Inhalt

Aufgabenteil 1

1.1 Einleitung

Die folgende Ausarbeitung der Lehrveranstaltung Datenbanken betrachtet die Modellierung einer Datenbank und die daran anknüpfende Umsetzung mittels SQLite. Fachlicher Hintergrund der Datenbank liefert das Konzept des „Amazon Marketplace“. Amazon selbst wurde 1994 gegründet und ist im Online- und Versandhandel tätig. Mit 1.298.000 Mitarbeitern und einem Umsatz von 386 Mrd. US-Dollar im Jahr 2020 ist Amazon nach eigenen Angaben der Marktführer im Internethandel[1]. Die integrierte Marketplace-Plattform kann dabei von Privatpersonen und Unternehmen genutzt werden, um Produkte zum Verkauf anzubieten. Experten gehen davon aus, dass der Umsatz der Marketplace-Händler mittlerweile größer als der Amazon-eigene Umsatz ist.[2]

Die Arbeit umfasst dabei eine Fallbeschreibung des erwähnten Amazon Marketplace, an welche eine Modellierung mittels ER-Diagramms angeschlossen wird. Es folgt eine Modellierung in Form eines relationalen Datenmodells. Abschließend werden verschiedene Abfragen auf die erstellte Datenbank erstellt und genauer erläutert.

1.2 Fallbeschreibung

Grundlage für die weiteren Modelle ist folgende Fallbeschreibung:

Ein Kunde registriert sich bei Amazon (und damit beim Amazon Marketplace) mittels E-Mail, Passwort, Telefonnummer, Adresse und Name. Jeder Kunde kann dabei eindeutig über seine E-Mail verifiziert und identifiziert werden. Die Adresse des Kunden setzt sich aus der Postleitzahl, der Straße, der Hausnummer und ggf. einem Adresszusatz zusammen. Die Telefonnummer ist als Kontaktmöglichkeit und zur 2-Faktor-Authentifizierung hinterlegt. Aus diesem Grund können auch

[1] Vgl. https://de.wikipedia.org/wiki/Virtueller_Marktplatz

[2] Vgl. https://de.wikipedia.org/wiki/Amazon

mehrere Telefonnummern hinterlegt werden, sodass bei einer Benutzung des Kontos durch mehrere Benutzer jeder einzelne die Authentifizierung mittels des zweiten Faktors durchführen kann.

Auf dem Marketplace sind verschiedene Anbieter vertreten. Jedem Anbieter ist eine eindeutige ID zugeordnet. Zudem besitzt jeder Anbieter einen Namen. Da Amazon daran interessiert ist dem Kunden das bestmögliche Erlebnis zu ermöglichen und ein möglicher Missbrauch oder Betrugsversuche seitens der Anbieter direkte Auswirkungen auf die Reputation von Amazon selbst hätten, sollen diese durch ein Anti-Fraud-System verhindert werden. Aus diesem Grund wird jedem Anbieter ein „Trust-Score“ zugeordnet. Dieser wird fortlaufend durch erfolgreiche Bestellungen, Bewertungen oder auch Beschwerden seitens der Käufer berechnet. Das Unterschreiten eines gesetzten Wertes kann dabei zur Sperrung des Anbieter-Kontos führen. Damit der Kunde sich zudem über den Anbieter informieren kann, ist eine Mail-Adresse und die Website des Anbieters hinterlegt. Ohne einen Anbieter können Produkte nicht existieren. Bei Löschung oder Sperrung eines Anbieters, werden seine Produkte somit ebenfalls entfernt. Ein Produkt ist immer nur einem Anbieter zugeordnet.

Viele Kunden haben viele Produkte zur Auswahl. Ein Produkt ist fest einem Anbieter zugeordnet, welcher dieses Produkt über den Marketplace vertreibt. Jedes Produkt hat eine feste und eindeutige ID, einen konkreten Namen, einen Beschreibungstext, ein Produktbild (bzw. genau genommen den Pfad zur Bilddatei) und einen Preis.

Da der Marketplace sehr produktorientiert arbeitet, werden Bewertungen durch den Kunden zum Produkt getätigt und nicht zum Anbieter. Eine solche Bewertung besitzt eine eindeutige Bewertungs-ID, ein Datum, an welchem die Bewertung abgegeben wurde, das Alter der Bewertung in Tagen, das genaue Rating (1-5 Sterne), einen Bewertungstitel und einen Bewertungstext.

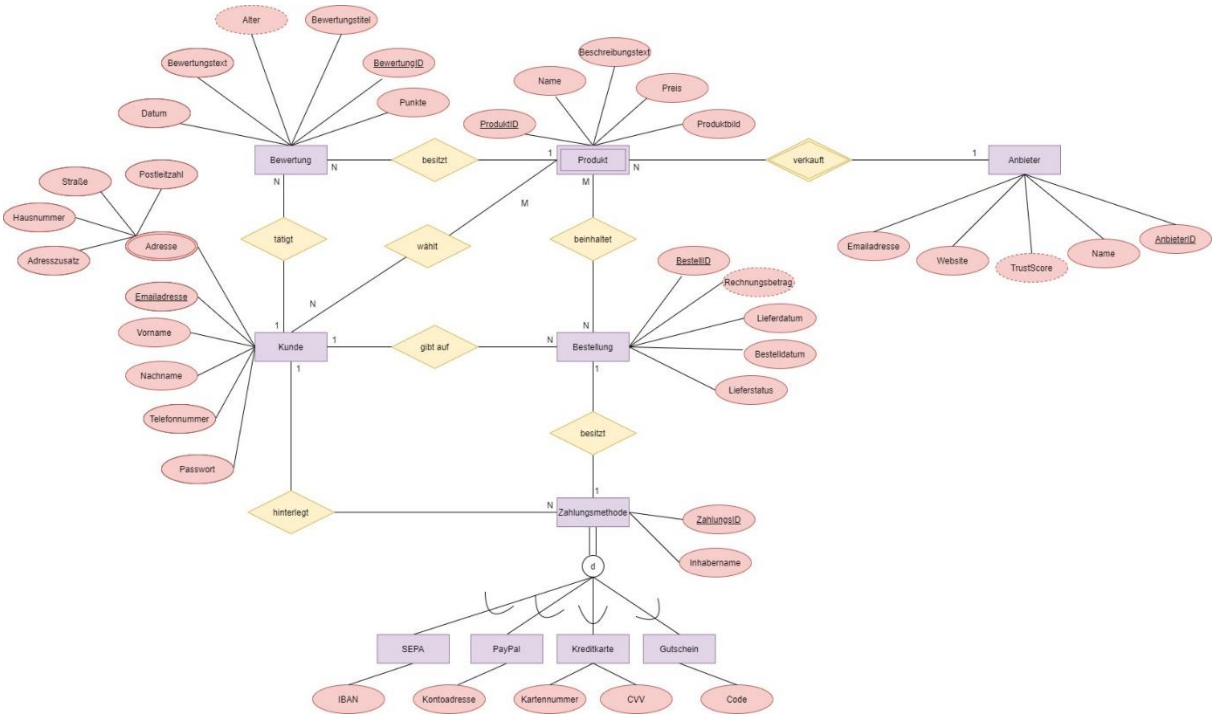
Ein Kunde kann auf der Plattform eine oder mehrere Bestellungen durchführen. Die Bestellung kann dabei einen oder mehrere Produkte enthalten. Jede Bestellung besitzt eine Bestell-ID, einen Rechnungsbetrag, welcher sich wiederum aus den Preisen der enthaltenen Produkte berechnen lässt, ein Bestelldatum, das Lieferdatum und einen Lieferstatus. Die N:M-Beziehung zwischen Bestellungen und Produkten wird dabei im späteren Verlauf durch eine zusätzliche Mapping-Tabelle aufgelöst.

Bei jeder Bestellung hat der Kunde die Wahl zwischen verschiedenen Zahlungsmethoden. Alle Zahlungsmethoden haben eine Zahlungs-ID und einen Zahlungsinhaber. Die konkreten Zahlungsmethoden sind PayPal, Kreditkarte, Gutscheincodes und SEPA-Lastschrift. Bei der Zahlungsart PayPal ist ein PayPal-Nutzerkonto anzugeben. Neben der Kartennummer und dem Ablaufdatum muss bei einer Kreditkarte auch der Sicherheitscode (CVV-Code) angegeben werden. Bei einem Gutschein-Code ist die korrekte Zeichenfolge des Codes einzugeben. Das SEPA-Lastschriftverfahren erfordert die IBAN des Käufers. Der Kunde kann zudem verschiedene Zahlungsmethoden in seinem Konto speichern (1:N).

Aufgabenteil 2

2.1 ER-Diagramm

Aus den Angaben aus Kapitel 1.2 ergibt sich folgendes ER-Modell:



Aufgabenteil 3

3.1 Relationales Datenmodell

Das folgende relationale Datenmodell wurde aus dem ER-Diagramm abgeleitet und aufbereitet. Die Typen werden bereits im relationalen Modell angegeben, sodass eine Erstellung der jeweiligen Tabellen später einfacher fällt. Passwörter werden dabei als einfache varChars dargestellt, weil die Passwörter nicht im Klartext, sondern als Hash abgespeichert werden sollen. Die Mail-Adressen wurden sehr lange gewählt, da diese erfahrungsgemäß durch die Nutzung verschiedenster Dienst (wie Azure) durch das Anhängen weiterer Domainsuffixes sehr lange werden können und somit durch Authentifizierungen über oAuth oder SAML Probleme auf Datenbankebene verursachen können, sofern auf dieser die Attribute zu klein gesized wurden. Bei den Produktbildern handelt es sich tatsächlich nicht um BLOB-Speicher, sondern lediglich um Pfade in Form von Strings, sodass die Datenbank nicht künstlich aufgebläht wird und auch unter weniger performanter Hardware schnell läuft. Auf das Hinterlegen des Ortes bei Adressen wurde verzichtet, da diese aus den Postleitzahlen herleitbar sind.

Kunde(Emailadresse: varChar(100), Telefonnummer: varChar(30), Name: varChar(50), Passwort:varChar(100))

Adresse (Emailadresse: varChar(100), Postleitzahl varChar(10), Strasse: varChar(50), Hausnummer: INT, Adresszusatz: varChar(50), AdresseID:INT)

Waehlt(Emailadresse: varChar(100), ProduktID:INT)

Produkt(ProduktID:INT, Name:varChar(50), Beschreibungstext:varChar(250), Preis:REAL, Produktbild:varChar(250), AnbieterID:INT)

Beinhaltet(ProduktID:INT,BestellID:INT)

Bestellung(BestellID:INT,Rechnungsbetrag:REAL,Lieferdatum:DATE,Bestelldatum:DATE,Lieferstatus:varChar(30),Emailadresse: varChar(100), ZahlungsID:INT)

Anbieter(AnbieterID:INT,Name:varChar(50),Trustscore:INT,Website:varChar(50),Emailadresse:varChar(50))

Bewertung(BewertungID:INT,Bewertungstitel:varChar(50),Punkte:INT,Alter:INT,Bewertungstext:varChar(250),Datum:DATE, Emailadresse: varChar(100), ProduktID:INT)

Zahlungsmethode(ZahlungsID:INT,Inhabername:varChar(50), Emailadresse: varChar(100))

SEPA(ZahlungsID:INT,Inhabername:varChar(50), Emailadresse: varChar(100), IBAN:varChar(30))

PayPal(ZahlungsID:INT,Inhabername:varChar(50), Emailadresse: varChar(100), Kontoadresse:varChar(30))

Kreditkarte(ZahlungsID:INT,Inhabername:varChar(50), Emailadresse: varChar(100), Kartennummer:varChar(30),CVV:INT)

Gutschein(ZahlungsID:INT,Inhabername:varChar(50), Emailadresse: varChar(100),Code:varChar(15))

Alternativ bei Zahlungsmethode (wird auch so in Datenbank abgebildet):

Zahlungsmethode(ZahlungsID:INT,Inhabername:varChar(50),Zahlungsart:varChar(30), IBAN:varChar(30), Kontoadresse:varChar(30),Kartennummer:varChar(30),CVV:INT, Code:varChar(15), Emailadresse: varChar(100))

Aufgabenteil 4

Sämtliche SQL-bezogenen Abfragen, sowie die Erstellung der einzelnen Tabellen wurden mit der Software *„BeeKeeper-Studio“*[3] durchgeführt.

4.1 Erstellung der Datenbank

Folgende Screenshots zeigen die Befehle zum Erstellen der Tabellen. Anschließend wurden die einzelnen Tabellen mit konkreten Werten befüllt. Aus Gründen der Übersichtlichkeit wird in der Ausarbeitung nur jeweils ein Beispiel dargestellt. Alle weiteren Befehle, inklusive der Datenbank, befinden sich in den angehängten Dateien und in diesem GIT-Repo: https://github.com/philiplorenz-code/DatenbankenHausarbeit

4.1.1 Erstellen einer Tabelle

Durch den folgenden Befehl wird die Tabelle „Produkt“ angelegt. Diese beinhaltet das Attribut „ProduktID“ in Form des Primärschlüssels. „ProduktID“ wird dabei automatisch inkrementiert (hochgezählt), darf nicht „null“ sein und ist immer eindeutig. Daneben befinden sich weitere Attribute (vergleiche ER-Modell und Rationale Darstellung). Das Attribut „AnbieterID“ stellt den Fremdschlüssel dar, über welchen die Tabelle „Produkt“ mit den Einträgen aus Tabelle „Anbieter“ verknüpft werden kann:

PRODUKT.SQL

```sql
CREATE TABLE "Produkt" (
    "ProduktID"         INTEGER NOT NULL UNIQUE,
    "Name"              VARCHAR(50),
    "Beschreibungstext" VARCHAR(250),
    "Preis"             REAL,
    "Produktbild"       VARCHAR(250),
    "AnbieterID"        INTEGER,
    PRIMARY KEY("ProduktID" AUTOINCREMENT),
    FOREIGN KEY("AnbieterID") REFERENCES "Anbieter"("AnbieterID")
)
```

Abschließend wurden die Tabellen noch mit Werten befüllt. Die folgende Abbildung zeigt den INSERT-Befehl für die Anbieter-Tabelle. Jeder Anbieter erhält zu Beginn den Trust-Score 10:

INSERTVALUES > ANBIETER.SQL

```sql
INSERT INTO 'Anbieter' (Name,Trustscore,Website,Emailadresse) VALUES
('Moebel GmbH',    '10', 'moebel-gmbh.de',      'service@moebel-gmbh.de'),
('Adidas',         '10', 'adidas.de',           'info@adidas'),
('FanartikelShop','10', 'shop.fanartikel.de', 'request@fanartikel.de'),
('Angelzubehoer', '10', 'angel-zubehoer.de',   'ticket@angel-zubehoer.de'),
('Dampfershop',    '10', 'dampfer-shop.de',     'service@dampfer-shop.de')
```

[3] https://www.beekeeperstudio.io

Aufgabenteil 5

Im abschließenden Kapitel dieser Ausarbeitung werden verschiedene Abfragen auf die zuvor erstellte Datenbank durchgeführt. Ziel ist es die Abfragen sowohl in Form von SQL-Queries, als auch in Form von relationaler Algebra darzustellen. Zudem sollen die Ergebnisse, welche von der Datenbank geliefert werden, angegeben werden. Mein persönlicher Anspruch war es hierbei auf generische Abfragen zu verzichten und einen möglichst großen Praxisbezug herzustellen. Hinweis: viele der benutzten SQL-Statements lassen sich nicht detailgetreu in relationale Algebra übersetzen. Abweichung werden im Folgenden textlich gekennzeichnet.

Aufgabe 1: Es gibt eine allgemeine Störung bei den PayPal-Zahlungen. Der Automatismus zum Deaktivieren zur Zahlungsmethode hat aber nicht gegriffen, weshalb Kunden weiterhin auf die Methode zugreifen konnten (diese hat im Hintergrund aber nicht mehr funktioniert, weshalb die Zahlungstransaktionen nicht abgeschlossen werden konnten). Finde heraus welche Bestellungen am 29.11.2021 mittels PayPal durchgeführt wurden, sodass eine Korrektur im System vorgenommen werden kann!

SQL-Query:

```sql
SELECT
Bestellung.Bestelldatum,
Zahlungsmethode.Zahlungsart,
Zahlungsmethode.Emailadresse

FROM Bestellung

INNER JOIN Zahlungsmethode
ON Bestellung.ZahlungsID = Zahlungsmethode.ZahlungsID

WHERE Zahlungsmethode.Zahlungsart = 'PayPal'
AND Bestellung.Bestelldatum = '26.11.2021'
```

Ergebnis der Datenbank:

Bestelldatum	Zahlungsart	Emailadresse
26.11.2021	PayPal	hendrik.schmitt@web.de

Relationale Algebra:

```
π bestellung.bestelldatum, zahlungsmethode.zahlungsart, zahlungsmethode.emailadresse
σ zahlungsmethode.zahlungsart = "PayPal" ∧ bestellung.bestelldatum = "26.11.2021" (
  BESTELLUNG ⋈ bestellung.zahlungsid = zahlungsmethode.zahlungsid ZAHLUNGSMETHODE
)
```

Aufgabe 2: Welche Kunden kauften ein Produkt, dessen Preis über 20€ liegt? Gib sowohl die Kunden als auch das bestellte Produkt an.

SQL-Query:

```sql
SELECT Kunde.Vorname || ' ' || Kunde.Nachname AS Kundenname, Produkt.Name
FROM Produkt

INNER JOIN Beinhaltet
ON Produkt.ProduktID = Beinhaltet.ProduktID

INNER JOIN Bestellung
ON Bestellung.BestellID = Beinhaltet.BestellID

INNER JOIN Kunde
ON Kunde.Emailadresse = Bestellung.Emailadresse

WHERE Produkt.Preis > 20
```

Ergebnis der Datenbank:

Kundenname	Name
Hendrik Schmitt	Vape
Max Mustermann	Lampe
Helga Maier	Sportshirt

Relationale Algebra:

```
π kunde.vorname, kunde.nachname, produkt.name
σ produkt.preis > 20 (
  PRODUKT ⋈
  produkt.produktid = beinhaltet.produktid BEINHALTET ⋈
  BESTELLUNG.bestellid = beinhaltet.bestellid BESTELLUNG ⋈
  kunde.emailadresse = bestellung.emailadresse KUNDE
)
```

Aufgabe 3: Was ist der durchschnittliche Bestellwert? Welche Kunden liegen darüber?

SQL-Query:

Abfrage des Bestellwerts:

```
SELECT AVG(Rechnungsbetrag)
FROM Bestellung
```

Abfrage der Kunden, die über dem Durchschnitt bestellt hatten:

```
SELECT Kunde.Emailadresse, Kunde.Vorname, Kunde.Nachname

FROM Bestellung

INNER JOIN Kunde
ON Kunde.Emailadresse = Bestellung.Emailadresse

WHERE Bestellung.Rechnungsbetrag > (
  SELECT AVG(Rechnungsbetrag)
  FROM Bestellung
)
```

Ergebnis der Datenbank:

Bestellwert:

AVG(Rechnungsbetrag)
63.386

Kunden:

Emailadresse	Vorname	Nachname
helga-maier@mail.de	Helga	Maier

Relationale Algebra:

```
π kunde.emailadresse, kunde.vorname, kunde.nachname
σ bestellung.rechnungsbetrag > (π AVG(rechnungsbetrag))
(BESTELLUNG ⋈ kunde.emailadresse = bestellung.emailadresse KUNDE)
```

Hinweis: Inner Queries gibt es in relationaler Algebra nicht, weshalb die abgebildete Syntax nicht vollständig gültig ist.

Aufgabe 4: Amazon Marketplace updated sein Anti-Fraud-System. Bevor sie durch die neue Sicherheitsrichtlinie Punkte auf ihrem Trust-Score Konto abgezogen bekommen, müssen alle Anbieter mit einem Trustscore unter 40 ermittelt werden, die Produkte anbieten, welchen kein Bild zugeordnet ist. Konkret sollen die Mails, der entsprechenden Anbieter ermittelt werden, da an die Query anknüpfend automatisiert eine Mail an die Anbieter gesendet werden soll, sodass diese ihren Produkten Bilder hinzufügen können, bevor ihr Trustscore negativ belastet wird.

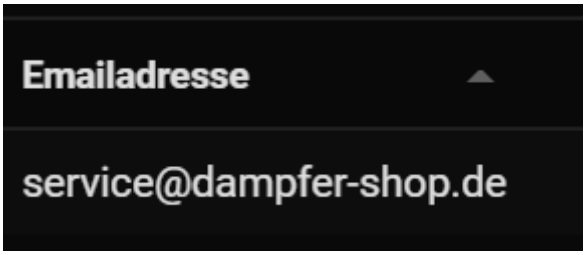
SQL-Query:

```sql
SELECT Anbieter.Emailadresse
FROM Produkt

INNER JOIN Anbieter
ON Anbieter.AnbieterID = Produkt.AnbieterID

WHERE Produkt.Produktbild = '' AND Anbieter.Trustscore < 40
```

Ergebnis der Datenbank:



Relationale Algebra:

```
π anbieter.emailadresse
σ produkt.produktbild = "" ∧ anbieter.trustscore < 40
(PRODUKT ⋈ anbieter.anbieterid = produkt.anbieterid ANBIETER)
```

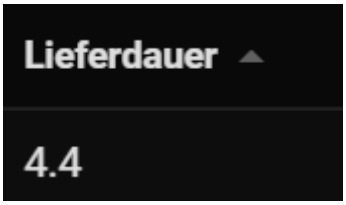
Hinweis: Leere Strings bzw NULL gibt es in relationaler Algebra nicht, weshalb die abgebildete Syntax nicht vollständig gültig ist.

Aufgabe 5: Wie lange dauert eine Lieferung im Schnitt? (Differenz zwischen Bestell- und Lieferdatum)

SQL-Query:

```
SELECT AVG(Bestellung.Lieferdatum - Bestellung.Bestelldatum) AS Lieferdauer
FROM Bestellung
```

Ergebnis der Datenbank:



Relationale Algebra:

```
π AVG (lieferdatum) - AVG (bestelldatum) BESTELLUNG
```

Hinweis: Mathematische Operatoren können in relationaler Algebra nicht verwendet werden, weshalb die abgebildete Syntax nicht vollständig gültig ist.