Projektaufgabe
„haushaltsnahe Dienstleistungen“

HOCHSCHULE
DER MEDIEN

Erstellen Sie ein Szenario, das einen von Ihnen gewählten Ausschnitt einer Datenbank zur Unterstützung einer Plattform, über die haushaltsnahe Dienstleistungen vermittelt werden. Dabei handelte es sich z.B. um die Vermittlung von Handwerkern, Haushaltshilfen, Hausmeisterdienstleistungen oder stundenweise Betreuung von Kindern u.v.m. Der Begriff ist also weitergefasst als im steuerrechtlichen Sinn. Mögliche Ansätze für das Datenmodell sind:

- Verwaltung der Anbieter von haushaltsnahen Dienstleistungen oder
- Verwaltung der Kunden und deren Anfragen zu Dienstleistungen oder
- Abbildung der Vertragsbeziehung,
- Abbildung einer Rückwärtsauktion, um den besten Preis zu ermitteln oder, oder, oder ...

Diese Ansätze sind nur unverbindliche Vorschläge. Sie sind in der inhaltlichen Gestaltung des Datenmodells sehr frei. Auch kann Ihr Modell mehrere Aspekte der oben genannten Ansätze umsetzen.

Die Projektaufgabe muss mindestens 8 und soll maximal 12 Tabellen umfassen. **Sie muss in Gruppenarbeit mit 3 oder 4 Personen bearbeitet werden.** Die Anzahl der Spalten in den Tabellen kann begrenzt werden, wenn dies keinen Einfluss auf die Modellbildung hat. Beispielsweise müssen nicht alle Eigenschaften einer Dienstleistung oder von Personen abgebildet werden. Es müssen jedoch im Gesamtmodell die Datentypen ganze Zahlen, Fließkommazahlen, Datum, Zeichenkette mit fester Länge und Zeichenkette mit einer variablen Länge vorkommen und der Gesamtkontext sichtbar sein. Im Datenmodell müssen sinnvolle Randbedingungen hinterlegt werden.

Erstellen Sie

- eine Systembeschreibung, die Ihre Entwurfsentscheidung erläutert (2 bis 4 DIN-A4-Seiten),
- ER-Diagramme des logischen und physischen Datenmodells,
- Restartfähige DDL-Skripte ohne kaskadierendes Löschen von Daten oder Tabellen und
- Insert-Skripte mit mindestens drei Datensätzen je Tabelle, die als Testdaten geeignet sind.

Das Datenmodell muss mindestens einen View enthalten, der Daten aus drei oder mehr Tabellen zusammenführt

Verwenden Sie das vorgegebene Deckblatt und folgende Formatvorgaben:

- Texte in Arial 11pt mit 1,5-fachem Zeilenabstand
- Skripte in Courier 10pt mit 1-fachem Zeilenabstand
- ER-Diagramme können auf dem Rechner mit einem Modellierungstool oder Zeichenprogramm erstellt oder als gescannte handschriftliche Zeichnung eingebunden werden.

Die Abgabe erfolgt in Papierform (1 Exemplar je Gruppe) mit Unterschrift aller Teilnehmer der Projektgruppe unmittelbar vor Start der Klausur bei der Klausuraufsicht und elektronisch in Form einer einzelnen PDF-Datei, die per Mail bis spätestens 24h nach der Klausur gem. Klausurenplan an den Dozenten zu senden ist und mit der bei der Klausur abgegebenen Version übereinstimmen muss.

In der PDF-Datei (verpflichtender Dateiname mit xx=Gruppennummer: Gruppe_xx.pdf) muss das Kopieren der Skript-Elemente zugelassen sein, so dass die Skript-Elemente in ein SQL-Tool übertragen werden können. Die Lauffähigkeit der Skripte ist Teil der Bewertung und wird mit dem SQL-Tool aus PGAdmin und dem DBMS Postgres 16. Die Art (Qualität) der grafischen Darstellung der ER-Diagramme ist dann für die Bewertung relevant, wenn die Lesbarkeit eingeschränkt ist. Die Originalität und die formale Korrektheit des Entwurfs spielt hingegen eine wesentliche Rolle.

Mit der Projektaufgabe haushaltsnahe Dienstleistungen kann 1/5 der Punkte der Prüfungsleistung im Modul 113222 erreicht werden. In der Bearbeitungsphase werden weder vom Dozenten noch vom Betreuer der Übungen Feedbackfragen zum Lösungsentwurf beantwortet.

Viel Erfolg!

Prof. Dr. Mathias Hinkelmann

Projektaufgabe

„haushaltsnahe Dienstleistungen“

Teil der Prüfungsleistung im Modul

Datenbanken (EDV-Nr. 113222)

Wintersemester 2024

Projektgruppe 14 mit den Gruppenteilnehmer:

Demharter, Luca	5012624	Medieninformatik
Curca, Julian	5015941	Medieninformatik
Zahn, Jannes	5016598	Medieninformatik
Hörmann, Sandro	5013489	Medieninformatik

Hiermit bestätigen die oben genannten Gruppenteilnehmer, dass die Projektaufgabe E-Bike-Leasing ohne Unterstützung dritter und unter Nennung aller Quellen und eingesetzten Hilfsmittel erstellt wurde. Der Einsatz von generativer KI ist explizit ausgeschlossen. Es ist uns bekannt, dass ein Verstoß gem. § 17 Abs. 5 Satz 1 zum Nichtbestehen der kompletten Prüfungsleistung im Modul Datenbanken (EDV-Nr. 113222) führt und der Einsatz von generativer KI einen schwerwiegenden Täuschungsversuch darstellt, der zum Ausschluss von der Erbringung weiterer Prüfungsleistungen und somit zum Verlust des Prüfungsanspruchs führen kann.

Stuttgart, den

Demharter, Luca

Curca, Julian

Zahn, Jannes

Hörmann, Sandro

Bewertungsschema (bitte als letzte Seite der Ausarbeitung einbinden):

Kriterium		max.	ist	Kommentar
Originalität / Einhaltung der formalen Vorgaben		20		
Relative Qualität		10		
Spezifikation / Systembeschreibung		15		
ER-Modell inkl. formaler Korrektheit im Sinn der Normalisierung		15		
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		
	View mit einem Join über mehr als 2 Tabellen	5		
Summe				

Projektaufgabe

„haushaltsnahe Dienstleistungen“

Meldung als Projektgruppe für den

Teil „Projektaufgabe“ der Prüfungsleistung im Modul

Datenbanken (EDV-Nr. 113222)

Wintersemester 2024

Wir bilden eine Projektgruppe mit den Gruppenteilnehmern:

Demharter, Luca	5012624	Medieninformatik
Curca, Julian	5015941	Medieninformatik
Zahn, Jannes	5016598	Medieninformatik
Hörmann, Sandro	5013489	Medieninformatik

Zugewiesene Gruppennummer: 14

Systembeschreibung zum Datenbank-Modell der online Plattform zur Vermittlung von Haushaltshilfen

1. Abgrenzung

Die nachfolgend beschriebene Implementierung des Datenmodells ist ein vereinfachtes Beispiel, das die Zusammenhänge in einer Vermittlungsplattform aufzeigen soll. Funktionen zur Zugriffssteuerung, Anmeldung, Sicherheit und Identitätsprüfung von Nutzern werden nicht berücksichtigt, obwohl sie für eine Plattform, auf der sich kurzfristig Arbeitsuchende anmelden können, üblicherweise erforderlich wären. So ließen sich kriminelle Aktionen und gesetzeswidrige Handlungen besser nachverfolgen.

2. Funktionsprinzip: Erläuterung zur Plattform

Die Projektaufgabe beschreibt das Datenbankmodell einer Onlineplattform oder App, die hilfsbedürftige Kunden mit flexiblen Arbeitskräften aus dem Pflege- und Haushaltsdienstbereich zusammenbringt. Dienstleistungen werden von den Kunden angefragt und als „Service-Requests“ veröffentlicht, wobei die Kategorien (z. B. Einkaufen, Gartenarbeit, medizinische Unterstützung) nicht frei definierbar sind. Ein Beschreibungsfeld ermöglicht nähere Angaben zur Tätigkeit. Selbstständige Pfleger, Ferienjobber oder Privatpersonen können Angebote abgeben und sich so auf die Anfragen bewerben. Die Plattform ist auf Deutschland beschränkt und nimmt nur deutsche Wohnorte an.

3. Genereller Entwurf

Bei der Registrierung werden Kunden im Entitätstyp Kunden gespeichert, mit Pflichtfeldern wie Vor- und Nachname, Alter, Kontaktdaten und Wohnortadresse (letztere in einer separaten Adressen-Tabelle). Pfleger werden nach denselben Vorgaben erfasst, verfügen jedoch zusätzlich über ein Credits-Feld. Dienstleistungen sind in einer statischen Entität hinterlegt und enthalten Kategorien, Beschreibung und Preis; falls „Essen“ gewählt wird, kann eine Mahlzeit zugeordnet werden. Anzeigen referenzieren die Dienstleistung per Foreign Key und führen Anfrage-Datum, Abschluss-Datum sowie Status. Für jede Dienstleistung ist eine eigene Anzeige nötig, auch wenn mehrere Aufgaben theoretisch von einer Person übernommen werden könnten. Auf die Anzeigen antworten Pflegekräfte mit Angeboten; der Kunde wählt eines aus, dessen Preis fest in der Dienstleistungstabelle steht. Bei Annahme entsteht eine Buchung mit Verweis auf Angebot, Kunde, Pfleger und Dienstleistung. Abschließend dokumentiert eine Entität Bezahlung den Zahlungseingang, indem sie Buchung und Dienstleistung referenziert.

4. Dokumentation der Entitätstypen

4.1 Normierungsdaten

Entität Kundenadressen

Speichert die Adressen eines Kunden.

- kundenadresse_id (Pflicht, Integer, Primärschlüssel): Eindeutige ID der Kundenadresse.
- plz (Pflicht, Text, genau 5 Zeichen): Postleitzahl der Adresse.
- strasse (Pflicht, Text, max. 255 Zeichen): Straßenname und Hausnummer.
- ort (Pflicht, Text, max. 100 Zeichen): Name des Wohnorts.

Trigger:

- Insert-Trigger: Die Postleitzahl muss genau 5 Zeichen lang sein. Falls nicht, wird eine Exception geworfen (Fehlernummer -30003, Meldung "Invalid ZIP code length!").

Entität Kunden

Repräsentiert einen Kunden, der eine Dienstleistung in Anspruch nimmt.

- kunde_id (Pflicht, Integer, Primärschlüssel): Eindeutige ID des Kunden.
- vorname (Pflicht, Text, max. 100 Zeichen): Vorname des Kunden.
- nachname (Pflicht, Text, max. 100 Zeichen): Nachname des Kunden.
- geb_datum (Pflicht, Datum): Geburtsdatum des Kunden (muss zwischen 01.01.1900 und dem aktuellen Datum liegen).
- tel_nr (Pflicht, Text, genau 11 Zeichen): Telefonnummer des Kunden.
- email (Pflicht, Text, max. 255 Zeichen): E-Mail-Adresse des Kunden.
- kundenadressen (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität kundenadressen.

Trigger:

- Insert-Trigger: Das Geburtsdatum muss vor dem aktuellen Datum und nach dem 01.01.1900 liegen. Falls nicht, wird eine Exception geworfen (Fehlernummer -30001, Meldung "Invalid birth date!").
- Update-Trigger: Die Telefonnummer muss genau 11 Zeichen lang sein. Falls nicht, wird eine Exception geworfen (Fehlernummer -30002, Meldung "Invalid phone number length!").

Entität Pfleger

Repräsentiert eine Pflegekraft, die Dienstleistungen erbringt.

- pfleger_id (Pflicht, Integer, Primärschlüssel): Eindeutige ID des Pflegers.
- vorname (Pflicht, Text, max. 100 Zeichen): Vorname des Pflegers.
- nachname (Pflicht, Text, max. 100 Zeichen): Nachname des Pflegers.

- geb_datum (Pflicht, Datum): Geburtsdatum des Pflegers (muss zwischen 01.01.1900 und dem aktuellen Datum liegen).
- tel_nr (Pflicht, Text, genau 11 Zeichen): Telefonnummer für Kontaktaufnahmen.
- credits (Pflicht, Integer, Standard 0): Erfahrungspunkte basierend auf geleisteten Dienstleistungen (muss >= 0 sein).

Entität Mahlzeiten
Repräsentiert eine Mahlzeit als spezielle Dienstleistung.

- mahlzeit_id (Pflicht, Integer, Primärschlüssel): Eindeutige ID der Mahlzeit.
- beschreibung (Pflicht, Text): Beschreibung der Mahlzeit.
- preis (Pflicht, Dezimal(10,2), >= 0): Preis der Mahlzeit.

Entität Dienstleistungen
Repräsentiert verschiedene Arten von angebotenen Dienstleistungen.

- dienstleistung_id (Pflicht, Integer, Primärschlüssel): Eindeutige ID der Dienstleistung.
- bezeichnung (Pflicht, Text, max. 100 Zeichen): Name der Dienstleistung.
- beschreibung (Pflicht, Text): Beschreibung der Dienstleistung.
- betrag (Pflicht, Dezimal(10,2), >= 0): Kosten der Dienstleistung.
- mahlzeit (Optional, Fremdschlüssel, Integer): Referenz auf die Entität mahlzeiten.

3.2 Bewegungsdaten
Entität Anzeigen
Repräsentiert eine Kundenanfrage für eine Dienstleistung.

- anzeige_id (Pflicht, Integer, Primärschlüssel): Eindeutige ID der Anfrage.
- kunde (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität kunden.
- dienstleistung (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität dienstleistungen.
- beschreibung (Optional, Text): Beschreibung der Anfrage.
- aufgabe_datum (Optional, Datum): Datum der Anfrage.
- abschluss_datum (Optional, Datum): Datum des Abschlusses der Anfrage.
- status (Pflicht, Text): Status der Anfrage ("offen", "akzeptiert", "abgeschlossen").

Trigger:

- Insert-Trigger: Das Aufgabendatum darf nicht in der Zukunft liegen. Falls doch, wird eine Exception geworfen (Fehlernummer -30007, Meldung "Task date cannot be in the future!").
- Update-Trigger: Der Status der Anzeige darf nur von "offen" auf "akzeptiert" oder "abgeschlossen" geändert werden. Ein Wechsel zu "abgelehnt" erfordert zusätzliche Validierungen, z.B. das Setzen eines Abschlussdatums.

- Delete-Trigger: Eine Anzeige kann nicht gelöscht werden, wenn bereits ein Angebot für diese Anzeige existiert.

Entität Angebote
Repräsentiert ein Angebot eines Pflegers für eine Anzeige.

- angebot_id (Pflicht, Integer, Primärschlüssel): Eindeutige ID des Angebots.
- pfleger (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität pfleger.
- anzeige (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität anzeigen.
- status (Optional, Text): Status des Angebots ("offen", "angenommen", "abgelehnt").
- beschreibung (Optional, Text): Beschreibung des Angebots.

Trigger:

- Insert-Trigger: Ein Angebot darf nur erstellt werden, wenn die zugehörige Anzeige den Status "offen" hat. Falls nicht, wird eine Exception geworfen (Fehlernummer -30008, Meldung "Cannot create offer for non-open advertisement!").
- Update-Trigger: Der Status des Angebots kann nur von "offen" auf "angenommen" geändert werden, wenn auch der Status der Anzeige auf "akzeptiert" geändert wurde. Andernfalls wird eine Exception geworfen (Fehlernummer -30010, Meldung "Cannot accept offer for non-open advertisement!").
- Delete-Trigger: Ein Angebot kann nicht gelöscht werden, wenn es bereits akzeptiert oder abgelehnt wurde, oder wenn es keine gültige Verbindung zu einer Anzeige mehr hat.

Entität Buchungen
Beschreibt eine Buchung einer Dienstleistung oder eines Angebots.

- buchung_id (Pflicht, Integer, Primärschlüssel): Eindeutige ID der Buchung.
- angebot (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität angebote.
- kunde (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität kunden.
- pfleger (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität pfleger.
- dienstleistung (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität dienstleistungen.
- buchungsdatum (Pflicht, Datum): Datum, an dem die Buchung vorgenommen wurde.

Trigger:

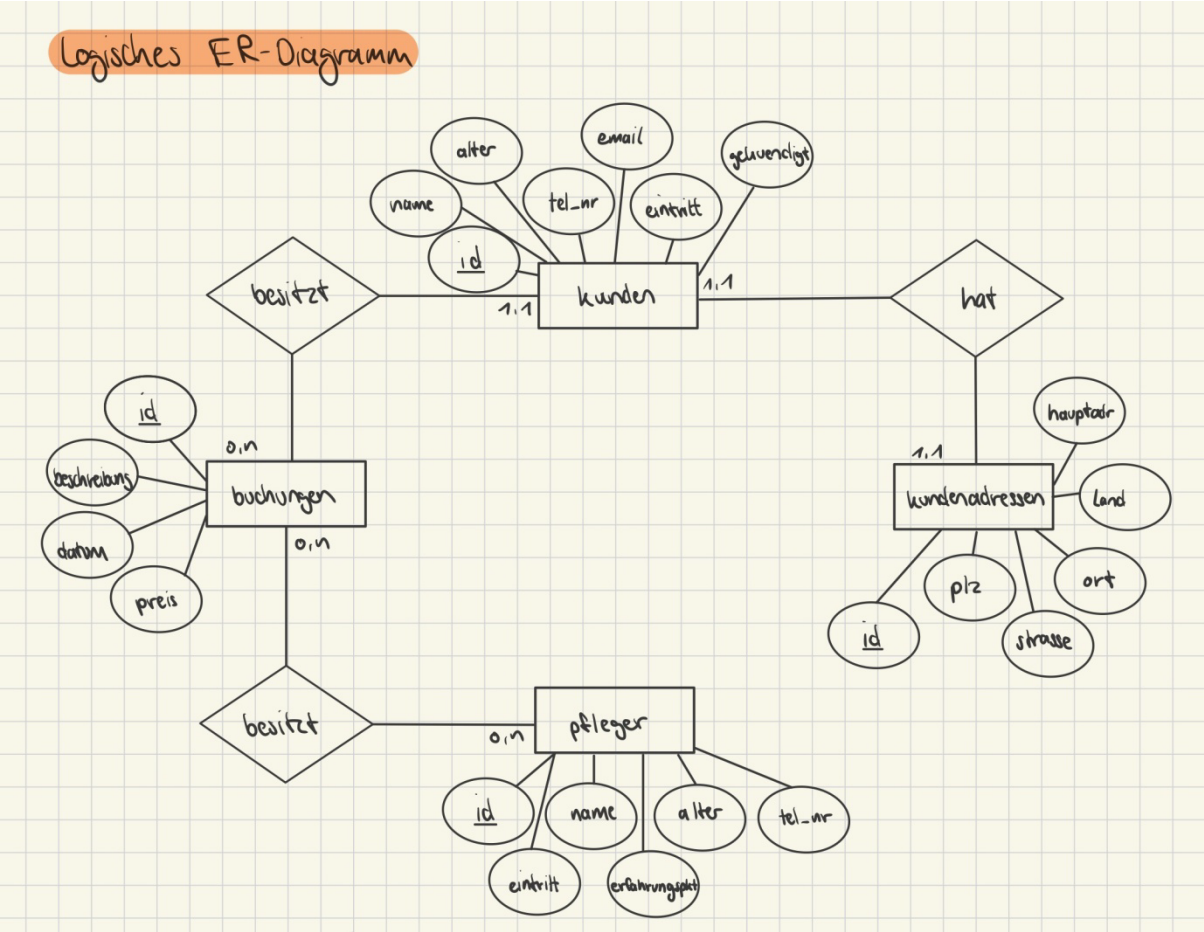
- Insert-Trigger: Das Buchungsdatum darf nicht in der Vergangenheit liegen. Falls doch, wird eine Exception geworfen (Fehlernummer -30005, Meldung "Booking date cannot be in the past!").

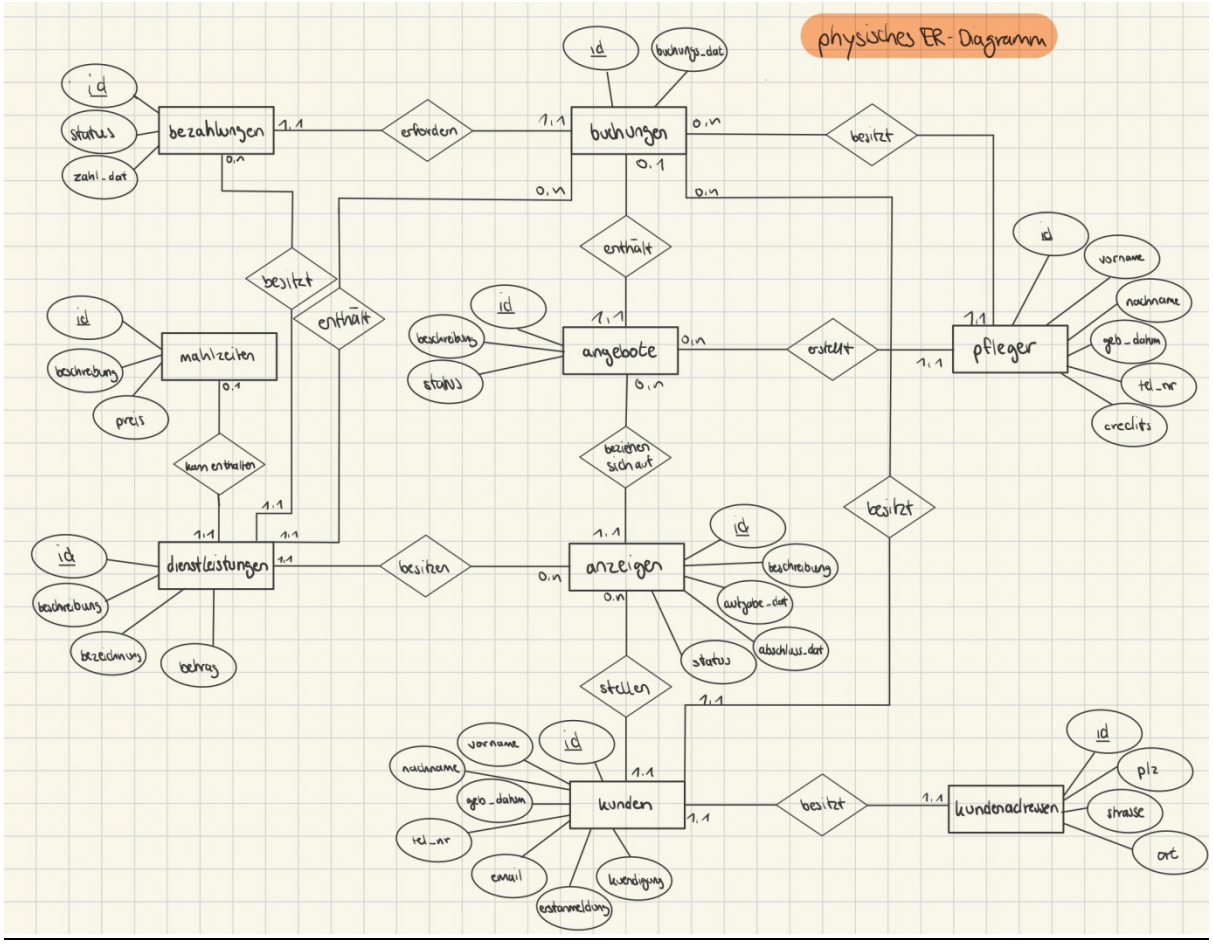
Entität Bezahlungen
Repräsentiert eine Zahlung innerhalb des Systems.

- bezahlung_id (Pflicht, Integer, Primärschlüssel): Eindeutige ID der Zahlung.
- buchung (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität buchungen.

- dienstleistung (Pflicht, Fremdschlüssel, Integer): Referenz auf die Entität dienstleistungen.
- status (Pflicht, Text): Status der Zahlung ("offen", "bezahlt", "fehlgeschlagen").
- zahlungsdatum (Optional, Datum): Datum der Zahlung.
  Trigger:
- Insert-Trigger: Eine Zahlung darf nur dann als "bezahlt" markiert werden, wenn eine Buchung existiert. Falls nicht, wird eine Exception geworfen (Fehlernummer -30006, Meldung "Cannot mark payment as paid without valid booking!").

5. ER Diagramme





6. Restartfähige DDL Skripte

```
DROP TABLE IF EXISTS bezahlungen;

DROP TABLE IF EXISTS buchungen;

DROP TABLE IF EXISTS angebote;

DROP TABLE IF EXISTS anzeigen;

DROP TABLE IF EXISTS mahlzeiten;

DROP TABLE IF EXISTS kunden;

DROP TABLE IF EXISTS pfleger;

DROP TABLE IF EXISTS dienstleistungen;

DROP TABLE IF EXISTS kundenadressen;
```

```
CREATE TABLE kundenadressen(
 kundenadresse_id            SERIAL,
 plz                         VARCHAR(5)     NOT NULL   CHECK (LENGTH(plz) = 5),
 strasse                     VARCHAR(255)   NOT NULL,
 ort                         VARCHAR(100)   NOT NULL,
 PRIMARY KEY (kundenadresse_id)
);
```

```
CREATE TABLE kunden (
  kunde_id              SERIAL,
  vorname               VARCHAR(100)      NOT NULL,
  nachname              VARCHAR(100)      NOT NULL,
  geb_datum             DATE              NOT NULL
      CHECK(geb_datum < CURRENT_DATE AND geb_datum > DATE '1900-1-1'),
  tel_nr                VARCHAR(11)       NOT NULL
      CHECK (LENGTH(tel_nr) = 11),
  email                 VARCHAR(255)      NOT NULL,
  kundenadressen        SERIAL            NOT NULL,
  PRIMARY KEY(kunde_id),
  FOREIGN KEY (kundenadressen) REFERENCES kundenadressen(kundenadresse_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT
);

CREATE TABLE pfleger (
  pfleger_id            SERIAL,
  vorname               VARCHAR(100)      NOT NULL,
  nachname              VARCHAR(100)      NOT NULL,
  geb_datum             DATE              NOT NULL
      CHECK(geb_datum < CURRENT_DATE AND geb_datum > DATE '1900-1-1'),
  tel_nr                VARCHAR(11)       NOT NULL
      CHECK (LENGTH(tel_nr) = 11),
  credits               INT               NOT NULL    DEFAULT 0
      CHECK (credits >= 0),
  PRIMARY KEY(pfleger_id)
);

CREATE TABLE mahlzeiten (
  mahlzeit_id           SERIAL,
  beschreibung          TEXT              NOT NULL,
  preis                 DECIMAL(10,2)     NOT NULL    CHECK (preis >=0),
  PRIMARY KEY (mahlzeit_id)
);

CREATE TABLE dienstleistungen (
  dienstleistung_id     SERIAL,
  bezeichnung           VARCHAR(100)      NOT NULL,
  beschreibung          TEXT              NOT NULL,
  betrag                DECIMAL(10,2)     NOT NULL
    CHECK (betrag >= 0),
  mahlzeit              SERIAL,
  PRIMARY KEY (dienstleistung_id),
  FOREIGN KEY (mahlzeit) REFERENCES mahlzeiten(mahlzeit_id)
);

CREATE TABLE anzeigen (
  anzeige_id            SERIAL,
  kunde                 SERIAL            NOT NULL,
  dienstleistung        SERIAL            NOT NULL,
  beschreibung          TEXT              NULL,
  aufgabe_datum         DATE              NULL,
  abschluss_datum       DATE              NULL,
  status                VARCHAR(50)
      CHECK (status IN('offen', 'akzeptiert', 'abgeschlossen')),
  PRIMARY KEY (anzeige_id),
  FOREIGN KEY (kunde) REFERENCES kunden(kunde_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT,
  FOREIGN KEY (dienstleistung) REFERENCES dienstleistungen(dienstleistung_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT );
```

```
CREATE TABLE angebote (
  angebot_id           SERIAL,
  pfleger              SERIAL           NOT NULL,
  anzeige              SERIAL           NOT NULL,
  status               VARCHAR(50)      NULL
   CHECK(status IN('offen', 'angenommen', 'abgelehnt')),
  beschreibung         TEXT             NULL,
  PRIMARY KEY (angebot_id),
  FOREIGN KEY (pfleger) REFERENCES pfleger(pfleger_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT,
  FOREIGN KEY (anzeige) REFERENCES anzeigen(anzeige_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT
);

CREATE TABLE buchungen (
  buchung_id           SERIAL,
  angebot              SERIAL           NOT NULL,
  kunde                SERIAL           NOT NULL,
  pfleger              SERIAL           NOT NULL,
  dienstleistung       SERIAL           NOT NULL,
  buchungsdatum        DATE             NOT NULL,
  PRIMARY KEY (buchung_id),
  FOREIGN KEY (angebot) REFERENCES angebote(angebot_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT,
  FOREIGN KEY (kunde) REFERENCES kunden(kunde_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT,
  FOREIGN KEY (pfleger) REFERENCES pfleger(pfleger_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT,
  FOREIGN KEY (dienstleistung) REFERENCES dienstleistungen(dienstleistung_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT
);

CREATE TABLE bezahlungen (
  bezahlung_id         SERIAL,
  buchung              SERIAL           NOT NULL,
  dienstleistung       SERIAL           NOT NULL,
  status               VARCHAR(50)      NULL,
  zahlungsdatum        DATE             NULL,
  PRIMARY KEY (bezahlung_id),
  FOREIGN KEY (buchung) REFERENCES buchungen(buchung_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT,
  FOREIGN KEY (dienstleistung) REFERENCES dienstleistungen(dienstleistung_id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT
);
```

```
-- Views (Zahlungsüberblick und Auskunft zu Angeboten je Anzeige)

CREATE OR REPLACE VIEW zahlungsdetails_bezahlt AS SELECT

kunden.vorname AS vorname_kunde,
kunden.nachname AS nachname_kunde,
pfleger.vorname AS vorname_pfleger,
pfleger.nachname AS nachname_pfleger,
dienstleistungen.bezeichnung AS dienstleistung,
dienstleistungen.beschreibung AS beschreibung,
dienstleistungen.betrag AS betrag,
buchungen.buchung_id AS buchungs_id,
bezahlungen.zahlungsdatum AS zahldatum

      FROM buchungen

LEFT JOIN bezahlungen ON buchungen.buchung_id = bezahlungen.buchung
LEFT JOIN dienstleistungen ON buchungen.dienstleistung =
      dienstleistungen.dienstleistung_id
LEFT JOIN kunden ON buchungen.kunde= kunden.kunde_id
LEFT JOIN pfleger ON buchungen.pfleger= pfleger.pfleger_id

      WHERE bezahlungen.status = 'bezahlt';



CREATE OR REPLACE VIEW zahlungsdetails_offen AS SELECT

kunden.vorname AS vorname_kunde,
kunden.nachname AS nachname_kunde,
pfleger.vorname AS vorname_pfleger,
pfleger.nachname AS nachname_pfleger,
dienstleistungen.bezeichnung AS dienstleistung,
dienstleistungen.beschreibung AS beschreibung,
dienstleistungen.betrag AS betrag,
buchungen.buchung_id AS buchungs_id,
bezahlungen.zahlungsdatum AS zahldatum

      FROM buchungen

LEFT JOIN bezahlungen ON buchungen.buchung_id = bezahlungen.buchung
LEFT JOIN dienstleistungen ON buchungen.dienstleistung =
      dienstleistungen.dienstleistung_id
LEFT JOIN kunden ON buchungen.kunde = kunden.kunde_id
LEFT JOIN pfleger ON buchungen.pfleger = pfleger.pfleger_id

      WHERE bezahlungen.status = 'offen';



CREATE OR REPLACE VIEW angebote_ueberblick AS SELECT

angebote.angebot_id AS angebot,
angebote.beschreibung AS beschreibung,
angebote.status AS status,
anzeigen.anzeige_id AS bezogen_auf_Anzeige,
anzeigen.aufgabe_datum AS vom,
pfleger.vorname AS vorname_pfleger,
pfleger.nachname AS nachname_pfleger,
dienstleistungen.bezeichnung AS dienstleistung,
anzeigen.beschreibung AS anzeigen_beschreibung
```

```
      FROM angebote

LEFT JOIN anzeigen ON angebote.anzeige = anzeigen.anzeige_id
LEFT JOIN pfleger ON angebote.pfleger = pfleger.pfleger_id
LEFT JOIN dienstleistungen ON anzeigen.dienstleistung =
      dienstleistungen.dienstleistung_id;
```

7. Testdaten

```
INSERT INTO kundenadressen (kundenadresse_id, plz, strasse, ort) VALUES
(1,'10115','Musterstraße 1','Berlin'),
(2,'60549','Beispielweg 10','Frankfurt'),
(3,'80331','Hauptplatz 5','München');

INSERT INTO kunden (kunde_id,vorname, nachname, geb_datum, tel_nr, email, kun-
denadressen) VALUES
(1,'Max','Muster','1980-05-12','03012345671','max@muster.de',1),
(2,'Anna','Beispiel','1975-11-02','06998765431','anna@example.com',2),
(3,'John','Doe','1990-01-20','08922244441','john.doe@example.com',3);

INSERT INTO pfleger (pfleger_id,vorname, nachname, geb_datum, tel_nr, credits)
VALUES
(1,'Peter','Pfleger','1985-07-10','01761234567',10),
(2,'Julia','Hilfsbereit','1992-03-25','01769876543',25),
(3,'Michael','Support','1978-12-01','01765554433',5);

INSERT INTO mahlzeiten (mahlzeit_id, beschreibung, preis) VALUES
(1, 'Spaghetti Bolognese', 8.50),
(2, 'Gemüsepfanne', 7.00),
(3, 'Schnitzel mit Pommes', 9.90);

INSERT INTO dienstleistungen (dienstleistung_id,bezeichnung, beschreibung, be-
trag,mahlzeit) VALUES
(1,'Einkaufen','Einkauf von Lebensmitteln und Waren',20.00,1),
(2,'Gartenarbeit','Pflege von Garten und Pflanzen',15.00,2),
(3,'Essen','Zubereitung von Mahlzeiten',30.00,3);


INSERT INTO anzeigen (anzeige_id,kunde, dienstleistung, beschreibung, aufga-
be_datum, abschluss_datum, status) VALUES
(1,1, 1, 'Brauche Hilfe beim Einkaufen', '2025-02-01', NULL, 'offen'),
(2,2, 2, 'Gartenarbeit dringend gesucht', '2025-02-05', NULL, 'offen'),
(3,3, 3, 'Suche Koch für 1 Woche', '2025-02-06', NULL, 'offen');

INSERT INTO angebote (angebot_id,pfleger, anzeige, status, beschreibung) VALUES
(1,1, 1, 'offen', 'Ich kann die Einkäufe übernehmen'),
(2,2, 2, 'offen', 'Habe viel Erfahrung in Gartenpflege'),
(3,3, 3, 'offen', 'Koche gerne abwechslungsreiche Gerichte');

INSERT INTO buchungen (buchung_id,angebot, kunde, pfleger, dienstleistung, bu-
chungsdatum) VALUES
(1,1, 1, 1, 1, '2025-02-02'),
(2,2, 2, 2, 2, '2025-02-06'),
(3,3, 3, 3, 3, '2025-02-07');

INSERT INTO bezahlungen (bezahlung_id,buchung, dienstleistung, status, zah-
lungsdatum) VALUES
(1,1, 1, 'offen', NULL),
(2,2, 2, 'bezahlt', '2025-02-08'),
(3,3, 3, 'offen', NULL);
```