OPTIMIZAREA INTEROGĂRILOR

Victor REABOI

Departamentul Ingineria Software și Automatică, grupa TI-201FR, Facultatea Calculatoare Informatică și Microelectronică, Universitatea Tehnică a Moldovei, Chișinău, Republica Moldova

Autorul corespondent: Reaboi Victor, victor.reaboi@isa.utm.md

Îndrumătorul/coordonatorul științific **Dorian SARANCIUC**, lector universitar

Abstract. *Bazele de date joacă un rol vital în gestionarea informațiilor în era digitală, iar optimizarea interogărilor reprezintă un aspect esențial pentru asigurarea recuperării și procesării eficiente a datelor. Acest articol explorează importanța și strategiile de optimizare a interogărilor în bazele de date, cu accent pe planurile de execuție, algoritmii și strategiile practice. Se discută despre normalizarea bazelor de date și impactul acesteia în optimizarea interogărilor. De asemenea, sunt prezentate tipurile de indexuri și rolul lor în accelerarea căutărilor. Articolul oferă, de asemenea, recomandări practice pentru a îmbunătăți eficiența interogărilor, inclusiv evitarea unor comenzi ineficiente și preferarea unor metode mai rapide. Se concluzionează cu sublinierea importanței optimizării continue în gestionarea bazelor de date în era digitală.*

Cuvinte cheie: *Optimizare interogări; Planuri de execuție; Algoritmi de optimizare; Normalizare baze de date; Indexuri în optimizare; Strategii de performanță; Structuri normalizate; Recomandări practică interogări.*

Planurile de Execuție a Interogărilor

Un aspect fundamental în optimizarea interogărilor este înțelegerea planurilor de execuție. Un plan de execuție este o reprezentare a modului în care sistemul de bază de date va executa o interogare. Acesta evidențiază pașii, operațiile și algoritmii pe care sistemul îi va utiliza pentru a accesa, uni, filtra, sorta, agregă și returna datele. Swapnil Parmar, Arhitect Enterprise la Kim Technologies, subliniază importanța acestor planuri în identificarea și modificarea indexilor pentru performanță optimă.

Algoritmi de Optimizare a Interogărilor

Algoritmii de optimizare a interogărilor sunt metodele pe care sistemul de bază de date le utilizează pentru a genera și a selecta cel mai bun plan de execuție pentru o anumită interogare. Kamlesh Ujgare, Oracle Apps DBA, evidențiază utilizarea predominantă a Optimizării Bazate pe Cost în bazele de date Oracle. Acest algoritm, dependent de statistici colectate, estimează costul și beneficiul fiecărui plan posibil de execuție. Algoritmi bazate pe reguli și heuristici oferă alternative, bazându-se pe reguli predefinite sau modele matematice pentru a simplifica și rescrie interogările [2].

Strategii de Optimizare a Interogărilor

Îmbunătățirea performanței interogărilor implică utilizarea unor strategii eficiente. Indexarea, partiționarea, statisticile și sugestiile sunt tehnici comune care afectează algoritmii de optimizare a interogărilor. Indexarea, așa cum menționează Ujgare, implică crearea și menținerea de indexi pentru a accelera accesul la date. Partiționarea împarte o tabelă mare în porțiuni mai mici, facilitând accesul la datele relevante și paralelizarea execuției interogărilor [4].

Monitorizarea Performanței Interogărilor

Monitorizarea performanței interogărilor reprezintă un proces esențial pentru măsurarea și analizarea în timp real a performanței interogărilor. Acest proces implică colectarea și analizarea unor metrici, cum ar fi timpul de execuție, consumul de resurse, evenimentele de

așteptare și erorile interogărilor. Utilizarea unor instrumente de monitorizare a performanței, precum SQL Server Profiler, Oracle Enterprise Manager sau MySQL Workbench, permite capturarea și afișarea datelor de performanță ale interogărilor. De asemenea, scripturile de monitorizare a performanței interogărilor, cum ar fi Dynamic Management Views în SQL Server, Automatic Workload Repository în Oracle sau Performance Schema în MySQL, oferă posibilitatea de a interoga și raporta datele de performanță ale interogărilor.

Normalizarea Bazelor de Date și Importanța sa în Optimizarea Interogărilor

Normalizarea reprezintă un proces esențial în proiectarea bazei de date, având ca obiectiv eliminarea redundanțelor și organizarea eficientă a datelor. Această practică este esențială pentru optimizarea interogărilor și asigurarea unei structuri coerente și flexibile a bazei de date [3].

Structura și Nivelele de Normalizare

Normalizarea implică împărțirea datelor în tabele pentru a evita redundanțele și a asigura dependența funcțională între coloane. Modelele normale, cum ar fi Forma Normală de Prima, a Doua și a Treia, precum și Forma Normală Boyce-Codd, oferă ghiduri pentru proiectarea eficientă a bazelor de date.

Eliminarea Redundanțelor

Prin eliminarea duplicării datelor, normalizarea contribuie la reducerea spațiului de stocare necesar și la menținerea coerenței datelor. De exemplu, într-un sistem normalizat, informațiile despre clienți sau produse pot fi stocate într-o singură tabelă, evitându-se replicarea necontrolată a datelor.

Optimizarea Interogărilor

Bazele de date normalizate facilitează optimizarea interogărilor prin structuri de date mai simple și relații bine definite între tabele. Această organizare eficientă permite accesul rapid la date și reduce complexitatea interogărilor, contribuind la o performanță crescută a acestora.

Flexibilitate în Gestionarea Datelor

Normalizarea conferă flexibilitate în gestionarea datelor și permite modificările structurale fără a afecta integritatea informațiilor existente. Într-o bază de date normalizată, adăugarea sau ștergerea de informații este mai ușoară și mai puțin predispusă la erori.

Importanța și Tipurile de Indexuri în Optimizarea Interogărilor

În lumea gestionării bazelor de date, optimizarea performanței interogărilor este esențială pentru asigurarea unui acces eficient la date. Unul dintre instrumentele cheie în acest proces este utilizarea indexurilor. Acest articol explorează importanța indexurilor, tipurile acestora și oferă exemple practice de optimizare a interogărilor în funcție de situații specifice [5].

Indexurile reprezintă structuri de date care permit o căutare rapidă și eficientă într-o bază de date. Ele funcționează asemenea unor cărți de referință care indică locația exactă a informațiilor dorite. Importanța indexurilor poate fi înțeleasă prin următoarele aspecte:

1. Accelerarea Căutărilor. Indexarea permite bazelor de date să localizeze rapid informațiile dorite, reducând timpul de căutare. În loc să parcurgă întreaga tabelă, sistemul poate utiliza indexul pentru a ajunge direct la locul unde se află datele căutate.
2. Optimizarea Unirilor. Atunci când se realizează unirea între mai multe tabele, indexurile pot îmbunătăți semnificativ performanța. Ele permit sistemului să găsească corespondențele între cheile din diferite tabele mai eficient.
3. Eficiența Sortării și a Grupării. Indexurile pot accelera operațiile de sortare și grupare, deoarece datele sunt deja organizate într-un anumit mod. Acest aspect este esențial în interogările care necesită prelucrarea și prezentarea datelor într-un anumit mod.

Există diverse tipuri de indexuri, fiecare adaptat pentru anumite scenarii și tipuri de interogări. Iată câteva dintre cele mai comune tipuri:

1. Index Simplu (Single-Column Index). Este cel mai comun tip de index și se aplică unei singure coloane. Util pentru interogări care se bazează pe acea coloană.
2. Index Compus (Composite Index). Implică două sau mai multe coloane. Util pentru optimizarea interogărilor care implică condiții pe mai multe coloane.
3. Index Unic (Unique Index). Asigură că valorile din coloana indexată sunt unice. Util pentru chei primare sau restricții de unicitate.
4. Index Descendent (Descending Index). Permite sortarea descrescătoare a valorilor în index. Util pentru interogări care necesită sortare descrescătoare.
5. Index Spatial (Spatial Index). Utilizat în baze de date spațiale pentru optimizarea interogărilor spațiale.

Exemple și recomandări

Selecția

Selectarea rândurilor necesare în loc să fie selectate toate rândurile ar trebui să fie urmată. SELECT * este foarte ineficient deoarece scanează întreaga bază de date [1].

```
SET STATISTICS TIME ON
SELECT * FROM SalesLT.Product
```

```
(295 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 125 ms.

Completion time: 2021-10-03T17:45:01.8602635+05:30
```

```
SET STATISTICS TIME ON
SELECT ProductNumber, Name, Color,Weight FROM SalesLT.Product
```

```
 (295 rows affected)

  SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 41 ms.

 Completion time: 2021-10-03T17:45:58.3009888+05:30
```

Evitați folosirea comenzii SELECT DISTINCT

Comanda SELECT DISTINCT în SQL este folosită pentru a obține rezultate unice și pentru a elimina rândurile duplicate în relație. Pentru a realiza această sarcină, ea grupează în mod fundamental rândurile relevante și apoi le elimină. Operația GROUP BY este o operație costisitoare. Așadar, pentru a obține rânduri distincte și pentru a elimina rândurile duplicate, este posibil să utilizați mai multe atribute în operația SELECT [1].

```
SET STATISTICS TIME ON
SELECT DISTINCT Name, Color, StandardCost, Weight FROM SalesLT.Product
```

```
SQL Server Execution Times:
  CPU time = 0 ms,  elapsed time = 111 ms.

Completion time: 2021-10-03T23:13:53.9723237+05:30
```

```
SET STATISTICS TIME ON
SELECT  Name, Color, StandardCost, Weight, SellEndDate, SellEndDate FROM SalesLT.Product
```

```
SQL Server Execution Times:
  CPU time = 0 ms,  elapsed time = 69 ms.

Completion time: 2021-10-03T23:13:19.7751003+05:30
```

Inner joins vs WHERE clause

Ar trebui să folosim INNER JOIN pentru a uni două sau mai multe tabele în loc să folosim clauza WHERE. Clauza WHERE creează o îmbinare CROSS / produs CARTEZIAN al tabelelor. Produsul CARTEZIAN al două tabele necesită mult timp [1].

```
SET STATISTICS IO ON
SELECT p.Name, Color, ListPrice
FROM SalesLT.Product p, SalesLT.ProductCategory pc
WHERE P.ProductCategoryID = pc.ProductCategoryID
```

```
(295 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0,
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0
Table 'Product'. Scan count 1, logical reads 103, physical reads 0, page server reads 0, read-ahead reads 0
Table 'ProductCategory'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead re

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 90 ms.

Completion time: 2021-10-03T23:33:29.2651153+05:30
```

```
SET STATISTICS TIME ON
SELECT p.Name, Color, ListPrice FROM SalesLT.Product p
INNER JOIN  SalesLT.ProductCategory pc
ON P.ProductCategoryID = pc.ProductCategoryID
```

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 21 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(295 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0,
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0,
Table 'Product'. Scan count 1, logical reads 103, physical reads 0, page server reads 0, read-ahead reads 0,
Table 'ProductCategory'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead r
```

Comanda limit

Comanda LIMIT este folosită pentru a controla numărul de rânduri care trebuie afișate din setul de rezultate. Setul de rezultate ar trebui să afișeze doar acele rânduri care sunt necesare. Prin urmare, trebuie să se utilizeze LIMIT cu setul de date de producție și să se ofere o calculare la cerere a rândurilor în scopuri de producție [1].

```
SET STATISTICS IO ON
SELECT Name, Color, ListPrice
FROM SalesLT.Product
LIMIT 10
```

	Name	Color	ListPrice
1	HL Road Frame - Black, 58	Black	1431.50
2	HL Road Frame - Red, 58	Red	1431.50
3	Sport-100 Helmet, Red	Red	34.99
4	Sport-100 Helmet, Black	Black	34.99
5	Mountain Bike Socks, M	White	9.50
6	Mountain Bike Socks, L	White	9.50
7	Sport-100 Helmet, Blue	Blue	34.99
8	AWC Logo Cap	Multi	8.99
9	Long-Sleeve Logo Jersey, S	Multi	49.99
10	Long-Sleeve Logo Jersey, M	Multi	49.99

IN versus EXISTS

Operatorul IN este mai costisitor decât EXISTS în ceea ce privește scanările, în special atunci când rezultatul subinterogării este un set mare de date. Prin urmare, ar trebui să încercăm să folosim EXISTS în loc să folosim IN pentru a obține rezultate cu o subinterogare [1].

```
SET STATISTICS TIME ON
SELECT ProductNumber,Name,Color FROM SalesLT.Product
WHERE ProductID IN
(SELECT ProductID FROM SalesLT.ProductDescription)
```

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 2 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(295 rows affected)
Table 'ProductDescription'. Scan count 1, logical reads 590
Table 'Product'. Scan count 1, logical reads 103, physical

 SQL Server Execution Times:
   CPU time = 16 ms,  elapsed time = 111 ms.
```

```
SET STATISTICS TIME ON
SELECT ProductNumber,Name,Color FROM SalesLT.Product
WHERE  EXISTS
(SELECT ProductID FROM SalesLT.ProductDescription)
```

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 4 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(295 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 53 ms.

Completion time: 2021-10-04T17:41:52.7784161+05:30
```

Concluzie

În concluzie, optimizarea interogărilor în bazele de date reprezintă un aspect crucial pentru eficiența și performanța sistemelor informatice în era digitală. Articolul a acoperit o gamă variată de aspecte, de la înțelegerea planurilor de execuție a interogărilor și a algoritmilor de optimizare, până la strategii practice precum indexarea, partiționarea și monitorizarea performanței.

Importanța normalizării în proiectarea bazelor de date a fost evidențiată ca fiind esențială pentru eliminarea redundanțelor, optimizarea interogărilor și asigurarea unei structuri flexibile și coerente. Structurile normalizate au fost prezentate ca având beneficii semnificative în accelerarea accesului la date, reducerea complexității interogărilor și oferirea flexibilității în gestionarea informațiilor.

De asemenea, s-a subliniat importanța indexurilor în optimizarea performanței interogărilor, evidențiind diversele tipuri de indexuri și rolul lor în accelerarea căutărilor, optimizarea unirilor și eficiența sortării și grupării.

Recomandările de optimizare a interogărilor menționate anterior sunt esențiale pentru îmbunătățirea eficienței și performanței în gestionarea bazelor de date. Selecția adecvată a

datelor, cum ar fi evitarea folosirii comenzi SELECT * și limitarea rezultatelor la doar rândurile necesare, contribuie la reducerea timpului de execuție și a resurselor necesare, optimizând astfel procesul de interogare.

De asemenea, evitarea comenzii SELECT DISTINCT în favoarea unei abordări mai eficiente, care implică utilizarea mai multor atribute în operația SELECT, ajută la prevenirea costurilor ridicate asociate operației GROUP BY. Aceasta este o practică esențială pentru a asigura că rezultatele distincte sunt obținute într-un mod eficient și economic.

În ceea ce privește unirea tabelelor, preferarea clauzei INNER JOIN în locul clauzei WHERE pentru a evita îmbinarea CROSS / produsul CARTEZIAN al tabelelor duce la îmbunătățirea semnificativă a performanței interogărilor.

Utilizarea comenzii LIMIT pentru controlul numărului de rânduri afișate în setul de rezultate este o practică utilă pentru a asigura că doar datele necesare sunt recuperate și prezentate utilizatorilor. Aceasta contribuie la o gestionare eficientă a volumelor mari de date și la reducerea timpului de răspuns al interogărilor.

În plus, preferarea operatorului EXISTS în locul operatorului IN în cazul subinterogărilor cu seturi mari de date ajută la minimizarea costurilor de scanare și îmbunătățirea performanței interogărilor.

Prin aplicarea acestor recomandări în practică, se poate realiza o optimizare semnificativă a interogărilor, asigurând o experiență eficientă și rapidă în gestionarea bazelor de date în mediul digital.

Articolul a evidențiat că optimizarea interogărilor nu este doar un proces teoretic, ci necesită o abordare practică și continuă, utilizând instrumente specializate pentru monitorizare și tunare a performanței. În contextul evoluției tehnologice, preocuparea constantă pentru eficiență rămâne un pilon esențial în gestionarea eficientă a datelor în era digitală, oferind cititorilor cunoștințe și instrumente esențiale pentru abordarea complexității optimizării interogărilor în mediul digital.

Bibliografie

[1] IACOB, N. (fără an). *OPTIMIZAREA INTEROGĂRILOR*. Preluat de pe https://www.utgjiu.ro/revista/ec/pdf/2010-04.I/17_NICOLETA_IACOB.pdf

[2] *Query optimization*. (fără an). Preluat de pe Wikipedia: https://en.wikipedia.org/wiki/Query_optimization

[3] *Normalizarea bazei de date*. (fără an). Preluat de pe :: Departamentul de Electrotehnica :: Facultatea de Inginerie Electrica :: Universitatea Politehnica din Bucuresti ::: http://www.elth.pub.ro/~preda/teaching/BDE/BDE_5.pdf

[4] *The Different Types of Indexes in Databases: A Comprehensive Overview*. (fără an). Preluat de pe Medium – Where good ideas find you.: https://londondataconsulting.medium.com/the-different-types-of-indexes-in-databases-a-comprehensive-overview-559a0c4f5fb5

[5] *A Detailed Guide on SQL Query Optimization*. (fără an). Preluat de pe Analytics Community | Analytics Discussions | Big Data Discussion: https://www.analyticsvidhya.com/blog/2021/10/a-detailed-guide-on-sql-queryoptimization/#:~:text=The%20query%20optimization%20process%20involves,plan%20based%20on%20cost%20estimations