27. Lekérdezés átírás relációs adatbázisokban

Az első kötetben (lásd 12. fejezet) bevezettük a relációs adatbázisok alapfogalmait, többek között a relációs séma, reláció, reláció példány fogalmát. Az adatbázisokat tervezői oldalról közelítettük meg, a fő kérdés az volt, hogyan érhetjük el, hogy elkerüljük adatok redundáns tárolását, illetve az adatbázis használata során fellépő különböző anomáliákat.

Jelen fejezetben a sémát adottnak tekintjük és megpróbáljuk a felhasználó kérdéseit minél gyorsabban és teljesebben megválaszolni. Ehhez először áttekintjük az alapvető (elméleti) lekérdezési nyelveket, az ezek közti kapcsolatokat.

A fejezet második részében a nézeteket tárgyaljuk. Informálisan, egy nézet nem más, mint egy lekérdezés eredménye. Bemutatjuk a nézetek kapcsolatát lekérdezések gyorsításával, a fizikai adatfüggetlenség biztosításával, valamint az adatok integrálásával.

A fejezet harmadik részében lekérdezések átírásával foglalkozunk.

27.1. Lekérdezések

Tekintsük a budapesti mozihálózat adatbázisát. Tegyük fel, hogy a séma három relációból áll:

$$\textbf{PestiMűsor} = \{\mathit{Filmek}, \mathit{Mozik}, \mathit{Műsor}\}. \tag{27.1}$$

Az egyes relációk sémái a következőek:

$$\begin{aligned} \mathit{Filmek} &= \{\textbf{C}\mathit{ím}, \textbf{R}\mathit{endező}, \textbf{Sz}\mathit{ínész}\}, \\ \mathit{Mozik} &= \{\textbf{M}\mathit{ozi}, \textbf{U}\mathit{tca}, \textbf{T}\mathit{elefon}\}, \\ \mathit{Műsor} &= \{\textbf{M}\mathit{ozi}, \textbf{C}\mathit{ím}, \textbf{I}\mathit{dőpont}\}. \end{aligned} \tag{27.2}$$

Az egyes relációk példányainak lehetséges részletei a 27.1. ábrán láthatók.

Tipikus felhasználói kérdések lehetnek:

27.1 Ki rendezte a “Kontroll”-t?

27.2 Listázzuk az összes olyan mozi nevét és címét, ahol Kuroszava filmet játszanak!

27.3 Adjuk meg azon rendezők nevét, akik szerepeltek valamelyik saját filmjükben!

Ezek a kérdések leképezéseket definiálnak a **PestiMűsor** adatbázis séma relációiból valamilyen másik sémába (jelen esetben egyetlen relációból álló sémákba). Formálisan meg kell különböztetnünk a ***lekérdezést*** és a ***lekérdezés függvényt.*** Az előbbi szintaktikus fogalom, az utóbbi pedig leképezés a bemeneti sémához tartozó példányok halmazából a kimeneti

Filmek

C*ím*	**R***endező*	**Sz***ínész*
Kontroll	Antal Nimród	Csányi Sándor
Kontroll	Antal Nimród	Mucsi Zoltán
Kontroll	Antal Nimród	Pindroch Csaba
⋮	⋮	⋮
A vihar kapujában	Kuroszava Akira	Mifune Tosiro
A vihar kapujában	Kuroszava Akira	Kjó Macsiko
A vihar kapujában	Kuroszava Akira	Maszajuki Mori

Mozik

M*ozi*	**U***tca*	**T***elefon*
Bem	II., Margit krt. 5/b.	316-8708
Corvin Budapest Filmpalota	VIII., Corvin köz 1.	459-5050
Európa	VII., Rákóczi út 82.	322-5419
Művész	VI., Teréz krt. 30.	332-6726
⋮	⋮	⋮
Uránia Nemzeti Filmszínház	VIII., Rákóczi út 21.	486-3413
Vörösmarty	VIII., Üllői út 4.	317-4542

Műsor

M*ozi*	**C***ím*	**I***dőpont*
Bem	A vihar kapujában	19:00
Bem	A vihar kapujában	21:30
Uránia Nemzeti Filmszínház	Kontroll	18:15
Művész	A vihar kapujában	16:30
Művész	Kontroll	17:00
⋮	⋮	⋮
Corvin Budapest Filmpalota	Kontroll	10:15

27.1. ábra. A **PestiMűsor** adatbázis.

séma példányainak halmazába, amit a lekérdezés határoz meg, valamilyen alkalmas szemantikus értelmezés szerint. Azonban az egyszerűség kedvéért mindkét fogalomra a "lekérdezés" szót használjuk, a környezetből mindig világos lesz, hogy éppen melyikről beszélünk.

27.1. definíció. *Az* $\mathbf{R}$ *bemeneti séma feletti* q_1 *és* q_2 *lekérdezések* ***ekvivalensek,*** *jelölésben* $q_1 \equiv q_2$*, ha ugyanaz a kimeneti sémájuk, és minden* $\mathbf{R}$*-hez tartozó* $\mathcal{I}$ *példányra* $q_1(\mathcal{I}) = q_2(\mathcal{I})$*.*

A fejezet további részében áttekintjük a legfontosabb lekérdezési nyelveket. Szükségünk lesz a lekérdezési nyelvek kifejező erejének összehasonlítására.

27.2. definíció. *Legyenek* $\mathcal{Q}_1$ *és* $\mathcal{Q}_2$ *lekérdezési nyelvek (a megfelelő értelmezéssel).* $\mathcal{Q}_2$ ***gazdagabb,*** *mint* $\mathcal{Q}_1$ *(* $\mathcal{Q}_1$ ***szűkebb,*** *mint* $\mathcal{Q}_2$*), jelölésben* $\mathcal{Q}_1 \sqsubseteq \mathcal{Q}_2$*, ha minden* q_1 $\mathcal{Q}_1$*-beli lekérdezéshez van* $q_2 \in \mathcal{Q}_2$*, amelyre* $q_1 \equiv q_2$*.* $\mathcal{Q}_1$ *és* $\mathcal{Q}_2$ ***ekvivalensek,*** *ha* $\mathcal{Q}_1 \sqsubseteq \mathcal{Q}_2$ *és* $\mathcal{Q}_1 \sqsupseteq \mathcal{Q}_2$*.*

27.1. példa. *Lekérdezés.* Tekintsük a 27.2. kérdést. Első közelítésben a következő megoldást találjuk

> **if** léteznek a *Filmek*, *Mozik* és *Műsor* relációkban $(x_C$, "Kuroszava Akira", $x_{Sz})$, (x_M, x_U, x_T) és (x_M, x_C, x_I) sorok
> **then** vegyük be a $(Mozi : x_M, Utca : x_U)$ sort az eredmény relációba.

$x_C, x_{Sz}, x_M, x_I, x_U, x_T$ különböző változókat jelölnek, amelyek az értékeiket a megfelelő attribútum értelmezési tartományából veszik fel. Ugyanazon változók használatával közvetetten jeleztük, hogy a különböző sorokban hol kell egyenlő értékeknek szerepelniük.

27.1.1. Konjunktív lekérdezések

A lekérdezések legegyszerűbb és legtöbb jó tulajdonsággal rendelkező fajtája. Három, egymással ekvivalens változatát ismertetjük, amelyek közül kettő logikai alapú, a harmadik pedig algebrai. A név a logikai változatból ered, olyan elsőrendű kifejezéseken alapszik, amelyek csak egzisztenciális kvantorokat ($\exists$), valamint "és"-el (konjunkcióval) összekötött atomi kifejezéseket tartalmaznak.

Datalog – szabály alapú lekérdezés

Az $(x_1, x_2, \ldots, x_m)$ sort ***szabad sornak*** nevezzük, ha az x_i-k változók vagy konstansok. A szabad sor a reláció példány egy sorának általánosítása. A 27.1. példában szereplő $(x_C$, "Kuroszava Akira", $x_{Sz})$ sor szabad.

27.3. definíció. *Legyen* $\mathbf{R}$ *relációs adatbázis séma.* ***Szabály alapú konjunktív lekérdezésen*** *a következő alakú kifejezést értjük*

$$val(u) \leftarrow R_1(u_1), R_2(u_2), \ldots, R_n(u_n), \tag{27.3}$$

ahol $n \geq 0$*,* $R_1, R_2, \ldots, R_n$ $\mathbf{R}$*-beli reláció nevek, val olyan reláció név, ami nincs* $\mathbf{R}$*-ben;* $u, u_1, u_2, \ldots, u_n$ *szabad sorok. Minden u-ban előforduló változónak elő kell fordulnia* $u_1, u_2, \ldots, u_n$ *valamelyikében is.*

A szabály alapú konjunktív lekérdezéseket egyszerűbben csak ***szabályoknak*** is nevezzük. $val(u)$ a szabály ***feje,*** $R_1(u_1), R_2(u_2), \ldots, R_n(u_n)$ a szabály ***teste.*** $R_i(u_i)$-t ***(relációs) atomnak*** nevezzük. Feltesszük, hogy a fejben előforduló összes változó előfordul valamelyik testbeli atomban is.

Egy szabályt úgy tekinthetünk, mint valamilyen eszközt, ami megmondja hogyan vezethetünk le újabb és újabb ***tényeket,*** azaz sorokat, a lekérdezés eredmény relációjába. Ha találunk a szabályban előforduló változóknak olyan értékeket, hogy minden $R_i(u_i)$ atom igaz (azaz a megfelelő sor az R_i relációban van), akkor a val relációba bevesszük az u sort. Mivel a fejben előforduló változók előfordulnak testbeli atomokban is, elérjük hogy sohasem kell ***végtelen*** értelmezési tartományokkal foglalkozzunk, hiszen a változók csak az éppen lekérdezett példányban előforduló konstans értékeket vehetik fel. Formálisan, legyen $\mathcal{I}$ az $\mathbf{R}$ relációs séma feletti példány, q pedig (27.3)-mal adott lekérdezés. Jelölje $var(q)$ a q-ban előforduló változók halmazát, $dom(\mathcal{I})$ pedig az $\mathcal{I}$-beli konstansok halmazát. $\mathcal{I}$ q ***alatti képe***

$$q(\mathcal{I}) = \{\nu(u) | \nu\colon var(q) \to dom(\mathcal{I}) \text{ és } \nu(u_i) \in R_i \; i = 1, 2, \ldots, n\}. \tag{27.4}$$

$q(\mathcal{I})$ kiszámításának triviális módja, hogy valamely rendszer szerint végignézzük a lehetséges ν kiértékeléseket. Ennél hatékonyabb algoritmus is lehetséges, egyrészt a lekérdezés ekvivalens átalakításával, másrészt megfelelő indexek használatával.

Fontos különbség a fejben és a testben szereplő atomok között, hogy az $R_1, R_2, \ldots, R_n$ relációkat adottnak, (fizikailag) tároltnak tekintjük, míg a val relációt nem, azt úgy gondoljuk, hogy a szabály segítségével számoljuk ki. Ebből adódik az elnevezés: R_i-k ***extenzionális relációk,*** val ***intenzionális reláció.***

Az $\mathbf{R}$ séma feletti q lekérdezés ***monoton,*** ha az $\mathcal{I}$ és $\mathcal{J}$ $\mathbf{R}$ feletti példányokra $\mathcal{I} \subseteq \mathcal{J}$-ből $q(\mathcal{I}) \subseteq q(\mathcal{J})$ következik. q ***kielégíthető,*** ha létezik olyan $\mathcal{I}$ példány, amelyre $q(\mathcal{I}) \neq \emptyset$. A következő egyszerű észrevétel bizonyítását az Olvasóra bízzuk (27.1-1. gyakorlat).

27.4. állítás. *Szabály alapú lekérdezések monotonok és kielégíthetőek.*

A 27.4. állítás rámutat a szabály alapú lekérdezések korlátaira. Például a ***Mely mozikban játszanak csak Jancsó filmeket?*** lekérdezés nyilvánvalóan nem monoton, tehát nem is fejezhető ki (27.3) formájú szabállyal.

Táblázatos lekérdezések

Ha a változók és konstansok közti különbséget nem vesszük figyelembe, akkor egy szabály teste a séma feletti példánynak is tekinthető. Ez a nézőpont elvezet a konjunktív lekérdezések táblázatos formájához, ami leginkább hasonlatos a Microsoft Access adatbázis-kezelő rendszer (QBE: Query By Example) szemléletes lekérdezéseihez.

27.5. definíció. *Az* $\mathbf{R}$ *séma feletti* ***tábla*** *az* $\mathbf{R}$ *séma feletti példány általánosítása, a sorokban konstansok mellett változók is előfordulhatnak. A* $(\mathbf{T}, u)$ *pár* ***táblázatos lekérdezés,*** *ha* $\mathbf{T}$ *egy tábla és* u *egy szabad sor, úgy, hogy minden* u*-ban előforduló változó előfordul* $\mathbf{T}$*-ben is. Az* u *szabad sor a táblázatos lekérdezés* ***összegzése.***

A $(\mathbf{T}, u)$ táblázatos lekérdezés u összegzés sora mutatja meg, hogy mely sorok alkotják a lekérdezés eredményét. Az eljárás lényege, hogy megpróbáljuk a $\mathbf{T}$ táblával adott mintát az adatbázisban megtalálni, és ha sikerül, akkor az u-nak megfelelő sort bevesszük az eredmény relációba. Pontosabban, $\nu\colon var(\mathbf{T}) \to dom(\mathcal{I})$ a $(\mathbf{T}, u)$ tábla ***beágyazása*** az $\mathcal{I}$

példányba, ha $\nu(\mathbf{T}) \subseteq \mathcal{I}$. A $(\mathbf{T}, u)$ táblázatos lekérdezés eredmény relációja mindazon $\nu(u)$ sorokból áll, amelyekre ν a $(\mathbf{T}, u)$ tábla beágyazása az $\mathcal{I}$ példányba.

27.2. példa. *Táblázatos lekérdezés.* Legyen $\mathbf{T}$ a következő tábla.

Filmek	*Cím*	*Rendező*	*Színész*
	x_C	"Kuroszava Akira"	x_{Sz}

Mozik	*Mozi*	*Utca*	*Telefon*
	x_M	x_U	x_T

Műsor	*Mozi*	*Cím*	*Időpont*
	x_M	x_C	x_I

A $(\mathbf{T}, \langle \mathit{Mozi}: x_M, \mathit{Utca}: x_U \rangle)$ táblás lekérdezés a bevezetés 27.2. kérdését válaszolja meg.

A táblázatos lekérdezések szintaxisa nagyon hasonló a szabály alapú lekérdezésekéhez. A későbbiekben hasznos lesz számunkra, hogy a táblázatos lekérdezések nyelvén könnyen megfogalmazható lesz annak feltétele, hogy két lekérdezés közül az egyik mikor tartalmazza a másikat.

Relációs algebra

Az adatbázis séma relációkból áll, a relációk pedig sorok halmazai. A lekérdezések eredménye is adott attribútum halmazzal rendelkező reláció. Természetes gondolat, hogy a lekérdezés eredményét a bemeneti relációkból halmazalgebrai, illetve a relációkon értelmezett egyéb műveletekkel fejezzük ki. A ***relációs algebra*** a következő műveleteket tartalmazza.

Szelekció: Az operáció alakja $\sigma_{A=c}$ vagy $\sigma_{A=B}$, ahol A és B attribútumok, c pedig konstans. A művelet alkalmazható minden olyan R relációra, amelyiknek A (és B) attribútuma, eredménye pedig értelemszerűen a *val* reláció, amelynek ugyanazok az attribútumai, mint R-nek, és azon R-beli sorokat tartalmazza, amelyekre igaz a kiválasztási feltétel.

Vetítés: Az operáció alakja $\pi_{A_1,A_2,\ldots,A_n}$, $n \geq 0$, ahol A_i-k különböző attribútumok. A művelet alkalmazható minden olyan R relációra, amelyik attribútumai közt minden A_i előfordul, eredménye pedig a *val* reláció, amelyik attribútum halmaza $\{A_1, A_2, \ldots, A_n\}$,

$$val = \{t[A_1, A_2, \ldots, A_n] | t \in R\},$$

azaz az R-beli sorok $\{A_1, A_2, \ldots, A_n\}$ attribútum halmazra való megszorításaiból áll.

Természetes összekapcsolás: Ezt a műveletet már az első kötet 12. fejezetében definiáltuk. Jelölése $\bowtie$, bemenete kettő R_1, R_2 (vagy több) reláció, V_1, V_2 attribútum halmazokkal. Az eredmény reláció attribútum halmaza $V_1 \cup V_2$.

$$R_1 \bowtie R_2 = \{t \text{ sor } V_1 \cup V_2 \text{ felett } | \exists v \in R_1, \exists w \in R_2, t[V_1] = v \text{ és } t[V_2] = w\}.$$

Átnevezés: Attribútum átnevezés nem más, mint egy-egy értelmű leképezés a véges U attribútum halmazról az összes attribútumok halmazába. Az f attribútum átnevezés megadható az $(A, f(A))$ párok listájával, ahol $A \neq f(A)$, ezt általában $A_1 A_2 \ldots A_n \rightarrow$

$B_1 B_2 \dots B_n$ alakban írjuk, $f(A_i) = B_i$. Az ***átnevezés operátor*** δ_f, U feletti bemenetekről $f[U]$ feletti kimenetekre képez. Ha R az U feletti reláció, akkor

$$\delta_f(R) = \{v\ f[U] \text{ felett} | \exists u \in R, v(f(A)) = u(A),\ \forall A \in U\}.$$

Relációs algebrai lekérdezéseket a fenti műveletek ismételt alkalmazásával nyerünk a ***relációs algebrai alap lekérdezésekből,*** amelyek

Bemenet reláció: R.

Egyetlen konstans: $\{\langle A : a\rangle\}$, ahol a konstans, A attribútum név.

27.3. példa. *Relációs algebrai lekérdezés.* A bevezetés 27.2. kérdése relációs algebrai műveletekkel a következőképpen fejezhető ki

$$\pi_{Mozi,Utca}\left((\sigma_{Rendező=\text{“Kuroszava Akira”}}(Filmek) \bowtie Műsor) \bowtie Mozik\right).$$

A relációs algebrai lekérdezés által megvalósított leképezést a műveleti fa szerinti indukcióval definiálhatjuk értelemszerűen. Könnyen látható (27.1-2. gyakorlat), hogy relációs algebrával megadható olyan lekérdezés, ami nem elégíthető ki. Az ilyennel ekvivalens szabály alapú, illetve táblázatos lekérdezés nyilván nem létezik. Azonban igaz a következő.

27.6. tétel. *A szabály alapú lekérdezések, a táblázatos lekérdezések és a kielégíthető relációs algebrai lekérdezések nyelvei ekvivalensek.*

A tétel bizonyítása három fő lépésből áll:

1. Szabály alapú ≡ Táblázatos
2. Kielégíthető relációs algebrai ⊑ Szabály alapú
3. Szabály alapú ⊑ Kielégíthető relációs algebrai

Az első lépést az Olvasóra bízzuk (27.1-3. gyakorlat). A második lépéshez először be kell látni, hogy a szabály alapú lekérdezések nyelve zárt a lekérdezések egymásba ágyazására nézve. Pontosabban, legyen $\mathbf{R} = \{R_1, R_2, \dots, R_n\}$ adatbázis, q lekérdezés $\mathbf{R}$ felett. Ha q eredmény relációja S_1, akkor egy következő lekérdezésben már S_1 is használható ugyanúgy, mint $\mathbf{R}$ tetszőleges extenzionális relációja. Így definiálhatjuk az S_2, majd annak segítségével az S_3, és így tovább, relációkat. Az S_i relációk ***intenzionális*** relációk. A P ***konjunktív lekérdezés program*** szabályok sorozata

$$\begin{array}{rcl} S_1(u_1) & \leftarrow & test_1 \\ S_2(u_2) & \leftarrow & test_2 \\ & \vdots & \\ S_m(u_m) & \leftarrow & test_m, \end{array} \tag{27.5}$$

ahol S_i-k különbözőek, és nincsenek $\mathbf{R}$-ben. A $test_i$ szabály testben csak az $R_1, R_2, \dots R_n$ és $S_1, S_2, \dots, S_{i-1}$ relációk fordulhatnak elő. P eredmény relációjának S_m-t tekintjük, kiszámítása a szabályok sorrend szerinti kiértékelésével történik. Nem nehéz látni, hogy a változók megfelelő átnevezésével P egyetlen szabállyal helyettesíthető, ahogy azt a következő példa

mutatja.

27.4. példa. *Konjunktív lekérdezés program.* Legyen $\mathbf{R} = \{Q, R\}$, és tekintsük a következő konjunktív lekérdezés programot

$$\begin{array}{rcl} S_1(x,z) & \leftarrow & Q(x,y), R(y,z,w) \\ S_2(x,y,z) & \leftarrow & S_1(x,w), R(w,y,v), S_1(v,z) \\ S_3(x,z) & \leftarrow & S_2(x,u,v), Q(v,z). \end{array} \tag{27.6}$$

Az (27.6) első két szabálya alapján S_2 felírható csak Q és R használatával

$$S_2(x,y,z) \leftarrow Q(x,y_1), R(y_1,w,w_1), R(w,y,v), Q(v,y_2), R(y_2,z,w_2). \tag{27.7}$$

Látható, hogy bizonyos változókat át kellett nevezni, hogy elkerüljük a különböző szabály testek nem kívánt egymásra hatását. A (27.7) kifejezést (27.6) harmadik szabályába írva S_2 helyére, és megfelelően átnevezve a változókat kapjuk

$$S_3(x,z) \leftarrow Q(x,y_1), R(y_1,w,w_1), R(w,u,v_1), Q(v_1,y_2), R(y_2,v,w_2), Q(v,z). \tag{27.8}$$

Ezek után elegendő az egyes relációs algebrai műveleteket egyenként megvalósítani szabállyal.

$P \bowtie Q$: Jelölje $\overrightarrow{x}$ a P és Q közös attribútumainak megfelelő változók (konstansok) listáját, $\overrightarrow{y}$ a csak P-ben, míg $\overrightarrow{z}$ a csak Q-ban előforduló attribútumoknak megfelelő változókat (konstansokat). Ekkor a $val(\overrightarrow{x},\overrightarrow{y},\overrightarrow{z}) \leftarrow P(\overrightarrow{x},\overrightarrow{y}), Q(\overrightarrow{x},\overrightarrow{z})$ szabály a $P \bowtie Q$ relációt adja eredményül.

$\sigma_F(R)$: Tegyük fel, hogy $R = R(A_1, A_2, \ldots, A_n)$. F a szelekciós feltétel, $A_i = a$ vagy $A_i = A_j$ alakú, ahol A_i, A_j attribútumok, a konstans. Ekkor

$$val(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n) \leftarrow R(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n),$$

illetve

$$val(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_{j-1}, y, x_{j+1}, \ldots, x_n) \leftarrow R(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_{j-1}, y, x_{j+1}, \ldots, x_n)$$

megfelelő szabály.

$\pi_{A_{i_1},A_{i_2},\ldots,A_{i_m}}(R)$: Ha $R = R(A_1, A_2, \ldots, A_n)$, akkor

$$val(x_{i_1}, x_{i_2}, \ldots, x_{i_m}) \leftarrow R(x_1, x_2, \ldots, x_n)$$

jó lesz.

$A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_n$: Az átnevezést a megfelelő változók átnevezésével oldhatjuk meg, amint azt a 27.4. példában láttuk.

A harmadik lépés bizonyításához tekintsünk egy

$$val(\overrightarrow{x}) \leftarrow R_1(\overrightarrow{x_1}), R_2(\overrightarrow{x_2}), \ldots, R_n(\overrightarrow{x_n}) \tag{27.9}$$

szabályt. Az R_i relációk attribútumainak megfelelő átnevezésével elérhető, hogy csupa különböző attribútumnév szerepeljen. Ezek után képezhetjük az $R = R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$ relációt, ami ténylegesen az R_i relációk direkt szorzata lesz, az attribútum nevek különbözősége miatt. A (27.9) szabályban szereplő konstansokat és változó egyezéseket megfelelő szelekciós operátorok alkalmazásával szimulálhatjuk. Végül a $val(\overrightarrow{x})$ reláció változóinak megfelelő attribútum halmazra való vetítéssel kapjuk a végeredményt.

27.1.2. Kiterjesztések

A konjunktív lekérdezések a lekérdezési nyelvek jól kezelhető fajtája. Azonban viszonylag szűk az általuk kifejezhető kérdések köre. Tekintsük a következőket.

27.4 Adjuk meg az olyan párokat, amelyeknek az első tagja rendezte a második tagot valamilyen filmben, valamint fordítva, a második tag is rendezte az elsőt valamilyen filmben.

27.5 Melyik moziban megy a "Szegénylegények" vagy a "Vihar kapujában"?

27.6 Melyek azok a Hitchcock filmek, amelyekben nem szerepelt Hitchcock maga?

27.7 Listázzuk azokat a filmeket, amelyek összes színésze szerepelt Jancsó Miklós valamelyik filmjében.

27.8 Ismeretes a "Színészlánc" játék. Az első játékos mond egy színészt, a sorban következő pedig egy olyat, aki vele egy filmben játszott. Ez így folytatódik, mindig olyan színészt kell mondani, aki az előzővel játszott egy filmben, és még nem szerepelt a játék folyamán. (Az nyer, aki utolsóként tudja még folytatni a láncot.) Listázzuk azokat a színészeket, akikhez "Latinovits Zoltán"-tól el lehet jutni a játék folyamán.

Egyenlőség atomok

A **27.4** kérdésre könnyen megadhatjuk a választ, ha a szabály testében a relációs atomokon kívül egyenlőséget is megengedünk

$$val(y_1, y_2) \leftarrow Filmek(x_1, y_1, z_1), Filmek(x_2, y_2, z_2), y_1 = z_2, y_2 = z_1. \tag{27.10}$$

Az egyenlőség megengedése két problémát is felvet. Az első, hogy a lekérdezés eredménye végtelen is lehet. Például a

$$val(x, y) \leftarrow R(x), y = z \tag{27.11}$$

lekérdezés eredménye végtelen sok sor, hiszen az y és z változókat az R reláció nem korlátozza, így végtelen sok kiértékelés lehetséges, ami a szabály testet kielégíti. Ezért a q szabály alapú lekérdezés ***tartomány-korlátozott,*** ha minden változó, amelyik q testében előfordul, az előfordul valamely relációs atomban is.

A második probléma az, hogy egyenlőség atomok a szabály testben a testbeli feltétel kielégíthetetlenségét okozhatják, ellentétben a 27.4. állítással. Például a

$$val(x) \leftarrow R(x), x = a, x = b \tag{27.12}$$

szabály tartomány-korlátozott, viszont ha a és b különböző konstansok, akkor a válasz az üres reláció lesz. Könnyen eldönthető, hogy egy q egyenlőségeket is tartalmazó szabály alapú lekérdezés kielégíthető-e.

KIELÉGÍTHETŐ(q)

1 Számítsuk ki a q testében levő egyenlőségek tranzitív lezártját.
2 **if** Két különböző konstans kell egyenlő legyen a tranzitivitás miatt
3 **then return** "Nem elégíthető ki."
4 **else return** "Kielégíthető."

Igaz az is (27.1-4. gyakorlat), hogy ha a q egyenlőségeket is tartalmazó szabály alapú lekérdezés kielégíthető, akkor van vele ekvivalens q', amelyik egyenlőség nélküli.

Diszjunkció – egyesítés

A **27.5** kérdés nem fejezhető ki konjunktív lekérdezéssel, azonban ha az egyesítés műveletét hozzávesszük a relációs algebrához, akkor az így kiterjesztett algebrával már kifejezhető:

$$\pi_{Mozi}\left(\sigma_{Cím=\text{“Szegénylegények”}}(Műsor) \cup \sigma_{Cím=\text{“A vihar kapujában”}}(Műsor)\right). \tag{27.13}$$

Szabály alapú lekérdezések is képesek a **27.5** kérdés kifejezésére, ha megengedjük, hogy több különböző szabálynak is ugyanaz a reláció legyen a fejében:

$$\begin{aligned} val(x_M) &\leftarrow Műsor(x_M, \text{“Szegénylegények”}, x_V), \\ val(x_M) &\leftarrow Műsor(x_M, \text{“A vihar kapujában”}, x_V). \end{aligned} \tag{27.14}$$

Ennek általánosítása a ***nemrekurzív datalog program.***

27.7. definíció. *Az* **R** *séma feletti* ***nemrekurzív datalog program***

$$\begin{aligned} S_1(u_1) &\leftarrow test_1 \\ S_2(u_2) &\leftarrow test_2 \\ &\vdots \\ S_m(u_m) &\leftarrow test_m \end{aligned} \tag{27.15}$$

szabályok halmaza, ahol **R***-beli reláció nem szerepel szabály fejben, ugyanaz a reláció több szabály fejében is szerepelhet, valamint létezik a szabályok olyan $r_1, r_2, \ldots, r_m$ sorrendje, hogy az r_i szabály fejében levő reláció nem fordul elő semelyik r_j szabály testében sem, ha $j \leq i$.*

A (27.15) nemrekurzív datalog program eredményének kiszámítása a (27.5) konjunktív lekérdezés program kiszámításához hasonló. A szabályokat a 27.7. definícióban szereplő sorrendben számoljuk, ha több szabály fejében ugyanaz a reláció szerepel, akkor a szabályok által adott sorok halmazainak egyesítését vesszük.

A $(\mathbf{T}_i, u)$ $i = 1, 2, \ldots, n$ táblázatos lekérdezések egyesítését $(\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n\}, u)$ jelöli. Kiszámításához az egyes $(\mathbf{T}_i, u)$ lekérdezéseket külön-külön kiszámítjuk, majd az eredmények egyesítését vesszük. Igaz a következő tétel.

27.8. tétel. *Az egyetlen célrelációval rendelkező nemrekurzív datalog programok nyelve és a relációs algebra kiegészítve az unió műveletével ekvivalensek.*

A 27.8. tétel bizonyítása hasonló a 27.6. tétel bizonyításához, az Olvasóra bízzuk (27.1-5. gyakorlat). Megjegyezzük, hogy a táblázatos lekérdezések egyesítésének kifejező ereje gyengébb, aminek az az oka, hogy ugyanazt az összegző sort követeljük meg minden egyes táblázathoz. A

$$\begin{aligned} val(a) &\leftarrow \\ val(b) &\leftarrow \end{aligned} \tag{27.16}$$

nemrekurzív datalog program nem valósítható meg táblázatos lekérdezések egyesítéseként.

Tagadás

A **27.6**. kérdés nyilvánvalóan nem monoton. Tegyük fel, hogy a *Filmek* relációban léteznek sorok Hitchcock *Psycho* című filmjéről, például (“Psycho”,”Hitchcock”,”A. Perkins”), (“Psycho”,”Hitchcock”,”J. Leigh”), ... , azonban

nincs (“Psycho”,”Hitchcock”,”Hitchcock”) sor. Ekkor a **27.6**. lekérdezés eredményében a (“Psycho”) sor is szerepel. Azonban, kicsit alaposabb kutatással kideríthető, hogy Hitchcock szerepel a *Psycho* című filmben, mint “egy ember cowboy kalapban”. Ha ezek után hozzávesszük a *Filmek* relációhoz a (“Psycho”,”Hitchcock”,”Hitchcock”) sort, akkor a **PestiMűsor** séma aktuális példánya bővebb lesz, viszont a **27.6** lekérdezés eredménye szűkebb.

Könnyű látni, hogy az eddigi fejezetekben tárgyalt lekérdezési nyelvek által megvalósított lekérdezések monotonok, azaz a **27.6**. lekérdezés nem valósítható meg nemrekurzív datalog programmal, vagy vele ekvivalens nyelvvel. Azonban, ha a relációs algebrai műveletek közé felvesszük a kivonás ($-$) operátort is, akkor alkalmas a **27.6**. típusú lekérdezések megvalósítására. Például,

$$\pi_{Cím}\,(\sigma_{Rendező=\text{“Hitchcock”}}(Filmek)) - \pi_{Cím}\,(\sigma_{Színész=\text{“Hitchcock”}}(Filmek)) \tag{27.17}$$

pontosan a **27.6** lekérdezést valósítja meg. A (teljes) relációs algebrát tehát a $\{\sigma, \pi, \bowtie, \delta, \cup, -\}$ műveletek alkotják. A relációs algebra fontosságát az is mutatja, hogy Codd a Q lekérdezési nyelvet pontosan akkor nevezi ***relációsan teljesnek,*** ha minden q algebrai lekérdezéshez van $q' \in Q$, hogy $q \equiv q'$.

Ha megengedünk szabály testekben ***negatív literálokat,*** azaz $\neg R(u)$ alakú atomokat, akkor az így kapott ***nemrekurzív datalog tagadással,*** jelölésben ***nr-datalog***$^\neg$ már relációsan teljes lesz.

27.9. definíció. *Nemrekurzív datalog*$^\neg$ *(nr-datalog*$^\neg$*) szabály a*

$$q: \quad S(u) \leftarrow L_1, L_2, \ldots, L_n \tag{27.18}$$

*alakú szabály, ahol S egy reláció, u egy szabad sor, az L_i pedig **literál,** azaz $R(v)$ vagy $\neg R(v)$ alakú kifejezés, $i = 1, 2, \ldots, n$, v szabad sor. S nem fordul elő a szabály testében. A szabály **tartomány-korlátozott,** ha minden x változó, ami előfordul a szabályban, előfordul valamely **pozitív literálban** ($R(v)$ alakú kifejezésben) is a szabály testében. Ha másképp nem jelezzük, akkor minden nr-datalog*$^\neg$ *szabályt tartomány-korlátozottnak tekintünk.*

A (27.18) szabály jelentése a következő. Legyen $\mathbf{R}$ relációs séma, amelyik tartalmazza a q testében szereplő összes relációt, továbbá legyen $\mathcal{I}$ $\mathbf{R}$ feletti példány. $\mathcal{I}$ ***q-szerinti képe***

$$\begin{aligned} q(\mathcal{I}) = \{\nu(u) |\ & \nu \text{ a változók kiértékelése és } i = 1, 2, \ldots, n\text{-re} \\ & \nu(u_i) \in \mathcal{I}(R_i), \text{ ha } L_i = R_i(u_i) \text{ és} \\ & \nu(u_i) \notin \mathcal{I}(R_i), \text{ ha } L_i = \neg R_i(u_i)\}. \end{aligned} \tag{27.19}$$

Az $\mathbf{R}$ séma feletti ***nr-datalog***$^\neg$ ***program*** nr-datalog$^\neg$ szabályok

$$\begin{array}{rcl} S_1(u_1) & \leftarrow & test_1 \\ S_2(u_2) & \leftarrow & test_2 \\ & \vdots & \\ S_m(u_m) & \leftarrow & test_m \end{array} \tag{27.20}$$

halmaza, ahol $\mathbf{R}$-beli reláció nem szerepel szabály fejben, ugyanaz a reláció több szabály fejében is szerepelhet, valamint létezik a szabályok olyan $r_1, r_2, \ldots, r_m$ sorrendje, hogy az r_i szabály fejében levő reláció nem fordul elő semelyik r_j szabály testében sem, ha $j \leq i$.

A (27.20) nr-datalog¬ program eredményének kiszámítása az **R** séma $\mathcal{I}$ példányára alkalmazva megegyezik a (27.15) nemrekurzív datalog program esetén használt módszerrel, azzal a különbséggel, hogy az egyes nr-datalog¬ szabályokat (27.19) szerint értelmezzük.

27.5. példa. *Nr-datalog¬ program.* Tegyük fel, hogy minden filmnek, amelyik a *Filmekben* szerepel, egyetlen rendezője van. (Nem mindig teljesül a valóságban!) A **27.6** lekérdezést

$$val(x) \leftarrow Filmek(x, \text{“Hitchcock”}, z), \neg Filmek(x, \text{“Hitchcock”}, \text{“Hitchcock”}) \tag{27.21}$$

nr-datalog¬ szabály valósítja meg. A **27.7** lekérdezést pedig a

$$\begin{aligned} Jancsó\text{-}színész(z) &\leftarrow Filmek(x, \text{”Jancsó”}, z) \\ Nem\text{-}ez\text{-}a\text{-}válasz(x) &\leftarrow Filmek(x, y, z), \neg Jancsó\text{-}színész(z) \\ Válasz(x) &\leftarrow Filmek(x, y, z), \neg Nem\text{-}ez\text{-}a\text{-}válasz(x) \end{aligned} \tag{27.22}$$

nr-datalog¬ program válaszolja meg. Óvatosnak kell lennünk azonban nr-datalog¬ program írásakor. Ha a (27.22) program első két szabályát egybe vonnánk a 27.4. példához hasonlóan

$$\begin{aligned} Rossz\text{-}nem\text{-}v(x) &\leftarrow Filmek(x, y, z), \neg Filmek(x', \text{”Jancsó”}, z), Filmek(x', \text{”Jancsó”}, z') \\ Válasz(x) &\leftarrow Filmek(x, y, z), \neg Rossz\text{-}nem\text{-}v(x), \end{aligned} \tag{27.23}$$

akkor (27.23) nem a **27.7** lekérdezést adná, hanem (feltéve, hogy minden filmnek egy rendezője van) a következő lekérdezést.

27.9 Listázzuk azokat a filmeket, amelyek minden színésze az ***összes*** Jancsó filmben szerepelt.

Könnyen látható, hogy minden nr-datalog¬ program, amelyik tartalmaz egyenlőség atomokat is, helyettesíthető olyannal, amelyikben nem szerepelnek egyenlőség atomok. Továbbá igaz az alábbi állítás is.

27.10. állítás. *A (teljes) relációs algebra és az egyetlen cél relációval rendelkező nr-datalog¬ program ekvivalens lekérdezési nyelvek.*

Rekurzió

A **27.8**. kérdést az eddigi lekérdezési nyelvekkel nem tudjuk megfogalmazni. Szükségünk lenne valamilyen ***a priori*** információra arról, hogy legfeljebb milyen hosszú *színészlánc* képezhető a kiindulási színészből. Tegyük fel, hogy tudjuk, “Latinovits”-ból indulva legfeljebb 117 hosszú lánc képezhető. (Érdekes lenne tudni a tényleges értéket!) Ekkor a következő nemrekurzív datalog program megadja a választ.

$$\begin{aligned} Film\text{-}társ(z_1, z_2) &\leftarrow Filmek(x, y, z_1), Filmek(x, y, z_2), z_1 < z_2{}^{1} \\ Rész\text{-}válasz_1(z) &\leftarrow Film\text{-}társ(z, \text{“Latinovits”}) \\ Rész\text{-}válasz_1(z) &\leftarrow Film\text{-}társ(\text{“Latinovits”}, z) \\ Rész\text{-}válasz_2(z) &\leftarrow Film\text{-}társ(z, y), Rész\text{-}válasz_1(y) \\ Rész\text{-}válasz_2(z) &\leftarrow Film\text{-}társ(y, z), Rész\text{-}válasz_1(y) \\ &\vdots \quad\quad \vdots \\ Rész\text{-}válasz_{117}(z) &\leftarrow Film\text{-}társ(z, y), Rész\text{-}válasz_{116}(y) \\ Rész\text{-}válasz_{117}(z) &\leftarrow Film\text{-}társ(y, z), Rész\text{-}válasz_{116}(y) \\ Latinovits\text{-}lánc(z) &\leftarrow Rész\text{-}válasz_1(z) \\ Latinovits\text{-}lánc(z) &\leftarrow Rész\text{-}válasz_2(z) \\ &\vdots \quad\quad \vdots \\ Latinovits\text{-}lánc(z) &\leftarrow Rész\text{-}válasz_{117}(z) \end{aligned} \tag{27.24}$$

Sokkal egyszerűbben fejezhető ki a **27.8**. kérdés ***rekurzió*** segítségével. Ténylegesen egy gráf, a *Film-társ*, ***tranzitív lezártját*** akarjuk kiszámolni. Az egyszerűség kedvéért kicsit megváltoztatjuk a *Film-társ* definícióját, (körülbelül megkétszerezve a tárigényt).

$$\begin{array}{rcl} \textit{Film-társ}(z_1, z_2) & \leftarrow & \textit{Filmek}(x, y, z_1), \textit{Filmek}(x, y, z_2) \\ \textit{Színészlánc-társ}(x, y) & \leftarrow & \textit{Film-társ}(x, y) \\ \textit{Színészlánc-társ}(x, y) & \leftarrow & \textit{Film-társ}(x, z), \textit{Színészlánc-társ}(z, y)\ . \end{array} \tag{27.25}$$

A (27.25) datalog program rekurzív, mivel a *Színészlánc-társ* reláció definíciójában önmagát használjuk. Tegyük fel, hogy ez értelmezhető (lásd később), ekkor a **27.8** kérdésre a választ megadja a

$$\textit{Latinovits-lánc}(x) \leftarrow \textit{Színészlánc-társ}(x, \text{“Latinovits”}) \tag{27.26}$$

szabály.

27.11. definíció. *Az*

$$R_1(u_1) \leftarrow R_2(u_2), R_3(u_3), \ldots, R_n(u_n) \tag{27.27}$$

kifejezés ***datalog szabály,*** *ha $n \geq 1$, R_i-k reláció nevek, u_i-k megfelelő hosszúságú szabad sorok. Minden u_1-beli változó elő kell forduljon $u_2, \ldots u_n$ valamelyikében is. A szabály feje $R_1(u_1)$, teste $R_2(u_2), R_3(u_3), \ldots, R_n(u_n)$.* ***Datalog program*** *(27.27) alakú szabályok véges halmaza. Legyen P datalog program. A P-ben szereplő R reláció* ***extenzionális,*** *ha csak szabály testekben fordul elő.* ***Intenzionális*** *a reláció, ha valamelyik szabály fejében előfordul.*

Ha ν a (27.27) szabályban szereplő változók valamely kiértékelése, akkor $R_1(\nu(u_1)) \leftarrow R_2(\nu(u_2)), R_3(\nu(u_3)), \ldots, R_n(\nu(u_n))$ a (27.27) szabály ***megvalósítása.*** Az ***extenzionális (adatbázis) séma*** P extenzionális relációiból áll, jelölésben $edb(P)$. $idb(P)$, az ***intenzionális (adatbázis) séma*** hasonlóképpen P intenzionális relációit tartalmazza. Jelölje $sch(P) = edb(P) \cup idb(P)$. A P datalog program szemantikus jelentése egy leképezés az $edb(P)$ feletti példányok halmazáról az $idb(P)$ feletti példányok halmazába. Ezt a jelentést modellelméletileg, bizonyítás-elméletileg, illetve valamely leképezés fixpontjának segítségével lehet megadni. Mivel az első kettő lényegileg ekvivalens a harmadikkal, és tárgyalásuk túl messzire vezetne, ezért csak a fixpont alapú definícióval foglalkozunk.

A 27.11. definícióban nem használtunk negatív literálokat. Ennek fő oka, hogy általában rekurzió és tagadás együtt értelmetlen lehet. Azonban bizonyos esetekben szükségünk lehet negatív atomokra is, akkor majd speciálisan foglalkozunk a program értelmezésével.

Fixpont szemantika

Legyen P datalog program, $\mathcal{K}$ $sch(P)$ feletti példány. Az A ***tény,*** azaz konstansokból álló sor, $\mathcal{K}$ és P ***közvetlen következménye,*** ha vagy $A \in \mathcal{K}(R)$ valamely $R \in sch(P)$ relációra, vagy $A \leftarrow A_1, A_2, \ldots, A_n$ a P valamelyik szabályának megvalósítása és minden A_i $\mathcal{K}$-ban van. P ***közvetlen következmény operátora*** T_P az $sch(P)$ feletti példányok halmazából képez önmagára. $T_P(\mathcal{K})$ a $\mathcal{K}$ és P összes közvetlen következményéből áll.

27.12. állítás. *A T_P közvetlen következmény operátor monoton.*

[1] Az egyenlőség atomokhoz hasonlóan használhatunk más összehasonlítási atomokat is. Itt a $z_1 < z_2$ azt biztosítja, hogy minden pár csak egyszer szerepel a felsorolásban.

Bizonyítás. Tegyük fel, hogy $\mathcal{I}$ és $\mathcal{J}$ $sch(P)$ feletti példányok, valamint $\mathcal{I} \subseteq \mathcal{J}$. Legyen A $T_P(\mathcal{I})$-be tartozó tény. Ha $A \in \mathcal{I}(R)$ valamely $R \in sch(P)$ relációra, akkor $\mathcal{I} \subseteq \mathcal{J}$ alapján $A \in \mathcal{J}(R)$ is teljesül. Ha pedig $A \leftarrow A_1, A_2, \ldots, A_n$ a P valamelyik szabályának megvalósítása és minden A_i $\mathcal{I}$-ben van, akkor $A_i \in \mathcal{J}$ teljesül. ■

T_P definíciójából következik, hogy $\mathcal{K} \subseteq T_P(\mathcal{K})$. Innen, felhasználva a 27.12. állítást kapjuk, hogy

$$\mathcal{K} \subseteq T_P(\mathcal{K}) \subseteq T_P(T_P(\mathcal{K})) \subseteq \ldots \tag{27.28}$$

27.13. tétel. *Minden $sch(P)$ feletti $\mathcal{I}$ példányhoz létezik egy egyértelmű minimális $\mathcal{I} \subseteq \mathcal{K}$ példány, ami a T_P **fixpontja,** azaz $\mathcal{K} = T_P(\mathcal{K})$.*

Bizonyítás. Jelölje $T_P^i(\mathcal{I})$ a T_P operátor i-szeres egymás utáni alkalmazását, és legyen

$$\mathcal{K} = \bigcup_{i=0}^{\infty} T_P^i(\mathcal{I}). \tag{27.29}$$

(27.29) és T_P monotonitása alapján

$$T_P(\mathcal{K}) = \bigcup_{i=1}^{\infty} T_P^i(\mathcal{I}) \subseteq \bigcup_{i=0}^{\infty} T_P^i(\mathcal{I}) = \mathcal{K} \subseteq T_P(\mathcal{K}), \tag{27.30}$$

azaz $\mathcal{K}$ fixpont. Könnyen látható, hogy minden olyan fixpont, ami tartalmazza $\mathcal{I}$-t, tartalmazza $T_P^i(\mathcal{I})$-t is minden $i = 1, 2, \ldots$-re, azaz $\mathcal{K}$-t is. ■

27.14. definíció. *A P datalog program **eredménye** az $edb(P)$ feletti $\mathcal{I}$ példányon T_P $\mathcal{I}$-t tartalmazó minimális fixpontja, jelölésben $P(\mathcal{I})$.*

Belátható, lásd 27.1-6. gyakorlat, hogy a (27.28)-beli lánc véges, azaz létezik olyan n, hogy $T_P(T_P^n(\mathcal{I})) = T_P^n(\mathcal{I})$. Ez az alapja a datalog program eredménye naiv kiszámítási módjának.

NAIV-DATALOG(P,$\mathcal{I}$)

```
1  K ← I
2  while T_P(K) ≠ K
3        do K ← T_P(K)
4  return K
```

Természetesen a NAIV-DATALOG eljárás nem optimális, hiszen minden egyes tényt, ami $\mathcal{K}$-ba belekerül, a **while** ciklus minden további végrehajtásánál újra kiszámol.

A FÉLIG-NAIV-DATALOG eljárás elve, hogy amennyire csak lehet, az éppen kiszámított új tényeket használja csak a **while** ciklus során, elkerülve ezzel a már ismert tények újraszámolását. Tekintsük a P datalog programot, $edb(P) = \mathbf{R}$, $idb(P) = \mathbf{T}$. A P-beli

$$S(u) \leftarrow R_1(v_1), \ldots, R_n(v_n), T_1(w_1), \ldots, T_m(w_m) \tag{27.31}$$

szabályhoz, ahol $R_k \in \mathbf{R}$ és $T_j \in \mathbf{T}$, elkészítjük a következő szabályokat $j = 1, 2, \ldots, m$ és $i \geq 1$-re

$$temp_S^{i+1}(u) \leftarrow R_1(v_1), \ldots, R_n(v_n), T_1^i(w_1), \ldots, T_{j-1}^i(w_1), \Delta_{T_j}^i(w_j), T_{j+1}^{i-1}(w_1), \ldots, T_m^{i-1}(w_1). \tag{27.32}$$

A $\Delta_{T_j}^i$ reláció a T_j i-edik iterációban történő változását jelöli. Az i-edik szint S-re vonatkozó szabályainak együttesét P_S^i jelöli (azaz (27.32) szabályokat $temp_S^{i+1}$-re, $j = 1, 2, \ldots, m$ esetén). Tegyük fel, hogy $T_1, T_2, \ldots, T_\ell$ az S idb relációt meghatározó szabályok testében szereplő idb relációk listája. Jelölje

$$P_S^i(\mathcal{I}, T_1^{i-1}, \ldots, T_\ell^{i-1}, T_1^i, \ldots, T_\ell^i, \Delta_{T_1}^i, \ldots, \Delta_{T_\ell}^i) \tag{27.33}$$

a (27.32) szabályok az $\mathcal{I}$ bemeneti példányra és a $T_j^{i-1}, T_j^i, \Delta_{T_j}^i$ idb relációkra való alkalmazásával kapott tényeket (sorokat). Az $\mathcal{I}$ bemeneti példány az $edb(P)$ relációinak aktuális értéke.

FÉLIG-NAIV-DATALOG(P,$\mathcal{I}$)

```
1  P' ← azon P-beli szabályok, amelyek testében nincs idb reláció
2  for S ∈ idb(P)
3      do S^0 ← ∅
4         Δ_S^1 ← P'(I)(S)
5  i ← 1
6  repeat
7         for S ∈ idb(P)
8                 ▷ T_1, ..., T_ℓ az S-t meghatározó szabályokban előforduló idb relációk.
9             do S^i ← S^{i-1} ∪ Δ_S^i
10               Δ_S^{i+1} ← P_S^i(I, T_1^{i-1}, ..., T_ℓ^{i-1}, T_1^i, ..., T_ℓ^i, Δ_{T_1}^i, ..., Δ_{T_ℓ}^i) − S^i
11        i ← i + 1
12 until Δ_S^i = ∅ minden S ∈ idb(P)-re
13 for S ∈ idb(P)
14     do S ← S^i
```

27.15. tétel. *A* FÉLIG-NAIV-DATALOG *helyesen számítja ki a P datalog program eredményét.*

Bizonyítás. i-szerinti teljes indukcióval belátjuk, hogy tetszőleges $S \in idb(P)$-re a 6–12. sorok ciklusának i-edik végrehajtása után az S^i értéke $T_P^i(\mathcal{I})(S)$, Δ_S^{i+1} pedig $T_P^{i+1}(\mathcal{I})(S) - T_P^i(\mathcal{I})(S)$-sel egyenlő. $T_P^i(\mathcal{I})(S)$ értelemszerűen a T_P közvetlen következmény operátor i-szeres alkalmazásával az $\mathcal{I}$ példányból kiindulva az S relációra kapott érték.

$i = 0$-ra a 4. sor pontosan $T_P(\mathcal{I})(S)$-t számítja ki minden $S \in idb(P)$-re. Az indukciós lépéshez mindössze azt kell látnunk, hogy $P_S^i(\mathcal{I}, T_1^{i-1}, \ldots, T_\ell^{i-1}, T_1^i, \ldots, T_\ell^i, \Delta_{T_1}^i, \ldots, \Delta_{T_\ell}^i) \cup S^i$ pontosan $T_P^{i+1}(\mathcal{I})(S)$-sel egyenlő, hiszen a 9–10. sorokban a FÉLIG-NAIV-DATALOG eljárás ennek használatával állítja elő S^i-t és Δ_S^{i+1}-t. Az indukciós feltétel szerint S^i értéke $T_P^i(\mathcal{I})(S)$, ehhez képest új sorokat csak úgy kaphatunk, ha valamely S-t meghatározó idb relációnak olyan sorait vesszük figyelembe, amelyek T_P legutolsó alkalmazásakor keletkeztek, ezek pedig szintén az indukciós feltétel miatt a $\Delta_{T_1}^i, \ldots, \Delta_{T_\ell}^i$ relációkban vannak.

A 12. sor feltétele pont azt jelenti, hogy minden $S \in idb(P)$ reláció változatlan marad a T_P közvetlen következmény operátor alkalmazása során, tehát az algoritmus megtalálta annak minimális fixpontját. Az pedig a 27.14. definíció szerint pontosan a P datalog program eredménye az $\mathcal{I}$ bemeneti példányra. ■

A FÉLIG-NAIV-DATALOG eljárás ugyan sok felesleges számítást kiküszöböl, azonban bizonyos datalog programokon ez sem optimális (27.1-7. gyakorlat). Azonban a datalog program elemzésével és a számítás azon alapuló módosításával a legtöbb felesleges lépést meg lehet takarítani.

27.16. definíció. *Legyen P datalog program. P **előzmény gráfja** G_P a következő irányított gráf. Csúcshalmaza $idb(P)$ relációi, $R, R' \in idb(P)$ esetén (R, R') irányított él, ha létezik olyan szabály P-ben, amelynek feje R' és R a testében van. P **rekurzív,** ha G_P-ben van irányított kör. Az R és R' relációk **kölcsönösen rekurzívak,** ha G_P ugyanazon erősen összefüggő komponensébe esnek.*

A kölcsönös rekurzivitás ekvivalencia reláció $idb(P)$-be tartozó relációk halmazán. A JAVÍTOTT-FÉLIG-NAIV-DATALOG eljárás alapgondolata, hogy az $R \in idb(P)$ relációval "egyszerre" csak a vele kölcsönösen rekurzív relációkat kell számolni, minden más, R-t definiáló szabályban előforduló relációt már "előre" kiszámíthatunk, és *edb* relációnak tekinthetünk.

JAVÍTOTT-FÉLIG-NAIV-DATALOG(P,$\mathcal{I}$)

1 Határozzuk meg $idb(P)$ kölcsönös rekurzivitás szerinti ekvivalencia osztályait.
2 Készítsük el az $[R_1], [R_2], \ldots, [R_n]$ ekvivalencia osztályok listáját G_P topologikus rendezése szerint.
3 ▷ Minden $i < j$-re teljesül, hogy G_P-ben nincs irányított út R_j-ből R_i-be.
4 **for** $i \leftarrow 1$ **to** n
5 **do** Használjuk a FÉLIG-NAIV-DATALOG eljárást az $[R_i]$-beli relációk kiszámítására, az $[R_j]$-beli relációkat *edb* relációkként kezelve $j < i$-re.

Mélységi keresés alkalmazásával az 1–2. sorok $O(v_{G_P}+e_{G_P})$ időben végrehajthatók, ahol v_{G_P} és e_{G_P} a G_P gráf csúcs-, illetve élszámát jelölik. Az eljárás helyességének bizonyítását az Olvasóra bízzuk (27.1-8. gyakorlat).

27.1.3. Bonyolultsági kérdések lekérdezések közti tartalmazásról

A jelen részben visszatérünk a konjunktív lekérdezésekhez. Lekérdezések eredményének számításakor a legköltségesebb feladat relációk természetes összekapcsolásának elvégzése. Különösen igaz ez, ha a közös attribútumokhoz nincs index megadva, és így csak TELJES-SORONKÉNTI-ÖSSZEKAPCSOLÁS eljárás alkalmazható.

```
TELJES-SORONKÉNTI-ÖSSZEKAPCSOLÁS(R1, R2)
1  S ← ∅
2  for minden u ∈ R1
3      do for minden v ∈ R2
4             do if u és v összekapcsolható
5                then S ← S ∪ {u ⋈ v}
6  return S
```

Világos, hogy TELJES-SORONKÉNTI-ÖSSZEKAPCSOLÁS futási ideje $O(|R_1| \times |R_2|)$. Nem mindegy tehát, hogy egy lekérdezést milyen sorrendben számítunk ki, hiszen az eljárás folyamán különböző méretű relációk természetes összekapcsoltjait kell képezni. Táblázatos lekérdezések esetén a ***Homomorfizmus tétel*** lehetőséget ad a lekérdezés olyan átírására, amelyik kevesebb összekapcsolást használ, mint az eredeti.

Az $\mathbf{R}$ séma feletti q_1, q_2 lekérdezésekre q_2 ***tartalmazza*** q_1-t, jelben $q_1 \sqsubseteq q_2$, ha minden $\mathbf{R}$ feletti $\mathcal{I}$ példányra $q_1(\mathcal{I}) \subseteq q_2(\mathcal{I})$ teljesül. $q_1 \equiv q_2$ a 27.1. definíció értelmében pontosan akkor, ha $q_1 \sqsubseteq q_2$ és $q_1 \sqsupseteq q_2$. Szükségünk lesz a kiértékelések általánosítására. ***Helyettesítésen*** olyan leképezést értünk, amelyik a változók halmazából képez a változók és a konstansok halmazának egyesítésébe, és amelyiket konstansokra identitásként terjesztünk ki. Természetesen értelmezhető a helyettesítés kiterjesztése szabad sorokra, illetve táblázatokra.

27.17. definíció. *Legyen $q = (\mathbf{T}, u)$ és $q' = (\mathbf{T}', u')$ két táblázatos lekérdezés az $\mathbf{R}$ séma felett. A θ helyettesítés* ***homomorfizmus*** *q'-ről q-ra, ha $\theta(\mathbf{T}') = \mathbf{T}$ és $\theta(u') = u$.*

27.18. tétel (homomorfizmus tétel). *Legyen $q = (\mathbf{T}, u)$ és $q' = (\mathbf{T}', u')$ két táblázatos lekérdezés az $\mathbf{R}$ séma felett. $q \subseteq q'$ akkor és csak akkor, ha létezik homomorfizmus q'-ről q-ra.*

Bizonyítás. Tegyük fel először, hogy θ homomorfizmus q'-ről q-ra, és legyen $\mathcal{I}$ az $\mathbf{R}$ séma feletti példány. Legyen $w \in q(\mathcal{I})$. Ez pontosan akkor teljesül, ha létezik egy ν kiértékelés, amelyik a $\mathbf{T}$ táblát $\mathcal{I}$-be képezi és $\nu(u) = w$. Könnyen látható, hogy $\theta \circ \nu$ a $\mathbf{T}'$ táblát képezi $\mathcal{I}$-be és $\theta \circ \nu(u') = w$, azaz $w \in q'(\mathcal{I})$. Tehát $w \in q(\mathcal{I}) \Longrightarrow w \in q'(\mathcal{I})$, ami pontosan $q \subseteq q'$-vel egyenértékű.

Másik oldalról, tegyük fel, hogy $q \subseteq q'$. A bizonyítás gondolata, hogy q-t és q'-t is alkalmazzuk a $\mathbf{T}$ "példányra". q eredménye az u szabad sor, tehát q' eredménye szintén tartalmazza az u sort, vagyis létezik $\mathbf{T}'$ egy θ beágyazása $\mathbf{T}$-be, amelyik u'-t u-ra képezi. A gondolatmenet szabatossá tételéhez elkészítjük a $\mathbf{T}$-vel izomorf $\mathcal{I}_{\mathbf{T}}$ példányt.

Legyen V a $\mathbf{T}$-ben előforduló változók halmaza. Minden $x \in V$-hez rendeljük az a_x konstanst, ami különbözik a $\mathbf{T}$-ben illetve $\mathbf{T}'$-ben előforduló konstansoktól, valamint $x \neq x' \Longrightarrow a_x \neq a_{x'}$. Legyen μ az a kiértékelés, amelyik $x \in V$-hez a_x-t rendeli, valamint legyen $\mathcal{I}_{\mathbf{T}} = \mu(\mathbf{T})$. μ bijekció V-ről $\mu(V)$-re, és $\mu(V)$-ben nem fordulnak elő $\mathbf{T}$-beli konstansok, ezért μ^{-1} jól definiált az $\mathcal{I}_{\mathbf{T}}$-ben előforduló konstansokon.

Világos, hogy $\mu(u) \in q(\mathcal{I}_{\mathbf{T}})$, tehát $q \subseteq q'$ alapján $\mu(u) \in q'(\mathcal{I}_{\mathbf{T}})$ is teljesül. Azaz, létezik egy ν kiértékelés, ami a $\mathbf{T}'$ táblát beágyazza $\mathcal{I}_{\mathbf{T}}$-be, úgy hogy $\nu(u') = \mu(u)$. Könnyen látható, hogy $\nu \circ \mu^{-1}$ homomorfizmus q'-ről q-ra. ■

Lekérdezés optimalizálás tábla minimalizálással

A 27.6. tétel alapján táblázatos lekérdezések és a kielégíthető relációs algebrai (kivonás nélkül) ekvivalensek. A bizonyítás során kiderül, hogy a táblázatos lekérdezéssel ekvivalens relációs algebra kifejezés $\pi_{\vec{j}}(\sigma_F(R_1 \bowtie R_2 \bowtie \cdots \bowtie R_k))$ alakú, ahol k a tábla sorainak száma. Ebből következik, hogy ha a természetes összekapcsolások számát akarjuk minimalizálni, akkor az ekvivalens tábla sorainak számát kell lehető legkisebbre csökkenteni.

A $(\mathbf{T}, u)$ táblázatos lekérdezés ***minimális,*** ha nincs olyan $(\mathbf{S}, v)$ lekérdezés, amelyik ekvivalens $(\mathbf{T}, u)$-val és $|\mathbf{S}| < |\mathbf{T}|$, azaz $\mathbf{S}$-nek kevesebb sora van. Meglepő, de igaz, hogy a $(\mathbf{T}, u)$-val ekvivalens minimális lekérdezés megkapható egyszerűen néhány sor elhagyásával $\mathbf{T}$-ből.

27.19. tétel. *Legyen $q = (\mathbf{T}, u)$ táblázatos lekérdezés. Van $\mathbf{T}'$ részhalmaza $\mathbf{T}$-nek, hogy $q' = (\mathbf{T}', u)$ minimális és ekvivalens $q = (\mathbf{T}, u)$-val.*

Bizonyítás. Legyen $(\mathbf{S}, v)$ minimális, q-val ekvivalens lekérdezés. A Homomorfizmus tétel szerint létezik θ homomorfizmus q-ról $(\mathbf{S}, v)$-re, valamint λ $(\mathbf{S}, v)$-ről q-ra. Legyen $\mathbf{T}' = \theta \circ \lambda(\mathbf{S})$. Könnyen ellenőrizhető, hogy $(\mathbf{T}', u) \equiv q$ és $|\mathbf{T}'| \leq |\mathbf{S}|$. Azonban $(\mathbf{S}, v)$ minimális, így $(\mathbf{T}', u)$ is az. ■

27.6. példa. *Tábla minimalizálás alkalmazása.* Tekintsük az $\{A, B, C\}$ attribútum halmazú $\mathbf{R}$ séma feletti

$$q = \pi_{AB}(\sigma_{B=5}(R)) \bowtie \pi_{BC}(\pi_{AB}(R) \bowtie \pi_{AC}(\sigma_{B=5}(R))) \tag{27.34}$$

relációs algebrai lekérdezést. A q-nak megfelelő táblás lekérdezés a következő $\mathbf{T}$ tábla:

R	A	B	C
	x	5	z_1
	x_1	5	z_2
	x_1	5	z
u	x	5	z

(27.35)

Olyan homomorfizmust keresünk, ami a $\mathbf{T}$ tábla néhány sorát $\mathbf{T}$ más soraira képezi, ezáltal mintegy "összehajtogatja" a táblát. Az első sor nem hagyható el, mert a homomorfizmus az u szabad soron azonosság, tehát x-t önmagának kell megfeleltesse. Hasonló a helyzet a harmadik sorral, mert z-nek is saját maga a képe minden homomorfizmusnál. Azonban a második sort ki lehet küszöbölni, x_1-t x-re, z_2-t z-re képezve. Tehát a $\mathbf{T}$-vel ekvivalens minimális tábla $\mathbf{T}$ első és harmadik sorát tartalmazza. Visszaírva algebrai lekérdezésre,

$$\pi_{AB}(\sigma_{B=5}(R)) \bowtie \pi_{BC}(\sigma_{B=5}(R)) \tag{27.36}$$

az eredmény. A (27.36) lekérdezés a (27.34) lekérdezéshez képest eggyel kevesebb összekapcsolás műveletet tartalmaz.

A következő tétel azt mondja ki, hogy táblák közti tartalmazás és ekvivalencia eldöntésének kérdése NP-teljes, következésképpen a tábla minimalizálás NP-nehéz feladat.

27.20. tétel. *Adott q és q' táblázatos lekérdezések esetén az alábbi döntési feladatok NP-teljesek:*

27.10. $q \subseteq q'$?

27.11. $q \equiv q'$*?*

27.12. Tegyük fel, hogy q'*-ből néhány szabad sor elhagyásával kaptuk* q'*-t. Igaz-e ekkor, hogy* $q \equiv q'$*?*

Bizonyítás. A Pontos Fedés feladatot vezetjük vissza a különböző tábla feladatokra. A Pontos Fedés feladat bemenete egy $X = \{x_1, x_2, \ldots, x_n\}$ halmaz valamint részhalmazainak $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ rendszere. Eldöntendő, hogy létezik-e olyan $\mathcal{S}' \subseteq \mathcal{S}$, hogy az $\mathcal{S}'$-beli részhalmazok pontosan lefedik X-t (azaz, minden $x \in X$-hez pontosan egy $S \in \mathcal{S}'$ létezik, amelyre $x \in S$). A Pontos Fedés ismert NP-teljes feladat.

Legyen $\mathcal{E} = (X, \mathcal{S})$ a Pontos Fedés bemenete. Vázolunk egy konstrukciót, ami $\mathcal{E}$-hez táblázatos $q_\mathcal{E}, q'_\mathcal{E}$ lekérdezés párt készít polinomiális időben. Ezt a konstrukciót lehet aztán a különböző NP-teljességi állítások bizonyítására használni.

Az $\mathbf{R}$ séma attribútumai legyenek a páronként különböző $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$ attribútumok. $q_\mathcal{E} = (\mathbf{T}_\mathcal{E}, t)$ és $q'_\mathcal{E} = (\mathbf{T}'_\mathcal{E}, t)$ az $\mathbf{R}$ séma feletti táblázatos lekérdezések, mindkettő összegzése a $t = \langle A_1 : a_1, A_2 : a_2, \ldots, A_n : a_n \rangle$ szabad sor, ahol $a_1, a_2, \ldots, a_n$ páronként különböző változók.

Legyenek $b_1, b_2, \ldots, b_m, c_1, c_2, \ldots c_m$ további páronként különböző változók. $\mathbf{T}_\mathcal{E}$ n sorból áll, X minden elemének megfelel egy. Az x_i elem sorában a_i áll az A_i attribútum oszlopában, b_j a B_j attribútum oszlopában minden olyan j-re, amelyre $x_i \in S_j$ teljesül. A $\mathbf{T}_\mathcal{E}$ n tábla többi helyén csupa különböző új változó áll.

Hasonlóan, $\mathbf{T}'_\mathcal{E}$ m sorból áll, $\mathcal{S}$ minden elemének megfelel egy. Az S_j részhalmaz sorában a_i áll az A_i attribútum oszlopában, minden olyan i-re, amelyre $x_i \in S_j$, valamint $c_{j'}$ a $B_{j'}$ attribútum oszlopában, minden $j' \neq j$-re. A $\mathbf{T}'_\mathcal{E}$ n tábla többi helyén csupa különböző új változó áll.

A 27.10. kérdés NP-teljessége következik abból, hogy X-nek akkor és csak akkor létezik pontos fedése $\mathcal{S}$-beli halmazokkal, ha $q'_\mathcal{E} \subseteq q_\mathcal{E}$ teljesül. A bizonyítást, valamint a 27.11. és 27.12. kérdések NP-teljességének bizonyítását az Olvasóra bízzuk (27.1-9. gyakorlat). ■

Gyakorlatok

27.1-1. Bizonyítsuk be a 27.4. állítást, azaz hogy minden szabály alapú q lekérdezés monoton és kielégíthető. *Útmutatás.* A kielégíthetőség bizonyításához legyen a q lekérdezésben szereplő összes konstans halmaza K, $a \notin K$ pedig egy további konstans. Minden (27.3)-beli R_i reláció sémához készítsük el az összes olyan $(a_1, a_2, \ldots, a_r)$ sort, ahol $a_i \in K \cup \{a\}$, és r az R_i attribútumainak száma. Legyen $\mathcal{I}$ az így kapott példány. Lássuk be, hogy $q(\mathcal{I})$ nem üres.

27.1-2. Adjunk meg egy $\mathbf{R}$ relációs sémát és q relációs algebrai lekérdezést $\mathbf{R}$ felett, amelynek eredménye üres halmaz tetszőleges $\mathbf{R}$ feletti példányra.

27.1-3. Bizonyítsuk be, hogy a szabály alapú lekérdezések nyelve és a táblázatos lekérdezések nyelve ekvivalens.

27.1-4. Bizonyítsuk be, hogy minden szabály alapú q lekérdezés, amelyik egyenlőség atomokat is tartalmazhat, vagy ekvivalens a $q^\emptyset$ üres lekérdezéssel, vagy létezik egy q' szabály alapú lekérdezés, amely nem tartalmaz egyenlőség atomokat úgy, hogy $q \equiv q'$. Adjunk polinomiális algoritmust, ami egy adott egyenlőségeket is tartalmazó szabály alapú q lekérdezésről eldönti, hogy $q \equiv q^\emptyset$ teljesül-e, és ha nem, akkor elkészít egy q' szabály alapú lekérdezést, amely nem tartalmaz egyenlőség atomokat úgy, hogy $q \equiv q'$.

27.1-5. A 27.6. tétel bizonyításának gondolatát általánosítva bizonyítsuk be a 27.8. tételt.
27.1-6. Legyen P datalog program, $\mathcal{I}$ példány $edb(P)$ felett, $C(P,\mathcal{I})$ az $\mathcal{I}$-ben és P-ben szereplő konstansok (véges) halmaza. Legyen $\mathbf{B}(P,\mathcal{I})$ az alábbi $sch(P)$ feletti példány:

1. Minden $edb(P)$-beli R relációra az $R(u)$ tény $\mathbf{B}(P,\mathcal{I})$-ben van pontosan akkor, ha $\mathcal{I}$-ben van, valamint
2. minden $idb(P)$-beli R relációra minden $C(P,\mathcal{I})$-beli konstansokból képzett $R(u)$ tény $\mathbf{B}(P,\mathcal{I})$-ben van.

Bizonyítsuk be, hogy

$$\mathcal{I} \subseteq T_P(\mathcal{I}) \subseteq T_P^2(\mathcal{K}) \subseteq T_P^3(\mathcal{K}) \subseteq \cdots \subseteq \mathbf{B}(P,\mathcal{I}). \tag{27.37}$$

27.1-7. Adjunk példát olyan bemenetre, P datalog programra és $\mathcal{I}$ edb példányra, amelyre ugyanaz a sort a FÉLIG-NAIV-DATALOG ciklusának különböző végrehajtásai is előállítják.
27.1-8. Bizonyítsuk be, hogy a JAVÍTOTT-FÉLIG-NAIV-DATALOG eljárás minden bemenetre véges időben megáll, és helyes eredményt ad. Mutassunk példát olyan bemenetre, amelyiken a JAVÍTOTT-FÉLIG-NAIV-DATALOG kevesebb sort számít ki többször, mint a FÉLIG-NAIV-DATALOG eljárás.
27.1-9.

1. Bizonyítsuk be, hogy a 27.20. tétel bizonyításában szereplő $q_{\mathcal{E}} = (\mathbf{T}_{\mathcal{E}}, t)$ és $q'_{\mathcal{E}} = (\mathbf{T}'_{\mathcal{E}}, t)$ táblázatos lekérdezésekre pontosan akkor létezik homomorfizmus $(\mathbf{T}_{\mathcal{E}}, t)$-ről $(\mathbf{T}'_{\mathcal{E}}, t)$-re, ha az $\mathcal{E} = (X, \mathcal{S})$ PONTOS FEDÉS feladatnak van megoldása.
2. Bizonyítsuk be, hogy a 27.11. és 27.12. kérdések eldöntése NP-teljes feladat.

27.2. Nézetek

Egy adatbázis rendszer felépítésének három fő szintje van:

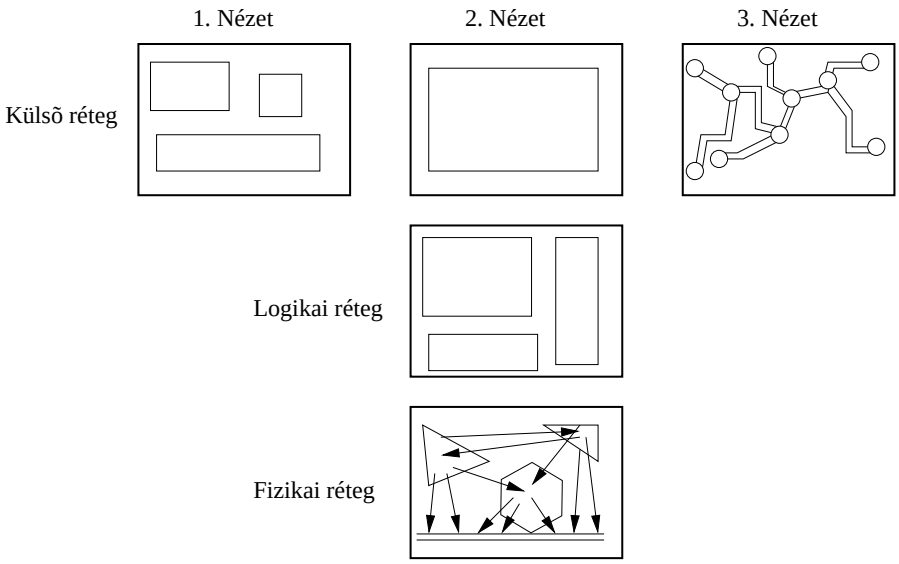
- fizikai réteg;
- logikai réteg;
- külső (felhasználói) réteg.

A szintek elválasztásának célja a fizikai adatfüggetlenség elérése, valamint a felhasználói kényelem. A 27.2. ábrán a három nézet lehetséges felhasználói felületeket mutat: multirelációs, univerzális relációs, illetve grafikus felület.

A ***fizikai réteg*** a ténylegesen tárolt adatállományokat, a rájuk épített sűrű és ritka indexeket jelenti.

A ***logikai réteg*** elválasztása a fizikai rétegtől lehetővé teszi, hogy a felhasználó az adatok logikai összefüggéseire koncentráljon, ami sokkal jobban közelíti a modellezni kívánt valóságról alkotott képét. A logikai réteghez tartozik az adatbázis séma leírása a különböző integritási feltételekkel, függőségekkel. Ez az a szint, ahol az adatbázis adminisztrátorok kezelik a rendszert. A logikai és a fizikai réteg közti kapcsolatot az adatbáziskezelő program tartja fent.

A ***külső réteg*** és a logikai réteg elválasztásának célja, hogy a végfelhasználók az adatbázist a saját (szűk) igényeik és szempontjaik szerint lássák. Például egy banki adatbázis



27.2. ábra. Az adatbázis rendszerek felépítésének három szintje.

esetén a külső réteg egyik nagyon egyszerű nézete lehet a pénzkiadó automata, vagy egy jóval bonyolultabb nézete lehet kölcsön igénylés elbírálásához az ügyfél teljes banki kredit története.

27.2.1. Nézet, mint lekérdezés eredménye

Kérdés az, hogy a külső réteghez tartozó nézeteket hogyan adhatjuk meg. Ha egy relációs algebrai kifejezéssel adott lekérdezést úgy tekintünk, mint valamely formulát, amit majd reláció példányokra alkalmazunk, akkor kapjuk a ***nézetet***. Datalog szabályok jól szemléltetik a lekérdezések és nézetek közti különbséget. A szabályok által meghatározott relációkat ***intenzionálisnak*** neveztük, mivel ezek azok a relációk, amelyeknek nem kell külső tárolókon, extenzionálisan létezniük, szemben az ***extenzionális*** relációkkal.

27.21. definíció. *Az* $\mathbf{R}$ *séma feletti valamely* Q *lekérdezési nyelven megadott* $\mathcal{V}$ *kifejezést* $\mathbf{R}$ *séma feletti* ***nézetnek*** *nevezünk.*

Természetesen datalog intenzionális relációkhoz hasonlóan nézeteket is használhatunk lekérdezések megadásakor, illetve újabb nézetek meghatározásakor.

27.7. példa. *SQL nézet.* Az SQL adatbázis-kezelő nyelvben az alábbi módon lehet nézetet megadni. Tegyük fel, hogy a **PestiMűsor** sémából számunkra érdekes adat csak annyi, hogy mikor és hol játszanak Kuroszava filmeket. A **KuroszavaIdőpontok** nézetet

KuroszavaIdőpontok

```
1 create view KuroszavaIdőpontok as
2        select Mozi, Időpont
3        from Filmek, Műsor
4        where Filmek.Cím=Műsor.Cím and Filmek.Rendező="Kuroszava Akira"
```

SQL program definiálja. Relációs algebrai alakban a következő.

$$KuroszavaIdőpontok(Mozi,\ Vetítés) = \pi_{Mozi,\ Vetítés}(Mozi \bowtie \sigma_{Rendező="\text{Kuroszava Akira}"}(Filmek)) \quad (27.38)$$

Végül datalog szabállyal ugyanez

$$KuroszavaIdőpontok(x_M, x_V) \leftarrow Mozik(x_M, x_C, x_V), Filmek(x_C, \text{"Kuroszava Akira"}, x_{Sz}). \quad (27.39)$$

A KuroszavaIdőpontok eljárás 2. sora jelöli ki a használt szelekciós operátort, a 3. sor, hogy melyik két relációt kell összekapcsolni, végül a 4. sor feltétele mutatja meg, hogy természetes összekapcsolásról van szó, nem pedig direkt szorzatról.

Amennyiben a $\mathcal{V}$ nézetet már definiáltuk, akkor a további lekérdezésekben, illetve nézet definíciókban ugyanúgy alkalmazható, mint akármilyen másik (extenzionális) reláció.

Nézetek használatának előnyei

- Adatok automatikus elrejtése: Olyan adatok, amelyek nem részei a használt nézetnek nyilván nem is jelennek meg a felhasználó előtt, így azokat nem is olvashatja illetéktelenül, illetve nem is módosíthatja. Tehát azzal, hogy az adatbázis hozzáférést nézeteken keresztül engedjük meg, egyszerű, de hatékony biztonsági mechanizmust üzemeltetünk.
- Nézetek egyszerű "makró képességet" szolgáltatnak. A 27.7. példában definiált KuroszavaIdőpontok nézet használatával könnyedén meg tudjuk keresni melyik moziban adnak délelőtt Kuroszava filmet:

  $$KuroszavaDélelőtt(Mozi) \leftarrow KuroszavaIdőpontok(Mozi, x_V), x_V < 12. \quad (27.40)$$

  Természetesen a felhasználó beírhatná a kódba a *KuroszavaIdőpontok* definícióját közvetlenül, de a makró utasításokkal szoros hasonlatosságban, itt is a kényelmi szempontok az elsők.
- A nézetek lehetővé teszik, hogy ugyanazt az adatot különböző felhasználók különböző módon lássák ugyanabban az időben.
- Nézetek biztosítják a ***logikai adatfüggetlenséget.*** A logikai adatfüggetlenség lényege, hogy a felhasználók és programjaik védettek legyenek az adatbázis séma szerkezeti változtatásaitól. Ezt úgy lehet elérni, hogy a szerkezet–változtatás előtti relációkat mint nézetek definiáljuk a szerkezet változtatás utáni új sémában.

Materializált nézet

Előfordulhat, hogy valamely nézetet több különböző lekérdezésben is használunk. Ilyen esetekben hasznos lehet ha nézet által meghatározott reláció(k) sorait nem kell minden esetben újra kiszámolnunk, hanem a nézet definíciójában szereplő lekérdezés eredményét tároljuk, és a további feldolgozásokkor csak beolvassuk. Az ilyen tárolt eredményt nevezzük

materializált nézetnek.

Gyakorlatok

27.2-1. Tekintsük az alábbi sémát:

Filmsztár(*Név*,*Cím*,*Nem*,*SzülDátum*)
FilmMogul(*Név*,*Cím*,*Igazolvány#*,*Vagyon*)
Stúdió(*Név*,*Cím*,*ElnökIg#*)

A *FilmMogul* reláció a filmszakma nagymenőinek (stúdió elnökök, producerek, stb) adatait tartalmazza. Az egyes attribútumok nevei értelemszerűen megadják jelentésüket, illetve az *Igazolvány#* a mogul működési engedélyének száma, az *ElnökIg#* a stúdió elnöke működési engedélyének száma. Adjuk meg az alábbi nézeteket datalog szabállyal, relációs algebrai kifejezéssel, illetve SQL nyelven:

1. *GazdagMogul*: a legalább 100,000,000 forint vagyonú filmmogulok nevét, címét igazolvány számát és vagyonát listázza.
2. *StúdióElnök*: az olyan filmmogulok nevét, címét igazolvány számát listázza, akik stúdió elnökök is egyben.
3. *MogulSztár*: az összes olyan személy nevét, címét igazolvány számát és vagyonát listázza, aki egyszerre filmsztár és filmmogul is.

27.2-2. A **PestiMűsor** séma felett adjuk meg az alábbi nézeteket datalog szabállyal, relációs algebrai kifejezéssel, illetve SQL nyelven:

1. *Marilyn*(*Cím*): Marilyn Monroe szereplésével készült filmek címét listázza.
2. *CorvinInfo*(*Cím*,*Időpont*,*Telefon*): a Corvin moziban játszott filmek címét, vetítési időpontjait, valamint a mozi telefonszámát listázza.

27.3. Lekérdezés átírás

Lekérdezések megválaszolása nézetek felhasználásával, más néven lekérdezések átírása nézetek felhasználásával a közelmúltban nagyon sok figyelmet és érdeklődést kiváltó feladattá vált. Ennek oka a feladat széles körű alkalmazhatósága a legkülönfélébb adatkezelési feladatokban: lekérdezés optimalizálásban, fizikai adatfüggetlenség megvalósításában, adat– , illetve információ–egyesítésnél, valamint adattárházak tervezésénél.

A feladat lényege a következő. Tegyük fel. hogy az $\mathbf{R}$ séma felett adott a Q lekérdezés, valamint $V_1, V_2, \ldots, V_n$ nézetek. Meg lehet-e válaszolni a Q lekérdezést pusztán a $V_1, V_2, \ldots, V_n$ nézetek eredményeinek ismeretében? Avagy, mi azon sorok legbővebb halmaza, amit a nézetek ismeretében meg tudunk határozni? Ha elérhetjük a nézeteket és az alapséma relációit is, melyik a legolcsóbb kiszámítási mód Q megválaszolására?

27.3.1. Motiváció

Mielőtt a lekérdezés átírás algoritmusait részletesen tárgyalnánk néhány alkalmazás bemutatásával indokoljuk, hogy miért érdemes a kérdéssel foglalkozni. A példákhoz az alábbi

Egyetem adatbázist használjuk:

$$\textbf{Egyetem} = \{Tanár, Kurzus, Tanít, Felvett, Szakirány, Dolgozik, Témavezető\} \tag{27.41}$$

Ahol az egyes relációk sémái a következőek:

$$\begin{aligned} Tanár &= \{TNév, Szakterület\} \\ Kurzus &= \{K\text{-}szám, Cím\} \\ Tanít &= \{TNév, K\text{-}szám, Félév, Véleményezés\} \\ Felvett &= \{Diák, K\text{-}szám, Félév\} \\ Szakirány &= \{Diák, Tanszék\} \\ Dolgozik &= \{TNév, Tanszék\} \\ Témavezető &= \{TNév, Diák\} \end{aligned} \tag{27.42}$$

Feltesszük, hogy a tanárokat, diákokat és tanszékeket a nevük egyértelműen meghatározza, valamint a kurzusokat a számuk. A *Felvett* reláció sorai azt mutatják, hogy melyik diák melyik tárgyat melyik félévben vette fel, a *Szakirány* pedig azt, hogy melyik tanszéket választotta szakosodáskor (feltesszük az egyszerűség kedvéért, hogy egy tanszéken csak egy szakirány van hirdetve).

Lekérdezés optimalizálás

Ha egy lekérdezés megválaszolásához szükséges számítások egy részét már előzőleg elvégeztük és tároltuk valamely materializált nézetben, akkor használhatjuk a nézetet a lekérdezés megválaszolásának meggyorsítására.

Tekintsük azt a lekérdezést, amelyik azokat a (*Diák*, *Cím*) párokat keresi, ahol a diák az adott doktorandusz tárgyat felvette, a tárgyat *adatbázis* szakterületű tanár tanítja (választható tárgyak K-száma legalább 400, ebből a doktorandusz tárgyak azok, amelyek K-száma ≥ 500).

$$\begin{aligned} val(x_D, x_C) \quad \leftarrow \quad & Tanít(x_T, x_K, x_F, y_1), Tanár(x_T, \text{“adatbázis”}), \\ & Felvett(x_D, x_K, x_F), Kurzus(x_k, x_C), x_K \geq 500 \end{aligned} \tag{27.43}$$

Tegyük fel, hogy rendelkezésre áll az alábbi materializált nézet, amelyik választható tárgyak regisztrációs adatait tartalmazza

$$Választható(x_D, x_C, x_K, x_F) \leftarrow Felvett(x_D, x_K, x_F), Kurzus(x_K, x_C), x_K \geq 400. \tag{27.44}$$

A *Választható* nézet felhasználható (27.43) megválaszolására

$$\begin{aligned} val(x_D, x_C) \quad \leftarrow \quad & Tanít(x_T, x_K, x_F, y_1), Tanár(x_T, \text{“adatbázis”}), \\ & Választható(x_D, x_C, x_K, x_F), x_K \geq 500 \end{aligned} \tag{27.45}$$

(27.45) kiszámítása gyorsabb lesz, mint (27.43)-é, mivel a *Felvett* és a *Kurzus* természetes összekapcsolását már a *Választható* nézet elvégezte, valamint leválasztotta a kötelező tárgyakat (amelyek a regisztráció legnagyobb részét teszik ki a legtöbb egyetemen). Érdemes megjegyezni, hogy a *Választható* nézet annak ellenére használható, hogy ***szintaktikusan*** a (27.43) lekérdezés egyetlen részével sem egyezik meg.

Másfelől azonban előfordulhat, hogy az eredeti lekérdezést gyorsabban tudjuk megválaszolni. Ha a *Felvett* és a *Kurzus* relációknak a *K-szám* attribútumon létezik indexük,

viszont a *Választható*-hoz semmilyen index sincs felépítve, akkor gyorsabb lehet a (27.43) lekérdezést közvetlenül az adatbázis relációkból számítani. Az igazi kihívás tehát nem csak az, hogy eldöntsük egy materializált nézetről, hogy logikailag használható-e valamely lekérdezés megválaszolására, hanem alapos költség elemzést kell végeznünk, hogy mikor érdemes használni a létező nézeteket.

Fizikai adatfüggetlenség
A mai modern adatbázis rendszerek egyik alapelve a az adatok logikai szerkezetének és fizikai tárolási módjának szétválasztása. A relációs adatbázis rendszerek esetében, eltekintve a relációk vízszintes vagy függőleges szétvágásától, még mindig lényegileg egy-az-egyhez megfeleltetés van a séma relációi és az őket tartalmazó fizikai állományok között. Objektum orientált rendszerek esetében a fizikai–logikai elválasztás elengedhetetlen, mivel a logikai séma jelentős redundanciát tartalmaz, ezért nem felel meg jó fizikai elhelyezésnek. Másik példa a fizikai adatfüggetlenség fontosságára az, amikor a logikai modellt mint közbenső réteget határozzuk meg azután, hogy a fizikai megjelentetést már eldöntöttük. Ez általános amikor XML adatokat tárolunk relációs adatbázisokban, illetve adat egyesítésnél. A STORED rendszer például XML adatokat tárol relációs adatbázisban úgy, hogy nézetek használ az XML→relációk leképezés leírására.

A fizikai adatfüggetlenség fenntartására az egyik elterjedt módszer, hogy az adat tényleges fizikai tárolását a logikai séma feletti nézetek segítségével írjuk le. Például Tsatalos, Solomon és Ioannidis ÁTEU-kat (Általánosított Többszintű Elérési Utakat) használnak az adattárolás leírására.

Az ÁTEU leírja a tárolási szerkezet fizikai szervezését és indexeit. Az első kulcsszó (**as**) leírja a használt adatszerkezetet, amelyben a sorokat tárolják (B^+-fa, hasítási index, stb). A leírás többi része a tartalmat adja meg, nézetek megadásához nagyon hasonlóan. A **given** és a **select** kulcsszavak leírják a rendelkezésre álló attribútumokat, ahol a **given** adja meg, hogy melyik attribútumok alapján indexeljük a struktúrát. A **where** kulcsszóval adott nézet definícióban infix jelölést használunk.

A 27.3. ábrán látható példában az A1 ÁTEU olyan (*Diák,Tanszék*) párokat tartalmaz, ahol a *Diák* a *Tanszéket* választotta szakosodáskor. Ezek a párok egy B^+ fával vannak indexelve a *Diák.név* attribútumon. Az A2 ÁTEU egy indexet tárol a diákok nevéből azon kurzusok *K-számaiba,* amelyeket felvettek. Az A3 ÁTEU a kurzusok *K-számaiból* azon *Tanszékekbe* mutató indexet tartalmaz, amely *Tanszékeknél* szakosodott diákok felvették az adott tárgyat.

Mivel az adatokat az ÁTEU-kban leírt adatszerkezetekben tároljuk, felmerül a kérdés, hogyan lehet ezeket az adatszerkezeteket felhasználni lekérdezések megválaszolására. Az ÁTEU-k logikai tartalmát nézetek írják le, ezért a lekérdezés megválaszolása pontosan az a feladat, hogy találjunk a lekérdezésnek egy olyan átírását, ami az adott nézeteket használja. Ha több átírás is lehetséges, akkor a legolcsóbb átírást keressük. Jegyezzük meg, hogy a lekérdezés optimalizálással szemben itt ***szükségszerűen*** használnunk kell a nézeteket, mivel az adatokat ÁTEU-kban tároljuk.

Tekintsük a következő lekérdezést, amelyik azon diákok nevét és a szakosodásnál választott tanszékét kérdezi, akik doktorandusz kurzust vettek fel.

$$val(Diák,Tanszék) \leftarrow Felvett(Diák,K\text{-}szám,y), Szakirány(Diák,Tanszék), K\text{-}szám \geq 500. \tag{27.46}$$

def.áteu A1 **as** b^+-fa **by**
given *Diák.név*
select *Tanszék*
where *Diák* szakosodik *Tanszék*

def.áteu A2 **as** b^+-fa **by**
given *Diák.név*
select *Kurzus.K-szám*
where *Diák* felvett *Kurzus*

def.áteu A3 **as** b^+-fa **by**
given *Kurzus.K-szám*
select *Tanszék*
where *Diák* felvett *Kurzus* **and** *Diák* szakosodik *Tanszék*

27.3. ábra. Az **Egyetem** tartomány ÁTEU-i.

A lekérdezést két módon is megválaszolhatjuk. Először, mivel a *Diák.név* egyértelműen meghatározza a diákot, vehetjük A1 és A2 természetes összekapcsolását, majd alkalmazhatunk egy szelekciós operátort amivel kiválasztjuk azokat a sorokat, amelyekre $K\text{-}szám \geq 500$, végül egy vetítéssel kiküszöbölhetjük a szükségtelen attribútumokat. Másik végrehajtási terv lehet, hogy A3-t és A2-t kapcsoljuk össze, majd elvégezzük a $K\text{-}szám \geq 500$ szelekciót. Ez utóbbi megoldás hatékonyabb lehet, mert A3 tartalmaz indexet a *K-szám* attribútumon, így a közbenső összekapcsolások sokkal kisebbek lehetnek.

Adategyesítés

Egy ***adategyesítési rendszer*** (más néven ***adatközvetítő rendszer***) célja, hogy ***egységes*** lekérdezési felületet biztosítson nagyszámú és eltérő szerkezetű adatforráshoz. Legfontosabb példák a vállalati integráció, több különböző webes forrás egyidejű lekérdezése, valamint elosztott tudományos kísérletek eredményének egyesítése.

Az egységes lekérdezési felület elérése érdekében az adategyesítési rendszer a felhasználó felé egy ***közvetített sémát*** mutat. A közvetített séma ***virtuális*** relációkból áll, abban az értelemben, hogy fizikai valóságukban ezeket a relációkat sehol nem tárolják ebben a formában. A közvetített sémát az adategyesítési alkalmazás szempontjából kell egyedileg megtervezni. Ahhoz, hogy a lekérdezéseket meg tudja válaszolni, a rendszernek tartalmaznia kell ***forrás leírásokat.*** Egy adatforrás leírása meghatározza a forrás tartalmát, a benne megtalálható attribútumokat, valamint a forrás tartalmára vonatkozó integritási feltételeket. Az adatforrás leírás egyik elterjedt megközelítési módja, hogy a forrás tartalmát a közvetített séma feletti ***nézetként*** adjuk meg. Ez lehetővé teszi új adatforrások beillesztését, illetve az integritási feltételek megadását.

A lekérdezés megválaszolásához az adategyesítési rendszernek a közvetített sémában megfogalmazott lekérdezést le kell fordítania olyanra, amelyik közvetlenül az adatforrásokra hivatkozik. Mivel a források nézetként adottak, a fordítás azzal egyenértékű, hogy a lekérdezést nézetek segítségével válaszoljuk meg.

Példaként tekintsük az **Egyetem** sémát, mint a felhasználó felé közvetített sémát, azzal

a különbséggel, hogy a *Tanít* és a *Kurzus* relációknak eggyel több attribútumuk van, nevezetesen az *Egyetem* attribútum, amelyik megmondja, hogy az adott tárgyat melyik egyetemen tanítják.

$$\begin{aligned} Kurzus &= \{K\text{-}szám, Cím, Egyetem\} \\ Tanít &= \{TNév, K\text{-}szám, Félév, Véleményezés, Egyetem\} \end{aligned} \tag{27.47}$$

Tegyük fel, hogy az alábbi két adatforrás áll rendelkezésre. Az első az összes "Adatbázisok" című tárgyakat, és azok előadóját listázza az összes egyetemről. A következő nézet meghatározással írható le:

$$\begin{aligned} DBkurz(Cím, Tnév, K\text{-}szám, Egyetem) \leftarrow\ & Kurzus(K\text{-}szám, Cím, Egyetem), \\ & Tanít(TNév, K\text{-}szám, Félév, Véleményezés, Egyetem), \\ & Cím = \text{“Adatbázisok”} \end{aligned} \tag{27.48}$$

A második adatforrás a Budapesti Műszaki és Gazdaságtudományi Egyetem doktorandusz kurzusait adja meg, az alábbi módon.

$$\begin{aligned} BMEPhD(Cím, Tnév, K\text{-}szám, Egyetem) \leftarrow\ & Kurzus(K\text{-}szám, Cím, Egyetem), \\ & Tanít(TNév, K\text{-}szám, Félév, Véleményezés, Egyetem), \\ & Egyetem = \text{“BME”}, K\text{-}szám \geq 500. \end{aligned} \tag{27.49}$$

Ha az adategyesítési rendszertől azt kérdezzük, hogy ki tanít *Adatbázisokat* a Műegyetemen, akkor az a lekérdezést egyszerűen megválaszolhatja a *DBkurz* relációra alkalmazott szelekciós operátorral:

$$Val(Tnév) \leftarrow DBkurz(Cím, Tnév, K\text{-}szám, Egyetem), Egyetem = \text{"BME"}. \tag{27.50}$$

Azonban, ha azt szeretnénk tudni, hogy milyen választható (nem csak adatbázis) tárgyak léteznek a Műegyetemen, akkor az adategyesítési rendszer nem tudja a lekérdezés eredményéhez tartozó összes sort megtalálni, mivel csak a (27.48) és a (27.49) források állnak a rendelkezésére. Ehelyett, a források alapján meghatározható legbővebb sorhalmazt keresheti meg. Azaz, meghatározhatja az összes műegyetemi választható *adatbázis* kurzust a *DBkurz* forrásból, valamint a doktorandusz tárgyakat a *BMEPhD* forrásból. Tehát a következő nemrekurzív datalog program megadja a legbővebb választ:

$$\begin{aligned} val(Cím, K\text{-}szám) \leftarrow\ & DBkurz(Cím, Tnév, K\text{-}szám, Egyetem), \\ & Egyetem = \text{"BME"}, K\text{-}szám \geq 400 \\ val(Cím, K\text{-}szám) \leftarrow\ & BMEPhD(Cím, Tnév, K\text{-}szám, Egyetem). \end{aligned} \tag{27.51}$$

Vegyük észre, hogy azok a választható tárgyak, amelyek nem doktorandusz tárgyak, vagy nem adatbázis témájúak, nem szerepelnek az eredményben. A lekérdezés optimalizálás és fizikai adatfüggetlenség esetében ***ekvivalens*** lekérdezés átírást kellett találni, itt olyan lekérdezés kifejezést keresünk, amelyik a nézetekből kapható ***legbővebb*** eredményhalmazt találja meg.

Szemantikus gyorstárolás

Kliens-szerver felépítésű adatbázis elérés esetén a kliens által már letöltött adatok szemantikusan modellezhetők, mint bizonyos nézetek eredményei. Tehát a kliens gépen eltárolt adatokat nem mint fizikai adategységeket, lapokat, sorokat, hanem mint nézeteket tekintjük. Ekkor annak eldöntésére, hogy a kliens újabb lekérdezésének megválaszolásához mely

további adatok letöltése szükséges, a szervernek azt a feladatot kell megoldania, hogy a kliens oldalon létező nézetek segítségével a lekérdezés mely részei válaszolhatók meg.

27.3.2. Átírás bonyolultsági kérdései

Ebben az alfejezetben a lekérdezés átírás ***elméleti*** bonyolultságát tárgyaljuk. Elsősorban konjunktív lekérdezésekkel foglalkozunk. Megkülönböztetjük a ***minimális*** és a ***teljes*** átírásokat. Belátjuk, hogy ha a lekérdezés konjunktív és a nézetek is konjunktív lekérdezések materializált eredményeként adottak, akkor az átírási probléma ***NP-teljes,*** feltéve, hogy sem a lekérdezés, sem a nézetek nem tartalmaznak összehasonlítási atomokat. A konjunktív lekérdezéseket szabály alakban tekintjük, ez a legkényelmesebb számunkra.

Tegyük fel, hogy Q lekérdezés adott a $\mathbf{R}$ séma felett.

27.22. definíció. *A Q' konjunktív lekérdezés a Q lekérdezés $\mathcal{V} = V_1, V_2, \ldots, V_m$ nézeteket használó* ***átírása,*** *ha*

- *Q és Q' ekvivalensek, és*
- *Q' egy, vagy több $\mathcal{V}$-beli literált tartalmaz.*

Azt mondjuk, hogy Q' ***lokálisan minimális,*** *ha Q'-ből nem hagyható el literál anélkül, hogy az ekvivalencia megsérülne. Az átírás* ***globálisan minimális,*** *ha nem létezik kevesebb literált használó átírás. (A literálok számába az összehasonlítási atomok $=, \neq, \leq, <$ nem számítanak bele!)*

27.8. példa. *Lekérdezés átírás.* Tekintsük a következő Q lekérdezést és V nézet.

$$\begin{array}{rcl} Q\colon q(X,U) & \leftarrow & p(X,Y), p_0(Y,Z), p_1(X,W), p_2(W,U) \\ V\colon v(A,B) & \leftarrow & p(A,C), p_0(C,B), p_1(A,D) \end{array} \tag{27.52}$$

Q átírható V használatával:

$$Q'\colon q(X,U) \leftarrow v(X,Z), p_1(X,W), p_2(W,U). \tag{27.53}$$

A V nézet a Q lekérdezés első két literálját helyettesíti. Vegyük észre, hogy a lekérdezés harmadik literálját is biztosan kielégíti a nézet, azonban nem hagyható el az átírásból, mert a D változó már nem szerepel V fejében, ezért ha a p_1 literált is elhagynánk, akkor a p_1 és p_2 közti természetes összekapcsolást már nem kényszerítené ki semmi.

Mivel az alkalmazások egy részében az adatbázis relációkat nem érjük el, csak a nézeteket, például az adategyesítés és adattárházak esetében, ezért szükségünk van a ***teljes átírás*** fogalmára.

27.23. definíció. *A Q lekérdezés $\mathcal{V} = V_1, V_2, \ldots, V_m$ nézeteket használó Q' átírása* ***teljes átírás,*** *ha Q' csak $\mathcal{V}$-beli literálokat és összehasonlítási atomokat tartalmaz.*

27.9. példa. *Teljes átírás.* Tegyük fel, hogy a 27.8. példában szereplő V nézet mellett még a

$$V_2\colon v_2(A,B) \leftarrow p_1(A,C), p_2(C,B), p_0(D,E) \tag{27.54}$$

nézet is adott. A Q lekérdezés át teljesen átírható:

$$Q''\colon q(X,U) \leftarrow v(X,Z), v_2(X,U). \tag{27.55}$$

Fontos látnunk, hogy ezt az átírást nem kaphatjuk meg ***lépésenként,*** először csak V-t használva, majd megpróbálni V_2-t beépíteni (vagy éppen a másik sorrendben), hiszen p_0 reláció amelyik V_2-ben szerepel, nem szerepel Q'-ben. Tehát a teljes átírás megtalálásához a két nézet használatát egyszerre, ***párhuzamosan*** kell tekinteni.

A lekérdezés átírás megtalálása szoros kapcsolatban áll a lekérdezések közti tartalmazás eldöntésének feladatával. Ez utóbbit táblázatos lekérdezésekre már a 27.1.3. pontban tárgyaltuk. Táblázatos lekérdezések közti homomorfizmus értelmezhető szabály alapú konjunktív lekérdezésekre is. Az egyetlen különbség, hogy nem követeljük meg ebben a fejezetben, hogy az szabály fejét a homomorfizmus a másik szabály fejére képezze. (A táblázatos lekérdezés összegző sorának a szabály feje felel meg.) A 27.20. tétel szerint annak eldöntése, hogy a Q_1 konjunktív lekérdezés tartalmazza-e Q_2 konjunktív lekérdezést NP-teljes. Ez igaz marad akkor is, ha Q_2 összehasonlítási atomokat is tartalmaz. Azonban, ha Q_1 is tartalmaz összehasonlítási atomokat, akkor homomorfizmus létezése Q_1-ről Q_2-re csak elégséges feltételt ad a lekérdezések tartalmazására, amelyik ebben az esetben Π_2^p-teljes feladat. Ez utóbbi feladatosztály tárgyalása túlmutat a fejezet keretein, ezért nem részletezzük. A következő állítás szükséges és elégséges feltételt ad arra, hogy létezik-e a Q lekérdezésnek a V nézetet használó átírása.

27.24. állítás. *Tegyük fel, hogy Q és V konjunktív lekérdezések, amelyek tartalmazhatnak összehasonlítási atomokat is. Akkor és csak akkor létezik Q-nak V-t használó átírása, ha $\pi_\emptyset(Q) \subseteq \pi_\emptyset(V)$, azaz V vetítése az üres attribútum halmazra tartalmazza Q vetítését.*

Bizonyítás. Vegyük észre, hogy $\pi_\emptyset(Q) \subseteq \pi_\emptyset(V)$ ekvivalens az alábbi állítással: Ha V eredménye üres halmaz valamely adatbázis példányon, akkor Q eredménye is üres.

Tegyük fel először, hogy létezik átírás, azaz olyan Q-val ekvivalens szabály, amelynek testében V szerepel. Ha r olyan adatbázis példány, amelyiken V eredménye üres halmaz, akkor minden olyan szabály eredménye is üres, amelyiknek testében V szerepel.

Tegyük fel fordítva, hogy ha V eredménye üres halmaz valamely adatbázis példányon, akkor Q eredménye is üres. Legyen

$$\begin{array}{lcl} Q\colon q(\tilde{x}) & \leftarrow & q_1(\tilde{x}), q_2(\tilde{x}), \ldots, q_m(\tilde{x}) \\ V\colon v(\tilde{a}) & \leftarrow & v_1(\tilde{a}), v_2(\tilde{a}), \ldots, v_n(\tilde{a}) \end{array} \tag{27.56}$$

Legyen $\tilde{y}$ az $\tilde{x}$-beli változóktól diszjunkt változók listája. Ekkor a

$$Q'\colon q'(\tilde{x}) \leftarrow q_1(\tilde{x}), q_2(\tilde{x}), \ldots, q_m(\tilde{x}), v_1(\tilde{y}), v_2(\tilde{y}), \ldots, v_n(\tilde{y}) \tag{27.57}$$

lekérdezésre teljesül, hogy $Q \equiv Q'$. Világos, hogy $Q' \subseteq Q$. Másfelől, ha valamely r adatbázis példányra létezik az $\tilde{y}$ változóinak olyan kiértékelése, amelyik kielégíti V testét, akkor ezt rögzítve, az $\tilde{x}$-beli változók tetszőleges kiértékelésére pontosan akkor kapunk egy eredménysort Q-ban, amikor a rögzített $\tilde{y}$ kiértékeléssel együtt Q'-ben. ■

A 27.24. állítás és a 27.20. tétel következménye az alábbi tétel.

27.25. tétel. *Legyen Q konjunktív lekérdezés, amelyik tartalmazhat összehasonlítási atomokat, $\mathcal{V}$ nézetek halmaza. Ha a $\mathcal{V}$-beli nézetek összehasonlítási atomokat nem tartalmazó konjunktív lekérdezéssel adottak, akkor NP teljes annak eldöntése, hogy létezik-e Q-nak $\mathcal{V}$-t használó átírása.*

A 27.25. tétel bizonyítását az Olvasóra bízzuk (27.3-1. gyakorlat).

A 27.24. állítás bizonyításában új változókat vezettünk be. Azonban, a következő lemma szerint erre nincs szükség. Másik fontos észrevétel, hogy elegendő az eredeti lekérdezésben szereplő adatbázis relációk egy részhalmazát tekinteni, amikor lokálisan minimális átírást keresünk, új adatbázis relációkat nem kell felhasználni.

27.26. lemma. *Legyen Q konjunktív lekérdezés, amelyben nem szerepelnek összehasonlítási atomok*

$$Q\colon q(\tilde{X}) \leftarrow p_1(\tilde{U}_1), p_2(\tilde{U}_2), \ldots, p_n(\tilde{U}_n), \tag{27.58}$$

valamint legyen $\mathcal{V}$ nézetek halmaza, szintén összehasonlítási atomok nélkül.

1. *Ha Q $\mathcal{V}$-t használó lokálisan minimális átírása Q', akkor a Q'-ben szereplő adatbázis literálok halmaza izomorf a Q-beli adatbázis literálok egy részhalmazával.*
2. *Ha*

$$q(\tilde{X}) \leftarrow p_1(\tilde{U}_1), p_2(\tilde{U}_2), \ldots, p_n(\tilde{U}_n), v_1(\tilde{Y}_1), v_2(\tilde{Y}_2), \ldots v_k(\tilde{Y}_k) \tag{27.59}$$

a Q egy a nézeteket használó átírása, akkor létezik egy

$$q(\tilde{X}) \leftarrow p_1(\tilde{U}_1), p_2(\tilde{U}_2), \ldots, p_n(\tilde{U}_n), v_1(\tilde{Y'}_1), v_2(\tilde{Y'}_2), \ldots v_k(\tilde{Y'}_k) \tag{27.60}$$

átírás is, amelyre teljesül, hogy $\{\tilde{Y'}_1 \cup \cdots \cup \tilde{Y'}_k\} \subseteq \{\tilde{U}_1 \cup \cdots \cup \tilde{U}_n\}$, azaz az átírás nem vezet be új változókat.

A 27.26. lemma bizonyításának részleteit az Olvasóra hagyjuk (27.3-2. gyakorlat). A következő lemma alapvető fontosságú: A Q $\mathcal{V}$-t használó minimális átírása nem ***növelheti*** a literálok számát.

27.27. lemma. *Legyen Q konjunktív lekérdezés, $\mathcal{V}$ konjunktív lekérdezésekkel megadott nézetek halmaza, mindkettő összehasonlítási atomok nélkül. Ha Q teste p literált tartalmaz, és Q' a Q $\mathcal{V}$-t használó lokálisan minimális teljes átírása, akkor Q' legfeljebb p literált tartalmaz.*

Bizonyítás. A Q'-beli nézet literálok helyére írjuk be a definíciójukat, így kapjuk a Q'' lekérdezést. Legyen φ homomorfizmus Q testéből Q''-be. φ létezik a Homomorfizmus tétel (27.18. tétel) és $Q \equiv Q''$ alapján. Q testében szereplő $l_1, l_2, \ldots, l_p$ literálok mindegyike legfeljebb egy nézet literál kifejtéséből kapott tagra képződik. Ha Q'-ben több, mint p nézet literál szerepel, akkor Q'' testében néhány nézet literál kifejtése diszjunkt φ képétől. Ezek a nézet literálok elhagyhatók Q'-ből, úgy, hogy az ekvivalencia nem változik. ■

A 27.27. lemma alapján a következő tétel mondható ki minimális átírások bonyolultságáról.

27.28. tétel. *Legyen Q konjunktív lekérdezés, $\mathcal{V}$ konjunktív lekérdezésekkel megadott nézetek halmaza, mindkettő összehasonlítási atomok nélkül. Tegyük fel, hogy Q testében p literál szerepel.*

1. *Annak eldöntése, hogy létezik-e Q-nak $\mathcal{V}$-t használó Q' átírása, amelyik legfeljebb $k(\leq p)$ literált használ, NP-teljes.*
2. *Annak eldöntése, hogy Q-nak $\mathcal{V}$-t használó Q' átírása, amelyik legfeljebb $k(\leq p)$ adatbázis literált használ, NP-teljes.*

3. *Annak eldöntése, hogy létezik-e Q-nak $\mathcal{V}$-t használó teljes átírása, NP-teljes.*

Bizonyítás. Az első állítást bizonyítjuk, a másik kettő bizonyítása hasonló. A 27.27. és 27.26. lemmák alapján csak olyan átírásokat kell tekintenünk, amelyekben legfeljebb annyi literál szerepel, mint a lekérdezésben, a lekérdezés literáljainak egy részhalmazát tartalmazza, és nem használ új változókat. Egy ilyen átírást valamint az ekvivalenciát bizonyító homomorfizmusokat polinomiális időben tudunk ellenőrizni, tehát a feladat NP-beli. Az NP-nehézség bizonyításához a 27.25. tételt használjuk. Adott Q lekérdezéshez és V nézethez legyen V' az a nézet, amelyiknek a feje megegyezik V fejével, a teste pedig Q és V testének konjunkciója. Könnyen látható, hogy pontosan akkor létezik V'-t használó átírás 1 literállal, amikor létezik V-t használó (korlátozások nélküli) átírás. ■

27.3.3. Gyakorlati algoritmusok

Ebben a fejezetben csak teljes átírásokkal foglalkozunk. Ez nem jelent igazi korlátozást, mert ha adatbázis literálokat is szeretnénk használni, akkor bevezethetünk olyan nézeteket, amelyek egy az egyben tükrözik az adatbázis relációkat. A 27.22. definícióban bevezetett ***ekvivalens*** átírás fogalma megfelelő, ha az átírás célja lekérdezés optimalizálás, illetve a fizikai adatfüggetlenség biztosítása. Azonban, adategyesítési, illetve adattárház környezetben nem törekedhetünk arra, hogy az átírás ekvivalens legyen, mert nem feltétlenül áll rendelkezésre minden adat. Ezért bevezetjük a ***maximálisan tartalmazott átírás*** fogalmát, amelyik függ attól, melyik lekérdezési nyelvet használjuk, ellentétben az ekvivalens átírásokkal.

27.29. definíció. *Legyen Q lekérdezés, $\mathcal{V}$ nézetek halmaza, $\mathcal{L}$ pedig lekérdezési nyelv. Q-nak $\mathcal{V}$-t használó $\mathcal{L}$-re vonatkozó **maximálisan tartalmazott átírása** Q', ha*

1. *Q' az $\mathcal{L}$ nyelv olyan lekérdezése, amelyik csak a $\mathcal{V}$-beli nézeteket használja,*
2. *Q tartalmazza Q'-t,*
3. *ha $Q_1 \in \mathcal{L}$ lekérdezésre teljesül, hogy $Q' \sqsubseteq Q_1 \sqsubseteq Q$, akkor $Q' \equiv Q_1$.*

Lekérdezés optimalizálás materializált nézetek használatával

Mielőtt rátérünk arra, hogyan lehet egy hagyományos optimalizáló eljárást módosítani, hogy adatbázis relációk helyett materializált nézetekkel dolgozzon, át kell tekintenünk, mikor használható egy nézet valamely lekérdezés megválaszolásához. Lényegileg a V nézet használható a Q lekérdezéshez, ha V definíciójában szereplő adatbázis relációk és a Q-ban szereplő relációk halmazainak közös része nem üres, és V olyan attribútumokat is kiválaszt, amelyeket Q is. Ezen kívül, ekvivalens átírás esetén, ha V-ben vannak összehasonlítási atomok olyan attribútumokra, amelyek Q-ban is szereplenek, akkor a nézet logikailag ekvivalens, vagy gyengébb összehasonlítási feltételt kell alkalmazzon, mint a lekérdezés. Ha logikailag erősebb feltételt alkalmaz, akkor a nézet egy (maximálisan) tartalmazott átírás része lehet. Ezt legegyszerűbben egy példán keresztül világíthatjuk meg. Tekintsük a Q lekérdezést az **Egyetem** séma felett, amelyik az olyan tanár, diák, félév hármasokat sorolja fel, ahol a tanár a diák témavezetője és az adott félévben a diák a tanár valamelyik óráját felvette.

$$Q\colon q(Tnév,Diák,Félév) \leftarrow Felvett(Diák,K\text{-}szám,Félév), Témavezető(Tnév,Diák), \\ Tanít(Tnév, K\text{-}szám,Félév, x_V), Félév \geq \text{“2000ősz”}. \tag{27.61}$$

Az alábbi V_1 nézet használható Q megválaszolásánál, mivel ugyanazt az összekapcsolási feltételt használja a *Felvett* és *Tanít* relációkra, mint Q, amint azt az azonos nevű változók mutatják. Ezen kívül, V_1 kiválasztja a *Diák, Tnév, Félév* attribútumokat, ami szükséges ahhoz, hogy a *Témavezető* relációval megfelelően összekapcsolhassuk, és a végeredménybe kiválaszthassuk a három attribútumot. Végül, a *Félév* > "1999ősz" összehasonlítási atom logikailag gyengébb feltétel, mint a Q-ban szereplő *Félév* ≥ "2000ősz".

$$V_1: v_1(Diák,Tnév,Félév) \leftarrow Tanít(Tnév,K\text{-}szám,Félév,x_V), \\ Felvett(Diák,K\text{-}szám,Félév), Félév > \text{"1999ősz"}. \tag{27.62}$$

A következő négy nézet mutatja, hogy V_1-t csak kicsit módosítva hogyan változik a felhasználhatóság.

$$V_2: v_2(Diák,Félév) \leftarrow Tanít(x_T,K\text{-}szám,Félév,x_V), \\ Felvett(Diák,K\text{-}szám,Félév), Félév > \text{"2000ősz"}. \tag{27.63}$$

$$V_3: v_3(Diák,Tnév,Félév) \leftarrow Tanít(Tnév, K\text{-}szám, x_F, x_V), \\ Felvett(Diák,K\text{-}szám,Félév), Félév > \text{"2000ősz"}. \tag{27.64}$$

$$V_4: v_4(Diák,Tnév,Félév) \leftarrow Tanít(Tnév, K\text{-}szám,Félév,x_V), \\ Témavezető(Tnév,x_D), Tanár(Tnév,x_{Sz}), \\ Felvett(Diák,K\text{-}szám,Félév), Félév > \text{"2000ősz"}. \tag{27.65}$$

$$V_5: v_5(Diák,Tnév,Félév) \leftarrow Tanít(Tnév, K\text{-}szám,Félév,x_V), \\ Felvett(Diák,K\text{-}szám,Félév), Félév > \text{"2001ősz"}. \tag{27.66}$$

A V_2 nézet majdnem ugyanaz, mint V_1, az egyetlen különbség, hogy nem választja ki a *Tnév* attribútumot a *Tanít* relációból. Az azonban szükséges a *Témavezető* relációval való természetes összekapcsoláshoz, valamint Q eredményében is szerepel. Így, ha használni akarjuk V_2-t az átíráshoz, akkor újra össze kell kapcsolni a *Tanít* relációval. Azonban, ha a *Felvett* és a *Tanít* relációk természetes összekapcsolása már csak kevés sort tartalmaz az eredeti relációk sorszámához képest, akkor megérheti az újabb összekapcsolás.

A V_3 nézetben a *Felvett* és a *Tanít* relációk összekapcsolása csak a *K-szám* attribútum szerint történik, a *Félév* és a x_F változók egyenlőségét nem követeli meg. Mivel a x_F attribútumot V_3 nem választja ki, ezért nem lehet később összekapcsolási feltételben alkalmazni, így V_3 használata nem jelent nyereséget.

A V_4 nézet csak olyan tanárokat vesz figyelembe, akiknek van legalább egy szakterületük. Ezzel több feltételt alkalmaz, mint az eredeti Q lekérdezés, tehát nem alkalmazható ekvivalens átírásra, ha nem engedjük meg egyesítés és tagadás használatát is a lekérdezési nyelvben. Azonban, ha az adatbázisban van olyan integritási feltétel, hogy minden tanárnak van legalább egy szakterülete, akkor a lekérdezés optimalizáló észre kell vegye, hogy V_4 használható.

Végül, V_5 összehasonlítási operátora erősebb feltételt használ, mint a Q-ban szereplő, így ekvivalens átíráshoz nem, csak (maximálisan) tartalmazott átíráshoz használható.

System-R stílusú optimalizálás

Mielőtt a hagyományos optimizálás változtatásait tárgyalnánk, röviden összefoglaljuk hogyan dolgozik a ***System-R stílusú optimizáló.*** A legjobb végrehajtási sorrendet alulról

felfelé építkezve keresi meg. Első fázisban 1 méretű rész-lekérdezéseket tekint, azaz a lekérdezésben szereplő minden egyes relációs táblához megkeresi a legjobb elérési utat. Az n-edik fázisban N méretű végrehajtási terveket tekint, amelyeket kisebbek (k és $n-k$ méretűek) kombinációjával kap. Az eljárás akkor fejeződik be, ha olyan tervet talál, amelyik a lekérdezés összes relációját lefedi. Az eljárás hatékonysága abból adódik, hogy a végrehajtási terveket ***ekvivalencia osztályokra*** bontja, és minden osztályból csak egyetlen tervet tekint. Két terv ugyanabban az osztályban van, ha

- a lekérdezésben szereplő relációk közül ugyanazokat fedik le (tehát ugyanaz a méretük), valamint
- a választ ugyanabban a (lekérdezés szempontjából) érdekes sorrendben adják.

A mi esetünkben az optimizáló a lekérdezés végrehajtási tervét nem adatbázis relációkra, hanem nézetekre alapozza. Ezért a szokásosan rendelkezésre álló meta-adatokon kívül (pl. statisztikák, indexek) az optimalizáló rendelkezésére állnak a nézeteket definiáló lekérdezés kifejezések is. Az alábbi változtatásokra van szükség.

A. A lekérdezés megválaszolásához használható nézeteket ki kell választani, a fentiekben vázolt feltételek alapján. A hagyományos optimizáló esetében ez triviális, hiszen egy adatbázis reláció pontosan akkor használható a lekérdezés megválaszolásához, ha szerepel a lekérdezésben.

B. Mivel a lekérdezés végrehajtás terve nézetek összekapcsolását jelenti, nem pedig adatbázis relációkét, a tervek nem oszthatók szépen ekvivalencia osztályokba, mint a hagyományos esetben, és így nem lehet őket méret szerint növekvő sorrendben végig nézni. Ezért a következő módosítások szükségesek.

1. ***Megállási feltétel***: Az eljárás megkülönbözteti a *részleges lekérdezés végrehajtási terveket* a *teljes lekérdezés végrehajtási tervektől*. A lehetséges összekapcsolási sorrendek végig tekintése akkor fejeződik be, ha már nincs több ellenőrizetlen részleges lekérdezés végrehajtási terv. Ezzel szemben, a hagyományos optimizáló eljárás akkor ér véget, ha áttekintette azon ekvivalencia osztályokat, amelyek a lekérdezés összes relációját tartalmazzák.
2. ***Végrehajtási tervek elhagyása***: A hagyományos optimizáló eljárás az *egy ekvivalencia osztályba* tartozó terveket hasonlítja össze páronként, és csak a legolcsóbbat tárolja el minden osztályból. A mi esetünkben *tetszőleges* két eddig előállított tervet össze hasonlít. A p tervet elhagyja, ha létezik olyan p' terv, amelyikre igaz, hogy

   (a) p' olcsóbb, mint p, és
   (b) p' legalább annyit hozzájárul a lekérdezés megválaszolásához, mint p. Ez lényegileg azt jelenti, hogy legalább annyi adatbázis relációt lefed, mint p, és legalább annyi szükséges attribútumot ki is választ.

3. ***Részleges tervek társítása***: A hagyományos esetben, ha két részleges tervet társítunk egy nagyobb tervvé, akkor a hozzájuk tartozó összekapcsolási feltétel, egyértelműen adott a lekérdezésben, az optimalizáló eljárás csak a leghatékonyabb megvalósítást kell megtalálja. A mi esetünkben azonban *a priori* – előzetesen – egyáltalán nem világos, hogy milyen összekapcsolási feltétel eredményez ekvivalens átírást. Tehát az optimalizáló eljárás több, különböző összekapcsolási feltételt kell

megvizsgáljon. Szerencsére, a gyakorlatban a rendelkezésre álló meta-adatok lényegesen szűkítik a vizsgálandó feltételek körét. Például, nincs túl sok értelme megpróbálni összekapcsolni egy szöveg típusú attribútumot egy numerikus attribútummal. Hasonlóan az integritási feltételek is csökkenthetik a számba vehető összekapcsolások számát. Miután az összes lehetséges összekapcsolást végig vizsgálta, az optimalizáló azt is ellenőrzi, hogy a kapott terv az még mindig a lekérdezés részleges megoldása-e.

A fentieket az alábbi összehasonlító táblázatban foglalhatjuk össze.

Hagyományos optimalizáló	**Nézeteket használó optimalizáló**
1. Fázis	***1. Fázis***
a) Keressük meg az összes lehetséges elérési utat.	a1) Keressük meg az összes nézetet, amelyik használható a lekérdezés megválaszolásához a2) Különböztessük meg a részleges és teljes terveket.
b) Hasonlítsuk össze a költségüket és tartsuk meg a legolcsóbbat.	b) Hasonlítsuk össze páronként a nézeteket. Ha valamelyik nem járul többel hozzá a lekérdezés megválaszolásához mint egy másik, és nem is olcsóbb annál, akkor hagyjuk el.
c) Ha a lekérdezésben egy reláció szerepel, **stop**.	Ha nincs részleges megvalósítási terv, **stop**.
2. Fázis	***2. Fázis***
Tekintsük az előző fázisban talált elérési utak összekapcsolásait, amelyek a lekérdezésnek megfelelőek, az összes lehetséges összekapcsolási eljárással.	a1) Tekintsük az előző fázisban talált részleges megoldások összekapcsolásait minden lehetséges összekapcsolási eljárással, minden lehetséges összekapcsolási feltételt alkalmazva. a2) Különböztessük meg a részleges és teljes terveket.
b) Hasonlítsuk össze a kapott összekapcsolási tervek költségét, és tartsuk meg a legolcsóbbat.	Ha valamelyik újonnan előállított megoldás nem használható a lekérdezés megválaszolásához, vagy valamelyik másik minden tekintetben jobb nála, akkor hagyjuk el.
c) Ha a lekérdezésben két reláció szerepel, **stop**.	Ha nincs részleges megvalósítási terv, **stop**.
3. Fázis	***3. Fázis***
⋮	⋮

Ekvivalens átírások másik módozata az átalakítási szabályok alkalmazása. A közös gondolat, hogy a hagyományos optimizáló átalakítási szabályaihoz hozzáveszik azt, hogy a lekérdezés valamely részét helyettesíteni lehet egy nézettel. Ezekkel részletesebben nem foglalkozunk.

A fentiekben tárgyalt optimalizáló eljárások elsősorban olyan helyzetekre készültek, amikor a szereplő nézetek száma nem nagy, legalábbis összehasonlítható az adatbázis relációk számával. Ezzel ellentétben az adategyesítés környezetben nagyszámú nézet kezelésére kell felkészülnünk, mivel minden egyes adatforrás egy–egy újabb nézetet jelent. Ezenkívül a nézetek bonyolult predikátumokat tartalmazhatnak, hiszen a céljuk, hogy az egyes adatforrások közti finom különbségek leírják. További különbség, hogy az adategyesítési környezetben nézetekről általában feltesszük, hogy nem teljesek, azaz nem tartalmazzák a definíciójukat kielégítő összes sort, csak azok egy részhalmazát. A továbbiakban ismertetünk néhány, az adategyesítés céljára kifejlesztett eljárást.

Vödör algoritmus

A vödör algoritmus célja, hogy a felhasználó közvetített sémában megfogalmazott lekérde-

zését átfogalmazza olyan lekérdezésre, amelyik közvetlenül a rendelkezésre álló adatforrásokra hivatkozik. Feltesszük, hogy konjunktív lekérdezésekről van szó, melyekben lehetnek összehasonlítási atomok. A Q lekérdezés összehasonlítási atomjainak halmazát $C(Q)$-val jelöljük.

Mivel a lehetséges átírások száma exponenciális a lekérdezés méretében, ezért a vödör algoritmus fő gondolata, hogy a lehetséges átírások számát drasztikusan csökkenthetjük, ha a lekérdezés ***részcéljait*** – a benne szereplő relációs atomokat – egyenként tekintjük és meghatározzuk mely nézetek használhatók a részcélokhoz külön–külön.

Az eljárás általános menete a következő. Először minden részcélhoz egy ***vödröt*** rendelünk, amelyik azon nézeteket tartalmazza, ahonnan a részcél sorait vehetjük. A második lépésben az összes olyan összekapcsolást tekintjük, amelyik minden vödörből tartalmaz egy nézetet, és ellenőrizzük, hogy az így kapott V konjunktív lekérdezés átírás szemantikusan helyes-e, azaz $V \sqsubseteq Q$ teljesül-e, vagy szemantikusan helyessé tehető-e összehasonlítási atomok hozzáadásával. Végül a megmaradó terveket minimalizáljuk a redundáns részcélok elhagyásával. Az alábbi Vödör-készítő eljárás az első lépést hajtja végre. A bemenet adatforrások leírásának $\mathcal{V}$ halmaza, valamint Q konjunktív lekérdezés,

$$Q\colon\ Q(\tilde{X})\ \leftarrow\ R_1(\tilde{X}_1), R_2(\tilde{X}_2), \ldots, R_m(\tilde{X}_m), C(Q) \tag{27.67}$$

formában.

Vödör-készítő(Q,$\mathcal{V}$)

```
1  for i ← 1 to m
2      do Vödör_i ← ∅
3         for minden V ∈ 𝒱
4                                          ▷ V: V(Ỹ) ← S_1(Ỹ_1),...S_n(Ỹ_n), C(V) formájú.
5             do for j ← 1 to n
6                if R_i = S_j
7                  then Legyen φ a V változóin következőképpen definiált leképezés:
8                       if Ỹ_j k-adik változója y és y ∈ Ỹ
9                         then φ(y) = x_k, ahol x_k az X̃_i k-adik változója
10                        else φ(y) egy új változó, ami nem szerepel sem Q-ban sem V-ben.
11                      Q'() ← R_1(X̃_1), R_m(X̃_m), C(Q), S_1(φ(Ỹ_1)), ..., S_n(φ(Ỹ_n)), φ(C(V))
12                      if Kielégíthető≥(Q')
13                        then adjuk φ(V)-t Vödör_i-hez.
```

A Kielégíthető$^{\geq}$ eljárás a 27.1.2. pontban leírt Kielégíthető eljárás kiterjesztése arra az esetre, ha egyenlőség atomok mellett egyenlőtlenség atomok is szerepelhetnek a szabály testében. A szükséges változtatás annyi, hogy minden olyan y változóra, amelyik egyenlőtlenség atomban szerepel, ellenőrizni kell, hogy az y-ra kirótt egyenlőtlenségek egyszerre teljesíthetőek-e.

A Vödör-készítő eljárás polinomiális lépésszámú Q és $\mathcal{V}$ méretének függvényében. Valóban, a 2. és 5. sorok egymásba ágyazott ciklusának magja $n\sum_{V\in\mathcal{V}}|V|$-szer fut le. A 6–13. sorok utasításai a 12. sor kivételével konstans sok lépést jelentenek. A 12. sor **if** utasításának feltételét polinomiális időben lehet ellenőrizni.

A Vödör-készítő eljárás helyességének igazolásához nézzük meg, hogy milyen feltételekkel teszi be a V nézetet $Vödör_i$-be. A 6. sorban ellenőrzi, hogy V-ben szerepel-e rész-

célként az R_i reláció. Ha nem, akkor nyilván V nem adhat használható információt a Q-beli R_i részcélhoz. Ha V-ben szerepel részcélként az R_i reláció, akkor a 9–10. sorokban elkészíti azt a megfeleltetést amelyet a változókra alkalmazva az S_j és R_i részcélok megfeleltethetők egymásnak a Q illetve V fejében levő relációkkal összhangban. Végül a 12. sor ellenőrzi, hogy az így kapott változó megfeleltetésekkel az összehasonlítási atomok nem mondanak-e ellent.

A második lépésben, miután a vödröket a VÖDÖR-KÉSZÍTŐ eljárással elkészítette, a vödör algoritmus ***konjunktív lekérdezés átírások*** halmazát állítja elő. Minden átírás olyan konjunktív lekérdezés, amelyik minden vödörből tartalmaz pontosan egy tényezőt. Az algoritmus eredménye ezen konjunktív lekérdezés átírások egyesítése, hiszen a különböző átírások különböző sorokat adhatnak az eredményhez. Adott Q' konjunktív lekérdezés ***konjunktív lekérdezés átírás,*** ha

1. $Q' \sqsubseteq Q$, vagy
2. Q' kiegészíthető összehasonlítási atomokkal úgy, hogy az előző teljesüljön.

27.10. példa. *Vödör algoritmus.* Tekintsük a következő Q lekérdezést, amelyik azokat az x cikkeket listázza, amely cikkekhez létezik y cikk ugyanabban a témában, hogy x és y kölcsönösen hivatkoznak egymásra. Rendelkezésre áll három nézet V_1, V_2, V_3.

$$\begin{array}{lcl} Q(x) & \leftarrow & idéz(x,y), idéz(y,x), uaTéma(x,y) \\ V_1(a) & \leftarrow & idéz(a,b), idéz(b,a) \\ V_2(c,d) & \leftarrow & uaTéma(c,d) \\ V_3(f,h) & \leftarrow & idéz(f,g), idéz(g,h), uaTéma(f,g) \end{array} \tag{27.68}$$

Első lépésben a VÖDÖR-KÉSZÍTŐ eljárással az alábbi vödröket állítjuk elő.

$idéz(x,y)$	$idéz(y,x)$	$uaTéma(x,y)$
$V_1(x)$	$V_1(x)$	$V_2(x)$
$V_3(x)$	$V_3(x)$	$V_3(x)$

(27.69)

A második lépésben a vödrök direkt szorzatának minden eleméből szerkeszt az algoritmus egy Q' konjunktív lekérdezést, és ellenőrzi, hogy Q tartalmazza-e Q'-t. Ha igen, akkor hozzáadja a válaszhoz.

Esetünkben megpróbálja V_1-t a többi nézettel összerakni, de így nem kap helyes eredményt. Ennek oka, hogy b nem szerepel V_1 fejében, így a Q-ban szereplő összekapcsolási feltételt – az x és y változók szerepelnek *uaTéma* relációban is – nem tudja alkalmazni. Ezek után a V_3-t tartalmazó átírásokat tekinti, és észreveszi, hogy V_3 fejében található változókat egyenlővé téve tartalmazott átírást kap. Végül, az algoritmus azt is megtalálja, hogy V_3-t és V_2-t kombinálva is átírást kap. Egyszerű további ellenőrzéssel kapjuk, hogy ez utóbbi átírás redundáns, V_2-t el lehet hagyni belőle. Tehát a vödör algoritmus eredménye a (27.68) lekérdezésre és nézetekre a (ténylegesen ekvivalens)

$$Q'(x) \leftarrow V_3(x,x)\,. \tag{27.70}$$

A vödör algoritmus előnye, hogy jelentősen lecsökkenti az ellenőrizendő konjunktív átírás jelöltek számát. Ha az adatforrások alapvetően az összehasonlítási atomokban különböznek egymástól, akkor várhatóan a vödrök mérete kicsi lesz.

A vödör algoritmus fő hátránya pont abban rejlik, amiben az előnye is. Semmilyen becslésünk nincs arra, hogy a vödrök direkt szorzatának a mérete mekkora lesz, lehet, hogy nagy. Továbbá az eljárás minden egyes lehetséges átírásra elvégez egy lekérdezés tartalmazás ellenőrzést, ami már akkor is NP-teljes, ha nincsenek összehasonlítási atomok.

Inverz szabályok

A vödör algoritmusnál általánosabban használható eljárás az ***inverz szabályok*** alkalmazása. Tetszőleges, negáció nélküli, de rekurziót megengedő datalog programmal adott lekérdezéshez megtalálja a maximálisan tartalmazott átírást polinomiális időben.

Az első kérdés az, hogy adott $\mathcal{P}$ datalog program és $\mathcal{V}$ konjunktív nézetek halmaza esetén létezik-e olyan $\mathcal{P}$-vel ekvivalens $\mathcal{P}_v$ datalog program, amelynek EDB relációi a $\mathcal{V}$-beli $v_1, v_2, \ldots, v_n$ relációk. Sajnos, azonban ez a kérdés algoritmikusan eldönthetetlen. Meglepő viszont az, hogy el tudjuk készíteni a lehető legjobb, maximálisan tartalmazott átírást. Abban az esetben, ha létezik $\mathcal{P}$-vel ekvivalens $\mathcal{P}_v$ datalog program, akkor az eljárásunk azt fogja előállítani, hiszen a maximálisan tartalmazott átírás tartalmazza $\mathcal{P}_v$-t is. Ez csak látszólagos ellentmondás avval az állítással, hogy az ekvivalens átírás algoritmikusan eldönthetetlen, hiszen az inverz szabályokkal előállított maximálisan tartalmazott átírásról nem tudjuk eldönteni, hogy ténylegesen ekvivalens-e.

27.11. példa. *Ekvivalens átírás.* Tekintsük a következő $\mathcal{P}$ datalog programot, ahol *él* és $fekete$ relációk EDB relációk, egy G gráf éleit, illetve feketére színezett csúcsait tartalmazzák.

$$\begin{array}{lll} \mathcal{P}: & q(X,Y) & \leftarrow \quad él(X,Z), él(Z,Y), fekete(Z) \\ & q(X,Y) & \leftarrow \quad él(X,Z), fekete(Z), q(Z,Y) \end{array} \tag{27.71}$$

Könnyen ellenőrizhető, hogy $\mathcal{P}$ a G gráf olyan útjainak (pontosabban sétáinak) végpontjait adja meg, amelyek minden belső pontja fekete. Tegyük fel, hogy csak az alábbi két nézet érhető el.

$$\begin{array}{lll} v_1(X,Y) & \leftarrow \quad él(X,Y), fekete(X) \\ v_2(X,Y) & \leftarrow \quad él(X,Y), fekete(Y) \end{array} \tag{27.72}$$

v_1 a fekete kezdőpontú, v_2 a fekete végpontú éleket tárolja. Ekkor a $\mathcal{P}$ datalog programnak létezik ekvivalens $\mathcal{P}_v$ átírása, amelyik csak a v_1 és v_2 nézeteket használja EDB relációként:

$$\begin{array}{lll} \mathcal{P}_v: & q(X,Y) & \leftarrow \quad v_2(X,Z), v_1(Z,Y) \\ & q(X,Y) & \leftarrow \quad v_2(X,Z), q(Z,Y) \end{array} \tag{27.73}$$

Azonban, ha csak az v_1, vagy v_2 nézet érhető el, akkor nem lehetséges az ekvivalens átírás, mert csak olyan utakat kaphatunk, amelyeknek a kezdő, illetve végpontja fekete.

Az inverz szabály eljárás leírásához szükségünk lesz a ***datalog program,*** illetve a ***datalog szabály*** általánosítására, a ***Horn szabályra.*** Ha a 27.11. definícióban szereplő (27.27) szabály u_i szabad soraiban a változók és konstansok mellett még ***függvény szimbólumokat*** is megengedünk, akkor ***Horn szabályról*** beszélünk. Horn szabályok halmazát ***logikai programnak*** nevezzük. Ebben az értelemben egy függvény szimbólum mentes logikai program lesz datalog program. A 27.11. definíció *edb*, *idb* fogalma logikai programra ugyanúgy értelmezhető.

Az inverz szabály eljárás két lépésből áll. Először olyan logikai programot készítünk, amelyik tartalmazhat függvény szimbólumokat. Azonban ezek a függvény szimbólumok nem szerepelnek rekurzív szabályokban, így a második lépében a logikai programot datalog programmá lehet alakítani.

27.30. definíció. *A*

$$v(X_1, \ldots, X_m) \leftarrow v_1(\tilde{Y}_1), \ldots, v_n(\tilde{Y}_n) \tag{27.74}$$

szabállyal meghatározott v nézet v^{-1} ***inverze*** *Horn szabályok következő halmaza. Minden*

$v_i(\tilde{Y}_i)$ részcélnak megfelel egy szabály, amelyiknek teste a $v(X_1, \ldots, X_m)$ literál. A szabály feje $v_i(\tilde{Z}_i)$, ahol a $\tilde{Z}_i$-t $\tilde{Y}_i$-ból úgy kapjuk, hogy a (27.74) szabály fejében szereplő változókat meghagyjuk, ezen kívül minden, a fejben nem szereplő y változó helyére pedig az $f_y(X_1, \ldots, X_m)$ függvény szimbólumot írjuk. Különböző változókhoz különböző függvény szimbólumok tartoznak. A $\mathcal{V}$ nézet halmaz $\mathcal{V}^{-1}$ inverze a $\{v^{-1} : v \in \mathcal{V}\}$ halmaz, ahol a különböző nézetek inverzeiben különböző függvény szimbólumok szereplenek.

Az inverz definíció gondolata, hogy ha a v nézetben megjelenik a $(x_1, \ldots x_m)$ sor valamilyen $x_1, \ldots x_m$ konstansokkal, akkor minden a fejben nem szereplő y változónak van valamilyen kiértékelése, ami a szabály testét igazzá teszi. Ezt az "ismeretlen" kiértékelést jelöljük az $f_y(X_1, \ldots, X_m)$ szimbólummal.

27.12. példa. *Nézetek inverze.* Legyen $\mathcal{V}$ az alábbi nézetek halmaza.

$$\begin{array}{lcl} v_1(X,Y) & \leftarrow & \acute{e}l(X,Z), \acute{e}l(Z,W), \acute{e}l(W,Y) \\ v_2(X)) & \leftarrow & \acute{e}l(X,Z) \end{array} \tag{27.75}$$

Ekkor $\mathcal{V}^{-1}$ a következő Horn szabályokból áll.

$$\begin{array}{lcl} \acute{e}l(X, f_{1,Z}(X,Y)) & \leftarrow & v_1(X,Y) \\ \acute{e}l(f_{1,Z}(X,Y), f_{1,W}(X,Y)) & \leftarrow & v_1(X,Y) \\ \acute{e}l(f_{1,W}(X,Y), Y) & \leftarrow & v_1(X,Y) \\ \acute{e}l(X, f_{2,Z}(X)) & \leftarrow & v_2(X) \end{array} \tag{27.76}$$

Ezek után $\mathcal{P}$ datalog programhoz és konjunktív nézetek $\mathcal{V}$ halmazához könnyű elkészíteni azt a logikai programot, amelyből majd azt a datalog programot kapjuk, ami $\mathcal{P}$ $\mathcal{V}$-t használó maximálisan tartalmazott átírása.

$\mathcal{P}$-ből töröljük az összes olyan szabályt, amelyikben olyan EDB reláció szerepel, ami nem fordul elő $\mathcal{V}$-beli nézet definíciójában. Az így kapott $\mathcal{P}^-$ programhoz hozzávesszük a $\mathcal{V}^{-1}$ szabályait, és ezáltal nyerjük a $(\mathcal{P}^-, \mathcal{V}^{-1})$ logikai programot. Vegyük észre, hogy $\mathcal{P}$ megmaradt EDB relációi a $(\mathcal{P}^-, \mathcal{V}^{-1})$ logikai programban IDB relációk, mivel a $\mathcal{V}^{-1}$ szabályai fejében szerepelnek. Az IDB relációk elnevezése tetszőleges, így átnevezhetjük őket, hogy ne egyezzen a nevük $\mathcal{P}$ EDB relációinak nevével. Ezt azonban itt a könnyebb érthetőség kedvéért nem tesszük meg.

27.13. példa. *Logikai program.* Tekintsük az alábbi datalog programot, ami az *él* reláció tranzitív lezártját számítja ki.

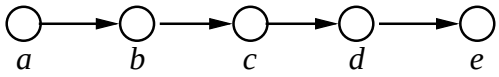
$$\begin{array}{lcl} \mathcal{P}: \quad q(X,Y) & \leftarrow & \acute{e}l(X,Y) \\ q(X,Y) & \leftarrow & \acute{e}l(X,Z), q(Z,Y) \end{array} \tag{27.77}$$

Tegyük fel, hogy csak a

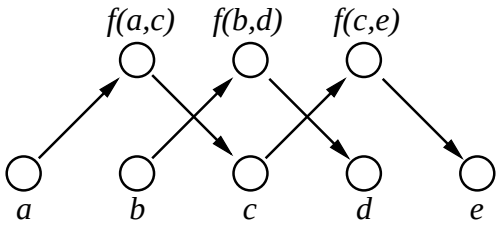
$$v(X,Y) \leftarrow \acute{e}l(X,Z), \acute{e}l(X,Y) \tag{27.78}$$

materializált nézet érhető el, amelyik a kettő hosszúságú utak végpontjait tárolja. Ha csak ezt a nézetet használhatjuk, akkor a legtöbb amit remélhetünk, hogy a páros hosszúságú utak végpontjait tudjuk előállítani. Mivel $\mathcal{P}$ egyetlen EDB relációja az *él*, ami szerepel v definíciójában, ezért $(\mathcal{P}^-, \mathcal{V}^{-1})$ logikai programot úgy kapjuk, hogy $\mathcal{P}$-hez hozzávesszük $\mathcal{V}^{-1}$ szabályait.

$$\begin{array}{lcl} (\mathcal{P}^-, \mathcal{V}^{-1}): \quad q(X,Y) & \leftarrow & \acute{e}l(X,Y) \\ q(X,Y) & \leftarrow & \acute{e}l(X,Z), q(Z,Y) \\ \acute{e}l(X, f(X,Y)) & \leftarrow & v(X,Y) \\ \acute{e}l(f(X,Y), Y) & \leftarrow & v(X,Y) \end{array} \tag{27.79}$$



27.4. ábra. A G gráf.



27.5. ábra. A G' gráf.

A $\mathcal{P}$ datalog program *él* EDB relációjának példánya legyen a 27.4. ábrán látható G gráf. Ekkor $(\mathcal{P}^-, \mathcal{V}^{-1})$ három új konstanst vezet be, melyek $f(a,c)$, $f(b,d)$ és $f(c,e)$. A $\mathcal{V}^{-1}$ program *él* IDB relációja a 27.5. ábrán látható G' gráfot adja. $\mathcal{P}^-$ a G' gráf tranzitív lezártját számítja ki. Vegyük észre, hogy azok a párok a tranzitív lezártban, amelyek nem tartalmaznak egyet sem az új konstansokból, pontosan a G-beli páros hosszú utak végpontjai.

A 27.13. példában a $(\mathcal{P}^-, \mathcal{V}^{-1})$ logikai program eredményét például a NAIV-DATALOG eljárással számíthatjuk ki. Azonban, logikai programokra általában nem igaz, hogy az eljárás véget ér. Tekintsük ugyanis a

$$\begin{array}{lcl} q(X) & \leftarrow & p(X) \\ q(f(X)) & \leftarrow & q(X) \end{array} \tag{27.80}$$

logikai programot. Ha a p EDB reláció az a konstanst tartalmazza, akkor a program eredménye az $a, f(a), f(f(a)), f(f(f(a))), \ldots$ végtelen sorozat lesz. Ezzel ellentétben az inverz szabály eljárás által adott $(\mathcal{P}^-, \mathcal{V}^{-1})$ logikai program eredménye ***garantáltan*** véges, és a kiszámítási algoritmus véges időben véget ér.

27.31. tétel. *Tetszőleges $\mathcal{P}$ datalog programra, konjunktív nézetek $\mathcal{V}$ halmazára és a nézetek tetszőleges véges példányaira teljesül, hogy a $(\mathcal{P}^-, \mathcal{V}^{-1})$ logikai programnak egyértelmű minimális fixpontja van, valamint a* NAIV-DATALOG *és* FÉLIG-NAIV-DATALOG *eljárások ezt a fixpontot adják eredményül.*

A 27.31. tétel bizonyításának lényege, hogy függvény szimbólumokat csak az inverz szabályok vezetnek be, amelyek azonban nem rekurzívak, így egymásba ágyazott függvény szimbólumokat tartalmazó tényezők nem keletkeznek. A bizonyítás részletezését az Olvasóra bízzuk (27.3-3. gyakorlat).

Még ha egy adatbázis EDB relációiból indulunk is ki, egy logikai program eredményében lehetnek olyan sorok, amelyek függvény szimbólumokat tartalmaznak. Ezért bevezetünk egy szűrőt, ami eltávolítja a szükségtelen sorokat. Ha a $\mathcal{P}$ logikai program EDB relációinak példánya a D adatbázis, akkor $\mathcal{P}(D)\downarrow$ jelöli azon $\mathcal{P}(D)$-beli sorok halmazát, amelyek

nem tartalmaznak függvény szimbólumokat. Jelölje $\mathcal{P}\downarrow$ azt a programot, amelyik adott D példányra $\mathcal{P}(D)\downarrow$-t számítja ki. Az alábbi tétel bizonyítása meghaladja jelen fejezet kereteit.

27.32. tétel. *Tetszőleges $\mathcal{P}$ datalog programra, valamint konjunktív nézetek $\mathcal{V}$ halmazára teljesül, hogy $(\mathcal{P}^-, \mathcal{V}^{-1})\downarrow$ logikai program $\mathcal{P}$ $\mathcal{V}$-t használó maximálisan tartalmazott átírása. Továbbá $(\mathcal{P}^-, \mathcal{V}^{-1})$ előállítható $\mathcal{P}$ és $\mathcal{V}$ méretében polinomiális időben.*

A 27.32. tétel jelentése, hogy az az egyszerű eljárás, hogy a nézet definíciók inverzeit hozzáadjuk a datalog programhoz olyan logikai programot eredményez, ami a lehető legjobban használja fel a nézeteket. Az hogy $(\mathcal{P}^-, \mathcal{V}^{-1})$ előállítható $\mathcal{P}$ és $\mathcal{V}$ méretében polinomiális időben, könnyen látható, hiszen minden $v_i \in \mathcal{V}$ minden részcéljához egyetlen inverz szabályt kell elkészíteni.

Az átírási feladat teljes megoldásához szükséges azonban egy olyan datalog programot előállítani, amelyik ekvivalens a $(\mathcal{P}^-, \mathcal{V}^{-1})\downarrow$ logikai programmal. Ehhez adja a kulcsot az az észrevétel, hogy $(\mathcal{P}^-, \mathcal{V}^{-1})$-ben csak véges sok függvény szimbólum van, továbbá az alulról felfelé történő kiszámításban, mint a Naiv-datalog eljárás és változatai, egymásba ágyazott függvény szimbólumok nem jönnek létre. Megfelelő könyveléssel nyomon követhetjük a függvény szimbólumok megjelenését, anélkül, hogy ténylegesen előállítanánk azokat tartalmazó sorokat.

Az átalakítást alulról felfelé végezzük, a Naiv-datalog eljáráshoz hasonlóan. $\mathcal{V}^{-1}$-beli IDB relációban megjelenő $f(X_1, \ldots, X_k)$ függvény szimbólumot a $X_1, \ldots, X_k$ változó listával helyettesítjük. Ugyanakkor az IDB reláció nevet meg kell jelölni, hogy tudjuk, a $X_1, \ldots, X_k$ lista a $f(X_1, \ldots, X_k)$ függvényhez tartozott. Ezzel új, "ideiglenes" reláció neveket vezetünk be. Tekintsük a 27.13. példa (27.79) logikai programjában szereplő

$$\acute{e}l(X, f(X, Y)) \leftarrow v(X, Y) \tag{27.81}$$

szabályt a

$$\acute{e}l^{\langle 1, f(2,3)\rangle}(X, X, Y) \leftarrow v(X, Y) \tag{27.82}$$

szabállyal helyettesítjük. A $\langle 1, f(2,3)\rangle$ jelölés értelmezése, hogy $\acute{e}l^{\langle 1, f(2,3)\rangle}$ első argumentuma megegyezik *él* első argumentumával, a második és harmadik argumentum $\acute{e}l^{\langle 1, f(2,3)\rangle}$-ben az f függvény szimbólummal együtt adja *él* második argumentumát. Ha $(\mathcal{P}^-, \mathcal{V}^{-1})$ alulról felfelé kiszámítása során $\mathcal{P}^-$ valamelyik IDB relációjának argumentumába függvény szimbólum kerülne, akkor egy új szabályt adunk a programhoz, a megfelelően megjelölt reláció nevekkel.

27.14. példa. *Logikai program átalakítása datalog programmá.* A 27.13. példa logikai programját az alábbi datalog programmá alakítja át a fent vázolt eljárás. A Naiv-datalog alulról felfelé végrehajtásának különböző fázisait vonalak határolják el.

$$\begin{array}{lcl}
\acute{e}l^{\langle 1, f(2,3)\rangle}(X, X, Y) & \leftarrow & v(X, Y) \\
\acute{e}l^{\langle f(1,2), 3\rangle}(X, Y, Y) & \leftarrow & v(X, Y) \\
\hline
q^{\langle 1, f(2,3)\rangle}(X, Y_1, Y_2) & \leftarrow & \acute{e}l^{\langle 1, f(2,3)\rangle}(X, Y_1, Y_2) \\
q^{\langle f(1,2), 3\rangle}(X_1, X_2, Y) & \leftarrow & \acute{e}l^{\langle f(1,2), 3\rangle}(X_1, X_2, Y) \\
\hline
q(X, Y) & \leftarrow & \acute{e}l^{\langle 1, f(2,3)\rangle}(X, Z_1, Z_2), q^{\langle f(1,2), 3\rangle}(Z_1, Z_2, Y) \\
q^{\langle f(1,2), f(3,4)\rangle}(X_1, X_2, Y_1, Y_2) & \leftarrow & \acute{e}l^{\langle f(1,2), 3\rangle}(X_1, X_2, Z), q^{\langle 1, f(2,3)\rangle}(Z, Y_1, Y_2) \\
\hline
q^{\langle f(1,2), 3\rangle}(X_1, X_2, Y) & \leftarrow & \acute{e}l^{\langle f(1,2), 3\rangle}(X_1, X_2, Z), q(Z, Y) \\
q^{\langle 1, f(2,3)\rangle}(X, Y_1, Y_2) & \leftarrow & \acute{e}l^{\langle 1, f(2,3)\rangle}(X, Z_1, Z_2), q^{\langle f(1,2), f(3,4)\rangle}(Z_1, Z_2, Y_1, Y_2)
\end{array} \tag{27.83}$$

Az így kapott datalog program egyértelműen mutatja, hogy melyik argumentumokban keletkezhet függvény szimbólum az eredeti logikai programban. Azonban, bizonyos függvény szimbólumokat tartalmazó sorok sohasem eredményeznek függvény szimbólumokat nem tartalmazó sorokat a program eredményének kiszámítása során.

A p relációt ***fontosnak*** nevezzük, ha a 27.16. definícióban megadott előzmény gráfban[2] p-ből van irányított út a program q eredmény relációjába. Ha p nem fontos, akkor a program eredményének kiszámításához nincs szükség p-beli sorokra, így p elhagyható a programból.

27.15. példa. *Nem fontos relációk elhagyása.* A 27.14. példában kapott datalog program előzmény gráfjában a $q^{\langle 1,f(2,3)\rangle}$ és $q^{\langle f(1,2),f(3,4)\rangle}$ relációkból nincs irányított út a program q eredmény relációjába, ezért nem fontosak, azaz el lehet hagyni őket és a hozzájuk tartozó szabályokat. Az alábbi datalog programot kapjuk ezután:

$$\begin{array}{lcl} \acute{e}l^{\langle 1,f(2,3)\rangle}(X,X,Y) & \leftarrow & v(X,Y) \\ \acute{e}l^{\langle f(1,2),3\rangle}(X,Y,Y) & \leftarrow & v(X,Y) \\ q^{\langle f(1,2),3\rangle}(X_1,X_2,Y) & \leftarrow & \acute{e}l^{\langle f(1,2),3\rangle}(X_1,X_2,Y) \\ q^{\langle f(1,2),3\rangle}(X_1,X_2,Y) & \leftarrow & \acute{e}l^{\langle f(1,2),3\rangle}(X_1,X_2,Z),q(Z,Y) \\ q(X,Y) & \leftarrow & \acute{e}l^{\langle 1,f(2,3)\rangle}(X,Z_1,Z_2),q^{\langle f(1,2),3\rangle}(Z_1,Z_2,Y) \end{array} \tag{27.84}$$

Még egy egyszerűsítő lépést hajthatunk végre, ami ugyan nem csökkenti a szükséges levezetések számát az eredmény kiszámítása során, viszont elkerül felesleges adat másolásokat. Ha p olyan reláció egy datalog programban, amelyet egyetlen szabály definiál, és annak az egyetlen szabálynak a testében csak egyetlen reláció áll, akkor p elhagyható a programból: Minden olyan szabályban, amelyiknek a testében p előfordul, p-t helyettesíthetjük a p-t definiáló szabály testében szereplő relációval, a változók megfelelő egyenlővé tétele után.

27.16. példa. *Adatmásolás elkerülése.* A 27.14. példában $\acute{e}l^{\langle 1,f(2,3)\rangle}$ és $\acute{e}l^{\langle f(1,2),3\rangle}$ relációkat egyetlen szabály határozza meg, amely szabályok testében egyetlen reláció szerepel. Ezért a (27.84) program tovább egyszerűsíthető.

$$\begin{array}{lcl} q^{\langle f(1,2),3\rangle}(X,Y,Y) & \leftarrow & v(X,Y) \\ q^{\langle f(1,2),3\rangle}(X,Z,Y) & \leftarrow & v(X,Z),q(Z,Y) \\ q(X,Y) & \leftarrow & v(X,Z),q^{\langle f(1,2),3\rangle}(X,Z,Y) \end{array} \tag{27.85}$$

A fenti két egyszerűsítés eredményeként kapott datalog programot $(\mathcal{P}^-,\mathcal{V}^{-1})^{datalog}$-al jelöljük. Világos, hogy $(\mathcal{P}^-,\mathcal{V}^{-1})$ és $(\mathcal{P}^-,\mathcal{V}^{-1})^{datalog}$ alulról felfelé történő kiszámítása közt kölcsönösen egyértelmű megfeleltetés létezik. Mivel a függvény szimbólumokat $(\mathcal{P}^-,\mathcal{V}^{-1})^{datalog}$-ban nyomon követjük, ezért tudjuk, hogy az eredményül kapott példány megegyezik $(\mathcal{P}^-,\mathcal{V}^{-1})$ eredménye függvény szimbólumot nem tartalmazó sorainak részhalmazával.

27.33. tétel. *Tetszőleges $\mathcal{P}$ datalog programra és konjunktív nézetek $\mathcal{V}$ halmazára, a $(\mathcal{P}^-,\mathcal{V}^{-1})\downarrow$ program ekvivalens a $(\mathcal{P}^-,\mathcal{V}^{-1})^{datalog}$ programmal.*

[2] Itt az előzmény gráf definícióját ki kell terjesztenünk a datalog program EDB relációira is.

MiniCon

A vödör algoritmus hátránya alapvetően az, hogy a nézetek részcéljai közti kölcsönhatások nagy részét nem figyeli meg azáltal, hogy minden egyes részcélt elkülönítve vizsgál. Így a vödrök sok használhatatlan nézetet tartalmazhatnak, és az algoritmus második fázisa nagyon költségessé válhat.

Az inverz szabály eljárás előnye a fogalmi egyszerűsége és modularitása. A nézetek inverzeit csak egyszer kell kiszámítani, utána már tetszőleges datalog programmal adott lekérdezéshez használhatóak. Azonban, az inverzek használatával elveszíthetjük azt az előnyt, hogy a nézet már kiszámított néhány szükséges összekapcsolást. A lekérdezés megválaszolása során ugyanis az inverz szabály eljárás lényegileg újra előállítja az adatbázis relációkat.

A ***MiniCon*** algoritmus az előző kettő hátrányait próbálja kiküszöbölni. A kulcs gondolata, hogy ahelyett, hogy a lekérdezés ***részcéljaihoz*** keresnénk átírásokat, azt vizsgálja, hogy a lekérdezés ***változói*** hogyan működnek együtt a rendelkezésre álló nézetekkel. A továbbiakban visszatérünk a konjunktív lekérdezésekhez, és a könnyebb érthetőség kedvéért csak olyan lekérdezéseket és nézeteket tekintünk, amelyek nem tartalmaznak konstansokat.

A MiniCon eljárás úgy kezdődik, mint a vödör algoritmus, azt vizsgálja, melyik nézetek tartalmaznak a lekérdezés valamelyik részcéljának megfelelő részcélt. Azonban, amikor az algoritmus talál egy részleges leképezést a lekérdezés g részcéljáról valamelyik V nézet g_1 részcéljára, nézőpontot vált, és a lekérdezés változóit tekinti. Az algoritmus megvizsgálja a lekérdezés összekapcsolási feltételeit – amelyeket a változók többszörös előfordulásai határoznak meg – és megkeresi további részcélok olyan minimális halmazát, amit még a V részcéljaira kell képezni, feltéve, hogy g-t g_1-re képezzük. Részcéloknak ez a halmaza, valamint a leképezési információ együttese lesz a ***MiniCon leírás*** (MCL). A második fázisban az MCL-ket kapcsolja össze a MiniCon algoritmus. Az MCL-k előállítása szükségtelenné teszi a vödör algoritmus legköltségesebb részének, az átírások és a lekérdezés közti tartalmazás ellenőrzésének végrehajtását, mert az MCL-k előállítási szabálya biztosítja, hogy az összekapcsolásuk helyes eredményt ad.

Adott $\tau\colon Var(Q) \longrightarrow Var(V)$ leképezés esetén azt mondjuk, hogy a V nézet g_1 részcélja ***fedi*** a Q lekérdezés g részcélját, ha $\tau(g) = g_1$. $Var(Q)$, illetve $Var(V)$ a lekérdezés, valamint a nézet változóinak halmazát jelöli. Ahhoz, hogy belássuk egy átírásról, hogy csupa, a lekérdezés eredményéhez tartozó sort ad, meg kell adnunk egy homomorfizmust a lekérdezésről az átírásra. Egy MCL ezen homomorfizmus egy részletének tekinthető, így a részletek majd könnyen összekapcsolhatók lesznek.

A Q lekérdezés átírása a nézeteket használó konjunktív lekérdezések egyesítése. Ezekben az egyes nézetek fejében szereplő változók közül néhány lehet, hogy azonosítva lett, mint a 27.10. példában kapott (27.70) ekvivalens átírásban. Tehát célszerű bevezetnünk a ***fej homomorfizmus*** fogalmát. A $h\colon Var(V) \longrightarrow Var(V)$ leképezés fej homomorfizmus, ha identitás a V fejében nem szereplő változókon, de azonosíthat fejben szereplő változókat. Minden x V fejében szereplő változóra $h(x)$ is V fejében szerepel, továbbá $h(x) = h(h(x))$. Ezek után megadhatjuk az MCL pontos definícióját.

27.34. definíció. *A Q lekérdezéshez a V nézet feletti **MiniCon leírás (MCL)** a $C = (h_C, V(\tilde{Y})_C, \varphi_C, G_C)$ négyes, ahol*

- *h_C fej homomorfizmus V-n,*
- *$V(\tilde{Y})_C$-t a h_C alkalmazásával kapjuk V-ből, azaz $\tilde{Y} = h_C(\tilde{A})$, ahol $\tilde{A}$ a V fejében szereplő változók halmaza,*

- φ_C *részleges leképezés* $Var(Q)$*-ról* $h_C(Var(V))$*-be,*
- G_C *a* Q *olyan részcéljainak egy halmaza, amelyeket valamelyik* $H_C(V)$*-beli részcél fed, a* φ_C *leképezéssel.*

Az alábbi állításon alapszik az MCL-ket előállító eljárás.

27.35. állítás. *Legyen* C *a* V *nézet feletti MiniCon leírás a* Q *lekérdezéshez.* C *csak akkor használható a* Q *nem redundáns átírásához, ha*

F1. *minden olyan* x *változóra, amelyik* Q *fejében van és* φ_C *értelmezési tartományában is,* $\varphi_C(x)$ *a* $h_C(V)$ *fejében található, valamint*

F2. *ha* $\varphi_C(y)$ *nem szerepel* $h_C(V)$ *fejében, akkor* Q *minden olyan* g *részcéljára, amelyik tartalmazza* y*-t, teljesül, hogy*

1. *g minden változója szerepel φ_C értelmezési tartományában, és*
2. $\varphi_C(g) \in h_C(V)$.

F1. feltétel lényegileg ugyanaz, mint a vödör algoritmus feltétele arra, mikor kerül bele egy nézet egy vödörbe. **F2.** jelentése, hogy ha az x változó szerepel a lekérdezés valamelyik összekapcsolási feltételében, amelyik feltételt a nézet nem teljesíti, akkor x-nek szerepelnie kell a nézet fejében, hogy az őt tartalmazó összekapcsolási feltétel később alkalmazható legyen valamilyen másik részcéllal az átírás folyamán. A MCL-készítő eljárás Q konjunktív lekérdezéshez és konjunktív nézetek $\mathcal{V}$ halmazához megadja a használható MiniCon leírásokat.

MCL-készítő$(Q, \mathcal{V})$

```
1  C ← ∅
2  for Q minden g részcéljára
3      do for V ∈ 𝒱
4          do for V minden v részcéljára
5              do Legyen h a legáltalánosabb fej homomorfizmus V-n, amelyikre van
                  φ leképezés, hogy φ(g) = h(v).
6                 if φ és h létezik
7                   then Adjuk C-hez azon C MCL-ket, amelyek magadhatók úgy, hogy:
8                            (a) φ_C (h_C) a φ (h) kiterjesztése,
9                            (b) G_C a Q részcéljainak olyan minimális részhalmaza, melyre
                                 27.35. állítás teljesül, és
10                           (c) φ és h nem terjeszthető ki φ'_C és h'_C-re úgy, hogy (b) teljesül,
                                 és a (b)-ben meghatározott G'_C-re G'_C ⊊ G_C.
11 return C
```

Tekintsük újra a 27.10. példa (27.68) lekérdezését és nézeteit. Az MCL-készítő eljárás először a lekérdezés *idéz*(x, y) részcélját tekinti. Nem állít elő MCL-t a V_1 nézethez, mivel 27.35. állítás **F2.** feltétele megsérülne. Ugyanis V_1-nek a *uaTéma*(x, y) részcélt is fednie kéne a $\varphi(x) = a$, $\varphi(y) = b$ leképezésnél, hiszen b nincs V_1 fejében.[3] Ugyanezen ok miatt

[3] $\varphi(x) = b$, $\varphi(y) = a$ eset hasonló.

a lekérdezés többi részcéljánál sem készít MCL-t a V_1 nézethez. Valamilyen értelemben a MiniCon eljárás a vödör algoritmus második lépésének bizonyos részeit az MCL-készítő eljárásban elvégzi. Az alábbi táblázat mutatja az MCL-készítő eredményét.

$V(\tilde{Y})$	h	φ	G
$V_2(c,d)$	$c \to c, d \to d$	$x \to c, y \to d$	3
$V_3(f,f)$	$f \to f, h \to f$	$x \to f, y \to f$	1, 2, 3

(27.86)

Az MCL-készítő eljárás minimális olyan G_C részcél halmazt ad meg, ami teljesíti a 27.35 állítás feltételeit. Ez lehetővé teszi, hogy a MiniCon algoritmus második fázisában csak olyan MCL-ket kapcsoljunk össze, amelyek a lekérdezés részcéljainak ***páronként diszjunkt*** részhalmazait fedik.

27.36. állítás. *Adott Q lekérdezésre és nézetek $\mathcal{V}$, valamint MCL-k C halmazára csak olyan $C_1, \ldots C_l$ MCL-k kapcsolhatók össze Q nem redundáns átírásává, amelyekre teljesül, hogy*

F3. *$G_{C_1} \cup \cdots \cup G_{C_l}$ Q összes részcélját tartalmazza, és*

F4. *minden $i \neq j$-re $G_{C_i} \cap G_{C_j} = \emptyset$.*

Az, hogy csak páronként diszjunkt MCL-ket érdemes összekapcsolni, jelentősen lecsökkenti a keresési teret. A MCL-összekapcsoló eljáráshoz még egy jelölést be kell vezetnünk. A C MCL φ_C leképezése Q változóinak egy egész halmazát képezheti $h_C(V)$ ugyanazon változójára. E halmaz egy tetszőlegesen választott reprezentánsát kiválasztjuk, arra ügyelve, hogy ha a halmazban van Q fejében található változó, akkor egy olyat. $EC_{\varphi_C}(x)$ jelöli annak a halmaznak a reprezentáns változóját, amelyikbe x tartozik. A C MiniCon leírást EC_{φ_C}-vel kiegészítve a $(h_C, V(\tilde{Y}), \varphi_C, G_C, EC_{\varphi_C})$ ötösként kezeljük. Ha a $C_1, \ldots, C_k$ MCL-ket akarjuk összekapcsolni, és valamilyen $i \neq j$-re $EC_{\varphi_{C_i}}(x) = EC_{\varphi_{C_i}}(y)$ és $EC_{\varphi_{C_j}}(y) = EC_{\varphi_{C_j}}(z)$ teljesül, akkor az összekapcsolással kapott konjunktív átírásban x, y és z is ugyanarra a változóra lesz leképezve. Jelölje S_C azt az ekvivalencia relációt Q változóin, amelynek osztályai a φ_C által ugyanarra a változóra képezett elemek, azaz $xS_Cy \iff EC_{\varphi_C}(x) = EC_{\varphi_C}(y)$. $\mathcal{C}$ az MCL-készítő eljárás eredményeként kapott MCL-k halmaza.

MCL-ÖSSZEKAPCSOLÓ(C)

```
1  Válasz ← ∅
2  for {C_1, ..., C_n} ⊆ C amelyre G_{C_1}, ..., G_{C_n} a Q részcéljainak partíciója
3      Definiáljuk a Ψ_i leképezést az Ỹ_i-n a következőképpen:
4      if Q-ban van x változó melyre φ_i(x) = y
5          then Ψ_i(y) = x
6          else Ψ_i(y) az y egy új példánya.
7      Legyen S az S_{C_1} ∪ ··· ∪ S_{C_n} tranzitív lezártja.
8                                          ▷ S ekvivalencia reláció Q változóin.
9      Az S minden osztályához jelöljünk ki egy reprezentánst.
10     Definiáljuk az EC leképezést a következőképpen:
11     if x ∈ Var(Q)
12         then EC(x) az S x-t tartalmazó osztályának reprezentánsa
13         else EC(x) = x
14     Legyen Q' a Q'(EC(X̃)) ← V_{C_1}(EC(Ψ_1(Ỹ_1))), ..., V_{C_n}(EC(Ψ_n(Ỹ_n)))
15     Válasz ← Válasz ∪ {Q'}
16 return Válasz
```

Igaz az alábbi tétel.

27.37. tétel. *Adott konjunktív Q lekérdezésre és konjunktív nézetek $\mathcal{V}$ halmazára a MiniCon algoritmus által előállított konjunktív lekérdezések egyesítése a Q $\mathcal{V}$-t használó maximálisan tartalmazott átírása.*

A 27.37. tétel teljes bizonyítása meghaladja e fejezet kereteit. A 27-1. feladatban az Olvasóra bízzuk annak belátását, hogy az MCL-ÖSSZEKAPCSOLÓ eljárás eredményeként kapott konjunktív lekérdezések egyesítését Q tartalmazza.

Megjegyezzük, hogy a vödör algoritmus, az inverz szabályok és a MiniCon algoritmus futási ideje legrosszabb esetben megegyező: $O(nmM^n)$, ahol n a lekérdezés részcéljainak száma, m a nézetek részcéljainak maximális száma és M a nézetek száma. Azonban gyakorlati futási eredmények azt mutatják, hogy nagyszámú nézet esetén (3–400 nézet) a MiniCon algoritmus lényegesen gyorsabb, mint a másik kettő.

Gyakorlatok

27.3-1. A 27.24. állítás és a 27.20. tétel felhasználásával bizonyítsuk be a 27.25 . tételt.

27.3-2. Bizonyítsuk be a 27.26. lemma két állítását. *Útmutatás.* Az első állításhoz Q'-ben a $v_i(\tilde{Y}_i)$ nézetek helyére írjuk be a definíciójukat. Az így kapott Q'' lekérdezést minimalizáljuk a 27.19. tétel segítségével. A második bizonyítandó állításhoz használjuk a 27.24. állítást, amellyel bizonyítsuk be, hogy létezik h_i homomorfizmus a $v_i(\tilde{Y}_i)$ nézetet definiáló konjunktív leképezés testéből Q testébe. Lássuk be, hogy a $\tilde{Y}'_i = h_i(\tilde{Y}_i)$ választás megfelelő.

27.3-3. Bizonyítsuk be a 27.31. tételt, felhasználva, hogy a datalog programok minimális fixpontja egyértelmű.

Feladatok

27-1. MiniCon helyes
Bizonyítsuk be, hogy a MiniCon algoritmus helyes eredményt ad. *Útmutatás.* Elegendő belátni, hogy ha bármelyik, az MCL-összekapcsoló eljárás 14. sorában megadott Q' konjunktív lekérdezésre igaz, hogy $Q' \sqsubseteq Q$. Ez utóbbihoz készítsünk homomorfizmust Q-ról Q'-re.

27-2. $(\mathcal{P}^-, \mathcal{V}^{-1})\downarrow$ ***helyes***
Bizonyítsuk be, hogy a $(\mathcal{P}^-, \mathcal{V}^{-1})\downarrow$ logikai program eredményének minden sora benne van $\mathcal{P}$ eredményében. (A 27.32. tétel bizonyításának része.) *Útmutatás.* Legyen t olyan sor $(\mathcal{P}^-, \mathcal{V}^{-1})$ eredményében, melyik nem tartalmaz függvény szimbólumot. Tekintsük t levezetési fáját. Ennek levelei nézet literálok, hiszen azok a $(\mathcal{P}^-, \mathcal{V}^{-1})$ program extenzionális relációi. Ha ezeket a leveleket elhagyjuk a levezetési fából, a maradék fa levelei már $\mathcal{P}$ EDB relációi. Mutassuk meg, hogy az így kapott fa t levezetési fája a $\mathcal{P}$ datalog programban.

27-3. Datalog nézet
Ezzel a feladattal azt szeretnénk megindokolni, miért csak konjunktív nézet definíciókat tekintettünk. Legyen $\mathcal{V}$ nézetek halmaza, Q pedig lekérdezés. A nézetek adott $\mathcal{I}$ példánya esetén a t sor a Q lekérdezés ***biztos válasza,*** ha tetszőleges olyan $\mathcal{D}$ adatbázis példányra, amelyikre teljesül, hogy $\mathcal{I} \subseteq \mathcal{V}(\mathcal{D})$, $t \in Q(\mathcal{D})$ fenn áll.

a. Bizonyítsuk be, hogy ha a $\mathcal{V}$-beli nézetek datalog programmal vannak meghatározva, a Q lekérdezés konjunktív, amelyik nem-egyenlőségi ($\neq$) atomokat tartalmazhat, az a kérdés, hogy a nézetek adott $\mathcal{I}$ példánya esetén egy t sor a Q lekérdezés biztos válasza-e, algoritmikusan eldönthetetlen. *Útmutatás.* Vezessük vissza rá a ***Post Megfeleltetési Problémát,*** ami a következő: Adott az $\{a, b\}$ ábécé feletti szavak két, $\{w_1, w_2, \ldots, w_n\}$ és $\{w'_1, w'_2, \ldots, w'_n\}$ halmaza. A kérdés az, hogy létezik-e olyan index sorozat $i_1, i_2, \ldots, i_k$ (ismétlések megengedettek), hogy

$$w_{i_1} w_{i_2} \cdots w_{i_k} = w'_{i_1} w'_{i_2} \cdots w'_{i_k} \tag{27.87}$$

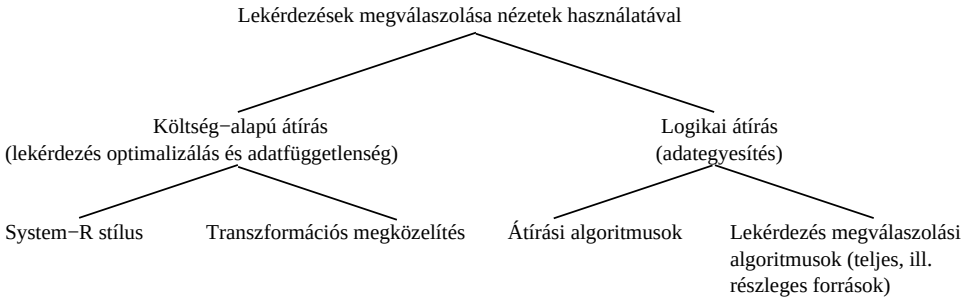
A Post Megfeleltetési Problémáról ismeretes, hogy algoritmikusan eldönthetetlen. Legyen a V nézet az alábbi datalog programmal adva:

$$\begin{array}{lcl} V(0,0) & \leftarrow & S(e,e,e) \\ V(X,Y) & \leftarrow & V(X_0,Y_0), S(X_0,X_1,\alpha_1), \ldots, S(X_{g-1},Y,\alpha_g), \\ & & S(Y_0,Y_1,\beta_1), \ldots, S(Y_{h-1},Y,\beta_h) \\ & & \text{ahol } w_i = \alpha_1 \ldots \alpha_g \text{ és } w'_i = \beta_1 \ldots \beta_h \\ & & \text{egy szabály minden } i \in \{1,2,\ldots,k\}\text{-ra} \\ S(X,Y,Z) & \leftarrow & P(X,X,Y), P(X,Y,Z) \end{array} \tag{27.88}$$

Legyen továbbá Q a következő konjunktív lekérdezés.

$$Q(c) \leftarrow P(X,Y,Z), P(X,Y,Z'), Z \neq Z' \tag{27.89}$$

Lássuk be, hogy a V nézet $\mathcal{I}$ példánya esetén, melyre $\mathcal{I}(V) = \{\langle e,e\rangle\}$ és $\mathcal{I}(S) = \{\}$, a $\langle c\rangle$ sor pontosan akkor lesz Q biztos válasza, ha a $\{w_1, w_2, \ldots, w_n\}$ és $\{w'_1, w'_2, \ldots, w'_n\}$ halmazokra a Post Megfeleltetési Problémának *nincs* megoldása.



27.6. ábra. Lekérdezés átírás használatának osztályozása.

b. Az *a.* pontban leírt kiszámíthatatlansági eredménnyel ellentétben, ha $\mathcal{V}$ konjunktív nézetek halmaza, és a Q lekérdezés a $\mathcal{P}$ datalog programmal adott, akkor tetszőleges t sorról könnyen eldönthető, hogy a Q lekérdezés biztos válasza-e adott $\mathcal{I}$ nézet példány esetén. Bizonyítsuk be, hogy a $(\mathcal{P}^{-}, \mathcal{V}^{-1})^{datalog}$ datalog program pontosan a Q biztos válaszában található sorokat adja eredményül.

Megjegyzések a fejezethez

A "lekérdezések megválaszolása nézetek használatával" feladat kezelését több szempontból lehet osztályozni. A 27.6. ábrán az osztályozás vázlata látható.

A legfontosabb választóvonal a különböző munkák közt, hogy a céljuk adategyesítés, vagy pedig lekérdezés optimalizálás és a fizikai adatfüggetlenség elérése. A két megközelítés közti különbség kulcsa a lekérdezéseket megválaszoló nézeteket használó algoritmus kimenete. Az első esetben, adott Q lekérdezéshez és nézetek $\mathcal{V}$ halmazához az eljárás célja olyan Q' lekérdezés előállítása, amelyik a nézetekre hivatkozik, és ekvivalens Q-val, vagy Q tartalmazza Q'-t. A második esetben az eljárásnak tovább kell lépnie, és egy (remélhetőleg) optimális végrehajtási tervet is elő kell állítania a Q megválaszolására a nézetek (és esetleg adatbázis relációk) használatával. Ebben az esetben csak ekvivalens átírásokat tekinthetünk.

A hasonlóság a két megközelítés között az, hogy mindkettő alapkérdése, hogy egy átírás vajon ekvivalens-e, vagy tartalmazza-e a lekérdezés. Azonban, amíg logikai helyesség elegendő az adategyesítés nézőpontból, addig lekérdezés optimalizálási szempontból nem, ott a ***legolcsóbb,*** a nézeteket használó végrehajtási tervet kell megtalálni. A nehézségek abból adódnak, hogy az optimalizáló algoritmusok olyan nézeteket is figyelembe kell vegyenek, amelyek ugyan nem járulnak hozzá az átírás logikai helyességéhez, de csökkentik a végrehajtási terv költségét. Ezért az adategyesítési algoritmusok helyességének indoklása főként logikai, míg az optimizálóké logikai és költség-alapú is. Másfelől, adategyesítési problémakörben alapvető adottság a nézetek nagy száma, amelyek a különböző adatforrásoknak felelnek meg. Ezzel ellentétben, az optimalizálási feladatnál általában (de nem mindig!) feltesszük, hogy a nézetek száma a séma nagyságával összehasonlítható.

A lekérdezés optimalizálás téma munkái tovább oszthatók ***System-R stílusú,*** valamint transzformációs optimizálókra. Az előbbiekhez tartoznak Chaudhuri, Krishnamurty, Poto-

mianos és Shim [6]; Tsatalos, Solomon, és Ioannidis [28] munkái. Az utóbbihoz Florescu, Raschid, és Valduriez [13]; Bello és társai [3]; Deutsch, Popa és Tannen [8], Zaharioudakis és társai [33], valamint Goldstein és Larson[16] cikkei.

Az adategyesítési munkák közül átírási algoritmusokkal foglalkoznak Yang és Larson [31]; Levy, Mendelzon, Sagiv és Srivastava [21]; Qian [26]; valamint Lambrecht, Kambhampati és Gnanaprakasam [20] cikkei. A vödör algoritmust Levy, Rajaraman és Ordille [23] vezette be. Az inverz szabályok eljárás Duschka és Genesereth [9, 10] munkája. A MiniCon algoritmust Pottinger és Halevy fejlesztették ki [25, 24].

Lekérdezés megválaszolási algoritmusokkal, illetve feladat bonyolultságával foglalkozik Abiteboul és Duschka [2]; Grahne és Mendelzon [17]; valamint Calvanese, De Giacomo, Lenzerini és Vardi [5].

A STORED rendszert Deutsch, Fernandez és Suciu dolgozták ki [7]. A szemantikus gyorstárolást Yang, Karlapalem és Li [32] tárgyalja. Az átírási feladat különböző kiterjesztéseivel [4, 14, 15, 19, 32] foglalkoznak.

A témakör összefoglalása található Abiteboul [1], Florescu, Levy és Mendelzon [12], Halevy [22, 18] valamint Ullman[29] munkáiban.

A fejezet témakörében magyar nyelven a datalog alapjai olvashatók Ullman és Widom [30] könyvében. Az NP-teljesség és algoritmikus eldönthetőség kérdéseit Ivanyos Gábor, Rónyai Lajos és Szabó Réka tankönyve tárgyalja [27]. Logikai programozásról magyar nyelven Peter Flach könyve olvasható [11].

Irodalomjegyzék

[1] S. Abiteboul. Querying semi-structured data. In F. Afrati, P. Kolaitis (szerkesztők), *Proceedings of ICDT'97*, *Lecture Notes in Computer Science* 1186. kötete, 1–18. o. Springer-Verlag, 1997. 2647

[2] S. Abiteboul, O. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 254–263. o. ACM-Press, 1998. 2647

[3] R. Bello, K. Dias, A. Downing, J. Freenan, T. Finnerty, W. D. Norcott, H. Sun, A. Witkowski, M. Ziauddin. Materialized views in Oracle. In *Proceedings of Very Large Data Bases'98*, 659–664. o., 1998. 2647

[4] P. Buneman. Semistructured data. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 117–121. o. ACM-Press, 1997. 2647

[5] D. Calvanese, D. De Giacomo, M. Lenzerini, M. Varni. Answering regular path queries using views. In *Proceedings of the Sixteenth International Conference on Data Engineering*, 190–200. o., 2000. 2647

[6] S. Chaudhury, R. Krishnamurty, S. Potomianos, K. Shim. Optimizing queries with materialized views. In *Proceedings of the Eleventh International Conference on Data Engineering*, 190–200. o., 1995. 2647

[7] A. Deutsch, M. Fernandez, D. Suciu. Storing semistructured data with stored. In *Proceedings of SIGMOD'99*, 431–442. o., 1999. 2647

[8] A. Deutsch, L. Popa, D. Tannen. Physical data independence, constraints and optimization with universal plans. In *Proceedings of VLDB'99*, 459–470. o., 1999. 2647

[9] O. Duschka, M. Genesereth. Answering recursive queries using views. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 109–116. o. ACM-Press, 1997. 2647

[10] O. Duschka, M. Genesereth. Query planning in infomaster. In *Proceedings of ACM Symposium on Applied Computing*, 109–111. o. ACM-Press, 1997. 2647

[11] P. Flach. *Logical programming (???)*. ????, ???? (magyarul: *Logikai programozás*, Panem, 2001). 2647

[12] D. Florescu, A. Levy, A. O. Mendelzon. Database techniques for the world-wide web: a survey. *SIGMOD Record*, 27(3):59–74, 1998. 2647

[13] D. Florescu, L. Raschid, P. Valduriez. Answering queries using oql view expressions. In *Workshop on Materialized views, in cooperation with ACM SIGMOD*, 627–638. o., 1996. 2647

[14] D. D. Florescu. *Search spaces for object-oriented query optimization*. PhD thesis, University of Paris VI, 1996. 2647

[15] M. Friedman, D. S. Weld. Efficient execution of information gathering plans. In *Proceedings International Joint Conference on Artificial Intelligence*, 785–791. o., 1997. 2647

[16] J. Goldstein, P. A. Larson. Optimizing queries using materialized views: a practical, scalable solution. In *Optimizing queries using materialized views: a practical, scalable solution*, 331–342. o., 2001. 2647

[17] G. Grahne, A. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proceedings of ICDT'99*, *Lecture Notes in Computer Science* 1540. kötete, 332–347. o. Springer-Verlag, 1999. 2647

[18] A. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10:270–294, 2001. 2647

[19] C. T. Kwok, D. Weld. Planning to gather information. In *Proceedings of AAAI 13th National Conference on Artificial Intelligence*, 32–39. o., 1996. 2647

[20] E. T. Lambrecht, S. Kambhampati, S. Gnanaprakasam. Optimizing recursive information gathering plans. In *Proceedings of 16th International Joint Conference on Artificial Intelligence*, 1204–1211. o., 1999. 2647

[21] A. Levy, A. Mendelzon, Y. Sagiv, D. Srivastava. Answering queries using views. In *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 95–104. o. ACM-Press, 1995. 2647

[22] A. Levy. Logic based techniques in data integration. In J. Minker (szerkesztő), *Logic-based Artificial Intelligence*, 575–595. o. Kluwer Academic Publishers, 2000. 2647

[23] A. Levy, A. Rajaraman, J. J. Ordille, D. Srivastava. Querying heterogeneous information sources using source descriptions. In *Proceedings of Very Large Data Bases*, 251–262. o., 1996. 2647

[24] R. Pottinger. A scalable algorithm for answering queries using views. *The VLDB Journal*, 10(2):182–198, 2001. 2647

[25] R. Pottinger, A. Halevy. Minicon: A scalable algorithm for answering queries using views. In *Proceedings of Very Large Data Bases'00*, 484–495. o., 2000. 2647

[26] X. Qian. Query folding. In *Proceedings of International Conference on Data Engineering*, 48–55. o., 1996. 2647

[27] L. Rónyai, G. Ivanyos, R. Szabó. *Algoritmusok* (*Algorithms*). Typotex, 1999. 2647

[28] O. G. Tsatalos, M. C. Solomon, Y. Ioannidis. The GMAP: a versatile tool for physical data independence. *The VLDB Journal*, 5(2):101–118, 1996. 2647

[29] J. D. Ullman. Information integration using logical views. In *Proceedings of ICDT'97*, *Lecture Notes in Computer Science* 1186. kötete, 19–40. o. Springer-Verlag, 1997. 2647

[30] J. D. Ullman, J. Widom. *A First Course in Database Systems*. Prentice Hall, 1997 (Magyarul: *Adatbázis-rendszerek. Alapvetés*. Panem, Budapest, 1998). 2647

[31] H. Z. Yang, P. A. Larson. Query transformation for PSJ-queries. In *Proceedings of Very Large Data Bases'87*, 245–254. o., 1987. 2647

[32] J. Yang, K., Karlapalem, Q. Li. Algorithms for materialized view design in data warehousing environment. In *Proceedings of Very Large Data Bases'97*, 136–145. o., 1997. 2647

[33] M. Zaharioudakis, R. Cochrane, G. Lapis, H. Pirahesh, M. Urata. Answering complex SQL queries using automatic summary tables. In *Proceedings of SIGMOD'00*, 105–116. o., 2000. 2647

Névmutató

Tárgymutató

Tartalomjegyzék