29. fejezet

Adatbázisok – lekérdezés és időszerűsítés

A relációs algebra műveletei, használata, műveleti tulajdonságok. Lekérdező nyelvek ekvivalenciája, átírás egyik nyelvből a másikba. Rekurzív lekérdezések az SQL-ben. Lekérdezések kiértékelése és optimalizálási stratégiák.

29.1. A relációs algebra

A relációs algebra egy algebrai alapokon nyugvó *adatmanipulációs nyelv*nek (DML) tekinthető a relációs modell felett.

A relációs algebra alapelemei a relációk (ld. az adatmodellezésről szóló, 28. fejezetet).

29.1.1. A relációs algebra műveletei

Halmazműveletek

Mivel a relációk sorok halmazai, ezért a hagyományos halmazműveleteket $(\cup, \cap, \setminus)$ könnyen értelmezhetjük rájuk, ha kikötjük, hogy *az operandusok sémája azonos* kell legyen.

Vetítés

A vetítéssel néhány oszlopot elhagyhatunk a relációból. A

$$\pi_{A_1,\ldots,A_n}(R)$$

vetület az a reláció, mely R oszlopaiból csak az $A_1, \ldots, A_n$-et tartalmazza (illetve ezek közül azokat, melyek R-nek attribútumai).

Mivel a relációk halmazok, ezért a vetület elemszáma esetleg kisebb lehet, mint $|R|$.

Kiválasztás

A kiválasztással a relációnak csak bizonyos feltételt teljesítő sorait tartjuk meg.

$$\sigma_C(R)$$

tehát R azon sorainak halmaza, melyek kielégítik a C feltételt, ahol C attribútumnevekre vonatkozó egyszerű relációkat $(=, \neq, >, <, \leq, \geq)$ tartalmaz *és*, illetve *vagy* műveletekkel összekapcsolva.

Descartes-szorzat

A Descartes-szorzat a megszokott módon értelmezett: két tetszőleges reláció $R \times S$ Descartes szorzatának egy sora az R és S egy-egy sorának összefűzése, és a szorzat minden ilyen párt tartalmaz.

A Descartes-szorzat sémája R és S sémájának egyesítése úgy, hogy ha azonos attribútumaik vannak, azokat átnevezzük (pl. $R.A$).

Természetes összekapcsolás

A természetes összekapcsolás jele: $R \bowtie S$. Természetes összekapcsolás esetén R-nek és S-nek csak azokat a sorait párosítjuk össze, melyeken a közös attribútumok megegyeznek.

Theta-összekapcsolás

A természetes összekapcsolás kiterjesztése a theta-összekapcsolás: $R \underset{\theta}{\bowtie} S$. Itt a direktszorzatból csak a θ feltételnek megfelelő sorok jelennek meg, a művelet minden egyéb tekintetben a direktszorzattal egyezik meg. Röviden: $R \underset{\theta}{\bowtie} S = \sigma_C(R \times S)$.

Átnevezés

Néha kényelmes, ha egy kifejezésre rövidebb vagy más névvel hivatkozhatunk. A

$$\rho_{S(A_1,\ldots,A_n)}(R)$$

kifejezés az R relációt átnevezi S-re úgy, hogy attribútumait balról jobbra $A_1, \ldots, A_n$ névvel látja el.

29.1.2. A kiterjesztett relációs algebra

A kiterjesztett relációs algebra alapját multihalmazok adják. A korábbi relációs algebrai műveleteket könnyen átfogalmazhatjuk multihalmazokra, azon kívül bevezetünk néhány új műveletet. Nem minden algebrai azonosság vihető át multihalmazokra is.

A kiterjesztett relációs algebra szerepe a gyakorlati felhasználásban rejlik: sok lekérdező nyelv, például az SQL is lényegében kiterjesztett relációs algebrán alapul.

Ismétlődések megszüntetése

Multihalmazokról lévén szó, szükségünk lehet a multihalmaz „halmazzá alakítására", azaz az ismétlődő sorok eltávolítására. Az erre szolgáló művelet jele: $\delta(R)$.

Csoportosítás, összesítő műveletek

Az összesítő műveletek (pl. összeg, átlag, minimum, maximum, számlálás) egy-egy attribútumra vonatkoznak, és a csoportosítással együtt is használhatók.

A reláció sorait csoportokba foghatjuk egy vagy több attribútum értékei alapján. Ezután az egyes oszlopok csoportjaira összesítő műveleteket alkalmazhatunk.

A $\gamma_L(R)$ csoportosítási művelet egy csoportosítást és egy összesítést hajt végre, ahol L attribútumlista, esetleg összesítő függvényekkel.

Ilyen módon δ a csoportosítási művelet speciális esete, ahol nem alkalmazunk összesítést, viszont minden attribútumon csoportosítunk.

Rendezés

A multihalmazok sorai rendezhetők egy attribútum értékei szerint. A rendezés jele: $\tau_L(R)$, ahol L attribútumlista.

Külső összekapcsolások

A külső összekapcsolások lényege, hogy megengedik a „lógó sorokat", azaz olyan sorok is megjelennek az összekapcsolásban, amelyeknek nincs párjuk –

a hiányzó adatok üres értéket ($\perp$) vesznek fel.

Megkülönböztetünk jobb oldali ($\overset{\circ}{\bowtie}_R$), bal oldali ($\overset{\circ}{\bowtie}_L$) és kétoldali ($\overset{\circ}{\bowtie}$) összekapcsolásokat aszerint, hogy melyik reláció „lógó sorait" tartjuk meg.

29.2. Lekérdező nyelvek ekvivalenciája

A különböző ismert lekérdező nyelvek (Datalog, kiterjesztett relációs algebra, SQL) kifejező ereje hasonló, de nem teljesen azonos.

29.2.1. Műveletek átírása

Unió

Relációs algebrában: $E = R \cup S$.

Datalogban:

$E(\dots) \leftarrow R(\dots)$
$E(\dots) \leftarrow S(\dots)$.

SQL-ben: `SELECT * FROM R UNION S`.

Metszet

Relációs algebrában: $E = R \cap S$.

Datalogban: $E(\dots) \leftarrow R(\dots)$ `AND` $S(\dots)$.

SQL-ben: `SELECT * FROM R INTERSECTION S`.

Különbség

Relációs algebrában: $E = R \backslash S$.

Datalogban: $E(\dots) \leftarrow R(\dots)$ `AND NOT` $S(\dots)$.

SQL-ben: `SELECT * FROM R MINUS S`.

Vetítés

Relációs algebrában: $E = \pi_L(R)$.

Datalogban: $E(< L >) \leftarrow R(\dots)$.

SQL-ben: `SELECT <L> FROM R`.

Kiválasztás

Relációs algebrában: $E = \sigma_C(R)$.

Datalogban: $E(\dots) \leftarrow R(\dots)$ `AND` $< C >$.

SQL-ben: `SELECT * FROM R WHERE <C>`.

Szorzat

Relációs algebrában: $E = R \times S$.

Datalogban: $E(< r >< s >) \leftarrow R(< r >)$ `AND` $S(< s >)$.

SQL-ben: `SELECT * FROM R, S`.

Összekapcsolás

A különböző összekapcsolások a Datalogban viszonylag természetesen, aritmetikai részcélokkal, SQL-ben pedig a megfelelő `[NATURAL | INNER | OUTER | LEFT | RIGHT | ...] JOIN` kifejezésekkel fogalmazhatók meg.

29.3. Rekurzív lekérdezések SQL-ben

Az SQL-99 szabvány tartalmazza a rekurzív lekérdezések lehetőségét, bár nem szigorú elvárás ennek implementációja.

Rekurzív lekérdezések a `WITH RECURSIVE` utasítás segítségével fogalmazhatók meg:

```
WITH RECURSIVE <reláció neve> <attribútumlista> AS
                                                (
                                    <kiindulási lekérdezés>
                                                UNION ALL
                                     <rekurzív lekérdezés>
                                                )
```

A `WITH RECURSIVE` utasítás elején lehetőségünk van több lokális lekérdezés definíciójára is.

Az SQL csak lineáris (nem kölcsönös) rekurziót enged meg, ahol a rekurzív felhasználások monotonak (azaz ha a rekurzívan használt relációhoz hozzáadva egy sort, az őt felhasználó lekérdezés eredménye nem csökkenhet).

29.4. Lekérdezések kiértékelése, optimalizálása

A lekérdezések kiértékelése 3 fő lépésből áll:

1. lekérdezés átalakítása elemzőfává,
2. elemzőfa átalakítása relációs algebrai kifejezésfává (*logikai lekérdezésterv*),
3. az elemzőfa optimalizálása és átalakítása fizikai lekérdezéstervvé, mely tartalmazza a végrehajtás sorrendjét és a használt algoritmusokat.

A lekérdezések fordításának első lépésében elemzőfává alakulnak. Ez egy egyszerű szemantikus fa felépítéséhez hasonló folyamat: megtörténik a *relációk*, *attribútumnevek* és *típusok ellenőrzése*, valamint esetleges egyéb szemantikus ellenőrzések. Ezt már könnyű átírni relációs algebrai kifejezésfává, ahol elkezdődhet az optimalizáció.

29.4.1. Algebrai optimalizálás

Az algebrai optimalizálás heurisztikus eljárás, melynek során a logikai lekérdezésterv kifejezésfáját vele ekvivalens, de várhatóan hatékonyabban kiértékelhető formára hozzuk.

Az ekvivalens átalakítás azonosságai

- kommutativitás: $\bowtie$, $\times$, $\cup$, $\cap$,
- asszociativitás: $\bowtie$, $\times$, $\cup$, $\cap$,
- $A \subseteq B \Rightarrow \pi_A(\pi_B(E)) \equiv \pi_A(E)$,
- $\sigma_{C \wedge D}(E) \equiv \sigma_C(\sigma_D(E))$,
- $\sigma_D(\sigma_C(E)) \equiv \sigma_C(\sigma_D(E))$,
- $\sigma_{F_1}(E_1 \times E_2) \equiv \sigma_{F_1}(E_1) \times E_2$,
- unió, különbség és természetes összekapcsolás szűrése átírható szűrések uniójára, különbségére, illetve természetes összekapcsolására,
- $\pi_{A_1 \cup A_2}(E_1 \times E_2) \equiv \pi_{A_1}(E_1) \times \pi_{A_2}(E_2)$,
- $\pi_{A_1 \cup A_2}(E_1 \cup E_2) \equiv \pi_{A_1}(E_1) \cup \pi_{A_2}(E_2)$.

Az optimalizálás elvei

1. az összetett feltételekkel felírt kiválasztásokat szétvágjuk,
2. a vetítéseket mozgassuk minél lejjebb a fában,
3. a kiválasztásokat is vigyük minél lejjebb a fában (lehetőleg a vetítések alá),
4. a szorzás utáni kiválasztásokat próbáljuk meg természetes összekapcsolássá alakítani,
5. vonjuk össze az ágymásba ágyazott kiválasztásokat, illetve vetítéseket,
6. keressünk közös részkifejezéseket és jegyezzük meg, hogy ezeket elég egyszer kiszámolni a végrehajtás során.

A végrehajtás sorrendje

Partícionáljuk úgy a kifejezésfát, hogy minden bináris műveletet a fölötte a következő bináris műveletig elhelyezkedő unáris műveletekkel tekintsünk egy csúcsnak – ezt a fát értékeljük ki alulról felfelé.

29.4.2. Fizikai lekérdezésterv optimalizálása

A fizikai lekérdezésterv optimalizálása az egyes műveletek végrehajtási költségének és be-, illetve kimeneti méretének becslésén nyugszik a táblaméretek ismeretében.

Az alapvető relációs algebrai műveletek eredményének mérete az *egyenletességi feltevéssel* könnyen becsülhető.

Kiválasztás

Lineáris kereséssel. Műveletigénye az elemszámmal arányos.

Logaritmikus kereséssel. Csak rendezett adatsorban alkalmazható.

Elsődleges indexszel. A költség indexszintek számával arányos.

Másodlagos indexszel. A keresett rekordok kiválasztási számosságának és az indexszintek számának összegével arányos a költség. Sok megfelelő rekord esetén a lináris keresés hatékonyabb.

Halmazműveletek

Feladat a duplikációk kiszűrése, és esetleg a művelet hatékony elvégzéséhez az adatok rendezése. A rendezés lehet *külső* vagy *belső rendezés* attól függően, hogy az adatok elférnek-e a memóriában.

Vetítés

A mezők számával arányos műveletigény.

Duplikációk szűrése (`DISTINCT`)

A szűréshez először rendezni kell az adatsort az összes mező szerint, majd törölni a szomszédos azonos sorokat.

Összekapcsolás

Skatulyázott ciklusok (nested loop). Lényege, hogy egy ciklusban végighaladunk az egyik (R, külső) reláció elemein, egy belső ciklusban pedig R minden sorához megpróbáljuk hozzákapcsolni a másik (belső) reláció minden sorát.

Szerencsés eset, ha a kisebb reláció elfér a memóriában, ekkor ezt használjuk belső relációnak és a műveletigény a két reláció elemszámának összege.

Hatékonyabb belső ciklust készíthetünk, ha rendelkezünk indexszel a belső reláción.

Összefésüléses rendező összekapcsolás. Feltétele, hogy a relációk az összekapcsolás mező(k) szerint rendezettek legyenek. Ekkor könnyű kiválasztani az összekapcsolható sorokat.

Hasításos összekapcsolás. Az összekapcsolási mezőn értelmezett hasítófüggvény segítségével skatulyázzuk a két reláció rekordjait a memóriában elférő csoportokra, és ezeket kapcsoljuk össze.

Több tábla összekapcsolása

Több tábla összekapcsolását *bal-mély kiértékelési fa* szerint érdemes elvégezni, mivel ezzel könnyű a *csővezeték* (*futószalagosítás*) használata, csökkenthető a *materializálás*, és az összekapcsolási algoritmusok hatékonyan megvalósíthatók.

Az asszociativitás mentén az összekapcsolásokat érdemes olyan alakra is hozni, ahol a legkevesebb eredménysor keletkezik. Ezen kívül figyelembe vehető még a szükséges lemezhozzáférések, távoli elérések esetében a sávszélesség, stb.

Semi-join

Az adatátviteli költségek csökkentésére használt módszer: a távoli táblának csak az összekapcsolási attribútumra vonatkozó vetületét visszük át, elvégezzük az összekapcsolást, majd lekérjük az eredmény előállításához szükséges sorokat/átvisszük.

29.4.3. Egyebek

- lekérdezéstervezés irányítása (tippek),
- szabályalapú optimalizálás.