Wort vorab:
In Cassandra denkt man erst über die Queries, dann über die Tabelle nach. Man baut die Tabelle so, dass die Daten, die man haben will, über den Partition Key erreicht werden kann.

Außerdem muss beachtet werden, dass man ausschließlich nach Partition Keys Querien kann. Also eine WHERE muss mindestens einen Partition key enthalten.

Zum Thema Primary Key:
Der Primary Key setzt sich auch 2 teilen zusammen:

1. Der Partition Key, dieser gibt vor, wie Daten in Partitionen aufgesplittet werden.
2. Eine oder mehrere Clustering Columns.

Die Kombination aus allen Columns des Primary Key muss Unique sein. Wenn mein ein `INSERT` ausführt, ist es eigentlich eher ein `UPSERT`. Das heißt wenn man `INSERT` mit einem bereits vorhandenden Compound Key macht. (Alle werte der Columns des Primary Keys sind identisch). Dann wird eine `UPDATE` ausgeführt.

Zum starten einer Cassandra Instanz nutze ich Docker.

```
docker run --name cassandra-test -d -p 9042:9042 cassandra
```

Jetzt über das Docker Dashboard in die Shell des Containers gehen und dort dann die Cassandra Shell öffnen.

```
cqlsh
```

Zunächst erstellen wir einen Keyspace. In einem Keyspace können wir Tabellen zusammenfassen. Cassandra empfiehlt pro Anwendung einen Keyspace zu erstellen. Dann lassen wir uns den Keyspace anzeigen und anschließend nutzen wir den Keyspace für alle weiteren Befehle.

```
CREATE KEYSPACE hsbo WITH replication = {'class': 'SimpleStrategy',
'replication_factor': 1};

DESCRIBE KEYSPACE hsbo;

USE hsbo;
```

Als Beispiel für eine einfach Musik Playlist, legen wir eine Tabelle an. Wir nutzen einen Composit Partition Key (Aus `Singer` und `Year`) und als Clustering Column nehmen wir den SongName. Das heißt, dass jeder Singer und jedes Jahr eine eigene Partition bekommen. Dadurch können wir schnell zb. alle Songs eines Jahre oder alle Songs eines Singers bekommen. Innerhalb der Partition sind die Songs dann nach dem SongNamen sortiert. (Nicht zwingend aufsteigend oder absteigen, da die Daten gehasht werden.)

```

```

```
Create table MusicPlaylist
(
  SongName text,
  Year int,
  Singer text,
  Primary key((Singer, Year), SongName)
);

DESCRIBE MusicPlaylist;

INSERT INTO musicplaylist (songname, year, singer) VALUES ('Song 1', 2020, 'Singer
1');

INSERT INTO musicplaylist (songname, year, singer) VALUES ('Song 2', 2020, 'Singer
2');

INSERT INTO musicplaylist (songname, year, singer) VALUES ('Song 3', 2020, 'Singer
2');

SELECT * FROM musicplaylist;
```

Relations

One to one

Wenn wir uns jetzt angucken wollen, wie wir Relationen machen, dann schauen wir uns einfach an, wie wir die 3 traditionellen Relationen hinbekommen. Wichtig ist, dass es das Konzept von Foreign Keys in Cassandra nicht gibt. Das grundlegende Konzept hier ist die De-Normalisierung der Daten.
Also quasi genau das Gegenteil zu einer Relationalen Datenbank.

Wenn wir also eine 1 zu 1 Beziehung erstellen wollen, schreiben wir einfach die Daten in eine gemeinsame Tabelle, die beide Informationen enthält. Beispiel hier ist der Student und sein Course. Wir schreiben also in die Tabelle die Daten des Student (`rollno` und `name`) sowie die Daten des Kurses (`name`).

```
Create table Student_Course
(
  Student_rollno int primary key,
  Student_name text,
  Course_name text,
);


INSERT INTO hsbo.student_course (Student_rollno, Student_name, Course_name) VALUES
(1, 'Max Mustermann', 'E-Technik 1');

INSERT INTO hsbo.student_course (Student_rollno, Student_name, Course_name) VALUES
(2, 'Klaus Kleber', 'E-Technik 1');
```

```
SELECT * FROM student_course;
```

One to many

Bei einer 1-m Beziehung sieht das ganze nicht viel anders aus. Wir müssen lediglich den primary key ein wenig anders wählen, da dieser Unique sein muss.
Das Problem oben wäre, dass wir für eine rollno nur einen Studenten anlegen können. Deshalb erstellen wir jetzt einen Composit Key aus `rollno` und `course_name`. Das heißt, die Kombination aus beiden muss `UNIQUE` sein. Das erlaubt es uns jetzt für jeden Studierenden (Im traditionellen ist der Primary Key des Studierenden hier die `rollno`) mehrer Kurse anzulegen ohne das ein Studierender in einem Kurs zweimal sein kann.

```
Create table Student_Course
(
  Student_rollno int,
  Student_name text,
  Course_name text,
  Primary key(Student_rollno, Course_name), Student_name)
);

INSERT INTO hsbo.student_course (Student_rollno, Student_name, Course_name) VALUES
(1, 'Max Mustermann', 'E-Technik 1');

INSERT INTO hsbo.student_course (Student_rollno, Student_name, Course_name) VALUES
(1, 'Max Mustermann', 'E-Technik 2');

INSERT INTO hsbo.student_course (Student_rollno, Student_name, Course_name) VALUES
(2, 'Klaus Kleber', 'E-Technik 1');

SELECT * FROM student_course;

Select * FROM Student_Course WHERE Course_name='E-Technik 1' ALLOW FILTERING;
```

Many to many

Wenn wir eine n-m Beziehung erschaffen wollen, dann müssen wir zwei Tabellen anlegen um die Relation von beiden Seiten modellieren zu können. Wir erstellen also quasi zwei 1-m Beziehungen.

Die erste Tabelle `Student_Course` erlaubt die Verbindung von einem Studieren mit einem oder mehreren Kursen.

Die zweite Tabelle `Course_Student` speichert für jeden Kurs, die dazugehörigen Students.

```
Create table Student_Course
(
  Student_rollno int,
  Student_name text,
  Course_name text,
```

```
  Primary key(Student_rollno, Course_name)
);

--Mock Data
INSERT INTO student_course (student_rollno, student_name, course_name) VALUES (1,
'Max Mustermann', 'E-Technik 1');
INSERT INTO student_course (student_rollno, student_name, course_name) VALUES (1,
'Max Mustermann', 'E-Technik 2');
INSERT INTO student_course (student_rollno, student_name, course_name) VALUES (1,
'Max Mustermann', 'Informatik 1');
INSERT INTO student_course (student_rollno, student_name, course_name) VALUES (2,
'Klaus Kleber', 'Informatik 1');
INSERT INTO student_course (student_rollno, student_name, course_name) VALUES (2,
'Klaus Kleber', 'Informatik 2');

SELECT * FROM student_course;

-- Get all courses of student 1
Select * from Student_Course where student_rollno=1;

Create table Course_Student
(
  Course_name text,
  Student_name text,
  student_rollno int,
  Primary key(Course_name, Student_rollno)
);

INSERT INTO Course_Student (student_rollno, student_name, course_name) VALUES (1,
'Max Mustermann', 'E-Technik 1');
INSERT INTO Course_Student (student_rollno, student_name, course_name) VALUES (1,
'Max Mustermann', 'E-Technik 2');
INSERT INTO Course_Student (student_rollno, student_name, course_name) VALUES (1,
'Max Mustermann', 'Informatik 1');
INSERT INTO Course_Student (student_rollno, student_name, course_name) VALUES (2,
'Klaus Kleber', 'Informatik 1');
INSERT INTO Course_Student (student_rollno, student_name, course_name) VALUES (2,
'Klaus Kleber', 'Informatik 2');

--- Get all students of Informatik 1
Select * from Course_Student WHERE Course_name='Informatik 1';
```