Eötvös Loránd Tudományegyetem
Informatikai Kar
Információs Rendszerek Tanszék

Integrált adatbázis-fejlesztői környezet SQL-hez

dr. Nikovits Tibor
mestertanár

Lestár Norbert
programtervező informatikus BSc

Budapest, 2017

Tartalomjegyzék

1. fejezet

Bevezetés

A szoftverfejlesztést nagyban meghatározza milyen fejlesztői környezetben végezzük a fejlesztést. A fejlesztői környezet az, amiben létrejön a szoftverfejlesztés folyamata során a kész program, avagy amiben lehet fordítani, és tesztelni az általunk írt kódot. Egy integrált fejlesztői környezet ezt bővíti ki olyan eszközökkel, amelyek segítségével a fejlesztés kényelmesebb, gyorsabb lesz, így egy hatékonyabb eszközt adva a fejlesztő kezébe. Ezzel szemben nem csak tapasztalt felhasználók tarthatnak igényt egy ilyen környezetre, hanem kevésbé tapasztalt, jelenleg tanuló felhasználók is.

A program ezeknek a felhasználóknak is hasznos lehet, akik jelenleg csak ismerkednek az SQL-el. Ezen felhasználók többféleképpen is nekikezdhetnek a tanulásnak, erre az egyik mód egy integrált fejlesztői környezet használata, ami kényelmes felületet biztosít a tanuláshoz, többek között a szintaxis kiemelésnek, illetve a könnyen szerkeszthető, és futtatható lekérdezések segítségével.

Felmerülhet a kérdés, hogy pontosan mire is jó az SQL? Az SQL, azon belül is az Oracle SQL nyelv mögötti szerver képes biztonságosan tárolni nagy mennyiségű adatokat, hiba esetén helyreállítani azokat, illetve lehetőség van a különböző adattáblák indexelésére ami felgyorsíthatja a lekérdezéseket, de ezen kívül számos pozitív (és persze negatív) tulajdonsága is van, ami jelenlegi program szempontjából nem annyira lényeges. Felmerülhet az igény, hogy ezeket az adatmennyiségeket (például napi adatgyűjtés a felhasználók szokásairól) valamilyen szempont alapján megfigyeljük, megjelenítsük. Erre szolgál az ebben a programban létrehozható diagramok: ha megfelelően lekérdezzük az adatokat jól látható eredményeket kapunk, amik segíthetnek bizonyos szolgáltatások fejlesztésében.

A szerkesztőfelületnek köszönhetően ha hiba történik az könnyen javítható, valamint elmentheti munkáját, betölthet korábban létrehozott SQL fájlokat is. A szintaxis kiemelésnek köszönhetően könnyű észrevenni, és javítani a hibákat, sőt lényegesen gyorsabban meg lehet tanulni az SQL nyelvet. Bizonyos műveleteket akár az SQL teljes ismerete nélkül is elvégezhet bárki: adatbázis objektumok megte-

kintését, törlését, esetlegesen korábban létrehozott lekérdezésekből diagramok megjelenítését. Továbbá tapasztalt felhasználóknak is hasznos eszköz, könnyebben tudnak SQL lekérdezéseket, illetve szkripteket készíteni, és futtatni. Emellett képes megjeleníteni SQL lekérdezéseknek a lekérdezési tervét.

Mindezen funkciók megvalósítása volt a cél a program írása közben. Mivel az implementáció Qt keretrendszer segítségével történt, így könnyen bővíthető a program további adatbázis szerverek használatával, azonban a program csak Oracle SQL adatbázis szerverhez képes csatlakozni. A programra *DiagramQuery* néven fogok a továbbiakban hivatkozni.

2. fejezet

Felhasználói dokumentáció

A *DiagramQuery* első sorban lehetőséget biztosít egy Oracle SQL adatbázishoz való kapcsolódáshoz, illetve eme adatbázison lekérdezések, szkriptek végrehajtására, továbbá diagramok létrehozására ezen lekérdezések segítségével. A kapcsolatokat tartalmazó adatokat el is lehet menteni (valamint lehetőség van kézzel leírni, ez kifejtésre kerül az első szekcióban.), illetve betölteni egy .xml formátumú fájlból.

A kapcsolat felépítése után juthatunk a fő ablakba, itt lehetőség van a fentebb említett SQL lekérdezések végrehajtására, azonkívül PL/SQL szkriptek futtatására, illetve diagramokat is itt lehet készíteni, ennek részletezése a 2.3.-ban történik meg.

2.1. Telepítés és beállítás

A *DiagramQuery* telepítése nem szükséges, ám a lemezen található minden állomány szükséges a futásához, így futtatáshoz másoljon át minden fájlt, és futtassa a *DiagramQuery.exe*-t. Windows 10 alatt lett tesztelve, a program bármilyen rendszeren képes megfelelően működni ahol a Windows 10 is elfut alkalmas módon.

Érdemes megjegyezni, hogy a DiagramQuery Qt keretrendszerben lett megírva. Ez azt jelenti, hogy nem csak Windowson lehet futtatni fordítás után, hanem bármin amin Qt és OCI elérhető. Linuxon is tesztelve lett, és helyesen működik. A program telepítése forráskódból:

1. Qt 5.6, vagy frissebb letöltése és telepítése.
2. Oracle Instant Client letöltése, valamint telepítése (Oracle oldalán elérhető, regisztrációhoz kötött).
3. Qt OCI pluginjának fordítása.
4. Utolsó lépésként egyszerűen fordítható a program, immáron működőképes kell legyen.

Lehetőség van későbbi fejlesztések során további adatbázis-kezelők hozzáadására a forráskód birtokában, azonban ekkor manuálisan kell telepíteni az előbb leírtak szerint. Telepítés után lehetőség van manuálisan létrehozni kapcsolatokat. Ha ezt kívánja tenni, akkor hozzon létre egy *Connections* mappát a futtatható állomány könyvtárába, és hozzon létre egy .xml kiterjesztésű fájlt a kívánt névvel.

Egy példa a fájlra:

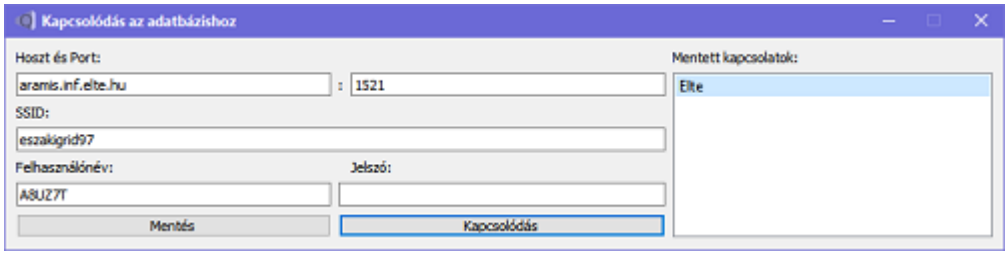
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Connection name="Elte">
3      <host>aramis.inf.elte.hu</host>
4      <port>1521</port>
5      <service>eszakigrid97</service>
6      <username>A8UZ7T</username>
7  </Connection>
```

A fájl felépítése a következő legyen:

- A második (és hetedik) sorhoz hasonlóan **Connection** címkék között legyen elhelyezve az egész kód, illetve legyen a nyitócímkének egy attribútuma: name, a kapcsolat nevével,
- **host** címkék között a kapcsolat címe legyen a harmadik sorban látható módon,
- **port** címkék között legyen megadva a kapcsolat portja úgy, ahogy a negyedik sorban látható,
- **service** címkék között legyen a szerviz neve az ötödik sorhoz hasonlóan,
- illetve **username** címkék között a bejelentkezéshez szükséges felhasználónév ugyanúgy mint a hatodik sorban.

Ha valamelyik mezőt nem szeretné eltárolni, akkor azt címkével együtt hagyja ki.

Miután létrehoztuk a fájlt, utána el is kezdhetjük a program használatát, további beállításra nincs szükség. A mentett kapcsolatok kezelésére itt is lehetőség van.

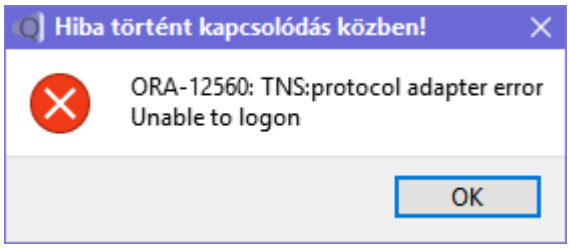


2.1. ábra. Kapcsolódási ablak

Duplán kattintva egy kapcsolat nevére betölthetjük annak adatait. Ha valamelyik

mező üresen lett hagyva, akkor az ablakban levő mező is üres lesz. (Abban az esetben is, ha korábban lett bele írva.) A *delete* billentyű lenyomására a kijelölt kapcsolat törlődik, de mielőtt véglegesen törölné a program kér egy megerősítést a felhasználótól. Lehetőség van az aktuális, képernyőn látható kapcsolati adatok mentésére is. Ekkor egy nevet kell megadni a kapcsolatnak, ami nem lehet üres, valamint két különböző kapcsolat nem szerepelhet ugyanazon névvel, ennek ellenére felül lehet írni jelenleg létező kapcsolatot ha módosítani kívánjuk. Az adatok helyes megadása után, illetve ha létrejön sikeresen a kapcsolat az adatbázissal, akkor a program beléptet a fő ablakhoz. Hiba esetén kiírja a hibakódot, ezekről részletesebben 2.3-as szakaszban olvashat.

2.2. A kapcsolódási ablaknál előforduló hibák



2.2. ábra. Kapcsolódási hibára példa

Kapcsolódás közben több hiba is előfordulhat, csakhogy ezeket az adatbázis szerver kezeli, nem a DiagramQuery. Néhány gyakori hibát megemlítenék, ami hasznos lehet a program első használata során. A következő hibakódok fordulhatnak elő a leggyakrabban:

- **ORA-12560**: TNS:protocol adapter error; Unable to logon. Ezt okozhatja internetkapcsolat hiánya, vagy hiányzó szervernév, azaz ha nem lehet kiépíteni kapcsolatot a szerverrel bármilyen okból.
- **ORA-01005**: null password given; logon denied; Unable to logon. Ezt a hibaüzenetet akkor fogja kapni, ha a jelszó mezőt üresen hagyja. (Ha üres a jelszó mező akkor hibás felhasználónév esetén is ez a hibaüzenet jön elő)
- **ORA-01017**: invalid username/password; logon denied; Unable to logon. Hibás felhasználónév vagy jelszó esetén jelenik meg ez a hibaüzenet.

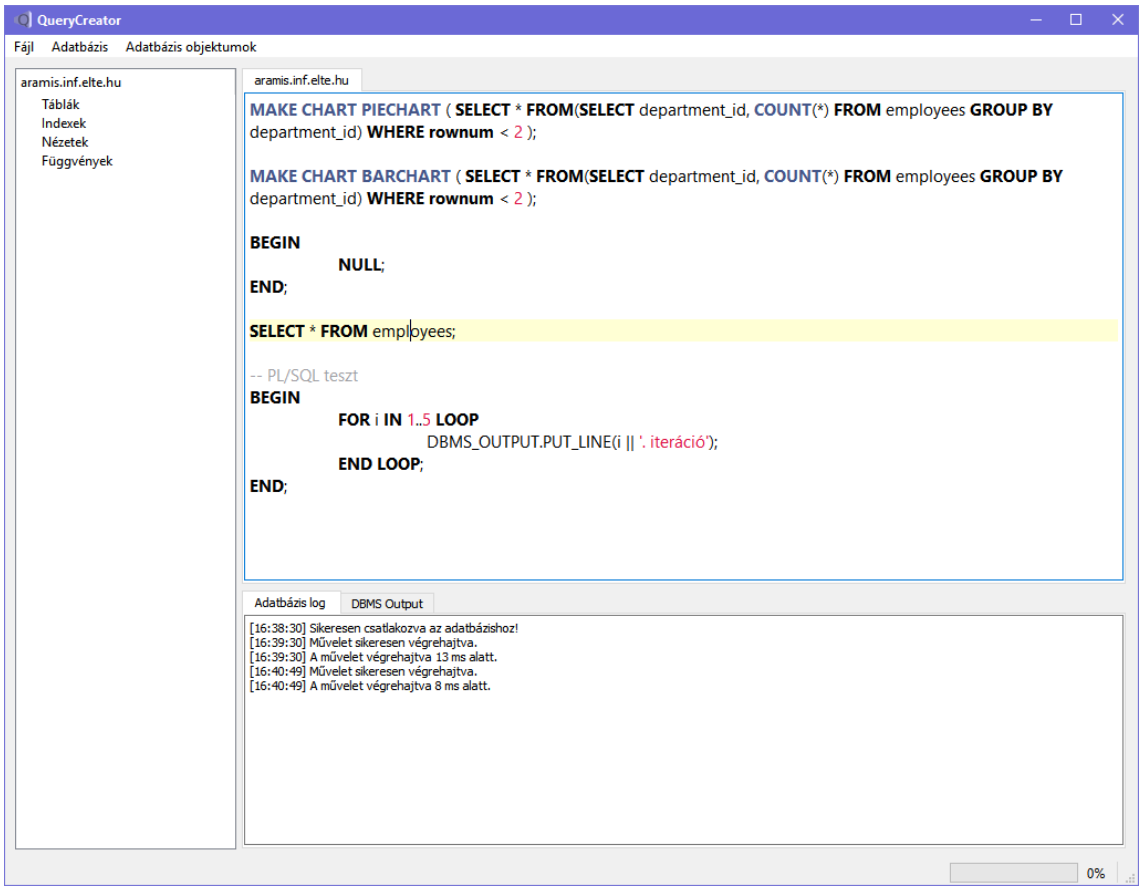
Ezek a hibakódok természetesen más esetben is előjöhetnek, ha biztos benne hogy minden rendben van az adatokkal, és a kapcsolattal, akkor kérjen meg egy hozzáértőt a probléma elhárításában (mivel ebben az esetben nagy valószínűséggel magával az adatbázis szerverrel van probléma).

Az Oracle Docs[**?**] oldalán megtalálható minden hibakód, és a hibák okai (angolul). Legtöbb esetben elég beszédesek, így meg lehet őket érteni dokumentáció nélkül is, valamint ha magyar nyelvű *Oracle Instantclient*-et tölt le, abban az esetben magyarul jelennek meg a hibaüzenetek magyarázatai. A későbbiekben is ezen hibakódok vannak használva (és logolva), így ezt az oldalt érdemes a későbbiekben is látogatni.

A továbbiakban a kapcsolódási ablak saját hibaüzeneteit részletezném:

- **Kapcsolat törlése sikertelen!**: Ez akkor fordulhat elő, ha nincs joga ahhoz a fájlhoz ami ezt a kapcsolatot tartalmazza, vagy abban az esetben ha futás közben már törölve lett.
- **Kérem ellenőrizze jogosultságát a program könyvtárához.**: A program a Connections nevű mappát próbálja létrehozni abban az esetben, ha nem létezik, és ez a hibaüzenet fog megjelenni akkor, ha ez a kísérlet meghiúsul.
- **Kérem adjon meg egy nevet, amivel később majd el lehet érni a kapcsolatot!**: akkor jelenik meg, hogyha üres szöveget próbált megadni a kapcsolat nevének.
- **Fájl megnyitása sikertelen!**: akkor jelenik meg, ha mentés közben a program nem fér hozzá a Connections/<**kapcsolatneve**>.xml fájlhoz.

2.3. Grafikus felület



2.3. ábra. Felhasználói felület

Sikeres belépés esetén egy ehhez hasonló képernyő fogadja a felhasználókat. A következő billentyűparancsok használhatóak az ablakon belül:

- **CTRL+W, illetve CTRL+SHIFT+W**: bezárja alsó, illetve felső lapot. Az alapértelmezett lapokat nem lehet bezárni.
- **CTRL+ENTER**: A szerkesztőben lévő lekérdezést vagy szkriptet végrehajtja kurzor helyzetétől függően.
- **F4**: Lekérdezési terv létrehozása.
- **F3**: Kijelölt utasítás(ok) végrehajtása.
- **DELETE**: Adatbázis objektum törlése a bal oldali fából.

Ezeket a parancsokat nem csak billentyűkombinációk révén lehet elérni, hanem kattintással, illetve menüből is elérhetőek.

A menüben lévő parancsokról röviden:

- **Fájl** → **Új** - Új lapot hoz létre a program, azaz töröl minden szöveget a jelenlegi szerkesztőből. Mielőtt ezt megtenné, természetesen figyelmezteti erről a felhasználót. CTRL + N billentyűkombinációval is elérhető.

- **Fájl** → **Megnyitás...** - Törli a jelenlegi lap tartalmát és betölti a kiválasztott fájlt. Hiba vagy visszalépés esetén megmarad a korábban írt lap tartalma. CTRL + O billentyűkombinációval is elérhető.

- **Fájl** → **Mentés másként...** - Elmenti a jelenlegi lap tartalmát. Hiba esetén jelez, és nem történik mentés. CTRL + S billentyűkombinációval is elérhető.

- **Fájl** → **Kilépés** - Kilépés a programból. A nem mentett munka elveszik, a program nem figyelmeztet nem mentett tranzakciókról, se a szerkesztőbe írt, de nem mentett adatokról, azonban megerősítést kér, mielőtt kilép. CTRL + Q billentyűkombinációval is elérhető.

- **Szerkesztés** → **Parancs futtatása** - Ugyan az a hatás érhető el vele, mint a *CTRL+ENTER* billentyűkombinációval.

- **Szerkesztés** → **Kijelölés futtatása** - *F3*-hoz hasonlóan végrehajta a kijelölt utasítást.

- **Szerkesztés** → **Lekérdezési terv megjelenítése** - Nem különb a hatása, mint az *F4* megnyomásának: a kurzor helyén lévő lekérdezéshez, vagy a kijelölt lekérdezéshez készít lekérdezési tervet, majd jeleníti meg azt.

- **Adatbázis** → **Újrakapcsolódás...** - Megpróbál új kapcsolatot kiépíteni az adatbázis szerverrel. Az újrakapcsolódáshoz meg kell adnia ismét a jelszavát.

- **Adatbázis** → **Új kapcsolat** - Bontja a jelenlegi kapcsolatot a felhasználó beleegyezése után, és nyit egy új kapcsolódási ablakot.

- **Adatbázis objektumok** → **Megtekintés** - Az adatbázis objektumok fájában jelenleg kijelölt objektum tulajdonságairól megnyit egy új lapot.

- **Adatbázis objektumok** → **Törlés** - A jelenleg kijelölt adatbázis objektumot törli beleegyezés után.

Itt még egy dologra felhívnám a figyelmet: Ha időtúllépés miatt kiléptet az adatbázis (ezt az adatbázis szerver beállításaitól függ mennyi idő), akkor érdemes az újrakapcsolódást választani, mivel ha lekérdezést próbál végrehajtani, akkor az adatbázis sokáig nem fog válaszolni.

Az ablak bal oldalán található az adatbázis objektumok fája. Dupla kattintás esetén betöltődnek ezek az objektumok, majd ezután lehetőség nyílik ezek megtekintésére (dupla kattintás), illetve törlésére (*delete* billentyű). Ismételt dupla kattintásra frissítésre kerülnek az adatok. Táblák megtekintése esetén előjön az attribútumainak neve, típusa, hossza, illetve hogy lehet-e az adott attribútum null. Indexeknél megtekinthető az index típusa, az hogy melyik táblához készült az index, illetve ennek a táblának a tulajdonosa, továbbá az hogy egyedi-e az index. Nézeteknél meg lehet nézni a lekérdezést amivel a nézet létre lett hozva, a nézet típusát, illetve hogy csak olvasható-e az adott nézet. Függvényeknél a függvény neve, id-ja, illetve létrehozási dátuma tekinthető meg. Ezt a fát igény esetén el is lehet tüntetni, és akármikor vissza is lehet hozni. Ez hasznos lehet zavartalan íráshoz, vagy diagramok nagy méretben való megtekintésére.

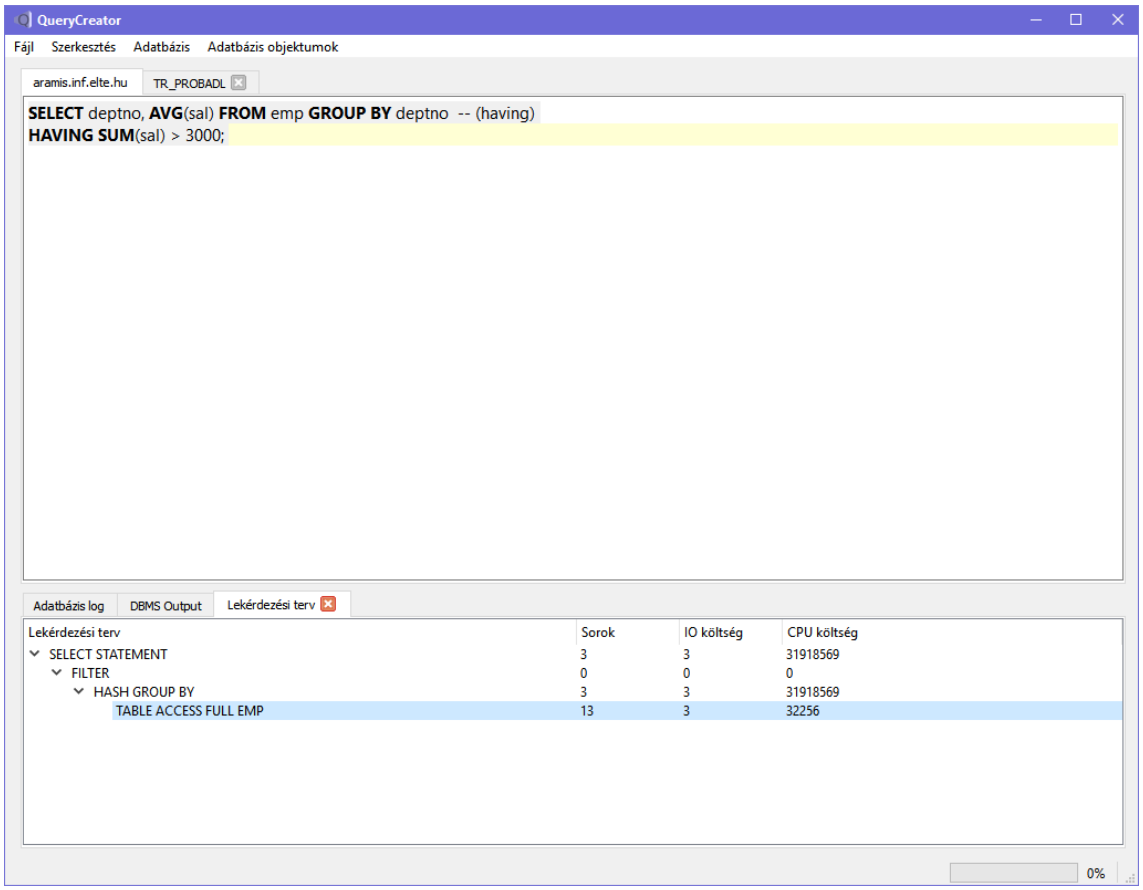
A jobb alsó sarokban találhatóak az Adatbázis logja illetve kimenete, ezen felül ide kerülnek a végrehajtott lekérdezések, és végrehajtási tervek is. Az adatbázis minden sikertelen műveletet logol a hiba okával együtt, illetve a sikeres műveleteket is, a végrehajtáshoz szükséges idővel együtt. Ez a log (ha sikeresen létrehozható) megtalálható lesz később a háttértáron is, *Logs* mappában *ev_honap_nap.log* formában. Ha nem lépünk ki a programból a nap váltása előtt, akkor előző napi loghoz fog írni. Az adatbázis kimenete minden esetben működik, nincs szükség külön parancs beírására hogy megjelenítsük azt. Mindemellett ez a komponens is eltüntethető, abban az esetben ha a felhasználó számára éppen nem szükséges.

2.3.1. Lekérdezések

A program szerkesztő felületében lehetőség van SQL lekérdezések, illetve PL/SQL szkriptek végrehajtására. SQL* Plus szkriptek viszont nem működnek, így olyan utasításokat mint a **set serveroutput on** nem hajthatunk végre, csakhogy erre nincs is szükség, az adatbázis kimenet alapértelmezetten használható. A programban szintaxis kiemeléssel jelennek meg ezek az utasítások, így tanulás közben is használható. Azonban szemantikus elemzés és automatikus kitöltés nincsen a programban. Az SQL parancsokat érdemes pontosvesszővel elválasztani, ennek ellenére nem kötelező, pontosvessző hiányában is végrehajthatóak. Nincs szükség PL/SQL parancsok lezárására '/' karakterrel mint SQLDeveloper esetén.

2.3.2. Diagramok

Két különböző típusú diagram lett implementálva a programban, a kör- és oszlopdiagram. A továbbiakban ezeknek a szintaxisát részletezem, illetve leírom mikre használhatóak, mivel ezek különböző adatok reprezentációjára alkalmasak.



2.4. ábra. Lekérdezési terv megtekintése

Kördiagram

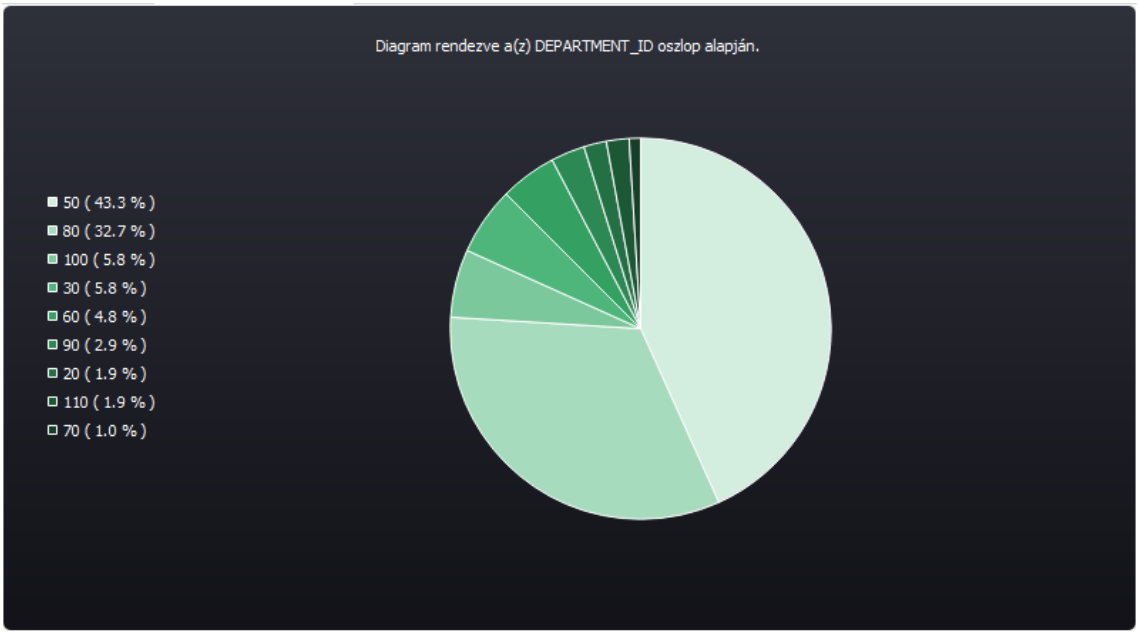
A kördiagram kategorikus adatok megjelenítésére használható, azaz egy adott mennyiség megoszlásának arányát lehet vele ábrázolni. Erre láthatunk egy példát a 2.5-ös ábrán. A kört akkora részekre osztjuk, amekkora részt az adott mennyiségek kitesznek az egészből. Akkor érdemes ezt a fajta diagramot használni, ha egyes részeknek az egészhez való viszonyát kívánjuk megvizsgálni. Lehetőség lenne egy-egy körcikket kirobbantani, azonban ez zavarólag hathat, így erre jelenleg nincs lehetőség a programban. Többet olvashat erről a Számítógépes adatfeldolgozás[**?**] című könyvben, ami az interneten is elérhető.

Szintaxisa a következő:

MAKE CHART PIECHART (<**SQL lekérdezés**>);

Az SQL lekérdezésnek ezeket a feltételeket kell teljesítenie:

- A lekérdezés nem lehet üres.
- Legalább kettő, legfeljebb húsz rekordos lehet a lekérdezés.
- Két attribútumosnak kell lennie a lekérdezésnek.

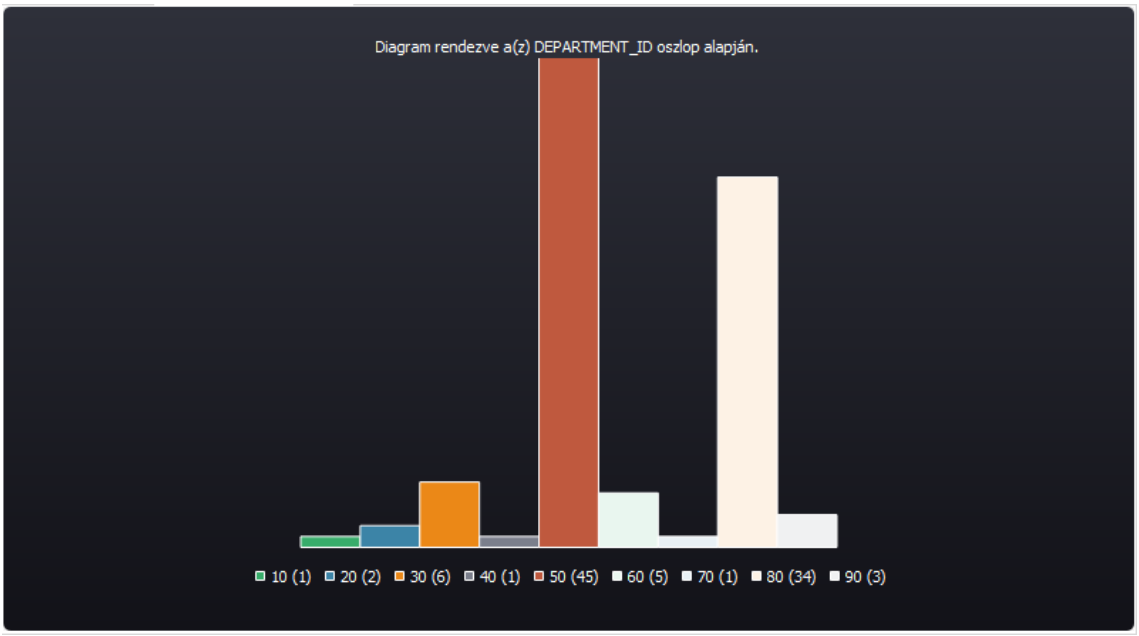


2.5. ábra. Kördiagram

- Második attribútuma lehet egy aggregáló függvény eredménye, vagy egy természetes szám. Negatív szám nem használható, hiszen ebben a kontextusban értelmetlen.
- Érdemes csökkenő sorrendben lekérni az elemeket, de ez teljes mértékben a felhasználóra van bízva.
- Felesleges attribútum nem szerepelhet a lekérdezésben, azaz ha kettőnél többet tartalmaz, akkor hibát fog adni a program.

A címkékben százalékosan jelennek meg az adatok, így érdemes lehet úgy lekérni az adatokat, hogy ne történjen hiba a kerekítés során.

Oszlopdiagram



2.6. ábra. Oszlopdiagram

Az oszlopdiagram különböző, ennek ellenére egynemű, időszakosan változó adatok megjelenítésére használható. Erről a típusról láthat egy példát a 2.6-os ábrán. Ez lehet például egy bolt napi nyeresége, ami természetesen nem feltétlen pozitív minden nap, így szükség lehet negatív számok megjelenítése is. Az adatok változása így könnyen leolvasható, és összehasonlítható lesz. Erről a diagramtípusról bővebben olvashat Nyesőné Marton Mária könyvében[**?**].

Szintaxisa a következő:

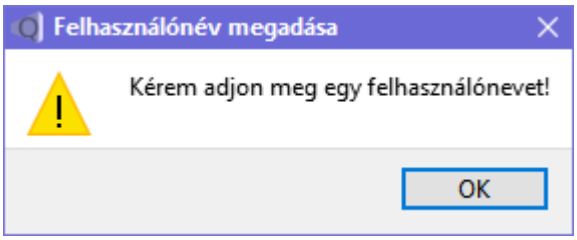
MAKE CHART BARCHART (<**SQL lekérdezés**>);

Az SQL lekérdezésnek ezeket a feltételeket kell teljesítenie:

- A lekérdezés nem lehet üres.
- Legalább kettő, legfeljebb tíz rekordot tartalmazhat.
- Két attribútumosnak kell lennie a lekérdezésnek.
- Második attribútuma lehet egy aggregáló függvény eredménye, vagy egy egész szám.
- Felesleges attribútum nem szerepelhet a lekérdezésben, azaz ha kettőnél többet tartalmaz hibát fog adni a program.

Érdemes eredeti sorrendben meghagyni az adatokat, hogy könnyen megfigyelhessük a kiugró adatokat.

2.4. Hibák a fő ablaknál



2.7. ábra. Újrakapcsolódás közben fellépő hibára példa

Számtalan hiba fordulhat elő fő ablak használata közben, ennek ellenére a felugró hibák száma elenyésző. Röviden most ezeket, és ezeknek a magyarázatát sorolnám fel:

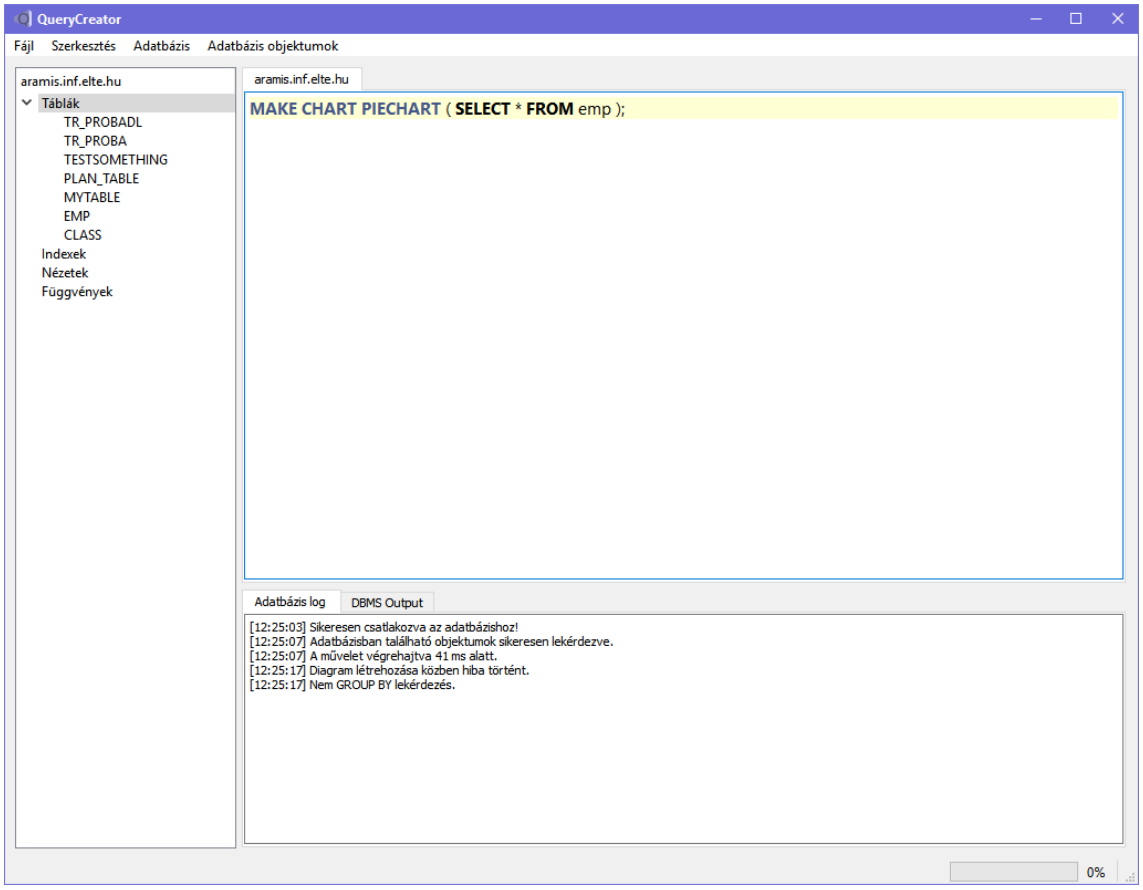
- **Kérem adjon meg egy felhasználónevet!**: akkor jelenik meg, ha újracsatlakozásnál nem adott meg egy felhasználónevet se. A program automatikusan beírja a jelenleg kapcsolódott felhasználót, csakhogy lehetőség van más felhasználóval való belépésre is.
- **Kérem adja meg jelszavát!**: abban az esetben jelenik meg, ha újrakapcsolódásnál nem adott meg jelszót.
- **Sikertelen kapcsolódás**: akkor, ha valamilyen okból (internetkapcsolat megszűnt, hibás felhasználónév/jelszó páros) nem sikerült újra kiépíteni a kapcsolatot az adatbázissal. Egy hibaüzenet is megjelenik, ami a korábban említett Oracle oldalán[?] megtekinthető.

Az utasítások futtatása során azonban ennél jóval többféle hibaüzenet is megjelenhet. Az alábbi üzenetek az adatbázis logjában találhatóak meg, ezek később is visszanézhetőek. Ezeknek a nagy részét lefedi az Oracle oldala[?], csakhogy vannak a programnak saját hibaüzenetei is, ezeket részletezném:

- **Kérem hozza létre a PLAN_TABLE táblát ezen parancs használata előtt!** - a lekérdezési terv létrehozásához szükséges egy PLAN_TABLE nevű tábla, erről bővebben az Oracle dokumentációjában[?] olvashat.
- **Hiba: üres lekérdezést próbált végrehajtani!** - ha kijelölést próbál végrehajtani, de nincs semmi kijelölve, akkor kapja ezt a hibaüzenetet. Illetve ha nincs semmi a szerkesztőablakba, és akkor próbál egy utasítást végrehajtani.

- **Nem GROUP BY lekérdezés.** - ez a hibaüzenet akkor jelenik meg, ha megpróbált egy diagramout létrehozni olyan lekérdezéssel, amelyben nem szerepel **group by**.

- **A diagram működéséhez 2 oszlop szükséges.** - akkor jelenik ez meg ha kettőnél kevesebb, vagy több attribútumu lekérdezéssel próbál egy diagramot létrehozni.



2.8. ábra. Diagram létrehozása közben fellépő hiba

- **Második oszlopnak számot kell tartalmaznia!** - ha a felhasználó nem aggregációs lekérdezést használ, vagy olyan oszlopot ad meg második attribútumként ami nem egy szám, akkor ez a hibaüzenet fogadja.

- **Kérem kördiagramot csak pozitív számokkal használjon!** - mivel a kördiagram adatok egymáshoz való viszonyítását szemlélteti, és az egyes adatok százalékban jelennek meg az egészhez viszonyítva, így nem lenne értelme negatív számokkal használni azt. Ez a hibaüzenet erre a hibára figyelmeztet.

- **Nem támogatott diagram típus: ¡diagramtípus¿** - ez az üzenet akkor jelenik meg ha egy nem támogatott diagramtípust próbál létrehozni. Jelenleg támogatott típusok: piechart, barchart.
- **Kérem ezt a diagramtípust kevesebb rekorddal használja.** - abban az esetben ha túl sok rekorddal rendelkező lekérdezéssel próbál egy diagramot készíteni, akkor fog ezzel a hibaüzenettel találkozni. Kördiagram esetén húsz, míg oszlopdiagram esetén tíz a maximális rekordszám.
- **Üres lekérdezés.** - amennyiben egy rekorddal se rendelkezik a lekérdezés amivel a diagramot próbálja létrehozni, akkor ez az üzenet fogadja.

Ezeknek a hibaüzeneteknek nagy része elkerülhető ha körültekintően hozza létre a diagramokat, illetve hajtja végre a lekérdezéseket. Mielőtt használná a programot érdemes elolvasni a 2.3.2-es szakaszban lévő leírást a diagramok használatáról, illetve hasznukról.

DiagramQuery arról is figyelmeztet, hogy ha egy törlés / megtekintés / lekérdezés nem sikerül, és ezen felül az Oracle saját hibakódját is megjeleníti, így pontosan meg lehet tudni milyen hiba történt az utasítás végrehajtása során.

1. beállításaitól
2. kapcsolat
3. given
4. magyar

3. fejezet

Fejlesztői dokumentáció

Egy integrált fejlesztői környezetnél többféle igény is megfogalmazható. Valamelyik fejlesztőnek olyanra van szüksége, amely ugyan kevesebbet tud, de gyorsabban indítható, esetlegesen gyorsabban végezhető vele a munka. SQL-hez található több integrált fejlesztői környezet, ilyen az Oracle-nél például az SQL Developer[**?**]. Az én programom az SQL Developerhez képest kevesebb komponenst tartalmaz, valamint nem Javaban, hanem C++ban lett írva, így gyorsabb is. Tökéletes eszköz, ha valaki csak a lekérdezésekre próbál koncentrálni.

A megvalósított programban lehetőség van...

- Oracle adatbázishoz való kapcsolódáshoz,
- kapcsolat adatainak mentésére,
- ezen kapcsolati adatok törlésére,
- PL/SQL szkriptek végrehajtására,
- SQL lekérdezések végrehajtására,
- adatbázis objektumok megtekintésére, törlésére,
- lekérdezési tervek létrehozására,
- kör, illetve oszlopdiagramok készítésére.

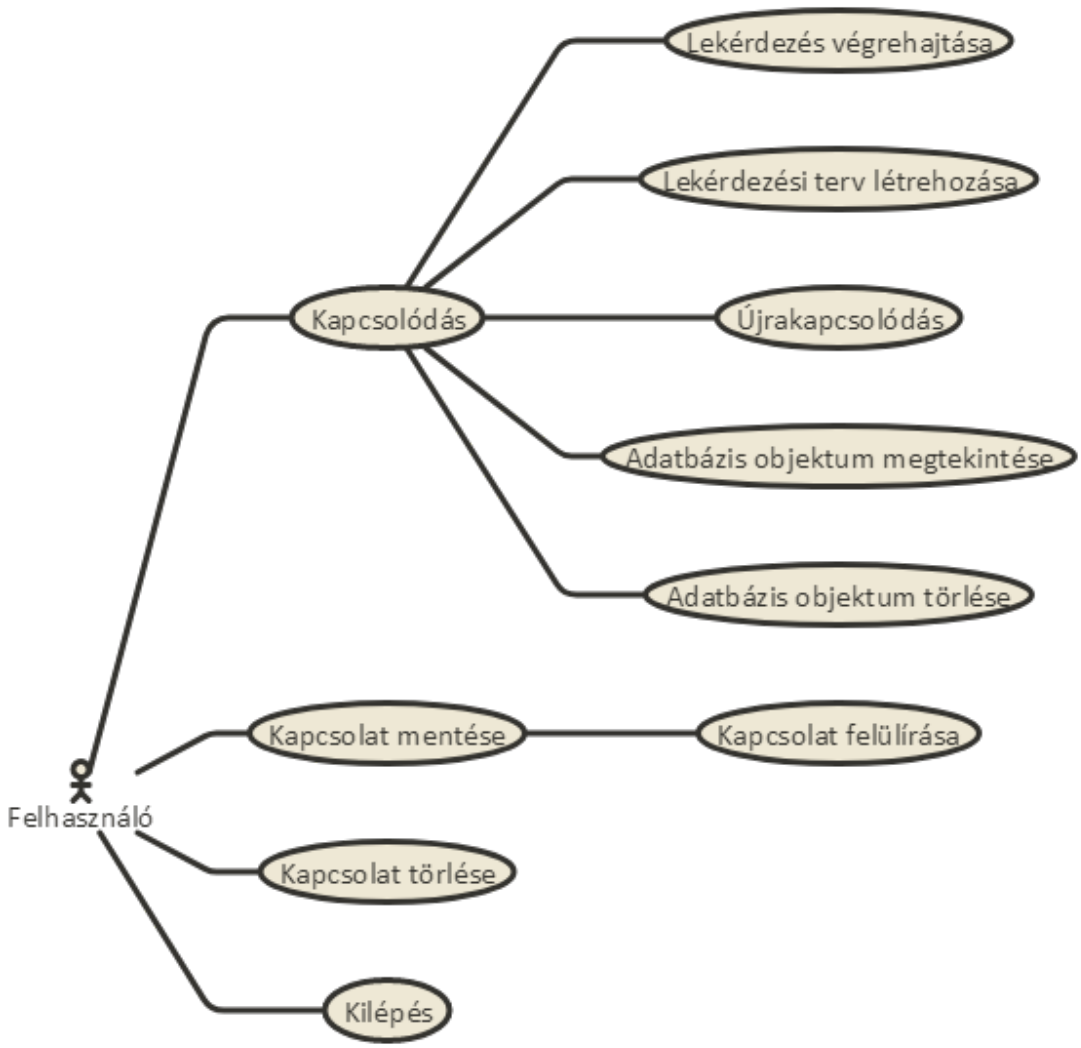
A fejlesztéshez szükséges eszközök kiválasztásnál tehát szempont volt a hatékonyság, valamint hogy ennek ellenére többet nyújtson az eddig elérhető programoknál, ezért esett a választásom a Qt-ra, ami egy C++ keretrendszer. Nem csak hatékony kódot lehet vele készíteni, szükség esetén magas absztrakció is használható, és nem csak egy operációs rendszeren lehet a Qt-ban készített programokat futtatni. Ezen felül elvárás hogy legyen intuitív kezelőfelülete, és biztonságosan működjön.

3.1. Fogalmak

Itt részleteznék néhány fogalmat amivel jobb tisztában lenni mielőtt tovább olvasná a dokumentációt.

SQL egy lekérdezésekre megfogalmazására specializálódott nyelv, azaz egy szakterület-specifikus nyelv. PL/SQL az Oracle által az SQL kiterjesztéseként kifejlesztett procedurális programozási nyelv. Továbbá egy host nyelv, amin belül bizonyos kódrészleteket SQL segítségével írhatunk meg. Ezen fogalmak mélyebb ismerete ajánlott a továbbiakban. Névtelen függvény egy helyben létrehozott és (akár) meghívott függvény.

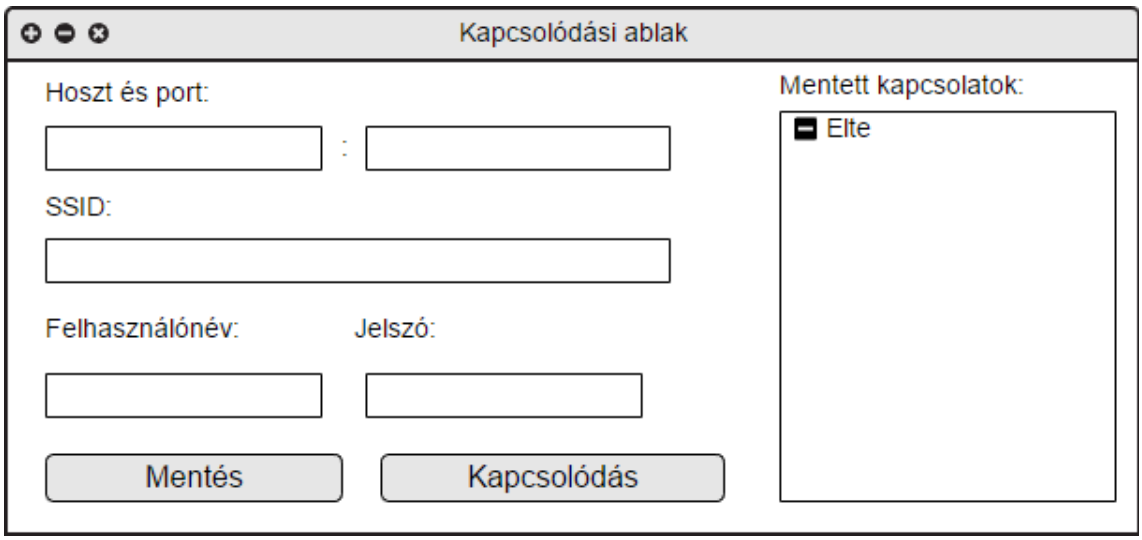
3.2. Megoldási terv



3.1. ábra. Felhasználói esetek

Felhasználóknak ezen felül lehetősége nyílhat más funkciók használatára is, ám ezek a legfontosabbak, így ezeknek az implementálása az elsődleges. Ezeknek az implementálása két külön ablakban történt a gyorsabb elérhetőség, és a modularitás miatt: így bármikor lehetőség nyílik új ablak nyitására, esetleg ha valaki akar felhasználhatja a programot, és kódbol csatlakozhat az adatbázishoz. Biztonsági szempontból ez nem ajánlatos, de ez is egy lehetőség ha nincs a számítógépen biztonsági kockázat (pl. belső hálózatról működik csak, illetéktelen felhasználó nem férhet hozzá).

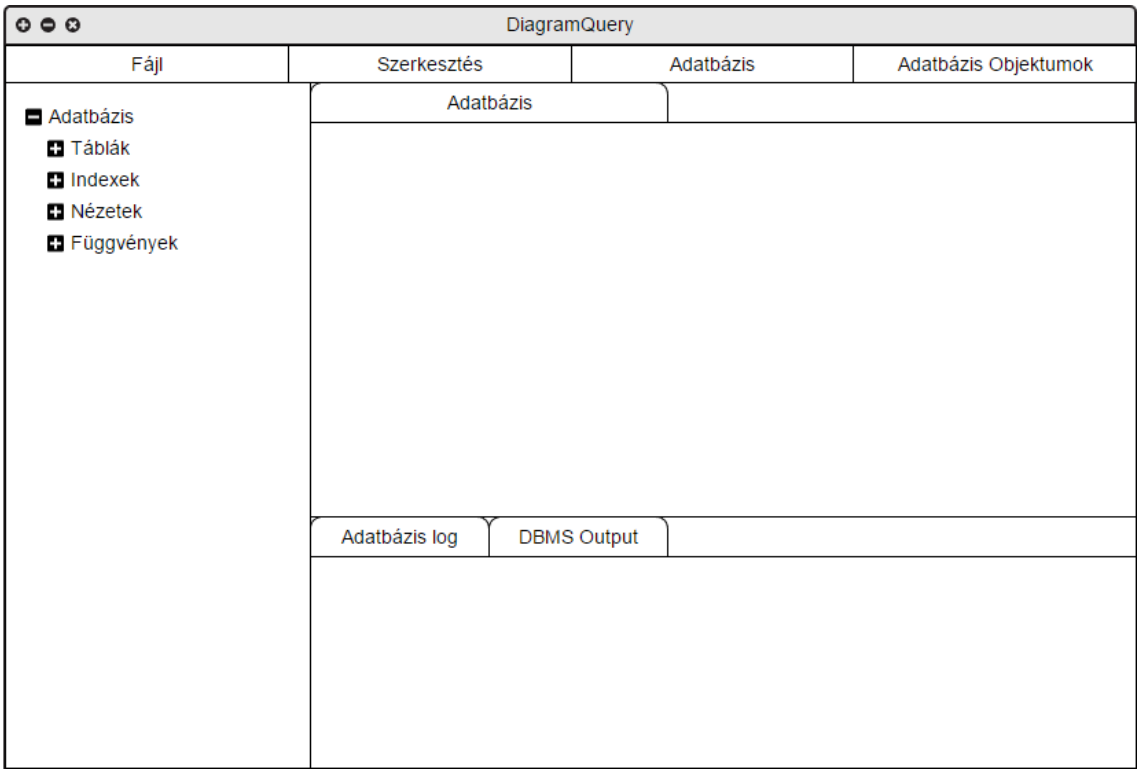
3.2.1. Felhasználói felület terve



3.2. ábra. Kapcsolódási ablak terve

A kapcsolódási ablaknál szempont volt az egyszerűség és könnyen kezelhetőség. Első használat során már el lehet igazodni rajta bárkinek. Implementálás során fontos megjegyezni néhány szempontot amit érdemes figyelmbe venni: a tabok sorrendje legyen meghatározott - balról jobbra, fentről lefele haladjon -, valamint a jelszó mező tartalmát ne lehessen látni. Továbbá adatbázishoz kapcsolódás esetén legyen minél hamarabb eltűntetve a jelszó a memóriából. (Megjegyzés: C++ esetén vigyázni kell az optimalizálásra, fordítótol függően könnyen lehet hogy a nem használt értékadást eltűnteti a fordító) A kapcsolatokat lehessen szerkeszteni, azaz lehessen felülírni. Erről jelenjen meg egy figyelmeztetés is, hogy véletlen se írja felül a felhasználó a korábbi kapcsolatát. Ezen felül ha bármilyen hiba (nem lehet fájlt/mappát megnyitni, vagy létrehozni) történik, akkor azt jelezze a program ennek megfelelően. Kapcsolat törlése esetén is kérjen megerősítést a program.

A kapcsolódás után fogadja a felhasználókat a fő ablak. Legfontosabb itt, hogy a felhasználó rögtön munkához tudjon látni. Érdemes úgy készíteni, hogy a képernyő

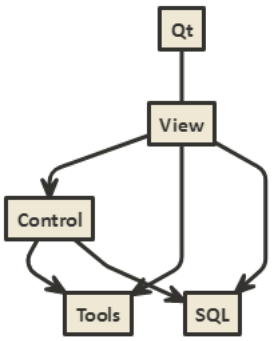


3.3. ábra. Fő ablak terve

nagy részét a szerkesztő tegye ki. Bal oldalon egy fában jelenjenek meg az adatbázis objektumai (táblák, nézetek, függvények, indexek) és lehessen megtekinteni illetve törölni őket. A képernyőn jelenjen meg egy logban a futási idő, illetve ha hiba történik, az legyen szemmel látható. PL/SQL szkriptek miatt érdemes egy külön lapon megjeleníteni a kiírt üzeneteket is. Azonkívül legyen lehetőség a szerkesztő törlésére, mentésére, valamint már meglévő .sql kiterjesztésű fájl betöltésére is. A program adjon figyelmeztetést törlés esetén. Mindemelett lehessen újra kapcsolódni az adatbázishoz arra az esetre, ha megszakadna a kapcsolat az adatbázissal, valamint lehessen új kapcsolatot is nyitni, ami zárja be a jelenlegit. Legyen lehetőség az adatbázis objektumok megtekintésére és törlésére is. A szerkesztőablakba beírt SQL lekérdezések, PL/SQL szkriptek, illetve egyedi parancsok (diagramok készítésére) hajtódjanak végre megfelelően, és legyen látható pontosan mit hajtott végre a program. A szerkesztő betűtípusa legyen jól olvasható, a tabulátor mérete alapértelmezetten legyen 4, illetve az SQL és PL/SQL kulcsszavak legyenek kiemelve.

3.2.2. Komponensek

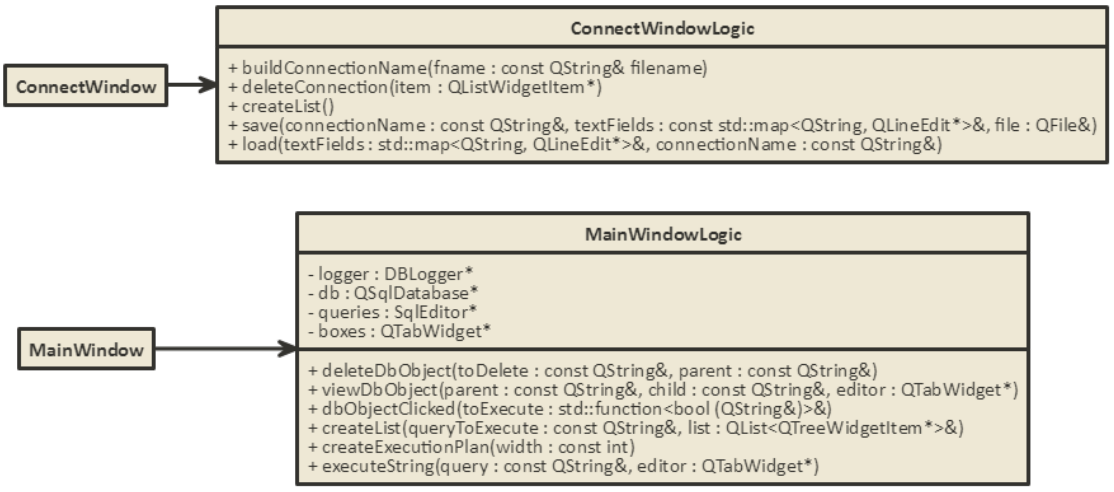
A program négy fő komponensből áll: Control, View, SQL, Tools. Mindegyik komponense a programnak külön felhasználható, valamint bővíthetőek. Az SQL komponens



3.4. ábra. Komponensek

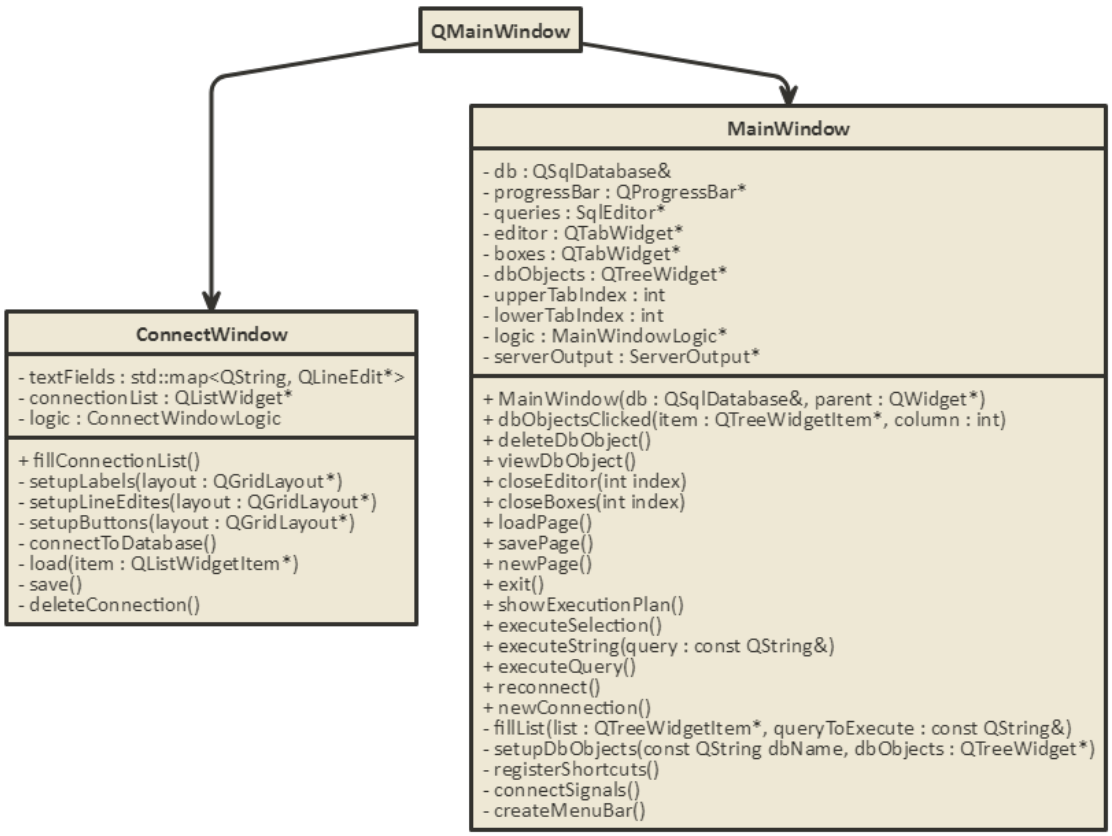
azonban főként Oracle SQL-hez lett írva, így nem feltétlen működik minden SQL variánsal, de ez később bővíthető. Az osztályokon kívűl egyéb függvényeket is tartalmaznak ezen komponensek, valamint konstansokat is: ezek a jobb érthetőség és átláthatóság miatt kerültek a programba.

Control



3.5. ábra. Kapcsolódási- és főablak logikájának diagramja

Ebben a komponensben található a program logikája. Pontosabban itt, ebben a rétegben kerülnek a diagramok előállításra, valamint itt kerülnek végrehajtásra a különböző lekérdezések. Miután a program végrehajtott egy utasítást, azután logolásra kerül a végrehajtás sikerettségét jelző üzenet, valamint a hibaüzenet is sikertelen futás esetén.



3.6. ábra. Kapcsolódási- és főablak diagramja

View

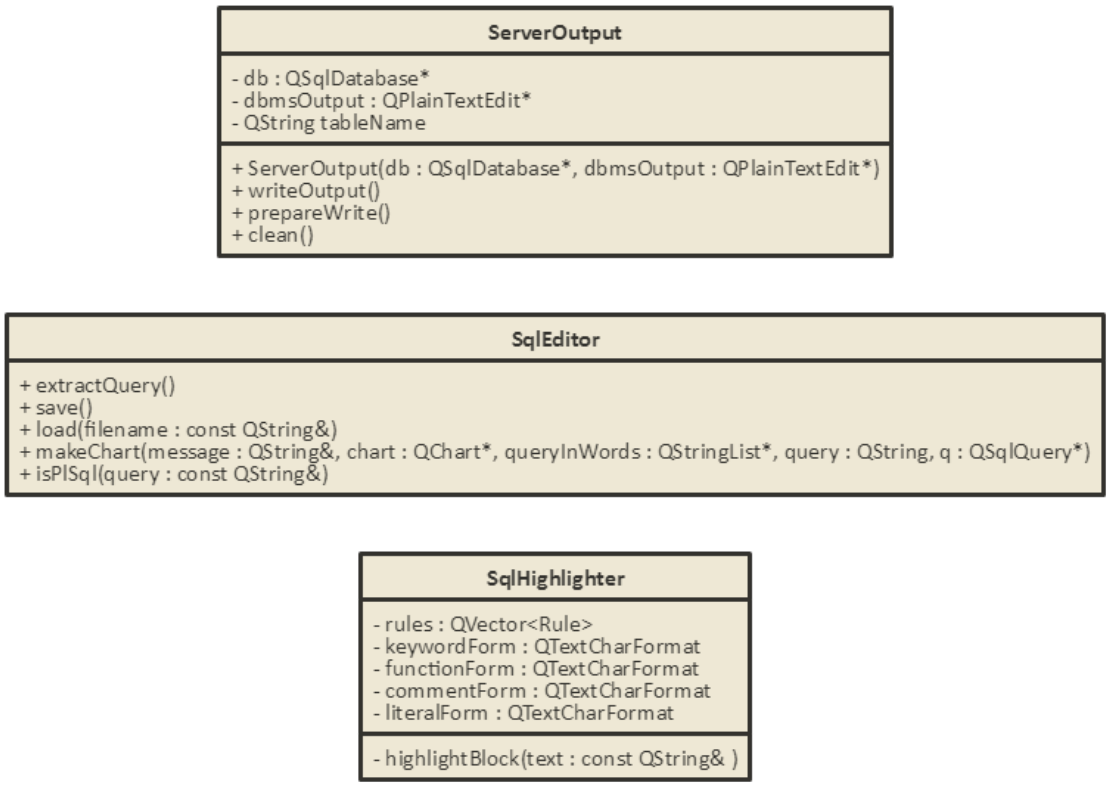
A View komponens felel a program kinézetéért. Itt van pontosan elrendezve minden nézetbeli elem, és itt vannak az események is lekezelve. Fontos hogy jól érthető hibaüzeneteket jelenítsen meg ha valami nem rendeltetésszerűen történik, és ezt a felhasználó egyből észrevegye. Későbbi szekcióban megtalálható a pontos terve a felhasználói felületnek, az alapján készítsük ezt el. A két legfontosabb függvény amit kiemelnék a *ConnectWindow::connectToDatabase()* és a *MainWindow::reconnect()*.

ConnectWindow::connectToDatabase() metódus végzi el az adatbázishoz való kapcsolódást. Szüksége lesz az adatbázis, és a felhasználó adataira. Az adatbázis adatai úgy mint a hoszt, port, serviceid (később esetlegesen a driver ami szükséges az adatbázishoz való kapcsolódáshoz). Valamint a felhasználó adatai úgy mint a neve és jelszava. A függvény írja ki ha hiba történik belépés során, illetve ha nem akkor csatlakozzon az adatbázishoz és törölje a felhasználó jelszavát. Továbbá ez a metódus köti össze a kapcsolódási és főablakot.

A MainWindow::reconnect() metódus felelős a kapcsolat újboli kiépítésére. Bemenő adatai a felhasználó neve és jelszava. Abban az esetben ha a felhasználó hibás adatokat ad meg, természetesen nem dobja el a régi kapcsolatot, hanem értesíti a

felhasználót a hibáról. Megfelelő adatok esetén belépteti a felhasználót, és folytathatja tovább eddigi munkáját. Erre legfőképp azért lehet szükség, mert bizonyos adatbázis szerverek (pl. Oracle) egy idő után kidobják a felhasználót. Ilyenkor az egész programot újra kéne indítani ha erre nem adnánk lehetőséget.

SQL



3.7. ábra. SQL komponens osztályainak diagramja

Az SQL komponens végzi el a nyelvhez kapcsolódó feladatokat: szerkesztőfelületet kialakítása, szintaxis kiemelése, a lekérdezések és szkriptek megfelelő kiszedése. A cél az, hogy akár önmagában is használható legyen (feltéve, ha van egy adatbázis biztosítva hozzá), és minden SQL művelet elvégezhető legyen a segítségével.

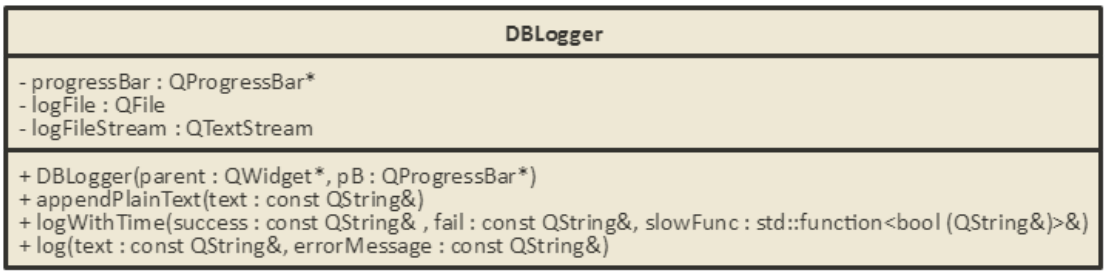
SQLHighlighter osztály reguláris nyelvtannal végzi el a kapott szöveg elemzését. Az összes SQL és PL/SQL kulcsszavak[**?**] kiemelését elvégzi, továbbá a kommenteket, szám- és szövegliterálokat is másképp emeli ki a könnyebb érthetőség kedvéért. Ennek az osztálynak egy metódusa van, a *highlightBlock()* aminek a bemenete egy szövegliterál. Ezt a bemeneti szöveget elemzi a metódus, majd reguláris kifejezések segítségével kiemeli a kulcsszavakat, és literálokat a szabályoknak megfelelően.

ServerOutput osztály a DBMS_OUTPUT kiírására szolgál. Egy temporáris táblába tárolja el a kiírt sorokat amit a bufferből olvas, majd ezt kérésre kiírja,

és törli. Külön osztály azért szükséges ehhez, hogy kilépés esetén törölje maga után a létrehozott temporáris objektumokat a program a RAII elvei szerint. Egyetlen fontos metódusa a *writeOutput()*, aminek nincs bemenete és kimenete, csupán kiírja a megfelelő helyre a buffer tartalmát.

Az SQLEditor osztály valósítja meg a fő szerkesztőfelületét az osztálynak. Ezen felül ez az osztály hozza létre a diagramokat, és kezeli le az esetlegesen hibás bemeneteket. A szerkesztéssel kapcsolatos logika is itt található: itt történik a végrehajtandó utasítások kiemelése (persze a végrehajtásuk nem ebben az osztályban történik), a diagramok feldolgozása és kirajzolása, valamint a lapok mentése és betöltése is. Annak érdekében hogy jobban hasonlítson egy szerkesztőhöz, a jelenleg aktív sorokat is kiemeli, valamint ha futtatunk egy utasítást, vagy utasításokat, akkor azokat is kiemeli hogy a felhasználó tisztában legyen azzal, hogy a program mit hajtott végre.

Tools



3.8. ábra. Tools komponens osztályának diagramja

Ebben a komponensben a két osztályon felül egyéb eljárások illetve konstansok kerülnek eltárolásra.

A DBLogger osztály fontos szerepet játszik, mivel annak segítségével fogjuk a végrehajtandó utasításokat végrehajtani, és így lemérni a futás idejüket. SQL lekérdezések készítésénél kritikus ennek a megléte, mivel így tudja a felhasználó javítani a lekérdezéseit szükség esetén. *DBLogger::logWithTime* metódusa fogja megoldani ezt, mégpedig egy függvény pointer segítségével. Szükséges bemenete a függvény amit a programban kivánunk futtatni, és visszaad egy üzenetet arról hogy a művelet sikeres avagy sikertelen volt-e, valamint egy hibaüzenetet sikertelen lefutás esetén.

3.3. Megvalósítás

3.3.1. Eszközválasztás

3.3.2. Control

ConnectWindowLogic

Függvények

buildConnectionName(QString&): Felépíti a kapcsolat nevét, azaz az átadott QString nevével létrehoz egy Connections/<string neve>.xml tartalmú QStringet majd visszaadja azt. A függvényben a QDir::separator() elválasztót használjuk, így rendszerfüggetlen elválasztót kapva.

deleteConnection(QListWidgetItem*): A fentebb említett függvény segítségével megépíti a kapcsolat nevét ahhoz az elemhez amit paraméterként kap. Ha sikerül a törlés akkor törli az elemet is, és igazat ad vissza, különben hamisat. Itt megemlíteném az elágazásban használt makrót: Q_LIKELY. Qt saját makrója, ami segít a branch predictingben, így hatékonyabb kódot generálva.

3.3.3. SQL eszközök

3.3.4. Felhasználói felület

3.3.5. Fejlesztői környezet

3.4. Továbbfejlesztési lehetőségek

3.5. Tesztelés

4. fejezet

Irodalomjegyzék

[1] Oracle: Error messages, https://docs.oracle.com/cd/B28359_01/server.111/b28278/toc.htm , megtekintés dátuma: 2017. március 03.

[2] Nyesőné Marton Mária: Számítógépes adatfeldolgozás, *Eszterházy Károly Főiskola*, 2011. 08. 31.

[3] Oracle: SQL Language Reference, https://docs.oracle.com/database/121/SQLRF/ , megtekintés dátuma: 2017. március 10.

[4] Oracle: SQL Developer, http://www.oracle.com/technetwork/developer-tools/sql-developer/ , megtekintés dátuma: 2017. április 12.

[5] Oracle: SQL reserved words, https://docs.oracle.com/database/121/SQLRF/ap_keywd001.htm , megtekintés dátuma: 2017. március 10.