

3.3

Optimieren eines Modells
für bessere Ergebnisse

In 90% der Fälle entstehen Leistungsprobleme durch ungünstige Datenmodelle und/oder DAX.

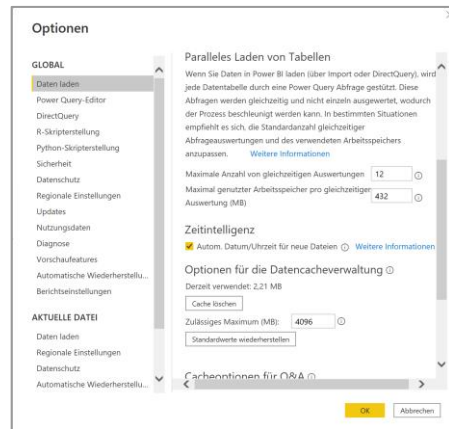
- richtige Datentypen
- Löschen unnötiger Spalten und Zeilen
- Vermeiden von sich wiederholenden Werten
- Ersetzen numerischer Spalten durch Measures
- Reduzieren von Kardinalitäten (n:m vermeiden), Zusammenfassen von Daten, Granularität
- Analyse von Modellmetadaten

Mit Hilfe der Leistungsanalyse lassen sich Elemente identifizieren, die zu Leistungsproblemen beitragen. Für jede Interaktion wird die Dauer angezeigt.

Ansicht → Leistungsanalyse → Aufzeichnungen starten

Vorher

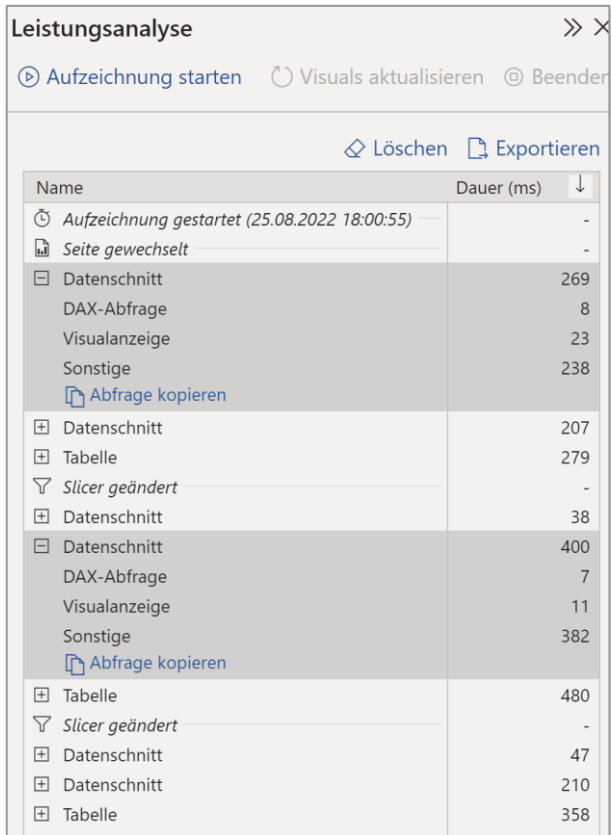
- Leeren des Visualcache: Leere Seite hinzufügen, diese Seite auswählen, Datei abspeichern, Neustart von PowerBI)
- Leeren des Daten-Engine-Cache: Neustart von PowerBI (wird beim Leeren des Visualcache eh gemacht) oder unter Datei → Optionen → Daten laden → Cache löschen



Für jede Interaktion wird die Dauer angezeigt.

- **DAX-Abfragen:** Zeit, die das Visual zum Senden der Abfrage und der Analysis Service zum Zurückgeben der Ergebnisse benötigt
- **Visualanzeige:** Zeit zum Rendern der Darstellung inkl. Abruf von Webbildern oder Geokodierungen.
- **Sonstiges:** Alles weitere wie Vorbereiten von Abfragen, Warten auf die Ausführung anderer Visuals usw.

DAX Studio ist ein Open-Source-Tool, um Abfragen ausführlicher zu analysieren



The screenshot shows the 'Leistungsanalyse' (Performance Analysis) window in DAX Studio. It contains a table with two columns: 'Name' and 'Dauer (ms)'. The table lists various events such as 'Aufzeichnung gestartet', 'Seite gewechselt', and several 'Datenschnitt' (Data Slice) events, each with a corresponding duration in milliseconds. Some rows are expanded to show sub-events like 'DAX-Abfrage' and 'Visualanzeige'.

Name	Dauer (ms)
Aufzeichnung gestartet (25.08.2022 18:00:55)	-
Seite gewechselt	-
Datenschnitt	269
DAX-Abfrage	8
Visualanzeige	23
Sonstige	238
Datenschnitt	207
Tabelle	279
Slicer geändert	-
Datenschnitt	38
Datenschnitt	400
DAX-Abfrage	7
Visualanzeige	11
Sonstige	382
Tabelle	480
Slicer geändert	-
Datenschnitt	47
Datenschnitt	210
Tabelle	358

- **Visuals:** Weniger Visuals je Berichtsseite. Gegebenenfalls andere Möglichkeiten in Betracht ziehen, weitere Informationen darzustellen, z.B. Drillthroughseiten und Quickinfos
- **DAX-Abfragen:** In DAX Studio die einzelnen Schritte überprüfen. Gegebenenfalls Austausch von Funktionen, z.B. FILTER durch KEEPFILTERS
- **Datenmodell:** Reduktion des Datenmodells, z.B. unnötige Spalten und Zeilen löschen. Auslagern von Berechnungen in die vorgelagerten Systeme (z.B. SQL-Datenbank)
- **Autom. Datum/Uhrzeit:** Die Option Autom. Datum/Uhrzeit ist standardmäßig aktiviert, um mit Zeitintelligenz bei Kalendern zu arbeiten. Das ist häufig nicht nötig. Deaktivieren über Datei → Optionen und Einstellungen → Optionen → Daten laden → Zeitintelligenz

Das Verwenden von Variablen in DAX-Ausdrücken hat mehrere Vorteile

- Verbesserte Leistung: Der selbe Ausdruck muss nicht mehrfach ausgewertet werden.
- Verbesserte Lesbarkeit: Variablen haben kurze und selbsterklärende Namen
- Einfacheres Debuggen: Verwendung von Variablen, um Ausdrücke zu testen
- Reduzierte Komplexität: Formeln sind z.T. einfacher

Variablen werden mit dem Schlüsselwort **VAR** definiert.

Ohne Variable

```
Umsatzdelta % ggü VJ =  
    DIVIDE(  
        ([Umsatz] - CALCULATE([Umsatz], PARALLELPERIOD('Datum'[Datum], -12, MONTH))),  
        CALCULATE([Umsatz], PARALLELPERIOD('Datum'[Datum], -12, MONTH))  
    )
```

Mit Variable

```
Umsatzdelta % ggü VJ =  
    VAR UmsatzVJ = CALCULATE([Umsatz], PARALLELPERIOD('Datum'[Datum], -12, MONTH))  
  
    RETURN DIVIDE(([Umsatz] - UmsatzVJ), UmsatzVJ)
```

- Bei Verknüpfungen zwischen Tabellen muss ID-Spalte den gleichen Datentyp haben
- Performanceverbesserung durch Verringerung der Granularität, z.B.
 - Summieren des Umsatzes auf Tages-/Wochenebene (statt Zeitstempel)
 - Mitteln der Temperatur eines Temperatursensors auf 5-Minuten-Werte
- Nachteil ist, dass die Drilldown-Möglichkeit verloren geht. Diese könnten ggf. auf einer gesonderten Drilldown-Seite über DirectQuery-Tabellen realisiert werden

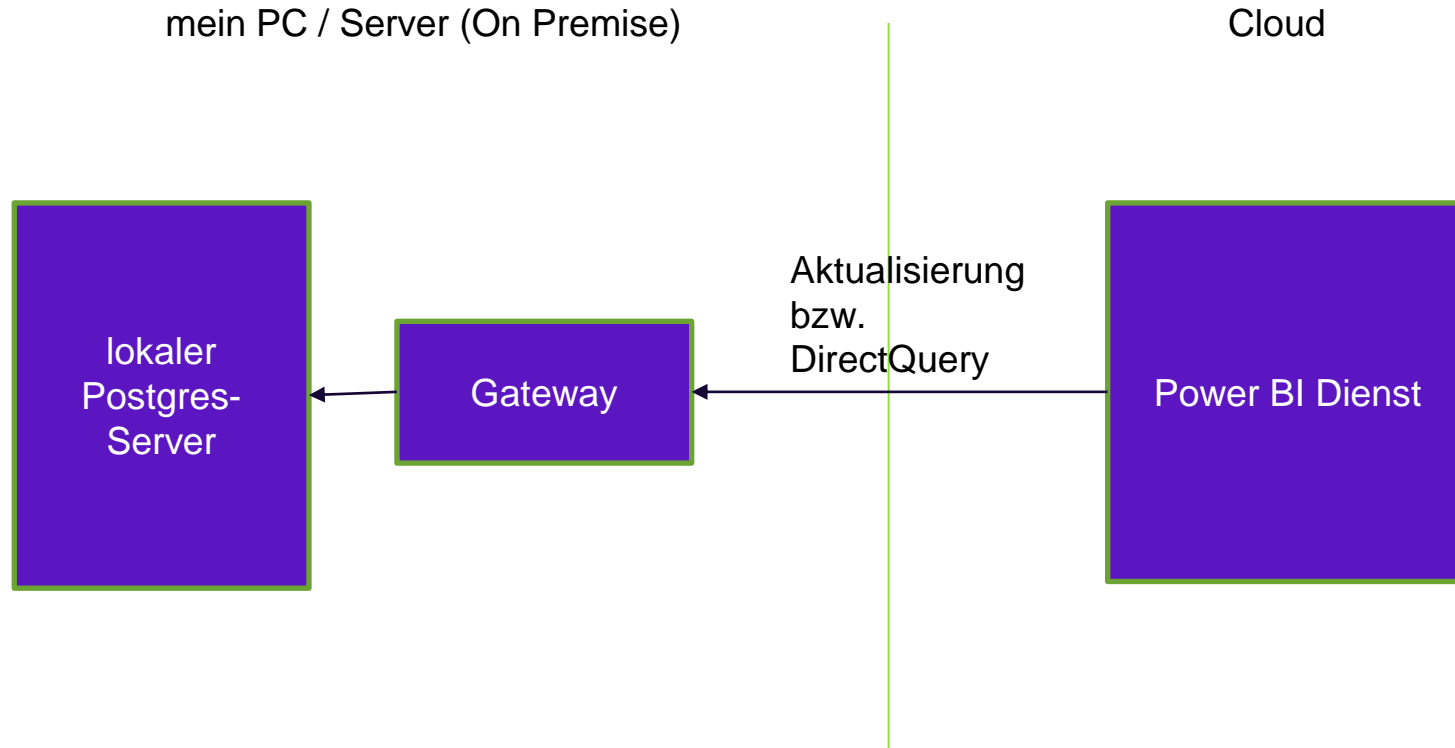
Import: Alles geladen



DirectQuery: Ladevorgang erst bei Anzeige
Das erlaubt z.B. nur einen Ausschnitt zu laden



- Statt Import der gesamten Tabelle wird bei DirectQuery zum Zeitpunkt der Abfrage eine Query an den Server geschickt
- Damit das flüssig funktioniert, ist eine schnelle Antwortzeit des Servers notwendig
 - Jede Benutzeranfrage erzeugt Serverlast, z.B. 20 Visuals x 10 Benutzer = 200 Anfragen an den Server
 - Antwortzeit hängt von Netzwerkgeschwindigkeit und Leistung des Servers ab
 - Ggf. Optimierung der Datenstruktur auf dem Server, z.B. Indizes und keine komplexen Berechnungen in der Abfrage
- Vorteile von DirectQuery:
 - Bei häufigen Datenänderungen, z.B. 15-minütige Aktualisierungen
 - Es können größere Datenmengen verarbeitet werden
 - Datenschutz: Sensible Daten liegen nicht in PowerBI-Datei, sondern nur auf Server
 - Umgang mit mehrdimensionalen Datenquellen (Data Cubes), z.B. SAP Business Warehouse
- Es gibt Einschränkungen bzgl. Datentransformation und Modellierung

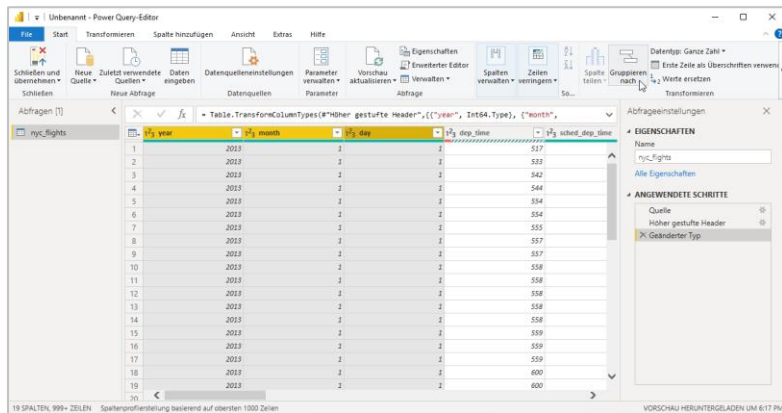


Konfiguration der Abfrageverringeroptionsen zur Verbesserung der Leistung unter
Datei > Optionen und Einstellungen > Optionen > Abfrageverringeroptionsen

- Eine Verringerung der Interaktionen der Visuals untereinander verringert auch die Anzahl Abfragen
- Hinzufügen eines Buttons „Anwenden“ für Slicer- und Filteränderungen

Aggregation der Daten auf das benötigte Detaillevel ist die wichtigste Optimierungsmöglichkeit

- Am besten direkt in der Datenbank durch Anlegen entsprechender Tabellen oder Views
- PowerQuery-Editor: Spalten auswählen, dann „Gruppieren nach“



Verwalten von erstellten Aggregationen über Felder > Tabelle > ... > Aggregationen verwalten

- Anpassen der Aggregationsart

