

Adatvezérelt rendszerek

Lekérdezés optimalizálás



Automatizálási és
Alkalmazott
Informatikai Tanszék

Mi a lekérdezés optimalizálás célja?



Optimalizálás alapelvek

- Statisztikák alapján értékkel
 - > Költség = Válaszidő (CPU + I/O idő)
- Triviális terv
 - > Egyszerűbb lekérdezéshez egyértelműen generálható
 - > Szabály alapú
- Ha nem készíthető triviális terv
 - > Összetett lekérdezések
 - > 3 fázisú optimalizáló

Fizikai terv

- Fizikai terv elemei
 - > Relációt beolvasó operátorok
 - Logikai terv levél eleminek beolvasása
 - > Relációs algebrai műveletet végrehajtó operátor
- Tervek készítése
 - > Szabály alapú
 - > Költségbecslés alapú
 - Tábla elérési módok
 - Join operátorok megvalósítási módjai
 - Join sorrend

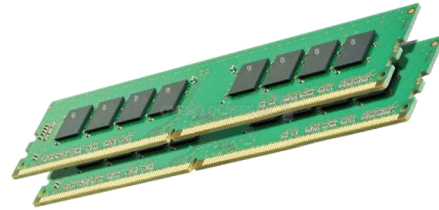
Jó tanácsok – 1

- Statisztikák legyenek naprakészek
 - > Elavult statisztika → rossz végrehajtási terv
 - > (Automatikus, ha csak nem kapcsoljuk ki stb.)
- Lekérdezés struktúrája
 - > SQL deklaratív környezet
 - Gondoljunk procedurálisan is!
 - > Többféleképp is megfogalmazható ugyanaz
 - > Törekedjünk az egyszerűsége
 - > Kerüljük a select * -ot
 - > Jó struktúrával sokat lehet nyerni
 - Csak ezután kísérletezzünk a hintek használatával

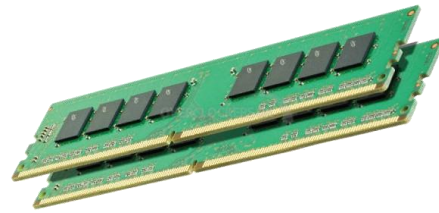
MongoDB indexek

- Index „csak” keresési célt szolgál
- Index típusok
 - > Egyszerű & összetett
 - > Unique index
 - primary key jellegű attribútumot így lehet garantálni
 - > Tömbök tartalmát is indexeli
 - > Beágyazott dokumentumokat is indexeli
 - > TTL, Geospatial, full text
- Indexet létre kell hozni
 - > Kivéve: _id-ra unique

Mi a lekérdezés optimalizálás célja?



Mi a lekérdezés optimalizálás célja?



Válaszidőt befolyásoló tényezők

- I/O költség
 - > Adatbázisokban meghatározó
 - > Moore törvény nem igaz rá
 - > Speciális kezelési módok
 - > Írás, olvasás, pozicionálás (seek)
- CPU használat
 - > Komplex lekérdezések
 - > Összetett számítások
- Memória használat
 - > Cache hatás

Rövid történelem

- A '70-es évek: „sötét idők”, manuális optimalizáció
- '70-es, '80-as évek
 - > Relációs adat és deklaratív SQL születése
 - > Az optimalizálás a rendszer feladata
 - > Join-ok sorrendjének optimalizációja stb.
 - Nagyságrendi különbségek!
 - > Heurisztikus optimalizálás
- '80-as, '90-es évek
 - > Költség alapú, teljesebb körű optimalizáció, push-downs
- '90-es évek
 - > Indexelhető, materializált nézetek, adattárházak

Optimalizálás alapelvek

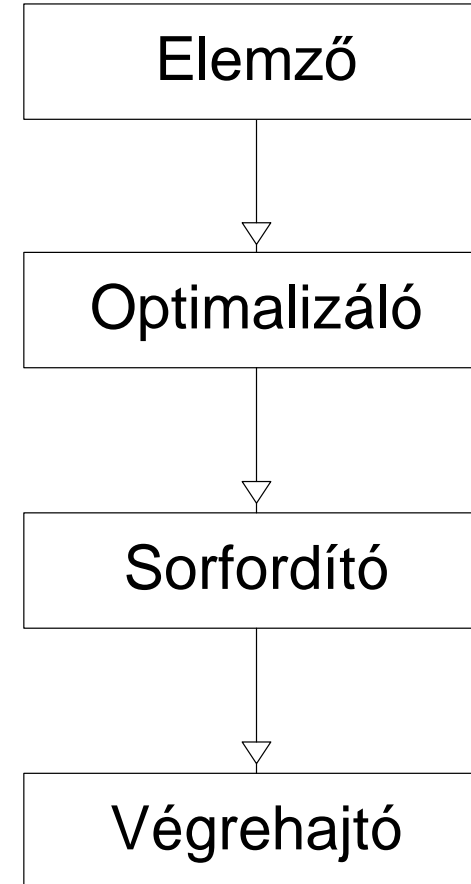
- Statisztikák alapján értékel
 - > Költség = Válaszidő (CPU + I/O idő)
- Triviális terv
 - > Egyszerűbb lekérdezéshez egyértelműen generálható
 - > Szabály alapú
- Ha nem készíthető triviális terv
 - > Összetett lekérdezések
 - > 3 fázisú optimalizáló

Háromfázisú optimalizáció

- Ha nincs triviális terv
- 0. Fázis
 - > Egyszerű átalakítások
 - > Preferált hash join
 - > Ha a költség $< X \rightarrow$ végrehajtás
- 1. Fázis
 - > Kibővített átalakítások
 - > Ha a költség $< Y \rightarrow$ végrehajtás
- 2. Fázis
 - > Párhuzamos végrehajtás vizsgálata

Lekérdezés feldolgozás menete

- Elemző
 - > Lekérdezés fordítása
 - > Logikai terv készítése
- Optimalizáló
 - > Fizikai terv elkészítése
 - > Táblák bejárása
 - > Táblák összekapcsolása
- Sorfordító
 - > Fizikai terv leképezése I/O műveletekre
- Végrehajtó
 - > Műveletek végrehajtása



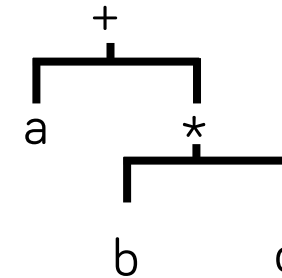
Microsoft SQL Server

Logikai végrehajtási terv

Logikai végrehajtási terv elemei

- Elemző fa
 - > Relációk (levél elemek)
 - > Műveletek (csomópontok)
 - > Adatok áramlása (lentről fölfelé)
- Relációs algebra műveletek
 - > Descartes-szorzat ($R \times S$)
 - > Projekció ($\pi_L(R)$)
 - > Szelekció/kiválasztás ($\sigma_F(R)$)
 - > Összekapcsolás ($R \bowtie S$)
 - > Ismétlődések szűrése ($\delta(R)$)
 - > Csoportosítás ($\gamma_L(R)$)
 - > Rendezés ($\tau_L(R)$)

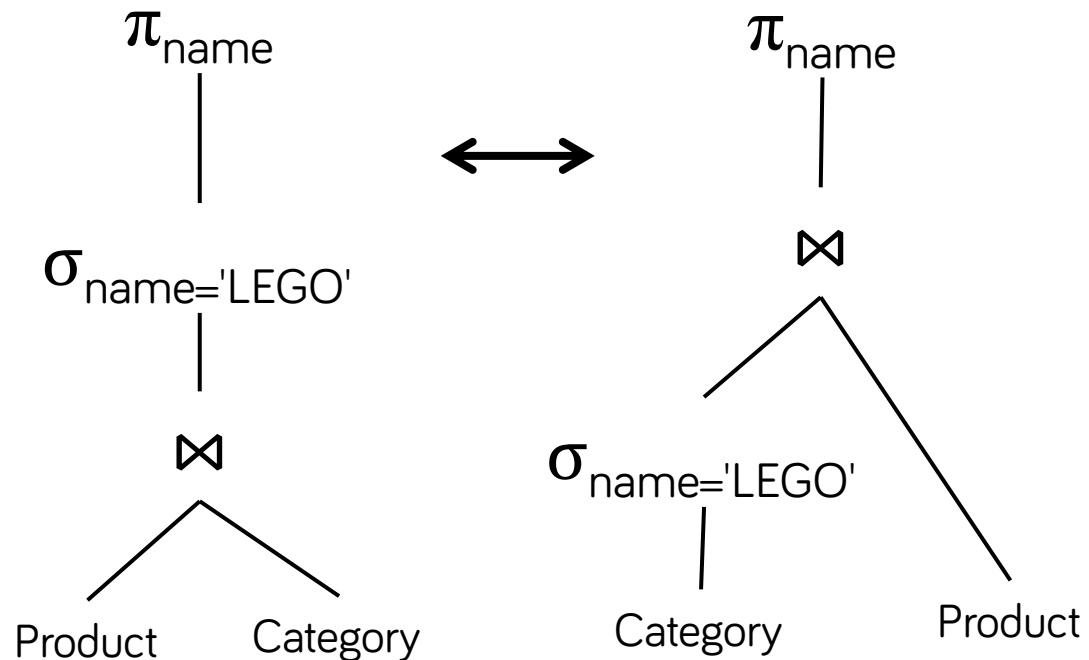
$$a + b * c$$



Egyszerű lekérdezés elemző fája

- Egyszerű lekérdezés, több, ekvivalens logikai terv

```
SELECT p.name  
from Product p  
join Category c on p.CategoryID = c.ID  
where c.Name = 'LEGO'
```



Elemzőfa átalakítása

- Ekvivalens átalakítások
 - > Ugyanazt az eredményhalmazt adják
- Legkisebb költségű logikai terv keresése
 - > Dinamikus programozás
 - > Heurisztika alkalmazása
- A végrehajtási terv keresési terének vágása
 - > Költség becslés alapján

Ekvivalens átalakítások 1

- Kiválasztás / szűrés / WHERE

- > Felcserélési szabály: $\sigma_{F_1}(\sigma_{F_2}(R)) = \sigma_{F_2}(\sigma_{F_1}(R))$

- > Szétvágási szabály:

- $\sigma_{F \text{ and } G}(R) = \sigma_F(\sigma_G(R))$

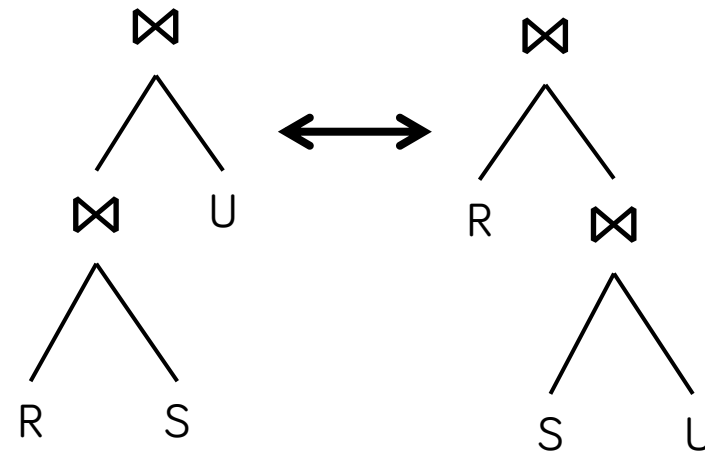
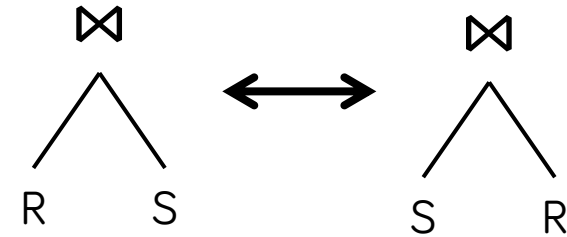
- $\sigma_{F \text{ or } G}(R) = \sigma_F(R) \text{ UNION } \sigma_G(R)$

- Összekapcsolás / join

- > $R \bowtie_F S = \sigma_F(R \times S)$

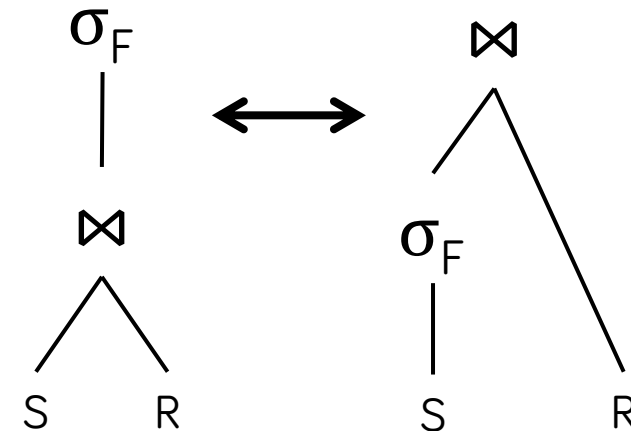
- > $R \bowtie S = S \bowtie R$

- > $(R \bowtie S) \bowtie U = R \bowtie (S \bowtie U)$



Ekvivalens átalakítások 2

- Összekapcsolás és kiválasztás
 - > $\sigma_F(R \bowtie S) = \sigma_F(R) \bowtie S$
 - ha R-ben szerepel minden F-ben vizsgált attribútum
 - > $\sigma_F(R \bowtie S) = R \bowtie \sigma_F(S)$
 - ha S-ben szerepelnek az F-ben vizsgált attribútumok
 - > $\sigma_F(R \bowtie S) = \sigma_F(R) \bowtie \sigma_F(S)$
 - ha R-ben és S-ben is szerepelnek F attribútumai
- Ismétlődések
 - > $\delta(\gamma_L(\mathbf{R})) = \gamma_L(\mathbf{R})$
 - > $\delta(\mathbf{R} \times \mathbf{S}) = \delta(\mathbf{R}) \times \delta(\mathbf{S})$
 - Join operátorokra is igaz



Optimalizálás

- Mi a legjobb sorrendje a $r_1 \bowtie r_2 \bowtie \dots \bowtie r_n$ kifejezésnek?
- Ha $n=7$ -> 665280 lehetőség
- Ha $n=10$ -> 176 milliárd lehetőség
 - > Dinamikus programozással 59000 alternatíva
- Rengeteg egyéb tényező
 - > Merge-join lassabb, de rendezett halmazt ad
- ...
- -> Túl nagy az optimalizálás költsége

Heurisztika: szabályok

- Szűrjünk minél előbb
 - > Kiválasztás (szűrés) műveletek lefelé mozgatása a fában
- Projektáljunk minél előbb
 - > Csökkenti az attribútumok számát
- A legerősebb szűréssel kezdjük
- A legszűkebb joinokkal kezdjük
 - > Az átmeneti halmaz kisebb
 - > Direkt szorzat csak akkor ha a lekérdezés erre utasít
 - `select * from Category, Product`

Fizikai terv

- Fizikai terv elemei
 - > Relációt beolvasó operátorok
 - Logikai terv levél eleminek beolvasása
 - > Relációs algebrai műveletet végrehajtó operátor
- Tervek készítése
 - > Szabály alapú
 - > Költségbecslés alapú
 - **Tábla elérési módok**
 - Join operátorok megvalósítási módjai
 - Join sorrend

Tábla-elérési módok

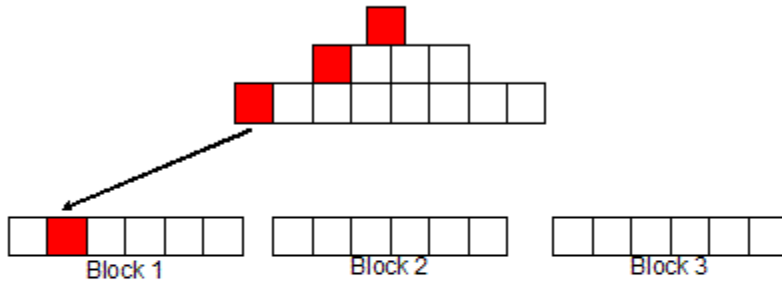
- Alapvetően két féle megközelítés
 1. **Table scan - Teljes átvizsgálás**
 - Ha nincs alkalmazható index
 - Ha minden rekordra szükség van
 - Kis táblák esetén
 - A táblára vonatkozó szűrési feltételt is kiértékeli
 2. **Index alapú átvizsgálás**
 - Szűrés esetén, ha a szűrési feltételre létezik index
 - Rendezés esetén, ha a rendezési feltételre létezik index

Indexelt táblaelérési módok

- **Clustered index scan**
 - > Nyalábolt adatolvasás
 - > Az adatblokkok index szerint vannak rendezve
 - > Clustered index primary key mentén létrejön
 - > Table scan helyett ezt preferálja
- **Nonclustered index scan**
 - > Hasonló, mint a clustered index scan
 - > Alapvetően az '=' operátor kiértékelésére
- **Clustered/Nonclustered index seek**
 - > Hasonló az index scan-hez
 - > B* fa leveleinek bejárása egy kezdőelemtől
 - A '>', 'between', '<' operátorok kiértékelése

MS SQL Server indexei

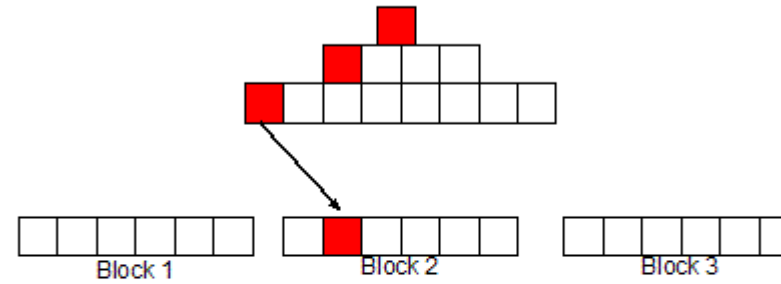
- Clustered / non-clustered



Select order_nbr, item_name from ordor natural join item;

Clustered table rows

Clustering_factor \approx blocks



Select order_nbr, item_name from ordor natural join item;

Un-Clustered table rows

Clustering_factor \approx num_rows

Forrás: http://www.dba-oracle.com/t_table_row_resequencing.htm

MS SQL Server indexei

- B* fa alapú indexek
 - > Egyszerű
 - > Összetett
 - Hierarchikus
 - > Clustered
 - Adatblokkok sorrendje index szerinti
 - Egy táblán egy lehet
 - Definiált elsődleges kulcs mentén automatikusan létrejön

Indexek sajátosságai

- Cover index (included column)
 - > B* fa levelének bővítése oszlopokkal
 - > Nem kell kiolvasni a tényleges rekordot
- Clustered és non clustered indexek együttes használata
 - > Nonclustered index levél eleme
 - Nem fizikai címet tartalmaz
 - Kulcs érték a clustered indexre
 - > Indirekció, dupla index olvasás
- Indexelt nézetek
 - > Nézet eredményének tárolása: Materialized view / materializált nézet

Fizikai terv

- Fizikai terv elemei
 - > Relációt beolvasó operátorok
 - Logikai terv levél eleminek beolvasása
 - > Relációs algebrai műveletet végrehajtó operátor
- Tervek készítése
 - > Szabály alapú
 - > Költségbecslés alapú
 - Tábla elérési módok
 - **Join operátorok megvalósítási módjai**
 - Join sorrend

Nested loop join

- Egymásba ágyazott kettős for ciklus $r \bowtie_{\theta} s$ számítására

```
for each tuple  $t_r$  in  $r$  do begin
  for each tuple  $t_s$  in  $s$  do begin
    test pair  $(t_r, t_s)$  to see if they satisfy the join condition  $\theta$ 
    if they do, add  $t_r \cdot t_s$  to the result.
  end
end
```

- **Tetszőleges méretű táblákra** működik
 - > Nagy méret esetén: a két tábla egy-egy blokkját tartja memóriában
- **Nagyon kis táblákra a leggyorsabb módszer**
- Lehetőséget ad csővezeték használatára
- I/O költség
 - > $O(\text{blokk_szám}_1 * \text{blokk_szám}_2)$

Hash join

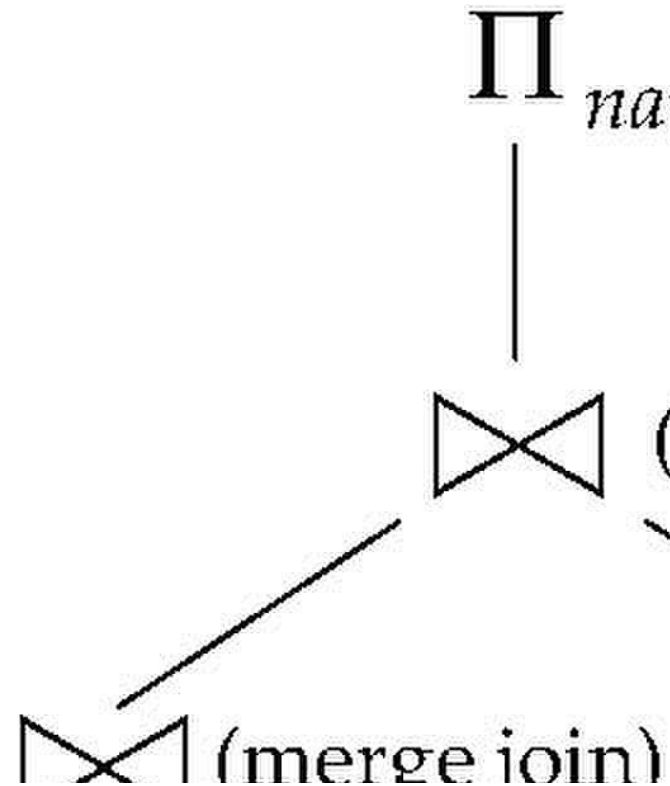
- Nem rendezett táblák, legalább az egyik kicsi
- Első menetben
 - > Kisebb reláció beolvasása
 - > Vödrös hash építése a memóriában / diszken
 - Kulcs a join operátorban szereplő oszlop
- Második menet
 - > A nagyobbik reláció beolvasása
 - > Kapcsolódó rekordok keresése a vödrös hashben
- I/O költség
 - > $O(\text{blokk_szám}_1 + \text{blokk_szám}_2)$

Sort Merge Join

- Ha már mind a két tábla rendezve van
 - > Ha nincs, a rendezés az összekapcsolási kulcs szerint plusz költség
 - > Lehet memóriában vagy diszken
- A két *rendezett listát* összefésüli
 - > Listák közös bejárása
- Azonos méretű relációk esetén
 - > Különösen, ha rendezett a két tábla
- I/O költség
 - > $O(\text{blokk_szám}_1 + \text{blokk_szám}_2)$
 - > A hash join gyorsabb, ha az egyik tábla jelentősen kisebb

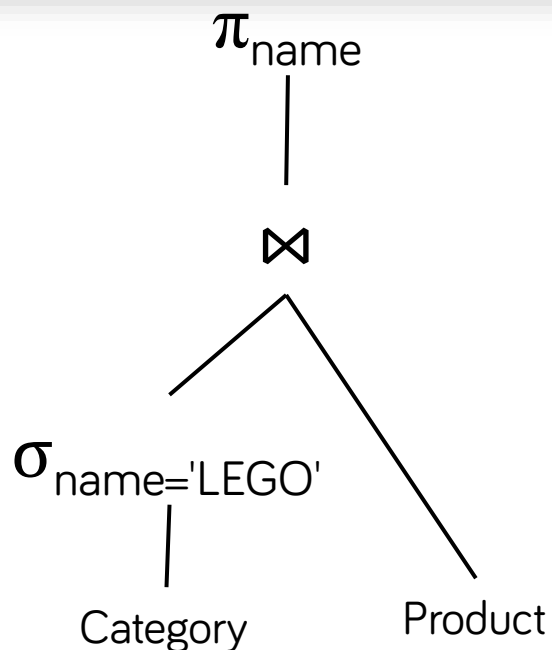
Fizikai végrehajtási terv

- Definiálja, hogy pontosan milyen műveletet hajt végre az egyes csomópontokban

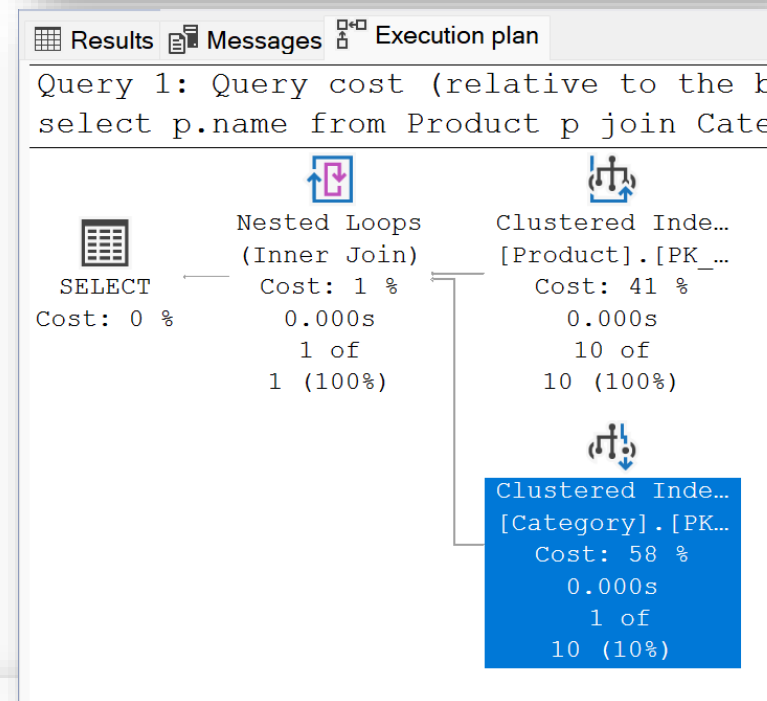
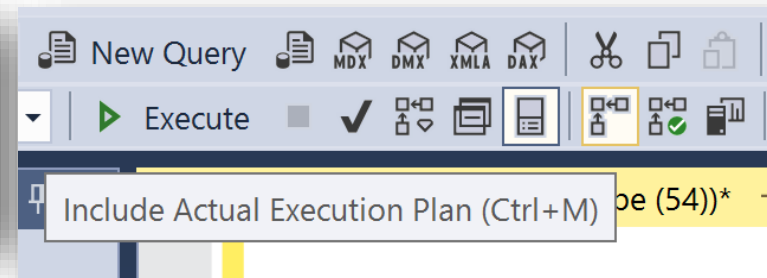


Végrehajtási terv megnézése

```
select p.name from Product p
join Category c on p.CategoryID = c.ID
where c.Name = 'LEGO'
```



Parallel	False
Physical Operation	Clustered Index Seek
Predicate	[M22XDS].[dbo].[Category].[Name] as [c].[Name]=N'LEGO'
Scan Direction	FORWARD



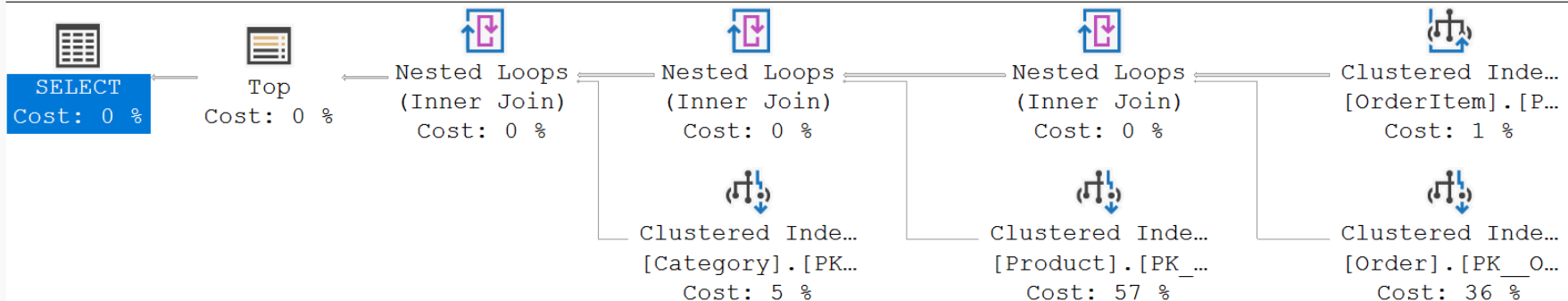
Query plan in SQL Server

- CTRL+L a lekérdezés futtatása után

```
select top 100 O.ID, p.name, c.name, oi.price
from
  OrderItem oi join [Order] o on oi.OrderID = o.ID
  join Product p on oi.ProductID = p.ID
  join Category c on p.CategoryID = c.ID
```

Query 1: Query cost (relative to the batch): 100%

```
select top 100 O.ID, p.name, c.name, oi.price from OrderItem oi join [Order] o on oi.Ord
```



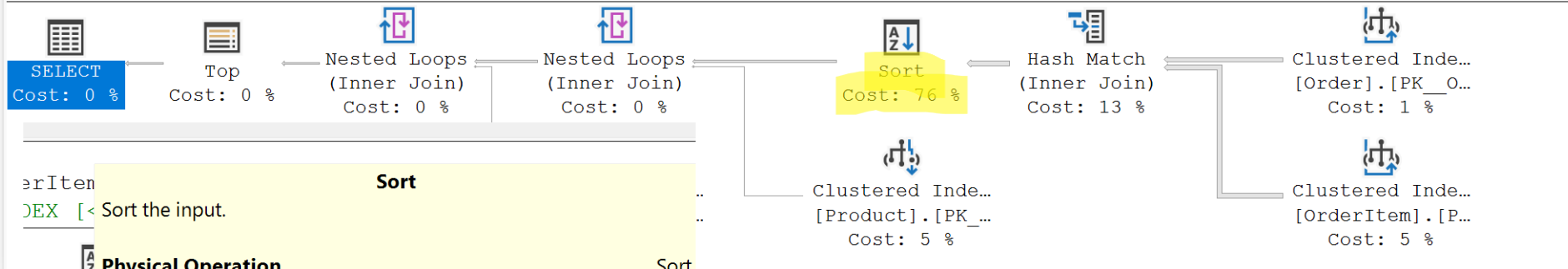
Index

```

select top 100 O.ID, p.name, c.name, oi.price
from
    OrderItem oi join [Order] o on oi.OrderID = o.ID
    join Product p on oi.ProductID = p.ID
    join Category c on p.CategoryID = c.ID
order by o.Deadline desc
    
```

Query 1: Query cost (relative to the batch): 100%

select top 100 O.ID, p.name, c.name, oi.price from OrderItem oi join [Order] o on oi.OrderID = o.ID joi...
 Missing Index (Impact 17.5168): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo]...

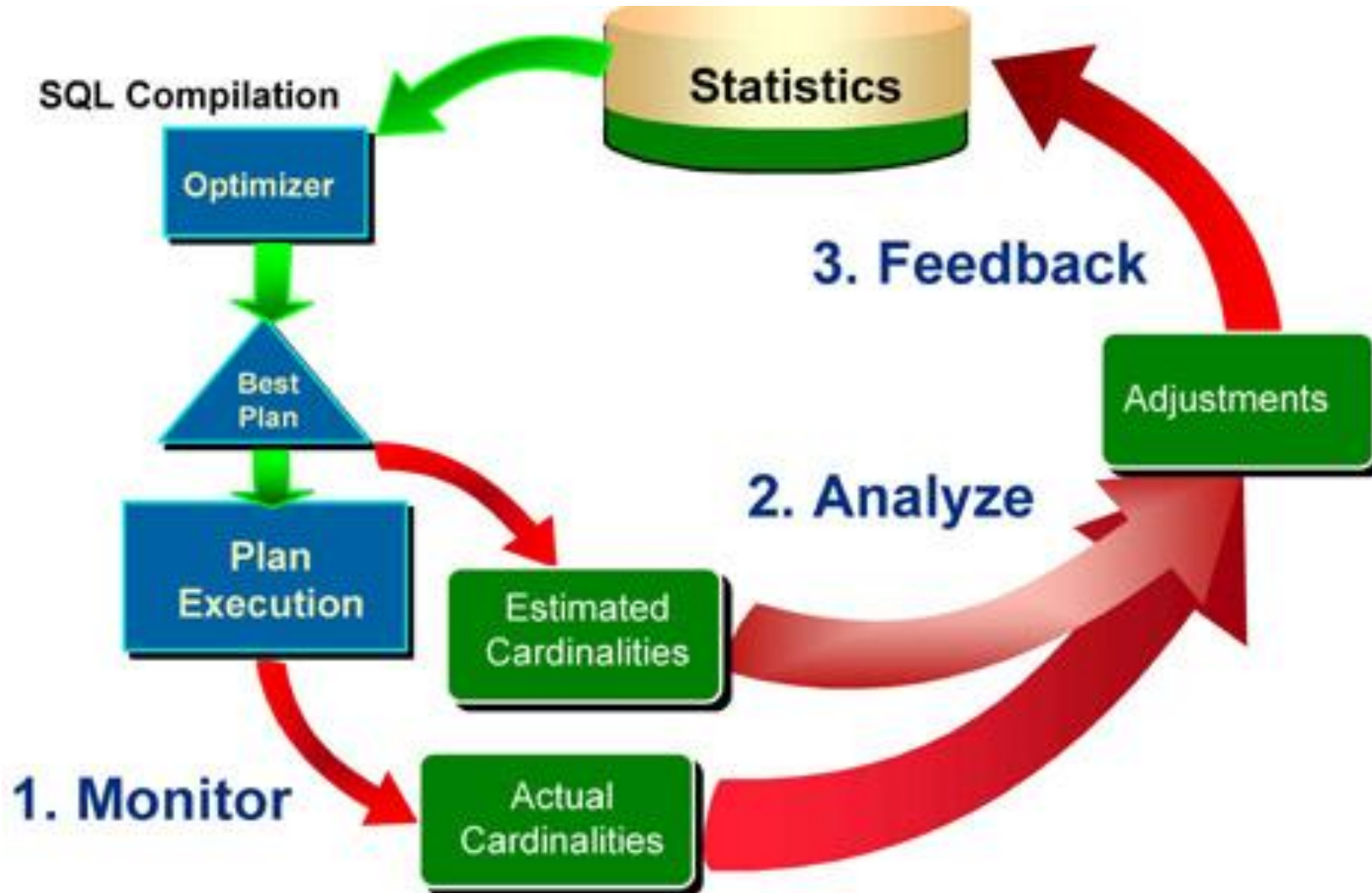


Sort	
Physical Operation	Sort
Logical Operation	Sort
Estimated Execution Mode	Row
Estimated I/O Cost	0,0112613
Estimated Operator Cost	3,40483 (76%)
Estimated CPU Cost	3,39357
Estimated Subtree Cost	4,24715
Estimated Number of Executions	1
Estimated Number of Rows for All Executions	100
Estimated Number of Rows Per Execution	100
Estimated Row Size	31 B
Node ID	4
Output List	
[M22XDS].[dbo].[OrderItem].Price; [M22XDS].[dbo].	
[OrderItem].ProductID; [M22XDS].[dbo].[Order].ID; [M22XDS].	
[dbo].[Order].Deadline	
Order By	
[M22XDS].[dbo].[Order].Deadline Descending	

Végrehajtási terv alternatívák

- Több terv alternatíva lehet, melyik optimális?
 - > Hatalmas különbségek: másodpercek vagy napok
 - > A költség azon múlik, hogy az egyes fázisoknál hány sor az eredmény
 - > Ezt statisztikák alapján becsli a rendszer
 - -> Önhangoló adatbázis, szükség esetén újratervez
- Plan cache
 - > végrehajtási terv cache
 - > Ha ugyan olyan struktúrájú lekérdezés jött -> ugyanaz a terv!
 - > Statisztikák nem változtak
 - Lehet, hogy kézzel kell frissíteni a cache-t!
 - A statisztika frissítést be kell állítani!

Önhangolás



Jó tanácsok – 1

- Statisztikák legyenek naprakészek
 - > Elavult statisztika → rossz végrehajtási terv
 - > (Automatikus, ha csak nem kapcsoljuk ki stb.)
- Lekérdezés struktúrája
 - > SQL deklaratív környezet
 - Gondolkodjunk procedurálisan is!
 - > Többféleképp is megfogalmazható ugyanaz
 - > Törekedjünk az egyszerűsége
 - > Kerüljük a select * -ot
 - > Jó struktúrával sokat lehet nyerni
 - Csak ezután kísérletezzünk a hintek használatával

Jó tanácsok – 2

- Inkább join, mint
 - > In / Not in
 - > Exists / Not exists
- Exists helyett inkább in
- Nézetek
 - > Ha lehet kerüljük
 - > Főleg ne kapcsoljuk egymáshoz
- Kerüljük a vagy feltételeket → Union all
- Union helyett Union all (ha lehet)
 - > Megtartja a duplikátumokat

Példa: nem bizonylatolt tételek

```
select *  
from Invoice i  
where not exists  
(  
    select 1  
    from Invoicetem ii  
    where i.Id=ii.InvoiceID  
)
```

```
select i.*  
from Invoice i  
where i.id not in  
(  
    select InvoiceID  
    from Invoicetem  
)
```

```
select i.*  
from Invoice i left outer join Invoicetem ii  
on i.Id=ii.InvoiceID  
where ii.id is null
```

Egyszerű esetekben ma már mindegy, de általában nem

Jó tanácsok – 4

- Indexek használata

- > Egy táblán egy lekérdezésben általában csak egyet tud használni → Join művelet el is használhatja
- > Összetett index
 - Hierarchia számít
- > Kulcs bármilyen kifejezésben szerepel akkor nem tudja használni az optimalizáló
 - Akár: kulcs+0 (← Ezt ma már észreveszik)

Jó tanácsok – 5

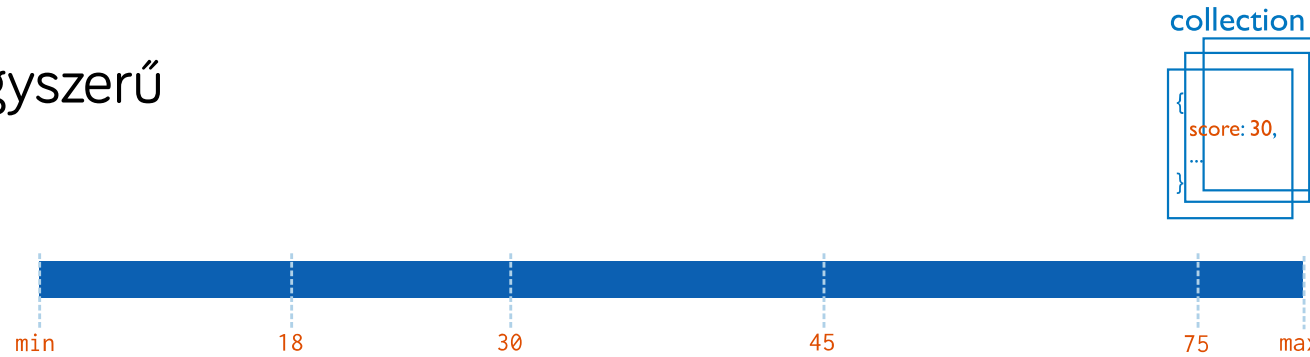
- Függvények használata
 - > Select listán nyugodtan
 - Nem befolyásolja a végrehajtási tervet
 - > Where feltételben lehetőleg ne használjuk
 - Minden rekordra le kell futtatni
 - Nehezen mozgatható a kifejezés fában
 - Kimenetére nem készül statisztika → nehéz optimalizálni

MongoDB indexek

- Index „csak” keresési célt szolgál
- Index típusok
 - > Egyszerű & összetett
 - > Unique index
 - primary key jellegű attribútumot így lehet garantálni
 - > Tömbök tartalmát is indexeli
 - > Beágyazott dokumentumokat is indexeli
 - > TTL, Geospatial, full text
- Indexet létre kell hozni
 - > Kivéve: _id-ra unique

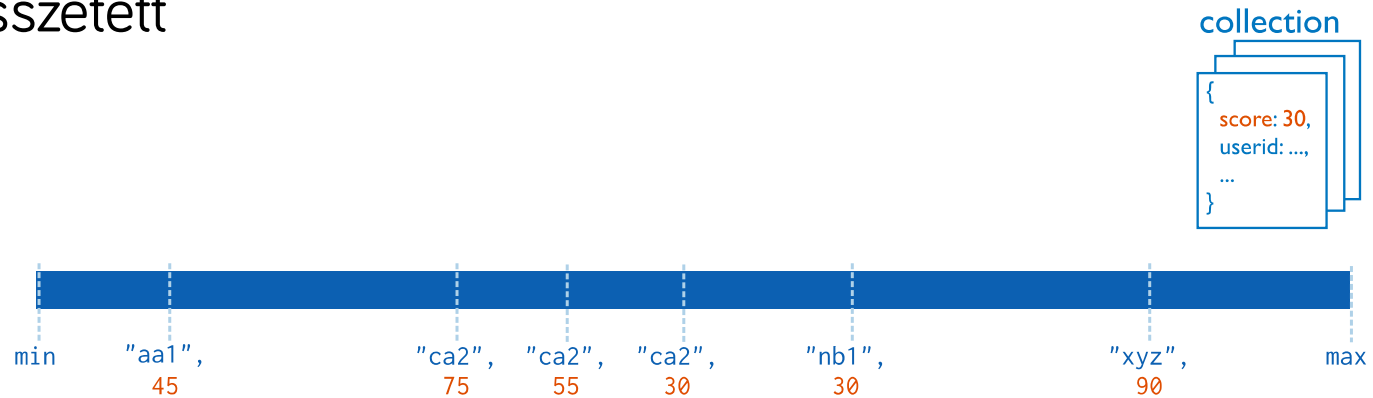
Index típusok

Egyszerű



{ score: 1 } Index

Összetett



{ userid: 1, score: -1 } Index

Képek forrása: <https://docs.mongodb.com/manual/indexes/>

Optimalizálási alapelvek

- **Nem** használ statisztikákat
- Több lehetséges terv közül választás
 - > Mindegyiket elkezdi végrehajtani, amelyik „legolcsóbban” adja vissza az első 101 db eredményt, az a legjobb
- Miért lehet több terv?
 - > Több index is lefedi a lekérdezést

Optimalizációs lépések

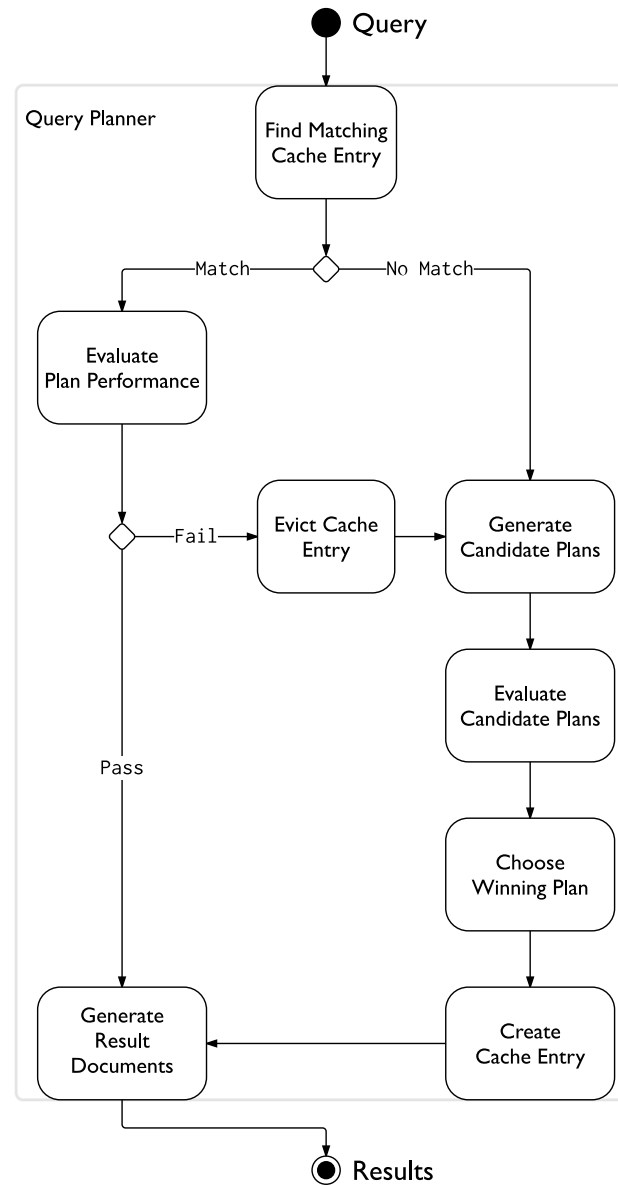
- Szűrések "előre" mozgatása
 - > Projekciók elé, ha kell, felbontva a szűrést több darabra
 - > Sorrendezés elé
- Skip és limit "előre" mozgatása
 - > Projekció elé
- Összevonások
 - > Limit + limit, skip + skip
- ...

Match mozgatása

```
{ $addField: {  
  maxTime: { $max: "$times" },  
  minTime: { $min: "$times" }  
} },  
{ $project: {  
  _id: 1, name: 1, times: 1, maxTime: 1, minTime: 1,  
  avgTime: { $avg: ["$maxTime", "$minTime"] }  
} },  
{ $match: {  
  name: "Joe Schmoe",  
  maxTime: { $lt: 20 },  
  minTime: { $gt: 5 },  
  avgTime: { $gt: 7 }  
} }
```

```
{ $match: { name: "Joe Schmoe" } },  
{ $addField: {  
  maxTime: { $max: "$times" },  
  minTime: { $min: "$times" }  
} },  
{ $match: { maxTime: { $lt: 20 }, minTime: { $gt: 5 } } },  
{ $project: {  
  _id: 1, name: 1, times: 1, maxTime: 1, minTime: 1,  
  avgTime: { $avg: ["$maxTime", "$minTime"] }  
} },  
{ $match: { avgTime: { $gt: 7 } } }
```

Terv cache



Kép forrása
<https://docs.mongodb.com/manual/core/query-plans/>

Terv cache

- Terv cache
 - > Strukturájában hasonló tervek
 - > Találat esetén pass/fail kiértékelés
- Terv „alakja” (*query shape*)
 - > Használt szűrések, rendezések és projekciók
 - > Értékek nincsenek benne
 - Pl. szűrésnél csak a mező(k) nevei szerepelnek, a szűrt érték nem

Explain

- query.explain()

```
"winningPlan" : {  
  "stage" : <STAGE1>,  
  ...  
  "inputStage" : {  
    "stage" : <STAGE2>,  
    ...  
    "inputStage" : {  
      "stage" : <STAGE3>,  
      ...  
    }  
  }  
},  
"rejectedPlans" : [  
  <candidate plan 1>,  
  ...  
]
```

Stage-ek

- *COLLSCAN*
- *IXSCAN*
- *FETCH*
- ...

Tervek vizuális nézete

sample_mflix.movies

23.5k DOCUMENTS 2 INDEXES

Documents Aggregations Schema **Explain Plan** Indexes Validation

Filter `{ "title": "Jurassic Park" }`

Project `{ field: 0 }`

Sort `{ field: -1 } or [['field', -1]]`

Collation `{ locale: 'simple' }`

Reset Explain `</>` Less Options

MaxTimeMS 60000

Limit 0

VIEW **VISUAL TREE** RAW JSON

Query Performance Summary

Documents Returned: 1

Index Keys Examined: 1

Documents Examined: 1

Actual Query Execution Time (ms): 3

Sorted in Memory: no

Query used the following index:

title

FETCH

nReturned: 1

Execution Time: 0 ms

DETAILS

IXSCAN

nReturned: 1

Execution Time: 2 ms

Index Name: title_1

Multi Key Index: no

DETAILS