Übungsblatt 1

# Modellierung & Mapping

## Chen & Min-Max Diagramme

Die Diagramme befinden sich auf den nächsten beiden Seiten (zuerst Chen, dann Min-Max).

**Bitte beachten**: User hat zur Relation „is friend of“ nur einen Strich, obwohl hier eigentlich zwei sein müssten. Ich habe das aus Platz- und Formatierungsgründen so gemacht, damit mir die beiden Linien nicht zusammenfallen. Die Kardinalität ist n:m (Chen) bzw. (0,N)(0,N) (Min-Max) und befindet sich über und unter der Linie um sie auszudrücken.





## Verfeinerung

Ich habe das Modell in Aufgabe 1a nach bestem Wissen und Gewissen erstellt und weiß nicht genau, was ich hier genau tun soll. Meiner Meinung nach habe ich dabei die Regeln für gute E/R-Modellierung eingehalten und beachtet.

## Relationales Schema

### Relationen direkt aus den Entitäten, 1:n-Beziehungen bereits aufgelöst

Location: {[

location\_id: int,

street: string,

city: string,

country: string

]}

Schlüsselkandidaten: {location\_id}, {street, city, country}

Primärschlüssel: {location\_id}

Sekundärschlüssel: {country, city}

User: {[

user\_id: int,

email: string,

profile\_image: binary,

name: string,

hobbies: string

]}

Schlüsselkandidaten: {user\_id}, {email}, {name}

Primärschlüssel: {user\_id}

Sekundärschlüssel: {name}, {email}

Das Feld „hobbies“ ist mehrwertig und müsste durch eine eigene Tabelle ausgedrückt werden.

Group: {[group\_id: int, name: string, user\_id, location\_id]}

Schlüsselkandidaten: {group\_id}, {name, user\_id}

Primärschlüssel: {group\_id}

Sekundärschlüssel: {name}

GroupRole: {[role\_id: int, name: string]}

Schlüsselkandidaten: {role\_id}, {name}

Primärschlüssel: {role\_id}

Sekundärschlüssel: {name}

EducationalInstitution: {[institution\_id: int, name: string, location\_id]}

Primärschlüssel: {institution\_id}, {name,location\_id}

Sekundärschlüssel: {name}

Education: {[education\_id: int, name: string]}

Schlüsselkandidaten: {education\_id}, {name}

Primärschlüssel: {education\_id}

Sekundärschlüssel: {name}

Gallery: {[gallery\_id: int, name: string, user\_id]}

Primärschlüssel: {gallery\_id}, {user\_id, name}

Sekundärschlüssel: {user\_id, name}

Image: {[image\_id: int, title: string, description: string, gallery\_id]}

Schlüsselkandidaten: {image\_id,gallery\_id}

Primärschlüssel: {image\_id, gallery\_id}

Sekundärschlüssel: {title}

### Relationen aus den Beziehungen

UserGroupRole: {[user\_id, group\_id, role\_id]}

Primärschlüssel: {user\_id, group\_id, role\_id}

Sekundärschlüssel: {}

UserLocations: {[user\_id, location\_id, type: string]}

Schlüsselkandidaten: {user\_id, location\_id}

Primärschlüssel: {user\_id, location\_id}

Sekundärschlüssel: {type}

Der Typ der UserLocation müsste auch über eine Tabelle aufgelöst werden, da sonst in der Datenbank die Typen nicht validiert werden können. Ansonsten muss das in der Applikation geschehen, was aber unsauber ist.

UserEducations: {[user\_id, institution\_id, education\_id, graduation\_year: date]}

Primärschlüssel: {user\_id, institution\_id, education\_id}

Sekundärschlüssel: {}

UserTaggedImages: {[user\_id, gallery\_id, image\_id, pos\_x: int, pos\_y: int]}

Primärschlüssel: {user\_id, gallery\_id, image\_id}

Sekundärschlüssel: {}

# SQL

**SELECT**

`fh1`**.**`name` **AS** `Von`**,**

`fh2`**.**`name` **AS** `Nach`**,**

**DATE\_FORMAT(**`f`**.**`abflug`**,** '%k:%i'**)** **AS** `Uhrzeit`

**FROM**

`flug` **AS** `f`

**INNER** **JOIN** `flughafen` **AS** `fh1`

**ON** `f`**.**`von` **=** `fh1`**.**`flughafen\_id`

**INNER** **JOIN** `flughafen` **AS** `fh2`

**ON** `f`**.**`nach` **=** `fh2`**.**`flughafen\_id`

**WHERE**

`fh1`**.**`iata` **=** 'TEE'

**AND**

`f`**.**`abflug` **BETWEEN** **CAST(**'2011-08-01' **AS** DATETIME**)** **AND** **CAST(**'2011-08-30' **AS** DATETIME**)**

**;**

Hier gibt es keine Ergebnistupel weil vom Flughafen „TEE“ keine Flüge ausgehen.

## 

**SELECT**

`p`**.**`vorname`**,**

`p`**.**`nachname`**,**

`pd`**.**`geburtsdatum`

**FROM**

`passagier` **AS** `p`

**INNER** **JOIN** `passagierdetails` **AS** `pd`

**ON** `p`**.**`passagier\_id` **=** `pd`**.**`passagier\_id`

**INNER** **JOIN** `buchung` **AS** `b`

**ON** `b`**.**`passagier\_id` **=** `p`**.**`passagier\_id`

**INNER** **JOIN** `flug` **AS** `f`

**ON** `f`**.**`flug\_id` **=** `b`**.**`flug\_id`

**INNER** **JOIN** `fluglinie` **AS** `fl`

**ON** `fl`**.**`fluglinie\_id` **=** `f`**.**`fluglinie\_id`

**WHERE**

`fl`**.**`firmenname` **=** 'Australia Airlines'

**AND**

`pd`**.**`geburtsdatum` **BETWEEN** **CAST(**CURDATE**()** **-** 250000 **AS** **DATE)** **AND** **CAST(**CURDATE**()** **-** 200000 **AS** **DATE)**

**;**

## 

**SELECT**

`p`**.\***

**FROM**

`passagier` **AS** `p`

**INNER** **JOIN** `buchung` **AS** `b`

**ON** `b`**.**`passagier\_id` **=** `p`**.**`passagier\_id`

**INNER** **JOIN** `flug` **AS** `f`

**ON** `f`**.**`flug\_id` **=** `b`**.**`flug\_id`

**INNER** **JOIN** `flugzeug` **AS** `fz`

**ON** `fz`**.**`flugzeug\_id` **=** `f`**.**`flugzeug\_id`

**INNER** **JOIN** `flugzeug\_typ` `ft`

**ON** `ft`**.**`typ\_id` **=** `fz`**.**`typ\_id`

**WHERE**

`ft`**.**`bezeichnung` **=** 'Airbus 380'

**;**

Die Bezeichnung „Airbus 380“ stimmte in der Angabe nicht, da er in der Datenbank als „Airbus A380“ abgespeichert ist.

## 

**SELECT**

**\***

**FROM**

`flugzeug` **AS** `fz`

**WHERE**

`fz`**.**`flugzeug\_id` **NOT** **IN** **(SELECT** `flugzeug\_id` **FROM** `flug`**)**

**;**

Korrelierte Queries sind nicht gut für die Performance, deswegen habe ich diese Aufgabe zunächst mit einem normalen Subquery gelöst. Hier nachfolgend noch eine Lösung mit einem korrelierten Query, die ich allerdings nicht testen konnte, da PHPMyadmin dabei out of sync geht.

**SELECT**

**\***

**FROM**

`flugzeug`

**WHERE**

**(**

**SELECT**

`flug`**.**`flugzeug\_id`

**FROM**

`flug`

**WHERE**

`flug`**.**`flugzeug\_id` **=** `flugzeug`**.**`flugzeug\_id`

**)** **IS** **NULL**

**;**

## 

**SELECT**

**MAX(**`fz`**.**`kapazitaet`**)**

**FROM**

`flugzeug` **AS** `fz`

**INNER** **JOIN** `flug` **AS** `f`

**ON** `f`**.**`flugzeug\_id` **=** `fz`**.**`flugzeug\_id`

**INNER** **JOIN** `fluglinie` `fl`

**ON** `fl`**.**`fluglinie\_id` **=** `f`**.**`fluglinie\_id`

**WHERE**

`fl`**.**`firmenname` **=** 'Thailand Airlines'

**;**

## 

**SELECT** **DISTINCT**

`fh1`**.**`name` **AS** `Startflughafen`**,**

`fh2`**.**`name` **AS** `Direkt erreichbar`**,**

**IFNULL(**`fh3`**.**`name`**,** ""**)** **AS** `Erreichbar über zwei Hops`**,**

**IFNULL(**`fh4`**.**`name`**,** ""**)** **AS** `Erreichbar über drei Hops`**,**

**IFNULL(**`fh5`**.**`name`**,** ""**)** **AS** `Erreichbar über vier Hops`

**FROM**

`flughafen` **AS** `fh1`

**INNER** **JOIN** `flug` **AS** `f1` **ON** `f1`**.**`von` **=** `fh1`**.**`flughafen\_id`

**INNER** **JOIN** `flughafen` **AS** `fh2` **ON** `fh2`**.**`flughafen\_id` **=** `f1`**.**`nach`

**LEFT** **JOIN** `flug` **AS** `f2` **ON** `f2`**.**`von` **=** `fh2`**.**`flughafen\_id`

**LEFT** **JOIN** `flughafen` **AS** `fh3` **ON** `fh3`**.**`flughafen\_id` **=** `f2`**.**`nach`

**LEFT** **JOIN** `flug` **AS** `f3` **ON** `f3`**.**`von` **=** `fh3`**.**`flughafen\_id`

**LEFT** **JOIN** `flughafen` **AS** `fh4` **ON** `fh4`**.**`flughafen\_id` **=** `f3`**.**`nach`

**LEFT** **JOIN** `flug` **AS** `f4` **ON** `f4`**.**`von` **=** `fh4`**.**`flughafen\_id`

**LEFT** **JOIN** `flughafen` **AS** `fh5` **ON** `fh5`**.**`flughafen\_id` **=** `f4`**.**`nach`

**WHERE**

`fh1`**.**`flughafen\_id` **=** 2233

**;**

**SELECT** **DISTINCT**

`fh1`**.**`name` **AS** `Startflughafen`**,**

`fh2`**.**`name` **AS** `Direkt erreichbar`**,**

**IFNULL(**`fh3`**.**`name`**,** ""**)** **AS** `Erreichbar über zwei Hops`**,**

**IFNULL(**`fh4`**.**`name`**,** ""**)** **AS** `Erreichbar über drei Hops`**,**

**IFNULL(**`fh5`**.**`name`**,** ""**)** **AS** `Erreichbar über vier Hops`

**FROM**

`flughafen` **AS** `fh1`

**INNER** **JOIN** `flugplan` **AS** `f1` **ON** `f1`**.**`von` **=** `fh1`**.**`flughafen\_id`

**INNER** **JOIN** `flughafen` **AS** `fh2` **ON** `fh2`**.**`flughafen\_id` **=** `f1`**.**`nach`

**LEFT** **JOIN** `flugplan` **AS** `f2` **ON** `f2`**.**`von` **=** `fh2`**.**`flughafen\_id`

**LEFT** **JOIN** `flughafen` **AS** `fh3` **ON** `fh3`**.**`flughafen\_id` **=** `f2`**.**`nach`

**LEFT** **JOIN** `flugplan` **AS** `f3` **ON** `f3`**.**`von` **=** `fh3`**.**`flughafen\_id`

**LEFT** **JOIN** `flughafen` **AS** `fh4` **ON** `fh4`**.**`flughafen\_id` **=** `f3`**.**`nach`

**LEFT** **JOIN** `flugplan` **AS** `f4` **ON** `f4`**.**`von` **=** `fh4`**.**`flughafen\_id`

**LEFT** **JOIN** `flughafen` **AS** `fh5` **ON** `fh5`**.**`flughafen\_id` **=** `f4`**.**`nach`

**WHERE**

`fh1`**.**`flughafen\_id` **=** 2233

**;**

Wieder keine Ergebnisse, da vom Flughafen mit der ID 2233 keine Flüge ausgehen.

Es ist unklar ob im Flugplan das „nach” sich auf die Endstation oder den letzten Flughafen bezieht, selbiges gilt natürlich für Flug.

Die Tabelle „flughafen\_erreichbar” sieht aus, als würde sie das Ergebnis eines „SELECT INTO”-Querys sein, auf jeden Fall hilft sie nicht die Aufgabenstellung zu lösen.

# Normalformen

## R1

R1(A,B,C,D) mit FD1 = {A B, B C, A D}

Schlüsselkandidaten: {A}

R1{A, B, C, D}

1.NF: Ja – Alle Attributwerte sind atomar laut Angabe.

2.NF: Ja – kein Attribut ist abhängig von einer Schlüsselkandidatenteilmenge.

3.NF: Nein – C ist von A über B transitiv abhängig.

BCNF: Nein – Relation ist nicht in 3.NF.

R2(A,B,C,D) mit FD2 = {AB C, B D}

Schlüsselkandidaten: {A,B}

R2{A, B, C, D}

1.NF: Ja – Alle Attributwerte sind atomar laut Angabe.

2.NF: Nein – D ist abhängig von Schlüsselkandidatenteilmenge (B).

3.NF: Nein – Relation ist nicht in 2.NF.

BCNF: Nein – Relation ist nicht in 3.NF.

R3(A,B,C,D) mit FD3 = {AB C, AC D, AD B}

Schlüsselkandidaten: {A,B,C,D}

R3{\_A\_, \_B\_, \_C\_, \_D\_}

1.NF: Ja – Alle Attributwerte sind atomar laut Angabe.

2.NF: Ja – Es gibt keine Nichtschlüsselattribute.

3.NF: Ja – Es gibt keine Nichtschlüsselattribute.

BCNF: Nein – keine Determinante ist Superschlüssel.

# B-Baum

### 5 einfügen





### 12 löschen



### 2 löschen



### 123 löschen





### 555 löschen



Eine Formel habe ich im Skriptum leider nicht gefunden dazu. Die folgende Formel habe ich aus einem Skriptum von Udo Kelter.

b Blockgröße in Bytes: 2048 B

t Platzbedarf für eine Medienadresse (Verweis auf Teilbaum): 8 B

k Platzbedarf für einen Schlüsselwert: 8 B

i Platzbedarf für Satzinhalt: 20 B

Ordnung des Baumes n = 56.