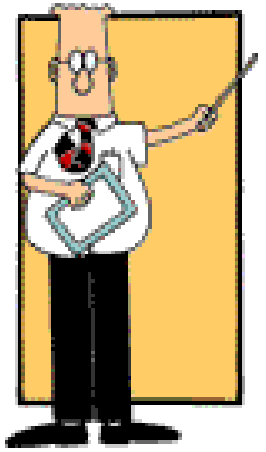
Oracle SQL Tuning

Áttekintés

- Alapozás
  - Optimalizáló, költség vs. szabály, adattárolás, SQL végrehajtási fázisok, …
- Végrehajtási tervek létrehozása és olvasása
  - Elérési utak, egyetlen tábla, összekapcsolás, …
- Eszközök
  - Követőfájlok, SQL tippek, analyze/dbms_stat
- Adattárház jellemzők
  - Csillag lekérdezés és bittérkép indexelés
  - ETL



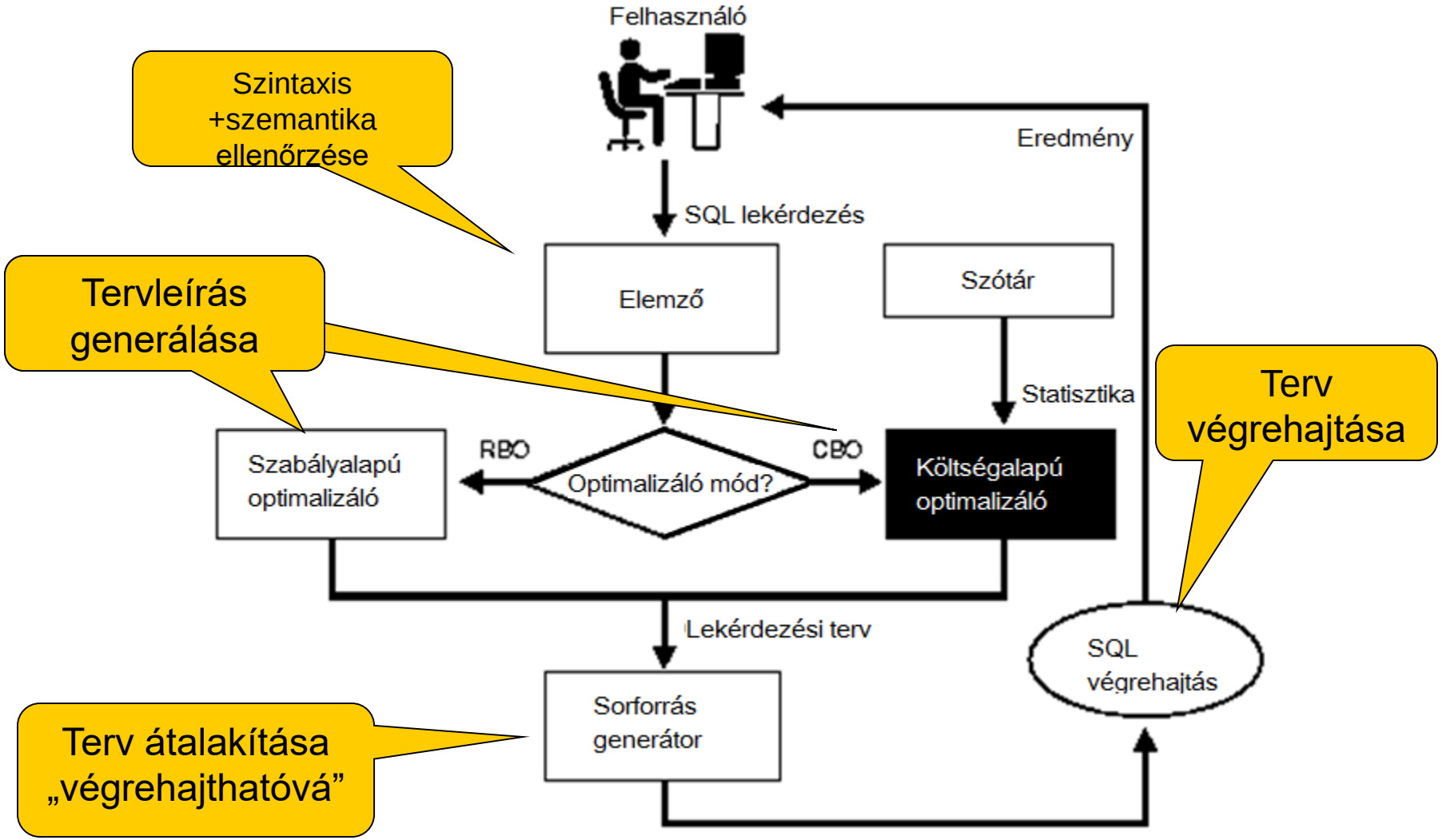
Célok

- Végrehajtási tervek olvasása
  - Táblaelérés
  - Indexelérés
  - Összekapcsolás
  - Allekérdezések
- Végrehajtási tervek megértése
  - Teljesítmény megértése
  - SQL optimalizáció alapjainak megértése
- Úgy gondolkodjunk, hogy mi hogy hajtanánk végre

Következik…

- Alapfogalmak
  - Háttérinformáció
- SQL végrehajtás
  - Olvasás + értés

Optimalizáló áttekintés



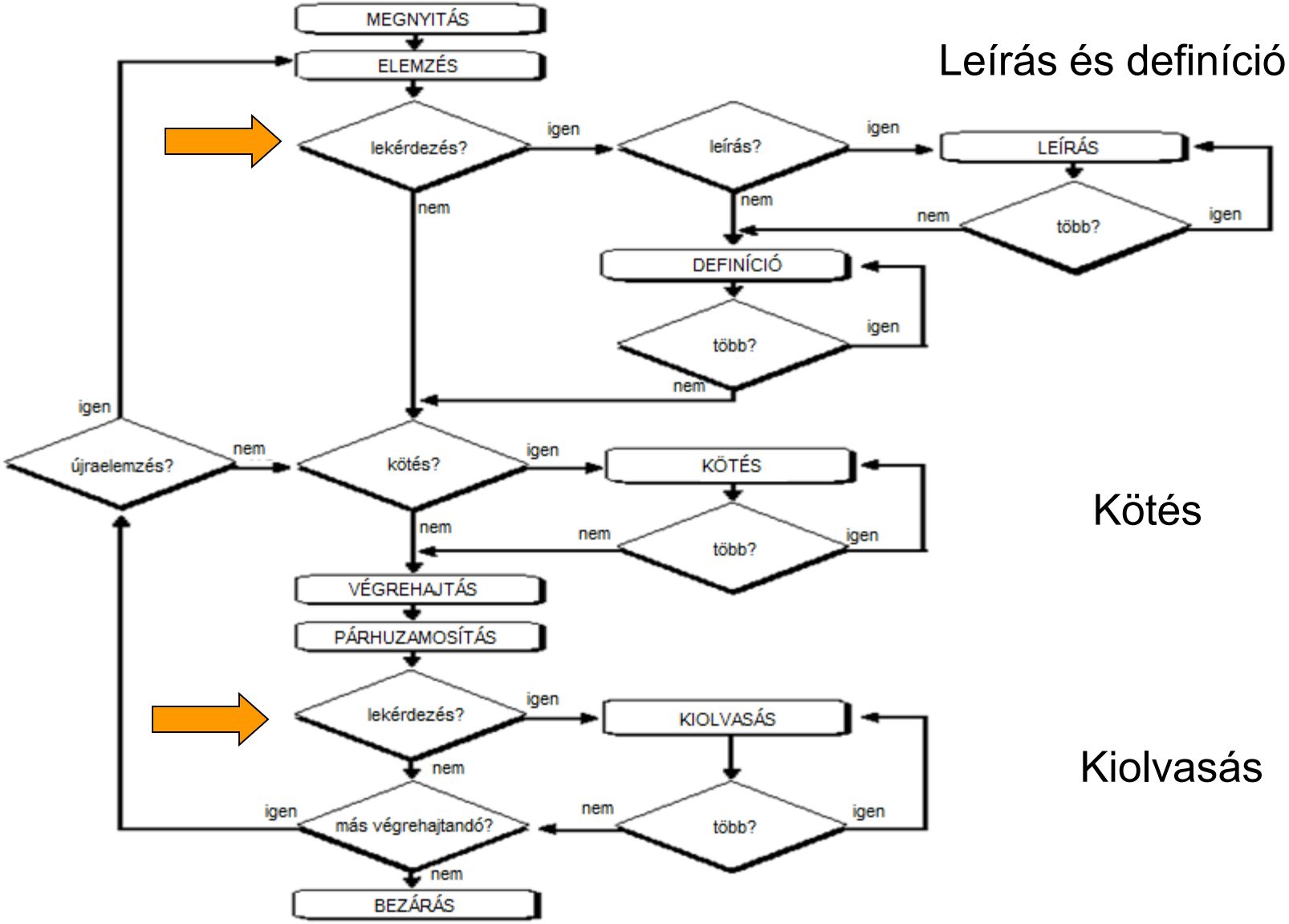
Költség vs. Szabály

- Szabály
  - Rögzített heurisztikus szabályok határozzák meg a tervet
    - „Indexen keresztül elérés gyorsabb, mint az egész tábla átnézése”
    - „teljesen megegyező index jobb, mint a részben megegyező index”
    - …
- Költség (2 mód)
  - Az adatstatisztikák szerepet játszanak a terv meghatározásában
    - Legjobb átfutás: **minden sort** minél hamarabb
      - Először számoljon, aztán gyorsan térjen vissza
    - Legjobb válaszidő: az **első sort** minél hamarabb
      - ~~Számítás közben már térjen vissza (ha lehetséges)~~

Melyiket hogyan állítjuk be?

- Példány szinten: Optimizer_Mode paraméter
  - Szabály
  - Választás
    - statisztikáknál CBO (all_rows), egyébként RBO
  - First_rows, First_rows_n (1, 10, 100, 1000)
  - All_rows
- Munkamenet szinten:
  - Alter session set optimizer_mode=<mode>;
- Utasítás szinten:
  - SQL szövegben elhelyezett tippek mutatják a használandó módot

SQL végrehajtás: DML vs. lekérdezések



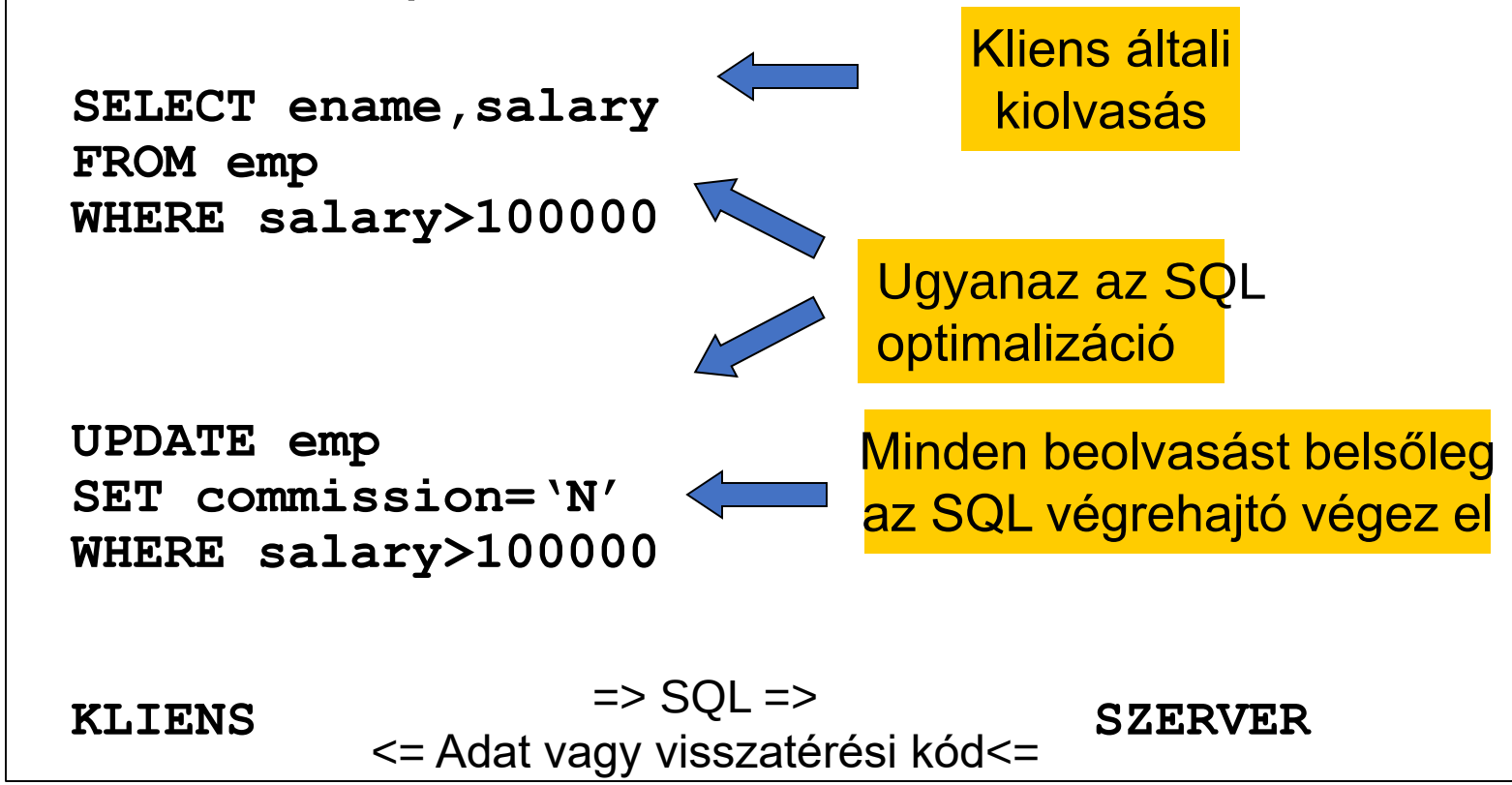
Leírás és definíció

Kötés

Kiolvasás

DML vs. Lekérdezések

- Megnyitás => Elemzés => Végrehajtás (=> Kiolvasás^{n})



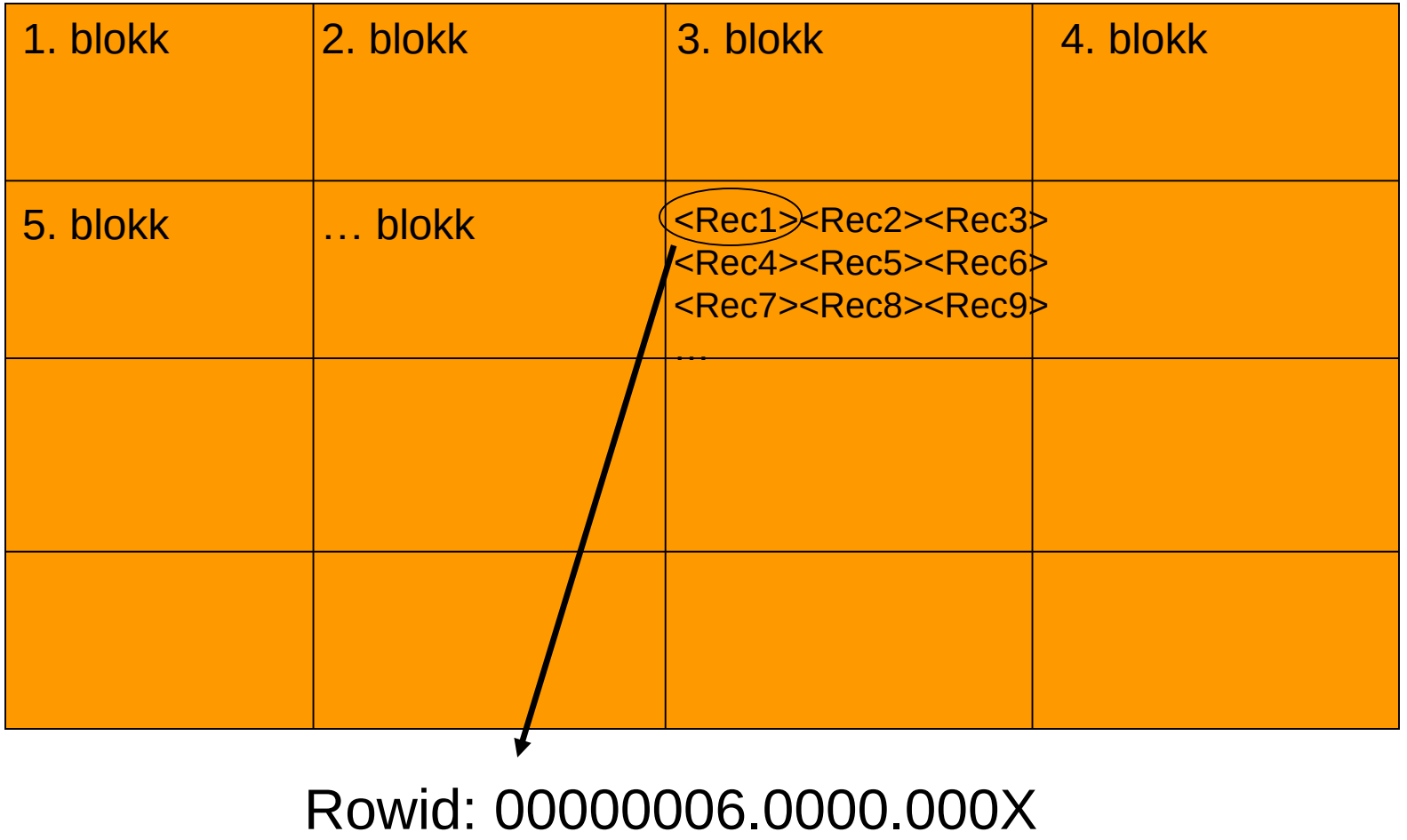
Adattárolás: Táblák

- Az Oracle az összes adatot adatfájlokban tárolja
  - Hely és méret DBA által meghatározott
  - Logikailag táblaterekbe csoportosítva
  - Minden fájlt egy relatív fájlszám (fno) azonosít
- Az adatfájl adatblokkokból áll
  - Mérete egyenlő a *db_block_size* paraméterrel
  - Minden blokkot a fájlbeli eltolása azonosít
- Az adatblokkok sorokat tartalmaznak
  - Minden sort a blokkban elfoglalt helye azonosít

ROWID: <Blokk>.<Sor>.<Fájl>

Adattárolás: Táblák

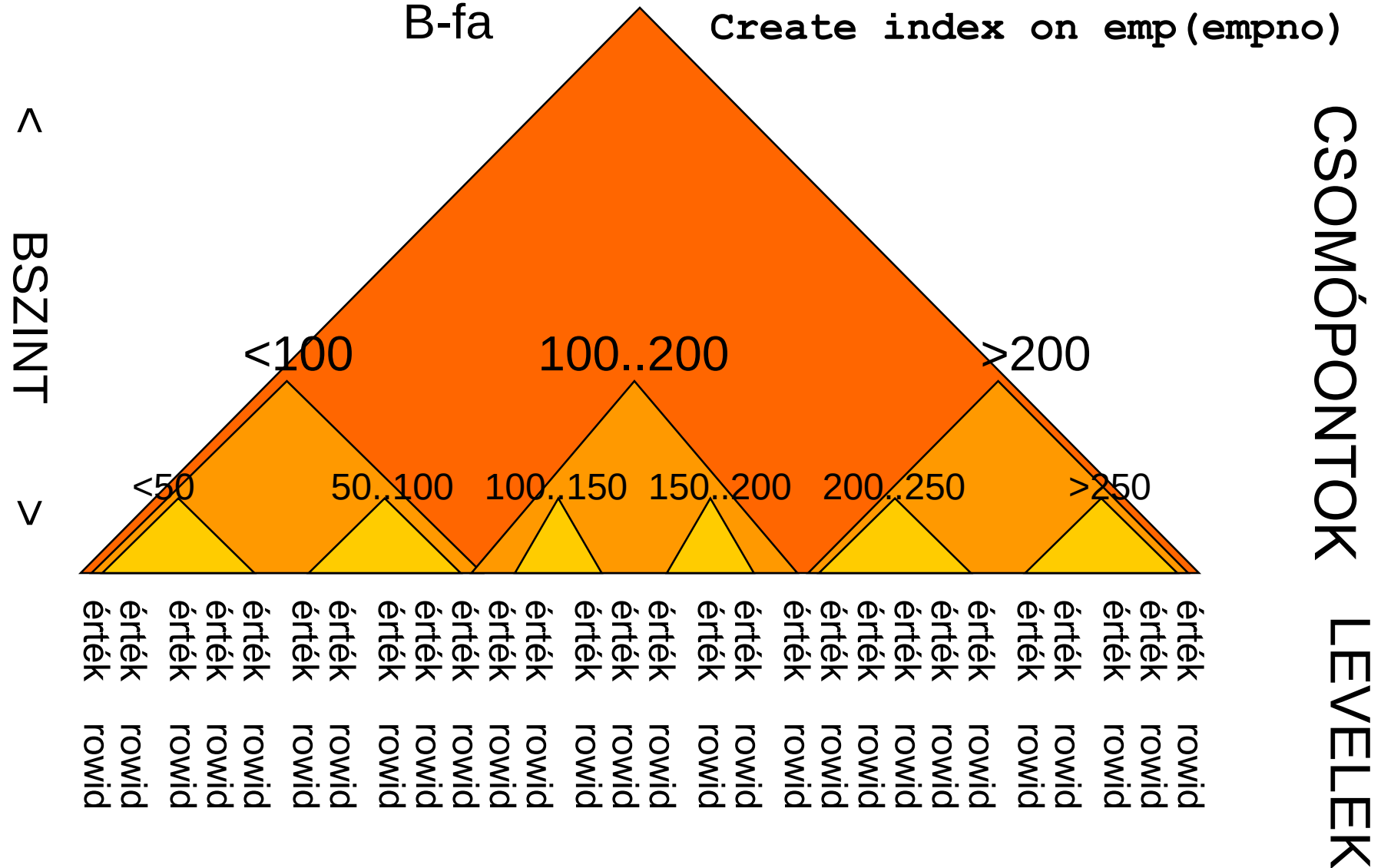
x. fájl



Adattárolás: Indexek

- Kiegyensúlyozott fák
  - Indexelt oszlop(ok) rendezett tárolása külön
    - a NULL érték kimarad az indexből
  - A mutatószerkezet logaritmikus keresést tesz lehetővé
    - Először az indexet érjük el, megkeressük a táblamutatót, aztán elérjük a táblát
- B-fa tartalma:
  - Csomópont blokkok
    - Más csomópontokhoz vagy levelekhez tartalmaz mutatókat
  - Levélblokkok
    - A tényleges indexelt adatot tartalmazzák
    - Tartalmaznak rowid-ket (sormutatókat)
- Szintén blokkokban tárolódik az adatfájlokban
  - Szabadalmazott formátum

Adattárolás: Indexek



Adattárolás: Indexek

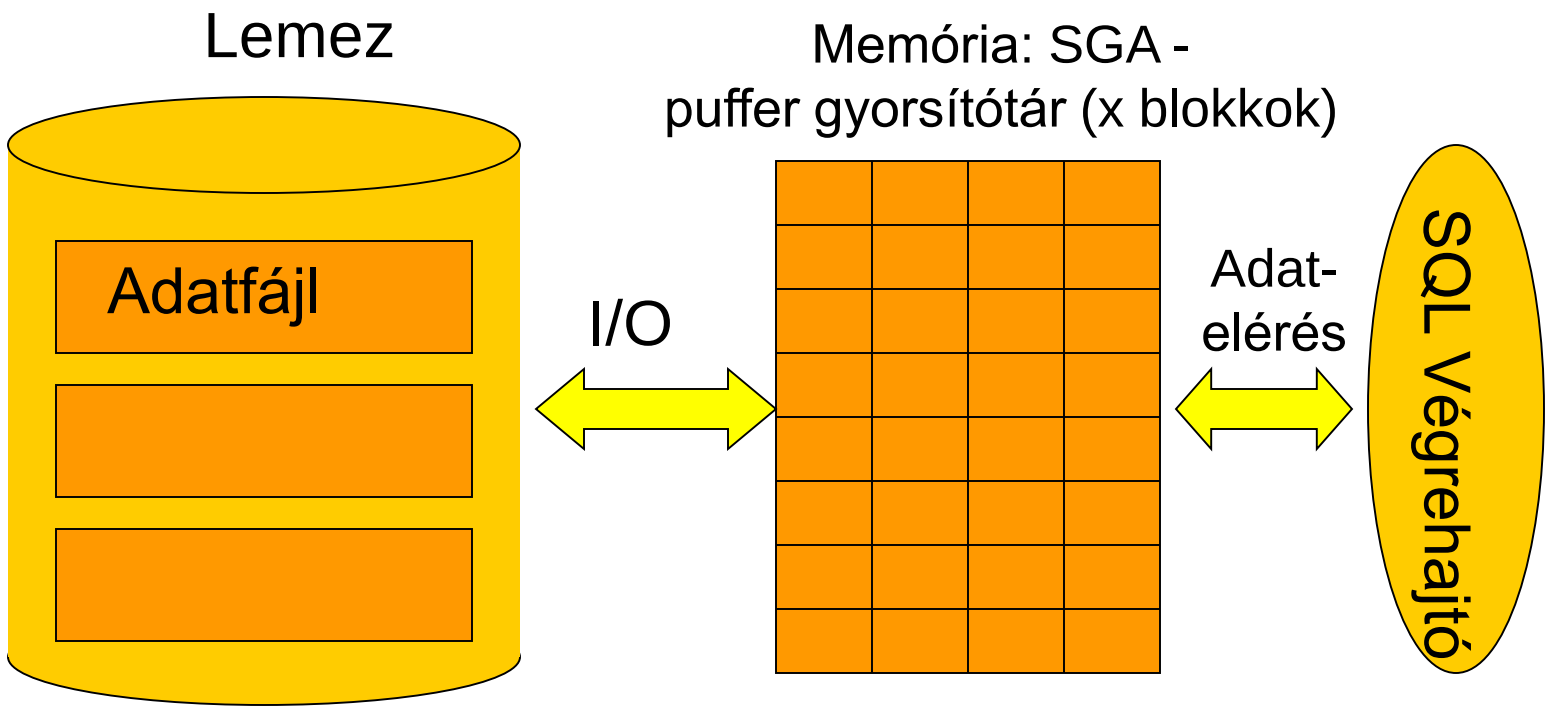
Adatfájl

1. blokk	2. blokk	3. blokk	4. blokk
5. blokk	…blokk	Index csomópont blokk	Index levél blokk
Index levél blokk			

Nincs kitüntetett sorrendje a csomópont és levél blokkoknak

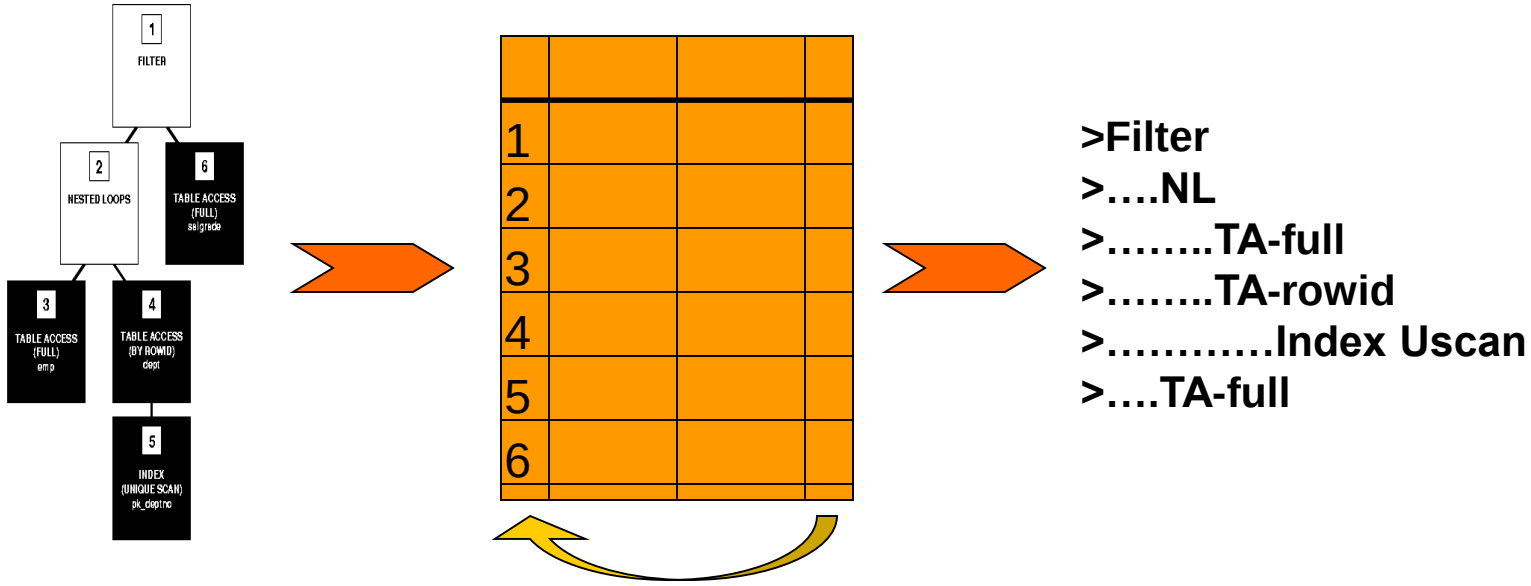
Tábla és Index I/O

- Az I/O blokk szinten történik
  - LRU lista vezérli, kinek „jut hely” a gyorsítótárban



Tervmagyarázó eszköz

- “Explain plan for <SQL-utasítás>”
  - Elmenti a tervet (sorforrások + műveletek) Plan_Table-be
  - Plan_Table nézete (vagy külső eszköz) formázza olvasható tervvé



Tervmagyarázó eszköz

```
create table PLAN_TABLE (
  statement_id    varchar2(30),   operation       varchar2(30),
  options         varchar2(30),   object_owner    varchar2(30),
  object_name     varchar2(30),   id              numeric,
  parent_id       numeric,        position        numeric,
  cost            numeric,        bytes           numeric);
```

```
create or replace view PLANS(STATEMENT_ID,PLAN,POSITION) as
select statement_id,
  rpad('>',2*level,'.')||operation||
  decode(options,NULL,'',' (')||nvl(options,' ')||
  decode(options,NULL,'',') ')||
  decode(object_owner,NULL,'',object_owner||'.')||object_name plan,
  position
from plan_table
start with id=0
connect by prior id=parent_id
       and prior nvl(statement_id,'NULL')=nvl(statement_id,'NULL')
```

Végrehajtási tervek

1. Egyetlen tábla index nélkül
2. Egyetlen tábla indexszel
3. Összekapcsolások
    1. Skatulyázott ciklusok
    2. Összefésüléses rendezés
    3. Hasítás1 (kicsi/nagy), hasítás2 (nagy/nagy)
4. Speciális műveletek

Egyetlen tábla, nincs index (1.1)

```
SELECT *
FROM emp;
```

```
>.SELECT STATEMENT
>...TABLE ACCESS full emp
```

- Teljes táblabeolvasás (FTS)
  - Minden blokk beolvasása sorozatban a puffer gyorsítótárba
    - Másik neve “buffer-gets”
    - Többszörös blokk I/O-val (db_file_multiblock_read_count)
    - Amíg a magas vízszintjelzőt el nem érjük (truncate újraindítja, delete nem)
  - Blokkonként: kiolvasás + minden sor visszaadása
    - Aztán a blokk visszarakása a LRU-végen az LRU listába (!)
    - Minden más művelet a blokkot az MRU-végre rakja

Egyetlen tábla, nincs index(1.2)

```
SELECT *
FROM emp
WHERE sal > 100000;
```

```
>.SELECT STATEMENT
>...TABLE ACCESS full emp
```

- Teljes táblabeolvasás szűréssel
  - Minden blokk beolvasása
  - Blokkonként beolvasás, szűrés, aztán sor visszaadása
    - **Az egyszerű where-feltételek nem látszanak a tervben**
    - FTS-nél: sorok-be < sorok-ki

Egyetlen tábla, nincs index (1.3)

```
SELECT *
FROM emp
ORDER BY ename;
```

```
>.SELECT STATEMENT
>...SORT order by
>.....TABLE ACCESS full emp
```

- FTS, aztán rendezés a rendezendő mező(kö)n
  - „Aztán”, tehát a rendezés addig nem ad vissza adatot, amíg a szülő sorforrás nem teljes
  - SORT order by: sorok-be = sorok-ki
  - Kis rendezések a memóriában (SORT_AREA_SIZE)
  - Nagy rendezések a TEMPORARY táblatéren
    - Lehet, hogy nagy mennyiségű I/O

Egyetlen tábla, nincs index (1.3)

```
SELECT *
FROM emp
ORDER BY ename;

Emp(ename)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS full emp
>.....INDEX full scan i_emp_ename
```

- Ha a rendezendő mező(kö)n van index
  - Index Full Scan
  - CBO használja az indexet, ha a mód = First_Rows
  - Ha használja az indexet => nem kell rendezni

Egyetlen tábla, nincs index(1.4)

```
SELECT job,sum(sal)
FROM emp
GROUP BY job;
```

```
>.SELECT STATEMENT
>...SORT group by
>.....TABLE ACCESS full emp
```

- FTS , aztán rendezés a csoportosító mező(kö)n
  - FTS csak a job és sal mezőket olvassa ki
    - Kis köztes sorméret => gyakrabban rendezhető a memóriában
  - SORT group by: sorok-be >> sorok-ki
  - A rendezés kiszámolja az aggregátumokat is

Egyetlen tábla, nincs index (1.5)

```
SELECT job,sum(sal)
FROM emp
GROUP BY job
HAVING sum(sal)>200000;
```

```
>.SELECT STATEMENT
>...FILTER
>.....SORT group by
>.......TABLE ACCESS full emp
```

- HAVING szűrés
  - Csak a having feltételnek megfelelő sorokat hagyja meg

Egyetlen tábla, nincs index(1.6)

```
SELECT *
FROM emp
WHERE rowid=
    '00004F2A.00A2.000C'
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
```

- Táblaelérés rowid alapján
  - Egy sor megkeresése
  - Azonnal a blokkra megy és kiszűri a sort
  - A leggyorsabb módszer egy sor kinyerésére
    - Ha tudjuk a rowid-t

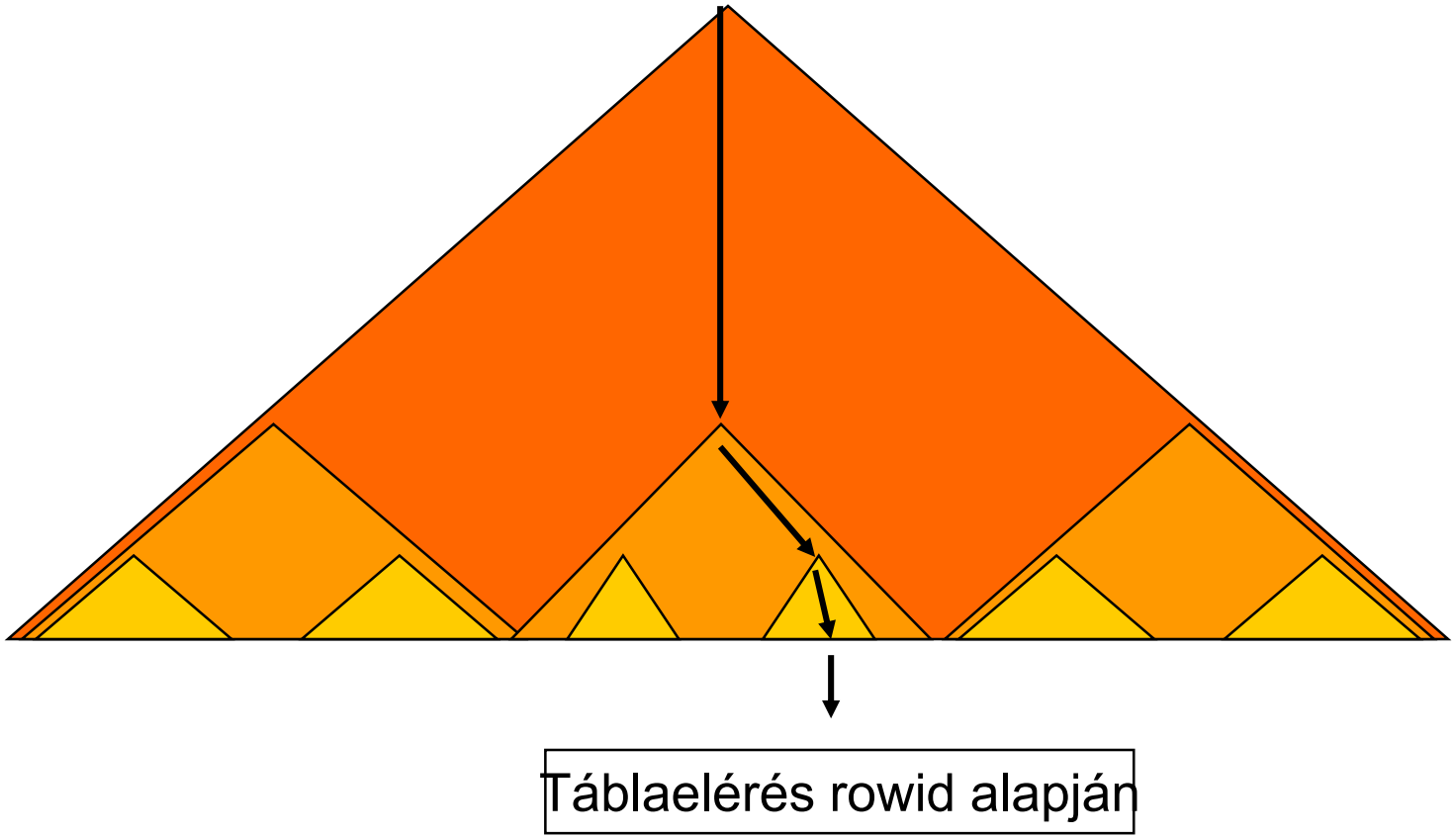
Egyetlen tábla, index(2.1)

```
SELECT *
FROM emp
WHERE empno=174;

Unique emp(empno)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX unique scan i_emp_pk
```

- Index egyedi keresés
  - Bejárja a csomópont blokkokat, hogy megtalálja a megfelelő levélblokkot
  - Megkeresi az értéket a levélblokkban (ha nem találja => kész)
  - Visszaadja a rowid-t a szülő sorforrásnak
    - Szülő: eléri a fájl+blokkot és visszaadja a sort

Index egyedi keresés (2.1)

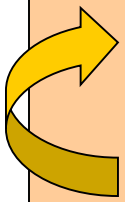


Egyetlen tábla, index(2.2)

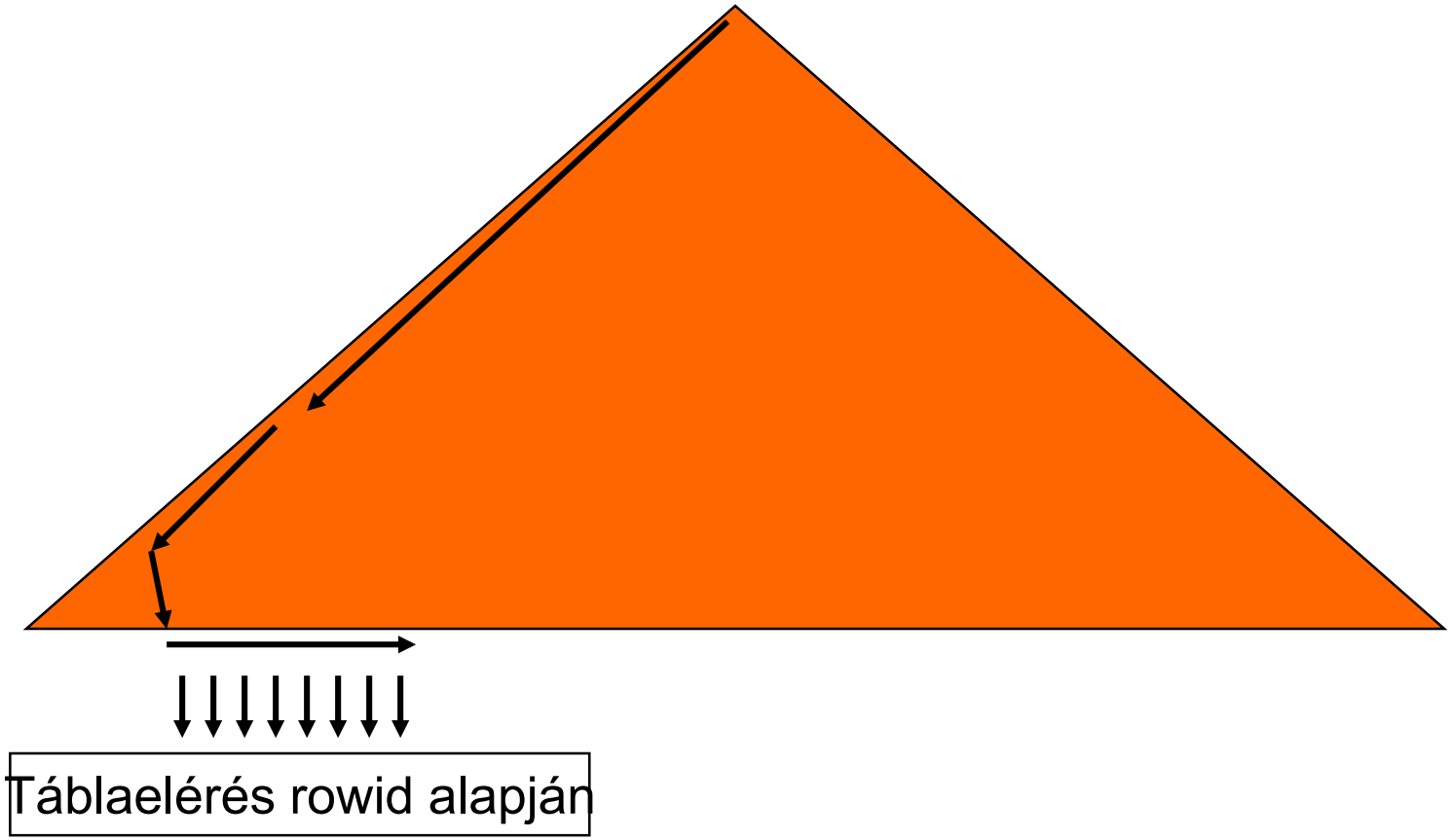
```
SELECT *
FROM emp
WHERE job='manager';


emp(job)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_job
```

- (Nem egyedi) index intervallum keresés
  - Bejárja a csomópont blokkokat, hogy megtalálja a bal szélső levélblokkot
  - Megkeresi az érték első előfordulását
  - Visszaadja a rowid-t a szülő sorforrásnak
    - Szülő: eléri a fájl+blokkot és visszaadja a sort
  - Folytatja az érték minden előfordulására
    - Amíg van még előfordulás



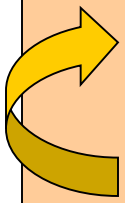
Index intervallum keresés (2.2)



Egyetlen tábla, index(2.3)

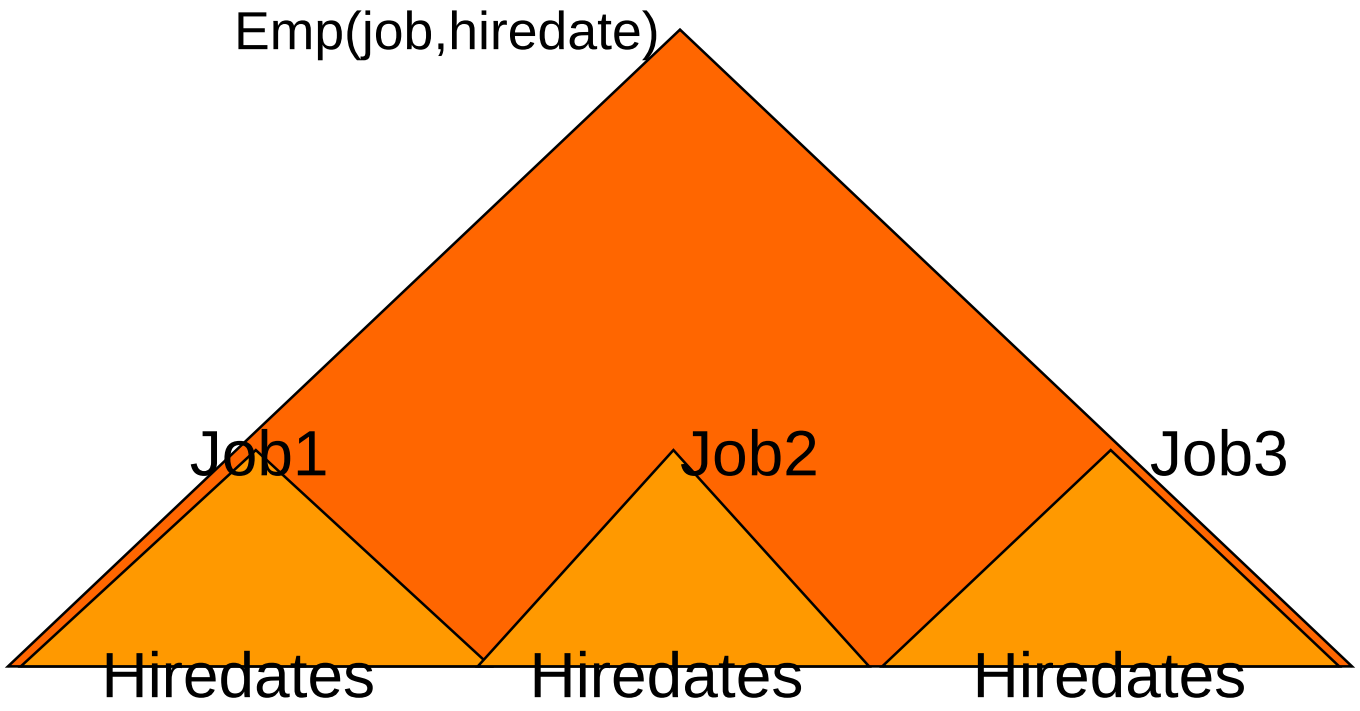
```
SELECT *
FROM emp
WHERE empno>100;

Unique emp(empno)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_pk
```



- Egyedi index intervallum keresés
  - Bejárja a csomópont blokkokat, hogy megtalálja a bal szélső levélblokkot a kezdőértékkel
  - Megkeresi az intervallumbeli első előforduló értéket
  - Visszaadja a rowid-t a szülő sorforrásnak
    - Szülő: eléri a fájl+blokkot és visszaadja a sort
  - Folytatja a következő érvényes előfordulással
    - Amíg van előfordulás az intervallumban

Összefűzött indexek



Többszintű B-fa, mezők szerinti sorrendben

Egyetlen tábla, index(2.4)

```
SELECT *
FROM emp
WHERE job='manager'
AND hiredate='01-01-2001';

Emp(job,hiredate)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_j_h
```

- Teljes összefűzött index
  - Felhasználja a job értékét az al-B-fához navigálásra
  - Aztán megkeresi az alkalmas hiredate-eket

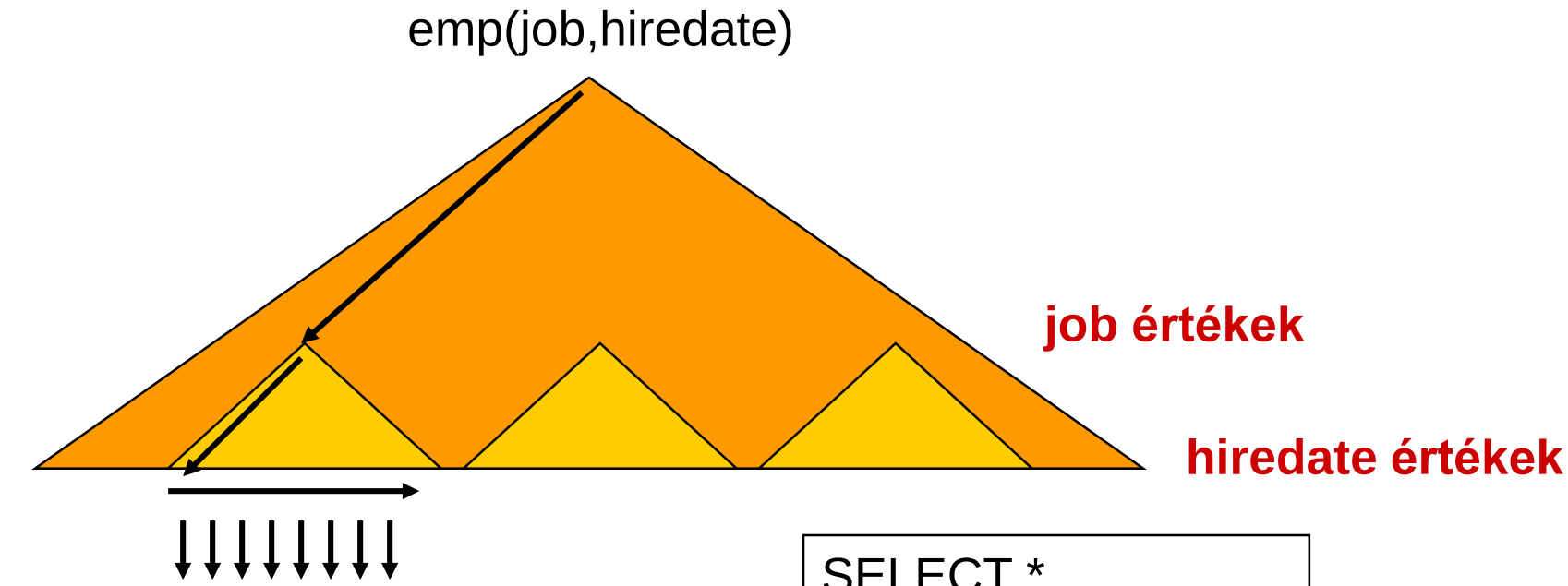
Egyetlen tábla, index(2.5)

```
SELECT *
FROM emp
WHERE job='manager';

Emp(job,hiredate)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_j_h
```

- (Bevezető) Összefűzött index prefixe
  - Végignézi a teljes al-B-fát a nagy B-fán belül

Index intervallumkeresés (2.5)




```
SELECT *
FROM emp
WHERE job='manager';
```

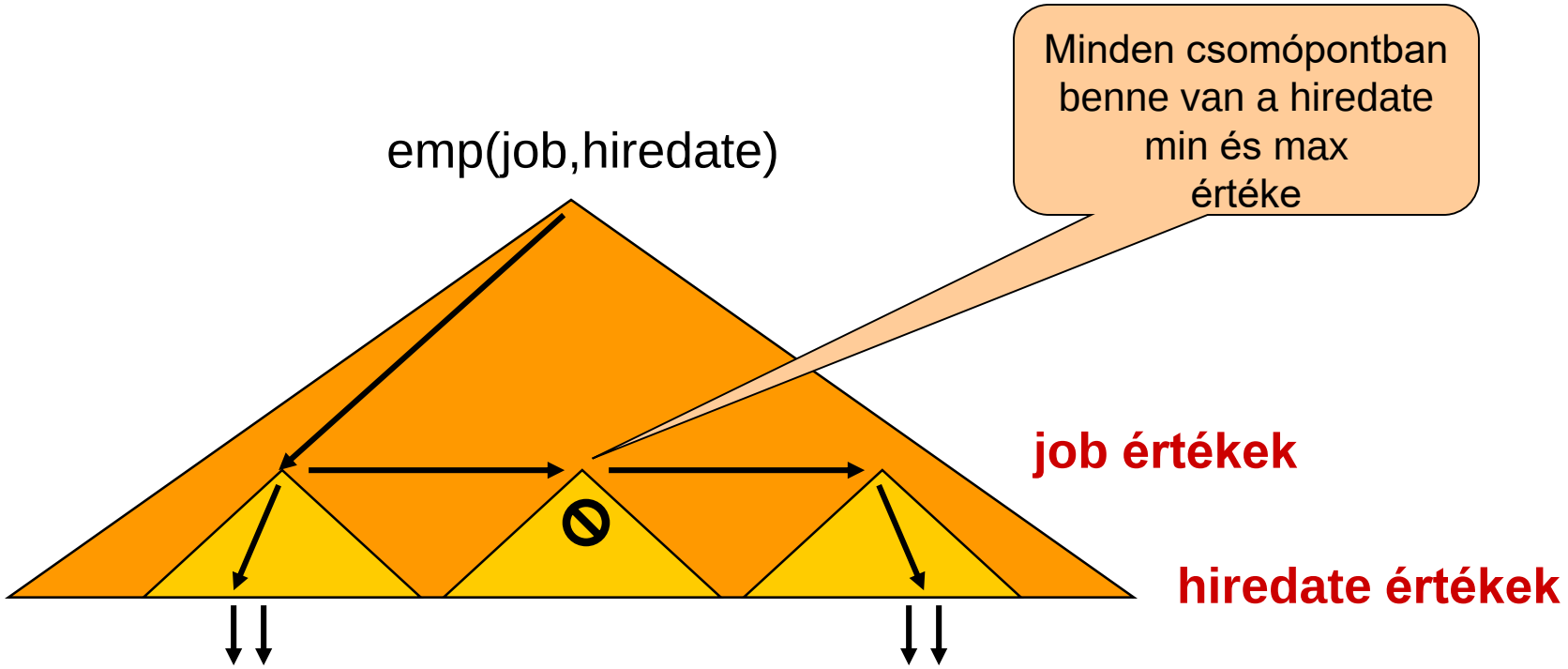
Egyetlen tábla, index(2.6)

```
SELECT *
FROM emp
WHERE hiredate='01-01-2001';

Emp(job,hiredate)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_j_h
```

- Index kihagyásos keresés (korábbi verziókban FTS)
  - „Ott használjunk indexet, ahol eddig soha nem használtuk”
  - A bevezető mezőkön már nem kell predikátum
  - A B-fát sok kis al-B-fa gyűjteményének tekinti
  - Legjobban kis számosságú bevezető mezőkre működik

Index kihagyásos keresés (2.6)




```
SELECT *
FROM emp
WHERE hiredate='01-01-2001';
```

Egyetlen tábla, index(2.7)

```
SELECT *
FROM emp
WHERE empno>100
AND job='manager';

Unique Emp(empno)
Emp(job)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_job
```

- Több index
  - Szabály: heurisztikus döntéslista alapján választ
    - Az elérhető indexeket rangsorolja
  - Költség: kiszámolja a legtöbbet kiválasztót (azaz a legkisebb költségűt)
    - Statisztikát használ

RBO heurisztikák

- Több elérhető index rangsorolása
  1. Egyenlőség egy mezős egyedi indexen
  2. Egyenlőség láncolt egyedi indexen
  3. Egyenlőség láncolt indexen
  4. Egyenlőség egy mezős indexen
  5. Korlátos intervallum keresés indexben
     - Like, Between, Leading-part, …
  6. Nem korlátos intervallum keresés indexen
     - Kisebb, nagyobb (a bevezető részen)

  Általában tippel választjuk ki, melyiket használjuk

CBO költségszámítás

- Statisztikák különböző szinteken
  - Tábla:
    - Num_rows, Blocks, Empty_blocks, Avg_space
  - Mező:
    - Num_values, Low_value, High_value, Num_nulls
  - Index:
    - Distinct_keys, Blevel, Avg_leaf_blocks_per_key, Avg_data_blocks_per_key, Leaf_blocks
- Az egyes indexek kiválasztóképességének számításához használjuk
  - Kiválasztóképesség = a sorok hány százalékát adja vissza
    - az I/O száma fontos szerepet játszik
  - FTS-t is figyelembe vesszük most!

Egyetlen tábla, index(2.1)

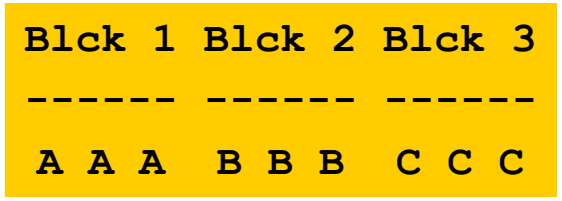
```
SELECT *
FROM emp
WHERE empno=174;

Unique emp(empno)
```

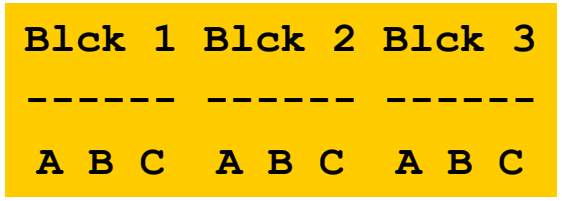
```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX unique scan i_emp_pk
Or,
>.SELECT STATEMENT
>...TABLE ACCESS full emp
```

- CBO teljes táblabeolvasást használ, ha FTS-hez szükséges I/O < # IRS-hez szükséges I/O
  - FTS I/O a db_file_multiblock_read_count (dfmrc)-t használja
    - Typically 16
  - Egyedi keresés: (bszint + 1) +1 I/O
  - FTS: ⌈táblasorok száma / dfmrc⌉ I/O

CBO: csomósodási tényező

- Index szintű statisztika
  - Mennyire jól rendezettek a sorok az indexelt értékekhez képest?
  - Átlagos blokkszám, hogy elérjünk egyetlen értéket
    - **1** azt jelenti, hogy az intervallumkeresés olcsó
    - **<táblasorok száma>** azt jelenti, hogy az intervallumkeresés drága
  - Arra használja, hogy több elérhető intervallumkeresést rangsoroljon





Clust.fact = 1

Clust.fact = 3

Egyetlen tábla, index(2.2)

```
SELECT *
FROM emp
WHERE job='manager';


emp(job)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_job
Or,
>.SELECT STATEMENT
>...TABLE ACCESS full emp
```

- Csomósodási tényező IRS és FTS összehasonlításában
  - Ha (táblasorok / dfmrc)
    <
    (értékek száma * csomó.tény.) + bszint + meglátogatandó levél blokkok
    akkor FTS-t használunk

Egyetlen tábla, index(2.7)

```
SELECT *
FROM emp
WHERE empno>100
AND job='manager';

Unique Emp(empno)
Emp(job)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_job
Or,
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_empno
```

- Csomó.tényező több IRS összehasonlításában
  - Feltesszük, hogy a FTS túl sok I/O
  - Hasonlítsuk össze (értékek száma * csomó.tény.)-t, hogy válasszunk az indexek közül
    - Empno-kiválasztóképesség => értékek száma * 1 => I/O szám
    - Job-kiválasztóképesség => 1 * csomó.tény. => I/O szám

Egyetlen tábla, index(2.8)

```
SELECT *
FROM emp
WHERE job='manager'
AND depno=10

Emp(job)
Emp(depno)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....AND-EQUAL
>.......INDEX range scan i_emp_job
>.......INDEX range scan i_emp_depno
```

- Több azonos rangú, egymezős index
  - ÉS-EGYENLŐ: legfeljebb 5 egymezős intervallumkeresést von össze
  - Kombinál több index intervallumkeresést táblaelérés előtt
    - Az egye intervallumkeresések rowid-halmazait összemetszi
  - CBO-nál ritkán fordul elő

Egyetlen tábla, index(2.9)

```
SELECT ename
FROM emp
WHERE job='manager';

Emp(job,ename)
```

```
>.SELECT STATEMENT
>...INDEX range scan i_emp_j_e
```

- Indexek használata táblaelérés elkerülésére
  - A SELECT listán levő mezőktől és a WHERE feltétel bizonyos részein
  - Nincs táblaelérés, ha az összes mező indexben van

Egyetlen tábla, index(2.10)

```
SELECT count(*)
FROM big_emp;

Big_emp(empno)
```

```
>.SELECT STATEMENT
>...INDEX fast full scan i_emp_empno
```

- Gyors teljes index keresés (CBO only)
  - Ugyanazt a több blokkos I/O-t használja, mint az FTS
  - A kiválasztható indexeknek legalább egy NOT NULL mezőt kell tartalmazniuk
  - A sorok levélblokk sorrendben adódnak vissza
    - Nem indexelt mezők sorrendben

Összekapcsolás, skatulyázott ciklusok(3.1)

```
SELECT *
FROM dept, emp;
```

```
>.SELECT STATEMENT
>...NESTED LOOPS
>.....TABLE ACCESS full dept
>.....TABLE ACCESS full emp
```

- Teljes direkt szorzat skatulyázott ciklusos összekapcsolással (NLJ)
  - Init(RowSource1);
    While not eof(RowSource1)
    Loop Init(RowSource2);
    While not eof(RowSource2)
    Loop return(CurRec(RowSource1)+CurRec(RowSource2));
    NxtRec(RowSource2);
    End Loop;
    NxtRec(RowSource1);
    End Loop;

Két ciklus, skatulyázott

Összekapcsolás, összefésüléses rendező(3.2)

```
SELECT *
FROM emp, dept
WHERE emp.d# = dept.d#;
```

```
>.SELECT STATEMENT
>...MERGE JOIN
>.....SORT join
>.......TABLE ACCESS full emp
>.....SORT join
>.......TABLE ACCESS full dept
```

- Belső összekapcsolás, nincs index: összefésüléses rendező összekapcsolás (SMJ)

```
Tmp1 := Sort(RowSource1,JoinColumn);
Tmp2 := Sort(RowSource2,JoinColumn);
Init(Tmp1); Init(Tmp2);
While Sync(Tmp1,Tmp2,JoinColumn)
Loop return(CurRec(Tmp1)+CurRec(Tmp2));
End Loop;
```

Sync továbbviszi a mutató(ka)t a következő egyezésre

Összekapcsolás (3.3)

```
SELECT *
FROM emp, dept
WHERE emp.d# = dept.d#;

Emp(d#)
```

```
>.SELECT STATEMENT
>...NESTED LOOPS
>.....TABLE ACCESS full dept
>.....TABLE ACCESS by rowid emp
>.......INDEX range scan e_emp_fk
```

- Belső összekapcsolás, csak az egyik oldal indexelt
  - NLJ a nem indexelt tábla teljes beolvasásával kezd
  - Minden kinyert sornál az indexben keresünk egyező sorokat
    - A 2. ciklusban a d# (jelenlegi) értéke elérhető!
    - És felhasználható intervallumkeresésre

Összekapcsolások (3.4)

```
SELECT *
FROM emp, dept
WHERE emp.d# = dept.d#

Emp(d#)
Unique Dept(d#)
```

```
>.SELECT STATEMENT
>...NESTED LOOPS
>.....TABLE ACCESS full dept
>.....TABLE ACCESS by rowid emp
>.......INDEX range scan e_emp_fk
Or,
>.SELECT STATEMENT
>...NESTED LOOPS
>.....TABLE ACCESS full emp
>.....TABLE ACCESS by rowid dept
>.......INDEX unique scan e_dept_pk
```

- Belső összekapcsolás, mindkét oldal indexelt
  - RBO: NLJ, először a FROM utolsó tábláján FTS
  - CBO: NLJ, először a FROM legnagyobb tábláján FTS
    - A legnagyobb I/O nyereség FTS-nél
    - Általában kisebb tábla lesz a puffer gyorsítótárban

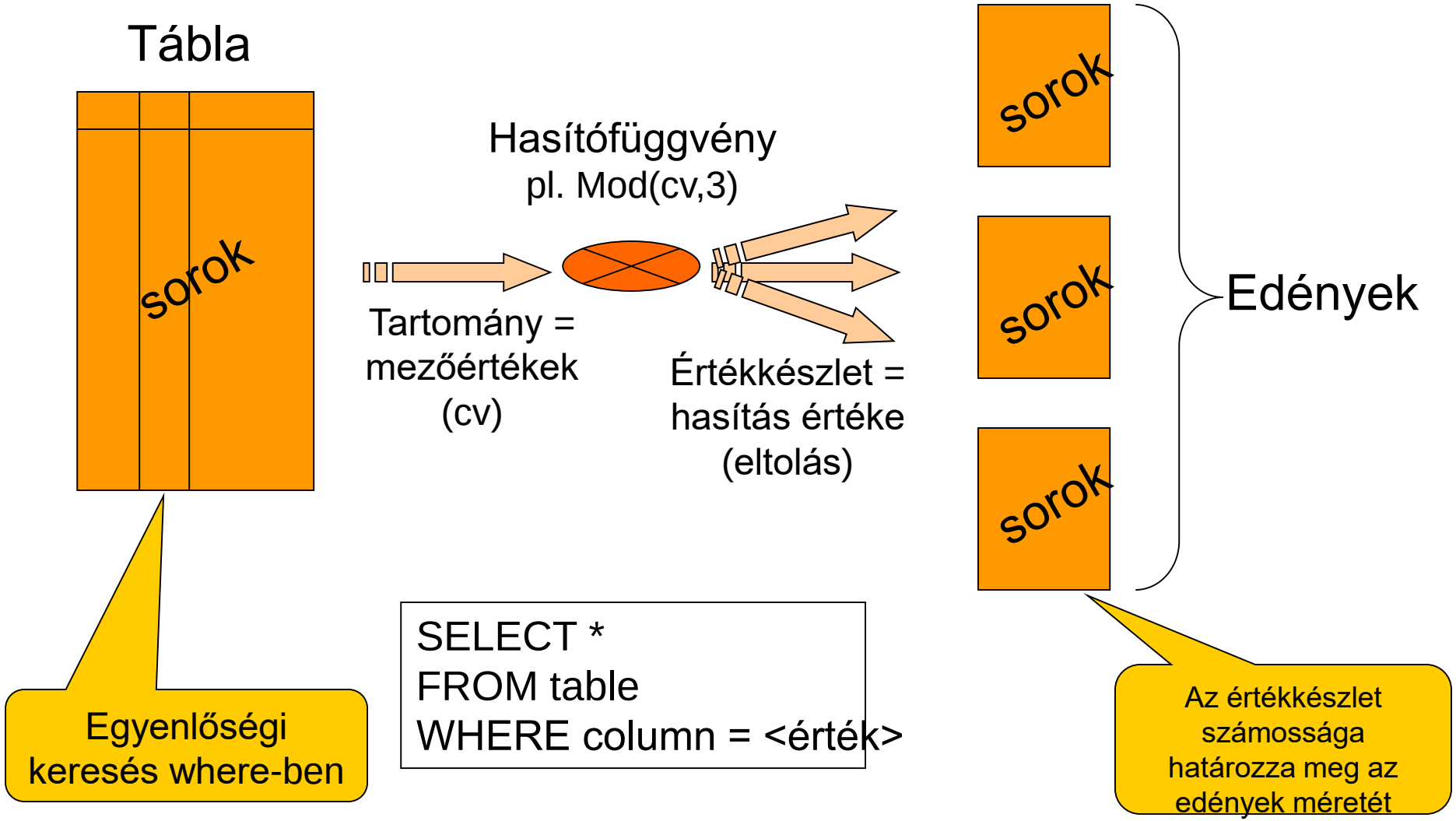
Összekapcsolások (3.5)

```
SELECT *
FROM emp, dept
WHERE emp.d# = dept.d#
AND dept.loc = ‘DALLAS’

Emp(d#)
Unique Dept(d#)
```

```
>.SELECT STATEMENT
>...NESTED LOOPS
>.....TABLE ACCESS full dept
>.....TABLE ACCESS by rowid emp
>.......INDEX range scan e_emp_fk
```

- Belső összekapcsolás plusz feltételekkel
  - Skatulyázott ciklusok
  - Mindig azzal a táblával kezdjük, amelyiken plusz feltétel van

Hasítás


Tábla
sorok
Hasítófüggvény
pl. Mod(cv,3)
Tartomány =
mezőértékek
(cv)
Értékkészlet =
hasítás értéke
(eltolás)
sorok
sorok
sorok
Edények
Egyenlőségi
keresés where-ben
SELECT *
FROM table
WHERE column = <érték>
Az értékkészlet
számossága
határozza meg az
edények méretét

Összekapcsolások, Hasítás (3.6)

```
SELECT *
FROM dept, emp
WHERE dept.d# = emp.d#

Emp(d#), Unique Dept(d#)
```

```
>.SELECT STATEMENT
>...HASH JOIN
>.....TABLE ACCESS full dept
>.....TABLE ACCESS full emp
```

- Tmp1 := Hash(RowSource1,JoinColumn); -- memóriában
  Init(RowSource2);
  While not eof(RowSource2)
  Loop HashInit(Tmp1,JoinValue); -- edény megtalálása
  While not eof(Tmp1)
  Loop return(CurRec(RowSource2)+CurRec(Tmp1));
  NxtHashRec(Tmp1,JoinValue);
  End Loop; NxtRec(RowSource2);
  End Loop;

Összekapcsolások, Hasítás (3.6)

- Explicit engedélyezni kell az init.ora fájlban:
  - Hash_Join_Enabled = True
  - Hash_Area_Size = <bytes>
- Ha a hasított tábla nem fér bele a memóriába
  - 1. sorforrás: átmeneti hasító cluster keletkezik
    - És kiíródik a lemezre (I/O) partíciónként
  - 2. sorforrás szintén konvertálódik ugyanazzal a hasítófüggvénnyel
  - Edényenként a sorok összehasonlításra kerülnek
    - Egy edénynek bele kell férnie a memóriába, különben rossz teljesítmény

Allekérdezés (4.1)

```
SELECT dname, deptno
FROM dept
WHERE d# IN
   (SELECT d#
    FROM emp);
```

```
>.SELECT STATEMENT
>...NESTED LOOPS
>.....VIEW
>.......SORT unique
>.........TABLE ACCESS full emp
>.....TABLE ACCESS by rowid dept
>.......INDEX unique scan i_dept_pk
```

- Átalakítás összekapcsolássá
  - Átmeneti nézet keletkezik, amely hajtja a skatulyázott ciklust

Allekérdezés, korrelált(4.2)

```
SELECT *
FROM emp e
WHERE sal >
  (SELECT sal
   FROM emp m
   WHERE m.e#=e.mgr#)
```

```
>.SELECT STATEMENT
>...FILTER
>.....TABLE ACCESS full emp
>.....TABLE ACCESS by rowid emp
>.......INDEX unique scan i_emp_pk
```

- Skatulyázott ciklus-szerű FILTER
  - Az 1. sorforrás minden sorára végrehajtja a 2. sorforrást és szűri az allekérdezés feltételére
  - Az allekérdezés átírható az EMP tábla ön-összekapcsolásává

Allekérdezés, korrelált (4.2)

```
SELECT *
FROM emp e, emp m
WHERE m.e#=e.mgr#
AND e.sal > m.sal;
```

```
>.SELECT STATEMENT
>...NESTED LOOPS
>.....TABLE ACCESS full emp
>.....TABLE ACCESS by rowid emp
>.......INDEX unique scan i_emp_pk
```

- Allekérdezés átírása összekapcsolássá
  - Az allekérdezés átírható EXISTS-allekérdezéssé is

Allekérdezés, korrelált(4.2)

```
SELECT *
FROM emp e
WHERE exists
 (SELECT ‘less salary'
  FROM emp m
  WHERE e.mgr# = m.e#
              and m.sal < e.sal);
```

```
>.SELECT STATEMENT
>...FILTER
>.....TABLE ACCESS full emp
>.....TABLE ACCESS by rowid emp
>.......INDEX unique scan i_emp_pk
```

- Allekérdezés átírása EXISTS allekérdezéssé
  - Az 1. sorforrás minden sorára végrehajtja a 2. sorforrást és szűri a 2. sorforrás kinyerését

Összefűzés (4.3)

```
SELECT *
FROM emp
WHERE mgr# = 100
OR job = 'CLERK';

Emp(mgr#)
Emp(job)
```

```
>.SELECT STATEMENT
>...CONCATENATION
>.....TABLE ACCESS by rowid emp
>.......INDEX range scan i_emp_m
>.....TABLE ACCESS by rowid emp
>.......INDEX range scan i_emp_j
```

- Összefűzés (VAGY-feldolgoás)
  - Hasonló, mint amikor átírjuk 2 külön lekérdezésre
  - Amelyeket azután összefűzünk
  - Ha hiányzik az egyik index => teljes táblabeolvasás

Bel-lista iterátor (4.4)

```
SELECT *
FROM dept
WHERE d# in (10,20,30);


Unique Dept(d#)
```

```
>.SELECT STATEMENT
>...INLIST ITERATOR
>.....TABLE ACCESS by rowid dept
>.......INDEX unique scan i_dept_pk
```

- Iteráció felsorolt értéklistán
  - Minden értékre külön végrehajtja
- Ugyanaz, mint 3 VAGY-olt érték összefűzése

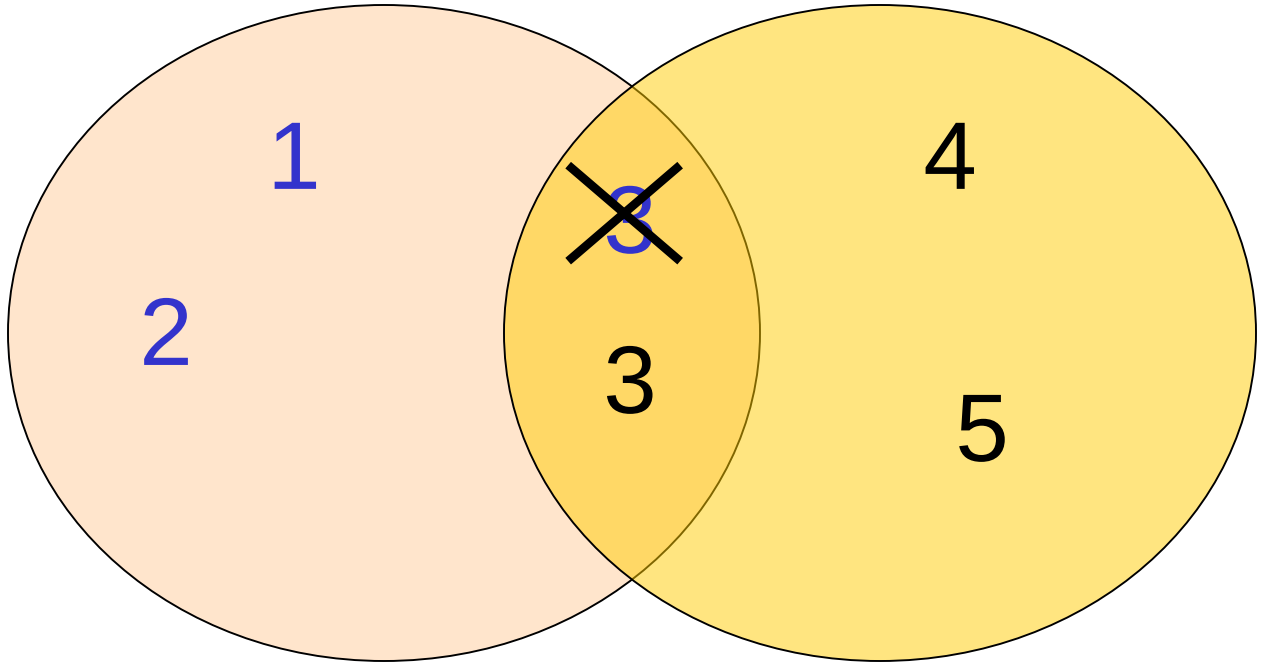
Unió (4.5)

```
SELECT empno
FROM emp
UNION
SELECT deptno
FROM dept;
```

```
>.SELECT STATEMENT
>...SORT unique
>.....UNION
>.......TABLE ACCESS full emp
>.......TABLE ACCESS full dept
```

- Unió, majd egyedi rendezés
  - Az al-sorforrások külön kerülnek optimalizálásra/végrehajtásra
  - A kinyert sorokat összefűzzük
  - A halmazelmélet miatt az elemeknek egyedinek kell lenniük (rendezés)

UNION



Minden-unió (4.6)

```
SELECT empno
FROM emp
UNION ALL
SELECT deptno
FROM dept;
```

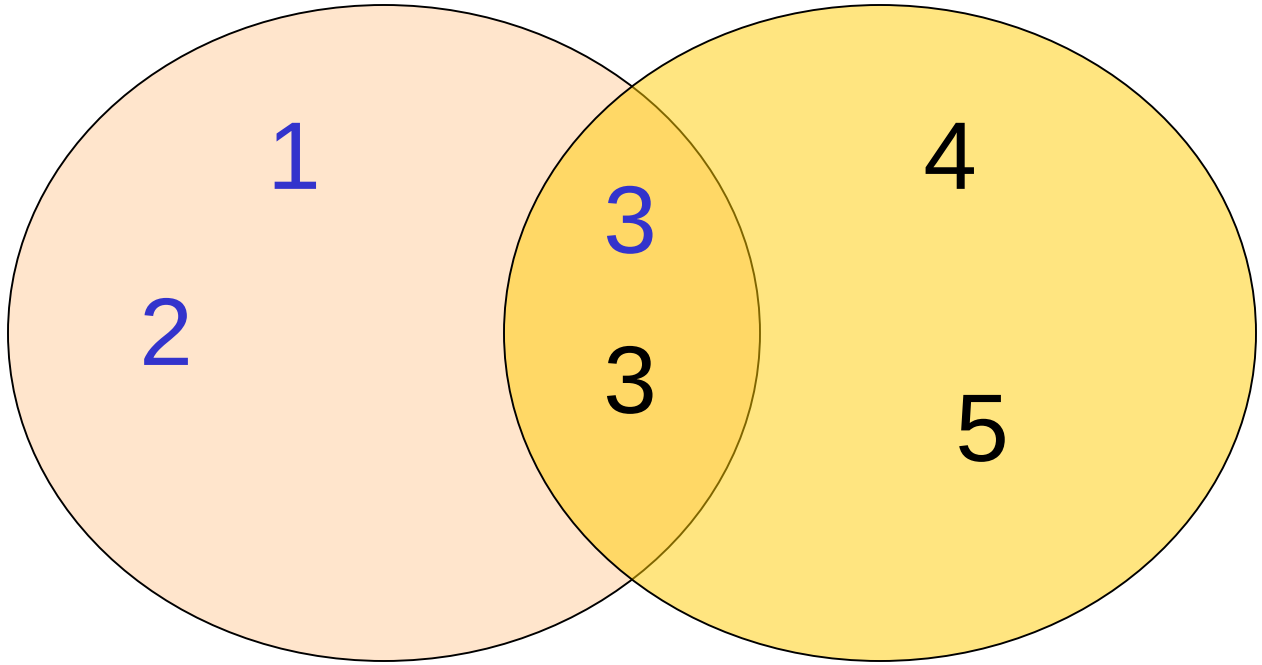
```
>.SELECT STATEMENT
>...UNION-ALL
>.....TABLE ACCESS full emp
>.....TABLE ACCESS full dept
```

- Minden-unió: az eredmény zsák, nem halmaz
  - (Drága) rendezésre nincs szükség

Használjunk UNION ALL-t, ha tudjuk, hogy a zsák halmaz

(megspórolunk egy drága rendezést)

UNION ALL



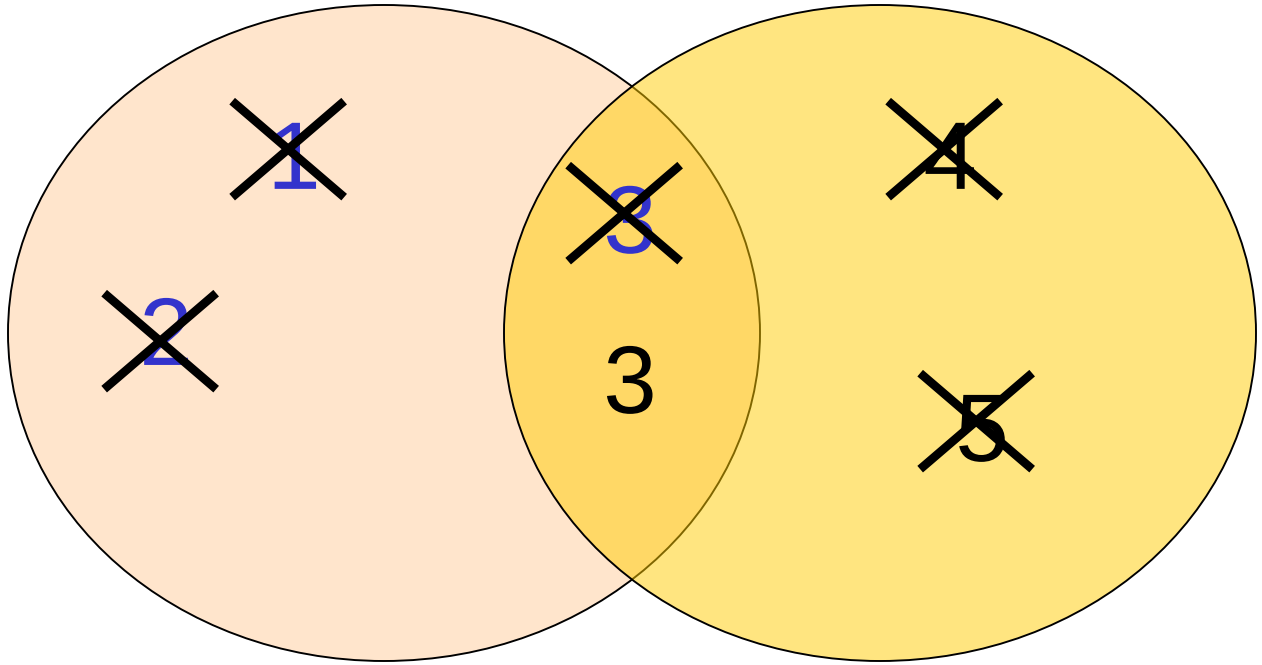
Metszet (4.7)

```
SELECT empno
FROM emp
INTERSECT
SELECT deptno
FROM dept;
```

```
>.SELECT STATEMENT
>...INTERSECTION
>.....SORT unique
>.......TABLE ACCESS full emp
>.....SORT unique
>.......TABLE ACCESS full dept
```

- INTERSECT
  - Az al-sorforrások külön kerülnek optimalizálásra/végrehajtásra
  - Nagyon hasonlít az összefésüléses rendezéshez
  - A teljes sorokat rendezi és összehasonlítja

INTERSECT



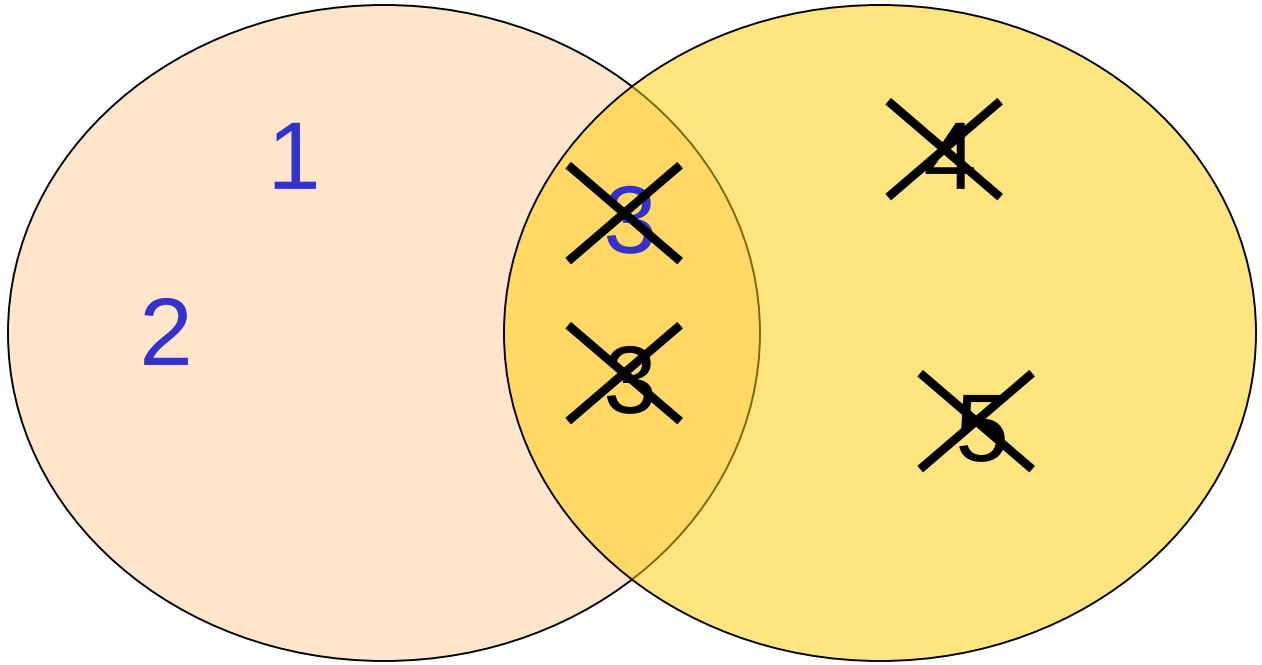
Különbség (4.8)

```
SELECT empno
FROM emp
MINUS
SELECT deptno
FROM dept;
```

```
>.SELECT STATEMENT
>...MINUS
>.....SORT unique
>.......TABLE ACCESS full emp
>.....SORT unique
>.......TABLE ACCESS full dept
```

- MINUS
  - Az al-sorforrások külön kerülnek optimalizálásra/végrehajtásra
  - Hasonlít a metszet feldolgozására
    - Összehasonlítás és visszaadás helyett összehasonlítás és kizárás

MINUS



Eszközök

- Nyomkövetés
- SQL tippek
- Analizáló parancs
- Dbms_Stats csomag

Nyomkövető fájlok

- Tervmagyarázat: beletekintés végrehajtás előtt
- Nyomkövetés: beletekintés végrehajtás közben
    - Felhasznált CPU idő
    - Eltelt idő
    - Fizikai blokk I/O száma
    - Gyorsítótárazott blokk I/O száma
    - Sorforrásonként feldolgozott sorok száma
- A munkamenetet nyomkövető módba kell állítani
    - Alter session set sql_trace=true;
    - Exec dbms_system.set_sql_trace_in_session(sid,s#,T/F);

Nyomkövető fájlok

- A nyomkövető fájl az adatbázisszerveren generálódik
  - TKPROF eszközzel kell formázni

    tkprof <nyomkövető fájl> <tkp-fájl> <user>/<pw>

  - SQL utasításonként 2 szakasz:

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.06       0.07          0          0          0           0
Execute      1      0.01       0.01          0          0          0           0
Fetch        1      0.11       0.13          0         37          2           2
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.18       0.21          0         37          2           2
```

Nyomkövető fájlok

- 2. szakasz: bővített végrehajtási terv
  - Példa 4.2 (dolgozó fizetése nagyobb, mint a menedzseréé),

```
#R  Plan                                                    .
 2  SELECT STATEMENT
14    FILTER
14      TABLE ACCESS (FULL) OF 'EMP‘
11      TABLE ACCESS (BY ROWID) OF 'EMP‘
12        INDEX (UNIQUE SCAN) OF 'I_EMP_PK' (UNIQUE)
```

- Emp tartalmaz 14 rekordot
- Kettőben nincs menedzser (NULL mgr mezőérték)
- Az egyik nem létező alkalmazottra mutat
- Ketten többet keresnek, mint a menedzserük

Tippek

- Kényszerítik az optimalizálót egy konkrét lehetőség kiválasztására
  - Beágyazott megjegyzéssel valósítjuk meg

```
SELECT /*+ <tipp> */ ….
FROM ….
WHERE ….

UPDATE /*+ <tipp> */ ….
WHERE ….

DELETE /*+ <tipp> */ ….
WHERE ….

INSERT (ld. SELECT)
```

Tippek

- Gyakori tippek
  - Full(<tab>)
  - Index(<tab> <ind>)
  - Index_asc(<tab> <ind>)
  - Index_desc(<tab> <ind>)
  - Ordered
  - Use_NL(<tab> <tab>)
  - Use_Merge(<tab> <tab>)
  - Use_Hash(<tab> <tab>)
  - Leading(<tab>)
  - First_rows, All_rows, Rule

Analizáló parancs

- A statisztikát időnként generálni kell
  - Az ‘ANALYZE’ paranccsal tehető meg

Analyze <Table | Index> <x>
<compute | estimate | delete> statistics
<sample <x> <Rows | Percent>>

Analyze table emp estimate statistics sample 30 percent;

Az ANALYZE támogatása megszűnik

Dbms_Stats csomag

- Az analizáló parancs utódja
  - Dbms_stats.gather_index_stats(<owner>,<index>, <blocksample>,<est.percent>)
  - Dbms_stats.gather_table_stats(<owner>,<table>, <blocksample>,<est.percent>)
  - Dbms_stats.delete_index_stats(<owner>,<index>)
  - Dbms_stats.delete_table_stats(<owner>,<table>)

SQL>exec dbms_stats.gather_table_status(‘scott’,’emp’,null,30);

Adattárház jellemzők

- Hagyományos csillag lekérdezés
- Bittérkép indexek
  - Bittérkép egyesítése, átalakítása rowid-dé
  - Egyetlen táblás lekérdezés
- Csillag lekérdezés
  - Több táblás

Hagyományos csillag lekérdezés

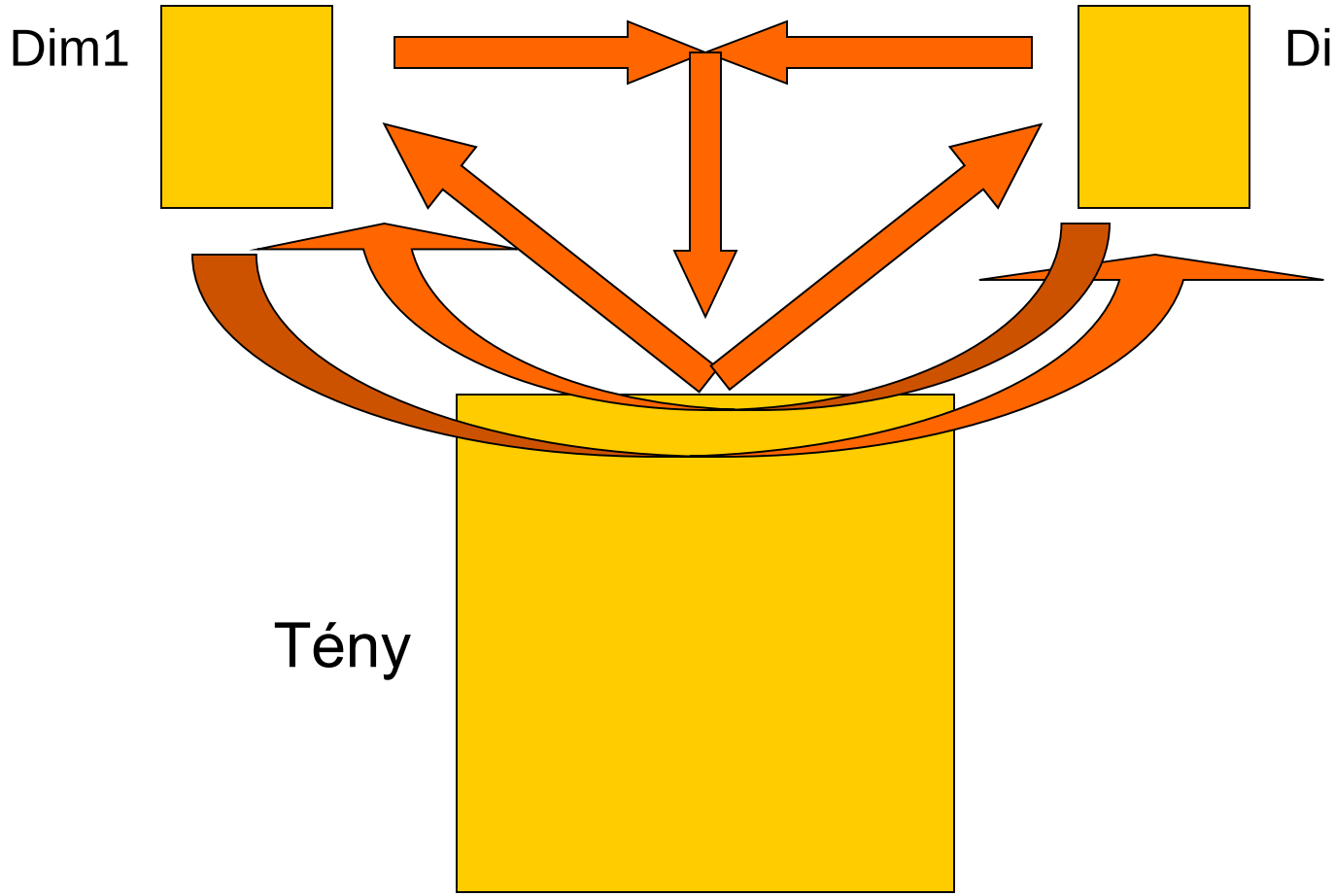
```
SELECT f.*
FROM a,b,f
WHERE a.pk = f.a_fk
AND b.pk = f.b_fk
AND a.t = …  AND b.s = …

A(pk), B(pk)
F(a_fk), F(b_fk)
```

```
>.SELECT STATEMENT
>...NESTED LOOPS
>.....NESTED LOOPS
>.......TABLE ACCESS full b
>.......TABLE ACCESS by rowid fact
>.........INDEX range scan i_fact_b
>.....TABLE ACCESS by rowid a
>.......INDEX unique scan a_pk
```

- Dupla skatulyázott ciklus
  - Válasszunk kezdp táblát (A vagy B)
  - Aztán kövessük az összekapcsolási feltételeket skatulyázott ciklusokkal

Túl bonyolult az ÉS-EGYENLŐ-höz

Hagyományos csillag lekérdezés



Négy lehetséges elérési sorrend!

Hagyományos csillag lekérdezés

```
SELECT f.*
FROM a,b,f
WHERE a.pk = f.a_fk
AND b.pk = f.b_fk
AND a.t = …  AND b.s = …

F(a_fk,b_fk,…)
```

```
>.SELECT STATEMENT
>...NESTED LOOPS
>.....MERGE JOIN cartesian
>.......TABLE ACCESS full a
>.......SORT join
>.........TABLE ACCESS full b
>.....TABLE ACCESS by rowid fact
>.......INDEX range scan I_f_abc
```

- Összefűzött index intervallumkeresés csillag lekérdezéshez
  - Legalább két dimenzió
  - Legalább eggyel több indexelt mező, mint dimenzió
  - Összevonás-Összekapcsolás-Direkt szorzat adja az összes lehetséges dimenziókombinációt
  - Minden kombinációhoz keresünk az összefűzött indexben

Bittérkép index

Empno	Status	Region	Gender	Info
101	single	east	male	bracket_1
102	married	central	female	bracket_4
103	married	west	female	bracket_2
104	divorced	west	male	bracket_4
105	single	central	female	bracket_2
106	married	central	female	bracket_3

REGION='east'	REGION='central'	REGION='west'
1	0	0
0	1	0
0	0	1
0	0	1
0	1	0
0	1	0

Bittérkép index

```
SELECT COUNT(*)
FROM CUSTOMER
WHERE MARITAL_STATUS = 'married‘
AND REGION IN ('central','west');
```

status = 'married'		region = 'central'		region = 'west'						
0		0		0		0		0		0
1		1		0		1		1		1
1	AND	0	OR	1	=	1	AND	1	=	1
0		0		1		0		1		0
0		1		0		0		1		0
1		1		0		1		1		1

Bittérkép elérés, egyetlen tábla

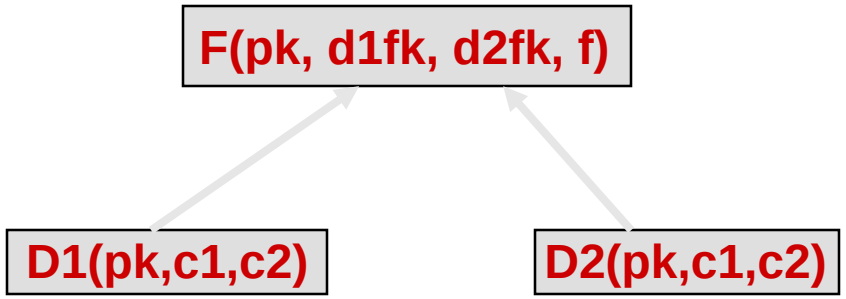
```
SELECT count(*)
FROM customer
WHERE status='M'
AND region in ('C','W');
```

```
>.......TABLE ACCESS (BY INDEX ROWID) cust
>.........BITMAP CONVERSION to rowids
>...........BITMAP AND
>.............BITMAP INDEX single érték cs
>.............BITMAP MERGE
>...............BITMAP KEY ITERATION
>.................BITMAP INDEX range scan cr
```

- Bittérkép ÉS, VAGY és ÁTALAKÍTÁS
  - 'C' és 'W' bitsorozatok megkeresése (bittérképkulcs-iteráció)
  - Logikai VAGY végrehajtása (bittérkép összevonás)
  - Az 'M' bitsorozat megkeresése
  - Logikai ÉS a régió bitsorozattal (bittérkép és)
  - Átalakítás rowid-kké
  - Táblaelérés

Bittérkép elérés, csillag lekérdezés

Bittérkép indexek: id1, id2




```
SELECT sum(f)
FROM F,D1,D2
WHERE F=D1 and F=D2
AND D1.C1=<…>
AND D2.C2=<…>
```

```
>.......TABLE ACCESS (BY INDEX ROWID) f
>.........BITMAP CONVERSION (TO ROWIDS)
>...........BITMAP AND
>.............BITMAP MERGE
>...............BITMAP KEY ITERATION
>.................TABLE ACCESS (FULL) d1
>.................BITMAP INDEX (RANGE SCAN) id1
>.............BITMAP MERGE
>...............BITMAP KEY ITERATION
>.................TABLE ACCESS (FULL) d2
>.................BITMAP INDEX (RANGE SCAN) id2
```

Adattárház tippek

- Csillag lekérdezésre jellemző tippek
  - Star
    - Hagyományos: összevonásos index intervallumkeresés
  - Star_transformation
    - Egymezős bittérkép index összevonás/ÉS-ek
  - Fact(t) / No_fact(t)
    - Segíti a star_transformation-t
  - Index_combine(t i1 i2 …)
    - Explicit megadja, mely indexeket vonja össze/ÉS-elje

ETL lehetőségek

- Új a 9i-ben
  - Külső táblák
    - Külső ASCII fájl elérése SQL-ből (csak FTS)
  - Összevonás (aka UpSert)
    - Feltételes beszúrás vagy frissítés végrehajtása
  - Többtáblás beszúrás (Multi-Table Insert, MTI)
    - Feltételesen beszúrja az allekérdezések eredményét több táblába

Elérhetőség

- Oracle7
  - Költségalapú optimalizáció
  - Hasításos összekapcsolás
- Oracle r8.0
  - Bittérkép indexek (hibamentesen)
  - Star_transformation
  - Rowid formátum (dbms_rowid)
- Oracle 8i
  - Dbms_Stats
- Oracle9i
  - Index SkipScans
  - First_rows(n)-tipp

Egy bevezetés…

- Nem fedtük le:
  - Elosztott SQL
  - Skatulyázott SQL
  - PL/SQL függvények SQL-en belül
  - Ellen-összekapcsolások
  - Nézetek feldolgozása
  - Index+hasító clusterek
  - Partícionálás / Párhuzamosítás
  - Index szervezett táblák
  - …

SQL Tuning: Útirány

- Képes beolvasni tervet
- Képes átírni a tervet 3GL programmá
    - Ismerjük a sorforrás műveleteinket
- Képes beolvasni SQL-t
- Képes átalakítani az SQL-t üzleti lekérdezéssé
    - Ismerjük az adatmodellünket
- Képes megítélni a kimenetelt
    - Ismerjük az üzleti szabályokat / adatstatisztikákat
        - Jobban, mint a CBO
- Szakértők:
    - Optimalizáljuk az SQL-t az SQL írása közben...