

Performance-Check – Beispielbefund

Arbeitsprobe auf synthetischen Testdaten aus dem datrexx-SQL-Labor

Was dieses Dokument ist: eine Arbeitsprobe, die Aufbau und Detailtiefe eines Befunds zeigt. Die Befunde sind typische Ursachen langsamer SQL-Server-Datenbanken, die im Labor gezielt nachgestellt und per Skript gemessen wurden. Es gibt keinen Kunden und keine Kundendaten. In einem echten Performance-Check stehen hier Ihre Abfragen, Ihre Messwerte und die Maßnahmen für Ihr System; dazu gehört dann auch die Analyse von Wartezeiten und Sperren, die im Labor nicht nachgestellt ist.

Gegenstand	Labor-Datenbanken auf Microsoft SQL Server, synthetische Testdaten: Kunden-, Konten- und Betriebstabelle mit je 10 Mio. Zeilen (Befunde 1–3, 5), dazu 20 Mio. Bestellungen und 40 Mio. Positionen (Befund 4); für die Hardware-Frage 100 Mio. Kunden, für die Partitionierung 100 Mio. Buchungen
Messstand	Messläufe vom 15.10.2026
Messverfahren	Per Skript; Ausführungsplan vorab erzeugt und nicht mitgemessen. Kalter Cache heißt: vor jeder Messung Datenpuffer geleert – das geht nur auf einem Testsystem, nicht im Produktivbetrieb. Je Messreihe: Befunde 1 und 4 Mittel aus 3 Messungen, bei Mittel unter 100 ms aus 20, kalter Cache · Befunde 2 und 3 sowie Abschnitt 4 Mittel aus 3 Messungen, kalter Cache · ID-Listen (Befund 5) Mittel aus 5 Aufrufen, kalter Cache, Kompilieren mitgemessen · Plan-Cache (Befund 5) bei warmem Datencache, weil der Plan-Speicher Gegenstand ist
Messgrößen	Laufzeit (ms), logische Lesevorgänge (8-KB-Seiten), CPU-Zeit, Plan-Cache-Belegung, Speicherbedarf der Indizes

1. Kurzfassung für die Leitung

Die Datenbank ist nicht zu klein dimensioniert, sondern liest zu viel. Fünf Ursachen erklären die langen Antwortzeiten; alle lassen sich ohne neue Hardware beheben, die mit dem kleinsten Eingriff (Befund 1) sogar ohne Änderung an Anwendung oder Abfragen. Mehr Kerne und mehr Speicher brachten in der Messung ohne Index nur Faktor 1,7 (Abschnitt 4).

RANG	BEFUND	VORHER	NACHHER	EINGRIFF
1	Häufige Suchabfrage liest die ganze Tabelle (fehlender abdeckender Index)	1.906 ms	5,3 ms	ein Index; Anlegen im Wartungsfenster
2	Anwendung übergibt Text als nvarchar an eine varchar-Spalte mit SQL-Sortierung	446 ms	2,7 ms	Parametertyp in der Anwendung
3	Endungssuche LIKE '%berg' kann den Index nicht gezielt nutzen	3.890 ms	6,3 ms	berechnete Spalte, Index, Abfrage
4	Prozedur ohne Parameter rechnet alles, gefiltert wird erst außen	134.172 ms	54 ms	Prozedur und Aufrufer
5	Ad-hoc-SQL und lange IN-Listen füllen den Plan-Cache	781 MB	0,1 MB	Aufrufe in der Anwendung
–	Mehr Kerne und Speicher (8 statt 2 Kerne, zehnfacher Speicher)		nur Faktor 1,7	nicht empfohlen

Rang nach Nutzen und Eingriff: zuerst, was ohne Änderung an Anwendung oder Abfragen wirkt; dann Änderungen mit Anteil in Anwendung, Abfrage oder Aufrufen, gewichtet nach Wirkung und Aufwand; Hardware zuletzt. „vorher/nachher“ ist die Laufzeit der

2. Befunde im Detail

Befund 1 – Suchabfrage liest die ganze Tabelle

Die Kundensuche nach Nachname, Stadt und Geschäftsjahr findet 20 Treffer, liest dafür aber die gesamte Tabelle mit 10 Mio. Zeilen (Ausführungsplan: *Clustered Index Scan* mit Parallelisierung). Ein Index nur auf dem Nachnamen hilft nicht: Der Optimizer ignoriert ihn, weil die vielen Nachschlagezugriffe teurer wären als der Scan.

```
SELECT Email FROM dbo.Kunde
WHERE Nachname = @n AND Stadt = @s AND GeschaeftsJahr >= @j;
```

VARIANTE	LAUFZEIT	LESEVORGÄNGE	INDEX-SPEICHER
ohne passenden Index	1.906 ms	349.716	–
Index nur auf Nachname	1.843 ms	349.667	409 MB
Kombi-Index (Nachname, Stadt)	71 ms	2.686	625 MB
abdeckender Index (empfohlen)	5,3 ms	11	1.159 MB

Die Laufzeiten stammen aus der Variante mit Literalwerten; die parametrisierte Abfrage nutzt denselben Index Seek mit gleich vielen Lesevorgängen (alle Varianten: www.datrexx.de/sql-labor-messdaten.html).

Maßnahme:

```
CREATE NONCLUSTERED INDEX IX_Kunde_Nachname_Stadt_Cov ON dbo.Kunde (Nachname, Stadt)
INCLUDE (GeschaeftsJahr, Email) WITH (MAXDOP = 4, SORT_IN_TEMPDB = ON);
```

Wirkung: ×358 schneller, ×31.792 weniger Lesevorgänge.

Erfolgskriterium: unter 100 Lesevorgänge und unter 50 ms bei kaltem Cache – erfüllt.

Nebenwirkung: 1.159 MB zusätzlicher Speicher; das Einfügen des Testpakets dauerte mit diesem Index 5,1 s. Ohne `ONLINE = ON` (nur Enterprise Edition) sperrt das Anlegen Schreibzugriffe – daher im Wartungsfenster anlegen.

Rückweg: `DROP INDEX IX_Kunde_Nachname_Stadt_Cov ON dbo.Kunde;`

Geprüfte und verworfene Variante: Ein gefilterter Index nur für aktive Geschäftsjahre braucht lediglich 86 MB, das Einfügen dauerte 1,7 s. Mit Literalwerten liest die Abfrage 5 Seiten – mit Parameter ohne `OPTION (RECOMPILE)` fällt sie aber auf 1.878 ms und 349.538 Lesevorgänge zurück: Ein parametrisierter Plan muss für jeden möglichen Wert gelten und darf den gefilterten Index deshalb schon beim Kompilieren nicht wählen. Für eine Anwendung mit parametrisierten Aufrufen daher nicht empfohlen.

Befund 2 – Falscher Datentyp im Parameter

Die Kontonummer ist als `varchar` mit SQL-Sortierung gespeichert; die Anwendung übergibt den Suchwert als `nvarchar` (Standard vieler .NET-Zugriffe). SQL Server muss jeden Indexeintrag umwandeln: Der Plan zeigt die Warnung `CONVERT_IMPLICIT` und einen Index Scan statt eines Index Seek. Mit Windows-Sortierung tritt der Effekt nicht auf.

VARIANTE	LAUFZEIT	LESEVORGÄNGE	CPU-ZEIT
Parameter <code>nvarchar</code>	446 ms	27.469	3.011 ms
Parameter <code>varchar</code> (passend)	2,7 ms	3	0 ms

Maßnahme: Parametertyp in der Anwendung an den Spaltentyp angleichen (z. B. Spaltentyp im Entity-Framework-Mapping als varchar festlegen).

Nebenwirkung: keine an der Datenbank; Änderung in der Anwendung mit deren Tests absichern.

Erfolgskriterium: keine CONVERT_IMPLICIT-Warnung, Index Seek statt Index Scan – erfüllt.

Rückweg: Mapping-Änderung zurücknehmen (ein Commit).

Befund 3 – Endungssuche ohne gezielt nutzbaren Index

WHERE Firma LIKE '%berg' beginnt mit einem Platzhalter; selbst mit Index auf Firma liest SQL Server alle Indexseiten. Gemessen wurde eine Zählabfrage (COUNT); liest die Abfrage weitere Spalten, kommen je Treffer Nachschlagezugriffe hinzu.

VARIANTE (ZÄHLABFRAGE)	LAUFZEIT	LESEVORGÄNGE	CPU-ZEIT
LIKE '%berg' mit Index auf Firma	3.890 ms	49.022	22.146 ms
berechnete Spalte REVERSE(Firma) mit Index	6,3 ms	43	5,3 ms

```
ALTER TABLE dbo.Betrieb ADD FirmaRev AS REVERSE(Firma) PERSISTED;  
CREATE NONCLUSTERED INDEX IX_Betrieb_FirmaRev ON dbo.Betrieb (FirmaRev);  
-- Abfrage: WHERE REVERSE(Firma) LIKE N'greb%'
```

Wirkung: ×617 schneller, ×1.140 weniger Lesevorgänge (Zählabfrage).

Nebenwirkung: PERSISTED schreibt jede Zeile der Tabelle neu (Wartungsfenster; da REVERSE deterministisch ist, geht der Index auch ohne PERSISTED). Ein Index auf einer berechneten Spalte verlangt bestimmte SET-Optionen (u. a. QUOTED_IDENTIFIER, ANSI_NULLS); schreibende Aufrufer mit abweichenden Einstellungen schlagen danach mit Fehler 1934 fehl – vorher prüfen.

Erfolgskriterium: Index Seek, unter 100 Lesevorgänge – erfüllt.

Rückweg: Index und berechnete Spalte entfernen, alte Abfrage wiederherstellen.

Befund 4 – Prozedur rechnet alles, gefiltert wird erst außen

Eine Prozedur ohne Parameter liefert den Umsatz aller 10 Mio. Kunden; der Aufrufer filtert danach auf eine Stadt. Über eine View oder eine Prozedur mit Parameter erreicht der Filter dagegen den Index.

VARIANTE	LAUFZEIT	LESEVORGÄNGE
Prozedur ohne Parameter, außen gefiltert	134.172 ms	44.110.987
Mehranweisungs-Funktion, außen gefiltert	53.095 ms	13.668.491
View, Filter von außen	52 ms	3.408
Prozedur mit Parameter (empfohlen)	54 ms	3.620
Vergleich: direkte Abfrage	50 ms	3.410

Maßnahme: Prozedur mit Parameter für die Stadt; die Aufrufer übergeben ihn.

Nebenwirkung: Aufrufer müssen angepasst werden; die Ergebnismenge vorher und nachher per Prüfsumme vergleichen.

Erfolgskriterium: Laufzeit höchstens 10 % über der direkten Abfrage – erfüllt.

Rückweg: neue Prozedur unter neuem Namen anlegen (oder den Parameter mit Vorgabewert NULL ergänzen); die alte bleibt unverändert, bis alle Aufrufer umgestellt sind.

Befund 5 – Ad-hoc-SQL und lange IN-Listen füllen den Plan-Cache

Die Anwendung baut Abfragen als Text zusammen. Jede Variante erzeugt einen eigenen Ausführungsplan, der kompiliert und gespeichert werden muss.

MESSUNG	ALS TEXT (LITERAL)	PARAMETRISIERT / TVP
50.000 Einzelaufrufe: Plan-Cache (warmer Datencache)	781 MB (50.000 Pläne)	0,1 MB
Liste mit 2.000 IDs: Laufzeit gesamt	589 ms	222 ms
Liste mit 2.000 IDs: davon Kompilieren	185 ms	0 ms
Liste mit 2.000 IDs: Plan-Cache	6.376 KB	200 KB

Maßnahme: Einzelaufrufe parametrisieren; lange ID-Listen als Table-Valued Parameter (TVP) übergeben.

Nebenwirkung: bei stark ungleich verteilten Werten auf Parameter Sniffing achten (ein für den ersten Wert erzeugter Plan gilt dann für alle); gefilterte Indizes nutzt ein parametrisierter Aufruf ohnehin nicht (Befund 1).

Erfolgskriterium: kein Kompilieren je Aufruf, Plan-Cache-Anteil der Einmal-Pläne sinkt – erfüllt.

Rückweg: Änderung in der Anwendung zurücknehmen.

3. Maßnahmenliste nach Nutzen und Eingriff

RANG	MASSNAHME	WO	ERFOLGSKRITERIUM	RÜCKWEG
1	Abdeckender Index für die Kundensuche	Datenbank	< 100 Lesevorgänge, < 50 ms	DROP INDEX
2	Parametertyp varchar statt nvarchar	Anwendung	keine CONVERT_IMPLICIT-Warnung	Commit zurücknehmen
3	Berechnete Spalte und Index für die Endungssuche	Datenbank + Abfrage	Index Seek, < 100 Lesevorgänge	Spalte und Index entfernen
4	Prozedur mit Parameter für den Umsatz je Stadt	Datenbank + Aufrufer	≤ 10 % über direkter Abfrage	alte Prozedur bleibt unverändert
5	Parametrisierte Aufrufe, TVP für ID-Listen	Anwendung	0 ms Kompilieren je Aufruf	Commit zurücknehmen

Zuerst, was ohne Änderung an Anwendung oder Abfragen wirkt; danach Änderungen mit Anteil in Anwendung, Abfrage oder Aufrufern, gewichtet nach Wirkung und Aufwand; Hardware zuletzt. Jede Maßnahme wird einzeln umgesetzt und nachgemessen.

4. Was nicht empfohlen wird

Mehr Kerne und Speicher

Dieselbe Suchabfrage auf 100 Mio. Zeilen, auf demselben Rechner nachgebildet: einmal gedrosselt (2 Kerne, MAXDOP 1, 1.300 MB Speicher), einmal mit 8 Kernen und 13.000 MB. Plattengeschwindigkeit und Speicherbandbreite sind nicht nachgebildet.

VARIANTE	2 KERNE, 1.300 MB	8 KERNE, 13.000 MB
ohne Index (8-Kern-Profil: 3.465.122 Lesevorgänge)	30.934 ms	17.837 ms
mit abdeckendem Index (8 Lesevorgänge)	1,2 ms	4,6 ms

Ohne Index brachten viermal so viele Kerne und zehnfacher Speicher nur Faktor 1,7. Mit Index war schon das gedrosselte Profil 14.864-mal so schnell wie das starke ohne Index; die Werte mit Index liegen im Bereich weniger Millisekunden, sind grob und sagen nichts über die Hardware aus. Der Index wurde mit vollem Speicher angelegt; im gedrosselten Profil kam das Anlegen in einem einzelnen Lauf nach knapp 2 Stunden nicht voran (Beobachtung, nicht wiederholt). Im Core-Lizenzmodell kosten zusätzliche Kerne außerdem SQL-Server-Kernlizenzen. Neue Hardware wäre erst richtig, wenn nach den Maßnahmen die Messung physische Grenzen zeigt – etwa bei Berichten, die konstruktionsbedingt alle Zeilen lesen müssen.

Partitionierung als Beschleuniger

Auf 100 Mio. Buchungen war die Summe eines Kontos partitioniert langsamer (285 ms) als mit einfachem Index (45 ms), weil die Abfrage alle Partitionen durchsuchen muss. Partitionierung dient vor allem der Datenpflege (einen Monat entfernen: 8,5 ms statt 14.082 ms per DELETE); für Abfragen ersetzt sie den passenden Index nicht.

5. Verwendete Skripte

Alle Werte dieses Dokuments stammen aus diesen Skripten des datrexx-SQL-Labors; in einem echten Check erhalten Sie die Skripte mit Ihren Abfragen, damit Ihr Team jederzeit nachmessen kann.

SKRIPT	MISST
Run-SqlLabor.ps1	Indexstufen 0 bis 5, Laufzeit, Lesevorgänge, Indexgröße, Einfügezeit (Befund 1)
Run-SqlLaborKonvertierung.ps1	Datentyp- und Sortierungskonflikte (Befund 2)
Run-SqlLaborLike.ps1	LIKE-Suchen und berechnete Spalte (Befund 3)
Run-SqlLaborProzedurView.ps1	Prozedur, View, Tabellenfunktionen (Befund 4)
Run-SqlLaborPlanCache.ps1, Run-SqlLaborLangeListen.ps1	Plan-Cache, IN-Listen, TVP (Befund 5)
Run-SqlLaborHardwareVsIndex.ps1, Run-SqlLaborPartition.ps1	Hardware gegen Index, Partitionierung (Abschnitt 4)
Erzeuge-Kennzahlen.ps1, Prüfe-Kennzahlen.ps1	übertragen die Messwerte in dieses Dokument und prüfen jede Zahl

Ausführungspläne und weitere Messreihen: www.datrexx.de/sql-labor-messdaten.html

Ihr System statt Labor: Auf Ihrem Produktivsystem messe ich ohne Eingriff aus Abfragestatistiken, Wartezeiten und Ausführungsplänen; Kaltmessungen wie im Labor nur auf einer Testkopie und nach Ihrer Freigabe. Die wichtigste Maßnahme setze ich nach Ihrer Freigabe um und messe nach. Erstgespräch (20 Minuten, unverbindlich):
manuel.hagen@datrexx.de · +49 178 9364265 · www.datrexx.de