* 1. Osztott adatbázisok lekérdezése

A fő különbség az osztott és nem osztott adatbázison történő lekérdezés végrehajtásában az, hogy az adatbázis és a feldolgozó egységek több részre vannak osztva és az adatok mozgatása közöttük általában eléggé költséges.

**Fragmentáció – töredékekre bontás**

Legelőször annak módját kell tisztázni, hogy a relációkat miként lehet töredékekre feldarabolni és szétszórni az elosztott adatbázis egyes csomópontjaira. A mi szemszögünkből olyan logikai relációk vannak az adatbázisban, amelyek valójában nem léteznek, hanem töredékekből állnak össze, az unió és természetes összekapcsolás operátorok segítségéve. A lekérdezéseket és módosításokat a felhasználó ezeken a logikai relációkon hajtja végre. Vannak továbbá a fizikai relációk, amelyek a logikai relációk töredékeiként léteznek az adatbázisban.

Bármilyen reláció előállítható a töredékekből, amihez két művelet javasolt:

* Az reláció lehet töredékek természetes összekapcsolása: Ha az -re táblaként tekintünk, akkor minden -re tekinthetünk úgy, mint a tábla oszlopainak egy halmaza és nevezzük ezeket a tábla vertikális töredékeinek.
* Az reláció lehet töredékek uniója, . Ebben az esetben az reláció táblái sorokra vannak felosztva, így ezt nevezzük horizontális töredékeknek.

Például:

Vegyük egy bank adatbázisát, ahol az ügyfeleknek az egyes fiókokhoz tartozó adataik külön vannak tárolva. Ennek egyik előnye, hogy ha az ügyfél egy fiókon belül akar ügyet intézni, akkor elég csak az ott tárolt adatait belevenni a tranzakcióba. Ezzel időt spórolunk, mert kevesebb adatot kell mozgatni az alacsony átviteli sebességű összeköttetéseken.

Tegyük fel, hogy egy bank adatbázisában a relációk így néznek ki:

SZÁMLÁK(FIÓK, SZSZÁM, EGYENLEG)  
HITELEK(FIÓK, HSZÁM, ÖSSZEG)  
TULAJ(SZSZÁM, ÜNÉV)  
TARTOZIK(HSZÁM, ÜNÉV)  
ÜGYFELEK(ÜNÉV, CÍM)

Feltételezhetjük, hogy az és a táblák közötti reláció horizontálisan töredezett, minden fiókhoz van egy, így azok az a számlák, amik a töredékben vannak tárolva és a fiókban találhatók. Hasonlóképpen a reláció a horizontális töredékekből, amelyek a fiókban találhatók.

Az relációnak szintén horizontálisa töredezettnek kell lennie, esetleg lehetnek olyan sorok, amiben az ügyfél duplikálva szerepel, ha több fiókban is van számlája vagy hitelkerete. Így a címének ezek közül bármely ágban láthatónak kell lennie. Viszont a példa kedvéért,most feltesszük, hogy a reláció egyáltalán nem töredezett, csak megmarad mint fizikai reláció a fő ágazatban.

Ha olyan nézetet szeretnénk látni, ami kettő vagy több fizikai reláció összekapcsolása, akkor az vertikális dekompozíció. Például, ha egy olyan nézetet szeretnénk látni, ami a és az relációk természetes összekapcsolása, akkor a címek és a számlák úgy fognak megjelenni, mintha azok egy reláció részei lennének. Ekkor a és a a vertikális töredékei ennek a nézetnek. Ezek a vertikális töredékek el vannak osztva a számos csomópont között, bár feltettük, hogy a csak a fő fiókban van, amíg a el van osztva az fiókok között.

**Lekérdezések töredékek bevonásával**

Tegyük fel, hogy a felhasználók csak a logikai relációkat látják, és lekérdezéseket csak ezeken a logikai relációkon hajtanak végre. Ekkor a lekérdezés, logikai relációkból felépülő termként jelenik meg, és a logikai relációk pedig a relációk töredékeiből természetes összekapcsolás és unió operátorok használatával előálló termek. Így, olyan kifejezéseket állítunk elő, amelynek operandusai a fizikai relációk. A sorrendhez és a transzformációs fa operátorainak pontos implementációhoz alapos megfontolások szükségesek.

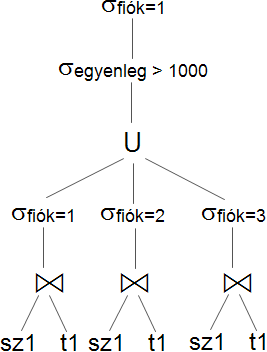
**Őrfeltételek**

A horizontális töredékkel kapcsolatban gyakran keresünk egy őrfeltételt, amelyet bármely a töredék bármely halmaza kielégít. Például a töredéknek a banki adatbázisban a az őrfeltétele. Minden logikai és fizikai relációnak van őrfeltétele, ami lehet akár a mindig-igaz feltétel is, ami egyenértékű azzal, ha nincs feltétel. Ekkor reláció teljesen eliminálható a kifejezésből, ha találunk olyan kiválasztást, amelyben ellentmond a őrfeltételnek.

Ha relációnak az őrfeltétele, akkor az bármely előfordulását a kifejezésben lecserélhetjük kifejezésre, anélkül, hogy az eredmény megváltozna.

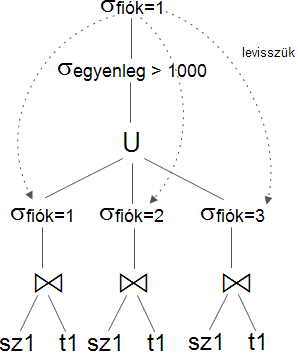
**Példa**

Tegyük fel, hogy a korábbi példában felírt banki adatbázisban három fiók van: 1, 2, 3. A szükséges lekérdezésnek pedig az összes olyan ügyfelet kell megjelenítetnie, akiknek több mint 1000 $ megtakarítása van az 1. fiókban. Ha a korábban felírt sémát használjuk, akkor azt nem lesz jó, mert nem tudjuk azt a tényt kifejezni, hogy 2. és 3. fiókban található töredékek nem tartalmaznak elemeket az 1. fiókban található számlákhoz. Ennek oka, hogy a az nem attribútuma a relációnak, így nem tudjuk kifejezni a szükséges feltételt őrfeltételek használatával. Így át kell tervezni a töredezési sémát, hogy ezt a feltételt is ki tudjuk fejezni. Így be kell vezetni egy további relációt:

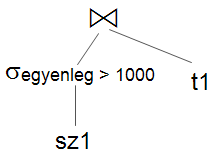


Az horizontálisan töredezett relációkra, és olyan -beli elemeket tartalmaznak, ahol . A továbbiakban legyen minden felosztva vertikálisan töredékekre, amely attribútumai és . A attribútumai és . Ezek a töredékek a és a relációkat helyettesítik.

Legyen ekkor reláción az őrfeltétel , így reláció helyett kifejezést használhatjuk, és ezen a vonalon elindulva és is kiváltható. A fizikai töredékek termjeiben kifejezéssel kapjuk meg -t és természetesen kifejezés vezet az relációhoz a lekérdezésben.



A felső lekérdezésben található szelekció levihető az unió mellett azokba a részfákba, amelyekre vonatkozik. A attribútum természetesen jelen van mindhárom unióból álló termben, így a kiválasztás levihető az unió csomópont gyermek csúcsaiba. A második és a harmadik részfára a és a feltételek teljesülnek, ami logika ellentmondáshoz vezet az eredeti feltétellel kombinálva: és .Ennek alapján a második és a harmadik részfát el is távolíthatjuk, így az unió művelet csak egy részfára fog vonatkozni, tehát az unió művelet is eliminálható. Ha a felső szelekciót levisszük az első részfába, akkor az meg fog egyezni azzal a feltétellel, ami a részfát meghatározza így ez a feltétel is törölhető. Az következő feltétel csak -re tolható le, mivel a -nek nincs attribútuma. Végül a fa így fog kinézni:



**Logikai relációk módosítása**

Ahhoz, hogy egy sort be tudjunk szúrni egy logikai relációba, ahhoz töredékeket kell beszúrni -be. Ehhez egy rekurzív algoritmust kell definiálni, mivel a beszúrása egy vagy több töredék beszúrását is eredményezheti. Ha ezek a töredékek fizikai relációk, akkor a beszúrás egyértelmű, logikai relációk esetén viszont rekurziót kell használni.

**Az algoritmus:**

Egy sor törlése relációból még több problémát vet fel. Ha horizontális töredékekből áll, akkor simán töröljük -t azokból a töredékekből, amelyekben megtalálható. Azonban, ha vertikálisan töredezett, akkor nem tudjuk, mit kell tenni. Mert ha simán töröljük -t az -ből mindenesetén, akkor véletlenül törölhetjük -ből az összes olyan sort is, amire a -= feltétel igaz valamely esetén.

A megoldás az, hogy minden sorhoz egy egyedi sor azonosítót rendelünk. Ennek hatására a relációs séma egy attribútummal bővül. Beszúrás esetén a rendszer generálja ezt az egyedi azonosítót, és egy olyan projekció eredménye kerül beszúrásra, ami már tartalmazza ezt a azonosítót is.

Tegyük fel, hogy törölni szeretnénk a sort -ből. Keressük meg minden relációhoz azt a sor halmazt, ami az -ben és a pedig a komponens és

Miután megkaptuk az halmazt minden esetében, egy természetes összekapcsolást kell végrehajtanunk:

Az eredmény 0 vagy több alakú sor lesz, ahol a sor azonosítója, pedig a sor, amit törölni akarunk. Most már tudjuk a azonosítóját (illetve azonosítóit, ha többször is szerepel), így most már biztonsággal tudjuk törölni az az relációból, minden ilyen megtalált -ra.

* 1. Aciklikus hipergráf:

Azon lekérdezésekre, amelyeknek a hipergráfja „aciklikus”, létezik számos egyszerűbb és hatékonyabb optimalizáló algoritmus.

***Aciklikus***: (itt a legegyszerűbb esetet használjuk) annak az ötletnek a megvalósítása, hogy egy fában (ami az egyetlen típusa az általános gráfnak, ami aciklikus) el tudunk távolítani leveleket egyesével, amíg nem marad egy sem.

* 1. GYO-redukció:

***Definíció***: Legyen E és F hiperélek. E fül (F miatt), ha E-F élei nem szerepelnek más hiperélben E-n kívül.

E fül eltávolításának problémáját nevezzük füllevágásnak.

Speciális eset: Ha egy hiperél nem metszik más hiperélt, akkor ez a hiperél egy fül és eltávolítható a hipergráfból.

***GYO-redukció***: Egy füllevágás sorozat egy hipergráfon, amíg nem marad egy hiperél sem.

***Definíció***: Egy hipergráf aciklikus, ha a GYO-redukálás eredménye az üres hipergráf, egyébként ciklikus.

***Tétel***: A füllevágások különböző sorrendje ugyanahhoz az irreducibilis (felbonthatatlan) hipergráfhoz vezetnek, vagyis a GYO-redukció egy hipergráfon egyedi.

***Bizonyítás***:

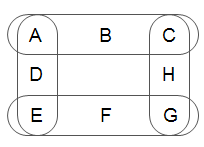
*Megjegyzés*: ha egy potenciális eltávolítás akkor is elérhető marad, ha egy másik eltávolítást választunk.

Tfh.: E1 fül E2-re nézve. Ekkor ha elvégezzük egy hiperél füllevágását (úgy, hogy ez nem az E2), akkor E1 még mindig eltávolítható marad.

Nehéz eset: E2 fül E3-ra és E2-t távolítjuk el. Kell: E1-E3 részhalmaza E1-E2, vagyis E1 továbbra is fül (E3-ra nézve), és eltávolítható.

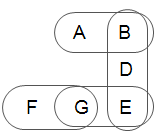
Indirekten lássuk be a fentit:

Ekkor kell lennie egy N élnek az E1-E3-ban, ami nem található meg E1-E2-ben. Tehát N benne van E1-ben és E2-ben, de nincs benne E3-ban. Innen következik, hogy E2-E3-ban benne van N, ezért N nem egyedi E2-re, ami ellent mond annak a feltevésnek, hogy E2 él és ezért eltávolítható. //Bizonyítva//

*Példa 1*:

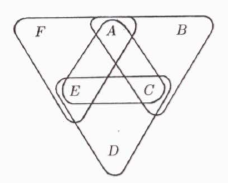
Ebben az esetben a hipergáf ciklikus, ugyanis nincs benne fül.

*Példa 2*:

Itt a következőeket állapíthatjuk meg:

* {F, G} fül GE-re //a halmazos jelölésből elhagyható a kapcsos zárójel és az elválasztó szimbólum// Ekkor FG hiperél eltávolítható. ( {F, G} – {G, E} = {F} tehát F egyedi az FG hiperélre )
* GE fül BDE-re, ezért GE eltávolítható
* AB fül BDE-re, ezért AB eltávolítható
* BDE fül, hiszen egyedül van és nincs más hiperél, ezért ő is eltávolítható.

Ez a hipergráf aciklikus.

*Példa 3*:

(Fül-re vonatkozó) Definíció alapján az ABC, CDE és AEF hiperélek egy ciklikus hipergráfot adnának ki. Viszont a fenti ábrán ACE is egy hiperél, ezért a következőek teljesülnek:

* ABC fül ACE miatt
* CDE fül ACE miatt
* AEF fül ACE miatt

Ha egyesével eltávolítjuk ezeket, akkor már csak ACE hiperél marad, ami fül, ezért eltávolítható, tehát a hipergráf aciklikus.

Ez a példa felvet egy tényt, amire figyelni kell: Aciklikus hipergráfnak lehet ciklikus részgráfja.

* 1. Félig összekapcsolás

**Átviteli költség csökkentése félig-összekapcsolásokkal**

A Wong-Yussefi algoritmus félig-összekapcsolásokkal gyakran a relációk méretet csökkenti. Viszont az egyprocesszoros rendszerekben, amiben ez az algoritmus tervezve lett, általában nem javít vagy ront a hatékonyságon olyan egyszerű esetekben, mint például amikor egy hipergráf csak két hiperélből áll, -ből és -ből. Ilyenkor egyszerűbb kiszámolni a kifejezést és projektálni azt egy kitüntetett csomópontra.

Viszont az elosztott környezetekben, ahol az átviteli költséget minimalizálni kell, gyakran sokkal hatékonyabb félig összekapcsolásokat csinálni. Így a eredményét vagy módon kell meghatározni. Feltesszük, hogy az és olyan relációk vagy töredékek, amelyek nem azonos csomóponton találhatók, és illetve sorokból állnak. Az átvitel költsége ekkor , ahol fix költség minden átvitelre, és pedig a további költség n sor átvitelére.

**Egyenes módszer:** Amennyiben átvisszük a relációt az -t is tartalmazó csomópontra, és ott hajtjuk végre az összekapcsolást a költség lesz. Amennyiben viszont a teljes S relációt visszük át az R relációt tartalmazó csomópontra, úgy a költség lesz. Tehát elmondhat, hogy az átviteli költség az összekapcsolás végrehajtásához a kisebb költségű átvitelt választva lesz.

**Félig-összekapcsolásos módszer:** Másik megközelítésben az relációból vesszük az attribútumok szerinti projekció eredményét, és csak azt szállítjuk át az relációt tartalmazó csomópontra. Majd ott természetes összekapcsolás segítségével kiszűrjük relációból a fityegő sorokat, azaz amelyek nem kapcsolható egy sorhoz sem az relációból. Így előáll az reláció oldalán a eredménye, amit visszaszállíthatunk az oldalára, és a join végrehajtható, mivel .

A végrehajtása 5 lépésben:

1. kiszámítása az oldalán
2. átszállítása az oldalára
3. kiszámítása az oldalán, a összefüggés felhasználásával
4. átszállítása oldalára
5. kiszámítása oldalán, a összefüggés felhasználásával

Az és természetesen felcserélhető.

Legyen a sorok száma -ben, pedig a sorok száma -ben. Legyen a a sorok száma -ben, és pedig a sorok száma -ben. A második lépés költsége , és a 4. lépés költsége pedig . Így ha figyelembe vesszük, hogy az és felcserélhető, akkor a minimális átviteli költség félig-összekapcsolásos stratégia esetén lesz. Így a félig-összekapcsolásos stratégia akkor hatékonyabb, mint az egyenes módszer, ha igaz az alábbi egyenlőtlenség:

Ami miatt ez igaz lesz:

* A fix költség nem nagy, még akkor sem ha kevés rekord átviteli költségéhez hasonlítjuk
* és nagyjából azonos méretűek
* A és lényegesen kisebb mint a , és ez ugyanúgy igaz -re is. Tehát minden több olyan sora van, amelyeknek a főbb attribútumai megegyeznek R-ben és S-ben és számos fityegő sor keletkezik.

Tehát, vannak esetek, amikor a félig-összekapcsolásos megközelítés két reláció összekapcsolásában előnyökkel jár és vannak helyzetek, amikor nem. Viszont, ha több táblát kell összekapcsolni, akkor a fityegő sorok előfordulása megnőhet. Ennek oka, hogy a sorához a relációnak sorokat kell találni a többi relációból, oly módon, hogy a főbb attribútumaik megegyeznek. Ekkor, ha a sor szerepel az eredményben, akkor a nem fityegő sor ebben az összekapcsolásban. Így minél több reláció van egy összekapcsolásban, annál több eliminálható fityegő sor fordulhat elő.

* 1. Félig-összekapcsolásos program

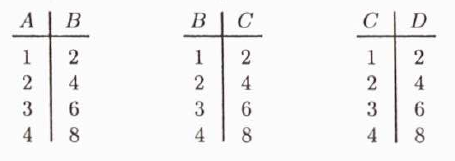
**Félig-összekapcsolásos program**

Legyen adott egy relációk kollekciója, amely elemeinek természetes összekapcsolását szeretnénk meghatározni. Tanulmányozzuk annak hatását, ha a félig-összekapcsolásokat különböző sorrendben hajtjuk végre a kollekción. Írjuk fel a félig-összekapcsolás lépését, mint .

A cél nem az eredeti reláció módosítása, hanem olyan ideiglenes relációként való kezelése, amibe az új értéket elrakhatjuk. A félig-összekapcsolásos program a félig-összekapcsolás lépés egy sorozata.

**Példa**

Vegyük az ábrán látható relációkat és határozzuk meg az összekapcsolás eredményét. Ha a három reláció mindegyike más-más helyen található, akkor valamilyen félig-összekapcsolásos programot kellene futtatni, hogy csökkentsük a sorok számát, mielőtt a két relációt átvisszük a harmadik relációt tartalmazó csomópontra.



A program egy lehetséges futtatása:

Első lépésben eliminálódnak a (3,6) és (4,8) sorok az relációból és a második lépésben ugyanezek a relációból. A harmadik lépés eliminálja az (1,2) és (3,6) sorokat a relációból. Ha végrehajtjuk az összekapcsolást a három reláció között, akkor mindössze egy sort kapunk az összekapcsolás eredményeként, ami az (1,2,4,8). Tehát, a (2,4) fityegő sor lesz mind az mind pedig a összekapcsolásban, és fityegő sor marad a összekapcsolásban is. Így program nem egészen megfelelően működik, mivel néhány fityegő sort eliminált, de nem mindet.

* 1. Teljes redukáló:

Legyenek R1, … , Rk relációk, és ezeknek a természetes összekapcsolása ()

Ri-t redukáltnak hívjuk, ha nincsenek benne „lógó” sorok. ()

***Definíció:*** Egy félig-összekapcsolásos programot teljes redukálónak nevezünk, ha minden Ri redukált a program lefutása után, függetlenül a relációk kiindulási értékétől.

*Példa 1*:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** |  | **B** | **C** |  | **C** | **D** |
| 1 | 2 |  | 1 | 2 |  | 1 | 2 |
| 2 | 4 |  | 2 | 4 |  | 2 | 4 |
| 3 | 6 |  | 3 | 6 |  | 3 | 6 |
| 4 | 8 |  | 4 | 7 |  | 4 | 8 |

A fenti relációkra egy teljes redukáló a következő lesz:

*Példa 2*: Nincs mindig teljes redukáló

Legyen egy a fentihez hasonló táblázat adott, ahol az értékek rendre ai, bi, ci, di-k legyenek (i=1..n).

//Vagyis az első tábla első sora ai-t és bi-t tartalmazza, n-ik sora pedig an-t és bn-t, a második tábla hasonlóan, csak b1 és c1, valamint bn és cn, stb...//

Ezek az értékek legyenek mind különbözőek.

Ekkor az lesz. Vagyis minden sor fityegő (lógó), ezért egy félig-összekapcsolásos programmal el kellene távolítani.

Legyen ez a program a következő utasítások egymás utáni ismétlése:

// kiesik (a1, b1)

// kiesik (b1, c1)

// kiesik (a2, c1)

Ezt a három lépést ismételgetjük.

Indukcióval belátható, hogy i-ik lépés után csak olyan sorok törlődhettek, amelyben az egyik tag indexe <= i vagy n-i+1-nél nagyobb vagy egyenlő (az indexe).

Legyen egy félig-összekapcsolásos program, aminek k lépése van, valamint n = 2k+2

Ekkor k lépés után:

(ak+1, bk+1) nem törlődik AB-ből,

(bk+1, ck+1) nem törlődik BC-ből,

(ak+2, ck+1) nem törlődik AC-ből

*Következmény*: Nem létezik teljes redukáló, mert lépésszáma függ a konkrét értékektől.

*Példa 3*: Hasonlóan az előzőhöz, csak AC-ben (an+1, cn) helyett (a1, cn) legyen

Ekkor

Viszont a félig-összekapcsolások nem törölnek egy sort sem.

***Tétel***: Az -hoz létezik teljes redukáló pontosan akkor, ha hipergráfja aciklikus.

***Bizonyítás***: Két irányt kell belátni.

:

Az előző példát kell csak átalakítani a belátásához.

:

k = 1 esetén: Üres program teljes redukáló.

k > 1 és aciklikus G esetén: Létezik S fül T-re.

Ekkor hagyjuk el S-T és S csúcsokat a G-ből. Az így kapott K hipergráf aciklikus, ezért létezik teljes redukálója.

|

| |

| K teljes reduálója | G teljes redukálója, mivel nem marad benne fityegő sor.

| |

|

Ha s  S (a program után) esetén s-hez kapcsolható t  T. Indukció miatt t-hez K minden relációjából kapcsolható sor, ezért s nem fityegő. (Ha még máshoz is kapcsolni kell, akkor az csak T attribútumon keresztül történhet.)

Elég: K-ban nem fityeg  G-ben sem fityeg.

miatt minden t  T sor S-beli s-hez és indukció miatt K többi relációjánál valamelyik sorhoz kapcsolható.

K bármely relációjának u sorához van a többi relációban hozzá kapcsolható sor, ezek között T-beli sor is szerepel, amihez S-beli sor is kapcsolható, így u nem fityegő sor. //Bizonyítva//