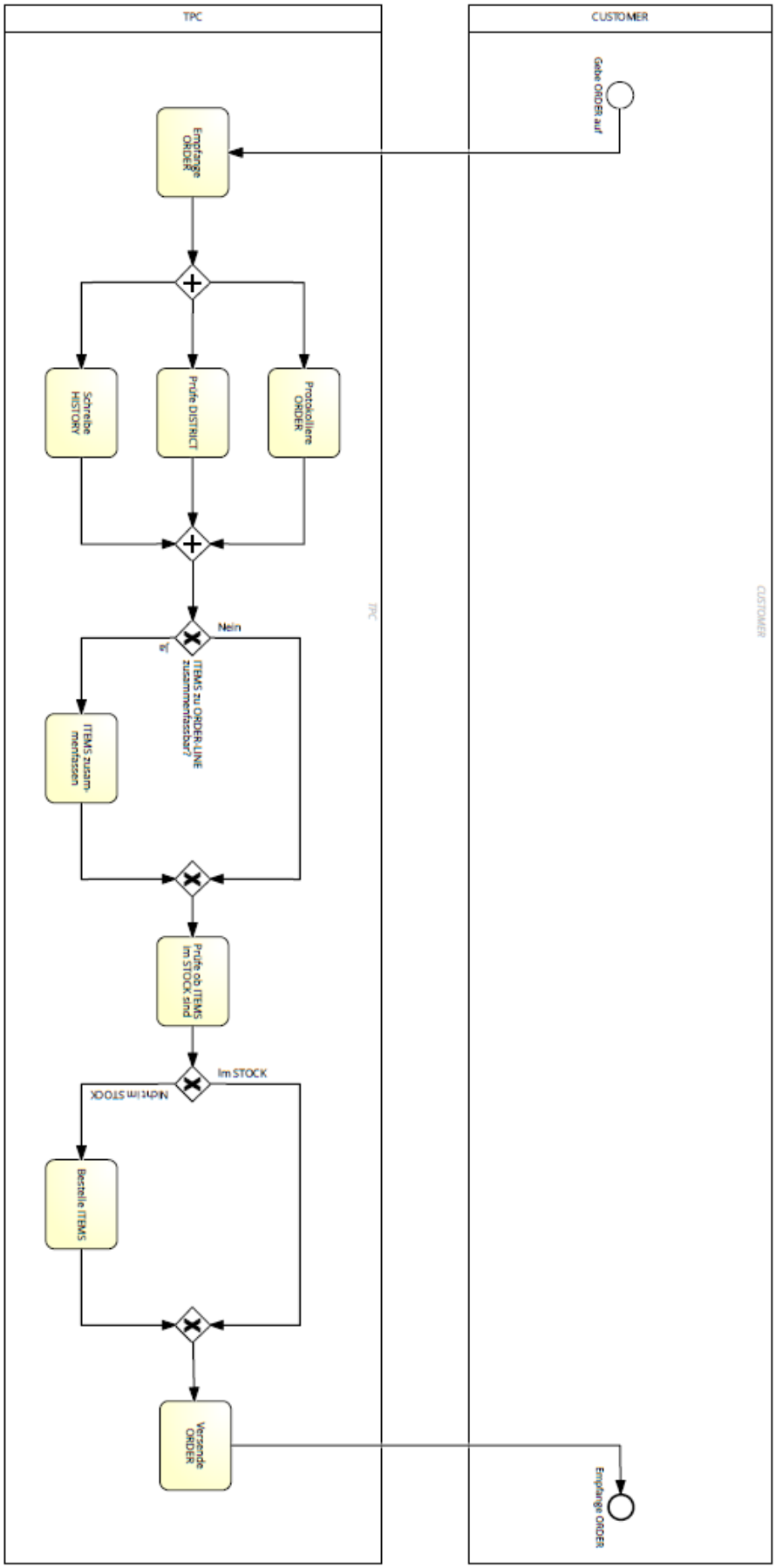
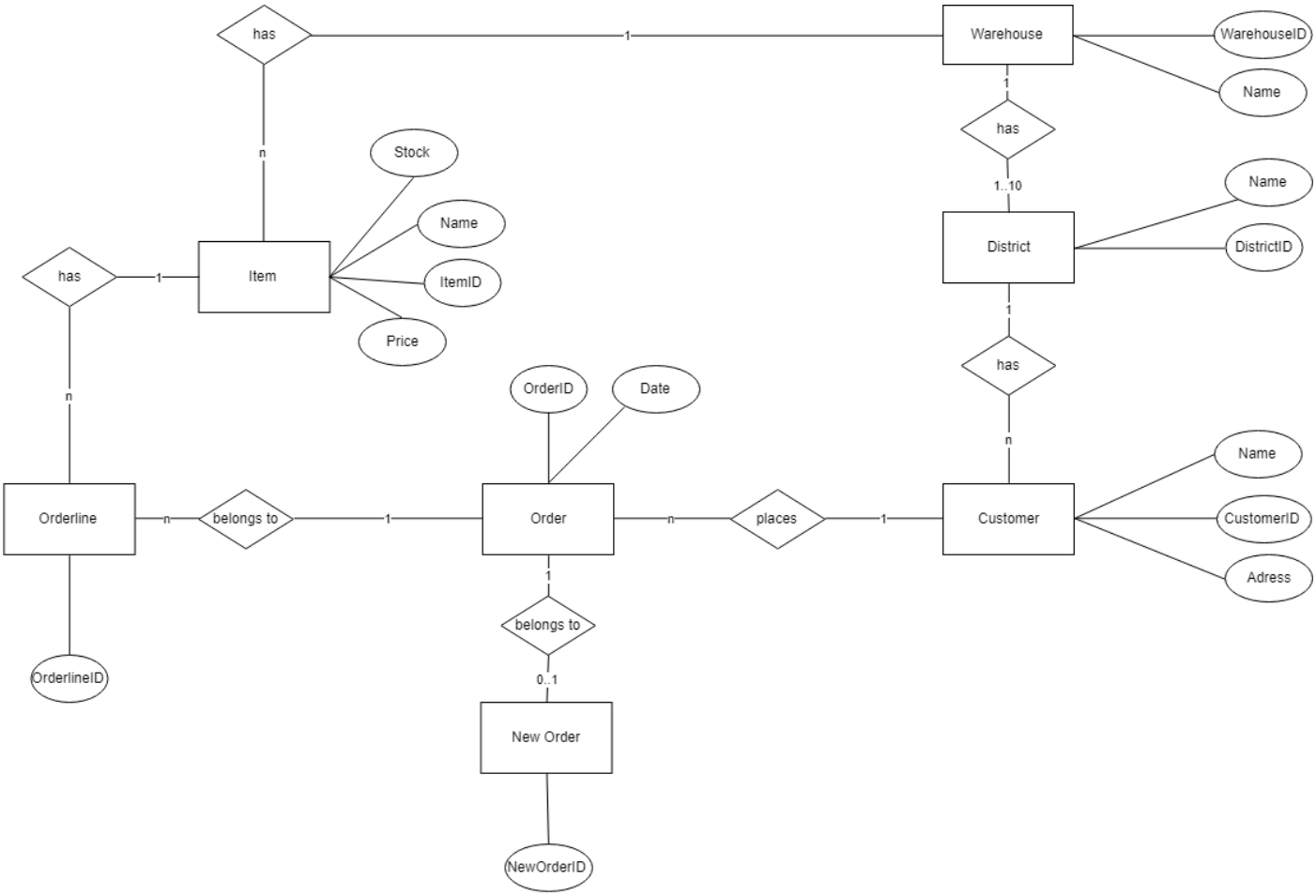
Jens Windisch 1526760, Markus Cöllen 1527307

Aufgabe 1



Jens Windisch 1526760, Markus Cöllen 1527307

Aufgabe 2 ER-Modell



Aufgabe 2 Impedance Mismatch

Problem	**Klasse**	**Tabelle**
Struktur	Attribute, Methoden	Spalten, Zeilen
Instanz	Objekte	-
Kapselung	Information Hiding	-
Identität	ID des Objekts	Primärschlüssel
Verarbeitungsmodell	Objektverarbeitung	Mengenverarbeitung
Wartung und Pflege	Entwickler	DB-Administrator

Jens Windisch 1526760, Markus Cöllen 1527307

Aufgabe 2 Nested Transaction

```
public class HibernateNestedTransactions {
   private static SessionFactory factory;
   public static void main(String[] args) {
      factory = new Configuration().configure("../resources/hibernate.cfg.xml").buildSessionFactory();
      Session session = factory.openSession();
      Transaction transaction1 = null;
      Transaction transaction2 = null;
      try {
         transaction1 = session.beginTransaction();
         Customer customer = new Customer("Igt Test", "Teststraße 1", new Date());
         session.saveOrUpdate(customer);
         CustomerOrder order = new CustomerOrder(customer, new Date(), 13.37f);
         transaction2 = session.beginTransaction();
         session.saveOrUpdate(order);
         transaction1.commit();
      } catch (Exception e) {
         if (transaction1 != null) {
            transaction1.rollback();
         }
         if (transaction2 != null) {
            transaction2.rollback();
         }
         e.printStackTrace();
      }
   }
}
```

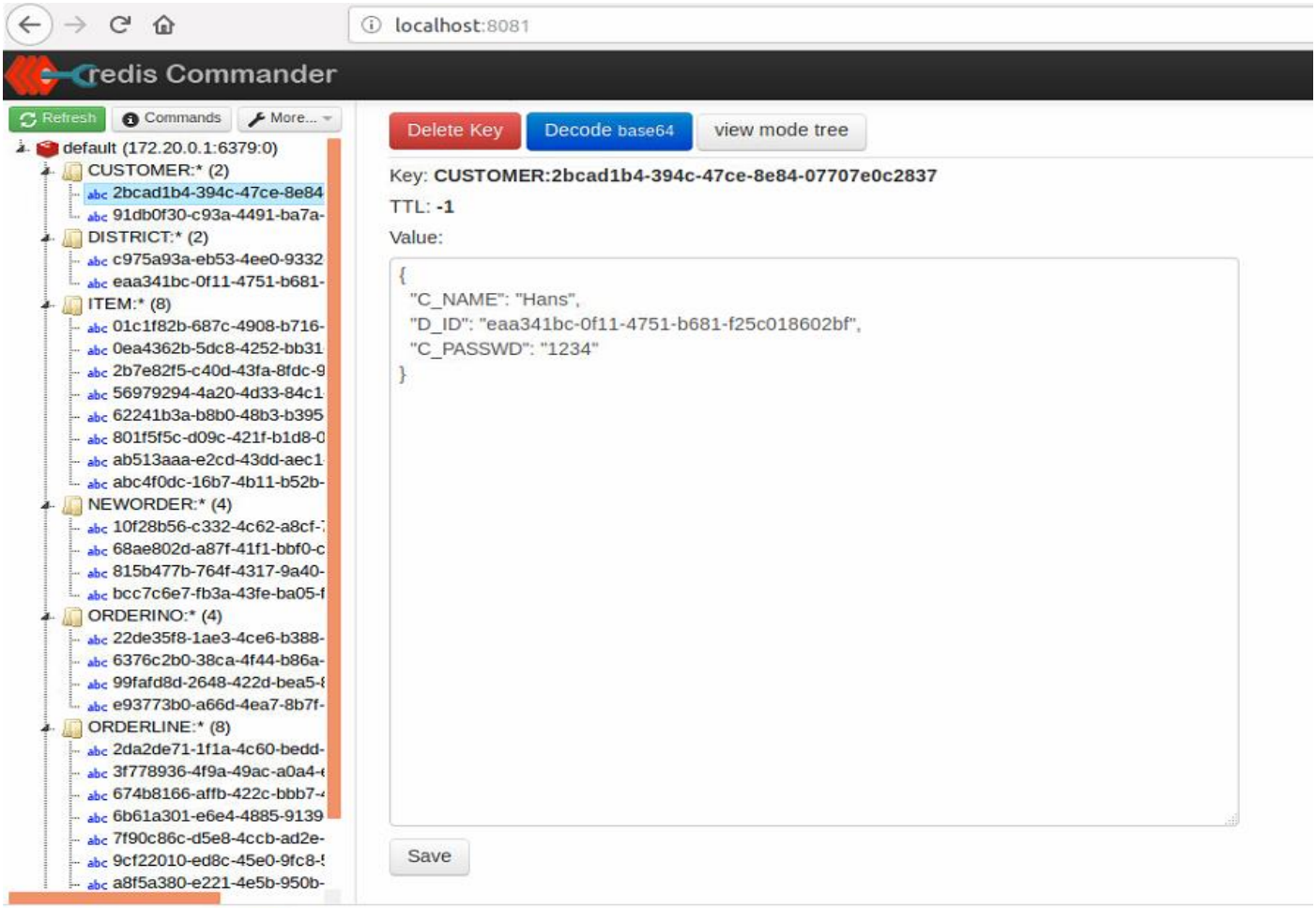
Exception:

```
java.lang.IllegalStateException: Transaction already active at
org.hibernate.engine.transaction.internal.TransactionImpl.begin(TransactionImpl.java:52)
at org.hibernate.internal.AbstractSharedSessionContract.beginTransaction(AbstractSharedSessionContract.java:409)
        at main.HibernateNestedTransactions.main(HibernateNestedTransactions.java:34)
```

Jens Windisch 1526760, Markus Cöllen 1527307

Aufgabe 3 Datenmodelle NoSQL

REDIS



CASSANDRA

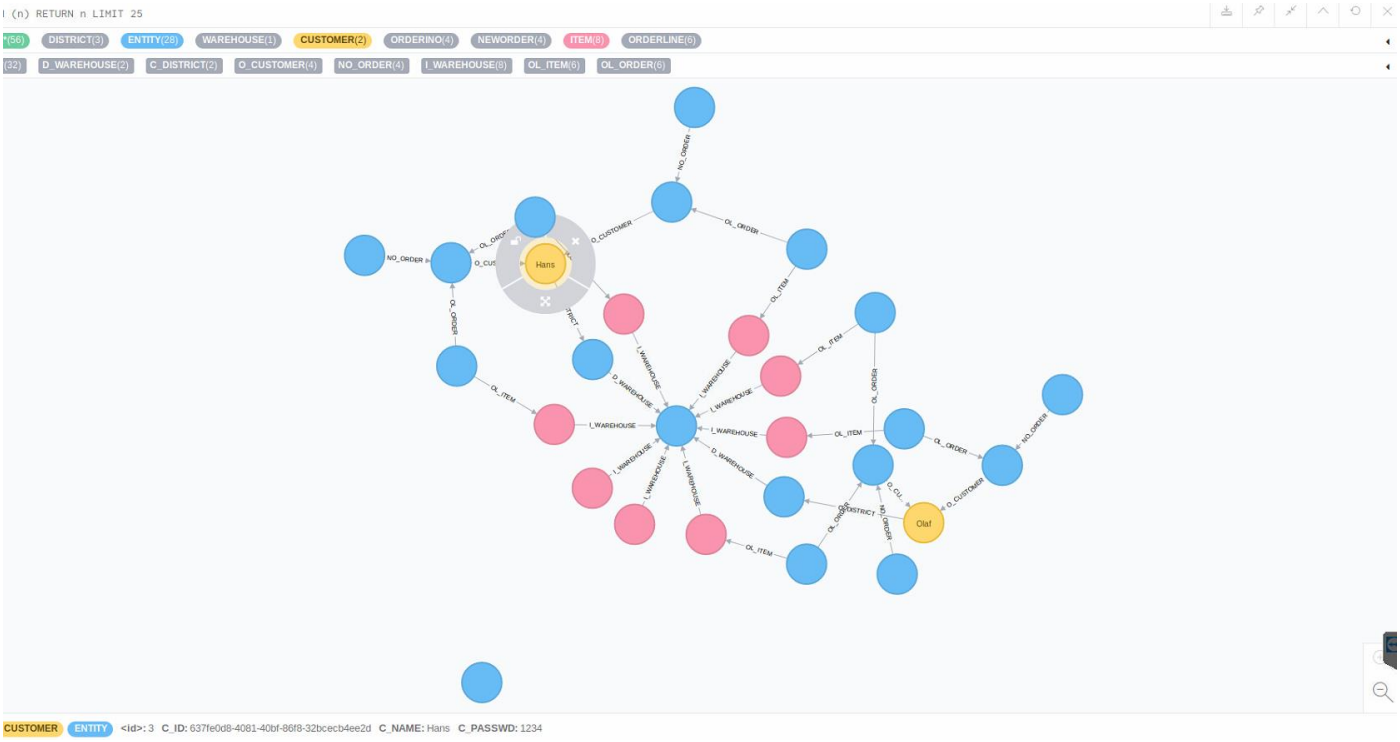
```
cqlsh> use ogm_hibernate;
cqlsh:ogm_hibernate> select * from "CUSTOMER";

 C_ID                                 | C_NAME | C_PASSWD | D_ID
--------------------------------------+--------+----------+--------------------------------------
 1b88c2d7-fbe6-457d-87b4-8bdfdd2c4f86 |   Olaf |     1234 | 3cb14313-989a-4351-9173-fe985cb67a80
 3edbb2cb-f068-4f79-a7de-7feae72bda3c |   Hans |     1234 | 5f8e2c8b-05e8-4bf8-a859-b6a591fa262d

(2 rows)
cqlsh:ogm_hibernate>
```

Jens Windisch 1526760, Markus Cöllen 1527307

Neo4j



MONGODB

```
MONGO DB
> db.CUSTOMER.find()
{ "_id" : "47a2da95-dc57-4b24-9213-8c91c9a0c33a", "C_NAME" : "Hans", "D_ID" : "c69d769a-96d2-4484-99f8-3b848777121a",
"C_PASSWD" : "1234" }
{ "_id" : "5ce59e5c-cd9b-4edf-90ce-014a91833a0b", "C_NAME" : "Olaf", "D_ID" : "e1156052-cf21-4c80-9734-8791f256e0bb",
"C_PASSWD" : "1234" }
```

Jens Windisch 1526760, Markus Cöllen 1527307

Aufgabe 3 Impedance Mismatch NoSQL

Problem	**Key-Value-Stores**	**Document-Store**	**Graphenbasiert**	**Spaltenorientiert**
Struktur	Tupel	JSON-Dokument	Knoten, Kanten	Spalten, Zeilen
Instanz	-	-	-	-
Kapselung	-	Geschachtelte Collections	-	-
Identität	ID	UUID	ID	Spalte + Zeile
Verarbeitungsmodell	Key-Abfrage	MapReduce	Iteration	Mengenverarbeitung
Wartung und Pflege	DB-Administrator	DB-Administrator	DB-Administrator	DB-Administrator

UUID = Universally Unique Identifier

Jens Windisch 1526760, Markus Cöllen 1527307

Aufgabe 4

Aufgabe 4.1

Mit einem Typ2-Hypervisor (z.B. Oracle Virtualbox) ist es möglich eine virtuelle Maschine innerhalb einer anderen zu starten.

Die gesamte Hardware wird durch Software virtualisiert, daher laufen quasi alle Prozesse des „Gastsystems“ in einem einzigen Prozess des „Hostsystems“. Je nachdem wieviel Ressourcen der ersten virtuellen Maschine zugewiesen werden ist es bis zur Erschöpfung dieser Ressourcen möglich, VMs in VMs zu starten

Aufgabe 4.2

Unsere Lösung ist eine Classic System VM, da Oracle Virtualbox eingesetzt wird.

Aufgabe 4.3

Ein Microservice stellt eine feingranulare Einheit eines Software-Projekts dar. Diese Einheit kann jederzeit von verschiedenen Diensten genutzt werden. Dank loser Kopplung kann der Service mit wenig Aufwand auf verschiedene Systeme redundant verbreitet werden oder ausgetauscht werden.

Implementierungsdetails werden nach außen verschleiert und sind nur durch exportierte Schnittstellen erreichbar

Im Vergleich mit Komponente oder Service ist ein Microservice vom Umfang her kleiner und grobgranularer

Aufgabe 4.4

Docker virtualisiert nicht die vollwertige Hardware eines Computers sowie ein komplettes Betriebssystem, sondern nutzt die Hardware des Host-Systems und Teile des Betriebssystems mit.

Es werden nur diverse Softwareschichten darübergelegt, welche Bibliotheken und Programme, welche nur in einem Container verwendet werden, bereitstellen. Über diese Schichten können gewisse Verzeichnisse bei Bedarf ein- oder ausgeblendet werden

Docker nutzt Bibliotheken wie libcontainer um mit dem Kernel des darunterliegenden Linux zu kommunizieren und auf die Computerhardware zuzugreifen

Ein Container bringt nur wenige spezifische Daten mit sich, wodurch die Images deutlich kleiner als virtuelle Festplatten-Dateien ausfallen