2.előadás: Adatbázisok-I.

dr. Hajas Csilla (ELTE IK)
http://sila.hajas.elte.hu/

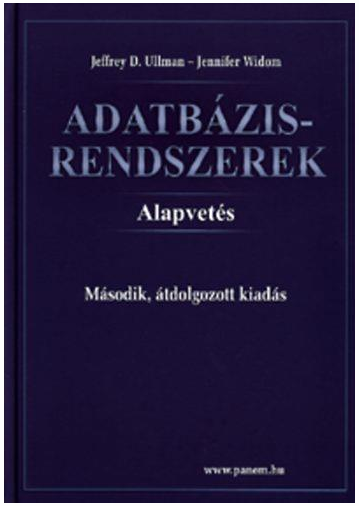
Relációs algebra alapműveletei és SQL SELECT - FROM - WHERE

Tankönyv:

2.4. Lekérdezések: Relációs algebra

6.1. Egy táblára vonatkozó lekérdezések

6.2. Több tábla lekérdezése SQL-ben



Bevezetőben/Ismeri az SQL-t?

- Ki ismeri az SQL-t? Van-e különbség?
- Tetszőleges táblát lekérdezve megegyezik-e az eredmény?

R

A	B
5	20
10	30
20	40
…	…

(1)
```
SELECT B FROM R
    WHERE A<10 OR A>=10;
```

(2)
```
SELECT B FROM R;
```

- Itt mi a helyzet ezzel?

(3)
```
SELECT A FROM R, S
    WHERE R.B = S.B;
```

(4)
```
SELECT A FROM R
    WHERE B IN (SELECT B FROM S);
```

Lekérdezések: Mi az algebra?

- Nyelv: a kérdés szintaktikai alakja és a kérdés kiértékelése (algoritmus) kiértékelési szemantika
- Algebra műveleteket és atomi operandusokat tartalmaz.
- Relációs algebra: az atomi operandusokon és az algebrai kifejezéseken végzett műveletek alkalmazásával kapott relációkon műveleteket adunk meg, kifejezéseket építünk (a kifejezés felel meg a kérdés szintaktikai alakjának).
- Fontos tehát, hogy minden művelet végeredménye reláció, amelyen további műveletek adhatók meg.
- A relációs algebra atomi operandusai a következők:
  - a relációkhoz tartozó változók,
  - konstansok, amelyek véges relációt fejeznek ki.

Relációs algebrai lekérdező nyelv ---1

Relációs algebrai kifejezés, mint lekérdező nyelv

Lekérdező nyelv: L -nyelv

Adott az adatbázis sémája: $\mathbb{R} = \{R_1, \ldots, R_k\}$

$q \in L$ $\quad q: R_1, \ldots, R_k \rightarrow V$ (eredmény-reláció)

E - relációs algebrai kifejezés: $E(R_1, \ldots, R_k) = V$ (output)

Relációs algebrai kifejezések formális felépítése

- Elemi kifejezések (alapkifejezések)

  (i) $R_i \in \mathbb{R}$ (az adatbázis-sémában levő relációnevek)

  R_i kiértékelése: az aktuális előfordulása

  (ii) konstans reláció (véges sok, konstansból álló sor)

- Összetett kifejezések (folyt. köv.oldalon)

Relációs algebrai lekérdező nyelv ---2

(folyt.) Relációs algebrai kifejezések felépítése

- Összetett kifejezések
- Ha E_1, E_2 kifejezések, akkor a következő E is kifejezés
  - $E := \Pi_{lista} (E_1)$ vetítés (típus a lista szerint)
  - $E := \sigma_{Feltétel} (E_1)$ kiválasztás (típus nem változik)
  - $E := E_1 \cup E_2$ unió, ha azonos típusúak (és ez a típusa)
  - $E := E_1 - E_2$ különbség, ha E_1, E_2 azonos típusúak (típus)
  - $E := E_1 \bowtie E_2$ term. összekapcsolás (típus attr-ok uniója)
  - $E := \rho_{S(B1, \ldots, Bk)} (E_1 (A_1, \ldots A_k))$ átnevezés (típ.új attr.nevek)
  - $E := (E_1)$ kifejezést zárójelezve is kifejezést kapunk
- Ezek és csak ezek a kifejezések, amit így meg tudunk adni

Vetítés (project, jelölése pí: Π)

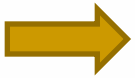
- Vetítés (projekció). Adott relációt vetít le az alsó indexben szereplő attribútumokra (attribútumok számát csökkentik)
- $\Pi_{lista}(R)$ ahol lista: $\{A_{i_1}, \ldots, A_{i_k}\}$ R-sémájában levő attribútumok egy részhalmazának felsorolása

  eredmény típusa $<A_{i_1}$: értéktípus$_{i_1}$, … , A_{i_k}:értéktípus$_{i_k}>$

  $\Pi_{lista}(R) := \{ t.A_{i_1}, t.A_{i_2}, \ldots, t.A_{i_k} \mid t \in R\} = \{ t[lista] \mid t \in R\}$
- Reláció soraiból kiválasztja az attribútumoknak megfelelő $A_{i_1}, \ldots, A_{i_k}$ -n előforduló értékeket, ha többször előfordul akkor a duplikátumokat kiszűrjük (hogy halmazt kapjunk)
- **Példa:**

A	B	C
a	b	c
c	d	e
c	d	d

$\Pi_{A, B}(R)$



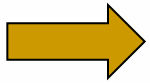
A	B
a	b
c	d

Kiválasztás (select, jelölése szigma: σ)

- Kiválasztás (szűrés). Kiválasztja az argumentumban szereplő reláció azon sorait, amelyek eleget tesznek az alsó indexben szereplő feltételnek.
- $\sigma_{\text{Feltétel}}(R)$ és R sémája megegyezik
- $\sigma_{\text{Feltétel}}(R) := \{ t \mid t \in R \text{ és } t \text{ kielégíti az F feltételt} \}$
- $R(A_1, \ldots, A_n)$ séma feletti reláció esetén a σ_F kiválasztás F feltétele a következőképpen épül fel:
  - elemi feltétel: $A_i\ \theta\ A_j$, $A_i\ \theta$ c, ahol c konstans, θ pedig =, ≠, <, >, ≤, ≥
  - összetett feltétel: ha B_1, B_2 feltételek, akkor $\neg B_1$, $B_1 \wedge B_2$, $B_1 \vee B_2$ és zárójelezésekkel is feltételek
- **Példa:**

A	B	C
a	b	c
c	d	e
g	a	d

$\sigma_{A= 'a' \vee C<= 'd'}(R)$



A	B	C
a	b	c
g	a	d

Halmazműveletek (jelölése a szokásos)

- Reláció előfordulás véges sok sorból álló halmaz. Így értelmezhetők a szokásos halmazműveletek: az unió (az eredmény halmaz, csak egyszer szerepel egy sor) értelmezhető a metszet és a különbség. Milyen művelet van még halmazokon? Értelmezhető-e relációkon?
- R, S és azonos típusú, $R \cup S$ és $R - S$ típusa ugyanez $R \cup S := \{t \mid t \in R \vee t \in S\}$, $R - S := \{ t \mid t \in R \wedge t \notin S\}$
- Az alapműveletekhez az unió és különbség tartozik, metszet műveletet származtatjuk $R \cap S = R - (R - S)$
-

A	B	C
a	b	c
c	d	e
g	a	d

A	B	C
a	b	c
c	d	e
g	d	f

Példa: különbségre

R – S

A	B	C
g	a	d

Példák: Halmazműveletek

Felszolgál1:

Söröző	Sör	Ár
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50

Felszolgál2:

Söröző	Sör	Ár
Joe's	Bud	2.50
Jack's	Bud	2.75

Felszolgál1 ∪ Felszolgál2:

Söröző	Sör	Ár
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Jack's	Bud	2.75

Felszolgál1 ∩ Felszolgál2:

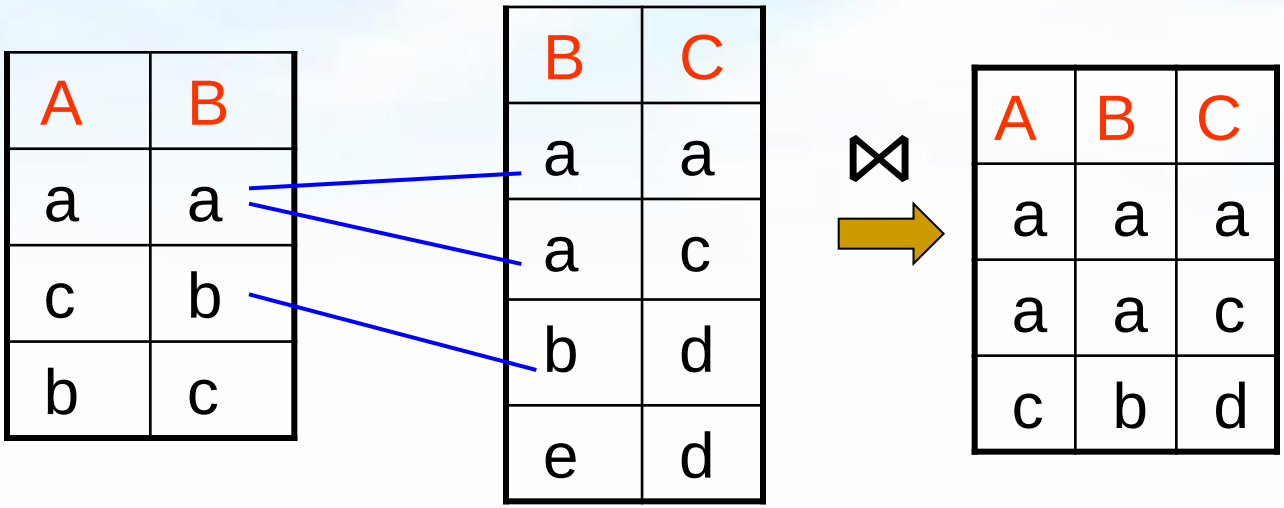
Söröző	Sör	Ár
Joe's	Bud	2.50

Felszolgál2 \ Felszolgál1:

Söröző	Sör	Ár
Jack's	Bud	2.75

Természetes összekapcsolás ---1

- Szorzás jellegű műveletek (attribútumok számát növeli) többféle lehetőség, amelyekből csak egyik alapművelet:
- Angolul: Natural Join (jelölése: „csokornyakkendő”)
- Természetes összekapcsolás: közös attribútum-nevekre épül. R ⋈ S azon sorpárokat tartalmazza R-ből illetve S-ből, amelyek R és S azonos attribútumain megegyeznek.



Természetes összekapcsolás ---2

- Természetes összekapcsolás:
- Legyen $R(A_1,\ldots,A_k,B_1,\ldots,B_n)$, illetve $S(B_1,\ldots,B_n,C_1,\ldots,C_m)$
- $R \bowtie S$ típusa $(A_1,\ldots,A_k,B_1,\ldots,B_n,C_1,\ldots,C_m)$ vagyis a két attribútum-halmaz uniója
- $R \bowtie S = \{ <A_1: t(A_1),\ldots, A_k: t(A_k), B_1: t(B_1),\ldots, B_n: t(B_n),$
  $C_1: s(C_1),\ldots, C_m: s(C_m)> \mid t \in R, s \in S,$
  $t(B_i) = s(B_i)\ i=1, \ldots, n \}$
- $R \bowtie S$ elemei $v \in R \bowtie S$

  $R \bowtie S = \{ v \mid \exists\ t \in R, \exists\ s \in S: t[B_1,\ldots,B_n] = s[B_1,\ldots,B_n] \wedge$
  $\wedge\ v[A_1,\ldots,A_k] = t[A_1,\ldots,A_k] \wedge v[B_1,\ldots,B_n] = t[B_1,\ldots,B_n] \wedge$
  $\wedge\ v[C_1,\ldots,C_m] = s[C_1,\ldots,C_m] \}$

Természetes összekapcsolás ---3

- Példákban: két azonos nevű attribútumot úgy tekintünk, hogy ugyanazt jelenti és a közös érték alapján fűzzük össze a sorokat.
- Milyen problémák lehetnek?
- Filmek adatbázisban ugyanarra a tulajdonságra más névvel hivatkozunk: Filmek.év és SzerepelBenne.filmÉv, illetve FilmSzínész.név és SzerepelBenne.színészNév
- Termékek adatbázisban pedig ugyanaz az azonosító mást jelent: Termék.típus más, mint Nyomtató.típus
- Emiatt a Filmek és a Termékek adatbázisokban ahhoz, hogy jól működjön az összekapcsolás szükségünk van egy technikai műveletre, és ez: az átnevezés (rename)

Átnevezés (rename, jelölése ró: ϱ)

- Miért van erre szükség? Nem tudjuk a reláció saját magával való szorzatát kifejezni, $R \bowtie R = R$ lesz.
- Láttuk, hogy egyes esetekben szükség lehet relációnak vagy a reláció attribútumainak átnevezésére:

  $$\rho_{T(B_1, \ldots, B_k)}(R(A_1, \ldots A_k))$$

- Ha az attribútumokat nem szeretnénk átnevezni, csak a relációt, ezt $\rho_T(R)$-rel jelöljük. Ha ugyanazt a táblát használjuk többször, akkor a táblának adunk másik hivatkozási (alias) nevet.
- Az attribútumok átnevezése helyett alternatíva: R.A (vagyis relációnév.attribútumnév hivatkozás) amivel meg tudjuk különböztetni a különböző táblákból származó azonos nevű attribútumokat.

Szorzás jellegű műveletek ---1

- Szorzás jellegű műveletek többféle lehetősége közül csak az egyiket vesszük alapműveletnek: join vagy természetes összekapcsolást, amely közös attribútumnevekre épül. $R \bowtie S$ azon sorpárokat tartalmazza R-ből illetve S-ből, amelyek R és S azonos attribútumain megegyeznek.
- Egy másik lehetőség: direkt-szorzat (Descartes-szorzat) Ez is tekinthető alapműveletnek (és bizonyos esetekben egyszerűbb ezt venni alapműveletnek) az ennél sokkal gyakrabban használt természetes összekapcsolás helyett.
- $R \times S$: az R és S minden sora párban összefűződik, az első tábla minden sorához hozzáfűzzük a második tábla minden sorát

$$R \times S := \{ t \mid t[R] \in R \text{ és } t[S] \in S \}$$

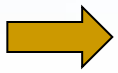
Szorzás jellegű műveletek ---2

- A direkt-szorzat (vagy szorzat, Descartes-szorzat) esetén természetesen nem fontos az attribútumok egyenlősége. A két vagy több reláció azonos nevű attribútumait azonban meg kell különböztetni egymástól.
  Hivatkozás séma: oszlopok átnevezése illetve azonos nevű oszlop esetén: $R.A_1, \ldots, R.A_k, S.A_1, \ldots, S.A_k$
- **Példa:**

$R \times S$

A	B	C
a	b	c
c	d	e
g	a	d

B	D
b	r
q	s



A	R.B	C	S.B	D
a	b	c	b	r
a	b	c	q	s
c	d	e	b	r
c	d	e	q	s
g	a	d	b	r
g	a	d	q	s

Szorzás jellegű műveletek ---3

Théta-join

Felszolgál:

Söröző	Sör	Ár
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

Söröző

Név	Cím
Joe's	Maple st.
Sue's	River rd.

Barinfo= Felszolgál $\bowtie_{\text{Felszolgál.söröző = Söröző.név}}$ Söröző

Söröző	Sör	Ár	Név	Cím
Joe's	Bud	2.50	Joe's	Maple st.
Joe's	Miller	2.75	Joe's	Maple st.
Sue's	Bud	2.50	Sue's	River rd.
Sue's	Miller	3.00	Sue's	River rd.

Szorzás jellegű műveletek ---4

- Ha R, S sémái megegyeznek, akkor $R \bowtie S = R \cap S$.
- Ha R, S sémáiban nincs közös attribútum, akkor
  $R \bowtie S = R \times S$.
- Théta összekapcsolás $\bowtie_\theta$, félig összekapcsolás $\ltimes$, és a rel.algebra kiterjesztésénél külső összekapcsolásokat a következő 3.előadáson folytatjuk.
- Feladatok: Hogyan fejezhető ki az R x S direkt szorzat relációs algebrában? (a természetes összekapcsolást tekintjük alapműveletnek, ebből és az átnevezés segítségével felírható a direkt szorzat).
- Hogyan fejezhető ki a természetes összekapcsolás, ha a direkt szorzatot sorolnánk az alapműveletek közé?

Egyszerű egy táblára vonatkozó lekérdezések az SQL-ben

Példa – Sörivók adatbázisséma

- Az előadások SQL lekérdezései az alábbi Sörivók adatbázissémán alapulnak

  (aláhúzás jelöli a kulcs attribútumokat)

Sörök(<u>név</u>, gyártó)
Sörözők(<u>név</u>, város, tulaj, engedély)
Sörivók(<u>név</u>, város, tel)

Szeret(<u>név</u>, <u>sör</u>)
Felszolgál(<u>söröző</u>, <u>sör</u>, ár)
Látogat(<u>név</u>, <u>söröző</u>)

Egyszerű példa Select-From-Where-re

- Használjuk Sörök(név, gyártó) relációsémát, mely söröket gyártja a Dreher?

```
SELECT név
FROM Sörök
WHERE gyártó = 'Dreher';
```

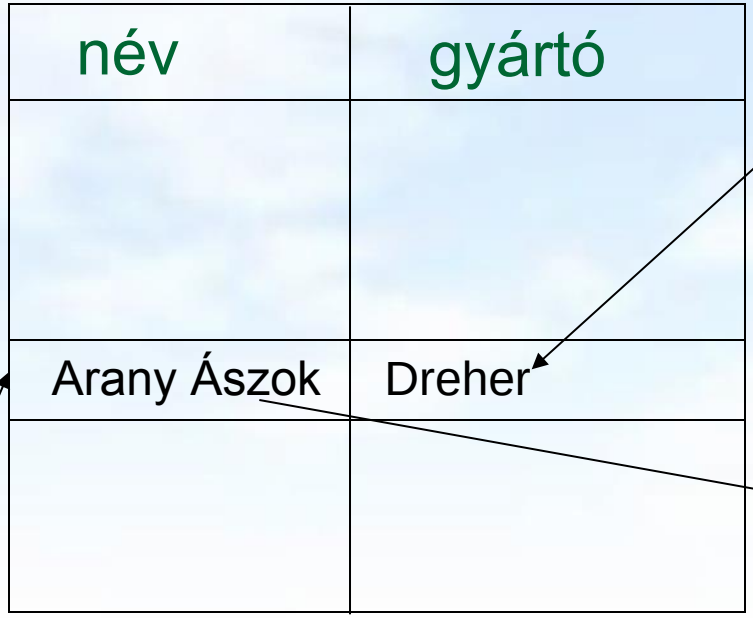
A lekérdezés eredménye

név
Arany Ászok
Dreher Classic
. . .

A lekérdezés eredménye egy reláció, amelynek egy attribútuma van (név) és a sorai az összes olyan sör neve, amelyet a Dreher gyárt.

Eltérés a relációs algebrától: Az SQL alapértelmezésben nem szűri ki a duplikátumokat, az eredmény multihalmaz.

A műveletek szemantikája



1) Ellenőrizzük a feltételt, hogy a gyártó Dreher-e

2) Ha a feltétel teljesült, akkor képezünk egy t eredménysort

3) Ebből a t sorból a SELECT listának megfelelő típusú sort képezzük, példa: t.név

Az egytáblás SFW alapértelmezése

SELECT [DISTINCT] kif_1 [[AS] $onév_1$], … , kif_n [$onév_n$]
FROM táblanév [sorváltozó]
[WHERE feltétel]

Alapértelmezés (a műveletek szemantikája -- általában)

- A FROM záradékban levő relációhoz tekintünk egy **sorváltozót**, amely a reláció minden sorát bejárja
- Minden egyes „aktuális” sorhoz **kiértékeljük** a WHERE záradékot
- Ha helyes (vagyis **igaz**) választ kaptunk, akkor képezünk egy sort a SELECT záradékban szereplő kifejezéseknek megfelelően.

SELECT záradékban * jelentése

- Amikor csak egy reláció van a FROM záradékban, akkor a SELECT záradékban levő * jelentése: „a reláció minden attribútuma”
- Példa: Keressük a Sörök(név, gyártó) tábla alapján a Dreher-sörök adatait.
- A lekérdezés eredménye

```
SELECT *
FROM Sörök
WHERE gyártó = 'Dreher';
```

- A lekérdezés eredménye a Sörök tábla összes attribútumát tartalmazza. Első lépésben (kezdő gyakorlásnál kicsi táblákra) mindig lekérdezzük előbb a tábla tartalmát: SELECT * FROM Táblanév;

Attribútumok átnevezése

- Ha az eredményben (a fejlécben) más attribútumnevet szeretnénk használni, akkor “[AS] új_oszlopnév” segítségével tudunk más oszlopnevet kiírni.
  (Oracle: másodnévben nem kell ‘AS', csak szóköz)
- Listán azt értjük, hogy vesszővel vannak elválasztva az elemek (attribútumnevek), ha a másodnévben szóköz szerepel, akkor azt macskaköröm közé kell tenni: ”… ”
- Példa: Sörök(név, gyártó)

```
SELECT név sör, gyártó ”Dreher gyártó”
FROM Sörök
WHERE gyártó = 'Dreher';
```

- A lekérdezés eredményében az új oszlopnevek lesznek.

SELECT záradékban levő kifejezések

- Az attribútumnevek helyett tetszőleges kifejezések állhatnak (amelyek megfelelnek az adott típusra) a SELECT záradék elemeként.
- Lásd bővebben majd a gyakorlatok példáit, feladatait, felhasználjuk az Oracle DB SQL Language Reference megfelelő fejezeteit: Operators, Functions, Expressions.
- Példa: Felszolgál(söröző, sör, ár)

```
SELECT söröző, sör, ár*114 árYenben
FROM Felszolgál;
```

- Konstansok a kifejezésekben Szeret(név, sör):

```
SELECT név DABkedvelő
FROM Szeret
WHERE upper(sör) = 'DAB';
```

WHERE záradék (összetett feltételek)

- Hasonlóan, mint a relációs algebra kiválasztás (σ) feltételében elemi feltételekből építkezünk, ahol elemi feltételen két kifejezés =, <>, <, >, <=, >= aritmetikai összehasonlítását, a theta műveletet értjük.
- Logikai műveletek AND, OR, NOT és zárójel ( ) segítségével kapjuk az összetett feltételeket.
- Példa: Felszolgál (söröző, sör, ár) relációséma esetén keressük a „Joe's Bar”-ban árult „DAB” sörök árát:

```
SELECT ár
FROM Felszolgál
WHERE söröző = 'Joe''s Bar' AND
      sör = 'DAB';
```

WHERE záradék (további lehetőségek)

SQL specialitások, amelyek könnyen átírhatóak relációs algebrai kifejezésre (összetett kiválasztási feltételre)

- BETWEEN .. AND .. intervallumba tartozás
- IN (értékhalmaz) egyszerű értékek halmaza

SQL specialitások, nem írhatók át relációs algebrába:

(--- ezek jönnek a köv.lapon…)

- Karakterláncok LIKE összehasonlítása mintákkal
- IS NULL összehasonlítás

LIKE

- Karakterláncok összehasonlítása mintákkal:
  - <attribútum> LIKE <minta> vagy
  - <attribútum> NOT LIKE <minta>
- Minta egy olyan karakterlánc, amelyben használhatjuk a speciális % és _ karaktereket. A mintában % megfelel bármilyen karakterláncnak és _ bármilyen karakternek.
- **Példa:** Azokat a sörözőket keressük, amelyik nevének a második betűje „a” vagy a nevében van „‘s”, mint ahogyan például a „*Joe's Bar*” névben is szerepel:

```
SELECT név FROM Sörözők
WHERE név LIKE '_a%' OR
      név LIKE '%''s%';
```

NULL (hiányzó) értékek

- Az SQL lehetővé teszi, hogy a relációk soraiban az attribútum értéke egy speciális NULL nullérték legyen.
- **A nullérték értelmezésére** több lehetőségünk is van:
  - Hiányzó érték: például tudom, „Joe's Bár”-jának van valamilyen címe, de nem tudom, hogy mi az.
  - Nem-definiált érték: például a házastárs attribútumnak egyedülálló embereknél nincs olyan értéke, aminek itt értelme lenne, nincs házastársa, ezért nullérték.
- **Where záradékban a nullérték** vizsgálata:
  - IS NULL
  - IS NOT NULL

NULL értékek használata

- **Where záradékban a nullérték** használata:
  - Amikor egy aritmetikai műveletben az egyik tag NULL, akkor az eredmény is NULL.
  - Amikor egy NULL értéket hasonlítunk össze bármely más értékkel (beleértve a NULL-t is) az összehasonlítási operátorok (=, <>, <, <=, >, >=) segítségével, akkor az eredmény UNKNOWN (ismeretlen).

Az ismeretlen (unknown) igazságérték

- Az SQL-ben szereplő logikai feltételek valójában háromértékű logika: TRUE, FALSE, UNKNOWN (magyarban igaz, hamis, ismeretlen rövidítése miatt inkább meghagyjuk az angol T, F, U rövidítéseket).
- A WHERE záradékban szereplő logikai feltételt a rendszer minden egyes sorra ellenőrzi és a logikai érték TRUE, FALSE vagy UNKNOWN valamelyike lehet, de az eredménybe csak azok a sorok kerülnek, amelyeknek a feltétel kiértékelése TRUE értéket adott.

A háromértékű logika

- Hogyan működnek az AND, OR, és NOT logikai műveletek a 3-értékű logikában?
- A szabályt könnyű megjegyezni, ha úgy tekintjük, hogy TRUE = 1, FALSE = 0, és UNKNOWN = ½.
- Ekkor AND = MIN, OR = MAX, NOT(*x*) = 1-*x*.
- Példa:

  TRUE AND (FALSE OR NOT(UNKNOWN)) =

  MIN(1, MAX(0, (1 - ½))) =

  MIN(1, MAX(0, ½)) =

  MIN(1, ½) = ½ = UNKNOWN
- A 3-értékű logika AND, OR és NOT igazságtáblázatát lásd a Tk. 6.2.ábráját (vagy kitöltése a fenti szabállyal)

A háromértékű logika (Tk.6.2. ábra)

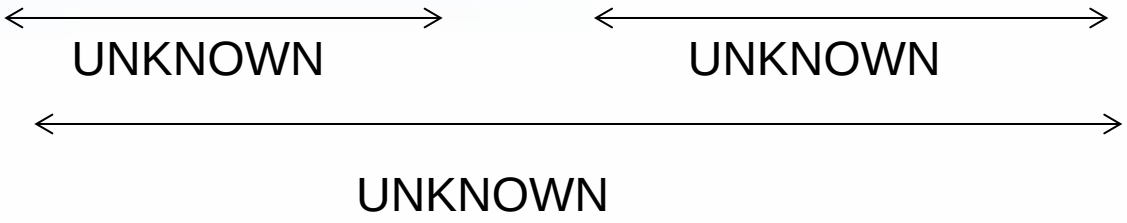
x	*y*	*x* AND *y*	*x* OR *y*	NOT *x*
T	T	T	T	F
T	U	U	T	F
T	F	F	T	F
U	T	U	T	U
U	U	U	U	U
U	F	F	U	U
F	T	F	T	T
F	U	F	U	T
F	F	F	F	T

Egy meglepő példa

- Példa: Felszolgál reláció legyen az alábbi:

söröző	sör	ár
Joe's Bar	Bud	NULL

```
SELECT söröző
FROM Felszolgál
WHERE ár < 2.00   OR   ár >= 2.00;
```



Oka: a 2-értékű != 3-értékű szabályok

- Bizonyos általános szabályok, mint például, hogy az AND kommutatív érvényes a 3-értékű logikában is.
- Ellenben nem igaz, például a kizáró szabály, vagyis

  p OR NOT *p* = TRUE nem teljesül, ha *p* = UNKNOWN, mert ekkor a baloldal: MAX(½, (1 – ½)) = ½ != 1 vagyis a 3-értékű logikában baloldal értéke nem TRUE.
- Ezért az előző példában nem az eredeti egy soros táblát, hanem az üres táblát (amelynek egy sora sincs) kaptuk meg az eredménytáblaként.

Az eredmény rendezése

- SQL SELECT utasításban a záradékok
- Az SQL lehetővé teszi, hogy a lekérdezés eredménye bizonyos sorrendben legyen rendezve. Az első attribútum egyenlősége esetén a 2.attribútum szerint rendezve, stb, minden attribútumra lehet növekvő vagy csökkenő sorrend.
- Select-From-Where utasításhoz a következő záradékot adjuk, a WHERE záradék és minden más záradék (mint például GROUP BY és HAVING) után következik:

  SELECT … FROM … [WHERE …][…]

  ORDER BY {attribútum [DESC], …}

- **Példa: SELECT * FROM Felszolgál**

  ORDER BY ár DESC, sör

Több táblára vonatkozó lekérdezések az SQL-ben

Select-From-Where (SFW) utasítás

- Gyakran előforduló relációs algebrai kifejezés

  Π_{Lista} (σ_{Felt} (R_1 x… x R_n)) típusú kifejezések
- Szorzat és összekapcsolás az SQL-ben
- SELECT s-lista -- milyen típusú sort szeretnénk az eredményben látni?

  FROM f-lista -- relációk (táblák) összekapcsolása, illetve szorzata

  WHERE felt -- milyen feltételeknek eleget tevő sorokat kell kiválasztani?
- FROM f-lista elemei (ezek ismétlődhetnek)

  táblanév [[AS] sorváltozó, …]

  Itt: a from lista elemei a táblák direkt szorzatát jelenti, az összekapcsolási feltételt where-ben adjuk meg, később bevezetünk majd tovább lehetőségeket a különböző összekapcsolásokra az SQL from záradékában.

Attribútumok megkülönböztetése ---1

- Milyen problémák merülnek fel?
- (1) Ha egy attribútumnév több sémában is előfordul, akkor nem elég az attribútumnév használata, mert ekkor nem tudjuk, hogy melyik sémához tartozik.
- Ezt a problémát az SQL úgy oldja meg, hogy megengedi egy relációnévnek és egy pontnak a használatát egy attribútum előtt: **R.A** (az R reláció A attribútumát jelenti).
- Természetes összekapcsolás legyen R(A, B), S(B,C)

```
SELECT A,  R.B  B,  C
FROM R, S
WHERE R.B=S.B;
```

Attribútumok megkülönböztetése ---2

- Milyen problémák merülnek még fel?
- (2) Ugyanaz a reláció többször is szerepelhet, vagyis szükség lehet arra, hogy ugyanaz a relációnév többször is előforduljon a FROM listában.
- Ekkor a FROM listában a táblához másodnevet kell megadni, erre sorváltozóként is szoktak hivatkozni, megadjuk, h. melyik sorváltozó melyik relációt képviseli:

  FROM R_1 [t_1], …, R_n [t_n]

  Ekkor a SELECT és WHERE záradékok kifejezésekben a hivatkozás: **t_i.A** (vagyis sorváltozó.attribútumnév)

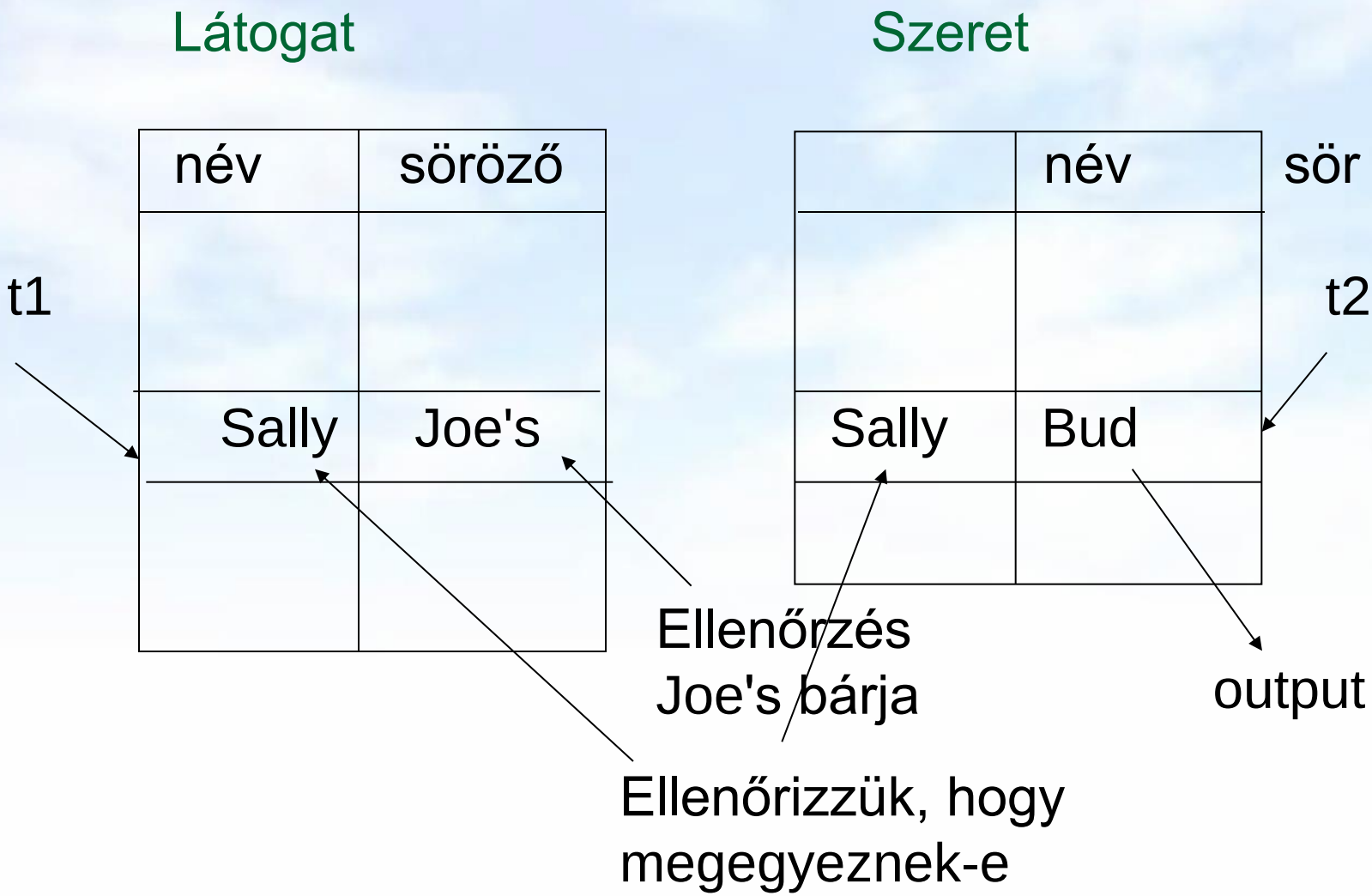
Példa: Két tábla összekapcsolása ---1

- Mely söröket szeretik a Joe's Bárba járó sörivók?

```
SELECT sör
   FROM Szeret, Látogat
   WHERE söröző = 'Joe''s Bar'
      AND Látogat.név = Szeret.név;
```

- Kiválasztási feltétel: `söröző = 'Joe''s Bar'`
- Összekapcsolási feltétel: `Látogat.név = Szeret.név`
- Alapértelmezése a következő oldalon a mai órán
- Összekapcsolások szintaxisát később nézzük majd

Példa: Két tábla összekapcsolása ---2



SFW szabvány alapértelmezése ---1

- Kiindulunk a **FROM záradékból**: a FROM lista minden eleméhez egy beágyazott ciklus, végigfut az adott tábla sorain a ciklus minden lépésénél az n darab sorváltozónak lesz egy-egy értéke
- ehhez kiértékeljük a WHERE feltételt, vagyis elvégezzük a **WHERE záradékban** szereplő feltételnek eleget tevő sorok kiválasztását (csak a helyesek, ahol TRUE=igaz választ kapunk), azok a sorok kerülnek az eredménybe.
- Alkalmazzuk a **SELECT záradékban** jelölt kiterjesztett projekciót. Az SQL-ben az eredmény alapértelmezés szerint itt sem halmaz, hanem multihalmaz.

  Ahhoz, hogy halmazt kapjunk, azt külön kérni kell: SELECT **DISTINCT** Lista

SFW szabvány alapértelmezése ---2

FOR *t*1 sorra az *R*1 relációban DO

FOR *t*2 sorra az *R*2 relációban DO

...

FOR *tn* sorra az *Rn* relációban DO

IF a where záradék igaz, amikor az attribútumokban *t*1, *t*2, ..., *tn* megfelelő értékei találhatóak

THEN

*t*1, *t*2, ..., *tn* -nek megfelelően kiértékeljük a select záradék attribútumait

és az értékekből alkotott sort

az eredményhez adjuk

SFW szabvány alapértelmezése ---3

SELECT [DISTINCT] kif_1 [[AS] $onév_1$], …, kif_n [[AS] $onév_n$]
FROM R_1 [t_1], …, R_n [t_n]
WHERE feltétel (vagyis logikai kifejezés)

Alapértelmezés (a műveletek szemantikája -- általában)

- A FROM záradékban levő relációkhoz tekintünk egy-egy sorváltozót, amelyek a megfelelő reláció minden sorát bejárják (beágyazott ciklusban)
- Minden egyes „aktuális” sorhoz kiértékeljük a WHERE záradékot
- Ha helyes (vagyis igaz) választ kaptunk, akkor képezünk egy sort a SELECT záradékban szereplő kifejezéseknek megfelelően.

Megj.: konverzió relációs algebrába

SELECT [DISTINCT] kif_1 [[AS] $onév_1$], …, kif_n [[AS] $onév_n$]

FROM R_1 [t_1], …, R_n [t_n]

WHERE feltétel (vagyis logikai kifejezés)

- 1.) A FROM záradék sorváltozóiból indulunk ki, és tekintjük a hozzájuk tartozó relációk Descartes-szorzatát. Átnevezéssel valamint R.A jelöléssel elérjük, hogy minden attribútumnak egyedi neve legyen.
- 2.) A WHERE záradékot átalakítjuk egy kiválasztási feltétellé, melyet alkalmazunk az elkészített szorzatra.
- 3.) Végül a SELECT záradék alapján létrehozzuk a kifejezések listáját, a (kiterjesztett) vetítési művelethez.

$$\Pi_{onév1,\ldots,\ onévn}\ (\ \sigma_{feltétel}\ (R_1 \times \ldots \times R_n\ ))$$

Tábla önmagával való szorzata ---1

- Bizonyos lekérdezéseknél arra van szükségünk, hogy ugyanannak a relációnak több példányát vegyük.
- Ahhoz, hogy meg tudjuk különböztetni a példányokat a relációkat átnevezzük, másodnevet adunk, vagyis sorváltozókat írunk mellé a FROM záradékban.
- A relációkat mindig átnevezhetjük ily módon, akkor is, ha egyébként nincs rá szükség (csak kényelmesebb).
- **Példa: R(Szülő, Gyerek)** séma feletti relációban adott szülő-gyerek adatpárokból állítsuk elő a megállapítható Nagyszülő-Unoka párokat!

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

Tábla önmagával való szorzata ---2

- **Példa: Sörök(név, gyártó)** tábla felhasználásával keressük meg az összes olyan sörpárt, amelyeknek ugyanaz a gyártója.
  - Ne állítsunk elő (Bud, Bud) sörpárokat.
  - A sörpárokat ábécé sorrendben képezzük, például ha (Bud, Miller) szerepel az eredményben, akkor (Miller, Bud) ne szerepeljen.

```
SELECT s1.név, s2.név
FROM Sörök s1, Sörök s2
WHERE s1.gyártó = s2.gyártó
   AND s1.név < s2.név;
```

Halmazműveletek az SQL-ben

- Mi hiányzik még, hogy a relációs algebra alapműveleteit mindet az SQL-ben vissza tudjuk adni?
- A relációs algebrai halmazműveletek: unió, különbség mellett az SQL-ben ide soroljuk a metszetet is (ugyanis fontos a metszet és az SQL-ben is implementálva van).
- Az SQL-ben a halmazműveleteket úgy vezették be, hogy azt mindig két lekérdezés között lehet értelmezni, vagyis nem relációk között, mint R U S, hanem lekérdezem az egyiket is és a másikat is, majd a lekérdezések unióját veszem.

  (SFW-lekérdezés1)
  [**UNION** | **INTERSECT** | {**EXCEPT** | **MINUS**}]
  (SFW-lekérdezés2);

Példa: Intersect (metszet)

- Szeret(név, sör), Felszolgál(söröző, sör, ár) és Látogat(név, söröző) táblák felhasználásával keressük azokat (név,sör) párokat, ahol a név = sörivó látogat olyan söröző̋t, ahol felszolgálnak olyan sört, amelyet szeret (a „boldog” sörivók).

Trükk: itt ez az alkérdés valójában az adatbázisban tárolt tábla

```
(SELECT * FROM Szeret)
  INTERSECT
(SELECT név, sör
 FROM Látogat L, Felszolgál F
 WHERE L.söröző = F.söröző);
```

(név, sör) párok, ahol a sörivó látogat olyan söröző̋t, ahol ezt a sört felszolgálják

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?
- Összefoglalva: SQL lekérdezések megalapozása: Az alap relációs algebra (Tk.2.4)
- SELECT - FROM - WHERE lekérdezések és halmazműveletek SQL-ben (Tk.6.1-6.2)
- Következő előadáson:
  Alkérdések használata az SQL SELECT utasítás FROM és WHERE záradékaiban, továbbá összekapcsolások az SQL-ben (Tankönyv 6.3)

Rel.algebrai feladatok a gyakorlatra

Legyen adva a szeret(Név, Gyümölcs) sémájú reláció

Fejezzük ki alap relációs algebrában a lekérdezéseket:

- Kik szeretik az almát?
- Kik nem szeretik az almát? (de valami mást igen)
- Kik szeretik vagy az almát vagy a körtét?
- Kik szeretik az almát is és a körtét is?
- Kik szeretik az almát, de nem szeretik a körtét?
- Kik szeretnek legalább kétféle gyümölcsöt?
- Kik szeretnek legalább háromféle gyümölcsöt?
- Kik szeretnek legfeljebb kétféle gyümölcsöt?
- Kik szeretnek pontosan kétféle gyümölcsöt?