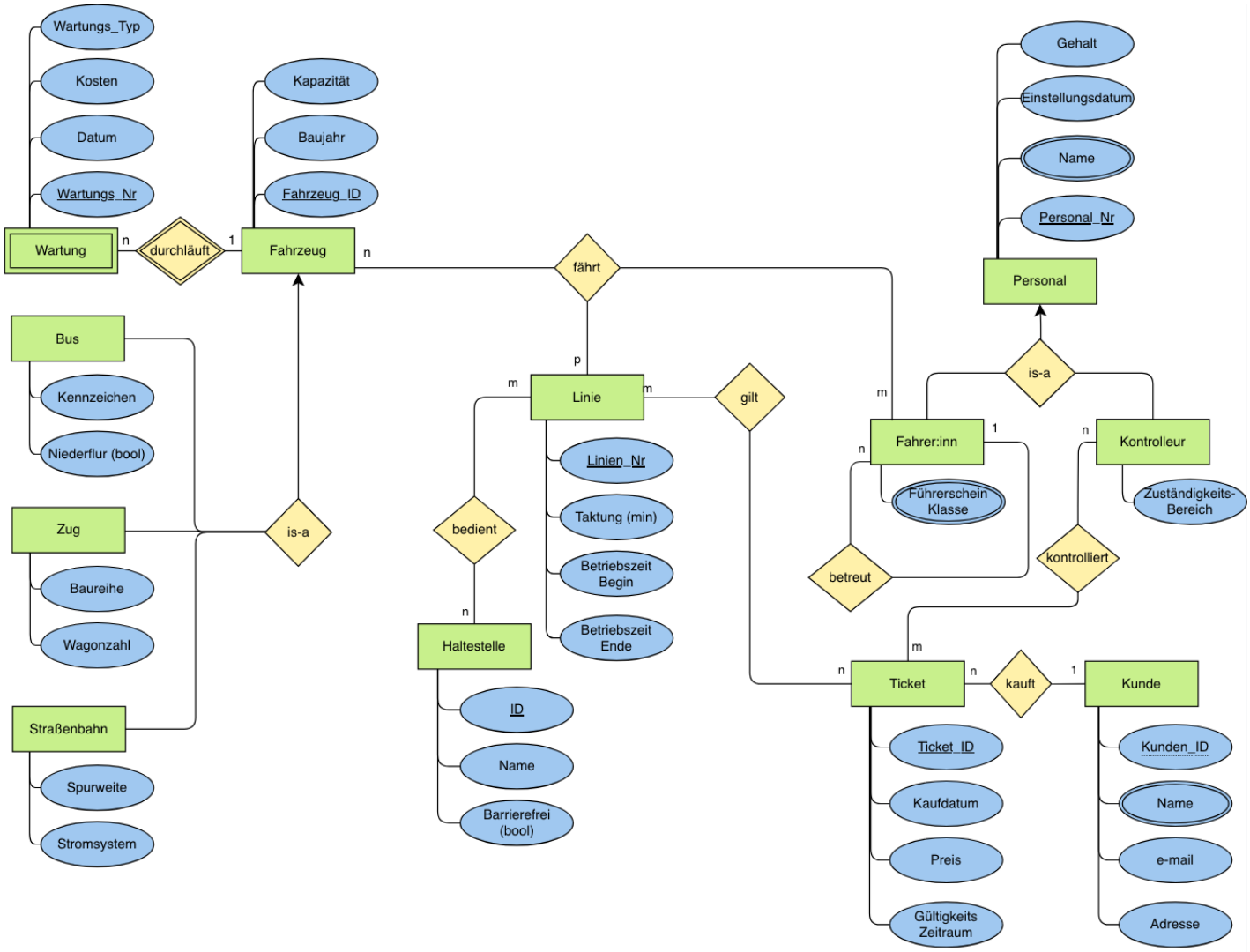
Meilenstein 2: Logischer und Physischer Entwurf

Gruppe 30
Boiko Bohdana, Alexander Grath

Logischer und Physischer Entwurf

universität wien

1 ER-Diagramm



Kopie des ER-Diagramms aus Meilenstein 1.

2 Logischer Entwurf

Fahrzeug(FahrzeugID, Baujahr, Kapazität)
Primarey Key: FahrzeugID

Bus(*FahrzeugID*, Kennzeichen, Niederflur)
Primary Key: FahrzeugID
Foreign Key: FahrzeugID ⋄ Fahrzeug.FahrzeugID

Zug(*FahrzeugID*, Baureihe, Wagonzahl)
Primary Key: FahrzeugID
Foreign Key: FahrzeugID ⋄ Fahrzeug.FahrzeugID

Straßenbahn(*FahrzeugID*, Spurweite, Stromsystem)
Primary Key: FahrzeugID
Foreign Key: FahrzeugID ⋄ Fahrzeug.FahrzeugID

Wartung(WartungsID, Datum, Kosten, Art, *FahrzeugID*)
Primary Key: WartungsID
Foreign Key: FahrzeugID ⋄ Fahrzeug.FahrzeugID

Personal(PersonalNr, Name, Gehalt, Einstellungsdatum)
Primary Key: PersonalNr

Fahrer(*PersonalNr*, Führerscheinklasse, *MentorNr*)
Primary Key: PersonalNr
Foreign Key: PersonalNr ⋄ Personal.PersonalNr
Foreign Key: MentorNr ⋄ Farhrer.PersonalNr

Kontrolleur(*PersonalNr*, Zuständigkeitsbereich)
Primary Key: PersonalNr
Foreign Key: PersonalNr ⋄ Personal.PersonalNr

Halstestelle(HalstestellenID, Name, Barrierefrei)
Primary Key: HalstestellenID

Linie(LinienNr, Taktung, Betriebsbeginn, Betriebsende)
Primary Key: LinienNr

Linie_Haltestelle(*LinienNr*, *HaltestellenID*, Reihenfolge)
Primary Key: (LinienNr, HaltestellenID)
Foreign Key: LinienNr ⋄ Linie.LinienNr
Foreign Key: HaltestellenID ⋄ Haltestelle.HaltestellenID

Kunde(KundenID, Name, Email, Adresse)
Primary Key: KundenID

Ticket(TicketID, Kaufdatum, Preis, GültigVon, GültigBis, *KundenID*)
Primary Key: TicketID
Foreign Key: KundenID ⋄ Kunde.KundenID

Ticket_Linie(*TicketID*, *LinienNr*)
Primary Key: (TicketID, LinienNr)
Foreign Key: TicketID ⋄ Ticket.TicketID
Foreign Key: LinienNr ⋄ Linie.LinienNr

3 Physischer Entwurf

Wie in den Übungen besprochen wurden alle keys und constraints benannt, im Falle eines Fehlers sollte damit ein lesbarer Errorcode erstellt werden.

Aus der Angabe zu verwendenden Bedingungen:

- **Primärschlüssel**
  alle Tables haben einen Primärschlüssel
- **Sekundärschlüssel**
  Beispiel aus 'Haltestelle':

```
CONSTRAINT UQ_Haltestelle_Name UNIQUE (Name)
```

- **Fremdschlüssel**
  Beispiel aus 'Wartung':

```
CONSTRAINT FK_Wartung_Fahrzeug FOREIGN KEY (FahrzeugID)
    REFERENCES Fahrzeug(FahrzeugID)
```

  'Wartung' wird mit 'Fahrzeug' verbunden.
- **Verbieten von Nullwerten**
  Beispiel aus 'Fahrzeug':

```
Baujahr SMALLINT NOT NULL
```

- **Domänenconstraints**
  Beispiel aus 'Fahrzeug':

```
CONSTRAINT CC_Fahrzeug_Kapazitaet CHECK (Kapazitaet > 0)
```

- **Standardwerte**
  Beispiel aus 'Bus':

```
Niederflur CHAR(1) DEFAULT 'N'
```

- **Deletion Rules**
  - *ON DELETE CASCADE*
    Beispiel aus 'Linie':

```
CONSTRAINT FK_Linie_Haltestelle_Linie FOREIGN KEY (LinienNr)
    REFERENCES Linie(LinienNr) ON DELETE CASCADE
```

  - *ON DELETE SET NULL*
    Wird in 'Fahrer' verwendet:

```
CONSTRAINT FK_Fahrer_Mentor FOREIGN KEY (MentorNr)
    REFERENCES Fahrer(PersonalNr)
    ON DELETE SET NULL
```

  - *NO ACTION*
    sollte default Verhalten sein wenn nichts anderes definiert ist, Beispiel bei 'Wartung':

```
CONSTRAINT FK_Wartung_Fahrzeug FOREIGN KEY (FahrzeugID)
    REFERENCES Fahrzeug(FahrzeugID)
    -- defaults to NO ACTION, can't delete Fahrzeug with Wartung
```

    In diesem Fall wollen wir verhindern das Fahrzeuge die gewartet werden, gelöscht werden können.

3.1 Fahrzeug

```
CREATE TABLE Fahrzeug (
  FahrzeugID INTEGER,
  Baujahr    SMALLINT NOT NULL,
  Kapazitaet INTEGER,
  -- PK: primary key
  CONSTRAINT PK_Fahrzeug PRIMARY KEY (FahrzeugID),
  -- CC: check constraint
  CONSTRAINT CC_Fahrzeug_Kapazitaet CHECK (Kapazitaet > 0)
);
```

3.2 Bus

```
CREATE TABLE Bus (
  FahrzeugID  INTEGER,
  Kennzeichen VARCHAR2(20) NOT NULL,
  Niederflur  CHAR(1) DEFAULT 'N',
  CONSTRAINT PK_Bus PRIMARY KEY (FahrzeugID),
  -- UQ: unique constraint
  CONSTRAINT UQ_Bus_Kennzeichen UNIQUE (Kennzeichen),
  -- FK: foreign key
  CONSTRAINT FK_Bus_Fahrzeug FOREIGN KEY (FahrzeugID)
    REFERENCES Fahrzeug(FahrzeugID)
    ON DELETE CASCADE,
  CONSTRAINT CC_Bus_Niederflur CHECK (Niederflur IN ('J','N'))
);
```

3.3 Zug

```
CREATE TABLE Zug (
  FahrzeugID  INTEGER,
  Baureihe    VARCHAR2(30) NOT NULL,
  Wagonanzahl INTEGER,
  CONSTRAINT PK_Zug PRIMARY KEY (FahrzeugID),
  CONSTRAINT FK_Zug_Fahrzeug FOREIGN KEY (FahrzeugID)
    REFERENCES Fahrzeug(FahrzeugID)
    ON DELETE CASCADE,
  CONSTRAINT CC_Zug_Wagon CHECK (Wagonanzahl > 0)
);
```

3.4 Straßenbahn

```
CREATE TABLE Strassenbahn (
  FahrzeugID  INTEGER,
  Spurweite   INTEGER NOT NULL,
  Stromsystem VARCHAR2(30) NOT NULL,
  CONSTRAINT PK_Strassenbahn PRIMARY KEY (FahrzeugID),
  CONSTRAINT FK_Strassenbahn_Fahrzeug FOREIGN KEY (FahrzeugID)
    REFERENCES Fahrzeug(FahrzeugID)
    ON DELETE CASCADE
);
```

3.5 Wartung

```
CREATE TABLE Wartung (
  WartungID  INTEGER,
  Datum      DATE NOT NULL,
  Kosten     NUMERIC(10,2),
  Art        VARCHAR2(50) NOT NULL,
  FahrzeugID INTEGER NOT NULL,
  CONSTRAINT PK_Wartung PRIMARY KEY (WartungID),
  -- defaults to NO ACTION, can't delete Fahrzeug with Wartung
  CONSTRAINT FK_Wartung_Fahrzeug FOREIGN KEY (FahrzeugID)
    REFERENCES Fahrzeug(FahrzeugID),
  CONSTRAINT CC_Wartung_Kosten CHECK (Kosten >= 0)
);
```

3.6 Personal

```
CREATE TABLE Personal (
  PersonalNr       INTEGER,
  Name             VARCHAR2(50) NOT NULL,
  Gehalt           NUMERIC(10,2),
  Einstellungsdatum DATE NOT NULL,
  CONSTRAINT PK_Personal PRIMARY KEY (PersonalNr),
  CONSTRAINT CC_Personal_Gehalt CHECK (Gehalt > 0)
);
```

3.7 Fahrer

```
CREATE TABLE Fahrer (
  PersonalNr         INTEGER,
  Fuehrerscheinklasse VARCHAR2(10) NOT NULL,
  MentorNr           INTEGER,
  CONSTRAINT PK_Fahrer PRIMARY KEY (PersonalNr),
  CONSTRAINT FK_Fahrer_Personal FOREIGN KEY (PersonalNr)
    REFERENCES Personal(PersonalNr)
    ON DELETE CASCADE,
  CONSTRAINT FK_Fahrer_Mentor FOREIGN KEY (MentorNr)
    REFERENCES Fahrer(PersonalNr)
    ON DELETE SET NULL
);
```

3.8 Kontrolleur

```
CREATE TABLE Kontrolleur (
  PersonalNr          INTEGER,
  Zustaendigkeitsbereich VARCHAR2(50) NOT NULL,
  CONSTRAINT PK_Kontrolleur PRIMARY KEY (PersonalNr),
  CONSTRAINT FK_Kontrolleur_Personal FOREIGN KEY (PersonalNr)
    REFERENCES Personal(PersonalNr)
    ON DELETE CASCADE
);
```

3.9 Haltestelle

```
CREATE TABLE Haltestelle (
  HaltestellenID INTEGER,
  Name           VARCHAR2(50) NOT NULL,
  Barrierefrei   CHAR(1) DEFAULT 'N',
  CONSTRAINT PK_Haltestelle PRIMARY KEY (HaltestellenID),
  -- UQ: unique constraint
  CONSTRAINT UQ_Haltestelle_Name UNIQUE (Name),
  CONSTRAINT CC_Haltestelle_Barrierefrei CHECK (Barrierefrei IN ('J','N'))
);
```

3.10 Linie

```
CREATE TABLE Linie (
  LinienNr      INTEGER,
  Taktung       INTEGER,
  Betriebsbeginn TIME NOT NULL,
  Betriebsende   TIME NOT NULL,
  CONSTRAINT PK_Linie PRIMARY KEY (LinienNr),
  CONSTRAINT CC_Linie_Taktung CHECK (Taktung > 0)
);
```

3.11 Linie_Haltestelle

```
CREATE TABLE Linie_Haltestelle (
  LinienNr       INTEGER,
  HaltestellenID INTEGER,
  Reihenfolge    INTEGER,
  CONSTRAINT PK_Linie_Haltestelle PRIMARY KEY (LinienNr, HaltestellenID),
  CONSTRAINT FK_Linie_Haltestelle_Linie FOREIGN KEY (LinienNr)
    REFERENCES Linie(LinienNr)
    ON DELETE CASCADE,
  CONSTRAINT FK_Linie_Haltestelle_Haltestelle FOREIGN KEY (HaltestellenID)
    REFERENCES Haltestelle(HaltestellenID)
    ON DELETE CASCADE,
  CONSTRAINT CC_LH_Reihenfolge CHECK (Reihenfolge > 0)
);
```

3.12 Kunde

```
CREATE TABLE Kunde (
  KundenID INTEGER,
  Name     VARCHAR2(50) NOT NULL,
  Email    VARCHAR2(100) NOT NULL,
  Adresse  VARCHAR2(100),
  CONSTRAINT PK_Kunde PRIMARY KEY (KundenID),
  CONSTRAINT UQ_Kunde_Email UNIQUE (Email)
);
```

3.13 Ticket

```
CREATE TABLE Ticket (
  TicketID   INTEGER,
  Kaufdatum  DATE NOT NULL,
  Preis      NUMERIC(8,2),
  GueltigVon DATE NOT NULL,
  GueltigBis DATE NOT NULL,
  KundenID   INTEGER NOT NULL,
  CONSTRAINT PK_Ticket PRIMARY KEY (TicketID),
  CONSTRAINT FK_Ticket_Kunde FOREIGN KEY (KundenID)
    REFERENCES Kunde(KundenID)
  CONSTRAINT CC_Ticket_Preis CHECK (Preis > 0),
  CONSTRAINT CC_Ticket_Datum CHECK (GueltigBis >= GueltigVon)
);
```

3.14 Ticket_Linie

```
CREATE TABLE Ticket_Linie (
  TicketID INTEGER,
  LinienNr INTEGER,
  CONSTRAINT PK_Ticket_Linie PRIMARY KEY (TicketID, LinienNr),
  CONSTRAINT FK_Ticket_Line_Ticket FOREIGN KEY (TicketID)
    REFERENCES Ticket(TicketID)
    ON DELETE CASCADE,
  CONSTRAINT FK_Ticket_Line_Linie FOREIGN KEY (LinienNr)
    REFERENCES Linie(LinienNr)
    ON DELETE CASCADE
);
```

4 Delete Script

Im Delete-Script muss beachtet werden dass die Tables in der richtigen Reihenfolge gelöscht werden. Zuerst die 'Child' Tables und alle Tables die Fremdschlüssel beinhalten und dann die 'Parent' Tables. Beispiel bei Tickets: $Ticket\_Linie \rightarrow Ticket \rightarrow Kunde$.

```
DROP TABLE Ticket_Linie;
DROP TABLE Ticket;
DROP TABLE Kunde;
DROP TABLE Linie_Haltestelle;
DROP TABLE Haltestelle;
DROP TABLE Linie;
DROP TABLE Wartung;
DROP TABLE Bus;
DROP TABLE Zug;
DROP TABLE Strassenbahn;
DROP TABLE Fahrzeug;
DROP TABLE Fahrer;
DROP TABLE Kontrolleur;
DROP TABLE Personal;
```

5 Physischer Entwurf - Erweitert

Alle Tables werden mit zumindest einem Element befüllt.

```
INSERT INTO Fahrzeug VALUES (1, 2018, 80);
INSERT INTO Fahrzeug VALUES (2, 2020, 200);
INSERT INTO Fahrzeug VALUES (3, 2015, 150);

INSERT INTO Bus VALUES (1, 'W-1234AB', 'J');
INSERT INTO Zug VALUES (2, 'Railjet', 6);
INSERT INTO Strassenbahn VALUES (3, 1435, 'DC 600V');

INSERT INTO Wartung VALUES (1, DATE '2025-01-10', 500.00, 'Inspektion', 1);

INSERT INTO Personal VALUES (1, 'Max Fahrer', 2500, DATE '2020-03-23');
INSERT INTO Personal VALUES (2, 'Anna Fahrerin', 2300, DATE '2022-07-19');
INSERT INTO Personal VALUES (3, 'Karl Kontrolle', 2200, DATE '2021-01-15');

INSERT INTO Fahrer VALUES (1, 'B', NULL);
INSERT INTO Fahrer VALUES (2, 'B', 1);

INSERT INTO Kontrolleur VALUES (3, 'Linie 15');

INSERT INTO Haltestelle VALUES (1, 'Praterstern', 'J');
INSERT INTO Haltestelle VALUES (2, 'Karlsplatz', 'N');

INSERT INTO Linie VALUES (1, 5, TIME '05:00:00', TIME '23:30:00');

INSERT INTO Linie_Haltestelle VALUES (1, 1, 1);
INSERT INTO Linie_Haltestelle VALUES (1, 2, 2);

INSERT INTO Kunde VALUES (1, 'Lisa Kunde', 'lisa@beispiel.com', 'Wien');

INSERT INTO Ticket
    VALUES (1, DATE '2025-02-01', 45.00, DATE '2025-02-01', DATE '2025-02-28', 1);

INSERT INTO Ticket_Linie VALUES (1, 1);
```

5.1 Beispiele zu Primär- und Fremdschlüssel:

Korrekte Einträge können natürlich gemacht werden (und Primär- und Fremdschlüssel verwenden.
Hier zwei Beispiele wie falsche Einträge verhindert werden können:

```
-- Primary Key Test:
INSERT INTO Personal VALUES (1, 'Zwilling Fahrer', 2000, DATE '2023-01-01');
-- PersonalNr already exists
-- Foreign Key Test:
INSERT INTO Bus VALUES (99, 'W-1235AB', 'N');
-- there is no FahrzeugID 99
```

Einmal wird versucht einen bestehenden Primärschlüssel zu verwenden → ERROR.
Einmal wird versucht einen Eintrag anzulegen der einen Fremdschlüssel hat den es nicht gibt → ERROR.

5.2 Beispiele zu den Deletion Rules:

Wenn ein Fahrzeug gewartet wird, kann es nicht gelöscht werden (NO ACTION). Wenn Teil des Personals gelöscht wird, wird der korrekte Arbeiter gelöscht (CASCADE) und in diesem Fall war der Fahrer auch ein Mentor und somit wird beim anderen Fahrer der Mentor auf NULL gesetzt (SET NULL).

```
-- Deletion Rule Tests
-- 1) NO ACTION
DELETE FROM Fahrzeug WHERE FahrzeugID = 1;
-- can't be deleted is is part of Wartung 1
-- 2) CASCADE and SET NULL
DELETE FROM Personal WHERE PersonalNr = 1;
-- Fahrer 1 will be deleted (CASCADE)
-- Fahrer 2's Mentor will be deleted (SET NULL)
```

5.3 View und Abfrage:

Hier eine Ticket Übersicht zu den Linien:

```
CREATE VIEW Ticket_Uebersicht AS
SELECT
  k.Name AS Kunde,
  t.TicketID,
  t.Preis,
  l.LinienNr
FROM Kunde k
JOIN Ticket t ON k.KundenID = t.KundenID
JOIN Ticket_Linie tl ON t.TicketID = tl.TicketID
JOIN Linie l ON tl.LinienNr = l.LinienNr;

SELECT * FROM Ticket_Uebersicht;
```