# Erstellung und Evaluierung einer Datenbank für Kochrezepte

Praxisbericht im Studiengang

Bachelor Business Information Management

an der

Provadis - School of International Management and Technology

vorgelegt von

Florian Zoia

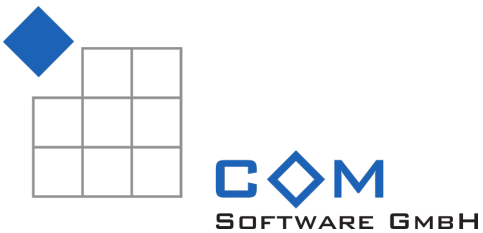
im Fach

Datenbanken und Datenmodellierung

Betreuer

Martin Gergeleit

Frankfurt am Main, 15. November 2020




# Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Thesis mit dem Titel

„Erstellung und Evaluierung einer Datenbank für Kochrezepte“

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Frankfurt am Main, den 15. November 2020

_____

Florian Zoia

# Inhaltsverzeichnis

**Selbstständigkeitserklärung** **II**

**Abbildungsverzeichnis** **IV**

**1 Einführung in das Thema** **1**
1.1 Einordnung in das Thema . . . 1
1.2 Leitfrage . . . 1
1.3 Ziel der Arbeit . . . 1
1.4 Aufbau der Arbeit . . . 1
1.5 Forschungsmethode . . . 2

**2 Grundlagen zur Bearbeitung** **3**
2.1 Datenbanken . . . 3
2.2 Relationale Datenbanken . . . 3
2.3 SQL . . . 4
2.4 ER-Diagramm . . . 5

**3 Einführung in die Erstellung der Datenbank** **6**
3.1 Erstellung des ER-Diagramms . . . 6
3.2 Erläuterung der einzelnen Queries . . . 7
3.2.1 Erstellung der Tabellen . . . 7
3.2.2 Einpflegen der Daten in die Datenbank . . . 10
3.2.3 Auslesen der Datensätze . . . 10

**4 Evaluation der Datenbank** **13**
4.1 Vor- und Nachteile . . . 13
4.2 Dritte und Vierte Normalform . . . 14

**5 Fazit** **16**

**Literatur** **V**

# Abbildungsverzeichnis


1 ER-Diagramm Kochrezepte (eigene Abbildung) . . . . . 7
2 Create Table 'Rezept' (eigene Abbildung) . . . . . 7
3 Create Table 'Kategorie' (eigene Abbildung) . . . . . 8
4 Create Table 'Zubereitung' (eigene Abbildung) . . . . . 8
5 Create Table 'Zwischentabelle Zubereitung-Rezept' (eigene Abbildung) . . 8
6 Create Table 'Zutaten' (eigene Abbildung) . . . . . 9
7 Create Table 'Einheit' (eigene Abbildung) . . . . . 9
8 Create Table 'Zwischentabelle Zutaten-Rezept' (eigene Abbildung) . . . . 9
9 Insert 'Rezept' (eigene Abbildung) . . . . . 10
10 Insert 'Zwischentabelle Zubereitung-Rezept' (eigene Abbildung) . . . . . 10
11 Select 'Zutaten' (eigene Abbildung) . . . . . 11
12 Datensätze der 11 (eigene Abbildung) . . . . . 11
13 Select 'Zubereitung' (eigene Abbildung) . . . . . 12
14 Datensätze der 13 (eigene Abbildung) . . . . . 12

# 1 Einführung in das Thema

Diese Arbeit behandelt das Thema 'Erstellung und Evaluierung einer Datenbank für Kochrezepte'. Dafür wurde eine Datenbank von Grund auf aufgebaut und im folgenden wird beschrieben, welche Schritte dabei durchlaufen wurden.

## 1.1 Einordnung in das Thema

Die Datenbank speichert die Daten für Kochrezepte. Sie kann die Zubereitung für die Rezepte sowie die Zutatenliste mit Mengenangaben ausgeben. Somit bietet sie die richtigen Voraussetzungen, um an eine Web Applikation angebunden zu werden. Insofern wäre es möglich eine Online Plattform zu erstellen, um Rezepte zu speichern und abzurufen. Dadurch könnten die Anwender Ihre Kochrezepte untereinander austauschen und verbreiten.
Um eine Datenbank zu erstellen muss erst die Architektur der Datenbank aufgestellt und ein Modell entworfen werden. Dies geschieht bei dieser Datenbank mit Hilfe eines ER-Modells. Daraufhin kann begonnen werden die Datenbank zu entwickeln. Hierfür wird die Datenbanksprache SQL verwendet. Die daraus resultierenden Abfragen werden im Laufe dieses Praxisberichts näher beschrieben und erläutert. Außerdem wird die Vor- und Nachteile der gewählten Datenstruktur eingegangen.

## 1.2 Leitfrage

Welche Architektur ist am besten geeignet um eine Datenbank für Kochrezepte zu entwickeln und welcher Aufwand muss für die Entwicklung aufgebracht werden ? Welche Vor- und Nachteile können aus der gewählten Architektur und der Umsetzung der Datenbank resultieren ?

## 1.3 Ziel der Arbeit

Diese Arbeit beschäftigt sich mit einer eigens für diese Arbeit erstellten Datenbank und ihrer Architektur. Dabei ist das Ziel dieser Arbeit die Architektur näher zu erläutern. Genauso wird aufgezeigt inwiefern die gewählten Strukturen Vor- und Nachteile aufwerfen.

## 1.4 Aufbau der Arbeit

Anfangs wird diese Arbeit die Grundlagen aufführen, welche benötigt werden, um die Arbeit zu verstehen. Darunter zählt die Datenbanksprache 'SQL', 'Relationale Datenbanken' und 'Datenbanken' als ganzes.
Darauf folgt die Einführung in die Datenbank für Kochrezepte. Hierbei wird der Prozess beschrieben, welcher bei der Erstellung der Datenbank durchlaufen wird. Unter diesen Punkt vereinigen sich die unterschiedlichen 'Queries', die für die Datenbank genutzt werden und die Erstellung des ER-Diagramms.
Danach kommt die Evaluation der Datenbank, welche die Vor und Nachteile der Datenbank aufzählt, sowie eine Beweisführung für die angewendete Normalform beinhaltet.

Der letzte Punkt ist das Fazit. Hierbei bezieht sich die Arbeit auf die Leitfrage und wirft einen letzten Rückblick auf die Erstellung und Funktionalität der Datenbank.

## 1.5 Forschungsmethode

Die Grundlagen der Arbeit wurden durch eine Literaturrecherche mit Hilfe der angegebenen Fachliteratur und einer Internetrecherche erarbeitet.

# 2 Grundlagen zur Bearbeitung

Für das Projektverständnis sind Grundlagen zu schaffen. Sowohl die Einordnung des Projektthemas als auch die unterschiedlichen Tools, Programmiersprachen, Informationssysteme und genutzten Methoden müssen verständlich erläutert sein, um den Gesamtkontext der Arbeit einordnen zu können.

## 2.1 Datenbanken

Datenbanken sind ein Instrument in der Informatik, welches genutzt wird um große Mengen an Daten zu speichern. Diese Daten werden dann von Applikationen oder Programmen aufgerufen. Man unterscheidet bei Datenbanken zwischen relationalen und nicht relationalen Datenbanken. Diese Arbeit befasst sich mit relationalen Datenbanken. Diese sind auf der Programmiersprache SQL aufgebaut.

## 2.2 Relationale Datenbanken

Relationale Datenbanken sollten so konstruiert werden, dass sie jegliche Redundanzen vermeiden. Sie bestehen aus mehreren Tabellen, welche die verschiedenen Entitäten der jeweiligen Datenbank darstellen.

Erstellt man zum Beispiel eine Datenbank für eine Schule hat man die Entitäten Lehrer, Fächer und viele weitere, wobei jede Entität ihre eigene Tabelle erhält. Diese Tabellen besitzen jeweils einen Primary Key, dieser ist ein einzigartiger Schlüssel den jeder Eintrag in der Tabelle erhält, um zwischen den verschiedenen Einträgen zu unterscheiden. Neben dem Primary Key gibt es auch die 'Foreign Key', welche den 'Primary Key' einer anderen Tabelle verweisen. Somit können Tabellen miteinander verbunden und die Daten die zusammengehören werden verlinkt.

Um Redundanzen zu vermeiden wird die Normalform auf Relationale Datenbanken angewendet. Diese besagt, dass Daten untereinander aufgeteilt werden müssen. Die Nullte Normalform einer Datenbank schaut so aus, dass die Daten grob separiert, die Datentypen allerdings vermischt sind. Außerdem ist alles in einer einzelnen Tabelle gespeichert. Dies ist die simpelste Form Daten abzuspeichern, allerdings ist diese Normalform eine unnormalisierte Relation und sehr anfällig für Redundanzen.

Für die Erste Normalform ist es notwendig, die Daten so aufzuteilen, dass die Wertebereiche der Relation atomar sind. Das bedeutet, dass die einzelnen Datentypen untereinander in verschiedene Spalten aufgeteilt sind. Allerdings sind die Daten weiterhin in einer einzelnen Relation gespeichert.

Bei der Zweiten Normalform wird die einzelne Relation in mehrere Tabellen aufgeteilt. Dies verteilt die Daten. Zwar müssen die Relationen untereinander verbunden werden, jedoch können mehrere Datensätze aus der einen Relation mit einem einzelnen Datensatz aus einer anderen Relation verbunden werden. Somit kommen noch weniger Redundanzen auf

und zum Beispiel werden bei einer Datenbank für Rechnungen die Rechnung und der Kunde klar aufgeteilt. So muss ein Kunde nicht bei jedem Einkauf neu angelegt werden.
(siehe Beratungsgesellschaft (2020))

Die Dritte Normalform teilt die vorhanden Relationen noch weiter auf. Hier werden die transitiven Spalten wieder in eine neue Relation ausgelagert. Somit sind die Daten, in ihrer normalen Form, soweit aufgeteilt wie möglich. Es gibt noch die Vierte und Fünfte Normalform. Diese werden allerdings hauptsächlich auf wissenschaftliche oder sehr präzise Datenbanken angewendet. Die Dritte Normalform wird für die meisten laufenden Datenbanken verwendet.
(vgl. Matthiessen und Unterstein (2007))

## 2.3 SQL

SQL steht für 'Structured Query Language' und wird genutzt, um auf relationale Datenbanken zuzugreifen. Die Hauptbestandteile von SQL bestehen aus den 'select' Abfragen, dem erstellen einer Tabelle mit 'create', dem Löschen von Daten aus einer Tabelle mit 'delete' und dem Hinzufügen von Daten in eine Tabelle mit 'insert'.

Das 'Select' ist eine Abfrage, die sich die entsprechenden Daten aus einer Tabelle zieht. Hierbei wird zunächst angegeben, welche Spalten ausgegeben und aus welcher Tabelle diese gezogen werden sollen. Dabei kann mit der 'where' Klausel nach bestimmten Daten gefiltert werden. Somit gibt die Datenbank nur die Daten ausgegeben, welche wirklich benötigt werden. Mit einem 'Join' können mehrere Tabellen miteinander verbunden werden. Diese Funktion sorgt dafür, dass weniger Abfragen gestartet werden müssen, um die Daten auszulesen.

Für die Ergebnismenge eines 'Select' Befehl können mehrere Tabellen miteinander verbunden werden, um die Datensätze aus den verschiedenen Tabellen zusammenzuführen. Dies funktioniert über den 'Join' Befehl. Es gibt vier verschiedene Arten von einem 'Join'.
Der 'Inner Join' verbindet zwei Tabellen miteinander und gibt alle Einträge aus, die übereinstimmende Werte haben. Dadurch werden automatisch die Einträge, welche keine zugehörigen Daten in der zweiten Tabelle haben aussortiert.
Im Gegensatz dazu gibt der 'Full Outer Join' alle Datensätze aus, unabhängig davon ob sie übereinstimmende Werte in der jeweils anderen Tabelle haben.

Der 'Left Join' gibt alle Datensätze aus der linken und die Datensätze mit übereinstimmenden Werten aus der rechten Tabelle aus.
Der 'Right Join' funktioniert wie der 'Left Join', nur dass hier alle Datensätze von der rechten Tabelle ausgegeben werden und nur die Datensätze mit übereinstimmenden Werten von der linken Tabelle.

Mit dem 'Create' Befehl erstellt SQL eine neue Tabelle in der Datenbank. Hier werden die Reihen einer Tabelle bestimmt und auch zugewiesen, welcher Datentyp innerhalb einer Reihe gespeichert werden darf. Des weiteren werden sowohl der 'Primary Key' als auch optional

der ’Foreign Key’ festgelegt. Diese verweisen auf die anderen Relationen.

Der ’Delete’ Befehl löscht Zeilen oder Werte aus einer bestimmten Tabelle. Dabei wird, wie bei dem ’Select’ Befehl, mit einer ’Where’ Bedingung gearbeitet, um nicht die gesamte Tabelle zu löschen.

Um Daten in die Tabellen einzupflegen wird der ’Insert’ Befehlt verwendet. Hierbei wird angegeben in welche Tabelle die Daten eingefügt werden sollen. Die Daten selbst müssen für das einfügen in der selben Reihenfolge eingespeichert werden wie die Tabelle aufgebaut ist. Sollte die ID die erste Spalte in der Tabelle sein, so muss bei einem ’Insert’ Befehl auch die ID als erstes angegeben werden.
(vgl. Geisler (2011))

## 2.4 ER-Diagramm

Ein Entity Relationship Modell zeigt die einzelnen Relationen auf und wie diese verbunden sind. Hier wird eine Relation aufgeteilt in den Namen der Tabelle und welche Spalten die Relation hat. Als erstes wird der ’Primary Key’ aufgelistet und danach kommen die restlichen Spalten. Die Fremdschlüssel und ’Primary Keys’ werden mit einem Pfeil verbunden. So ergibt sich ein Diagramm, welches die einzelnen Schlüssel, Verbindungen und Relationen aufzeigt.
(vgl. Gogolla (1994))

# 3 Einführung in die Erstellung der Datenbank

Für die Architektur der Datenbank wurde als Erstes ein ER-Diagramm erstellt, um die einzelnen Entitäten, Tabellen und Verbindungen darzustellen.Hierbei mussten die benötigten Daten beachtet werden, um ein Kochrezept darzustellen und inwiefern eine Redundanz entstehen kann. Dies darf in Relationalen Datenbanken nicht vorkommen und muss mit Hilfe von Zwischentabellen und einer gut aufgebauten Datenstruktur vermieden werden. Als Nächstes war es wichtig welche Datentypen innerhalb der einzelnen Spalten gespeichert werden sollen. Hierbei darf nicht vergessen werden was in der jeweiligen Spalte gespeichert werden soll, da einige Aspekte größere oder längere Werte haben als andere. Zum Beispiel benötigt die Zubereitung eines Rezepts mehr 'Characters' als der Name des Rezepts.
Nachdem das ER-Diagramm fertiggestellt war, konnten die einzelnen *Queries* entworfen werden. Diese beinhalten das Erstellen der verschiedenen Tabellen, das Einfügen der Datensätze in die Tabellen und die unterschiedlichen Abfragen der Daten. Dabei war es nötig sich an die Vorlage durch das ER-Diagramm zu halten, damit nicht doch eine Redundanz entsteht oder einzelne Tabellen nicht miteinander verlinkt sind.

## 3.1 Erstellung des ER-Diagramms

Um ein Rezept optimal darzustellen benötigt man den Namen, die Zutaten, die Menge der Zutaten und die Zubereitung. Damit diese Facetten am Besten aufgeteilt werden, wurde für jeden Aspekt des Rezepts eine Tabelle erstellt. Daraus entstanden die Tabellen *'Rezept'*, *'Zutaten'*, *'Einheit'* und *'Zubereitung'*. Da Rezepte untereinander noch in Kategorien wie 'Vorspeise', 'Hauptspeise' und 'Nachspeise' unterteilt werden können, kam die Tabelle *'Kategorien'* hinzu.
Um die ganzen Tabellen miteinander verbinden zu können wurden zwei weitere Tabellen benötigt, da es mehrere Tabellen mit einer *N* zu *N* Beziehung gibt. Damit diese Beziehungen in einer Relationalen Datenbank dargestellt werden können, werden Verbindungstabellen benötigt.
Diese sind dafür zuständig die Tabellen *'Zutaten'* und *'Einheit'* mit der Tabelle *'Rezept'* zu verbinden. Dies geschieht über die zugehörigen ID's. Die Tabelle *'Zutaten'* hat die Zutaten_ID, die Einheit_ID gehört zur *'Einheit'* Tabelle und in der *'Rezept'* Tabelle gibt es die Rezept_ID. Diese ID's sind die 'Primary Keys' der einzelnen Tabellen. Somit wird per 'Foreign Key' auf die unterschiedlichen Tabellen verwiesen und diese sind miteinander verbunden.
Die zweite Verbindungstabelle funktioniert genau wie die erste. Sie verbindet die Tabellen *'Zubereitung'* und *'Rezept'* miteinander.
Somit kommt man wie in Abbildung 1 auf eine Summe von sieben Tabellen, die wie folgt aussehen, um eine Relationale Datenbank für Kochrezepte zu erstellen.

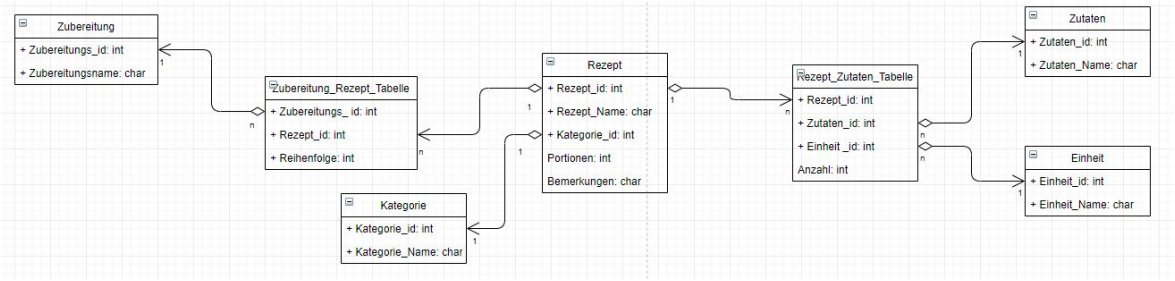



Abbildung 1: ER-Diagramm Kochrezepte (eigene Abbildung)

## 3.2 Erläuterung der einzelnen Queries

Für die Erstellung der Datenbank die Datenbanksprache ’SQL’ verwendet. Diese besteht aus ’Queries’ welche, sobald ausgeführt, die Tabellen erstellen, löschen oder lesen können. Zur Erstellung der einzelnen Tabellen wird für jede einzelne Tabelle eine eigene ’Query’ benötigt.

### 3.2.1 Erstellung der Tabellen

Die Tabelle *’Rezept’* hat, um die jeweiligen Rezepte zu unterscheiden, den ’Primary Key’ der Rezept_ID. Dieser wird auch genutzt um zwischen den einzelnen Tabellen eine Verbindung zu schaffen. Des Weiteren beinhaltet die Tabelle die grundsätzlichen Informationen zu den Rezepten, wie den Namen des Rezepts, wie viele Portionen das Rezept ergibt, zu welcher Kategorie das Rezept zählt. Also ob es eine Vorspeise, Hauptspeise oder ein Nachtisch ist. Als Letztes können Bemerkungen zu dem Rezept eingepflegt werden, was ebenfalls in der Tabelle *’Rezept’* gespeichert wird. Die zugehörige ’Query’ in SQL sieht folgendermaßen aus.

```
create table Rezept (
Rezept_id int,
Rezept_Name char(75),
Kategorie_id int,
Portionen int,
Bemerkungen char(255),
Primary key(Rezept_id),
Foreign Key (Kategorie_id) References Kategorie(Kategorie_id)
);
```

Abbildung 2: Create Table ’Rezept’ (eigene Abbildung)

Bei dieser ’Query’ muss gleich mit angegeben werden, welche ’Primary Keys’ und ’Foreign Key’ vorliegen sollen. In diesem Fall verweist der ’Foreign Key’ auf die Tabelle *’Kategorie’*.

Durch diesen Verweis wird eine Redundanz vermieden, da zum Beispiel mehrere Hauptspeisen in die Datenbank eingepflegt werden können. Somit hat die *'Rezept'* Tabelle den 'Foreign Key' Kategorie_ID erhalten. Dieser zeigt auf die Tabelle *'Kategorie'*.

```
create table Kategorie (
Kategorie_id int,
Kategorie_Name char(75),
Primary Key(Kategorie_id)
);
```

Abbildung 3: Create Table 'Kategorie' (eigene Abbildung)

Hier werden die Namen der Kategorien gespeichert. Diese lauten bis jetzt 'Vorspeise', 'Hauptspeise' und 'Nachspeise'. Durch den 'Primary Key' Kategorie_ID kann sich die Tabelle *'Rezept'* die zum Rezept zugehörige Kategorie aus der Tabelle ziehen.
Die Tabelle *'Zubereitung'* beinhaltet die Zubereitungs_ID und die Beschreibung der jeweiligen Zubereitung. Da hier allerdings eine *N* zu *N* Beziehung zwischen den Tabellen *'Zubereitung'* und *'Rezept'* entsteht wird eine Zwischentabelle benötigt. Diese beinhaltet die Zubereitungs_ID und die Rezept_ID, um die einzelnen Schritte der Zubereitung dem Rezept zuzuordnen. Damit diese auch noch in der richtigen Reihenfolge ausgegeben werden können, hat die Tabelle die Spalte 'Reihenfolge'. Somit können die Tabellen *'Zubereitung'* und *'Rezept'* miteinander verbunden werden.

```
create table Zubereitung (
Zubereitungs_id int,
Zubereitung char(255),
Primary Key(Zubereitungs_id)
);
```

Abbildung 4: Create Table 'Zubereitung' (eigene Abbildung)

```
create table Zubereitung_Rezept_Tabelle(
Zubereitungs_id int,
Rezept_id int,
Reihenfolge int,
Foreign Key (Zubereitungs_id) References Zubereitung(Zubereitungs_id),
Foreign Key (Rezept_id) References Rezept(Rezept_id)
);
```

Abbildung 5: Create Table 'Zwischentabelle Zubereitung-Rezept' (eigene Abbildung)

Um die Zutaten sowie die zugehörige Einheit abspeichern zu können, musste dasselbe wie bei den vorherigen Tabellen angewendet werden. Um eine Redundanz bei den Einheiten zu vermeiden, wurden diese in eine eigene Tabelle ausgelagert. Diese ist über die ID's mit der Tabelle *'Zutaten'* verlinkt. Des Weiteren sind in der *'Einheit'* Tabelle noch der Name der

jeweiligen Einheit, wie 'Milliliter' oder 'Gramm' abgespeichert.
In der Tabelle *'Zutaten'* wird die Zutaten_ID, als 'Primary Key' und der Name der Zutat gespeichert.

```
create table Zutaten (
Zutaten_id int,
Zutaten_Name char(75),
Primary Key (Zutaten_id)
);
```

Abbildung 6: Create Table 'Zutaten' (eigene Abbildung)

```
create table Einheit (
Einheit_id int,
Einheit_Name char(75),
Primary Key (Einheit_id)
);
```

Abbildung 7: Create Table 'Einheit' (eigene Abbildung)

Da zwischen den beiden Tabellen jedoch eine *N* zu *N* Beziehung entsteht, wird ebenfalls eine Zwischentabelle benötigt. Diese beinhaltet die jeweiligen 'Primary Keys' der verlinkten Tabellen. Also die Rezept_ID, die Zutaten_ID und die Einheit_ID. Dadurch sind die Tabellen *'Zutaten'*, *'Rezept'* und *'Einheit'* miteinander verbunden und es können die Zutaten eines Rezepts abgerufen werden. Genauso beinhaltet die Zwischentabelle die Anzahl, damit es ersichtlich wird wie viel man von welcher Zutat benötigt.

```
create table Rezept_Zutaten_Tabelle (
Rezept_id int,
Zutaten_id int,
Einheiten_id int,
Anzahl int,
Foreign Key (Rezept_id) References Rezept(Rezept_id),
Foreign Key (Zutaten_id) References Zutaten(Zutaten_id),
Foreign Key (Einheiten_id) References Einheit(Einheit_id)
);
```

Abbildung 8: Create Table 'Zwischentabelle Zutaten-Rezept' (eigene Abbildung)

Für die Erstellung der einzelnen Relationen ist es notwendig die richtige Reihenfolge für das Ausführen der Befehle beizubehalten. Es können keine Relationen mit Fremdschlüsseln erstellt werden, deren Verweis noch nicht existiert. Im Falle dieser Datenbank bedeutet das, dass die Tabelle *'Rezept'* als erstes erstellt werden kann, jedoch die Tabelle *'Zutaten'* nicht. Diese verweist auf eine Zwischentabelle und ohne diese kann die Tabelle nicht erstellt werden.

### 3.2.2 Einpflegen der Daten in die Datenbank

Damit Daten in die Datenbank eingepflegt werden können besitzt SQL den *'insert into'* Befehl. Dieser ist so aufgebaut, dass man die Tabelle angibt in welche man die Daten einpflegen möchte und daraufhin in derselben Reihenfolge, wie die Tabelle aufgebaut ist, die zu erfassenden Daten angibt.
Um ein Rezept für das Nudelgericht 'Aglio e Olio' in die Datenbank einzupflegen sehen die jeweiligen 'Queries wie folgt aus.

```
insert into rezept
values (2, 'Aglio e Olio', 2, 2, 'Leckeres, einfaches und schnelles Nudelgericht');
```

Abbildung 9: Insert 'Rezept' (eigene Abbildung)

Diese 'Query' pflegt die grundsätzlichen Informationen des Rezepts in die Tabelle ein. Die Tabelle ist so aufgebaut, dass die Rezept_ID an erster Stelle steht, darauf folgt der Name des Rezepts und die Kategorie_ID. Die letzten beiden Spalten sind die Portionen, die bei den angegebenen Mengen raus kommen und eine kleine Beschreibung.
Bei Zwischentabellen sieht eine 'Query' für das Einpflegen der Daten etwas anders aus. Diese Tabellen bestehen zum Großteil aus 'Foreign Key' und ID's, weshalb die Datensätze meistens nur Zahlen sind.

```
insert into zubereitung_rezept_tabelle
values (12, 2, 1), (13, 2, 2), (14, 2, 3), (15, 2, 4), (16, 2, 5), (17, 2, 6), (21, 2, 7), (18, 2, 8), (19, 2, 9), (20, 2, 10);
```

Abbildung 10: Insert 'Zwischentabelle Zubereitung-Rezept' (eigene Abbildung)

Bei dieser Tabelle, die die beiden Tabellen *'Rezept'* und *'Zubereitung'* miteinander verbindet, wird keine Spalte am Ende ausgegeben, da sie nur für die Verbindung der Tabellen und die richtige Reihenfolge bei der Ausgabe der Datensätze zuständig ist. Die erste Spalte speichert die Zubereitungs_ID, die zweite Spalte beinhaltet die Rezept_ID und als Letztes wird die Reihenfolge gespeichert. Diese sorgt dafür, dass die einzelnen Schritte der Zubereitung eines Rezepts in der richtigen Reihenfolge ausgegeben werden können.

### 3.2.3 Auslesen der Datensätze

Sobald Daten in der Datenbank eingepflegt sind kann man diese auslesen. Bei SQL funktioniert dies mit dem 'Select' Befehl. Dieser ist so aufgebaut, dass entweder jede Spalte ausgelesen wird oder nur ausgewählte, die vorher angegeben werden. Im Regelfall werden bei einer Datenbank, die hinter einer Applikation steht nicht alle Spalten ausgelesen, sondern nur diejenigen, welche benötigt werden. In diesem Fall ist das zum Einen die Zubereitung, welche ausgelesen werden soll und die Zutaten.
Für die Zutaten müssen die Tabellen *'Rezept'*, *'Zutaten'*, *Einheit* und die zugehörige Zwischentabelle zusammengeführt werden. Dies funktioniert am besten über einen 'Inner Join', da hier nur die übereinstimmenden Daten zusammengeführt werden. Somit kann über die

Rezept_ID mit Hilfe des 'Where' Befehls auf ein bestimmtes Rezept gefiltert werden. Es werden nicht alle Spalten, die bei der Zusammenführung der Tabellen entstehen, benötigt. Aus diesem Grund liest diese 'Query' nur die Spalten 'Rezept_Name', 'Portionen', 'Zutaten_Name', 'Anzahl' und 'Einheit_Name' aus.

```
# Select auf die Zutaten für das Rezept
select Rezept_Name, Portionen, Zutaten_Name, Anzahl, Einheit_name from rezept
inner join rezept_zutaten_tabelle on Rezept.rezept_id = Rezept_Zutaten_tabelle.rezept_id
inner join Zutaten on rezept_zutaten_tabelle.Zutaten_id = zutaten.Zutaten_id
inner join Einheit on rezept_zutaten_tabelle.Einheiten_id = Einheit.Einheit_id
inner join kategorie on kategorie.Kategorie_id = rezept.kategorie_id
where Rezept.Rezept_id = 2;
```

Abbildung 11: Select 'Zutaten' (eigene Abbildung)

Diese 'Query' liest folgende Datensätze aus der Datenbank aus:

| Rezept_Name | Portionen | Zutaten_Name | Anzahl | Einheit_name |
|---|---|---|---|---|
| Aglio e Olio | 2 | Knoblauch Kopf | 1 | Kopf |
| Aglio e Olio | 2 | Petersilie | 100 | Gramm |
| Aglio e Olio | 2 | Oliven Öl | 118 | Milliliter |
| Aglio e Olio | 2 | Chiliflocken | 1 | Teelöffel |
| Aglio e Olio | 2 | Nudeln | 226 | Gramm |
| Aglio e Olio | 2 | Zitrone | 1 | |
| Aglio e Olio | 2 | grobe Salzflocken | 0 | |

Abbildung 12: Datensätze der 11 (eigene Abbildung)

Sollte die Datenbank an eine Applikation oder ein Programm angebunden werden, so kann diese 'Query' genutzt werden, um die Zutaten darzustellen.
Um die Datensätze für die Zubereitung eines Kochrezepts ausgeben zu können schaut die Vorgehensweise ziemlich ähnlich aus. Dabei werden die Tabellen *'Rezept'*, *'Zubereitung'* und die zugehörige Zwischentabelle zusammengeführt. Der große Unterschied hierbei liegt darin, dass die Datensätze noch in einer bestimmten Reihenfolge ausgegeben werden müssen. Aus diesem Grund beinhaltet die 'Query' für die Zubereitung noch den Befehl 'Order By'. Dadurch wird gewährleistet, dass bei jeder Datenabfrage die richtige Reihenfolge ausgegeben wird und die Daten sortiert sind.

```
#Select auf Zubereitung
select Rezept_Name, Portionen, zubereitung from rezept
inner join zubereitung_rezept_tabelle on zubereitung_rezept_tabelle.rezept_id = rezept.rezept_id
inner join zubereitung on zubereitung.zubereitungs_id = zubereitung_rezept_tabelle.zubereitungs_id
where rezept.rezept_id = 2
order by Reihenfolge;
```

Abbildung 13: Select ’Zubereitung’ (eigene Abbildung)

Aus der oben abgebildeten ’Query’ resultieren die folgenden Datensätze:

| | Rezept_Name | Portionen | zubereitung |
|---|---|---|---|
| ▶ | Aglio e Olio | 2 | Wasser aufkochen und salzen |
| | Aglio e Olio | 2 | Öl erhitzen |
| | Aglio e Olio | 2 | Knoblauch anbraten |
| | Aglio e Olio | 2 | Chiliflocken hinzufügen |
| | Aglio e Olio | 2 | Nudeln in die Pfanne geben |
| | Aglio e Olio | 2 | Pastawasser hinzugeben |
| | Aglio e Olio | 2 | köcheln lassen |
| | Aglio e Olio | 2 | Zitronensaft drüber geben |
| | Aglio e Olio | 2 | Petersilie hinzugeben |
| | Aglio e Olio | 2 | Mit Salz und Pfeffer abschmecken |

Abbildung 14: Datensätze der 13 (eigene Abbildung)

# 4 Evaluation der Datenbank

Um die in dieser Arbeit vorgestellte Datenbank ordnungsgemäß zu evaluieren, werde ich auf deren Vor- und Nachteile eingehen. Genauso wird im folgenden noch einmal auf die Dritte und Vierte Normalform eingegangen.

## 4.1 Vor- und Nachteile

Durch die vorgestellte Architektur der Datenbank und die Anwendung der Dritten Normalform hat die vorliegende Datenbank eine sehr geringe Redundanz. Die transitiven Spalten wurden soweit aufgeteilt, dass eine Wiederholung innerhalb der Datenbank nur vorliegen kann, wenn deren Nutzer Rezepte doppelt einspeichern. Dies kann jedoch durch die anzubindende Applikation unterbunden werden, indem vor jedem 'Insert' eine Abfrage gestartet wird ob dieses Rezept schon vorhanden ist.
Des Weiteren ist eine Vermeidung von Datenbank Anomalien gewährleistet. Dies ist durch die Anwendung der Dritten Normalform gegeben. Anomalien entstehen durch ein fehlerhaftes Datenbankdesign und sorgen dafür, dass zum Beispiel bei einem 'Insert' nicht alle Daten gespeichert oder inkonsistent werden. Bei dem 'Update' Befehl zeigt sich eine Anomalie, indem die Daten nicht geändert werden und der 'Delete' Befehl löscht bei einer Datenbank Anomalie zu viele Daten aus den Relationen, da diese fälschlicherweise miteinander verbunden wurden.
Außerdem ist durch den recht simplen Aufbau der Datenbank eine einfache Anbindung an Applikationen gewährleistet. Zum einen könnte die Datenbank mit Hilfe eines 'REST Services' an die entsprechende Applikation angebunden werden. Dadurch würde der Rest Service die Funktionen der Datenbank aufrufen. Innerhalb dieser Funktionen wären die 'Queries' gespeichert und, je nachdem was benötigt wird ein 'Insert', 'Select' oder 'Delete' ausführen. Beim 'Select' würden die ausgelesenen Datensätze an den 'REST Service' gesendet werden und dieser leitet die Datensätze weiter an die angebundene Applikation.
Der Aufbau der Datenbank ermöglicht es außerdem, dass diese ohne Probleme erweiterbar ist. Sollten zum Beispiel Rezepte für 'Snacks' oder 'warme' beziehungsweise 'kalte Vorspeisen' gespeichert werden wollen, so kann man die Datenbank einfach mit einem 'Insert' Befehl erweitern. Dabei müssen keine neuen Tabellen oder Spalten hinzugefügt werden. Der Aufbau bietet den perfekten Rahmen, um weitere Rezepte in die Datenbank einzupflegen.

Der größte Nachteil der vorgestellten Datenbank ist ihr langsames Laufzeitverhalten. Durch die Dritte Normalform ist die Datenbank auf sehr viele Relationen aufgeteilt. Es gibt zwei Relationen deren einzige Aufgabe es ist die anderen Relationen zu Verbinden. Diese Zwischentabellen müssen die anderen Relationen über ein 'Join' verbinden. Soll die Zutatenliste ausgegeben werden, so gibt müssen insgesamt vier Tabellen miteinander verbunden werden. Dadurch läuft die Datenbank wesentlich langsamer, als wären alle Datensätze in nur einer Relation gespeichert. Jedoch wird so jegliche Redundanz vermieden und es gibt weniger Datensätze über die, die Datenbank iterieren muss.
Dazu kommt, dass die meisten Verbindungen zwischen den Relationen *1 zu N* Beziehungen sind. Dies verlangsamt die Laufzeit der Datenbank um ein weiteres.
Ein weiterer Nachteil liegt darin, dass ein hoher Normalisierungsaufwand nötig war um die

Architektur der Datenbank aufzustellen. Die einzelnen Relationen mussten im laufe der Architektur mehrfach neu aufgestellt und sogar weiter aufgeteilt werden. Dies führt dazu, dass die Erstellung der Datenbank weitaus länger gedauert hat, als wenn die ganzen Datensätze in einer einzelnen, simple Tabelle gespeichert wären.

Dennoch ist es Sinnvoll eine Datenbank zu normalisieren. Datenbanken befinden sich in einem ständigen Wachstum, da immer mehr Daten eingepflegt werden. Sollte eine stetig wachsende, große Datenbank nicht normalisiert sein, ist der Wartungsaufwand schier unmöglich und auch nicht lesbar. Dies kommt daher, dass die Datenbank alle Daten in wenigen, wenn nicht in einer Relation speichert, sollte diese nicht normalisiert sein. Dadurch entsteht zum einen eine sehr hohe Redundanz und zum Anderen kann man händisch die Daten nicht auslesen und eine einzelne Abfrage auf die Datenbank wird unverständlich groß.

## 4.2 Dritte und Vierte Normalform

Für die Vierte Normalform von Datenbanken ist es notwendig jegliche mehrwertige Abhängigkeiten innerhalb der Relationen zu entfernen. Sodass eine Relation nur aus Datensätzen besteht, die eine funktionale Abhängigkeit aufweisen. In der Dritten Normalform ist es gestattet, dass die Relationen noch Datensätze mit trivialen Werten beinhalten. Jegliche mehrwertige Abhängigkeit aus Relationen zu entfernen bedeutet, dass alle Spalten, die trivial sind und mit einer weiteren Spalte beliebig ausgetauscht werden können, ausgelagert werden. Dadurch entstehen mehr Relationen durch die Auslagerung der Spalten und es müssen noch Zwischentabellen erstellt werden, die die genannten Relationen verbinden.
Für die Datenbank, welche in dieser Arbeit behandelt wird, ist es nicht notwendig die Datensätze so weit aufzuteilen, dass nur noch funktionale Abhängigkeiten enthalten sind. Durch das weitere Aufteilen der Datensätze entsteht ein größerer Mehraufwand bei der Erstellung der Datenbank. Dieser lohnt sich in den meisten Fällen gar nicht, da die Daten zwar mehr Integrität haben und die Datenbank auch weniger Redundanz aufweist, jedoch eine schlechtere Performance und mehr administrativer Aufwand entsteht. Dieser beinhaltet, dass die Datenbank mehr Speicherplatz benötigt.

Aus den sieben Tabellen, die aus der Dritten Normalform entstanden sind würden in der Vierten Normalform Zwölf oder mehr Tabellen werden. Diese Datenbank würde wesentlich mehr Aufwand benötigen, um gewartet und gepflegt zu werden.
Sollte eine Applikation an die Datenbank angebunden werden ist es in der Dritten Normalform einfacher Funktionen für den 'Select' und 'Insert' Befehl zu schreiben. Diese benötigen weniger Aufwand erstellt zu werden, da weniger 'Joins' darin vorkommen. Das Selbe gilt für den 'Insert' Befehl, da hier für jeden neuen Eintrag, also jedes neue Rezept, mehrere 'Insert' Befehle benötigt werden. Es müssen in der vorliegenden Datenbank fünf 'Inserts' ausgeführt werden, um ein ganzes Rezept in die Datenbank einzuspeichern. Würde man die Vierte Normalform auf die Datenbank anwenden, so würde die Anzahl der benötigten 'Insert' Befehle enorm ansteigen. Dadurch müssten mehr Anfragen an die Datenbank geschickt werden und die angebundene Applikation würde ebenfalls langsamer laufen, wenn die Vierte Normalform auf die Datenbank angewendet werden würde.
In der Praxis kommt die Vierte Normalform selten auf, da sie ihren aufwand teilweise nicht

Wert ist. Dennoch findet sie in der Wissenschaft noch Anwendung, genauso in Datenbanken, die nur wenig Redundanz aufweisen dürfen. Allerdings wäre sie für diese Datenbank, wie vorher erwähnt, nicht geeignet und wurde deshalb auch nicht angewandt.
(vgl. Schicker (2017))

# 5 Fazit

Die Leitfrage der Arbeit befasst sich mit dem aufgebrachten Aufwand und welche Vor- und Nachteile aus dem gewählten Datenbank Konzept entstehen. Auf die Vor- und Nachteile wurde in der Evaluation eingegangen und der folgende Abschnitt bezieht sich auf den aufgetretenen Aufwand.
Der entstandene Aufwand eine Datenbank, die Kochrezepte speichert, ist ein vergleichsweise geringer Aufwand.
Der größte Aufwand entsteht bei dem Entwurf der einzelnen Relationen. Hier muss darauf geachtet werden, dass die Dritte Normalform eingehalten wird. Dies hat zur Folge, dass der erste Entwurf meistens komplett neu überarbeitet werden muss, um die ganzen transitiven Daten auf neue Relationen aufzuteilen. Dabei kann es sich, wie in diesem Fall um die 'Einheit' der Datenbanken handeln oder auch die 'Kategorie'. Beides Relationen in der vorliegenden Datenbank, welche nun in einzelnen Tabellen gespeichert sind.
Für die Erstellung der 'Queries' und der Datenbank selbst, ist der Aufwand recht simpel. Dies hängt ganz Individuell von dem Wissensstand der jeweiligen Person ab. Je nachdem wie gut der/die Entwickler/In mit der Datenbanksprache 'SQL' vertraut ist, wird ein anderer Zeitaufwand benötigt.

Rückblickend wäre es effizienter gewesen erst die Architektur der Datenbank aufzustellen und danach mit dem Erstellen des ER-Diagramms anzufangen. In dieser Arbeit wurden beide Schritte anfangs vereint, sodass ein höherer Zeitaufwand entstanden ist. Das ER-Diagramm musste mehrfach neu erstellt werden. Des weiteren könnte man die Relationen so aufteilen, dass weniger 'Joins' benötigt werden. Somit hätte die Datenbank eine bessere Laufzeit und könnte schneller Daten an eine angebundene Applikation schicken.

# Literatur

Beratungsgesellschaft, Begerow (2020): Datenbanken Verstehen.

Geisler, Frank (2011): *Datenbanken Grunlagen und Design*. Verlagsgruppe Hüthig Jehle Rehm GmbH.

Gogolla, Martin (1994): *Grundlagen von Entity-Relationship-Modellen*.

Matthiessen, Günter und Michael Unterstein (Dez. 2007): *Relationale Datenbanken und Standard-SQL*. Addison-Wesley Verlag, München u.a., 4. Aufl.

Schicker, Edwin (2017): *Datenbankendesign*. Springer Verlag.