

12.12.2019

## Übungsblatt 8 – Lösungsvorschlag

### Diskussionsteil (im PS zu lösen; keine Abgabe nötig)

#### a) Rekursion und Common Table Expressions (CTE)

a) Beantworten Sie folgende Fragen zu Common Table Expressions:

i. Was macht eine Common Table Expression?

#### Lösung



Eine CTE produziert ein temporäres Ergebnis, auf das man in der folgenden Query zugreifen kann. CTEs werden mit der **WITH** Klausel erstellt.

ii. Wofür werden Common Table Expressions eingesetzt?

#### Lösung



CTEs erlauben es, komplexe Queries zu vereinfachen und hierarchische Daten abzufragen.

iii. Wie hängen CTEs mit rekursiven Abfragen zusammen?

#### Lösung



Rekursive Abfragen sind CTEs, die auf sich selbst verweisen.

iv. Geben Sie zwei Anwendungsfälle für rekursive Abfragen an.

#### Lösung



Wegsuche, Vorfahrenermittlung

b) Öffnen Sie das im Rahmen einer DBIS-Bachelorarbeit entwickelte Rekursions-Tool unter der Adresse: <https://dbis-uibk.github.io/recursion/>.

Wählen Sie das Datenset *Flights Simple* aus. Nun können Sie anhand von zwei Examples die Ausführung der Abfrage schrittweise verfolgen. Versuchen Sie zu verstehen, wie die Rekursion funktioniert.

A	B	C	D	E	F	G
1	x	S	1.4	ja	101	klein
2	y	M	2.4	nein	102	groß
3	x	XL	1.5	ja	101	mittel
4	z	L	1.4	ja	105	klein
5	z	M	0.0	ja	110	mittel

b) **Funktionale Abhängigkeiten:**

Prüfen Sie, ob folgende funktionale Abhängigkeiten auf den in der Tabelle gegebenen Daten gelten:

a) Gilt  $A \rightarrow E$ ?

<b>Lösung</b>	✓
Gilt.	

b) Gilt  $B \rightarrow D$ ?

<b>Lösung</b>	✓
Gilt nicht, da für z unterschiedliche Werte vorhanden sind.	

c) Gilt  $G \rightarrow E$ ?

<b>Lösung</b>	✓
Gilt.	

d) Gilt  $AB \rightarrow D$ ?

<b>Lösung</b>	✓
Gilt (allein dadurch überprüfbar, dass A eindeutig ist).	

e) Gilt  $EG \rightarrow C$ ?

<b>Lösung</b>	✓
Gilt nicht, da für ja/klein verschiedene Werte (S, L) vorhanden sind.	

c) **Theorie:** Diskutieren Sie folgende Fragen zu Schlüsseln:

- Bei der Modellierung von Personen wird oft z.B. die Sozialversicherungsnummer als Schlüssel verwendet. Welche funktionalen Abhängigkeiten ergeben sich daraus? Zählen Sie einige Beispiele auf.

**Lösung**

Von der Sozialversicherungsnummer kann eindeutig auf alle Attribute der Person z.B. auf den Namen oder das Geburtsdatum der Person geschlossen werden.

- Wie unterscheiden sich funktionale Abhängigkeit und voll funktionale Abhängigkeit?

**Lösung**

Eine funktionale Abhängigkeit  $\alpha \rightarrow \beta$  bedeutet, dass für alle vorhandenen Tupel einer Relation  $R$  die Werte von  $\beta$  eindeutig durch  $\alpha$  bestimmt sind (d.h. von  $\alpha$  abgeleitet werden können). Voll funktionale Abhängigkeit ist hingegen strenger, da eine Attributkombination  $\beta$  nur dann als voll funktional abhängig von  $\alpha$  gilt, wenn  $\alpha$  nicht verkleinerbar ist (d.h. es keine echte Teilmenge von  $\alpha$  gibt, von der  $\beta$  voll funktional abhängig ist).

- Warum erfüllt eine Relation in 1NF, die nur einelementige Schlüsselkandidaten besitzt, automatisch die 2NF?

**Lösung**

Die 2NF wird erfüllt, da kein Nichtschlüsselattribut von einem Teil eines Schlüssels abhängen kann (die Schlüssel sind einelementig und können nicht mehr verkleinert werden).

- d) **NF-Bestimmung:** Gegeben seien die abstrakten Relationenschemata  $R$  über die Attribute  $A, B, C, D, E$  mit den zugehörigen funktionalen Abhängigkeiten  $FA$ . Sie können dabei davon ausgehen, dass alle Attributwerte atomar sind.

Bestimmen Sie für jedes Schema alle möglichen Schlüsselkandidaten und prüfen Sie, ob dieses der 1NF, 2NF, 3NF oder BCNF genügt.

- $R(A, B, C, D, E)$  mit  $FA = \{DE \rightarrow AC, B \rightarrow ADE\}$

**Lösung**

$SK = \{B\}$

$\mathcal{R}$  befindet sich in der 2. NF, da die folgende NF durch die FA  $DE \rightarrow AC$  verletzt wird.

- $R(A, B, C, D, E)$  mit  $FA = \{BC \rightarrow AE, E \rightarrow A, AC \rightarrow BD\}$

**Lösung**

$SK = \{CE, BC, AC\}$

$\mathcal{R}$  befindet sich in der 3. NF, da die folgende NF durch die FA  $E \rightarrow A$  verletzt wird.

- $R(A, B, C, D, E)$  mit  $FA = \{AE \rightarrow BC, CE \rightarrow ABD\}$

## Lösung



$SK = \{CE, AE\}$

$\mathcal{R}$  befindet sich in der BCNF (auf der linken Seite stehen nur Superschlüssel).

## Hausaufgabenteil (Zuhause zu lösen; Abgabe nötig)

### Aufgabe 1 (Rekursion)

[5 Punkte]

Gegeben sei eine Datenbank zum Verwalten eines sozialen Netzwerks mit folgenden Tabellen:

```
person(id, firstname, lastname, year_of_birth, country_id)
friendship(person1_id, person2_id, friends_since)
follow(person_id, followed_person_id, follows_since)
```

Ein Eintrag in *friendship* besagt, dass *person1\_id* mit *person2\_id* befreundet ist. Da Freundschaften bidirektional sind, ist immer nur ein Paar pro Freundschaft verzeichnet: (x, y) bedeutet, dass sowohl Person x mit Person y befreundet ist, als auch Person y mit Person x. Bitte beachten Sie, dass es keinen expliziten Eintrag (y,x) in der Tabelle gibt (obwohl dies semantisch korrekt wäre). Ein Eintrag in *follow* besagt, dass *person\_id* (der follower) der Person *followed\_person\_id* (der followee) folgt. In diesem Fall ist es nicht zwingend notwendig, dass auch die Gegenrichtung gilt. D.h. wenn dies der Fall wäre, dann gäbe es dafür auch einen eigenen Eintrag.

Friendship- und follow-Beziehungen kann man übersichtlich in einem Graph darstellen. Der Graph in Abbildung 1 enthält einen kleinen Auszug der follow-Beziehungen der fakebook-Datenbank. In diesem Graphen können Sie zum Beispiel sehen, dass Laura drei Personen direkt folgt (und damit ihr *follower* ist): Lori, Kathleen und Evelyn (die sogenannten *followees*). Indirekt (also über mehrere Zwischenschritte) folgt Laura aber auch z.B. Matthew (über Lori). Die Erreichbarkeit bzw. Nähe von anderen Knoten (in diesem Fall Personen) werden meist mit der Anzahl der sogenannten Hops (Zwischenschritte) im Graph gemessen. Wie im Graphen für einige Beispiele in rot gekennzeichnet, folgt Lori Evelyn über einen Hop (also direkt) und Matthew über zwei Hops. Laura folgt Lori einerseits direkt, aber auch über drei Hops (Laura->Kathleen->Beverly->Lori).

Die fakebook-Datenbank(fakebook-backup.sql), die im OLAT zu finden ist, können Sie mithilfe des Query Tools von PgAdmin einspielen.

- a) 1 Punkt Geben Sie **Wayne** (Vorname) **Hayes'** (Nachname) Followees aus, denen er maximal über 4 Hops folgt. Berechnen Sie zudem den Follow-Pfad (string), aus dem hervorgeht, wie die Beziehung zustande kommt.

Der Follow-Pfad soll folgendem Format entsprechen: ID->ID->...->ID (fügen Sie keine Leerzeichen ein!!!).

Das Ergebnis mit folgenden Spalten soll aufsteigend nach step ausgegeben werden:

- person\_id (Wayne Hayes' ID)
- following (ID des Followees)
- path (der Pfad als string)
- step (die Anzahl der Hops als integer)

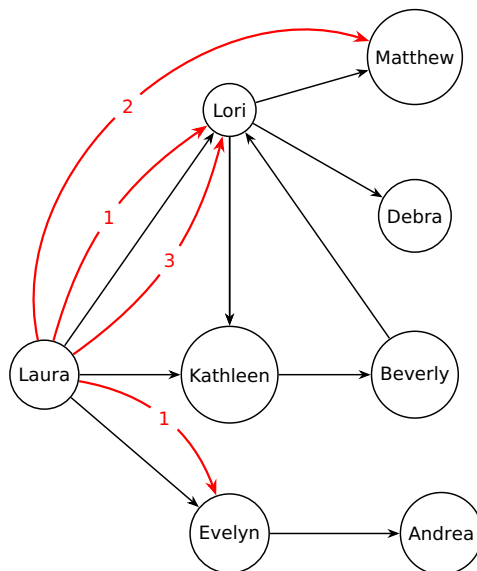


Abbildung 1: follow-Graph (Auszug)

Das Ergebnis sollte wie folgt aussehen:

person_id	following	path	step
50	113	50->113	1
...	...	...	...
50	854	50->113->201->854	3
...	...	...	...
50	752	50->331->767->998->752	4

Tabelle 1: Ausgabe zur Aufgabe 1a.

#### Hinweis



Sie können zum Verketteten von Zeichenketten die Funktion CONCAT<sup>a</sup> verwenden.

<sup>a</sup><https://www.postgresql.org/docs/9.1/functions-string.html>

#### Abgabe



exercise1\_a.sql

#### Lösung



```
WITH RECURSIVE wh_follow_path(person_id, FOLLOWING, PATH, step) AS
  (SELECT f.person_id,
    f.followed_person_id,
    concat(f.person_id, '->', f.followed_person_id),
    1
  FROM follow AS f
```

```

INNER JOIN person AS p ON f.person_id=p.id
WHERE p.firstname='Wayne'
      AND p.lastname='Hayes'
UNION ALL SELECT t.following,
                  follow.followed_person_id,
                  concat(PATH, '->', follow.followed_person_id),
                  step+1
FROM wh_follow_path AS t
INNER JOIN follow ON t.following=follow.person_id
WHERE t.step < 4 )
SELECT
  (SELECT id
   FROM person
   WHERE firstname='Wayne'
        AND lastname='Hayes' ), FOLLOWING,
        PATH,
        step
FROM wh_follow_path
ORDER BY step ASC

```

person_id	following	path	step
50	113	50->113	1
50	191	50->191	1
...	...	...	...
50	854	50->113->201->854	3
...	...	...	...
50	752	50->331->767->998->752	4

**548** Tupel gesamt

- b) 1 Punkt In einem Graphen ist es auch möglich, über verschiedene Pfade von Person A zu Person B zu gelangen, dies hat zur Folge, dass bestimmte Followees über mehrere Pfade erreicht werden können. Geben Sie nun für jeden Followee von **Wayne Hayes**, der direkt (1 Hop) oder indirekt mit bis zu 4 Hops erreicht werden kann, die Anzahl der Pfade aus, über die der Followee erreicht werden kann. Schließen Sie dabei Zyklen aus, die wie folgt definiert sind:

*Zyklus* :  $\exists k_s \in P$ , wobei  $P$  der Pfad mit  $n$  Knoten ist, sodass  $k_i \rightarrow \dots \rightarrow k_s \rightarrow \dots \rightarrow k_j \rightarrow \dots \rightarrow k_s \rightarrow \dots \rightarrow k_k$  mit  $j \neq s$  gegeben ist.

Laura aus dem obigen Beispiel kann mit 4 Hops ohne Zyklus Matthew und Debra erreichen, Kathleen kommt nicht hinzu (Laura->Kathleen->Beverly->Kathleen - Zyklus!!).

Das Ergebnis mit folgenden Spalten soll aufsteigend nach Count ausgegeben werden:

- firstname
- lastname
- count

Das Ergebnis sollte wie folgt aussehen:

firstname	lastname	count
Albert	Foster	1
...	...	...
Helen	Garcia	2
...	...	...
Nathan	Lee	3

Tabelle 2: Ausgabe zur Aufgabe 1b.

#### Hinweis



Die Zykelbestimmung lässt sich über String-Funktionen realisieren.

#### Abgabe



exercise1\_b.sql

#### Lösung



```
WITH RECURSIVE wh_follow_path(person_id, FOLLOWING, PATH, step) AS
  (SELECT f.person_id,
         f.followed_person_id,
         concat(f.person_id, '->', f.followed_person_id),
         1
   FROM follow AS f
   INNER JOIN person AS p ON f.person_id=p.id
   WHERE p.firstname='Wayne'
        AND p.lastname='Hayes'
   UNION ALL SELECT t.following,
                  follow.followed_person_id,
                  concat(PATH, '->', follow.followed_person_id),
                  step+1
   FROM wh_follow_path as t
   INNER JOIN follow ON t.following=follow.person_id
   INNER JOIN person AS p ON follow.followed_person_id = p.id
   WHERE t.path NOT LIKE '%' || follow.followed_person_id || '%'
        AND t.step < 4 )
SELECT p.firstname,
       p.lastname,
       count(FOLLOWING)
FROM wh_follow_path as t
INNER JOIN person AS p ON t.following=p.id
```

```
GROUP BY FOLLOWING,
        p.firstname,
        p.lastname
ORDER BY COUNT ASC
```

firstname	lastname	count
Albert	Foster	1
Brenda	James	1
Carolyn	Sanchez	1
...	...	...
Helen	Garcia	2
...	...	...
Nathan	Lee	3

**402** Tupel gesamt.

- c) 1 Punkt Geben Sie **Wayne Hayes'** Followee aus, denen er exakt nach 4 Hops folgt. Berücksichtigen Sie zudem folgende Kriterien:

- Followees, die in weniger Hops erreicht werden können, sollen ausgeschlossen werden
- Der Pfad darf keine Zyklen enthalten.


Laura aus dem Beispiel kann exakt über 2 Hops Andrea, Beverly, Debra und Matthew, ohne diese über einen kürzeren Pfad (1 Hop) gesehen zu haben, erreichen, die anderen Kontakte erreicht sie über mehrere Pfade, sodass ein kürzerer Pfad immer gegeben ist. Das Ergebnis mit folgenden Spalten soll aufsteigend nach Hops ausgegeben werden:

- person\_id (Wayne Hayes' id)
- following (ID des Followees)
- path
- step

Die Ausgabe soll gemäß Tabelle 1 erfolgen.

#### Abgabe



 exercise1\_c.sql

#### Lösung



```
WITH RECURSIVE wh_follow_path(person_id, FOLLOWING, PATH, step) AS
    (SELECT f.person_id,
        f.followed_person_id,
        concat(f.person_id, '->', f.followed_person_id),
        1
    FROM follow AS f
    INNER JOIN person AS p ON f.person_id=p.id
```



```

WHERE p.firstname='Wayne'
      AND p.lastname='Hayes'
UNION ALL SELECT t.following,
                  follow.followed_person_id,
                  concat(PATH, '->', follow.followed_person_id),
                  step+1
FROM wh_follow_path AS t
INNER JOIN follow ON t.following=follow.person_id
INNER JOIN person AS p ON follow.followed_person_id = p.id
WHERE t.path NOT LIKE '%' || follow.followed_person_id || '%'
      AND t.step < 4 )
SELECT
  (SELECT id
   FROM person
   WHERE firstname='Wayne'
        AND lastname='Hayes' ) AS id,
        FOLLOWING AS follows,
        PATH,
        step
FROM wh_follow_path AS t
WHERE t.step=4
      AND t.following IN
      (SELECT t.following
       FROM wh_follow_path AS t
       GROUP BY t.following
       HAVING count(t.following) = 1)
ORDER BY person_id ASC

```

person_id	following	path	step
50	172	50->331->614->30->172	4
50	29	50->331->614->30->29	4
...	...	...	...

**234** Tupel gesamt

- d) 1 Punkt Berechnen Sie für **Shirley** (Vorname) **Moore** (Nachname) die Metrik Freundes-Freunde, die wie folgt definiert ist:

$Freunde(x)$  : alle Freunde von  $x$

$Freunde'(X)$  :  $Freunde(x_1) \cup Freunde(x_2) \cup \dots \cup Freundw(x_n)$

$FreundesFreunde(x)$  :  $|Freunde'(Freunde(x))|$

Die Ergebnisspalte soll **number\_of\_friends** lauten.

## Hinweis



Die für diese Aufgabe benötigte friendship-Relation enthält wie eingangs beschrieben nur einen Eintrag pro Freundschaftspaar, d.h. wenn 3 mit 10 befreundet ist, dann gibt es einen Eintrag (3,10), aber keinen für (10,3). Bereiten Sie sich mithilfe einer Common Table Expression (CTE) die Ausgangsrelation deshalb so vor, dass sie für eine anschließende Rekursion geeignet ist. Beachten Sie, dass das Schlüsselwort **RECURSIVE** am Anfang stehen muss, ohne **RECURSIVE** darf eine CTE nur auf etwas verweisen, das lexikalisch vorher definiert wurde.

## Abgabe



exercise1\_d.sql

## Lösung



```
WITH RECURSIVE table_friends(id, friend, step) AS
    (SELECT i.person1_id,
           i.person2_id,
           1
    FROM initial_set AS i
    INNER JOIN person AS p ON i.person1_id=p.id
    WHERE p.firstname='Shirley'
           AND p.lastname='Moore'
    UNION ALL SELECT f.person1_id,
                    f.person2_id,
                    t.step+1
    FROM table_friends AS t
    INNER JOIN friendship AS f ON t.friend=f.person1_id
    WHERE step<2 ),
    initial_set AS
    (SELECT *
    FROM friendship
    UNION SELECT person2_id,
                person1_id,
                friends_since
    FROM friendship)
SELECT count(DISTINCT(friend))-1 AS number_of_friends
FROM table_friends
WHERE step=2
```

**number\_of\_friends**

190


- e) 1 Punkt Berechnen Sie für **Wayne Hayes** die Metrik Freund-Und-Freundes-Freunde, die wie folgt definiert ist:

$$\text{FreundUndFreundesFreund}(x) : |\text{Freunde}(x) \cup \text{Freunde}'(\text{Freunde}(x))|$$

Die Ergebnisspalte soll **number\_of\_friends** lauten.

#### Abgabe



 exercise1\_e.sql

#### Lösung



```
WITH RECURSIVE table_friends(id, friend, step) AS
  (SELECT i.person1_id,
         i.person2_id,
         1
   FROM initial_set AS i
  INNER JOIN person AS p ON i.person1_id=p.id
 WHERE p.firstname='Wayne'
       AND p.lastname='Hayes'
  UNION ALL SELECT f.person1_id,
                  f.person2_id,
                  t.step+1
   FROM table_friends AS t
  JOIN friendship AS f ON t.friend=f.person1_id
 WHERE step<2 ),
      initial_set AS
  (SELECT *
   FROM friendship
  UNION SELECT person2_id,
              person1_id,
              friends_since
   FROM friendship)
SELECT count(DISTINCT(t.friend))-1 AS number_of_friends
FROM table_friends AS t
```

<b>number_of_friends</b>
<hr/>
157
<hr/>

## Aufgabe 2 (Normalformenbestimmung)

**[1 Punkt]**

Gegeben seien die abstrakten Relationenschemata  $R$  über die Attribute  $A, B, C, D, E$  mit den zugehörigen funktionalen Abhängigkeiten  $FA$ .

Bestimmen Sie für jedes Schema alle möglichen Schlüsselkandidaten und prüfen Sie, ob diese der 1NF, 2NF, 3NF oder BCNF genügen und begründen Sie Ihre Entscheidung. Sie können dabei davon ausgehen, dass alle Attributwerte atomar sind.

- a) 0.25 Punkte  $FA = \{AE \rightarrow B, C \rightarrow ABD, B \rightarrow CDE\}$

**Lösung**



$SK = \{C, B, AE\}$   
 $\mathcal{R}$  befindet sich in der BCNF.

**Abgabe**



exercise2\_a.txt

- b) 0.25 Punkte  $FA = \{BD \rightarrow E, CE \rightarrow BD, BC \rightarrow AD, A \rightarrow BCE\}$

**Lösung**



$SK = \{CE, BC, A\}$   
 $\mathcal{R}$  befindet sich in der 3. NF, da die folgende NF durch die FA  $BD \rightarrow E$  verletzt wurde.

**Abgabe**



exercise2\_b.txt

- c) 0.25 Punkte  $FA = \{ABD \rightarrow C, CDE \rightarrow A, BCD \rightarrow E\}$

**Lösung**



$SK = \{BCD, ABD\}$   
 $\mathcal{R}$  befindet sich in der 3. NF, da die folgende NF durch die FA  $CDE \rightarrow A$  verletzt wurde.

**Abgabe**



exercise2\_c.txt

- d) 0.25 Punkte  $FA = \{CDE \rightarrow B, BDE \rightarrow C, AE \rightarrow BD\}$

**Lösung**



$SK = \{AE\}$   
 $\mathcal{R}$  befindet sich in der 2. NF, da die folgende NF durch die FA  $CDE \rightarrow B, BDE \rightarrow C$  verletzt wurde.

**Abgabe**



exercise2\_d.txt

### Aufgabe 3 (Algorithmen)

[4 Punkte]

- a) 2 Punkte **Kanonische Überdeckung:** Berechnen Sie für das Schema  $R(A, B, C, D, E, F, G)$  mit den folgenden funktionalen Abhängigkeiten die kanonische Überdeckung:  $FA = \{A \rightarrow D, C \rightarrow EF, F \rightarrow AD, G \rightarrow BC, EF \rightarrow G, AGF \rightarrow E\}$

#### Lösung



1) Linksreduktion:

Überprüfen, ob  $\{\alpha\}^+ = \{\alpha - B\}^+$ ; dann kann B gestrichen werden

$AGF \rightarrow E$  reduzieren zu  $G \rightarrow E$

aktueller Stand:

$A \rightarrow D,$

$C \rightarrow EF,$

$F \rightarrow AD,$

$G \rightarrow BC,$

$EF \rightarrow G,$

$G \rightarrow E$

2) Rechtsreduktion:

$F \rightarrow AD$  reduzieren zu  $F \rightarrow A$

aktueller Stand:

$A \rightarrow D,$

$C \rightarrow EF,$

$F \rightarrow A,$

$G \rightarrow BC,$

$G \rightarrow E$

3) Zusammenfassen:

$A \rightarrow D,$

$C \rightarrow F,$

$F \rightarrow A,$

$G \rightarrow B, C, E$

4) kanonische Überdeckung ist somit:

$A \rightarrow D,$


$C \rightarrow F,$


$F \rightarrow A,$

$G \rightarrow B, C, E$

#### Abgabe



 exercise3\_a.txt oder

 exercise3\_a.pdf

Bitte geben Sie im Abgabefile nachvollziehbar die durchgeführten Schritte an.

- b) 2 Punkte **Relationensynthese:** Gegeben sei das Schema  $R(A, B, C, D, E, F, G, H)$  mit der

kanonischen Überdeckung  $F_C = \{A \rightarrow DF, G \rightarrow B, C \rightarrow AH, E \rightarrow G, B \rightarrow C\}$ . Berechnen Sie eine verlustfreie und abhängigkeitsbewahrende Relationenzerlegung von R in 3NF mithilfe der Relationensynthese.


### Lösung




- 1) (Schritt 1) kanonische Überdeckung bereits gegeben
- 2) (Schritt 2) erzeuge Relation  $R_1(A, D, F)$  mit  $F_1 = \{A \rightarrow DF\}$
- 3) (Schritt 3) erzeuge Relation  $R_2(B, G)$  mit  $F_2 = \{G \rightarrow B\}$
- 4) (Schritt 4) erzeuge Relation  $R_3(A, C, H)$  mit  $F_3 = \{C \rightarrow AH\}$
- 5) (Schritt 5) erzeuge Relation  $R_4(E, G)$  mit  $F_4 = \{E \rightarrow G\}$
- 6) (Schritt 6) erzeuge Relation  $R_5(B, C)$  mit  $F_5 = \{B \rightarrow C\}$
- 7) (Schritt 7) Schlüsselkandidat  $E$ , daher keine weitere Relation nötig.

### Abgabe



 exercise3\_b.txt oder

 exercise3\_b.pdf

Bitte geben Sie im Abgabefile nachvollziehbar die durchgeführten Schritte an.

**Wichtig:** Laden Sie bitte Ihre Lösung in OLAT hoch und geben Sie mittels der Ankreuzliste auch unbedingt an, welche Aufgaben Sie gelöst haben. Die Deadline dafür läuft am Vortag des Proseminars um 23:59 (Mitternacht) ab.