**Adatbázisrendszerek elméleti alapjai**

*Egyetemi jegyzet*

2012

**ELTE Információs Rendszerek Tanszék**

Készítette Kiss Attila előadásai alapján

Csubák Dániel, Varga Péter, Vörös Péter, Barabás Gábor, Nagy Dávid, Nemes Tamás, Bodnár István, Fodor Krisztián, Gyimesi Gábor, Torma Balázs, Müller Dávid, Tornóczky Zoltán, Szücs Ádám, Verő Anita, Morvai Mihály, Fokin Miklós, Hodosy Gábor, Tóth Tamás

**Tartalomjegyzék**

[Bevezetés 3](#_Toc342491094)

[1. Relációs algebra és SQL 4](#_Toc342491095)

[Alfejezet címe **Hiba! A könyvjelző nem létezik.**](#_Toc342491096)

[Irodalomjegyzék **Hiba! A könyvjelző nem létezik.**](#_Toc342491113)

# Bevezetés

A jegyzet a programtervező informatikus mesterszak Adatbázisrendszerek elméleti alapjai kurzushoz készült Kiss Attila előadásai és segédanyagjai alapján.

# Relációs algebra és SQL

A Relációs algebra és SQL fejezet négy részre oszlik. Ezen fejezeten belül az anyag foglalkozik az **Adatbázis-kezelő rendszerek** általános jellemzőivel, **A relációs adatmodellel**, a relációs algebra műveleteivel illetve annak használatával. Ide taroznak **az SQL nyelv egyes részei** (ORACLE specifikusan), bővebben a DDL, DML QL, triggerek, jogosultságok, PL/SQL, függvények, procedúrák, kurzorok. Ezen kívül ez a fejezet foglalkozik az **Adatmodellezéssel** ezen belül is az egyed-kapcsolat modellel illetve ennek átalakításával.

## Adatbázis-kezelő rendszerek

**Irodalomjegyzék**

Jeffrey D. Ullman, Jennifer Widom: Adatbázisrendszerek Alapvetés

Hector Garcia, H. G. Molina, Jeffrey D. Ullman, Jennifer Widom: Adatbázisrendszerek megvalósítása

Stolnicki Gyula: SQL programozóknak

Kende Mária, Kotsis Domokos, Nagy István: Adatbázis-kezelés az Oracle rendszerben

Iványi Antal: Informatikai algoritmusok

### Edgar Frank Codd 12 szabálya

Edgar Frank Codd a relációs adatmodellezés úttörője által megalkotott tizenkét szabály, amelyek teljesítése elvárt egy adatbázisrendszertől, hogy racionálisnak nevezhessük.

1. Az egységes megjelenésű információ szabálya
   * Az adatbázisban szereplő összes információt egy, és csak egy megadott formában (adatmodellben) lehet ábrázolni, nevezetesen táblázatok sorainak oszlopértékeiben.
2. Garantált lokalizálhatóság szabálya
   * Az adatbázisban minden egyes skaláris értékre logikailag úgy kell hivatkozni, hogy megadjuk az azt tartalmazó táblázat és az oszlop nevét, valamint a megfelelő sor elsődleges kulcsának az értékét.
3. A NULL értékek egységes kezelése
   * Az adatbázis-kezelő rendszernek (DBMS) olyan egységes módszerrel kell támogatnia a hiányzó vagy nem ismert információ kezelését, amely eltér az összes „rendes” érték kezelésétől, továbbá független az adattípustól.
4. A relációs modell alapján aktív online katalógust kell üzemben tartani
   * A rendszernek támogatnia kell egy online, beépített katalógust, amelyet a feljogosított felhasználók a lekérdező nyelv segítségével ugyanúgy le tudnak kérdezni, mint a közönséges táblákat.
5. A teljes körű „adatnyelv” szabálya
   * A rendszernek legalább egy olyan relációs nyelvet kell támogatnia, amelynek
     1. lineáris a szintaxisa,
     2. interaktívan és az alkalmazásokhoz készített programokon belül is lehet használni,
     3. támogatja az adatdefiniáló műveleteket, a visszakereső és adatmódosító (manipulációs) műveleteket, biztonsági és jósági (integritási) korlátokat, valamint a tranzakció-kezelési műveleteket (begin, commit, rollback: elkezdés, jóváhagyás és visszagörgetés).
6. A nézetek frissítésének szabálya
   * A rendszernek képesnek kell lennie az adatok összes nézetének frissítésére.
7. Magas szintű beszúrás, frissítés és törlés
   * A rendszernek támogatnia kell az INSERT, UPDATE, és DELETE (új adat, módosítás, törlés) operátorok halmaz szintű, egyidejű működését.
8. Fizikai szintű adatfüggetlenség
   * A fizikai adatfüggetlenség akkor áll fenn, ha az alkalmazások (programok) és a felhasználók adatelérési módja független az adatok tényleges (fizikai) tárolási és elérési módjától.
9. Logikai szintű adatfüggetlenség
   * Logikai adatfüggetlenség akkor áll fenn, ha az adatbázis logikai szerkezetének bővítése nem igényli az adatbázist használó alkalmazások (programok) megváltoztatását.
10. Jóság (integritás) függetlenség
    * Az adatok jóságának (érvényességének) korlátait az adatfeldolgozási programoktól függetlenül kell tudni meghatározni, és azokat katalógusban kell nyilvántartani. Legyen lehetséges a szóban forgó korlátokat megváltoztatni, anélkül hogy a meglévő alkalmazásokon változtatni kelljen.
11. Elosztástól való függetlenség
    * A meglévő alkalmazások működése zavartalan kell, hogy maradjon
      1. amikor sor kerül az adatbázis-kezelő osztott változatának bevezetésére
      2. amikor a meglévő osztott adatokat a rendszer újra szétosztja.
12. Megkerülhetetlenség szabálya
    * Ha a rendszernek van egy alacsony szintű (egyszerre egy rekordot érintő) interfésze, akkor ezt az interfészt ne lehessen a rendszer megkerülésére használni, például a relációs biztonsági vagy jósági (integritás védelmi) korlátok megsértésével.

### Adatbázisrendszerek

Az Adatbázis kezelés feladata: A háttértárolón tárolt, nagy adatmennyiség hatékony kezelése, lekérdezése, módosítása.

#### Adatmodell támogatása

Az adatbázis rendszereknek támogatniuk kell bizonyos adatmodelleket. Az adatmodell a valóság folyamatainak, tevékenységeinek magasabb szintű ábrázolása.

Ilyen adatmodell a *hálós, hierarchikus adatmodell,* amely apa-fiú kapcsolatok gráfja, amelyben ez által a keresés hatékony művelet lesz. Egy másik gyakran használt adatmodell a *relációs adatmodell,* amelynek a megvalósítása táblák rendszere, könnyen megfogalmazható műveletekkel, erre épül az *Objektum-orientált adatmodell.* Ezen kívül léteznek még *félig strukturált adatmodellek*, ilyen például az XML, illetve *logikai adatmodellek,* amelyek tényekre illetve következtetési szabályokra épülnek és szakértői rendszerekben használatosak.

#### Adatbázis-kezelő nyelvek

Egy adatbázis kezelő rendszernek támogatnia kell bizonyos adatbázis-kezelő nyelveket is, hogy a benne tárolt adatok használhatóak legyenek. Ilyen nyelv a DDL (Data Definition Language) azaz az adatdefiniáló nyelv, amely sémák illetve adatstruktúrák megadására használható. A DML (Data Manipulation Language) tehát az adatkezelő nyelv, amellyel beszúrni,törölni illetve módosítani tudjuk a rendszer által tárolt adatokat. A QL lesz a rendszer lekérdező nyelve, amelynek típusai lehetnek deklaratív (SQL, kalkulusok) illetve procedurális (relációs algebra). Ezen kívül létezik még a PL/SQL, amely programozási szerkezetek ötvözése SQL-lel. Illetve az előfordítóval programozási nyelvbe ágyazás és a 4GL (amelyekkel alkalmazásokat generálhatunk) nyelvek használata is.

#### Több felhasználó támogatása

Mivel az adatbázis rendszereket általában többen is használják, ezért mindenképp támogatnia kell ennek megfelelő kezelését, például felhasználói csoportok létrehozásával. Az adatbázisrendszerbeli legfelsőbb hatalommal az adatbázis-rendszergazda (DBA) rendelkezik. A megfelelő szabályozás érdekében minden egyes műveletet jogosultságok vezérelnek. Ezen műveletek lehetnek például: objektumok olvasása, írása, módosítása, készítése, törlése, jogok továbbadása, jogok visszavonása. Ezen jogosultságok tárolása rendszertáblákban történik.

#### Tranzakció-kezelés

Az adatintegritás biztosításának érdekében az adatbázis kezelő a rajta végrehajtott műveleteket tranzakcióként kezeli. Egy tranzakció adatkezelő műveletekből (DML) álló sorozat, a rendszer célja, hogy az adatintegritás fenntartása mellett a tranzakciókat párhuzamosak tudja végrehajtani.

Read S

S≔S + 1

T1

Idő

A tranzakció-kezelő biztosítja az adatbázisrendszer ACID tulajdonságát. Az ***A****tomicity* (Atomosság) azt jelenti, hogy minden tranzakció vagy egységesen lefut vagy nem. A ***C****onsitency* (Következetesség) azaz, hogy az adatbázis konzisztens a tranzakció futása után. ***I****solation* (Elkülönítés) miszerint a párhuzamos végrehajtás eredménye egymás utáni végrehajtás eredményével kell, hogy megegyezzen. A ***D****urability* (Tartósság) pedig, hogy a befejezett tranzakciók eredménye rendszerhiba esetén sem veszhet el.

T2

Write S

S≔S + 1

Read S

A tranzakció-kezelés megoldásához szükség van zárolásokra is ezeket a **Lock, Unlock** műveletek használatával lehet elérni.

T1: (Lock S, Read S, S:=S+1, Write S, Unlock S)

T2: (Lock S, Read S, S:=S-1, Write S, Unlock S)

0Minden zár kiadása előtt meg kell várni, amíg a zárat előzőlegesen feloldották. Csak a Lock, Unlock műveletek használata csökkenti a párhuzamosíthatóságot, de zárakat finomsága (zárolt adategység nagysága, zárolás típusa) javíthat ezen a csökkenésen.

A zárolások használata közben kialakulhatnak holtpontok, amikor két tranzakció vár egymásra.

T1 vár T2-re B miatt

T2

T1

D≔A + B

Read A

T2

Lock A

Read B

Lock B

C≔A + B

Read B

Lock B

Read A

T1

Lock A

T2 vár T1-re A miatt

Az adatbázisrendszerek kétfázisú protokollt használnak a zárolások kezelésére, amely minden tranzakció elején zárolja a szükséges adatokat és annak végeztével feloldja a zárakat. Fontos része a tranzakció-kezelésnek a tranzakciók érvényesítése, a naplózás, illetve a Commit ,Rollback, Checkpoint lehetőségek használata is. Az ütemező felel a tranzakciók végrehajtásának sorrendjéért felelős. Minden tranzakció sorozatnak ki kell elégítenie a szérializálhatóság feltételét, miszerint az ütemezés ekvivalens a tranzakciók egymás utáni végrehajtásával. A tranzakciók állapotát és az elvégzett műveleteket rendszertáblákban tárolja a rendszer.

#### Helyreállíthatóság

Minden rendszernél előfordulhatnak hibák, legyenek azok szoftveresek vagy hardveresek az adatbázis-kezelő rendszernek képesnek kell lennie az utolsó konzisztens állapot visszaállítására. Ennek a megoldására a rendszeres mentések használhatóak természetesen más stratégiával statikus (ritkán módosított) illetve dinamikus (gyakran módosított) adatbázisok esetén. A helyreállíthatóság összefügg a tranzakció kezeléssel, a tranzakciók állapotát naplófájlokba menti a rendszer, amiből később visszaállíthatóak.

#### Kliens-szerver felépítés

Az adatbázisrendszerek általában kliens-szerver architektúrára épülnek, de persze léteznek más architektúrák ahol egy köztes réteg található a kliens és a szerver között. Adatbázis szerverek általában olyan gyors gépek, amelyek nagy tárhellyel rendelkeznek, feladatuk pedig az adatbázis műveletek optimalizált, párhuzamos végrehajtása. Ezzel szemben az kliensek feladata, az adatbázis műveletek megfogalmazása, elküldése, az eredmények fogadása majd azok megjelenítése.

#### Adatvédelem, adatbiztonság

Mivel az adatbázisokban sokszor érzékeny adatokat is tárolunk, nagyon fontos, hogy ezek az adatok megfelelően biztosítva legyenek. A rendszer feladata a jogosultságok kezelése, amibe a felhasználók és jelszavaik adminisztrálása illetve az hozzáférési jogok kezelése tartozik. Az adatbázissémák korlátozása virtuális nézettáblák segítségével történik. A tárolt adatok illetve a hálózati adatforgalom titkosítása is feladata a rendszernek, ez megoldható például nagy prímszámok, RSA, DES felhasználásával.

### Adatbázis-kezelők felépítése

#### Lekérdezés-feldolgozó

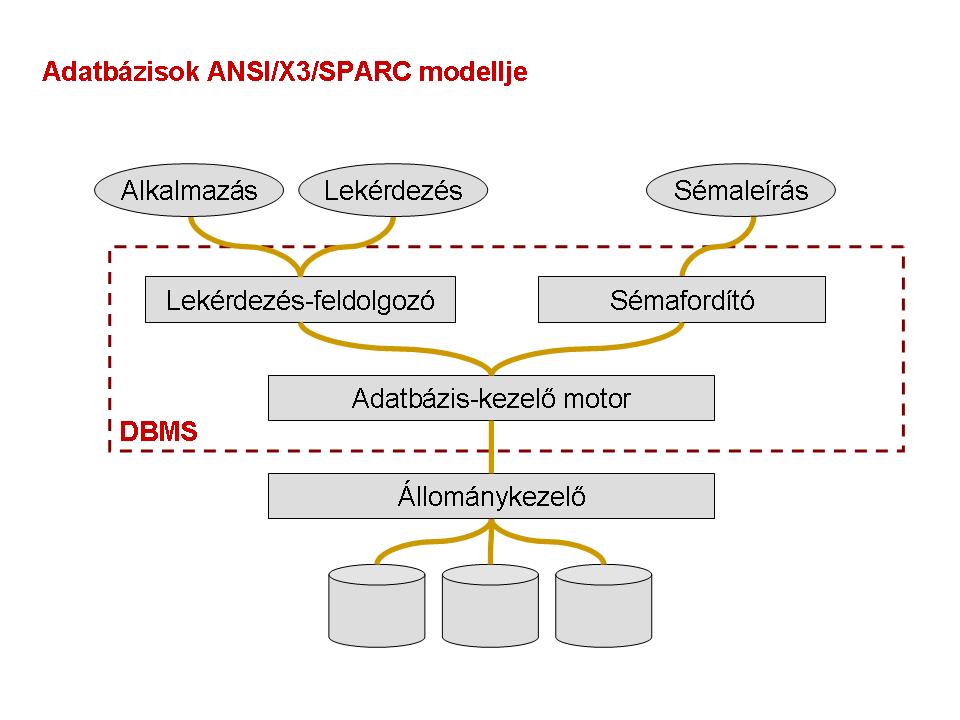
A lekérdezés-feldolgozó feladata, hogy szintaktikusan ellenőrizze a lekérdezéseket, az adatbázis-objektumok létezését illetve a hozzáférési jogokat. Ezután a lekérdezéseket optimálisra kell átfogalmaznia. Az optimalizálást végrehajtási tervek készítésével, aztán a végrehajtási tervek és statisztikai adatok (adatstruktúrák, méretek) alapján a várhatóan minimális költségű végrehajtási terv kiválasztásával éri el. Ezután már csak az optimális végrehajtási tervet kell végrehajtani.

#### Tranzakció-kezelő

A tranzakciók párhuzamos végrehajtásának biztosítása, az rendszer ACID tulajdonságának megtartása a feladata.

#### Tárkezelő

Feladata, hogy a fizikai adatstruktúrákat, indexeket, táblákat, puffereket kezelje



### Adatbázisok különböző szintjei

Az adatbázis kezelő rendszerek szintjei erősen elkülönülnek egymástól, ezt a sémák (tervek, leírások) és előfordulások (konkrét adatok, megvalósulások) leírásával lehet szemléltetni a fizikai, logikai és alkalmazói rétegen:

|  |  |  |
| --- | --- | --- |
|  | **Séma** | **Egy előfordulás** |
| **Alkalmazások** | Select sum(fiz) as összfiz from Bér; | 30 |
| **Logikai adatbázis** | Bér(név, fiz) | **név fiz**  Kiss 10  Nagy 20 |
| **Fizikai adatbázis** | szekvenciális | (Bér,név,fiz,#2,Kiss,10,Nagy,20) |

Fontos, hogy az adatbázis rendszerek szintjei a megvalósításban is elkülönüljenek, az adatok fizikailag függetlenek legyenek. Ez azt jelenti, hogy amennyiben a fizikai adatbázis szintje megváltozik (indexek készülnek, az adatok más struktúrában tárolódnak) akkor a felette lévő szinteken ez ne látszódjon. A fizikai adatfüggetlenség biztosításával később könnyebben növelhető a hatékonyság a tárolási struktúrák jobb használatával.

Az adatok logikailag is függetlenek kell, legyenek, tehát a logikai adatbázis szintjén alkalmazott változások (új táblák, oszlopok) esetén a régi alkalmazások változtatás nélkül képesek legyenek futni.

## Relációs algebra

### Relációs adatmodell

#### Relációséma

A relációséma egy olyan R(A1, A2, A3… An) konstrukció, ahol R a reláció neve , Ai-k pedigaz attribútum/tulajdonság/oszlop nevek. Dom(Ai) egy attribútum lehetséges értékeinek halmaza, az attribútumok pedig különbözőek egy sémán belül.

#### Reláció-előfordulás

r az R relációséma egy előfordulása, gyakorlatilag, egy tábla, reláció vagy sorhalmaz. r-ben minden sor egyszer szerepel és a sorok illetve oszlopok sorrendje lényegtelen.

r-re igaz,hogy:

t ∈ r esetén t egy sor (angolul: tuple azaz n-es). Ekkor t(Ai) vagy t($i) a t i-edik komponense, a t[Ai\_1,…,Ai\_n] a t sor i\_1..i\_n komponenseiből álló vektor. Amennyiben két séma azonos attribútummal rendelkezik akkor R.A prefixszel különböztetjük ezeket meg.

Egy t sort tekinthetünk függvénynek is:

Példa:

Bér

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **név** | **fiz** | **kor** | **név** | t1(név)=„Kiss”  t3($3)=22  t2(név,kor)=(„Nagy”,45)  t1(Bér.fiz)=10 |
| Kiss | 10 | 35 | Kiss | t1 |
| Nagy | 20 | 45 | Nagy | t2 |
| Kovács | 15 | 22 | Kovács | t3 |

### SQL lekérdezések felbontása: Relációs algebra

Az SQL nyelv alkalmas összetett, több táblás, alkérdéseket is tartalmazó lekérdések megfogalmazására is. Ezeknek az összetett lekérdezéseknek az áttekinthetőbbé tételének illetve optimalizálhatóságának érdekében felbontjuk őket egyszerű SQL utasításokra.

Először definiálni kell, hogy mi számít egyszerű SQL műveletnek. Legyenek ilyenek az egyszerű kiválasztásra épülő SQL lekérdezések, a többtáblás lekérdezések. Legyenek halmazműveleteink, legyünk képesek átnevezni táblákat, illetve lehessen felhasználni egy lekérdezés eredményét egy másik lekérdezésben (nézettáblák).

#### Egyesítés, unió

select \* from r union select \* from s;

Ekkor legyen r,s és r ⋃ s azonos sémájú, legyen r ⋃ s ≔ { t | t ∈ r vagy t ∈ s }, |r ⋃ s |≤ |r| + |s|, ahol |r| az r reláció sorainak száma, és ahol minden sor csak egyszer szerepelhet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 0 | 1 | | ⋃ | |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 1 | 0 | | = | |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 0 | 1 | | 1 | 0 | |

#### Kivonás, különbség

select \* from r minus select \* from s;

vagy

select \* from r where not exsits   
( select \* from s where r.A=s.A and r.B=s.B);

Ekkor legyen r,s és r - s azonos sémájú, legyen r - s ≔ { t | t ∈ r és t ∉ s }, |r - s |≤ |r|

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 0 | 1 | | - | |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 1 | 0 | | = | |  |  | | --- | --- | | **A** | **B** | | 0 | 1 | |

#### Szorzás, direktszorzat vagy Decartes-szorzat

select \* from r,s;

Ekkor r,s sémáiban nincs közös attribútum, legyen r × s sémája a sémák egyesítése és

r × s ≔ { t | t[R] ∈ r és t[S] ∈ s } , illetve | r × s| = |r| ∙ |s|

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 0 | 1 | | × | |  |  | | --- | --- | | **C** | **D** | | 0 | 0 | | 1 | 0 | | = | |  |  |  |  | | --- | --- | --- | --- | | A | B | C | D | | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 0 | |

#### Vetítés, projekció

select distinct A1,…,Ak from r;

Ekkor X ⊆ { A1,…,An} és ΠX(r) sémája X, ahol ΠX(r) ≔ { t | ∃ t’ ∈ r, t’[X] = t } és |ΠX(r)|≤ |r|

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | r: | |  |  |  |  | | --- | --- | --- | --- | | **A** | **B** | **C** | **D** | | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 0 | | | |  |  | | --- | --- | | select distinct B,D from r; | | | ΠBD(r)= | |  |  | | --- | --- | | **B** | **D** | | 0 | 0 | | 0 | 1 | | |
| |  |  | | --- | --- | | select distinct D,A from r; | | | ΠDA(r)= | |  |  | | --- | --- | | **D** | **A** | | 0 | 0 | | |

#### Kiválasztás, szűrés, szelekció

select \* from r where A = B;

select \* from r where A < B;

select \* from r where A > B;

select \* from r where A <> B;

select \* from r where A ≤ B;

select \* from r where A ≥ B;

select \* from r where A = konstans;

select \* from r where A < konstans;

select \* from r where A > konstans;

select \* from r where A <> konstans;

select \* from r where A ≤ konstans;

select \* from r where A ≥ konstans;

select \* from r where feltétel1 and feltétel2;

select \* from r where feltétel1 or feltétel2;

select \* from r where not (feltétel);

Ekkor σF(r) és r sémája megegyezik, ahol σF(r) ≔ { t | t∈ r F(t) = IGAZ }. Itt F lehet atomi feltétel azaz : Ai Θ Aj ahol Θ ∈ { =, ≠ , <, >, ≤, ≥ } vagy Ai Θ c ahol c konstans, ezen kívül atomi feltételekből zárójelekkel és logikai kapcsolókkal (⋀ ,⋁, ¬ )

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | r: | |  |  |  |  | | --- | --- | --- | --- | | **A** | **B** | **C** | **D** | | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | | | |  |  | | --- | --- | | select \* from r where A=B and not B < 1; | | | σ**A=C ∧ ¬ (B<1)** (r)= | |  |  |  |  | | --- | --- | --- | --- | | **A** | **B** | **C** | **D** | | 0 | 1 | 0 | 0 | | |

|σF(r)| ≤ |r| és a feltételben nem használhatóak függvények,

tehát F nem lehet például A + B < 5.

Az összetett feltételeket átírhatjuk elemi feltételeket használó kifejezésekké a következő azonosságok felhasználásával:

* σF1∧F2(r) ≅ σF1(σF2(r)) ≅ σF2(σF1(r))
* σF1∨F2(r) ≅ σF1(r) ∪ σF2(r)
* A De Morgan azonosság felhasználásával beljebb vihetjük a negációkat
  + ¬ (F1 ∧ F2) helyett (¬ F1) ∨ (¬ F2)
  + ¬ (F1 ∨ F2) helyett (¬ F1) ∧ (¬ F2)
  + elemi feltétel tagadása helyett a fordított összehasonlítást használjuk:
    - például ¬ (A < B) helyett (A >= B)
* Az elemi feltételek jók, mivel könnyen készíthetünk hozzájuk indexeket, illetve lekérdezést gyorsító adatszerkezeteket.

#### Átnevezés

select oszlop [AS]újnév, … from r [AS] újnév;

Átnevezéskor a relációnak és az attribútumainak új nevet adhatunk, amennyiben r sémája R(A1,…,An) akkor ρS(B1,…Bn)(r) sémája S(B1,…,Bn) és | ρS(B1,…Bn)(r)| = |r|

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | BÉR  r: | |  |  | | --- | --- | | **C** | **D** | | 0 | 0 | | 1 | 0 | | | |  |  | | --- | --- | | ρMUNKA(dolg,jöv)(r) | | | MUNKA | |  |  | | --- | --- | | **C** | **D** | | 0 | 0 | | 1 | 0 | |   select név dolg, fiz jöv from BÉR MUNKA |

#### Kifejezések kompozíciója

Amennyiben szeretnénk egyszerű SQL lekérdezésekből összetett lekérdezéseket felépíteni akkor nézettáblákat kell használnunk. Egy SQL lekérdezés FROM listájában elhelyezhetjük a nézettáblákat. Például:

create view T1 as select ... from ... where ... ;

create view T2 as select ... from ... where ... ;

...

create view Tk as select ... from ... where ... ;

create view S as select ... from T1,...,Tk where ... ;

#### Összefoglalás

Bonyolult relációs algebrai műveletek kifejezésekor használhatunk konstans relációkat, relációs változókat illetve a relációs algebra alapoperátorait véges sok alkalommal. A relációs algebra alapoperátorai SQL-ben kifejezve a következőek:

1. Egyesítés
   * select \* from r union select \* from s;
2. Különbség
   * select \* from r minus select \* from s;
3. Szorzat
   * select \* from r,s;
4. Vetítés
   * select distinct A1,...,Ak from r;
5. Kiválasztás
   * select \* from r where feltétel;
6. Átnevezés
   * select oszlop [AS] újnév,... from r [AS] újnév;

A relációs algebra gyakorlatilag kifejezések halmaza.

#### A relációs algebra kifejező ereje

Relációs algebrával ki tudjuk fejezni a legfontosabb lekérdezéseket, de sajnos nem mindent.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ÉL | |  |  | | --- | --- | | **honnan** | **hova** | | 1 | 2 | | 2 | 4 | | 2 | 3 | | 3 | 3 | | ÚT | |  |  | | --- | --- | | **honnan** | **hova** | | 1 | 2 | | 2 | 4 | | 2 | 3 | | 3 | 3 | | 1 | 3 | | 1 | 4 | |

Ahol az ÚT tábla az ÉL tábla tranzitív lezártja.

**Tétel:** Nem létezik olyan relációs algebrai kifejezés, amelyet *tetszőleges* ÉL táblára alkalmazva a neki megfelelő ÚT táblát eredményezi.

### Származtatott műveletek

#### Gyakran használt kifejezések helyett új műveleteket vezetünk be, ezek lesznek a származtatott műveletek.

#### Metszet

select \* from r intersect select \* from s;

r ⋂ s = { t | t ∈ r és t ∈ s }, a metszet relációs algebrában többféleképpen is kifejezhető.

r ⋂ s = r – (r – s) = s – (s – r ) = r ⋃ s – ( (r – s) ⋃ (s – r) )

#### Téta összekapcsolás

select \* from r,s where r.Ai összehasonlítás s.Bj;

Ekkor r,s sémáiban R(A1,…,An) és S(B1,…,Bn) nincs közös attribútum .

r ⋈ Ai Θ Bj s = σAi Θ Bj(r × s), Ai=Bj feltétel esetén egyen összekapcsolásnak hívjuk.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 0 | 1 | | ⋈B=C | |  |  | | --- | --- | | **C** | **D** | | 0 | 0 | | 1 | 0 | | = | |  |  |  |  | | --- | --- | --- | --- | | **A** | **B** | **C** | **D** | | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | |

#### Természetes összekapcsolás

select distinct R.A1,...,R.An,R.B1,...,R.Bk,S.C1,...,S.Cm from r,s where R.B1=S.B1 and R.B2=S.B2 and ... and R.Bk=S.Bk;

Ekkor r,s sémái az R(A1,…,An,B1,…,Bn) illetve S(B1,…,Bn,C1,…,Cn).

Illetve r ⋈ s = ρP(A1,…,An,B1,…,Bk,C1,…,Cm)∏A1,…,An,R.B1,…,R.Bk,C1,…,CmσR.B1=S.B1∧…∧R.Bk=S.Bk (r×s)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 2 | 1 | | 1 | 2 | | ⋈ | |  |  | | --- | --- | | **B** | **C** | | 0 | 0 | | 0 | 2 | | 1 | 3 | | 4 | 3 | | = | |  |  |  | | --- | --- | --- | | **A** | **B** | **C** | | 0 | 0 | 0 | | 0 | 0 | 2 | | 2 | 1 | 3 | |

#### Félig összekapcsolás

select distinct R.A1,...,R.An,R.B1,...,R.Bk from r,s where R.B1=S.B1 and R.B2=S.B2 and ... and R.Bk=S.Bk;

Ekkor r,s sémái az R(A1,…,An,B1,…,Bn) illetve S(B1,…,Bn,C1,…,Cn).

Illetve r ⋉ s = ρP(A1,…,An,B1,…,Bk,C1,…,Cm)∏A1,…,An,R.B1,…,R.Bk,C1,…,Cm (r⋈ s)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 2 | 1 | | 1 | 2 | | ⋉ | |  |  | | --- | --- | | **B** | **C** | | 0 | 0 | | 0 | 2 | | 1 | 3 | | 4 | 3 | | = | |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 2 | 1 | |

#### Külső összekapcsolás

select A.r.B,C from r outer join s on r.B = s.B;

A külső összekapcsolás nem relációs algebrai művelet mivel annak a modelljéből kilép.

Ekkor r,s sémái az R(A1,…,An,B1,…,Bn) illetve S(B1,…,Bn,C1,…,Cn).

A r  s = r ⋈ s reláció kiegészítése az r és s soraival, úgy, hogy a hiányzó helyekre NULL értéket írunk.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **A** | **B** | | 0 | 0 | | 2 | 1 | | 1 | 2 | |  | |  |  | | --- | --- | | **B** | **C** | | 0 | 0 | | 0 | 2 | | 1 | 3 | | 4 | 3 | | = | |  |  |  | | --- | --- | --- | | A | B | C | | 0 | 0 | 0 | | 0 | 0 | 2 | | 2 | 1 | 3 | | 1 | 2 | NULL | | NULL | 4 | 3 | |

#### Összekapcsolások azonosságai

A relációs algebrai összekapcsolások esetében van néhány megjegyzendő azonosság. Amennyiben r,s sémái megegyeznek akkor r ⋈ s = r ⋂ s, amennyiben a két sémában nincs közös attribútum akkor r ⋈ s = r × s.

Amennyiben r = ∅ akkor r × s = ∅ és r ⋈ s = ∅.

A külső összekapcsolás lehet bal (jobb) oldali, amennyiben csak r (s) sorait vesszük hozzá a természetes összekapcsoláshoz ekkor jelölése: r B s (r J s ), SQL ben:

select A,r.B,C from r left outer join s on r.B=s.B; vagy select A,r.B,C from r,s where r.B = s.B(+);

(select A,r.B,C from r right outer join s on r.B=s.B; vagy select A,r.B,C from r,s where r.B(+) = s.B;)

#### Osztás, hányados

A maradékos osztás analógiájára bevezethetjük a relációk közötti osztást. A maradékos osztás esetében 7 % 3 = 2, mivel 2 az a legnagyobb egész, amire 2 ∙ 3 ≤ 7, amennyiben relációkról van szó akkor a ≤ helyett tartalmazásról van szó,

Osztásnál r és s sémája az R(A1,…,An,B1,…,Bn) illetve S(B1,…,Bn), ekkor r % s sémája a R(A1,…,An) lesz. r % s a legnagyobb (legtöbb sort tartalmazó) reláció amelyre ( r % s) × s ⊆ r, ez a reláció kifejezhető relációs algebrában méghozzá a ∏A1,…,An(r) – ∏A1,…,An( ∏A1,…,An(r)×s – r) kifejezés segítségével.

Ezen kívül kifejezhetjük SQL segítségével a NOT EXISTS illetve a MINUS operátorok használatával:

MINUS:

r(a,b)÷s(b):

Πa(r)- Πa(Πa(r)×s-r)

create view rsz as

select distinct r.a,s.b from r, s

minus

select \* from r;

select distinct a from r

minus

select distinct a from rsz;

NOT EXISTS:

Πa(r)- Πa(Πa(r)×s-r)

select distinct r2.a from r r2  
 where not exists  
 (select \* from r r1, s s1  
 where r2.a=r1.a and  
 not exists  
 (select \* from r r3  
 where r3.a=r1.a  
 and s1.b=r3.b));

#### Monotonitás

Egy kifejezést monoton nem csökkenőnek (röviden monotonnak) nevezünk, amennyiben bővebb relációra alkalmazva az eredmény is bővebb, tehát ha Ri⊆ Si i=1..n akkor   
E(R1,…,Rn) ⊆ E(S1,…,Sn).

Az alapműveletek monotonak a kivonás kivételével, ennek ellenére egy kivonásokból álló kifejezés lehet monoton, például r ⋂ s = r – ( r – s) monoton. Amennyiben E,E1,Ek monoton kifejezések és E(E1(…),…,Ek(…)) helyes kifejezés, akkor monoton is. Ezen két állítás következménye: A kivonás nem fejezhető ki a többi alapművelettel.

## Lekérdezések optimalizálása

Szeretnénk, hogy a lekérdezéseink minél gyorsabban fussanak le, a lekérdezések gyorsításához felhasználhatunk táblákra vonatkozó paramétereket, indexeke, illetve általános érvényű tulajdonságokat, heurisztikákat.

Legyen egy példa: select B,D from R,S where R.A = ’c’ and S.E = 2 and R.C = S.C;

Ekkor a kiválasztás eredménye

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **A** | **B** | **C** | | a | 1 | 10 | | b | 1 | 20 | | c | 2 | 10 | | d | 2 | 45 | | e | 3 | 45 | | |  |  |  | | --- | --- | --- | | **C** | **D** | **E** | | 10 | x | 2 | | 20 | y | 2 | | 30 | z | 2 | | 40 | x | 1 | | 50 | y | 3 | |
| |  |  | | --- | --- | | **A** | **B** | | 2 | x | | |

A lekérdezés eredményének egyik kiszámítási módja, hogy fogjuk a két tábla szorzatát, kiválasztjuk a megfelelő sorokat és végrehajthatjuk a vetítést. Ez a direktszorzaton alapuló összekapcsolás (Oracleben a NESTED LOOP) ami nagyon költséges. Ez a terv relációs algebrában: ΠB,D [ σR.A='c'∧ S.E=2 ∧ R.C = S.C (RXS)] .   
Ugyanennek a feladatnak egy másik lehetséges kiszámítási módszere: ΠB,D(σR.A=’c’(R) ⋈ σS.E = 2(S)

Indexek felhasználásával tovább gyorsíthatjuk a lekérdezést. A feladat kiszámolható a következő lépésekkel:

1. Az R.A index alapján keressük meg az R azon sorait, amelyekre R.A = 'c'!
2. Minden megtalált R.C értékhez az S.C index alapján keressük meg az S-ből az ilyen értékű sorokat!
3. Válasszuk ki a kapott S-beli sorok közül azokat, amelyekre S.E = 2!
4. Kapcsoljuk össze az R és S így kapott sorait, és végül vetítsünk a B és D oszlopokra.

#### Algebrai optimalizáció

Amikor SQL kifejezéseket szeretnénk optimalizálni első lépésként felépítünk egy elemzőfát, amiből logikai lekérdező tervet, amelyet optimalizálunk az algebrai optimalizáció szabályainak alkalmazásával.

Az algebrai optimalizáció célja, hogy a relációs algebrai kifejezéseket minél gyorsabban ki tudjuk számolni. Ennek eléréséhez feltesszük, hogy a kiszámítás költsége arányos a relációs algebrai kifejezés részkifejezéseinek megfelelő relációk tárolási méretének összegével. Az optimalizáció felhasználja a műveleti tulajdonságokat, amelyek felhasználásával ekvivalens átalakításokat alkalmazva kisseb méretű relációkat készít. Ez az eljárás heurisztikus, azaz nem az argumentum relációk valódi méretével számol, és nem egyértelmű, mivel az átalakítások sorrendje nem determinisztikus ezért más sorrendben végrehajtva őket más (általában a kiindulásinál minden esetben jobb) relációkat kapunk. Az SQL optimalizáláshoz nem elég a relációs algebrai optimalizáció, lévén, hogy az bővebb annál, ezért ki kell terjeszteni a bővített relációs algebrára is a kifejezéseket.

A relációs algebrai kifejezések ábrázolására kifejezésfát alkalmazunk, amely egy gráf, ahol, amelynek a csúcsai amennyiben nem levelek a relációs algebrai műveletek, és amelyeknek egy vagy két gyereke van a műveletek argumentumainak számától függően. Azok a csúcsok, amelyek levelek vagy konstans relációk vagy relációs változók.

E1(r1,…,rk) és E2(r1,…,rk) ekvivalensek (jel.: E1 ≅ E2) amennyiben tetszőleges r1,…,rk esetén E1(r1,…,rk) = E2(r1,…,rk) . Az algebrai optimalizálás 11 szabálya olyan állítások, amelyek a kifejezések ekvivalenciáját fogalmazzák meg.

1. Kommutativitás
   * szorzás
   * természetes összekapcsolás,
   * téta összekapcsolás
2. Asszociativitás
   * szorzás
   * természetes összekapcsolás,
   * téta összekapcsolás
3. Vetítések összevonása, bővítése
   * Amennyiben A,B ⊂ E oszlopainak és A ⊆ B akkor ΠA(ΠBE) ≅ ΠA(E)
4. Kiválasztások felcserélhetőségem, felbontása
   * Amennyiben F és G az E oszlopain értelmezett kiválasztási feltétel akkor:   
     σF ⋁ G(E)≅ σF(σG(E) ≅ σG(σF(E)
5. Kiválasztás és vetítés felcserélhetősége
   * Legyen F az E relációnak az A ⋃ B oszlopain értelmezett kiválasztási feltétel ,   
     ahol A ⋂ B = ∅. Ekkor: ΠA(σF(E)) ≅ ΠA(σF(ΠA ⋃ B(E)))
6. Kiválasztás és szorzás felcserélhetősége
   * Legyen F1 az E1 reláció oszlopainak részhalmazán értelmezett kiválasztási feltétel, legyen F2 az E1 × E2 reláció oszlopainak egy részhalmazán értelmezett kiválasztási feltétel, hogy mindkét sémából legalább egy oszlop szerepel benne, legyen továbbá F = F1 ⋀ F2.  
     Ekkor: σF(E1×E2) ≅ σF2 (σF1(E1)×E2)
7. Kiválasztás és egyesítés felcserélhetősége
   * Legyen az E1 és E2 relációk sémája megegyező és F a közös sémán értelmezett kiválasztási feltétel. EkkorσF(E1∪E2) ≅ σF(E1)∪σF(E2)
8. Kiválasztás és kivonás felcserélhetősége
   * Legyen az E1 és E2 relációk sémája megegyező és F a közös sémán értelmezett kiválasztási feltétel. Ekkor σF(E1 - E2) ≅ σF(E1) - σF(E2)
9. Kiválasztás és természetes összekapcsolás felcserélhetősége.
   * Legyen F az E1 és E2 közös oszlopainak egy részhalmazán értelmezett kiválasztási feltétel. Ekkor σF(E1⋈ E2) ≅ σF(E1) ⋈ σF(E2)
10. Vetítés és szorzás felcserélhetősége
    * Legyen i=1,2 esetén Ai az Ei reláció oszlopainak halmaza, valamit legyen A = A1 ⋃ A2. EkkorΠA(E1×E2) ≅ ΠA1(E1)×ΠA2(E2)
11. Vetítés és egyesítés felcserélhetősége
    * Legyen E1 és E2 relációk sémája megegyező és legyen A a sémájában szereplő oszlopok egy részhalmaza. Ekkor ΠA(E1∪E2) ≅ ΠA(E1)∪ΠA(E2)

Az algebrai optimalizáció heurisztikus módszer, amely a következő elveken alapul. Minél hamarabb szeretnénk szelektálni, mivel a szelekciók után keletkező relációk kisebbek lesznek. A szorzás utáni kiválasztásokat próbáljuk természetes összekapcsolássá alakítani, mivel az hatékonyabban számolható. Amennyiben egymás utáni unáris műveleteket találunk, próbáljuk összevonni őket és amennyiben lehet kiválasztást, vetítést vagy kiválasztás utáni vetítést próbáljunk képezni, mivel így csökken a műveletek száma illetve általában a relációk mérete is. Keressünk közös kifejezéseket, amelyeket később újra felhasználhatunk.

#### Algebrai optimalizálás algoritmusa

INPUT: relációs algebrai kifejezés kifejezésfája

OUTPUT: optimalizált kifejezésfa optimalizált kiértékelése

Hajtsuk végre az alábbi lépéseket a megadott sorrendben:

1. A kiválasztásokat bontsuk fel a 4. szabály segítségével:
   * σF1∧...∧Fn(E) ≅ σF1(...(σFn(E)))
2. A kiválasztásokat a 4., 5., 6., 7., 8., 9. szabályok segítségével vigyük olyan mélyre a kifejezésfában, amilyen mélyre csak lehet.
3. A vetítéseket a 3., 5., 10., 11. szabályok segítségével vigyük olyan mélyre a kifejezésfában, amilyen mélyre csak lehet. Hagyjuk el a triviális vetítéseket, azaz az olyanokat, amelyek az argumentum reláció összes attribútumára vetítenek.
4. Ha egy relációs változóra vagy konstans relációra közvetlenül egymás után kiválasztásokat vagy vetítéseket alkalmazunk, akkor ezeket a 3., 4., 5. szabályok segítségével vonjuk össze egy kiválasztássá, vagy egy vetítéssé, vagy egy kiválasztás utáni vetítéssé, ha lehet (azaz egy Π.(σ.()) alakú kifejezéssé). Ezzel megkaptuk az optimalizált kifejezésfát.
5. A gráfot a bináris műveletek alapján bontsuk részgráfokra. Minden részgráf egy bináris műveletnek feleljen meg. A részgráf csúcsai legyenek: a bináris műveletnek (∪, ⎯, ×) megfelelő csúcs és a csúcs felett a következő bináris műveletig szereplő kiválasztások (σ) és vetítések (Π). Ha a bináris művelet szorzás (×), és a részgráf equi-joinnak felel meg, és a szorzás valamelyik ága nem tartalmaz bináris műveletet, akkor ezt az ágat is vegyük hozzá a részgráfhoz.
6. Az előző lépésben kapott részgráfok is fát képeznek. Az optimális kiértékeléshez ezt a fát értékeljük ki alulról felfelé haladva, tetszőleges sorrendben.