

# Projektaufgabe Vegi-Bio-Lieferservice

Datenbanken (EDV-Nr. 113222)

Wintersemester 2022/23

Stuttgart, den 14.01.2023

# Sachverzeichnis

## 1. Systembeschreibung

## 2. Entity Relationship

### 2.1 Logisch

### 2.2 Physisch

### 2.3 Dokumentation der Entitätstypen

## 3. Skripte

### 3.1 DDL

### 3.2 Trigger und Funktion

# 1. Systembeschreibung

Das Datenbankmodell ist, gemäß der Aufgabe, gedacht, einen BIO-Vegi-Lieferservice zu unterstützen. Beachtet wurde, dass es dem Kunden möglich sein soll, sowohl selbst Rezepte zu erstellen als auch aus bestehenden Rezepten verschiedene auszusuchen. Die Menge der gelieferten Zutaten wird durch die Anzahl an bestellten Portionen (Eine Portion = Eine Person) bestimmt. Die Lieferung erfolgt sowohl über Bio-Lebensmittelhändler als auch Bio-Standard zertifizierten lokalen Hofläden und Bio zertifizierten Einzelhändlern. Bewusst wurde bei einem Lieferservice auf einen Unverpackt-Laden als Händler Art verzichtet. Da die Aufgabe nur ein von uns gewählter Ausschnitt einer Datenbank ist, wurde auf Zahlungsmethoden wie Paypal oder Lastschrift als eigene Tabelle verzichtet. Deswegen ist an keiner Stelle etwas zum Bezahlvorgang zu finden.

Zwei grundlegende Abschnitte strukturieren das gesamte Datenbank-Modell, die Rezepte und deren Bestellungen. Die Systembeschreibung geht zuerst auf die Zutaten wie folgt ein. Als wichtige Grunderkenntnis gilt jedoch noch zu erwähnen, dass die Besonderheit des Lieferservices die Herkunft und Qualität der Zutaten ist. Dieser Mehrwert lässt den BIO-Vegi-Lieferservice vom Wettbewerb abheben und führt zur Schlussfolgerung, dass, am Sinnvollsten dargestellt, der Ansatz zur Strukturierung des Datenmodells auf der Verwaltung der Zutaten sorgfältigst und in großem Detail wiedergegeben wird. Alle Zutaten die der Lieferservice verwaltet, sind in der Tabelle 'Alle\_Zutaten' gespeichert mit den Attributen Id, Bezeichnung, Einheit, Menge und Preis (z.B: {1255, "Milch", "Liter", 2, 2,50}). Hierbei gibt es zwei grundlegende Eigenschaften einer jeden Zutat, die als absolut notwendige Bedingungen zu betrachten sind, sprich: Eine Zutat muss mindestens eine oder mehrere Bedingungen beider Eigenschaften erfüllen .

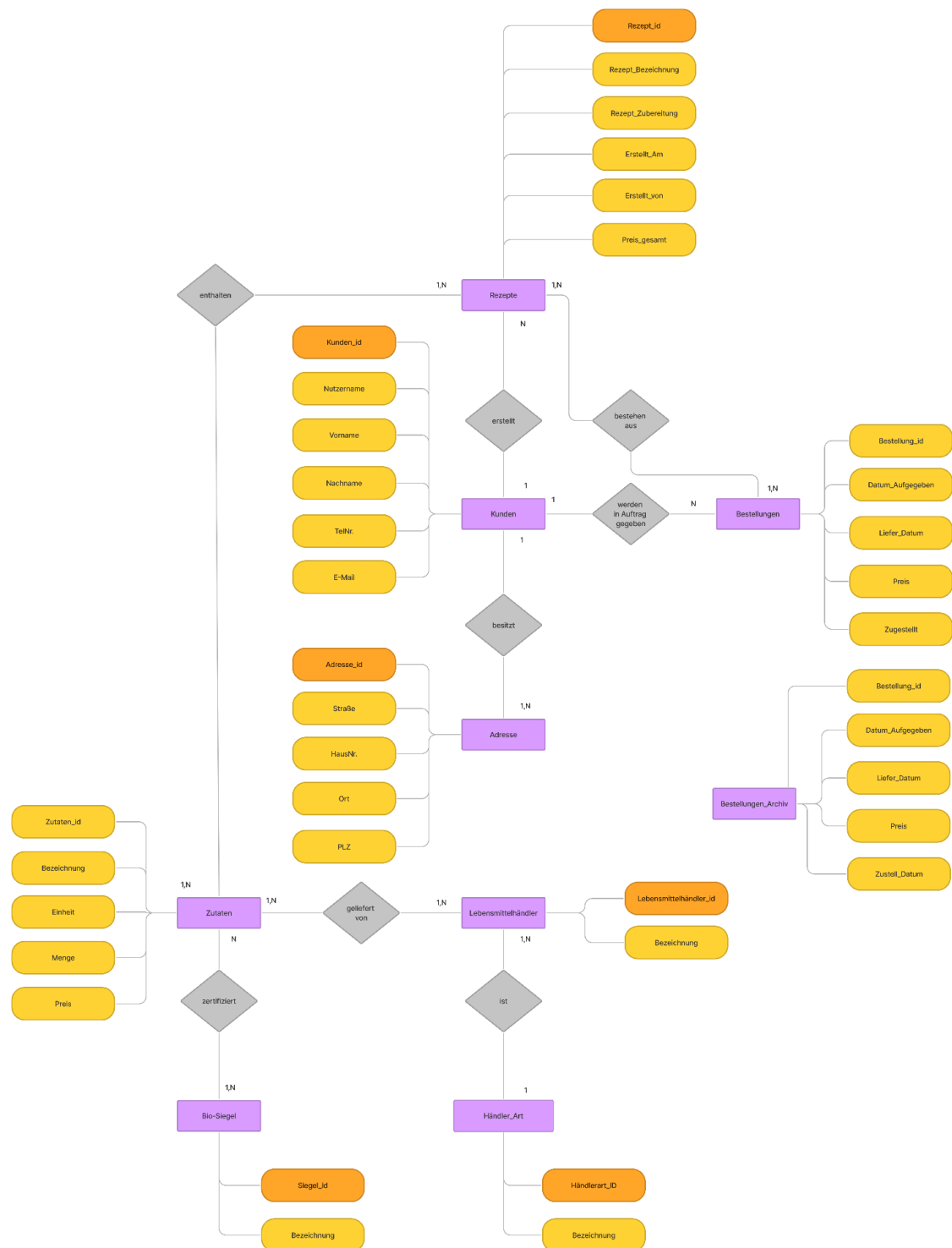
Die erste Eigenschaft ist, dass alle Zutaten durch mindestens eines der drei gängigsten Bio-Zertifikaten zertifiziert sein müssen. Die drei meistverbreiteten Bio-Zertifikate in Deutschland sind das EU-Bio-Siegel, Deutsches Bio-Siegel und Demeter-Siegel. Diese Information bietet Transparenz und ermöglicht den bewussten Einkauf von Lebensmitteln. Sie wird gespeichert in der Normierungstabelle 'Bio-Siegel'. Diese zeigt mit einer Primär Fremdschlüssel Beziehung auf die Tabelle aller Zutaten ('Alle\_Zutaten'). Die andere Eigenschaft ist, wie zuvor erwähnt, dass alle Zutaten ebenfalls nach Art ihrer Händler weiterhin zu unterscheiden sind. Jede Zutat wird von einem oder mehreren Händlern geliefert. Grundlegende Informationen der Händler sind in der Tabelle 'Lebensmittelhändler' gespeichert. Ob Bio-Lebensmittelhändler, Bio-Standard Zertifizierter Lokaler Hofladen oder Bio-Standard Zertifizierter Einzelhändler wird in einer eigenen Tabelle

`Händler\_Art` gespeichert. Die Art des Händlers ist eine wichtige Eigenschaft jeder Zutat und darauf hinaus eines der Qualitätsmerkmale des gesamten Online-Shops, weshalb die Entscheidung einer extra Tabelle getroffen wurde. Da Händler und Kunden beide eine oder mehrere Adressen besitzen zeigen diese auf eine eigene Tabelle `Adresse` in der grundlegende Informationen wie Straße, Hausnummer, Ort und Postleitzahl gespeichert sind (z.B. {“Bernhardusstraße”, 5, 73540, “Heubach”}).

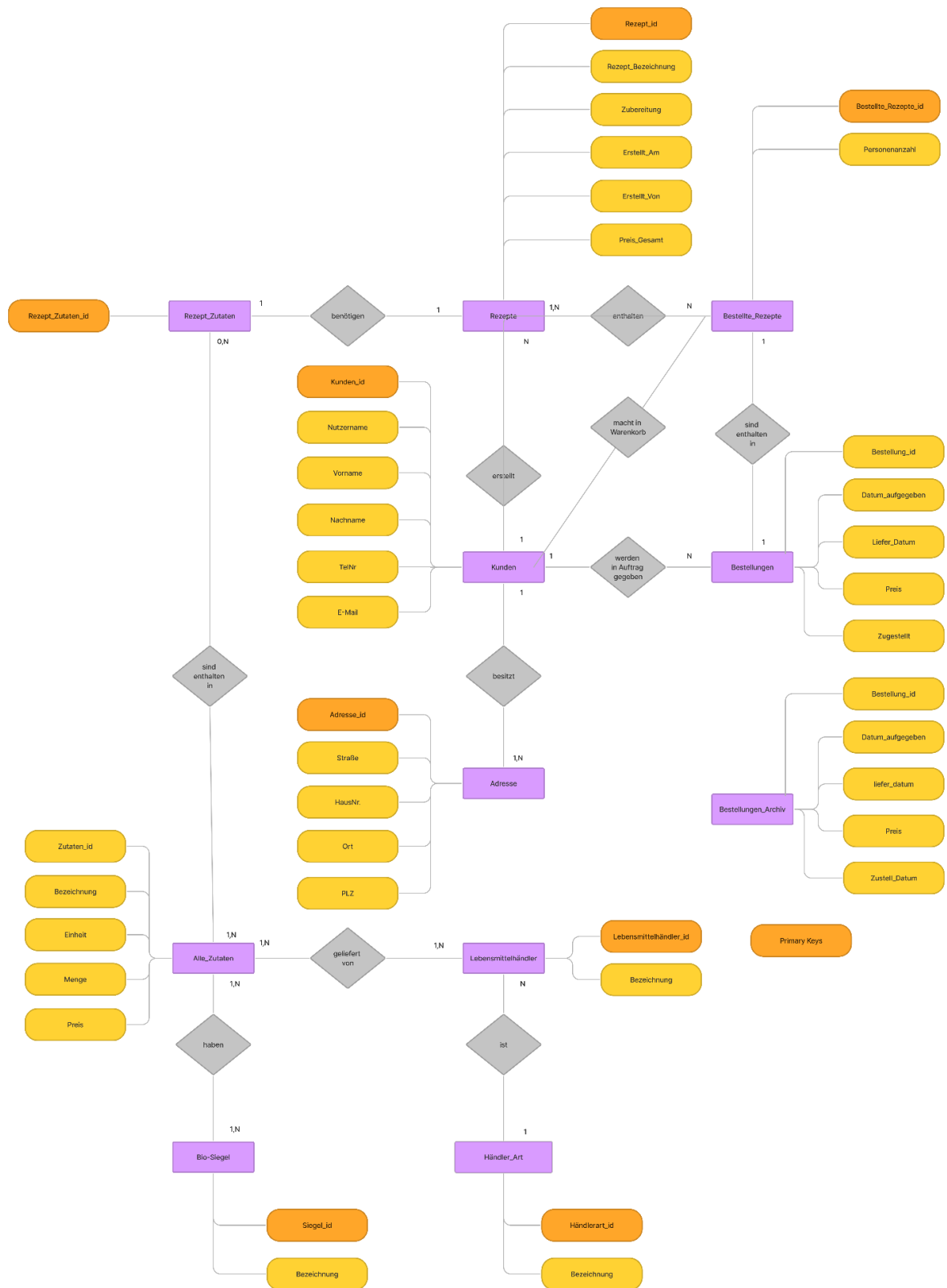
Doch nicht nur die Zutaten sollen Struktur geben, auch die Entität Rezepte spielt eine der zentralen Rollen bei der Strukturierung des Datenmodells. Alle Rezepte, ob bereits bestehende oder vom Kunden erstellte Rezepte, sind in der Tabelle `Rezepte` mit den Attributen Bezeichnung, Zubereitung, Erstellt\_Am und Preis\_Gesamt gespeichert (z.B. {“Spiegelei”, “Ei in der Pfanne braten”, 26.08.2022, 7,50}). Hier gilt es zu beachten, dass jedes Rezept eine andere Liste von Zutaten besitzen kann. Die Tabelle `Rezepte` und die Tabelle `Alle\_Zutaten` zeigen beide mittels einer Primär Fremdschlüssel Beziehung auf eine dritte Tabelle, `Rezept\_Zutaten`. Diese dritte Tabelle kümmert sich um die Zuweisung der einzelnen Zutaten zu den Rezepten. Wird ein eigenes Rezept vom Kunden erstellt, so wird dieses zur Tabelle der Rezepten hinzugefügt. Der Gesamtpreis eines jeden Gerichts wird anhand der Zutaten berechnet. Jede Zutat besitzt einen festen Preis, was es ermöglicht, einen Preis für selbst erstellte Rezepte zu kalkulieren. Der Kunde besitzt die Option, aus unterschiedlichen Mengen einer jeden Zutat zu wählen, welche dementsprechend angepasste Preise besitzen. So kann z.B ein Liter Milch zu einem festen Preis oder zwei Liter Milch zu einem festen Preis ausgewählt werden. Jedes Rezept kann anhand von Portionen vervielfacht werden. Die Information über die Anzahl an Portionen ist in der Tabelle `Bestellte\_Rezepte` enthalten. Die Anzahl an Portionen wirkt sich wiederum auf den Preis aus. Ähnlich wie bei den Zutaten und den Rezepten (es ist die Rede von der Tabelle `Rezept\_Zutaten`) stellt die Tabelle `Bestellte\_Rezepte` eine Brücke mittels Primär Fremdschlüssel Beziehung zwischen den Bestellungen und den Rezepten dar und kann als Liste der bestellten Rezepte vom Kunden einer jeden Bestellung, sprich dem Warenkorb gesehen werden. Sie ist dafür zuständig, dass die Rezepte den Bestellungen korrekt zugeordnet werden. Die Tabelle Bestellungen ist in direkter Verbindung zum Kunden. Kunden werden in der Tabelle `Kunden` gespeichert mit den Attributen Nutzername, Vorname, Nachname, Telefonnummer und Email (z.B. {“Susi64”, “Susanne”, “Retschler”, 015752789918, “[susi64@gmail.com](mailto:susi64@gmail.com)”}). Der Kunde gibt mindestens eine Bestellung in Auftrag, welche wiederum in der Tabelle `Bestellungen` gespeichert werden. Diese verfügt über die Attribute Datum Aufgegeben, Lieferdatum, Preis und Zugestellt (z.B. {22.04.2022, 30.04.2022, 27.50, y}). Die Tabelle `Bestellungen\_Archiv` ist für abgeschlossene Bestellungen vorgesehen. Das Entwicklerteam wünscht einen schönen Tag und viel Freude beim Lesen.

## 2. Entity Relationship

### 2.1 Logisch



## 2.2 Physisch



## 2.3 Dokumentation der Entitätstypen

### Bewegungsdaten

Entität Alle\_Zutaten

Enthält alle Zutaten die der Lieferservice zur Verfügung hat

Fachliche Attribute

Zutaten\_id (Pflicht, Integer) : Identifikationschlüssel

Bezeichnung (Pflicht, Text)

Einheit (Pflicht, Text)

Menge (Pflicht, Float)

Preis (Pflicht, Number)

Entität Adresse

Enthält die Adressen der Kunden als auch Lebensmittelhändler

Fachliche Attribute

Adress\_id (Pflicht, Integer) : Identifikationschlüssel

Straße (Pflicht, Text)

HausNr. (Pflicht, Integer)

Ort (Pflicht, Text)

PLZ (Pflicht, Text)

Entität Lebensmittelhändler

Enthält jeden Shop der Partner des Lieferservices ist

Fachliche Attribute

Lebensmittelhändler\_id (Pflicht, Integer) : Identifikationschlüssel

Bezeichnung (Pflicht, Text)

Entität Kunden

Eine Liste aller Kunden und deren persönlichen Daten

Fachliche Attribute

Kunden\_id (Pflicht, Integer) : Identifikationschlüssel

Nutzername (Pflicht, Text)

Vorname (Pflicht, Text)

Nachname (Pflicht, Text)

TelNr. (Pflicht, Text)

E-Mail (Pflicht, Text)

#### Entität Bestellungen

Alle Bestellungen des Lieferservices

##### Fachliche Attribute

Bestellun\_id (Pflicht, Integer) : Identifikationschlüssel

Datum\_aufgegeben (Pflicht, Date)

Liefer\_Datum (Pflicht, Date)

Preis (Pflicht, Number)

Zugestellt (Pflicht, Char)

#### Entität Bestellungen\_Archiv

Alle abgeschlossenen Bestellungen des Lieferservices

##### Fachliche Attribute

Bestellun\_id (Pflicht, Integer) : Identifikationschlüssel

Datum\_aufgegeben (Pflicht, Date)

Liefer\_Datum (Pflicht, Date)

Preis (Pflicht, Number)

Zustell\_Datum (Pflicht, Date)

#### Entität Bestellte\_Rezette

Die Liste der Rezepte einer Bestellung

##### Fachliche Attribute

Bestellte\_Rezette\_id (Pflicht, Integer) : Identifikationschlüssel

Personenzahl (Pflicht, Integer)

#### Entität Rezepte

Liste aller Rezepte, sowohl selbst erstellte als auch bestehende

##### Fachliche Attribute

Rezept\_id (Pflicht, Integer) : Identifikationschlüssel

Rezept\_Bezeichnung (Pflicht, Text)

Zubereitung (Pflicht, Text)

Erstellt\_Am (Pflicht, Date)

Preis\_Gesamt (Pflicht, Number)

#### Entität Rezept\_Zutaten

Liste der Zutaten eines bestimmten Rezepts

##### Fachliche Attribute

Rezept\_Zutaten\_id (Pflicht, Integer) : Identifikationschlüssel



## Normierungsdaten

Entität Bio-Siegel

Enthält die Siegel der Bio-Zertifikate

Fachliche Attribute

Siegel\_id (Pflicht, Integer) : Identifikationschlüssel

Bezeichnung (Pflicht, Text)

Entität Händler\_Art

Enthält die Information über die Art des Händlers

Fachliche Attribute

Händlerart\_id (Pflicht, Integer) : Identifikationschlüssel

Bezeichnung (Pflicht, Text)

## Index

- "index\_alle\_zutaten"
- Wir haben einen Index auf die Tabelle alle\_zutaten auf die Attribute zutaten\_id und bezeichnung gesetzt, da es eine große Tabelle ist und die Selektivität der beiden Attribute recht hoch ist. Allerdings kann es zu vielen Änderungen kommen, was nicht ideal für die Performance ist, da bei jeder Änderung der Index neu gepflegt werden muss und darunter die Performance leidet.

## Trigger

- Update-Trigger: "Archivieren",  
wenn in der Tabelle `Bestellungen` die Spalte zugestellt auf Y geupdated wird, wird der Datensatz in die Tabelle `Bestellungen\_Archiv` kopiert.
- Insert-Trigger (fehlerhaft): "Bestellung\_Löschen",  
löscht zugestellte Bestellungen aus der Tabelle `Bestellungen`.
- Insert-Trigger (fehlerhaft): "Trig1",  
updated den Gesamtpreis in der Tabelle `Bestellungen` mit der Funktion "GetBestellungGesamtPreis".

## Funktion

- "GetBestellungGesamtPreis",  
berechnet den Gesamtpreis einer Bestellung anhand der Preise der einzelnen Zutaten für die ausgewählten Rezepte sowie der Person Anzahl.

## 3. Skripte

### 3.1 DDL

-- CANDIDATE KEYS LÖSCHEN

ALTER TABLE kunde DROP CONSTRAINT kunde\_k1;

ALTER TABLE kunde DROP CONSTRAINT kunde\_k2;

-- FOREIGN KEYS LÖSCHEN

ALTER TABLE kunde DROP CONSTRAINT kunde\_fk1;

ALTER TABLE lebensmittelhändler DROP CONSTRAINT lmh\_fk1;

ALTER TABLE alle\_zutaten DROP CONSTRAINT alle\_zutaten\_fk1;

ALTER TABLE alle\_zutaten DROP CONSTRAINT alle\_zutaten\_fk2;

ALTER TABLE bestellungen DROP CONSTRAINT bestellungen\_fk1;

ALTER TABLE rezept\_zutaten DROP CONSTRAINT rezept\_zutaten\_fk1;

ALTER TABLE rezept\_zutaten DROP CONSTRAINT rezept\_zutaten\_fk2;

ALTER TABLE bestellte\_rezepte DROP CONSTRAINT bestellte\_rezepte\_fk1;

ALTER TABLE bestellte\_rezepte DROP CONSTRAINT bestellte\_rezepte\_fk2;

ALTER TABLE bestellte\_rezepte DROP CONSTRAINT bestellte\_rezepte\_fk3;

--PRIMARY KEYS LÖSCHEN

ALTER TABLE adresse DROP CONSTRAINT adresse\_pk;

ALTER TABLE kunde DROP CONSTRAINT kunde\_pk;

ALTER TABLE bio\_siegel DROP CONSTRAINT bio\_siegel\_pk;

ALTER TABLE händler\_art DROP CONSTRAINT händler\_art\_pk;

ALTER TABLE lebensmittelhändler DROP CONSTRAINT lebensmittelhändler\_pk;

ALTER TABLE alle\_zutaten DROP CONSTRAINT zutaten\_pk;

ALTER TABLE rezept\_zutaten DROP CONSTRAINT rezept\_zutaten\_pk;

ALTER TABLE bestellungen DROP CONSTRAINT bestellungen\_pk;

ALTER TABLE bestellte\_rezepte DROP CONSTRAINT bestellte\_rezepte\_pk;

ALTER TABLE rezepte DROP CONSTRAINT rezepte\_pk;

-- TABELLEN LEEREN & LÖSCHEN --

DELETE FROM bestellungen;

DROP TABLE bestellungen;

DELETE FROM bestellungen\_archiv;

DROP TABLE bestellungen\_archiv;

```

DELETE FROM rezepte;
DROP TABLE rezepte;
DELETE FROM kunde;
DROP TABLE kunde;
DELETE FROM adresse;
DROP TABLE adresse;
DELETE FROM alle_zutaten;
DROP TABLE alle_zutaten;
DELETE FROM rezept_zutaten;
DROP TABLE rezept_zutaten;
DELETE FROM bestellte_rezepte;
DROP TABLE bestellte_rezepte;
DELETE FROM bio_siegel;
DROP TABLE bio_siegel;
DELETE FROM lebensmittelhändler;
DROP TABLE lebensmittelhändler;
DELETE FROM händler_art;
DROP TABLE händler_art;

-- TABELLEN ERSTELLEN --
CREATE TABLE adresse(
    adresse_id INTEGER,
    straße VARCHAR(30) NOT NULL,
    hausnr VARCHAR(5),
    plz VARCHAR(10),
    ort VARCHAR(30) NOT NULL
);

CREATE TABLE kunde(
    kunden_id INTEGER,
    nutzername VARCHAR(15),
    vorname VARCHAR(20) NOT NULL,
    nachname VARCHAR(20) NOT NULL,
    telnr VARCHAR(20),
    email VARCHAR(40) NOT NULL,
    adresse_id INTEGER
);

CREATE TABLE bio_siegel(
    siegel_id INTEGER,
    bezeichnung VARCHAR(40)
);

CREATE TABLE händler_art(
    händlerart_id INTEGER,
    bezeichnung VARCHAR(40)
);

```

```
CREATE TABLE lebensmittelhändler(  
    lebensmittelhändler_id INTEGER,  
    bezeichnung VARCHAR(40),  
    händlerart_id INTEGER,  
    adresse_id INTEGER  
);
```

```
CREATE TABLE alle_zutaten(  
    zutaten_id INTEGER,  
    bezeichnung VARCHAR(40),  
    einheit VARCHAR(10) CHECK (einheit in ('g','kg','l','ml','Stück','Päckchen')),  
    geliefert_von INTEGER,  
    siegel_id INTEGER,  
    menge float,  
    preis number(5,2)  
);
```

```
CREATE TABLE rezept_zutaten(  
    rezept_zutaten_id INTEGER,  
    zutaten_id INTEGER,  
    rezept_id INTEGER  
);
```

```
CREATE TABLE bestellte_rezepte(  
    bestellte_rezepte_id INTEGER,  
    bestell_id INTEGER,  
    rezept_id INTEGER,  
    personenanzahl INTEGER,  
    kunden_id INTEGER  
);
```

```
CREATE TABLE bestellungen(  
    bestellung_id INTEGER,  
    datum_aufgegeben DATE,  
    bestellt_von INTEGER,  
    preis NUMBER(5,2),  
    zugestellt CHAR(1) DEFAULT 'N' CHECK (zugestellt in ('Y','N'))  
);
```

```
CREATE TABLE bestellungen_archiv(  
    bestellung_archiv_id INTEGER,  
    datum_aufgegeben DATE,  
    bestellt_von INTEGER,  
    preis NUMBER(5,2),
```

```
    zustelldatum DATE
);
```

```
CREATE TABLE rezepte(
    rezept_id INTEGER,
    rezept_bezeichnung VARCHAR(40),
    zubereitung VARCHAR(1500),
    erstellt_am DATE,
    preis_gesamt NUMBER(5,2),
    erstellt_von VARCHAR(15)
);
```

-- ALLE PRIMARY KEYS ERSTELLEN

```
ALTER TABLE adresse ADD CONSTRAINT adresse_pk PRIMARY KEY (adresse_id);
ALTER TABLE kunde ADD CONSTRAINT kunde_pk PRIMARY KEY (kunden_id);
ALTER TABLE bio_siegel ADD CONSTRAINT bio_siegel_pk PRIMARY KEY (siegel_id);
ALTER TABLE händler_art ADD CONSTRAINT händler_art_pk PRIMARY KEY
(händlerart_id);
ALTER TABLE lebensmittelhändler ADD CONSTRAINT lebensmittelhändler_pk PRIMARY
KEY (lebensmittelhändler_id);
ALTER TABLE alle_zutaten ADD CONSTRAINT zutaten_pk PRIMARY KEY (zutaten_id);
ALTER TABLE rezept_zutaten ADD CONSTRAINT rezept_zutaten_pk PRIMARY KEY
(rezept_zutaten_id);
ALTER TABLE bestellungen ADD CONSTRAINT bestellungen_pk PRIMARY KEY
(bestellung_id);
ALTER TABLE bestellte_rezepte ADD CONSTRAINT bestellte_rezepte_pk PRIMARY KEY
(bestellte_rezepte_id);
ALTER TABLE rezepte ADD CONSTRAINT rezepte_pk PRIMARY KEY (rezept_id);
```

-- CANDIDATE KEYS ERSTELLEN

```
ALTER TABLE kunde ADD CONSTRAINT kunde_k1 UNIQUE (nutzername);
ALTER TABLE kunde ADD CONSTRAINT kunde_k2 UNIQUE (email);
```

-- FOREIGN KEYS ERSTELLEN

```
ALTER TABLE bestellungen ADD CONSTRAINT bestellungen_fk1 FOREIGN KEY
(bestellt_von) REFERENCES kunde (kunden_id);
ALTER TABLE bestellte_rezepte ADD CONSTRAINT bestellte_rezepte_fk1 FOREIGN KEY
(rezept_id) REFERENCES rezepte (rezept_id);
```

```

ALTER TABLE bestellte_rezepte ADD CONSTRAINT bestellte_rezepte_fk2 FOREIGN KEY
(bestell_id) REFERENCES bestellungen (bestellung_id);
ALTER TABLE bestellte_rezepte ADD CONSTRAINT bestellte_rezepte_fk3 FOREIGN KEY
(kunden_id) REFERENCES kunde (kunden_id);
ALTER TABLE kunde ADD CONSTRAINT kunde_fk1 FOREIGN KEY (adresse_id)
REFERENCES adresse (adresse_id);
ALTER TABLE lebensmittelhändler ADD CONSTRAINT lmh_fk1 FOREIGN KEY
(händlerart_id) REFERENCES händler_art (händlerart_id);
ALTER TABLE rezept_zutaten ADD CONSTRAINT rezept_zutaten_fk1 FOREIGN KEY
(zutaten_id) REFERENCES alle_zutaten (zutaten_id);
ALTER TABLE rezept_zutaten ADD CONSTRAINT rezept_zutaten_fk2 FOREIGN KEY
(rezept_id) REFERENCES rezepte (rezept_id);
ALTER TABLE alle_zutaten ADD CONSTRAINT alle_zutaten_fk1 FOREIGN KEY
(geliefert_von) REFERENCES lebensmittelhändler (lebensmittelhändler_id);
ALTER TABLE alle_zutaten ADD CONSTRAINT alle_zutaten_fk2 FOREIGN KEY (siegel_id)
REFERENCES bio_siegel (siegel_id);

```

-- Beispiel-Datensätze einfügen --

```

INSERT INTO bio_siegel (siegel_id, bezeichnung) VALUES (1, 'demeter');
INSERT INTO bio_siegel (siegel_id, bezeichnung) VALUES (2, 'Deutsches Bio-Siegel');
INSERT INTO bio_siegel (siegel_id, bezeichnung) VALUES (3, 'EU-Bio-Siegel');

INSERT INTO händler_art (händlerart_id, bezeichnung) VALUES (1,
'Bio-Lebensmittelhandel');
INSERT INTO händler_art (händlerart_id, bezeichnung) VALUES (2, 'Lokaler Bio-Hofladen');
INSERT INTO händler_art (händlerart_id, bezeichnung) VALUES (3, 'Bio-zertifizierter
Einzelhandel');

INSERT INTO adresse (adresse_id, straße, hausnr, plz, ort) VALUES (1, 'Bernardostr. ',
'5/1', '71640', 'Ludwigsburg');
INSERT INTO adresse (adresse_id, straße, hausnr, plz, ort) VALUES (2, 'Mozartstr. ', '77',
'12345', 'Weil der Stadt');
INSERT INTO adresse (adresse_id, straße, hausnr, plz, ort) VALUES (3, 'Asphaltstr. ', '3',
'65833', 'Dorfstadt');

INSERT INTO lebensmittelhändler (lebensmittelhändler_id, bezeichnung, händlerart_id)
VALUES (1, 'Henns-Biohof', 2);
INSERT INTO lebensmittelhändler (lebensmittelhändler_id, bezeichnung, händlerart_id)
VALUES (2, 'Mias Futterecke', 3);
INSERT INTO lebensmittelhändler (lebensmittelhändler_id, bezeichnung, händlerart_id)
VALUES (3, 'Milchbauernhof Dobler', 2);
INSERT INTO lebensmittelhändler (lebensmittelhändler_id, bezeichnung, händlerart_id)
VALUES (4, 'Emmas Bioladen', 1);

INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)

```

```

VALUES (1, 'Hafermilch 500ml', 'ml', 4, 3, 500, 0.75);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (2, 'Sojamilch 500ml', 'ml', 3, 1, 500, 1.15);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (3, 'Mehl 250g', 'g', 1, 2, 250, 1.25);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (4, 'Mehl 1000g', 'g', 1, 2, 1000, 2.00);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (5, 'Rohrzucker 10g', 'g', 1, 2, 10, 0.15);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (6, 'Rohrzucker 500g', 'g', 1, 2, 500, 1.30);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (7, 'Vanillinzucker 1x Päckchen', 'Päckchen', 2, 3, 1, 0.35);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (8, 'Backpulver 1x Päckchen', 'Päckchen', 4, 3, 1, 0.20);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (9, 'Eiersatz 1x Päckchen', 'Päckchen', 2, 1, 1, 2.97);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (10, 'Eiersatz 2x Päckchen', 'Päckchen', 1, 1, 2, 1.55);
INSERT INTO alle_zutaten (zutaten_id, bezeichnung, einheit, geliefert_von, siegel_id,
menge, preis)
VALUES (11, 'Margarine 500g', 'g', 3, 1, 500, 1.17);

```

```

INSERT INTO rezepte(rezept_id, rezept_bezeichnung, zubereitung, erstellt_am,
preis_gesamt, erstellt_von)
values (1, 'Florian`s Waffeln', 'Zutaten vermischen und in Waffeleisen goldbraun backen und
mit Obst oder Puderzucker servieren', to_Date('01.01.2023','DD.MM.YYYY'), 5.99,
'flomagWaffeln');
INSERT INTO rezepte(rezept_id, rezept_bezeichnung, zubereitung, erstellt_am,
preis_gesamt, erstellt_von)
values (2, 'Leckere Pfannkuchen', 'Zutaten verrühren und in der Pfanne goldbraun braten
und mit Nussnougatcreme oder Apfelmus servieren', to_Date('10.01.2023','DD.MM.YYYY'),
7.99, 'flotteagathe1');
INSERT INTO rezepte(rezept_id, rezept_bezeichnung, zubereitung, erstellt_am,
preis_gesamt, erstellt_von)
values (3, 'Schneller Tassenkuchen', 'Zutaten in eine Tasse geben und in eine Minute bei
800 W in der Mikrowelle backen', to_Date('12.01.2023','DD.MM.YYYY'), 8.99, 'mariechen5');

```

```

INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (1, 1, 1);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (2, 3, 1);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (3, 5, 1);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (4, 8, 1);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (5, 9, 1);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (6, 2, 2);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (7, 3, 2);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (8, 7, 2);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (9, 9, 2);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (10, 3, 3 );
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (11, 5, 3);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (12, 7, 3);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (13, 8, 3);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (14, 10, 3);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (15, 11, 3);
INSERT INTO rezept_zutaten(rezept_zutaten_id, zutaten_id, rezept_id)
values (16, 11, 1);

```

```

INSERT INTO kunde (kunden_id, nutzername, vorname, nachname, telnr, email,
adresse_id)
VALUES (1, 'mariechen5', 'Marie', 'Müller', '0174 3325689', 'mariemueller@gmail.com', 3);
INSERT INTO kunde (kunden_id, nutzername, vorname, nachname, telnr, email,
adresse_id)
VALUES (2, 'flotteagathe10', 'Agathe', 'Windig', '0667 62742944', 'awindig@gmx.de', 1);
INSERT INTO kunde (kunden_id, nutzername, vorname, nachname, telnr, email,
adresse_id)
VALUES (3, 'flomagWaffeln', 'Florian', 'Simmer', '0163 3724948', 'flofloflo@yahoo.de', 2);

```

```

INSERT INTO bestellungen (bestellung_id, datum_aufgegeben, bestellt_von, preis)
VALUES (1, to_date('15.01.2023', 'DD.MM.YYYY'),1,27.96);
INSERT INTO bestellungen (bestellung_id, datum_aufgegeben, bestellt_von, preis)
VALUES (2, to_date('16.01.2023', 'DD.MM.YYYY'),2, 8.99);

```



```
INSERT INTO bestellungen (bestellung_id, datum_aufgegeben, bestellt_von, preis)
VALUES (3,to_date('17.01.2023', 'DD.MM.YYYY'), 3,23.96 );
```

```
INSERT INTO bestellte_rezepte (bestellte_rezepte_id, bestell_id, rezept_id,
personenanzahl, kunden_id)
VALUES (1, 1, 1, 2, 1);
INSERT INTO bestellte_rezepte (bestellte_rezepte_id, bestell_id, rezept_id,
personenanzahl, kunden_id)
VALUES (2, 1, 2, 2, 1);
INSERT INTO bestellte_rezepte (bestellte_rezepte_id, bestell_id, rezept_id,
personenanzahl, kunden_id)
VALUES (3, 2, 3, 1, 2);
INSERT INTO bestellte_rezepte (bestellte_rezepte_id, bestell_id, rezept_id,
personenanzahl, kunden_id)
VALUES (4, 3, 1, 4, 3);
```

## 3.2 Index

```
CREATE INDEX index_alle_zutaten ON alle_zutaten (zutaten_id, bezeichnung);
```

## 3.3 Trigger und Funktion

```
--UPDATE-TRIGGER
-- wenn in bestellungen die spalte zugestellt auf 'Y' geupdatet wird, kopieren wir den
Datensatz in bestellungen_archiv und löschen den Datensatz in Bestellungen

CREATE OR REPLACE TRIGGER archivieren
AFTER UPDATE ON bestellungen
REFERENCING NEW AS NEW OLD AS OLD FOR EACH ROW
BEGIN
    IF (:NEW.zugestellt = 'Y') THEN
        INSERT INTO bestellungen_archiv(bestellung_archiv_id, datum_aufgegeben,
bestellt_von, preis, zustelldatum)
        VALUES(:OLD.bestellung_id, :OLD.datum_aufgegeben, :OLD.bestellt_von, :OLD.preis,
sysdate);
    END IF;
END;
/
```

```
-- proof für update-trigger
```

```
UPDATE bestellungen SET zugestellt = 'Y' WHERE bestellung_id = 3; -- bestellung 3 wurde
zugestellt
```

```

SELECT * FROM bestellungen; -- bei datensatz 3 wurde zugestellt auf 'Y' gesetzt
SELECT * FROM bestellungen_archiv; -- datensatz wurde in bestellungen_archiv eingefügt

```

```

/*

```

```

-- INSERT-TRIGGER
-- funktioniert leider nicht, sollte zugestellte/abgeschlossene bestellung aus bestellungen
löschen

```

```

CREATE OR REPLACE TRIGGER bestellung_löschen
AFTER INSERT ON bestellungen_archiv
REFERENCING NEW AS NEW OLD AS OLD FOR EACH ROW
BEGIN
    DELETE FROM bestellungen WHERE zugestellt = 'Y';
END;
/

```

```

-- proof für insert-trigger
SELECT * FROM bestellungen; -- bestellung mit bestell_id 3 wurde gelöscht
*/

```

```

--FUNKTION
-- berechnet den Gesamtpreis einer Bestellung anhand der Preise der einzelnen Zutaten für
die ausgewählten Rezepte sowie der Personenanzahl

```

```

CREATE OR REPLACE FUNCTION getBestellungGesamtpreis(fürBestellung IN int)
RETURN number
IS varPreisGesamt NUMBER := 0;
BEGIN
    SELECT sum(rezeptpreis) * panzahl INTO varPreisGesamt
    FROM
    (
        SELECT bestellungen.bestellung_id, sum(alle_zutaten.preis) as rezeptpreis,
        rezepte.rezept_bezeichnung, bestellte_rezepte.personenanzahl panzahl
        FROM bestellungen, bestellte_rezepte, rezepte, rezept_zutaten, alle_zutaten
        WHERE bestellungen.bestellung_id = bestellte_rezepte.bestell_id
        AND bestellte_rezepte.rezept_id = rezepte.rezept_id
        AND rezept_zutaten.rezept_id = rezepte.rezept_id
        AND rezept_zutaten.zutaten_id = alle_zutaten.zutaten_id
        AND bestellung_id = fürBestellung
        GROUP BY rezepte.rezept_bezeichnung, bestellungen.bestellung_id,
        bestellte_rezepte.personenanzahl
    ) GROUP BY panzahl;
    RETURN (varPreisGesamt);
END;

```

/

--proof funktion -- funktioniert

SELECT getBestellungGesamtpreis(1) from DUAL;

/\*

--INSERT-TRIGGER

-- Sollte mit Funktion den Gesamtpreis in der Tabelle Bestellungen updaten

CREATE OR REPLACE TRIGGER trig1

AFTER INSERT ON bestellte\_rezepte

REFERENCING NEW AS NEW OLD AS OLD

FOR EACH ROW

DECLARE

gpreis number := 0;

bestell\_id\_index integer := :NEW.bestell\_id;

BEGIN

select getBestellungGesamtpreis(bestell\_id\_index) into gpreis from dual; -- geht nicht

INSERT INTO bestellungen (bestellung\_id, datum\_aufgegeben, bestellt\_von, preis)

VALUES (:NEW.bestell\_id, sysdate, :NEW.kunden\_id, gpreis);

END;

/

\*/

Bewertungsschema (bitte als letzte Seite der Ausarbeitung einbinden):

Kriterium		max.	ist	Kommentar
Originalität		10		
Relative Qualität		15		
Spezifikation / Systembeschreibung		15		
ER-Modell inkl. formaler Korrektheit im Sinn der Normalisierung		10		
Sinnvolle Nutzung von Normierungsdaten		5		
Lauffähigkeit		5		
Restartfähigkeit		5		
Vollständigkeit der Testdaten		5		
Umfang der Umsetzung des Datenmodells		-	-	
	Anzahl der Tabellen	5		
	Vollständige Nutzung der Datentypen	5		
	Sinnvolle Nutzung von Randbedingungen	5		
	Sinnvolle Nutzung von Indexierung	5		
	Insert-Trigger	5		
	Update-Trigger	5		
Summe				