Térbeli adatbázisok kezelése

Jegyzet geoinformatika mesterszakos hallgatóknak

Dr. Ungvári Zsuzsanna

egyetemi adjunktus

Térképtudományi és Geoinformatikai Intézet

Tartalomjegyzék

Előszó, követelmények

A jegyzet a Térbeli adatbázisok című elsőéves, Geoinformatika MSc kurzushoz készült. Azonban fontos tudni, ahhoz, hogy sikeresen boldoguljunk a térbeli lekérdezések világában, szükség van alapismeretekre az adatbázis-kezelés, valamint a térinformatika terén is egyaránt.

Azt, hogy mit illik tudni adatbázis-kezelés témakörben, az első fejezet jól összefoglalja. Az ELTE térképész és geoinformatikus képzés sajátosságai miatt földtudományi alapszakon bevezető tantárgy az Adatbázis-kezelés, amelyen MySQL rendszerben sajátítjuk el egy adatbáziskezelő működését. Mivel a MySQL térbeli eszköztára egyelőre kevésbé professzionális, mint a PostgreSQL+PostGIS rendszer, ezért térünk át az utóbbi rendszerre. Az árnyalatnyi „nyelvjárásbeli" különbségek leküzdésében segít az első fejezet.

Mivel térbeli adatokkal dolgozunk, ezért nem hátrány, ha az olvasó járatos bármely térinformatikai szoftverben, illetve ismeri az ehhez kapcsolódó adatmodellek és függvények elméleti hátterét is. Itt elsősorban a geometriai illetve a térbeli műveleteket végző függvények ismeretére gondolok. A jegyzet szempontjából a QGIS ismerete praktikusabb, mivel némely fejezetben egy-egy művelet erejéig használni fogjuk.

Ezen fejezet végén pedig álljon itt néhány link, és hivatkozás, amelyek segítségével az olvasó fejlesztheti alapismereteit.

Megjegyzés: a jegyzethez PostgreSQL 13-as és a PostGIS 3.1 verziót használtam. Árnyalatnyi eltérések lehetnek a lekérdezések szintaktikájában más verziójú PostgreSQL+ PostGIS kiegészítő esetén.

Jó tanulást kívánok!

A szerző

Jó/illik tudni:

- Gede Mátyás: Bevezetés a MySQL-be, gyakorló feladatsor.
  http://mercator.elte.hu/~saman/hu/okt/db.html
- Elek István: Adatmodellek, térképek, információs rendszerek.
  Az adatbázis-kezelés elméletéről, vektos és raszteres adatmodell
  http://mapw.elte.hu/elek/adatmodellek.pdf
- Ungvári Zsuzsanna: Bevezetés a QGIS-be. Jegyzet.
  http://mercator.elte.hu/~ungvarizs/

Lektorálta: Dr. Gede Mátyás.

1. FEJEZET: Hogyan mondjuk PostgreSQL-ben?

Ebben az első, bevezető fejezetben egy összehasonlító táblázatban tárgyalom, hogy az egyes lekérdezések, kulcsszavak vagy klauzulák hogyan működnek MySQL-ben és PostgreSQL-ben. Bár az SQL egy szabvány, azért adódnak a két adatbáziskezelő rendszer között amolyan „nyelvjárásbeli" különbségek, időnként éppen annyi, hogy ne fusson le a lekérdezés. Sárgával emeltem ki a változásokat ahhoz képest, mint amit a MySQL-ben megszokhattunk.

Mi akarok csinálni?[vagy függvény neve MySQL-ben]	MySQL	PostgreSQL
ORDER BY két mező szerinti rendezés	`select * from kiadvany order by kv_ev desc, kv_kiado`	`select * from kiadvany order by kv_ev desc, kv_kiado`
WHERE 1 Feltétel számmal	`select * from katalogus where kt_ar>2000;`	`select * from katalogus where kt_ar>2000;`
1 feltétel szöveggel	`select * from konyvtaros where ks_beosztas='vezető'` `select * from konyvtaros where ks_beosztas= "vezető"`	`select * from konyvtaros where ks_beosztas='vezető'` **Figyelem: beállítás kérdése, hogy a karakterkészlet(collation), amit használunk case sensitive vagy sem. Ha case sensitive a karakterkészlet, akkor nemcsak az idézőjel típusa számít, hanem a pontos szövegegyezés: kis/nagybetű érzékeny az egyenlőségjel!!!**

1 feltétel dátummal	`select * from konyvtaros where ks_belep>='1992-01-03'`
	`select * from konyvtaros where ks_belep>='1992.01.03'`
	`select * from konyvtaros where ks_belep>="1992-01-03"`
	`select * from konyvtaros where ks_belep>="1992.01.03"`
		`select * from konyvtaros where ks_belep>='1992-01-03'`
		`select * from konyvtaros where ks_belep>='1992.01.03'`

IN	`select * from konyvtaros where ks_beosztas in ('Pultos', 'Raktáros')`	`select * from konyvtaros where ks_beosztas in ('Pultos', 'Raktáros')`
NOT IN	`select * from konyvtaros where ks_beosztas not in ('Vezető', 'Pultos')`	`select * from konyvtaros where ks_beosztas not in ('Vezető', 'Pultos')`
BETWEEN	`select * from kiadvany where kv_ev between 2001 and 2005`	`select * from kiadvany where kv_ev between 2001 and 2005`
NOT BETWEEN	`select * from kiadvany where kv_ev not between 2001 and 2005`	`select * from kiadvany where kv_ev not between 2001 and 2005`
LIKE Case Sensitive!	`select * from kiadvany where kv_cim not ilike '%atlasz%'`	`select * from kiadvany where kv_cim like '%atlasz%'` **Case Sensitive,lásd fentebb a magyarázatot**
NOT LIKE Case Sensitive!	`select * from kiadvany where kv_cim not ilike '%atlasz%'`	`select * from kiadvany where kv_cim not like '%atlasz%'`

		Case Sensitive,lásd fentebb a magyarázatot
ILIKE **NEM Case Sensitive!**	-	`select * from kiadvany where kv_cim not ilike '%atlasz%'` **NEM Case Sensitive!**
NOT ILIKE **NEM Case Sensitive!**	-	`select * from kiadvany where kv_cim not ilike '%atlasz%'` **NEM Case Sensitive!**
IS NULL	`select * from kiadvany where kv_isbn is null`	`select * from kiadvany where kv_isbn is null`
IS NOT NULL	`select * from kiadvany where kv_isbn is not null`	`select * from kiadvany where kv_isbn is not null`
AND	`select * from kiadvany where kv_isbn is not null and kv_kiado='KISKAPU'`	`select * from kiadvany where kv_isbn is not null and kv_kiado='KISKAPU'`
OR	`select * from kiadvany where kv_isbn is not null OR kv_kiado='KISKAPU'`	`select * from kiadvany where kv_isbn is not null OR kv_kiado='KISKAPU'`
DISTINCT	`select distinct ks_beosztas from konyvtaros`	`select distinct ks_beosztas from konyvtaros`
UPPER()	`select upper(ks_nev) from konyvtaros`	`select upper(ks_nev) from konyvtaros`
LOWER()	`select lower(kv_kiado) from kiadvany`	`select lower(kv_kiado) from kiadvany`
CONCAT()	`select concat('ISBN szám: ', kv_isbn) from kiadvany`	`select concat('ISBN szám: ', kv_isbn) from kiadvany`
COALESCE()	`select coalesce(kv_isbn,'****') from kiadvany`	`select coalesce(kv_isbn,'****') from kiadvany`
SUBSTRING()	`select substring(ks_nev, 2,3) from konyvtaros`	`select substring(ks_nev, 2,3) from konyvtaros`
SUBSTRING()	`select substring(ks_nev, 2) from konyvtaros`	`select substring(ks_nev, 2) from konyvtaros`
SUBSTRING()	`select substring(ks_nev, -2,1) from konyvtaros`	- **RIGHT-tal**
SUBSTRING()	`select substring(ks_nev, -2) from konyvtaros`	- **RIGHT-tal**
LEFT()	`select left(ks_nev, 3) from konyvtaros`	`select left(ks_nev, 3) from konyvtaros`
RIGHT()	`select right(ks_nev, 3) from konyvtaros`	`select right(ks_nev, 3) from konyvtaros`
INSTR() **STRPOS()**	`select instr(ks_nev,' ') from konyvtaros`	`select strpos(ks_nev,' ') from konyvtaros`
LENGTH()	`select ks_nev, length(ks_nev) from konyvtaros` /bináris hossz: ékezetes betű kétszeres hossznak számít/	`select ks_nev, length(ks_nev) from konyvtaros` **A karakterek száma** **A MySQL-beli length-szel az octet_length(string) egyezik meg.**
CHAR_LENGTH()	`select ks_nev, char_length(ks_nev) from konyvtaros` **A karakterek száma**	`select ks_nev, char_length(ks_nev) from konyvtaros` **A karakterek száma**
DIV	`select ks_fizetes, ks_fizetes div 5000 from konyvtaros`	`select ks_fizetes, div(ks_fizetes,5000) from konyvtaros`
MOD(), %	`select ks_fizetes, mod (ks_fizetes,5000) from konyvtaros` vagy	`select ks_fizetes, mod (ks_fizetes,5000) from konyvtaros` vagy

	select ks_fizetes, ks_fizetes mod 5000 from konyvtaros vagy select ks_fizetes, ks_fizetes %5000 from konyvtaros	select ks_fizetes, ks_fizetes%5000 from konyvtaros
ABS()	select abs(-2.1)	select abs(-2.1)
SIGN()	select sign(-2.1)	select sign(-2.1)
POWER() vagy POW()	select POWER(2,4)	select POWER(2,4)
ROUND()	select round(2.1234,0) select round(200.1234,-2) select round(200.1234,2)	select round(2.1234) select round(2.1234,0) select round(200.1234,-2) select round(200.1234,2)
TRUNCATE()	select truncate(200.1234,0) select truncate(200.1234, 2) select truncate(200.1234,-2)	select trunc(200.1234) select trunc(200.1234,0) select trunc(200.1234,2) select trunc(200.1234,-2)
CEIL()	select ceil(-2.1234)	select ceil(-2.1234)
FLOOR()	select floorl(-2.1234)	select floor(-2.1234)
SQRT()	select sqrt(200)	select sqrt(200)
DATEDIFF()	select * from kolcsonzes where datediff(kz_kidatum,kz_bedatum)>7	- helyette: select kz_bedatum-kz_kidatum from kolcsonzes
NOW()	select now()	select now() timezone-nal
YEAR()	select year(now())	select extract(year from now())
MONTH()	select month(now())	select extract(month from now())
DAY()	select day(now())	select extract(day from now())
HOUR()	select hour(now())	select extract(hour from now())
MINUTE()	select minute(now())	select extract(minute from now())
SECOND()	select second(now())	select extract(second from now())
DAYOFWEEK()	select dayofweek(now())	select extract(dow from now())
timezone		select extract(timezone_hour from now())
MIN()	select min(ks_fizetes) from konyvtaros	select min(ks_fizetes) from konyvtaros
MAX()	select max(ks_fizetes) from konyvtaros	select max(ks_fizetes) from konyvtaros
AVG()	select avg(ks_fizetes) from konyvtaros	select avg(ks_fizetes) from konyvtaros
SUM()	select sum(ks_fizetes) from konyvtaros	select sum(ks_fizetes) from konyvtaros
COUNT()	select count(*) from konyvtaros Olyan mező számosságát vizsgáljuk ált. ahol nincs NULL!	select count(*) from konyvtaros Olyan mező számosságát vizsgáljuk ált. ahol nincs NULL!
Group by	select ks_beosztas,count(*) from konyvtaros group by ks_beosztas VAGY select ks_beosztas,count(*) from konyvtaros group by 1	select ks_beosztas,count(*) from konyvtaros group by ks_beosztas VAGY select ks_beosztas,count(*) from konyvtaros group by 1
HAVING	select ks_beosztas,count(*) from konyvtaros group by ks_beosztas having count(*)>6	select ks_beosztas,count(*) from konyvtaros group by ks_beosztas having count(*)>6

TÁBLAKAPCSOLÁS (INNER JOIN) A és B tábla metszete	`select ts_nev, ks_nev from konyvtaros, tanszek where ks_tskod=ts_kod VAGY select ts_nev, ks_nev from konyvtaros join tanszek ON ks_tskod=ts_kod VAGY select ts_nev, ks_nev from konyvtaros inner join tanszek ON ks_tskod=ts_kod`	`select ts_nev, ks_nev from konyvtaros, tanszek where ks_tskod=ts_kod VAGY select ts_nev, ks_nev from konyvtaros join tanszek ON ks_tskod=ts_kod VAGY select ts_nev, ks_nev from konyvtaros inner join tanszek ON ks_tskod=ts_kod`
LEFT VAGY RIGHT JOIN	`select ts_nev, ks_nev from konyvtaros right join tanszek ON ks_tskod=ts_kod`	`select ts_nev, ks_nev from konyvtaros right join tanszek ON ks_tskod=ts_kod`
Egymásba ágyazott lekérdezés (egy érték relációval)	`select ks_nev, ks_fizetes from konyvtaros where ks_fizetes>(select ks_fizetes from konyvtaros where ks_nev='Balázs Mária')`	`select ks_nev, ks_fizetes from konyvtaros where ks_fizetes>(select ks_fizetes from konyvtaros where ks_nev='Balázs Mária')`
Egymásba ágyazott lekérdezés (IN több értékkel)	`SELECT * FROM tanszek WHERE ts_telepules IN (select ts_telepules FROM tanszek WHERE ts_nev IN ('Geofizikai Tanszék','Geodéziai Tanszék'));`	`SELECT * FROM tanszek WHERE ts_telepules IN (select ts_telepules FROM tanszek WHERE ts_nev IN ('Geofizikai Tanszék','Geodéziai Tanszék'));`
EXISTS, NOT EXISTS	`SELECT * FROM tanszek where NOT EXISTS (SELECT 1 FROM konyvtaros WHERE ks_tskod=ts_kod);`	`SELECT * FROM tanszek where NOT EXISTS (SELECT 1 FROM konyvtaros WHERE ks_tskod=ts_kod);`
LIMIT első x találat mutatása	`SELECT * FROM konyvtaros limit 5`	`SELECT * FROM konyvtaros limit 5`
LIMIT X, Y LIMIT X OFFSET Y	`SELECT * FROM konyvtaros limit 5,3 VAGY SELECT * FROM konyvtaros limit 5 offset 3 Írjon ki 3 találatot, a 6. rekordtól kezdve (offset száma+1)`	`SELECT * FROM konyvtaros limit 3 offset 5 Írjon ki 3 találatot, a 6. rekordtól kezdve (offset száma+1)`

1. FEJEZET: A pgAdmin4, PostgreSQL és POSTGIS telepítése, a pgAdmin 4 rövid bemutatása

Ahhoz, hogy PostgreSQL adatbázist tudjunk kezelni, nemcsak egy felületre, hanem magára az adatbáziskezelőre is szükség van. Tehát önmagában, Postgres szerver nélkül feltenni egy pgAdmin-t nem sok értelme van. A telepítéshez a következő hivatalos oldalt ajánlom:

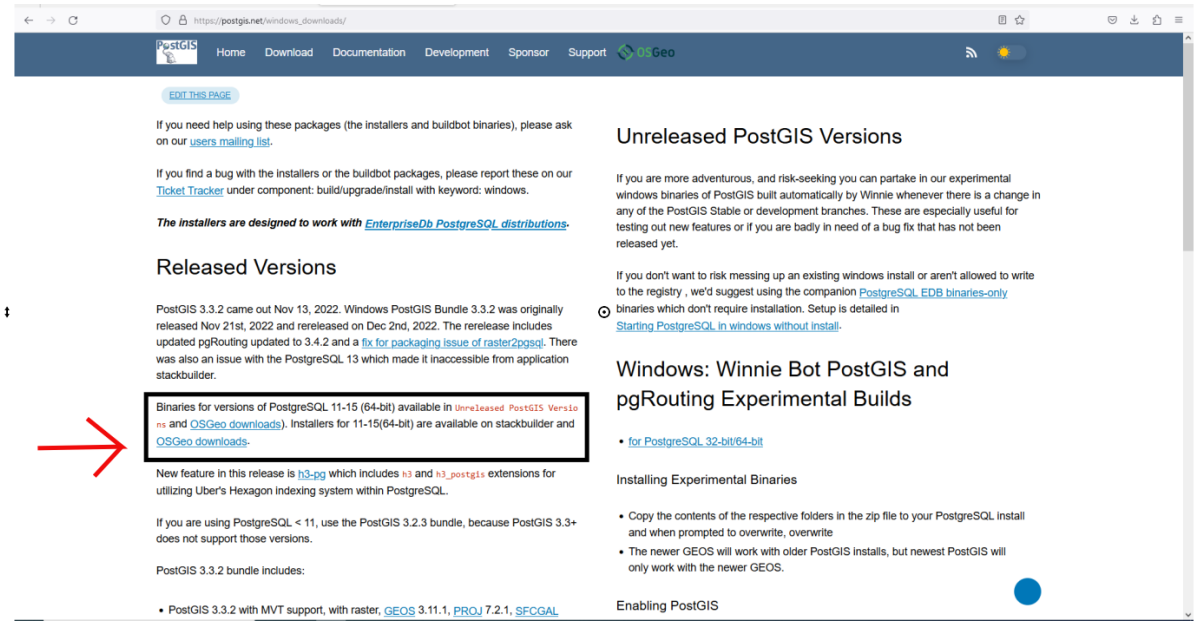
https://www.postgresql.org/download/windows/

A telepítő *next-next* jellegű. A telepítéskor kell beállítani azt a portot, amelyen az adatbázis-kezelő fut. Alapértelmezettként (ha nem foglalt) érdemes a rendszer által felajánlott 5432-portot választani. Emellett ekkor kell megadni egy felhasználónevet és jelszót is, amellyel hozzáférünk a PostgreSQL-hez. Ezt jegyezzük meg.

Ami még települ az adatbázis szerver mellé, az a pgAdmin4. Ez a szoftver PostgreSQL adatbázisok kezelésére lett kitalálva. Igen robusztus, stabil. Azonban szerintem a kezelőfelülete nem túl kényelmes, éppen ezért a következő fejezetben mutatok egy alternatív megoldást, amellyel praktikusabban tudunk SQL adatbázisokat kezelni. Ez az univerzális adatbázis-kezelő lesz a DBeaver. Sajnos, mint kiderült, ennek a szoftvernek van (jelenleg) néhány gyenge pontja, ezért egy-egy példa erejéig, igénybe fogjuk venni a pgAdmin-t is. Akinek ez szimpatikusabb, természetesen az alábbi feladatok nagy része megvalósítható pgAdmin környezetben is.

Azonban térbeli adatokat nem lehet kezelni PostGIS nélkül, telepítsük ezt is!

https://postgis.net/windows_downloads/

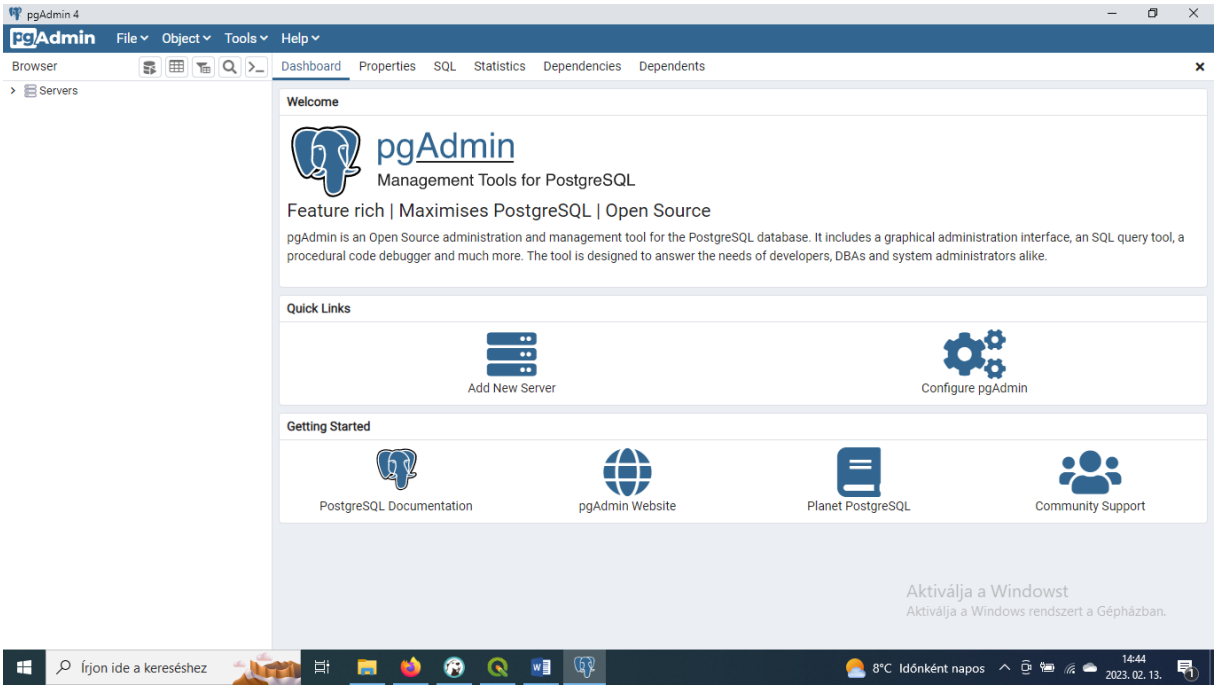


vagy direkt: https://download.osgeo.org/postgis/windows/

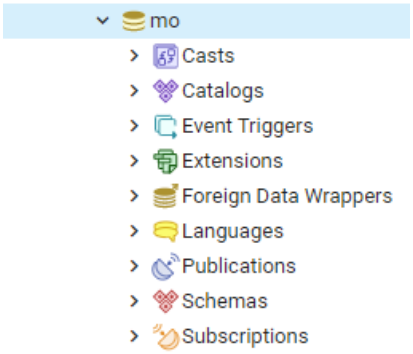
Innen lehet letölteni a mi PostgreSQL verziónknak megfelelő PostGIS-t.

A pgAdmin-ról

A szoftver néhány alapfunkcióját mutatom be. Első lépés a szoftver elindításakor a bejelentkezés, a telepítéskor megadott jelszót kell beírni, majd ez az üdvözlőképernyő fogad minket:

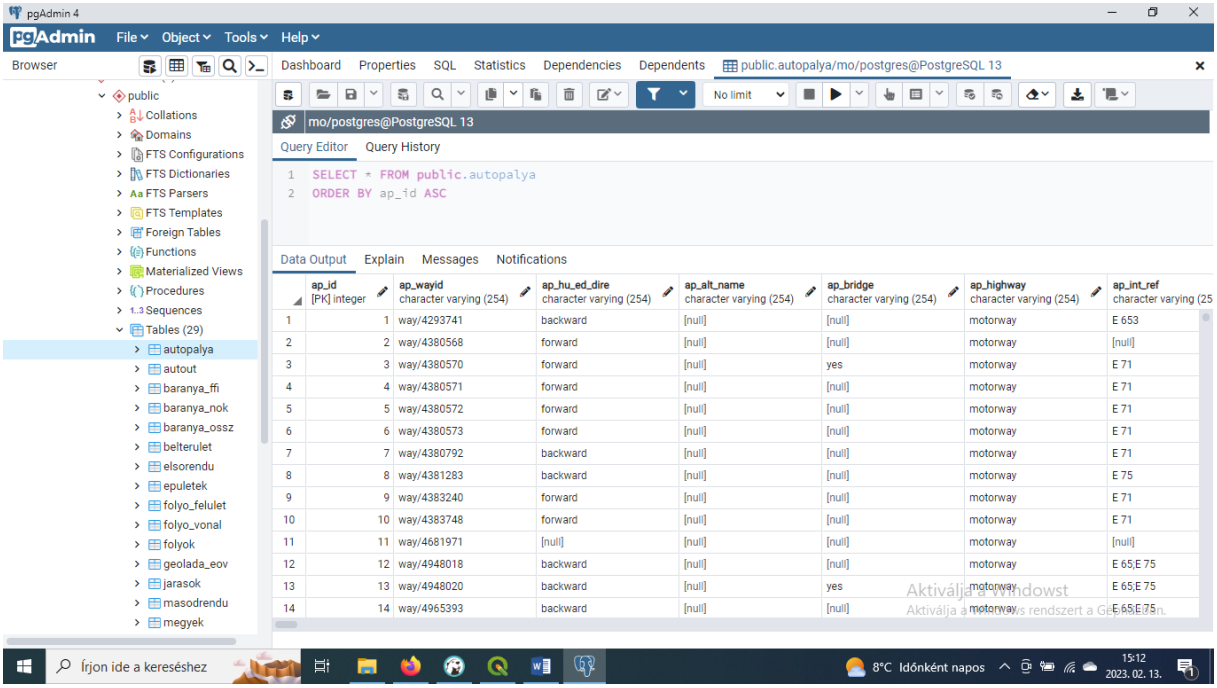


Lenyitva a *Servers* menüpontot, megtalálhatóak az adatbázisok, így pl. a `mo` adatbázis is (feltéve, ha már importáltuk). A *Schemas* itt az utolsó előtti pont, ebben található a `public` séma, azon belül pedig a táblák. Az *Extension*nél lehet hozzáadni a kiegészítőket (Jobb egérgombbal kattintás az *Extension*ön, majd *Create* → Itt a lenyíló listából választható ki pl. a PostGIS).

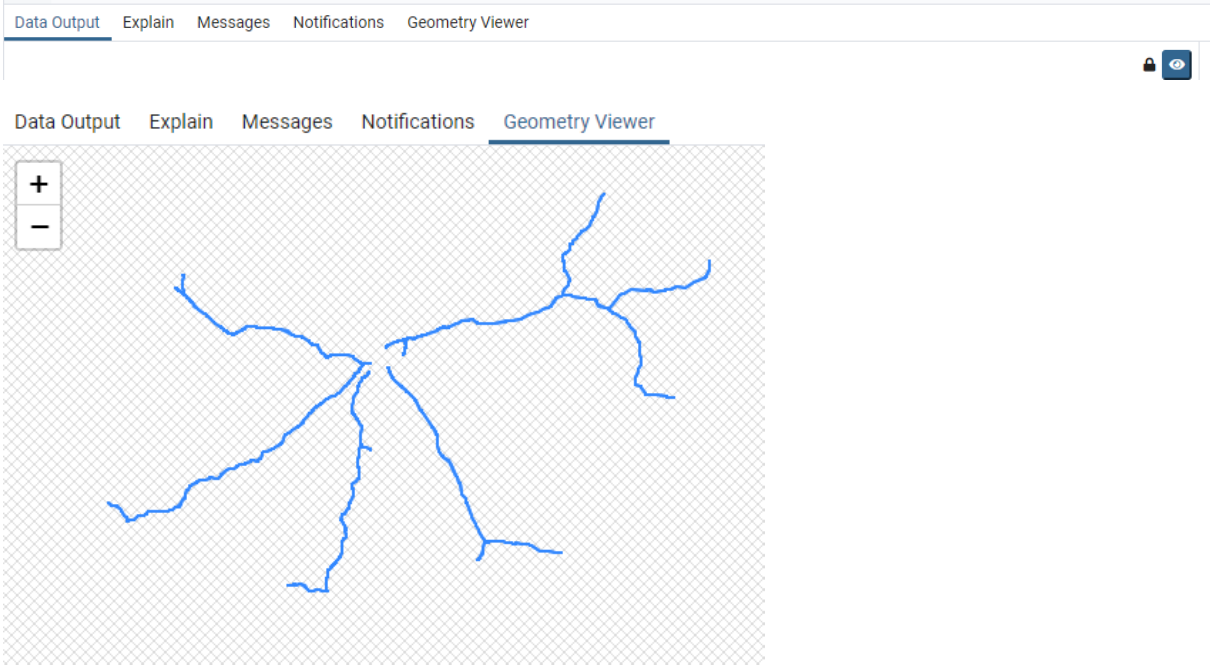


A *Tables* almenün belül a táblát kiválasztva (pl.: `autopalya`), a *Browser* ablakban a táblázat ikonra kattintva nyílik meg a tábla nézete a jobb oldali munkaablakban.

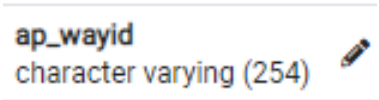
Ilyenkor egy `select *` utasítás fut le a táblán, és minden rekord látszik. Ha vannak térbeli adataink a geometriát tartalmazó mező fejlécében találunk egy szem ikont, vagy a *GeometryViewer* fülre kell kattintani. Ezáltal elérhető egy (web)térképes nézet. Amennyiben az adatok WGS-84 földrajzi koordinátákkal vannak definiálva (EPSG:4326), egy OSM térkép kerül a háttérbe, azonban más rendszereknél (pl. EOV vetület) a háttértérkép nélküli nézet aktiválódik.



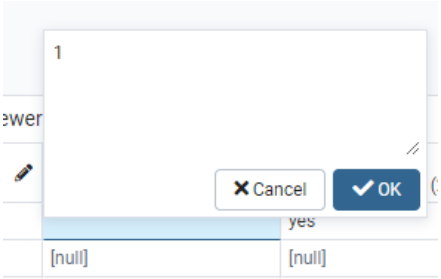
A táblázatos nézetre való visszaváltáshoz a *Data Output*ra kell kattintani.



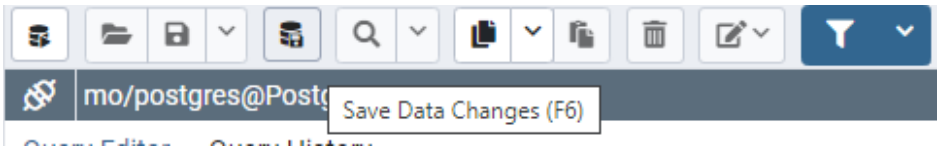
Az adatok szerkesztése a táblázat fejlécében található ceruzára való kattintással lehetséges.



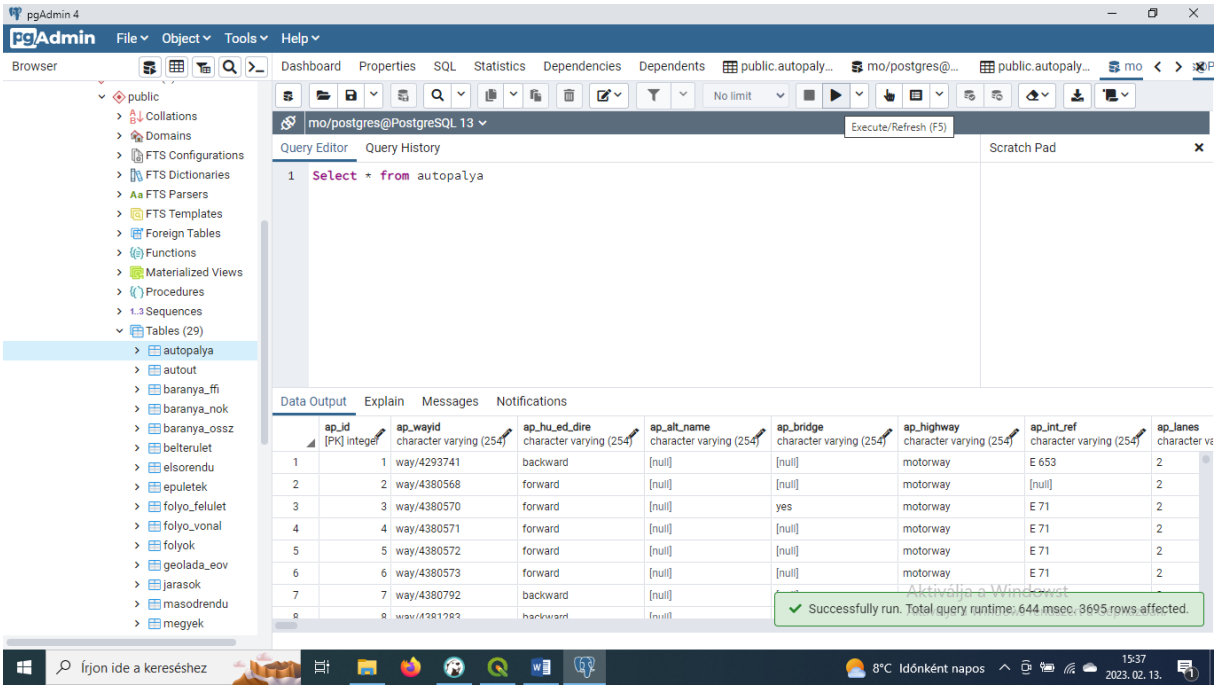
Majd szerkesztjük az adatot.



Mentjük a változásokat. A lekérdező ablak felett a hordó jel lakattal aktívvá fog válni, ha változtattunk a táblán. Ez a *Save Data Changes*.

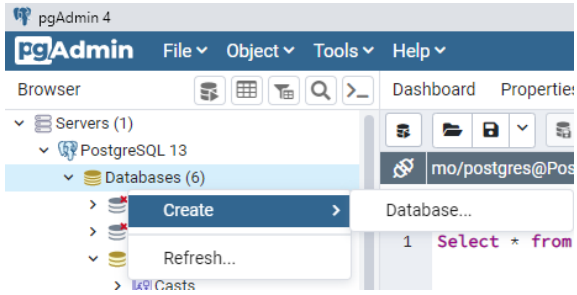


Ha lekérdezéseket szeretnénk végrehajtani a kiválasztott adatbázison a *Query Tool*t kell aktiválni (hordó jel lejátszás gombbal). Beírjuk a lekérdezést, majd az F5 billentyűvel, vagy a fekete lejátszás gombbal lefuttatjuk.



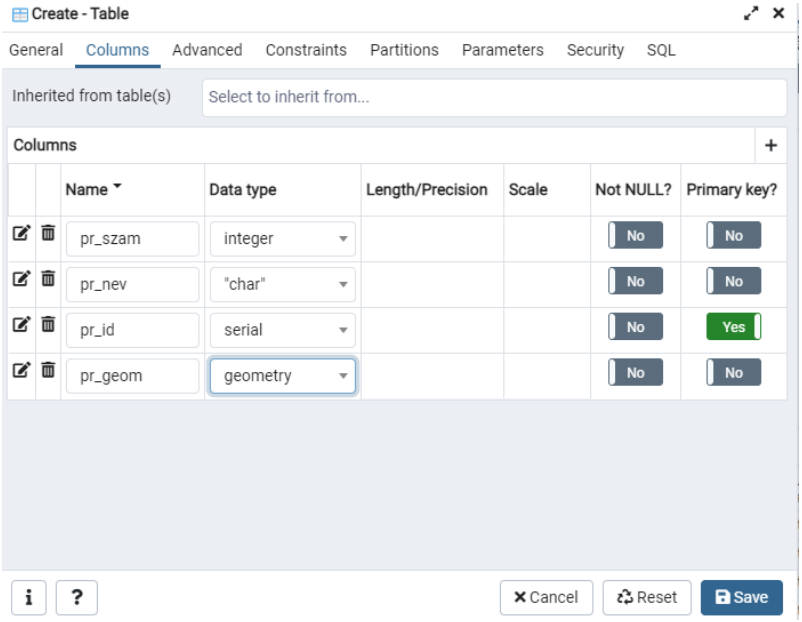
Új adatbázis hozzáadása: egy kattintás a jobb egérgombbal a *Database*-en → *Create* → *Database*

A párbeszédablakban minimum az adatbázis nevét, a karakterkódolást, és a felhasználó tulajdonos (pl. postgres) nevét kell megadni, de egyéb beállításokra is van lehetőség.



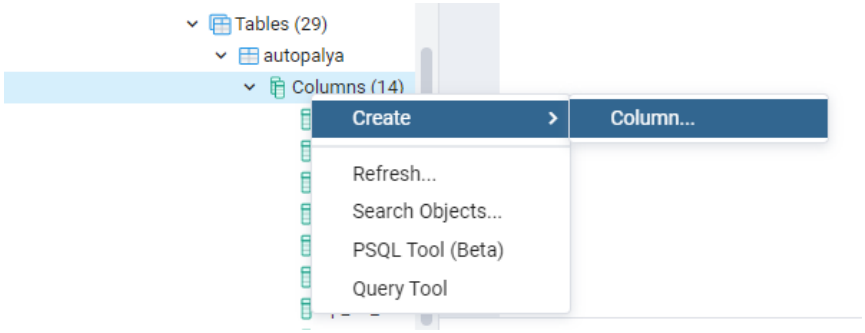
Új tábla létrehozása: egy kattintás a jobb egérgombbal a *Table*-ön → *Create* → *Table*

A párbeszédpanelen először a tábla nevét adjuk meg, majd a második fülön a mezők is megadhatók (+), névvel, adattípussal, a kulcs mező és `Null` érték definiálásával. Az SQL fülön olvasható a lekérdezés.



Ha pedig utólag szeretnénk mezőt hozzáadni egy létező táblához, akkor a *Tables* menü *Columns* alpontjában egy kattintás a jobb egérgombbal →*Create* → *Column.*

A két legfontosabb beállítás a mező neve és adattípusa (*General* és *Definition* fülek a megnyíló párbeszédablakban).



Az előbb felsorolt elemeket (adatbázis, tábla, mező) lehet törölni is: ehhez egy kattintás a jobb egérgombbal a törölni kívánt elemen és *Delete/Drop*.

Az előbb felsorolt elemek nevét/adattípusát (adatbázis, tábla, mező) lehet módosítani is: ehhez egy kattintás a jobb egérgombbal a módosítani kívánt elemen → *Properties*. A dialógusablakban módosítható a kívánt érték.

A *Refresh* gombbal frissíthető a nézet.

Meglévő SQL szkriptek importálását lásd a 7. fejezetben.

2. FEJEZET: A DBeaver univerzális adatbáziskezelő-szoftver

A DBeaver Community egy ingyenes adatbázis-kezelő felület, amely nemcsak a PostgreSQL, hanem más SQL és NoSQL adatbázisok kezelésére is alkalmas. Vannak kereskedelmi verziói is, amelyek még több funkcionalitást ígérnek.

Előnyei:

- kényelmes kezelőfelület
- támogatja a felhő alapú megoldásokat is
- kompatibilis Excel táblákkal, adat export és import akár CSV-ből
- multiplatform támogatás (Windows, Linux, MacOSX)

Mi a PostgreSQL+ PostGIS adatbázisok kezeléséhez fogjuk használni.

Letölthető innen: https://dbeaver.io/download/

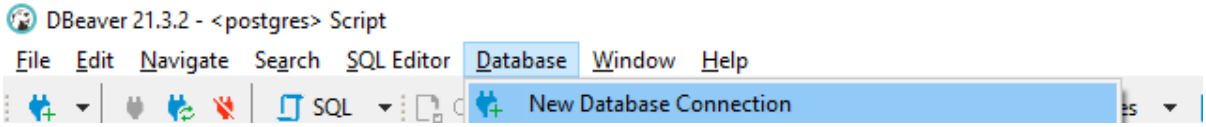
Windows install:

Next-next jellegű. Ahhoz, hogy a PostgreSQL adatbázissal együtt tudjuk használni szükséges, hogy egy PostgreSQL adatbáziskezelő szerver legyen telepítve.

DBeaver – New connection: Új adatbázis kapcsolat

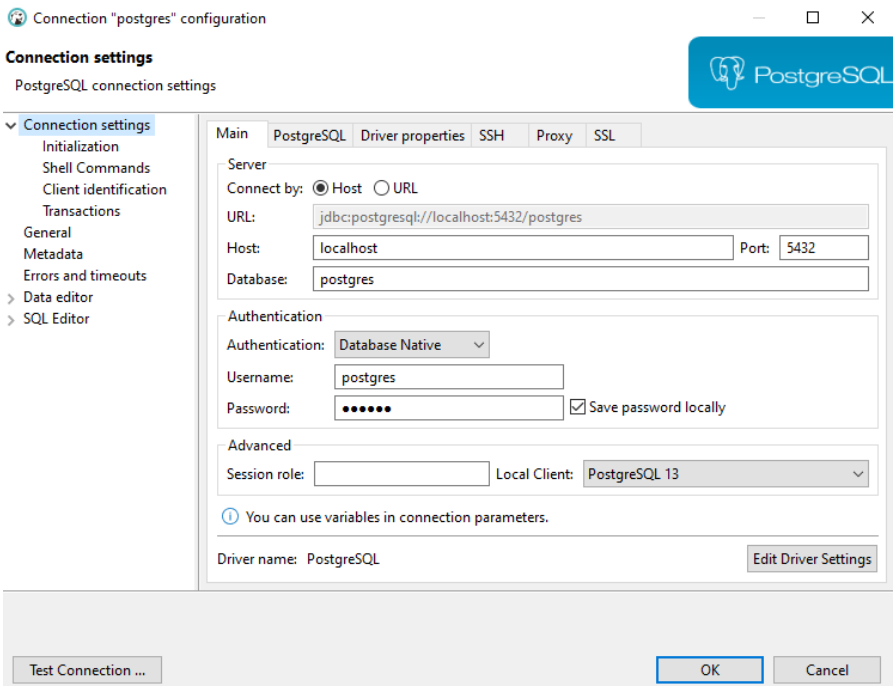
Ha első alkalommal indítjuk el a szoftvert, egy kapcsolat-varázslóval fog találkozni. Egyébként, ha új kapcsolatot szeretne, akkor is ezt a menüt kell előhívni.

Ez a *Database* → *New Database Connection* menüpontban található.



Válasszuk ki a PostgreSQL-t. → *Next* → Itt meg kell adni a szerver nevét, portját, az adatbázis felhasználónevét és jelszavát. Itt a PostgreSQL adatbázis szerverrel megegyező felhasználónevet, jelszót és portot kell megadni. Ha máshogy nem döntünk és még nem foglalt, a PostgreSQL alapértelmezettként az 5432-es portot ajánlja fel.

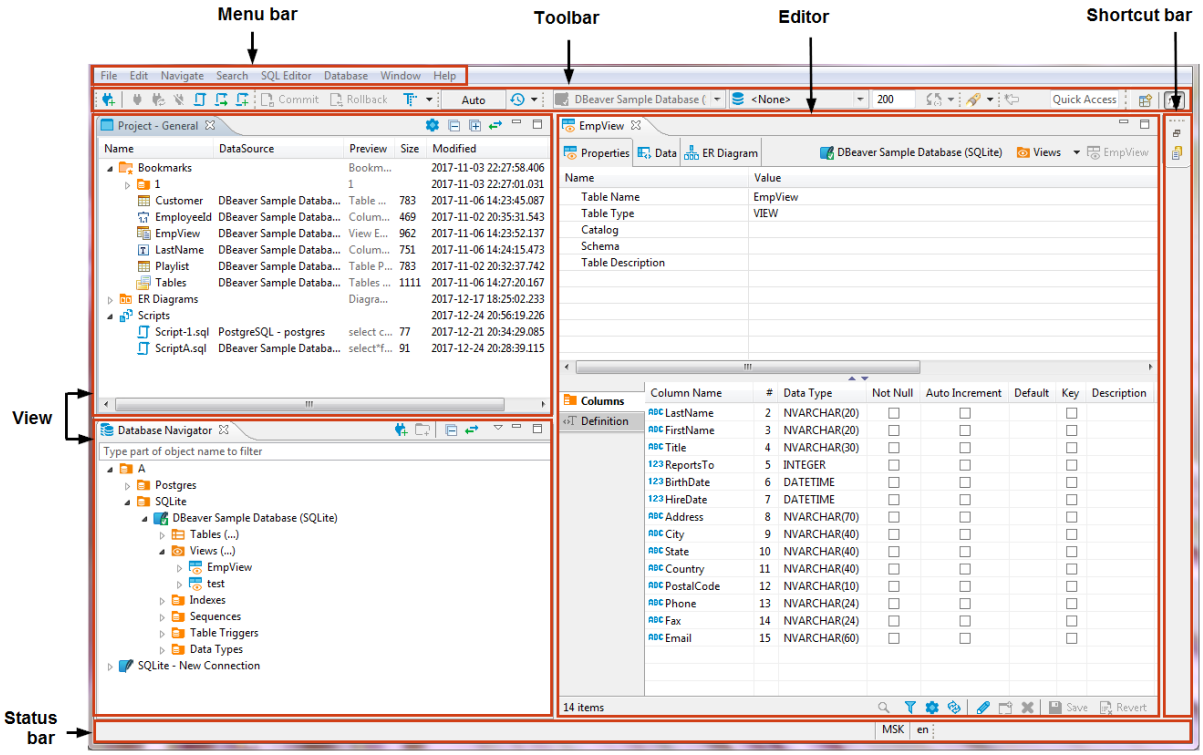
Megjegyezném, amikor először létesítünk kapcsolatot egy PostgreSQL szerverrel, a DBeaver le fog tölteni néhány komponenst, kiegészítőt, amely az adatbázis-kapcsolat felépítését szolgálja.



A fenti ábrán a PostgreSQL fülön van egy *Show all Databases* opció. Érdemes kipipálni, hogy mindig mutassa az összes adatbázist.

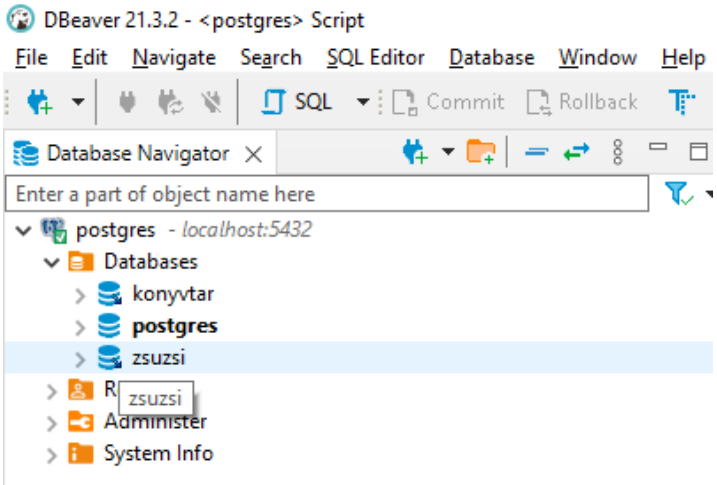
A DBeaver User Interface

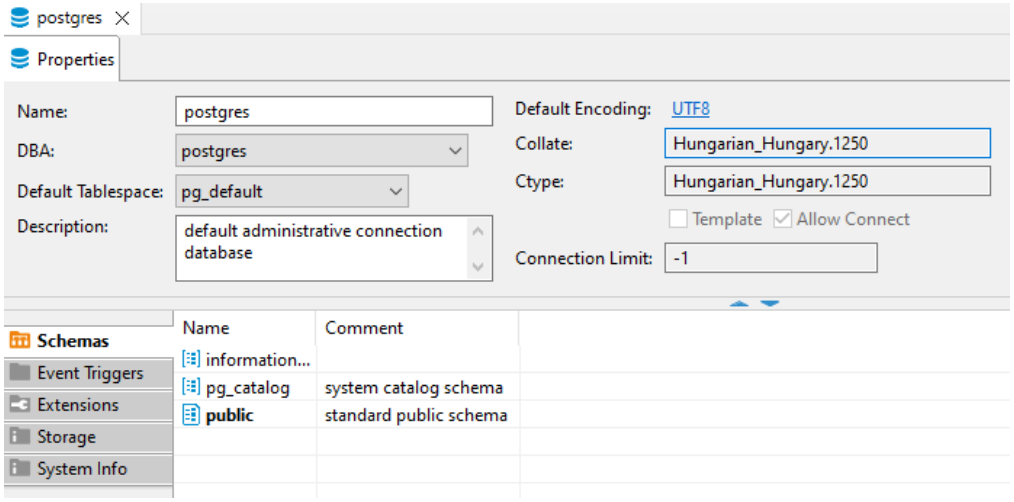
A DBeaver kezelőfelülete könnyen alakítható, felhasználóbarátabb a pgAdmin robosztusabb felületével szemben, jól kiemeli a felhasználónak lényeges menüpontokat. Baloldalon a *Database Navigator* menüben láthatjuk az egyes adatbázisokat (amihez kapcsolódtunk), jobb oldalon a munkaablakban pedig az egyes adatbázisok tábláit, azok szerkezetét és adatait hívhatjuk be, valamint egy újabb fülön futtathatjuk a lekérdezéseinket.



Az adatbázisok

A Database Navigatorban dupla kattintással nyithatjuk meg az adatbázis adatlapját. Itt lehet definiálni az adatbázis nevét és a karakterkódolást. A default UTF-8.

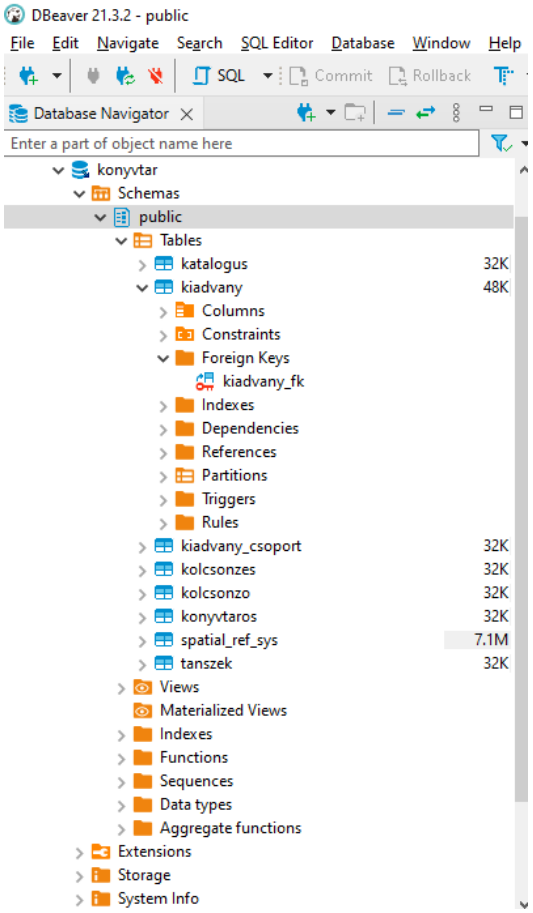


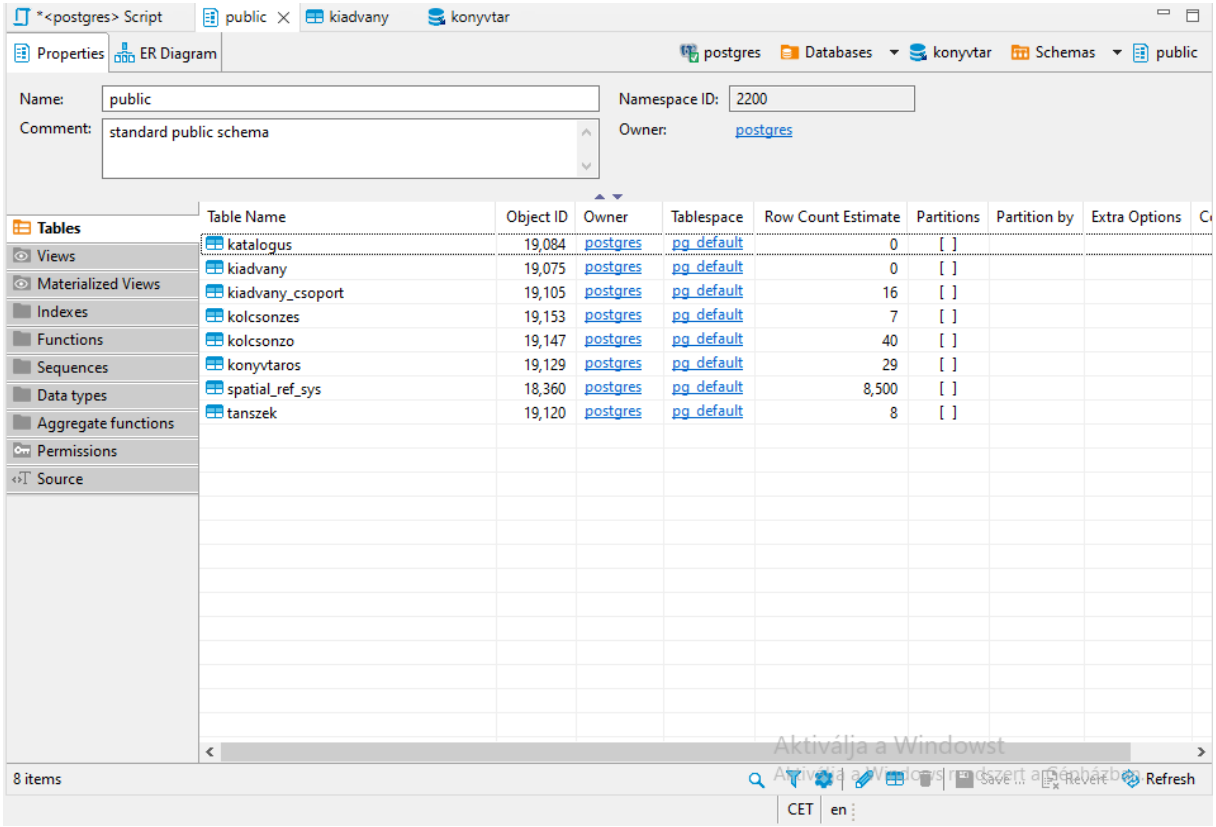


Ha az adatbázist kinyitjuk, számunkra talán a legfontosabbak a sémák és a kiegészítők (kiterjesztések).

- *Schema*: *public* → alapértelmezett adatbázisséma, az adatbázison belül egymástól lazán elkülönülő egységek a sémák.
- *Extension*: itt látható/adható hozzá, hogy milyen extrák/kiegészítők vannak telepítve az adott adatbázishoz. Egyik ilyen a PostGIS.
  Jobb kattintás, vagy alul kis ikon: *Create New Extension*. A listából kikereshető a PostGIS (feltéve, hogy telepítve van).

A sémát lenyitva láthatók a táblák. Dupla kattintás vagy a jobb egérgomb és *View Table* menü aktiválja a tábla nézetét a jobb oldali munkaablakba.





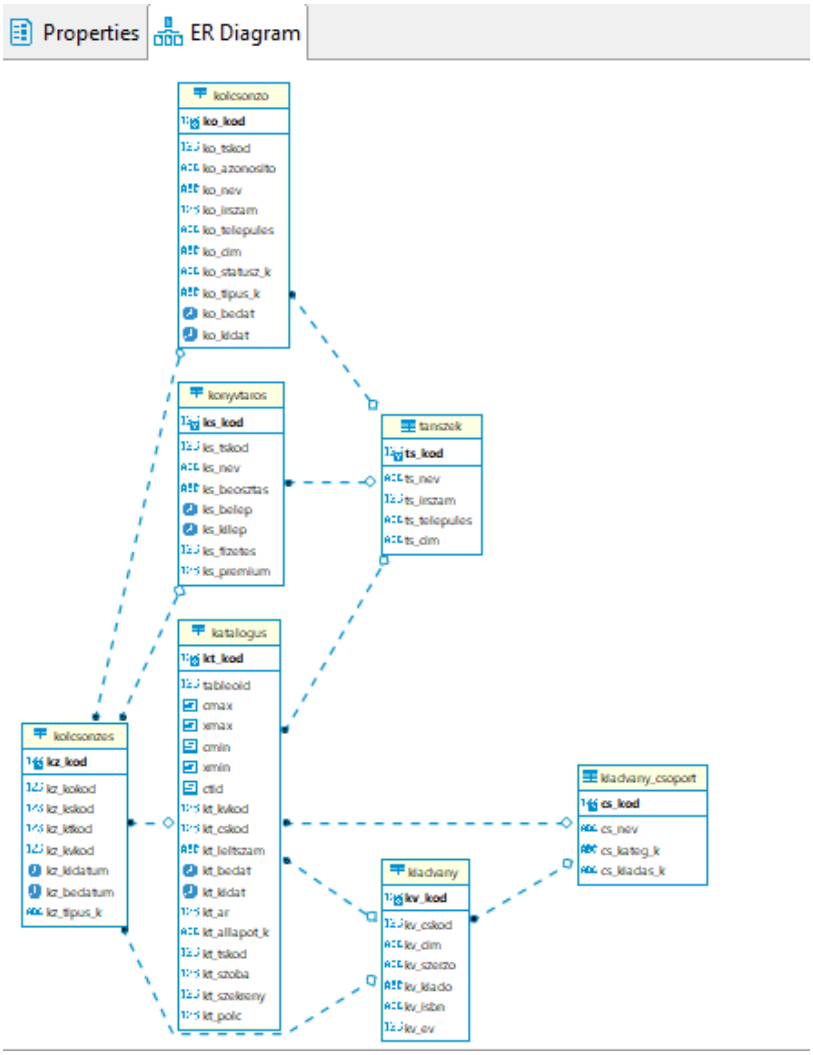
A táblák

A baloldali menürendszerben is láthatóak egy tábla beállítási lehetőségei, számunkra fontosabb a jobb oldali munkaablak. Egy tábla munkaablakja három fülből áll:

- *Properties*: a tábla beállításai, mezői stb.
- *Data*: az adatok nézete
- *ER Diagram*: az adott tábla kapcsolatai diagramon ábrázolva (elsődleges és idegen kulcsok)

ER Diagram: Ha az adatbázisban be van állítva az összes elsődleges és idegen kulcs, akkor a szoftver képes ezeket a függőségeket egy diagramon bemutatni. Itt például a konyvtar adatbázis ER diagramja látható.

-



Nézzünk egy példát a könyvtár adatbázis katalógus táblájára (Csak a legfontosabb lehetőségeket sorolom fel)!

Columns*:* a tábla mezői, adattípussal, karakterkódolással, `Null` vagy `Not Null` értékkel.

Új mező létrehozásához kattintani kell a jobb egérgombbal → *Create new column* (vagy alul keresse meg a fehér/kék négyzetes ikont).

Ugyanitt az alsó ikonsoron van a mező törlésére illetve a módosítására szolgáló ikon. Attól, hogy bármit változtattunk, még nem hajtódott végre a változás. A változás végrehajtását a *SAVE* ikonnal menthetjük. Ekkor láthatóvá válik a módosítást megvalósító (automatikusan létrehozott) SQL parancs is.

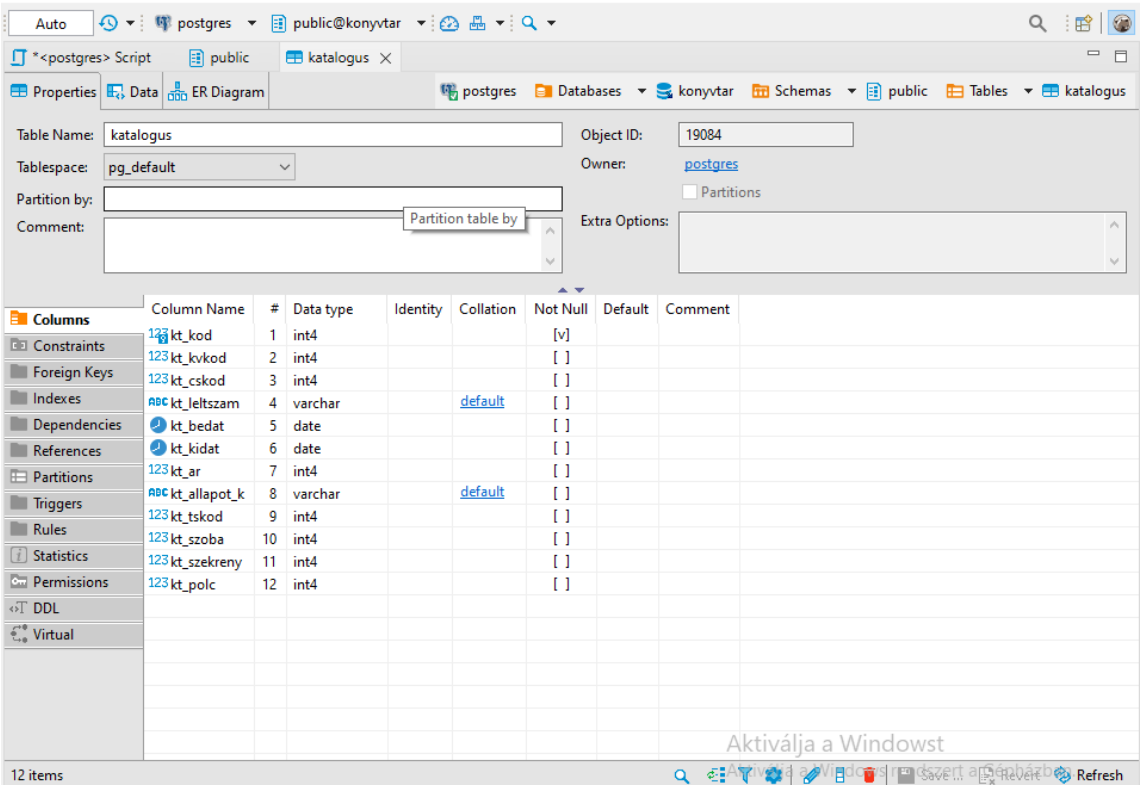
Constraints: Primary key (elsődleges kulcs) vagy unique key beállítása.

Foreign keys: idegen kulcsok definiálása és a velük együtt végrehajtandó műveletek pl. törlési és frissítési műveletek: (delete és update rules: cascade és restrict)

Cascade delete and update /Rekurzív törlés és frissítés/: ha törlünk vagy frissítünk egy elsődleges kulcsot, akkor az idegen kulccsal kapcsolódó tábla is törlődik vagy frissül.

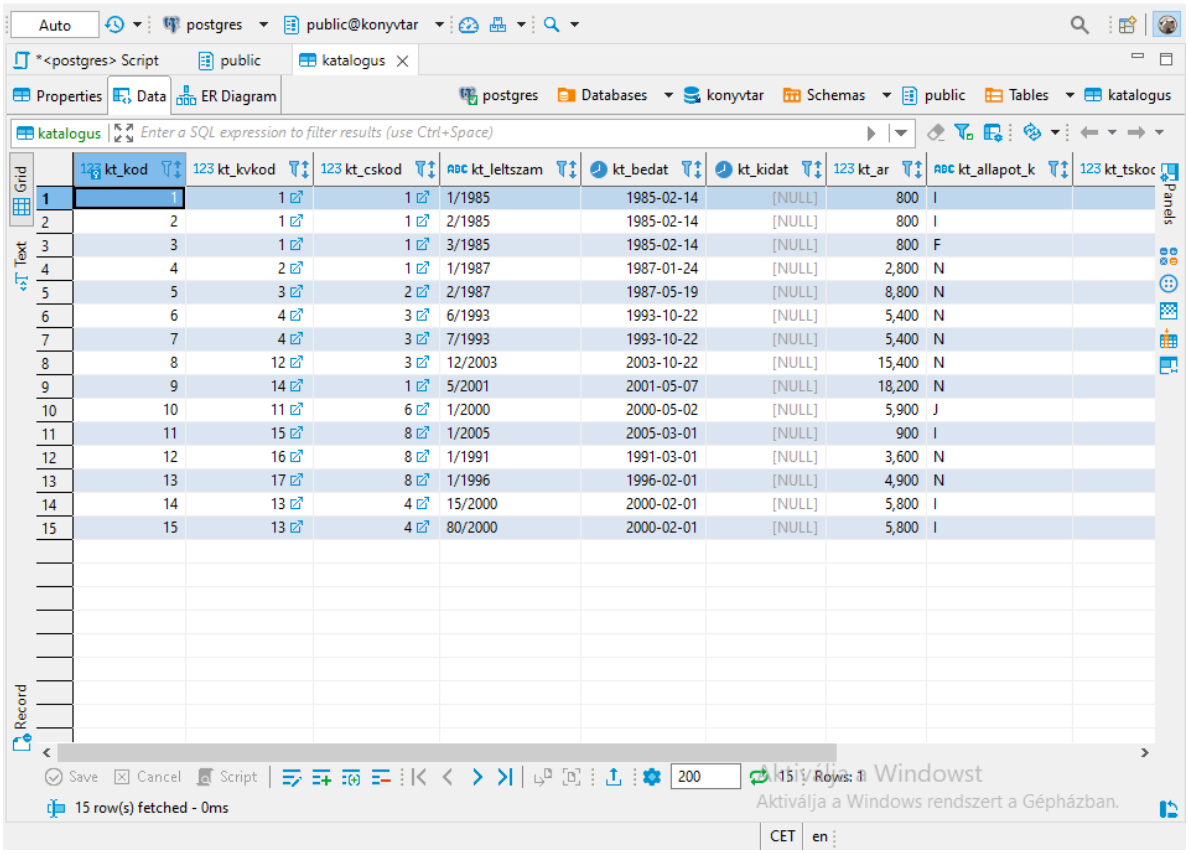
Restricted delete and update /Korlátozott törlés és frissítés/: ha módosítanánk vagy törölnénk az adott rekordot a szülő táblában, akkor letiltja a módosítás vagy törlés műveletét az SQL, hogy a gyermektáblában ne maradjon az idegen kulcs referencia nélkül.

***Indexes*:** a keresés gyorsítására index definiálható.



Egy tábla DATA füle

A táblázat mezői egyszerűen szerkeszthetők ezen a felületen. Az előbbi megállapításom itt is igaz: attól, hogy bármit változtattunk, még nem hajtódott végre a változás. A változás végrehajtását a *SAVE* ikonnal menthetjük. Ekkor láthatóvá válik az elküldendő lekérdezés.



Data filtering

Itt hajthatjuk végre az adott táblán a megjelenő adatok szűrését. A fenti képen a *Properties* fül alatt látszik a tábla neve (ez jelenleg katalogus), ebbe a sorba írhatjuk a feltételeket. Persze ilyenkor nem kell teljes lekérdezést írni, csak a `where` utáni feltételt.

```
/select * from katalogus where/ FELTÉTEL.
```

Az egyes mezők fejlécén lévő nyilakra kattintva az adott mező szerinti növekvő, illetve csökkenő sorrend kapható. A tölcsér pedig az adatok szűrését engedi az adott mezőben (a lekérdezés a fenti sorba kerül).

Lekérdezés egy adatbázison

Az általános SQL lekérdező ablak több helyről is megnyitható. Jobb egérgomb kattintás az adatbázison → *SQL Editor* → *Open SQL Console*.

VAGY ikonsor → *SQL* → *Open SQL Console* (ekkor fontos, hogy a választott adatbázis megfelelő sémáján (pl. `public`) álljon a kijelölés, mert ezen fog futni a lekérdezés).

Lekérdezés futtatása a narancsszínű lejátszás gombbal lehetséges. 

Pl. Írassuk ki az a budapesti tanszéket, ahol a legnagyobb az átlagfizetés. Csak azok a tanszékek szerepeljenek, ahol legalább két alkalmazott van!

```
select ts_nev, avg(ks_fizetes), count(*) from konyvtaros join tanszek on ks_tskod=ts_kod where ts_telepules='Budapest' group by ts_nev having count(*)>2 order by 1 desc limit 1
```

3. FEJEZET: A POSTGIS elmélete

Ez a fejezet a hivatalos angol nyelvű POSTGIS dokumentáció alapján íródott. Igyekeztem kiemelni azokat a függvényeket, amelyek térinformatikai szempontból fontosak számunkra, és azok egyes eseteit példákkal megmagyarázni. Az elméleti áttekintés után a jegyzet következő fejezetében lesz lehetőség gyakorolni az egyes függvényeket.

A hivatalos és aktuális POSTGIS verzió elérhető itt: https://postgis.net/docs/manual-3.3/

Ismerkedés a POSTGIS-szel

A POSTGIS a PostgreSQL térbeli adatok kezelésére kitalált bővítménye.

Írjunk ki egy pontot.

```
SELECT ST_Point(1, 2) AS MyFirstPoint;
```

Írjuk ki egy valós pont földrajzi koordinátáit, adjunk hozzá koordináta rendszert.

```
SELECT ST_SetSRID(ST_Point(-77.036548, 38.895108),4326);
```

Írjunk ki egy pontot földrajzi koordináta rendszerben. Használjuk az `ST_GeomFromText()` függvényt.

```
SELECT ST_GeomFromText('POINT(-77.036548 38.895108)', 4326);
```

Mi az a WKT formátum?

A Well-known text format egy OGC Standard a geometria reprezentációjára. Szöveges adattípus, amely az ember által is könnyen érthető. Konverzióhoz az `ST_GeomFromText()` és az `ST_AsText()` függvényeket használjuk.

A koordinátákat (hosszúság szélesség) sorrendben adjuk meg mindig.

Az egyes elemek geometriai reprezentációját WKT-ben lásd a következő oldalon.

Mi az a WKB formátum?

A Well-known binary format, egy OGC Standard a geometria reprezentációjára. Bináris adattípus (hexadecimális string).

A bináris adatok igen nehezen értelmezhetők az embereknek, bár az adatttárolás szempontjából praktikusabban. A következő példákban a bináris adattípusra láthatunk egy-egy példát.

```
SELECT ST_AsEWKT('0101000020E6100000FD2E6CCD564253C0A93121E692724340');
```

Az `ST_AsEWKT()` függvény nemcsak a bináris geometriát adja vissza WKT-ként, hanem az SRID-t is.

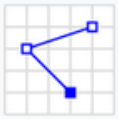
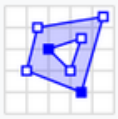
```
SELECT ST_AsText('0101000020E6100000FD2E6CCD564253C0A93121E692724340');
```

Az `ST_AsText()` függvény csak a bináris geometriát adja vissza WKT-ként.

Pontok, vonalláncok és poligonok reprezentációja WKT-ben

Az alábbi két táblázat bemutatja, hogyan lehet megadni a pontokat, vonalláncokat és felületeket, valamint ezek multi (csoport) változatait. A poligonnál az első zárójel a poligon körvonalát, a második ill. további zárójelek a gyűrűket tartalmazzák.

Geometry primitives (2D)

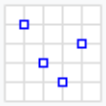
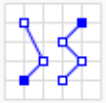
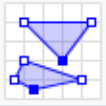
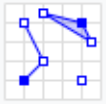
Type	Examples	
Point		`POINT (30 10)`
LineString		`LINESTRING (30 10, 10 30, 40 40)`
Polygon		`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
		`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

A kép forrása: Wikipédia.

A multi geometriák pontok, vonalláncok és poligonok csoportjai. Tipikus eset az Egyesült Királyság minden szigete egy csoportba szervezve. Ezeket reprezentáják a további zárójelek. A geometry collection-ben (geometriai gyűjtemény) az előbbi geometriák bármelyike szerepelhet, vegyes adattípus (ennek ellenére megfontolandó ennek használata, mert egyes térinformatikai szoftverek szétbontják az ilyen jelleg adatokat rétegekre).

A felsoroltakon kívül másfajta geometriákat is támogat a PostGIS. Ezekről később olvashat.

Multipart geometries (2D)

Type	Examples	
MultiPoint		`MULTIPOINT ((10 40), (40 30), (20 20), (30 10))`
		`MULTIPOINT (10 40, 40 30, 20 20, 30 10)`
MultiLineString		`MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))`
MultiPolygon		`MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))`
		`MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))`
GeometryCollection		`GEOMETRYCOLLECTION (POINT (40 10), LINESTRING (10 10, 20 20, 10 40), POLYGON ((40 40, 20 45, 45 30, 40 40)))`

A kép forrása: Wikipédia.

Mi az az EWKT/EWKB?

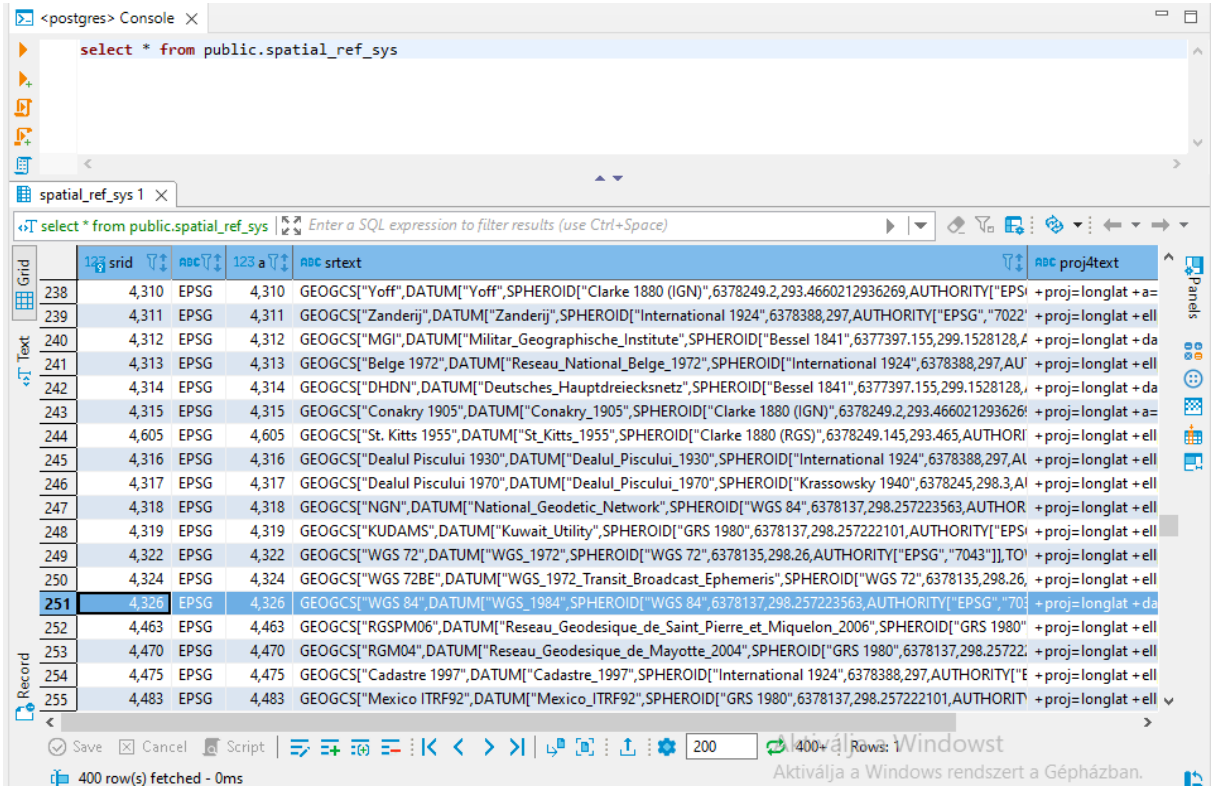
Extended Well-know text or binary format. Egy PostGIS specifikus formátum, amelyben a térbeli referenciarendszert is tárolni tudjuk a következő formában.

```
'SRID=4326;POINT(19.1 47.5)'
```

Mi az a SRID?

Spatial Reference Identifier. A térbeli koordináta-rendszer azonosítója. Ehhez van egy „gyári” (=ha a PostGIS hozzá van adva az adatbázishoz, mint kiegészítő, automatikusan létrejön ez a tábla is) spatial_ref_sys tábla, melyben az összes térbeli azonosító megtaláljuk, amely elérhető a PostGIS-ben.

Az SRID az EPSG rendszerét követi (EPSG: http://epsg.io/). Benne van az EPSG azonosító és a vetület szöveges leírása.



Hány dimenziós lehet a koordináta?

Koordináta dimenzió: hány tengelye van a koordináta-rendszerünknek.

A térképeken és valóságban 2 (X, Y) vagy 3 (X, Y, Z vagy X, Y, M) dimenziós adatokkal dolgozunk, ezt a PostGIS is támogatja.

De a PostGIS-ben lehetőség van egy 4. dimenziót is bevezetni: ez lesz az M koordináta. Az M koordináta (méret/mérték) egy double adattípusként is, minden egyes ponthoz, vagy node-hoz tárolható.

Néhány példa M-re:

- Ha egy pont időbeli mozgását vizsgáljuk, időt tudunk tárolni. Pl. busz útvonal tárolása esetén az M lehet a végállomástól megtett idő a szóban forgó megállóig (pontig).

- Ha egy autópálya melletti található vészhívók koordinátáit vizsgáljuk, ezek sorszámát is tárolhatjuk az M-ben.

Görbék, különleges geometriák és a PostGIS

Görbék: A legnépszerűbb Bézier-görbék vagy egyéb spline-ok tárolására egyelőre nincs lehetőség PostGIS-ben. Ezek tárolását ajánlott LineStringgel vagy Polygonnal lehet megoldani (az íves/görbe részeken sűrűbben vegyük fel a csomópontokat.)

CircularString: legegyszerűbb görbe (**kör vagy körív**), létrehozásához, legalább 3 pont kell. Ebből az első és az utolsó a végpont, a köztesek a görbén helyezkednek el. Ha zárt kört szeretnénk, akkor a kezdő és a végpont megegyezik, a köztes pont a kör átmérőjének ellentétes oldalán helyezkedik el.

CompoundCurve: körívevek és egyenes vonalszakaszok kombinációja. Fel kell benne tüntetni, hogy melyik szakasz lesz benne a körív.

```
COMPOUNDCURVE(CIRCULARSTRING(0 0, 1 1, 1 0),(1 0, 0 1))
```

CurvePolygon: A sima poligonhoz hasonló, de a körvonala és a lyuk körvonala lehet CircularString, CompoundCurve és sima LineString.

MultiCurve: Egy olyan görbe, amelyben lehet CircularString, CompoundCurve és LineString.

MultiSurface: Egy olyan felületgyűjtemény, amely állhat egy lineáris poligonból vagy CurvePolygon-ból.

PolyhedralSurface, Triangle, Tin: További lehetőségek geometriák tárolására, lásd: https://postgis.net/docs/using_postgis_dbmanagement.html#PolyhedralSurface

Miért ST-vel kezdődnek a függvények?

S=Spatial, T=Temporal. Térbeli és időbeli adatok kezelésérére lettek kitalálva (habár az adatok időbeli kezelését kevésbe használjuk).

Hogyan adjunk meg az egyes geometriákat és a térbeli referenciarendszert?

`ST_Point(X,Y)` → X: hosszúság, Y: szélesség, pont ismeretlen SRID-vel

`ST_Point(X,Y,SRID)` → X: hosszúság, Y: szélesség, SRID.

Ezen kívül a következő függvényekkel megadható Z vagy M koordináta, vagy mindkettő. SRID az előző analógiájára definiálható.

`ST_PointM(X,Y,M)` vagy `ST_PointZ(X,Y,Z)` vagy `ST_PointZM(X,Y,Z,M)`

`ST_PointM(X,Y,M, SRID)` vagy `ST_PointZ(X,Y,Z,SRID)` vagy `ST_PointZM(X,Y,Z,M,SRID)`

`ST_MakePoint(X,Y)` vagy `ST_MakePoint(X,Y, Z)` vagy `ST_MakePoint(X,Y, Z, M)` → Gyorsabb és pontosabb, mint az `ST_GeomFromText`

`ST_MakeLine(Point, Point)` → Pontokból vonalat készít

`ST_MakePolygon(LineString)` → Vonalat poligonná konvertálja

Elmélyedés a POSTGIS függvényeiben

Térbeli referenciarendszerre (vetületre) vonatkozó lekérdezések

Mi a geometria SRID-je?

`ST_SRID(geometriát tartalmazó mező neve továbbiakban geom)` → lekérdezi az SRID-t az egyes rekordokra

```
select ST_Srid(t_geom) from telepulesek
```

Frissítsük az SRID-t!

`UpdateGeometrySrid('táblanév', 'geom ', 'EPSG-szám')` → frissíti az SRID-t a megadott rekordnál.

```
select UpdateGeometrySRID('telepulesek','t_geom','23700')
```

Vetületi transzformációk

`ST_Transform(geom, EPSG-szám)` → áttranszformálja a geometriát az adott vetületbe.

```
select t_name, ST_Transform(t_geom, 4326) from telepulesek
```

Geometriára vonatkozó lekérdezések

Kérdezzük le a geometria típusát!

`ST_GeometryType(geom)` → Az eredmény tartalmazza az ST előtagot (új függvény), pl. `ST_Multipolygon`

`GeometryType(geom)` → Az eredmény az adattípust tartalmazza pl. MULTIPOLYGON.

```
SELECT ST_GeometryType(t_geom) As new_name, GeometryType(t_geom) As old_name from
telepulesek
```

`ST_NumGeometries(geom)` →A geometria számossága

Kérdezzük le a geometria és a koordináták dimenzióját!

`ST_Dimension(geom)` → a geometria dimenziója

`ST_CoordDim(geom)` → a koordináták dimenziója

Pl. Z és/vagy M koordináta esetén a kettő eltérhet

```
SELECT ST_Dimension(t_geom) As gdim, ST_CoordDim(t_geom) as cdim from telepulesek
```

A geometria érvényessége

`ST_isValid(geom)` → megvizsgálja, szabályos-e a geometria, pl. nincs-e önmetszés stb.

```
SELECT ST_IsValid(t_geom)  from telepulesek
```

A csomópontok számossága

Egy objektum (feature) csomópontjainak számosságát, attól függően, hogy sima, vagy multigeometriáról van szó, a következő két függvénnyel lehet kimutatni.

`ST_NPoints(geom)` → csomópontok száma poligon és bármely multigeometria esetén is (összes elem összes csomópontja)

`ST_NumPoints(geom)` → csomópontok száma egyszerű geometria esetén LineStringnél, multigeometria esetén `Null`-t ad vissza

```
SELECT ST_NPoints(t_geom) As npoints, ST_NumPoints(t_geom) As numpoints from
telepulesek
```

Mérések

Ahhoz, hogy megmérjük egyes elemek hosszát, kerületét, területét, vagy elemek egymástól való távolságát stb. a következő dolgoknak kell a tudatában lennünk.

- Hány dimenziósak a koordinátáink (2 v. 3.)

- Síkon vagy térben mérünk (melyik PostGIS függvényt használjuk?)

- Melyik koordinátarendszerben (vetületben) mérjük a távolságot?

- Síkon vagy gömbön/alapfelületen érdemes azt a távolságot mérni? Pl. több tíz/száz kilométeres távolságok esetén már jelentős különbség lehet!

Síkbeli mérések (→ térképi méretek)

A síkbeli méréseknél az eredmény a kiindulási adattól függ. Ha az elem koordinátái méterben vannak, akkor az eredmény is méterben vagy négyzetméterben lesz. (Angolszász mértékegységeknél láb, négyzetláb stb.). Ha fokban van a vetület, akkor az eredmény fokban lesz, ilyenkor a föld görbületét nem veszi figyelembe, így nagyobb területeknél már jelentős eltérések lehetnek.

`ST_Length(geom)` → hossz (`ST_Length2D` ugyanez)

```
SELECT utszam, ST_Length(f_geom) from foutak
```

`ST_Length3D(geom)` → hossz, figyelembe veszi a Z koordinátát

`ST_Perimeter(geom)` → poligon körvonalának hossza (ha lyukas a poligon, a gyűrű körvonalát is beleveszi). `ST_Perimeter2D` ugyanígy dolgozik.

```
SELECT j_name, ST_Perimeter(j_geom) from jarasok
```

`ST_Area(geom)` → poligon területe

```
SELECT j_name, ST_Area(j_geom) from jarasok
```

Geodéziai / gömbi / ellipszoidi mérések

`ST_LengthSpheroid(geom, ellipszoid adatai)` → adott elem ellipszoidi méretei metrikus rendszerben (m).

```
select ST_LengthSpheroid(ST_GeomFromText('LINESTRING(19.1 47.5, 18.5 47.2)') ,
'SPHEROID["WGS_84",6378137,298.257223563]')
```

Egyéb mérések

`ST_Distance(geom A, geom B)` → két geometria távolsága (méterben/fokban/szögmértékben)

```
select ST_Distance(ST_GeomFromText('LINESTRING(628200 215000, 629000
216000)'),ST_GeomFromText('POINT(632500 215500)'))
```

`ST_3DDistance(geom A, geom B)` → függvénnyel térképi távolságot tudunk számítani két térképi pont között, eközben a Z koordinátát is figyelembe vesszük.

`ST_MaxDistance(geom A, geom B)` → két geometria között a legnagyobb távolság (nyilván két pont között ugyanazt fogja adni, mint a sima `ST_Distance`)

```
select ST_MaxDistance(ST_GeomFromText('LINESTRING(628200 215000, 629000
216000)'),ST_GeomFromText('POINT(632500 215500)'))
```

`ST_DistanceSphere(pont A fokban, pont B fokban)` → két pont közötti gömbi távolságot adja vissza méterben. A gömb sugara: r=6 371 008 m

```
 select ST_DistanceSphere(ST_GeomFromText('POINT(19.1 47.5)'),
ST_GeomFromText('POINT(18.5 47.2)'))
```

`ST_DistanceSpheroid(pont A fokban, pont B fokban, ellipszoidi felület leírása)` → két pont közötti ellipszoidi távolságot adja vissza méterben.

```
select ST_DistanceSpheroid(ST_GeomFromText('POINT(19.1 47.5)') ,
ST_GeomFromText('POINT(18.5 47.2)'),'SPHEROID["WGS 84",6378137,298.257223563]')
```

`ST_ShortestLine(geom A, geom B)` → két geometria közötti legrövidebb egyenes vonalat adja vissza (kezdő és végpont koordinátái)

Nézzük meg Nógrád és Somogy megye között, mi a legrövidebb vonal?

```
select ST_ShortestLine((select m_geom from megyek where m_megye_kod=15),(select
m_geom from megyek where m_megye_kod=13))
```

`ST_3DShortestLine(geom A, geom B)` → legrövidebb háromdimenziós vonal, vagyis a Z koordinátát is figyelembe veszi

`ST_LongestLine(geom A, geom B)` → két geometria között a leghosszabb vonal

Nézzük meg Nógrád és Somogy megye között, mi a leghosszabb vonal?

```
select ST_ShortestLine((select m_geom from megyek where m_megye_kod=15),(select
m_geom from megyek where m_megye_kod=13))
```

`ST_3DLongestLine(geom A, geom B)`→ leghosszabb 3D-s vonal, vagyis a Z koordinátát is figyelembe veszi

`ST_ClosestPoint(geom A, geom B)` → az A geometria B-hez legközelebbi pontjával tér vissza. Ez a legrövidebb vonal kezdőpontja is egyben.

```
select ST_ClosestPoint((select m_geom from megyek where m_megye_kod=15),(select
m_geom from megyek where m_megye_kod=13))
```

`ST_3DClosestPoint(geom A, geom B)`→ az A geometria B-hez legközelebbi pontjával tér vissza. Ez a legrövidebb vonal kezdőpontja is egyben. Z koordinátáit figyelembe veszi.

`ST_Azimuth(pont A, pont B)` → két pont által megadott egyenes Északi iránnyal bezárt szöge, vagyis az azimut, ahol észak= 0; északkelet = π/4; kelet = π/2; délkelet = 3π/4; dél = π; délnyugat 5π/4; nyugat = 3π/2; északnyugat = 7π/4.

```
select degrees(ST_azimuth(ST_GeomFromText('POINT(628200 215000)'),
ST_GeomFromText('POINT(632500 215500)')))
```

`degrees(fok)`→ visszaadja a radián értéket fokban.

`ST_Angle(vonal A, vonal B)` vagy `ST_Angle(pont A, pont B, pont C)` vagy `ST_Angle(pont A, pont B, pont C, pont D)` → két vonal vagy néhány pont által bezárt szöget számolja ki. Három pont esetén (ABC), a B-nél lévő szöget, négy megadott pontnál: AB és CD szakaszok által bezárt szöget adja vissza radiánban.

Két vonal által bezárt szög

```
select degrees(ST_Angle(ST_GeomFromText('LINESTRING(628200 215000,632500
215500)'), ST_GeomFromText('LINESTRING(629500 212500,628200 212500)')))
```

Három pont által bezárt szög

```
select degrees(ST_Angle(ST_GeomFromText('POINT(628200 215000)'),
ST_GeomFromText('POINT(632500 215500)'), ST_GeomFromText('POINT(629500 212500)')))
```

Négy pont által bezárt szög

```
select degrees(ST_Angle(ST_GeomFromText('POINT(628200 215000)'),
ST_GeomFromText('POINT(632500 215500)'), ST_GeomFromText('POINT(629500 212500)'),
ST_GeomFromText('POINT(628200 212500)')))
```

Hausdorff-távolság: két halmaz elemeinek távolsága egymástól

`ST_HausdorffDistance(geom A, geom B)`

Fréchet-távolság: két görbe (vonal) hasonlósága/különbsége (~távolsága) egymástól

`ST_FrechetDistance(geom A, geom B, float densifyFrac = -1)`

Befoglalók

`ST_Envelope(geom)` → Befoglaló téglalap sarokkoordinátáival tér vissza

`Box2D(geom)` → Bal alsó sarokpont és jobb felső sarokpont koordinátáival tér vissza

```
select ST_Envelope(geom), Box2D(geom) from cshapes
```

```
POLYGON ((-124.71430894851119 24.94638210705216, -124.71430894851119
49.37156646313332, -66.9708358947334 49.37156646313332, -66.9708358947334
24.94638210705216, -124.71430894851119 24.94638210705216))
```

és

```
BOX(-124.71430894851119 24.94638210705216,-66.9708358947334 49.37156646313332)
```

Megemlíteném, hogy van Box3D függvény is.

Hozzáférés egyéb geometriai információkhoz

`ST_Boundary(geom)` → Eredményül az elem határoló vonalát adja.

Ha az elem poligon (akár belső gyűrűvel), akkor az eredmény egy MultiLineString lesz, amely az elem körvonala (a belső gyűrű is számít). Ha az elem vonal, az első és utolsó csomópontja lesz a határoló vonal (MultiPoint-ként)

```
SELECT ST_Boundary(geom)FROM (SELECT 'LINESTRING( 10 130, 50 190, 110 190, 140
150, 150 80, 100 10, 20 40 )'::geometry As geom) As f;
→MULTIPOINT ((10 130), (20 40))
```

`ST_StartPoint(geom)` és `ST_EndPoint (geom)` →

A LineString és CircularLineString első vagy utolsó pontjával tér vissza. `Null` ha nem ilyen típusú az elem, akkor is ha MultiLineString(!).

```
SELECT ST_StartPoint(geom)FROM (SELECT 'LINESTRING( 10 130, 50 190, 110 190, 140
150, 150 80, 100 10, 20 40 )'::geometry As geom) As f;
→POINT (10 130)
```

```
SELECT ST_EndPoint(ST_Boundary(geom)) FROM (SELECT 'POLYGON (( 10 130, 50 190, 110
190, 140 150, 150 80, 100 10, 20 40, 10 130 ), ( 70 40, 100 50, 120 80, 80 110, 50
90, 70 40 ))'::geometry As geom) As f; → NULL
```

`ST_ExteriorRing(geom)` → A poligon külső gyűrűjével tér vissza, LineString-ként. `Null` ha a geometria nem polygon, és akkor is `Null`, ha MultiPolygon az adattípus!

```
SELECT ST_ExteriorRing(geom) FROM (SELECT 'Polygon(( 10 130, 50 190, 110 190, 140
150, 150 80, 100 10, 20 40, 10 130 ))'::geometry As geom) As f;
→ LINESTRING (10 130, 50 190, 110 190, 140 150, 150 80, 100 10, 20 40, 10 130)
```

`ST_IsRing(geom)` → Azt teszteli, hogy a LineString zárt-e és nincs-e önmetszése. True vagy False értékkel tér vissza

`ST_IsSimple(geom)` → Ha az elemnek nincsenek önmetszései vagy ön-érintése, akkor True, ellenkező esetben False.

`ST_IsClosed(geom)` → Megnézi, hogy a LineString első és utolsó pontja megegyezik-e? (érvényes még CircularString-re, Curves-re és PolyhedralSurface-re).

```
SELECT ST_IsClosed('LINESTRING(0 0, 0 1, 1 1,0 0)');
→True
SELECT ST_IsClosed('LINESTRING(0 0, 0 1, 1 1)');
→False
```

`ST_X(geom), ST_Y(geom), ST_Z(geom)` → a megadott pont X vagy Y vagy Z koordinátájával tér vissza

```
SELECT ST_X('POINT(1 2 3)');
→1
```

`ST_IsPolygonCW(geom)` vagy `ST_IsPolygonCCW(geom)` →

Azt teszteli, hogy a poligon (multipoligon is) az óra mutató járásával megegyező vagy ellentétes irányban van a külső vagy belső gyűrűje.

CCW- Külső gyűrű óramutatóval ellentétes, belső gyűrű megegyező irányú (True)

CW –Külső gyűrűje óramutatóval megegyező, a belső ellentétes irányú (True)

`ST_NumGeometries(geom)` → A csoportgeometriák vagy a GeometryColletion-ök elemszámának tesztelésére. Hány részből áll a geometria?

```
SELECT ST_NumGeometries(geom) from cshapes → 129
```

A geometria szerkesztése, feldolgozása

`ST_Buffer(geom, a pufferzóna szélessége (sugara), egyebek)` → Pufferzónát képez a megadott szélességgel. Negatív szám is lehet, ekkor a keletkező poligon kisebb lesz (akár zéró méretű is lehet). A szélesség az épp használt SRID mértékegységében értendő. A harmadik paraméter opcionális:

quad_segs = a vonalszegmensek száma, amivel egy negyed kör alakú pufferzóna-poligont létrehozunk. A default = 8.

endcap = round | flat | square →vonalvég stílusa: kerek, egyenes és egyenes, de továbbhúzott. default = round.

join = round | mitre | bevel → vonaltörés stílusa: kerek, hegyes, vagy levágott. Default = round

mitre_limit→ vonalvég hegyességének/levágottságának mértéke (miter-t is elfogad).

side = both | left | right → egy vagy kétoldali pufferzóna?

```
select t_name, ST_Buffer(t_geom, 1000) from telepulesek where
t_name='Százhalombatta'
select p_name, ST_Buffer(p_geom, 500,'endcap=flat' ) from patak where
p_name='Benta-patak'
```

`ST_Centroid(geom)` → A geometria (bármely) centroidját adja vissza

```
select t_name, ST_Centroid(t_geom) from telepulesek
```

ST_ConvexHull(geom) → A legkisebb területű konvex geometriát adja vissza.

```
select t_name, ST_ConvexHull(t_geom) from telepulesek
```

ST_ConcaveHull(geom, target_percent, allow_holes) → Egy lehetséges konkáv geometriát ad vissza. A target_percent 0-1 között (1: konvex burok), közötte az elem körvonalának egyszerűsítését adja meg. Az allow_holes boolean változó megadja, hogy lehet-e lyuk a poligonban.

```
select t_name, ST_ConcaveHull(t_geom, 0.5) from telepulesek where
t_name='Százhalombatta'
```

ST_DelaunayTriangles(geom, tolerancia) → Az elemet felosztja Delauney-háromszögekre.

ST_LineMerge(geom) → A megadott geometriákat egy MultiLineStringgé vonja össze.

ST_OrientedEnvelope(geom) →A legkisebb (elforgatott) befoglaló téglalappal tér vissza.

```
select t_name, ST_OrientedEnvelope(t_geom) from telepulesek where
t_name='Százhalombatta'
```

ST_Simplify(geom, tolerance)→ A Douglas-Peucker-algoritmust használja vonalegyszerűsítéshez

```
select t_name, ST_Simplify(t_geom, 500) from telepulesek where
t_name='Százhalombatta'
```

ST_SimplifyVW(geom, tolerance) →Visvalingam-Whyatt-algoritmust használja a vonalegyszerűsítéshez.

```
select t_name, ST_SimplifyVW(t_geom, 500) from telepulesek where
t_name='Százhalombatta'
```

ST_ChaikinSmoothing(geom, iterációk_száma, preserve_endPoints) → A Chaikin-algoritmust használja, hogy simítsa és görbe jellegűre formálja a vonalat. Meg kell adni az iterációk számát, és hogy a végpontokat megőrizzük-e. Az iterációk száma max. 5.

```
select p_name, ST_ChaikinSmoothing(p_geom,3,true ) from patak where p_name='Benta-
patak'
```

A topológia kapcsolatok vizsgálata

A KÖVETKEZŐ FÜGGVÉNYEK MINDEGYIKE TÉRBELI KAPCSOLATOKAT VIZSGÁL: VISSZATÉRÉSI ÉRTÉKÜK TRUE VAGY FALSE.

Ez a weboldal részletesen tárgyalja, és ábrákkal segíti a kapcsolatok megértését:

http://postgis.net/workshops/postgis-intro/spatial_relationships.html

ST_Intersects(geom A, geom B) → Ha az A és B geometriának valahol a síkban/térben van közös pontja, a kettő metszi egymást. (pl. két pont megegyező helyen van, valahol a vonalon van egy pont, a felület érintkezik bárhol egy vonallal, vagy a felület érinti vagy átfedi a másik felületet, a felület érintkezik vagy átfed egy ponttal)

`ST_Disjoint(geom A, geom B)` → Az intersects ellentéte: ha két geometria sehol sem érintkezik.

`ST_Equals(geom A, geom B)` → Két geometria térbeli egyezőségét vizsgálja. Ha a két geometria tökéletesen megegyezik, akkor igaz a kapcsolat. (pont-pont, vonal-vonal, poligon-poligon stb. ezek multi/csoport változataik).

`ST_Crosses(geom A, geom B)` → Két különböző geometria metszésének tesztelésére. multipoint/polygon, multipoint/linestring, linestring/linestring, linestring/polygon és linestring/multipolygon

`ST_Overlaps(geom A, geom B)` → Két egyező dimenziójú geometriát hasonlít össze, hogy átfednek-e. Pont-Pont, vonal-vonal és poligon-poligon.

`ST_Touches(geom A geom B)` → azt teszteli, hogy két geometria érinti-e egymást. Pl. a pont rajta van-e a vonalon, vagy poligon körvonalán, a vonalak érintik-e egymást közös szakaszon, vagy a poligon körvonalát érinti a vonal vagy pont.

`ST_Contains(geom A, geom B)` → Azt teszteli, hogy az A geometria tökéletesen tartalmazza-e B-t. A B geometria akkor esik bele tökéletesen A-ba, ha B-ből egy pont sem fekszik az A körvonalán (külső gyűrűjén), és minimum egy pont a B geometriából az A geometria belsejére esik. Tehát ha egy vonal vagy egy pont fekszik a poligon körvonalán, akkor hamis értéket kapunk vissza, mert a poligon belsejére nem esik pont! Két egyforma poligonnál igaz érték tér vissza.

`ST_Within(geome A, geom B)` → Azt teszteli, hogy az A geometria tökéletesen tartalmazza-e a B-t. A B geometria akkor esik bele tökéletesen A-ba, ha B-ből egy pont sem fekszik túl az A körvonalán (külső gyűrűjén), és minimum egy pont a B geometriából az A geometria belsejére esik. Tehát ha egy vonal vagy egy pont fekszik a poligon körvonalán, akkor hamis értéket kapunk vissza, mert a poligon belsejére nem esik pont! Két egyforma poligonnál igaz érték tér vissza.

Ne felejtsük el, hogy:

`ST_Contains(geom A, geom B) = ST_Within(geom B, geom A)`

`ST_Covers(geom A, geom B)` → Azt teszteli, hogy az A geometria tökéletesen tartalmazza-e B-t. A B geometria akkor esik bele tökéletesen A-ba, ha B-ből egy pont sem fekszik túl az A körvonalán (külső gyűrűjén). Tehát ami a körvonalon van pl. poligon körvonalán fekvő vonal vagy pont akkor igaz értéket kapunk vissza

`ST_CoveredBy(geom A, geom B)` → Azt teszteli, hogy az A geometria tökéletesen tartalmazza-e B-t. A B geometria akkor esik bele tökéletesen A-ba, ha B-ből egy pont sem fekszik túl az A körvonalán (külső gyűrűjén). Tehát ami a körvonalon van pl. poligon körvonalán fekvő vonal vagy pont akkor igaz értéket kapunk vissza.

Ne felejtsük el, hogy:

`ST_Covers(geom A, geom B)=ST_CoveredBy(geom B, geom A)`

Két geometria egymással való kombinálása

`ST_Intersection(geom A, geom B, gridSize)` → Két geometria közös részét adja vissza. Ha adott a gridSize paraméter (opcionális), akkor az input geometria a megadott gridhez igazodik, az eredmény geometriában a csomópontok ugyanarra a grid méretre lesznek kiszámítva.

`ST_Difference(geom A, geom B, gridSize)` → Visszatér az A geometria azon részével, amely nem fed át B-vel. → Másképpen: `GeomA - ST_Intersection(geom A, geom B)`. Ha adott a gridSize paraméter (opcionális), akkor az input geometria a megadott gridhez igazodik, és az eredmény geometriában a csomópontok ugyanarra a grid méretre lesznek kiszámítva.

`ST_SymDifference(geom A, geom B, gridSize)` → Két geometria azon részét kivonatolja, amelyik nem fed át egymással. → Másképpen: `ST_Union(geom A, geom B) - ST_Intersection(geom A, geom B)`. → Másképpen: `ST_SymDifference(geom A, geom B) = ST_SymDifference(geom B, geom A).` Ha adott a gridSize paraméter (opcionális), akkor az input geometria a megadott gridhez igazodik, és az eredmény geometriában a csomópontok ugyanarra a grid méretre lesznek kiszámítva.

`ST_Union(geom A, geom B, gridSize)` → Minden elemet összeolvaszt egy nagy elemmé (Multi típus, vagy GeometryCollection). Nem maradnak átfedések. Felhívnám a figyelmet, hogy nem csak két geometria összeillesztése működik, hanem tetszőleges számúéra. Ilyenkor a mező vagy geometriát tartalmazó adattömb nevét kell megadni.

```
SELECT ST_Union(b_geom) from belteruletek
```

Ha adott a gridSize paraméter (opcionális), akkor az input geometria a megadott gridhez igazodik, és az eredmény geometriában a csomópontok ugyanarra a grid méretre lesznek kiszámítva.

`ST_Split(geome A, geom B)` → A geometriát A elvágjuk a B geometriával. Például: vonal vágása ponttal, vonallal és poligonnal valamint ezek multi/csoport változataival. Poligon vagy multipoligon vágása vonallal.

Formátum konverzióra alkalmas függvények

A formátum konverzióra alkalmas függvények arra jók, hogyha az adatbázisból szeretném kiíratni a WKB-ben vagy WKT-ben tárolt geometriákat például egy szöveges fájlba, akkor ezek elvégzik az adatkonverziót. Gyakori feladat, hogy a lekérdezés eredményét szeretnénk viszontlátni a weben. Ilyenkor használhatjuk pl. a KML és GeoJSON formátumokat. Illetve ennek inverze, hogy KML-ben vagy GeoJSON-ben tárolt adatokat szeretnénk az adatbázisba importálni, akkor is szükség van akonverzióra. Alább a leggyakrabban használt függvények olvashatóak.

`ST_GeomFromGeoJSON(text)` → WKT geometriát készít GeoJSON-ből.

`ST_GeomFromKML(text)` → WKT geometriát készít KML-ből

`ST_AsGeoJSON(geom)` → GeoJSON geometriát készít. Példa a 7. fejezetben.

`ST_AsKML(geom )` → KML geometriát készít

```
SELECT ST_AsKML(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

`ST_AsSVG()` → SVG geometriát készít

`ST_AsLatLonText(geom)` → Szélességet és hosszúságot ad meg

Egyebek

`String_Agg()` → ez a függvény a PostgreSQL része, aggregáló (csoportfüggvény). Arra szolgál, hogy szövegeket fűzzön össze, de csak azokat, amelyek a megadott csoport részei. Meg lehet adni az elemek közti elválasztó karaktert is. Pl. `string_agg(j_name, ',')`

5. FEJEZET: GYAKORLÓ FELADATOK POSTGIS-ben

0. rész: SQL kiegészítés: elágazások

CASE
WHEN feltétel THEN eredmény
WHEN feltétel THEN eredmény
….
ELSE eredmény
END

1. Írjuk ki a könyvtár adatbázisból a könyvtárosok mellé, milyen régóta állnak alkalmazásban: Aki 1995-01-01 előtt lépett be az 'törzsgárda tagja', aki 1995-01-01 és 2000-01-01 között az régi motoros, és aki 2000-01-01 után lépett be az zöldfülű.
2. Emeljük meg a könyvtárosok fizetését! Írassuk ki a nevüket, beosztásukat, a régi és az új fizetésüket. A fizetés emelése beosztás szerint történjen.
   PULTOS: 20%-os emelés
   RAKTÁROS: 15%-os emelés
   ADMINISZTRÁTOR: 25%-os emelés
   VEZETŐ: 17%-os emelés

Megyék tábla frissítése vármegyékre

3. Szúrjon be egy új mezőt a `megyek` táblába, de ott már vármegye legyen a földrajzi köznév a megye tulajdonneve után!

1. rész: Geometria létrehozása, lekérdezése

1. Adjunk meg egy pontot Budapestnél (47.5° és 19.1°), (ST_Point).
2. Adjunk meg ugyanezt a pontot, de definiáljunk hozzá térbeli referenciarendszert is (EPSG:4326)!
3. Adjunk meg egy pontot WKT formátumban és készítsünk belőle geometriát (Budapest 47.5° és 19.1°).
4. Adjunk meg egy pontot WKT formátumban és készítsünk belőle geometriát! Definiáljuk hozzá a 4326-os vetületet (Budapest 47.5° és 19.1°).
5. Adjuk meg ugyanezt a pontot és referencia rendszert az GeomFromEWKT() függvénnyel!
6. Készítsünk egy vonalat Budapest–Sárbogárd–Siófok irányban, WKT formátumban (47.5° 19.1°, 46.88° 18.62°, 46.9° 18.05°).
7. Készítsünk egy poligont WKT formátumban, amelynek sarokpontjai a következő nagyvárosokban vannak: Budapest–Sárbogárd–Siófok–Tatabánya (47.5° 19.1°, 46.88° 18.62°, 46.9° 18.05°, 47.58° 18.4°).
8. Tegyünk az előző poligonba egy gyűrűt (47.22° 18.66°, 47.24° 18.6°, 47.18° 18.55°)!
9. Készítsünk a Budapest–Sárbogárd–Siófok–Tatabánya (47.5° 19.1°, 46.88° 18.62°, 46.9° 18.05°, 47.58° 18.4°) poligonból vonalat (ST_MakePolygon).
10. Készítsünk egy vonalat Budapest és Székesfehérvár koordinátájából (47.5° 19.1°, 47.2° 18.4°) (ST_MakePoint, ST_MakeLine)!
11. Kérdezzük le a Magyarország adatbázisból a geometriákat! Figyeljük meg a geometriák leírását: a „telepulescentroid”, a „patak”, és a „jarasok” táblákban.
12. Kérdezzük le a geometria típusát, a geometria dimenzióját és a koordináták dimenzióját az előző táblákból. Figyeljük meg, mi történik az eredménytáblában!
13. Nézzük meg a multigeometriák (csoportgeometriák) és az egyszerű geometriák csomópontjainak számosságát (ST_Npoints, ST_NumPoints). Ehhez vizsgáljuk meg a járások és a patakok táblát. Egyszerű geometriákhoz használhatjuk a 6. és a 7. feladatban elkészült lekérdezést.
14. Vizsgáljuk meg a geometria számosságát a belterület táblában!

2. rész: Mérések, koordináták, vetületi konverzió

1. Kérdezzük le a települések táblából az adatokat. Nézzük meg a geometriát a megjelenítő felületen.
2. Mi a megye táblában lévő elemek vetülete? Írassuk ki a megye nevét és vetületének EPSG számát!
3. Mekkora az egyes megyék területe? Írassa ki a nevüket és területüket! Mi a mértékegysége?
4. Hány km^2 az egyes megyék területe? Rendezzük a listát úgy, hogy a legnagyobb területű megye legyen legelöl a listában!
5. Írassuk ki, hogy a megye réteg vetületében (EOV: 23700) és Web Mercator vetületben (3857) mekkora az egyes megyék területe km^2-ben! Az adatokat rendezzük növekvő sorrendben az EOV-ben vett méret szerint.
6. Mekkora az egyes megyék kerülete? Mi a mértékegysége?
7. Írassuk ki, hogy a megye réteg vetületében (EOV: 23700) és Web Mercator vetületben (3857) mekkora az egyes megyék kerülete méterben! Az adatokat rendezzük növekvő sorrendben az EOV-ben vett méret szerint.
8. Milyen hosszúak a patakok (írassa ki a nevüket és a hosszúságukat)!
9. Milyen hosszúak a patakok a megye réteg vetületében (EOV: 23700) és Web Mercator vetületben? Az adatokat rendezzük növekvő sorrendben az EOV-ben vett méret szerint.
10. Milyen messze van egymástól Sárospatak és Barcs (vetületi távolság)?
11. Milyen messze van egymástól Sárospatak és Barcs centroidja?
12. Milyen messze van egymástól Sárospatak és Barcs centroidja, ha a gömbi távolságra vagyunk kíváncsiak?
13. Készítsük el a belterületek réteg centroidjai alapján a telepulescentroid táblát![1]
14. Mi Barcs centroidjának EOV koordinátája és mi a földrajzi (4326)?
15. Milyen messze van Barcs a X=600 000 és Y=200 000 EOV koordinátáktól?
16. Milyen messze van Barcs a 47.5° és 19.1° földrajzi koordinátájú ponttól (méterben)?
17. Mi a legkisebb és a legnagyobb távolság Barcs centroidja és Pécs poligonja között?
18. Adjuk meg légvonalban a legrövidebb és leghosszabb vonalat Barcs centroidja, és Pécs poligonja között?
19. Írassuk ki, hogy ha Barcsról indulunk, a Balaton partjának melyik pontja lesz a legközelebb légvonalban? Mekkora ez a távolság?
20. Ez a pont melyik településhez tartozik?[2]
21. Melyik az a település Mo-on, amelyik légvonalban a legmesszebb esik Barcstól, és milyen messze van?
22. Mekkora szöget zár be északkal a Barcs és Magosliget által megadott egyenes? [3]
23. És a fordított irányú, Magosliget és Barcs közötti?

1 Az adatfeltöltés órához kapcsolódó kérdés

2 A Térbeli kapcsolatokat vizsgáló függvények témakörhöz kapcsolódik: ST_Intersects

3 Matematikai függvények: https://www.postgresql.org/docs/9.5/functions-math.html

3. rész: Hozzáférés geometriai információkhoz, a geometria szerkesztése, feldolgozása.

1. Írassuk ki Barcs település befoglaló téglalapját!
2. Írassuk ki Barcs település befoglaló téglalapjának azt a 4 koordinátáját, amelyből a befoglaló visszaállítható (box)!
3. Most adjuk meg ugyanezt a 4 koordinátát földrajzi koordinátaként!
4. Adjuk meg Pest megye körvonalát (vonalként, ST_Boundary)!
5. Nézzük meg a 1.8-as feladatban definiált gyűrűs poligon külső gyűrűjét (ST_ExteriorRing)!
6. Nézzük meg, hogy szintén az 1.8 feladatban szereplő poligon gyűrűje az óramutató járásával megegyező irányú-e vagy ellentétes (ST_IsPolygonCW, ST_IsPolygonCCW)?
7. Készítsünk egy vonalat WKT formátumban, amelynek sarokpontjai a következő nagyvárosokban vannak: Budapest–Székesfehérvár–Siófok–Tatabánya–Budapest. (47.5° 19.1°, 47.2° 18.4°, 46.9° 18.05°, 47.58° 18.4°, 47.5° 19.1°). Nézzük meg, hogy önmagába visszatér-e a LineString (ST_IsClosed)?
8. Az előző feladatban szereplő vonalnak van-e önmetszése (ST_IsSimple)?
9. Nézzük meg, hogy a 7. feladatban szereplő vonal, zárt vonal-e (ST_IsRing)?
10. Készítsünk egy vonalat WKT formátumban, amelynek sarokpontjai a következő nagyvárosokban vannak: Budapest–Székesfehérvár–Siófok–Tatabánya (47.5° 19.1°, 47.2° 18.4°, 46.9° 18.05°, 47.58° 18.4°). Ezt is vizsgáljuk meg, hogy zárt-e a vonal?
11. Írjuk ki a településcentroid rétegről Barcs település X és Y koordinátáját!
12. Határozzuk meg a település réteg centroidjait, és az egyes települések befoglaló téglalapjait, és konvex burkát!
13. Mekkora a konvex burok területe és a település területe?
14. Készítsük el a Dera-patak 1 km széles pufferzónáját (mindkét irányban 1-1 km a patak vonalától)!
15. Az előző pufferzóna a végeknél legyen egyenes, és érjen véget pontosan a vonalvégnél. A vonal törése lekerekített.
16. Írassuk ki csak a jobboldali pufferzónát!

4. rész: Térbeli kapcsolatokat elemző, és geometriát készítő függvények

1. Vizsgáljuk meg, hogy metszi-e valahol a Tisza és a Duna egymást?
2. Nézzük meg, hogy a Duna és a Tisza „nem metszi”-e egymást? (disjoint)
3. Vizsgáljuk meg, hogy a 616100, 206300 EOV koordinátájú pont Fejér megyében van-e?
4. Vizsgáljuk meg, hogy 47.2° 18.6° Fejér megyében van-e?
5. Vizsgáljuk meg, hogy Fejér érintkezik-e Pest megyével? Az alábbiak közül melyik térbeli relációnál kapunk még igaz eredményt és miért? (intersects, touches, disjoint, crosses, overlaps, within, contains)
6. Vizsgáljuk meg, hogy Fejér megye tartalmazza-e Székesfehérvár poligonját!
7. Vizsgáljuk meg, hogy Fejér megye átfed-e (overlap) vagy keresztezi-e (crosses) Székesfehérvár poligonját!
8. Vizsgáljuk meg az ST_COVERS/ST_CONTAINS és az ST_COVEREDBY/ST_WITHIN és működését. Vegyük észre az árnyalati különbséget!
   -egy ’POLYGON((0 0,0 1,1 1,1 0,0 0))’ benne van-e egy ugyanilyen poligonban? POLYGON((0 0,0 1,1 1,1 0,0 0))
   - egy ’LINESTRING(0 0,0 1,1 1,1 0)’ benne van-e a következő poligonban? POLYGON((0 0,0 1,1 1,1 0,0 0))
   egy ’LINESTRING(0 0,0.5 0.5,0 1,1 1,1 0)’ benne van-e a következő poligonban? POLYGON((0 0,0 1,1 1,1 0,0 0))
   egy ’POINT(0 0)’ benne van-e a következő poligonban? POLYGON((0 0,0 1,1 1,1 0,0 0))
9. Rajzoltassuk ki a Duna Fejér megyei szakaszát!
10. Rajzoljuk ki azokat a nemzeti parkokat, amelyek Veszprém megyén kívül esnek! Ha a nemzeti park határát kettévágja a megyehatár, akkor a másik megyébe eső rész szerepeljen.
11. Írassuk ki a Fejér megyei települések neveit!
12. Írassuk ki a Fejér megyei települések neveit és rajzoljuk ki a geometriájukat! Csak azok a települések szerepeljenek a listában, amelyek Fejér megyében vannak.
13. Írassuk ki a nem Fejér megyei települések neveit!
14. Írassuk ki a nem Fejér megyei települések neveit és rajzoljuk ki a geometriájukat!
15. Írassuk ki az M7-es autópálya Fejér megyei szakaszát!
16. Nézzük meg, hogy milyen hosszú az M7-es autópálya Fejér megyei szakasza (összesen)!
17. Milyen hosszú egy irányban az út? (forward és backward)
18. Nézzük meg, hogy mekkora a Veszprém megyei védett területek mérete km^2-ben?
19. Ez a megye területének hány százaléka?
20. Az egész országterületének hány százaléka védett terület?
21. Mely geoládák találhatók Fejér megyében?
22. Hány ilyen geoláda van?
23. Mely geoládák vannak Fejér megyén kívül?
24. Mely geoládák találhatók a Duna 10 kilométeres pufferzónájában?
25. Mely geoládák találhatók az M7es autópálya 10 kilométeres pufferzónájában?
26. Mely geoládák találhatók az M7es autópálya Fejér megyei szakaszának 10 kilométeres pufferzónájában?
27. Melyik patak fekszik a legközelebb Hollókő várához, mekkora ez a távolság?
28. Hol van a pataknak az a pontja, amely a legközelebb esik a várhoz? (koordinátát kérek!)
29. Vágjuk félbe Magyarországot (országpoligon) a 47°-es szélességi körnél (a 16° és 22° hosszúsági kör között).
30. Rajzoljuk ki azokat a területeket, amelyek vagy a Budai Tájvédelmi körzet részei, vagy Budapest közigazgatási területe. A közös részek ne látszanak a geometriában.
31. Írassa ki, hogy melyik járásban melyik vár található. Rendezzen a járás neve szerint növekvő sorrendbe.

32. Fogalmazza meg a kérdést, az alábbi lekérdezéshez, anélkül, hogy lefuttatná!

```
select st_intersection(
st_buffer(st_setsrid(st_makepoint(
(st_x(st_endpoint(st_shortestline((select st_transform(va_geom,23700) from
var where va_name='Bory-vár'), (select st_union(el_geom) from elsorendu
where el_ref='8'))))
+
st_x(st_endpoint(st_shortestline((select st_transform(va_geom,23700) from
var where va_name='Thury vár'), (select st_union(el_geom) from elsorendu
where el_ref='8'))) )
/2,
(st_y(st_endpoint(st_shortestline((select st_transform(va_geom,23700) from
var where va_name='Bory-vár'), (select st_union(el_geom) from elsorendu
where el_ref='8'))))
+
st_y(st_endpoint(st_shortestline((select st_transform(va_geom,23700) from
var where va_name='Thury vár'), (select st_union(el_geom) from elsorendu
where el_ref='8'))))/2),23700),
st_distance(
st_endpoint(st_shortestline((select st_transform(va_geom,23700) from var
where va_name='Bory-vár'), (select st_union(el_geom) from elsorendu where
el_ref='8'))),
st_endpoint(st_shortestline((select st_transform(va_geom,23700) from var
where va_name='Thury vár'), (select st_union(el_geom) from elsorendu where
el_ref='8'))))
/2),
(select st_union(el_geom) from elsorendu where el_ref='8'));
```

33. Írassuk ki azokat a vetületpárokat, amelyek EPSG számának összege 23700-at ad!
34. Melyek Székesfehérvári járás szomszédos járásai?
35. Írassuk ki az összes járás nevét, és a szomszédaikat egy listába. Szerepeljen az egyik oszlopban a szomszédok száma is.

6. FEJEZET: MEGOLDÁSOK

0. rész: A CASE elágazás

1.
```
select ks_nev, ks_beosztas,
case
      when ks_belep between '1995-01-01' and '2000-01-01' then 'Régi motoros'
      when ks_belep >'2000-01-01' then 'zöldfülű'
      else 'öreg tata'
end alkalmazasiido
   from konyvtaros
```

2.
```
select ks_nev , ks_beosztas, ks_fizetes,
case
      when ks_beosztas='Pultos' then ks_fizetes*1.2
      when ks_beosztas='Raktáros' then ks_fizetes*1.15
      when ks_beosztas='Adminisztrátor' then ks_fizetes*1.25
      when ks_beosztas='Vezető' then ks_fizetes*1.17
end as ujfizetes
from konyvtaros
```

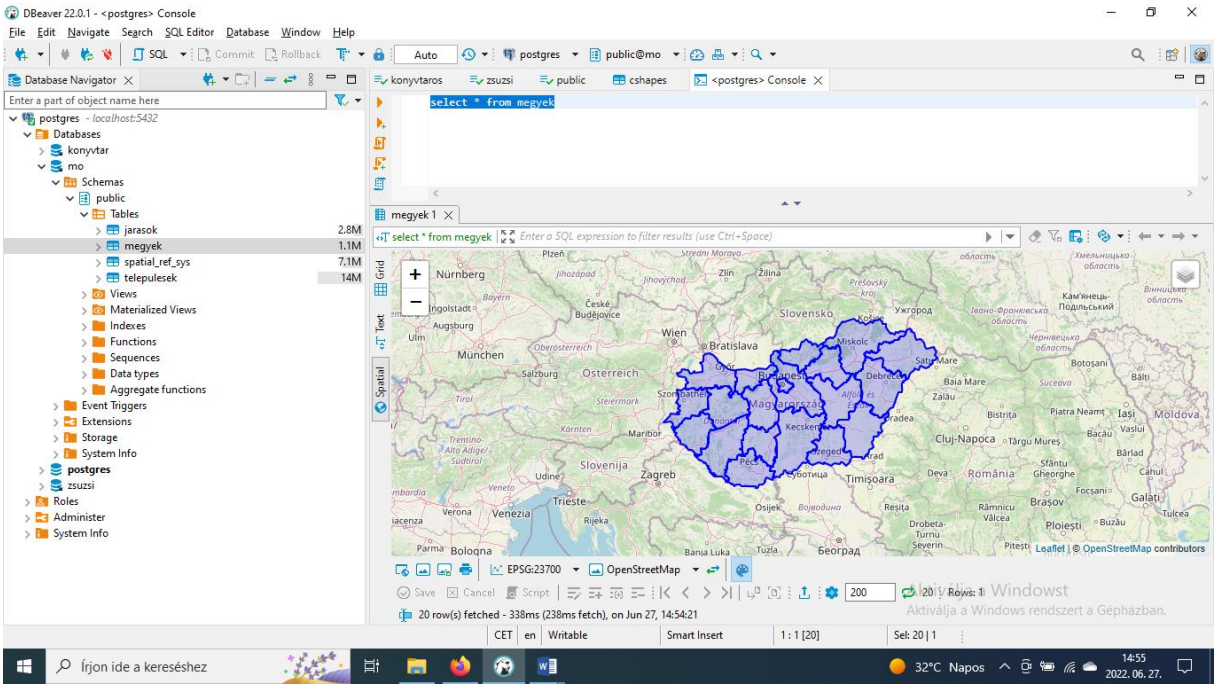
3.
```
ALTER TABLE public.megyek ADD m_varmegye varchar NULL;

update megyek set m_varmegye =(concat(substring(m_name,1,(length(m_name)-
5)),'vármegye')) where substring(m_name,(length(m_name)-5))=' megye'
```

1. rész: Geometria létrehozása, lekérdezése

```
1. SELECT ST_Point( 19.1, 47.5);

2. SELECT ST_SetSRID(ST_Point(19.1, 47.5), 4326);

3.SELECT ST_GeomFromText('POINT(19.1 47.5)');

4. SELECT ST_GeomFromText('POINT(19.1 47.5)', 4326);

5. SELECT ST_GeomFromEWKT('SRID=4326;POINT(19.1 47.5)');

6.     SELECT ST_GeomFromText('LINESTRING(19.1 47.5, 18.62 46.88, 18.05 46.9)',
4326);

7. SELECT ST_GeomFromText('POLYGON((19.1 47.5, 18.62 46.88, 18.05 46.9, 18.4
47.58, 19.1 47.5))', 4326);

8. SELECT ST_GeomFromText('POLYGON((19.1 47.5, 18.62 46.88, 18.05 46.9, 18.4
47.58, 19.1 47.5),(18.66 47.22,18.6 47.24, 18.55 47.18,18.66 47.22))', 4326);
9. SELECT ST_MakePolygon('LINESTRING(19.1 47.5, 18.62 46.88, 18.05 46.9, 18.4
47.58, 19.1 47.5)');
10. SELECT ST_MakeLine(ST_MakePoint(19.1, 47.5), ST_MakePoint(18.4, 47.2));

11. select tc_geom from telepulescentroid;

select p_geom from patak;

select j_geom from jarasok;

12. select ST_CoordDim(j_geom), ST_Dimension(j_geom), ST_GeometryType(j_geom)
from jarasok

select ST_CoordDim(tc_geom), ST_Dimension(tc_geom), ST_GeometryType(tc_geom)  from
telepulescentroid;

select ST_CoordDim(p_geom), ST_Dimension(p_geom), ST_GeometryType(p_geom)  from
patak;

13. select ST_NPoints(j_geom)  from jarasok;

select ST_NPoints(p_geom)  from patak;

SELECT ST_NumPoints(ST_GeomFromText('LINESTRING(19.1 47.5, 18.4 47.2, 18.05
46.9)', 4326));

SELECT ST_NPoints(ST_GeomFromText('POLYGON((19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4
47.58, 19.1 47.5))', 4326));

14. SELECT ST_NumGeometries(b_geom) from belterulet;
```

2. rész: Mérések, koordináták, vetületi konverzió

1. `select * from megyek`



2. `select m_name, ST_Srid(m_geom) from megyek`
3. `select m_name, ST_Area(m_geom) from megyek`
4. `select m_name, ST_Area(m_geom)/1000000 from megyek order by 2 desc`
5. `select m_name, ST_Area(m_geom)/1000000, ST_Area(ST_Transform(m_geom,3857))/1000000 from megyek order by 2 asc`
6. `select m_name, ST_Perimeter(m_geom) from megyek`
7. `select m_name, ST_Perimeter(m_geom), ST_Perimeter(ST_Transform(m_geom, 3857)) from megyek order by 2`
8. `select p_name, ST_Length(p_geom) from patak`
9. `select p_name, ST_Length(p_geom), ST_Length(ST_Transform(p_geom, 3857)) from patak`
10. `select ST_Distance((select t_geom from telepulesek where t_name='Sárospatak'),(select t_geom from telepulesek where t_name='Barcs'))`

11. `select ST_Distance((select ST_Centroid(t_geom) from telepulesek where t_name='Sárospatak'),(select ST_Centroid(t_geom) from telepulesek where t_name='Barcs'))`
12. `select ST_DistanceSphere((select ST_Centroid(ST_Transform(t_geom,4326)) from telepulesek where t_name='Sárospatak'),(select ST_Centroid(ST_Transform(t_geom,4326)) from telepulesek where t_name='Barcs'))`
13. `CREATE TABLE public.telepulescentroid (tc_id int4 NULL, tc_name varchar NULL, tc_geom public.geometry NULL);`

    `INSERT INTO telepulescentroid (tc_id, tc_name, tc_geom) select b_id, b_name, ST_Centroid(b_geom) from belterulet;`

14. `select tc_name, ST_AsText(tc_geom), ST_AsText(ST_Transform(tc_geom,4326)) from telepulescentroid where tc_name='Barcs'`

15. `select ST_Distance((select tc_geom from telepulescentroid where tc_name='Barcs'), ST_GeomFromText('POINT(600000 200000)',23700))`
16. `select ST_Distance((select tc_geom from telepulescentroid where tc_name='Barcs'), ST_Transform(ST_GeomFromText('POINT(19.1 47.5)',4326),23700))`

17. `select ST_Distance((select tc_geom from telepulescentroid where tc_name='Barcs'),(select b_geom from belterulet where b_name='Pécs')), ST_MaxDistance((select tc_geom from telepulescentroid where tc_name='Barcs'),(select b_geom from belterulet where b_name='Pécs'))`

18. `select ST_ShortestLine((select tc_geom from telepulescentroid where tc_name='Barcs'),(select b_geom from belterulet where b_name='Pécs')), ST_LongestLine((select tc_geom from telepulescentroid where tc_name='Barcs'),(select b_geom from belterulet where b_name='Pécs'))`

19. `select ST_Distance((select tc_geom from telepulescentroid where tc_name='Barcs'),(select to_geom from tavak where to_name='Balaton')), ST_Endpoint(ST_Shortestline((select tc_geom from telepulescentroid where tc_name='Barcs'),(select to_geom from tavak where to_name='Balaton')))`

20. `select t_name, ST_Distance((select tc_geom from telepulescentroid where tc_name='Barcs'),(select to_geom from tavak where to_name='Balaton')), ST_Intersects(t_geom, ST_Endpoint(ST_ShortestLine((select tc_geom from telepulescentroid where tc_name='Barcs'),(select to_geom from tavak where o_name='Balaton')))) as pont from telepulesek order by pont desc limit 1`

21. `select t_name, ST_Distance((select tc_geom from telepulescentroid where tc_name='Barcs'), t_geom) from telepulesek order by 2 desc limit 1`

22. `select ST_Azimuth((select tc_geom from telepulescentroid where tc_name='Barcs'), (select tc_geom from telepulescentroid where tc_name='Magosliget') )*180/pi()`

    `select Degrees(ST_Azimuth((select tc_geom from telepulescentroid where tc_name='Barcs'), (select tc_geom from telepulescentroid where tc_name='Magosliget')))`

23. `select ST_Azimuth((select tc_geom from telepulescentroid where tc_name='Magosliget'), (select tc_geom from telepulescentroid where tc_name='Barcs'))*180/pi()`

    `select Degrees(ST_Azimuth((select tc_geom from telepulescentroid where tc_name='Magosliget'), (select tc_geom from telepulescentroid where tc_name='Barcs')))`

3. rész: Hozzáférés geometriai információkhoz, a geometria szerkesztése, feldolgozása.

1. `SELECT ST_Envelope(t_geom) from telepulesek where t_name='Barcs';`
2. `SELECT Box(t_geom) from telepulesek  where t_name='Barcs';`
3. `SELECT Box(ST_Transform(t_geom, 4326)) from telepulesek where t_name='Barcs';`
4. `SELECT ST_Boundary(m_geom) from megyek where m_name like 'Pest%';`
5. `SELECT ST_ExteriorRing(ST_GeomFromText('POLYGON((19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4 47.58, 19.1 47.5),(18.66 47.22,18.6 47.24, 18.55 47.18,18.66 47.22))', 4326));`
6. `SELECT ST_IsPolygonCW(ST_GeomFromText('POLYGON((19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4 47.58, 19.1 47.5),(18.66 47.22,18.6 47.24, 18.55 47.18,18.66 47.22))', 4326));`
   → True
   `SELECT ST_IsPolygonCCW(ST_GeomFromText('POLYGON((19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4 47.58, 19.1 47.5),(18.66 47.22,18.6 47.24, 18.55 47.18,18.66 47.22))', 4326));` → False
7. `SELECT ST_IsClosed(ST_GeomFromText('LINESTRING(19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4 47.58, 19.1 47.5)', 4326));` → True
8. `SELECT ST_IsSimple(ST_GeomFromText('LINESTRING(19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4 47.58, 19.1 47.5)', 4326));` → Nincs önmetszése, tehát True.
9. `SELECT ST_IsRing(ST_GeomFromText('LINESTRING(19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4 47.58, 19.1 47.5)', 4326));` → True
10. `SELECT ST_IsRing(ST_GeomFromText('LINESTRING(19.1 47.5, 18.4 47.2, 18.05 46.9, 18.4 47.58)', 4326));` → False
11. `select ST_X(tp_geom), ST_Y(tp_geom) from telep_pont;`
12. `select ST_Centroid(t_geom), ST_Envelope(t_geom), ST_ConvexHull(t_geom) from telepulesek;`
13. `select ST_Area(ST_ConvexHull(t_geom)) as Burokter, ST_Area(t_geom) as telepulester from telepulesek;`
14. `select ST_Buffer(p_geom,1000) from patak where p_name ilike 'Dera%'`
15. `select ST_Buffer(p_geom,1000, 'join=round endcap=flat') from patak where p_name ilike 'Dera%'`
16. `select ST_Buffer(p_geom,1000, 'join=round endcap=flat side=right') from patak where p_name ilike 'Dera%'`

4. rész: Térbeli kapcsolatokat elemző, és geometriát készítő függvények

1. `select ST_Intersects((select ff_geom from folyo_felulet where ff_name='Duna'),(select ff_geom from folyo_felulet where ff_name='Tisza'))`
2. `select ST_Disjoint((select ff_geom from folyo_felulet where ff_name='Duna'),(select ff_geom from folyo_felulet where ff_name='Tisza'))`

3. `select ST_Intersects(ST_SetSRID(ST_Point(616100, 206300),23700), (select m_geom from megyek where m_name='Fejér megye'))`

   `select ST_Intersects(ST_GeomFromText('POINT(616100 206300)',23700), (select m_geom from megyek where m_name='Fejér megye'))`

4. `select ST_Intersects(ST_Transform(ST_SetSRID(ST_Point(18.6, 47.2),4326),23700), (select m_geom from megyek where m_name='Fejér megye'))`

   `select ST_Intersects(ST_Transform(ST_GeomFromText('POINT(18.6 47.2)',4326),23700), (select m_geom from megyek where m_name='Fejér megye'))`

5. `select ST_Touches((select m_geom from megyek where m_name='Fejér megye'),(select m_geom from megyek where m_name='Pest megye'))`

   Cseréljük ki és futassuk a feladatban felsorolt térbeli relációkra is!

6. `select ST_Contains((select m_geom from megyek where m_name='Fejér megye'),(select b_geom from belterulet b  where b_name='Székesfehérvár'))`

   `select ST_Within((select b_geom from belterulet b  where b_name='Székesfehérvár'),(select m_geom from megyek where m_name='Fejér megye'))`

7. `select ST_Crosses((select b_geom from belterulet b  where b_name='Székesfehérvár'),(select m_geom from megyek where m_name='Fejér megye'))`
   `select ST_Overlaps((select b_geom from belterulet b  where b_name='Székesfehérvár'),(select m_geom from megyek where m_name='Fejér megye'))`

8. `select ST_Within((select ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))')),(select ST_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True`
   `select ST_CoveredBy((select ST_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))')),(select ST_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) →True`

   `select ST_Within((select ST_GeomFromText('LINESTRING(0 0,0 1,1 1,1 0)')),(select ST_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → False`

   `select ST_CoveredBy((select ST_GeomFromText('LINESTRING(0 0,0 1,1 1,1 0)')),(select ST_GeomFromText ('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True`

   `select ST_Within((select ST_GeomFromText('LINESTRING(0 0,0.5 0.5,0 1,1 1,1 0)')),(select ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True`

```
select ST_CoveredBy((select ST_GeomFromText('LINESTRING(0 0,0.5 0.5,0 1,1
1,1 0)')),(select ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))')))
→  True
select ST_Within((select ST_GeomFromText('POINT(0 0)')),(select
ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → False
select ST_CoveredBy((select ST_GeomFromText('POINT(0 0)')),(select
ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))) → True
```

Az ST_COVERS-hez és az ST_CONTAINS-hez fel kell cserélni a két geometriát!!!

9. `select ST_Intersection((select m_geom from megyek where m_name='Fejér megye'),(select ff_geom from folyo_felulet where ff_name='Duna'))`
10. `select ST_Difference(np_geom,(select m_geom from megyek where m_name='Veszprém megye')) from np_tk`
11. `select b_name from belterulet where ST_Intersects(b_geom,(select m_geom from megyek where m_name='Fejér megye'))`
12. `select b_name, ST_Intersection(b_geom,(select m_geom from megyek where m_name='Fejér megye') ) from belterulet where ST_Intersects(b_geom,(select m_geom from megyek where m_name='Fejér megye'))`
13. `select count(b_name) from belterulet where not ST_Contains(b_geom,(select m_geom from megyek where m_name='Fejér megye'))`
14. `select b_name, ST_Difference(b_geom,(select m_geom from megyek where m_name='Fejér megye')) from belterulet where not ST_Contains(b_geom,(select m_geom from megyek where m_name='Fejér megye'))`
15. `select ST_Intersection((select m_geom from megyek where m_name='Fejér megye'), ap_geom ) from autopalya where ap_ref ilike '%M7%' and ST_Intersects((select m_geom from megyek where m_name='Fejér megye'), ap_geom)`

    vagy ST_Unionnal egyszerűbben

    `select ST_Intersection((select m_geom from megyek where m_name='Fejér megye'), ST_Union(ap_geom)) from autopalya where ap_ref ilike '%M7%'`

    vagy elfogadható ez is de tartalmazza az üres (EMPTY/LINESTRING EMPTY) rekordokat is, ahol nincs metszés:

    `select ST_Intersection((select m_geom from megyek where m_name='Fejér megye'), ap_geom ) from autopalya where ap_ref ilike '%M7%'`
16. `select ST_Length(ST_Union(ST_Intersection((select m_geom from megyek where m_name='Fejér megye'), ap_geom )))/1000 from autopalya where ap_ref ilike '%M7%'`
17. `select ST_Length(ST_Union(ST_Intersection((select m_geom from megyek where m_name='Fejér megye'), ap_geom )))/1000 from autopalya where ap_ref ilike '%M7%' and ap_hu_ed_dire='forward'`

    `select ST_Length(ST_Union(ST_Intersection((select m_geom from megyek where m_name='Fejér megye'), ap_geom )))/1000 from autopalya where ap_ref ilike '%M7%' and ap_hu_ed_dire='backward'`
18. `select ST_Area(ST_Union(ST_Intersection((select m_geom from megyek where m_name='Veszprém megye'),np_geom)))/1000000 from np_tk`
19. `select ST_Area(ST_Union(ST_Intersection((select m_geom from megyek where m_name='Veszprém megye'),np_geom)))/1000000/(select ST_Area(m_geom)/1000000 from megyek where m_name='Veszprém megye')*100 from np_tk`
20. `select  ST_Area(ST_Union(np_geom))/1000000/(select ST_Area(ST_Union(m_geom))/1000000 from megyek )*100 from np_tk`
21. `select * from geolada_eov where ST_Contains((select m_geom from megyek where m_name='Fejér megye'), gl_geom)`

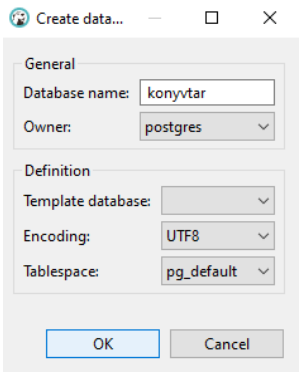
22. `select count(*) from geolada_eov where ST_Contains((select m_geom from megyek where m_name='Fejér megye'), gl_geom)`
23. `select * from geolada_eov where ST_Disjoint((select m_geom from megyek where m_name='Fejér megye'), gl_geom)`
    `select count(*) from geolada_eov where not ST_Contains((select m_geom from megyek where m_name='Fejér megye'), gl_geom)`

24. `select * from geolada_eov where ST_Intersects((select ST_Buffer(ff_geom, 10000) from folyo_felulet where ff_name='Duna'), gl_geom)`
25. `select * from geolada_eov where ST_Intersects((select ST_Union(ST_Buffer(ap_geom, 10000)) from autopalya where ap_ref ilike '%M7%'), gl_geom)`
26. `select * from geolada_eov where ST_Intersects(ST_Intersection((select ST_Union(ST_Buffer(ap_geom, 10000)) from autopalya where ap_ref ilike '%M7%'),(select m_geom from megyek where m_name='Fejér megye')), gl_geom)`
27. `select p_name, ST_Distance((select ST_Transform(va_geom,23700) from var where va_name='Hollókői vár'), p_geom) from patak order by 2 asc limit 1`
28. `select ST_Endpoint(ST_ShortestLine((select ST_Transform(va_geom,23700) from var where va_name='Hollókői vár'), p_geom)) from patak order by 1 asc limit 1`
29. `select ST_Split(ST_Union(m_geom), ST_Transform(ST_GeomFromText('LINESTRING(16.0 47.0, 22.0 47.0)',4326),23700)) from megyek`
30. `select ST_SymDifference((select ST_Union(np_geom) from np_tk where np_name='Budai Tájvédelmi Körzet' ),(select m_geom from megyek where m_name= 'Budapest'))`
31. `select j_name, va_name from jarasok, var where ST_Contains(j_geom, ST_Transform(va_geom,23700)) order by j_name`
    vagy
    `select j_name, va_name from jarasok, var where j_geom ~ ST_Transform(va_geom,23700) and ST_Contains (j_geom, ST_Transform(va_geom,23700)) order by j_name`
32. **Rajzoljuk ki a 8-as Út azon szakaszát, amely a székesfehérvári Bory-vár és a Várpalotai Thury vár között található. A vágásnál ügyeljen arra, hogy az út várhoz legközelebb eső pontjánál érjen véget a szakasz.**
33. `select SRS1.srid, SRS2.srid from spatial_ref_sys SRS1, spatial_ref_sys SRS2 where (SRS1.srid+SRS2.srid)=23700`
    végén elosztjuk kettővel, mert ismétli önmagát
34. `select j2.j_name from jarasok j1, jarasok j2 where ST_Touches(j1.j_geom, j2.j_geom) and j1.j_name='Székesfehérvári járás'`
35. `select j1.j_name, count(*), string_agg(j2.j_name,', ') from jarasok j1, jarasok j2 where st_touches(j1.j_geom, j2.j_geom) group by 1 order by 1`

7. FEJEZET: Adatimport, adatfeltöltés, adatmódosítás

Új adatbázis létrehozása és meglévő SQL szkript futtatása DBeaver-en

Jobb egérgombbal kattintás a *Database*-en, és létrehozzuk az új üres adatbázist, pl. `konyvtar` néven (`Create → Database`).



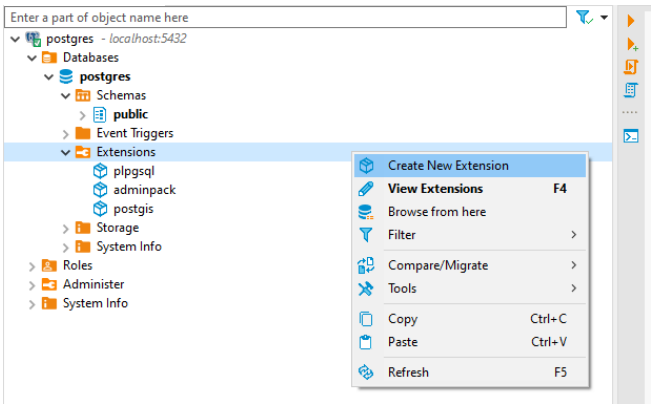
Nyissuk meg a `dump-KONYVTAR.sql` fájlt. *SQL editor* menü → *Open SQL script*.

Futtassuk le a szerkesztőben megnyitott szkriptfájlt → *Execute script* (3. ikon a szerkesztőablak bal oldalán) → F5: frissítés a könyvtár adatbázison.

Vigyázat a MySQL-ből exportált SQL szkript nem kompatibilis a PostgreSQL-lel, ilyen esetben célszerű lehet az adatokat CSV-ben kimenteni, majd a CSV-t importálni.

Kiegészítők hozzáadása (PostGIS, PostGIS Raster, PostGIS Topology)

Térbeli adatok kezeléséhez kiegészítőre van szükségünk, mivel az alap PostgresSQL nem ismeri fel a térbeli geometriákat. A hozzáadáshoz lenyitjuk a kiválasztott adatbázist, megkeressük az Extensions-t. Itt jobb egérgombbal kattintunk → *Create New Extension* → végül kikeressük a kívánt kiegészítőt/bővítményt, pl. PostGIS.

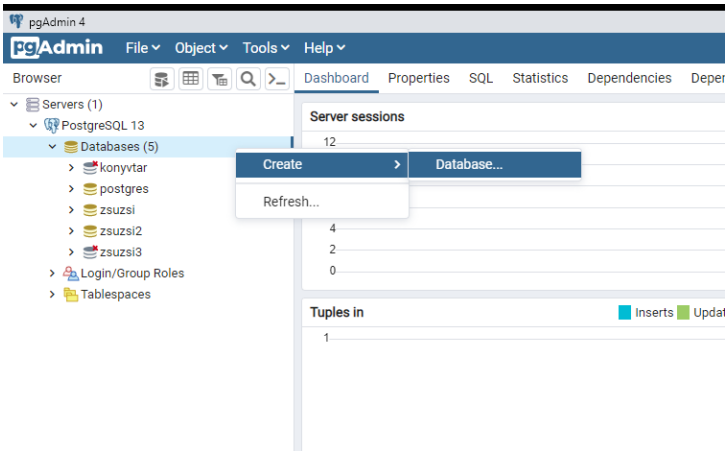


Nagyméretű adatbázisok importálása szkriptből (pgAdmin4 felületen)

Alapesetben a DBeaver-ben az előbbi fejezetben bemutatott módon tudjuk importálni SQL szkriptből az adatokat. De sajnos előfordulhat (a jegyzet készítése idején fenálló hibáról van szó), hogy a túl nagy méretű (kb. 30-50 MB+) szkriptfájlokat nem tudja lefuttatni a DBeaver, és összeomlik. Erre jelen pillanatban az alábbiakban vázolt megoldás működik. Sajnos, ehhez vissza kell menni a pgAdmin felületre, mivel az sokkal robosztusabb, stabilabb, meg fog birkózni a nagyméretű szöveges SQL fájlokkal is.

A Bejelentkezés pgAdminba ugyanazzal a felhasználónévvel és jelszóval működik, mint amit DBeavernél megadtunk.

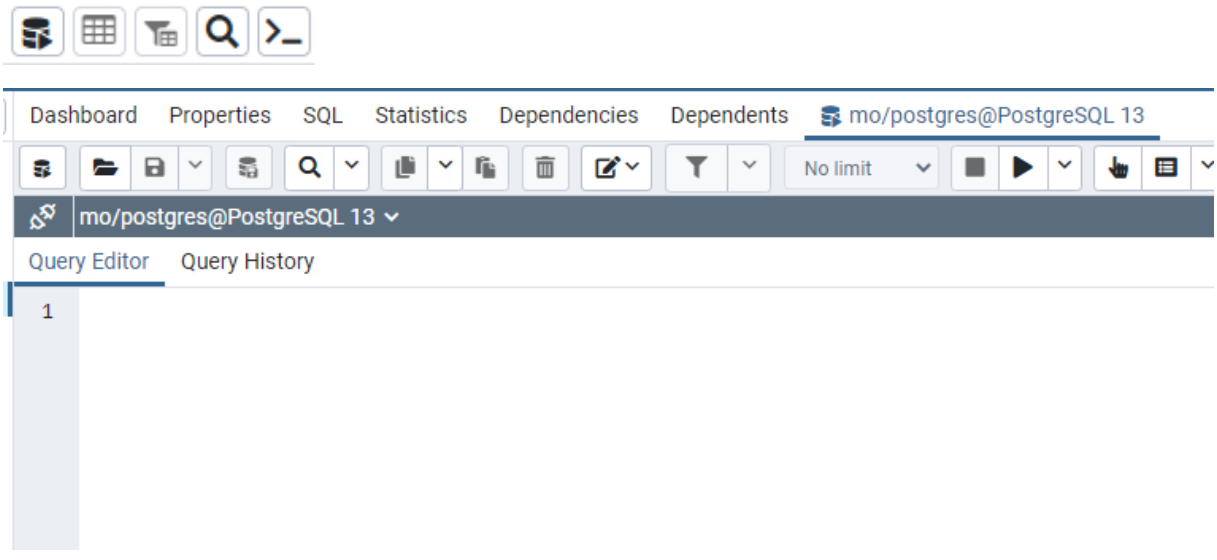
Jobb egérgombbal kattintás a *Database*-en→ *Create* → *Database*



Megadjuk az adatbázis nevét, pl. `mo`.

Majd itt is hozzáadjuk a térbeli adatokhoz tartozó adatbázishoz a PostGIS-t. Jobb egérgombbal kattintás az *Extension*ön, majd *Create*. Kiválasztjuk a PostGIS-t.

Majd megnyitjuk a *Query Tool*t (Első ikon, hordó lejátszás gombbal) vagy *Tools* menü→ *Query Tool*.

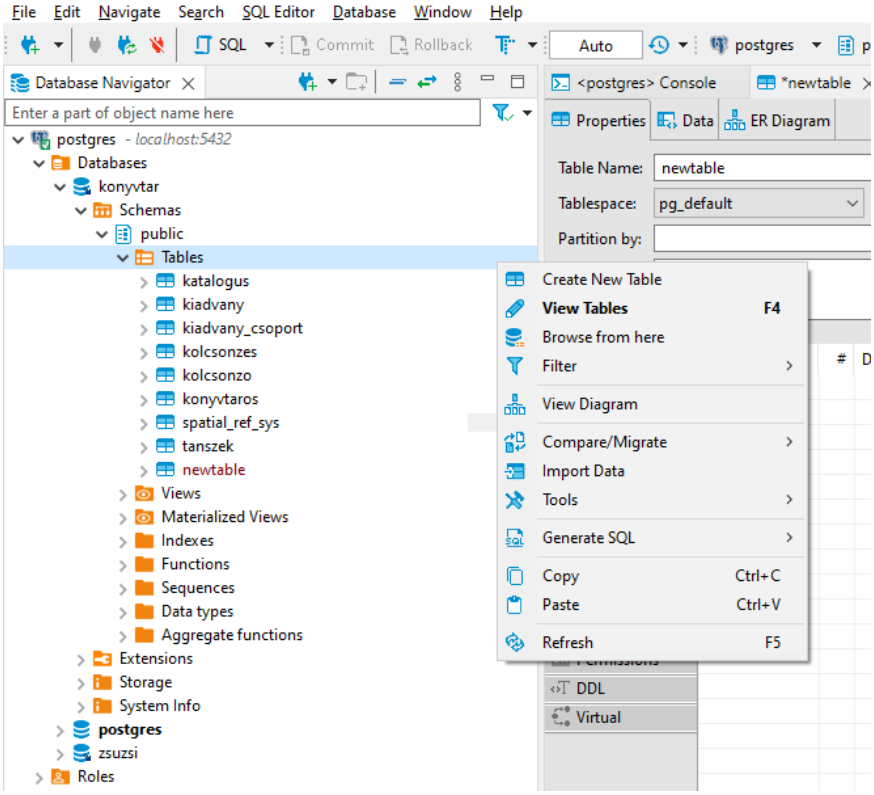


Megnyitjuk a szkriptet a mappa ikonnal, majd lejátszás gombbal lefuttatjuk (kb. 20-30 másodpercig futni fog). Ha a sikeres a lekérdezés, Refresh-sel frissíthető az adatbázis, és ekkor látnunk kell az új táblákat.

Adatfeltöltés (import) és táblák létrehozása CSV fájlokból

Ebben a gyakorlatban CSV fájlokat importálunk PosgreSQL adatbázisba DBeaver felületen, méghozzá úgy, hogy mi magunk készítjük elő a táblaszerkezetet.

Első lépésként hozzunk létre egy új táblát. Nyissuk le az adatbázis a bal oldali *Database Navigator*ban és jobb egérgombbal kattintsunk a *Table*s ponton→ *Create New Table*.

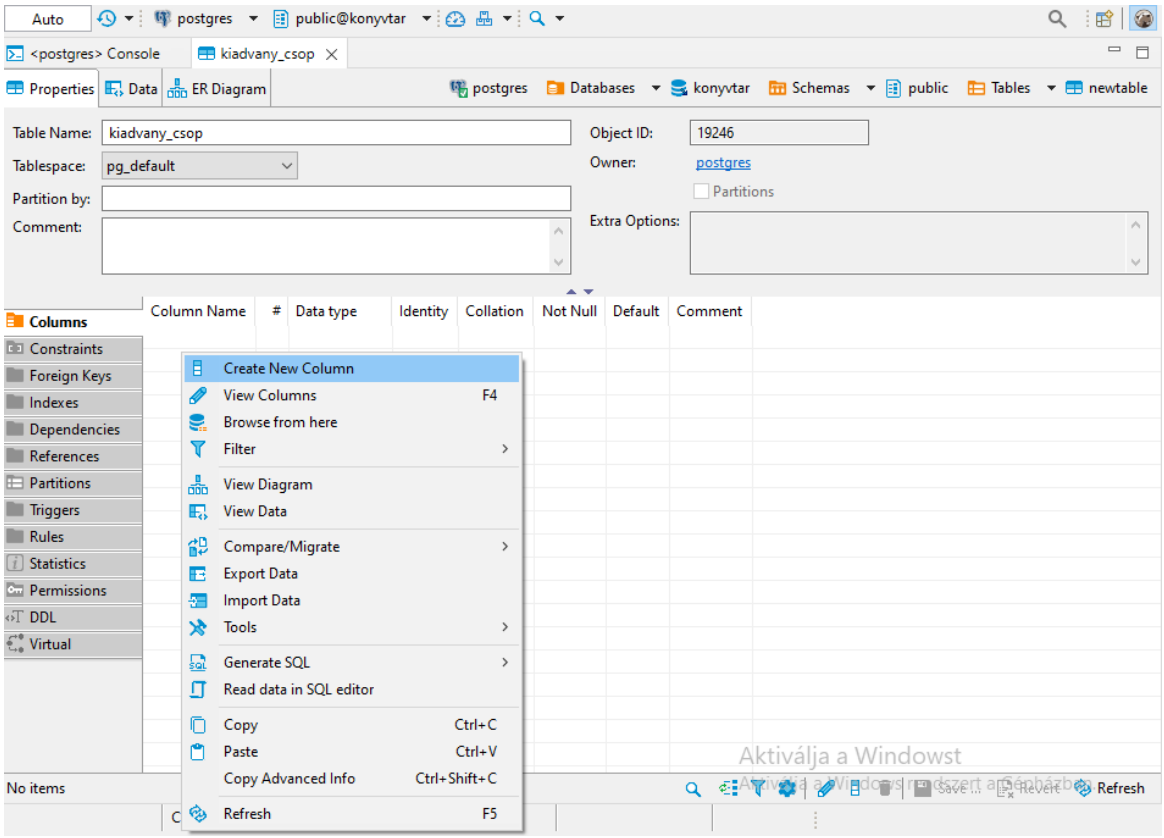


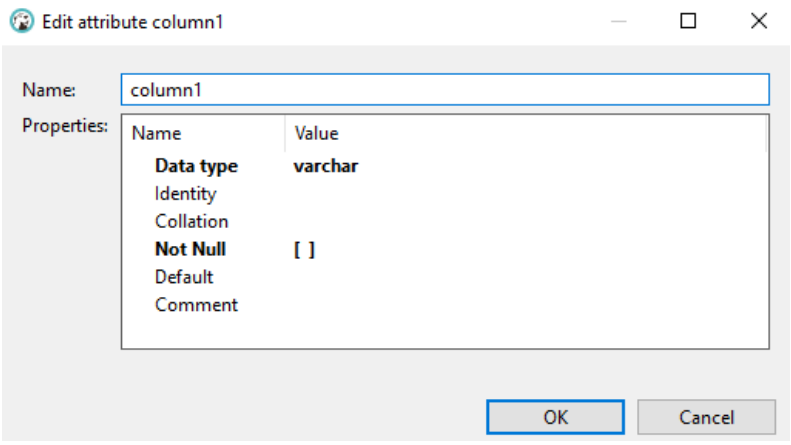
Adjuk meg a jobb oldali munkaablakban a tábla nevét a Properties fülön. *SAVE*. (A Save parancsot a munkaablak alján keressük egy floppylemez ikon mellett.)

Majd adjunk hozzá a táblához új mezőket: középen a *Columns* → jobb egérgombbal kattintás → *Create New Column*.

Adjuk meg a mező nevét, az adattípust, collation-t (a karaktertábla), *Not Null []* → ha üres, `Null` értékkel helyettesíti. ha *Not Null [v]* → ha üres a mező, nem tesz bele `Null` értéket, hanem üresen hagyja.

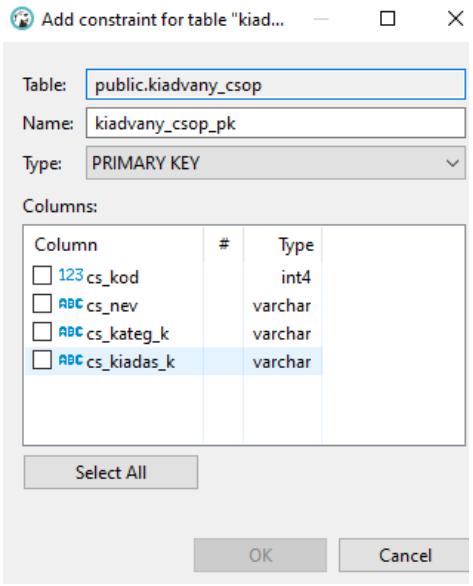
Ha készen vannak a mezők, a táblázatban zölddel jelöli az el nem mentett oszlopokat. *SAVE*.





A mezőkhöz be lehet állítani indexeket, kényszereket, kulcsokat.

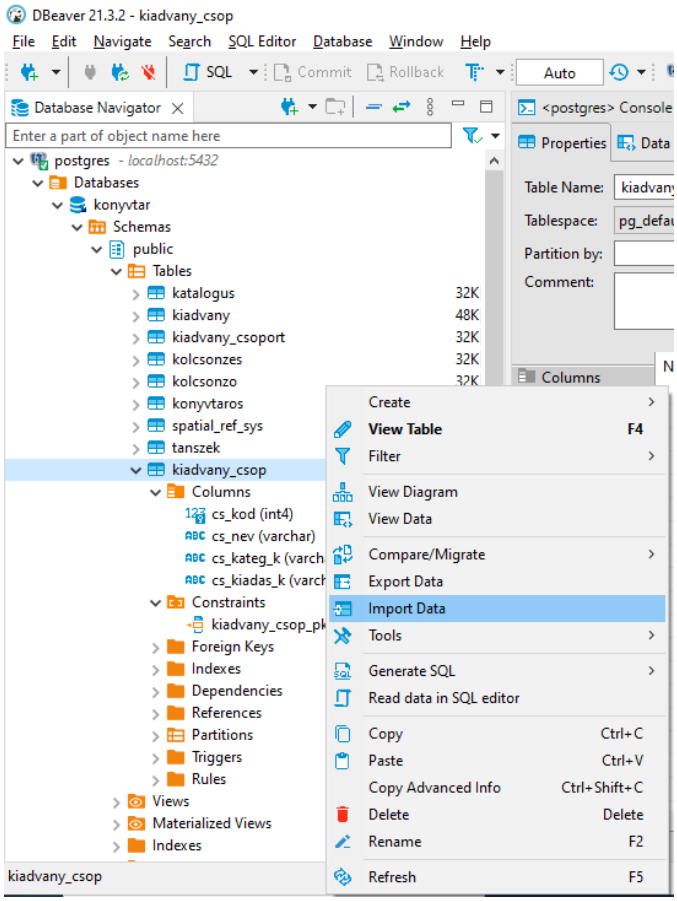
Állítsuk be az elsődleges kulcs mezőt (egyedi azonosító, nem lehet üres a cella.) és ahol szükséges a unique keyt (egyedi, nem ismétlődő adat, de nem elsődleges kulcs. A cella lehet üres is.).



További fontosabb beállításokat is a tábla fülön lehet megtenni: foreign key (idegen kulcs) és indexes, triggers stb.

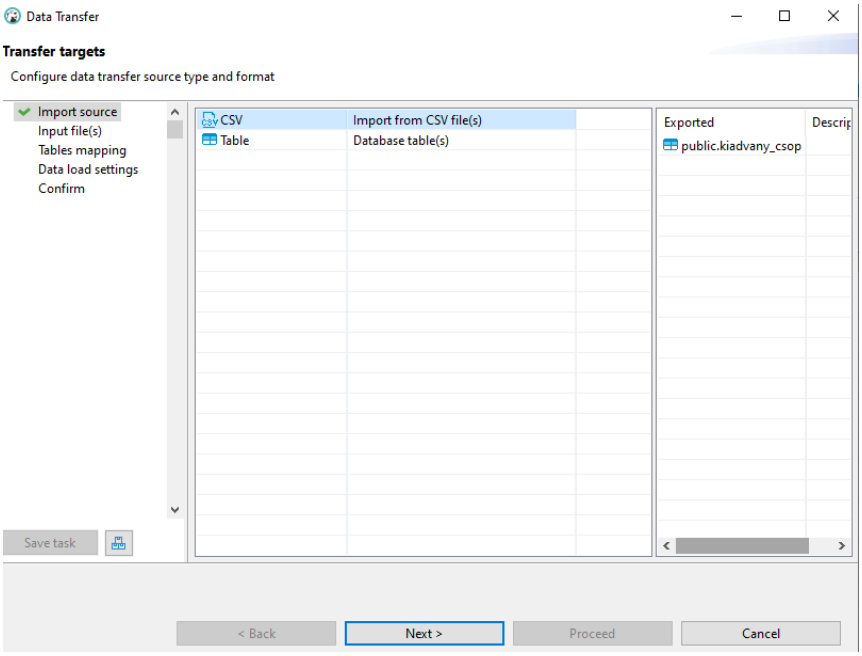
Minden módosítás után mentsünk (*SAVE*).

Ha kész a táblaszerkezet és van egy adattáblánk pl. Excel-ben/CSV-ben, akkor importálhatjuk az adattáblát.



Jobb egérgombbal kattintunk a tábla nevén a *Database Navigator*ban → *Import data*.

CSV files adattípus kiválasztása. *Next*.



Adatforrás kiválasztása. Utána állítsuk be a CSV fájl jellemzőit.

Extension: kiterjesztés (CSV, TSV,TXT)

Encoding: karakterkódolás

Column delimiter: oszlopokat elválasztó karakter. Ált. ; vagy , vagy TAB

Header position: van-e fejléc, ha igen hány sort tesz ki.

Quote char: van-e az oszlopokat határoló karakter, és mi az pl. ”

Escape char: ált. sorvége: \

Null value mark: mi jelöli a `Null` értéket, pl. `Null` vagy „”

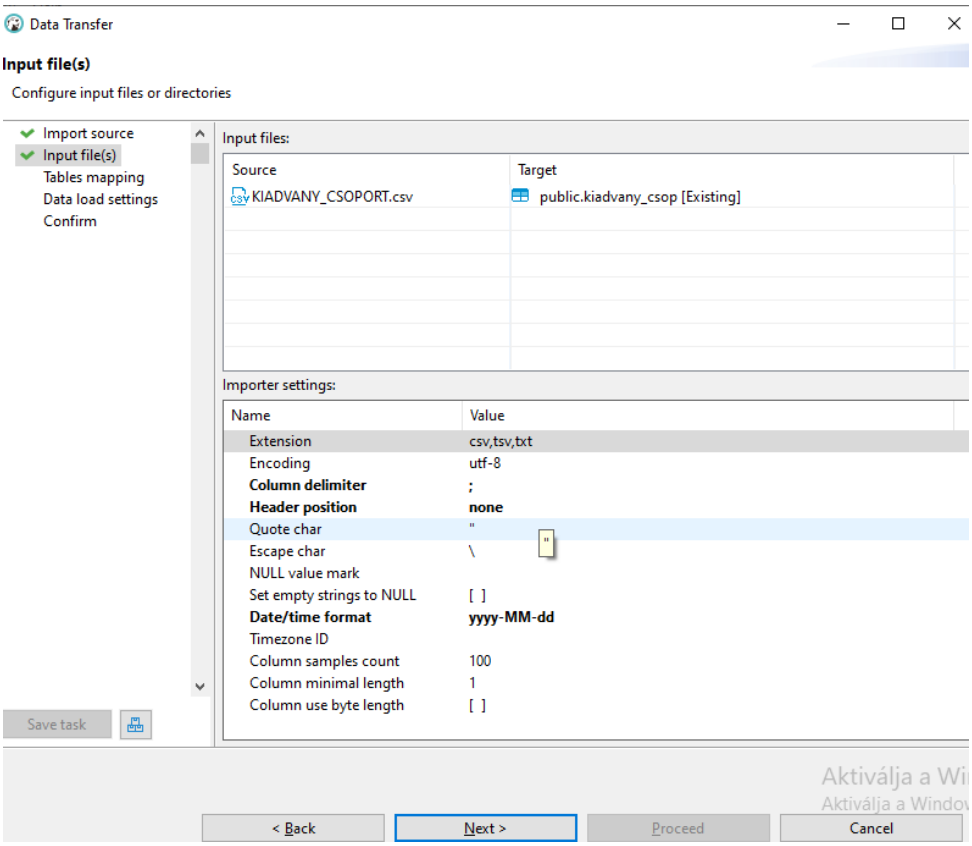
Set empty strings to `Null`: az üres mezőket `Null` értékre állítsa-e vagy sem.

Date Time format: Időformátum megadása.

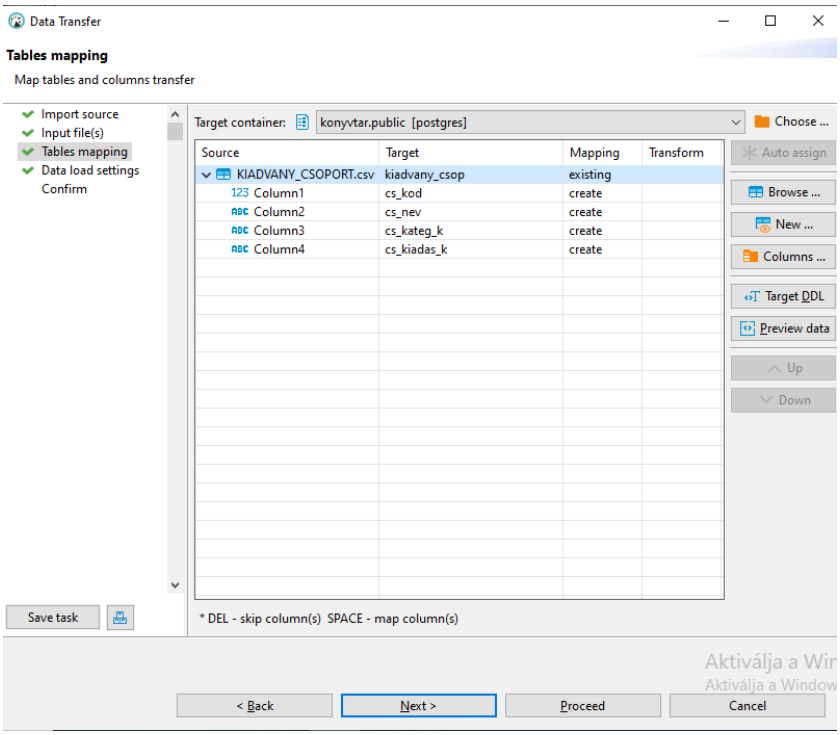
Timezone ID: (időzóna azonosító, ha releváns)

Column length-re: mezőkben lévő adatok hosszára vonatkozó beállítások

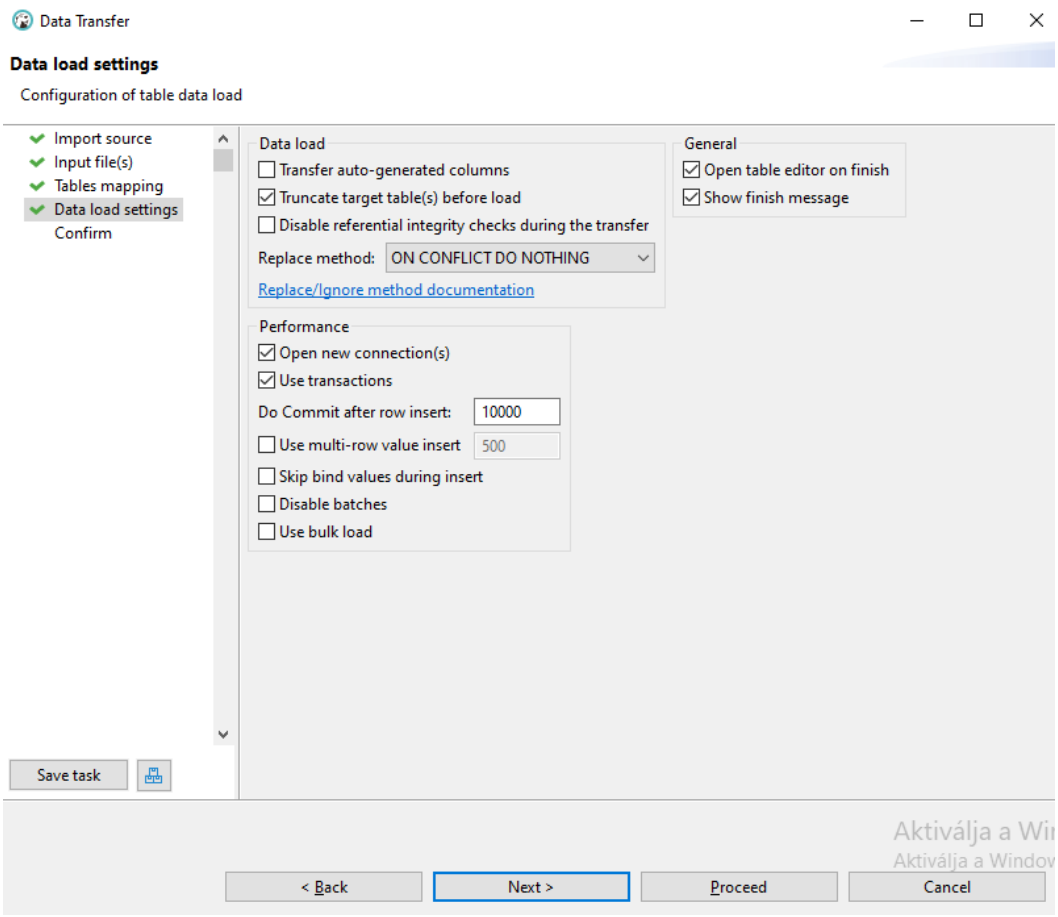
Next.



A következő ablakban vethető össze az importálandó állomány és az adattáblánk szerkezete. Ha van hiányzó oszlop, itt még módosíthatunk.



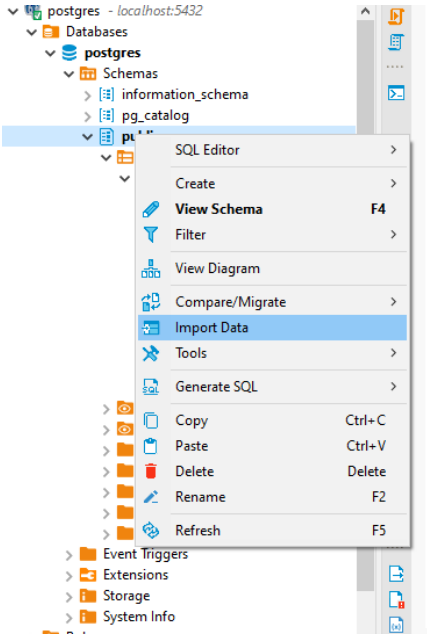
Végső beállítások. Amire felhívnám a figyelmet, a *Truncate target table(s) before load*. Ha ki van jelölve, akkor mielőtt elkezdené az importálást, kiüríti a tábla tartalmát. Időnként kívánatos lehet használni. *Next*, végül *Proceed*.



A fülön mindig ellenőrizzük az import eredményét! Ha nem látjuk elsőre, frissítsünk (F5 vagy jobb alsó sarokban refresh).

Adatfeltöltés (import) és tábla létrehozása nélkül

Ebben a feladatban adott egy CSV-t pl. Európa országai. Importáljuk a fájlt anélkül, hogy előzetesen létrehoznánk neki a táblát és a táblaszerkezetet.

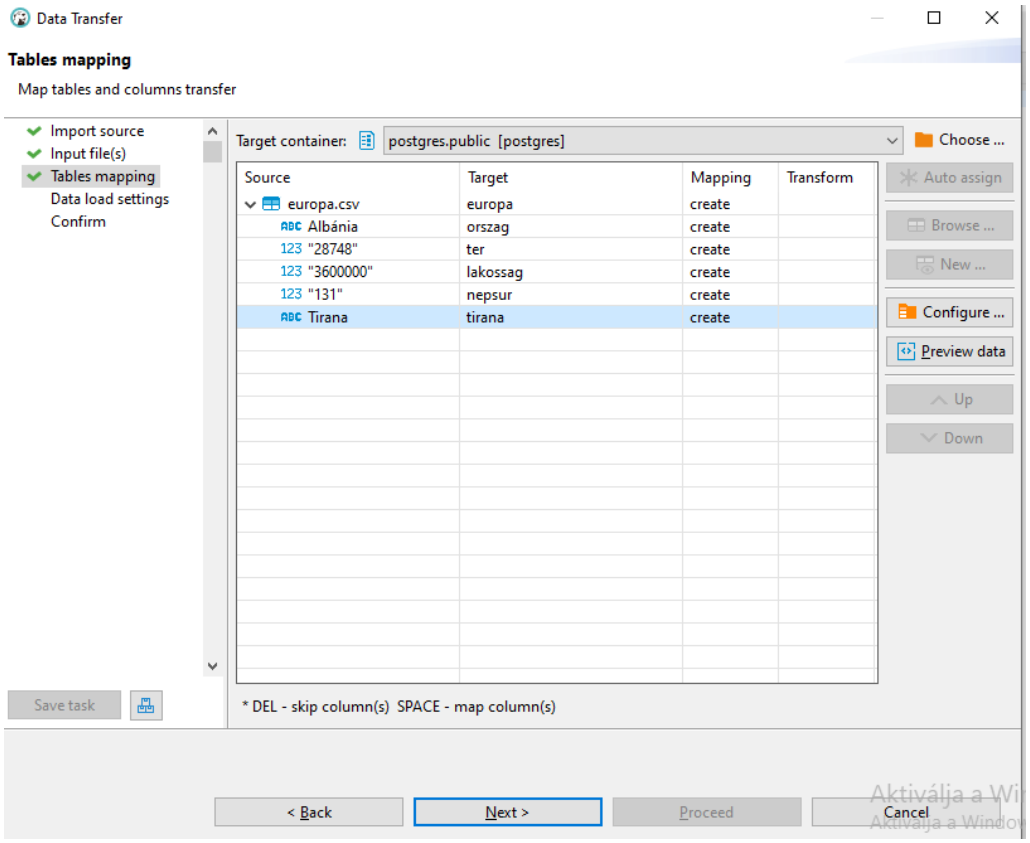


Az előbbi menüt használjuk. Jobb egérgombbal kattintás az adatbázis nevén vagy a sémán → *Import data*.

Kiválasztjuk a CSV-t, mint formátumot, illetve megadjuk a fájl forrását. Megadjuk a beolvasáshoz szükséges ismereteket, pl. elválasztó (ált. automatikusan megvizsgálja a fájlt, de érdemes ellenőrizni).

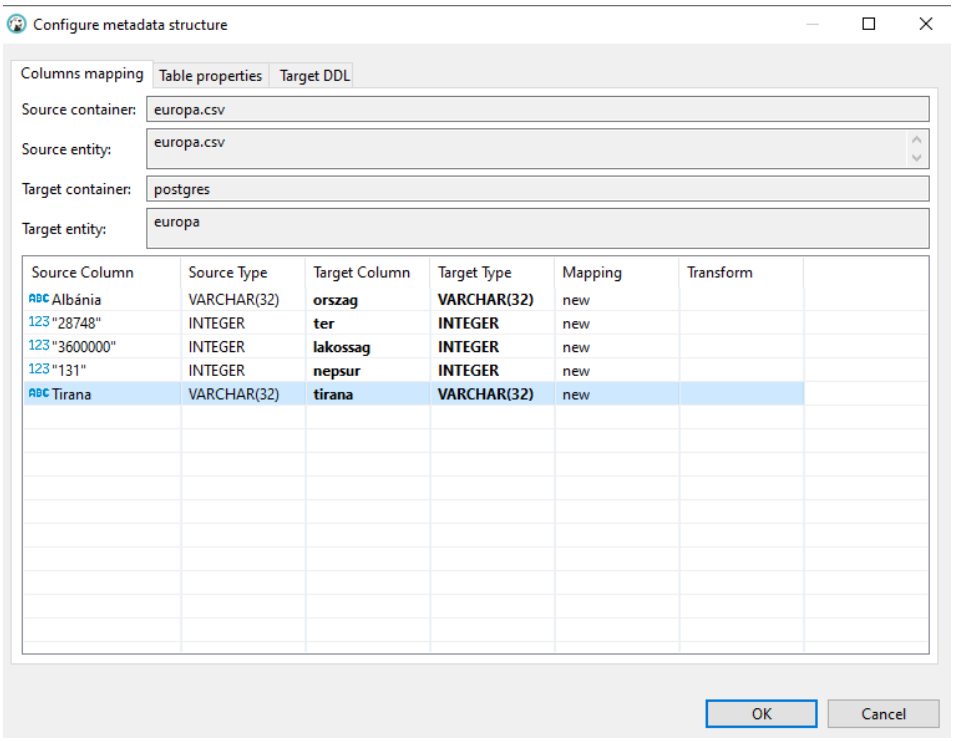
Mivel ez a CSV csak adatokat tartalmaz fejléc nélkül, ezért a következő menüben nekünk kell megadni a mezők neveit. Ugyanitt megjegyezném, hogy az adatbáziskezelők nem szeretik az ékezetes betűket, a szóközöket és a különleges karaktereket (pl.? ! %) a mezők neveiben. Így, ha ilyen előfordulna, még az import előtt írjuk át!

A *Mapping* oszlopban, ha a CSV-ből valamely mezőre nincs szükségünk az adatbázisban, azok ki is hagyhatók (*Skip*), vagy ha meglévő táblához adnánk hozzá a mezőket, akkor újra is generálhatók.



Ugyanitt érdemes a *Configure*-t is megnyitni. Itt lehet ellenőrizni és beállítani, hogy az egyes mezőknek mi lesz adattípusa.

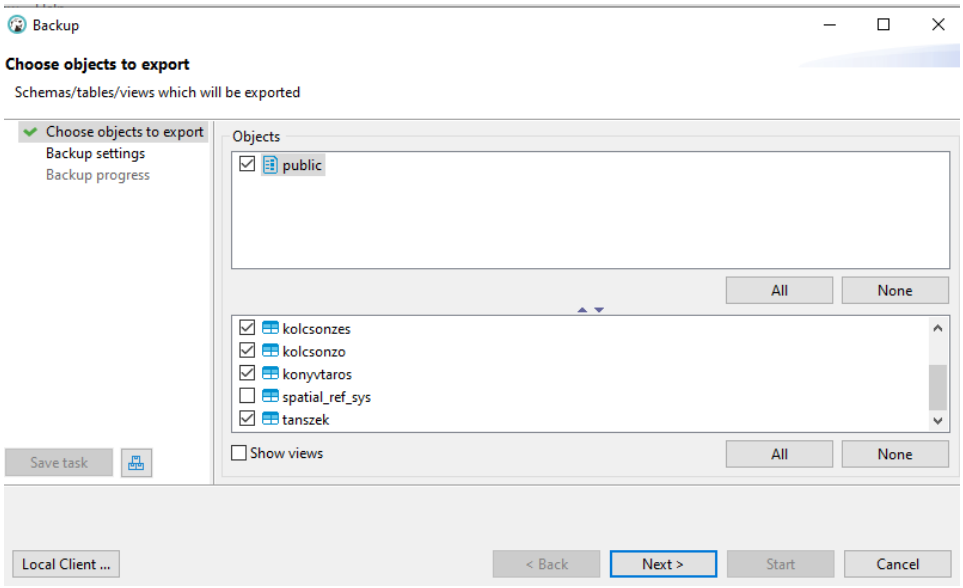
Ezek után már csak a további beállítások elfogadása van hátra. Frissítés után megjelenik az így behívott tábla az adatbázisban, ezt ellenőrizzük!



Adatbázisok biztonsági mentése (backup) és újra importálása

Jobb egérgombbal kattintás a kiválasztott adatbázison → *Tools* → *Backup*

Kiválasztjuk az exportálandó táblákat.



Ajánlott beállítások:

- *Formátum: Plain* (szöveges), ez könnyen szerkeszthető később is egy szöveg/kódszerkesztővel.
- *Encoding*: UTF8
- *Use SQL INSERT instead of copy of rows*: ne másolja, hanem egy újként szúrja be az táblákat

- *Output folder*: célmappa kiválasztása.

→ *START*.

Visszaállítás

Legyen egy adatbázis létrehozva pl. `konyvtar`. Nyissuk meg az sql szkriptet/vagy másoljuk be az *SQL Editor*ba. *Execute SQL script* gombbal indíthatjuk el. Ha lefut hiba nélkül, akkor frissítsünk. (Refresh).

Ha túl nagy a szkriptfájl, akkor érdemes pgAdmin felületen importálni a táblákat, mert egyelőre túl nagyméretű szkriptekkel (20MB+) nem biztos, hogy megbirkózik a DBeaver.

Üres tábla feltöltése másik táblából

Hozzunk létre egy ures nevű táblát! Ennek legyen egy `u_id serial`, egy `u_name varchar` és egy `u_geom geometry` mezője. Az elsődleges kulcs (primary key) az `u_id`.

Töltsük fel az ures táblát másik táblából lekérdezéssel! A feladat megoldható DBeaveren az ebben a fejezetben bemutatott módszerrel (Tábla létrehozása és szerkezetének kialakítása), vagy egyszerűen lekérdezésekkel.

```
CREATE TABLE public.ures ();

ALTER TABLE public.ures ADD ur_id serial NOT NULL;
ALTER TABLE public.ures ADD ur_name varchar NULL;
ALTER TABLE public.ures ADD ur_geom public.geometry NULL;
```

Szúrjunk be adatokat a geolada_eov táblából az új, ures táblába, de a csak Fejér megyei geoládákat. Az alábbi INSERT INTO + SELECT SQL parancs szolgál arra, hogy meglévő táblából leválogassunk adatot, és feltöltsünk egy másikat adatokkal.

```
insert into ures (ur_name, ur_geom) select gl_full_name, gl_geom from geolada_eov
where ST_Intersects(gl_geom, (select m_geom from megyek where m_name='Fejér
megye'))
```

Adatkonverzió

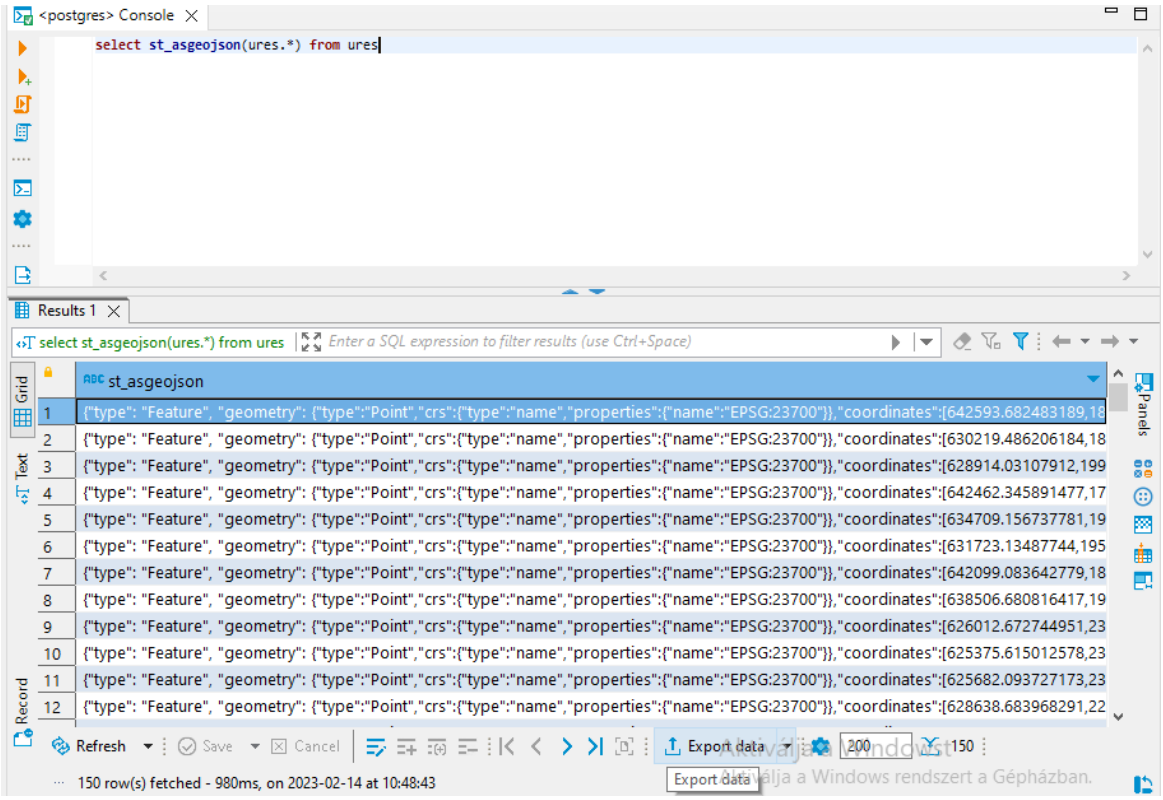
Készítsünk geojson geometriát az előbbi példában létrehozott táblából. Erre szolgál az `ST_AsGeoJSON` függvény. Ezt kétféleképpen tudjuk használni: egyszer a geometriát tudjuk átalakítani, a másik esetben pedig a teljes táblát átalakítom GeoJSON formátummá.

```
select ur_name, ST_AsGeoJSON (ur_geom) from ures
```

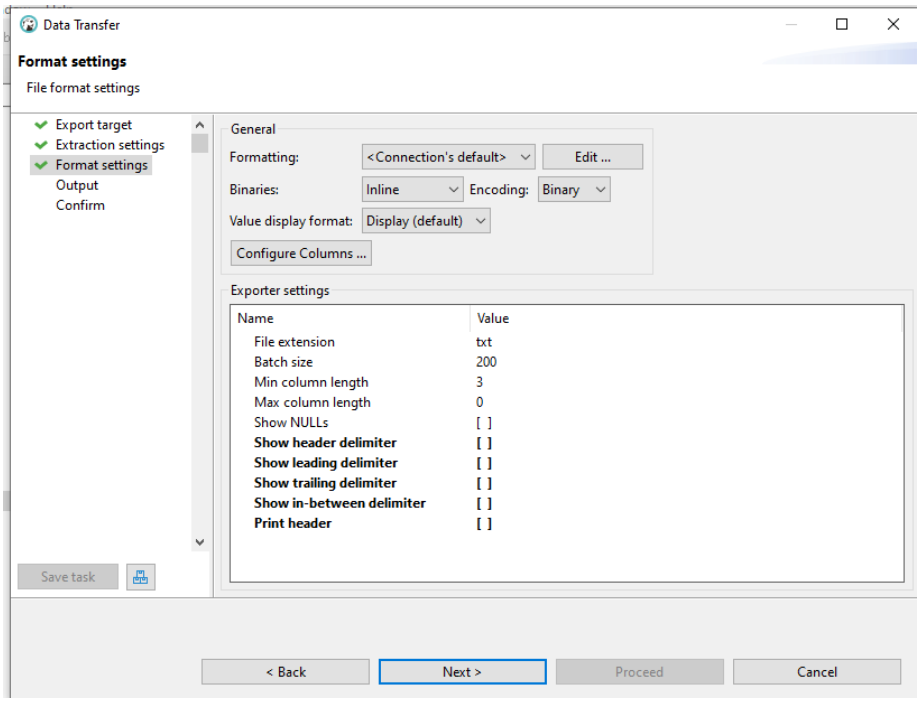
Készítsük el a rekord GeoJSON leírását:

```
select ST_AsGeoJSON(ures.*) from ures
```

A fenti táblát exportáljuk fájlba. Az eredménytábla alatt kattintsunk a tálca alakú jelre egy nyíllal.

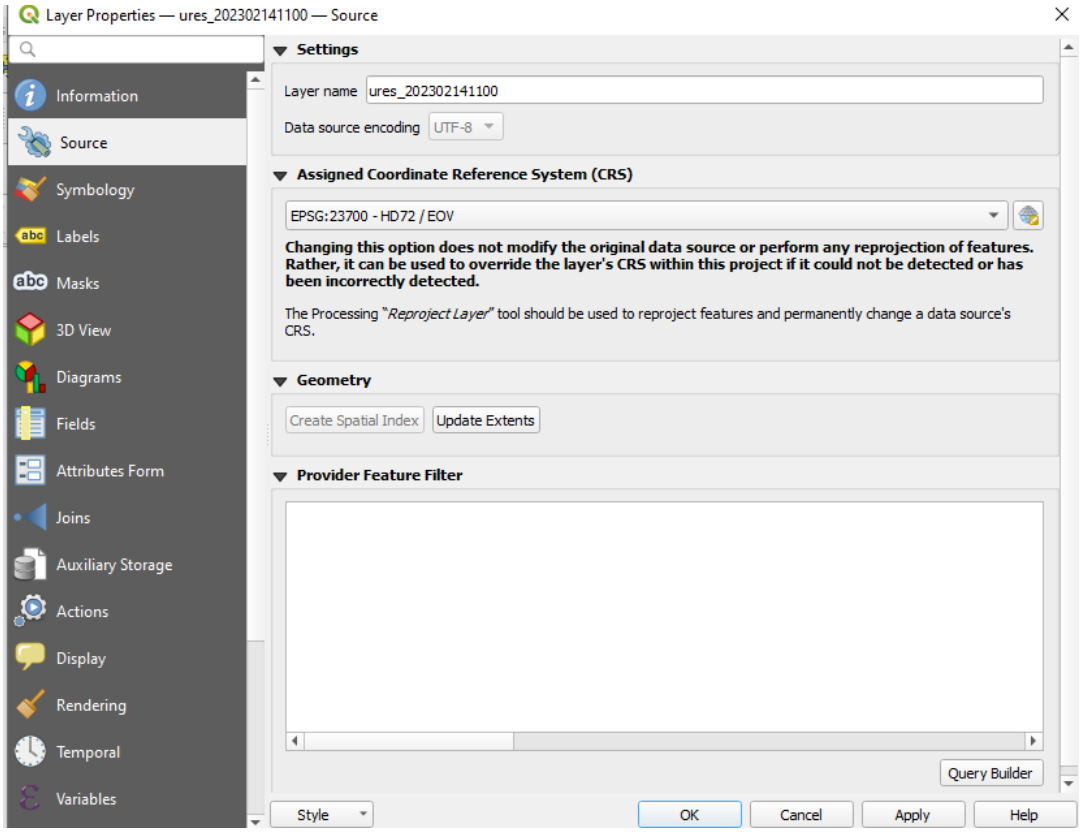


Itt a megnyíló párbeszédpanelen célszerű a TXT formátumot választani. Az exportáláskor beállíthatjuk, hogy legyenek-e a fájlban elválasztó karakterek. Ezeket kapcsoljuk ki. (Az alábbi képen az összes kiemelt sor). A többi menüpontban az alapbeállításokkal exportáltam. Az exportálás után a TXT formátumot átírtam GeoJSON-re.




```
{"type": "Feature", "geometry":
{"type":"Point","crs":{"type":"name","properties":{"name":"EPSG:23700"}},"coordinates":[642593.682
483189,180088.555515703]}, "properties": {"ur_id": 1, "ur_name": "Dunaújvárosi Madár Liget"}}
{"type": "Feature", "geometry":
{"type":"Point","crs":{"type":"name","properties":{"name":"EPSG:23700"}},"coordinates":[630219.486
206184,189545.599921337]}, "properties": {"ur id": 2, "ur name": "Kastélykutató"}}
```

Ezt a GeoJSON fájlt importáljuk QGIS-be. Vigyázzunk a vetülettel, ha nem adjuk meg, automatikusan 4326-ost vár a GeoJSON-től. Itt egyébként 23700. A fájl behívása után a réteg beállításainál a *Source* fülön állítsuk át.



Felmerül az ötlet, hogyha ilyen jól működik az adattábla átalakítása GeoJSON-be, akkor biztosan menne KML-lé alakítás is. Sajnos, az ST_asKML függvény paraméterül csak a geometriát fogadja el, az ures.* paramétert nem. Tehát csak a geometriát tudjuk KML formátumúvá tenni, leíró adatokat egyelőre nem tudunk hozzárendelni.

```
select ST_AsKML(ur_geom)  from ures
```

A Geography adattípus

A geography adattípus nem síkkoordinátákkal dolgozik, hanem gömbi koordinátákkal, földrajzi szélességgel és hosszúsággal. Ha a geometriát akarjuk átalakítani geography típussá, akkor az `ST_Transform` függvényt használjuk, és képezzük földrajzi koordinátákat (egyébként a síkon) a 4326-os vetületben. A földrajzi és síkkoordináták között a legnagyobb különbségek a távolságok számításakor jönnek elő (`ST_Distance` és `ST_DistanceSphere`). Az alábbi cikk erre hívja fel a figyelmet:

http://postgis.net/workshops/postgis-intro/geography.html

Ha nem feltétlenül szükséges, érdemes kerülni a geography adattípus használatát, inkább akkor használjuk csak, ha a teljes Földra vonatkozóan vannak földrajzi koordinátáink.

Készítsünk új mezőt geography adattípussal az új ures táblába.

```
ALTER TABLE public.ures ADD ur_geog public.geography NULL;
```

Töltsük fel geography adattípussal az új mezőt.

```
UPDATE ures SET ur_geog  = ST_Transform(ur_geom,4326);
```

vagy

```
UPDATE ures SET ur_geog  = ST_Transform(ur_geom,4326)::geography;
```

Kérdezzünk le rá. Melyik Fejér megyei geoládák vannak a Velencei-tó 10 km-es körzetében?

```
select ur_name from ures where ST_Intersects(ur_geog,(select
ST_Transform(ST_Buffer(to_geom, 10000),4326) from tavak where to_name='Velencei-
tó'))
```

Nézzük példát a távolságok számítására. A Dunaújvárosi Madár Liget (`ur_id=1`) és Bakonykúti (`ur_id=28`) közötti távolságokat számítsuk ki, egyszer a geometrián, majd a geography földrajzi koordinátákon az `ST_Distance` függvénnyel.

Ha a geometrián kérdezzük le a távolságot az `ST_Distance`-cel, akkor a síkon (vetületi) távolságot kapjuk meg. Ha az `ST_Distance`-et a geography földrajzi koordinátákon hívjuk meg, akkor az ellipszoidi alapfelületen vett távolságot kapjuk meg (amit az SRID meghatároz, itt WGS84).

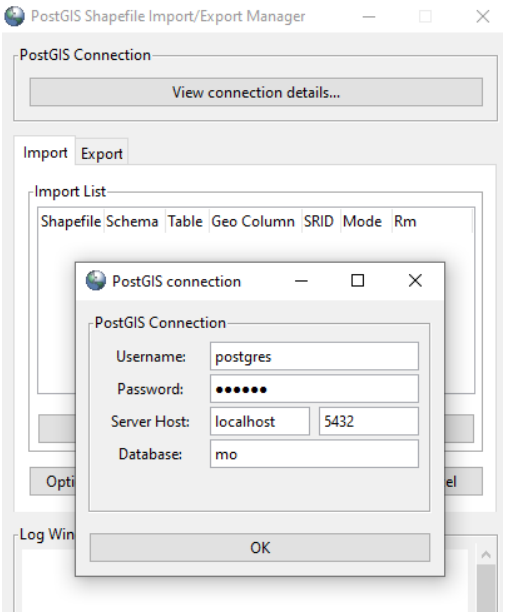
Ha az `ST_DistanceSphere`-t hívom meg a geometrián, (4326-os vetületbe transzformálva, mert az `ST_DistanceSphere` csak földrajzi koordinátákkal dolgozik), akkor a gömbi távolságot fogja visszaadni méterben. A gömb alakú Föld sugara r= 6 371 008 m.

```
select ST_Distance((select ur_geom from ures where ur_id=1),(select ur_geom from
ures where ur_id=28)),ST_Distance((select ur_geog from ures where ur_id=1),(select
ur_geog from ures where ur_id=28)) → 65 527.41 m és 65 531.6 m
```

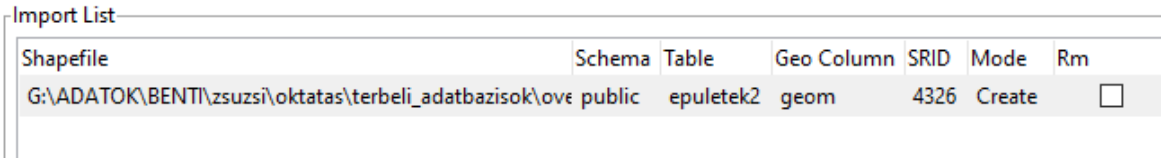
```
select ST_DistanceSphere((select ST_Transform(ur_geom,4326) from ures where
ur_id=1),(select ST_Transform(ur_geom,4326) from ures where
ur_id=28)),ST_Distance((select ur_geog from ures where ur_id=1),(select ur_geog
from ures where ur_id=28)) → 65 387.86 m és 65 531.6 m
```

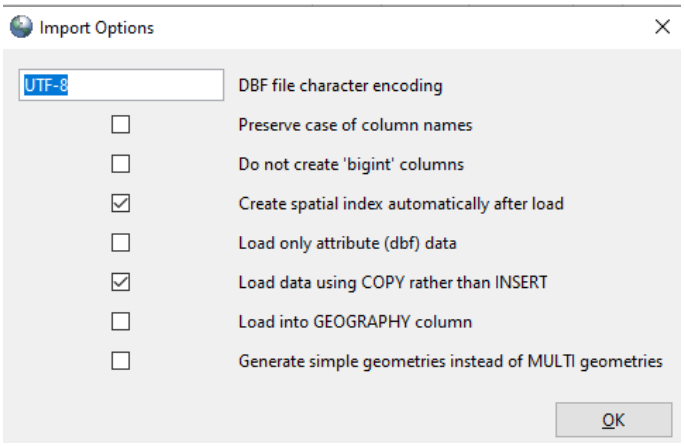
Shapefile-ok importja és exportja PostGIS Bundle-lel

Indítsuk el a *Start* menüből a PostGIS Bundle nevű programot (Windows). A program Shapefile-ok importálására és exportálására lett kitalálva PostgreSQL és PostGIS adatbáziskezelőhöz. Először adjuk meg a *View connection details* gombra kattintva az adatbázishoz való kapcsolódás adatait. Importáljunk az `epuletek.shp` fájlt.



Az *Add File*-nál válasszuk ki a Shapefile-t, a *Table* oszlopban a PostgreSQL-ben létrejövő tábla nevét tudjuk megadni. Ne felejtsük el az SRID-t is beállítani, mert automatikusan nem ismeri fel. Egyszerre több SHP fájl is importálható. Ha valamelyikre mégsincs szükségünk az *Rm* oszlopot kipipálva tudjuk elvetni a Shapefile-t. Megjegyezném, hogy az *Import List* alatt található egy *Options* gomb, ahol többek között a karakterkódolást, a térbeli indexek automatikus hozzáadását, vagy csak az attribútum táblázat importálását lehet beállítani.

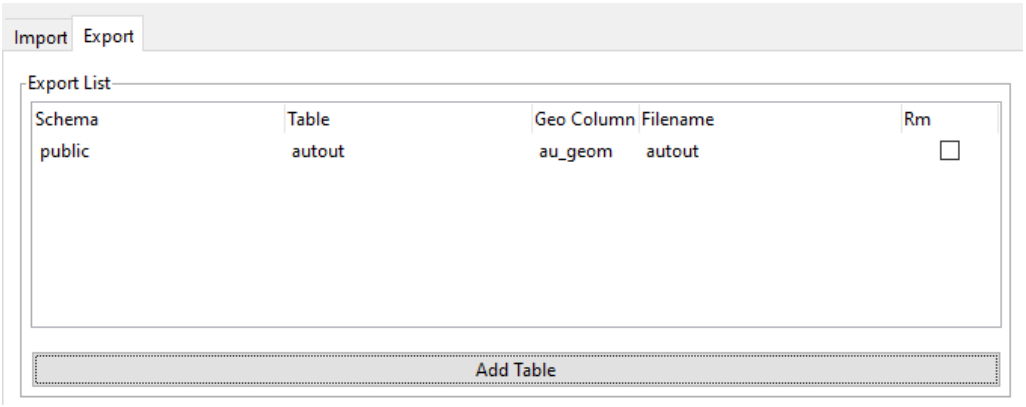




Ha a beállításokkal készen vagyunk, akkor az *Import* gombra kattintva megkezdhetjük a konvertálás folyamatát. Alul, a *Log Window*-ban nyerhetünk információt a folyamat sikerességéről, illetve az esetleges hibaüzenetek segíthetnek megfejteni a sikertelen importálás okait.

Szeretném felhívni a figyelmet, hogy nemcsak SHP fájl, hanem DBF (vagyis DBase formátumú adattábla is) importálható a PostGIS Bundle nevű programmal!

Export: Ha az adatbázisból szeretnénk az egyes táblákból shapefile-okat készíteni, akkor megadjuk a kijelölt táblát az *Export* fülön. Ha a táblában több geometriát tartalmazó oszlop, akkor azt ki kell választani. A fájl neve is megadható.

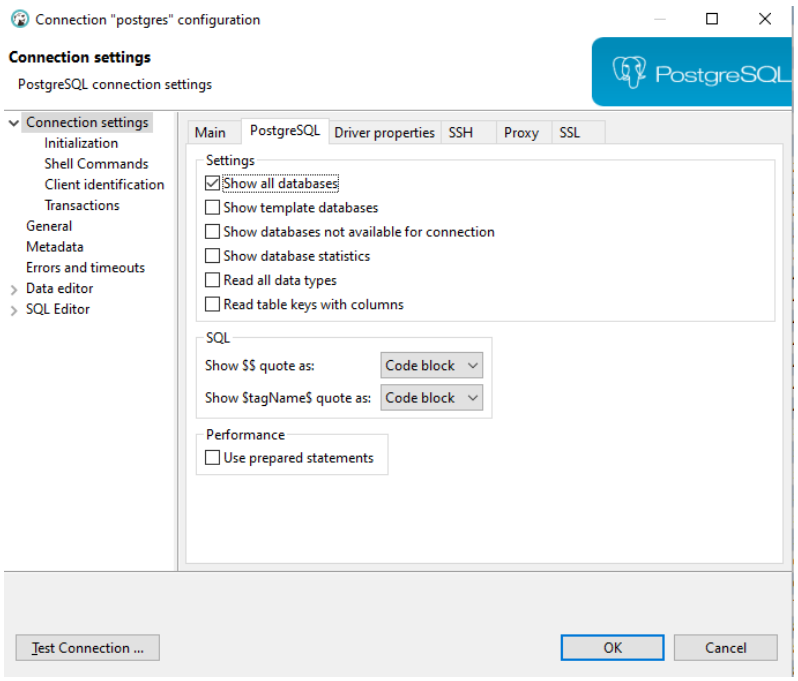


PROBLÉMAMEGOLDÁS GY.I.K.

Előfordul, hogy az alábbi jelenségekkel találkozik a felhasználó: a következőkben ezek megoldását ismertetem.

Hiba: nem látom az adatbázist, pedig már létrehoztam

Jobb egérgombbal klikkelünk a felhasználónéven (postgres) → *Edit connection*. Előjönnek a beállítások → *PostgreSQL fül* → *Show all database* jelölőnégyzet.



Hiba: Túl kis betűs az ablak, növelni szeretném a betűméretet.

Window → *Preferences* → *User Interface* → *Colors and Fonts* → *Basic* → *Text Font* → *Edit*tel megnövelni

Hiba: Egy sorban van egy hosszú lekérdezés az SQL Console-ban. Tördelni szeretném…

CTRL+SHIFT+F

8. FEJEZET: POSTGIS Topology

A PostGIS-ben lehetőség van nemcsak geometriák, hanem topológiák (topológikus adatszerkezet) építésére, tárolására, és a rajtuk műveletek végrehajtására. Erre a PostGIS Topology bővítményét kell hozzáadnunk az Extension-ök közöttt. A Topology részletes dokumentációja ezen a linken érhető el:

https://postgis.net/docs/Topology.html

Miért topológia, mint adattípus?

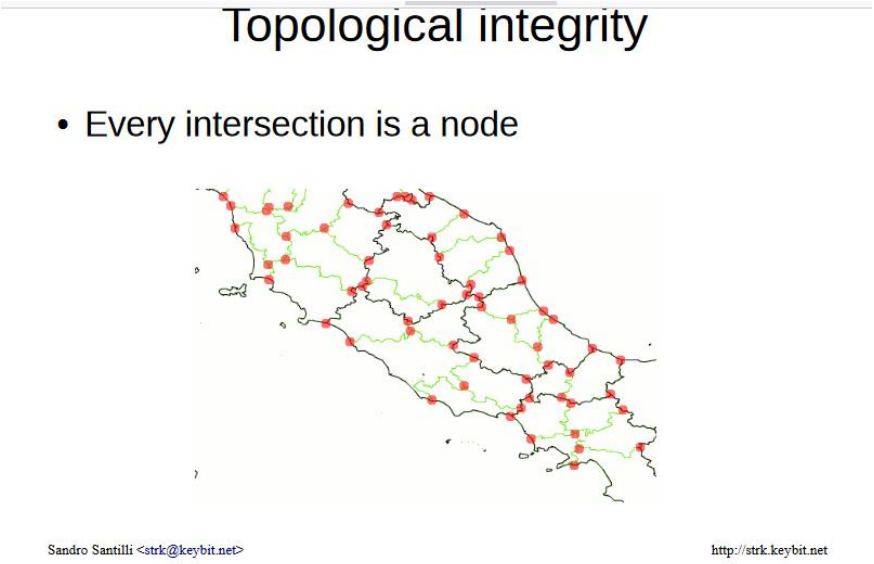
- helytakarékos geometria leírás

- topológikusan helyes tárolás

- térbeli helyzetek/kapcsolatok tárolása

-topológikus integritás: minden metszéspont biztos, hogy egy node. Megosztott élek.

- standard interfész



A kép forrása: http://strk.kbt.io/projects/postgis/Paris2011_TopologyWithPostGIS_2_0.pdf

A topológia tárolásához négy adattípust használ, ezek a következők:

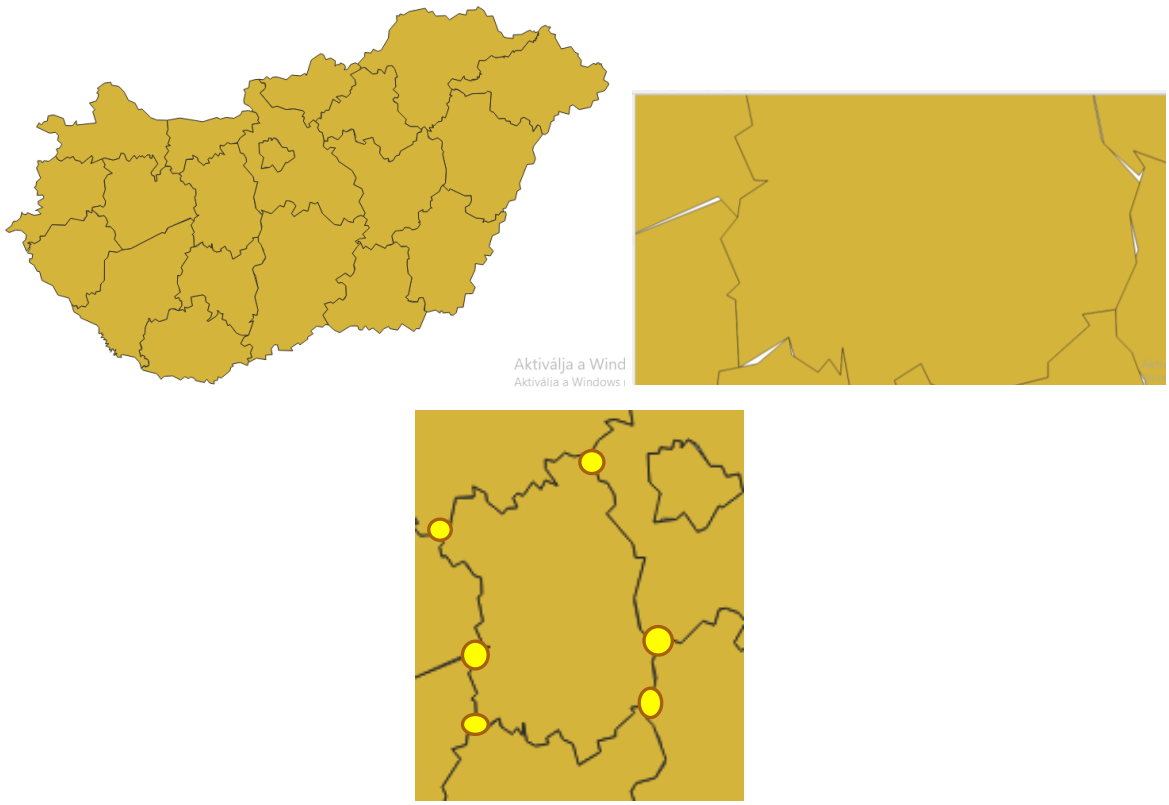
- node (csomópontok)
- edge (élek: a fenti ábrán kép piros pont közötti zöld vonalszakasz).
- face (~alakzatok, élekből felépülő elemek, gyakorlatilag poligonok helyettesítésére)
- relation (elemek közti kapcsolatok)

A topológikus adatszerkezetet bizonyos helyzetekben érdemes használni. A következő példa egy ilyen esetet szemléltet. A feladat az, hogy Magyarország megyéi réteget (amely jelenleg topológiai hibáktól mentes) egyszerűsítsük úgy, hogy a határrajz kisebb méretarányban is felhasználható legyen. Ez a

folyamat a kartográfiai generalizálás: a folyamat során a határvonalak csomópontjainak a száma csökken. Ezáltal a vonalak részletessége is csökken, ennek eredményeként a határrajz kisebb térképi méretarányban használható: pl. a kiindulási anyag nagyjából 1:500 000, a cél legyen az 1:2 000 000 térképi méretarány. Az egyszerűsítés vonalegyszerűsítő algoritmusokkal elvégezhető. Azonban, ha a poligonokat vonalnak értelmezzük, és egyenként egyszerűsítjük (gyakorlatilag mindegy melyik vonalegyszerűsítő eljárással), akkor a közös határszakaszokon lyukak és/vagy átfedések fognak keletkezni. Mindez az elemenkénti egyszerűsítés miatt. Tehát levonható a következtetés, hogy ilyen formában nem megfelelő.

A megoldást a topológiai adatszerkezet bevezetése kínálja. Ugyanis itt a két elem (esetünkben `megye`) közötti közös éleket egyszer tároljuk, vagyis ha ezt egyszerűsítjük, akkor mindkét elem továbbra is osztozik majd a közös, immáron egyszerűsített élen. A fenti ábrán pirossal jelölt pontokat pedig mindenképpen megtartjuk (ezek az élek végpontjai), így az eredmény is topológiailag helyes lesz.

A következő ábra az előbb elmondottakat szemlélteti.



Bal felső kép: Magyarország megyéi. Jobb felső: Az egyszerűsítés lehetséges hibái, ha nem vesszük figyelembe a szomszédos poligonokat. Alsó kép: a sárgával jelölt pontokat mindig meg kell tartani.

A munkafolyamat leegyszerűsítve így néz ki:

- Hozzunk létre egy üres topológiát, majd alakítsuk át a megye réteg geometriáját topológiává, és töltsük bele az üres topológiába
- Készítsünk még egy üres topológiát. Hívjuk meg a topológikus adatszerkezetű megye rétegen a vonalegyszerűsítést, és ezt töltük bele az új üres topológiába.
- Alakítsuk vissza az egyszerűsített topológiát geometriává.

A felhasznált vonalegyszerűsítő algoritmust a feladat után ismertetem. Most nézzük meg részletesen a feladatot.

A `megye` réteg (geometria) egyszerűsítése vonalegyszerűsítő algoritmussal hibás topológiát eredményez:

```
select megye_nev, ST_Simplify(geom, 1000) from megye
```

```
select megye_nev, ST_SimplifyPreserveTopology(geom, 1000) from megye
```

Ezért:

Másoljuk le az eredeti tábla tartalmát (hogy véletlenül se módosítsuk), és adjunk hozzá egy új mezőt az egyszerűsített geometriának!

```
create table ujmegye as (select gid,megye_nev, m_szekhely, geom from megye);
```

```
alter table ujmegye add column simplified_megye geometry(POLYGON, 23700);
```

Készítsünk egy üres topológiát.

```
select topology.CreateTopology('topo1',23700)
```

"Népesítsük be" az üres topológiát, vagyis töltsük bele a megye rétegről a poligonokat,

```
select ST_CreateTopoGeo('topo1', ST_Collect(geom)) from megye
```

Készítsünk egy új, üres topológiát az egyszerűsített geometriának.

```
select topology.CreateTopology('topo2',23700);
```

Töltsük fel a második topológiát az egyszerűsített geometriával.

```
select ST_CreateTopoGeo('topo2',geom) from (select
ST_Collect(ST_SimplifyPreserveTopology(geom, 1000)) as geom from topo1.edge_data)
As foo;
```

Alakítsuk újra geometriává a topológiát, és töltsük is fel az `ujmegye` táblába.

```
with simple_face As (Select ST_GetFaceGeometry('topo2',face_id) as geom from
topo2.face where face_id>0) Update ujmegye d set simplified_megye=sf.geom from
simple_face sf where ST_Intersects(d.geom, sf.geom) and
ST_Area(ST_Intersection(sf.geom,d.geom))/ST_Area(sf.geom)>0.5;
```

Az utolsó lekérdezésben a visszaalakítást eddig nem tanult módon oldom meg, hanem a `with` klauzulával (működését lásd alább). A következő sorban olvasható, hogyan állítom vissza az egyes poligonokat a face-ekből.

```
Select ST_GetFaceGeometry('topo2',face_id) as geom from topo2.face where face_id>0
```

Egy `update` paranccsal frissítem az ujmegye tábla `simplified_megye` (az egyszerűsített geometria) mezőjét a simple face lekérdezésből. A simple face pedig, mint láttuk az egyszerűsített topológiából visszanyert poligonokat tartalmazza.

```
Update ujmegye d set simplified_megye=sf.geom from simple_face sf
```

A visszaállításhoz pedig az ujmegye eredeti geometriájának és a face-ekből visszaállított poligonoknak metszeniük kell egymást, valamint az előző kettő terület metszetének területe (mérete) és a face-ekből

visszaállított poligonoknak területének hányadosa nagyobb, mint 0.5. Ez a feltétel azért kerül bele, mert nem tároltunk el leíró adatokat a topológiákhoz. Ezért az egyszerűsített megye poligonokat az alapján azonosítom, hogy melyik eredeti poligonnal fednek át nagymértékben (az egyszerűsítés után a szomszédos megyékkel kisebb átfedések adódhatnak, ezeket ki kell zárni).

```
where ST_Intersects(d.geom, sf.geom) and
ST_Area(ST_Intersection(sf.geom,d.geom))/ST_Area(sf.geom)>0.5;
```

A feladatban előkerülő függvények magyarázata:

`ST_Collect()` → geometry collection adattípust készít bármely geometriákból.

`ST_CreateTopoGeo()` → a geometriákat a topológiává alakítja

`ST_GetFaceGeometry()` Face-eket készít az azonos id-jű edge-kből.

A **WITH** klauzula:

A with klauzulával az allekérdezés elnevezhető, és akár többször is felhasználható az allekérdezés. Időleges kapcsolatot teremt a lekérdezés fő része és az allekérdezés között. Nem támogatja az összes adatbáziskezelő, de pl. a PostgreSQL igen.

Szintaxis:

```
WITH temporaryTable (averageValue) as
(SELECT avg(Attr1)
FROM Table)
SELECT Attr1
FROM Table, temporaryTable
WHERE Table.Attr1 > temporaryTable.averageValue;
```

Írjunk rá egy példát a `konyvtar` adatbázisból. Keresem azokat a fizetéseket a `konyvtaros` táblából, amelyek nagyobbak, mint az átlag.

```
WITH  konyvt(atlagfizu) as
    (SELECT avg(ks_fizetes)
    FROM konyvtaros)
    SELECT ks_beosztas from konyvtaros, konyvt
    WHERE konyvtaros.ks_fizetes > konyvt.atlagfizu;
```

Még egy példa: írassuk ki, hogy beosztásonként csoportosítva, hogy melyik beosztásnak nagyobb az átlagfizetése, mint a `konyvtaros` tábla átlagfizetése

```
WITH  konyvt(atlagfizu) as
    (SELECT avg(ks_fizetes)
    FROM konyvtaros),
    beoszt(atlagfizu,beosztas) as (select avg(ks_fizetes), ks_beosztas from
konyvtaros group by ks_beosztas)
    SELECT beoszt.beosztas from beoszt, konyvt
    WHERE beoszt.atlagfizu > konyvt.atlagfizu;
```

További elméleti ismertetők, dokumentációk a PosGIS Topology-ról:

http://strk.kbt.io/projects/postgis/Paris2011_TopologyWithPostGIS_2_0.pdf

https://trac.osgeo.org/postgis/wiki/UsersWikiPostgisTopology

Röviden az `ST_Simplify` és az `ST_SimplifyPreserveTopology` működéséről:

A legtöbb térinformatikai szoftverben már régóta rendelkezésünkre áll legalább egy lehetőség a vonalak és poligonok egyszerűsítésére. Ezek többségében a Douglas–Peucker-algoritmust használják. Alapesetben vonalak csomópontszámának csökkentésére találták ki, de felületek egyszerűsítésére is megfelelő. Ilyenkor az alap algoritmus a felület körvonalát vonalként értelmezi: ez abból a szempontból hátrány, hogy nem veszi figyelembe a szomszédos csatlakozó felületeket. Ennek következtében jönnek létre a lyukak vagy átfedések, vagyis a topológiai hibák.

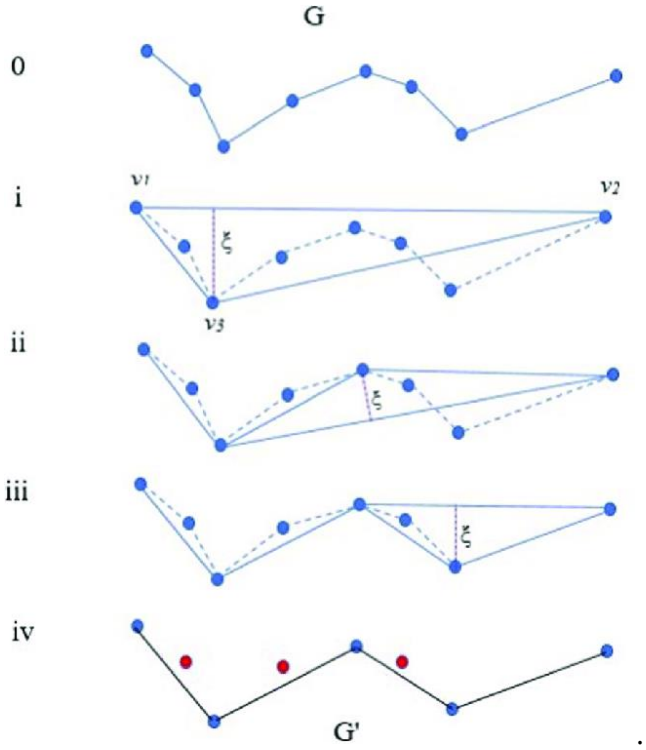
Az algoritmus működése a következő:

1. Bemeneti paraméter egy **ε** merőleges távolság méterben vagy fokban.
2. Kössük össze a vonal első és utolsó csomópontját, és nézzük meg, melyik csomópont esik legtávolabb merőlegesen az összekötő vonaltól. (Feltéve, ha a vonal több mint 3 csomópontból áll.)
3. Nézzük meg, hogy ez a távolság kisebb vagy nagyobb-e mint **ε.** Ha nagyobb, ennél a pontnál kettéválasztjuk a vonalat: egyszer összekötjük a kezdőponttal, majd a végponttal is. Újra, ezekre a szakaszokra is megvizsgáljuk a merőleges távolságot. Ezt ismérteljük, mindaddig, amíg a távolság kisebb nem, vagy nem marad további csomópont az épp vizsgált két csomópont között. Ha kisebb a távolság, a köztes csomóponto(ka)t elhagyjuk.

A Douglas–Peucker-algoritmus rekurzív. Eredményeként a vonal „karakterisztikusabb” részei őrződnek. Azonban ha túl nagy az **ε,** akkor elég durva eredmény kapunk, eléggé szögletes, inkább „túlgeneralizált” lesz a vonal. Ezt érdemes elkerülni: ennél az algoritmusnál, mint egyébként az összesnél van egy tartomány, amelyben optimálisan alkalmazható, ezen kívül kerülendő a használata.

Ajánlott olvasmány az automatizált generalizálásról:

- Ungvári Zsuzsanna: A térképi generalizálás vizsgálata különféle méretarány-tartományokban domborzatmodelleken. Doktori értekezés. ELTE Budapest, 2017. http://lazarus.elte.hu/hun/doktoran/ungvari/ungvari_zsuzsanna.pdf



A kép forrása: Wikipédia. (https://it.wikipedia.org/wiki/Algoritmo_Ramer-Douglas-Peucker)

9. FEJEZET: POSTGIS Raster: raszteres adatok kezelése

Kevésbé ismert és ritkábban használt lehetőség a raszteres adatok adatbázisban való tárolása. Pedig a PostGIS erre is biztosít nekünk eszközt, ez pedig a PostGIS Raster.

https://postgis.net/docs/RT_reference.html

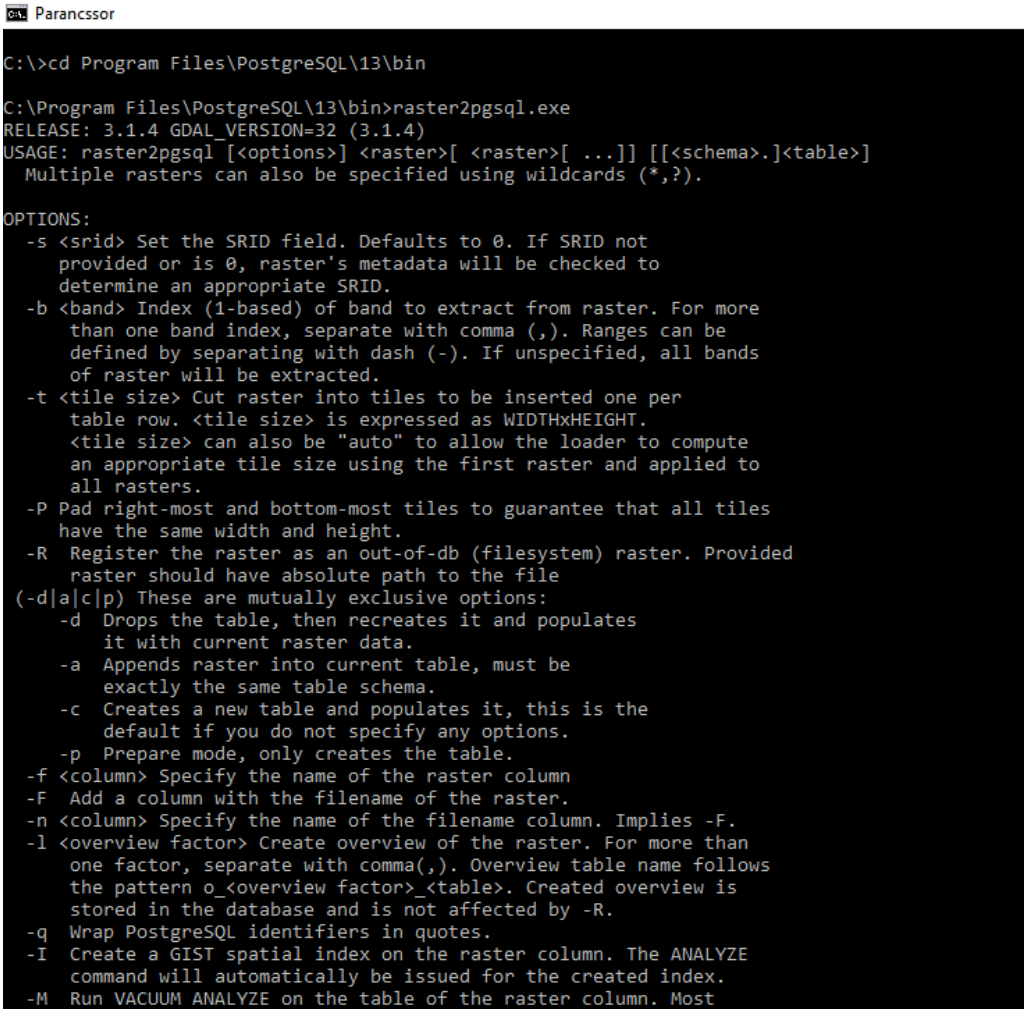
Véleményem szerint a talán a legnehezebb feladat a raszteres képek adatbázisba importálása és exportálása. Ehhez vagy a parancssort (vagy terminált), vagy a pl. a QGIS megfelelő PostGIS Raster Importer plugin-jét kell igénybe vennünk. Mindkettőre fognak ebben a jegyzetben példát találni. A parancssoros importálást Windows-os példán kereszül fogom illusztrálni.

Indítsuk el a cmd.exe-t. Lépjünk be a PostgeSQL mappájába a Program Files mappán belül. Itt található a **raster2pgsql.exe** program. Ez alkalmas raszteres fájlok importálására és exportálására.

```
cd Program Files\PostgreSQL\13\bin
```

Ez a parancssorból indítható program kapcsolókkal paraméterezhető. Kérjük le ezeket a program futtatásával.

```
raster2pgsql.exe
```



```
-C  Set the standard set of constraints on the raster
    column after the rasters are loaded. Some constraints may fail
    if one or more rasters violate the constraint.
-x  Disable setting the max extent constraint. Only applied if
    -C flag is also used.
-r  Set the constraints (spatially unique and coverage tile) for
    regular blocking. Only applied if -C flag is also used.
-T <tablespace> Specify the tablespace for the new table.
    Note that indices (including the primary key) will still use
    the default tablespace unless the -X flag is also used.
-X <tablespace> Specify the tablespace for the table's new index.
    This applies to the primary key and the spatial index if
    the -I flag is used.
-N <nodata> NODATA value to use on bands without a NODATA value.
-k  Skip NODATA value checks for each raster band.
-E <endian> Control endianness of generated binary output of
    raster. Use 0 for XDR and 1 for NDR (default). Only NDR
    is supported at this time.
-V <version> Specify version of output WKB format. Default
    is 0. Only 0 is supported at this time.
-e  Execute each statement individually, do not use a transaction.
-Y  Use COPY statements instead of INSERT statements.
-G  Print the supported GDAL raster formats.
-?  Display this help screen.
```

Írassuk ki a támogatott formátumokat:

```
raster2pgsql.exe -G
```

A következő feladat egy Sentinel II-es műholdkép részlet feltöltése lesz a sentinel táblába, amelyen aztán műveleteket fogunk végrehajtani. A feltöltéshez szükséges kapcsolók:

`-I` Ez a kapcsoló a GIST-nevű térbeli indexet (R-fa) fogja beállítani a táblában. (Create a GIST spatial index on the raster column. The ANALYZE command will automatically be issued for the created index.)

`-s` A térbeli referenciarendszer azonosítója. Ha nem adjuk meg, 0 a default, és megpróbálja a kép metaadataiból meghatározni az SRID-t. (<srid> Set the SRID field. Defaults to 0. If SRID not provided or is 0, raster's metadata will be checked to determine an appropriate SRID.)

Szükség lesz a psql.exe programra is, amely az adatbázishoz való kapcsolódást hajtja végre. Dokumentációja (a kapcsolók miatt) elérhető a weben: https://www.postgresql.org/docs/current/app-psql.html

`-U` felhasználónév

`-d` adatbázis neve

`-h` hoszt/szerver, itt localhost.

`-p` port. A PostgreSQL alapesetben, ha ezen a felhasználó a telepítéskor nem módosít, akkor az 5432-es porton fut. Ide a saját gépén beállított port számát kell értelemszerűen megadni.

Most már minden ismert a kép importálásához:

```
raster2pgsql.exe -I -s 32633
G:\ADATOK\BENTI\zsuzsi\oktatas\terbeli_adatbazisok\raszter\sentinel\sentinel_felve
tel.tif public.sentinel | psql -U postgres -d mo -h localhost -p 5432
```

vagy (ha egy hiányzó konstansra vonatkozó hibaüzenetet kapunk, adjunk megaz `-I` után egy random számot pl. 2.)

```
raster2pgsql.exe -I 2 -s 32633
G:\ADATOK\BENTI\zsuzsi\oktatas\terbeli_adatbazisok\raszter\sentinel\sentinel_felvetel.tif public.sentinel | psql -U postgres -d raszter -h localhost -p 5432
```

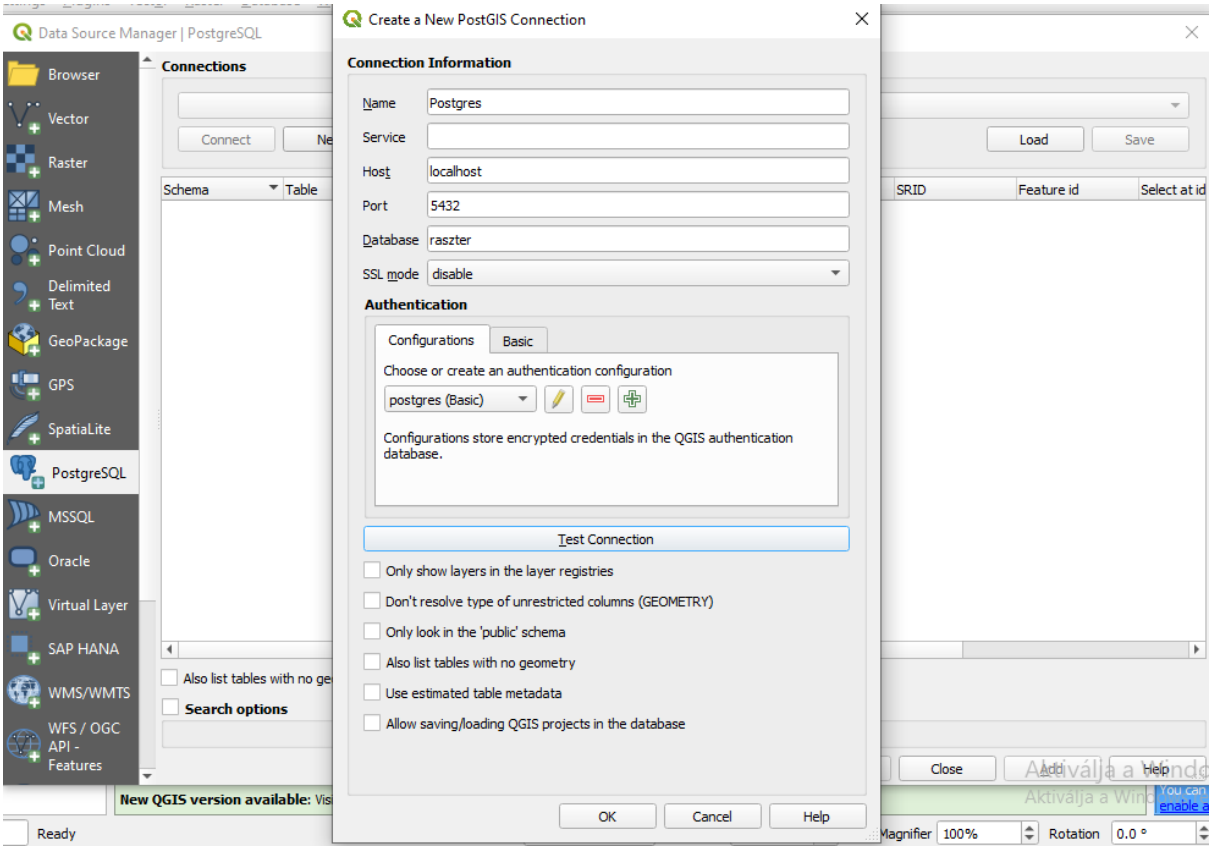
Kérni fogja a jelszót is a csatlakozás során, ezt adjuk meg.

Ezek után tekintsük át, hogyan lehet PostgreSQL + PostGIS-be raszteres adatot feltölteni QGIS plugin használatával.

A QGIS-ben a *PlugIns*→ *Manage and Install Plugins* menüben a *PostGIS Raster Import* modult kell keresni és telepíteni.

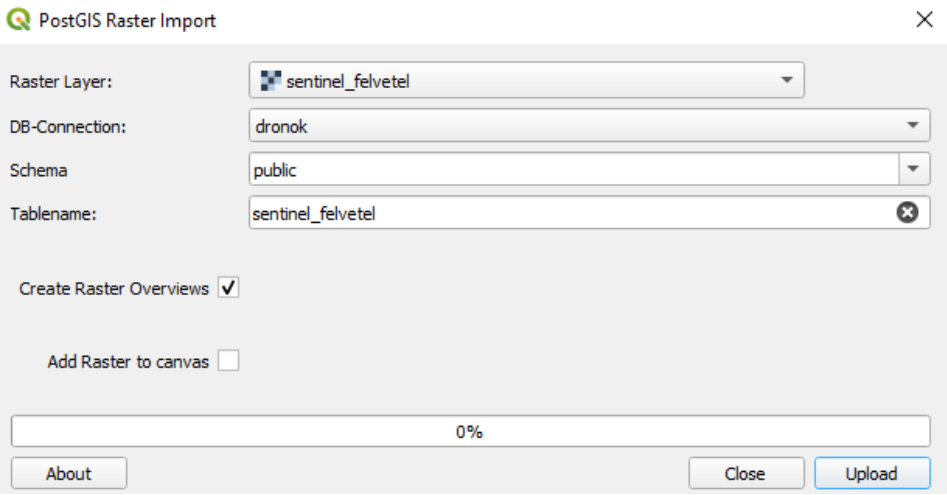
Majd először definiálni kell a kapcsolatot a *Data Source Manager*ben → *PostgreSQL* fül, majd *New*.

Megadjuk a felhasználónevet, szervert, portot és adatbázist.Egy teszt kapcsolatot érdemes létesíteni (*Test connection*). Majd a *Connect* gombra kattintva kapcsolódunk az adatbázishoz. Ezután tudjuk használni a *Raster Import*ot.



Itt meg kell adni a raszteres rétegből képzett tábla nevét. Lehetőség van az import során különböző felbontási szinteket készíttetni a felvételből. Ez inkább nagyméretű (pár száz MB méret feletti) képekből már érdemes lehet. Viszont ne felejtsük el, hogy ez igencsak megnöveli az adatbázis méretét. A kisebb felbontási szintű táblákat a következő előtaggal nevezi el automatikusan (*Create Raster Overviews*): o_2, o_4, o_8, o_16, o_32, o_64, o_128, o_256.

Bármely módszert is választjuk az importra, a létrejövő táblában egy rid és egy rast mező keletkezik. (megjegyezném, hogy amennyiben be van jelölve a *Create Raster Overviews*) tile-okra bontja a képeket.



Miután megnéztük az adatfeltöltési lehetőségeket, nézzük meg, hogy milyen lehetőségeink vannak az adatbáziskezelőben, milyen lekérdezésekkel tudjuk manipulálni a raszteres képeket.

Írassuk ki az SRID-t!

```
select ST_SRID(rast) FROM sentinel where rid=1
```

Ha nem megfelelő, az `UpdateRasterSRID` függvénnyel frissíthető.

```
select UpdateRasterSRID('sentinel','rast',32633)
```

Írassuk ki a raszteres kép csatornáinak számát:

```
select ST_NumBands(rast) from sentinel
```

Írassuk ki a kép georeferenciára vonatkozó adatait!

`ST_PixelHeight(raszter)` → A képfelbontása függőleges irányban = Az egyes pixelek mérete függőlegesen.

`ST_PixelWidth(raszter)` → A képfelbontása vízszintes irányban =Az egyes pixelek mérete vízszintesen

`ST_Height(raszter)` → A kép sorainak száma

`ST_Width(raszter)` → A kép oszlopainak száma

`ST_UpperLeftX(raszter)` → A kép bal felső sarkának X koordinátája

`ST_UpperLeftY(raszter)` → A kép bal felső sarkának Y koordinátája

```
select ST_PixelHeight(rast), ST_PixelWidth(rast), ST_Height(rast), ST_Width(rast),
ST_UpperLeftX(rast), ST_UpperLeftY(rast), st_srid(rast) from sentinel
```

Írassuk ki a felvétel 1. csatornájának metaadatait!

`ST_BandMetaData(raszter, csatorna száma)` → A visszatérő sztringben a következő adatok olvashatók ebben a sorrendben: a pixel típusa(`pixeltype`), NoData érték (`nodatavalue`), a fájl az adatbázisban van-e: true/false (`isoutdb`), az elérési út, ha nem adatbázisban van (`path`), a külső fájl csatornáinak száma (`outdbbandnum`), a külső fájl mérete (`filesize`), a külső fájl létrehozásának dátuma (`filetimestamp`).

```
select rid, ST_BandMetaData(rast, 1) FROM sentinel
```

Kérdezzük le a kép befoglalóját!

```
select ST_Envelope(rast) from sentinel
```

Kérdezzük le a kép konvex burkát!

```
select ST_ConvexHull(rast) from sentinel
```

Kérdezzük le a kép által lefedett területet, vagyis alakítsuk át a képet polygonná! A no data értékeket nem vesszük figyelembe. Az eredmény multipoligonként tér vissza (Lassú!)

```
select ST_Polygon(rast) from sentinel
```

Keressük meg, hogy melyik megyében van a kép. Ehhez szükség lesz a megyek táblára is.

```
select m_name from megyek where ST_Intersects(ST_Transform(m_geom,32633),(select
rast from sentinel where rid=1))
```

Vágjuk el Somogy megye poligonjával a sentinel műholdfelvételt!

```
select ST_Clip(rast, (select ST_Transform(m_geom,32633) from megyek where
m_name='Somogy megye')) from sentinel where rid=1
```

Írassuk ki az egyik csatornát a képből.

```
select ST_Band(rast,1) from sentinel
```

Statisztika kinyerése a felvételekről:

ST_Histogram(raszter, csatorna száma, tartományok) → A megadott csatornára kiszámítja a hisztogramot. A tartományok azt fejezi ki, hogy mennyi alegységbe szeretnénk sorolni a teljes tartományt (a default érték nem 1!).

```
select ST_Histogram(rast,1,1) from sentinel → a megadott kép 1. csatornájára:
(155,4527,474351,-1)
```

→Min pixel érték, Max. pixelérték, pixelek száma, a pixelek hány százaléka esik abba a tartományba

ST_Count(raszter, csatorna száma, nodata értéket figyelembe vegye-e: true/false) → Megadja, hány pixel van a képen (nodata értéket is beleértve/vagy nem)

```
select ST_Count(rast,1,true) from sentinel
```

ST_Quantile(raszter, csatorna száma, nodata érték figyelembe vétele: true/false,*quantiles*) → Megadja a pixelek kvantilisét, vagyis ha a tartomány 0-1 közötti, akkor mennyi a pixelérték pl. 0.25, 0.5 és 0.75-nél.

```
select ST_Quantile(rast,1,true) from sentinel → (0,155) | (0.25,326) | (0.5,510)
| (0.75,925) | (1,4527)
```

```
select ST_Quantile(rast,1,true, array[0.25,0.75]) from sentinel → (0.25,326) |
(0.75,925)
```

ST_SummaryStats(raszter, csatorna száma nodata értéket figyelembe vegye-e: true/false) → Aggregáló függvények készítenek statisztikát a képről. A kapott értékek sorrendben: a pixelek száma, a pixelek összértéke (sum), a pixelek átlagértéke (avg), standard deviation, mininális pixelérték, maximális pixelérték.

```
select ST_SummaryStats(rast,1, true) from sentinel →
(474351,299149086,630.6492154543787,333.5711420622201,155.0,4527.0)
```

`ST_ValueCount(raszter, csatorna száma nodata értéket figyelembe vegye-e: true/false)` → további paraméterek is lehetségesek lsd. dokumentáció. Megadja, hogy az adott értékű pixelből hány db van a felvételen.

```
select ST_ValueCount(rast,1, true) from sentinel → (209,32)| (302,1956)|
(357,953)…
```

Számítsuk ki az NDVI indexet az alábbi képlet alapján. A felvétel 3-as csatornája a vörös (Red), a 7-es pedig közeli infravörös (NIR).

$$NDVI = \frac{NIR - Red}{NIR + Red}$$

A `ST_MapAlgebra` függvényt többféleképpen is használhatjuk, attól függően, hogy egy kép több csatornájával dolgozunk, vagy esetleg több kép egyes csatornáira van szükségünk. Részletek a hivatalos dokumentációban, itt én egy példát fogok bemutatni.

`ST_MapAlgebra(táblanév.mezőnév, első csatorna száma, táblanév.mezőnév, második csatorna száma, kifejezés)` → eredményül a feldolgozott rasztert kapjuk vissza.

A kifejezésben a raszteres kép csatornáira az alábbi módon hivatkozunk, az elsőnek megadottra rast1.val, a másodikra pedig rast2.val. A két kettőspont után a kimeneti kép pixeleinek típusát adjuk meg.

```
select ST_MapAlgebra(sentinel.rast, 3, sentinel.rast, 7, '([rast2.val] -
[rast1.val]) / ([rast2.val] + [rast1.val])::float','32BF') from sentinel
```

Az eredménytáblából az eredményt igényünk támadhat fájlba írni, ezt a következőképpen lehet megtenni. Először is meg kell mondani, milyen formátumot szeretnénk kapni. Majd az az `lo` függvények használatával képesek leszünk a merevlemezre írni a fájlt.

A `lo_from_bytea()` függvény arra szolgál, hogy egy nagy objektumot hozzon létre, amelyben adatokat is tárolhat. Ha az oid 0 (első paraméter), akkor a rendszer hozza létre véletlenszerűen. A létrejövő objektumot egy táblába mentjük el. Majd innen exportáljuk. A `lo_export` függvénnyel vigyázzunk, előfordulhat, hogy pl. a C:\ (rendszer) meghajtóra nem tudunk fájlt menteni, viszont egy másik (nem rendszer) meghajtóra sikerül. Ennek oka, hogy a fájlok írásához az adatbázis tulajdonosának jogosultságait használja. Az `lo_unlink` pedig törli a nagyméretű objektumot (a végén pedig az üres tmp_out tábla marad meg).

Az lo függvényekhez: https://www.postgresql.org/docs/current/lo-funcs.html

```
create table tmp_out As
select lo_from_bytea(0,
      ST_AsGDALRaster((select ST_MapAlgebra(sentinel.rast, 3, sentinel.rast, 7,
      '([rast2.val] - [rast1.val]) / ([rast2.val] + [rast1.val])::float','32BF')
      from sentinel), 'GTiff',  ARRAY['COMPRESS=DEFLATE', 'PREDICTOR=2',
      'PZLEVEL=9'])
        ) AS loid
FROM sentinel;

select lo_export(loid, 'G:\myraster.tiff') from tmp_out;
```

```
Select lo_unlink(loid) from tmp_out;
```

Egy függvényt nem tárgyaltunk még az alábbi lekérdezésben, ez pedig a `ST_AsGDALRaster`.

`ST_AsGDALRaster(formátum, opciók, SRS)` → a visszatérési értéke a paraméterekben megadott fájlformátum. A paraméterek megadásában az alábbi oldalak nyújthatnak segítséget:

Összes formátum: https://gdal.org/drivers/raster/index.html

GeoTIFF: https://gdal.org/drivers/raster/gtiff.html#raster-gtiff

JPG: https://gdal.org/drivers/raster/jpeg.html#raster-jpeg

```
select ST_AsGDALRaster(ST_band(rast,1), 'JPEG', ARRAY['QUALITY=50']) As rastjpg
from sentinel where rid=1
```

A raszteres kép exportálása/mentése adatbázisból QGIS-szel is lehetséges. Ebben az esetben kapcsolódnunk kell az adatbázishoz és be kell tölteni a képet a munkafelületre (lásd fentebb). Innen már a réteg mentésével fájlba menthető a rekord (Jobb egérgombbal kattintás a rétegen. *Export* → *Save As*).

A PostGIS Raster kiegészítőjében számos más függvény található a fenti tárgyaltakon túl. Ehhez érdemes a fejezet bevezetőjében található hivatalos dokumentációt is meglátogatni. A Raster modul nemcsak űrfelvételekkel vagy ortofotókkal való műveletekre szolgál, hanem domborzatmodellekkel is tud dolgozni pl. árnyékolás `ST_Hillshade`, lejtőszög (`ST_Slope`) és kitettség (`ST_Aspect`) stb. képes számolni.

Egyéb ajánlott olvasnivaló a PostGIS raszterről:

https://github.com/lcalisto/workshop-postgis-raster

10. FEJEZET: Melléklet – a jegyzetben használt adatbázisokról, fájlokról

A CSV fájlok (Comma Separated Values) gyakorlatilag táblázatos adatokat tartalmaznak egy egyszerű szöveges formátumban. A táblázat sorai a textfájlban is külön sorba kerülnek, az oszlopokat egymástól valamilyen reguláris karakterrel választjuk el pl. vessző, pontosvessző, tabulátor.

europa.csv

Az európai országok nevét, területét, lakosságát, népsűrűségét, és fővárosukat tartalmazó fájl.

kiadvany_csoport.csv

A konyvtar adatbázis egy táblája. A `cs_kod` mező az elsődleges kulcs, a `cs_nev` a kiadvány csoport neve, a `cs_kateg_k` a csoport kategóriájának rövidítése és a `cs_kiadás_k` pedig a csoport elemeinek kiadhatóságát (kölcsönözhetőségét) mutatja.

Az SQL fájlok adatbázisok biztonsági mentésére készült fájlok, szkriptek. Előnyük, hogy a teljes táblaszerkezetet tartalmazzák Create és Insert utasításokkal (táblaszerkezet és adatok egyben). Így a felhasználóknak csak annyi a dolga, hogy egy szkriptként importálja és lefutattja a fájlt. Vigyázzunk, mert a MySQL és a PostgreSQL szkriptek nem feltétlenül kompatibilisek egymással, apróbb átalakítások szükségesek lehetnek.

konyvtar.sql

A `konyvtar` adatbázist tartalmazó szkript. Táblák: `katalogus, kiadvany, kiadvany_csoport, kolcsonzes, kolcsonzo, konyvtaros, tanszek`. Részletes leírás a szerkezetről: http://mercator.elte.hu/~saman/hu/okt/mysql/sql_tablaszerkezet.doc

mo.sql

A Magyarország adatbázis biztonsági mentése. Táblaszerkezet alább.

epuletek.shp

Épületek geometriáját tartalmazó Shapefile.

sentinel.tif

A Sentinel 2-es műholdcsalád egy felvételének részlete egy magyarországi mintaterületen. A felvételen tíz csatorna van (nem 13), ezek sorrendben: 1:b, 2: g, 3:r, 4,5,6: red edge, 7:nir, 8:red edge, 9: water vapour, 10: swir

NDVI=(ch7-ch3)/(ch7+ch3)

Adott egy `mo` adatbázis a következő táblákkal. Vetület: EOV 23700. (Kivéve `var`!)

		Mező/oszlop neve	Adattípus	
Az mo adatbázis táblái	**megyek**	m_id	serial4	Elsődleges kulcs
		m_name	varchar(254)	Megye neve
		m_megye_kod	int4	A megye kódja (véletlenszerű)
		m_geom	geometry (MultiPolygon)	Megye geometriája
	jarasok	j_id	serial4	Elsődleges kulcs
		j_name	varchar(254)	Járás neve
		j_megye_kod	int4	Megye kódja amiben van
		j_jaras_kod	int4	Járás kódja
		j_geom	geometry (MultiPolygon)	Járás geometriája
	telepulesek	t_id	serial4	Elsődleges kulcs
		t_name	varchar(254)	település neve
		t_name_de	varchar(254)	település német neve, ha van
		t_jaras_kod	int4	a járás kódja, amiben van
		t_geom	geometry (MultiPolygon)	település geometriája KÜLTERÜLETHATÁR
	telepulescentroid	tc_id	int4	Elsődleges kulcs
		tc_name	varchar(254)	Település neve
		tc_geom	geometry	Település centroidja a belterület alapján
	belterulet	b_id	int4	Elsődleges kulcs
		b_name	varchar(254)	Település neve
		b_geom	geometry (MultiPolygon)	Település BELTERÜLETE
	autopalya	ap_id	int4	Elsődleges kulcs
		ap_wayid	varchar(254)	OSM azonosító
		ap_hu_ed_dire	varchar(254)	Autópálya iránya Bp-ről
		ap_alt_name	varchar(254)	Autópálya régi neve (többnyire NULL)
		ap_alt_bridge	varchar(254)	a szakasz híd?
		ap_highway	varchar(254)	autópálya típus: motorway
		ap_int_ref	varchar(254)	Egyéb útszámozás
		ap_lanes	varchar(254)	sávok száma

		ap_maxspeed	varchar(254)	maximális sebesség a szakaszon
		ap_oneway	varchar(254)	egyirányú?
		ap_ref	varchar(254)	**Autópálya számozása**
		ap_surface	varchar(254)	burkolata
		ap_toll	varchar(254)	útdíjköteles?
		ap_geom	geometry(multilinestring)	geometriája
	autout	au_id	int4	Elsődleges kulcs
		au_wayid	varchar(254)	OSM azonosító
		au_hu_ed_dire	varchar(254)	Autópálya iránya Bp-ről
		au_bridge	varchar(254)	a szakasz híd?
		au_foot	varchar(254)	gyalogosan használható?
		au_highway	varchar(254)	út típus
		au_horse	varchar(254)	lóval járható?
		au_int_ref	varchar(254)	Egyéb útszámozás
		au_maxspeed	varchar(254)	maximális sebesség a szakaszon
		au_name	varchar(254)	a szakasz neve
		au_oneway	varchar(254)	egyirányú?
		au_ref	varchar(254)	**Autóút száma**
		au_surface	varchar(254)	burkolata
		au_toll	varchar(254)	díjköteles?
		au_tunnel	varchar(254)	Alagút?
		au_geom	geometry(multilinestring)	autóút geometriája
	elsorendu	el_id	int4	Elsődleges kulcs
		el_wayid	varchar(254)	OSM azonosító
		el_hu_ed_dire	varchar(254)	Autópálya iránya Bp-ről
		el_bicycle	varchar(254)	biciklivel járható?
		el_bridge	varchar(254)	a szakasz híd?
		el_foot	varchar(254)	gyalogosan használható?
		el_highway	varchar(254)	út típus
		el_horse	varchar(254)	lóval járható?
		el_int_ref	varchar(254)	Egyéb útszámozás

		el_maxspeed	varchar(254)	maximális sebesség a szakaszon
		el_name	varchar(254)	a szakasz neve
		el_oneway	varchar(254)	egyirányú?
		el_ref	varchar(254)	**Elsőrendű út száma**
		el_surface	varchar(254)	burkolata
		el_toll	varchar(254)	díjköteles?
		el_tunnel	varchar(254)	Alagút?
		el_geom	geometry(multilinestring)	út geometriája
	masodrendu	ma_id	int4	Elsődleges kulcs
		ma_wayid	varchar(254)	OSM azonosító
		ma_bicycle	varchar(254)	biciklivel járható?
		ma_bridge	varchar(254)	a szakasz híd?
		ma_foot	varchar(254)	gyalogosan használható?
		ma_highway	varchar(254)	út típus
		ma_horse	varchar(254)	lóval járható?
		ma_maxspeed	varchar(254)	maximális sebesség a szakaszon
		ma_name	varchar(254)	a szakasz neve
		ma_oneway	varchar(254)	egyirányú?
		ma_ref	varchar(254)	**Másodrendű út száma**
		ma_surface	varchar(254)	burkolata?
		ma_toll	varchar(254)	díjköteles?
		ma_tunnel	varchar(254)	alagút-e?
		ma_geom	geometry(multilinestring)	út geometriája
	tertiary	te_id	int4	Elsődleges kulcs
		te_wayid	varchar(254)	OSM azonosító
		te_bicycle	varchar(254)	biciklivel járható?
		te_bridge	varchar(254)	a szakasz híd?
		te_foot	varchar(254)	gyalogosan használható?
		te_highway	varchar(254)	út típus
		te_horse	varchar(254)	lóval járható?
		te_maxspeed	varchar(254)	maximális sebesség a szakaszon

		te_name	varchar(254)	a szakasz neve
		te_oneway	varchar(254)	egyirányú?
		te_ref	varchar(254)	**Másodrendű út száma**
		te_surface	varchar(254)	burkolata?
		te_toll	varchar(254)	díjköteles?
		te_tunnel	varchar(254)	alagút-e?
		te_geom	geometry(multilinestring)	út geometriája
	vasut	v_id	serial4	Elsődleges kulcs
		v_name	varchar(48)	Szakasz neve ha van
		v_tipus	varchar(16)	típusa (hév, keskeny nyomtáv, mellékvonal stb.)
		v_vonalszam	varchar(10)	Vonalszáma
		v_statusz	varchar(15)	A szakasz státusza (üzemel, megszűnt)
		v_uzemelteto	varchar(5)	Üzemeltető neve
		v_geom	geometry(multilinestring)	Vasút geometriája
	folyo_felulet	ff_id	int4	Elsődleges kulcs
		ff_name	varchar(254)	Folyó neve
		ff_geom	geometry(multipolygon)	Felületként reprezentált folyó geometriája
	folyo_vonal	fv_id	int4	Elsődleges kulcs
		fv_name	varchar(254)	Folyó neve
		fv_kategoria	varchar(50)	Folyó kategóriája
		fv_geologia	varchar(50)	Folyó geológiai környezete
		fv_eses	varchar(50)	Esés mértéke
		fv_torkolat	varchar(6)	Torkolat azonosító
		fv_leiras	varchar(250)	Folyó részletes leírása
		fv_geom	geometry(multilinestring)	Folyó geometriája vonalként
	folyok	f_id	int4	Elsődleges kulcs
		f_name	varchar(254)	Folyó neve
		f_name_hu	varchar(254)	Folyó magyar neve
		f_geom	geometry(multilinestring)	Folyó geometriája
	patak	p_id	int4	Elsődleges kulcs
		p_name	varchar(254)	Patak neve

		p_geom	geometry(multilinestring)	A patak geometriája
	to	to_id	serial4	Elsődleges kulcs
		to_name	varchar(100)	Tó neve
		to_kategoria	varchar(30)	Tó kategóriája (tározó stb.)
		to_tszf	numeric	A tó tengerszint feletti magassága
		to_altalanos	varchar(50)	A tó táji jellege
		to_melyseg	numeric	A tó mélysége
		to_leiras	varchar(150)	A tó leírása
		to_geom	geometry(multipolygon)	A tó geometriája
	mo_hatar	mo_id	int4	Elsődleges kulcs
		mo_name	varchar(254)	Ország neve
		mo_geom	geometry(multipolygon)	Geometria
	geolada_eov	gid	serial(4)	Elsődleges kulcs
		gl_id	float(8)	Másik azonosító
		gl_lat_d	float(8)	Szélesség fok
		gl_lat_m	numeric	Szélesség perc
		gl_lon_d	float(8)	Hosszúság fok
		gl_lon_m	numeric	Hosszúság perc
		gl_elevation	numeric	Tszf. magasság
		gl_full_name	varchar(254)	Láda neve
		gl_type	varchar(254)	Láda típusa
		gl_megye_orsz	varchar(254)	Megye/ország ahol van a láda
		gl_terep	numeric	Terep nehézsége
		gl_waypoint	varchar(254)	Rövid azonosító
		gl_description	varchar(254)	Láda leírása
		gl_felhaszn	varchar(254)	Tulajdonos
		gl_email	varchar(254)	Tulaj e-mail címe
		gl_field_30	numeric	Szélesség fokban
		gl_field_31	numeric	Hosszúság fokban
		gl_geom	geometry(point(23700))	Geometria
	nt_tk	np_id	in(4)	Elsődleges kulcs
		np_name	varchar(254)	Nemzeti park vagy tájvédelmi körzet neve

		np_tipus	varchar(50)	Nemzeti park vagy tájvédelmi körzet neve
		np_geom	geometry(MultiPolygon)	Geometria
	var	va_id	int(4)	Elsődleges kulcs
		va_name	varchar(254)	Vár neve
		va_geom	geometry(geometry,4326)	**geometria 4326! poligon vagy pont**
	spatial_ref_sys			**A PostGIS Extension részeként létrejövő tábla a térbeli referenciarendszerekkel!**