Meilenstein 3: Logischer und Physischer Entwurf

Gruppe 30
Boiko Bohdana, Alexander Grath

Logischer und Physischer Entwurf

universität wien

1 Normalisierung (3. Normalform)

Fahrzeug

Fahrzeug(FahrzeugID, Baujahr, Kapazität)
{FahrzeugID → Baujahr, Kapaziät}

Da der Primärschlüssel aus einem einzigen Attribut besteht, liegen keine partiellen Abhängigkeiten vor (2NF). Zudem existieren keine transitiven Abhängigkeiten zwischen nicht-Schlüsselattributen (3NF).

Bus

Bus(*FahrzeugID*, Kennzeichen, Niederflur)
{FahrzeugID → Kennzeichen, Niederflur}

Der Primärschlüssel besteht aus einem einzigen Attribut (FahrzeugID), daher liegen keine partiellen Abhängigkeiten vor (2NF).
Zwischen den Nicht-Schlüsselattributen existieren keine transitiven Abhängigkeiten. FahrzeugID ist zugleich ein Fremdschlüssel, der auf die Relation Fahrzeug verweist, begründet jedoch keine transitive Abhängigkeit innerhalb dieser Relation (3NF).

Zug

Zug(*FahrzeugID*, Baureihe, Wagonzahl)
{FahrzeugID → Baureihe, Wagonzahl}

Da der Primärschlüssel einattributig ist, liegen keine partiellen Abhängigkeiten vor (2NF).
Zwischen den Nicht-Schlüsselattributen bestehen keine transitiven Abhängigkeiten. FahrzeugID fungiert als Fremdschlüssel auf die Relation Fahrzeug und stellt keine transitive Abhängigkeit dar (3NF).

Straßenbahn

Straßenbahn(*FahrzeugID*, Spurweite, Stromsystem)
{FahrzeugID → Spurweite, Stromsystem}

Der Primärschlüssel besteht aus einem einzigen Attribut, wodurch keine partiellen Abhängigkeiten auftreten (2NF).
Es existieren keine transitiven Abhängigkeiten zwischen Nicht-Schlüsselattributen. Die Referenz auf Fahrzeug über den Fremdschlüssel FahrzeugID beeinflusst die Normalform nicht (3NF).

Wartung

Wartung(WartungsID, Datum, Kosten, Art, *FahrzeugID*)
{WartungsID → Datum, Kosten, Art}

Der Primärschlüssel besteht aus einem einzigen Attribut. Damit haben wir keine partiellen Abhängigkeiten (2NF).
Es gibt keine transitiven Abhängigkeiten zwischen den Nicht-Schlüsselattributen. FahrzeugID ist ein Fremdschlüssel, löst aber keine transitive Abhängigkeit der anderen Attribute aus.

Personal

Personal(PersonalNr, Name, Gehalt, Einstellungsdatum)
{PersonalNr → Name, Gehalt, Einstellungsdatum}

Der Primärschlüssel ist einattributig, somit sind keine partiellen Abhängigkeiten vorhanden. (2NF) Es bestehen keine transitiven Abhängigkeiten zwischen Nicht-Schlüsselattributen. (3NF)

Fahrer

Fahrer(*PersonalNr*, Führerscheinklasse, *MentorNr*)
{PersonalNr → Führerscheinklasse, MentorNr}

Der Primärschlüssel besteht aus einem einzigen Attribut (PersonalNr), daher liegen keine partiellen Abhängigkeiten vor (2NF).
Das Attribut MentorNr ist ein Fremdschlüssel, der auf die Relation Fahrer verweist, stellt jedoch keine transitive Abhängigkeit zwischen Nicht-Schlüsselattributen dar. Alle Nicht-Schlüsselattribute hängen ausschließlich vom Primärschlüssel ab (3NF).

Kontrolleur

Kontrolleur(*PersonalNr*, Zuständigkeitsbereich)
{PersonalNr → Zuständigkeitsbereich}

Da der Primärschlüssel einattributig ist, liegen keine partiellen Abhängigkeiten vor (2NF). Es gibt nur ein einziges Nicht-Schlüsselattribut (Zustaendigkeitsbereich), daher kann keine transitive Abhängigkeit zwischen Nicht-Schlüsselattributen existieren (3NF).

Halstestelle

Halstestelle(HalstestellenID, Name, Barrierefrei)
{HaltestellenID → Name, Barrierefrei}

Der Primärschlüssel besteht aus einem Attribut, wodurch keine partiellen Abhängigkeiten auftreten (2NF). Es liegen keine transitiven Abhängigkeiten zwischen Nicht-Schlüsselattributen vor (3NF).

Linie

Linie(LinienNr, Taktung, Betriebsbeginn, Betriebsende)
{LinienNr → Taktung, Betriebsbeginn, Betriebsende}

Da der Primärschlüssel einattributig ist, existieren keine partiellen Abhängigkeiten (2NF).
Zwischen den Nicht-Schlüsselattributen bestehen keine transitiven Abhängigkeiten (3NF).

Linie_Haltestelle

Linie_Haltestelle(*LinienNr*, *HaltestellenID*, Reihenfolge)
{LinienNr, HaltestellenID → Reihenfolge}

Der Primärschlüssel ist zusammengesetzt und besteht aus den Attributen LinienNr und HaltestellenID. Das Attribut Reihenfolge hängt vollständig von beiden Attributen des Primärschlüssels ab, sodass keine partiellen Abhängigkeiten vorliegen (2NF).
Da keine weiteren funktionalen Abhängigkeiten zwischen Nicht-Schlüsselattributen existieren, liegen auch keine transitiven Abhängigkeiten vor (3NF).

Kunde

Kunde(<u>KundenID</u>, Name, Email, Adresse)
{KundenID → Name, Email, Adresse}
{Email → KundenID, Name, Adresse} (Email UNIQUE)

Da sowohl KundenID als auch Email Schlüsselkandidaten sind, liegen keine partiellen oder transitiven Abhängigkeiten vor (3NF).

Kunde - Adresse, Erweiterung

Adresse alleine ist nicht sehr sinnvoll. Es wird daher mit Strasse, HausNr, PLZ und Ort ersetzt.
Kunde(<u>KundenID</u>, Name, Email, Strasse, HausNR, PLZ, Ort)
{KundenID → Name, Email, Strasse, HausNR, PLZ, Ort}

Jetzt haben wir ein Problem. Mit der Rechtsreduktion erkennen wir: KundenID → PLZ → Ort. Ort ist jetzt redundant.
Ort wird gestrichen.
{KundenID → Name, Email, Strasse, HausNR, PLZ}
{PLZ → Ort}

Wir haben jetzt zwei neue Relationen R_1 Kunde und R_2 Stadt:

Kunde(<u>KundenID</u>, Name, Email, Strasse, HausNR, *PLZ*)
Primary Key: KundenID
Foreign Key: PLZ ⋄ Stadt.PLZ

Ort(<u>PLZ</u>, Ort)
Primary Key: PLZ

Damit sind die transitiven Abhängigkeiten wieder entfernt und wir haben die 3. Normalform erreicht.

Physisches Modell, überführen

R_2

```
CREATE TABLE Stadt (
  PLZ      VARCHAR2(4) NOT NULL,
  Ort      VARCHAR2(50) NOT NULL,
  CONSTRAINT PK_Stadt PRIMARY KEY (PLZ)
);
```

R_1

```
CREATE TABLE Kunde (
  KundenID INTEGER,
  Name     VARCHAR2(50) NOT NULL,
  Email    VARCHAR2(100) NOT NULL,
  Strasse  VARCHAR2(50),
  Hausnr   VARCHAR2(10),
  PLZ      VARCHAR2(4),
  CONSTRAINT PK_Kunde PRIMARY KEY (KundenID),
  CONSTRAINT UQ_Kunde_Email UNIQUE (Email),
  CONSTRAINT FK_Kunde_Stadt FOREIGN KEY (PLZ) REFERENCES Stadt(PLZ)
);
```

Ticket

Ticket(<u>TicketID</u>, Kaufdatum, Preis, GültigVon, GültigBis, *KundenID*)
{TicektID → Kaufdatum, Preis, GültigVon, GültigBis, KundenID}

Da der Primärschlüssel aus einem Attribut besteht, liegen keine partiellen Abhängigkeiten vor. KundenID ist ein Fremdschlüssel und begründet keine transitive Abhängigkeit zwischen Nicht-Schlüsselattributen. (3NF)

Ticket_Linie

Ticket_Linie(<u>*TicketID*</u>, <u>*LinienNr*</u>)
{FahrzeugID → Spurweite, Stromsystem}

Der Primärschlüssel ist zusammengesetzt (TicketID, LinienNr). Da die Tabelle jedoch keine weiteren Attribute enthält, sind sowohl die 2. Normalform und die 3. Normalform erfüllt.

2 Datenbank Triggers

2.1 Auto-Increment

Information von w3schools und stackoverflow entnommen.

Zuerst die Sequenz:

```
CREATE SEQUENCE SEQ_TicketID
  START WITH 1
  INCREMENT BY 1
  NOCACHE;
```

Wenn die ID NULL ist wird mit der Sequenz eine neue ID generiert:

```
CREATE OR REPLACE TRIGGER TRG_Ticket_AutoIncrement
BEFORE INSERT ON Ticket
FOR EACH ROW
BEGIN
  IF :NEW.TicketID IS NULL THEN
    :NEW.TicketID := SEQ_TicketID.NEXTVAL;
  END IF;
END;
```

2.2 Trigger mit Business Logic

Die Idee wurde von stackoverflow entnommen.

Wir wollen zu hohe Kosten verhindern, bzw. eine Warnung generieren:

```
CREATE TABLE KostenWarnung (
  WarnungID INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  Nachricht VARCHAR2(200),
  ErfasstAm DATE
);
```

Einerseits wird NULL auf 0 gesetzt und wenn der Wert 5000 überschritten wird, wird ein Eintrag in die neue Tabelle gemacht.

```
CREATE OR REPLACE TRIGGER TRG_Wartung_Check
BEFORE INSERT OR UPDATE ON Wartung
FOR EACH ROW
BEGIN
  IF :NEW.Kosten IS NULL THEN
    :NEW.Kosten := 0;
  END IF;
  IF :NEW.Kosten > 5000 THEN
    INSERT INTO KostenWarnung (Nachricht, ErfasstAm)
    VALUES ('ACHTUNG: Hohe Wartungskosten für Fahrzeug ' || :NEW.FahrzeugID || ': ' || :NEW.Kosten || '
  END IF;
END;
```

3 Datengenerierung (Data-Generator mit Jupyter Notebook)

Siehe **Data_Generator.ipynb** in der Abgabe.

4 ChatGPT

Verwendeted Prompt, aus der Angabe:

Generiere mir ein ER-Modell, ein relationales Schema und die zugehörigen SQL Statements auf Basis der folgenden Informationen: + verbatim die Abgabe aus Aufgabe 1-1.

Modell: GPT-5.2 (free)
Link: `https://chatgpt.com/share/696ba0f8-d094-8013-a1ba-5c41bd8ecab5`

Antwort:

```
Unten findest du ein konsistentes, prüfungs- und übungstaugliches Ergebnis bestehend aus

ER-Modell (textuell beschrieben)

Relationalem Schema

SQL-DDL-Statements (CREATE TABLE ...)

Ich halte mich eng an deine Vorgaben
(Wien Praterstern, kein Zahlungssystem, Fokus auf Betrieb/Vertrieb).

1. ER-Modell (konzeptionell)
Entitäten und Spezialisierungen
Fahrzeug

FahrzeugID (PK)

Baujahr

Kapazität
Generalisierung

Subtypen (disjunkt, total):

Bus

Kennzeichen
```

Niederflur (Boolean)

Zug

Baureihe

Wagonanzahl

Straßenbahn

Spurweite

Stromsystem

Wartung

WartungsNr (PK)

Datum

Kosten

Wartungsart
Beziehung: Fahrzeug 1 | n Wartung

Personal

PersonalNr (PK)

Name

Gehalt

Einstellungsdatum
Generalisierung

Subtypen:

Fahrer:in

Führerscheinklasse

Kontrolleur:in

Zuständigkeitsbereich

Rekursive Beziehung:

Fahrer:in betreut Fahrer:in
(1:n, Mentor → Trainee)

Linie

LinienNr (PK)

Taktung (Minuten)

Betriebsbeginn

Betriebsende

Haltestelle

HaltestellenID (PK)

Name

Barrierefrei (Boolean)

m:n Beziehung
Linie | bedient | Haltestelle
(Reihenfolge relevant → eigene Beziehungstabelle)

Einsatz

EinsatzID (PK)

Datum
verbindet:

Fahrer:in

Fahrzeug

Linie

(„Fahrer fährt mit Fahrzeug auf Linie\)

Kunde

KundenID (PK)

Name

E-Mail

Adresse

Ticket

TicketID (PK)

Kaufdatum

Preis

GültigVon

GültigBis

Beziehungen:

Kunde 1 | n Ticket

Ticket m | n Linie

Ticket n | 1 Kontrolleur (wird geprüft von)

2. Relationales Schema
FAHRZEUG(FahrzeugID, Baujahr, Kapazität)

```
BUS(FahrzeugID, Kennzeichen, Niederflur)
ZUG(FahrzeugID, Baureihe, Wagonanzahl)
STRASSENBAHN(FahrzeugID, Spurweite, Stromsystem)

WARTUNG(WartungsNr, FahrzeugID, Datum, Kosten, Wartungsart)

PERSONAL(PersonalNr, Name, Gehalt, Einstellungsdatum)
FAHRER(PersonalNr, Fuehrerscheinklasse)
KONTROLLEUR(PersonalNr, Zustaendigkeitsbereich)

MENTORING(MentorNr, TraineeNr)

LINIE(LinienNr, Taktung, Betriebsbeginn, Betriebsende)
HALTESTELLE(HaltestellenID, Name, Barrierefrei)
LINIE_HALTESTELLE(LinienNr, HaltestellenID, Reihenfolge)

EINSATZ(EinsatzID, Datum, PersonalNr, FahrzeugID, LinienNr)

KUNDE(KundenID, Name, Email, Adresse)
TICKET(TicketID, Kaufdatum, Preis, GueltigVon,
    GueltigBis, KundenID, KontrolleurNr)
TICKET_LINIE(TicketID, LinienNr)

3. SQL-Statements (DDL)
CREATE TABLE Fahrzeug (
    FahrzeugID INT PRIMARY KEY,
    Baujahr INT,
    Kapazitaet INT
);

CREATE TABLE Bus (
    FahrzeugID INT PRIMARY KEY,
    Kennzeichen VARCHAR(20),
    Niederflur BOOLEAN,
    FOREIGN KEY (FahrzeugID) REFERENCES Fahrzeug(FahrzeugID)
);

CREATE TABLE Zug (
    FahrzeugID INT PRIMARY KEY,
    Baureihe VARCHAR(50),
    Wagonanzahl INT,
    FOREIGN KEY (FahrzeugID) REFERENCES Fahrzeug(FahrzeugID)
);

CREATE TABLE Strassenbahn (
    FahrzeugID INT PRIMARY KEY,
    Spurweite VARCHAR(30),
    Stromsystem VARCHAR(50),
    FOREIGN KEY (FahrzeugID) REFERENCES Fahrzeug(FahrzeugID)
);

CREATE TABLE Wartung (
    WartungsNr INT PRIMARY KEY,
    FahrzeugID INT,
    Datum DATE,
    Kosten DECIMAL(10,2),
    Wartungsart VARCHAR(100),
    FOREIGN KEY (FahrzeugID) REFERENCES Fahrzeug(FahrzeugID)
);

CREATE TABLE Personal (
```

```
    PersonalNr INT PRIMARY KEY,
    Name VARCHAR(100),
    Gehalt DECIMAL(10,2),
    Einstellungsdatum DATE
);

CREATE TABLE Fahrer (
    PersonalNr INT PRIMARY KEY,
    Fuehrerscheinklasse VARCHAR(10),
    FOREIGN KEY (PersonalNr) REFERENCES Personal(PersonalNr)
);

CREATE TABLE Kontrolleur (
    PersonalNr INT PRIMARY KEY,
    Zustaendigkeitsbereich VARCHAR(100),
    FOREIGN KEY (PersonalNr) REFERENCES Personal(PersonalNr)
);

CREATE TABLE Mentoring (
    MentorNr INT,
    TraineeNr INT,
    PRIMARY KEY (MentorNr, TraineeNr),
    FOREIGN KEY (MentorNr) REFERENCES Fahrer(PersonalNr),
    FOREIGN KEY (TraineeNr) REFERENCES Fahrer(PersonalNr)
);

CREATE TABLE Linie (
    LinienNr INT PRIMARY KEY,
    Taktung INT,
    Betriebsbeginn TIME,
    Betriebsende TIME
);

CREATE TABLE Haltestelle (
    HaltestellenID INT PRIMARY KEY,
    Name VARCHAR(100),
    Barrierefrei BOOLEAN
);

CREATE TABLE Linie_Haltestelle (
    LinienNr INT,
    HaltestellenID INT,
    Reihenfolge INT,
    PRIMARY KEY (LinienNr, HaltestellenID),
    FOREIGN KEY (LinienNr) REFERENCES Linie(LinienNr),
    FOREIGN KEY (HaltestellenID)
        REFERENCES Haltestelle(HaltestellenID)
);

CREATE TABLE Einsatz (
    EinsatzID INT PRIMARY KEY,
    Datum DATE,
    PersonalNr INT,
    FahrzeugID INT,
    LinienNr INT,
    FOREIGN KEY (PersonalNr) REFERENCES Fahrer(PersonalNr),
    FOREIGN KEY (FahrzeugID) REFERENCES Fahrzeug(FahrzeugID),
    FOREIGN KEY (LinienNr) REFERENCES Linie(LinienNr)
);

CREATE TABLE Kunde (
```

```
    KundenID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Adresse VARCHAR(200)
);

CREATE TABLE Ticket (
    TicketID INT PRIMARY KEY,
    Kaufdatum DATE,
    Preis DECIMAL(10,2),
    GueltigVon DATE,
    GueltigBis DATE,
    KundenID INT,
    KontrolleurNr INT,
    FOREIGN KEY (KundenID) REFERENCES Kunde(KundenID),
    FOREIGN KEY (KontrolleurNr) REFERENCES Kontrolleur(PersonalNr)
);

CREATE TABLE Ticket_Linie (
    TicketID INT,
    LinienNr INT,
    PRIMARY KEY (TicketID, LinienNr),
    FOREIGN KEY (TicketID) REFERENCES Ticket(TicketID),
    FOREIGN KEY (LinienNr) REFERENCES Linie(LinienNr)
);


Wenn du möchtest, kann ich dir das ER-Modell auch als
Diagramm (Chen oder Crow's Foot)
beschreiben oder vereinfachen für eine Klausur.
```

Resultat Bewertung

Ein Problem ergibt sich da wir PDFs verwenden, wenn Umlaute kopiert werden die Umlautpunkte separat kopiert. Aus ö wir ¨o.
Wird die Antwort kopiert ergibt sich ein ähnliches Problem. ChatGPT verwendet viele Symbole (vor allem Emojis und em-dash) die in anderen System nicht ordentlich dargestellt werden können.

Da unsere Angabe sehr exakt ist und auch vieles mit Textmarkierungen herausgehoben ist, hatte die KI keine Probleme den Text richtig zu interpretieren. Es würde definitiv einen guten Startpunkt bieten.

Probleme:

- die tertiäre Beziehung aus Fahrzeug, Linie und Fahrer wurde nicht erkannt bzw. es ist fraglich, wie es interpretiert worden ist da es laut KI auch einen Primärschlüssel haben sollte
- mit nur einem Prompt bekommt man kein ordentliches ER-Diagram, ich weiß nicht wie es mit der bezahlten Version funktioniert, aber die Prompts sind auf eine gewisse Zeichenanzahl limitiert. Desto länger die Ünterhaltung"desto höher ist die Chance das die KI anfängt zu halluzinieren.
- *Wartung* wurde nicht als schwache Identität erkannt
- die Fehler aus dem ersten Schritt setzen sich in das relationale Schema fort
- beim relationalen Schema wird nicht angezeigt was ein primary / foreign key ist
- Vererbungen werde nicht ordentlich dargestellt
- die Fehler aus Schritt 1 und 2 setzen sich in die SQL-Statements fort
- es werden keine constraints verwendet um das DBS sicherer zu machen

- es wird *BOOLEAN* verwendet, es ist als Alias zu *TINYINT* in Ordnung; man wird aber nicht gewarnt das es zu Kompatibilitäts-Problemen führen kann.
- es gibt keine default Werte
- es gibt keine DELETE Regeln
- es gibt kein NOT NULL
- die Normalformen werden nicht beachtet
- *CONSTRAINT* wurde nicht verwendet um error-Meldungen lesbarer zu machen