Adatbázisok 1
2013-14 tavaszi félév

Vizsgatételsor

1. **Relációs adatmodell alapjai**

Adatmodell:
Az adatmodell egy jelölésmód egy adatbázis adatszerkezetének a leírására, beleértve az adatra vonatkozó megszorításokat is. Az adatmodell általában az adatokon végezhető műveletek leírására alkalmas jelölést is magába foglalja. A műveletek lehetnek lekérdezések, illetve adatmódosítások.

Reláció:
A relációk információt reprezentáló táblák. Az oszlopok fejlécében **attribútumok** vannak, minden attribútumhoz tartozik egy tartomány vagy adattípus. A táblasorokat **soroknak** hívjuk és a reláció minden attribútumához tartozik a sornak egy komponense.

Relációséma:
A relációnév a reláció attribútumaival együtt adja a relációsémát. A relációsémák összessége az adatbázisséma. Egy relációhoz vagy relációk összességéhez tartozó adatokat az adott relációsémához vagy adatbázissémához tartozó előfordulásnak nevezzük.

Attribútumok:
A reláció oszlopait attribútumok látják el névvel, az oszlop fejrészében találhatók. Általában megadják az oszlopban szereplő adatok jelentését.

Sorok:
A reláció azon sorait, melyek különböznek az attribútumok által megadott fejléc soraitól, soroknak (tuple) nevezzük. A reláció minden attribútumához tartozik a sorban egy komponens.

Kulcsok:
Egy fontos megszorítási típus a táblákra nézve, hogy ki tudunk jelölni egy attribútumot vagy attribútumhalmazt, amely egy kulcsot alkot a relációra nézve. Egy reláció semelyik két sora nem egyezhet meg a kulcsattribútumokon, de egy részhalmazukon igen.

Külső kulcsok:

Egy külső kulcs megszorítás meghatározza, hogy egy tábla bizonyos oszlopa csak egy másik tábla egy bizonyos oszlopának értékeit veheti fel.

Hivatkozási épség:
A hivatkozási épség egy megszorítás, mely szerint ha egy érték megjelenik valahol egy környezetben, akkor ugyanez az érték egy másik, az előzővel összefüggő környezetben is megjelenik.

2. **Relációsémák**

Relációsémák definiálása SQL-ben:

Az SQL egy fő része az adatleíró rész (DDL), mely az adatbázis objektumainak leírására és módosítására használható. A relációkat az SQL-ben táblának (table) nevezzük, melyből az SQL alapvetően három típust különít el:

- Alaptáblák, ez a legalapvetőbb típus. Az SQL ezeket tárolja az adatbázisban.
- Nézettáblák, ezeket nem tároljuk, valamilyen számítások elvégzésével kapott relációk (táblák).
- Ideiglenes, átmeneti munkatáblák. Az SQL nyelvi feldolgozója hozza létre lekérdezések és adatmódosítások végzése során. Ezen relációkat eldobja, nem tárolja el az SQL feldolgozója.

CREATE TABLE:

Ezzel az utasítással határozható meg egy tárolt reláció sémája. Megadja a tábla nevét, attribútumait, valamint az attribútumok típusát. Emellett kulcs, illetve kulcsok megadását is lehetővé teszi. Egyéb megszorítások, valamint indexek meghatározását is végrehajthatjuk vele.

Formátum:

```
CREATE TABLE relációnév (
        Attribútum deklarációk listája,
        További kiegészítések
);
```

Példa:

```
CREATE TABLE Filmek (
        filmcím         CHAR(100),
        év              INT,
        hossz           INT,
        műfaj           CHAR(10),
        stúdióNév       CHAR(30),
        producerAzon    INT
);
```

Kulcsok megadása:

A kulcsok megadása során két típust különítünk el:

- Egyszerű kulcs (egy attribútum), vagy
- Összetett kulcs (attribútumok listája)

Jellegét tekintve lehet PRIMARY KEY, illetve UNIQUE. A kulcs megadásával a relációnak nem lehet két olyan sora, ahol a kulcsként megadott attribútumok értékei megegyeznének, még akkor sem ha azok NULL értékűek.

Csak egyetlen PRIMARY KEY szerepelhet egy relációban, UNIQUE azonban több is lehet. Ezen felül a PRIMARY KEY értéke nem lehet NULL, UNIQUE esetében azonban a NULL érték is megengedett.

Egyszerű kulcsot az attribútum deklarálása során adhatunk meg:
<attribútum neve> <típus> PRIMARY KEY / UNIQUE

Összetett kulcs esetén a kiegészítéseknél adhatjuk meg:
UNIQUE(attr1, … attr2)

Hivatkozási épség:

Idegen kulcs, más néven FOREIGN KEY esetén a hivatkozott oszlopnak a másik táblában kulcsnak kell lennie! Megkötés továbbá, hogy a tábla oszlopaiban szereplő értékeknek szerepelnie kell a hivatkozott tábla oszlopának értékei között.

Az idegen kulcsokat megadhatjuk:

Az attribútum deklarálása során:
<attribútum neve> <típus> REFERENCES Tábla(attribútum)

Sémaelemként a kiegészítések között:
FOREIGN KEY(attribútum) REFERENCES Tábla(attribútum)

3. **Relációs algebra**

Relációs algebra:
Ez az algebra egy fontos lekérdező nyelv a relációs modellben. Alapvető műveletei: egyesítés, metszet, különbség, kiválasztás, vetítés, Decartes-szorzat, természetes összekapcsolás, Théta-összekapcsolás és átnevezés.

Egyesítés: Két halmaz közötti unio művelet. Azon elemek halmaza, amelyek valamely halmaznak elemei. Minden elem csak egyszer szerepel az egyesítésben.

Metszet: Két halmaz közötti metszet művelet. Azon elemek halmaza, amelyek mindkét halmaznak elemei.

Különbség: R-S. Azon elemek halmaza, melyek benne vannak R-ben, de nem szerepelnek S-ben.

Kiválasztás: $\sigma_C(R)$. A kifejezés értéke az R reláció azon sorai, melyekre teljesül a C feltétel.

Vetítés: $\pi_{attr1, attr2,\ldots,attrn}(R)$ kifejezés értéke az a reláció, amelyik az R relációnak szak az att1, attr2, …, attrn attribútumokhoz tartozó oszlopait tartalmazza.

Decartes-szorzat: R x S: Két halmaz, R és S Decartes-szorzata egy olyan relációt ad eredményül, melynek attribútumai R és S attribútumainak egyesítése (előbb R, utóbb S attribútumai jelennek meg), elemeit pedig R minden sorához S minden sorát rendelve kapjuk. Ha a két sémának létezik megegyező nevű attribútuma, akkor legalább az egyiknek új nevet kell adni.

Természetes összekapcsolás: $R \bowtie S$ egy összekapcsolás, mely R és S azon sorait párosítja össze, amelyek értékei megegyeznek R és S sémájának összes közös attribútumán. Ehhez szükséges, hogy R és S rendelkezzen olyan attribútumokkal, mely mindkettőben megtalálható.

Théta-összekapcsolás: Olyan összekapcsolás, mely eredményként az összekapcsolásnak azon sorait kapjuk meg, melyek eleget tesznek egy megadott C feltételnek. Pl.: $R \bowtie_C S$

Átnevezés: $\rho_{S(A1,A2,\ldots,An)}(R)$ eredményreláció neve S, sorai megegyeznek R soraival, azonban attribútumainak neve balról jobbra átnevezésre kerülnek: A1, A2, …, An. Az attribútumok neveinek meghagyásához elhagyhatjuk az oszlopok nevét az azokat tartalmazó zárójellel együtt.

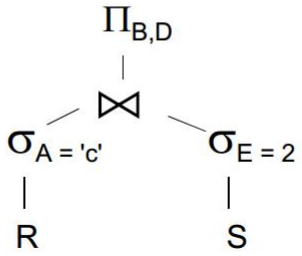
Kifejezések felépítése:

a. Elemi kifejezések:
   (i) $R_i \in R$ (az adatbázis-sémában lévő relációnevek)
   (ii) konstans reláció (véges sok, konstansból álló sor)
b. Összetett kifejezések
c. Ha E1 és E2 kifejezések, akkor a köztük értelmezett műveletekkel kapott E is egy kifejezés
   Ezek és csak ezek a kifejezések, amit így meg tudunk adni

Kifejezésfa példa:

$$\Pi_{B,D}(\sigma_{A='c'}(R) \bowtie \sigma_{E=2}(S))$$

- Ehhez is felrajzolva a kiértékelő fát:



4. **Az SQL adatbázisnyelv 1.**

Egyszerű lekérdezések:
Az SQL legegyszerűbb lekérdezései azon sorokra vonatkoznak, melyek egy bizonyos relációban eleget tesznek egy adott feltételnek. Egy ilyen lekérdezés a relációs algebra kiválasztási műveletének felel meg. Egy ilyen lekérdezés az SQL három alapvető kulcsszavát használja: SELECT, FROM és WHERE.

Példa egy lekérdezésre: SELECT * FROM Filmek WHERE stúdióNév = „Disney” AND év = 1990;

A FROM záradék azon relációt, vagy relációkat adja meg, melyekre a lekérdezés vonatkozik.

A WHERE záradék egy feltétel, mely hasonlóképp működik, mint a relációs algebra kiválasztási feltétele. A lekérdezés eredményébe azon sorok kerülnek, melyek kielégítik az adott feltételt.

A SELECT záradék megadja a feltételnek megfelelő sorok azon attribútumait, melyeket a lekérdezésre adott eredmény tartalmazni fog. A * érték esetén a reláció minden attribútumát tartalmazni fogja az eredmény. Lényeges különbség, hogy az SQL-ben az eredmény alapértelmezésben nem halmaz, hanem multihalmaz. Ahhoz, hogy halmazt kapjunk, a SELECT DISTINCT utasítás alkalmazandó.

Lehetőségünk van az eredményoszlopok átnevezésére is, ezt az AS kulcsszóval tehetjük meg, mely Oracle esetén elhagyható. SELECT [attr] AS [ujnev]. A SELECT feltételben megadhatunk kifejezéseket is, például SELECT ar*0,27

Használható elemi feltételek: =, <>, <, >, <=, >=
Logikai műveletek: AND, OR, NOT és a zárójelek ()

Speciális műveletek:
BETWEEN … AND …
IN (értékhalmaz)

Karakterláncok összehasonlítására lehetőség van a LIKE feltétellel, pl: WHERE név LIKE ’%s%’.

Az SQL lehetővé teszi, hogy a relációk soraiban NULL mint ismeretlen érték szerepeljen, ennek értelmezésére az IS NULL, illetve IS NOT NULL feltételeket használhatjuk. NULL értéket bármivel összehasonlítva az eredmény UNKNOWN.

Az SQL-ben szereplő logikai feltételeket háromértékű logikának nevezzük: TRUE, FALSE, UNKNOWN. A WHERE záradékban lévő feltételt a rendszer minden sorra kiértékeli, az eredményhalmazba pedig csak azon sorok kerülnek be, amelyeknek a feltétel kiértékelése TRUE értéket adott.

5. **Az SQL adatbázisnyelv 2.**

Többrelációs lekérdezések:
Egy többrelációs lekérdezés során a SELECT FROM WHERE záradékok használatakor a FROM záradék táblák listáját tartalmazza, melyek összekapcsolásával, illetve szorzatával kapjuk meg az eredményt.

Például:
SELECT A, R.B AS B, C FROM S, R WHERE R.B = S.B;

Előfordulhat, hogy az összekapcsolás, illetve szorzat során a két táblában valamely attribútum, vagy attribútumok neve megegyezik. Ezen attribútumok megkülönböztetésének érdekében az adott A attribútumra a t_i.A kifejezéssel hivatkozhatunk, ahol t_i egy a FROM záradékban megadott reláció neve.

Lehetőség van ugyanazon reláció többszöri felvételére is a FROM záradékban, ekkor a reláció neve után valamilyen másodnevet kell megadni. Pl.:
FROM R1 t1, …, Rn tn

Egy reláció önmagával történő szorzásakor a relációknak másodnevet kell megadni, vagyis sorváltozót írunk mellé a FROM záradékban.

Példa: SELECT t1.Szülő NagySzülő, t2.Gyerek Unoka FROM R t1, R t2 WHERE t1.Gyerek = t2.Szülő;

Halmazműveletek:
INTERSECT, UNION, EXCEPT (egyértelmű a működésük…)

6. **Az SQL adatbázisnyelv 3.**

Alkérdések:
A FROM záradékban alkérdéssel létrehozott ideiglenes táblát adhatunk meg, melyhez sorváltozóval egy másodnevet kell megadnunk.

Példa:
SELECT sör
FROM Kedvel, (SELECT név
FROM Látogat
WHERE bár = 'Joe''s bar') AS JD
WHERE Szeret.név = JD.név;

A WHERE záradékban:

a. az alkérdés eredménye lehet egyelten skalárérték
b. lehet skalár értékekből álló multihalmaz, mely logikai kifejezésekkel használható: [NOT] EXISTS, [NOT] IN, [ANY | ALL]
c. lehet egy többdimenziós tábla: [NOT] EXISTS, [NOT] IN

Nem korrelált alkérdés:
Ha az alkérdés nem korrelált, önállóan kiértékelhető és ez az eredmény a külső kérdés közben nem változik, a külső kérdés szempontjából ez egy konstanstábla, akkor a kiértékelés mindig a legbelsőből halad kifelé.

Korrelált alkérdés:
Olyan alkérdés, mely többször kerül kiértékelésre, minden egyes kiértékelés megfelel egy olyan értékadásnak, amely az alkérdésen kívüli sorváltozóból származik.

Példa korrelált alkérdésre:

```
SELECT DISTINCT cím
        FROM Filmek AS Régi
        WHERE év < ANY
                (SELECT év
                FROM Filmek
                WHERE cím = Régi.cím
);
```

7. **Kiterjesztett relációs algebra:**

A relációs algebra műveleteit multihalmazok fölött értelmezzük, ahol megengedett a sorok ismétlődése. Ezen felül új műveletekkel bővítjük:
- Ismétlődések megszűntetése: (δ(R)) – SELECT DISTINCT
- Összesítés, csoportosítás: (γ_{lista}(R)) – GROUP BY
- Vetítési művelet kiterjesztése (π_{lista}(R)) – SELECT kif AS onev
- Rendezési művelet (τ_{lista}(R)) – ORDER BY
- Külső összekapcsolások ($\bowtie$) – [left | right | full] outer join

UNIO, metszet, különbség értelmezve van

8. **Az SQL adatbázisnyelv 4.**

Összekapcsolások:
LEFT, RIGHT, FULL – rendre: csak a bal oldali, csak a jobb oldali, az összes lógó sort megőrzi.
Decartes-szorzat: CROSS JOIN
Természetes: NATURAL JOIN
Théta: R JOIN S ON <feltétel>
Külső: R OUTER JOIN S

9. **ORDER BY, GROUP BY, HAVING záradékok**

10. **INSERT, DELETE, UPDATE utasítások**

11. **Az egyed-kapcsolat (E/K) modell 1.**

    a. **Részei:**

    Egy egyed-kapcsolat modellnek alapvetően három különböző eleme lehet:

    Egyedhalmazok:
    Az egyed valamilyen típusú absztrakt objektum, például legyen ez egy film, egy filmekről szóló adatbázisban. Ezen egyedek halmaza az egyedhalmaz, mint például filmek összessége.

    Attribútumok:
    Az egyedhalmazokhoz tartoznak, melyek az egyedek tulajdonságait írják le. Ilyen lehet például egy adott film címe, hossza. Az attribútumok típusai lehetnek atomiak (általában ez a jellemző adatbázisok esetében), rekordok, valamint atomi vagy rekord típusú elemekből álló halmazok, adott elemtípussal.

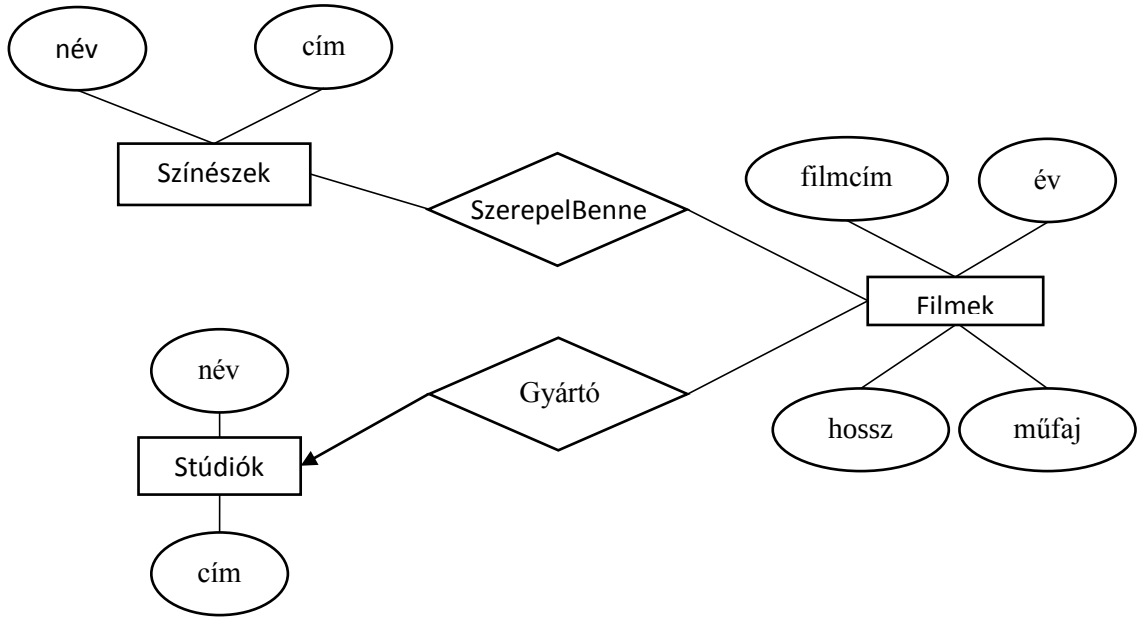
Kapcsolatok:
A kapcsolatok két egyedhalmazt kötnek össze egymással. Ilyen lehet például a filmek és színészek közötti kapcsolat, melyek között felírható egy SzerepelBenne kapcsolat. Ezen kapcsolatok leggyakrabban binárisak, azaz két halmazt kötnek össze, az E/K-modell azonban lehetővé teszi tetszőleges számú egyedhalmaz összekapcsolását, egyetlen kapcsolattal.

b. **E/K diagram**

Az E/K diagram egy gráf, melynek csúcspontjai egyedhalmazoknak, attribútumoknak és kapcsolatoknak felelnek meg. Ezeket különféle alakzatokkal ábrázoljuk:

- az egyedhalmazt téglalappal
- az attribútumot oválissal
- a kapcsolatot rombusszal

Az egyedhalmazokat összekötjük attribútumaikkal, a kapcsolatokat pedig a bennük részt vevő egyedhalmazokkal. Ezek az összekötések a gráf élei.



Az egyed-kapcsolat diagram által leírt adatbázist aktuális adataival együtt az adatbázis **előfordulásának** nevezzük. Ez persze nem azt jelenti, hogy az előfordulás egy létező dolog abban az értelemben, mint ahogy az adatbázisrendszerekben a relációk előfordulása létezik.

c. **Kapcsolatok típusai**

Bináris kapcsolat: egy-egy, sok-egy, sok-sok
ISA kapcsolat: speciális egy-egy kapcsolat, öröklődési kapcsolat. Pl.: a PC **is a** computer. Egy ISA kapcsolatot mint öröklődést, háromszöggel jelölünk.

d. **Szerepek:**

Ha egy egyedhalmaz kétszer vagy többször is szerepel egy kapcsolatban, szerepnek nevezzük.

e. **Kapcsolatok attribútumai:**

Lehetőség van a kapcsolatokhoz attribútumokat rendelni, melyek nem az egyedhalmazok elemeire érvényesek, hanem az azok közötti kapcsolatra, például egy adott színész egy adott filmért adott stúdiónál mekkora fizetést kap.

Kurzorok használata:

Egy SQL lekérdezés visszatérhet egy skalár értékkel, egyetlen sorral, vagy több sorból álló táblával. Az utolsó esetben az eredményt bejárhatjuk egy kurzorral.

Az eredményt elmenthetjük egy váltózóba a SET p = SELECT … paranccsal, vagy a SELECT INTO p parancs segítségével.

Ezután ledeklarálhatjuk a kurzort, megnyitjuk, majd lezárjuk:
DECLARE mutatónév CURSOR FOR (p);

OPEN mutatónév;
-- Köztes kód
CLOSE mutatónév;

Ciklusok:

LOOP
-- Ciklus törzs
ENDLOOP

Vagy el is nevezhetjük:
I:LOOP
-- Ciklus törzs
ENDLOOP

Több loop is egymásba ágyazható! A következő adat lekéréséhez a következő parancsot kell használni:
FETCH FROM mutatónév INTO v1, v2, …, vn; (ahol v1, v2, …, vn változónevek a kurzor által visszaadott értékekhez).

Logikai vizsgálat arra, hogy találtunk-e új sort:
IF mutatónév%NOTFOUND THEN
  LEAVE I;
END IF;

A lekérdezéshez DECLARE változónév TÍPUS; formátummal adhatjuk meg a tároló változókat.

Procedure:

Ugyanazokat használjuk itt is mint egy kurzornál, a felépítés a következő:
CREATE [OR REPLACE] PROCEDURE név
[(parameter_name [IN |OUT | IN OUT] type [, …])]
{IS | AS}
BEGIN
-- törzs
END;

Paraméterek:
IN – csak olvasható bemeneti paraméter
OUT – ezen keresztül térhetünk vissza valamilyen értékkel
IN OUT – mindkettő egyben

Meghívás:
EXECUTE név;
vagy
FüggvényszerűProcedúra(paraméter);

Nézettábla:
CREATE [MATERIALIZED] VIEW <név> AS <lekérdezés>;
Ugyanúgy lekérdezhetők mint egy rendes tábla.