Calea critică

Cum găsesc limita înainte
ca utilizatorul să ajungă la ea

Ioan Marian

MINI-CARTE · 30 DE PAGINI

DESCHIDERE

Am vrut să văd până unde merge

Un experiment făcut acasă, înainte de primul client.

În 1998 lucram acasă la o aplicație de contabilitate în Access. Înainte să o dau primului client, nu am verificat numai dacă se deschide, calculează și salvează. Am vrut să aflu ce se întâmplă atunci când datele cresc.

Am mărit treptat volumul. Cu câteva mii de înregistrări mergea bine. Cu câteva sute de mii își pierdea ritmul. În jurul unui milion nu mai răspundea. Am întâlnit limita eu, în camera mea, nu clientul după livrare.

Începusem să măsor încă din 1994, pe calculatoarele 286 și 386 ale facultății. Din experimentul din 1998 mi-a rămas o întrebare: până unde merge și ce se întâmplă atunci când volumul crește? Mini-cartea poate fi citită singură. Fiecare lecție explică problema, proba și rezultatul.

DE CE

Nu optimizez pentru o cifră frumoasă. Reduc munca inutilă pentru ca aplicația să rămână rapidă, ușor de înțeles și ușor de schimbat atunci când datele și cerințele cresc.

Un test îmi spune că aplicația merge. Un test de volum îmi spune până unde.

HARTA MINI-CĂRȚII

De la efect la cauză, apoi de la soluție la continuitate

Cele 26 de lecții urmează ordinea în care lucrez: întâi fac problema vizibilă, apoi elimin munca inutilă și, la final, las o metodă pe care o poate continua altcineva.

01 Înțeleg ce se întâmplă

Lecțiile 01-09
Incident, log, reper, volum și punct de rupere.

02 Reduc munca inutilă

Lecțiile 10-21
Calea critică, ID-first, paralelism, căutare și modele.

03 Construiesc continuitate

Lecțiile 22-26
Pregătire, progres, evaluare, protecție și transfer.

CUM O CITEȘTI

Poți parcurge toate cele 30 de pagini sau poți începe cu problema apropiată de sistemul tău. Fiecare lecție conține suficient context pentru a fi înțeleasă separat.

LECȚIA 01 | ÎNȚELEG CE SE ÎNTÂMPLĂ

O linie de cod a blocat o platformă întreagă

Era ora prânzului. Mă pregăteam să plec în pauză când managerul a venit la mine îngrijorat: «Aplicația intranet nu mai merge.»

Am întrebat liniștit: «Care dintre ele?» Mă gândeam că o pagină izolată se blocase temporar. Nu puteam concepe răspunsul primit: «Nicio pagină nu mai merge.»

Platforma era folosită zilnic de foarte mulți oameni și concentra documente și mai multe aplicații interne. Funcționase de mai mulți ani fără un incident asemănător.

A început o cursă contra cronometru. În aproximativ cincisprezece minute a fost izolată componenta care producea blocajul. Fusese dezvoltată cu mai multe luni înainte. Componenta a fost dezactivată imediat, iar utilizatorii și-au putut relua activitatea.

Abia după aceea am preluat codul sursă. Cunoșteam modelul după care trebuia construită componenta. De aceea, după ce am deschis codul, linia care nu respecta modelul mi-a atras atenția în aproximativ douăzeci de secunde.

Citirea datelor nu fusese păstrată ca etapă separată. Se citea o listă, iar în interiorul buclei fiecare element declanșa alte citiri.

La început, volumul fusese mic. Pagina funcționase normal și nimeni nu observase problema. Apoi volumul crescuse în tăcere. Fiecare citire părea scurtă, dar repetarea ei pentru fiecare element a schimbat complet rezultatul.

Am mutat citirea înaintea buclei și am refolosit rezultatul. După testarea pe toate mediile, componenta corectată a revenit în producție în aceeași zi.

ÎNCEARCĂ Alege un incident deja vizibil. Mai întâi restabilește serviciul; abia apoi reconstruiește cauza pas cu pas.

IDEEA **Platforma era deja blocată. Nu mai trebuia să măsor cât de lentă era. Trebuia să găsesc ce o oprise.**

LECȚIA 02 | ÎNȚELEG CE SE ÎNTÂMPLĂ

Până mâine să meargă de patru ori mai repede

Eram în mijlocul unei discuții despre arhitectură când a intrat managerul meu direct. Cerința privea o pagină de care nu mă ocupasem niciodată. Fusese dezvoltată de altă echipă și o moștenisem fără context și fără documentație.

La prânz venise mesajul: «Până mâine să meargă prima pagină de patru ori mai repede. Și până diseară să am pașii de implementare.»

Nu era o eroare. Pagina funcționa, dar era pagina principală a unei platforme interne SharePoint, folosită foarte des, și avea nevoie de câteva secunde pentru a se încărca.

Un singur fișier XSL — un șablon care transforma datele în pagina văzută de utilizator — grupa toate cazurile și era apelat de mai multe ori în timpul construirii aceleiași pagini.

Aveam mai multe posibilități. Puteam să rescriu complet afișarea. Puteam să păstrez temporar rezultatul într-un cache, caz în care pagina ar fi părut mai rapidă, dar cauza ar fi rămas. Sau puteam să împart fișierul mare și să execut numai partea necesară.

Nu am ales după preferință. În aproximativ zece minute am pus timpi în log și am măsurat separat componentele. În alte aproximativ zece minute am mutat alegerea cazului în C# și am împărțit fișierul inițial în mai multe fișiere XSL mai mici. Fiecare era apelat o singură dată, numai atunci când era necesar.

Nu am schimbat regulile și nici informațiile afișate. Am schimbat numai felul în care era împărțită munca. După testarea pe toate mediile, soluția a fost publicată în aceeași zi.

Ținta a fost depășită: pagina a ajuns să răspundă de aproape șase ori mai repede.

Presiunea a dispărut, dar lecția a rămas: uneori, viteza vine doar din felul în care împarți lucrurile.

ÎNCEARCĂ	Când timpul este scurt, măsoară componentele și caută schimbarea mică ce elimină cea mai multă muncă repetată.
IDEEA	**Presiunea nu mă obligă să schimb mult. Mă obligă să aleg precis.**

LECȚIA 03 | ÎNȚELEG CE SE ÎNTÂMPLĂ

Nu mai avem documentele

Mi s-a cerut să transfer o arhivă mare de documente dintr-o platformă internă către un alt spațiu de stocare. Erau proceduri, documente interne și arhive care adunaseră ani de muncă.

Am descărcat documentele, le-am organizat în foldere și le-am lăsat la destinația indicată. Totul părea în regulă.

Ceva timp mai târziu am aflat că documentele nu mai erau disponibile nici la sursă, nici la destinație. Întrebarea era simplă și grea: mai exista undeva o copie?

Din întâmplare, nu ștersesem încă folderele temporare de pe vechiul server. Acea copie, păstrată fără intenție, a făcut posibilă recuperarea întregii arhive.

Nu prezint întâmplarea ca pe o procedură bună. Am avut noroc. Dacă aș fi șters folderele temporare imediat după transfer, documentele nu ar mai fi putut fi recuperate.

Unele documente se pot reface în câteva ore. Altele nu se mai refac niciodată.

Backup faci mereu.

ÎNCEARCĂ Pentru un transfer important, verifică separat destinația, backupul și restaurarea înainte să închizi sursa.

IDEEA **Backupul care nu a fost restaurat niciodată este încă o promisiune.**

LECȚIA 04 | ÎNȚELEG CE SE ÎNTÂMPLĂ

Caută o limită pe care o poți reproduce

Nu începe cu întregul sistem. Alege o singură operație și fă punctul de rupere vizibil.

Alege o citire pe care o poți repeta. Păstrează aceleași coloane și mărește numărul rândurilor de zece ori la fiecare pas, până când timpul depășește pragul acceptat sau începe să crească mai repede decât volumul. Revino apoi la același număr de rânduri și schimbă numai câte coloane citești. Nu modifica ambele axe în aceeași probă.

Dacă înveți acum	Dacă coordonezi	Dacă proiectezi
O singură schimbare	Nu cere numai o cifră	Două axe

ÎNCEARCĂ Alege o citire și mărește numai numărul de rânduri până când curba își schimbă forma.

LECȚIA 05 | ÎNȚELEG CE SE ÎNTÂMPLĂ

Numele fișierului spune ce a pornit și când

Folosesc modelul ClientList_AAAALLZZ_OOMMSS_FFF.txt. De exemplu, ClientList_20260806_115522_012.txt arată că operația ClientList a pornit la 6 august 2026, ora 11:55:22,012.

Fiecare rulare primește propriul fișier. Astfel nu amestec două cereri care au avut volume, timpi sau rezultate diferite. Din nume găsesc repede operația și momentul. Din conținut refac pașii în ordinea în care au avut loc.

Înregistrez începutul fiecărei etape, durata ei, timpul adunat de la pornire, numărul de rânduri, fișiere sau apeluri și valorile de business necesare pentru a verifica un calcul. La final păstrez rezultatul și eroarea, dacă a existat. Nu scriu date personale sau conținut sensibil care nu ajută analiza.

01	**ClientList**	Operația de business care a pornit.
02	**AAAALLZZ**	Anul, luna și ziua pornirii.
03	**OOMMSS**	Ora, minutul și secunda pornirii.
04	**FFF**	Fracțiunea de secundă care deosebește două porniri apropiate.

ÎNCEARCĂ Creează un fișier nou pentru fiecare execuție importantă și fă numele suficient de clar pentru a-l găsi fără cod.

IDEEA

Un log bun nu spune doar că operația s-a terminat. Spune povestea completă a execuției.

LECȚIA 06 | ÎNȚELEG CE SE ÎNTÂMPLĂ

Observ câteva zile sau citesc istoricul care există deja

O valoare izolată devine mai întâi o întrebare, nu o optimizare urgentă.

Dacă riscul permite, observ câteva zile și verific dacă abaterea se repetă. Dacă am deja două săptămâni de istoric, pot vedea direct dacă tendința era deja prezentă.

Pentru fiecare pagină păstrez separat istoricul execuțiilor și calculez p50, p95 și p99. Calculez aceleași valori și pentru ansamblul aplicației, dar valoarea generală poate ascunde degradarea unei singure pagini. p95 este timpul sub care s-au încheiat 95% dintre execuții; restul de 5% au durat mai mult.

Dacă am 1.000 de execuții comparabile și p95 este 0,3 secunde, aproximativ 950 s-au terminat în cel mult 0,3 secunde, iar aproximativ 50 au fost mai lente. Analiza este manuală și automată: testele zilnice și rapoartele standardizate îmi arată fotografia fiecărei pagini și tendința ei în timp.

Astăzi O observație	**Câteva zile** Repetare	**Două săptămâni** Istoric
Periodic Intervenție		

ÎNCEARCĂ Calculează p95 pentru toate execuțiile și separat pentru fiecare pagină; apoi verifică dacă abaterea se repetă.

IDEEA **Privesc cifra în context, confirm tendința și intervin înainte să devină incident.**

LECȚIA 07 | ÎNȚELEG CE SE ÎNTÂMPLĂ

Algoritmul, datele de intrare, sortarea și rezultatul

Nu măsuram încă SQL Server sau milisecunde; învățam să fac problema verificabilă.

În COBOL porneam de la date de intrare pe care le cunoșteam, executam calculul și sortarea, apoi verificam raportul rezultat. În dBase IV și FoxPro păstram aceeași ordine: intrare cunoscută, pași explicabili și rezultat verificat.

Primul reper nu a fost un timp foarte mic. A fost un caz pe care îl puteam repeta și al cărui rezultat îl puteam verifica după fiecare schimbare.

Intrare	Pași	Ieșire
Date controlate	Calcul și sortare	Raport verificat

ÎNCEARCĂ Construiește un tabel simplu cu operația, volumul, timpul și rezultatul verificat.

IDEEA **Primul reper nu a fost o milisecundă. A fost un rezultat pe care îl puteam verifica.**

LECȚIA 08 | ÎNȚELEG CE SE ÎNTÂMPLĂ

Un milion de rânduri nu are un singur cost

Acesta este un laborator SQL separat; nu face parte din seria automată C# din repository. Vreau să văd separat efectul numărului de rânduri și al numărului de coloane.

1. Creez un model sintetic cu un milion de rânduri și cincizeci de coloane: ID, nouăsprezece numere, zece valori de tip dată, zece titluri și zece descrieri.

2. Selectez mai întâi numai ID-ul. Reiau aceeași probă cu 1, 10, 100, 1.000, 10.000, 100.000 și 1.000.000 de rânduri.

3. Reîncep de la un rând pentru fiecare grup de 5, 10, 20 și 50 de coloane. Nu aleg primele coloane fizice. Fiecare grup amestecă numere, date, titluri și descrieri.

4. Grupul de 5 conține ID, două numere, o dată și un titlu. Grupul de 10 adaugă alte numere și date, încă un titlu și două descrieri. Grupul de 20 are patru descrieri. Grupul de 50 le include pe toate zece.

5. Descrierile au lungimi diferite. Din acest motiv, zece coloane pot muta mult mai multe date decât cinci, iar creșterea nu urmează numai numărul coloanelor.

6. Tabelul arată rezultatele pentru volumul de un milion de rânduri. Volumele sunt estimări reproductibile ale încărcăturii sintetice. Timpii SQL depind de server, indexuri, cache și rețea și trebuie măsurați în mediul în care rulează baza de date.

SELECȚIE	VALORI ÎNTOARSE	VOLUM ESTIMAT (MB)
Numai ID	1.000.000	8,00
5 coloane	5.000.000	77,00
10 coloane	10.000.000	478,00
20 de coloane	20.000.000	2.820,00
50 de coloane	50.000.000	7.170,00

ÎNCEARCĂ Creează un tabel de test cu texte scurte și lungi. Compară aceleași rânduri selectând 1, 5, 10, 20 și 50 de coloane.

IDEEA **Un milion de rânduri nu are un singur cost. Contează și câte date poartă fiecare rând.**

LECȚIA 09 | ÎNȚELEG CE SE ÎNTÂMPLĂ

Aceeași muncă utilă, de la o instanță la un miliard de instanțe noi

Acesta este un calcul orientativ, nu un benchmark inclus în manifestul public. Păstrez exact un miliard de actualizări și arăt cum se schimbă alocarea când instrucțiunea new este mutată în interiorul repetării.

1. Presupun o clasă mică pentru care un obiect ocupă 24 de octeți, aceeași dimensiune observată în demo-ul public cu zece milioane de obiecte.

2. Execut un miliard de actualizări utile în toate variantele. Cu un singur obiect, el primește toate actualizările. Cu 1.000 de obiecte, fiecare primește un milion. Cu un miliard de obiecte, fiecare primește o singură actualizare.

3. Coloana de memorie este calculată: numărul obiectelor înmulțit cu 24 de octeți. La un miliard de obiecte rezultă aproximativ 24 GB de alocări cumulative.

4. Nu public aici timpi de execuție. Ei trebuie măsurați pe calculatorul cititorului, pentru că JIT-ul, procesorul și colectorul de memorie pot schimba mult rezultatul.

INSTANȚE CREATE	ACTUALIZĂRI / INSTANȚĂ	MEMORIE CALCULATĂ (MB)
1	1.000.000.000	0,000024
1.000	1.000.000	0,024000
10.000	100.000	0,240000
1.000.000	1.000	24,000000
1.000.000.000	1	24.000,000000

ÎNCEARCĂ

Păstrează un miliard de actualizări și schimbă numai numărul de obiecte create. Compară timpul și memoria alocată.

IDEEA

Numărul buclelor nu îmi spune singur numărul instanțelor. Locul instrucțiunii new îl decide.

LECȚIA 10 | REDUC MUNCA INUTILĂ

Prima versiune citea tot setul pentru o singură pagină

Exemplul istoric folosește valori aproximative: 60 de coloane, 100.000 de rânduri și câteva zeci de rânduri afișate.

Prima versiune întorcea aproximativ 60 de coloane pentru aproximativ 100.000 de rânduri. Însemna aproape 6.000.000 de valori pentru o singură pagină. Am schimbat ordinea operațiilor. Selectam mai întâi ID-urile, păstram pagina curentă și abia apoi citeam toate câmpurile pentru acele rânduri.

Cu două citiri întorceam aproximativ 100.000 de ID-uri și 3.000 de valori pentru pagina curentă. În total erau 103.000 de valori, față de 6.000.000. Cu trei citiri, la pagina 6, aveam un COUNT, aproximativ 300 de ID-uri și 3.000 de valori. Totalul era 3.301. La o pagină foarte adâncă erau necesare mai multe ID-uri, iar avantajul trebuia măsurat din nou.

Aceste rapoarte descriu informația întoarsă aplicației, nu pretind că lucrul intern al SQL Server scade automat în aceeași proporție. Filtrul, indexul și sortarea decid cât trebuie să citească și să ordoneze baza. ID-ul era ultimul criteriu al sortării; astfel, egalitatea celorlalte câmpuri nu putea schimba ordinea rândurilor de la o cerere la alta.

6.000.000 Forma inițială	**103.000** Două citiri	**3.301** Trei citiri, pagina 6

ÎNCEARCĂ Desenează traseul unei pagini în patru casete: citire, procesare, dependențe și afișare.

LECȚIA 11 | REDUC MUNCA INUTILĂ

Aceeași pagină, dar 7,65 GB, 4,38 MB sau 0,38 MB

Acesta este un alt laborator SQL separat, cu rânduri sintetice mai late decât în proba anterioară. Vreau să afișez aceleași cincizeci de rânduri dintr-o tabelă mare. Compar trei trasee și verific că pagina finală este identică.

1. Creez un milion de rânduri. Fiecare rând are cincizeci de coloane: ID, numere, date, titluri și descrieri care pot ajunge la 800 de caractere.

2. În primul traseu, un SELECT — instrucțiunea SQL care citește date — întoarce toate cele cincizeci de coloane pentru toate rândurile. Aplicația primește întregul set și păstrează numai cele cincizeci de rânduri ale paginii.

3. În al doilea traseu selectez toate ID-urile ordonate. Aleg ID-urile paginii și execut încă un SELECT pentru toate coloanele numai pentru acele cincizeci de ID-uri.

4. În al treilea traseu execut COUNT pentru numărul total, selectez numai ID-urile necesare până la pagina cerută și citesc toate coloanele numai pentru cele cincizeci de rânduri afișate.

5. Calculez aceeași valoare de control din toate câmpurile paginii. Cele trei trasee întorc aceeași pagină; diferă numai munca făcută pentru a ajunge la ea.

6. Acest laborator reproduce traseul istoric în trei citiri: COUNT, ID-urile până la sfârșitul paginii și rândurile late ale paginii. El nu folosește indexul îngust și Skip/Take din laboratorul următor. De aceea pagina 6 apare cu timpi diferiți: condițiile sunt diferite, nu rezultatul.

VARIANTĂ	VALORI ÎNTOARSE	VOLUM (MB)	TIMP (secunde)
Tot setul: 50 coloane	50.000.000	7.650,00	21,00
Toate ID-urile + pagina	1.002.500	4,38	0,63
COUNT + 300 ID + pagina 6	2.801	0,38	0,10
COUNT + 62.850 ID + pagina 1.257	65.351	0,63	0,15

ÎNCEARCĂ

Repetă experimentul cu date late. Compară tot setul, toate ID-urile și numai ID-urile necesare până la pagina cerută.

IDEEA

Nu mă interesează numai că o variantă este de multe ori mai rapidă. Mă interesează dacă timpul final rămâne mic atunci când volumul crește.

LECȚIA 12 | REDUC MUNCA INUTILĂ

Numărul citirilor era o decizie măsurată

Alegeam varianta care citea și transporta mai puține date.

În cărțile și exemplele SQL pe care le studiam atunci, pagina era prezentată de obicei printr-o singură interogare. Nu porneam de la un model cunoscut cu două sau trei citiri. Am ajuns la ele testând separat ID-urile ordonate, rândurile paginii și COUNT — operația care numără toate rândurile potrivite.

La trei citiri execut mai întâi COUNT, care numără totalul. Apoi folosesc TOP pentru a selecta ID-urile ordonate necesare până la sfârșitul paginii cerute. Din această listă păstrez ID-urile paginii curente și, în a treia citire, încarc toate coloanele numai pentru acele rânduri.

Dacă un filtru WHERE dificil restrângea deja rezultatul la câteva mii de rânduri, două citiri puteau fi mai bune decât trei. Dacă utilizatorul cerea totalul, îl măsuram separat. Nu transformam arhitectura într-o dogmă. Testam una, două și trei citiri cu aceleași date și alegeam costul total mai mic.

01	**O citire**	Filtru, ordine, pagină și câmpuri într-un plan eficient.
02	**Două citiri**	ID-uri, apoi obiecte; setul intermediar este mic.
03	**Trei citiri**	COUNT, ID-uri și obiecte sunt costuri justificate.
04	**Măsor**	Aleg după date, nu după numărul de comenzi.

ÎNCEARCĂ Măsoară separat varianta cu o citire, două citiri și trei citiri. Păstrează varianta care mută cea mai puțină informație.

IDEEA **Numărul interogărilor a fost secundar. Scopul a fost să citesc mai puțină informație.**

LECȚIA 13 | REDUC MUNCA INUTILĂ

Nouă secunde au devenit 0,17 secunde, dar timpul absolut spune povestea mai bine

Acesta este un laborator SQL separat, nu o continuare numerică a tabelului anterior. Reconstruiesc zece milioane de rânduri late, cu altă distribuție sintetică și alt plan SQL, apoi cer aceeași pagină prin două trasee.

1. Creez o tabelă sintetică cu zece milioane de rânduri, cincizeci de coloane și zece descrieri între 50 și 200 de caractere.

2. O pagină are cincizeci de rânduri. Sortez după câmpul ales și apoi după ID, ca aceeași cerere să producă mereu aceeași ordine.

3. Creez un index îngust care conține numai câmpul de sortare și ID-ul.

4. În varianta folosită de obicei, selectez toate cele cincizeci de coloane și aplic Skip/Take direct pe ele.

5. În varianta ID-first, aplic Skip/Take numai pe ID. Apoi selectez toate coloanele numai pentru cele cincizeci de ID-uri ale paginii.

6. Verific aceleași cincizeci de rânduri, aceleași 2.500 de valori și aceeași valoare de control.

7. Măsor timpul complet și citirile logice. O citire logică este un bloc de date consultat de SQL Server, nu un rând întors aplicației.

8. Repet fiecare variantă în trei procese independente și păstrez timpul reprezentativ.

PAGINĂ	RÂNDURI SĂRITE	TOATE COLOANELE (sec)	ID-FIRST COMPLET (sec)	RAPORT
1.257	62.800	0,1660	0,0040	aprox. 40×
10.000	499.950	1,3800	0,0240	aprox. 57×
50.000	2.499.950	6,5500	0,1200	aprox. 55×
70.000	3.499.950	9,2000	0,1670	aprox. 55×

ÎNCEARCĂ Compară Skip/Take pe toate coloanele cu Skip/Take numai pe ID urmat de citirea rândurilor paginii.

IDEEA

Chiar și după un câștig mare, verific dacă timpul final a coborât suficient pentru următoarea creștere.

LECȚIA 14 | REDUC MUNCA INUTILĂ

Aceleași 101.000 de rânduri: o secundă sau o treime de secundă

Acesta este un al treilea laborator SQL separat, cu douăzeci de coloane și o altă încărcătură sintetică. Vreau să văd dacă pot reduce timpul de citire fără să schimb datele. Modulo 4 înseamnă că împart rândurile după restul împărțirii ID-ului la 4; restul poate fi numai 0, 1, 2 sau 3, deci obțin patru grupe fără suprapunere.

1. Creez o tabelă mică de 1.000 de rânduri și o tabelă mare de 100.000 de rânduri. Fiecare rând are douăzeci de coloane: numere, date, titluri și descrieri. În total citesc 2.020.000 de valori, aproximativ 307,55 MB.

2. În prima variantă citesc tabela mică, apoi tabela mare. Sunt două interogări complete executate una după alta.

3. Pentru celelalte două variante păstrez tabela mică drept prima lucrare. Împart tabela mare în patru grupe după restul împărțirii ID-ului la 4: 0, 1, 2 și 3. Fiecare rând intră într-o singură grupă.

4. Rulez cele cinci lucrări mai întâi una după alta. Această probă arată costul împărțirii fără avantajul paralelismului.

5. Rulez apoi aceleași cinci lucrări simultan, pe conexiuni separate.

6. Reunesc rândurile, le ordonez după sursă și ID și calculez aceeași valoare din toate câmpurile. Toate variantele întorc același 101.000 de rânduri.

VARIANTĂ	LUCRĂRI	RÂNDURI	VOLUM (MB)	TIMP (secunde)
Două citiri complete, serial	2	101.000	307,55	1,02
Aceleași cinci părți, serial	5	101.000	307,55	1,10
Aceleași cinci părți, paralel	5	101.000	307,55	0,35

ÎNCEARCĂ Împarte tabela mare în patru grupe cu modulo 4. Rulează întâi serial, apoi paralel și verifică același rezultat.

IDEEA

Modulo 4 îmi dă patru grupe clare. Măsurarea îmi spune cât am câștigat. În acest test, aceeași citire s-a terminat în aproximativ o treime din timp.

LECȚIA 15 | REDUC MUNCA INUTILĂ

Aceleași 300.000 de rezultate: 16,035200 secunde sau 0,015229 secunde

Schimb numai calea de acces la obiectul-copil potrivit. Calculul aplicat fiecărui rezultat și valoarea finală rămân identice.

1. Creez patru niveluri. Primul are 300 de obiecte. Fiecare obiect are zece copii, deci nivelurile următoare au 3.000, 30.000 și 300.000 de obiecte.

2. Cele 300.000 de obiecte de pe ultimul nivel sunt rezultatele utile. Pentru fiecare execut același calcul.

3. În varianta LINQ/Where, pentru fiecare părinte pornesc din nou prin întreaga listă a nivelului următor și verific ce obiecte îi aparțin.

4. În varianta cu array, merg direct la pozițiile copiilor potriviți. Nu mai recitesc întreaga listă pentru fiecare părinte.

5. LINQ/Where face aproximativ 9,09 miliarde de verificări. Array-ul vizitează numai cele 333.000 de obiecte necesare.

6. Ambele variante produc 300.000 de rezultate și aceeași sumă finală. Schimb numai calea de acces la obiectul-copil.

VARIANTĂ	ELEMENTE VERIFICATE	TIMP MEDIU (secunde)	RAPORT	REZULTATE
LINQ/Where repetat	9.090.900.000	16,035200	1×	300.000
Array cu acces direct	333.000	0,015229	aprox. 1.053×	300.000

ÎNCEARCĂ Construiește aceleași legături cu LINQ/Where și cu un array de poziții. Numără verificările, nu numai secundele.

IDEEA **Căutarea poate costa mai mult decât operația făcută după ce am găsit rezultatul.**

LECȚIA 16 | REDUC MUNCA INUTILĂ

25.000 de termeni în 100.000 de documente: aproximativ 41 de ore sau 26 de minute

Vreau să găsesc 25.000 de termeni într-un document mare. Pentru un singur termen, IndexOf este soluția simplă și naturală: parcurge textul și arată unde apare cuvântul. Repetată de 25.000 de ori, această soluție recitește documentul de 25.000 de ori.

1. Creez un document sintetic de aproximativ 200 de pagini, cu 1.105.000 de caractere.

2. Creez 25.000 de termeni. Cele două metode trebuie să găsească exact aceleași 100.000 de potriviri.

3. IndexOf este soluția directă: caut primul termen în întregul text, apoi al doilea și continui până la termenul 25.000. Documentul este parcurs din nou pentru fiecare termen.

4. Aho-Corasick este metoda optimizată pentru multe căutări simultane. Construiesc o singură hartă a termenilor, apoi parcurg documentul o singură dată. Harta se construiește în aproximativ 0,006374 secunde și poate fi refolosită.

5. Tabelul proiectează timpul măsurat pentru un document asupra a 100.000 de documente similare. Nu pretinde că am rulat testul timp de 41 de ore.

METODĂ	UN DOCUMENT (secunde)	100.000 DOCUMENTE (secunde)	TIMP UȘOR DE CITIT	RAPORT
IndexOf repetat	1,479386	147.938,60	aprox. 41 h 6 min	1×
Aho-Corasick	0,015784	1.578,43	aprox. 26 min 18 s	aprox. 94×

ÎNCEARCĂ Construiește harta Aho-Corasick o singură dată, apoi compară aceleași potriviri cu IndexOf repetat.

IDEEA **Când caut 25.000 de lucruri, încerc să nu parcurg același document de 25.000 de ori.**

LECȚIA 17 | REDUC MUNCA INUTILĂ

Când mai multe fire pe același calculator nu rezolvă problema

Un fir este un traseu de lucru executat de calculator. Punctul de plecare este o procesare video în care un singur fir execută lucrările una după alta, pe un singur calculator. Estimarea este de douăsprezece zile.

1. În varianta inițială, un singur calculator procesează lucrările una după alta pe un singur fir. Timpul estimat pentru întregul lot este de douăsprezece zile.

2. Încerc mai multe fire pe același calculator. Câștigul nu este suficient, deoarece lucrările împart același procesor, aceeași memorie și aceleași limite locale.

3. Lucrările video sunt independente. Un coleg propune Azure Batch, un serviciu care poate porni mai multe calculatoare și trimite câte o lucrare fiecăruia.

4. Împart lotul, păstrez un identificator pentru fiecare lucrare și reunesc rezultatele la final. O lucrare eșuată poate fi reluată fără să repet tot lotul.

5. Tabelul arată împărțirea matematică ideală a celor douăsprezece zile. Într-o execuție reală adaug pornirea calculatoarelor, transferul, așteptarea, reluările, limitele serviciului și costul financiar.

LUCRĂRI SIMULTANE	CALCULATOARE	LIMITĂ IDEALĂ	CE ÎNSEAMNĂ
1	1	12 zile	punctul de plecare
50	până la 50	5 h 46 min	împărțire ideală
100	până la 100	2 h 53 min	împărțire ideală
200	până la 200	1 h 26 min	împărțire ideală

ÎNCEARCĂ Înainte de distribuție, verifică dacă lucrările sunt independente și dacă rezultatele pot fi reunite și reluate sigur.

IDEEA

Uneori nu trebuie să forțez același calculator. Trebuie să împart problema în unități care pot pleca pe calculatoare diferite.

LECȚIA 18 | REDUC MUNCA INUTILĂ

Un milion de linii și două soluții care costă prea mult

Un script PowerShell trebuia să creeze un fișier de aproximativ un milion de linii. Fișierul era necesar; problema era felul în care îl construiam.

Prima variantă scria fiecare linie imediat. Rezultatul era corect, dar producea aproximativ un milion de operații de fișier. Deschiderea, scrierea și sincronizarea repetate deveneau costul dominant.

A doua variantă păstra toate liniile într-un singur text. În PowerShell, textul este imuabil: la fiecare adăugare se creează un text nou, se copiază tot istoricul și se adaugă linia următoare. Pentru patru unități, munca este 1 + 2 + 3 + 4 = 10, deși rezultatul final are numai patru unități.

Aveam nevoie de același fișier final, dar fără un milion de scrieri și fără reconstruirea repetată a unui singur text uriaș.

Linie cu linie	Un singur text	Ținta
≈ 1.000.000 de scrieri	Istoricul este recopiat	Același fișier final

ÎNCEARCĂ Construiește același text prin cele patru metode. Verifică același hash, apoi compară timpul și alocarea cumulată.

IDEEA

Voiam același fișier fără un milion de scrieri și fără reconstruirea repetată a unui singur șir uriaș.

LECȚIA 19 | REDUC MUNCA INUTILĂ

Zece înregistrări spuneau da. Un milion spunea nu

În februarie 2004, Web Forms și Web Controls erau drumul obișnuit în ASP.NET. Componentele vizuale și evenimentele ajutau la construirea rapidă a primei pagini.

Am construit o probă în C#. Cu zece înregistrări, pagina funcționa bine. Cu un milion, se bloca. Volumul mare a arătat că nu voiam să continui în aceeași direcție.

Nu am încercat paginarea ID-first în interiorul paginii cu Web Controls. În aproximativ cinci minute am decis să construiesc un motor diferit, pornind de la obiectul de business, context și acțiune.

Schița era cli;list;detail;save;5. cli însemna obiectul Client. list era ecranul pe care voiam să ajung. detail era ecranul din care venea cererea. save era acțiunea. 5 era ID-ul clientului existent. Pentru creare foloseam ID-ul -1.

Linia nu era motorul terminat. Era harta lui. Aceleași etape puteau fi folosite pentru Client, Factură sau Utilizator. În propriul model am aplicat paginarea ID-first, iar proba cu un milion de rânduri a coborât sub o secundă.

cli Obiectul Client	**list / detail** Destinație și origine	**save / 5** Acțiune și ID

ÎNCEARCĂ Ia o pagină existentă și scrie pe o singură foaie ce este obiect, context, acțiune și identificator.

IDEEA

În cinci minute nu am construit motorul. Am ales direcția în care aveam să îl construiesc timp de nouă luni.

LECȚIA 20 | REDUC MUNCA INUTILĂ

O modificare trebuia făcută o dată, nu în fiecare pagină

Web Controls erau componente ASP.NET care ajutau la construirea rapidă a unei interfețe web. Prima pagină se scria ușor; întrebarea era ce se întâmplă când modelul ajunge la zeci de pagini.

Cu Web Controls puteam construi repede prima pagină, dar fiecare pagină nouă pornea cu multă muncă proprie. Dacă o regulă comună se schimba, trebuia să caut toate locurile în care fusese repetată. Motorul meu păstra pașii comuni într-un singur loc și lăsa aplicației configurația și comportamentul specific.

Configurațiile aplicației stăteau în fișiere XML — fișiere text structurate care descriau diferențele. Nu încărcam web.config, fișierul general de configurare al aplicației, cu detaliile fiecărui obiect. Astfel puteam vedea clar ce aparține motorului, ce aparține aplicației și ce trebuie schimbat atunci când apare un nou tip de pagină.

01	**3 pagini**	Duplicarea se ascunde; costul pare acceptabil.
02	**40 de pagini**	Duplicarea se multiplică în multe locuri.
03	**Motor**	Pașii comuni rămân într-un singur loc.
04	**XML**	Aplicația declară diferențele fără să rescrie motorul.

ÎNCEARCĂ Alege trei clase asemănătoare și notează ce poate fi comun fără să ascunzi diferențele reale.

IDEEA

Problema era numărul locurilor în care trebuia repetată aceeași decizie, nu numărul paginilor.

LECȚIA 21 | REDUC MUNCA INUTILĂ

AI poate construi și multiplica foarte repede un model bun

Odată ce primul exemplu este clar, clasele asemănătoare pot fi create mult mai repede.

Pot da AI-ului o clasă completă, convențiile, testele și diferențele noului obiect. El poate propune rapid Client, Factură sau Utilizator pe aceeași structură. Poate completa validările, logurile și testele repetitive și poate compara implementările.

Modelul îmi aduce rapiditate pentru că nu reiau de la zero aceleași decizii. Îmi aduce siguranță pentru că logul și testele sunt deja în exemplu. Îmi aduce un rezultat mai ușor de prevăzut pentru că următorul obiect urmează pași cunoscuți. AI mărește aceste avantaje, dar poate multiplica la fel de repede și o greșeală.

ÎNCEARCĂ	Înainte să generezi a doua clasă, scrie regulile pe care prima clasă le-a demonstrat deja.
IDEEA	**AI multiplică modelul. Decizia despre ce merită multiplicat rămâne la mine.**

LECȚIA 22 | CONSTRUIESC CONTINUITATE

Treizeci de minute pentru câțiva oameni. Mult mai mult pentru întregul șantier

Cele treizeci de minute nu au dispărut. S-au întors sub forma unei munci mai grele și mai scumpe.

În clipa alegerii, situația pare mică și personală: terminăm pauza sau ne ridicăm acum? Dar consecința nu se măsoară numai în timpul fiecărui om. Se măsoară în munca întregii echipe, în betonul pierdut, în transport, în întârzierea șantierului și în reluarea lucrării.

Această idee a rămas în modul meu de lucru. Privesc mai întâi decizia mică. Apoi adun toate efectele ei asupra echipei și asupra sistemului. Ce pare ieftin într-un singur loc poate deveni foarte scump după ce număr toate etapele noi.

ÎNCEARCĂ Alege o acțiune scurtă care blochează alte activități și notează costul dacă o amâni o zi.

IDEEA **Cele treizeci de minute economisite au mutat munca într-o formă mult mai grea și mai scumpă.**

LECȚIA 23 | CONSTRUIESC CONTINUITATE

De la 0,25 secunde la aproape 40 de secunde

Un cost mic, multiplicat săptămânal, nu mai rămâne mic.

Pornesc de la 0,25 secunde și adaug cinci procente în fiecare săptămână. După 104 săptămâni, rezultatul matematic este aproximativ 40 de secunde. Într-o aplicație reală, curba nu va fi atât de regulată. Unele săptămâni nu se schimbă nimic, iar într-altele apare un salt. Exemplul arată însă cât de ușor ignorăm o pierdere care pare suportabilă izolat.

Timpul poate crește din multe cauze. Pot exista mai multe date, o buclă nouă, un apel extern sau un index dispărut din baza de date. Dacă văd numai timpul final, toate aceste cauze se amestecă în aceeași propoziție: pagina merge greu.

ÎNCEARCĂ Alege o pagină și păstrează săptămânal p50, p95, volumul și versiunea aplicației.

IDEEA

Cinci procente nu sperie într-o vineri. Repetate timp de doi ani, schimbă natura problemei.

LECȚIA 24 | CONSTRUIESC CONTINUITATE

Faptul că merge astăzi este numai punctul de plecare

O demonstrație confirmă funcția de astăzi. Istoricul arată cum se schimbă aplicația.

Pornesc de la ceea ce există: utilizatori, rânduri, pagini, reguli, integrări, timpi și erori. Apoi caut istoria. Cât de repede au crescut datele? Câte excepții au apărut? Câte locuri trebuie schimbate pentru o regulă nouă? Cât din timp aparține sistemului și cât dependențelor externe?

Nu extrapolez mecanic o singură lună. Compar perioade, volume și tipuri de execuție. Normalizez timpul după volumul procesat înainte să spun că sistemul a devenit mai lent. Dacă numărul de rânduri rămâne stabil și timpul urcă, caut o versiune mai lentă, o cale nouă sau o dependență care s-a schimbat.

Astăzi Starea actuală	**Istoric** Schimbarea în timp	**Mâine** Scenariul

ÎNCEARCĂ Ia o funcție care merge astăzi și scrie trei scenarii de creștere pe care le poți testa.

IDEEA **Compar timpul cu volumul și construiesc scenariile pornind din istoric.**

LECȚIA 25 | CONSTRUIESC CONTINUITATE

Protecția putea fi uitată

Codul de test putea modifica date reale, iar oprirea depindea de un pas făcut manual.

Am folosit practica pentru că oferea acces la cazurile reale care lipseau din datele de test. Rulam puține exemple, alese atent, și câștigam timp. Execuțiile reușite au creat impresia că metoda era sub control.

Într-o zi, sub oboseală, pasul de protecție a fost uitat. Codul a executat operații acolo unde nu trebuia, iar date reale au trebuit refăcute. Problema nu a fost că am ales conștient să șterg date. Problema a fost că sistemul nu refuza operația atunci când contextul ieșea din atenția mea.

01	**Câștig imediat**	Testez repede cazuri reale și variate.
02	**Repet fără incident**	Practica începe să pară normală.
03	**Uit protecția**	Atenția cedează într-o zi obișnuită.
04	**Plătesc riscul**	Operația ajunge la datele reale.

ÎNCEARCĂ Transformă o regulă bazată pe atenție într-o verificare automată care poate refuza execuția.

IDEEA

Dacă siguranța depinde de faptul că îmi voi aminti de fiecare dată, nu am o protecție. Am doar o speranță.

LECȚIA 26 | CONSTRUIESC CONTINUITATE

Un sistem pe care altcineva îl poate continua păstrează metoda, nu persoana

Nu vreau ca următoarea problemă să înceapă cu întrebarea «unde este autorul?»

Logul păstrează execuția. Testele păstrează rezultatul. Reperele păstrează informația despre limită. Modelul păstrează separarea dintre comun și specific. Documentele păstrează deciziile. Exemplele arată cum se adaugă următorul obiect. Niciuna dintre acestea nu înlocuiește oamenii, dar le permite să lucreze fără dependență de memoria unei singure persoane.

Acesta este capătul firesc al căii critice: nu numai o pagină mai rapidă, ci un mod de lucru care poate fi văzut, verificat și transmis. Dacă sistemul poate continua fără mine, contribuția mea nu dispare. Devine parte din felul în care sistemul se explică și evoluează.

ÎNCEARCĂ Roagă un coleg să explice traseul unei funcții fără ajutorul tău și notează ce lipsește din model.

IDEEA

Cel mai bun semn că am lăsat ceva valoros este că lucrarea continuă fără să mă transforme într-un punct unic de eșec.

ÎNAINTE DE URMĂTOAREA SCHIMBARE

Zece întrebări care păstrează drumul clar

Nu am nevoie de toate răspunsurile înainte să încep. Am nevoie de o întrebare suficient de clară pentru următoarea probă.

01 **Ce rezultat funcțional trebuie să rămână identic?**	02 **Cât durează acum și care este volumul?**
03 **De câte ori se repetă operația?**	04 **Unde se multiplică munca?**
05 **Care este primul punct de rupere?**	06 **Ce îmi arată logul în producție?**
07 **Ce muncă pot elimina, nu doar accelera?**	08 **Cum verific că rezultatul a rămas corect?**
09 **Ce se întâmplă dacă volumul crește de zece ori?**	10 **Poate altcineva să continue fără memoria mea?**

CALEA CRITICĂ

Încep cu lucrurile simple, le măsor, urmăresc ce se întâmplă atunci când cresc datele și repetările și schimb numai ceea ce pot verifica. Apoi las metoda suficient de clară pentru a putea fi continuată.

COD ȘI REZULTATE PUBLICE

https://github.com/IoanMarian2001/DemoCriticalPath
Versiunea exactă folosită de carte: book-v1.0.0