FileMaker Konferenz

Performance-Bremsen in FileMaker Scripts

Wie unterschiedliche Techniken (Scriptschritte und Funktionen)
die Laufzeit eines Script beeinflussen können.

Ein Vortrag von Robert Kaiser

Robert Kaiser – karo.at

Robert Kaiser

- zertifizierter FileMaker Entwickler (10 bis 19)
- Claris Partner
- Werbegrafiker seit 1993
- FileMaker Entwicklung seit 1996
- Autor für FileMaker Magazin
- Sprecher auf der FileMaker Konferenz 2012, 2013, 2016, 2017, 2022

Robert Kaiser – karo.at

karo productions

- individuelle Lösungen auf der Basis von FileMaker Pro/Go/WebDirect
- DTP-Workflow-Optimierung
- Schulungen (Adobe InDesign, Photoshop, u.a.)
- FMM Award 2014, Beste FileMaker Lösung

Inhalt

- Problemstellung (Optimierungsbedarf) anhand eines Beispiels aus der Praxis
- Optimierungsmöglichkeiten für Scripts und Funktionen
- Benchmarks
- Alternativen
- Fragen bzw. Diskussion zum Thema

Warum dieser Vortrag?

- Beim Optimieren des Beispiel-Scripts entstanden einige Varianten
- Die unterschiedlichen Ansätze und Optimierungen sollten mit Benchmarks bestätigt werden, allerdings ergaben sich auch neue Erkenntnisse
- Keine offizielle Dokumentation über Speicherverwaltung

Bereiche in FileMaker-Lösungen, in denen man etwas gut oder besser machen kann…

Datenbankstrukturen

- Tabellenbreite („schmal“ vs „breit“)
- Feldtypen, Formelfelder
- Aufbau der Beziehungen (Abgleichfelder, Vergleichsoperatoren

Layout

- Anzahl der Elemente am Layout
- Themes, lokale Styles
- Portalfilter (Formeln vs Konstanten)
- Bedingte Formatierungen, Statistikfelder

Business Logik

- Vorabberechnung von Ergebnissen oder Teilergebnissen, z.B. für Statistiken
- Eigene Funktionen

Scripts

- Abläufe
- eingesetzte Scriptschritte und Funktionen

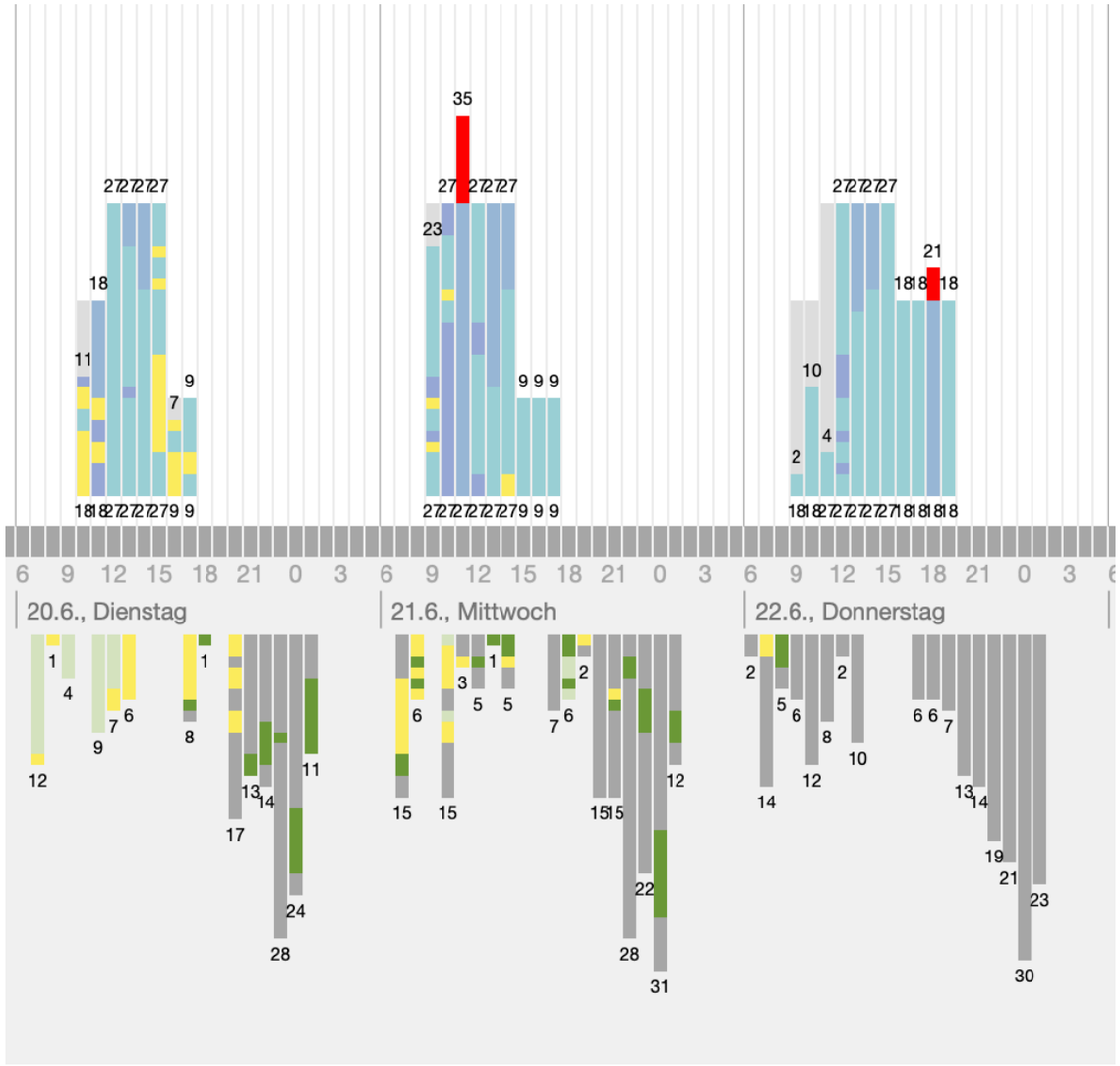
Sonstiges

- SQL ausführen auf offenen DS
- Sortierungen (GTRR)
- Umgang mit Containerfeld

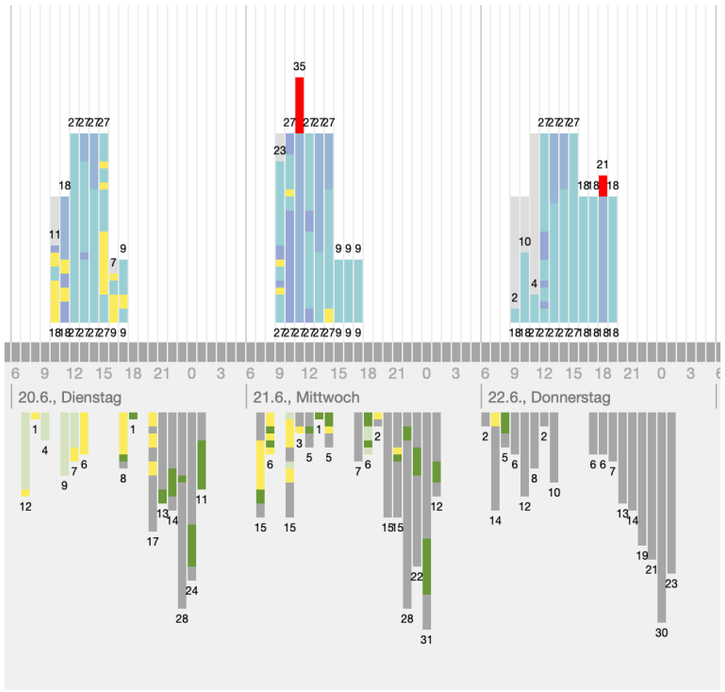
Beispiel aus der Praxis

Beispiel-Lösung „Rollenlogistik“

- Gezeigt als Beispiel für den FMK Vortrag 2022 „FileMaker und SVG Grafiken“



Beispiel-Lösung „Rollenlogistik“




```
----------
Gesamt: 2940 ms v7
Start Aufruf: 12.09.2022 14:08:36,771
Start lokal: 12.09.2022 14:08:36,771
- Scriptdauer: 2916 ms für 1738 DS
-- Einstellungen: 4 ms
-- Daten laden: 120 ms
-- Zeitblock: 63 ms
-- Kapazität: 11 ms
-- Datenausgabe: 2672 ms
-- Beschriftungen: 53 ms
-- Webviewer: 4 ms
```

- Umfangreicher HTML-Code und SVG Grafik werden per FileMaker Script anhand von Templates berechnet und in einem Webviewer ausgeben (ca. 400 KB)
- Daten ändern sich jederzeit und werden am Server laufend aktualisiert (Rollen werden verarbeiten / neu geliefert / vorbereitet)
- Personaleinsatz kann über Diagramm geändert werden
- Script dauert ca. 1 Sekunde (je nach Client / Netzverbindung bzw. Datenanzeige kann es auch bis zu 3 Sekunden dauern)

Erster Optimierungsansatz: Scriptberechnung auf Server statt lokal

Scriptberechnung auf Server statt lokal

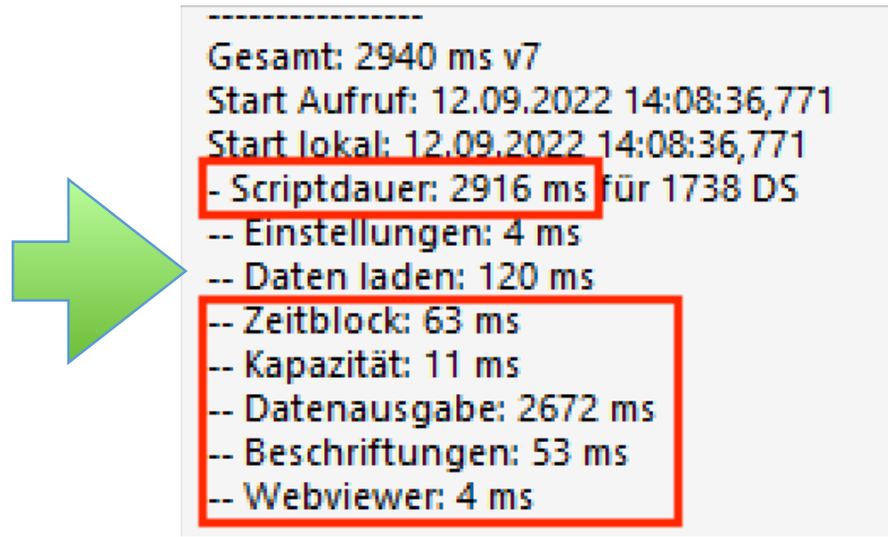
Überlegungen für Ablauf

- Umbau von globalen Feldern auf Session-Datensätze
  (jeder User soll einen Session-Datensatz haben)
- an Server werden nur Parameter für Ansicht (SessionID, Anzahl Tage, Fensterbreite) geschickt
- Overhead starten der PSOS Session?
- Zeit für Download der 400 KB HTML Daten?
- Zeitersparnis? …

Scriptberechnung auf Server statt lokal

Ergebnis

- Die Laufzeit des Scripts ist in diesem Beispiel um Faktor 4 langsamer!



Kein Vorteil, weil

- Die Datenselektion hier keinen großen Anteil an Laufzeit verursacht
- Ressourcen am Server aufgeteilt werden

Benchmarks zu Scriptschritte und Variablen

Benchmarks Scripts

Scripts:

- Jeder Scriptschritt kostet:
  - Auch Kommentarzeilen (inkl. leere Zeilen)
  - –> Dokumentation wenn möglich ausserhalb der Schleife einfügen
- Mehrere Variablen setzen:
  - Scriptschritt „Variable setzen“ vs Funktion „SetzeVar([..]…)“
  - Je nach Konstellation kann auch die neuerliche Anlage lokaler Variablen Zeit kosten:
    SetzeVar([lokaleVar = Wert]…) vs SetzeVar([$lokaleVar = Wert]…)
- Abfragebedingungen: Reihung beachten

Benchmarks Scripts

Scripts, Anmerkungen:

- Die Auswirkungen sind bei Schleifen mit vielen Wiederholungen relevant
- Eine Optimierung zu Lasten der Lesbarkeit ist abzuwägen

Benchmarks zu Funktionen

Benchmark Funktionen

Funktion „HoleWert( )“:

- Je größer ein Textblock, um so langsamer wird das Auslesen einer Zeile
- Das Wertelesen ist nicht indiziert (kein Array), es wird jedesmal neu nach entsprechenden Auftreten des ¶ Zeichen gesucht

Alternativen:

- Falls Daten im Script aufgebaut werden, können auch Variablen mit Wiederholungen verwendet werden:
  Im Beispiel $AnzahlRollen($Zeitstempel) = $AnzahlRollen($Zeitstempel) + 1
- Alternative bei großen Datenmengen: MBS Funktion QuickList
  https://www.mbsplugins.eu/QuickListGetValue.shtml

Benchmark Funktionen

Funktion „Liste( )“:

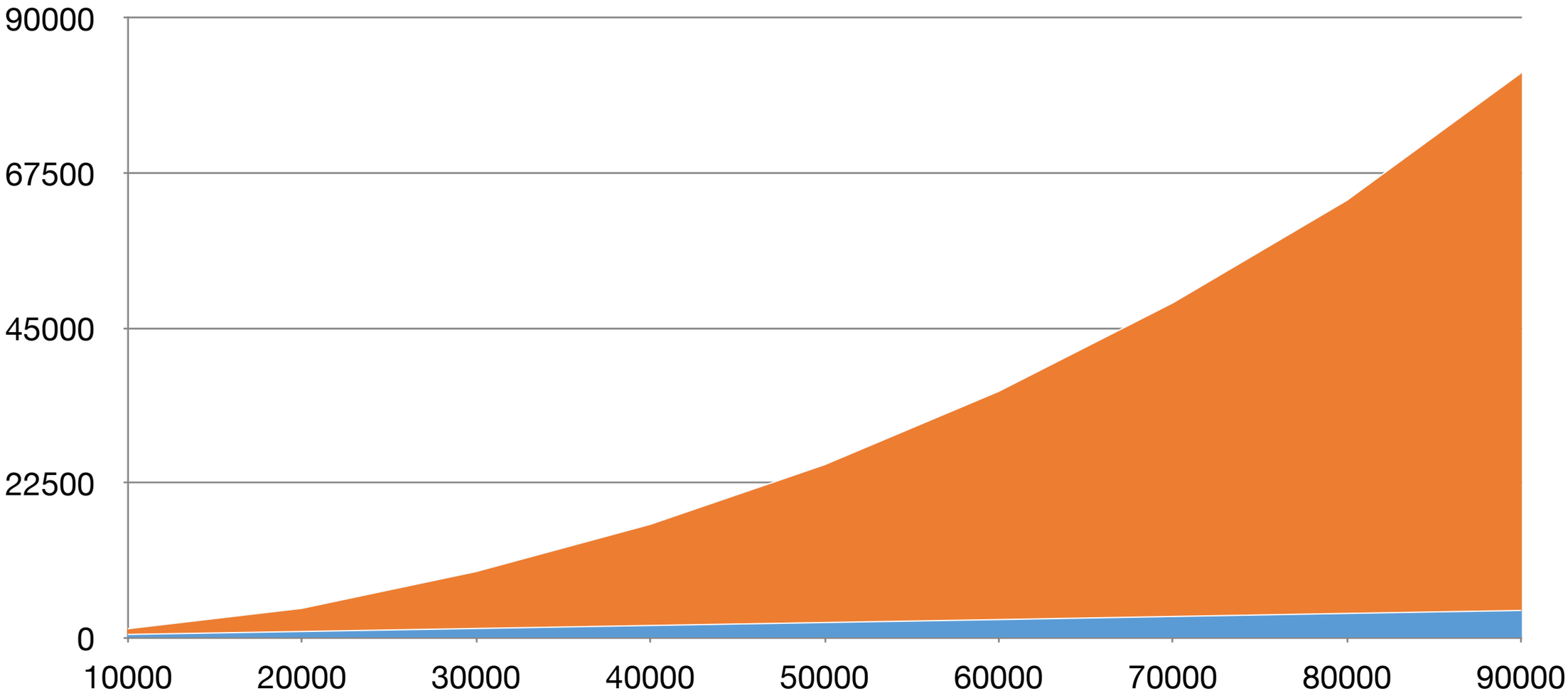
- $MeineListe = Liste($MeineListe; $NeuerWert)
- Bei jeder Zuordnung wird der Originalwert aus $MeineListe mit dem neuen Wert kopiert und neu in den Speicher geschrieben
- Je größer die Datenmenge, umso langsamer wird diese Funktion (**exponentielles** Wachstum der Ausführungszeit!)

Alternativen für Scripts:

- Scriptschritt „Berechneten Wert einfügen“
- Vorhandener Wert wird nicht neu geschrieben, neuer Wert wird „angehängt"
- Die Ausführungszeit steigt hier nur **linear** mit der Datenmenge

Benchmark Funktionen

Funktion „Liste( )“ vs Scriptschritt „Berechneten Wert einfügen“:



Benchmark Funktionen

Funktion Austausch( ):

- Bei Verwendung von Templates mit Platzhalter macht es Sinn zuerst kleineste Textmenge zu ersetzen

Benchmark Funktionen

JSON Funktionen:

- JSON "Objekte" liegen in FileMaker nur als Texte vor
  (in Textfelder oder Variablen)
- Erst bei Ausführung einer FileMaker JSON Funktion wird dieser Text in ein JSON Objekt kompiliert, dass sich dann optimiert nützen lässt.
- Ein neuerlicher Aufruf greift dann auf dieses Objekt zu,
  sodass nachfolgende JSON Funktionen schneller ausgeführt werden.
- **Es kann pro Client (!) nur ein Objekt kompiliert vorliegen!**

Benchmark Funktionen

Das bedeutet:

- Jeder neue Aufruf einer JSON Funktion mit einem anderen JSON Objekt kompiliert dann dieses Objekt neu
- Daher Vorsicht beim Auslesen von großen Datenmengen oder der Manipulation von JSON Objekten.

Die kompilierte Version kann schnell verloren gehen durch die Verwendung von:

- JSON Funktionen bei ScriptParameter/Ergebnis
- JSON Funktionen in eigenen Funktionen oder Formeln in Felddefinitionen
- JSON Funktionen im Layout
  (bedingte Formatierung, Ausblenden, virtuellen Listen …)
- JSON Funktionen in anderen Dateien (z.B. in Subscripts)

Alternativen?

Alternative: Datenaufbereitung in Javascript

Datenmanipulation und Ausgabe in JavaScript:

- Bereitstellung der Rohdaten für Javascript
- Die Berechnung des fertigen SVG Objekts ist in JS **massiv** schneller
- Arrays und JSON Objekte stehen zur Verfügung
- Textmenge des generierten HTML-Codes ist geringer (nur Daten und Javascripts), da SVG erst in JS berechnet wird (ca. 170 KB)

Q & R

Vielen Dank für Ihr Interesse!

Vielen Dank unseren Sponsoren

MonkeyBread Software

Claris

FileMaker Magazin

DATATRUST
BACKUP & ARCHIVIERUNG

24U SOFTWARE

gfm gofilemaker.de
FileMaker Software & News

Ottomatic
MODERN
FILEMAKER HOSTING

JK SOLUTION

karo.at
FileMaker Development