Performance Analyse der Optimierung von Datenbankabfragen in der HANA Calculation Engine

Projektarbeit 2

im Rahmen der Prüfung zum

Bachelor of Science (B.Sc.)

des Studienganges Wirtschaftsinformatik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Jared Heinrich

Abgabedatum:	26. August 2024
Bearbeitungszeitraum:	06.05.2024 - 25.08.2024
Matrikelnummer, Kurs:	5101479, WWI22SEA
Ausbildungsfirma:	SAP SE Dietmar-Hopp-Allee 16 69190 Walldorf, Deutschland
Unternehmensbetreuer:	Rainer Agelek
Wissenschaftlicher Betreuer:	Prof. Dr. Hans-Henning Pagnia

Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema:

> *Performance Analyse der Optimierung von Datenbankabfragen in der HANA Calculation Engine*

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Mannheim, den 8. Juli 2024

Heinrich, Jared

Abstract

- Deutsch -

Das ist der Abstract.

Inhaltsverzeichnis

Abkürzungsverzeichnis

CE Calculation Engine

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1. Einleitung

2. Grundlagen

2.1. Calculation Engine

2.2. Modelle und Datenbankabfragen

In der Calculation Engine (CE) werden Datenbankabfragen durch Modelle dargestellt. Für jede Abfrage wird zuerst das dazugehörige Modell instanziiert. Anschließend wird dieses Modell optimiert, um die Ausführungsdauer der Abfrage zu minimieren. Diese optimierte Abfrage, wird dann auf der Datenbank ausgeführt. Allgemein kann man ein Modell als azyklischen gerichteten Graphen nach Definition 1 und Definition 3 beschreiben. Des Weiteren ist zu beachten, dass dieser Graph nur eine Quelle nach Definition 5 hat, welche auch als Abfrageknoten bezeichnet wird. Diese Definitionen reichen jedoch nicht aus, da zwischen verschiedenen Arten von Knoten unterschieden wird, welche jeweils verschiedene Informationen beinhalten. Die wichtigsten dieser Knoten Typen sind Projection, Aggregation, Join, Union und Table.

Allgemein werden die Knoten in der CE in zwei Gruppen unterschieden. Die Datenquellenknoten, zu welchen z. B. der Table Knotentyp gehört. Diese werden im Folgenden auch Tabellenknoten genannt, da dieser der einzig relevante der Gruppe ist. Die Operationsknoten, zu welchen die anderen genannten Knotentypen gehören. Ein Table-Knoten spiegelt als Datenquellenknoten eine Tabelle einer Datenbank wider. Ein Operationsknoten bildet hingegen eine Datenbankoperation ab.

Zwar könnte man das Modell wie in Definition 1 beschreiben als eine Menge von Knoten und Kanten darstellen, tatsächlich wird es jedoch nur als eine Menge von Knoten gespeichert, wobei in jedem Knoten festgehalten ist, wer seine Kind- und Elternknoten sind. Die Kindknoten werden dabei als Eingangsknoten und die Elternknoten als Ausgangsknoten bezeichnet. Jeder Knoten kann beliebig viele Ausgangsknoten haben, wobei es wie bereits beschreiben nur einen Knoten mit 0 Ausgangsknoten gibt. Die Anzahl der Eingangsknoten unterscheidet sich dabei je nach Knotentyp. Projection- und Aggregation-Knoten haben

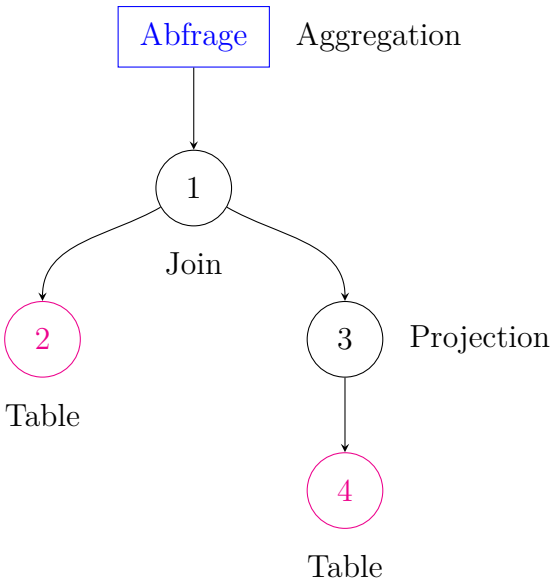

Abbildung 2.1.: Beispiel Modell

genau einen, ein Join-Knoten hat genau zwei, ein Union-Knoten hat zwei oder mehr und ein Table-Knoten hat keinen Eingangsknoten.

Abbildung 2.1 zeigt ein einfaches Beispielmodell, welches zur Veranschaulichung dient. Das Modell besteht aus fünf Knoten, dem Abfrageknoten, zwei Operationsknoten und zwei Tabellenknoten. In dem Knoten steht der Name des Knotens und neben ihm steht der Knotentyp. Bis auf den Abfrageknoten haben in diesem Beispiel alle Knoten eine Zahl als Namen. Der Abfrageknoten sowie alle Tabellenknoten sind zusätzlich farblich markiert. Knoten 1 ist ein Join-Knoten. Er hat zwei Eingangsknoten, den Table-Knoten 2 und den Projection-Knoten 3. Der einzige Ausgangsknoten ist in diesem Fall der Abfrageknoten, dieser ist hier vom Typ Aggregation. Der Abfrageknoten könnte jedoch ein beliebiger Operationsknoten sein.

Jeder Knoten beinhaltet eine Liste an Sichtattributen, diese legt fest, welche Sichtattribute seiner Eingangsknoten dieser Knoten an seine Ausgangsknoten weitergibt. Bei einem Tabellenknoten sind die Sichtattribute gleichbedeutend mit den Spalten der Tabelle. Die Sichtattribute des Abfrageknotens sind die Attribute, welche im Ergebnis der Abfrage angezeigt werden.

2.3. Verschiedene Arten von Performance Analyse

Benchmarking

Profiling

2.4. Notwendige Grundlagen aus der Graphentheorie

Definition 1 gibt eine Definition für einen gerichteten Graphen (Vgl. Knebl 2021, S.220).

> **Definition 1.** *Ein gerichteter Graph ist ein Tupel $G = (V, E)$. V heißt Menge der Knoten. E heißt Menge der gerichteten Kanten. Es gilt $V \neq \varnothing$ und $E \subset V \times V \setminus \{(v, v) | v \in V\}$. Zwischen zwei Knoten $u, v \in V$ gibt es eine Kante mit dem Anfangspunkt u und dem Endpunkt v, wenn $(u, v) \in E$.*

Definition 2 gibt eine Definition für einen Pfad in einem Graphen (Vgl. Knebl 2021, S.221f).

> **Definition 2.** *Ein Pfad P in G ist eine Folge von Knoten $v_0, \ldots, v_n$. Dabei muss $\forall i \in \{0, \ldots, n-1\} : (v_i, v_{i+1}) \in E$ gelten. v_0 heißt Anfangspunkt von P, v_n heißt Endpunkt von P und n heißt Länge von P.*

Definition 3 gibt eine Definition für Zyklen in einem gerichteten Graphen und definiert den Begriff des azyklischen Graphen (Vgl. Knebl 2021, S.222).

> **Definition 3.** *Ein Pfad heißt geschlossen, wenn $v_0 = v_n$. Ein geschlossener Pfad heißt einfach, wenn $\forall i, j \in \{0, \ldots, n-1\}, i \neq j : v_i \neq v_j$ gilt. In einem gerichteten Graphen heißt ein einfach geschlossener Pfad mit $n \geq 2$ auch Zyklus. Ein Graph heißt azyklisch, wenn er keine Zyklen besitzt.*

Definition 4 definiert den Begriff des gewichteten Graphen (Vgl. Knebl 2021, S.253).

Definition 4. *G heißt gewichtet, wenn es eine Abbildung $g : E \to \mathbb{R}$ gibt, welche jeder Kante ein Gewicht zuordnet. Für $e \in E$ heißt $g(e)$ Gewicht von e.*

Definition 5 definiert die Begriffe Quelle und Senke (Vgl. Knebl 2021, S.306).

Definition 5. *Ein Knoten $q \in V$ heißt Quelle, wenn $\neg\exists p \in V : (p, q) \in E$, q also nach Definition 6 keine Elternknoten hat. $s \in V$ heißt Senke, wenn $\neg\exists p \in V : (s, p) \in E$, s also nach Definition 6 keine Kindknoten hat.*

Definition 6 definiert die Begriffe Kindknoten, Elternknoten und Subknoten.

Definition 6. *Für zwei Knoten c und p gilt, c ist Kindknoten von p, wenn $(p, c) \in E$. Umgekehrt gilt, wenn $(p, c) \in E$, p ist Elternknoten von c. c ist Subknoten von p, wenn $\exists P : (v_0 = p) \wedge (v_n = c)$, es also einen Pfad von p nach c gibt.*

2.5. Statistische Grundlagen

In Definition 7 wird der Begriff metrisches Merkmal definiert (Vgl. Stocker und Steinke 2017, S.24).

Definition 7. *Bei einem metrisch skaliertem Merkmale werden einzelne Ausprägungen anhand einer Zahlenskala bewertet, häufig in Verbindung mit einer Maßeinheit. Sie lassen sich anhand der Größe ordnen und vergleichen. Außerdem kann man die Abstände zwischen verschiedenen Ausprägungen messen und interpretieren.*

Definition 8 definiert die, im Kontext dieser Arbeit gleichbedeutenden, Begriffe Durchschnitt, Mittelwert und Arithmetisches Mittel (Vgl. Stocker und Steinke 2017, S.52f).

Definition 8. *Das arithmetische Mittel von n metrisch skalierten Beobachtungswerten $x_1, \ldots, x_n$, genannt $\bar{x}$.*

Es gilt: $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$

Definition 9 beschreibt den Begriff der Varianz (Vgl. Fisher 1919, S.399).

Definition 9. *Die empirische Varianz von n metrisch skalierten Beobachtungswerten $x_1, \ldots, x_n$, genannt $\tilde{s}^2$, ist der Durchschnitt der quadratischen Abweichung von x_i von $\bar{x}$.*

Es gilt: $\tilde{s}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$

Definition 10 beschreibt den Begriff der Standardabweichung (Vgl. Pearson 1894, S.80).

Definition 10. *Die empirische Standardabweichung von n metrisch skalierten Beobachtungswerten $x_1, \ldots, x_n$, genannt $\tilde{s}$, ist die Quadratwurzel der Varianz.*

Es gilt: $\tilde{s} = \sqrt{\tilde{s}^2} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2}$

3. Aktueller Stand der Performance Analyse in der Calculation Engine

In der CE werden bereits verschiedene Wege genutzt, um die Performance und das Verhalten der HANA Datenbank zu analysieren.

3.1. Google Benchmark

Google Benchmark ist ein Open Source Benchmarking Tool von Google, welches es einem ermöglicht einzelne Funktionen in C++ zu benchmarken. Dazu kann man ähnlich zu den meisten Test Frameworks drei verschiedene Codeabschnitte definieren.

Der Hauptabschnitt definiert was genau im Benchmark untersucht werden soll. Diese legt die für die Messung relevante Logik fest. Der Setup Abschnitt wird einmal vor jeder Ausführung des Benchmarks aufgerufen. In dieser werden die Voraussetzungen für die Ausführung des Hauptabschnitts geschaffen. Man könnte sie beispielsweise nutzen, um Testdaten für den Benchmark zu generieren oder zu laden, da er nicht bei den Messungen beachtet wird. Der Teardown Abschnitt wird nach jeder Ausführung des Benchmarks aufgerufen. Dieser beeinflusst, wie der Setup Abschnitt, die Messung nicht.

Des Weiteren bietet Google Benchmark die Option, einen Benchmark mehrmals mit unterschiedlichen Parametern durchzuführen. Um beispielsweise die Auswirkung der Größe des Testdatensatzes auf das Ergebnis zu beobachten.

Abbildung 3.1 zeigt eine Beispielhafte Google Benchmark Ausgabe. Benchmark ist dabei der Name des Benchmarks und hinter dem Schrägstrich die Größe des Inputs. Time ist die durchschnittliche Dauer einer Ausführung des Hauptabschnitts über alle Iterationen hinweg. CPU ist die durchschnittliche CPU Zeit einer Ausführung des Hauptabschnitts über alle Iterationen hinweg. Iterations ist die Anzahl der durchgeführten Wiederholungen des Hauptabschnitts. Legt man keine Anzahl fest, wird die Anzahl der Wiederholungen

anhand der durchschnittlichen Dauer einer Iteration und der Varianz der Dauer über alle Iterationen hinweg. (Vgl. Google 2024)

```
-------------------------------------------------------------------
Benchmark                             Time           CPU Iterations
-------------------------------------------------------------------
BM_RemoveDetachedNodes/2              4 us         4 us       189823
BM_RemoveDetachedNodes/8              9 us         9 us        78165
BM_RemoveDetachedNodes/64            58 us        58 us        12106
BM_RemoveDetachedNodes/512          469 us       468 us         1515
BM_RemoveDetachedNodes/4096        4237 us      4234 us          166
BM_RemoveDetachedNodes/8192       10026 us     10019 us           60
```

Abbildung 3.1.: Google Benchmark Ausgabe

Google Benchmark wird in der CE meistens genutzt, um das Verhalten der Laufzeit bestimmter Funktionen in künstlich generierten Testfällen zu vergleichen. Hierbei wird keine HANA Instanz benötigt, da keine Operationen auf einer Datenbank ausgeführt werden, sondern nur einzelne Funktionen aufgerufen werden. Es werden folglich auch keine Testdatensätze für die Datenbank benötigt, sondern nur Daten, auf welchen man die zu analysierende Funktion aufrufen kann.

Im Vergleich zu anderen Methoden der Performance Messung, die Operationen auf einer HANA-Instanz ausführen, ist ein Google Benchmark relativ einfach. Dies liegt daran, dass sie weniger Voraussetzungen erfordern und sich lediglich auf einen kleinen Teil der Logik konzentrieren.

3.2. HANA Profiler

Eine weitere Methode die Performance und das Verhalten von Software zu analysieren ist, das bereits in Abschnitt 2.3 beschriebene, Profiling.

In der CE wird hauptsächlich der „BOOSS-Profiler“, ein in HANA integrierter Profiler, verwendet. Auf diesen kann entweder manuell oder im Code zugegriffen werden. Dabei sind die wichtigsten Befehle `profiler clear` um die aktuellen Profilinginformationen zurückzusetzen, `profiler start` um den Profiler zu starten, `profiler stop` um den

Profiler zu stoppen und `profiler print` um die gesammelten Profilinginformationen auszugeben.

Der Profiler kann in diesem Kontext auf zwei verschiedene Arten genutzt werden. Entweder in dem zur Laufzeit des Profilers Anfragen an eine bestehende HANA-Instanz gestellt werden. Oder der Profiler wird innerhalb eines Tests oder Benchmarks aufgerufen und es werden die dort aufgerufen Funktionen gemessen.

Die Ausgabe erzeugt dabei ist dabei zwei gewichtete gerichtete azyklische Graphen, nach Abschnitt 2.4, von denen einer die Verteilung der CPU-Zeit und der andere die Verteilung der Wartezeit, der aufgerufenen Funktionen beinhaltet. Im Folgenden werden die beiden Graphen CPU-Graph und Warte-Graph genannt. Abbildung 3.2 stellt eine mögliche Ausgabe des HANA Profilers dar. Die folgende Beschreibung gilt für den CPU- als auch den Warte-Graphen. Jeder Knoten des Graphen spiegelt eine zur Laufzeit des Profilers aufgerufene Funktion wider. Jeder Knoten beinhaltet drei Informationen. Den Namen der aufgerufenen Funktion, sowie den Wert I und den Wert E. I ist der Anteil der Gesamtzeit, welcher von diesem Knoten und all seinen Subknoten benötigt wurde. E ist der Anteil der Gesamtzeit, welcher von diesem Knoten benötigt wurde. Folglich gilt für alle Knoten $I \geq E$. Die Kindknoten eines Knoten K sind die Funktionen, welche von K aufgerufen wurden.

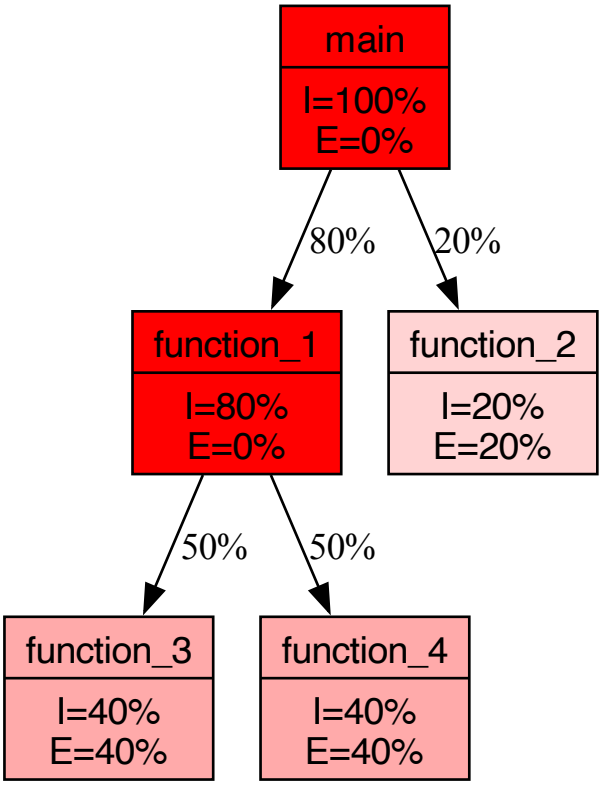

Abbildung 3.2.: Beispielausgabe des HANA Profilers

4. Benchmarking-Konzept

4.1. Messung

Um Messwerte sinnvoll miteinander vergleichen zu können, dürfen diese sich bis auf einen Parameter nicht unterscheiden. Bei der Analyse, wie sich verschiedene Modelle auf die Laufzeit der Optimierung von diesen auswirken, sind die Größe und der Aufbau des Modells als Parameter denkbar. Der Aufbau, im folgenden auch Art des Modells genannt, wird durch eine Funktion beschrieben, welche für einen Parameter n ein Modell der Größe n erzeugt. n ist dabei nicht zwingend die Anzahl der Knoten im Modell, sondern könnte auch festlegen, wie oft ein bestimmtes Muster in dem Modell wiederholt wurde.

Ist die Größe in einer Messreihe der veränderliche Parameter, muss der Aufbau des Modells unveränderlich sein.

4.2. Testdaten

5. Umsetzung

5.1. Implementation der neuen Methode

6. Verifikation

7. Fazit und Ausblick

A. Anhang

Hier ist der Anhang

Literaturverzeichnis

Knebl, H. (2021). *Algorithmen und Datenstrukturen: Grundlagen und probabilistische Methoden für den Entwurf und die Analyse / Helmut Knebl.* Springer eBook Collection. Springer Vieweg.

Stocker, T. C./ I. Steinke (2017). *Grundlagen und Methodik.* Berlin, Boston: De Gruyter Oldenbourg.

Fisher, R. A. (1919). „XV.—The Correlation between Relatives on the Supposition of Mendelian Inheritance.“ In: *Transactions of the Royal Society of Edinburgh* 52.2, S. 399–433.

Pearson, K. (1894). „Contributions to the Mathematical Theory of Evolution“. In: *Philosophical Transactions of the Royal Society of London. A* 185, S. 71–110. URL: http://www.jstor.org/stable/90667 (Einsichtnahme: 24. 06. 2024).

Google (2024). *benchmark A microbenchmark support library.* URL: https://google.github.io/benchmark/ (Einsichtnahme: 10. 06. 2024).