Datenbanksysteme
SS 2025

Übungsblatt 2

Group:
Attila Haraszti (5602072),
Moritz Ruge (5600961),
Joshua Zahn (5600969).
John ... (johnjesul01@)

Datum: May 16, 2025

Task 1: Relational Model I

1. Was ist ein Schlüsselattribut?

A key attribute can be used to identify each entity uniquely. It imposes a uniqueness constraint. In ER diagrammatic notation, each key attribute has its name underlined inside the oval.

2. Nennen Sie ein Beispiel für eine Relation vom Grad 4 (degree/arity)?

The degree (or arity) of a relation is the number of attributes n of its relation schema. An relation with an arity of 4 would be e.g. `STUDENT(FName, LName, CourseID, Email)`

3. Geben Sie ein Beispiel für eine nicht-rekursive 1-zu-N Relation an?

1 music producer produces N pieces of music:

```
Producer(ID, Name)
Music(ID, Title, ProducerID)
```

4. Geben Sie ein Beispiel für eine rekursive 1-zu-N Relation an?

Actors in a movie can be directed by other actors:

```
Actor(ID, Name, DirectedByID)
```

Task 2: Relational Model II

1. Erstellen Sie ein relationales Modell, das zu dem in Abbildung 1 gegebenen Entity Relationship Modell korrespondiert.

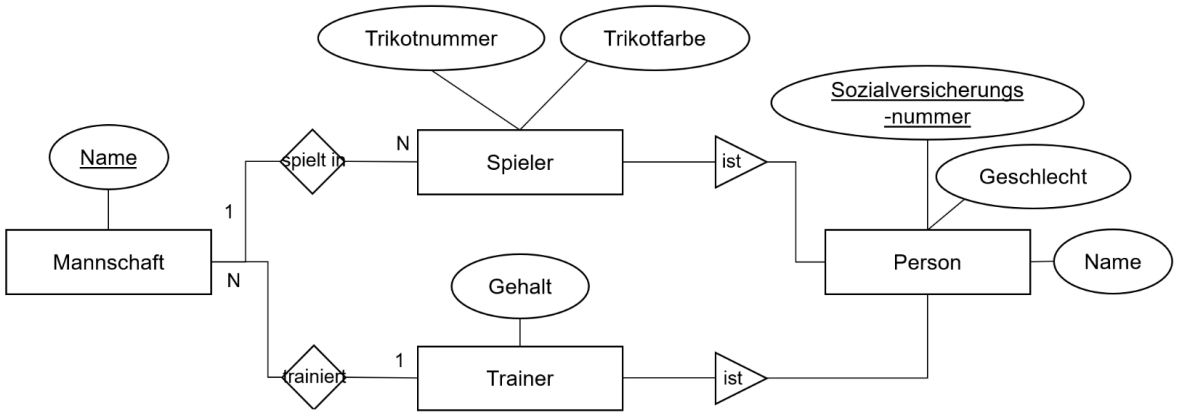


Figure 1: Entity Relationship model of a Soccer Team.

- SPIELER(Trikotnummer, Trikotfarbe)
- MANNSCHAFT(Name)
- TRAINER(Gehalt)
- PERSON(Sozialversicherungs-nummer, Geschlecht, Name)
- SPIELT-IN(Name, Trikotnummer)
- TRAINIERT(Sozialversicherungs-nummer, Name)

2. Erstellen Sie ein relationales Modell, das zu dem in Abbildung 2 gegebenen Entity Relationship Modell korrespondiert.?

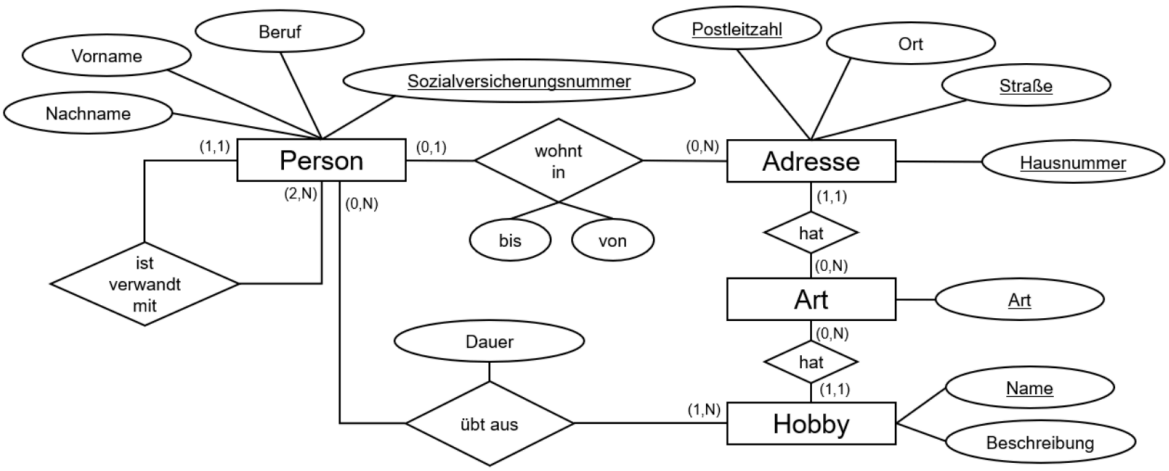


Figure 2: Entity Relationship model of a Person with Hobbies.

- PERSON(Vorname, Nachname, Beruf, Sozialversicherungsnummer)
- ADRESSE(Postleitzahl, Ort, Straße, Hausnummer)
- ART(Art)
- HOBBY(Name, Beschreibung)
- IstVerwandtMit(Sozialversicherungsnummer)
- WohntIn(bis, von, Sozialversicherungsnummer, Postleitzahl, Straße, Hausnummer)
- ÜbtAus(Dauer, Name, Sozialversicherungsnummer)
- HatArt(Sozialversicherungsnummer, Art)
- HatHobby(Sozialversicherungsnummer, Hobby)

Task 3: Relational Algebra I

EN: Relational Schema for Tasks 3, 4, 5, 6, 7

$\texttt{AIRLINE}(\underline{ID},\ Name)$
$\texttt{AIRPLANE}(\underline{ID},\ Name,\ Type,\ PassengerCapacity,\ Range)$
$\texttt{PASSENGER}(\underline{ID},\ FirstName,\ LastName,\ Age,\ Nationality,\ CreditCardNumber)$
$\texttt{CHARTER}(\underline{AirlineID, AirplaneID, FromDate, ToDate})$
$\texttt{FLIGHT}(\underline{AirlineID, PassengerID, Date})$
$\texttt{WEATHER}(\underline{Date},\ Temperature,\ PrecipitationAmount,\ SunshineDuration)$

DE: Relationales Schema für die Aufgaben 3, 4, 5, 6, 7

$\texttt{FLUGGESELLSCHAFT}(\underline{ID},\ Name)$
$\texttt{FLUGZEUG}(\underline{ID},\ Name,\ Typ,\ Passagierkapazitt,\ Reichweite)$
$\texttt{PASSAGIER}(\underline{ID},\ Vorname,\ Nachname,\ Alter,\ Nationalitt,\ Kreditkartennummer)$
$\texttt{CHARTER}(\underline{FluggesellschaftsID, FlugzeugID, VonDatum, BisDatum})$
$\texttt{FLUG}(\underline{FluggesellschaftsID, PassagierID, Datum})$
$\texttt{WETTER}(\underline{Datum},\ Temperatur,\ Niederschlagsmenge,\ Sonnenscheindauer)$

Betrachten Sie das auf Seite 3 angegebene relationale Modell. Drücken Sie die folgenden Abfragen in der relationalen Algebra aus.

1. Geben Sie die Vornamen aller Passagiere an, die mit Nachnamen ”Johansen” heißen

   ΠFirstName(σLastName=”Johansen”(PASSENGER))

2. Geben Sie die Kreditkartennummern aller Passagiere an, die am ”01.01.2020” einen Flug mit der ”Lufthansa” gebucht haben.

   PASS-FLI-AIRL <- PASSENGER 1.⋈ FLIGHT 2.⋈ AIRLINE
   1.PASSENGER.Passanger_ID=FLIGHT..Passanger_ID
   2.FLIGHT.Airline_ID=AIRLINE.Airline_ID
   ΠCreditCardNumber(σDate=”01.01.2020”∧AIRLINE.Name=”Lufthansa” (PASS-FLI-AIRL))

3. Geben Sie die Vornamen und Nachnamen aller Passagiere an, die im April 2022 einen Flug gebucht haben.
   PASS-FLI <- PASSENGER 1.⋈ FLIGHT
   1.PASSENGER.Passanger_ID=FLIGHT..Passanger_ID
   ΠFirstName,LastName(σDate=”04.2022”(PASS-FLI))

4. Geben Sie die Namen aller Fluggesellschaften an, die Passagiere an Tagen mit einer Temperatur mehr als 30 Grad Celsius und einer Regenmenge weniger als 20 Liter pro Stunde und einer Sonnenscheindauer von mindestens 8 Stunden befördert haben.

   AIRL-FLI-WEA <- AIRLINE 1.⋈ FLIGHT 2.⋈ WEATHER
   1.AIRLINE.Airline_ID=FLIGHT.Airline_ID
   2.FLIGHT.Date=WEATHER.Date
   ΠName(σTemperature > 30 ∧ PercipitationAmount < 20 ∧ SunshineDuration >= 8 (AIRL-FLI-WEA))

Task 4: Relational Algebra II

Betrachten Sie das auf Seite 3 angegebene relationale Modell. Drücken Sie die folgenden Abfragen in der relationalen Algebra aus.

1. Geben Sie alle Passagier-IDs an, die nie geflogen sind (Passagier≈Kunde)

   ΠPassenger_ID(PASSENGER - FLIGHT)

2. Geben Sie das Alter aller Passagiere an, die noch nie mit ”Lufthansa” an Tagen unter 10 Grad Celsius geflogen sind.

   PASS-FLI-AIR<-PASSENGER 1.⋈ FLIGHT 2.⋈ AIRLINE
   1.PASSENGER.Passanger_ID=FLIGHT..Passanger_ID
   2.FLIGHT.Airline_ID=AIRLINE.Airline_ID
   PASS-FLI-AIR-WEA<- PASS-FLI-AIR 3.⋈ WEATHER
   3.FLIGHT.Date=WEATHER.Date
   ΠAge(PASSAGIER - ΠPassagier_ID(σAIRLINE.Name = “Lufthansa” ∧ Temperature < 10 (PASS-FLI-AIR-WEA)))

Task 5: Relational Algebra III

Betrachten Sie das auf Seite 3 angegebene relationale Modell. Welche der unten angegebenen Abfragen lassen sich in relationaler Algebra ausdrücken? Geben Sie jeweils entweder "Kann nicht in relationaler Algebra ausgedrückt werden." oder den entsprechenden Ausdruck in relationaler Algebra an.

1. Geben Sie die Nationalität der Passagiere an, die am meisten geflogen sind.
    - Kann nicht in Relationale Algebra ausgedrückt werden
        - Es gibt keinen weg die Anzahl zu zählen
2. Geben Sie den Namen aller Flugzeuge an, die nie im Jahr 2010 geflogen sind.
    - Menge aller Flugzeuge
        - Π_Name(FLUGZEUG)
    - Menge der Flugzeuge, die im Jahr 2010 geflogen sind
        - Π_Name(FLUGZEUG ⋈ FLUGZEUG.ID = CHARTER.FlugzeugID
          σ_vonDatum <= '2010.12.31' ∧ BisDatum >= '2010.01.01'(CHARTER))
    - Menge aller Flugzeuge von der Menge im Jahr 2010 abziehen
        - Π_Name(FLUGZEUG) - Π_Name(FLUGZEUG ⋈ FLUGZEUG.ID = CHARTER.FlugzeugID
          σ_vonDatum <= '2010.12.31' ∧ BisDatum >= '2010.01.01'(CHARTER))
3. Geben Sie die Anzahl der Fluggesellschaften an, die nie Passagiere älter als 50 Jahre hatten.
    - Die genaue Anzahl kann man nicht berechnen, da es keine COUNT Funktion gibt, man kann aber die Menge wiedergeben
    - Menge von Passagieren über 50 Jahren
        - oldPeople ← σ_Alter > 50(PASSAGIER)
    - Menge von PassagierID von Passagieren über 50
        - ID50 ← FLUG ⋈ FLUG.Passagier.ID = oldPeople.ID (oldPeople)
    - Menge von Fluggesellschaften die über 50 Jährige befördert haben
        - AirlineOldPeople ← σ_FluggesellschaftsID(ID50)
    - Alle Fluggesellschaften
        - ALL_FS ← σ_ID(FLUGGESELLSCHAFTEN)
    - Fluggesellschaften, die nie Passagiere über 50 hatten
        - ERGEBNIS ← ALL_FS - AirlineOldPeople

Task 6: Structured Query Language – Basics

Betrachten Sie das auf Seite 3 angegebene relationale Modell. Geben Sie die folgenden Abfragen in SQL an.

1. Geben Sie die Namen aller Fluggesellschaften aus.

   SELECT Name FROM AIRLINE

2. Geben Sie die Vornamen und Nachnamen aller Passagiere aus, die jünger als 30 Jahre alt sind.

   SELECT FirstName, LastName FROM PASSENGER WHERE Age < 30

3. Geben Sie den Vornamen und Nachnamen aller Passagiere aus, die an Tagen mit mehr als 20 °C geflogen sind.

   SELECT FirstName,LastName FROM PASSENGER,FLIGHT,WEATHER
   WHERE PASSANGER.PASSANGER_ID = FLIGHT.PASSANGER_ID AND
   FLIGHT.Date = WEATHER.Date AND
   Temperature > 20

4. Geben Sie die Anzahl aller Passagiere aus, die 2020 mit “British Airways” geflogen sind.

   SELECT COUNT(*) FROM PASSENGER, AIRLINE, FLIGHT
   WHERE AIRLINE.Name = “British Airways” AND
   FLIGHT.Date = “2020” AND
   PASSANGER.PASSANGER_ID = FLIGHT.PASSANGER_ID AND
   FLIGHT.Airline_ID = AIRLINE.Airline_ID

5. Geben Sie die Namen aller Fluggesellschaften aus, die “Airline” im Namen tragen.

   SELECT Name FROM AIRLINE WHERE Name LIKE “*Airline*”

6. Geben Sie die Namen aller Fluggesellschaften an, die 2020 mindestens einen Airbus A380 gechartet haben.

```
SELECT AIRLINE.Name FROM AIRLINE, AIRPLANE, CHARTER, FLIGHT
WHERE FLIGHT.Airline_ID = AIRLINE.Airline_ID
        CHARTER.Airline_ID = AIRLINE.Airline_ID
        CHARTER.AIRPLANE_ID = AIRPLANE.Airplane_ID
        Date = “2020”
        AIRPLANE.Name = “Airbus A380”
```

7. Geben Sie die Namen der Top 5 Fluggesellschaften mit den höchsten Passagierzahlen 2020 an.

```
SELECT AIRLINE.Name COUNT(FLIGHT.Passanger_ID) AS Passagiere FROM FLIGHT,
AIRLINE
WHERE Date = “2020”
FLIGHT.Airline_ID = AIRLINE.Airline_ID
GROUP BY AIRLINE.Name
ORDER BY Passagiere DESC
LIMIT 5
```

Task 7: Structured Query Language – Joins & Nested Queries

```
COMPANY scheme:
EMPLOYEE(FName, LName, SSN, BDate, Address, Salary, BossSSN, NumDept)
DEPENDENT(Name, Gender, BDate, Relation, ESSN)
DEPARTMENT(DName, DNumber, MGRSSN, MGRStartDate) DEPTLOCATION(DNumber, Location)
PROJECT(PName, PNumber, Location, DNumber)
WORKSON(ESSN, PNO, Hours)
```

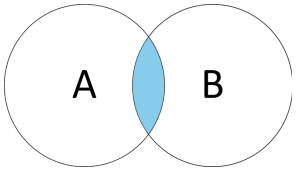
Betrachten Sie das auf Seite 3 angegebene relationale Modell. Geben Sie die SQL-Anweisungen und ggf. eine kurze Beschreibung des Join-Typs an.

1. Vor- und Nachnamen aller Mitarbeiter über einen Inner Join.

```
SELECT EMPLOYEE.Fname, EMPLOYEE.LName FROM DEPARTMENT INNER JOIN
EMPLOYEE ON DEPARTMENT.DNumber = EMPLOYEE.NumDept;
```

   a. Merkmale und Zweck eines Inner Join

An inner join, also known as *theta-join*, or θ-join ($\bowtie_\theta$), is a type of match-and- combine operation defined formally as a combination of CARTESIAN PRODUCT and SELECTION (Elmasri & Navathe, 2016, 254). In an INNER JOIN, only pairs of tuples that match the explicit join condition are retrieved.

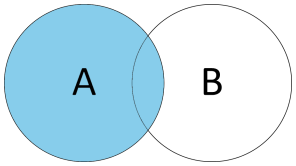


2. Namen aller Abteilungen mit SSN der Manager per Left Join.

```
SELECT DEPARTMENT.DName, EMPLOYEE.SSN AS ManagerSSN FROM DEPARTMENT LEFT
JOIN EMPLOYEE ON DEPARTMENT.MGRSSN = EMPLOYEE.SSN;
```

   a. Merkmale und Zweck eines Left Join.

Also known as LEFT OUTER JOIN, denoted by ⟕. The operation keeps every tuple in the first, or left, relation R in R if no matching tuple is found in S, then the attributes of S in the join result are filled or padded with NULL values. (Elmasri & Navathe, 2016, 264)

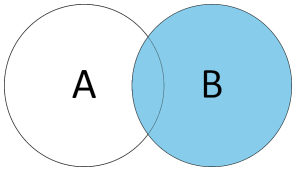


3. Namen und Geschlecht aller Angehörigen per Right Join.

```
SELECT DEPENDENT.Name, DEPENDENT.Gender FROM EMPLOYEE RIGHT JOIN
DEPENDENT ON EMPLOYEE.SSN = DEPENDENT.ESSN;
```

    a. Merkmale und Zweck eines Right Join.

Same as LEFT OUTER JOIN, but the operation keeps every tuple in the second, or right relation if no matching tuples are found. Denoted by ⟖.

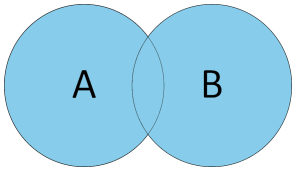


4. Vor- und Nachnamen der Mitarbeiter mit Abteilungsnamen per Full Join.

```
SELECT EMPLOYEE.FName, EMPLOYEE.LName, DEPARTMENT.DName FROM EMPLOYEE
FULL JOIN DEPARTMENT ON EMPLOYEE.NumDept = DEPARTMENT.DNumber;
```

    a. Merkmale und Zweck eines Full Join.

A FULL OUTER JOIN, denoted by ⟗, keeps all tuples in both the left and the right relations when no matching tuples are found, padding them with NULL values as needed. (Elmasri & Navathe, 2016, 264)



5. Vor- und Nachnamen aller Mitarbeiter, die mindestens in einem Projekt mit John Smith arbeiten – verschachtelte Abfragen.

```
SELECT EMPLOYEE.FName, EMPLOYEE.LName FROM EMPLOYEE
      WHERE EMPLOYEE.SSN IN
            (SELECT WORKSON.ESSN FROM WORKSON WHERE WORKSON.PNO =
                  (SELECT WORKSON.PNO FROM WORKSON WHERE WORKSON.ESSN =
                        (SELECT SSN FROM EMPLOYEE
                              WHERE FName = 'John' AND LName = 'Smith'))
                        AND WORKSON.Hours > 0)
      AND EMPLOYEE.SSN != (SELECT SSN FROM EMPLOYEE WHERE
                        FName = 'John' AND LName = 'Smith');
```

6. Wie oben, aber mit Join-Operation.

```
SELECT e1.FName, e1.LName FROM EMPLOYEE e1
      JOIN WORKSON w1 ON e1.SSN = w1.ESSN
      JOIN WORKSON w2 ON w1.PNO = w2.PNO
      JOIN EMPLOYEE e2 ON w2.ESSN = e2.SSN
                              AND e2.Fname = 'John'
                              AND e2.LName = 'Smith'
      WHERE e1.SSN <> e2.SSN AND w1.Hours > 0;
```

Task 8: Relational Algebra & SQL on COMPANY

Betrachten Sie das auf Seite 3 angegebene relationale Modell. Drücken Sie jede der folgenden Abfragen in der relationalen Algebra **und** SQL aus.

1. Standort der Abteilung und Geburtsdatum aller Mitarbeiter > 60 J.

```
SELECT dl.Location, e.BDate FROM DEPTLOCATION dl
      JOIN DEPARTMENT d ON dl.DNumber = d.DNumber
      JOIN EMPLOYEE e ON d.DNumber = e.NumDept
        WHERE e.BDate <= date('now','-60 year'); -- SQLite syntax :)
      -- WHERE e.BDate <= CURRENT_DATE - INTERVAL '60 years'; -- PostgreSQL
```

$dl = \text{DEPTLOCATION}$
$d = \text{DEPARTMENT}$
$e = \text{EMPLOYEE}$

$$TEMP = dl \bowtie_{dl.DNumber=d.DNumber} d \bowtie_{d.DNumber=e.NumDept} e$$

$$\Pi_{\text{Location, BDate}}\left(\sigma_{e.BDate \leq \text{date('now', '}-60\text{ year')}}(TEMP)\right)$$

2. Gehälter aller Mitarbeiter, die an allen Projekten der Abteilung „Forschung“ arbeiten

```
SELECT e.Salary FROM EMPLOYEE e
      JOIN WORKSON w ON e.SSN = w.ESSN
      JOIN DEPARTMENT d ON e.NumDept = d.DNumber
      JOIN PROJECT p ON p.PNumber = w.PNO AND p.DNumber = d.DNumber
            WHERE d.DName = 'Forschung';
```

$w = WORKSON$
$e = EMPLOYEE$
$d = DEPARTMENT$
$p = PROJECT$

$$TEMP = e \bowtie_{e.SSN=w.ESSN} w \bowtie_{e.NumDept=d.DNumber} d \bowtie_{p.PNumber=w.PNO \wedge p.DNumber=d.DNumber} p$$

$$\Pi_{Salary}(\sigma_{DName='Forschung'}(TEMP))$$

Quellenangabe

Elmasri, R., & Navathe, S. (2016). *Fundamentals of Database Systems*. Pearson.

Wikimedia Commons. (n.d.). *SQL Joins*. Wikimedia Commons.

https://commons.wikimedia.org/wiki/Category:SQL_Joins