**Dokumentation Cassandra**

1. **Installation**

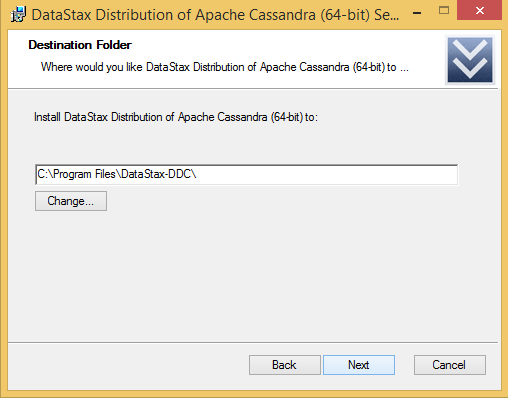
Bezogen kann die Installationssoftware für Cassandra von der Apache Foundation.

<http://cassandra.apache.org/download/>

Alternativ kann für einen einfachen Windows Installer auch die Version von DataStax Enterprise genutzt werden.

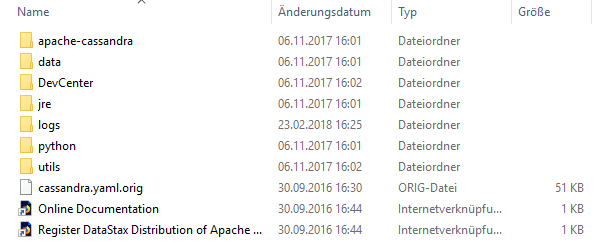
<https://academy.datastax.com/planet-cassandra/cassandra>

Wir haben uns für den Windows Installer von DataStax entschieden. Die Installation verlief problemlos und man konnte einfach den einzelnen Schritten folgen.

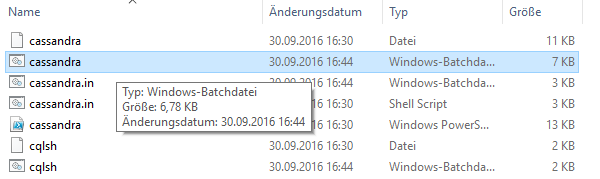


Der Installationsumfang besteht dabei aus einer voll funktionstüchtigen Version von Apache Cassandra welche aus einem Server und einer CQL-Shell(Cassandra Query Language) besteht. Zusätzlich gibt es eine Version des DataStax DevCenters die man kostenfrei nutzen kann.

Nach der erfolgreichen Installation erhält man folgende Verzeichnisstruktur.

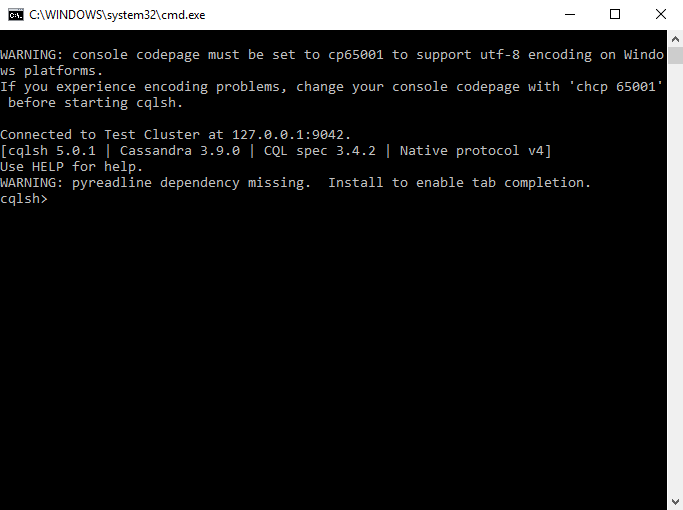


Durch das Auführen der Cassandra Batch-Datei wird der Server gestartet .

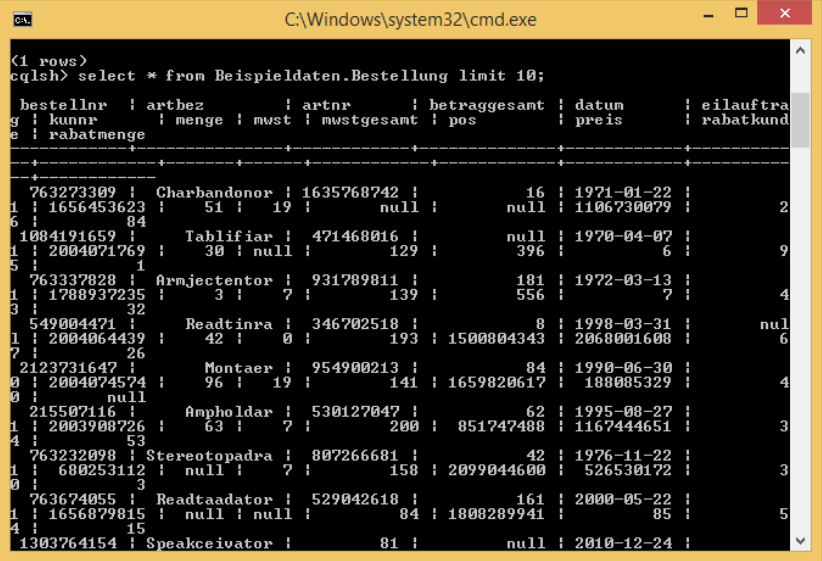


Dieser Server steht nun zur Verfügung und wartet auf eingehende CQL-Client-Anfragen.

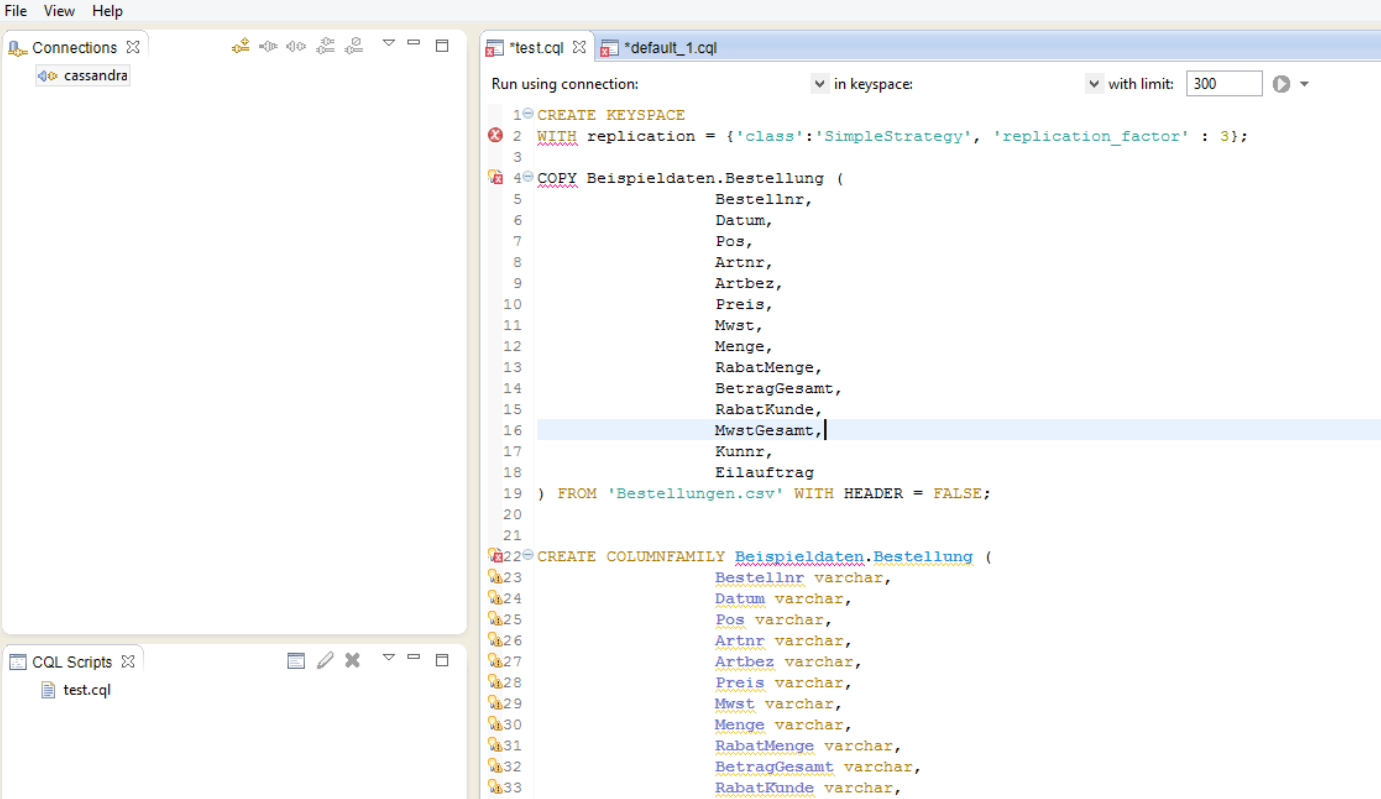
Eine Möglichkeit für einen CQL-Client bietet die in der Installation beinhaltete CQL-Shell die man ebenfalls einfach durch ausführen der „cqlsh.bat“ (Batchdatei) starten kann. Automatisch mit dem Server verbunden, steht diese bereit um Anfragen an den Cassandra-Server zu übertragen.



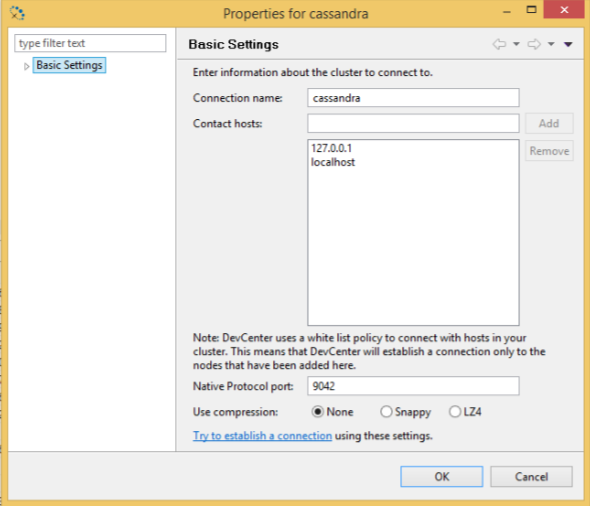
In der CQL-Shell hat man die Möglichkeit mit Hilfe von CQL-Befehlen mit einer Cassandra-Datenbank zu interagieren.



Wie im Bild zu sehen ist die Übersichtlichkeit in einer Shell stark eingeschränkt zu dem gibt es auch einige Probleme mit der Zeichenkodierung.

Die Vorteile der Cassandra Version von DataStax lagen dabei auch an dem Bereitstellen des DataStax DevCenters. Man hat die Möglichkeit das DevCenter mit dem Server zu verbinden und dieses ebenfalls als CQL-Client zu benutzen um mit der Datenbank arbeiten zu können. Dabei hat man die Vorteile einer übersichtlicheren Oberfläche die keine Probleme mit der Zeichenkodierung hat.

Die Verbindung zwischen dem DevCenter und dem Cassandra-Server stellt man über die Connections-Einstellungen im DevCenter her.

Der Server, der auf CQL-Clients im Localhost-Bereich wartet kann über den Port 9042 erreicht werden.

1. **Arbeit mit Cassandra Datenbank**

**2.1 Anlegen der Datenbank**

Nachdem das Fallbeispiel in den Systemen von MSSQL und SAPHana aufgesetzt wurde, konnte der Aufbau der Cassandra-Datenbank erfolgen.

Als Grundlage dienten hier die von der MSSQL-Datenbank exportierten CSV-Dateien der einzelnen Tabellen.

Um die CSV-Dateien erfolgreich importieren zu können, müssen zuerst ein Keyspace und alle Columnfamilies(=Tables) angelegt werden.

CREATE KEYSPACE Beispieldaten

WITH replication = {'class':'SimpleStrategy', 'replication\_factor' : 1};

Zu Testzwecken war die „SimpleStrategy“ mit einem „replication\_factor:1“ völlig ausreichend. Die Bedeutung von „SimpleStrategy“ und „replication\_factor:1“ ist, dass für den einen vorhanden Cassandra-Cluster eine einzelne Replikation als Backup dient.

Es gibt also auch noch die Möglichkeit für jeden weiteren Cluster einen anderen Replikationsfaktor auszuwählen.

Die Tabellen konnten dann ähnlich wie in SQL einfach mit einem „CREATE TABLE“ angelegt werden. Hierbei spielt es keine Rolle ob man „Columnfamily“ oder „Table“ benutzt.

*CREATE COLUMNFAMILY Beispieldaten.Kunden (*

*Kunnr varchar,*

*Name varchar,*

*Vorname varchar,*

*PLZ varchar,*

*Ort varchar,*

*Landkz varchar,*

*Land varchar,*

*Bundesland varchar,*

*Region varchar,*

*Debitornr varchar,*

*Kreditorennr varchar,*

*BLZ varchar,*

*IBAN varchar,*

*Kreditinstitut varchar,*

*Beziehungslvl varchar,*

*Werber varchar,*

*PRIMARY KEY(Kunnr));*

Der Import der CSV-Dateien erfolgt über den *COPY-*Befehl.

*COPY Beispieldaten.Kunden (*

*Kunnr,*

*Name,*

*Vorname,*

*PLZ,*

*Ort,*

*Landkz,*

*Land,*

*Bundesland,*

*Region,*

*Debitornr,*

*Kreditorennr,*

*BLZ,*

*IBAN,*

*Kreditinstitut,*

*Beziehungslvl,*

*Werber*

*) FROM 'KundeUTF.csv' with Header = True;*

Beim Importieren der Daten kam es dann zu ersten Problemen.

Das DataStax DevCenter unterstützt leider nicht die aktuellste Version der CQL(Cassandra Query Language) wodurch der *COPY*-Befehl nicht ausgeführt werden konnte. Das heißt man muss diesen in der Cassandra-Shell ausführen, welche jedoch auch nach Umstellen der Shell-Eigenschaften bzgl. der Codierung nicht mit den CSV-Dateien umgehen konnte.

Erst nachdem die CSV-Dateien via Notepad++ in UTF-8 Format gespeichert wurden konnten diese erfolgreich importiert werden.

Der Import erfolgt dann jedoch sehr schnell und dauert trotz 10 Millionen Zeilen lediglich ca. 5 Minuten.

* 1. **Einfache Abfragen für Vergleich finden und ausführen**

Um die Cassandra-Datenbank mit MSSQL, SAPHana und MariaDB mit Memchached zu vergleichen mussten Abfragen gefunden werden die auch auf einer spaltenorientierten No-SQL-Datenbank funktionieren, die nur begrenzt Aggregatfunktionen zur Verfügung stellt und Abfragen über mehrere Tabellen via JOINS nicht unterstützt.

Der Vergleich geschah auf der Grundlage dass Cassandra zwar In-Memory-Datenspeicherung unterstützt, man diese Funktion jedoch nur in einer lizenzpflichtigen Version der DataStax Enterprise verwenden kann und somit stand uns diese nicht zur Verfügung.

Da Aggregatfunktionen wie MIN, MAX, AVG, SUM und COUNT bereits unterstützt, beschränkten wir uns auf diese. Man hat jedoch die Möglichkeit selber Aggregat-Funktionen anzulegen.

Folgende einfache Abfragen wurden zum Vergleich von Cassandra mit den restlichen Systemen genutzt.

*SELECT sum(mitarbeiteranzahl) AS MA\_Bundesland*

*FROM beispieldaten.standorte*

*WHERE bundesland ='Berlin' allow filtering;*

*SELECT min(betraggesamt)AS Kleinster\_Gesamt\_Betrag*

*FROM beispieldaten.bestellung*

*WHERE menge <= 100 allow filtering;*

*SELECT max(preis),min(preis)*

*FROM beispieldaten.bestellung*

*WHERE menge <= 100 allow filtering;*

*SELECT max(fahrzeuganzahl),max(mitarbeiteranzahl)*

*FROM beispieldaten.lieferdienst*

*WHERE fahrzeugtyp = 'Auto' allow filtering;*

Auch hier entstanden erneut Probleme mit dem DataStax DevCenter da dieses keine Möglichkeit bot, den „Request-Timeout“ zu erweitern und somit nicht dafür geeignet war diese Abfragen auszuführen, ohne die Verbindung zum Server zu verlieren.

Anfangs war auch die CQL-Shell auf wenige Sekunden „Request-Timeout“ beschränkt, jedoch konnte man die Shell einfach wie folgt starten und den Timeout ausreichend verzögern.

*cqlsh --request-timeout=3600*

Das arbeiten in der Shell führte dann aber zu dem Problem, dass es keine Möglichkeit gab die Zeiten pro Abfrage zu sehen bzw. anzeigen zu lassen.

Folglich musste via *CAPTURE*-Befehl ein Textdatei angegeben werden, in der eine detaillierte Anzeige des Outputs samt Zeiten aufgelistet wird.

CAPTURE *'Logdatei.txt‘ ON*

Leider führte das Anlegen der Logdateien zu leichten Verfälschungen bei den Abfragezeiten was den allgemeinen Vergleich jedoch nicht weiter beeinträchtigte, da Cassandra von den Zeiten her , sowieso um längen geschlagen wird.

Folgende Ergebnisse ergaben die Abfragen bei 10-facher Durchführung via CQL-Shell.

Abfrage 1:



Abfrage 2:



Abfrage 3:



Abfrage 4:

