Funkcionális függések lekérdezések feldolgozása, kifejezésfák

Takács Gábor
mérnök informatikus, okl. mérnöktanár
takacsg@sze.hu
http://rs1.sze.hu/~takacsg/

Normalizálás

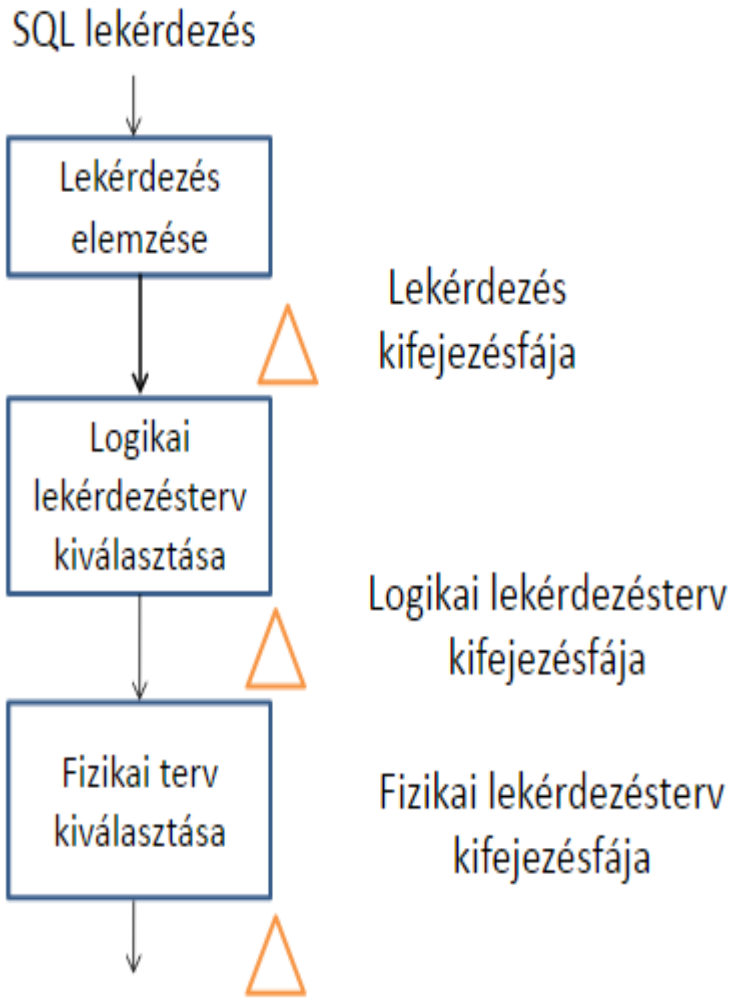
célja anomáliamentes relációséma létrehozása/előállítása.

- **Funkcionális függőség**: ha egy tulajdonságtípus bármely értékéhez egy másik tulajdonságtípus csakis egy értéke rendelhető hozzá. (Pl: személyi szám – név)
- **Kölcsönös funkcionális függőség:** ha az előző feltétel mindkét irányba igaz. (Pl: rendszám – motorszám; 1:1 kapcsolat)
- **Funkcionális függetlenség**: ha az előzőekben ismertetett viszony egyike sem áll fenn. (Pl: lakcím - testmagasság)
- **Tranzitív funkcionális függőség:** ha **egy egyedtípuson belül** egy tulajdonságtípus értékei meghatároznak **egy másik tulajdonságtípus értékeit**, és ezen tulajdonságtípusok **nem képezik a kulcs részét**. (Pl: FEOR kód – szakképzettség
  FEOR 3132 – Számítástechnikai programozó)

FEOR → **Foglalkozások Egységes Osztályozási Rendszere**

Lekérdezések bevezető

Adatbázis műveletek algoritmusai



- A lekérdezésfordítás három legfontosabb lépése:
  - elemzés (parsing): ennek során egy elemző fa készül,
  - lekérdezés átírása (query rewriting): az elemző fából egy kezdeti lekérdezéstervet készítünk, amely a műveleteknek egy algebrai megvalósítását tartalmazza. Ezt a tervet aztán egy olyan ekvivalens lekérdezési tervvé alakítjuk át, amely várhatóan rövidebb idő alatt végrehajtható. Ez a logikai lekérdezésterv (logical query plan).
  - fizikai terv előállítása (physical plan generation). A logikai terv valamennyi operátorának megvalósítására kiválasztunk egy algoritmust és meghatározzuk ezek végrehajtási sorrendjét. A fizikai tervet egy lekérdezésfával ábrázoljuk. Itt olyan részletek is szerepelnek, hogy az egyes relációkhoz miként férünk hozzá, illetve, hogy a relációkat kell-e rendezni, és ha igen, mikor.
- Az utóbbi két lépést lekérdezés optimalizálásnak (query optimization) nevezzük.

Tartalomjegyzék

- **Eddigiek áttekintése**(Lekérdezések, Relációs műveletek)

Ismétlés

Relációs műveletek

- UNIO (∪), METSZET (∩), KÜLÖNBSÉG (/)
- Descartes-szorzat: **R X S** [egyszerű szorzat] → két reláció kombinálása (összes lehetséges módon párosít sorokat)
- Projekció (vetítés) : $\pi_L(R)$ → oszlopok kiválasztása (**R** bizonyos oszlopait tartalmazza valamilyen sorrendben.)
- Szelekció (kiválasztás): $\sigma_F(R)$ → sorok kiválasztása (**R** reláció minden sorára megvizsgálja, hogy az **F** feltétel teljesül-e)
- Illeszkedés (természetes összekapcsolás): R ⋈ S (Descatres-szorzat + feltétel szerinti kiválasztás + vetítés)

$$\pi_{A, B, .., H, K, ..} (\sigma_{R.A=S.A} (R \times S))$$

- Csoportosítás: $\gamma_{A, SUM(B*C)->X}(R)$
- Rendezés $\tau_{A1, ..., An}(R)$.

Lekérdező nyelvek: adatok elérését teszik lehetővé DB-ból.

Lekérdezésekkel (SQL utasítások) információkat nyerhetünk az adatbázisból, számítások hajthatók végre, egyesíthetünk, hozzáadhatunk, vagy módosíthatunk táblaadatokat

SQL utasítások | SQL = Structured Query Language (struktúrált lekérdező nyelv).

Négy utasításcsoportot különböztetünk meg:

- **Adatdefiníciós utasítások** (Data Definition Language – DDL), amelyek objektumok létrehozására, módosítására, törlésére valók.
- **Adatmanipulációs utasítások** (Data Manipulation Language – DML), amelyek rekordok felvitelére, módosítására és törlésére alkalmazhatók.
- **Adatkezelő utasítások** (Data Query Language – DQL), amelyekkel a letárolt adatokat tudjuk visszakeresni.
- **Adatvezérlő utasítások** (Data Control Language – DCL), amelyekkel az adatvédelmi és a tranzakció-kezelő műveletek végrehajthatóak.

Adatdefiníciós utasítások (DDL)

- **Adattáblák létrehozása**

  CREATE TABLE táblanév
  (oszlopnév adattípus [feltétel],
  ...,
  oszlopnév adattípus [feltétel]
  [, táblaFeltételek]
  );

- **Oszlopfeltételek (egy adott oszlopra vonatkoznak)**:

  NOT NULL | NULL

  PRIMARY KEY: elsődleges kulcs

  UNIQUE: kulcs

  REFERENCES tábla(oszlop) [ON-feltételek]: külső kulcs

- **Táblafeltételek (az egész táblára vonatkoznak):**

  PRIMARY KEY (oszloplista): elsődleges kulcs

  UNIQUE (oszloplista): kulcs

  FOREIGN KEY (oszloplista) REFERENCES tábla(oszloplista) [ON-feltételek]: külső kulcs

- **Relációséma módosítása**

  ALTER TABLE táblanév
  [ADD (újelem, ..., újelem)]
  [MODIFY (módosítás, ..., módosítás)]
  [DROP (oszlop, ..., oszlop)];

A példa táblák létrehozása

Auto(rsz, tipus, szin, ar, evj, tul)

Tulaj(szazon, név, cím)

```
create table tulaj (
szazon text(11) primary key,
nev text(30),
cím text(40);

create table auto(
rsz text(7) primary key,
tipus text(20),
szin text(20),
ar money,
evj decimal,
tul references tulaj(szazon);
```

Adatmanipulációs utasítások (DML)

- **A táblába új sor felvétele**

  INSERT INTO táblanév [(oszloplista)]
  VALUES (értéklista);

  Példa:
  insert into tulaj
  values (27110142233, Kovács Ákos, Szerencs Sugár út 24);

  insert into auto
  values (abc-123, Trabi, kék, 130000, 1990, 27110142233);

- **Sor(ok) módosítása**

  UPDATE táblanév
  SET oszlop = kifejezés, ..., oszlop = kifejezés
  [WHERE feltétel];

- **Sor(ok) törlése**

  DELETE FROM táblanév
  [WHERE feltétel];

Adatlekérdező utasítás (DQL)

A lekérdező nyelv egyetlen utasításból áll, mely számos alparancsot tartalmazhat, és a lekérdező utasítások többszörös mélységben egymásba ágyazhatók. Arra használjuk, hogy egy vagy több adathalmazból (reláció) egy adathalmazt állítsunk elő. A bemeneti adatokon, a relációs algebra műveleteit hajtjuk végre, és kapunk egy eredmény táblát.

- **A SELECT utasítás az alábbi alparancsokból állhat:**

  SELECT [DISTINCT] oszloplista --> **projekció** (DISTINCT= azonos sorokból csak egyet jelenítsünk meg.)

  FROM táblanévlista --> **Descartes-szorzat**

  [WHERE feltétel] --> **szelekció**

  [GROUP BY oszloplista] --> **csoportosítás** (tábla sorait csoportonként szeretnénk összesíteni)

  [HAVING feltétel] --> **csoport-feltétel** (csak a feltételnek eleget tevő csoportok kerülnek összesítésre)

  [ORDER BY oszloplista]; --> **rendezés** (rendezett megjelenés: def: növekvő, [DESC]: csökkenő)

- **Megoldási sorrend a fentiek szerint, de végrehajtásuk az alábbiak szerint történik**

  1. FROM --> Descartes-szorzat
  2. WHERE --> szelekció
  3. GROUP BY --> csoportosítás
  4. HAVING --> csoport-szelekció
  5. SELECT --> projekció
  6. ORDER BY --> rendezés

Szelekció:

SELECT * FROM T WHERE feltétel;

Példa:

SELECT * FROM auto ar > 200000;

LEKÉRDEZÉS példák:

SELECT name, imdb_rating FROM movies;

Database Schema

movies	220 rows
id	INTEGER
name	TEXT
genre	TEXT
year	INTEGER
imdb_rating	REAL

Query Results

name	imdb_rating
Avatar	7.9
Jurassic World	7.3
The Avengers	8.1
The Dark Knight	9.0

SELECT * FROM movies WHERE imdb_rating > 8;

Database Schema

movies	220 rows
id	INTEGER
name	TEXT
genre	TEXT
year	INTEGER
imdb_rating	REAL

Query Results

id	name	genre	year	imdb_rating
3	The Avengers	action	2012	8.1
4	The Dark Knight	action	2008	9.0
6	Star Wars	action	1977	8.7
8	The Dark Knight	action	2012	8.5

Forrás: https://www.codecademy.com/

LEKÉRDEZÉS példák:

SELECT * FROM movies
WHERE name LIKE 'Se_en';

Query Results

id	name	genre	year	imdb_rating
219	Se7en	drama	1995	8.6
220	Seven	drama	1979	6.1

SELECT * FROM movies
WHERE name LIKE 'a%';

Query Results

id	name	genre	year	imdb_rating
1	Avatar	action	2009	7.9
7	Avengers: Age of Ultron	action	2015	7.9
16	American Sniper	action	2014	7.4
79	Alvin and the Chipmunks: The Squeakquel	comedy	2009	4.4
82	Aladdin	comedy	1992	8.0
83	Alvin and the Chipmunks	comedy	2007	5.3
87	Austin Powers in	comedy	2002	6.2

SELECT * FROM movies
WHERE year BETWEEN 1990 and 2000
AND genre = 'comedy';

Query Results

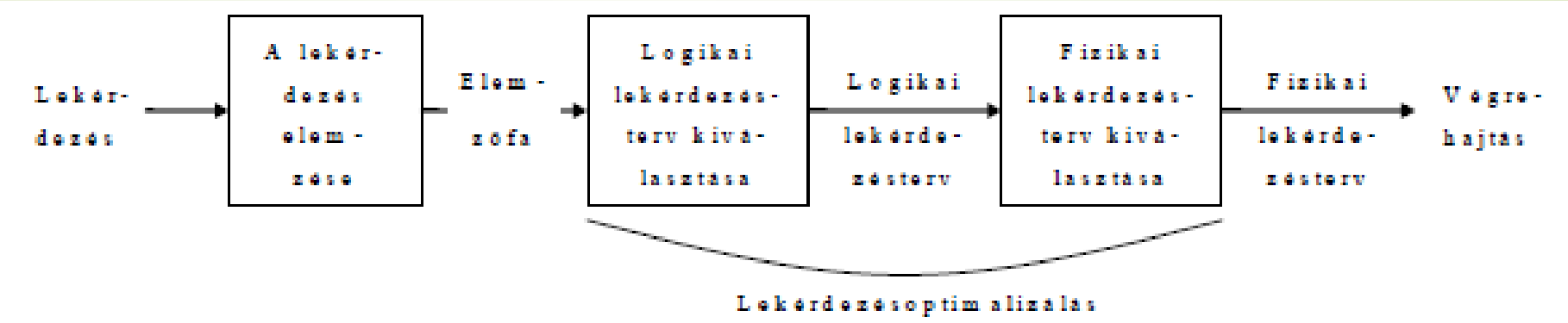
id	name	genre	year	imdb_rating
61	Home Alone	comedy	1990	7.4
66	How the Grinch Stole Christmas	comedy	2000	6.0
69	Men in Black	comedy	1997	7.2
70	Toy Story 2	comedy	1999	7.9
80	Mrs. Doubtfire	comedy	1993	6.9
82	Aladdin	comedy	1992	8.0

Forrás: https://www.codecademy.com/

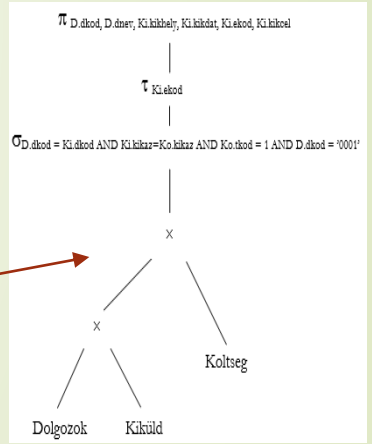
Tartalomjegyzék

- **Ismétlés** (Lekérdezések, Relációs műveletek)
- **Lekérdezések feldolgozása**
  - **FORDÍTÁS**

Lekérdezés feldolgozás → FORDÍTÁS



- A felhasználó lekérdezéseit, adatmódosító utasításait a **lekérdezés feldolgozó** *lefordítja* adatbázis-műveletekre, és végre is hajtja a műveleteket.
- A lekérdezés fordítás három fontos lépése:
  - **Elemzés**: a lekérdezést és annak szerkezetét jellemző ***elemzést készít***
  - **Lekérdezés átírás/optimalizálás**:
    az elemzést átkonvertálja → (**kezdeti) logikai lekérdezési terv**vé.
    Rendszerint *algebrai* megvalósítás.
  - **Fizikai terv előállítása**: a *logikai lekérdezés tervet* → ***fizikai lekérdezés-terv***vé.
    operátorokhoz algoritmust rendel, és meghatározza végrehajtási sorrendet.



A fizikai terv a végrehajtáshoz tartozó információkat is tartalmaz
pl.: a relációkhoz történő hozzáférés, relációt kell-e rendezni, stb.

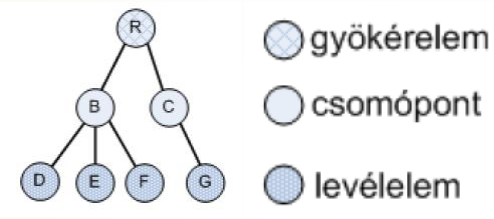
Tartalomjegyzék

- **Ismétlés** (Lekérdezések, Relációs műveletek)
- **Lekérdezések feldolgozása**
  - **FORDÍTÁS**
  - **Kifejezésfa**

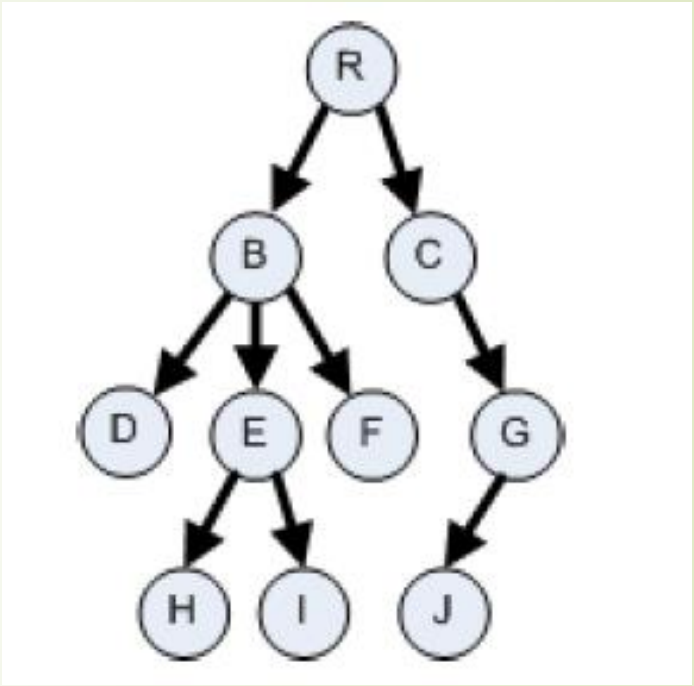
    Egy lekérdezéshez tartozó **különböző algebrai kifejezéseket** **logikai lekérdezések**nek nevezzük, melyeket kifejezésfákkal ábrázolunk.

Fa → ALAPOK

- A fa adatszerkezet egy speciális jellemzőkkel rendelkező ***irányított gráf***. Akkor mondhatjuk, hogy egy gráf fa, ha **körmentes** és **összefüggő.**



- Útvonalak: gyökér elemből minden más csúcshoz vezet út, méghozzá pontosan egy út – nincsenek alternatív útvonalak
- Csúcsokhoz rendeljük az értékeket (R,B,C,E,E,F,G), és nem az élekhez.
- Az informatikában a fák egy speciális változata terjedt el: a bináris fa. A **bináris fa** legfontosabb jellemzője, hogy a csúcsoknak maximum két gyermek elemük lehet.



Kifejezésfa → ALAPOK

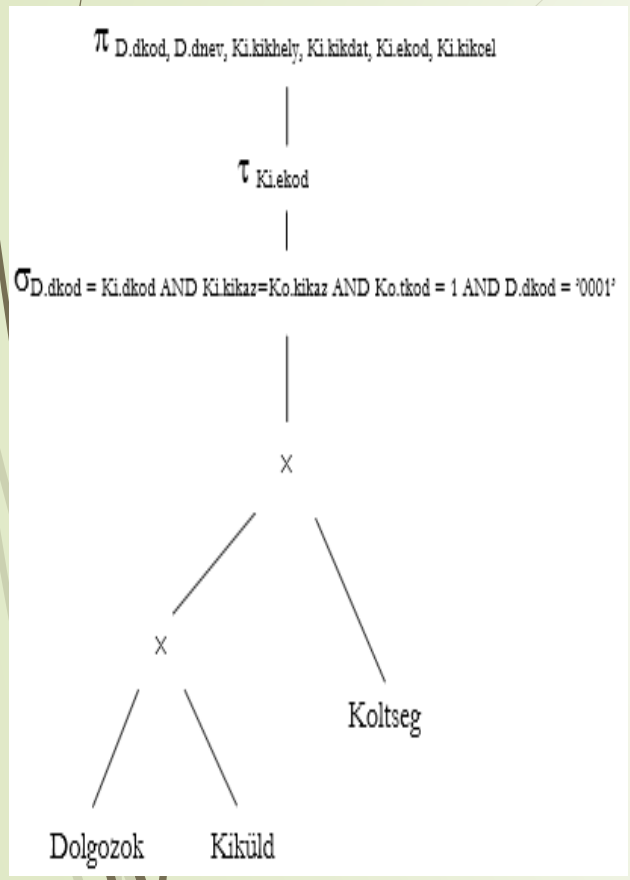
- A **zárójelezés és az operátor-precedencia** kiértékelése:

  A numerikus kifejezések esetén szokásos használni:

  1. meg kell keresni a legbelső zárójelek közé zárt részkifejezést, majd
  2. meg kell keresni a legmagasabb precedenciájú operátorokat, végül
  3. a kötési iránynak megfelelően meg kell keresni az első kiértékelendő operátort.

- **Hogyan építsünk, és alkalmazzuk a kifejezésfát?**
  Olyan részfákat keresünk, melyek csak konkrét számadatokat és egyetlen operátort tartalmaznak. Ekkor alkalmazzuk az operátort a két adatra, töröljük a részfát, és a kapott értéket az operátor csomópontjának helyére beszúrjuk. Pl.: *8/2+(5+7)*8 kifejezés fája*

Kifejezésfa (lekérdezés esetén)



- Lekérdezés esetén az operátorok egymás utáni alkalmazását egy *kifejezésfa* formájában rajzolhatjuk fel. **A fa leveleit relációk nevei alkotják, a csomópontokon pedig az attribútumokon végzett relációs operátorok (relációs műveletek) .**
- **Példa**: KIKÜLDETÉS adatbázisunk 4 relációja

  Dolgozok (dkod, dnev, dirsz, dtelep, dutca)

  Kikuld (kikaz, *dkod*, kikhely, kikdat, ekod, kikcel)

  KtgTipus (tkod, tnev)

  Koltseg (*kikaz*, *tkod*, km, osszeg)

- **Feladat:** Készítsük el a kifejezésfát az alábbi lekérdezéshez:
  Jelenítsük meg a '0001' azonosítóval bíró dolgozó közlekedési költségű (tkod = 1) kiküldetéseit, közlekedési eszköz rendezettségben!

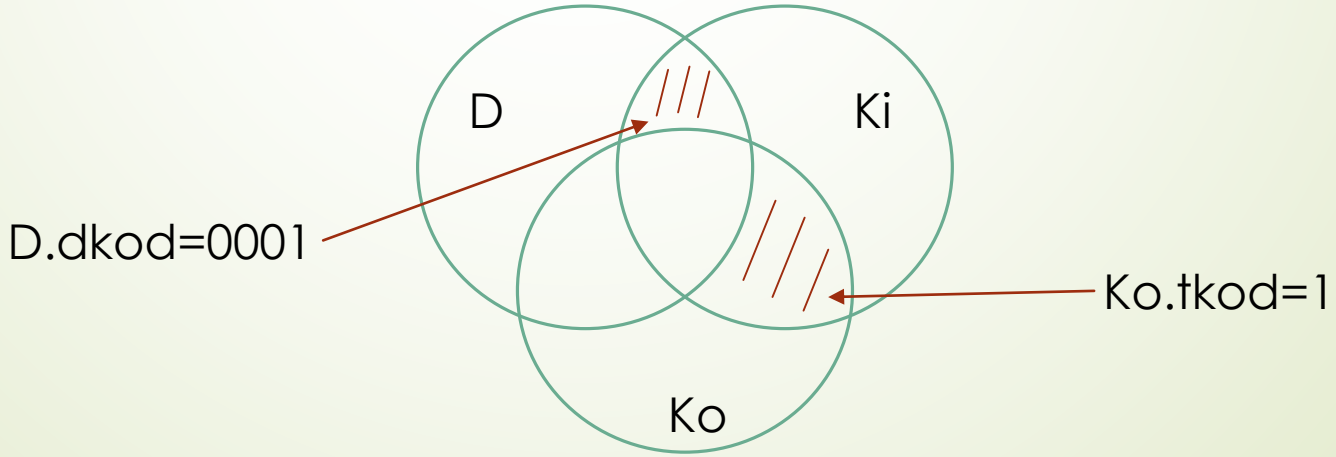
```
SELECT D.dkod, D.dnev, Ki.kikhely, Ki.kikdat, Ki.ekod, Ki.kikcel
FROM Dolgozok D, Kikuld Ki, Koltseg Ko
WHERE D.dkod = Ki.dkod AND Ki.kikaz=Ko.kikaz AND
    D.dkod = '0001' AND Ko.tkod = 1
ORDER BY Ki.ekod;
```

Kifejezésfa

```
SELECT D.dkod, D.dnev, Ki.kikhely, Ki.kikdat, Ki.ekod, Ki.kikcel
        FROM Dolgozok D, Kikuld Ki, Koltseg Ko
        WHERE D.dkod = Ki.dkod AND Ki.kikaz=Ko.kikaz AND D.dkod = '0001' AND Ko.tkod = 1
        ORDER BY Ki.ekod;
```

- A lekérdezés fordítás három fontos lépése:
  Elemzés készül → (kezdeti) logikai lekérdezési terv → *fizikai lekérdezés-terv*vé.

  Első lépés: a FROM utáni relációk összekapcsolása a Descartes-szorzat operátorral.
  Második lépés: a WHERE záradéknak megfelelő kiválasztás végrehajtása.
  Harmadik lépés: a rendezés végrehajtása (sorrend véglegesíts).
  Negyedik lépés: vetítés (megjelenítés) a SELECT záradékban szereplő listára.



A relációalgebrai operátorok precedencia sorrendje(zárójelben az SQL-beli megfelelőikkel):

- vetítés vagy projekció (SELECT);
- szorzat (FROM);
- kiválasztás vagy szelekció (WHERE, HAVING);
- összekapcsolás
  (JOIN, NATURAL JOIN, OUTER JOIN vagy FROM+WHERE);
- egyesítés vagy unió, metszet, különbség
  (UNION, INTERSECT, EXCEPT);
- ismétlődések kiküszöbölése (DISTINCT);
- csoportosítás (GROUP BY);
- rendezés (ORDER BY).

Elemzés készül → **(kezdeti) logikai lekérdezési terv** → *fizikai lekérdezés-terv*vé

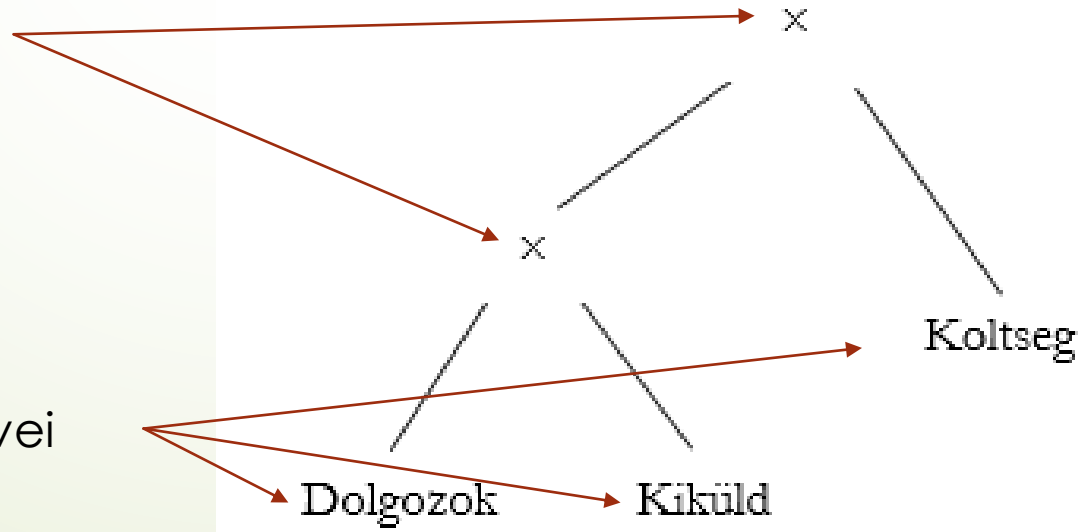
Projekció → π D.dkod, D.dnev, Ki.kikhely, Ki.kikdat, Ki.ekod, Ki.kikcel

Rendezés → τ Ki.ekod

Szelekció → σ D.dkod = Ki.dkod AND Ki.kikaz=Ko.kikaz AND Ko.tkod = 1 AND D.dkod = '0001'

Csomópontokon: relációs operátorok

Leveleken: relációk nevei



Módosított (optimalizált) lekérdezési terv

Projekció

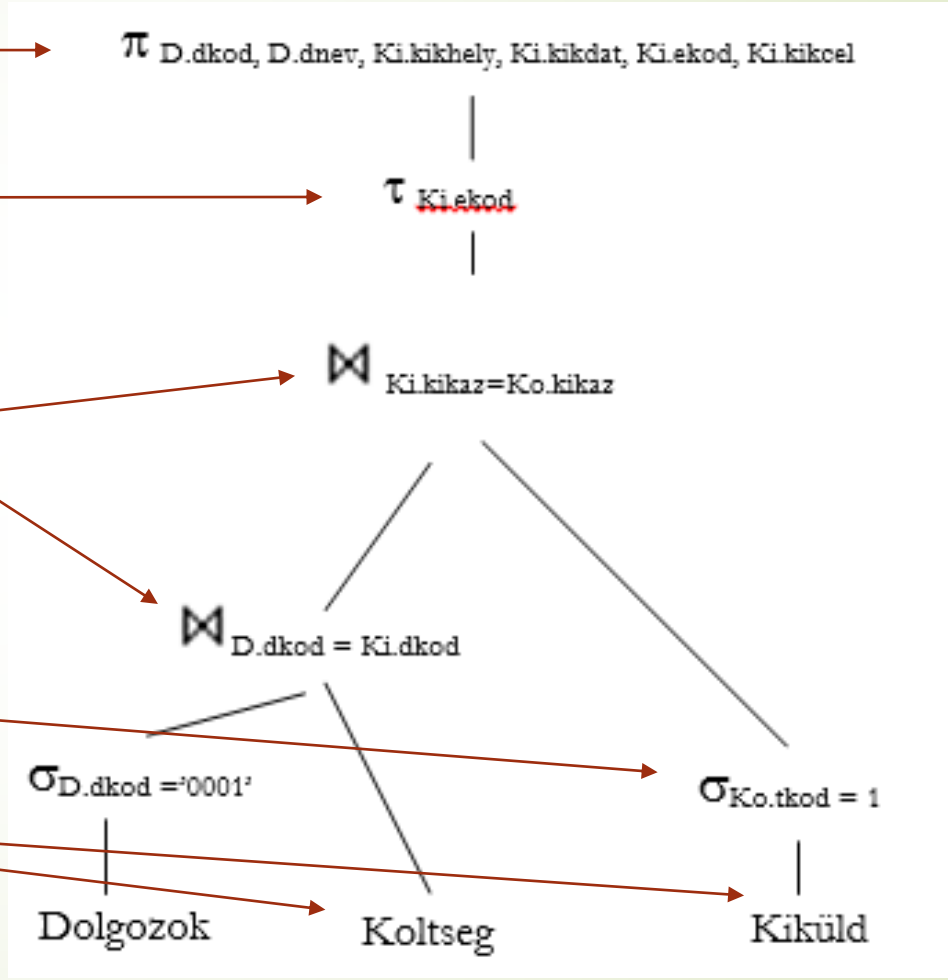
Rendezés

Illeszkedés:

Descatres-szorzat +
feltétel szerinti kiválasztás +
vetítés

Szelekció

Relációk



Mire optimalizálunk? → Általában válaszidőre. (I/O, CPU, Memória használat)

Tartalomjegyzék

Ekvivalens kifejezések

Lekérdezés-válaszoló rendszer

Ekvivalens kifejezések:

olyan kifejezések, amelyeket ha ugyanazokon a relációkon értékelünk ki, ugyanazt az eredményt adják

Minden adatbázisrendszernek *van* egy lekérdezés-válaszoló rendszere.

Egy felhasználó által megfogalmazott **lekérdezésnek létezhet több *ekvivalens kifejezése***, ezek között lehetnek olyanok, **amelyek gyorsabban kiértékelhetők**.

A lekérdezés-válaszolónak egyik fontos feladata, hogy egy relációs algebrai kifejezést olyan ekvivalens kifejezéssel helyettesítsen, amely **hatékonyabban értékelhető ki**.

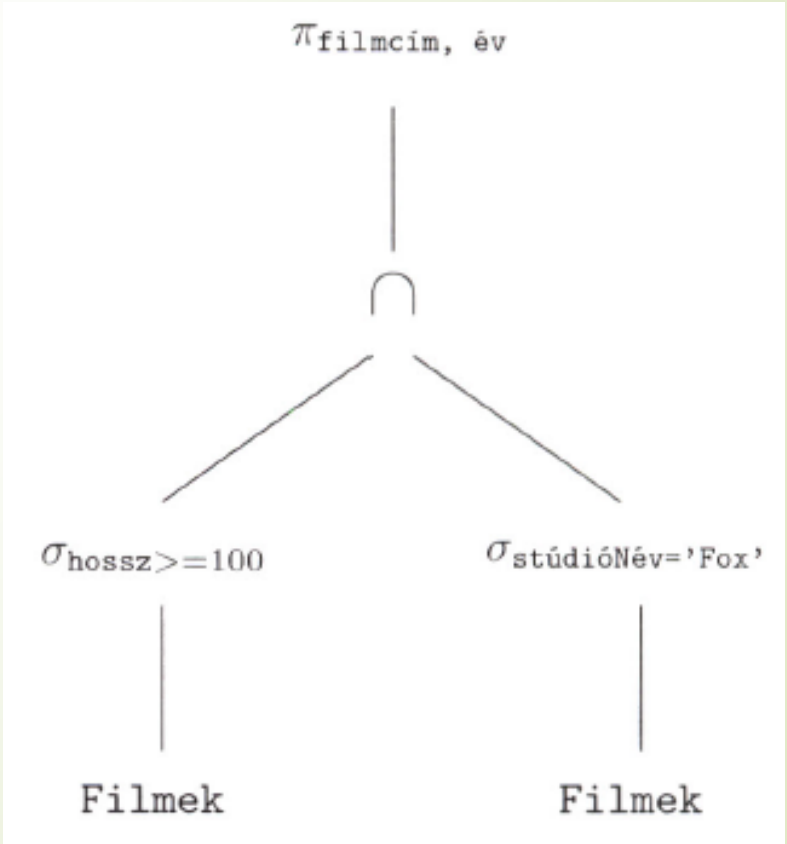
Ekvivalens kifejezések (a Filmek relación)

filmcím	*év*	*hossz*	*műfaj*	*stúdióNév*	*producerA.*
Csillagok háborúja	1977	124	sci-fi	Fox	12345
Galaktitkos küldetés	1999	104	vígjáték	DreamWorks	67890
Wayne világa	1992	95	vígjáték	Paramount	99999

Kérdés: "Melyek a Fox stúdióban készült, legalább 100 perc hosszúságú filmek, és ezek mikor készültek?"

Lépések:

1. Kiválasztjuk a **Filmek** relációból azokat a sorokat, amelyekre hossz >= 100.
2. Kiválasztjuk a **Filmek** relációból azokat a sorokat, amelyekre a stúdióNév = 'Fox'.
3. Kiszámoljuk az 1. és 2. metszetét.
4. A 3. lépésben megkapott relációt levetítjük a **filmcím** és **év** attribútumokra



Ekvivalens kifejezések

Kérdés: "Melyek a Fox stúdióban készült, legalább 100 perc hosszúságú filmek, és ezek mikor készültek?"

A korábban kifejezésfában meghatározott kifejezést felírhatjuk hagyományos lineáris jelöléssel is:

$$\pi_{\text{filmcim, év}} (\sigma_{\text{hossz>=100}}(\text{Filmek}) \cap \sigma_{\text{studioNev='FOX'}}(\text{FILMEK}))$$

Egyébként gyakori, hogy több relációs algebrai kifejezésnek is ugyanaz az eredménye. Például a fenti lekérdezés felírható egyetlen kiválasztás használatával, ha a metszetet az AND operátorral helyettesítjük.

$$\pi_{\text{filmcim, év}} (\sigma_{\text{hossz>=100}}(\text{Filmek}) \text{ AND } \sigma_{\text{studioNev='FOX'}}(\text{FILMEK}))$$

filmcím	*év*	*hossz*	*műfaj*	*stúdióNév*	*producerA.*
Csillagok háborúja	1977	124	sci-fi	Fox	12345
Galaktitkos küldetés	1999	104	vígjáték	DreamWorks	67890
Wayne világa	1992	95	vígjáték	Paramount	99999

Tartalomjegyzék

Algebrai szabályok lekérdezési tervek javítására

Kommunikatív és asszociatív szabályok

- Egy operátorra vonatkozó *kommutatív szabály* azt mondja ki, **hogy nem számít, hogy milyen sorrendben adjuk meg az operátor argumentumait, az eredmény ugyanaz lesz.** Pl. a + és × művelet kommutatív, mert x + y = y + x és x × y = y × x tetszőleges x és y esetén.
- Egy operátorra vonatkozó *asszociatív szabály* azt mondja ki, hogyha az **operátort kétszer használjuk, akkor egyaránt csoportosíthatunk balról vagy jobbról.** A + és x például asszociatív aritmetikai operátorok, ami azt jelenti, hogy (x + y) + z = x + (y + z) és (x × y) × z = x × (y × z).

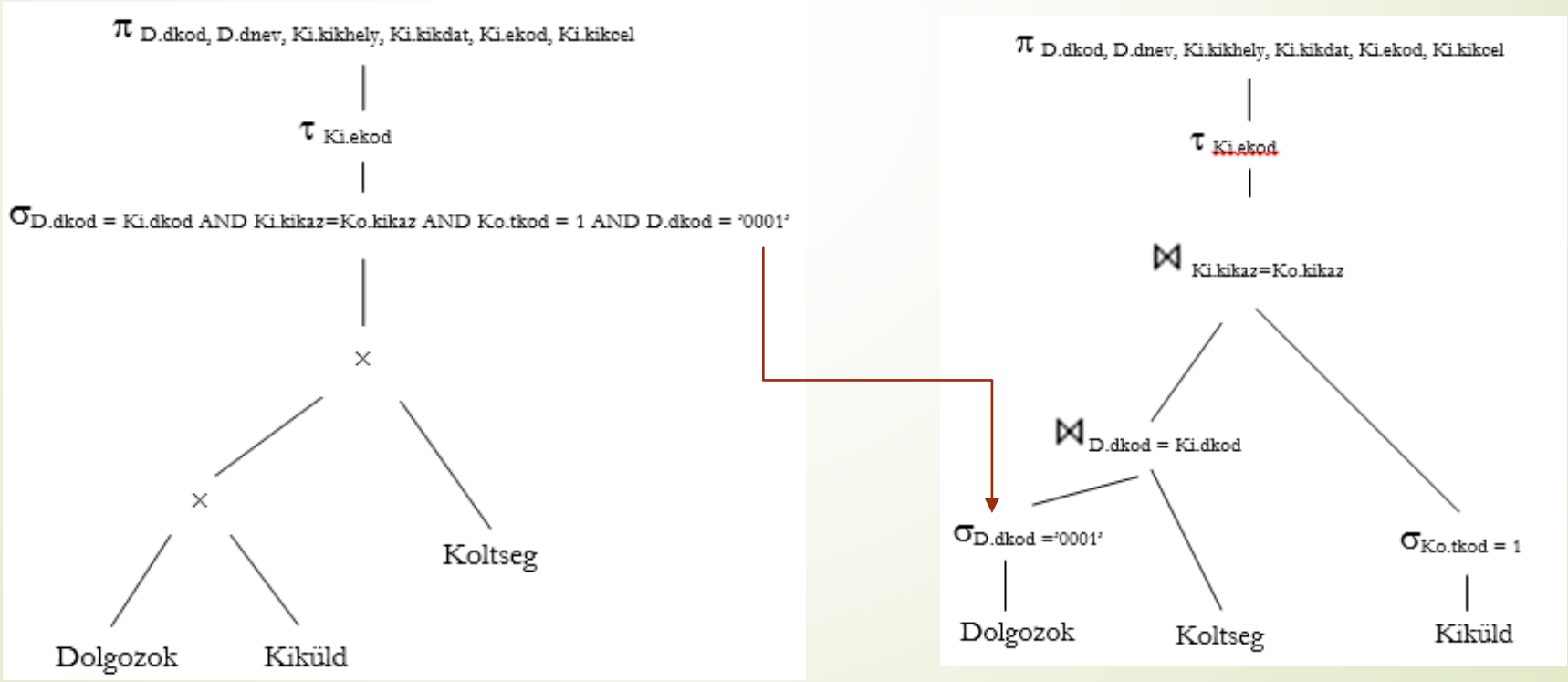
A relációs algebra néhány operátora **egyszerre** *kommutatív* és *asszociatív*:

	Kommunikativítás	asszociativitás
Descatres-szorzat	$R \times S = S \times R$;	$(R \times S) \times T = R \times (S \times T)$,
Illeszkedés	$R \bowtie S = S \bowtie R$;	$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$,
Unió	$R \cup S = S \cup R$;	$(R \cup S) \cup T = R \cup (S \cup T)$,
Metszet	$R \cap S = S \cap R$;	$(R \cap S) \cap T = R \cap (S \cap T)$.

Algebrai szabályok lekérdezési tervek javítására

Kiválasztással kapcsolatos szabályok

- A kiválasztások **lényegesen csökkenthetik** a *relációk méretét*, ezért az optimális lekérdezés eléréséhez a kiválasztásoknak a kifejezésfa aljában kell elhelyezkedniük. Ha egy kiválasztás feltétele összetett (azaz AND vagy OR által összekapcsolt feltételekből áll), akkor a feltételt szétvágjuk.
  A kiválasztásra vonatkozó első két szabályt *szétvágási szabálynak* nevezzük:



Algebrai szabályok lekérdezési tervek javítására

Kiválasztással kapcsolatos szabályok

Háromféle szabály van, attól függően, hogy opcionális vagy kötelező a kiválasztást az egyes argumentumokhoz odavinni:

1. ***Egyesítés* esetén (∪)** a kiválasztást mindkét argumentumra alkalmazni kell.
   $\sigma_F(R \cup S) = \sigma_F(R) \cup \sigma_F(S)$ Itt kötelezően le kell vinni a kiválasztást a fa mindkét ágán.
2. ***Különbség* esetén (\)**a kiválasztást az első argumentumra alkalmazni kell, a másodikra pedig lehet. (R [hallgatók] azon sorai, ami S-ben [jegyek] nem fordul elő)
   $\sigma_F(R \setminus S) = \sigma_F(R) \setminus S$ vagy $\sigma_F(R \setminus S) = \sigma_F(R) \setminus \sigma_F(S)$
3. **A többi operátor esetében (**x ; ⋈ ; ∩ **)** csak azt követeljük meg, hogy a kiválasztást egy argumentumra alkalmazzuk.

   $\sigma_F(R \times S) = \sigma_F(R) \times S$ (D-szorzat)

   $\sigma_F(R \bowtie S) = \sigma_F(R) \bowtie S$ (Illeszkedés)

   $\sigma_F(R \cap S) = \sigma_F(R) \cap S$ (Metszet)

Tartalomjegyzék

- **Ismétlés** (Lekérdezések, Relációs műveletek)
- **Lekérdezések feldolgozás**
  - **FORDÍTÁS**
  - **Kifejezésfa**
  - **Ekvivalens kifejezések**
  - **Algebrai szabályok lekérdezési tervek javítására**
  - **Fizikai lekérdezés terv-operátorok**
    Elemzést készítünk → (kezdeti) logikai lekérdezési terv → **fizikai lekérdezés-terv**

Fizikai lekérdezés-tervek

A fizikai lekérdezés-tervek **operátorokból épülnek fel**, melyek mindegyike a terv egy lépését reprezentálja. A folyamat során azonban szükség van olyan operátorokra, amik **nem kapcsolhatók a relációs algebrai műveletekhez**. Például:

Tábla ”beolvasása” memóriába.

A fizikai lekérdezések ezen kiegészítő elemeket is tartalmazzák.

Fizikai lekérdezés-terv operátorok

Táblák átvizsgálása

A lekérdezés-tervek megvalósítását nagyban segítik azok az operátorok, melyek különböző lekérdezés-lépéseket megvalósítva képesek például egy táblát (általában annak összes sorát) beolvasni a memóriába.

Két módon találhatjuk meg az **R** reláció megfelelő sorait:

- **Tábla alapú átvizsgálás:**
  Az R reláció sorait Blokkonként tároljuk, majd így lehetséges a beolvasásuk a másodlagos memóriából. A Blokkok ismertek a rendszer számára.

- ***Index alapú átvizsgálás***:
  Ha a létezik egy index az **R** valamelyik attribútumára, akkor használhatjuk ezt az indexet az **R** összes sorának a beolvasásához, még akkor is ha konkrét értéket, vagy intervallumot keresünk.

Fizikai lekérdezési terv-operátorok

Rendezés a táblák átvizsgálásakor

Több ok is lehet, amiért egy táblát rendezni szeretnénk:

- **Lehet a lekérdezésnek ORDER By záradéka**
- **A relációs algebrai műveletek implemetálására szolgáló algoritmus megköveteli**

A *rendezéses átvizsgálás* nevű fizikai lekérdezésterv-operátor:
veszi az **R** relációt azon attribútumok specifikációjával együtt, amelyeken el kell végezni a rendezést, és előállítja a rendezett **R** relációt.
A rendezéses átvizsgálás megvalósítására több lehetőség is létezik:

- ***Létező index bejárása*** lehetővé teszi a rendezett **R** reláció előállítását.
- Ha a rendezni kívánt **R** reláció elég kicsi ahhoz, hogy beférjen a memóriába, akkor táblaátvizsgálással, vagy indexátvisgálással kinyerhetjük a tábla sorait, és utána **választhatunk egyet a hatékony *memóriában rendező algoritmusok* közül**.
- Ha az **R** túl nagy ahhoz, hogy beférjen a memóriába, akkor jöhet a *többmenetes összefésülés*.

Fizikai lekérdezési terv-operátorok

Fizikai operátorok költségbecslése

Egy művelet költségének méréséhez a lemez I/O-műveleteinek számát használjuk.

Oka, hogy a lemezről történő beolvasás hosszabb, mint bármilyen művelet elvégzése a memóriában.

Feltételezzük, hogy egy tetszőleges operátor argumentumai a lemezen találhatók, az eredmény a memóriában marad. A kimeneten nem számolunk lemez I/O-költséget, mert az vagy nulla, vagy attól függ, hogy egy általunk ismeretlen alkalmazás mit tesz az adatokkal. (ugyanis az eredmény továbbadódik más programoknak, vagy alkalmazásnak.)

Csővezetékek módszer

Az eredmény a memóriában épül fel, és ott is adódik tovább. Az eredményt sosem írjuk ki a lemezre, így megspóroljuk annak visszaolvasási költségét.

Fizikai lekérdezési terv-operátorok

Fizikai operátorok költségbecslése

Költségbecslés paraméterei:

Paraméterek, amik a reláció adatainak méretét, és eloszlását becsülik meg.
A rendszer időnként újraszámolja őket, hogy segítse a lekérdezés optimalizálót.

Paraméter	Leírás
M	A rendelkezésre álló memória pufferek száma. (CSAK a bement, és a közbeeső operátorok eredményeinek tárolására szolgál.)
B(R) vagy B	Az R reláció hány blokkban fér el. (közelítő blokkszám)
T(R) vagy T T/B hányados	Az R reláció sorainak száma. R hány sora fér el egy blokkban
V(R, A)	Az R reláció A oszlopban található különböző értékek száma.

Költség	Leírás
B	Ha **R** befér a memóriába.
3B	Ha az **R** nyalábolt, de *kétfázisú többmenetes összefésüléses* rendezést igényel.
T + 2B	Ha az **R** nem nyalábolt, és kétfázisú rendezést igényel, akkor kezdetben **T** darab lemez I/O-műveletre van szükség a részcsoportok beolvasásához. A részlistákat azonban már tárolhatjuk (és később be is olvashatjuk) **nyalábolt** formában, így ez a lépés csak **2B** lemez I/O-műveletet igényel.

Fizikai lekérdezési terv-operátorok

Fizikai operátorok megvalósításához használatos iterátorok (gyorsabb adatfeldolgozás)

- Több fizikai operátor megvalósítható iterátorként, ami nem más, mint 3 függvény olyan együttese, amely lehetővé teszi, hogy az eredményt kérő soronként kapja az adatokat.
- Egy művelet iterátorát felépítő 3 függvény a következő:

1. **OPEN függvény** : Elindítja a sorok kinyerésének folyamatát
2. **GetNext függvény**: Visszaadja az eredmény következő sorát, és előkészíti következő sor beolvasását, amit ha szükséges, meg is tesz.
3. **Close függvény**: Befejezi az iterálást, miután végzett az összes sorral.

R reláció iterátor függvényeinek jelölései:

R.Open() ; R.GetNext() ; R.Close()

Iterátorok használatával egyszerre több művelet is aktív, ez csökkenti a tárolási szükségletet.

Tartalomjegyzék

- **Ismétlés** (Lekérdezések, Relációs műveletek)
- **Lekérdezések feldolgozás**
- **FORRÁS**
  - **Kifejezésfa**
  - **Ekvivalens kifejezések**
  - **Algebrai szabályok lekérdezési tervek javítására**
  - **Fizikai lekérdezés terv-operátorok**

    Elemzést készítünk → (kezdeti) logikai lekérdezési terv → **fizikai lekérdezés-terv**

Köszönöm a figyelmet!