másolat készül!

Tábla létrehozása

- Most azokat másoljuk át új táblába, akiknek 2000 dollárnál több a fizetése!

```
CREATE TABLE emp3 AS  
  SELECT * FROM emp  
  WHERE sal>2000;
```


Tábla létrehozása

- Teljesen új tábla létrehozása:

```
CREATE TABLE új tábla  
(számoszlop NUMBER(5,2),  
szövegoszlop VARCHAR2(10),  
dátumoszlop DATE);
```

- NUMBER típus paraméterei:
 - 1. paraméter: jegyek száma **összesen**,
 - 2. paraméter: **ebből** mennyi törtrész
- A CHAR kötött hosszúságú!

Módosítás

ALTER TABLE táblanév ADD (...)

ALTER TABLE táblanév MODIFY (...)

ALTER TABLE táblanév
DROP COLUMN oszlopnév

ALTER TABLE táblanév
RENAME COLUMN Réginév TO Újnév

Módosítás

- Adjunk hozzá egy új oszlopot a másolat-táblánkhoz, amely a dolgozók kedvenc színét tárolja!

```
ALTER TABLE emp3 ADD  
(színe VARCHAR2(10));
```

Átnevezés, törlés

- Tábla átnevezése:

```
RENAME emp3 TO emp23;
```

- Tábla törlése:

```
DROP TABLE emp23;
```

Megszorítások

- PRIMARY KEY – elsődleges kulcs
- FOREIGN KEY – idegen kulcs
- NOT NULL – az értéke nem lehet NULL
- UNIQUE – minden érték csak egyszer szerepel
- CHECK – az értékek meg kell felelnie a megadott feltételnek

PRIMARY KEY

```
CREATE TABLE dept(  
    deptno NUMBER(2),  
    dname VARCHAR2(14),  
    loc VARCHAR2(13),  
    CONSTRAINT dept_deptno_pk  
    PRIMARY KEY (deptno));
```

FOREIGN KEY

```
CREATE TABLE emp(  
    empno NUMBER(4),  
    ename VARCHAR2(10),  
    job VARCHAR2(9),  
    mgr NUMBER(4),  
    hiredate DATE,  
    sal NUMBER(7,2),  
    comm NUMBER(7,2),  
    deptno NUMBER(2),  
    CONSTRAINT emp_deptno_fk  
    FOREIGN KEY (deptno)  
    REFERENCES dept (deptno));
```

NOT NULL

```
CREATE TABLE emp(  
    empno NUMBER(4),  
    ename VARCHAR2(10) NOT NULL,  
    job VARCHAR2(9),  
    mgr NUMBER(4),  
    hiredate DATE,  
    sal NUMBER(7,2),  
    comm NUMBER(7,2),  
    deptno NUMBER(2) NOT NULL);
```


UNIQUE

```
CREATE TABLE dept(  
    deptno NUMBER(2),  
    dname VARCHAR2(14),  
    loc VARCHAR2(13),  
    CONSTRAINT dept_dname_uk  
    UNIQUE (dname));
```

CHECK

```
CREATE TABLE dept(  
    deptno NUMBER(2),  
    dname VARCHAR2(14),  
    loc VARCHAR2(13),  
    CONSTRAINT emp_deptno_ck  
    CHECK (DEPTNO BETWEEN 10 AND 99));
```

Megszorítás hozzáadása

ALTER TABLE táblanév
ADD CONSTRAINT ...

vagy

ALTER TABLE táblanév
MODIFY ...

- DROP CONSTRAINT megszorításnév;
- ENABLE CONSTRAINT megszorításnév;
- DISABLE CONSTRAINT megszorításnév;

Megszorítás hozzáadása

Példák:

```
ALTER TABLE emp ADD CONSTRAINT  
    minimálbér CHECK (sal>700);
```

vagy

```
ALTER TABLE emp  
MODIFY sal CHECK (sal>700);
```

```
ALTER TABLE emp  
DROP CONSTRAINT minimálbér;
```

Megszorítás hozzáadása

```
ALTER TABLE emp ADD CONSTRAINT  
    minimálbér CHECK (sal>1500);
```

- Nem működik! Csak olyan megszorítást adhatunk meglévő táblához (vagy engedélyezhetünk rá), ami nem mond ellent a benne levő adatoknak!

Gyakorlás

- Hozzuk létre a következő táblát **kutya** néven:
 - **ID**: legfeljebb 3 jegyű egész szám
 - **név**: legfeljebb 20 karakter hosszú szöveg
 - **nem**: legfeljebb 1 jegyű egész szám
 - **szüldátum**: dátum
 - Elsődleges kulcs: **ID**
 - A **nem** csak 0 vagy 1 lehet.
 - A **név** nem maradhat üresen.

Gyakorlás

Kutya tábla megoldása:

```
CREATE TABLE kutya (  
    ID NUMBER(3),  
    név VARCHAR2(20) NOT NULL,  
    nem NUMBER(1),  
    születésnap DATE,  
    CONSTRAINT kutya_id_pk PRIMARY KEY (id),  
    CONSTRAINT kutya_nem_ck CHECK (nem  
    BETWEEN 0 AND 1)  
);
```

Megszorítások lekérdezése

```
SELECT constraint_name,  
       constraint_type, table_name  
FROM   user_constraints  
WHERE  table_name IN  
( 'EMP', 'DEPT' )
```




Adatbázisok

5. gyakorlat:

DML utasítások

DML és DDL utasítások

- DML: Data Manipulation Language
 - **sorok** beszúrása, módosítása, törlése
- DDL: Data Definition Language
 - **táblák** létrehozása, módosítása, törlése

Előkészületek

- Hozzunk létre egy **emp2** táblát a teljes **emp** tábla alapján.
- Hozzunk létre egy **dept2** táblát a teljes **dept** tábla alapján.
- Legyen a **dept2**-ben elsődleges kulcs a deptno.
- Legyen az **emp2**-ben elsődleges kulcs az empno, idegen kulcs a deptno a **dept2**-re.

DML: beszúrás

- Helyezzünk el a **dept2** táblába egy újabb részleget.

```
INSERT INTO dept2 (deptno, dname, loc)  
VALUES (42, 'SAJTKÉSZÍTŐ', 'BUDAPEST')
```

- Ami érdekes:
 - sorrend
 - minden értéket megadunk?

DML: beszúrás

- Az alábbiak közül melyik működik? Miért?

```
INSERT INTO dept2  
VALUES (50, 'NÉV', 'VÁROS')
```

```
INSERT INTO dept2  
VALUES (30, 'NÉV', 'VÁROS')
```

```
INSERT INTO dept2 (dname, loc)  
VALUES ('NÉV', 'VÁROS')
```

DML: beszúrás

- Dolgozó is kell a sajtkészítők részlegébe: vegyünk fel az **emp2** táblába „SAJTKUKAC” munkakörbe egy 1111-es azonosítójú, Ödön nevű személyt.

```
INSERT INTO emp2  
  (empno, ename, deptno, job)  
VALUES (1111, 'ÖDÖN', 42, 'SAJTKUKAC')
```

DML: módosítás

- Legújabb dolgozónknak nem adtunk fizetést, főnöke sincs még.

```
UPDATE emp2
```

```
SET sal=500, mgr=7788
```

```
WHERE empno=1111
```

- Fontos a feltétel megadása, különben mindenhol átírja!!!

DML: módosítás

- Adjunk Ödönnek jutalékot is.
(legyen 10 dollár)

DML példa I

- Adjunk az **emp2** táblához egy *cardnumber* nevű mezőt, amely a dolgozó beléptető kártyájának 5 jegyű azonosítóját tartalmazza. A kártyaszámot a dolgozói azonosítóból képezzük a következő módon:

$$\text{cardnumber} = 10.000 + \text{empno}$$

Megoldás

```
ALTER TABLE emp2  
ADD cardnumber NUMBER(5)
```

```
UPDATE emp2  
SET cardnumber=10000+empno
```

DML példa 2

- Adjunk az **emp2** táblához egy *income* nevű szöveges mezőt, amely a dolgozó jövedelmének mértékét tárolja: akinek a jövedelme 2500 dollár feletti, legyen az income értéke "HIGH", a többieknek "LOW". A megfelelő megszorítással biztosítsuk, hogy mást ne is lehessen beírni ide.

Megoldás – DDL rész

```
ALTER TABLE emp2  
ADD income VARCHAR2(4)
```

```
ALTER TABLE emp2  
ADD CONSTRAINT emp2_ck  
CHECK (income IN ('HIGH', 'LOW'))
```

Megoldás – DML rész

```
UPDATE emp2  
SET income='HIGH'  
WHERE sal+NVL(comm,0)>2500
```

```
UPDATE emp2  
SET income='LOW'  
WHERE income IS NULL
```

DML: törlés

- A cég felszámolta sajtkészítő üzletágát, szegény Ödönt elbocsátották. Töröljük a táblából.

```
DELETE FROM emp2  
WHERE empno=1111
```

- Körültekintően határozzuk meg a feltételt!

DML: törlés

- Töröljük a sajtkészítők részlegét a **dept2** táblából.

DML: Gyakorló feladatok

- Kapjon 20% fizetésemelést minden olyan dolgozó, aki 1981-ben született és nincs jutaléka.

update emp set sal=sal *1.2 where
extract(year from hiredate)=1981 and
comm is null;

DML: Gyakorló feladatok

- Listázzuk ki azoknak a dolgozóknak a nevét, települését, akiknek a neve S-el kezdődik és T-re végződik vagy a településük tartalmaz szóközt.

```
SELECT ename, loc FROM emp INNER JOIN  
dept USING(deptno) WHERE ename LIKE 'S%T'  
OR loc LIKE '% %';
```

DML: Gyakorló feladatok

- Készítsünk egy FULL táblát, mely az emp és a dept táblák adatait tartalmazza.

```
CREATE TABLE emp2 AS SELECT * FROM emp  
INNER JOIN dept USING(deptno);
```

- Módosítsuk a FULL táblában a települést BUDAPEST-re, ahol nincs jutalék és a munkakör A betűvel kezdődik.

```
UPDATE emp2 SET loc = 'BUDAPEST' WHERE  
COMM IS NULL AND JOB LIKE 'A%';
```

DML: Gyakorló feladatok

- --Hozzunk létre egy myEmp táblát (név, születési dátum, munkakör).
- CREATE TABLE myEmp(
 - név VARCHAR2(30) PRIMARY KEY,
 - szüldátum DATE,
 - munkakör VARCHAR2(15)
-);
- --Adjunk hozzá a táblához egy új dolgozót (tetszőleges adatokkal).
- INSERT INTO myEmp VALUES ('Bob', TO_DATE('1979-10-01', 'YYYY-MM-DD'), 'Asszisztens');
- --Módosítsuk a munkakörét.
- UPDATE myEmp
- SET munkakör = 'Irodavezető'
- WHERE név = 'Bob';
- --Töröljük ki ezt a dolgozót.
- DELETE FROM myEmp
- WHERE név = 'Bob';

DML: Gyakorló feladatok

- Készítsen egy laptopok nevű táblát!
- Tábla oszlopai tételesen:
 - márka 32 hosszúságú szöveges mező, mely nem lehet üres
 - megnevezés 32 hosszúságú szöveges mező, mely nem lehet üres
 - memoria egész szám típusú 2 hosszúságú mező, melyre check megszorítást kell bevezetni és az értéke 4 és 64 közé kell essen
 - videokartya szöveges 10 hosszúságú mező, melyet check megszorítással ellenőrizni kell ,hogy vagy 'dedikalt' vagy 'integralt' értéket vehet fel
- Továbbá a táblában állítsa be elsődleges, összetett kulcsnak a márka és megnevezés oszlopot!
- Illesszen be tetszőleges HELYES adatokat a táblába!

DML: Gyakorló feladatok

```
Create table laptop(  
    marka varchar2(32) NOT NULL,  
    megnevezes varchar2(32) NOT NULL,  
    memoria number(2),  
    videokartya varchar2(10),  
    constraint laptop_pk primary key (marka,megnevezes),  
    constraint laptop_ck check (videokartya in ('dedikalt','integralt')),  
    constraint laptop_ck2 check (memoria between 4 and 64)  
);
```

```
insert into laptop values ('Apple','Macbook Air',8,'integralt');  
insert into laptop (marka,megnevezes,memoria,videokartya) values  
('Lenovo','Ideapad 530s',4,'integralt');
```

```
select * from laptop;
```

Gyakorló feladat

- Listázzuk ki azoknak a dolgozóknak a nevét, települését, akiknek a neve S-el kezdődik és T-re végződik vagy a településük tartalmaz szóközt.

```
SELECT ename, loc FROM emp INNER  
JOIN dept USING(deptno) WHERE ename  
LIKE 'S%T' OR loc LIKE '% %';
```

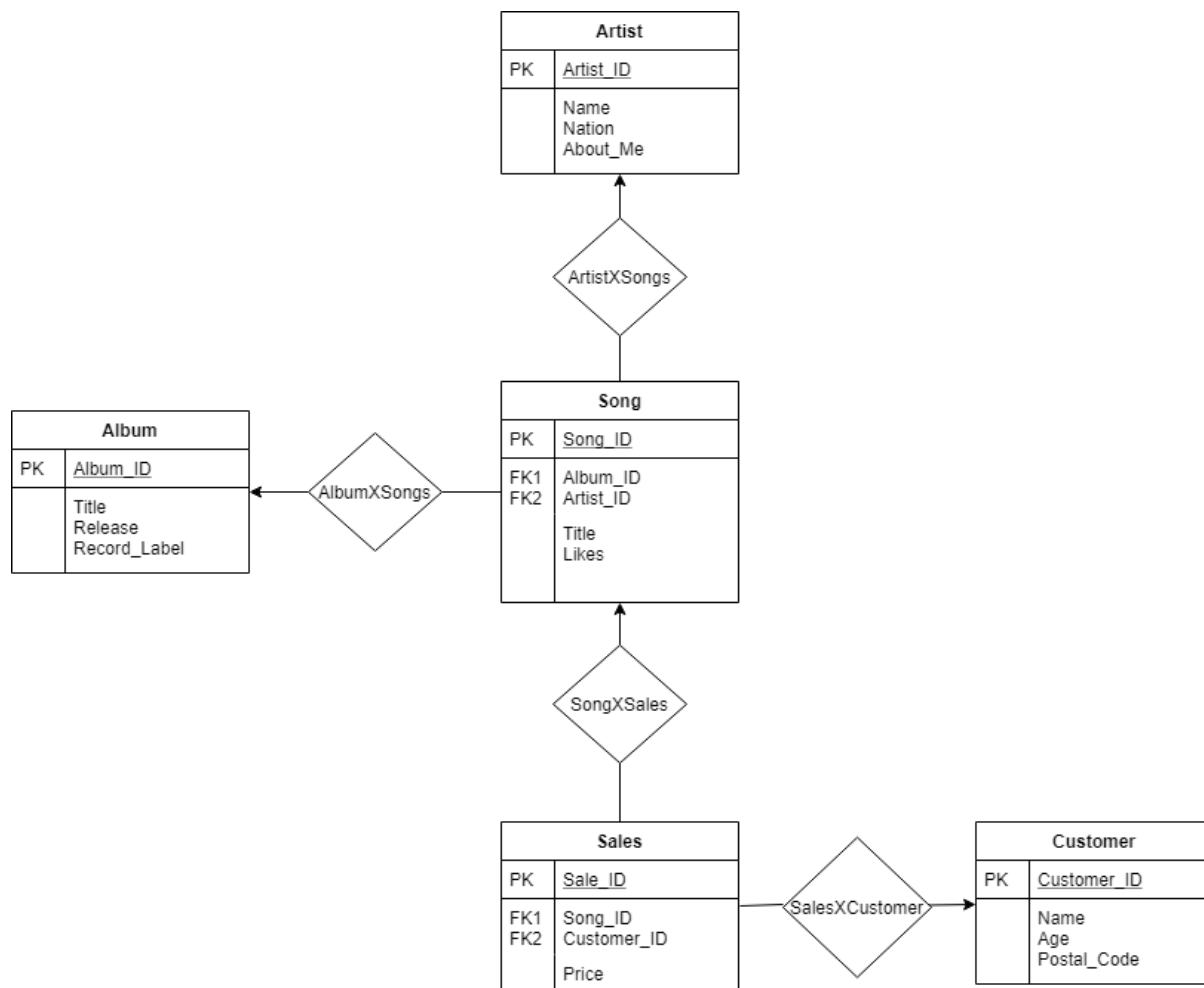


Adatbázisok

6. gyakorlat:

Gyakorlás

A gyakorló adatbázis



Előkészületek

- Jelentkezzünk be az online felületen:
<https://apex.oracle.com/en/>
- SQL Workshop menüpont
- SQL Scripts menüpont
- Upload gomb
- Nyomjuk meg a Run gombot
- Menjünk vissza az SQL Commands menübe

Feladatok

1. Listázzuk ki a tizenéves vásárlókat!

```
SELECT * FROM customer WHERE age < 20
```

2. Listázzuk ki azokat a dalcímeket, amelyeknek a kedvelésük 2 vagy 4. Rendezzük kedvelés szerint csökkenő sorrendbe!

```
SELECT title, likes FROM song WHERE likes IN(2, 4) ORDER BY likes DESC;
```

3. Listázzuk a 10 karakternél hosszabb dalcímeket!

```
SELECT title FROM song WHERE LENGTH(title) > 10;
```

4. Listázzuk azoknak az előadónak a nevét, nemzetiségét, kiknek a nemzetiségében szerepel a "United" szó. Az oszlopnevek magyarul jelenjenek meg!

```
SELECT name Előadó, nation Nemzetiség FROM artist WHERE nation LIKE '%United%';
```

Feladatok

5. Listázzuk azoknak a vásárlóknak a nevét, életkorát akik 20 és 25 év közöttiek!

```
SELECT name, age FROM customer WHERE age BETWEEN 20 AND 25;
```

6. Írjuk ki az adatbázisban lévő dalok összértékét!

```
SELECT SUM(price) FROM sales;
```

7. Mennyi a dalok kedvelésének az átlaga? Használjunk álnevet és 2 tizedes pontosságra kerekítsük!

```
SELECT ROUND(AVG(likes),2) Átlag FROM song;
```

Feladatok

8. Írjuk ki irányítószám alapján az életkorok átlagát! Kerekítsünk egészre!

```
SELECT postal_code, ROUND(AVG(age)) FROM customer GROUP BY postal_code;
```

9. Az előző feladatt bővítsük ki úgy, hogy odaírjuk, hogy az egyes irányítószám alatt hányan laknak!

```
SELECT postal_code, ROUND(AVG(age)), COUNT(*) FROM customer GROUP BY postal_code;
```

10. Írjuk ki a legfiatalabb vásárló életkorát!

```
SELECT MIN(age) FROM customer;
```

11. Listázzuk ki, hogy 2000 után hány albumot adtak ki az egyes kiadók!

```
SELECT record_label, COUNT(title) FROM album WHERE release > 2000 GROUP BY record_label;
```

Feladatok

12. Listázzuk ki azokat a kiadókat, akik 1-nél több albumot adtak ki!

```
SELECT record_label, COUNT(title) FROM album GROUP BY  
record_label HAVING COUNT(title) > 1;
```

13. Listázzuk ki melyik dalnak ki az előadója!

```
SELECT title, name FROM artist INNER JOIN song USING(artist_id);
```

14. Melyik dal mennyibe kerül?

```
SELECT title, price FROM song INNER JOIN sales USING(song_id);
```

15. Listázzuk ki a dalokat az album és az előadó nevével együtt.
Használjunk álneveket!

```
SELECT a.title Album, s.title Dal, a.name Előadó FROM album a  
INNER JOIN song s USING(album_id) INNER JOIN artist a  
USING(artist_id);
```

Feladatok

16. Listázzuk azokat a magyar dalokat, amelyek legalább 20 lájkot kaptak! Rendezzük az eredményt lájkok szerint csökkenő sorrendbe!

```
SELECT title, likes FROM song INNER JOIN artist  
USING(artist_id) WHERE likes > 19 AND nation =  
'Hungary' ORDER BY likes DESC;
```

17. Listázzuk azokat a 'J' vagy 'T' kezdőbetűs előadókat, akik 2010 és 2020 között adtak ki dalokat! Listázzuk a megjelenés évét is, az eredményhalmazt rendezzük ABC sorrendbe!

```
SELECT name, release FROM artist INNER JOIN song  
USING(artist_id) INNER JOIN album USING(album_id)  
WHERE release BETWEEN 2010 AND 2020 AND (name  
LIKE 'J%' OR name LIKE 'T%') ORDER BY name;
```

Feladatok

18. Listázzuk a dalok címét, lájkjuk számát, amelyek legalább 20 lájkot kaptak és 3-5 dollárba kerülnek? Egy cím csak egyszer jelenjen meg és az eredményt rendezzük cím alapján csökkenő sorrendbe!

```
SELECT DISTINCT title, likes FROM song INNER JOIN sales  
USING(song_id) WHERE price BETWEEN 3 AND 5 AND  
likes >= 20 ORDER BY title DESC;
```

19. Mennyit költöttek az egyes vásárlók dalokra? Listázzuk ki a nevüket és az elköltött összeget!

```
SELECT name, SUM(price) FROM sales INNER JOIN  
customer USING(customer_id) GROUP BY name;
```

Feladatok

20. Listázzuk ki hogy melyik dalt hányszor vásárolták meg!

```
SELECT title, COUNT(*) FROM song INNER JOIN sales  
USING(song_id) GROUP BY title;
```

21. Az előző feladatot egészítsük ki azzal, hogy a vásárlók átlagéletkorát is írjuk ki 2 tizedesre kerekítve!

```
SELECT title, COUNT(*), ROUND(AVG(age),2) FROM song INNER  
JOIN sales USING(song_id) INNER JOIN customer  
USING(customer_id) GROUP BY title;
```

22. Módosítsuk az előző feladatot úgy, hogy csak azok jelenjenek meg, akik 1-nél több dalt vásároltak!

```
SELECT title, COUNT(*), ROUND(AVG(age),2) FROM song INNER  
JOIN sales USING(song_id) INNER JOIN customer  
USING(customer_id) GROUP BY title HAVING COUNT(*) > 1;
```

23. Hány db. 4-5\$ közötti értékű dalt vásároltak meg?

```
SELECT COUNT(*) FROM song INNER JOIN sales USING(song_id)  
WHERE price BETWEEN 4 AND 5;
```


Feladatok

24. Listázzuk ki, melyik album hány dalt tartalmaz albumcím alapján!

```
SELECT a.title, COUNT(*) FROM album a INNER JOIN song s  
USING(album_id) GROUP BY a.title;
```

25. Melyik irányítószámról hány dalt vásároltak, milyen összegben és milyen átlagéletkorral? Használjunk álneveket, kerekítsük az átlagot és rendezzük az eredményhalmazt irányítószám alapján növekvő sorrendbe!

```
SELECT postal_code IrSzám, COUNT(*) EladottDB,  
SUM(price) Bevétel, ROUND(AVG(age),2) ÁtlagKor FROM  
sales INNER JOIN customer USING(customer_id) GROUP BY  
postal_code ORDER BY postal_code;
```

Feladatok

26. Készítsünk egy táblát 'eladasok' néven, amely megjeleníti a vásárlók:nevét, életkorát, irányítószámát, mennyi pénzt költött dalokra, magyar oszlopnevekkel

```
CREATE TABLE eladasok AS
```

```
SELECT name nev, age kor, postal_code irszam, SUM(price)  
vasarlas FROM sales INNER JOIN customer
```

```
USING(customer_id) GROUP BY name, age, postal_code;
```

27. Adjunk megszorítást a táblához, ami nem enged kiskorúakat beszúrni a táblába!

```
ALTER TABLE eladasok ADD CHECK (kor >= 18);
```

Feladatok

28. Próbáljuk meg beszúrni a következő adatokat:

Kornél, 16 éves, irányítószáma: 6721, a vásárlása összértéke: 0

```
INSERT INTO eladasok VALUES ('Kornél', 16, 6721, 0);
```

29. Telik az idő, minden vásárló öregedett +1 évet!

```
UPDATE eladasok SET kor = kor + 1;
```

30. Töröljük a 30 év feletti vásárlókat!

```
DELETE FROM eladasok WHERE kor >= 30;
```