FileMaker Konferenz

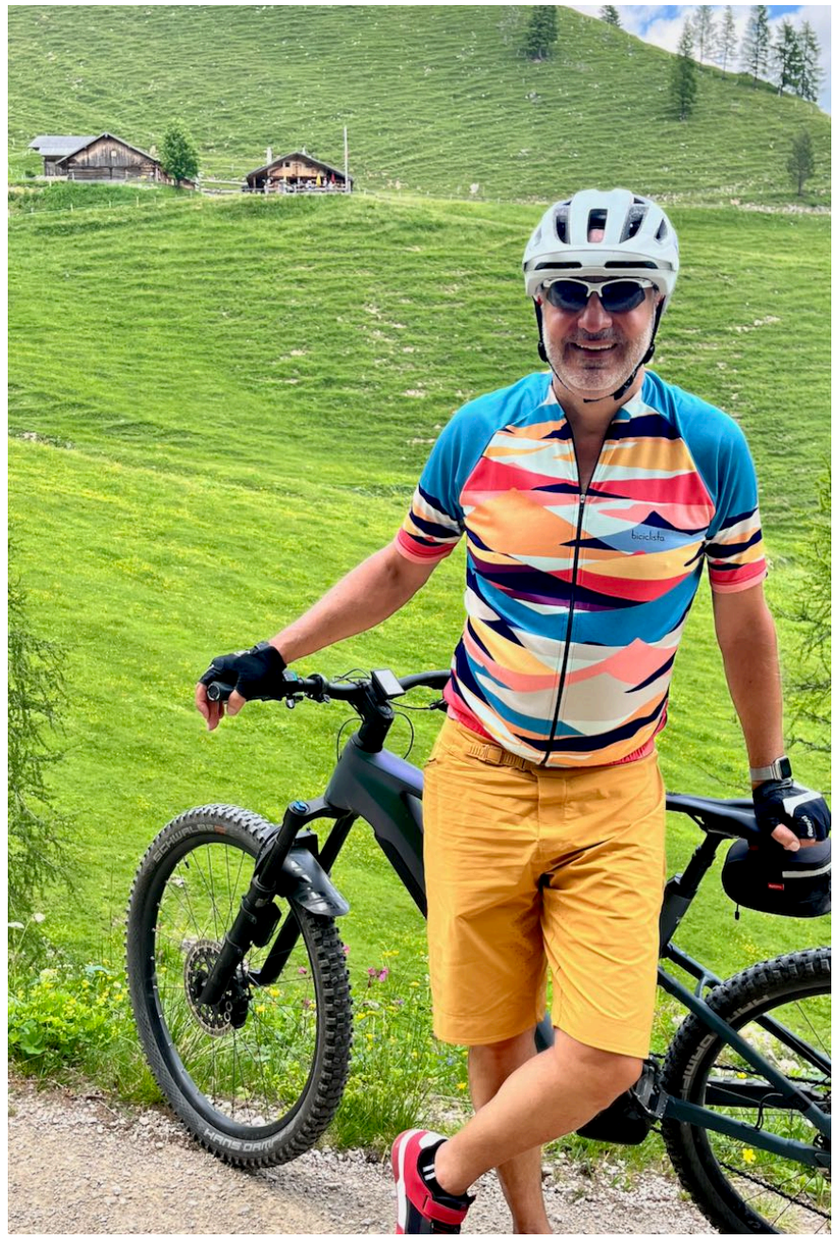
SQL in FileMaker - Pro und Contra

SQL oder nicht SQL, das ist hier die Frage

Stefan Tischler

Über mich

- Geboren und aufgewachsen in Frankfurt/M
- Ausbildung zum Bankkaufmann 1985
- Spezialist Meldewesen Kapitalanlagegesellschaften
- 2000 -2010 Senior Consultant Sungard Systeme
- Arbeit mit FileMaker seit 1985
- seit 2010 selbst. FileMaker Entwickler
- Entwicklung individueller Lösungen verschiedenster Branchen
- Hobby: Mountainbiken



Inhalt dieses Vortrags

- Was ist SQL?
- Warum SQL in FileMaker verwenden?
- Vorteile der Nutzung von SQL in Filemaker
- Nachteile und Grenzen der Nutzung von SQL in Filemaker
- Fallstricke bei der SQL Nutzung
- Performance: Mythen und Fakten
- Fazit
- Fragen und Diskussion
- Nicht Inhalt des Vortrages ist der Scriptschritte „SQL ausführen“ und „SQL-Abfrage in natürlicher Sprache ausführen“

Was ist SQL?

Die Bezeichnung *SQL* wird im allgemeinen Sprachgebrauch als Abkürzung für „**Structured Query Language**“ aufgefasst, obwohl sie laut Standard ein eigenständiger Name ist.

Die Sprache basiert auf der relationalen Algebra, ihre Syntax ist relativ einfach aufgebaut und semantisch an die englische Umgangssprache angelehnt. SQL ist eine Art Universalsprache für Datenbanken, die von den meisten der gebräuchlichsten Datenbanken genutzt wird.

SQLAusführen bzw. ExecuteSQL gibt es in FileMaker seit der Version 12.

SQLeAusführen gibt es in FileMaker seit der Version 21.1

FileMaker SQL Referenzhandbuch gibt es unter: https://help.claris.com/de/sql-reference/content/index.html

Warum SQL in FileMaker verwenden?

- **Was gegen den Einsatz von SQL spricht:**
- FileMaker untypische Logik
- Textbasierte Eingabe
- Limitierter Funktionsumfang (Nur SELECT)
- Evtl. Konvertierung von Ergebnissen

- **Was für den Einsatz von SQL spricht:**
- Aufgeräumter und performanter Beziehungsgraph
- Kontextunabhängige Abfragen (ohne Layoutwechsel oder Beziehung)
- Flexible scriptbare Abfragen statt vorgefertigter FM Suchabfragen
- Herstellung von Beziehungen on-the-fly mit Joins

Vorteile der SQL Nutzung in FileMaker

- Abfragen unabhängig vom Layout: SQLAusführen kann Daten aus Tabellen abrufen, ohne dass diese im aktuellen Layout angezeigt oder mit dem aktuellen Kontext verknüpft sein müssen

- Reduzierung der Komplexität im Beziehungsdiagramm: Entwickler können auf zusätzliche Verknüpfungen verzichten, was die Übersichtlichkeit und Wartbarkeit der Lösung verbessert

- Komplexe Abfragen möglich: Auch komplexe Filterungen, Gruppierungen und Aggregationen lassen sich mit einer einzigen Abfrage durchführen, was mit klassischen FileMaker-Funktionen oft nur mit Workarounds oder mehreren Schritten möglich wäre

- Schnelle Datenabfrage: Bei einfachen und gezielten Abfragen ist SQL oft schneller als klassische FileMaker-Suchvorgänge, da keine Layoutwechsel oder Script-Schleifen notwendig sind

Vorteile der SQL Nutzung in FileMaker

- Wiederverwendbare Skripte und Funktionen: Da die Abfragen unabhängig vom Layout sind, können Skripte und Funktionen universell eingesetzt werden, was die Portabilität und Wartbarkeit erhöht

- Dynamische Parameter: Die Funktion unterstützt Platzhalter und dynamische Parameter, was die Flexibilität erhöht.

- Virtuelle Listen und Berichte: SQL eignet sich hervorragend, um dynamische Listen (z. B. für Berichte, Dashboards oder Javascript Charts) zu erstellen, ohne zusätzliche Layouts oder Beziehungen

- Schnelle Lookup-Funktionen: Für Lookup- oder Validierungszwecke kann man mit SQL gezielt einzelne Werte abrufen, ohne den Kontext zu wechseln

- Suchfunktionen: „Search as you type“-Funktionen (Tannenbaumsuche) lassen sich effizient mit SQL realisieren

Nachteile der SQL Nutzung in FileMaker

- SQLAusführen unterstützt ausschließlich SELECT-Befehle. Datenmanipulation (INSERT, UPDATE, DELETE) ist nicht möglich (Möglich mit MBS-SQL Funktionen)

- Langsame Abfragen bei großen Datenmengen: Besonders bei komplexen Abfragen, JOINs oder GROUP BY kann SQL sehr langsam werden, insbesondere bei großen Tabellen

- Performance-Killer im falschen Kontext: Suchen per SQL in ungespeicherten Formelfeldern oder bei offenen Datensätzen kann die Performance massiv beeinträchtigen

- SQL ignoriert bestehende Beziehungen im Beziehungsdiagramm. Das kann zu Inkonsistenzen führen, wenn Entwickler erwarten, dass FileMaker-typische Zugriffsrechte oder Filter greifen.

- Keine Unterstützung für Wertelisten, Containerfelder oder Formate

- Lange und verschachtelte SQL-Statements sind schwer zu lesen und zu pflegen, besonders für Entwickler ohne SQL-Erfahrung

Performance SQL-Abfragen

- Eigene Tests haben leichte Vorteile für SQL-Suche gegenüber FM interner Suche ergeben (bei einfachen Suchen auf indizierte Felder)
- Suche mit MBS-SQL deutlich langsamer (ca. Faktor 4)
- SQL-Abfragen auf nicht gespeicherte Formelfelder sind langsam
- Der Mythos, dass bei einer Abfrage auf ungespeicherte Formelfelder die komplette Tabelle geladen wird kann nicht bestätigt werden

Fallstricke bei der SQL Nutzung

- Syntax- und Namenskonventionen: Feld- und TO-Namen müssen exakt angegeben werden. Sonderzeichen wie Umlaute oder Leerzeichen können zu Fehlern führen, wenn sie nicht korrekt in Anführungszeichen gesetzt werden. SQL sucht nur In TO-Namen nicht in Tabellennamen
- Änderungen an Feld- oder TO-Namen in FileMaker werden nicht automatisch in SQL-Abfragen übernommen, was zu Fehlern führen kann
- Rückgabe von Zahlen- bzw. Datumswerten im amerikanischen Format, d.h. vorherige Konvertierung vor Weiternutzung
- Verwendung von reservierten Wörtern als Feldnamen führt zu Fehlern
- Fehlende Fehlermeldungen: Bei Fehlern gibt SQLAusführen lediglich ein „?“ zurück. Die Fehlersuche ist dadurch erschwert, da keine detaillierten Fehlermeldungen ausgegeben werden. Auch SQLeAusführen nur bedingt hilfreich
- Es sind nur gleichartige Joins erlaubt (also nur Zahlen- zu Zahlenfeld, bzw. Text- zu Textfeld)

Fallstricke bei der SQL Nutzung

- Beispiel für eine komplexe SQL-Abfrage:

```
SQLeAusführen(
"SELECT Kunden.PLZ
FROM Kunden
JOIN Auftrag ON Kunden.Kundennummer = Auftrag.Kundennummer
WHERE Auftrag.Datum > ?
AND Auftrag.Summe > ?
GROUP BY Kunden.PLZ
ORDER BY Kunden.Kundennummer“;
"";
$Datum;
$Wert)
```

Tricks bei der SQL Nutzung

- Detaillierte Fehlermeldung durch Eingabe der SQL-Abfrage in Datenanzeige
- Indizierte Felder nutzen: Abfragen auf indizierte Felder sind deutlich performanter. In SQL möglichst auf indizierte Felder filtern oder sortieren
- Darauf achten, dass bei Abfragen keine offenen (nicht comittete) Datensätze vorhanden sind
- Feld- und TO-Namen maskieren: Bei Leerzeichen oder Sonderzeichen in Feld- und TO-Namen stets doppelte Anführungszeichen setzen Z.B.:

```
(„SELECT \“ID\“
FROM \“Auftrag\“
WHERE \“Datum\” < ?
AND \“Jahr\“ =?
ORDER BY \“Kundennummer\““;;
Auftrag::Suchdatum;
Auftrag::Jahr)
```

Tricks bei der SQL Nutzung

- WHERE Bedingung der Häufigkeit nach aufsteigend aufbauen
  Falsch: `(SELECT ID FROM Kunden WHERE Sex=? AND PLZ=?;;;"m";"52343")`
  Besser: `(SELECT ID FROM Kunden WHERE PLZ=? AND Sex=?;;;"52343";"m")`
- Dynamische Feldnamen können wie folgt übergeben werden, was allerdings auf Kosten der Lesbarkeit geht

```
SetzeVar (
[Feld = GFN*(HoleFeldname ( Auftrag::Auftrag_ID ))];
SQLAusführen (
 "SELECT " & Feld & "
FROM Auftrag"
; "" ; ""  ))
```

*CF GFN = GetFieldName

```
HoleWert(Austauschen(Feldname;"::";¶);2)
```

Fazit

- SQL ist kein Voodoo und kein Allheilmittel
- Es gibt keine Aufgabenstellung die einzig durch den Einsatz von SQL gelöst werden kann
- SQL sollte überlegt und dosiert verwendet werden
- Durch Verwendung von MBS-SQL (bzw. deren Update, Insert, Delete und InsertOrUpdateRecord Funktionen) kann erheblicher Programmieraufwand eingespart werden
- Durch Einsatz von SQL kann der Beziehungsgraph verschlankt und somit die Gesamtperformance der Anwendung gesteigert werden

Q & R

Vielen Dank für Ihr Interesse!

Vielen Dank unseren Sponsoren und Konferenz-Partnern