|  |
| --- |
| Wintersemester 2021/21 |
| Vergleich von MonetDB vs. InnoDB vs. ColumnStore |
|  |
| Semesteraufgabe |
| vorgelegt von  Benedikt Scheffbuch |

Inhaltsverzeichnis

[1 Einleitung 7](#_Toc282980784)

[1.1 Motivation 6](#_Toc282980785)

[1.2 Projektbeschreibung 6](#_Toc282980786)

[1.3 Aufbau der Thesis 6](#_Toc282980787)

[2 Kapitel 2 6](#_Toc282980788)

[2.1 Überschrift Ebene 2 6](#_Toc282980789)

[2.1.1 Überschrift Ebene 3 6](#_Toc282980790)

[2.1.2 Überschrift Ebene 3 6](#_Toc282980791)

[2.1.3 Überschrift Ebene 3 6](#_Toc282980792)

[2.2 Überschrift Ebene 2 6](#_Toc282980793)

[3 Kapitel 3 6](#_Toc282980794)

[4 Kapitel 4 6](#_Toc282980795)

[5 Kapitel 5 6](#_Toc282980796)

[6 Fazit und Aussichten 6](#_Toc282980797)

[Literaturverzeichnis 6](#_Toc282980798)

[Glossar 6](#_Toc282980799)

[Anhang 6](#_Toc282980800)

Abbildungsverzeichnis

[Abbildung 1 Logo der Hochschule Furtwangen 6](#_Toc282980473)

# Einleitung

Apache Kudu ist ein spaltenorientierter Open-Source Datenspeicher des Apache Hadoop-Ökosystems. Er ist mit den meisten Datenverarbeitungs-Frameworks in der Hadoop-Umgebung kompatibel. Er bietet Vollständigkeit für die Speicherschicht von Hadoop, um schnelle Analysen auf schnellen Daten zu ermöglichen. Die erste Version Apache Kudu 1.0 wurde am 19. September 2016 veröffentlicht.

Kudu wurde für OLAP-Arbeitslasten entworfen und optimiert. Wie HBase ist es ein Echtzeitspeicher, der schlüsselindizierte Datensatzsuche und -mutation unterstützt. Kudu unterscheidet sich von HBase, da das Datenmodell von Kudu ein traditionelleres relationales Modell ist, während HBase schemenlos ist. Die "On-Disk-Darstellung" von Kudu ist wirklich säulenförmig und folgt einem völlig anderen Speicherdesign.[[1]](#footnote-1)

## OLTP vs OLAP

**OLTP** ist durch große Datenbanktransaktionen gekennzeichnet, die Einfügungen, Aktualisierungen oder Löschungen vornehmen. Datenbanken vom Typ OLTP sind auf die schnelle Verarbeitung von Abfragen und die Wahrung der Datenintegrität spezialisiert, während auf sie in verschiedenen Umgebungen zugegriffen wird. Ihre Effektivität wird an der Anzahl der Transaktionen pro Sekunde (tps) gemessen. Es ist üblich, dass die Tabellen mit Eltern-Kind-Beziehungen (nach der Implementierung der Normalisierungsform) redundante Daten in einer Tabelle reduzieren.

Datensätze in einer Tabelle werden üblicherweise zeilenorientiert sequenziell verarbeitet und gespeichert und mit eindeutigen Schlüsseln indiziert, um das Abrufen oder Schreiben von Daten zu optimieren. Dies ist auch bei MySQL üblich, insbesondere wenn es um große Einfügungen oder hohe gleichzeitige Schreibvorgänge oder Masseneinfügungen geht. Die meisten der von MariaDB unterstützten Speicher-Engines sind für OLTP-Anwendungen geeignet - **InnoDB** (die Standardspeicher-Engine seit 10.2), XtraDB, TokuDB, MyRocks oder MyISAM/Aria.

Anwendungen wie CMS, FinTech, Web-Apps haben oft mit großen Schreib- und Lesevorgängen zu tun und erfordern oft einen hohen Durchsatz. Damit diese Anwendungen funktionieren, ist oft ein tiefes Fachwissen über Hochverfügbarkeit, Redundanz, Ausfallsicherheit und Wiederherstellung erforderlich.

**OLAP** befasst sich mit den gleichen Herausforderungen wie OLTP, verwendet jedoch einen anderen Ansatz (insbesondere bei der Datenabfrage). OLAP befasst sich mit größeren Datenmengen und wird häufig für Data Warehousing und für Business Intelligence-Anwendungen verwendet. Es wird häufig für Business Performance Management, Planung, Budgetierung, Forecasting, Finanzberichte, Analysen, Simulationsmodelle, Knowledge Discovery und Data Warehouse Reporting verwendet.

Daten, die in OLAP gespeichert werden, sind in der Regel nicht so kritisch wie die in OLTP gespeicherten. Das liegt daran, dass die meisten Daten simuliert werden können, die aus dem OLTP kommen und dann in Ihre OLAP-Datenbank eingespeist werden können. Diese Daten werden typischerweise für das Bulk-Loading verwendet, das oft für Geschäftsanalysen benötigt wird, die schließlich in visuelle Diagramme gerendert werden. OLAP führt auch multidimensionale Analysen von Geschäftsdaten durch und liefert Ergebnisse, die für komplexe Berechnungen, Trendanalysen oder anspruchsvolle Datenmodellierung verwendet werden können.

OLAP speichert Daten in der Regel persistent in einem spaltenförmigen Format. In MariaDB **ColumnStore** hingegen werden die Datensätze auf Basis ihrer Spalten aufgeschlüsselt und separat in einer Datei gespeichert. Auf diese Weise ist der Datenabruf sehr effizient, da nur die relevante Spalte durchsucht wird, auf die sich Ihre SELECT-Anweisung in der Abfrage bezieht.

Man kann sich das so vorstellen: Die OLTP-Verarbeitung kümmert sich um die täglichen und wichtigen Datentransaktionen, mit denen Ihre Geschäftsanwendung läuft, während OLAP hilft, das Produkt zu verwalten, vorherzusagen, zu analysieren und besser zu vermarkten - die Bausteine, die eine Geschäftsanwendung ausmachen.

## Spaltenorientierter Datenspeicher

Ein spaltenorientiertes DBMS (oder spaltenartiges Datenbankmanagementsystem) ist ein Datenbankmanagementsystem (DBMS), das Datentabellen spaltenweise und nicht zeilenweise speichert. Die praktische Verwendung eines Spaltenspeichers gegenüber einem Zeilenspeicher unterscheidet sich in der relationalen DBMS-Welt nur wenig. Sowohl spalten- als auch zeilenbasierte Datenbanken können herkömmliche Datenbankabfragesprachen wie SQL verwenden, um Daten zu laden und Abfragen durchzuführen. Sowohl Zeilen- als auch Spaltendatenbanken können zum Rückgrat in einem System werden, um Daten für gängige Extraktions-, Transformations-, Lade- (ETL) und Datenvisualisierungstools bereitzustellen. Durch die Speicherung von Daten in Spalten statt in Zeilen kann die Datenbank jedoch präziser auf die Daten zugreifen, die sie zur Beantwortung einer Abfrage benötigt, anstatt unerwünschte Daten in Zeilen zu scannen und zu verwerfen.

Kurzum, spaltenorientierte Datenspeicher sind effizienter, wenn nicht alle Eigenschaften einer Instanz gebraucht werden, während zeilenbasierte Datenspeicher effizienter sind, wenn (fast) alle Eigenschaften einer Instanz gebraucht werden.

## Arten von Datenbanken

# Installation

## Installation einer virtuellen Maschine (VM) und Betriebssystem (OS)

Für die Installation der kompletten virtuellen Maschine wird benötigt:

1. VMware Workstation Player ([Download](https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html))
2. CentOS ISO Datei ([Download](https://www.centos.org/download/))
   1. Bei Installation auf korrekte Sprache, Zeitzone und Tastaturlayout achten

* Login als root mit Passwort root für notwendige Berechtigungen
* Die Optimierungen für MariaDB ColumnStore vornehmen ([hier](https://mariadb.com/docs/deploy/community-single-columnstore-cs105-centos8/#install-community-single-columnstore-cs105-centos8-optimize-linux-kernel-parameters))
* CSV-Dateien via [Bulk Import](https://mariadb.com/docs/deploy/community-single-columnstore-cs105-centos8/#bulk-import-data) importieren
  + Schema erstellen (siehe create\_table.sql)
  + cpimport per Kommandozeile im user folder
  + sudo cpimport -s ',' flights flights flights\_01.csv

## MonetDB

1. Den Anweisungen für die Installation von MonetDB unter CentOS [hier](https://www.monetdb.org/downloads/epel/) folgen, konkret die Schritte:
   1. yum install <https://dev.monetdb.org/downloads/epel/MonetDB-release-epel.noarch.rpm>
   2. yum install MonetDB-SQL-server5 MonetDB-client
2. Anschließend einen Blick auf das [offizielle Tutorial](https://www.monetdb.org/Documentation/UserGuide/Tutorial) werfen, um das DBMS in Betrieb zu nehmen
3. Erstellen von Datenbank und Tabelle analog zum Tutorial
4. Importieren der CSV-Dateien über [CSV Bulk Loads](https://www.monetdb.org/Documentation/ServerAdministration/LoadingBulkData/CSVBulkLoads)

## Inbetriebnahme CentOS

## Installation DBMS

## Vorbereitung Benchmarking

## Resultate

# Bedienung von Kudu

Literaturverzeichnis

Wikipedia (2020): Apache Kudu. Online verfügbar unter https://en.wikipedia.org/wiki/Apache\_Kudu, zuletzt aktualisiert am 10.08.2020.

1. Wikipedia 2020 [↑](#footnote-ref-1)