Gyakorlati óravázlat 02 – Hogy kerülnek adatok egy adatbázisba?

1. A mai órán két fő feladatunk lesz, táblát létrehozni és abba adatokat belepakolni
2. Egy adatbázis általában nem egyetlen táblából áll, de a mai órán csak az alapokat nézzük meg, hogy hogyan tudunk egyetlen egyszerű táblát létrehozni illetve kitörölni. (CREATE TABLE, DROP TABLE)
3. Az adatbevitel utasítása az INSERT. A mai órán importálással fogunk adtokat írni egy táblába. Legyünk tudatában annak, hogy amikor az SQL Developerben importálunk, akkor valójában az történik, hogy az importálandó rekordoknak megfelelő számú INSERT utasítás jön létre, majd ezeket futtatja le a program.
4. Ahhoz, hogy egy táblát létrehozzunk, meg kell adnunk annak oszlopait és a kényszereket. A minimálisan szükséges adatok: a tábla neve, az oszlopok neve és adattípusa.
5. Fontosabb adattípusok megismerése:
   * (N)CHAR és (N)VARCHAR/(N)VARCHAR2 – N = nemzeti karaktertípus
   * NUMBER – ennek a formátuma, működése
   * DATE / TIMESTAMP
   * LOB = long object, CLOB/BLOB
   * ROWID vagy UROWID – sorazonosító a tárolási cím megjelenítésére és kezelésére nem valódi adattípus.

\_\_\_\_\_\_\_\_\_\_\_\_

**Kiegészítő információk gyakvezéreknek**

NUMBER(n, m) általános típus:

CREATE TABLE numberteszt(

n1 NUMBER

, n2 NUMBER(3,2)

, n3 NUMBER(2,3)

, n4 NUMBER(3,-2)

, n6 NUMBER(\*,2)

, n7 NUMBER(8)

--, n8 number(3,\*) -- ilyet állat nincs.

);

DROP TABLE numberteszt PURGE;

**Tudni kell**

* max. 38db számjegy 10-es számrendszerben! (max. 22 bájt)
* Jellemzően az m nemnagyobb, mint az n
* -9,99 és 9,99 közötti számnál NUMBER(3,2) írandó
* az n egy 1 és 38 közötti egész szám

**Jó, ha tudod – de nem számonkérendő –, hogy**

* más adatbázis-kezelőben van INTEGER, FLOAT, DOUBLE
* előfordulhat BOOLEAN, SMALLINT, TINYINT is.
* INTEGER = az operációs rendszer alapműveleti egységes. 32 bites operációs INTEGER = 32 bit, 64 bites operációs rendszer INTEGER = 64 bit. Nem egységes ebben az informatika! LONG az 2db INTEGER összefűzve, SMALLINT egy fél INTEGER. TINYINT = CHAR(1), azaz 1 bájt. Van, ahol éppen ezért BYTE-nak hívják.

**Egyáltalán nem kell tudni, csak mondom... (a fentieket is nehezen értik meg a hallgatók, ha ezzel is kiegészíted, gondok lesznek belőle!)**

* m egy -84 és 127 közötti szám lehet(!)
* NUMBER(3,-2) esetében kerekítés alkalmaz az Oracle, így például 745,89 valójában 700 lesz – vagyis m < 0 esetén a tizedesvesszőtől visszafelé m db „0“ írandó.
* Hasonlóan NUMBER(3,5) a -0,00999 és a 0,00999 közötti számokat jelöli, mivel m > n esetén a tizedesvesszőtől után (m-n) db „0“ írandó
* INTEGER = NUMBER(\*,0)
* Az m helyén \* nem állhat!
* NUMBER(n) = NUMBER(n,0)
* NUMBER = NUMBER(\*,\*) – logikai értelmben, mert ilyet nem lehet írni

Akit részletesebben érdekelnének az adattípusok, azok itt nézhetnek utána: https://docs.oracle.com/database/121/SQLRF/sql\_elements001.htm#SQLRF0021

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Feladatok**

Megmutathatjuk a dokumentációban a CREATE TABLE utasítást (16. fejezet), kicsit görgessünk bele, hogy mennyi minden van itt. Aztán menjünk vissza az elejére és kattintsunk az Examples linkre. Csak az első példára nézzünk rá. Az óra végére az abban szereplő dolgokat már ismerni fogják.

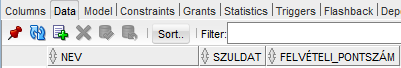
1. Hozzunk létre egy egyszerű táblát 2 oszloppal: alkalmazott(nev,szulev)
   * a nev adattípusa szöveg: VARCHAR2(40)
   * a szulev adattípusa szám, tételezzük fel, hogy 4 számjeggyel tároljuk az évet, akkor ez 4 jegyű egész szám: NUMBER(4)

CREATE TABLE alkalmazott(

nev VARCHAR2(40)

,szulev NUMBER(4)

);

* + Megnézhetjük a connecitons ablakban, ha a Tables-re ráfrissítünk, akkor látni fogjuk az ALKALMAZOTT táblánkat. (Következő órától megnézni is sql-lel fogjuk, ma ez a rész még menüből megy)
  + Ha rákattintunk, akkor látszanak az oszlopaink tulajdonságai, kattintgassuk végig a füleket, nem sok mindent fogunk még látni. Utolsó fül: SQL: itt az SQL Developer a kész objektumból visszafejti, hogy milyen SQL utasítással lehet létrehozni. Ez általában bonyolultabb, mint amivel mi létrehoztuk.
  + A data fülön látszanának a benne lévő adatok
  + 
  + A zöld plusz-ra kattintva tudunk új adatot bevinni. Mindenki vigye fel saját magát! Az adatbevitelt a zöld pipával (Commit changes) tudjuk befejezni. Alul látni fogjuk, hogy a lefutó utasítás egy INSERT utasítás.
  + Vigyük be magunkat még egyszer pontosan ugyanazokkal az adatokkal! Lehetséges? Miért ne lenne az? Hát nem arról volt szó, hogy egy táblának minden sora egyedi? – ezen a ponton mesélhetünk egy kicsit a ROWID-ről, nem kell túl részletesen, csak hogy tudjunk a létezéséről. Pl. töröljük ki a másodjára bevitt rekordunkat. A lefutó utasításban a ROWID-vel hivatkozza, hogy melyik rekordot törölte ki.
  + Menjünk vissza a worksheetre és töröljük a táblát:

DROP TABLE ALKALMAZOTT PURGE; -- a purge azért kell, hogy teljesen törlésre kerüljön a tábla, ne csak a lomtárba kerüljön.

1. Hozzuk létre az alkalmazott táblát újra, de most állítsuk be, hogy a név legyen elsődleges kulcs.

CREATE TABLE alkalmazott(

nev VARCHAR2(40) PRIMARY KEY

,szulev NUMBER(4)

);

* + frissítsünk rá a táblára, mi változott?
  + oszlopok (Columns) fülön a NULLABLE tulajdonság a NEV-nél No, a többinél maradt Yes
  + A Constraints fülön lett egy SYS\_... nevű Primary Key típusú feltételünk – a nevet a rendszer automatikusan adta. Ha akarjuk, van lehetőség a név megadására is, majd egy következő példánál megnézzük
  + Az Indexes fülön lett egy ugyanolyan nevű UNIQUE indexünk
  + Most próbáljuk felvinni magunkat mindkétszer azonos névvel – hibaüzenet „Egyediségre vonatkozó megszorítás nem teljesül”
  + Menjünk vissza a worksheetre és töröljük a táblát:

DROP TABLE ALKALMAZOTT PURGE;

1. Hozzuk létre az alkalmazott táblát újra de most állítsuk be, hogy a név és a születési év együtt legyen elsődleges kulcs.
   * Nem jó, ha mindkettő mellé odaírjuk, hogy Primary Key, mert az azt jelentené, hogy két elsődleges kulcsot akarunk létrehozni. Ez hibaüzenet mellett nem is fog létrejönni.
   * Két oszlopból álló elsődleges kulcs – nem oszlopkényszer, hanem táblázat kényszer, csak az oszlopok megadása után tudjuk megadni. Itt kötelező neki nevet adni. (Ha oszlopkényszernek szeretnénk nevet adni, akkor ugyanúgy ki kell írni elé a CONSTRAINT kulcsszót és a kényszer nevét, a folytatás ugyanaz)

CREATE TABLE alkalmazott(

nev VARCHAR2(40)

,szulev NUMBER(4)

);

1. Dobjuk el az alkalmazott táblát, majd hozzuk létre újra, de most legyen egy azonosítónk is, és az azonosító és a név+születési év is kulcs legyen.
   * az azonosító neve AID legyen, 3 jegyű szám.
   * Lehet mindkettő elsődleges? – Próbáljuk ki, nem fog lefutni.

DROP TABLE ALKALMAZOTT PURGE;

CREATE TABLE alkalmazott(

aid NUMBER(3) PRIMARY KEY

,nev VARCHAR2(40)

,szulev NUMBER(4)

,CONSTRAINT key\_nev\_szulev PRIMARY KEY (nev,szulev) --nem jó!

);

* + Akkor mi a megoldás? A kulcs azt jelenti, hogy unique és not null, úgyhogy ezt a kettőt adjuk meg együtt

CREATE TABLE alkalmazott(

aid NUMBER(3) PRIMARY KEY

,nev VARCHAR2(40) NOT NULL

,szulev NUMBER(4) NOT NULL

,CONSTRAINT key\_nev\_szulev UNIQUE (nev,szulev)

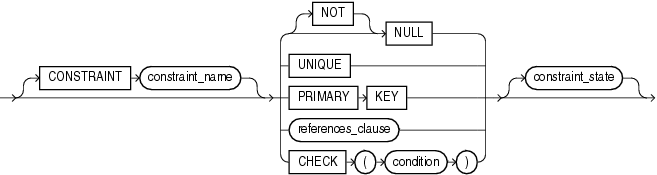
);

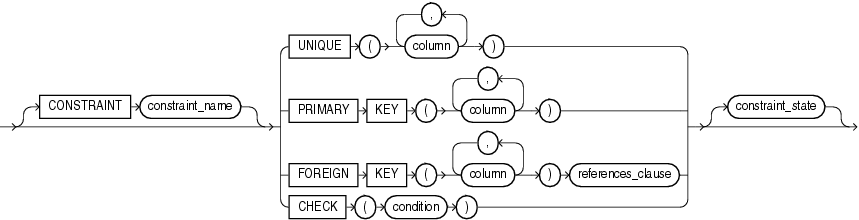
Eddig megismert kényszerek: Primary Key, Unique, Not null

Ezen kívül lehet még: check: lehet szűkíteni az adattípuson belül az értéktartományt. Lehet oszlop és tábla kényszer is. Tábla kényszer is vonatkozhat egyetlen oszlopra, de lehet több oszlopra együttesen vonatkozó feltétel is: pl. egyik oszlop nagyobb mint a másik.

Lehet még reference: idegen kulcs létrehozása. Ilyet a mai órán nem csinálunk.

Dokumentációban: Database SQL Language Reference, ott a tartalomjegyzékben kattintsunk a 8. pontra: Common SQL DDL Clauses (gyakori sql ddl egységek). Azon belül Constraints (kényszerek). Az oszlop kényszert inline constraint-nek nevezi, a tábla kényszert pedig out\_of\_line-nak. Ide teszem a fő ábráját mindkettőnek. Át lehet kicsit beszélni, hogy tudják értelmezni ezeket az ábrákat. Az ábra végén, az opcionáli constraint\_state-et nem kell tudni.





1. **Önállóan**: Dobjuk el újra a táblát, és hozzuk létre újra úgy, hogy a SZULEV oszlopra adunk CHECK feltételt, hogy 1900 és 2000 közötti számokat fogadunk csak el.

DROP TABLE alkalmazott PURGE;

CREATE TABLE alkalmazott(

aid NUMBER(3) PRIMARY KEY

,nev VARCHAR2(40) NOT NULL

,szulev NUMBER(4) NOT NULL CHECK (szulev BETWEEN 1900 AND 2000)

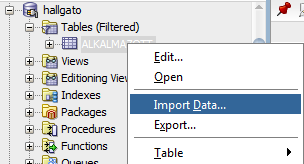
,CONSTRAINT key\_nev\_szulev UNIQUE (nev,szulev)

);

Feltételek megadása a dokumentációban: Database SQL Language Reference, ott a tartalomjegyzékben kattintsunk a Conditions-ra, nem kell most még mindegyiket érteni, ismerni, megnézni. Ami egyelőre érdekes lehet belőle: Comparison conditions (összehasonlító feltételek), Logical conditions (logikiai feltételek), Between conditions (között feltétel)

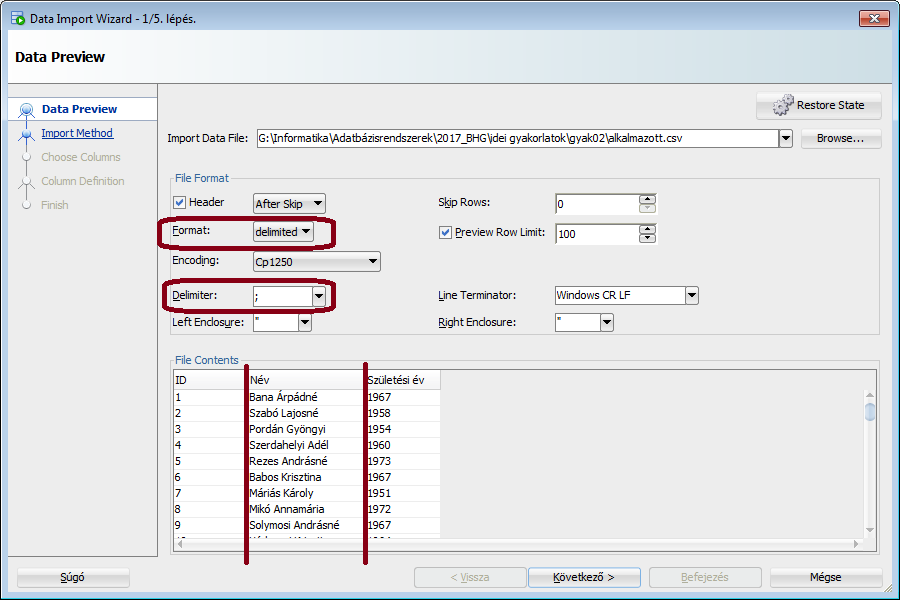
1. Importáljunk adatokat az utoljára létrehozott táblánkba. (alkalmazott.csv)

Jobb egérgomb a tábla nevén, gyorsmenüben Import Data

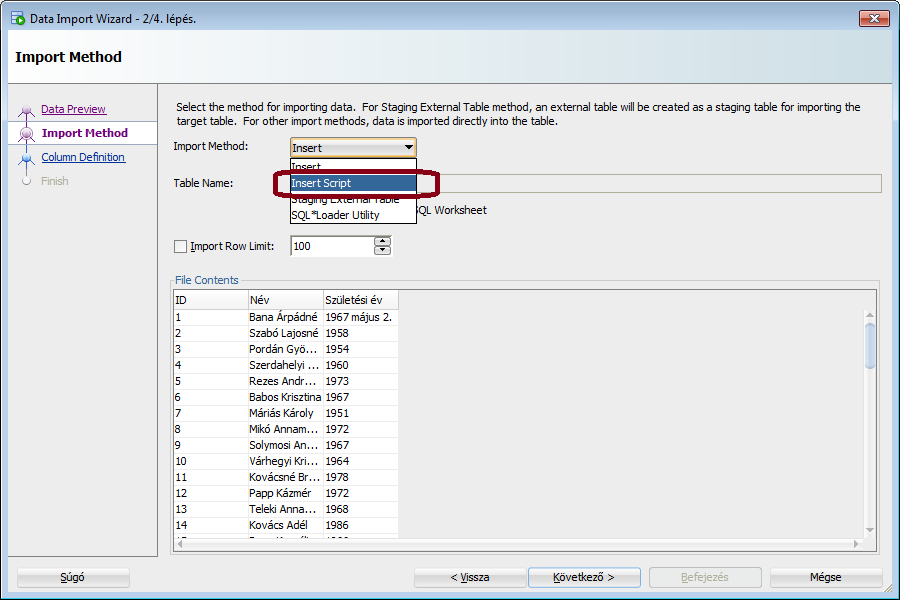


csv fájl: pontosvesszővel tagolt szövegfájl, a Format-nál meg kell adni, hogy delimited (tagolt) és a delimiter (tagoló) a pontosvessző (meg lehet még adni, hogy a szövegek idézőjelek között legyenek, ez az eszköz arra, ha a szöveg közben lenne olyan karakter, ami egyébként delimiter)

Nézzék mindig a mintát lent, ott látszik, hogy jól sikerült-e szétvágni az oszlopokat, esetleg változtatni kell-e a karakterkódoláson

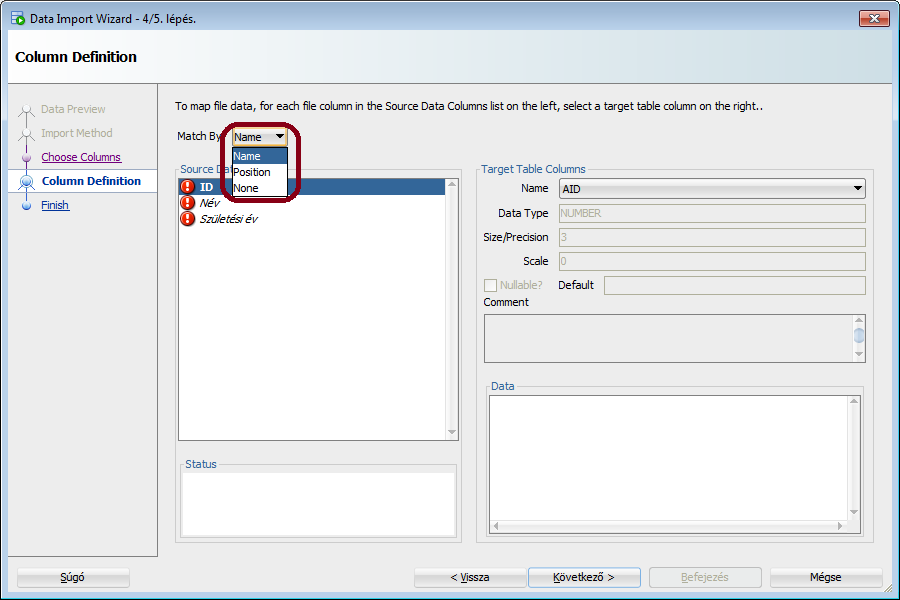


Az import method-nál állítsuk át Insert-ről Insert Script-re, mert akkor fogjuk látni, hogy valójában insert utasítások halmaza jön létre.

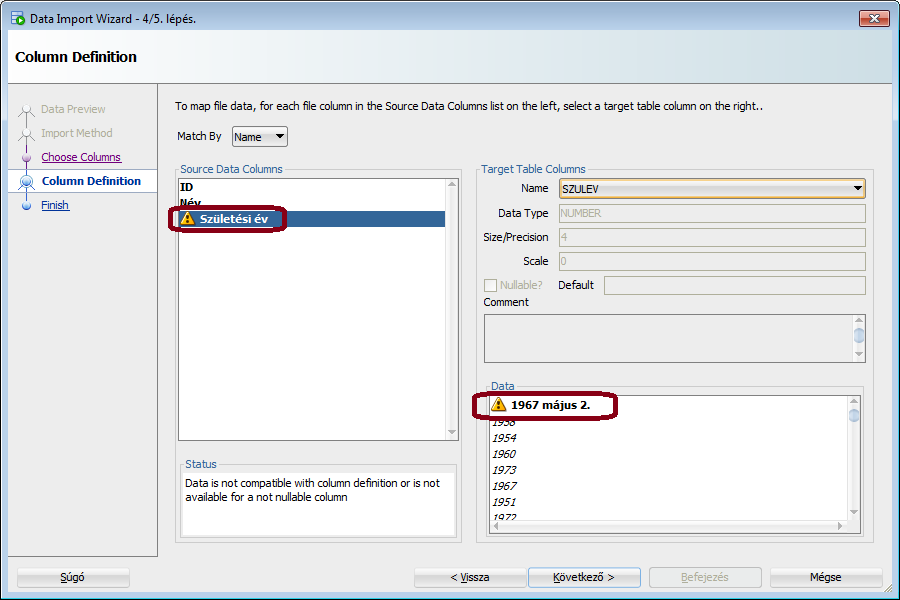


A 3. lépésben lehetne megadni, ha nincs szükségünk minden oszlopra (de most szükségünk van)

A 4. lépésben kell megmondani, hogy a forrásfájl melyik oszlopa a táblázat melyik oszlopába kerüljön. Mivel nem stimmelnek az oszlopnevek, alapértelmezetten mindegyiket az első oszlopba akarja beilleszteni, ezért vannak a piros felkiáltójelek, mert azt nem lehet. (Előfordulhatna, hogy vannak azonos nevű oszlopok, de rosszul, úgyhogy ez nem életbiztosítás így). Ha a pozíciót választjuk, akkor berakja sorrendben, de ezzel legyünk óvatosak, mert előfordulhat, hogy nem jó az oszlopsorrend. Az a biztos, ha az utolsó lehetőséget választjuk. Akkor is mindent az első oszlopba akar illeszteni, de aztán egyesével végig tudunk menni a mezőkön és ki tudjuk választani a megfelelőt. A forrás ablakban egyrészt a piros felkiáltójel mutatja, hogy baj van, másrészt a dőlt betű mutatja, hogy még nem ellenőriztük az adott mezőt.



A születési év oszlopban szándékosan az első értéket elrontottam. Teljes dátum szerepel, ami természetesen nem fér be az évszám helyére. Ezt jelzi a sárga felkiáltójel a mezőnévnél is, és az adatok listájában is. Érdemes végiggörgetni, hogy van-e másik is.



A hiba ellenére befejezhetjük az importálást, mert valójában csak az Insert utasítások fognak létrejönni, amin még futtatás előtt tudunk módosítani.

Létrejön egy új worksheet, rajta a 62 darab insert utasítás. Az elsőt javítsuk ki, töröljük ki a dátum felesleges elemeit, majd a run scrip gombbal (F5) futtassuk le (mivel új munkalap meg kell adnunk, hogy melyik kapcsolaton futtassa (még akkor is, ha csak egy kapcsolatunk van)

Számformátumok és dátumformátumok megbeszélése. Dokumentációban a 2. pont: Basic Elements of Oracle SQL (az Oracle SQL alapelemei). Azon belül a Format models. Beszéljünk itt a timestamp-ről is. **A sysdate-hez hasonlóan systimestamp** is létezik, ezt is mondjuk el! Nem kell a formátum modelleket fejből tudni, meg ezekkel még rengeteget fogunk foglalkozni a következő órán, de a dátum formátumból az év, hónap, nap, óra, perc különböző kiírásait kötelező fejből is tudni. Sprintben és minimumtesztben is lehetnek.

1. **Önállóan**: Hozzunk létre egy táblát ÁRFOLYAM néven. A táblának 3 oszlopa legyen. Egy dátum, ez legyen kulcs, egy JPY oszlop a japán jen árfolyamának tárolására, egy jegyű számot lehessen benne tárolni 4 tizedesjeggyel, feltételnek állítsuk be, hogy 0 és 5 között legyen és egy másik USD oszlop az amerikai dollár árfolyamának tárolására. 3 jegyű számot lehessen tárolni 2 tizedesjeggyel és állítsuk be, hogy 100 és 400 közötti számokat fogadjon csak el. Ha kész a tábla, importáljuk az arfolyam.csv fájl adatai közül csak a szükségeseket. Az importáláshoz a scriptet hozzuk létre, majd azt futtassuk!
2. **Önállóan**: Hozzunk létre egy táblát RENDELÉS néven. A tábla oszlopai: sorszám (max. 6 jegyű egész szám), dátum (időbélyeg), alapértelmezetten a rendszer órájából vegye az értéket, vevő (40 karakteres szöveg), fizetendő összeg (max 6 jegyű egész szám). A táblában a sorszám és a dátum is legyen kulcs. Vigyünk be kézileg 2-3 adatot a táblázatba.

Az utolsó feladatban a default érték megadása újdonság, de a create table example-ből jó lenne ha rájönnének. Küzdjenek kicsit.

# Kérdések a következő órán

adattípusokra, számformátumokra vonatkozó kérdések

CREATE, INSERT, DROP utasítások jelentése

dátumformátumra vonatkozó kérdések (importálandó adatok képe, mi legyen a dátumformátum?)