Adatbázis tervezés

Informatika 2 – Hallgatói segédlet

1. Szükséges előismeretek

A gyakorlat során a hallgatók megismerkednek az adatbázisok logikai tervezése során használatos Egyed/Kapcsolat (E/K) diagramokkal, valamint az elkészült adatbázis terv relációs adatmodellé történő leképezésével. A gyakorlat során egy szöveges specifikációból kiindulva adatbázis tervet kell készíteni, a megtervezett adatbázist relációs sémává kell alakítani, és a sémát létre is kell hozni MySQL Serveren.

A gyakorlat elvégzéséhez a következő ismeretek szükségesek:

- Egyed/Kapcsolat diagramokon használt jelölések
- Egyed/Kapcsolat diagramok leképezése relációs adatmodellé
- SQL nyelv séma létrehozásához kapcsolódó része, valamint egyszerű lekérdezések és adatmanipulációk

A gyakorlat elvégzéséhez ajánlott ismeretek:

- MySQL workbench használata, letölthető a http://www.mysql.com/products/workbench/ oldalról
- PHPMyAdmin használata, része a gyakorlatokon használt XAMPP csomagnak, ez letölthető a http://www.apachefriends.org/en/xampp.html oldalról

2. Egyed kapcsolat diagramok

Az Egyed/Kapcsolat diagramok az adatbázisban található elemek logikai struktúrájának modellezésére szolgálnak. A modellezés során az alábbi elemek találhatók meg egy E/K diagramban:

- ***Egyed***: Azon objektumok modellezésére szolgál, melyről információt szeretnénk tárolni az adatbázisban.
- ***Attribútum***: Egyedekhez vagy kapcsolatokhoz tartozó leíró tulajdonság. Azaz az attribútumok jelenítik meg a tárolandó tulajdonságokat.
- ***Kapcsolat***: A kapcsolatok az egyedek egymáshoz való viszonyát írják le. Azaz egy egyed milyen, ill. hány másik egyeddel áll kapcsolatban.

Az E/K diagramok ábrázolására nincs egységesen elfogadott szabvány, a két legelterjedtebb jelölésrendszer a Chen-féle és a Crow's foot jelölésmód. A továbbiakban először a Chen-féle jelölésmódot ismertetjük, majd bemutatjuk, hogy miben különbözik ettől a tervezőeszközökben inkább alkalmazott Crow's foot jelölésmód.

2.1. Egyed

Az egyedeket tulajdonképpen az információ hordozóinak lehet tekinteni, azaz az egyedek azok a fogalmak/dolgok amikről információt szeretnénk tárolni. Egyedek közt megkülönböztetünk egyed típust és egyed példányt. Az egyed típus egy gyűjtő fogalom, azaz adott típusú egyedeket tárolunk az adatbázisban és az egyed példány az egyed típus egy konkrét előfordulása. Például: a személy az egy

egyedtípus és Kiss Béla egy konkrét egyedpéldány. Az E/K diagramokon egyed típusokat ábrázolunk. Az egyed típusokat a következőképpen szokás jelölni:

1. ábra Egyedek jelölése

Egyedek esetében megkülönböztetünk erős és gyenge entitásokat:

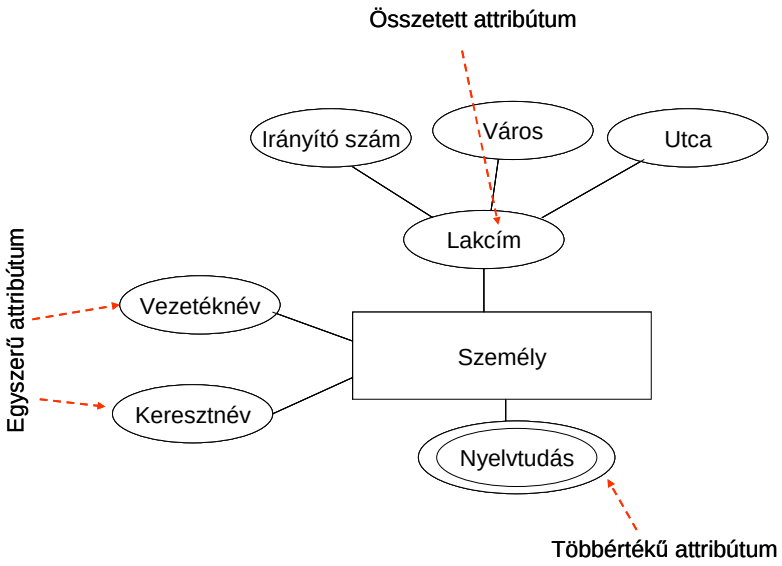
- ***Erős entitás***: egyedpéldányai önállóan létezhetnek, létezésük nem függ más egyedtől
- ***Gyenge entitás***: egyedpéldányai önállóan nem létezhetnek, csak erős entitáshoz kapcsolódóan létezhetnek.

2.2. Attribútum

Attribútumok szolgálnak a leíró tulajdonságok tárolására, azaz ezek írják le azt, hgy mit szeretnénk tárolni. Önmagukban nem állhatnak, csak egyedhez vagy kapcsolathoz kapcsolódva. Attribútumok között megkülönböztetünk egyszerű és összetett attribútumokat, valamint egyértékű és többértékű attribútumokat:

- ***Egyszerű attribútum****:* Egyszerű attribútum egy elemi információt jelent, amit tárolni szeretnénk. Például: Vezetéknév
- ***Összetett attribútum****:* Olyan attribútum, amely több másik attribútumból áll. Például: Lakcím
- ***Egyértékű attribútum****:* Olyan attribútum, amely egy egyed példány esetén egyszerre csak egy értéket vehet fel. Például: Személyi szám
- ***Többértékű attribútum****:* Olyan attribútum, amely egy egyed példány esetén egyszerre akár több értéket is felvehet.

Az attribútumokat az alábbi módon szokás jelölni az E/K diagramon:



2. ábra Attribútumok jelölése

2.3. Kapcsolat

A kapcsolatok az egyedek egymáshoz való viszonyát írják le. Ezek az összekötő elemei az adatmodellnek, ezek adják meg, hogy az egyes egyedek milyen kapcsolatban állnak egymással. Egy kapcsolat összeköthet akár egy egyedtípus több példányát (ezeket rekurzív kapcsolatoknak is hívják), általában két egyedtípus példányait rendeli össze, de elképzelhetők olyan kapcsolatok is, melyek több egyedtípust kapcsolnak össze. Az összetartozó egyedek alapján beszélhetünk:

- ***Egy – egy kapcsolat***ról: Egy egyedpéldányhoz legfeljebb egy másik egyedpéldány kapcsolódhat. Például:

3. ábra Egy-egy kapcsolat példa:
Egy alkalmazottnak egy parkolóhelye van és egy parkolóhelyet egy alkalmazott használhat

- ***Egy – több kapcsolat***ról: Egy egyedpéldányhoz több másik egyedpéldány is kapcsolódhat. Az egy – több kapcsolatban az szülőnek hívják azt az egyedet, melyből egy példány kapcsolódik, és gyereknek azokat az egyedeket, melyből több is lehet. Például:

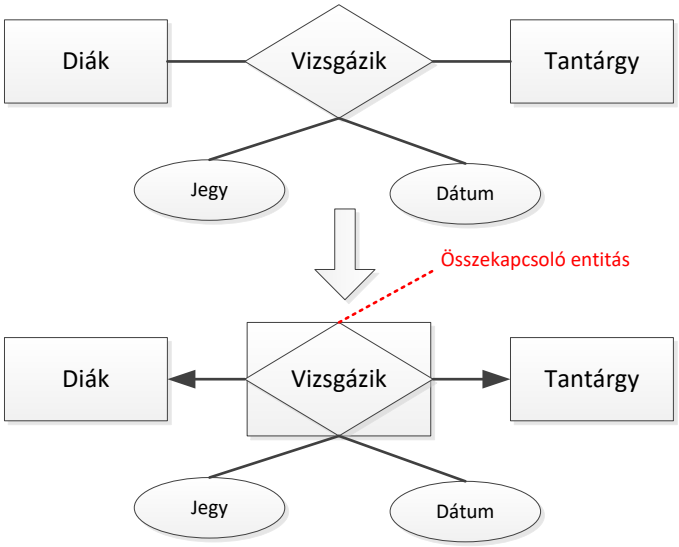
4. ábra Egy-több kapcsolat példa:
Egy termék egy kategóriába tartozik, de egy kategóriába több termék is tartozhat.
A kategória a szülő és a termék a gyerek ebben a példában

- ***Több – több kapcsolat***ról: Mindkét irányban megengedett, hogy egy egyedpéldányhoz több másik egyedpéldány kapcsolódjon. Például:

5. ábra Több-több kapcsolat példa:
Egy diák több tantárgyra is járhat, és egy tantárgyra több diák jár.

A kapcsolatok *számosságát (kardinalitását), tehát hogy a kapcsolat adott oldalán hány elem állhat,* a vonalvégeken elhelyezett *kényszerekkel* tovább pontosíthatjuk. (pl >1 vagy <2, <5).

A több-több kapcsolatokat általában ***összekapcsoló entitás***okkal szokták modellezni, mivel ekkor a több-több kapcsolat átalakul két darab egy-több kapcsolattá. Ennek oka, hogy a több-több kapcsolat nem normalizál, ezért nem is ábrázolható relációs adatmodellben közvetlenül. Viszont összekapcsoló entitás segítségével ez feloldható, és ábrázolhatóvá válik relációs adatmodellé. Például:

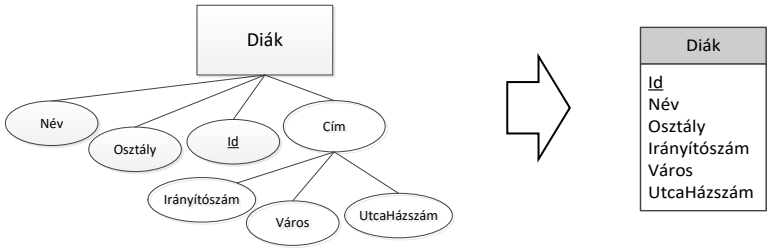


6. ábra Több-több kapcsolat átírása összekapcsoló entitással

2.4. Crow's foot jelölésmód

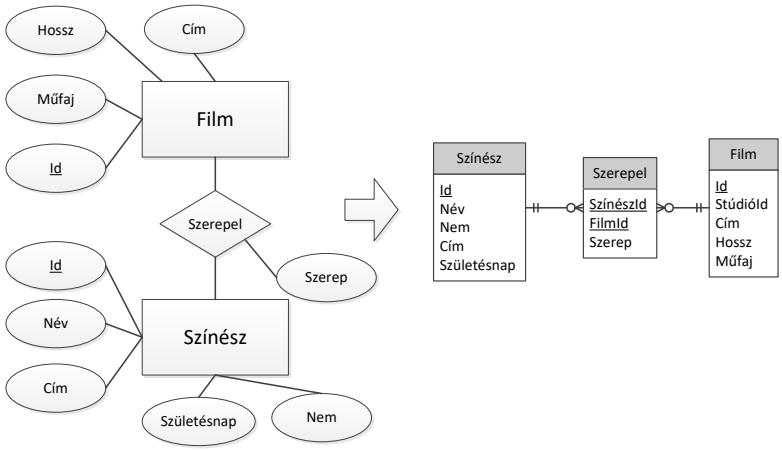
Az adatbázis-tervező eszközökben a Chen-féle jelölésmódtól némileg eltérő, u.n. ***Crow's foot*** jelölésmódot szokták többnyire alkalmazni. A két legfontosabb különbség, hogy

- az entitások attribútumait nem ellipszissel, hanem az entitás téglalapjának soraiként jelöljük



7. ábra Attribútumok jelölése Crow's foot jelölésmódban

- az attribútumokkal rendelkező kapcsolatokat entitásokként jelöljük (az attribútumokkal nem rendelkezőket egyszerű élekkel jelöljük). Ugyanakkor változik a kardinalitások jelölése is a vonalvégeken.



8. ábra Kapcsolatok jelölése Crow's foot jelölésmódban: a Szerepel több-több kapcsolatot egy *Szerepel* entitásra képeztük le, amely 1-több kapcsolatokkal kapcsolódik a *Színész* és *Film* entitásokhoz.

A kapcsolatok jellemzéséhez tartozik a minimum és maximum kardinalitásnak a meghatározása:

- ***Minimum kardinalitás:*** egy egyedpéldányhoz legalább ennyi egyedpéldány kapcsolódik
- ***Maximum kardinalitás:*** egy egyedpéldányhoz legfeljebb ennyi egyedpéldány kapcsolódhat

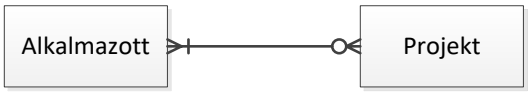
Ezeket részben már meghatározza a kapcsolat típusa (egy-egy, egy-több, több-több). A kardinalitások meghatározásánál a következő típusokat szokták megkülönböztetni:

- minimum kardinalitás 0: Azaz opcionális kapcsolat, nem feltétlenül kötelező, hogy legyen kapcsolódó elem
- minimum kardinalitás 1: Kötelező kapcsolat, minden esetben kell hivatkozás a másik entitásra
- maximum kardinalitás 1: Legfeljebb egy hivatkozás tartozhat hozzá
- Maximum kardinalitás sok: Sok hivatkozás tarozhat hozzá.

Jelölésük az E/K diagramon (Crow's foot) a következő:

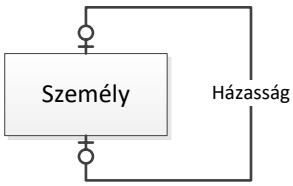
9. ábra Példa a minimum és maximum kardinalitások jelölésére

Egy kórtörténet bejegyzés minimum egy, maximum egy (azaz kötelezően egy) betegre vonatkozik, és egy betegnek egy vagy több kórtörténeti bejegyzése van



10. ábra Példa a minimum és maximum kardinalitások jelölésére

Egy alkalmazott több projekten is dolgozhat, de nem feltétlenül dolgozik projekten. Viszont egy projekten több alkalmazott is dolgozhat, de legalább egy alkalmazott dolgozik rajta.



11. ábra Példa a minimum és maximum kardinalitások jelölésére

Egy személy egy másik személlyel léphet házasságra, és legfeljebb egy házastársa lehet, mivel a házasság nem kötelező, ezért nem feltétlenül van házastársa.

A gyenge és erős entitások kérdését meg lehet közelíteni kapcsolatok felől is, a tervező eszközök is ezt a megközelítést támogatják általában. Ezen szempont szerint egy kapcsolat lehet:

- ***Azonosító kapcsolat:*** a gyerek rekord nem létezhet a szülő rekord nélkül (→gyenge entitás), ekkor tipikusan a gyerek rekord összetett kulcsot kap, melynek része a szülő rekord azonosítója.

- ***Nem azonosító kapcsolat:*** a gyerek rekord létezése független a szülő rekordtól (→erős entitás).

2.5. Egyed/Kapcsolat diagramok leképezése relációs adatmodellre

Az elkészült E/K diagramot relációs adatmodellé transzformálni viszonylag egyszerű algoritmussal lehet. Viszont az E/K diagramok tartalmazhatnak olyan részeket, melyek nem ábrázolhatók közvetlenül relációs adatmodellben, ezeket előbb meg kell szüntetni:

- Összetett attribútumok: Összetett attribútumoknál el kell dönteni, hogy szükséges-e önállóan az összetett attribútum. Ha igen akkor entitássá és kapcsolattá kell átalakítani, ha nem, akkor egyszerűen az egyes tagokat közvetlenül az egyedhez kell kapcsolni.
- Többértékű attribútumok: Entitássá és több-több kapcsolattá kell alakítani.
- Több-több kapcsolatok: Összekapcsoló entitás segítségével egy – több kapcsolatokká kell átalakítani.

A fenti átalakítások következtében a transzformált E/K diagram már csak egyszerű attribútumokat tartalmaz, valamint egy – egy és egy – több kapcsolatokat. A transzformáció menete:

- Egyedekből lesznek a táblák
- Az egyedekhez tartozó attribútumok adják a táblák oszlopait
- Egy – egy kapcsolatok esetén az egyik táblába idegen kulcsot kell elhelyezni a másik táblára
- Egy – több kapcsolatok esetében a gyerek táblában kell elhelyezni egy idegen kulcsot a szülő táblára

3. SQL nyelv alapjai

Az SQL nyelv elemeinek alapjait ismerteti a fejezet, csak azon részeket érintve, mely a gyakorlat elvégzéséhez feltétlenül szükséges.

3.1. Adatmodell létrehozása SQL nyelven

A relációs séma létrehozására az SQL nyelv DDL (Data Definition Language) utasításai szolgálnak. A táblák létrehozásakor meg kell adni az egyes oszlopok típusát. Az egyes MySQL Server adattípusokról ad rövid áttekintést a következő táblázat.

Típus	**Leírás**
Char(n)	Fix (n karakter) hosszúságú szöveg. A karakterkódolást az adatbázis collation paramétere határozza meg.
Varchar(n)	Változó hosszúságú szöveg, a szöveg maximális hossza n. A karakterkódolást az adatbázis collation paramétere határozza meg.
Numeric(sz,p)	Numerikus adattípus, sz jegynyi egész részből és p jegynyi tizedes részből áll.
Int	32 bites egész szám
Float	Lebegőpontos tört szám
Date	Dátum típus, napos felbontás

	Értékkészlet: 1000.01.01-től 9999.12.31
Datetime	Dátum és idő típus, másodperces felbontású Értékkészlet: 1000.01.01 00:00:00-tól 9999.12.31 23:59:59-ig

A táblák létrehozására a Create table utasítás szolgál:

```
CREATE TABLE táblanév (
oszlop név ADATTÍPUS [DEFAULT érték] [NULL|NOT NULL] oszlop szintű
     megszorítások,
…
oszlop név ADATTÍPUS [DEFAULT érték] [NULL|NOT NULL] oszlop szintű
     megszorítások,
tábla szintű megszorítások )
```

Tábla definiálása során megadható megszorítások:

- **PRIMARY KEY:** Elsődleges kulcs definiálása
- **UNIQUE [KEY]:** Kulcs jelöltek definiálása
- **[FOREIGN KEY] REFERENCES tábla(oszlopok):** Idegen kulcs másik táblára
- **CHECK [CONTRAINT] logikai kifejezés:** Tetszőleges logikai kifejezés helyességét lehet ellenőrizni.

Néhány példa táblák létrehozására.

```
Create table diak(
     ID int primary key,
     Nev Varchar(50) not null,
     Adoszam int not null unique,
     Szak Varchar(40) default 'Villamosmérnök')

Create table tantargy(
     ID int primary key,
     Nev Varchar(30))
Create table vizsgaeredmeny(
     DiakID int references diak(ID),
     TargyID int references tantargy(ID),
     VizsgaDatum datetime,
     Erdemjegy int,
Primary key (DiakID, TargyID, VizsgaDatum))
```

3.2. Egyszerű adatmanipuláció és lekérdezés

Ezen részben ismertetésre kerülnek az alapvető DML (Data Manipulation Language) utasítások szintaktikája, olyan mélységben, amely a gyakorlat elvégzéséhez szükséges

3.2.1. Rekordok létrehozása

Egy táblába új sort az insert utasítás segítségével lehetséges. Az insert utasítás szintaktikája a következő:

- Egy rekord beszúrása esetén:

```
insert into táblanév (oszlopok listája)
values (értékek listája)
```

ahol az oszlopok listája elhagyható a táblanév után, ekkor a tábla minden oszlopának értéket kell adni.

- Több rekord létrehozása esetén:

```
insert into táblanév (oszlopok listája)
select …
```

az oszlopok listája itt is elhagyható úgy, mint az előző esetben. A select rész szintaktikája a lekérdezéseknél kerül ismertetésre. A select rész egy lekérdezést fogalmaz meg és a lekérdezés által visszaadott összes sor beszúrásra kerül a táblába.

3.2.2. Meglévő rekordok törlése

Meglévő rekordok törlésére a delet utasítás szolgál, melynek a szintaktikája a következő:

```
delete
from táblanév
where logikai feltétel
```

Az utasítás feldolgozása során a végrehajtó motor minden sorra kiértékeli a feltételt, és azok a sorok kerülnek törlésre a táblából, ahol a where feltétel után megadott logikai kifejezés igaz. A where feltétel elhagyható, ekkor a tábla összes sora törlődni fog. A where feltételben megadható logikai kifejezések a lekérdezésekről szóló résznél kerülnek bemutatásra. Néhány példa:

1. Az összes termék törlése:

```
delete from termek
```

2. A 300 Ft-ba kerülő termékek törlése:

```
delete from termek
where ar = 300
```

3.2.3. Meglévő rekordok módosítása

Meglévő rekordok módosítására az update utasítás szolgál, melynek a szintaktikája a következő:

```
update táblanév
set oszlopnév1=érték1,
oszlopnév2 = érték2,
…
where logikai feltétel
```

Az update kulcsszó után megadott táblában azokat a rekordokat fogja módosítani, amelyekre a where után megadott logikai feltétel igaz. A where rész elhagyható, akkor a tábla összes rekordját módosítani fogja az update utasítás.

Példa: A 300 Ft-nál olcsóbb termékek árának emelése 10%-kal.

```
update termek
set ar=1.1*ar
where ar < 300
```

3.2.4. Rekordok egyszerű lekérdezése

Rekordok egyszerű lekérdezésére a select utasítás szolgál, melynek alapszintaktikája a következő:

```
select oszlop lista
```

```
from tábla lista
where logikai feltétel
```

A lekérdezés végrehajtását úgy lehet elképzelni, hogy a from részben felsorolt táblák Descartes szorzatának minden rekordjára (azaz az összes lehetséges rekord kombináción) kiértékelődik a where feltétel, és azok a rekordok kerülnek be a találati halmazba, ahol a where feltétel *igaz*. A select után felsorolt oszlopok adják az eredményreláció oszlopait. A select listában lehetőségünk van az oszlopok átnevezésére az **as** kulcsszó használatával. A select listában használható két speciális kifejezés:

- *: Összes oszlopot szeretnénk látni
- táblanév.* : Az adott tábla összes oszlopát szeretnénk látni.

A where feltételben használható legfontosabb operátorok rövid összefoglalóját tartalmazza az alábbi táblázat:

Operátor	**Jelentés**
Kif1 = Kif2	A két kifejezés értéke megegyezik.
Kif1 > Kif2	Kif1 értéke nagyobb, mint a Kif2 értéke.
Kif1 < Kif2	Kif1 értéke kisebb, mint a Kif2 értéke.
Kif1 >= Kif2	Kif1 értéke nagyobb egyenlő, mint a Kif2 értéke.
Kif1 <= Kif2	Kif1 értéke kisebb egyenlő, mint a Kif2 értéke.
Kif1 <> Kif2	Kif1 értéke nem egyenlő a Kif2 értékével.
Kif1 between Kif2 and Kif3	Kif1 értéke Kif2 és Kif3 közé esik, az intervallum határokat is beleértve.
Kif1 like String minta	Sting összehasonlító operátor. A Kif1-ben keres a mintának megfelelően, és ha a minta illeszthető a Kif1-re, akkor az operátor igaz értékkel tér vissza. A mintaillesztés case sensitive, a mintaillesztés során használható joker karakterek: _: egy betű helyettesítése %: tetszőleges hosszúságú szöveg helyettesítése '': szimpla ', mivel a szimpla ' jel önmagában a stringhatároló karakter.
Kif1 is null	Kif1 értéke null-e? Csak így lehet vizsgálni egy kifejezés nullitását. (A kif1=null kifejezés nem lesz igaz akkor sem ha a kif1 értéke egyébként null.)
Kif1 is not null	Kif1 értéke nem null.
Kif1 and Kif2	Két *logikai* kifejezés logikai és kapcsolata.
Kif1 or Kif2	Két *logikai* kifejezés logikai vagy kapcsolata.
not Kif1	*Logikai* kifejezés negálása.

Operátor	Jelentés
Kif1 in (allekérdezés)	Kif1 értéke szerepel-e az allekérdezéssel megadott rekordhalmazban.
Kif1 not in (allekérdezés)	Kif1 értéke nincs benne az allekérdezéssel megadott rekordhalmazban.
exists (allekérdezés)	Az allekérdezés tartalmaz-e rekordot.
not exists (allekérdezés)	Az allakerdezés nem ad vissza rekordot.

Néhány példa:

1. 200 Ft-nál drágább termékek listája:

```
select *
from termek
where ar > 200
```

2. 200 és 300 Ft közé eső termékek listája:

```
select *
from termek
where ar between 200 and 300
```

3. Azon termékek listája, ahol az ár nincs megadva:

```
select *
from termek
where ar is null
```

4. Azon termékek listája, amelyek neve tartalmazza az er karaktersorozatot és ára több mint 250 Ft:

```
select *
from termek
where nev like '%er%'
and ar > 250
```

4. MySQL Workbench használata

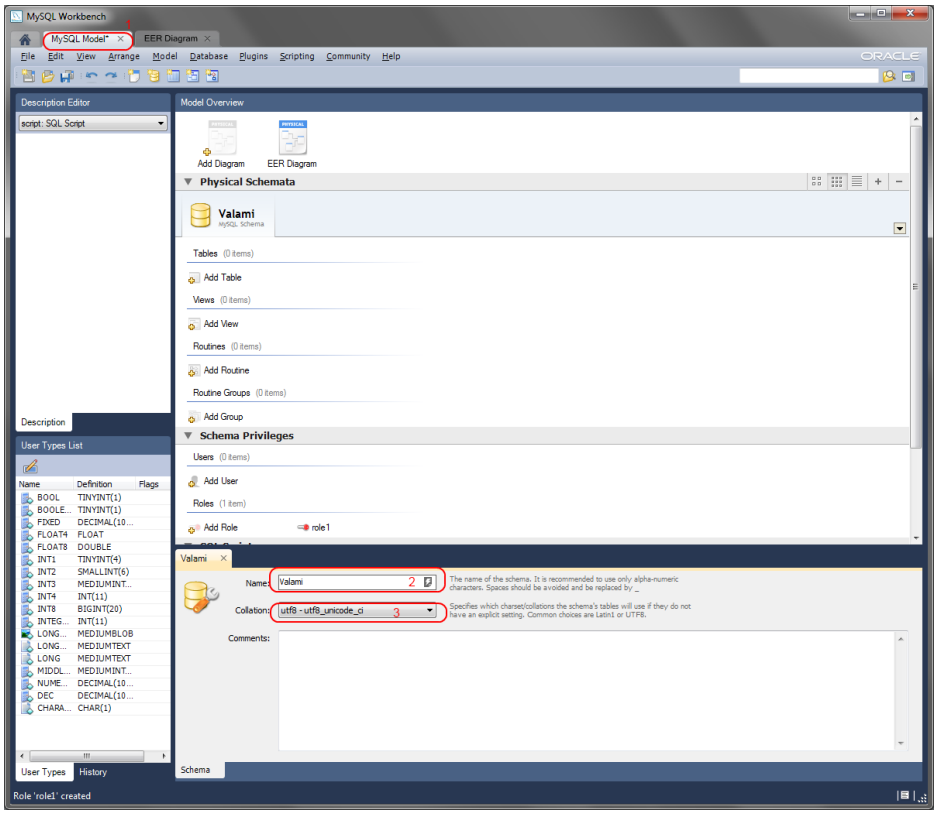
A MySQL Workbench egy olyan kliensalkalmazás, melynek segítségégével számos feladatot el lehet végezni a MySQL Serveres fejlesztések során:

- Adatbázis modellezés
- SQL fejlesztés és programozás
- Szerver adminisztráció

A gyakorlatok során csak az első két funkció szükséges, ezért csak azok kerülnek bemutatásra.

4.1. Adatbázis modellezés

Az adatmodellezés során csak a fizikai modellt lehet tervezni, logikai modellt nem lehet készíteni az eszközzel, csak olyan modellt lehet készíteni, mely relációs sémába közvetlenül leképezhető.

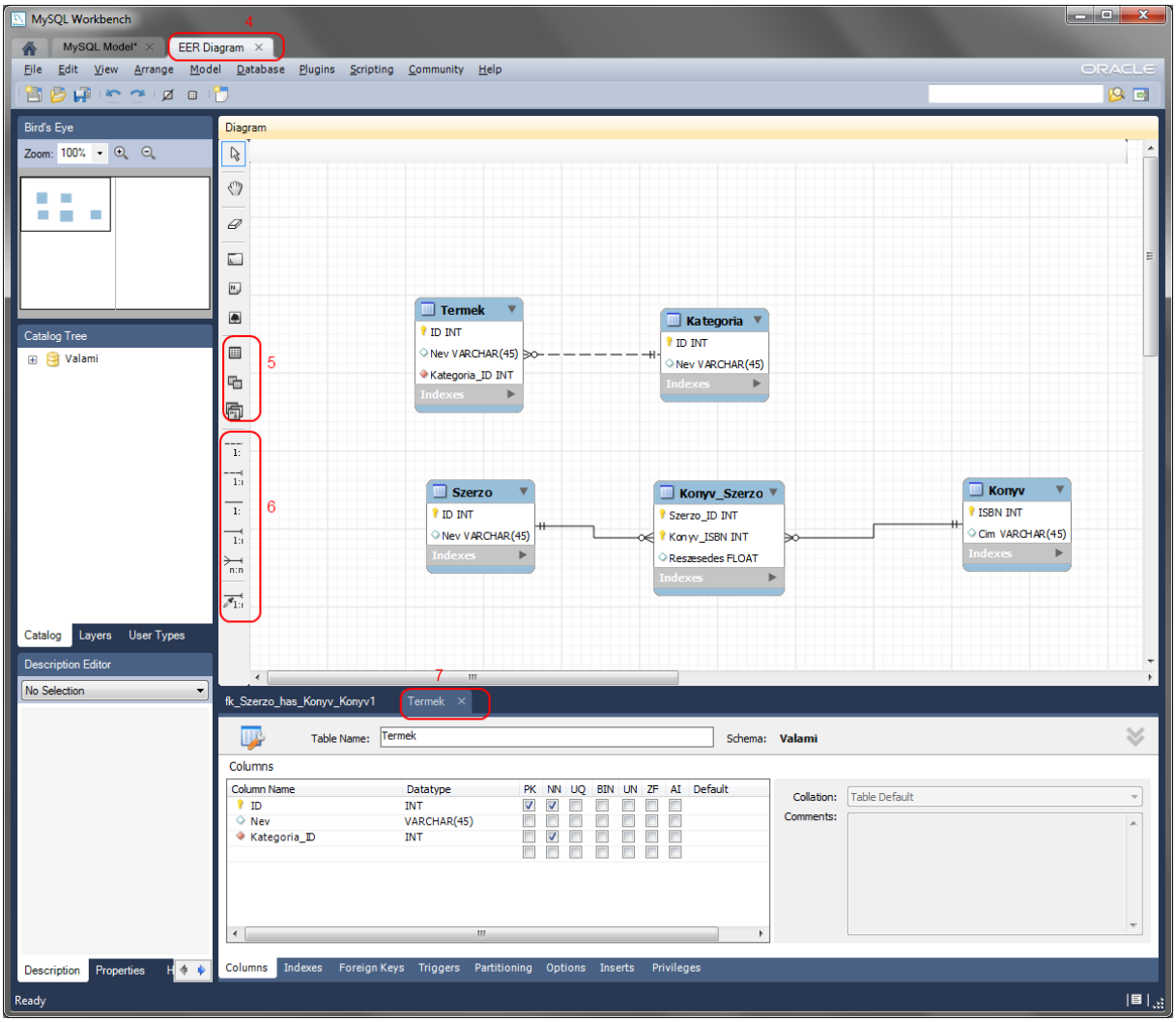


12. ábra MySQL Workbench modell elemek

A modell létrehozását követően megjelenik a modellszerkesztő ablak(1), ezt mutatja be az előző ábra. Egy modell számos elemből épülhet fel:

- *Fizikai adatbázis séma:* MySQL adatbázis sémára vonatkozó információk (pl.: séma név (2), collation(3))
- *Egyed kapcsolat modellek:* adatbázis tábláinak és kapcsolataiknak modellje
- *Jogosultságok:* felhasználók és jogosultságok a fizikai adatbázisban
- *Egyéb SQL szkriptek:* tetszőleges SQL szkriptek, amelyek például egyéb adatbázis elemek létrehozására szolgálhatnak

Egyed kapcsolat diagram hozzáadását követően megjelenik a szerkesztő felület (4), ezt szemlélteti az alábbi ábra.



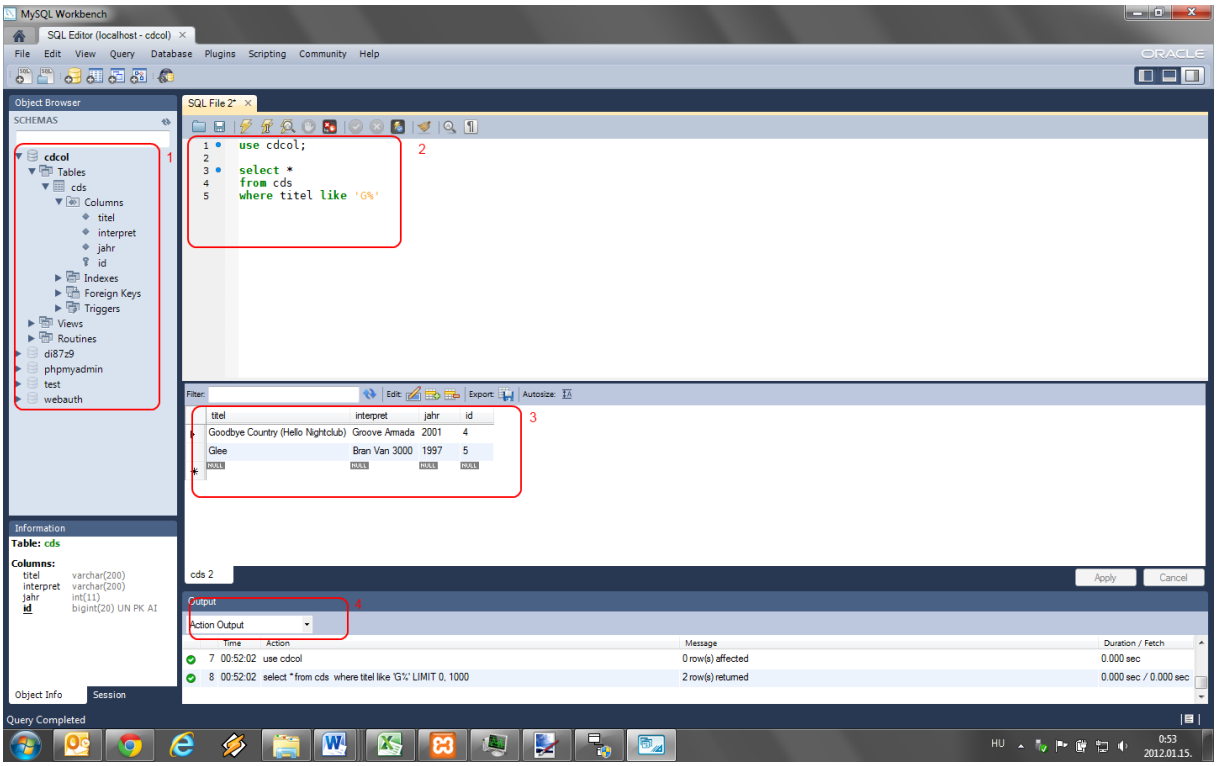
13. ábra Adatbázis modellezés MySQL Workbench használatával

A diagramra elhelyezhető elemeket a függőlegesen megjelenő eszköztárról lehet elhelyezni a diagramon: tábla elemek (5), kapcsolatok (6). A kapcsolatok modellezése során, a több-több kapcsolatokat rögtön átfordítja az eszköz összekapcsoló entitássá. Az egyes táblák szerkesztésére szolgál a táblaszerkesztő (7).

Az eszköz használatáról pár perces videó demonstráció letölthető a tárgy honlapjáról.

4.2. SQL fejlesztés

Alapvetően SQL utasítások írására, illetve szerver oldali programmodulok készítésére használatos eszköz. Az alkalmazást mutatja használat során az alábbi ábra.



14. ábra MySQL Workbench SQL szerkesztő

Adatbázishoz történő kapcsolódás után jelenik meg a fenti képernyő. Bal oldalt fa szerkezetbe rendezve láthatjuk az adatbázisokat és a bennük lévő objektumokat (1). Lekérdezést a szerkesztő felületen írhatunk (2) és futtathatunk le. Az eredmény halmazt táblázatosan jeleníti meg (3), valamint státusz visszajelzést is látunk a futtatásról (4).

Az eszköz használatáról pár perces videó demonstráció letölthető a tárgy honlapjáról.

5. Ellenőrző kérdések

1. Mi a különbség az egyértékű és többértékű attribútum között?
2. Mi a különbség az egyszerű és összetett attribútum között?
3. Mit értünk egy-egy kapcsolat alatt? Miképp jelöljük őket az E/K diagramon?
4. Mit értünk egy-több kapcsolat alatt? Miképp jelöljük őket az E/K diagramon?
5. Mit értünk több-több kapcsolat alatt? Miképp jelöljük őket az E/K diagramon?
6. Mit értünk erős entitás (entitás halmaz) alatt?
7. Mit értünk gyenge entitás (entitás halmaz) alatt?
8. Mit értünk azonosító (meghatározó) kapcsolat alatt?
9. Hogyan képezzük le az egy-egy kapcsolatot relációs adatmodellre?
10. Hogyan képezzük le az egy-több kapcsolatot relációs adatmodellre?
11. Hogyan képezzük le a több-több kapcsolatot relációs adatmodellre?
12. Írjon SQL utasítást, mely létrehoz egy táblát, aminek két oszlopa van és az első oszlopa a tábla elsődleges kulcsa!
13. Milyen szintaktikájú SQL utasítással lehet létrehozni (beszúrni) egy rekordot (sort) egy táblában?
14. Hogyan lehet vizsgálni egy kifejezés nullitását?
15. Miképp működik a like operátor? Milyen joker karaktereket lehet használni?