Párhuzamosítás adatbáziskezelő rendszerekben

Erős Levente, 2018-2022.

eros@db.bme.hu

Párhuzamosítás adatbázisokban

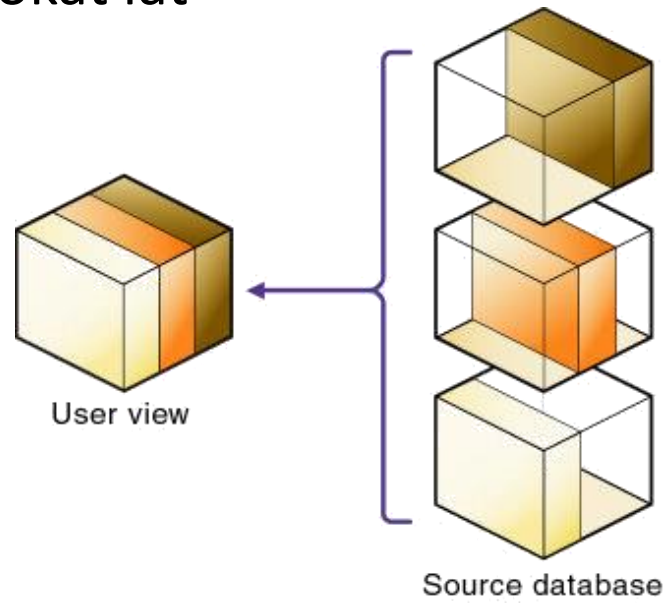
- Többprocesszoros környezet
- Párhuzamosítás szintjei
  - Adatok partícionáltan, szétszórva tárolódnak → I/O
  - Több lekérdezés szétszórva processzorokra → Lekérdezésközi (Interquery)
  - Lekérdezés műveletei szétszórva → Lekérdezésen belüli (Intraquery)
  - Művelet végrehajtása szétszórva → Műveleten belüli (Intraoperation)
- I/O – shared nothing architektúra
- Interquery, intraquery, intraoperation – bármely többproc. architektúra
- Cél (a tárgyban):
  - Relációs lekérdezések feldolgozása párhuzamosan, jobb válaszidőt elérve
  - A bemutatott módszerek egy része jól adaptálható más adatmodell esetén is.

Tartalom

- I/O párhuzamosítás
- Lekérdezések párhuzamosítása
  - Interquery
  - Intraquery
    - Interoperation
    - Intraoperation
- Intraoperation
  - Rendezés
    - 3 módszer
  - Join
    - 2 módszer - nem partícionált
    - 2 módszer - eleve prtícionált
    - Hálózatkihasználtság-optimalizálás
  - Szelekció
    - Ismétlődésszűrés

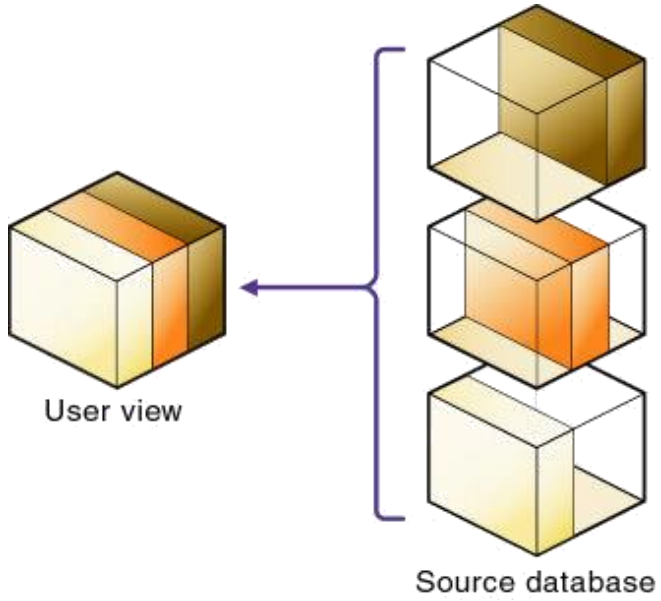
I/O párhuzamosítás

- Partícionáltan tárolt adathalmaz
  - Felhasználó számára transzparens – teljes relációkat lát
- Fajtái
  - Vertikális
    - Attribútumok mentén – sémadekompozíció
    - Lekérdezéshalmaztól függhet
  - **Horizontális**
    - Elemek (sorok) mentén
    - Lehet lekérdezéshalmaztól függő
    - **Lehet attól független**



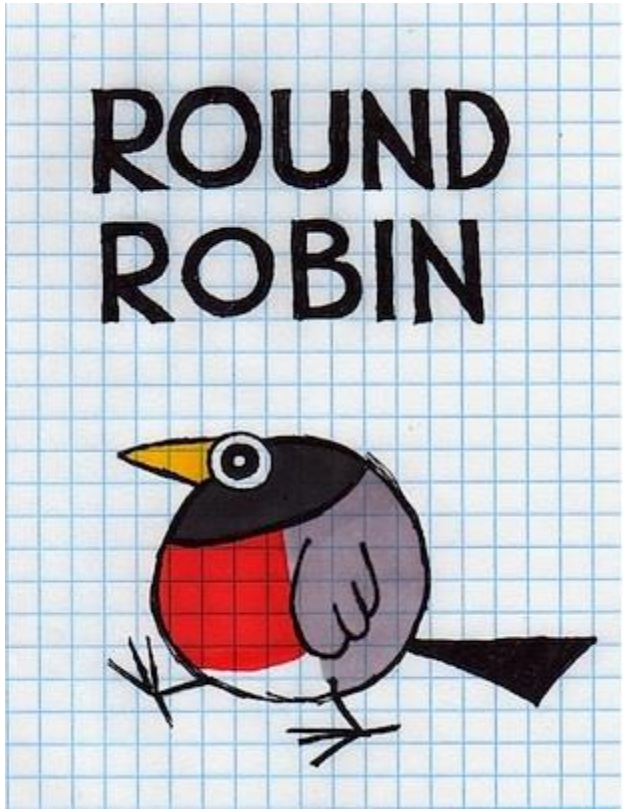
I/O párhuzamosítás

- Horizontális partícionálás
  - D[1],…,D[n] partíciók, köztük szórjuk szét a reláció elemeit
  - Round-robin partícionálás
  - Hash partícionálás
  - Tartomány alapú partícionálás (range partitioning)

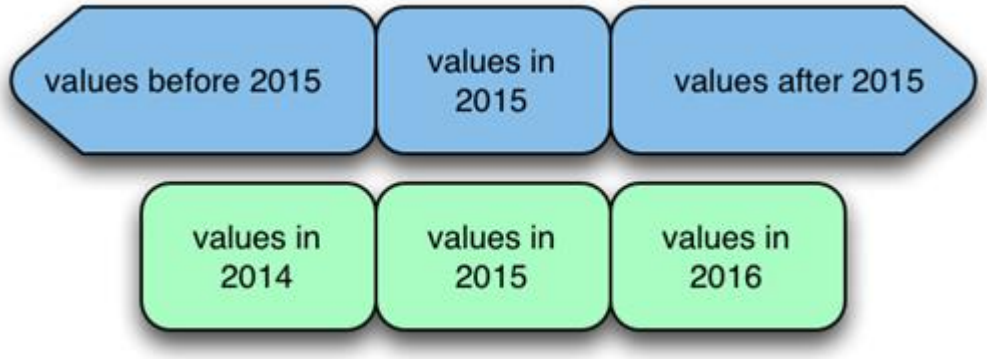


I/O párhuzamosítás

- Round-robin partícionálás
  - Elemek sorszámozása
  - i. elem a D[i mod n+1] partícióra kerül
  - Előny – elemek egyenletes elosztása
  - Hátrány – véletlenszerűség
    - Lekérdezés-végrehajtás hatékonysága bánja



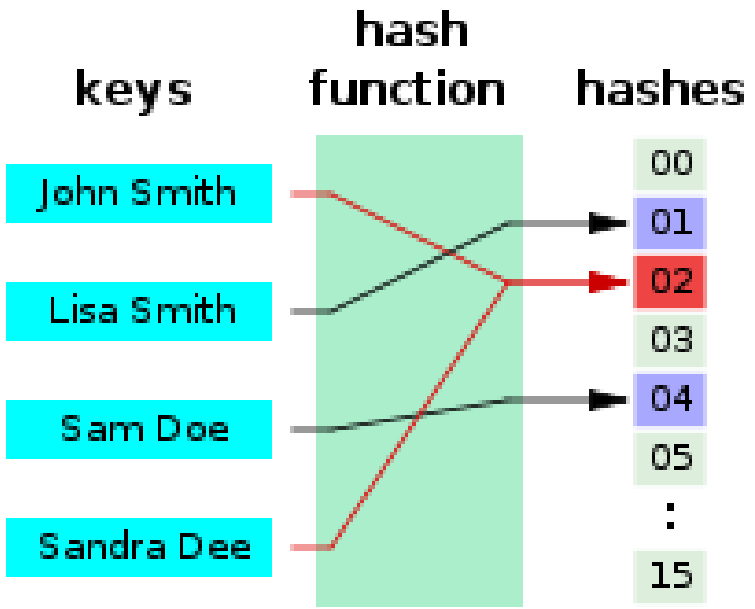
I/O párhuzamosítás



- Tartomány szerinti partícionálás
  - Partícionáló attribútum: A
  - Partícionáló vektor: (v[0],v[1],…,v[n-2]), szigorúan monoton növekvő elemek
  - Partícionálás:
    - D[1] partícióra kerülnek azon elemek, amelyekre A<v[0]
    - D[2] partícióra kerülnek azon elemek, amelyekre v[0]<=A<v[1]
    - D[3] partícióra kerülnek azon elemek, amelyekre v[1]<=A<v[2]
    - …
    - D[n] partícióra kerülnek azon elemek, amelyekre v[n-2]<=A
  - Előny: Lekérdezés-végrehajtás szempontjából a legjobb
  - Hátrány: Egyenetlen lehet a partíciók elemszáma – terheléselosztás
  - Megj: Lehetséges több attribútum alapján is

I/O párhuzamosítás

- Hash partícionálás
  - Hash-függvény:
    h(a1,…,am) → [0..n-1]
    partíció-sorszám
  - Hátrány – Csak jó hash-fv esetén egyenletes
  - Előny – bizonyos lekérdezések hatékonyabban hajthatók végre



I/O párhuzamosítás

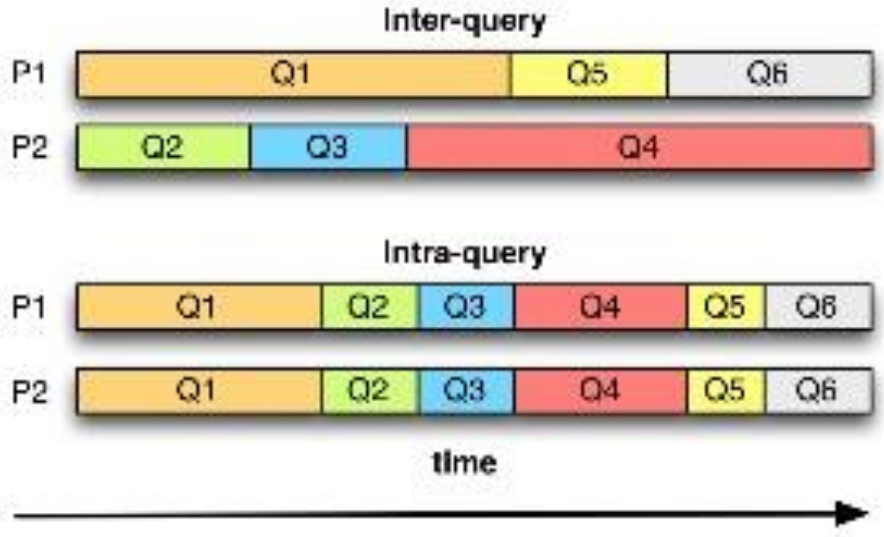
- Partícionálási módszerek összehasonlítása
- 3 szempont:

	RR	Hash	Tartomány
Teljes tábla olvasása	OK	OK	OK
Egy elem megtalálása (A=354, ahol A a partícionáló attribútum)		OK	OK
Intervallumkeresés (355<A<=553, ahol A a partícionáló attr)			OK

- Egyéb műveleteket is érinthet (pl. join – lásd később)

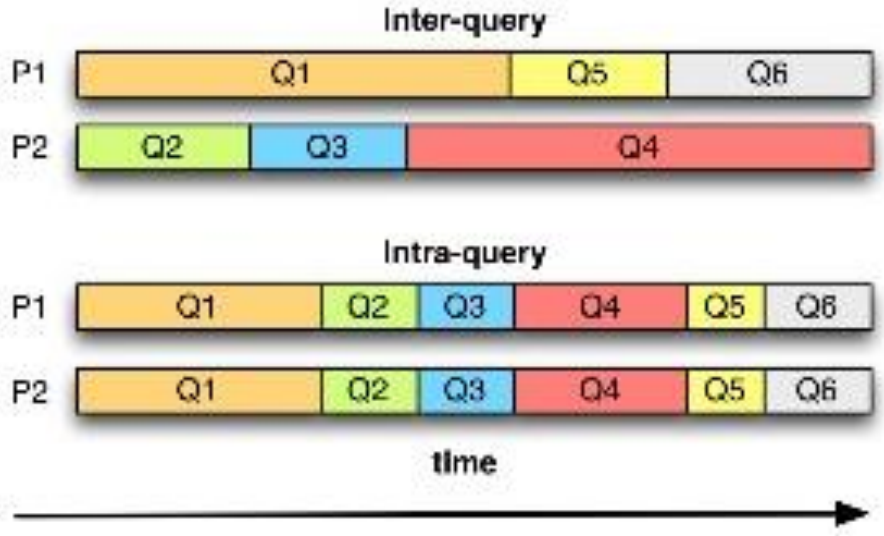
Lekérdezésközi párhuzamosítás (Interquery parallelism)

- Több lekérdezés fut párhuzamosan
- Cél: Több lekérdezés végrehajtása időegységenként
- Megoldás:
  - Eddig: Több lekérdezés egy processzoron
  - Most: Lekérdezések szétszórva több processzorra
  - Egy lekérdezés egy processzoron fut csak
  - Egy lekérdezés önmagában nem gyorsabb, mintha izoláltan futna
  - Durva granularitás, sok processzorra rosszul skálázódik



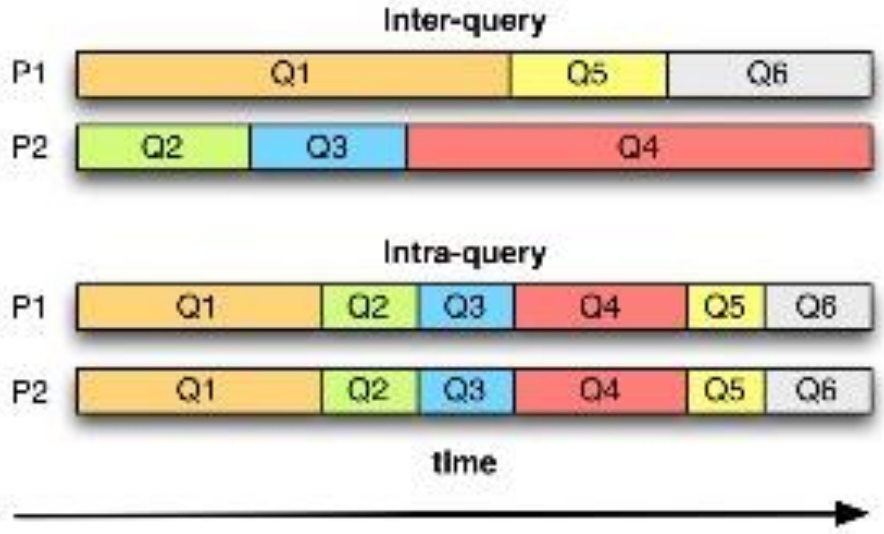
Lekérdezésen belüli párhuzamosítás (Intraquery parallelism)

- Egy lekérdezés több processzoron és háttértáron futhat
  - Egy lekérdezés végrehajtása **lehet** gyorsabb, mintha izoláltan futna
- 1. eset:
  - Összetett relációalgebrai kifejezés végrehajtása
  - Lekérdezési fa független műveletei párhuzamosan, külön processzoron futnak
  - Pipeline-ba szervezhető műveletek párhuzamosan, külön processzoron futnak
  - **Műveletek közötti párhuzamosítás (Interoperation parallelism)**
  - Finom/durva granularitás között, sok processzorra rosszul skálázódik



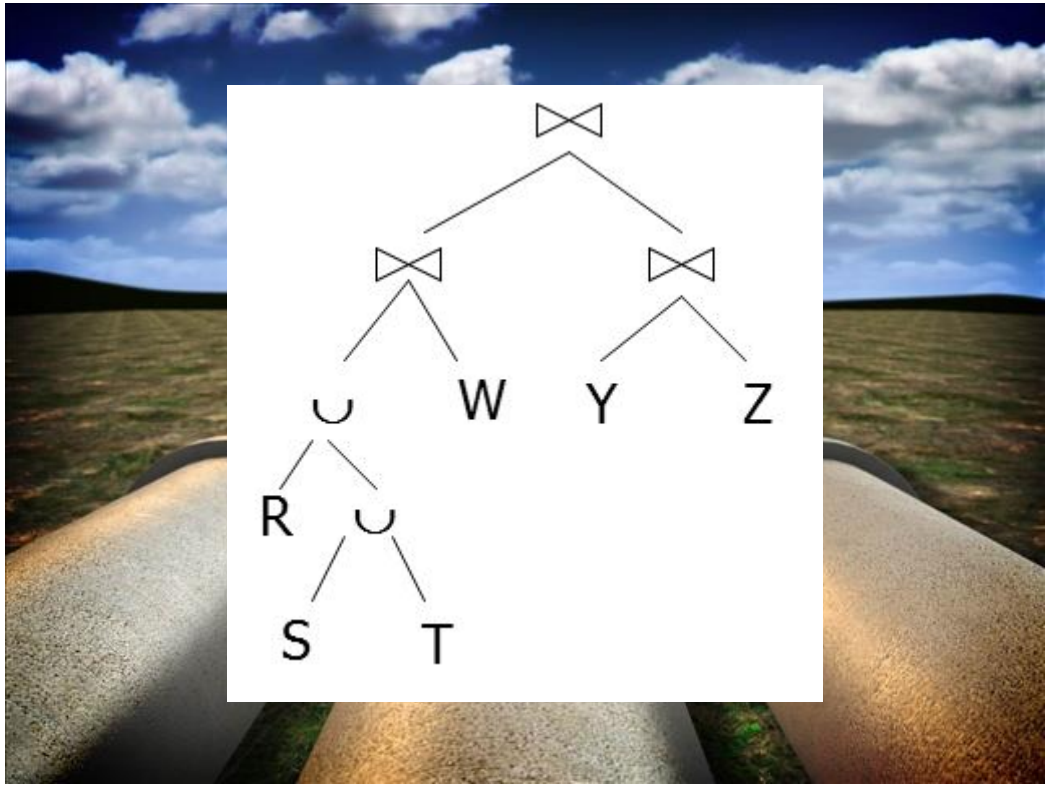
Lekérdezésen belüli párhuzamosítás (Intraquery parallelism)

- Egy lekérdezés több processzoron és háttértáron futhat
  - Egy lekérdezés végrehajtása **lehet** gyorsabb, mintha izoláltan futna
- 2. eset:
  - Feladat: Reláció rendezése
  - Tábla (horizontálisan) elosztottan tárolódik
  - Megoldás: Minden partíciót rendezünk külön, majd összefésüljük.
  - **Műveleten belüli párhuzamosítás (Intraoperation parallelism)**
  - Finom granularitás, jól skálázható



Műveletek közti párhuzamosítás (interoperation parallelism)

- Pipeline
  - Egymásra épülő műveletek pipeline-ba szervezhetőek
- Független párhuzamosítás (independent parallelism)
  - Egymástól független műveletek párhuzamosan végrehajthatók
- Durva granularitás



Műveleten belüli párhuzamosítás (Intraoperation parallelism)

- Rendezés
- Join
- Szelekció
- Ismétlődések szűrése
- Projekció

- **Shared-nothing architektúrán vizsgáljuk**

Rendezés

- Párhuzamos rendezés (Parallel sort)
  - **Tartomány szerint partícionált** D[1],…,D[n] partíciók eleve P[1],…,P[n] processzorok
  - Menete
    - Minden processzor rendezi a partícióját
    - Összefűzés (nem összefésülés) – triviális
    - Feltétel!
  - Példa1 – {1,4,5,3,2}, {20,13,16} a két partíció
- És ha nincs eleve tartomány szerint partícionálva?
- Tartomány szerinti partícionálás+rendezés (Range partitioning sort)
  - Újrapartícionálunk, más processzorokat (is) bevonhatunk, mint ahol az adat van
  - Menete
    - Rekordok elosztása, új tartományok szerint(, új processzorokra), ideiglenesen
      - Cél: hasonló méretű partíciók
    - Minden partíció rendezése függetlenül
    - Összefűzés
  - Példa2 – {1,13,5,16}, {20,2,4,3} bemenet, 4 processzorra osztjuk szét a [3, 5, 16] partíciós vektor szerint.

Rendezés

- Párhuzamos, külső összefésüléses rendezés (Parallel external sort-merge)
  - Mindegy, milyen módszerrel partícionált eleve az adathalmaz
  - Adott: P[1],…,P[n] processzorok; D[1],…,D[n] partíciók
  - Menete:
    - Minden processzor rendezi a saját partícióját
    - Összefésülés
      - Párhuzamosítható
        - P[i] processzor rendezett részeredményét tartomány szerint partícionáljuk → r[i1],…,r[im]
        - Minden P[i] processzor az r[ij] partícióját átküldi → P[j] processzor
        - Minden P[j] processzor összefésüli a többi processzortól bejövő (már rendezett) listákat
        - Összefűzés – triviális
- Példa3 – RR partícionálás után {10,40,30,20} {33,23,13,43} {46,36,26,16}

Rendezés – Példa3 levezetése – házi feladat

- Eredeti partíciók: r1={10,40,30,20} r2={33,23,13,43} r3={46,36,26,16}
- P1, P2, P3 processzor rendre rendezi a nála lévő partíciókat
- Rendezés után: r1={10,20,30,40} r2={13,23,33,43} r3={16,26,36,46}
- Most kéne összefésülni, de mi ezt párhuzamosítani szeretnénk, így újraelosztunk {22,35} partíciós vektor szerint
- Újraelosztás eredménye:
  r11={10,20} r12={30} r13={40}
  r21={13} r22={23,33} r23={43}
  r31={16} r32={26} r33={36,46}
- Összefésüléshez r11, r21, r31 → P1; r12, r22, r32 → P2; r13, r23, r33 → P3
- Mindhárom processzor *párhuzamosan* összefésüli a neki küldött partíciókat
- Tehát mindenki egyszerre dolgozik, azaz párhuzamosítottunk
- Összefésülés után: r1’={10,13,16,20} r2’={23,26,30,33} r3’={36,40,43,46}
- r1’ r2’ és r3’ össze**fűzésével** előáll a végeredmény, tehát nem kell összefésülni.

Join

- Partícionált join (Partitioned join)
  - Equijoin (ahol a join-feltétel egyenlőség) esetén működik
    - Pl. $r \bowtie_{r.A=s.B} s$
  - Adott: r és s relációk, akárhogyan partícionálva
  - Menete?
    - Relációk tartomány alapú (újra)partícionálása join attribútumok szerint → r[1],…,r[n]; s[1],…,s[n]
    - r[i] és s[i] → P[i] processzor: Lokálisan illesztjük a partíciókat
    - Unió

Join

- Partícionált join (Partitioned join)
  - Equijoin (ahol a join-feltétel egyenlőség) esetén működik
    - Pl. $r \bowtie_{r.A=s.B} s$
  - Adott: r és s relációk, akárhogyan partícionálva
  - Menete?
    - Milyen módon érdemes partícionálni?
      - Tartomány
      - Hash is jó!!
    - Partícionálás követelményei
      - Ugyanazon vektor
      - Ugyanazon hashfüggvény

Join

- Partícionált join (Partitioned join)
  - Equijoin (ahol a join-feltétel egyenlőség) esetén működik
    - Pl. $r \bowtie_{r.A=s.B} s$
  - Adott: r és s relációk, akárhogyan partícionálva
  - Menete?
    - Processzoron belül bármilyen join-algoritmussal r[i] és s[i] illesztése
  - Hash vagy tartomány szerint?
    - Hash: egyenletesebb a partíciók mérete, ha jó
      - Rossz, ha a join attribútumon belül nem egyenletes az értékek eloszlása
    - Tartomány szerint: Ha a végeredményt a join attribútum szerint utólag rendezni akarjuk, nem kell merge, csak összefűzés.

Join

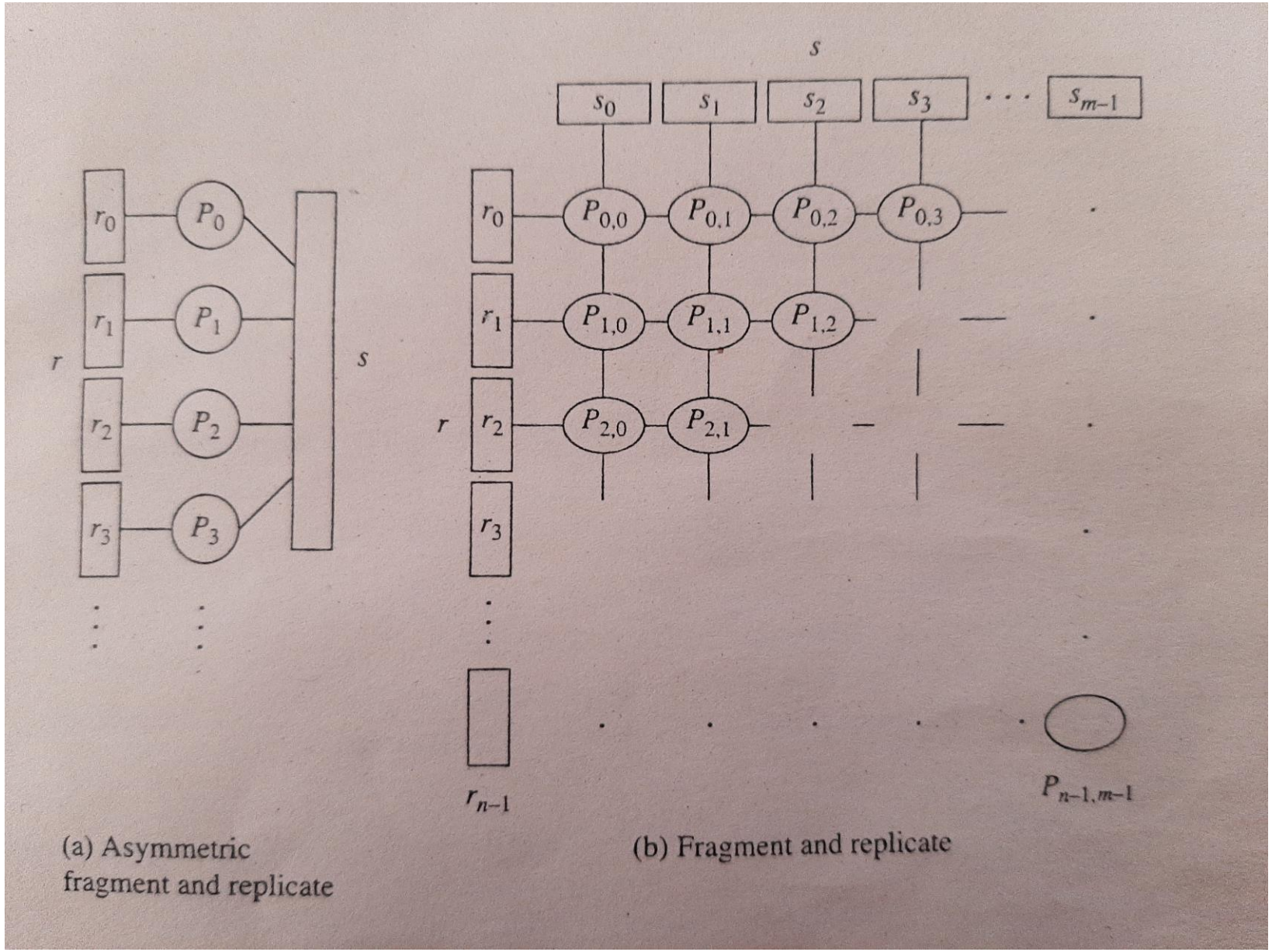
- Partícionált join (Partitioned join)
  - Equijoin (ahol a join-feltétel egyenlőség) esetén működik
    - Pl. $r \bowtie_{r.A=s.B} s$
  - Adott: r és s relációk, akárhogyan partícionálva
  - Könnyítés:
    - Ha egyik reláció már partícionált (és elég annyi partíción párhuzamosítani)
  - Példa4 – r natural join s
    - R(A,B), S(B,C)
    - r={(a,2), (b,3), (c,1), (d,2)}
    - s={(2,x), (1,y), (2,y), (1,z)}, eleve partícionált
    - Partíciós vektor: {2}
  - **PARTÍCIONÁLT JOIN: CSAK EQUIJOIN!!!** → és ha nem?

Join

- Felosztás, replikáció, join (fragment-and-replicate join) – Házi feladat
  - Nem equijoin esetére is – minden elemet minden elemmel össze kell vetni
    - Pl. $r \bowtie_{r.A<s.B} s$
  - Két módszer
    - Asszimmetrikus
      - Egyik reláció felosztása és replikációja: r[1],…,r[n] → P[1],…,P[n]
      - Másik reláció replikációja: s → P[1],…,P[n]
      - Elosztott join; merge
      - Példa5 → Következő dia
    - Szimmetrikus
      - Mindkét reláció felosztása és replikációja: r[0] – s[0],r[0] – s[1],…,r[n] – s[n] → P[0,0],P[0,1],…,P[n,n]
      - Elosztott join; merge
      - Példa6 → Következő dia
    - Minden r-elem összehasonlítható minden s-beli elemmel
    - Mikor melyik?
      - Ha egyik reláció kicsi → Érdemesebb szétszórni aszimmetrikusan

Join

- Felosztás, replikáció, join (fragment-and-replicate join) – Házi feladat



Join

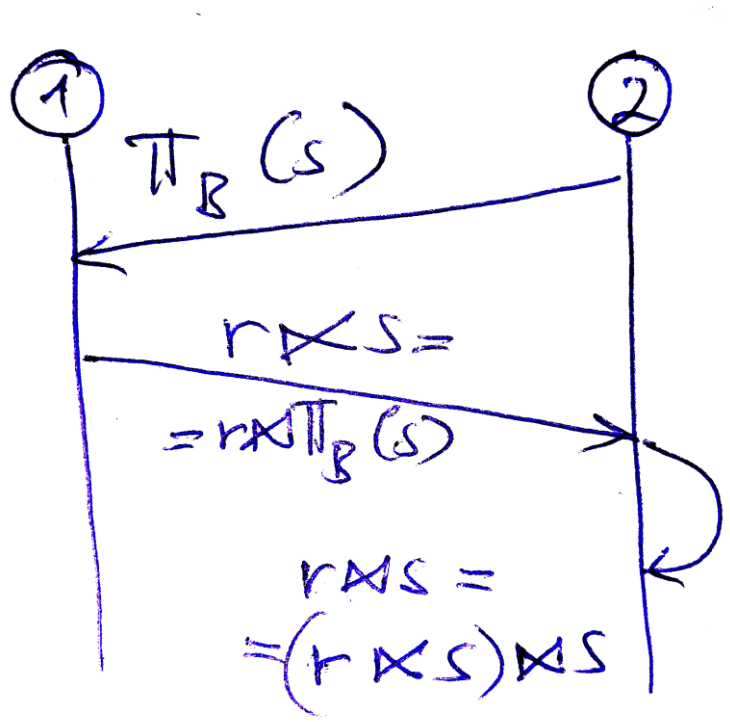
- Join eleve partícionált relációkon
  - Feltétel: Adathalmaz tartomány szerint partícionált
  - Join attribútum = partíciós attribútum mindkét relációban
- Megoldás:
  - Cél: Minden partíció illesztése minden partícióval
  - Lépés1: Lokalizált lekérdezés – partíciók behelyettesítése relációk helyére
  - Lépés2: Join-ok süllyesztése
  - Lépés3: Redukció – értelmetlen partíciópárok kiejtése → redukált lekérdezés
  - Példa7
    - Jármű JOIN személy ON ID=ownerID, ahol Jármű(ownerID, rendszám, típus), Személy(ID, név)
    - Két különböző partíciós vektor ID illetve ownerID szerint
    - v_jármű={20}
    - v_személy={30}

Join – Példa7 levezetése

```
jármű1 = sigma(ownerID<20) jármű

jármű2 = sigma(ownerID>=20) jármű

személy1 = sigma(ID<30) személy

személy2 = sigma(ID>=30) személy


jármű JOIN(ID=ownerID) személy

       <-- eredeti lekérdezés

(jármű1 U jármű2) JOIN(ID=ownerID) (személy1 U személy2)

       <-- lokalizált lekérdezés

(jármű1 JOIN személy1) U (jármű1 JOIN személy2) U (jármű2 JOIN személy1) U (jármű2
JOIN személy2)

       <-- JOIN-ok süllyesztve

(jármű1 JOIN személy1) U (jármű2 JOIN személy1) U (jármű2 JOIN személy2)
       <-- redukált lekérdezés
```

Join

- Partíciók illesztése equijoin esetén
  - Adott:
    - Előző módszerek bármelyikével (újra)partícionált relációk
      - r[i] P[g] csomópontban → „r” partíció 1. csomópontban
      - s[j] P[h] csomópontban → „s” partíció 2. csomópontban
  - Hogyan illesszük őket? – Két módszer:
    - 1. „r” átvitele 2. csomópontba, vagy „s” átvitele 1. csomópontba, majd join
    - 2. igény szerinti átvitel – csak a szükséges adatok átvitele
      - Példa6 – R(A,B), S(B,C) →
  - Mikor melyik módszer?
    - Kis partíciók → 1. módszer
    - Nagy partíciók → 2. módszer



Egyéb műveletek – Házi feladat

- Szelekció
  - 1. eset: Szelekciós attribútum =/= felosztási attribútum, vagy nem tartomány szerinti felosztás
    - Minden processzoron párhuzamosan történik a szelekció
  - 2. eset: Szelekciós attribútum = felosztási attribútum és tartomány szerinti felosztás
    - Lekérdezés átírása
    - Redukció – fölösleges partíciók eltávolítása a lekérdezésből
    - Példa8
      - 30 évnél idősebb emberek lekérdezése a Személy(név, kor, ID) séma szerinti relációból, ha kor szerint van partícionálva

Szelekció – Példa8 levezetése

```
Személy(név,kor,ID) személy(Személy)

személy1 = sigma(kor<20) személy
személy2 = sigma(kor>=20) személy

Eredeti lekérdezés: sigma(kor>30) személy

sigma(kor>30) (személy1 U személy2) <-- lokalizált lekérdezés
sigma(kor>30) (személy2) <-- redukált lekérdezés
```

Egyéb műveletek – Házi feladat

- Ismétlődések kiszűrése
  - 1. módszer:
    - Tetszőleges rendezési algoritmus használatával rendezés partíción belül
    - Ismétlődések kiszűrése, amint előkerülnek (csomóponton belül vagy összefésüléskor)
      - Hash, tartomány szerinti partícionálásnál csomóponton belül kerülnek elő
  - 2. módszer
    - Hash vagy tartomány szerinti (újra)partícionálás
    - Rendezés és szűrés lokálisan
    - MINDEN ismétlődés processzoron belül kerül elő.
    - Nagyobb egyenetlenség jelenhet meg a keletkező partíciók elemszámában.

Egyéb műveletek

- Projekció
  - Triviális
  - Bármilyen módszerrel való felosztás után, lokálisan végezhető
  - Ha ismétlődésszűrés is kell, ld. az előző diát.

--VÉGE--