

T.C
MARMARA ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
ENFORMATİK(ALM) ANABİLİM DALI
İŞLETME ENFORMATİĞİ(ALM) BİLİM DALI

**DIE IMPLEMENTIERUNG DER HIVE-ARCHITEKTUR IN BIG DATA IN
EINEM VIRTUELLEN UNTERNEHMEN**

Yüksek Lisans Tezi

Mehmet IŞIK

Istanbul, 2019

T.C
MARMARA ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
ENFORMATİK(ALM) ANABİLİM DALI
İŞLETME ENFORMATİĞİ(ALM) BİLİM DALI

**DIE IMPLEMENTIERUNG DER HIVE-ARCHITEKTUR IN BIG DATA IN
EINEM VIRTUELLEN UNTERNEHMEN**

Yüksek Lisans Tezi

Mehmet IŞIK

Danışman: DOÇ. DR. DENİZ HERAND

Istanbul, 2019



T.C.
MARMARA ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

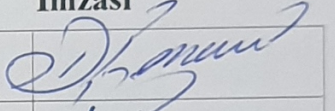
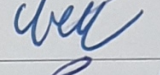
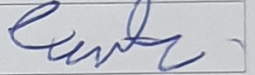
TEZ ONAY BELGESİ

ENFORMATİK (ALMANCA) Anabilim Dalı İŞLETME ENFORMATİĞİ
(ALMANCA) Bilim Dalı TEZLİ YÜKSEK LİSANS öğrencisi MEHMET IŞIK'ın DIE
IMPLEMENTIERUNG DER HIVE-ARCHITEKTUR IN BIG DATA IN EINEM
VIRTUELLEN UNTERNEHMEN adlı tez çalışması, Enstitümüz Yönetim Kurulunun
14.02.2019 tarih ve 2019-5/26 sayılı kararıyla oluşturulan jüri tarafından oy birliği / oy çokluğu
ile Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi 22/02/2019

Öğretim Üyesi Adı Soyadı

İmzası

1.	Tez Danışmanı	Doç. Dr. DENİZ HERAND	
2.	Jüri Üyesi	Prof. Dr. YÜCEL YILMAZ	
3.	Jüri Üyesi	Dr.Öğr.Üyesi ÖNDER TONBUŞ	

VORWORT

Die vorliegende Masterarbeit entstand im Rahmen meines Informatik-Studiums an der Marmara-Universität zu Istanbul.

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich bei Erstellung dieser Arbeit unterstützt haben. Besonderer Dank gilt an meine Betreuer in Herr Dozent Deniz HERAND. Ebenfalls besonderer Dank an meine Freunde, Kommilitonen und meine Familie die mich während dieser Arbeit unterstützt haben.

Istanbul, 2019

Mehmet IŞIK

Inhaltsverzeichnis

VORWORT	iii
ÖZET	vii
ZUSAMMENFASSUNG	viii
ABBILDUNGSVERZEICHNIS	ix
TABELLENVERZEICHNIS	xii
1. EINLEITUNG	1
2. DIE VERARBEITUNG IN HDFS.....	2
2.1 Die Erstellung eines neuen Ordners im HDFS:	2
2.2 Die Kopierung einer Datei vom lokalen Dateisystem auf das HDFS.....	3
2.3 Die Auflistung von allen Dateien und Ordner im HDFS-Ordner	4
2.3.1 Rekursiver Löschvorgang aus dem Verzeichnis.....	7
3. MAP-REDUCE.....	9
3.1 Zusammenfassung: Map Reduce	9
3.2 Die Einrichtung der Entwicklungsumgebung	11
3.2.1 Installation von Eclipse als IDE	11
3.2.2 Installation von Glassfish als Anwendungsserver	12
3.2.3 Erstellen eines Testprojekts	15
3.2.4 Die Installation von Maven-Plug	16
3.2.5 Die Installation von WinSCP	17
3.3 Map-Reduce Anwendung.....	17
3.3.1 Die Erstellung eines Maven-Projekts	20
3.3.2 Erstellen von Run-Einstellungen zum Generieren der JAR-Datei.....	23
3.3.3 Map Reduce Programmierung	24
3.3.4 Exportieren als JAR über Maven	35
3.3.5 Ausführen eines Jobs über eine Befehlszeile	37
4. ARCHITEKTUR VON HIVE	41
4.1 Die Architektur von Apache Hive	42
5. INSTALLATION VON HIVE	44

5.1 Die Einstellung von Apache Hive	44
6. HIVEQL ALS ABFRAGESPRACHE	46
6.1 Die Datenbank-Operationen	46
6.1.1 Anlegen von Datenbanken	46
6.1.2 Importieren von Daten	47
6.1.3 Das SELECT-Statement	54
6.1.4 Beschränken von SELECT über DISTINCT	57
6.1.5 SELECT auf partitionierte Tabellen	58
6.2 Die Datentypen und Views	60
6.2.1 Primitive Datentypen	60
6.2.2 Komplexe Datentypen	61
6.2.3 Erstellen von Views	61
6.2.4 Löschen einer View	62
6.3 Die Tabelle-Operationen	62
6.3.1 Anlegen von Tabellen	62
6.3.2 Partitionierung von Tabellen	64
6.3.3 Löschen, Leeren von Tabellen und count Befehle	64
6.3.4 Verändern von Tabellen	65
6.4. Die Gruppierung-Operationen	66
6.4.1 Gruppieren von Daten mittels GROUP BY	66
6.4.2 Subqueries verschachtelte Abfragen	68
6.4.3 Ergebnismengen vereinigen mit UNION ALL	68
6.4.4 HAVING	69
6.4.5 Tabellen zusammenführen mit JOINS	70
6.5 Die Funktionen	70
6.5.1 Mathematische Funktionen	71
6.5.2 String-Funktionen	73
6.5.3 Aggregatfunktionen	73
7. VIRTUELLE ANWENDUNG MIT HIVE	75

7.1 Erstellen eines dynamischen Webprojekts	76
7.2 Erstellen von JSP-Dateien für Hive-Yonetici	86
8. Schlussfolgerungen	101
LITERATURVERZEICHNIS.....	102



ÖZET

Apache HIVE SQL-Stil tabanlı sorgular geliřtirmemizi saęlayan Hadoop tabanlı olan veri ambarı (Datawarehouse) kütüphanesidir.

Büyük verilerde hızlı bir řekilde analiz yapmak için, Apache Hive tabanlı SQL sorguları geliřtirebiliriz.

Bu çalışmada Apache HIVE mimarisi Apache HIVE tabanlı SQL sorguları oluşturulacak ve sanal bir firma üzerinde bir uygulama yapılacaktır.

Anahtar Kelimeler: Hadoop, Apache Hive, HQL, büyük veri



ZUSAMMENFASSUNG

Apache HIVE SQL ist eine Hadoop-basierte Datawarehouse-Bibliothek, die SQL-Stil Abfrageentwicklung bietet.

Um schnell große Datenmengen analysieren zu können, können wir Apache HIVE basierte SQL-Abfragen entwickeln.

In dieser Studie wurden Apache HIVE-Architecture, Apache HIVE basierte SQL Abfragen und eine Anwendung in einem virtuellen Unternehmen erstellt.

Schlüsselwörter: Hadoop, Apache Hive, HQL, Big Data



ABBILDUNGSVERZEICHNIS

Abbildung 1: Auflistung von Dateien in HDFS.....	4
Abbildung 2: Auflistung aller Dateien in HDFS	5
Abbildung 3: Die Anzeige des letzten Bytes der Datei	6
Abbildung 4: Löschung von Dateien in HDFS.....	6
Abbildung 5: Ein Beispiel von Rekursion: Rückkopplung in einer Screenshot-Software	7
Abbildung 6: Der Datenfluss bei der MapReduce-Berechnung	11
Abbildung 7: Einstellen als Standard des JDK.....	12
Abbildung 8: GlassFish suchen	13
Abbildung 9: Servereinstellung	14
Abbildung 10: Server-Teil	14
Abbildung 11: Server läuft.....	15
Abbildung 12: Dynamisches Webprojekt.....	15
Abbildung 13: Testprojekt	16
Abbildung 14: Webserver-Ergebnis	16
Abbildung 15: Die Datei TestData.java.....	18
Abbildung 16: Die Erstellung eines Maven-Projekts	21
Abbildung 17: Die Einstellung des Mavens	22
Abbildung 18: Maven ID und Versionsauswahl.....	23
Abbildung 19: Maven Projekt.....	23
Abbildung 20: Die Einstellungen.....	24
Abbildung 21: Die Erstellung eines neuen Pakets	25
Abbildung 22: Die Erstellung einer neuen Klasse	26
Abbildung 23: Projektansicht	26
Abbildung 24: Programmeinstellungen	36
Abbildung 25: Die Erstellung der Datei ".jar"	37
Abbildung 26: Einloggen mit WinSCP.....	38
Abbildung 27: Kopieren von Dateien mit WinSCP	39
Abbildung 28: Anzeige mit Weboberfläche	40
Abbildung 29: Das Ergebnis wird im Terminal angezeigt	40
Abbildung 30: Architektur von Apache Hive im Zusammenspiel mit Hadoop.....	41
Abbildung 31: Architektur von Apache Hive	42

Abbildung 32: Ändern des Dateinamens	45
Abbildung 33: Erstellen einer Datenbank.....	46
Abbildung 34: Verwendung des USE-Befehls	46
Abbildung 35: Anzeige der Datenbank.....	47
Abbildung 36: Übertragen von Daten mit Hive.....	53
Abbildung 37: Erstellen der Tabelle "yoneticiler"	53
Abbildung 38: Übertragen Sie in die Tabelle "yoneticiler"	53
Abbildung 39: Bestätigung der Anzahl der Daten	54
Abbildung 40: SELECT-Befehl.....	54
Abbildung 41: Verwendung des WHERE-Befehls.....	55
Abbildung 42: Verwendung des AND-Befehls	56
Abbildung 43: Logischer Operatoren	56
Abbildung 44: Relationalen Operatoren	57
Abbildung 45: Verwendung des LIKE-Befehls.....	57
Abbildung 46: Verwendung des DISTINCT-Befehls.....	58
Abbildung 47: Verwendung des ORDER BY-Befehls.....	58
Abbildung 48: Verwendung des ASC-Befehls	59
Abbildung 49: Verwendung des VIEW-Befehls	61
Abbildung 50: Anzeigen mit View erstellten Tabelle	62
Abbildung 51: Löschen eines View	62
Abbildung 52: Erstellen einer Tabelle mit Hive	63
Abbildung 53: Auflisten der Tabellen	63
Abbildung 54: Verwendung des LIKE-Befehl	64
Abbildung 55: Verwendung des COUNT-Befehls	65
Abbildung 56: Verwendung des ALTER-Befehls	65
Abbildung 57: Ändern der Tabelleneigenschaften	66
Abbildung 58: Detaillierte Eigenschaften der Tabelle.....	66
Abbildung 59: Verwendung des GROUP BY-Befehls.....	66
Abbildung 60: Nutzungsergebnis des Befehls "GROUP BY".....	67
Abbildung 61: Fehlermeldung mit der GROUP BY-Befehl.....	67
Abbildung 62: Erstellung eines verschachtelte Abfragen.....	68
Abbildung 63: Verwendung des HAVING-Befehls	69
Abbildung 64: Verwendung des JOIN-Befehls	70
Abbildung 65: Verwendung des Befehls ROUND()	71

Abbildung 66: String-Funktionen in Hive	73
Abbildung 67: Aggregatfunktionen	74
Abbildung 68: Die Terminalansicht.....	89
Abbildung 69: Anzeige der Datenbank und Tabellen in Hive Manager.....	90
Abbildung 70: Anzeige von Daten in der Tabelle insan.....	90
Abbildung 71: Anzeige von Daten in der Teil “Abfragen”	100



TABELLENVERZEICHNIS

Tabelle 1: ToolRunner-Klassenargumente	34
Tabelle 2: Mathematische Funktionen	71



1. EINLEITUNG

Das Hadoop Ökosystem entstand als eine kostengünstige Möglichkeit, mit solchen großen Datensätzen zu arbeiten¹.

(Strata Hadoop, 2015) Die Geschichte von Big Data wird erzählt. Diese Webseite wird genutzt, um die Geschichte von Big Data zu beschreiben.

Das Hadoop Distributed File System (HDFS) ist ein verteiltes Dateisystem, das über ein Hadoop Cluster den Hochleistung-Zugriff auf Daten ermöglicht. Wie andere Hadoop-Technologien, ist HDFS heute zu einem wichtigen Werkzeug sowohl für die Verwaltung von Big Data Pools als auch zur Unterstützung von Big-Data Analytics Anwendungen geworden².

Hive stellt einen SQL-Dialekt namens Hive Query Language (abgekürzt HiveQL oder nur HQL) zum Abfragen von Daten bereit, die in einem Hadoop-Cluster gespeichert sind. SQL-Wissen ist weit verbreitet. Es ist ein effektives, einigermaßen intuitives Modell zum Organisieren und Verwenden von Daten. Hive übersetzt die meisten Abfragen in MapReduce-Jobs und nutzt dabei die Skalierbarkeit von Hadoop, während eine vertraute SQL-Abstraktion präsentiert wird.

Hive ist am besten für Data-Warehouse-Anwendungen geeignet, bei denen relativ statische Daten analysiert werden, schnelle Antwortzeiten nicht erforderlich sind und wenn sich die Daten nicht schnell ändern. Hive ist keine vollständige Datenbank. Hive eignet sich am besten für Data-Warehouse-Anwendungen, bei denen ein großer Datensatz für Einsichten, Berichte usw. verwaltet wird.

¹ Rutherglen, J., Wampler, D., Capriolo, W., Programming Hive, USA, 2012.

² <https://www.searchenterprisesoftware.de/definition/Hadoop-Distributed-File-System-HDFS>. 01 April 2015

2. DIE VERARBEITUNG IN HDFS

(Apache Hadoop, 2014) Webseite enthält Beschreibungen der in HDFS verwendeten Befehle. Hier wurden die HDFS-Befehle unter Linux implementiert. (Işık & Herand, 2019) Das ist ein Buch, in dem Big Data-Anwendungen Schritt für Schritt erklärt werden. Hive-Installation und Datenbankoperationen wurden aus diesem Buch genutzt. Dann wurde eine virtuelle Firmenanwendung mit Jsp durchgeführt. (Rouse, 2015) Die HDFS-Definition wird auf dieser Website vorgenommen. Die Dateisystem-Shell (FS) enthält verschiedene Shell-ähnliche Befehle, die direkt mit dem Hadoop Distributed File System (HDFS) und anderen von Hadoop unterstützten Dateisystemen wie Local FS, HFTP FS, S3 FS und anderen interagieren. Die FS-Shell wird aufgerufen von:

- `bin/hadoop fs <args>`

Alle FS-Shellbefehle verwenden Pfad-URIs als Argumente. Das URI-Format ist "schema://autorität/Pfad". Für HDFS ist das Schema HDFS, und für das Local FS ist das Schema Datei. Das Scheme und die Autorität sind optional. Wenn nicht angegeben, wird das in der Konfiguration angegebene Standardschema verwendet. Eine HDFS-Datei oder ein HDFS-Verzeichnis wie /parent/child kann als `hdfs://namenodehost/parent/child` oder einfach als `/parent/child` angegeben werden.

Die meisten Befehle in der FS-Shell verhalten sich wie entsprechende Unix-Befehle. Unterschiede werden bei jedem Befehl beschrieben. Fehlerinformationen werden an `stderr` gesendet und die Ausgabe wird an `stdout` gesendet³.

Das Folgende ist ein Überblick über HDFS-Befehle.

2.1 Die Erstellung eines neuen Ordners im HDFS:

- `hdfs dfs -mkdir [-p] ORDNER`
- `hdfs dfs -mkdir /hdfs/input/`

³ <https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html>. 21 June 2014

Die obigen Codes erstellt einen Ordner unter '/hdfs' im Hauptverzeichnis von HDFS.

Um eine leere Datei in HDFS zu erstellen, wird der folgende Befehl eingegeben.

- `hdfs dfs -touchz Datei` -> Die Verwendung des Befehls
- `hdfs dfs -touchz /hdfs/input/test3.txt`

Hier wird eine leere Textdatei "test3.txt" unter dem Verzeichnis "/hdfs/input/" erstellt.

Die folgenden Befehle werden verwendet, um eine Datei in HDFS zu kopieren:

- `hdfs dfs -cp Quelle` -> Die Verwendung des Befehls
- `hdfs dfs -cp /hdfs/input/test.txt /hdfs/input/test2.txt`

Die zuvor erstellte Datei "test.txt" wird in eine neue Datei im selben Verzeichnis mit dem Namen "test2.txt" kopiert.

Zuerst wird der Ordner erstellt, in den die Datei verschoben wird. Mit dem folgenden Befehl wird ein Verzeichnis mit dem Namen "input2" unter /hdfs erstellt.

- `hdfs dfs -mkdir /hdfs/input2/`

Dann wird die Datei "test.txt" unter dem Ordner "/hdfs/input" in den Ordner "/hdfs/input2" verschoben.

- `hdfs dfs -mv Quelle` -> Die Verwendung des Befehls
- `hdfs dfs -mv /hdfs/input/test.txt /hdfs/input2/test.txt`

2.2 Die Kopierung einer Datei vom lokalen Dateisystem auf das HDFS

Der Benutzer, der Hadoop installiert hat, wird übergeben und ein Ordner mit dem Namen input unter /usr wird erstellt. Zuletzt wird eine Textdatei mit dem Namen "test.txt" unter /usr/input erstellt. Für diese werden die folgenden Befehle auf dem Konsolenbildschirm eingegeben:

- su - hduser
- cd /usr
- sudo mkdir input
- sudo touch input/test.txt

Dann wird die Datei "test.txt" unter input in "/hdfs/input" kopiert.

- hdfs dfs -copyFromLocal QUELLE ZIEL -> Die Verwendung des Befehls
- hdfs dfs -copyFromLocal /usr/input/test.txt /hdfs/input/test.txt

Die folgenden Befehle werden eingegeben, um die Datei "test.txt" aus dem HDFS unter /usr/input in das lokale Dateisystem zu kopieren:

- hdfs dfs -copyToLocal QUELLE ZIEL -> Die Verwendung des Befehls
- hdfs dfs -copyToLocal /hdfs/input/test.txt /usr/input/test.txt

2.3 Die Auflistung von allen Dateien und Ordner im HDFS-Ordner

In HDFS werden die folgenden Befehle eingegeben, um die Dateien im Ordner "input" unter dem Hauptverzeichnis aufzulisten:

- hdfs dfs -ls Ordner -> Die Verwendung des Befehls
- hdfs dfs -ls /hdfs/input

Es wird "/hdfs/input/test.txt" ausgegeben. Das wird wie folgt generiert.

```
hduser@ubuntu:~$ hdfs dfs -ls /hdfs/input
Found 1 items
-rw-r--r-- 1 hduser supergroup 0 2017-08-16 05:37 /hdfs/input/test.txt
```

Abbildung 1: Auflistung von Dateien in HDFS

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

- hdfs dfs -ls /

Wenn es auf diese Weise verwendet wird, werden alle Dateien und Verzeichnisse im Verzeichnis hdfs ausgegeben.

```
hduser@ubuntu:~$ hdfs dfs -ls /  
Found 3 items  
drwxr-xr-x - hduser supergroup 0 2017-08-16 05:35 /hdfs  
drwxr-xr-x - hduser supergroup 0 2017-08-16 04:01 /input  
drwxrwx--- - hduser supergroup 0 2017-08-15 00:37 /tmp
```

Abbildung 2: Auflistung aller Dateien in HDFS

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Bevor dieser Befehl verwendet wird, wird die unter "/user/input" gespeicherte Datei "test.txt" in dem Verzeichnis "/ hdfs / input" kopiert.

- `hdfs dfs -copyFromLocal /usr/input/test.txt /hdfs/input/test.txt`

Jetzt wird der folgende Befehl verwendet, um der Ausgabe das letzte Kilobyte dieser Datei zu geben.

- `hdfs dfs -tail [-f] Datei` -> Die Verwendung des Befehls
- `hdfs dfs -tail -f /hdfs/input/test.txt`

Das letzte Byte des Textes in der Datei "test.txt" wird Ihrer Konsole wie folgt übergeben. Der Vorgang wird mit STRG + C beendet.

```
@ubuntu:~$ hdfs dfs -tail -f /hdfs/input/test.txt
Desteklenmeyen Universe (evrensel) Multiverse
Ubuntu yazılım depoları, dört sınıfa ayrılmaktadır.

Main: Canonical tarafından destekli özgür yazılımlara barındıran depodur. Ana depo niteliğindedir.

Universe: Ubuntu topluluğu tarafından bakılan özgür yazılımlara barındıran depodur.

Restricted: Bazı donanımlar için gerekli kapalı kaynak sürücüleri barındıran paketler bu depoda yer almaktadır.

Multiverse: Telif haklarıyla veya yasal sorunlar nedeniyle kısıtlanmış olan yazılımlar bu depoda yer almaktadır.

Tüm bu depolar Ubuntu'nun resmi yazılım depoları olmakla birlikte "Main ve Restricted" depolarındaki yazılımlar Ubuntu tarafından desteklenirken "Universe ve Multiverse" depolarındaki yazılımlar Ubuntu tarafından desteklenmemektedir. Bir yazılımın "desteklenmiyor" olması o yazılımın "önerilmediği" anlamına gelmez. Bir yazılımın Ubuntu tarafından desteklenmiyor olması o yazılımın güncelleştirmelerinin resmi olarak Ubuntu tarafından, ancak Ubuntu toplulukları tarafından sunulabileceği ile ilgilidir.
```

Abbildung 3: Die Anzeige des letzten Bytes der Datei

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Der folgende Befehl wird verwendet, um leere Verzeichnisse und Dateien in HDFS zu löschen.

- hdfs dfs -rm Leere_Datei -> Die Verwendung des Befehls
- hdfs dfs -rm /hdfs/input2/test.txt

```
hduser@ubuntu:~$ hdfs dfs -rm /hdfs/input2/test.txt
Deleted /hdfs/input2/test.txt
```

Abbildung 4: Löschung von Dateien in HDFS

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Mit dem obigen Befehl wurde das Verzeichnis /hdfs/input2 geleert. Jetzt wird das Verzeichnis /input mit dem folgenden Befehl gelöscht.

- hdfs dfs -rmdir Leer_Verzeichnis -> Die Verwendung des Befehls
- hdfs dfs -rmdir /hdfs/input2/

2.3.1 Rekursiver Löschvorgang aus dem Verzeichnis

"Als Rekursion (lateinisch recurrere ‚zurücklaufen‘) bezeichnet man den abstrakten Vorgang, dass Regeln auf ein Produkt, das sie hervorgebracht haben, von neuem angewandt werden. Hierdurch entstehen potenziell unendliche Schleifen. Regeln bzw. Regelsysteme heißen rekursiv, wenn sie die Eigenschaft haben, Rekursion im Prinzip zuzulassen.

Rekursion ist ein zentraler Begriff in Mathematik, Logik und Informatik und hat vielfältige Anwendungen darüber hinaus; diese reichen bis in die Kunst, wo das Phänomen auch als mise en abyme bezeichnet worden ist.

*Rekursion ist auch eine Problemlösungsstrategie. Komplexe Sachverhalte können oft mit rekursiv formulierten Regeln sehr elegant erfasst werden. Das Grundprinzip ist dabei dann das Zurückführen einer allgemeinen Aufgabe auf eine einfachere Aufgabe derselben Klasse. Beispielsweise ist die rekursive Programmierung Bestandteil vieler Programmiersprachen. Prozeduren oder Funktionen können sich dabei selbst aufrufen. Rekursion und Iteration sind im Wesentlichen gleichmächtige Sprachmittel. "*⁴.



Abbildung 5: Ein Beispiel von Rekursion: Rückkopplung in einer Screenshot-Software

Quelle : <https://de.wikipedia.org/wiki/Rekursion>. 17 Juli 2018

⁴ <https://de.wikipedia.org/wiki/Rekursion>. 17 Juli 2018

(Rekursion, 2018) Die ist eine Webseite, auf der die Rekursion beschrieben wird. Mit dem folgenden Befehl werden rekursiv alle Verzeichnisse mit ihren Unterverzeichnissen gelöscht.

- `hdfs dfs -rm -r Verzeichnis` -> Die Verwendung des Befehls
- `hdfs dfs -rm -r /hdfs/`

Hier wird das Verzeichnis `/hdfs/` und seine Unterverzeichnisse gelöscht.



3. MAP-REDUCE

(MapReduce, 2018) Die Mapreduce-Definition wird auf dieser Website vorgenommen. MapReduce ist ein vom Unternehmen Google eingeführtes Programmiermodell für nebenläufige Berechnungen über große Datenmengen auf Computer-Clustern. MapReduce ist auch der Name einer Implementierung des Programmiermodells in Form einer Software-Bibliothek. (Was ist MapReduce?, 2018) Diese Webseite enthält Informationen zu Mapreduce.

Beim MapReduce-Verfahren werden die Daten in drei Phasen verarbeitet (Map, Reduce und Shuffle), von denen zwei durch den Anwender spezifiziert werden (Map und Reduce). Dadurch lassen sich Berechnungen parallelisieren und auf mehrere Rechner verteilen. Bei sehr großen Datenmengen ist die Parallelisierung unter Umständen schon deshalb erforderlich, weil die Datenmengen für einen einzelnen Prozess zu groß sind.

Das Programmiermodell wurde durch die in der funktionalen Programmierung häufig verwendeten Funktionen map und reduce inspiriert, auch wenn die Arbeitsweise der Bibliothek davon abweicht. 2010 wurde für MapReduce ein US-Patent erteilt. Der wesentliche Beitrag von MapReduce ist jedoch das zu Grunde liegende System, das die Berechnungen stark parallelisiert, die Reorganisation der Daten im Shuffle-Schritt optimiert, und automatisch auf Fehler im Cluster reagieren kann, wie beispielsweise den Ausfall von kompletten Knoten⁵.

MapReduce wird häufig im Big-Data-Umfeld für beispielsweise Finanzanalysen, wissenschaftliche Simulationen oder das Data Mining verwendet. Es existiert eine ganze Reihe von Implementierungen des MapReduce-Verfahrens in Programmiersprachen wie Java, C++, Perl, Python oder Ruby⁶.

3.1 Zusammenfassung: Map Reduce

(Aunkofer, 2018) Webseite beschreibt den Map-Reduce und Algorithmus. Das Map-Reduce-Konzept und die Funktionsweise wurden aus dieser Quelle genutzt. MapReduce ist der bekannteste Algorithmus zur verteilten Verarbeitung von Daten und eignet sich für die

⁵ <https://de.wikipedia.org/wiki/MapReduce>. 17 Oktober 2018

⁶ <https://www.bigdata-insider.de/was-ist-mapreduce-a-624936/>. 28 Dezember 2018

Durchführung von komplexen Datenanalysen. Liegen Datensätze auf mehreren Computern (Client Nodes) vor, läuft der Algorithmus in der Regel in drei Schritten ab:

Map – Selektierung der Datensätze auf den Computern im gewünschten Format und Durchführung einer Berechnung, beispielsweise der Bildung einer Summe. Dieser Schritt ist ermöglicht das Prinzip Schema on Read, das aus Hadoop ein Tool zur Verarbeitung von unstrukturierten Daten macht.

Combine – Durchführung einer Aggregation, die ebenfalls auf jeden Client Node durchgeführt wird, zur Zusammenfassung von Map-Ergebnissen.

Reduce – Aggregation aller Ergebnisse auf dem zentralen Rechner (Name Node)

MapReduce ist dazu geeignet, unstrukturierte Daten zu verarbeiten, denn das Format der Daten wird über einen Map-Algorithmus bestimmt, der sehr flexibel programmiert werden kann. MapReduce ist kein eng definierter Algorithmus, sondern eine Hülle, die mit Inhalt befüllt werden muss. So müssen MapReduce-Algorithmen individuell über eine Programmiersprache wie Java, Scala oder Python programmiert werden.⁷

⁷ <https://data-science-blog.com/blog/2018/03/26/distributed-computing-mapreduce-algorithmus/>. 26 Februar 2018

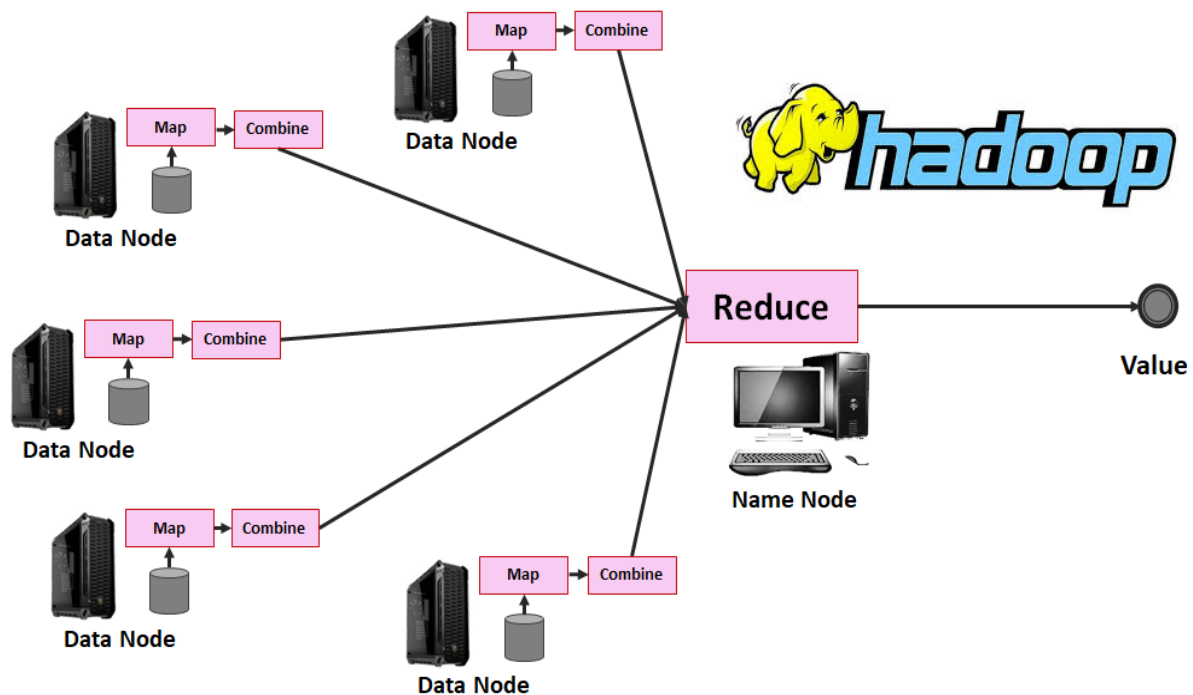


Abbildung 6: Der Datenfluss bei der MapReduce-Berechnung

Quelle : <https://data-science-blog.com/blog/2018/03/26/distributed-computing-mapreduce-algorithmus/> . 26 March 2018

3.2 Die Einrichtung der Entwicklungsumgebung

Um eine Anwendung zu entwickeln, werden eine IDE (Integrated Development Environment) und eine Java-EE benötigt.

3.2.1 Installation von Eclipse als IDE

Für die Eclipse-Installation muss Java auf Ihrem Computer installiert sein. Die Eclipse-Installation kann unter <http://www.eclipse.org/downloads/eclipse-packages/> über einen Internetbrowser erfolgen. Der Link "Eclipse IDE für Java-Entwickler" auf der Seite wird angeklickt. Und wird auf "64 Bit" angeklickt. Dann wird der Link "Download" angeklickt und die Datei "tar.gz" wird auf dem Computer gespeichert. Die Datei wird mit der rechten Maustaste "Extract Here" extrahiert. Es öffnet sich durch Klicken auf "eclipse" in dem Ordner, der geöffnet wird. Es muss als Standard festgelegt werden, um das in Eclipse installierte JDK zu starten. Dazu wird über das Hauptmenü „Window->Preferences“ eingeklickt. "Java->Installierte JREs" wird aus dem Menü auf der linken Seite ausgewählt. Dann wird auf Button

"Add" und dann auf "Standard-VM" angeklickt. Dann wird "Next" angeklickt und die Schaltfläche "Directory..." wird ausgewählt, wo das JDK installiert ist. Und mit der Schaltfläche „Finish“ wird die Installation abgeschlossen. Ein Bildschirm wird wie folgt angezeigt.

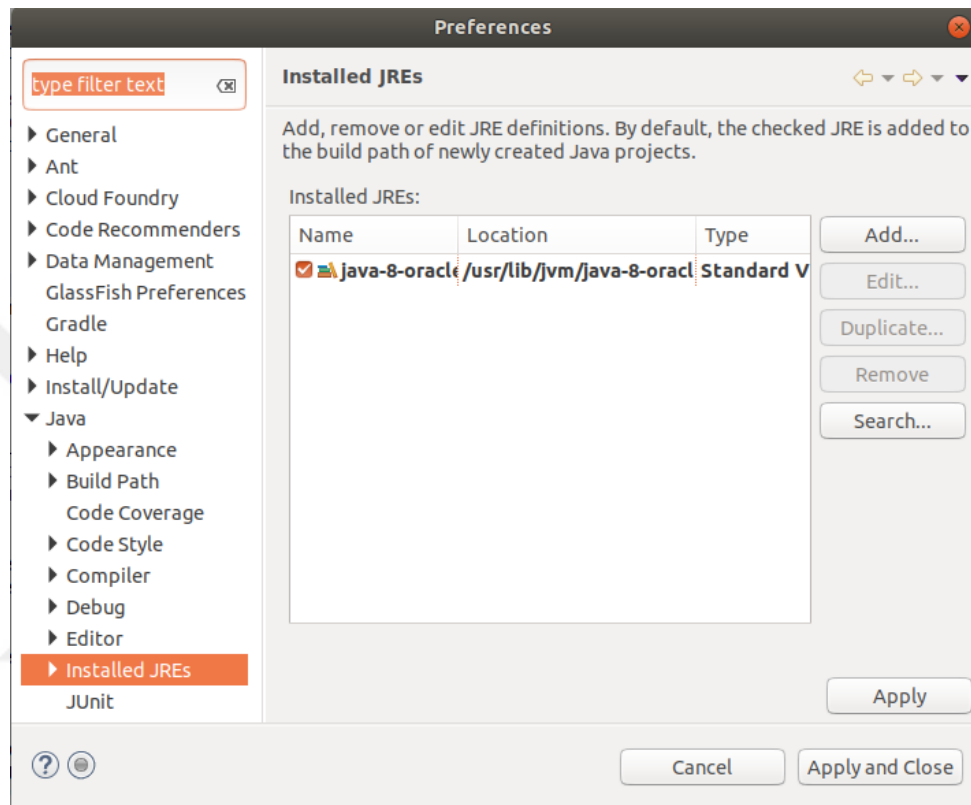


Abbildung 7: Einstellen als Standard des JDK

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Mit der Schaltfläche „Apply and Close“ wird das Dialogfenster geschlossen.

3.2.2 Installation von Glassfish als Anwendungsserver

Glassfish ist eine Open-Source-JEE-Anwendung. Die Verwendung der Admin-Seite ist einfach. Um die Ergebnisse von Hadoop-Webanwendungen anzuzeigen, muss das Glassfish-Plug-in installiert werden. Zur Installation von Plugin wird Hilfe-> Eclipse Marketplace aus dem oberen Menü verwendet. Glassfish wird im Suchbereich geschrieben. Ein Bildschirm wie der folgende erscheint.

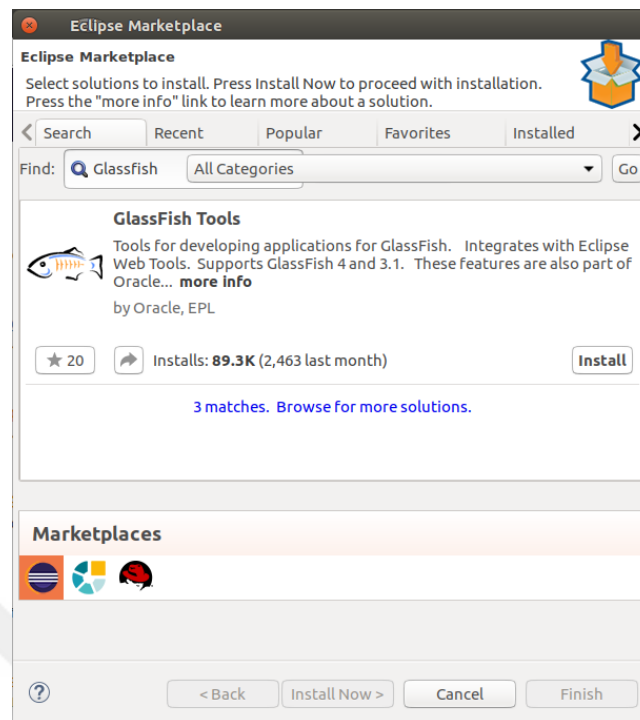


Abbildung 8: GlassFish suchen

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Die Schaltfläche „Install“ wird angeklickt, um hier zu installieren. Nachdem die Installation abgeschlossen ist, wird der Computer neu gestartet und der neue Server hinzugefügt. Dann werden die folgenden Codes eingegeben und Glassfish 4.1.2 Version wird auf dem Konsolenbildschirm heruntergeladen.

```
sudo apt-get install unzip
```

```
sudo wget http://download.java.net/glassfish/4.1.2/release/glassfish-4.1.2.zip
```

```
unzip glassfish-4.1.2.zip
```

Mit dem ersten Befehl wird das Programm "unzip" installiert. Glassfish wird mit dem zweiten Befehl heruntergeladen. Schließlich wird diese Datei glassfish-4.1.2.zip geöffnet.

Im Abschnitt Server wird "No servers are available Click this link to create a new server..." gezeigt. Dieser Link wird angeklickt und "GlassFish" ausgewählt und auf die

Schaltfläche "Weiter" geklickt. Die folgenden Einstellungen werden auf dem Display vorgenommen.

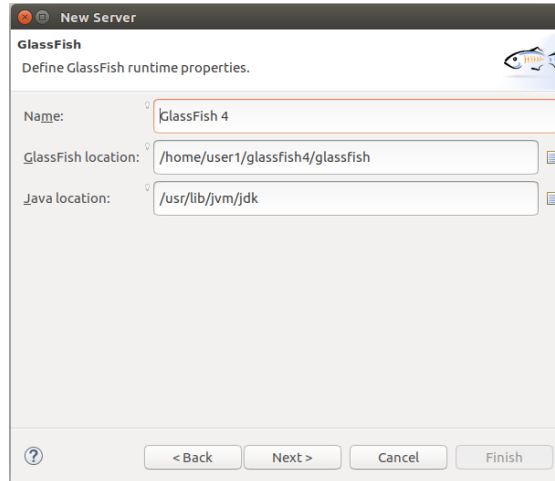


Abbildung 9: Servereinstellung

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Von hier aus wird die Schaltfläche "Weiter" geklickt, um den Vorgang abzuschließen. Nach dem Dialogfenster wird der folgende Bildschirm im Bereich Server angezeigt.

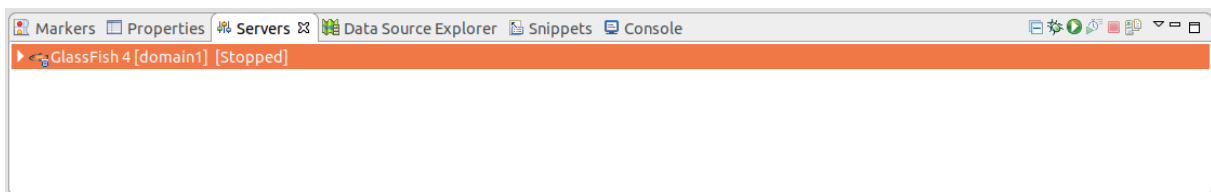


Abbildung 10: Server-Teil

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Um den Glassfish-Server zu starten, mit der rechten Maustaste darauf angeklickt und dann auf Start geklickt. Ein Ergebnis wie das Folgende ist erscheint.

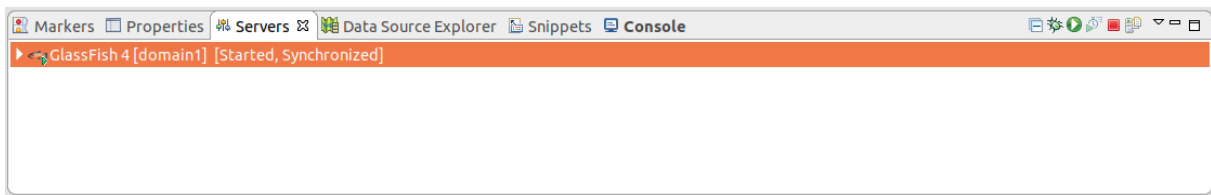


Abbildung 11: Server läuft

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

3.2.3 Erstellen eines Testprojekts

Mit einem kleinen Web-Test-Projekt kann nachgewiesen werden, dass Eclipse und Glassfish zusammenarbeiten. Mit der rechten Maustaste auf "Projekt-Explorer" wird angeklickt und "Neu-> Dynamisches Webprojekt" gewählt. Dem Projektname wird "TestProject" angegeben.

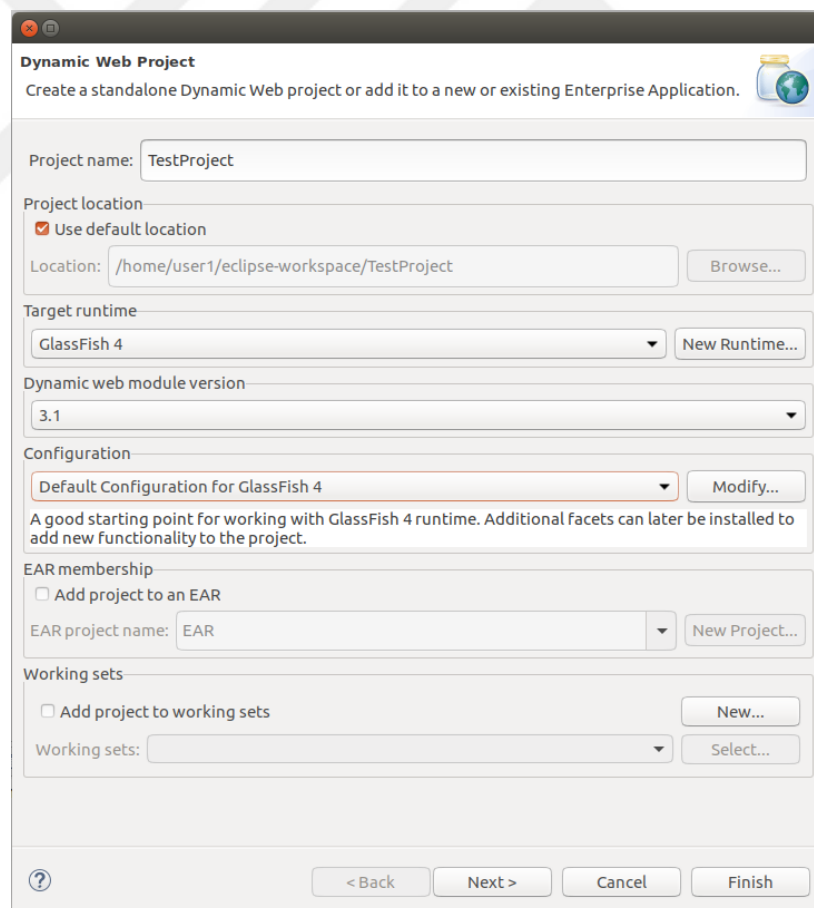


Abbildung 12: Dynamisches Webprojekt

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Es ist wie gezeigt eingestellt und auf die Schaltfläche "Finish" angeklickt. So wird das Projekt erstellt. Durch Drücken der kleinen Schaltfläche Pfeil wird das TestProject im Project Explorer geöffnet. Es wird auf den Abschnitt "Webcontent" angeklickt. Der rechte Knopf der Maus und danach "New-> JSP File" werden angeklickt. Die index.jsp wird als Name angegeben und durch Drücken der Schaltfläche Finish beendet. Die Datei index.jsp wird geöffnet und zwischen die Zeilen body "Bu bir test projesidir." geschrieben und gespeichert.



Abbildung 13: Testprojekt

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Im Projekt auf die Datei index.jsp wird mit der rechten Maustaste geklickt. "Run As->Run on Server" ist ausgewählt und auf die Schaltfläche Finish angeklickt. Das Ergebnis wird auf der Webseite angezeigt.

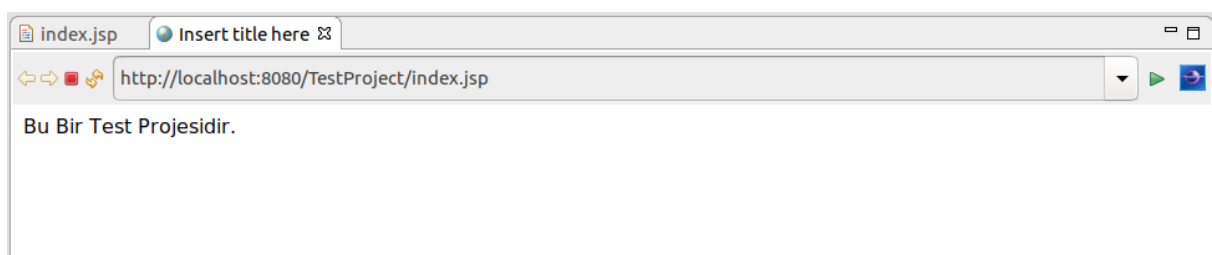


Abbildung 14: Webserver-Ergebnis

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

3.2.4 Die Installation von Maven-Plug

Maven ist ein Tool, das in dem Projekt einen Standard erstellt wird, das das Projekt vereinfacht wird, mit dem die Projektdokumentation effektiv erstellt werden kann, während das

Projekt entwickelt wird. Es ist in Eclipse vorinstalliert. Zur Installation wird 'Hilfe->Neue Software installieren ...' aus dem Hauptmenü aufgerufen. Der Link wird als "<http://download.eclipse.org/technology/m2e/milestones/1.0>" geschrieben. Die Abschnitte, die installiert wird, werden markiert und auf die Schaltfläche 'Next' angeklickt. Dann wird Eclipse neu gestartet und die Installation ist abgeschlossen.

3.2.5 Die Installation von WinSCP

Das WinSCP-Programm ist wie FTP-Programme. Das Kopieren und Speichern einiger Dateien werden in Zukunft erforderlich sein. Mit diesem Programm kann nicht nur auf die Domäne, sondern auf alle Server- und Systemdateien zugegriffen werden. Das Programm "wine" wird vor der Installation des WinSCP-Programms installiert. Wine ist ein Programm, das die Dateien der Erweiterung ".exe" in Windows unter Linux installiert. Um dieses Programm zu installieren, im Abschnitt "Software" wird "Wine" geschrieben und installiert. Dann wird das WinSCP-Programm unter "<https://winscp.net/eng/download.php>" heruntergeladen. Dann wird der folgende Befehl eingegeben.

- wine Downloads/WinSCP-5.9.6-Setup.exe

Ein Installationsdialogfeld wird geöffnet. Durch Drücken der Schaltfläche Next wird der Installationsprozess beendet.

3.3 Map-Reduce Anwendung

Erstens können die zu verarbeitenden Daten als Java-Projekt erstellt werden. Die erstellte Datei ist company_giro.txt und hat 5000 Datensätze. Im oberen Menü von Eclipse wird "Datei->New-> Other" ausgewählt und Java-Projekt wird aus der Liste ausgewählt. Ein Projekt mit dem Namen TestDataGenerator wird erstellt. Mit Eclipse unter dem Projekt wird Datei "src" angeklickt. Mit dem Rechtsklick "Neu" und dann "Paket" werden ausgewählt. Der Name wird als "Test" angegeben. Unter Test wird eine Java-Datei mit dem Namen TestData erstellt. Eine Hierarchie wird wie unten gezeigt erstellt.

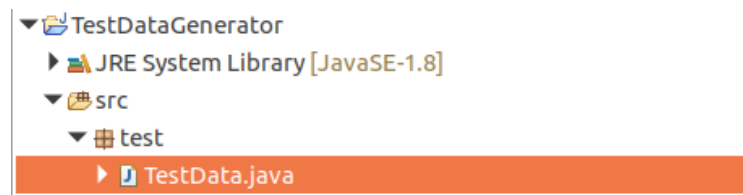


Abbildung 15: Die Datei TestData.java

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Die Datei TestData.java wird geöffnet und die folgenden Codes werden geschrieben.

```
package test;

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;

public class TestData {

    private final static Logger log = Logger.getLogger(TestData.class.getName());

    /**
     * Uygulamanin dizininde firma ismi, tarih ve firmanin cirosundan
     * olusan kayitlarin listesi olusturuluyor.
     * @param args - kullanilmiyor
     */
}
```

```

public static void main(String[] args) {

    int data_counter = 5000;

    log.log(Level.INFO, data_counter+" kayıt yaziliyor...");

    try {

        Random r = new Random();

        PrintWriter out = new PrintWriter("company_giro.txt");

        String company_name, giro_str;

        Date test_date = new Date();

        Calendar c = Calendar.getInstance();

        DateFormat dfmt = new SimpleDateFormat("ddMMyyyy");

        for(int i=0; i<data_counter; i++) {

            // Firma ismi

            company_name = " Kommunikationsfirma";

            // Cironun tarihi

            c.set(Calendar.YEAR,      2000 + r.nextInt(17));

            c.set(Calendar.MONTH,    1  + r.nextInt(11));

            c.set(Calendar.DAY_OF_MONTH, 1  + r.nextInt(30));

            test_date = c.getTime();

            // Ciro

            giro_str = String.valueOf(r.nextInt(100000));

            while( giro_str.length() < 5)

                giro_str = "0"+ giro_str;
        }
    }
}

```

```

        // Dosyaya yaz

        out.print(company_name);

        out.print(dfmt.format(test_date));

        out.print(giro_str);

        out.print(System.getProperty("line.separator"));

    }

    out.close();

    log.log(Level.INFO, "Bitti!");
} catch (FileNotFoundException e) {
    log.log(Level.SEVERE, "Yazmada Hata!");
    e.printStackTrace();
}
}
}

```

Im Projekt ist Run As-> Java Application ausgewählt. Als Ergebnis wurden 5.000 Datensätze mit dem Namen company_giro.txt erstellt. Beispiel Datensatz ist wie folgt.

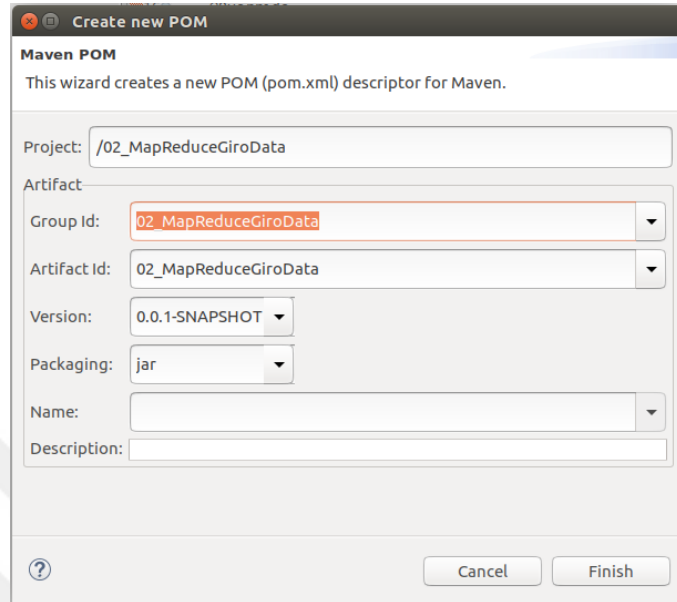
Der Firmenname Kommunikationsfirma, Tag (27), Monat (10), Jahr (2006) und Umsatz (99409)

Beispiel: Kommunikationsfirma2710200699409

3.3.1 Die Erstellung eines Maven-Projekts

Maven bietet einen Standard innerhalb des Projekts, vereinfacht den Projektentwicklungsprozess, erstellt effektiv Dokumentation und eliminiert Bibliotheksabhängigkeit und IDE-Abhängigkeit. Das erstellte Projekt in "Package Explorer" wird mit der rechten Maustaste angeklickt. Aus dem Popup-Menü "Configure->Convert to

Maven Project" wird ausgewählt. Ein Fenster wie das folgende erscheint und die Schaltfläche "Finish" wird angeklickt.



The screenshot shows a "Create new POM" dialog box. The title bar says "Create new POM". Inside, the title is "Maven POM" and the subtitle is "This wizard creates a new POM (pom.xml) descriptor for Maven." The fields are as follows:

- Project: /02_MapReduceGiroData
- Artifact:
 - Group Id: 02_MapReduceGiroData
 - Artifact Id: 02_MapReduceGiroData
- Version: 0.0.1-SNAPSHOT
- Packaging: jar
- Name: (empty)
- Description: (empty)

At the bottom right, there are "Cancel" and "Finish" buttons. A help icon (?) is at the bottom left.

Abbildung 16: Die Erstellung eines Maven-Projekts

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Nach diesem Schritt wird ein Dialogfenster wie das folgende erscheint.

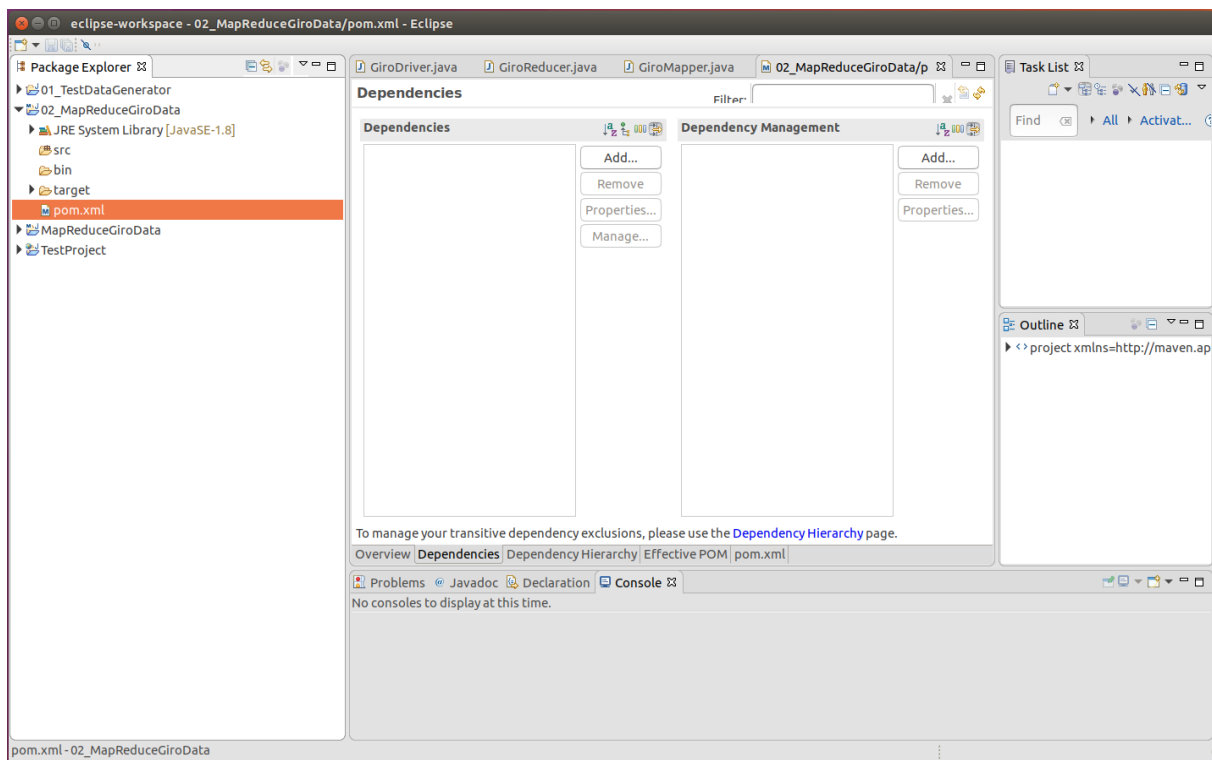


Abbildung 17: Die Einstellung des Mavens

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Durch einen Doppelklick auf die "pom.xml" unter dem Projekt kann dieser Bildschirm angezeigt werden. "Dependencies" wird am unteren Bildschirmrand ausgewählt. Auf diesem Bildschirm wird die Schaltfläche Add... angeklickt. Es ist wichtig, hier die korrekte Version von hadoop in das Feld Version zu schreiben. Andernfalls kann dies zu Verwirrung verursachen. Wenn die alte Version geschrieben wird, versteht jemand, der sie betrachtet, dass der hadoop-client anstelle von hadoop-core als "Artifact Id" verwendet wird. Das Bereichsfeld wird als "provided" ausgewählt und auf OK geklickt, um zu speichern und der Vorgang ist abgeschlossen.

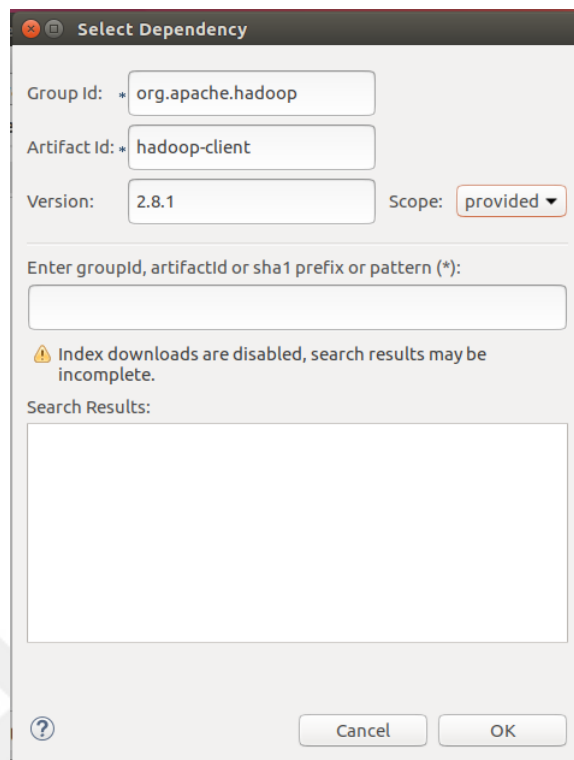


Abbildung 18: Maven ID und Versionsauswahl

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

3.3.2 Erstellen von Run-Einstellungen zum Generieren der JAR-Datei

Auf den Projektnamen wird mit der rechten Maustaste angeklickt. "Run As->Maven build" wird ausgewählt. Das Projekt sieht wie das Bild unten aus.

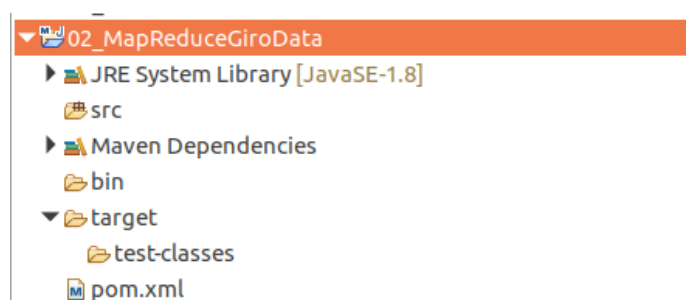


Abbildung 19: Maven Projekt

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Im Fenster "Run Configurations" wird "clean compile assembly: single" in den Abschnitt Goals geschrieben. Die Schaltflächen "Apply" und "Close" werden nacheinander angeklickt.

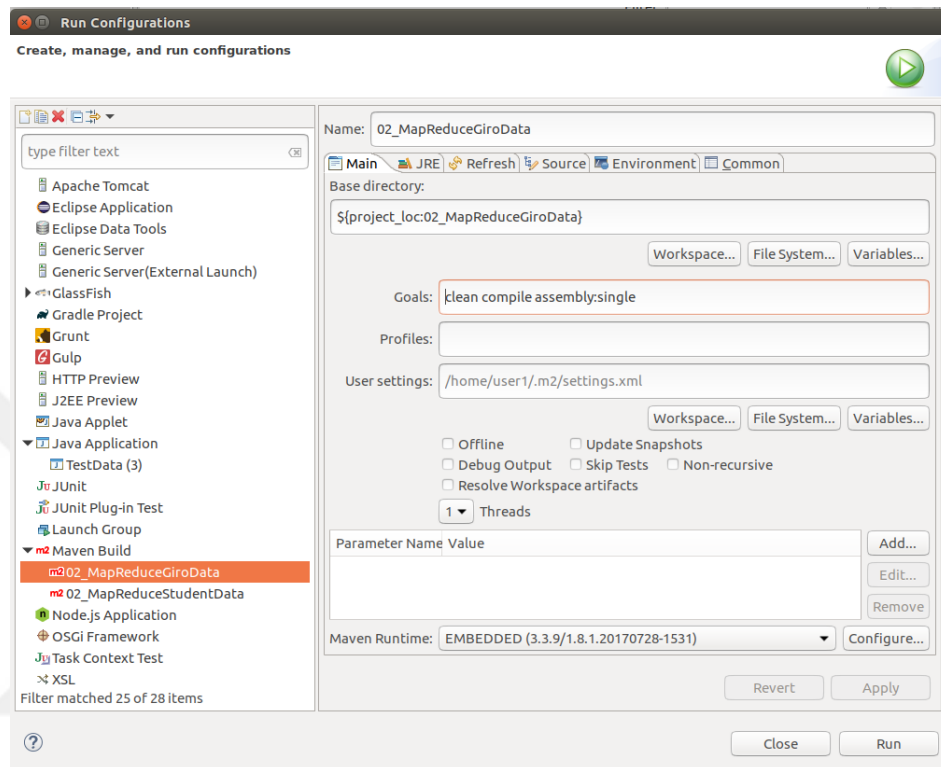


Abbildung 20: Die Einstellungen

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

3.3.3 Map Reduce Programmierung

Zuerst wird ein "Package" benötigt. Es werden drei Klassen erstellt, unter denen Map-Reduce Prozess gesteuert werden. Unter dem Projekt wird auf die Datei "src" mit der rechten Maustaste angeklickt. New->Package ist ausgewählt. "Mapreduce" wird als Name angegeben und die Schaltfläche "Finish" wird angeklickt.

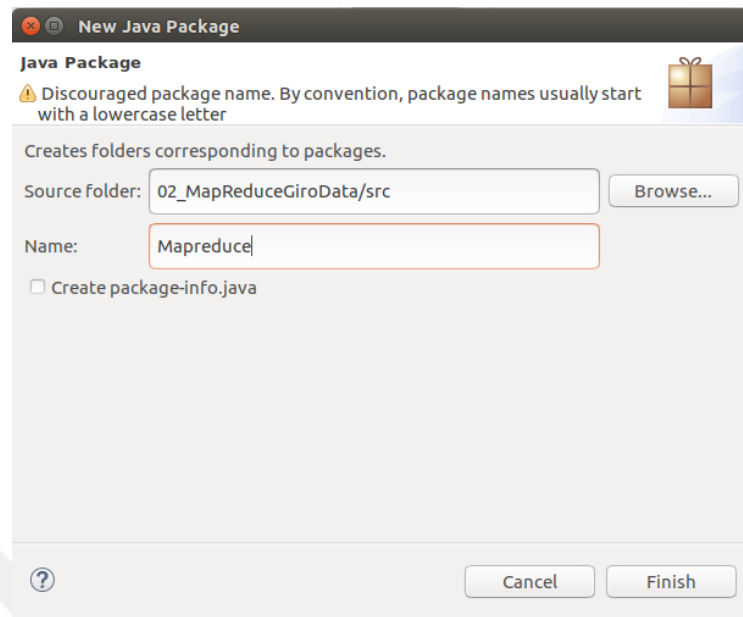


Abbildung 21: Die Erstellung eines neuen Pakets

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Auf der Projektname "Mapreduce" wird mit der rechten Maustaste geklickt und "New-> Class" ausgewählt. Neue Klassen werden mit den folgenden Namen erstellt.

- 1- GiroDriver
- 2- GiroMapper
- 3- GiroReducer

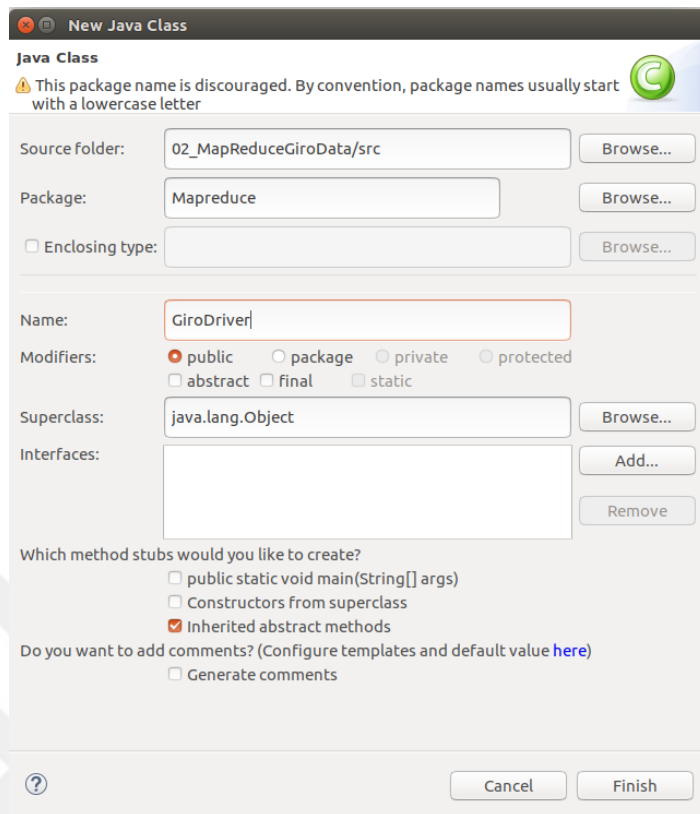


Abbildung 22: Die Erstellung einer neuen Klasse

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Das Projekt sieht wie folgt aus.

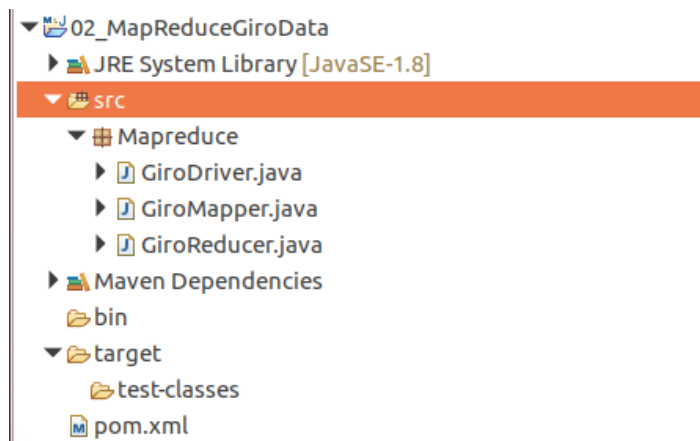


Abbildung 23: Projektansicht

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Es beginnt mit der Anwendung "Mapper". Das Programm "Mapper" wird die Zeile für Zeile (das Jahr und das Giro) aus der Textdatei gelesen. Für jedes Jahr wird aus den Schlüsselwertpaaren eine "HashMap" erstellt, die einen Giro gespeichert. Die Codes der Klasse „GiroMapper“ beziehungsweise Map-Operation sind wie folgt.

```
package Mapreduce;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

// Giris-Anahtar, Giris-Deger, Cikis-Anahtar, Cikis-Deger

public class GiroMapper extends Mapper<Text,Text,IntWritable,IntWritable> {

    private          final          static          Logger          log          =
    Logger.getLogger(GiroMapper.class.getName());

    private IntWritable year_int = null;

    private IntWritable giro_int = null;

    public void map(Text key, Text value, Context context) throws IOException,
        InterruptedException {

        // Kommunikationsfirma2710200699409 gibi bir String'den yıl ve
        cironun okunmasi

        if (key.toString().length() == 21) {

            String year_str = key.toString().substring(12,16);

            String giro_str = key.toString().substring(16,21);

            year_int = new IntWritable(Integer.parseInt(year_str));
```

```

        giro_int = new IntWritable(Integer.parseInt(giro_str));

        // Sunuclar biraraya getiriliyor
        context.write(year_int, giro_int);

    } else {

        log.log(Level.INFO, "Gecersiz veri uzunlugu bulundu (" + key.toString().length() + ").");

    }

}

}

```

Die GiroMapper-Klasse, die die Datentypen angibt, die auf die Eingänge und Ausgänge des "Mappers" angewendet werden müssen, wird angezeigt. "Mapper" erhält Eingabeschlüssel und Wert. Als Ausgabeschlüssel und wert wird der Integer-Wert gesendet.

Es gibt zwei Klassen für Mapper und Reducer in Hadoop. Die erste befindet sich unter org.apache.hadoop.mapreduce und org.apache.hadoop.mapred. Der zweite ist alt und wird nicht mehr benutzt.

Ein Schlüssel-Wert-Paar wird in der Map-Methode berücksichtigt. Jahres- und Giros Daten werden mit "substring" gelesen. Schließlich wird es mit "Integer.parseInt" in natürliche Zahlen umgewandelt. "Context.write" wird die Kombination von Jahr und Umsatz zusammenbringt. Diese Liste von Schlüssel-Wert-Paaren wird den Gesamtumsatz für jedes Jahr mit dem "Reducer" berechnet. Die Klasse " GiroReducer" erforderlichen Codes lauten wie folgt:

```

package Mapreduce;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.io.FloatWritable;

```

```

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Reducer;

// Giris-Anahtar, Giris-Deger, Cikis-Anahtar, Cikis-Deger

public class GiroReducer extends Reducer<IntWritable, IntWritable, IntWritable,
FloatWritable> {

    private final static Logger log =
Logger.getLogger(GiroMapper.class.getName());

    @Override
    protected void reduce(IntWritable key, Iterable<IntWritable> values, Context
context)

        throws IOException, InterruptedException {

        // Bir yilin bütün ciroları toplaniyor.

        float sum = 0;

        float count = 0;

        for (IntWritable val : values) {

            sum +=val.get();

            count +=1;

        }

        // Ortalamasi olusturuluyor

        float result = sum / count;

        log.log(Level.INFO, "Yıl: "+key+" Sonuc: "+result);

        // Yıl için toplam yaziliyor

        context.write(key, new FloatWritable(result));

    }

}

```

"Reducer" wird einen Schlüssel (Jahr) bekommen und einen (Umsatz) gesendet. Bei der Methode "Reduzieren" werden alle Schlüssel als ein Schlüssel abgeleitet. Gemeinsame Werte werden gesammelt. Alle Jahr-Giro Kombinationen werden über "context.write" in einer Liste gespeichert. An dieser Stelle wird die Klasse GiroDriver geschrieben, die Mapper und Reducer koordiniert. Der Code für die Klasse "GiroDriver" lautet wie folgt.

```
package Mapreduce;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.input.KeyValueTextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class GiroDriver extends Configured implements Tool {

    private final static Logger log = Logger.getLogger(GiroDriver.class.getName());

    public static void main(String[] args) {
```

```

int res = 1; // Eger 1 degistirilmiyorsa, sorun var demektir.

try {

    res = ToolRunner.run( new Configuration(), new GiroDriver(), args );

}

catch (Exception e) {

    log.log(Level.SEVERE, "Calisma sirasinda hata!");

    e.printStackTrace();

}

System.exit(res);
}

/*
 * isi tasarlar ve baslar.
 * @see org.apache.hadoop.util.Tool#run(java.lang.String[])
 */

@Override

public int run(String[] args) {

    log.log(Level.INFO, "Map-Reduce 'GiroDriver' basliyor... ");

    // Mevcut konfigürasyon getConf ile cagrilarak, Configured genisletilir

    Configuration conf = this.getConf();

    Job job = null;

    try {

        job = Job.getInstance(conf);

    } catch (IOException e1) {

        log.log(Level.SEVERE, "Orneklemede hata!");
    }
}

```

```

        e1.printStackTrace();
    }

    // Hadoop GiroDriver'in icerdigi mevcut bir Jar uygulaması gerekir
    job.setJarByClass(GiroDriver.class);

    // Mapper ve Reducer siniflari belirleniyor
    job.setMapperClass(GiroMapper.class);
    job.setReducerClass(GiroReducer.class);

    // Cikis turleri belirleniyor
    job.setOutputKeyClass(IntWritable.class);
    job.setOutputValueClass(FloatWritable.class);
    job.setMapOutputKeyClass(IntWritable.class);
    job.setMapOutputValueClass(IntWritable.class);
    job.setInputFormatClass(KeyValueTextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    // Jar baslarken Hadoop da girdi dosyalarini listeleyen yol ilk argüman
    olarak gecirilir.

    try {
        FileInputFormat.addInputPath(job, new Path(args[0]));
    } catch (IllegalArgumentException e) {
        log.log(Level.SEVERE, "Giris yolunun ayarlanmasında arguman
        hatasi!");

        e.printStackTrace();
    } catch (IOException e) {
        log.log(Level.SEVERE, "Giris yolunun ayarlanmasında arguman
        hatasi!");
    }

```

```

        e.printStackTrace();
    }

    // Cikis dizini ikinci arguman olarak gecilir.
    FileOutputStream.setOutputPath(job, new Path(args[1]));

    boolean result = false;

    try {

        // Bitene kadar isi gerceklestirir ve bekler

        result = job.waitForCompletion(true);

    } catch (ClassNotFoundException e) {

        log.log(Level.SEVERE, "job gerceklestirilmesinde ClassNotFoundException
hatasi!");

        e.printStackTrace();

    } catch (IOException e) {

        log.log(Level.SEVERE, "job gerceklestirilmesinde IOException
hatasi!");

        e.printStackTrace();

    } catch (InterruptedException e) {

        log.log(Level.SEVERE, "job gerceklestirilmesinde InterruptedException
hatasi!");

        e.printStackTrace();

    }

    log.log(Level.INFO, "Bitti!");

    return result ? 0 : 1;

}

}

```

Es gibt Tool- und ToolRunner-Klassen zum Isolieren von Vorgängen wie dem Parsen von Xml-Dateien und Starten von Jobs. Beim Codieren mit diesen Klassen werden die Parameter in den Konfigurationsdateien automatisch geladen und Jobs ausgeführt. Die Argumente der ToolRunner-Klasse sind in der folgenden Tabelle angegeben.

Tabelle 1
ToolRunner-Klassenargumente

Parameter	Definition
-D özellik=isim	Hadoop wird mit einem festgelegten Wert bereitgestellt. Zum Beispiel: mapreduce.framework.name=yarn
-conf dosyaismi	Es fügt der Konfiguration eine Datei mit vielen Eigenschaften und unterschiedlichen Werten hinzu. Wenn Sie verschiedene komplexe Einstellungen aktivieren und deaktivieren möchten, ist das sinnvoll.
-fs URI	Es setzt das Standard-Dateisystem auf einen bestimmten URI. Es kann auf hdfs://localhost:9000 gesetzt werden. Als alternativ kann -D fs.defaultFS=uri gesetzt werden.
-jt host:port	Job-Trackers (URI ve Port) werden eingestellt. Diese werden durch 'Resource Manager' und 'Job Scheduler'

	in 'YARN' geändert. Hadoop 1.x enthält diese Funktion ebenfalls.
-files Datei1, Datei2...	Kopiert die angegebenen Dateien in das Dateisystem, das im lokalen Dateisystem mit "fs.defaultFS" angegeben wurde. Dies ist in der Regel in HDFS verfügbar.
-archives Dosya1, Dosya2	Der Parameter verhält sich genauso wie die Dateien, die die Archivdateien akzeptieren und dieses Zielverzeichnis öffnen.
-libjars JAR1, JAR2...	Die angegebenen JARs stellen den Klassenpfad in Map-Reduce-Jobs bereit. Eine gültige Alternative zu dieser Methode besteht darin, alle JARs für unsere Methoden mit Maven zu öffnen. Es ist für die Anbringung unserer Job.

3.3.4 Exportieren als JAR über Maven

Die JAR unterscheidet sich nicht von Archivdateien. Diese Datei kann geöffnet und angezeigt werden.

Um Map-Reduce-Job nach der Programmierung zu implementieren, müssen wir es in ein Java-Archiv packen. Unter dem Menü neben dem weißen Dreieck und dem grünen Kreis wird auf die Schaltfläche "OK" (Schaltfläche "Run") angeklickt. Wenn "Build-Jobs" nicht ausgeführt wurde, wird "Run->Run Configurations ..." ausgewählt, dann wird Build-Job aus der Liste ausgewählt. "clean compile assembly:single" wird in dem Feld "Goals" geschrieben und auf die Schaltfläche Run geklickt.

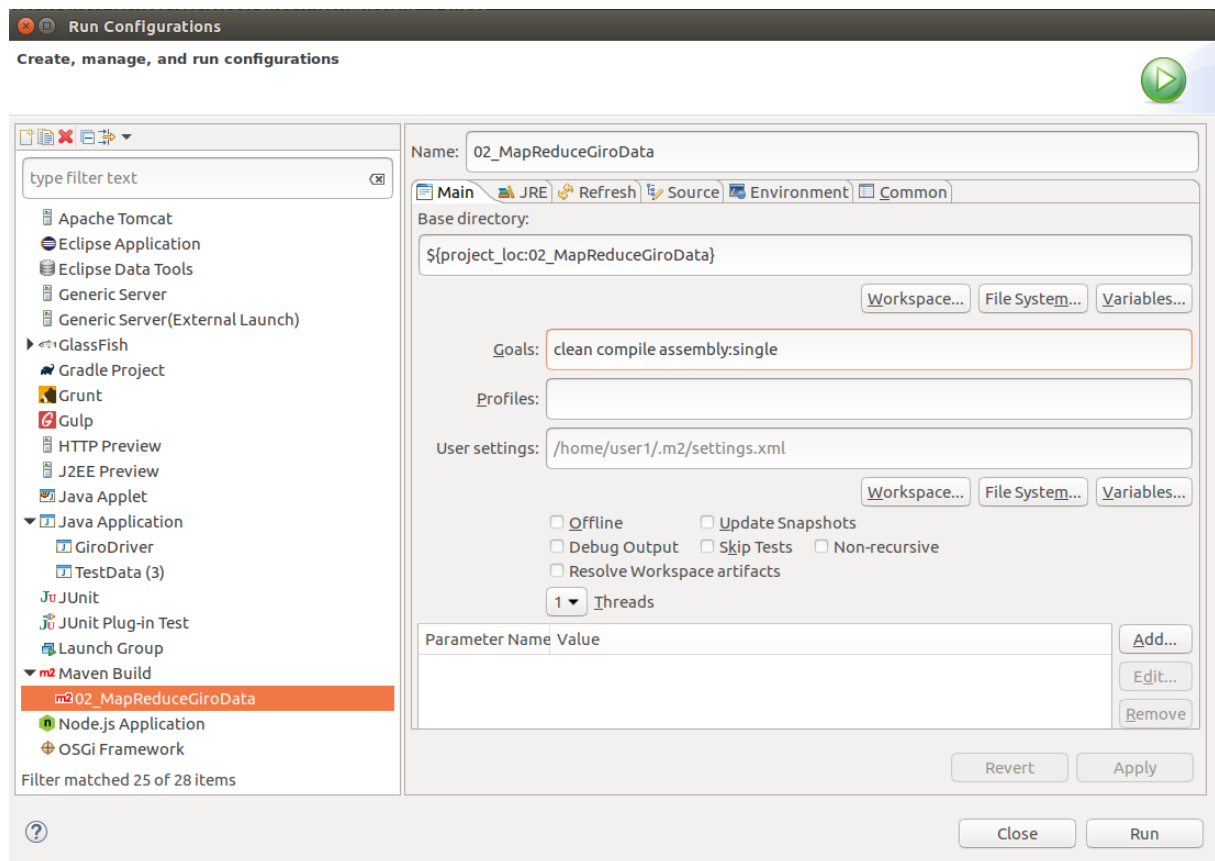


Abbildung 24: Programmeinstellungen

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Eine Datei mit dem Namen "02-MapReduceGirodata-0.0.1-SNAPSHOT-jar-with-dependencies.jar" wird unter der Zieldatei des Projekts erstellt.

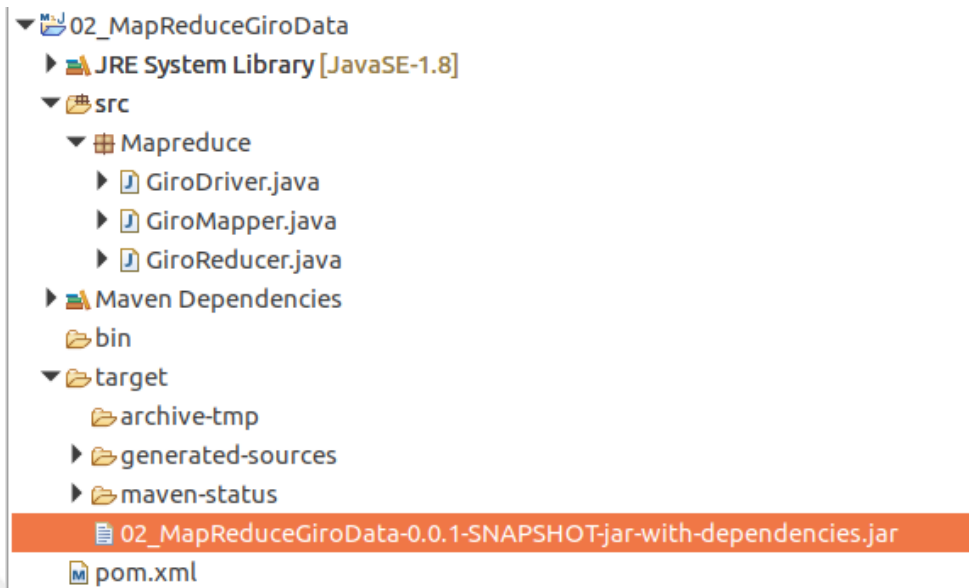


Abbildung 25: Die Erstellung der Datei ".jar"

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Damit ist der Entwicklungsteil der ersten Map-Reduce-Jobs abgeschlossen.

3.3.5 Ausführen einen Job über eine Befehlszeile

In Eclipse wird die JAR-Datei auf den Computer in Verzeichnis "/usr/local/hadoop" kopiert. Der Name wird als "02_MapReduceStudentData.jar" abgekürzt. Das WinSCP-Programm wird geöffnet. Im Registrierungsfenster wird "New Site" angeklickt. Die IP-Adresse der VM, auf der Hadoop installiert wurde, wird eingegeben, der Benutzername wird als "hduser" angegeben und durch Klicken auf "Speichern" gespeichert.

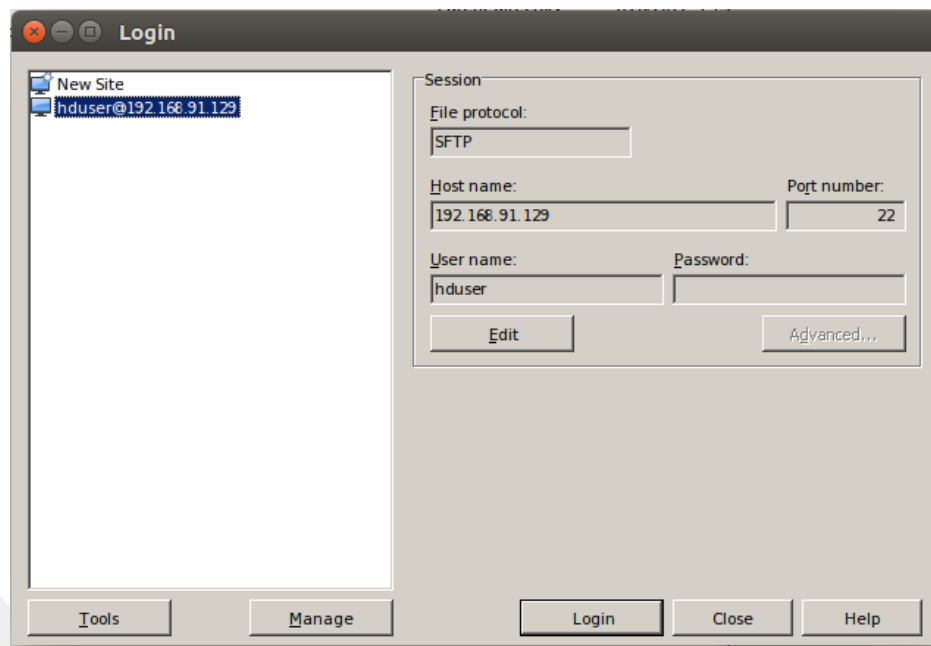
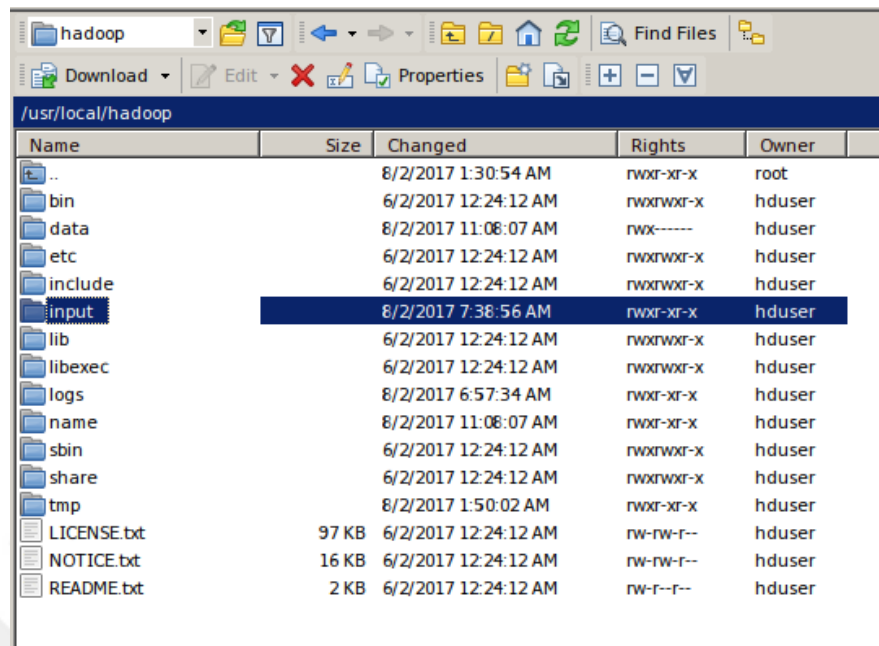


Abbildung 26: Einloggen mit WinSCP

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Auf den erstellten Benutzernamen und dann auf die Schaltfläche "Login" wird angeklickt. Das Passwort (hduser) wird in dem sich öffnenden Fenster eingegeben und der Schaltfläche "OK" wird angeklickt und eingeloggt. Das Verzeichnis wird als "/usr/local/hadoop" geändert. Rechtsklick auf "Neu-> Verzeichnis ..." ist ausgewählt und ein Ordner mit dem Namen "input" wird erstellt. Die im Projekt "01_TestDataGenerator" erstellte Datei "company_giro.txt" wird in den Ordner "input" kopiert. Der Vorgang ist abgeschlossen, wie unten gezeigt.



Name	Size	Changed	Rights	Owner
..		8/2/2017 1:30:54 AM	rw-r--r--	root
bin		6/2/2017 12:24:12 AM	rw-rw-r--	hduser
data		8/2/2017 11:08:07 AM	rw-r--r--	hduser
etc		6/2/2017 12:24:12 AM	rw-rw-r--	hduser
include		6/2/2017 12:24:12 AM	rw-rw-r--	hduser
input		8/2/2017 7:38:56 AM	rw-r--r--	hduser
lib		6/2/2017 12:24:12 AM	rw-rw-r--	hduser
libexec		6/2/2017 12:24:12 AM	rw-rw-r--	hduser
logs		8/2/2017 6:57:34 AM	rw-r--r--	hduser
name		8/2/2017 11:08:07 AM	rw-r--r--	hduser
sbin		6/2/2017 12:24:12 AM	rw-rw-r--	hduser
share		6/2/2017 12:24:12 AM	rw-rw-r--	hduser
tmp		8/2/2017 1:50:02 AM	rw-r--r--	hduser
LICENSE.txt	97 KB	6/2/2017 12:24:12 AM	rw-rw-r--	hduser
NOTICE.txt	16 KB	6/2/2017 12:24:12 AM	rw-rw-r--	hduser
README.txt	2 KB	6/2/2017 12:24:12 AM	rw-r--r--	hduser

Abbildung 27: Kopieren von Dateien mit WinSCP

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Der Benutzer wird als "hduser" geändert. Die Dienste werden gestartet. Wenn sich unter "/usr/local/hadoop" ein Ordner "output" befindet, wird dieser mit dem folgenden Befehl gelöscht.

```
cd /usr/local/hadoop
```

```
rm -rf output
```

Dann wird der folgende Befehl in der Befehlszeile eingegeben.

```
bin/hadoop jar 02_MapReduceGiroData.jar file:/usr/local/hadoop/input
file:/usr/local/hadoop/output
```

Dieser Vorgang kann einige Minuten dauern. Das Ergebnis der Vorgang kann auf der Weboberfläche unter "http://192.168.91.129:8088/cluster/apps" einsehen.

Show 20 entries													
ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	
application_1502109434886_0001	hduser	02_MapReduceGiroData.jar	MAPREDUCE	default	0	Mon Aug 7 05:38:30 -0700 2017	Mon Aug 7 05:38:48 -0700 2017	FINISHED	SUCCEEDED	N/A	N/A	N/A	

Showing 1 to 1 of 1 entries

Abbildung 28: Anzeige mit Weboberfläche

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Nach diesem Vorgang wird der Ordner "output" erstellt. Die Datei "part-r-00000" befindet sich im Ordner. Die Ausgabe wird im Ordner "output" auf dem Konsolenbildschirm eingegeben, um das Ergebnis anzuzeigen. Der folgende Befehl wird eingegeben.

```
nano part-r-00000
```

Als Ergebnis wird das folgende Bild erhalten.

2000	50585.652
2001	51394.43
2002	50680.855
2003	49488.27
2004	49381.152
2005	50919.586
2006	50217.215
2007	48974.285
2008	50456.656
2009	48328.406
2010	52025.387
2011	52328.86
2012	51567.29
2013	49450.875
2014	50100.652
2015	51798.29
2016	48295.98

[Read 17 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
 ^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

Abbildung 29: Das Ergebnis wird im Terminal angezeigt

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

So wird nach den Jahren das Giro des Unternehmens erhalten.

4. ARCHITEKTUR VON HIVE

(Freiknecht, 2014) Dieses Buch enthält Hadoop und Mapreduce in der Praxis. Die Voraussetzungen für die Anwendungen wurden aus diesem Buch genutzt. Die Apache Hive Architektur wird wie folgt dargestellt. HQL-Abfragen (Hive Query Language), die von der Webschnittstelle oder vom Hive-Client ausgeführt werden, werden im Hintergrund unter Verwendung der zuvor eingegebenen Schemainformationen in MapReduce-Jobs übersetzt. Hive kommt mit Thrift-Server, wenn es installiert ist. Der Zweck dieses Dienstes besteht darin, Zugriff auf bestimmte Ports auf Hive bereitzustellen.

(What is **Hive**? Architecture & Modes, 2018) Die Hive-Architektur wird auf dieser Webseite beschrieben. Hive funktioniert im Zusammenspiel mit Hadoop, da es einerseits das HDFS nutzt, um seine Daten dort abzulegen, und andererseits Queries in Map-Reduce-Jobs übersetzt und auf Hadoop ausführt⁸.

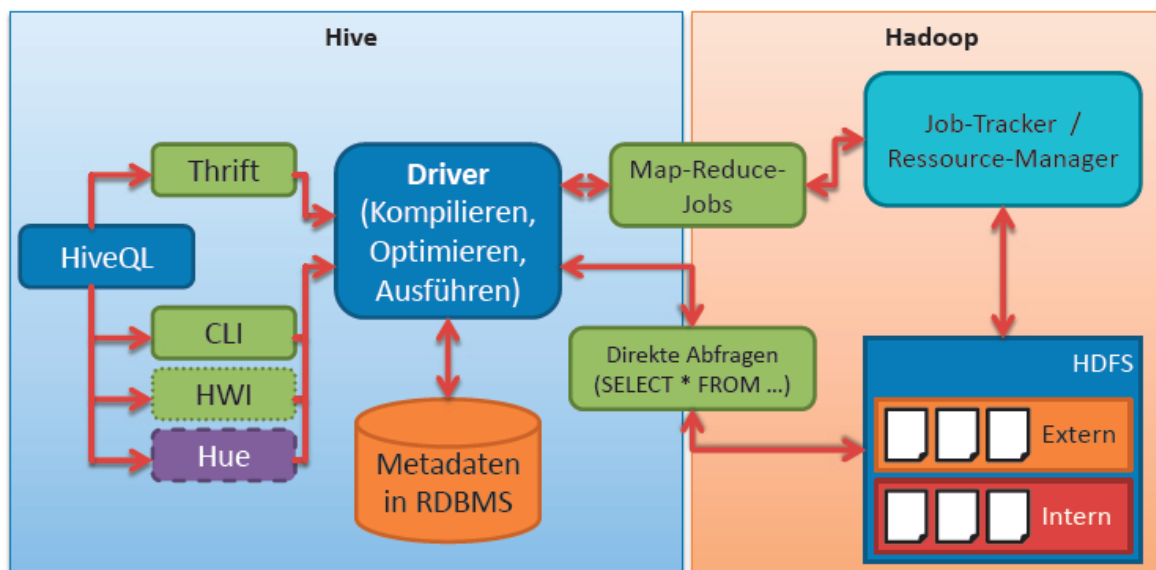


Abbildung 30: Architektur von Apache Hive im Zusammenspiel mit Hadoop

Quelle : Freiknecht, J., BIG DATA in der Praxis, Germany, 2014.

Der Metastore wird, falls nicht anders konfiguriert, in einer Derby-Datenbank angelegt. Auf Wunsch kann er jedoch auch in einer anderen relationalen Datenbank angelegt

⁸ Freiknecht, J., BIG DATA in der Praxis, Germany, 2014.

werden, etwa MySQL oder DB2. Da dieser Prozess jedoch nur in wenigen Fällen notwendig und etwas komplexer ist, hier wird nur auf das Hive-Wiki verwiesen, das die nötigen Schritte dazu auflistet⁹.

4.1 Die Architektur von Apache Hive

Hive besteht hauptsächlich aus 3 Hauptteilen

- Hive Clients
- Hive Services
- Hive Storage and compute

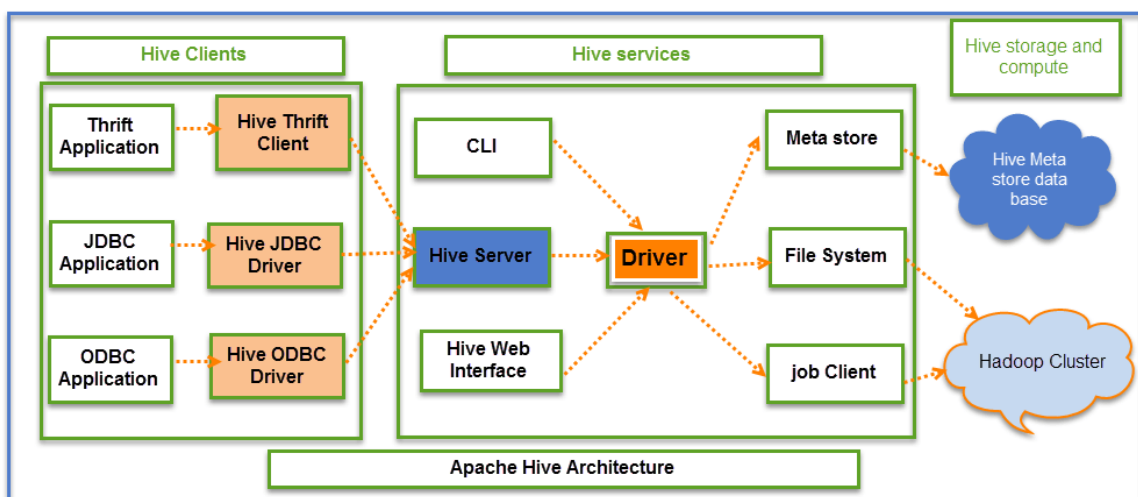


Abbildung 31: Architektur von Apache Hive

Quelle : <https://www.guru99.com/introduction-hive.html>. 23 November 2018

CLI ist die Befehlszeilenschnittstelle zu Hive (die Shell). Dies ist der Standarddienst. Beeline ist eine Befehlszeilenschnittstelle zu Hive, die im eingebetteten Modus (wie die reguläre CLI) oder durch die Verbindung mit einem HiveServer2-Prozess mithilfe von JDBC ausgeführt wird. Der Metastore wird im selben Prozess wie der Hive-Dienst ausgeführt. Mit diesem Service ist es möglich, den Metastore als eigenständigen (Remote-) Prozess auszuführen. Der Hive-Server wird als ein Thrift-Dienst verfügbar gemacht, so dass es möglich

⁹ Freiknecht, J., BIG DATA in der Praxis, Germany, 2014.

ist, mit ihm unter Verwendung jeder Programmiersprache, die Thrift unterstützt, zu interagieren. Hive stellt einen reinen Java-JDBC-Treiber bereit, der in der Klasse `org.apache.hadoop.hive.jdbc.HiveDriver` definiert ist. Ein ODBC-Treiber ermöglicht Anwendungen, die das ODBC-Protokoll unterstützen, eine Verbindung zu Hive herzustellen.



5. INSTALLATION VON HIVE

Zur Installation von Hive wird der Bildschirm der Linux-Konsole verwendet. Das Terminal wird geöffnet. Für die Hive-Einrichtung wurden die folgenden Befehle der Reihe nach eingegeben.

- `su - user1`
- `cd /usr/local`
- `sudo wget https://archive.apache.org/dist/hive/hive-1.2.1/apache-hive-1.2.1-bin.tar.gz`
- `sudo tar xzf apache-hive-1.2.1-bin.tar.gz`
- `sudo mv apache-hive-1.2.1-bin hive`
- `sudo chown -R hduser:hadoop hive`

5.1 Die Einstellung von Apache Hive

Wenn ein Verzeichnis erstellt wird, wird es mit einer Variablen bereitgestellt, die auf das Hive-Verzeichnis verweist. Die HCAT_HOME-Variable zeigt, wo HCatalog ist, die Tabellen und die für die Speicherverwaltung verwendeten Komponenten. Danach wurde ".bashrc" im hduser Benutzerverzeichnis mit nano geöffnet und folgendes hinzugefügt:

- `# Hive`
- `export HIVE_HOME=/usr/local/hive`
- `export HCAT_HOME=$HIVE_HOME/hcatalog`
- `export PATH=$HIVE_HOME/bin:$PATH`

Darüber hinaus benötigt HDFS zwei Sequenzen. Wenn das Verzeichnis tmp nicht verfügbar ist, wurde es erstellt. Sie müssen sicherstellen, dass Hadoop gestartet wurde und der folgende Code eingegeben wird, um die Verzeichnisse für Hive in HDFS zu erstellen:

- `hdfs dfs -mkdir /tmp`
- `hdfs dfs -mkdir -p /user/hive/warehouse`
- `hdfs dfs -chmod g+w /tmp`
- `hdfs dfs -chmod g+w /user/hive/warehouse`

Wenn im ersten Befehl ein tmp-Verzeichnis vorhanden ist, wird die Warnung "mkdir: '/tmp': Datei existiert" angezeigt. Hier ändern wir die Zugriffsrechte auf die gesamte Gruppe mit dem Befehl `chmod`. Es ist nicht möglich, eine Tabelle in Hive mit den ohne diesen Ordner gegebenen Rechten zu erstellen.

Um den Hive zu konfigurieren, wurde die Datei `hive-default.xml.template` im Verzeichnis `hive/conf` mit den Namen `hive-site.xml` geändert. Dazu wurde folgender Befehl eingegeben:

```
hduser@ubuntu:~$ cd /usr/local/hive/conf
hduser@ubuntu:/usr/local/hive/conf$ cp hive-default.xml.template hive-site.xml
```

Abbildung 32: Ändern des Dateinamens

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Da Hive die Rolle des Basisclients von Hadoop gespielt hat, muss es nur auf einem Server installiert werden.

6. HIVEQL ALS ABFRAGESPRACHE

HiveQL und SQL sind sich sehr ähnlich und ist leicht zu erlernen. HiveQL ist keine relationale Datenbank. Select und Insert werden verwendet, enthalten jedoch nicht Update, Delete und Merge.

6.1 Die Datenbank-Operationen

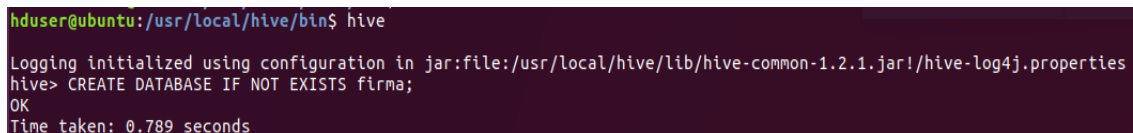
Diese Operationen können in fünf Abschnitte unterteilt werden.

6.1.1 Anlegen von Datenbanken

Das Erstellen einer Datenbank in Hive ist nicht schwierig. Dazu wurde der folgende Befehl eingegeben, indem die CLI (Command Line Interface) im bin-Verzeichnis von Hive aufgerufen wurde.

- hive

Wenn die Datenbank nicht existiert, wird der folgende Befehl eingegeben, um sie zu erstellen. Und das Ergebnis wurde angezeigt:

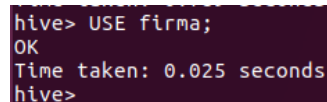


```
hduser@ubuntu:/usr/local/hive/bin$ hive
Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-1.2.1.jar!/hive-log4j.properties
hive> CREATE DATABASE IF NOT EXISTS firma;
OK
Time taken: 0.789 seconds
```

Abbildung 33: Erstellen einer Datenbank

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Der folgende Befehl wurde eingegeben, um die Ausführung der Datenbank zu benachrichtigen:

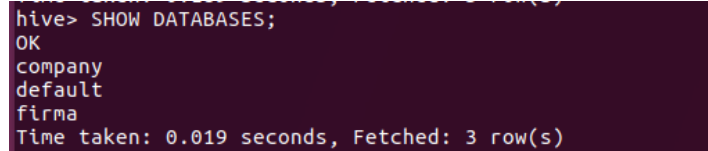


```
hive> USE firma;
OK
Time taken: 0.025 seconds
hive>
```

Abbildung 34: Verwendung des USE-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Der folgende Befehl wurde eingegeben, um sicherzustellen, dass Ihre Datenbank erstellt wurde:



```
hive> SHOW DATABASES;
OK
company
default
firma
Time taken: 0.019 seconds, Fetched: 3 row(s)
```

Abbildung 35: Anzeige der Datenbank

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Daher wurden drei Datenbanken mit den Namen company, default und Firma angesehen.

6.1.2 Importieren von Daten

Der Befehl LOAD wurde zum Importieren der Daten verwendet. Unsere Daten enthalten Informationen über Id, Name und Beruf. Eine Java-Anwendung wurde geschrieben, um diese Daten zu generieren. In dieser Anwendung wurden 30000 Daten generiert. Dieses Programm wurde mit Eclipse erstellt und in main.java wurde folgender Code geschrieben:

```
package test;

import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Main {

    private final static Logger log = Logger.getLogger(Main.class.getName());

    private static final int DATA_COUNTER = 30000;
```

```

private static final String jobs[] = {
    "Ogretmen", "Danisman", "Satici", "Bahcivan",
    "Sporcu", "MaliMusavir", "Muzisyen", "Cenazeci",
    "ArabaSatici", "Bankaci", "Bilisimci", "Ogrenci", "Rahip",
    "Polis", "HastaBakici", "Doktor", "SurucuOgretmeni",
    "Korsan", "BelediyeBaskani", "Politikaci", "Jeolog",
    "Profesor", "BilimAdami", "Cevirmen", "Garson", "Terzi",
    "Asci", "Güvenlik", "Aktris", "Sinemaci",
    "Ayakkabici", "Degirmenci", "Kasap"};

public static void main(String[] args) {
    log.log(Level.INFO, DATA_COUNTER + " veri yaziliyor...");
    try {
        Random r = new Random();
        Writer out = new BufferedWriter(new OutputStreamWriter(
            new FileOutputStream("hive_test_data.txt"), "UTF-8"));
        //PrintWriter out = new PrintWriter("hive_test_data.txt");

        for (int i = 0; i < DATA_COUNTER; i++) {
            out.write(i + "\t");

            // isimler
            String name = isimuretme.getRandomName();
            String lastName = isimuretme.getRandomLastName();
            out.write(name + " " + lastName + "\t");

            // gorev
            out.write(jobs[r.nextInt(jobs.length)]);
        }
    }
}

```

```

        // kelime ayirma

        if (i < DATA_COUNTER-1) {

            out.write(System.getProperty("line.separator"));

        }

    }

    out.close();

    log.log(Level.INFO, "Bitti!");

    } catch (Exception e) {

        log.log(Level.SEVERE, "yazmada hata!");

        e.printStackTrace();

    }

}

}

```

Die Klasse "*isimuretme.java*" wurde erstellt. Die folgenden Befehle wurden hier eingegeben:

```

package test;

import java.util.Random;

public class isimuretme {

    public static String getGender(String _name) {

        if (contains(_name, MALE_NAMES)) {

            return "male";

        }

        if (contains(_name, FEMALE_NAMES)) {

            return "female";

        }

    }

}

```

```

        return "unknown";
    }

    public static String getRandomName() {

        Random r = new Random();

        int gender = r.nextInt(2);

        if (gender == 0) {

            // Erkek

            int index = r.nextInt(MALE_NAMES.length);

            return MALE_NAMES[index];

        } else {

            // Kadin

            int index = r.nextInt(FEMALE_NAMES.length);

            return FEMALE_NAMES[index];

        }

    }

    public static String getRandomLastName() {

        Random r = new Random();

        int index = r.nextInt(LAST_NAMES.length);

        return LAST_NAMES[index];

    }

    private static boolean contains(String _str, String[] _list) {

        for (int i = 0; i < _list.length; i++) {

            if (_list[i].equalsIgnoreCase(_str)) {

                return true;

            }

        }

    }

```

```

    }

    return false;
}

public final static String[] LAST_NAMES = {"Abala", "Aga",
    "Akan", "Akar", "Alar", "Alim", "Alingan", "Alpman",
    "Ara", "Arpa", "Arsun", "Artuc", "Atalar", "Ayda", "Aydan",
    "Bakay", "Bakuy", "Bark", "Basgan", "Batak",
    "Bayar", "Baysal", "Beki", "Beril", "Cabalak", "Cici",
    "Cingil", "Dalan", "Dalbay", "Derin", "Devrim",
    "Dilim", "Dora", "Dursun", "El", "Elgin", "Erden",
    "Ger", "Gensu", "Mumcu", "Okuyan", "Park", "Tabu",
    "Taki", "Tala", "Tali", "Tang", "Tanla", "Tap", "Tar",
    "Taraka", "Tarban", "Tav", "Tedan", "Teker", "Tenik",
    "Tilki", "Toka", "Top", "Toy", "Tur", "Varol",
    "Yakuz", "Yanuk", "Yaray"};

public final static String[] MALE_NAMES = {
    "Mehmet", "Ahmet", "Ali", "Huseyin", "Veli",
    "Mustafa", "Sencer", "Oguz", "Semih", "Can",
    "Deniz", "Torehan", "Semir", "Serhan", "Halil", "Mustafa",
    "Ismail", "Selahattin", "Erdal", "Ibrahim", "Sarper", "Fevzi",
    "Suleyman", "Niyazi", "Eyup", "Adem", "Bilal", "Kemal",
    "Yuksel", "Cemal", "Cem", "Cemil", "Kaan", "Taner",
    "Tan", "Tekin", "Tumer", "Tahsin", "Selim", "Murat", "Bekir",
    "Salih", "Acun", "Akin", "Alp", "Alper", "Arda", "Ayhan",
    "Baran", "Bora", "Burhan", "Caner", "Ceyhun",

```

```

        "Celik", "Demir", "Emre", "Firat", "Furkan", "Hakan"
    };

    public final static String[] FEMALE_NAMES = {
        "Aydan", "Aynur", "Ayse", "Ayten", "Ajda",
        "Arzu", "Asena", "Ayben", "Ayda", "Benay",
        "Bergu", "Begum", "Beyza", "Birsen", "Buket",
        "Burcu", "Canan", "Candan", "Ceyda", "Ceylan", "Cagla",
        "Damla", "Defne", "Dilsah", "Duygu", "Demet", "Deniz",
        "Derya", "Didem", "Ebru", "Ece", "Emel", "Esen",
        "Ezgi", "Eda", "Elanur", "Elif", "Fatma", "Figen",
        "Fulya", "Funda", "Gamze", "Gizem", "Gokce",
        "Gulin", "Gulsun", "Hale", "Hande", "Irmak", "Idil",
        "Jale", "Kader", "Kevser", "Lale", "Leyla", "Melda",
        "Melike", "Merve", "Mine", "Naz", "Nazan", "Necla",
        "Nermin", "Nur", "Nuran", "Nurcan", "Oya", "Ozlem",
        "Pelin", "Sanem", "Seda", "Sema", "Serpil", "Suna"
    };
}

```

Wenn dieses Programm ausgeführt wird, werden 30000 Id, Name, Nachname und Beruf in hive_test_data.txt Datei geschrieben. Dann wurde das WinSCP-Programm geöffnet. Die generierte Datei "hive_test_data.txt" wurde in das Hive-Verzeichnis kopiert. Der folgende Befehl wurde eingegeben, um die Daten in Textdatei in die zuvor erstellte Tabelle "insan" zu übertragen:

```
hive> LOAD DATA LOCAL INPATH '/usr/local/hive/hive_test_data.txt' OVERWRITE INTO TABLE insan;
Loading data to table default.insan
Table default.insan stats: [numFiles=1, numRows=0, totalSize=788127, rawDataSize=0]
OK
Time taken: 1.016 seconds
```

Abbildung 36: Übertragen von Daten mit Hive

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Dann wurde eine Tabelle auf eine andere Tabelle übertragen. Dazu wurde zuerst eine Tabelle mit dem Namen "yoneticiler" erstellt.

```
hive> CREATE TABLE IF NOT EXISTS yoneticiler(yoneticiid INT, isim STRING);
OK
Time taken: 0.015 seconds
```

Abbildung 37: Erstellen der Tabelle "yoneticiler"

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Um die ID und die Namen aus der vorhandenen Tabelle "insan" in die Tabelle "yoneticiler" zu übertragen, wurde der folgende Befehl eingegeben. Das Ergebnis war wie folgt:

```
hive> INSERT INTO TABLE yoneticiler SELECT insanid as yoneticiid, isim as isim from insan;
Query ID = hduser_20171011014530_2cf1ddf7-23dc-4216-a31d-c71d322f2d88
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1507709534253_0004, Tracking URL = http://ubuntu:8088/proxy/application_1507709534253_0004/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1507709534253_0004
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 0
2017-10-11 01:45:37,587 Stage-1 map = 0%, reduce = 0%
2017-10-11 01:45:43,856 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.17 sec
MapReduce Total cumulative CPU time: 3 seconds 170 msec
Ended Job = job_1507709534253_0004
Stage-4 is filtered out by condition resolver.
Stage-3 is selected by condition resolver.
Stage-5 is filtered out by condition resolver.
Launching Job 3 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1507709534253_0005, Tracking URL = http://ubuntu:8088/proxy/application_1507709534253_0005/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1507709534253_0005
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2017-10-11 01:45:56,737 Stage-3 map = 0%, reduce = 0%
2017-10-11 01:46:01,974 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.05 sec
MapReduce Total cumulative CPU time: 1 seconds 50 msec
Ended Job = job_1507709534253_0005
Loading data to table default.yoneticiler
Table default.yoneticiler stats: [numFiles=1, numRows=30000, totalSize=180000, rawDataSize=150000]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2 Cumulative CPU: 3.17 sec HDFS Read: 794999 HDFS Write: 180127 SUCCESS
Stage-Stage-3: Map: 1 Cumulative CPU: 1.05 sec HDFS Read: 182125 HDFS Write: 180000 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 220 msec
OK
Time taken: 32.497 seconds
```

Abbildung 38: Übertragen Sie in die Tabelle "yoneticiler"

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Wie Sie sehen können, wurde dieser Vorgang in 32,497 Sekunden abgeschlossen. Um die Anzahl der übertragenen Daten zu sehen, wurde das Ergebnis mit dem folgenden Befehl angezeigt:

```
hive> SELECT COUNT(*) FROM yoneticiler;
Query ID = hduser_20171011014634_47cf676c-8745-45a5-8c8b-10c2c271573d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1507709534253_0006, Tracking URL = http://ubuntu:8088/proxy/application_1507709534253_0006/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1507709534253_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-10-11 01:46:41,659 Stage-1 map = 0%, reduce = 0%
2017-10-11 01:46:46,945 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.49 sec
2017-10-11 01:46:53,180 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.29 sec
MapReduce Total cumulative CPU time: 3 seconds 290 msec
Ended Job = job_1507709534253_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.29 sec HDFS Read: 186723 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 290 msec
OK
30000
Time taken: 19.656 seconds, Fetched: 1 row(s)
```

Abbildung 39: Bestätigung der Anzahl der Daten

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Es wurde hier gesehen, dass 30000 Einträge in die Tabelle "yoneticiler" übertragen wurden. Dies wurde bei 19,656 Sekunden durchgeführt.

6.1.3 Das SELECT-Statement

In Hive wurde jede Abfrage nicht für die Verwendung von Map-Reduce-Jobs ausgelegt. Um beispielsweise die ersten fünf Zeilen in der Tabelle "insan" mit dem folgenden Befehl anzuzeigen, wurde der folgende Befehl eingegeben:

```
hive> SELECT * FROM insan LIMIT 5;
OK
0      Birsen Taraka    Bahcivan
1      Baran Alpman   Ogretmen
2      Ismail Tenik   Politikaci
3      Burcu Tedan    Profesör
4      Elanur Aga     Bankaci
Time taken: 0.078 seconds, Fetched: 5 row(s)
```

Abbildung 40: SELECT-Befehl

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Es wurde festgestellt, dass das Ergebnis des Befehl "Select" weniger als 100 Millisekunden gedauert wurde. Das Zeichen * im SQL-Befehl bezieht sich auf alle Zeilen in der Tabelle. Die LIMIT-Anweisung gibt hier an, wie viele Zeilen angezeigt werden sollen.

Das AS-Schlüsselwort im SELECT-Befehl teilt uns in der Abfrage mit, welcher Name in der nächsten Abfrage verwendet werden soll. Dies gibt uns Bequemlichkeit in komplexen Abfragen. Die Syntax ist wie folgt.

```
SELECT isim FROM insan AS ins;
```

In diesem Befehl wurde das Wort "ins" anstelle des Namens der "insan" Tabelle verwendet. Außerdem wurden alle Namen in der Tabelle "insan" auf dem Bildschirm angezeigt.

Einer der am häufigsten verwendeten SQL-Befehle ist der WHERE-Befehl. Dieser Befehl gibt die Ergebnisse gemäß den gesuchten Kriterien an. Daher wird eine Einschränkung für die Abfrage eingeführt. Als ein Beispiel und ein Ergebnis wird der folgende Befehl eingegeben:

```
hive> SELECT * FROM insan WHERE gorev='Ogretmen' LIMIT 10;
OK
1      Baran Alpman      Ogretmen
26     Irmak Tur        Ogretmen
62     Funda Yakuz      Ogretmen
68     Kader Arsun      Ogretmen
76     Bora Tap         Ogretmen
81     Ayhan Teker      Ogretmen
91     Tan Beril        Ogretmen
145    Didem Akar       Ogretmen
216    Ceyda Ara        Ogretmen
248    Aynur Tabu       Ogretmen
Time taken: 0.087 seconds, Fetched: 10 row(s)
```

Abbildung 41: Verwendung des WHERE-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Hier wurden die ersten 10 Datensätze in der Tabelle "insan", die der Lehrer ist, auf den Bildschirm angezeigt.

Um Ergebnisse unter mehr als einer Bedingung zu erhalten, wurde die Bedingung neben dem WHERE-Befehl mit "AND" verbunden. Die Verwendung ist wie folgt:

```

> SELECT * FROM insan WHERE gorev='Ogretmen' AND insanid<200;
OK
1      Baran Alpman      Ogretmen
26     Irmak Tur        Ogretmen
62     Funda Yakuz      Ogretmen
68     Kader Arsun      Ogretmen
76     Bora Tap         Ogretmen
81     Ayhan Teker      Ogretmen
91     Tan Beril        Ogretmen
145    Didem Akar       Ogretmen
Time taken: 0.117 seconds, Fetched: 8 row(s)
hive>

```

Abbildung 42: Verwendung des AND-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Die Datensätze in der Tabelle "insan", die hier ein Lehrer ist und weniger als 200 ID, wurden auf den Bildschirm angezeigt. Anstelle von AND können andere logische Operatoren eingegeben werden. Andere logische Operatoren sind wie die folgende Tabelle.

Logischer Operator	Beschreibung
<i>a AND b</i>	<i>TRUE</i> , wenn a und b wahr sind. Andernfalls <i>FALSE</i> .
<i>a OR b</i>	<i>TRUE</i> , falls entweder a oder b wahr ist. Wenn keine Bedingung wahr ist, dann <i>FALSE</i> .
<i>NOT a</i>	<i>TRUE</i> , wenn a <i>FALSE</i> ist. Sonst <i>FALSE</i> . Alternativ kann <i>!a</i> verwendet werden.
<i>a IN (Wert1, Wert2...)</i>	<i>TRUE</i> , wenn a einen der Werte in der Liste angenommen hat. Ab Hive 0.13.0 dürfen in der Liste auch Subqueries (z. B. ein <i>SELECT</i> auf einen weiteren Wertebereich) verwendet werden.
<i>a NOT IN (Wert1, Wert2...)</i>	<i>TRUE</i> , wenn a keinem der Werte entspricht.
<i>EXISTS (Subquery)</i>	<i>TRUE</i> , falls die Subquery mindestens eine Zeile zurückliefert.

Abbildung 43: Logischer Operatoren

Quelle : Freiknecht, J., BIG DATA in der Praxis, Germany, 2014.

Neben logischen Operatoren können relationale Operatoren wie “=” und “<” verwendet werden.

Relationaler Operator	Beschreibung
$a = b$	<i>TRUE</i> , falls a gleich b ist. Andernfalls <i>FALSE</i> .
$a <=> b$	Wie $=$, nur dass wahr zurückgegeben wird, wenn a und b <i>NULL</i> sind. Ist a oder b <i>NULL</i> , wird <i>FALSE</i> zurückgegeben.
$a <> b$	<i>TRUE</i> , falls a ungleich b ist. <i>NULL</i> , wenn a oder b <i>NULL</i> ist. Alternativ kann auch $\neq$ verwendet werden.
$a < b$	<i>TRUE</i> , falls a kleiner als b ist. <i>FALSE</i> , falls b kleiner oder gleich a ist. Ist a oder b <i>NULL</i> , wird ein <i>NULL</i> zurückgegeben.
$a <= b$	<i>TRUE</i> , falls a kleiner oder gleich b ist. <i>FALSE</i> , falls b kleiner a ist. <i>NULL</i> , falls einer der beiden Werte <i>NULL</i> ist.
$a > b$	<i>TRUE</i> , falls b kleiner a ist. <i>FALSE</i> , falls a kleiner oder gleich b ist. Ist b oder a <i>NULL</i> , wird ein <i>NULL</i> zurückgegeben.
$a >= b$	<i>TRUE</i> , falls b kleiner oder gleich a ist. <i>FALSE</i> , falls a kleiner b ist. <i>NULL</i> , falls a oder b <i>NULL</i> ist.
$a \text{ [NOT] BETWEEN } b \text{ AND } c$	<i>NULL</i> , falls a, b oder c <i>NULL</i> ist. <i>TRUE</i> , falls a größer oder gleich b und kleiner oder gleich c ist. Andernfalls <i>FALSE</i> .

Abbildung 44: Relationalen Operatoren

Quelle : Freiknecht, J., BIG DATA in der Praxis, Germany, 2014.

LIKE wird verwendet, um Werte zu finden, die bestimmte Zeichen in String-Typen enthalten. Wenn zum Beispiel a LIKE b ist, werden die folgenden Tabellenergebnisse erhalten.

a	b	Ergebnis
hadoop	had	FALSE
hadoop	hadoop	TRUE
hadoop	had _ _ _	TRUE
hadoop	ha*	TRUE

Abbildung 45: Verwendung des LIKE-Befehls

Quelle : Freiknecht, J., BIG DATA in der Praxis, Germany, 2014.

6.1.4 Beschränken von SELECT über DISTINCT

Das Schlüsselwort DISTINCT verhindert, dass die Datensätze zweimal in die Abfrage geschrieben werden. Zum Beispiel suchen wir nach "Serhan Tur" in unserer Datenbank. Dieser Datensatz kann mehr als einmal gefunden werden. Um doppelte Einträge zu vermeiden, wurde folgender Befehl eingegeben und betrachtet:

```

hive> SELECT DISTINCT isim FROM insan WHERE isim='Serhan Tur';
Query ID = hduser_20171011074524_5f0369a1-45ab-469a-ad7d-9edee7e78030
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1507709534253_0010, Tracking URL = http://ubuntu:8088/proxy/application_1507709534253_0010/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1507709534253_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-10-11 07:45:30,454 Stage-1 map = 0%, reduce = 0%
2017-10-11 07:45:35,647 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.32 sec
2017-10-11 07:45:40,817 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.02 sec
MapReduce Total cumulative CPU time: 4 seconds 20 msec
Ended Job = job_1507709534253_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.02 sec HDFS Read: 795409 HDFS Write: 11 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 20 msec
OK
Serhan Tur
Time taken: 18.513 seconds, Fetched: 1 row(s)

```

Abbildung 46: Verwendung des DISTINCT-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Nur ein Datensatz wurde hier erhalten. Wenn die nach SELECT angegebene Spalte als "gorev" geändert wird, kann es mehr als ein sein.

6.1.5 SELECT auf partitionierte Tabellen

Die Ausgabe des SELECT-Befehls kann nach SORT BY und ORDER BY sortiert werden. Der folgende Befehl wurde eingegeben:

```

hive> SELECT * FROM insan ORDER BY insanid DESC LIMIT 5;
Query ID = hduser_20171011074803_4092bd48-c6d0-45cd-98b3-a2fdc41cfd39
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1507709534253_0011, Tracking URL = http://ubuntu:8088/proxy/application_1507709534253_0011/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1507709534253_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-10-11 07:48:09,437 Stage-1 map = 0%, reduce = 0%
2017-10-11 07:48:14,619 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.05 sec
2017-10-11 07:48:20,863 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.15 sec
MapReduce Total cumulative CPU time: 5 seconds 150 msec
Ended Job = job_1507709534253_0011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.15 sec HDFS Read: 794465 HDFS Write: 146 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 150 msec
OK
29999 Nazan Baysal Bahcivan
29998 Burhan Ara Bahcivan
29997 Selim Artuc Danisman
29996 Beyza Basgan MaliMusavir
29995 Fevzi Artuc BelediyeBaskani
Time taken: 18.891 seconds, Fetched: 5 row(s)

```

Abbildung 47: Verwendung des ORDER BY-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Der Teil, der dem Befehl ORDER BY folgt, ist der Teil der Sequenz. Es beginnt mit dem DESC-Befehl, der höchsten insanid-Nummer der Sequenz, und schreibt die ersten 5 Datensätze auf den Bildschirm. Es kann optional in ASC verwendet werden. Es beginnt mit der niedrigsten insanid und schreibt auf den Bildschirm. Ein relevantes Beispiel wurde gemacht.

```
> SELECT * FROM insan ORDER BY isim ASC, gorev ASC LIMIT 5;
Query ID = hduser_20171011090421_f4e496b7-d17d-46a6-a1d6-11a31b154f08
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1507709534253_0012, Tracking URL = http://ubuntu:8088/proxy/application_1507709534253_0012/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1507709534253_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-10-11 09:04:27,032 Stage-1 map = 0%, reduce = 0%
2017-10-11 09:04:32,341 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.37 sec
2017-10-11 09:04:37,500 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.34 sec
MapReduce Total cumulative CPU time: 4 seconds 340 msec
Ended Job = job_1507709534253_0012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.34 sec HDFS Read: 794536 HDFS Write: 133 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 340 msec
OK
22263 Acun Abala Cenazeci
17771 Acun Abala Degirmenci
21955 Acun Abala Doktor
23386 Acun Abala HastaBakici
14180 Acun Aga Ayakkabici
Time taken: 18.288 seconds, Fetched: 5 row(s)
```

Abbildung 48: Verwendung des ASC-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Hier wurde das erste Sortieren nach dem Namen gemacht und dies wurde vom kleinsten angefangen. Wenn es mehrere Namen gibt, erfolgt die zweite Sortierung nach "gorev". Dies wurde beginnend mit dem Buchstaben "A" gemacht.

Zum Sortieren wird SORT BY anstelle von ORDER BY verwendet. Was ist der Unterschied zwischen beiden? ORDER BY wird nur von einem Reducer implementiert und ausgegeben. Und die Reihenfolge nach dem Wert von LIMIT angewendet. Wenn der Befehl SORT BY benötigt wird, werden mehrere Reducer angewendet.

Es kann auch DISTRIBUTE BY und SORT BY in der Verwendung von SORT BY und ORDER BY enthalten. Jeder Reducer erzeugt einen nicht überlappenden Datensatz und wird im Reducer sortiert. Das Ergebnis ist eine fehlerfreie Ausgabe. CLUSTER BY kann mit einem verknüpften DISTRIBUTE BY und einem SORT BY verglichen werden. DISTRIBUTE

BY garantiert, dass jeder 'Reducer' einen nicht konfligierenden Datensatz erhält, aber nicht sortiert.

6.2 Die Datentypen und Views

Die Datentypen können in zwei Titeln angegeben werden. Das sind Primitive und Komplexe-Datentypen.

6.2.1 Primitive Datentypen

Einfache Datentypen werden in der folgenden Liste angezeigt.

BIGINT, INT, SMALLINT, TINYINT: Verschiedene Integer-Typen haben unterschiedliche Größen. TINYINT ist zwischen -128 und 127, SMALLINT zwischen -32768 und 32767, INT zwischen -2147483648 und 2147483647, BIGINT zwischen -9 223 372 036 854 775 808 und 9 223 372 036 854 775 807.

BOOLEAN : Es nimmt FALSE und TRUE Werte.

FLOAT : Es ist 4 Bytes groß und ist eine einzelne Ziffer nach dem Komma.

DOUBLE : Es ist 8 Bytes groß und es sind zwei Ziffern nach dem Komma.

DECIMAL : Es sind 38 Ziffern nach dem Komma.

STRING : Es kann in einzelne oder doppelte Gänsefüßchen geschrieben werden.

TIMESTAMP: Es ist Standard-Unix-Timestamp in Form von den Integer, Float oder YYYY-MM-DD HH:MM:SS.

BINARY : Es besteht aus Null und Eins.

VARCHAR : Es wird mit einer bestimmten Länge erstellt.

CHAR : Wenn ein VARCHAR eine konstante Länge hat, wird es erstellt. Nicht verwendete Zeichen werden durch Leerzeichen ersetzt.

DATE : Es wird durch die Kombination aus Jahr, Monat und Tag im Format "JJJJ-MM-TT" definiert.

6.2.2 Komplexe Datentypen

Die komplexen Datentypen wurden als UNIONTYPE nach Hive 0.7.0 entwickelt.

ARRAY: Zum Beispiel Array Hava['Yagmurlu','Gunesli','Sisli']. Wenn wir in unserer Serie Hava[0] aufrufen, erhalten wir einen "Yagmurlu" Wert.

MAP: Es ist in Form eines Schlüssel-Wert-Paares. Ein Wert von Satirismi['Element'] kann gefragt werden.

STRUCT: Es wird als STRUCT(isim STRING, yas INT) in einem Struct namens "Cocuk" erstellt. In der Form "Cocuk.isim" ist der Wert erreicht.

UNIONTYPE: Es funktioniert wie Union in C. In einem Bereich kann verschiedenen Datentypen adressiert werden. Zum Beispiel kann "UNIONTYPE<INT, DOUBLE, STRING>" für Alter den Wert 20, 20.0 oder 'Zwanzig' annehmen.

6.2.3 Erstellen von Views

Eine View beschreibt das Aussehen einer Tabelle. Es gibt keine physischen Daten wie die Tabelle. Sie müssen nicht darauf achten, dass eine View mit einer bereits erstellten View mit dem gleichen Namen erstellt wird. Eine View, die alle Schüler in der "insan" Tabelle zeigt, wurde wie folgt erstellt.

```
hive> CREATE VIEW ogrenciler(ogrenciid, isim) COMMENT "insan tablosundaki öğrenciler" AS SELECT  
iid, isim FROM insan WHERE gorev="Ogrenci";  
OK  
Time taken: 0.168 seconds
```

Abbildung 49: Verwendung des VIEW-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Mit diesem Befehl wurde eine View mit dem Namen "ogrenciler" erstellt. Die Datentypen der Spalten werden automatisch wie in der Tabelle "insan" angegeben. Die ersten

zehn Elemente der generierten View "ogrenciler" wurden auf dem Konsolenbildschirm angezeigt. Dazu wurde folgender Befehl eingegeben:

```
hive> SELECT * FROM ogrenciler LIMIT 10;  
OK  
66      Emre Bark  
95      Defne Ger  
119     Aynur Erden  
125     Alper Toka  
142     Canan Beki  
163     Bora Tanla  
169     Elif Okuyan  
170     Kemal Alim  
172     Dilsah Yaray  
218     Akin Yaray  
Time taken: 0.109 seconds, Fetched: 10 row(s)
```

Abbildung 50: Anzeigen mit View erstellten Tabelle

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

6.2.4 Löschen einer View

Ein View wie eine Tabelle wird mit dem Befehl DROP gelöscht. Mit dem folgenden Befehl wurde ein View mit dem Namen "ogrenci" gelöscht.

```
hive> DROP VIEW IF EXISTS ogrenci;  
OK  
Time taken: 0.013 seconds
```

Abbildung 51: Löschen eines View

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Achten Sie jedoch darauf, kein View zu löschen, die sich auf ein anderes View beziehen.

6.3 Die Tabelle-Operationen

Diese Operationen können in vier Abschnitte unterteilt werden.

6.3.1 Anlegen von Tabellen

Der folgende Befehl wurde eingegeben, um eine Tabelle zu erstellen:

```
hive> CREATE TABLE insan(insanid INT, isim STRING, gorev STRING);
OK
Time taken: 0.382 seconds
```

Abbildung 52: Erstellen einer Tabelle mit Hive

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Hier haben wir eine Tabelle mit dem Namen *insan* erstellt. Die Elemente dieser Tabelle sind *insanid*, *isim* und *gorev*. Wenn bereits eine Tabelle mit dem gleichen Namen existiert, gibt es eine Exception. Um die Kontrolle zu gewährleisten, können wir die Fehlermeldung nicht erhalten, indem wir "IF NOT EXISTS" zwischen das Schlüsselwort TABLE und den Namen der Tabelle setzen.

```
CREATE TABLE IF NOT EXISTS insan(insanid INT, isim STRING, gorev
STRING);
```

Der folgende Befehl wurde eingegeben, um die Tabellen in der Datenbank aufzulisten, und die Tabelle "man" wird als Ausgabe aufgelistet.

```
hive> SHOW TABLES;
OK
insan
Time taken: 0.036 seconds, Fetched: 1 row(s)
```

Abbildung 53: Auflisten der Tabellen

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Wenn schon keine andere Tabelle erstellt wurde, wird nur die Tabelle *insan* erhalten. Hive gibt nur die Tabellen der ausgewählten Datenbank.

Hive speichert eine Tabellendatei in HDFS. Andernfalls können Abfragen über Mapreduce nicht verarbeitet werden. Ein Beispiel für die Definition einer Tabelle von Zeilenformaten ist wie folgt.

```
CREATE TABLE insan(insanid INT, isim STRING, gorev STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n';
```

Mit diesem Befehl werden alle Felder in einer Spalte durch die Verwendung von '\t' getrennt. Und es ist durch "newline" getrennt wegen der Verwendung von '\n' in allen Zeilen.

Jetzt wurde die aktuelle Tabelle mit dem LIKE-Befehl kopiert.

```
hive> CREATE TABLE insan2 LIKE insan;  
OK  
Time taken: 0.104 seconds
```

Abbildung 54: Verwendung des LIKE-Befehl

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

6.3.2 Partitionierung von Tabellen

Zur Verbesserung der Abfrageleistung werden die Tabellen als partitionierte Tabellen bezeichnet, die durch einen oder mehrere Abschnitte der Daten getrennt sind. Wenn beispielsweise eine ismanager-Partitionsspalte für die Benutzertabelle mit den Werten 1 oder 0 erstellt wird, werden zwei Dateien für die Tabellen im Dateisystem erstellt. Der erste enthält die Aufzeichnungen des Geschäftsführer und der zweite die menschlichen Aufzeichnungen. Für jeden Wert in der Partitionsspalte eines Eintrags wird immer ein Ordner erstellt. So können Hunderte von separaten Dateien erstellt werden. Eine Beispieltabelle wurde erstellt:

```
CREATE TABLE universiteler(id INT, ulke STRING, isim STRING, yas INT);
```

Hier wurde die Spalte "ulke" ausgewählt und die Tabelle ist wie folgt erstellt:

```
CREATE TABLE universitelerim(id INT, isim STRING, yas INT) PARTITIONED  
BY (ulke STRING);
```

Mit dem letzten Befehl wird die Tabelle "universitelerim" nach den Länderdaten gruppiert und eine neue Tabelle erstellt.

6.3.3 Löschen, Leeren von Tabellen und count Befehle

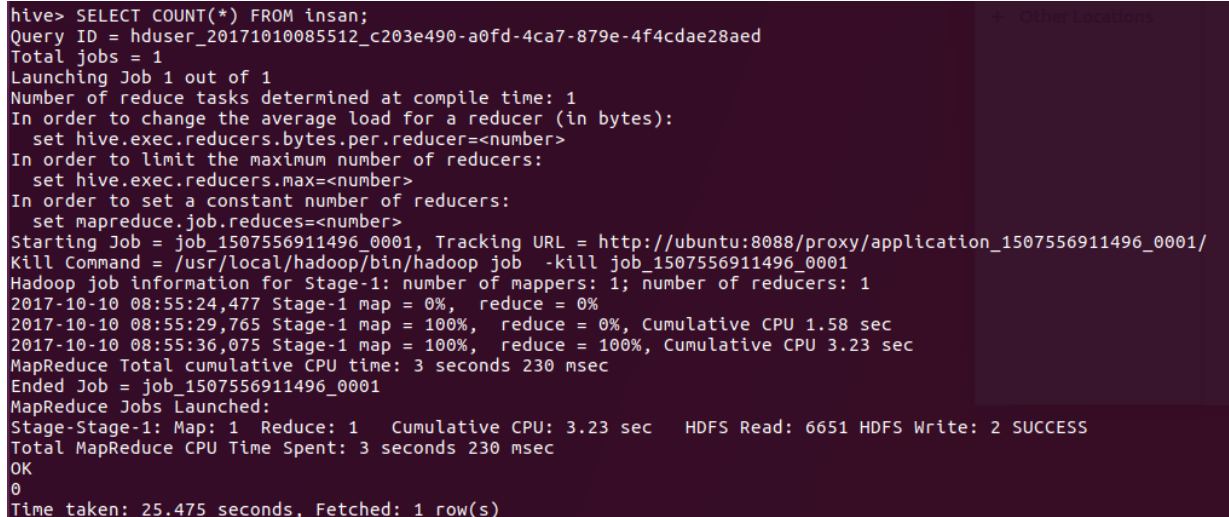
Der folgende Befehl wurde eingegeben, um die Daten in einer Tabelle zu leeren:

```
TRUNCATE TABLE insan;
```

Der folgende Befehl wurde auch zum Löschen einer Tabelle verwendet:

```
DROP TABLE insan;
```

Der COUNT() Befehl wurde verwendet, um die Anzahl der Zeilen in einer Tabelle herauszufinden, und die Ausgabe war wie folgt.



```
hive> SELECT COUNT(*) FROM insan;
Query ID = hduser_20171010085512_c203e490-a0fd-4ca7-879e-4f4cdade28aed
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1507556911496_0001, Tracking URL = http://ubuntu:8088/proxy/application_1507556911496_0001/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1507556911496_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-10-10 08:55:24,477 Stage-1 map = 0%, reduce = 0%
2017-10-10 08:55:29,765 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.58 sec
2017-10-10 08:55:36,075 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.23 sec
MapReduce Total cumulative CPU time: 3 seconds 230 msec
Ended Job = job_1507556911496_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.23 sec HDFS Read: 6651 HDFS Write: 2 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 230 msec
OK
0
Time taken: 25.475 seconds, Fetched: 1 row(s)
```

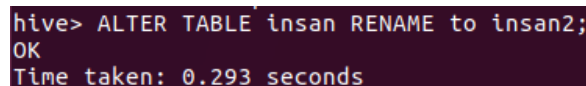
Abbildung 55: Verwendung des COUNT-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Hier wird zuerst die notwendige Map-Reduction-Operation durchgeführt. Da in dieser Tabelle keine Datensätze vorhanden sind, wurde der Wert 0 angezeigt.

6.3.4 Verändern von Tabellen

Mit dem Schlüsselwort ALTER kann der Name der gewünschten Tabelle in "Hive" geändert werden. Dafür müssen Sie die Tabelle wie in HBASE nicht deaktivieren. Die Tabelle "insan" wurde wie folgt in "insan2" geändert.



```
hive> ALTER TABLE insan RENAME to insan2;
OK
Time taken: 0.293 seconds
```

Abbildung 56: Verwendung des ALTER-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Die Tabelleneigenschaft kann auch geändert werden. Um der Tabelle eine Definition hinzuzufügen, ändern Sie die COMMENT-Eigenschaft wie folgt.

```
hive> ALTER TABLE insan SET TBLPROPERTIES ('COMMENT' = 'Firmadaki insanlar');
OK
Time taken: 0.101 seconds
```

Abbildung 57: Ändern der Tabelleneigenschaften

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Um die erweiterten Tabelleneigenschaften anzuzeigen, wurde der folgende Befehl eingegeben.

```
hive> DESCRIBE EXTENDED insan;
OK
insanid          int
isim             string
gorev            string

Detailed Table Information      Table(tableName:insan, dbName:default, owner:hduser, createTime:1507714377, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:insanid, type:int, comment:null), FieldSchema(name:isim, type:string, comment:null), FieldSchema(name:gorev, type:string, comment:null)], location:hdfs://localhost:9000/user/hive/warehouse/insan, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1, serdeInfo:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{line.delim=, field.delim=, serialization.format=}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), storedAsSubDirectories:false), partitionKeys:[], parameters:{last_modified_time=1508256705, totalSize=788127, numRows=-1, rawDataSize=-1, COLUMN_STATS_ACCURATE=false, numFiles=1, transient_lastDdlTime=1508256705, last_modified_by=hduser, COMMENT=Firmadaki insanlar}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE)
Time taken: 0.058 seconds, Fetched: 6 row(s)
```

Abbildung 58: Detaillierte Eigenschaften der Tabelle

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

6.4. Die Gruppierung-Operationen

Diese Operationen können in fünf Abschnitte unterteilt werden.

6.4.1 Gruppieren von Daten mittels GROUP BY

GROUP BY wird verwendet, um die Ergebnismenge zu gruppieren. Um nach verschiedenen Aufgabengruppen zu sortieren, wurde der folgende Befehl eingegeben:

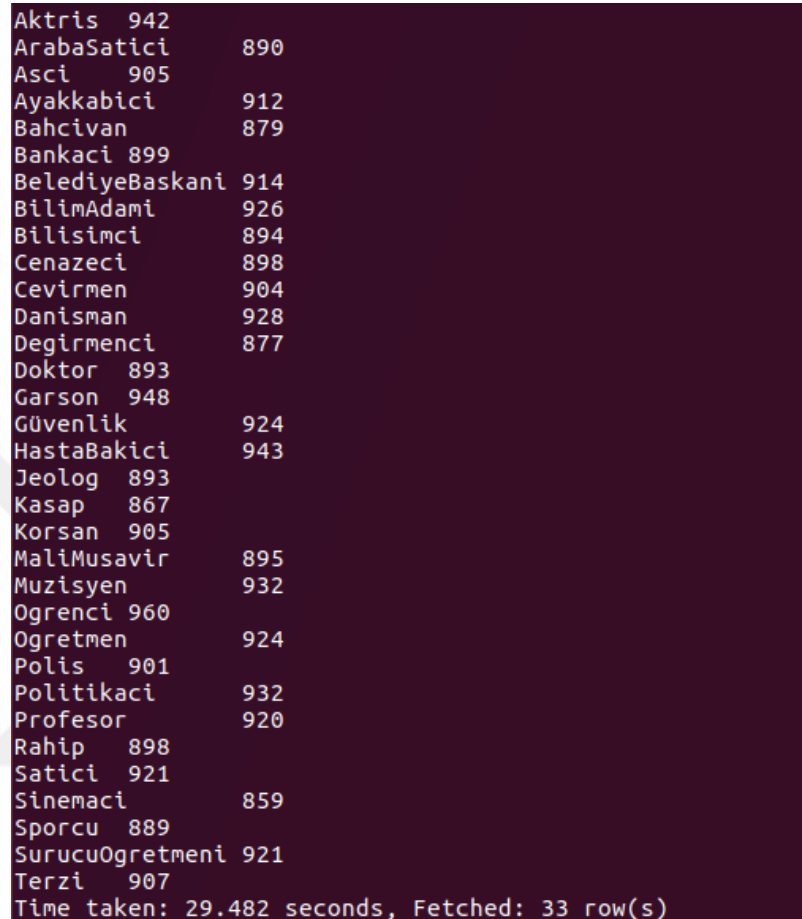
```
hive> SELECT gorev, count(DISTINCT insanid) FROM insan GROUP BY gorev;
```

Abbildung 59: Verwendung des GROUP BY-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Mit diesem Befehl wird nur gorev und insanid auf den Bildschirm angezeigt. Mit dem Befehl count (DISTINCT man) wurde zunächst einmal von jedem Datensatz auf dem

Bildschirm angezeigt. Schließlich wurde berichtet, dass die Gruppierung gemäß der Spalte "gorev" durchgeführt wurde. Bildschirmausgabe ist wie folgt:

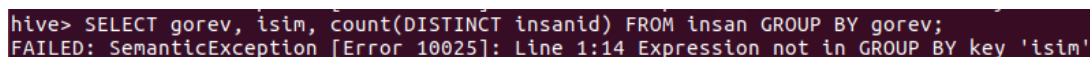


```
Aktris 942
ArabaSatici 890
Asci 905
Ayakkabici 912
Bahcivan 879
Bankaci 899
BelediyeBaskani 914
BilimAdami 926
Bilisimci 894
Cenazeci 898
Cevirmen 904
Danisman 928
Degirmenci 877
Doktor 893
Garson 948
Güvenlik 924
HastaBakici 943
Jeolog 893
Kasap 867
Korsan 905
MaliMusavir 895
Muzisyen 932
Ogrenci 960
Ogretmen 924
Polis 901
Politikaci 932
Profesor 920
Rahip 898
Satici 921
Sinemaci 859
Sporcu 889
SurucuOgretmeni 921
Terzi 907
Time taken: 29.482 seconds, Fetched: 33 row(s)
```

Abbildung 60: Nutzungsergebnis des Befehls "GROUP BY"

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Wenn in GROUP BY im SELECT-Abschnitt keine Spalten vorhanden sind, wird eine Fehlermeldung angezeigt. Das Beispiel und die Fehlermeldung wie folgt:



```
hive> SELECT gorev, isim, count(DISTINCT insanid) FROM insan GROUP BY gorev;
FAILED: SemanticException [Error 10025]: Line 1:14 Expression not in GROUP BY key 'isim'
```

Abbildung 61: Fehlermeldung mit der GROUP BY-Befehl

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Es ist ein Fehler aufgetreten, weil die Spalte "isim" nicht in GROUP BY gefunden wurde.

6.4.2 Subqueries verschachtelte Abfragen

Subqueries sind eine Abfrage, die in einer anderen Abfrage eingebettet ist. Nach dem Befehl "FROM" wird der Unterabfrageteil erstellt.

```
hive> SELECT tanımlama FROM (SELECT concat(isim, " ", gorev, " olarak calisiyor" ) AS tanımlama FROM insan) AS tuminsan LIMIT 20;
OK
Birsen Taraka Bahcivan olarak calisiyor
Baran Alpman Ogretmen olarak calisiyor
Ismail Tenik Politikaci olarak calisiyor
Burcu Tedan Profesör olarak calisiyor
Elanur Aga Bankaci olarak calisiyor
Akin Dursun Terzi olarak calisiyor
Mustafa Toka SurucuOgretmeni olarak calisiyor
Hakan Tilki Sinemaci olarak calisiyor
Ece Tilki BilimAdami olarak calisiyor
Tahsin Bakay Bilisimci olarak calisiyor
Sencer Arsun Jeolog olarak calisiyor
Canan Dalbay Sinemaci olarak calisiyor
Ibrahim Varol Korsan olarak calisiyor
Fatma Yanuk Garson olarak calisiyor
Salih Top Garson olarak calisiyor
Hakan Bakay Güvenlik olarak calisiyor
Beyza Yaray Güvenlik olarak calisiyor
Ceyhun Toka Cenazeci olarak calisiyor
Semih Dilim Güvenlik olarak calisiyor
Kaan Varol Korsan olarak calisiyor
Time taken: 0.08 seconds, Fetched: 20 row(s)
```

Abbildung 62: Erstellung eines verschachtelte Abfragen

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Innerhalb der FROM fragen wir nach einem Objekt namens "tanımlama" der Unterabfrage. Die als Funktion verwendete concat-Funktion kombiniert die Spalte "isim", einen Text und die Spalte "gorev". Die später erstellte Unterabfrage wurde als "tuminsan" genannt. Als Ergebnis wurden 20 Werte auf dem Bildschirm angezeigt und der Text, den wir mit den Spalten angegeben hatten, wurde kombiniert.

6.4.3 Ergebnismengen vereinigen mit UNION ALL

UNION wird verwendet, um viele SELECT-Ergebnismengen zu verbinden. Die Hive 0.13.1-Version führt durch, indem doppelte Ergebnisse mit UNION ALL nicht zurückgegeben werden. Als Beispiel wurde der folgende Befehl angegeben:

```
SELECT * FROM kitaplar UNION ALL SELECT * FROM videolar;
```

Hier wurden alle Bücher und alle Videos in einer Liste angezeigt.

6.4.4 HAVING

Da Aggregatfunktionen in der WHERE-Bedingung nicht verwendet werden können, wird HAVING dann in SQL angewendet. HAVING wird immer hinter dem GROUP BY-Befehl angewendet. Wenn Sie gegen diese Regel verstoßen, wird HIVE Sie darüber informieren. Aggregatfunktionen können ohne GROUP BY angewendet werden. Das Beispiel GROUP BY wurde zurückgegeben und die Berufe mit weniger als 1000 Mitarbeitern sind wie folgt begrenzt.

```
hive> SELECT gorev,count(1) FROM insan GROUP BY gorev HAVING count(gorev)<1000;
Query ID = hduser_20171017075037_b88cce9a-2177-4a29-bb21-72f2dabc431c
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1508249443792_0010, Tracking URL = http://ubuntu:8088/proxy/application_1508249443792_0010/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1508249443792_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-10-17 07:50:45,227 Stage-1 map = 0%, reduce = 0%
2017-10-17 07:50:50,477 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.7 sec
2017-10-17 07:50:55,697 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.96 sec
MapReduce Total cumulative CPU time: 3 seconds 960 msec
Ended Job = job_1508249443792_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.96 sec HDFS Read: 796346 HDFS Write: 432 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 960 msec
OK
Aktris 942
ArabaSatıcı 890
Asci 905
Ayakkabici 912
Bahçıvan 879
Bankacı 899
BelediyeBaskanı 914
BilimAdami 926
Bilisimci 894
Cenazeci 898
Cevirmen 904
Danisman 928
Degirmenci 877
Doktor 893
Garson 948
Güvenlik 924
HastaBakici 943
Jeolog 893
Kasap 867
Korsan 905
MaliMusavir 895
Muzisyen 932
Ogrenci 960
Ogretmen 924
Polis 901
Politikaci 932
Profesor 920
Rahip 898
Satıcı 921
Sinemaci 859
Sporcu 889
SurucuOgretmeni 921
Terzi 907
Time taken: 19.905 seconds, Fetched: 33 row(s)
```

Abbildung 63: Verwendung des HAVING-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

6.4.5 Tabellen zusammenführen mit JOINS

JOIN führt mehrere Tabellen zusammen. “ON” in der Syntax sagt uns, welche Spalten wir zusammenführen werden. Der folgende Befehl wird eingegeben und die Ergebnisse werden angezeigt:

```
hive> SELECT ogrenciler.ogrenciid, insan.isim FROM insan JOIN ogrenciler ON (ogrenciler.ogrenciid = insan.iid) LIMIT 20;
Query ID = hduser_20171027060059_1ebae22f-2f45-4d77-8f1c-81999e9dc4f0
Total jobs = 1
Execution log at: /tmp/hduser/hduser_20171027060059_1ebae22f-2f45-4d77-8f1c-81999e9dc4f0.log
2017-10-27 06:01:03 Starting to launch local task to process map join; maximum memory = 477626368
2017-10-27 06:01:04 Dump the side-table for tag: 0 with group count: 30000 into file: file:/tmp/hduser/df6869ed-2cf3-4b63-b5b0-9b103e704796/hive_2017-10-27_06-00-59_575_4788478370057134543-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile00-
-.hashtable
2017-10-27 06:01:04 Uploaded 1 File to: file:/tmp/hduser/df6869ed-2cf3-4b63-b5b0-9b103e704796/hive_2017-10-27_06-00-59_575_4788478370057134543-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile00-
-.hashtable (982726 bytes)
2017-10-27 06:01:04 End of local task; Time Taken: 1.271 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1509103583096_0001, Tracking URL = http://ubuntu:8088/proxy/application_1509103583096_0001/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1509103583096_0001
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2017-10-27 06:01:14,714 Stage-3 map = 0%, reduce = 0%
2017-10-27 06:01:21,090 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 2.85 sec
MapReduce Total cumulative CPU time: 2 seconds 850 msec
Ended Job = job_1509103583096_0001
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1 Cumulative CPU: 2.85 sec HDFS Read: 22397 HDFS Write: 303 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 850 msec
OK
66 Enre Bark
95 Defne Ger
119 Aynur Erden
125 Alper Toka
142 Canan Beki
163 Bora Tanla
169 Elif Okuyan
170 Kemal Alin
172 Dilsah Yaray
218 Akin Yaray
256 Hande Cabalak
353 Tan Beril
372 Necila Ara
391 Damla Alpman
431 Serpil Aga
462 Erdal Tar
467 Pelin Bakay
477 Bora Tap
507 Candan Derin
527 Selim Akan
Time taken: 23.682 seconds, Fetched: 20 row(s)
```

Abbildung 64: Verwendung des JOIN-Befehls

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Mit diesem Befehl wird berichtet, welche Spalten nach SELECT stehen. Nach “ON” wurde berichtet, unter welchen Bedingungen und die ersten 20 Daten angezeigt werden. Wenn Sie mehrere Tabellen kombinieren möchten, werden die Ausdrücke JOIN und ON für die anderen Tabellen erneut geschrieben. Darüber hinaus sind LINKS, RECHTS und FULL OUTER JOIN verfügbar.

6.5 Die Funktionen

Diese Funktionen können in drei Abschnitte unterteilt werden.

6.5.1 Mathematische Funktionen

Es gibt mathematische Funktionen im Hive, wie in anderen Datenbanksystemen. Mit diesen Funktionen können Spalteninhalte geändert und verglichen werden. Zum Beispiel erzeugen die Funktionen ROUND() und RAND() Zufallszahlen. Mit diesen Funktionen wurden die Mitarbeitern mit spezifischen "insanid" ausgewählt:

```
hive> SELECT * FROM insan WHERE insanid=ROUND(rand() * 100);
OK
48      Serhan Batak      Terzi
58      Halil Batak      Jeolog
99      Buket Akar       Ascı
Time taken: 0.115 seconds, Fetched: 3 row(s)
```

Abbildung 65: Verwendung des Befehls ROUND()

Quelle : Işık, M., Herand, D., Adım Adım Big Data ve Uygulamaları, Istanbul, 2019.

Hier erzeugt RAND() einen Wert zwischen 0 und 1. ROUND () konvertiert diesen Wert in eine natürliche Zahl. Andere mathematische Funktionen in Hive werden in Tabellenform wie folgt angezeigt:

Tabelle 2

Mathematische Funktionen

Rückgabetyyp	Verwendung	Beschreibung
DOUBLE	ROUND(DOUBLE a, [INT b])	Liefert den gerundeten Wert von a zurück. Wird b angegeben, so wird auf b Stellen gerundet.
BIGINT	FLOOR(DOUBLE a)	Gibt die größte Zahl als <i>BIGINT</i> zurück, die kleiner oder gleich a ist.
BIGINT	CEIL(DOUBLE a)	Gibt die kleinste Zahl als <i>BIGINT</i> zurück, die größer oder gleich a ist.

DOUBLE	RAND()	Gibt eine Zufallszahl zwischen 0 und 1 zurück. Mit dieser Funktion können Zufalls-IDs für Aufzeichnungen generiert werden.
DOUBLE	POW(DOUBLE a, DOUBLE b)	Gibt ab zurück.
DOUBLE	SQRT(DOUBLE a)	Gibt die Wurzel von a zurück.
DOUBLE	SIN(DOUBLE a), COS(DOUBLE a), TAN(DOUBLE a)	Gibt den Sinus/Kosinus/Tangens von a zurück. a wird im Bogenmaß erwartet.
DOUBLE	ASIN(DOUBLE a), ACOS(DOUBLE a), ATAN(DOUBLE a)	Gibt den Arkussinus/Arkuskosinus/Arkustangens von a zurück, falls $-1 \leq a \leq 1$ ist, ansonsten wird <i>NULL</i> zurückgeliefert.
DOUBLE	DEGREES(DOUBLE a)	Konvertiert den Wert von a vom Bogenmaß in Grad.
DOUBLE	RADIANS(DOUBLE a)	Konvertiert den Wert von a von Grad ins Bogenmaß.
DOUBLE	POSITIVE(INT a), NEGATIVE(INT a)	Gibt a als positiven bzw. als negativer Wert zurück.
DOUBLE	SIGN(DOUBLE a)	Gibt das Vorzeichen in Form von 1.0 oder -1.0 zurück.

DOUBLE	E() / PI()	Liefert den Wert von e/Pi zurück.
--------	------------	-------------------------------------

Quelle : Freiknecht, J., BIG DATA in der Praxis, Germany, 2014.

6.5.2 String-Funktionen

STRING-Funktionen und ihre Beschreibungen lauten wie folgt:

Rückgabewert	Bezeichner	Beschreibung
INT	<i>ASCII(STRING s)</i>	Gibt den ASCII-Code des ersten Zeichens von s zurück.
STRING	<i>CONCAT(STRING a, STRING b ...)</i>	Hängt a und b aneinander, kann noch beliebig viele weitere Parameter annehmen.
INT	<i>INSTR(STRING s, STRING substr)</i>	Gibt die Position zurück, an der der Substring <i>substr</i> in s vorkommt. Index beginnt mit 1. 0 wird zurückgegeben, wenn <i>substr</i> in s nicht vorkommt.
INT	<i>LENGTH(STRING s)</i>	Gibt die Länge von s zurück.
STRING	<i>LOWER(STRING s), UPPER(STRING s)</i>	Gibt s komplett klein- / großgeschrieben zurück.
STRING	<i>TRIM(STRING s)</i>	Schneidet Leerzeichen am Anfang und am Ende von s ab.

Abbildung 66: String-Funktionen in Hive

Quelle : Freiknecht, J., BIG DATA in der Praxis, Germany, 2014.

6.5.3 Aggregatfunktionen

Die Funktionen SELECT und GROUP BY bieten eine Kombination von Datensätzen in den resultierenden Spalten. Die wichtigsten Aggregatfunktionen sind:

Rückgabewert	Bezeichner	Beschreibung
<i>BIGINT</i>	<i>COUNT(*)</i>	Gibt die absolute Anzahl der enthaltenen Datensätze zurück inklusive derer, die <i>NULL</i> -Werte enthalten.
<i>BIGINT</i>	<i>COUNT (a)</i>	Gibt die Anzahl der Datensätze zurück, in denen der Ausdruck a (z. B. eine Spalte) nicht <i>NULL</i> ist.
<i>BIGINT</i>	<i>COUNT (DISTINCT a)</i>	Gibt die Anzahl der Datensätze zurück, in denen der Ausdruck a einzigartig und nicht <i>NULL</i> ist.
<i>DOUBLE</i>	<i>AVG([DISTINCT] spalte)</i>	Gibt den Durchschnittswert der Daten in der Spalte zurück.
<i>DOUBLE</i>	<i>MIN(spalte), MAX(spalte)</i>	Gibt das Minimum/das Maximum aus der Spalte zurück.
<i>DOUBLE</i>	<i>SUM([DISTINCT] spalte)</i>	Bildet die Summe über ein(e) (einzigartiges Element in einer) Spalte.

Abbildung 67: Aggregatfunktionen

Quelle : Freiknecht, J., BIG DATA in der Praxis, Germany, 2014.

7. VIRTUELLE ANWENDUNG MIT HIVE

(Rutherglen, Wampler, & Capriolo, 2012) Dieses Buch beschreibt die Hive-Technologie und Programmiertechniken. Java bietet eine API namens Java Database Connectivity (JDBC), die es Anwendungen ermöglicht, auf Daten zuzugreifen, die in einem RDBMS gespeichert sind, und die Art dieser Daten zu überprüfen. Die meisten Datenbankanbieter stellen einen JDBC-Treiber bereit, der die JDBC-API implementiert und den erforderlichen Code für die Verbindung mit ihrem Datenbankserver enthält.¹⁰

Das Java EE-Anwendungsmodell beginnt mit der Java-Programmiersprache und der Java Virtual Machine. Die bewährte Portabilität, Sicherheit und Entwicklerproduktivität, die sie bieten, bilden die Grundlage des Anwendungsmodells. Java EE wurde entwickelt, um Anwendungen zu unterstützen, die Unternehmensservices für Kunden, Mitarbeiter, Lieferanten, Partner und andere implementieren, die Anforderungen an das Unternehmen stellen oder Beiträge dazu leisten. Solche Anwendungen sind von Natur aus komplex und greifen möglicherweise auf Daten aus einer Vielzahl von Quellen zu und verteilen Anwendungen an eine Vielzahl von Clients. Um diese Anwendungen besser zu steuern und zu verwalten, werden die Geschäftsfunktionen zur Unterstützung dieser verschiedenen Benutzer auf der mittleren Ebene ausgeführt. Die mittlere Ebene stellt eine Umgebung dar, die von der IT-Abteilung eines Unternehmens streng kontrolliert wird. Die mittlere Schicht wird normalerweise auf dedizierter Server-Hardware ausgeführt und hat Zugriff auf die vollständigen Dienste des Unternehmens. Das Java EE-Anwendungsmodell definiert eine Architektur zum Implementieren von Diensten als Multi-Tier-Anwendungen, die die Skalierbarkeit, Zugänglichkeit und Verwaltbarkeit bieten, die für Anwendungen auf Unternehmensebene erforderlich sind. Dieses Modell unterteilt die Arbeit, die benötigt wird, um einen Multi-Tier-Service in folgende Teile zu implementieren:

Die Geschäfts- und Präsentationslogik, die vom Entwickler implementiert wird

Die Standardsystemdienste, die von der Java EE-Plattform bereitgestellt werden

¹⁰ White, T., Hadoop The Definitive Guide, California, 2012.

Der Entwickler kann sich darauf verlassen, dass die Plattform Lösungen für die Probleme auf Hardwareebene bei der Entwicklung eines Multi-Tier-Dienstes bereitstellt.¹¹

7.1 Erstellen eines dynamischen Webprojekts

Zunächst wird ein dynamisches Webprojekt in “eclipse ide java ee” erstellt. Maven wird aktiviert und die Datei pom.xml wird wie folgt erstellt.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi =  
http://www.w3.org/2001/XMLSchema-instance  
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
http://maven.apache.org/xsd/maven-4.0.0.xsd">  
    <modelVersion>4.0.0</modelVersion>  
    <groupId>HiveYonetici</groupId>  
    <artifactId>HiveYonetici</artifactId>  
    <version>0.0.1-SNAPSHOT</version>  
    <packaging>war</packaging>  
    <build>  
        <sourceDirectory>src</sourceDirectory>  
        <plugins>  
            <plugin>  
                <artifactId>maven-compiler-plugin</artifactId>  
                <version>3.1</version>  
                <configuration>  
                    <source>1.7</source>  
                    <target>1.7</target>
```

¹¹ <https://javaee.github.io/tutorial/overview003.html>, 2017.

```

        </configuration>
    </plugin>
    <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.3</version>
        <configuration>
            <warSourceDirectory>WebContent</warSourceDirectory>
            <failOnMissingWebXml>>false</failOnMissingWebXml>
        </configuration>
    </plugin>
</plugins>
</build>
<dependencies>
    <dependency>
        <groupId>jdk.tools</groupId>
        <artifactId>jdk.tools</artifactId>
        <version>1.7.0_45</version>
        <scope>system</scope>
        <systemPath>/usr/lib/jvm/jdk/lib/tools.jar</systemPath>
    </dependency>
    <dependency>
        <groupId>org.apache.hive</groupId>
        <artifactId>hive-jdbc</artifactId>
        <version>0.13.0</version>
    </dependency>

```

```

        <dependency>

            <groupId>org.apache.hadoop</groupId>

            <artifactId>hadoop-common</artifactId>

            <version>2.2.0</version>

        </dependency>

        <dependency>

            <groupId>commons-logging</groupId>

            <artifactId>commons-logging</artifactId>

            <version>1.2</version>

        </dependency>

        <dependency>

            <groupId>org.apache.commons</groupId>

            <artifactId>commons-vfs2</artifactId>

            <version>2.0</version>

        </dependency>

    </dependencies>

</project>

```

Hier ist die tools.jar-Datei unter jdk dargestellt. Dann werden die Informationen wie die Hive-Version und die Hadoop-Version eingegeben. Unter dem Paket werden 3 Klassen erstellt. Diese Klassen sind wie folgt:

HiveYonetici : bietet JDBC-Zugriff.

HiveKullanici: prüft FTP-Zugriff.

StringListe : bringt eine String liste auf einen bestimmten Stil.

HiveYonetici:

Eine Klasse wird mit dem Namen HiveYonetici erstellt. Es wird wie folgt geschrieben.

```
package de.jofre.hivemanager;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

public class HiveYonetici {

    private final static Logger log = Logger.getLogger(HiveYonetici.class.getName());

        private static String driverName = "org.apache.hive.jdbc.HiveDriver";

        Connection con = null;

        public static int PAGING_SIZE = 15;

        // Eger mevcutsa Hive ile baglantiyi kesin

        public void disconnect() {

            if (con != null) {

                try {

                    con.close();

                } catch (SQLException e) {

                    log.log(Level.SEVERE, "Baglanti kapatilirken hata olustu!");

                    e.printStackTrace();

                }

            }

        }

    }
```

```

    }
}

// Hive 'e baglanma

public boolean connect(String host, String port, String user,
    String password) {
    try {
        Class.forName(driverName);
    } catch (ClassNotFoundException e1) {
        log.log(Level.SEVERE,
            "Baglanti olustumada hata - sürücü sinifi bulunamadi!");
        e1.printStackTrace();
        return false;
    }
    try {
        con = DriverManager.getConnection("jdbc:hive2://" + host + ":"
            + port, user, password);
    } catch (SQLException e) {
        log.log(Level.SEVERE, "Baglanti olusturmada hata!");
        e.printStackTrace();
        return false;
    }
    if (con != null) {
        // Sayfalamaya izin vermek için, UDF 'yi kaydetme
        Statement stmt;

```

```

        try {

            stmt = con.createStatement();

            stmt.execute("ADD                                     JAR
/usr/local/hive/lib/15_HiveUDFs-0.0.1-SNAPSHOT-jar-with-dependencies.jar");

            stmt.execute("CREATE  TEMPORARY  FUNCTION
auto_inc as 'de.jofre.hive.udf.AutoIncrement'");

        } catch (SQLException e) {

            log.log(Level.SEVERE,
                "Gecici UDF'ler kaydedilirken hata olustu.");
            e.printStackTrace();

        }

        return true;

    } else {

        return false;

    }

}

```

// Rufe Datenbanken ab

```

public List<String> getDatabases() {

    log.log(Level.INFO, "Veritabanini sorgulama...");

    List<String> databases = new ArrayList<String>();

    try {

        Statement stmt = con.createStatement();

        String sql = "show databases";

        ResultSet res = stmt.executeQuery(sql);

        while (res.next()) {

```

```

        databases.add(res.getString(1));

        log.log(Level.INFO, "\t" + res.getString(1));

    }

} catch (SQLException e) {

    log.log(Level.SEVERE, "Veritabani sorgulamada hata!");

    e.printStackTrace();

}

return databases;

}

// Bir veritabanindaki tum tablolari al
public List<String> getTables(String database) {

    log.log(Level.INFO, "Erfrage Tabellen aus " + database + "...");

    List<String> tables = new ArrayList<String>();

    try {

        // Veritabani degistirme

        switchDatabase(database);

        Statement stmt = con.createStatement();

        String sql = "show tables";

        ResultSet res = stmt.executeQuery(sql);

        while (res.next()) {

            tables.add(res.getString(1));

            log.log(Level.INFO, "\t" + res.getString(1));

        }

    } catch (SQLException e) {

        log.log(Level.SEVERE,

```

```

        "Veritabanından tabloların sorgulanmasında hata"+ database + ".");

        e.printStackTrace();

    }

    return tables;

}

// Veritabani semasini degistir

public void switchDatabase(String database) {

    try {

        Statement stmt = con.createStatement();

        stmt.execute("use " + database);

    } catch (SQLException e) {

        log.log(Level.SEVERE, "Veritabani seciminde hata "+ database + ".");

        e.printStackTrace();

    }

}

// Bir tablodaki kayitların sayisi

public int getCount(String table) {

    int result = -1;

    try {

        Statement stmt = con.createStatement();

        String sql = "SELECT COUNT(*) FROM " + table;

        ResultSet res = stmt.executeQuery(sql);

        if (res.next()) {

            result = res.getInt(1);

        }

    }

```

```

        } catch (SQLException e) {

log.log(Level.SEVERE, "Kayitlar sorgulanirken hata olustu "+ table + ".");

        e.printStackTrace();

        }

        return result;

    }

// Sorguyu calistirma

public ResultSet executeQuery(String query) {

    log.log(Level.INFO, "Sorguyu calistir '" + query + "' dan.");

    ResultSet res = null;

    try {

        Statement stmt = con.createStatement();

        res = stmt.executeQuery(query);

    } catch (SQLException e) {

log.log(Level.SEVERE, "Ifadelerin sorgulanmasinda hata.");

        e.printStackTrace();

    }

    return res;

}

// Hive'deki tum rolleri listele

public List<String> getRoles() {

    List<String> roles = new ArrayList<String>();

    try {

        Statement stmt = con.createStatement();

        String sql = "show roles";

```

```

        ResultSet res = stmt.executeQuery(sql);

        while (res.next()) {

            roles.add(res.getString(1));

        }

    } catch (SQLException e) {

        log.log(Level.SEVERE, "Rollerde hata.");

        e.printStackTrace();

    }

    return roles;

}

public List<String> getRolesOfUser(String user) {

    List<String> roles = new ArrayList<String>();

    try {

        Statement stmt = con.createStatement();

        String sql = "show role grant user " + user;

        ResultSet res = stmt.executeQuery(sql);

        while (res.next()) {

            roles.add(res.getString(1));

        }

    } catch (SQLException e) {

        log.log(Level.SEVERE, "Rollerde hata.");

        e.printStackTrace();

    }

    return roles;

}

```

}

Mit dem String `driverName` wird die Klassennamen des Hive-Drivers bezeichnet. Die Verbindung zur Datenbank wird mit dem Objekt `“con”` und die Anzahl der Anzeigen der Tabellen mit dem Variablen `PAGING_SIZE` bestimmt. Dann wird die Verbindung mit der Methode `“connect”` benutzt. `JDBC-Connection` wird mit der Methode `“DriverManager.getConnection”` bestimmt.

In der Methode `“getDatabases”` wird ein Objekt `“stmt”` mit `“con.createStatement”` abgerufen. Das Query-Statement wird mit `“stmt.executeQuery(sql)”` ausgeführt und dann wird das Ergebnis in einem `“ResultSet”` gespeichert.

Mit der Methode `“getTables”` wird über die Methode `“switchDatabase”` den Befehl `“show tables”` verwendet. Dann wird eine Liste von Strings zurückgegeben.

Es folgt eine Methode mit dem Namen `“getCount”`. Diese Method wird ein Statement `“SELECT COUNT(*) FROM TABLE”` generiert. Dann wird mit der Abfrage `“res.next()”` der Zähler als Ganzzahl dargestellt.

7.2 Erstellen von JSP-Dateien für Hive-Yonetici

Für das Projekt Hive-Yonetici werden drei .jsp-Dateien erstellt. Diese Dateien sind wie folgt.

`index.jsp` listet die Tabellen der Datenbank auf.

`data.jsp` ist ein Link in der `index.jsp`. Es ruft die Daten ab und zeigt sie. Die Daten können sehr groß sein und der Paging-Mechanismus kann dazu verwendet werden.

`query.jsp` ist die Datei, in die Ihre eigenen Abfragen eingegeben werden. Der Paging-Mechanismus wird hier nicht verwendet.

Die Datei `index.jsp` wird erstellt und der folgende Code wird geschrieben.

```
<%@page import="java.util.ArrayList"%>
```

```
<%@page import="de.jofre.hivemanager.HiveYonetici"%>
```

```

<%@page import="java.util.List"%>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<link rel="shortcut icon" href="img/favicon.ico" type="image/x-icon" />
<link rel="stylesheet" href="css/style.css" type="text/css" />
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Hive-Yonetici</title>
</head>
<body>
    <div id="container">
        <div id="header">
            <h1>Hive-Yonetici</h1>
        </div>
        <div id="navigation">
            <ul>
                <li><a href="index.jsp"><b>Datenbanken und Tabellen </b></a></li>
                <li><a href="data.jsp">Daten</a></li>
                <li><a href="query.jsp">Abfragen</a></li>
                <li><a href="users.jsp">Benutzerverwaltung</a></li>
            </ul>
        </div>

```

```

        <div id="content">

            <%

                HiveYonetici hm = new HiveYonetici();

                boolean connected = hm.connect("localhost", "10000", "hduser", "hduser");

                if (connected) {

                    List<String> databases = hm.getDatabases();

                    out.println("<table style=\"border-width: 1px; border-style: solid; width: 100%;\">");

                    out.println("<tr><td><b>Datenbank/Schema</b></td><td><b>Tabellen</b></td></tr>");

                    for(int i=0; i<databases.size(); i++) {

                        out.println("<tr><td>" + databases.get(i) + "</td>");

                        List<String> tables = hm.getTables(databases.get(i));

                        out.println("<td>");

                        for(int j=0; j<tables.size(); j++) {

                            out.println("<a href=\"data.jsp?db="+databases.get(i)

                                + "&table="+tables.get(j)+"\">" + tables.get(j) + "</a>");

                            if (j<tables.size()-1) {

                                out.println(",");

                            }

                        }

                        out.println("</td>");

                    }

                    out.println("</tr>");

                }

                out.println("</table>");
            %>

```

```

        } else {

out.println("Hive ile baglanti kurulamadi!");

        }

        hm.disconnect();

    %>

</div>

<div id="footer">Marmara Universität | <a Enformatik Master</a></div>

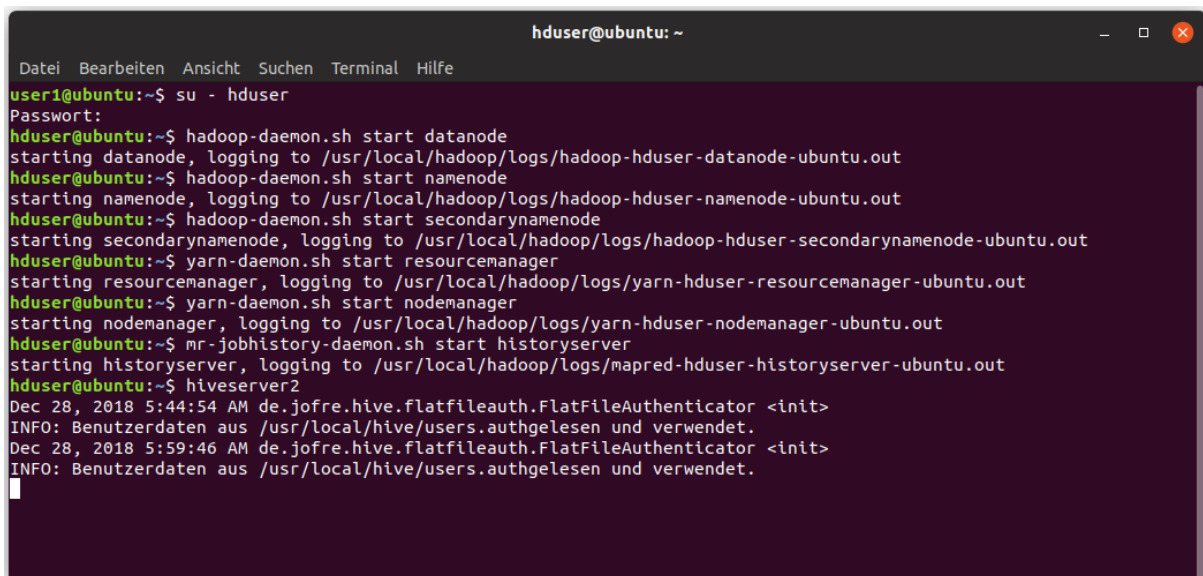
</div>

</body>

</html>

```

Nach dem Schreiben des Codes wird das Terminal geöffnet. Der Benutzer wird als "hduser" geändert. Die Dienste für Hadoop werden gestartet. Dann wird der hiveserver2 Dienst ausgeführt. Schließlich wird das Projekt als Server ausgeführt. Nach dem Herstellen der Verbindung zum Server hiveserver2 als Client ist das Terminal wie folgt.



```

hduser@ubuntu: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
user1@ubuntu:~$ su - hduser
Passwort:
hduser@ubuntu:~$ hadoop-daemon.sh start datanode
starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-ubuntu.out
hduser@ubuntu:~$ hadoop-daemon.sh start namenode
starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-ubuntu.out
hduser@ubuntu:~$ hadoop-daemon.sh start secondarynamenode
starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-ubuntu.out
hduser@ubuntu:~$ yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-ubuntu.out
hduser@ubuntu:~$ yarn-daemon.sh start nodemanager
starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-ubuntu.out
hduser@ubuntu:~$ mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /usr/local/hadoop/logs/mapred-hduser-historyserver-ubuntu.out
hduser@ubuntu:~$ hiveserver2
Dec 28, 2018 5:44:54 AM de.jofre.hive.flatfileauth.FlatFileAuthenticator <init>
INFO: Benutzerdaten aus /usr/local/hive/users.authgelesen und verwendet.
Dec 28, 2018 5:59:46 AM de.jofre.hive.flatfileauth.FlatFileAuthenticator <init>
INFO: Benutzerdaten aus /usr/local/hive/users.authgelesen und verwendet.

```

Abbildung 68: Die Terminalansicht

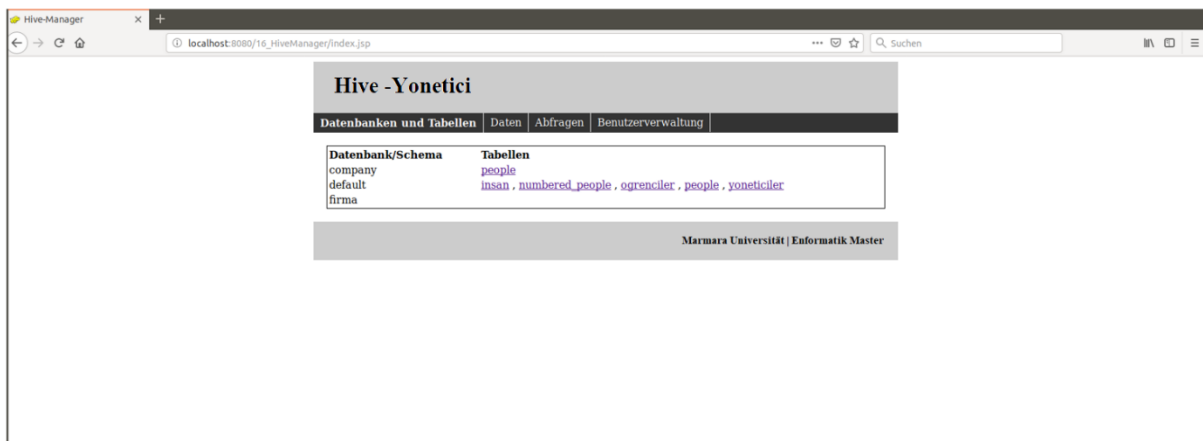


Abbildung 69: Anzeige der Datenbank und Tabellen in Hive-Yonetici

JSP ist eine JAVA-Technologie, die JAVA als Programmiersprache zum Erstellen dynamischer Websites verwendet. Auf JSP-Seiten werden JAVA-Codes zwischen `<%>` -Tags geschrieben und am Ende ein Semikolon eingefügt. Mit der Klasse `HiveYonetici` werden eine Verbindung zu Hive hergestellt. Wenn die Verbindung erfolgreich ist, wird die `getDatabases`-Methode aufgerufen und die zugehörigen Tabellen werden mit Hilfe von `getTables` erreicht. Jede Tabelle wird durch Aufrufen von `data.jsp` als Link angezeigt. Durch Klicken auf den Link wird ein Ergebnis wie im folgenden Bildschirm angezeigt.

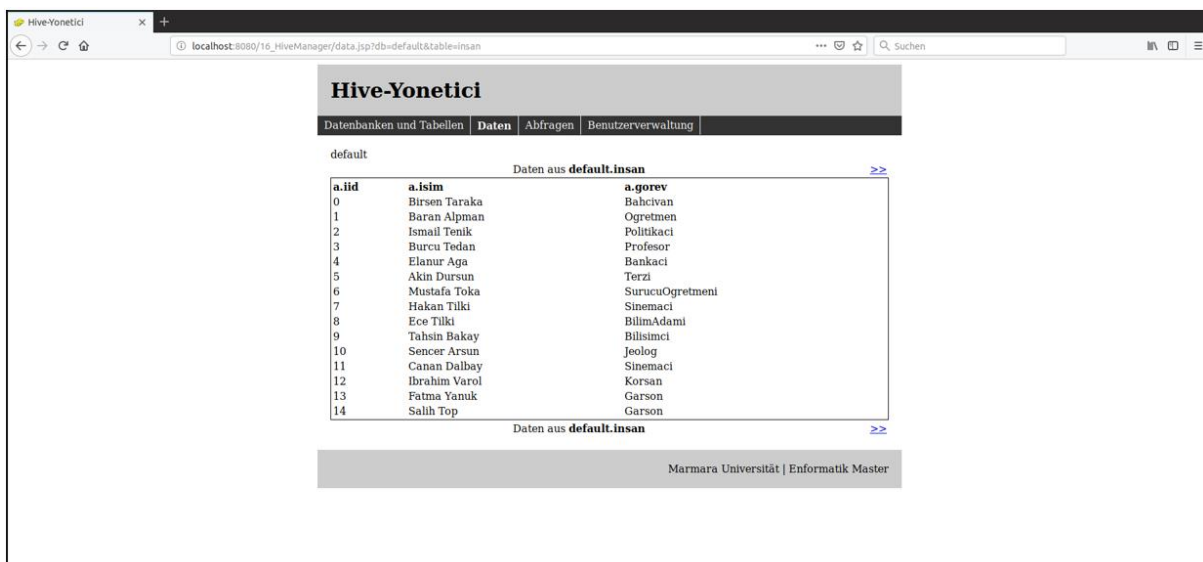


Abbildung 70: Anzeige von Daten in der Tabelle insan

Die Datei `data.jsp` wird erstellt und die folgenden Codes werden geschrieben. Die Codebeschreibung folgt nach den Codes.

```

<%@page import="java.sql.Types"%>

<%@page import="java.sql.ResultSetMetaData"%>

<%@page import="java.sql.ResultSet"%>

<%@page import="java.util.ArrayList"%>

<%@page import="de.jofre.hivemanager.HiveYonetici"%>

<%@page import="java.util.List"%>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<link rel="shortcut icon" href="img/favicon.ico" type="image/x-icon" />

<link rel="stylesheet" href="css/style.css" type="text/css" />

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Hive-Yonetici</title>

</head>

<body>

    <div id="container">

        <div id="header">

            <h1>Hive-Yonetici</h1>

        </div>

        <div id="navigation">

            <ul>

                <li><a href="index.jsp">Datenbanken und Tabellen</a></li>

```

```

        <li><a href="data.jsp"><b>Daten</b></a></li>

        <li><a href="query.jsp">Abfragen</a></li>

        <li><a href="users.jsp">Benutzerverwaltung</a></li>

    </ul>

</div>

<div id="content">

<%
    HiveYonetici hm = new HiveYonetici();
    boolean connected = hm.connect("localhost", "10000", "hduser", "hduser");

    if (connected) {
        // Veritabani ve tablo secili mi?
        String db = (String) request.getParameter("db");
        String table = (String) request.getParameter("table");

        out.println(db);

        // Sayfalama gerekli mi?
        String from = (String) request.getParameter("from");

        if (db != null && !db.trim().equals("") && table != null
            && !table.trim().equals("")) {

            hm.switchDatabase(db);

            ResultSet rs;

            if (from != null) {

                rs = hm.executeQuery("SELECT * FROM (SELECT *, auto_inc() as
row_count FROM " + table + ") A WHERE A.row_count >= " + from + " AND A.row_count
< " + (Integer.parseInt(from) + HiveYonetici.PAGING_SIZE) + " LIMIT " +
HiveYonetici.PAGING_SIZE);

```

```

    }

    else

    {

        rs = hm.executeQuery("SELECT * FROM (SELECT *, auto_inc() as
row_count FROM " + table + ") A WHERE A.row_count < " + HiveYonetici.PAGING_SIZE
+ " LIMIT " + HiveYonetici.PAGING_SIZE);

    }

    // Sayfalama butonu

    out.println("<table style=\"width: 100%;\">");

    if (from != null && !from.equals("0"))
    {

        int from_paging = Integer.parseInt(from)
        - HiveYonetici.PAGING_SIZE >= 0 ? Integer.parseInt(from) -
HiveYonetici.PAGING_SIZE : 0;

        int to_paging = Integer.parseInt(from) + HiveYonetici.PAGING_SIZE;

        out.println("<tr><td style=\"text-align: left;\"><a href=\"data.jsp?db="+ db +
"&table=" + table + "&from=" + from_paging
+ "\"><<</a></td><td style=\"text-align: center;\">Daten aus <b>"
+ db + "." + table+ "</b></td><td style=\"text-align: right;\"><a href=\"data.jsp?db="
+ db + "&table=" + table + "&from=" + to_paging + "\">>></a></td></tr>");

    }

    else

    {

        out.println("<tr><td></td><td style=\"text-align: center;\">Daten aus <b>" + db + "."
+ table + "</b></td><td style=\"text-align: right;\"><a href=\"data.jsp?db="
+ db + "&table=" + table + "&from=" + HiveYonetici.PAGING_SIZE

```

```

+ "\">>></a></td></tr>");

}

out.println("</table>");

// Veriler isleniyor

ResultSetMetaData rsmd = rs.getMetaData();

out.println("<table style=\"border-width: 1px; border-style: solid; width: 100%;\">");

// Sutun basligi

int columnCount = rsmd.getColumnCount();

out.println("<tr>");

for (int i = 1; i < columnCount; i++) {

    out.println("<td><b>" + rsmd.getColumnName(i) + "</b></td>");

}

out.println("</tr>");

// Tablodaki veriler cikisa veriliyor

while (rs.next()) {

out.println("<tr>");

for (int i = 1; i < columnCount; i++) {

    switch (rsmd.getColumnType(i)) {

        case Types.ARRAY:

            out.println("<td>" + rs.getArray(i) + "</td>");

            break;

        case Types.INTEGER:

            out.println("<td>" + rs.getInt(i) + "</td>");

            break;

        case Types.TINYINT:

```

```
        out.println("<td>" + rs.getBytes(i) + "</td>");
        break;
    case Types.SMALLINT:
        out.println("<td>" + rs.getShort(i) + "</td>");
        break;
    case Types.BIGINT:
        out.println("<td>" + rs.getLong(i) + "</td>");
        break;
    case Types.VARCHAR:
    case Types.LONGNVARCHAR:
    case Types.CHAR:
        out.println("<td>" + rs.getString(i) + "</td>");
        break;
    case Types.BIT:
    case Types.BOOLEAN:
        out.println("<td>" + rs.getBoolean(i) + "</td>");
        break;
    case Types.DATE:
        out.println("<td>" + rs.getDate(i) + "</td>");
        break;
    case Types.TIME:
        out.println("<td>" + rs.getTime(i) + "</td>");
        break;
    case Types.TIMESTAMP:
        out.println("<td>" + rs.getTimestamp(i) + "</td>");
```

```

        break;

    case Types.REAL:

    case Types.FLOAT:

        out.println("<td>" + rs.getFloat(i) + "</td>");

        break;

    case Types.DOUBLE:

        out.println("<td>" + rs.getDouble(i) + "</td>");

        break;

    case Types.BINARY:

    case Types.VARBINARY:

        out.println("<td>" + rs.getBinaryStream(i) + "</td>");

        break;

    default:

        out.println("<td>" + rs.getString(i) + "</td>");

    }

}

out.println("</tr>");

}

out.println("</table>");

// Sayfalama butonu

out.println("<table style=\"width: 100%;\">");

if (from != null && !from.equals("0")) {

    int from_paging = Integer.parseInt(from)

    - HiveYonetici.PAGING_SIZE >=0?Integer

    .parseInt(from) - HiveYonetici.PAGING_SIZE : 0;

```

```

        int to_paging = Integer.parseInt(from)

        + HiveYonetici.PAGING_SIZE;

        out.println("<tr><td style=\"text-align: left;\"><a href=\"data.jsp?db=\""
        +db+"&table="+table+"&from="+from_paging
        + "\"><<</a></td><td style=\"text-align: center;\">Daten aus <b>"
        + db + "." + table+ "</b></td><td style=\"text-align: right;\"><a
href=\"data.jsp?db=\"" + db + "&table=" + table + "&from=" + to_paging +
"\">>></a></td></tr>");

    }

    else {

        out.println("<tr><td></td><td style=\"text-align: center;\">Daten aus <b>"
        + db + "." + table+ "</b></td><td style=\"text-align: right;\"><a
href=\"data.jsp?db=\"" + db + "&table=" + table + "&from=" + HiveYonetici.PAGING_SIZE +
"\">>></a></td></tr>");

    }

    out.println("</table>");

}

else {

    out.println("Lutfen bir veritabani ve tablo seciniz.");

}

}

else {

    out.println("Hive ile baglanti kurulamadi!");

}

hm.disconnect();

%>

```

```

        </div>

        <div id="footer">

                Marmara Universität | Informatik Master

        </div>

</div>

</body>

</html>

```

Mit der Klasse HiveYonetici werden eine Verbindung zu Hive hergestellt. Die folgenden Codes werden geschrieben.

```

HiveYonetici hm = new HiveYonetici();

boolean connected = hm.connect("localhost", "10000", "hduser", "hduser");

```

Der HTTP-Request fragt, ob Datenbank und Tabelle vorhanden sind. Es wird geprüft, ob die Verbindung bereits besteht.

```

if (connected) {

        String db = (String) request.getParameter("db");

        String table = (String) request.getParameter("table");

        out.println(db);

        ...
}

```

Das erste n- Datensätze wird mit einer Select-Statement ausgeführt. Die Anzahl der Datensätze aus der angegebenen Tabelle wird mit der Ganzzahl "paging_size" geändert. Die Unterabfrage wird mit der Kennung A identifiziert, wodurch auf die Daten zugegriffen wird. Die Nummerierung wird mit "auto_inc" eingestellt. Die Bedingung wird durch WHERE begrenzt.

```
rs = hm.executeQuery("SELECT * FROM (SELECT *, auto_inc() as row_count
FROM " + table + ") A
```

```
WHERE A.row_count >= " + from + " AND A.row_count < " +
(Integer.parseInt(from) + HiveYonetici.PAGING_SIZE) + " LIMIT " +
HiveYonetici.PAGING_SIZE);
```

Dann werden Pfeile in Form von HTML-Links erstellt, die sich zwischen den Seiten hin und her blättern. Die Seitenanzahl wird mit "PAGING_SIZE" ermittelt. Da das Format der Daten nicht bekannt ist, werden diese zunächst mit den Metadaten des "resultSet" abgefragt und anschließend die entsprechende get-Methode verwendet (z. B. getFloat, getInt, getString ...). Und zur Anzeige der Daten wird die HTML-Tabelle dargestellt.

```
out.println("<table style=\"width: 100%;\">");
if (from != null && !from.equals("0")) {
    int from_paging = Integer.parseInt(from)
        - HiveYonetici.PAGING_SIZE >= 0 ? Integer.parseInt(from) -
HiveYonetici.PAGING_SIZE : 0;

    int to_paging = Integer.parseInt(from) + HiveYonetici.PAGING_SIZE;

    out.println("<tr><td style=\"text-align: left;\"><a href=\"data.jsp?db=\"" + db
        + "&table=\"" + table + "&from=\"" + from_paging + "\"><<</a></td><td style=\"text-
align: center;\">Daten aus <b>" + db
        + "." + table + "</b></td><td style=\"text-align: right;\"><a href=\"data.jsp?db=\"" + db
        + "&table=\"" + table + "&from=\"" + to_paging + "\">>></a></td></tr>");
}
else {
    out.println("<tr><td></td><td style=\"text-align: center;\">Daten aus <b>" + db
        + "." + table + "</b></td><td style=\"text-align: right;\"><a href=\"data.jsp?db=\"" + db +
        "&table=\"" + table + "&from=\"" + HiveYonetici.PAGING_SIZE + "\">>></a></td></tr>");
}
```

Der Abschnitt "Abfragen" wird ausgewählt, um die angeforderte SQL-Abfrage auszuführen. Der Abschnitt "Abfragen" wird unten dargestellt. Außerdem noch werden die Abfrage und die Ergebnisse angezeigt.

Hive-Yonetici

Datenbanken und Tabellen | Daten | **Abfragen** | Benutzerverwaltung

Query:

Datenbank: default

Absenden

insan.id	insan.isim
0	Birsen Taraka
1	Baran Alpman
2	Ismail Tenik
3	Burcu Tedan
4	Elanur Aga
5	Akin Dursun
6	Mustafa Toka
7	Hakan Tilki
8	Ece Tilki
9	Tahsin Bakay
10	Sencer Arsun
11	Canan Dalbay
12	Ibrahim Varol
13	Patma Yamuk
14	Salih Top
15	Hakan Bakay
16	Beyza Yaray
17	Ceyhan Toka
18	Semih Dillim
19	Kaan Varol
20	Oguz Akar
21	Akin Batak
22	Candan Akar

Marmara Universität | Enformatik Master

Abbildung 71: Anzeige von Daten in der Teil “Abfragen”

8. Schlussfolgerungen

Big Data ist ein englischer Begriff und bedeutet sehr große, komplexe und unstrukturierte Daten. Wir hören diesen Begriff oft und die Bedeutung von Big Data steigt täglich. Daher ist der Einsatz dieser neuen Technologie zur Notwendigkeit geworden. Die Unternehmen haben begonnen, ihre technologischen Investitionen in diese Bereiche zu verlagern, um in ihren Sektoren wettbewerbsfähig zu sein.

Die Ausarbeitung meiner Masterarbeit "Die implementierung der Hive-Architektur in Big-Data in einem virtuellen Unternehmen" beschreibt die Theorie und Anwendung der Hive-Warehouse. Dieses Projekt wurde mit Hive-Warehouse über ein Map-Reduce-Framework durchgeführt. Der Hauptzweck dieser Studie ist die Visualisierung von Daten mithilfe von Big Data.

Da es nicht möglich war, große Daten von der Quelle abzurufen, wurden die Daten mit einer Java-Anwendung generiert. Diese Daten wurden in eine Textdatei übertragen. Diese Textdatei wurde in einer Datenbank gespeichert, die zuvor mit den Hive-Abfragebefehlen im HDFS-System erstellt wurde. Zur Übertragung dieser Daten wurde die Macht von Mapreduce genutzt. Anschließend wurde ein JSP-Projekt erstellt, um diese Daten zu visualisieren. In diesem Projekt wurde eine Hive-Verbindung hergestellt und die Daten angezeigt und visualisiert.

Zusammenfassend wurde die Erstellung großer Datenmengen, die Verarbeitung und Visualisierung der Daten bereitgestellt. Darüber hinaus wurden die Hive-Architektur und die Zusammenarbeit von Hadoop mit der Leistungsfähigkeit des Mapreduce-Frameworks genutzt.

LITERATURVERZEICHNIS

- Apache Hadoop*. (21. 06 2014). Von <https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html> abgerufen
- Aunkofer, B. (26. 2 2018). *Distributed Computing – MapReduce Algorithmus*. Von <https://data-science-blog.com/blog/2018/03/26/distributed-computing-mapreduce-algorithmus/> abgerufen
- Freiknecht, J. (2014). *Big Data in der Praxis* (Bd. 1). München: Hanser.
- Işık, M., & Herand, D. (2019). *Adım Adım Big Data ve Uygulamaları* (Bd. 1). Istanbul: Pusula.
- MapReduce*. (7. 11 2018). Von Wikipedia: <https://en.wikipedia.org/wiki/MapReduce> abgerufen
- Rekursion*. (19. 12 2018). Von Wikipedia: <https://de.wikipedia.org/wiki/Rekursion> abgerufen
- Rouse, M. (1. 4 2015). *Hadoop Distributed File System (HDFS)*. Von <https://www.searchenterprisesoftware.de/definition/Hadoop-Distributed-File-System-HDFS> abgerufen
- Rutherglen, J., Wampler, D., & Capriolo, E. (2012). *Programming Hive* (Bd. 1). Sebastopol: O'Reilly.
- Strata Hadoop*. (6. 5 2015). Von <https://conferences.oreilly.com/strata/big-data-conference-uk-2015/public/schedule/detail/42456> abgerufen
- Was ist MapReduce?* (28. 12 2018). Von <https://www.bigdata-insider.de/was-ist-mapreduce-a-624936/> abgerufen
- What is Hive? Architecture & Modes*. (2018). Von <https://www.guru99.com/introduction-hive.html> abgerufen