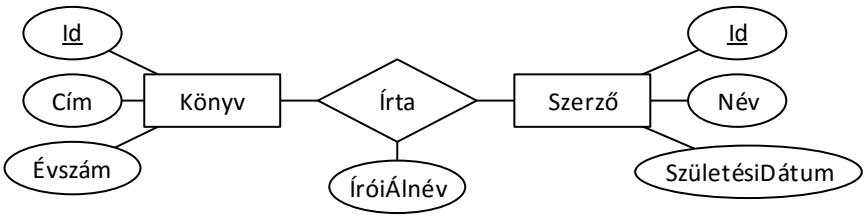
Informatika 2 Zárthelyi dolgozat 2016.04.18 **A csoport**		
A rendelkezésre álló idő 50 perc *Elégséges szint: 45% (18 pont).*	NÉV:	
	NEPTUN:	

Feladat	Pontszám
1 [6]	
2 [4]	
3 [4]	
4 [6]	
5 [7]	
6 [7]	
7 [6]	
∑ [40]	

1. Feladat: Adott az alábbi ábrán látható Entitás-Relációs diagram. Írjon SQL utasításokat, amelyek az ennek megfelelő táblákat létrehozzák! [**6 pont**]

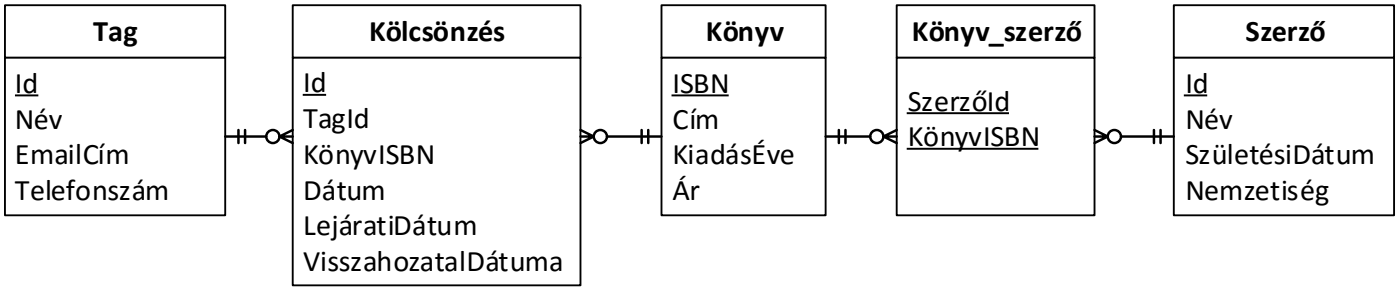



```
create table Konyv (
      Id int primary key,
    Cim nvarchar(200) not null,
    Evszam int);

create table Szerzo(
      Id int primary key,
    Nev nvarchar(100) not null,
    SzuletesiDatum datetime);

create table Irta(
      KonyvId int,
    SzerzoId int,
    IroiAlnev nvarchar(200),
    primary key(KonyvId, SzerzoId),
    foreign key (KonyvId) references Konyv(Id),
    foreign key (SzerzoId) references Szerzo(Id));
```

2. Feladat: Adott az alábbi ábrán látható adatbázis séma. Írjon SQL lekérdezést, amely megjeleníti azon szerzők neveit és könyveik címeit, amelyet már kikölcsönöztek a könyvtárból! [**4 pont**]




```
select distinct sz.nev, k.cim
from szerzo sz
    inner join konyv_szerzo ksz on sz.id = ksz.szerzoid
    inner join konyv k on k.isbn = ksz.konyvisbn
    inner join kolcsonzes kcs on kcs.konyvisbn = k.isbn
```

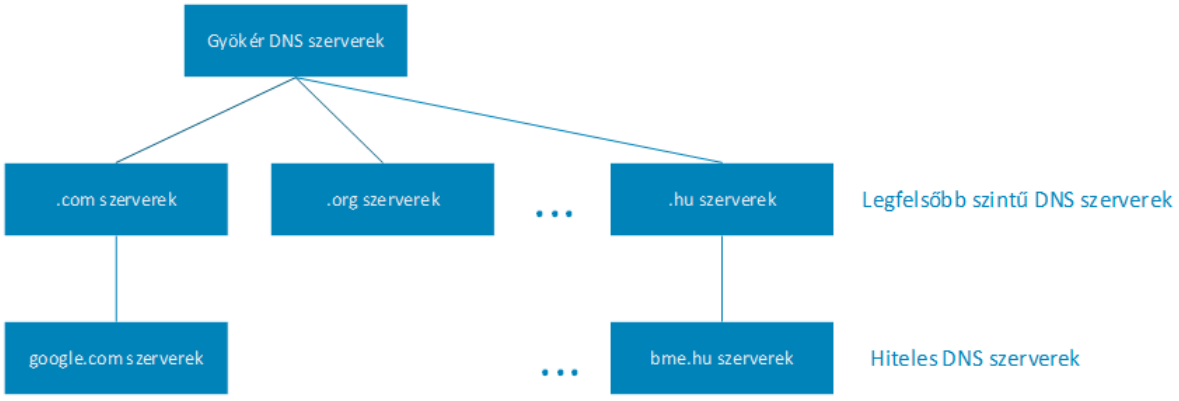
3. Feladat: Írjon SQL lekérdezést, amely megjeleníti, hogy az egyes tagok összesen hányszor kölcsönöztek a könyvtárból! [**4 pont**]

```
select t.*, count(kcs.id)
from tag t
      inner join kolcsonzes kcs on t.id = kcs.tagid
group by t.id
```

4. Feladat: Ismertesse, hogy milyen DNS szervereket különböztetünk meg, és hogy ezek milyen hierarchiába rendezve biztosítják a DNS szolgáltatást! [**6 pont**]

A DNS rendszert hierarchiába rendezett adatbázisszerverek alkotják.

- A **gyökér** (Root-Level) DNS szerverekből 13 darab van, amelyet 12 különböző önálló szervezet működtet.
- A gyökér szerverek a **legmagasabb szintű** (Top-Level – TLD) DNS szerverek elérhetőségét tárolják. A TLD szerverek felelősek a legfelsőbb szintű domainekért (pl. .com, .org, .eu, .hu).
- A következő szinten találhatók az egyes szervezetek saját **hiteles DNS szerverei**, amelyek a szervezeten belüli DNS leképezéseket tárolják. Minden, publikus IP címeket kezelő szervezet kötelezően meg kell adjon egy hiteles DNS szervert, amely mindig a legfrissebb név-cím leképezéseket kell tárolja az adott szervezeten belül.



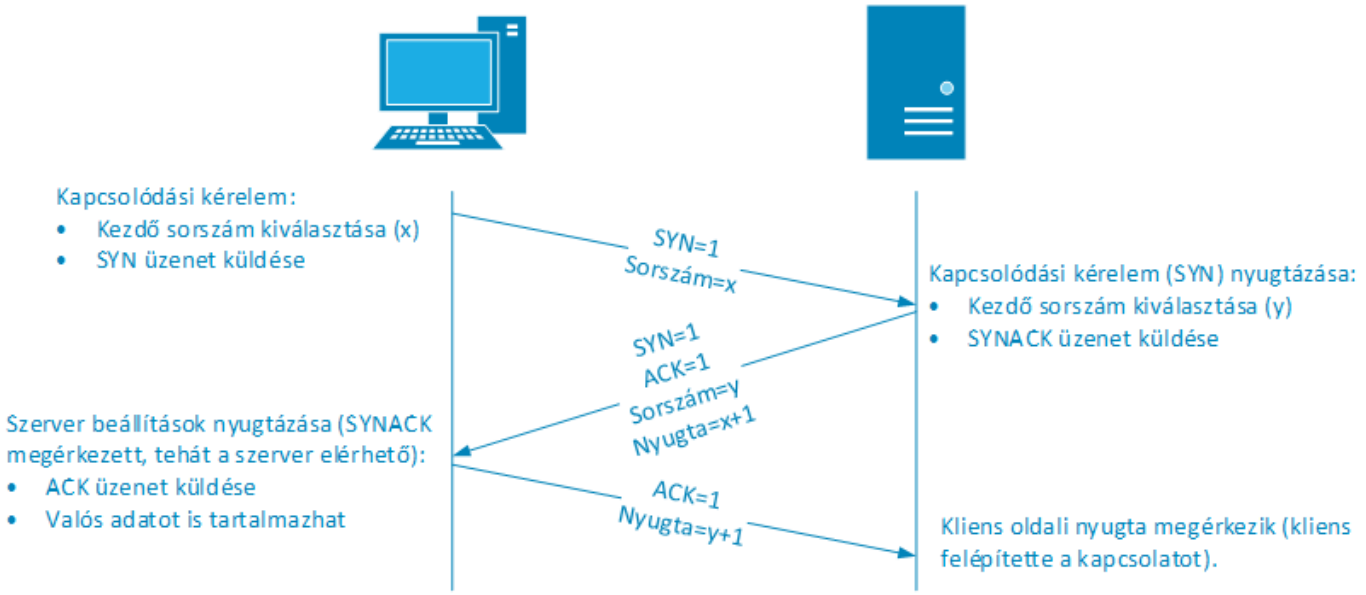
Hiba! Nincs ilyen stílusú szöveg a dokumentumban.-1 ábra: DNS szerverek hierarchiája

- Bár a hierarchiának nem képezik részét, nagyon fontos elemei a DNS rendszernek a **lokális (helyi) DNS szerver**ek. A legtöbb Internet szolgáltató kezel ilyen szervert, amelynek a feladata alapvetően a gyorsítótárazás.

5. Feladat: Ismertesse a socketkezelés szerveroldali függvényeit és hogy melyik mire szolgál! [**7 pont**]

1. Szerver socket létrehozása (`socket`).
2. A socket hozzárendelése egy hálózati címhez és portszámhoz (`bind`). Itt a programozó feladata, hogy egy még nem használt portszámot megadjon. Ha a portszám foglalt, akkor nem jön létre a socket, hanem hibát kapunk.
3. Harmadik lépésben a szerver socketet passzív állapotba helyezzük, felkészítve arra, hogy később beérkező kapcsolatokat fogadjon (`listen`).
4. Várakozás kliensek kapcsolódására (`accept`). Amennyiben egy kliens csatlakozik, akkor létrehozunk egy új socketet, amelyen keresztül az újonnan csatlakozott klienssel tudunk kommunikálni. Erre azért van szükség, hogy az eredeti socketet fenntartsuk további csatlakozni akaró kliensek számára.
5. Adatok fogadása (`recv`).
6. Adatok küldése (`send`) a socketen keresztül.
7. A socket lezárása (`closesocket`), ha már nincsen rá szükség. Erre azért van szükség, hogy az operációs rendszer erőforrásait ne foglaljuk feleslegesen.

6. Feladat: Ismertesse a TCP kapcsolat felépítésének forgatókönyvét! (Üzenetek típusa, sorrendje, jelzőbitek, sorszámok.) [**7 pont**]



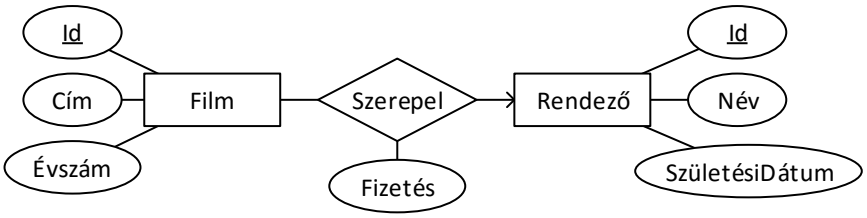
7 Feladat: Ismertesse a tranzakciókezelés 4 izolációs alapproblémáját! (Elnevezés és magyarázat is kell!) [**6 pont**]

- Az **elveszett módosítás** alatt azt a problémát értjük, amikor két (vagy több) párhuzamosan futó tranzakció ugyanazon adat azonos kommitált képéből kiindulva módosításokat hajt végre, majd a módosításokat kommitálják, és a különböző tranzakciók által elvégzett műveletek közül csak az egyik hatása fog érvényre jutni az adatbázisban, tipikusan az, amelyik időben utoljára kommitálódott.
- **Piszkos olvasás** akkor történik, amikor két, egymással párhuzamosan futó tranzakció közül az egyik módosít egy adatot, és még mielőtt kommitálná azt, egy másik tranzakció ezt az ideiglenes értéket kiolvassa és felhasználja.
- Egy **olvasás**t akkor nevezünk **nem megismételhető**nek, ha ugyanazon tranzakción belül ugyanazt az adatmezőt kétszer lekérdezve különböző eredményt kapunk. Ez akkor fordulhat elő, ha az egyik tranzakció által lekérdezett adatot egy másik tranzakció időközben módosítja (és kommitálja is, tehát nem piszkos adatot olvas), majd az első tranzakcióban ugyanezt az adatot újból lekérdezve már a módosított értéket kapjuk.
- A **fantom rekordok** problémája a nem megismételhető olvasás egy változata azzal a különbséggel, hogy nem csak egyszerű attribútumokra vonatkozik, hanem teljes rekordokra: ugyanazt a lekérdezést ismételten lefuttatva egy tranzakcióban új rekordok jelennek meg az eredményben, vagy korábbiak tűnnek el. A nem megismételhető olvasáshoz hasonlóan ez akkor fordulhat elő, ha időközben egy másik tranzakció új rekordokat szúr be vagy töröl a táblából, és kommitálja is a módosítást (tehát nem piszkos adat olvasása történik).

Informatika 2 Zárthelyi dolgozat 2016.04.18 **B csoport**		
A rendelkezésre álló idő 50 perc. *Elégséges szint: 45% (18 pont).*	NÉV:	
	NEPTUN:	

Feladat	Pontszám
1 [7]	
2 [4]	
3 [5]	
4 [6]	
5 [5]	
6 [7]	
7 [6]	
∑ [40]	

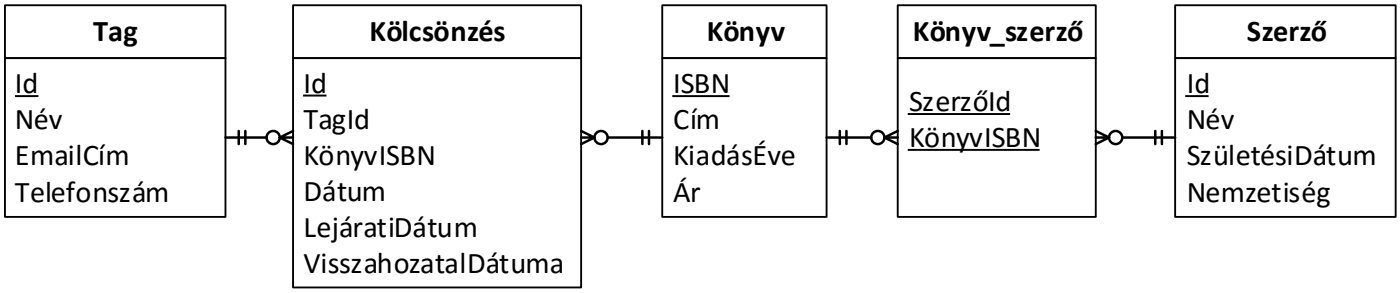
1. Feladat: Adott az alábbi ábrán látható Entitás-Relációs diagram. Írjon SQL utasításokat, amelyek az ennek megfelelő táblákat létrehozza! [**7 pont**]




```
create table Redezo(
      Id int primary key,
      Nev nvarchar(100) not null,
      SzuletesiDatum datetime);

create table Film(
      Id int primary key,
      Cím nvarchar(200) not null,
      Evszam int,
      RedezoId int not null,
      RendezoFizetes int,
      foreign key (RendezoId) references Rendezo(Id)
      )
```

2. Feladat: Adott az alábbi ábrán látható adatbázis séma. Írjon SQL lekérdezést, megjeleníti azokat a könyveknek a címét, amelyeknek van angol nemzetiségű szerzője! (Ugyanaz a cím ne szerepeljen kétszer az eredményben!) [**4 pont**]



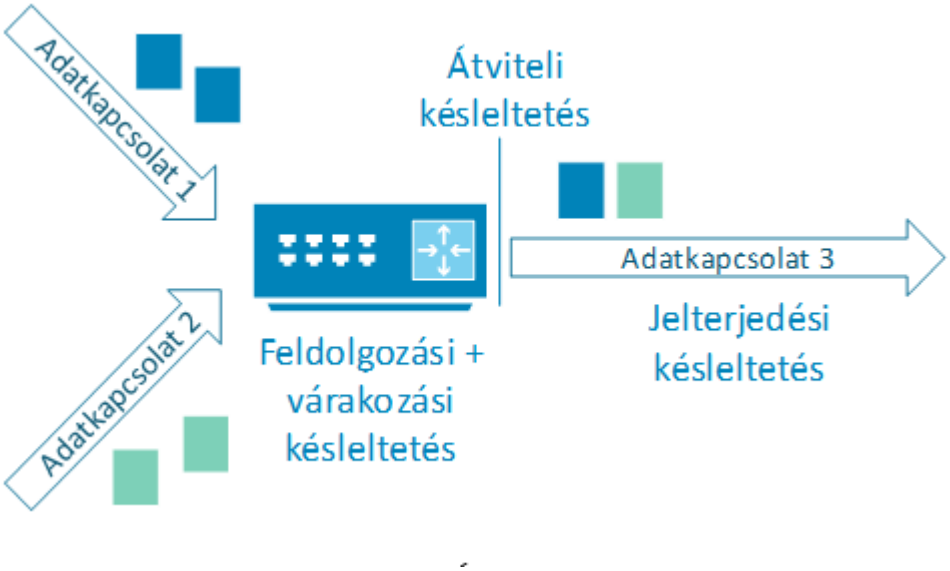

```
select distinct k.cim
from konyv k
      inner join konyv_szerzo ksz on ksz.konyvisbn = k.isbn
      inner join szerzo sz on sz.id = ksz.szerzoid
where sz.nemzetiseg = 'angol'
```

3. Feladat Írjon SQL lekérdezést, amely megjeleníti, hogy az egyes szerzőknek összesen hányszor kölcsönözték ki a könyveit! [**5 pont**]

```
select sz.*, count (k.id)
from szerzo sz
      inner join konyv_szerzo ksz on sz.id = ksz.szerzoid
      inner join kolcsonzes k on k.konyvisbn = ksz.konyvisbn
group by sz.id
```

4. Feladat: Ismertesse, hogy csomagkapcsolt hálózatokban milyen összetevőkből adódik egy csomag teljes késleltetése, amíg a hálózaton eljut a küldőtől a címzettig! Magyarázza el az egyes összetevők jelentését! [**6 pont**]

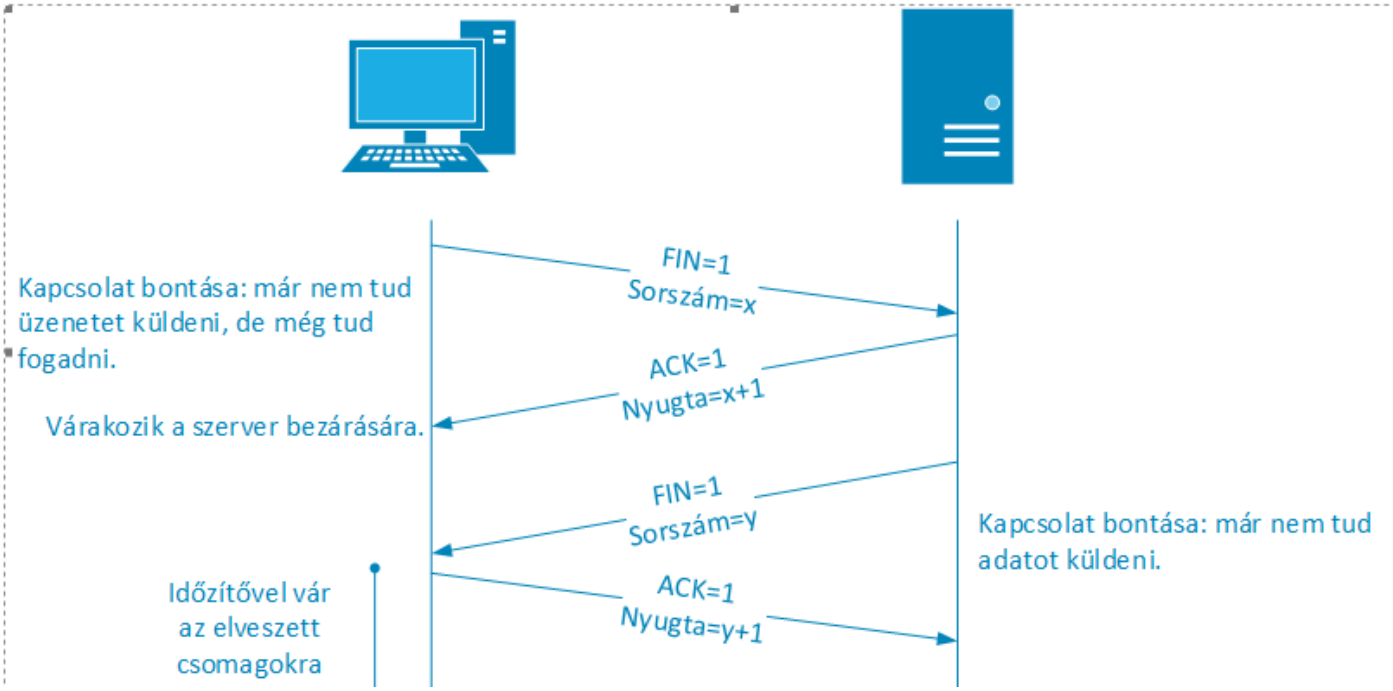
- Csomagkapcsolás során az üzeneteket csomagokra bontjuk. Tegyük fel, hogy egy csomag L bitből áll. Az adatkapcsolatok sebességét bit/másodpercben (bit/s) szoktuk megadni. A tárol és továbbít el alapján egy L hosszú csomag felküldése egy R sebességű csatornára L/R ideig (másodpercig) tart. Ezt nevezzük **átviteli késleletetés**nek.
- Ehhez hozzáadódik a jelterjedési **késleltetés**, amely az adott kapcsolat hosszával van összefüggésben.
- Amennyiben egy csomagot több csomóponton keresztül kell elküldenünk, akkor az egyes adatkapcsolatokon történő átvitelhez hozzáadódik az útválasztó által okozott ún. *csomagkapcsolási késleltetés*. Nézzük meg részletesen, hogy milyen tételekből adódik össze ez a késleltetés!
    - A továbbítás során el kell dönteni, hogy melyik kimenő adatkapcsolatra kell a csomagot továbbítani, ehhez pedig fel kell dolgozni a csomag fejlécét (első néhány bájtját), amely a célcsomópont címét tartalmazza. Ezt nevezzük **feldolgozási késleltetés**nek. A feldolgozás során gyakran bitszintű hibaellenőrzés is történik, amely szintén növeli a folyamat idejét. Nagysebességű útválasztóknál a feldolgozási késletetés igen gyors (kevesebb, mint msec).
    - Egy útválasztó több adatkapcsolati összeköttetéssel rendelkezik, egyszerre többön keresztül is érkezhetnek csomagok, továbbá lehetnek eltérések az egyes adatkapcsolatok sebessége között. Ilyenkor a bejövő bitek száma nagyobb lehet, mint a továbbítás sebessége. Ezen okok miatt előfordulhat, hogy egy csomagot várakoztatni kell, mielőtt az elküldésre kerülhet (**várakozási késleltetés**). Ennek nagyságrendje tipikusan mikro és milli szekundum közötti érték. Ilyenkor a csomag egy várakozási sorba kerül. Fontos megjegyezni, hogy az útválasztó memóriája megtelhet, ilyenkor a további bejövő csomagok eldobásra kerülnek, ezt nevezzük *csomagvesztés*nek.



5. Feladat: Ismertesse a socketkezelés kliensoldali függvényeit és hogy melyik mire szolgál! [**5 pont**]

1. Új socket létrehozása (`socket`). A létrehozás során meg kell adni, hogy milyen szállítási rétegbeli protokollt szeretnénk használni (TCP vagy UDP).
2. A socket segítségével kapcsolódás egy szerverhez (`connect`). A paraméternél meg kell adni, a szerver IP címét és portszámát.
3. Adatok küldése (`send`) és
4. fogadása (`recv`).
5. A socket lezárása (`closesocket`), ha már nincsen rá szükség.

6. Feladat: Ismertesse a TCP kapcsolatbontás forgatókönyvét! (Üzenetek sorrendje, jelzőbitek, sorszámok.) [**7 pont**]



7. Feladat: Ismertesse a tranzakciókezelés 4 szabványos izolációs szintjét és azt, hogy ezek milyen izolációs alapproblémákat oldanak meg! (Elnevezés és magyarázat is kell!) [**6 pont**]

A szabvány által meghatározott izolációs szintek a következők:

- **UNCOMMITTED READ**: semmilyen izolációt nem biztosít, egyik izolációs alapproblémát sem zárja ki
- **COMMITTED READ**: a piszkos olvasás nem lehetséges, az egyes tranzakciók mindig csak kommitált adatot olvashatnak.
- **REPEATABLE READ**: a piszkos olvasást és a nem megismételhető olvasást is kizárja.
- **SERIALIZABLE** (sorosítható): egyik alapprobléma sem fordulhat elő.