IDB SS23 Solutions & Notes

Igor Dimitrov Jacob Rose Jonathan Barthelmes

Table of contents

IDB SS 23 Loesungen & Notizen **4**

I Loesungen **5**

1 Blatt 1 **6**
1.1 Grundlagen der Logik 6
1.2 1-2 Relationale Algebra 7
1.3 1-3 Datenmanagementsysteme 8
1.4 Feedback 9

2 Blatt 2 **11**
2.1 Aufgabe 1 11
2.2 Aufgabe 2 11
2.3 Aufgabe 3 13
2.4 Aufgabe 4 13
2.5 Feedback 15

3 Blatt 3 **16**
3.1 Aufgabe 1 16
3.2 Aufgabe 2 16

4 Blatt 4 **19**
4.1 Aufgabe 1 19
4.2 Aufgabe 2 19

5 Blatt 5 **21**
5.1 Aufgabe 1 21
5.2 Aufgabe 2 23
5.3 Aufgabe 3 25
5.4 Aufgabe 4 27

6 Blatt 6 **30**
6.1 Aufgabe 1 30
6.2 Aufgabe 2 32
6.3 Aufgabe 3 34

7 Blatt 7 **36**
7.1 Aufgabe 1 36
7.2 Aufgabe 2 38

8 Blatt 8 **40**
8.1 Aufgabe 1 40
8.2 Aufgabe 2 42
8.3 Aufgabe 3 42

II Notes **44**

IDB SS 23 Loesungen & Notizen

Loesungen der Uebungsaufgaben und Notizen von der Vorlesung Datenbanken Sommersemester 23 Uni Heidelberg.

Part I

Loesungen

1 Blatt 1

1.1 Grundlagen der Logik

a)

1. Wenn zwei Tiere im selben Lebensraum leben, essen sie auch das selbe.

   falsch: t_1 (Gepard) und t_9 (Uganda-Grasantilope) haben den gleichen lebensraum "Regenwald" aber andere Ernaehrungen; Karnivore und bzw. Herbivore.

   erfuellbar: Jede Datenbank, die ein einziges Tier enthaelt erfuellt diese Aussage automatisch.

2. Fuer jedes Zootier existier ein anderes Zootier, welches entweder die selbe Nahrung isst oder im selben Lebensraum lebt. ***wahr***

3. Es existieren drei Zootiere, so dass erstes und zweites ,sowie zweites und drittes den gleichen Lebensraum teilen aber erstes und drittes nicht.

   falsch und nicht erfuellbar: $l(x, y)$ ist eine Aequivalenzrelation. Somit gilt Transitivitaet: $l(x, y) \wedge l(y, z) \rightarrow l(x, z)$

4. Es gibt keine zwei unterschiedliche Tiere, die sowohl der gleichen Familie zugehoerig sind als auch den gleichen Lebensraum teilen.

   falsch: t_6 und t_{10} sind beide Sakiaffen mit dem Lebensraum Regenwald.

   erfuellbar: Jede Datenbank mit einem einzigen Element efuellt diese Aussage automatisch.

b)

1. $\forall x \in T \exists y \in T : x \neq y \wedge fam(x, y) \wedge \neg ls(x, y)$
2. $\forall x \in T \forall y \in T : fam(x, y) \wedge le(x, y) \wedge er(x, y) \rightarrow x = y$
3. $\forall x \in T \forall y \in T : fam(x, y) \rightarrow er(x, y)$

1.2 1-2 Relationale Algebra

1. Gebe die Modelle von Flugzeugen, die so heissen, wie einer aus dem Personal.

Modell
Quack

2. Gebe die crew ID der Mitarbeiter, die nicht an den afugelisteten Fluegen beteiligt sind.

cid
c090

3. Gebe die Flugnummer der Fluege, die in Deutschland starten.

Flugnr
DB2013
DB2341

4. Gebe alle Modelle aus, fuer die eine Crew-Mitglied zugelassen ist.

Zulassung
A320
B787
A380
A340
B747

5. Gebe alle Namen von Piloten aus, die fuer eine Maschine zugelassen sind mit Reichweite ≤ 10000.

Name
Pan
Schmitt

6. Gebe Start, Ziel und Modell fuer alle Modelle aus, die ungeiegnet fuer einen Flug sind, weil sie die Strecke nicht fliegen koennen.

Start	**Ziel**	**Modell**
FRA	JFK	A320
JFK	FRA	A320
CDG	LAX	A320

7. Waehle aus Fluegen die gleichen Flugnummern, die an unterschiedlichen Tagen fliegen, d.h gebe Flugnummern der Fluegen, die Rundfahrten sind.

Flugnr
DB2013

8. Gebe die Laender aus, aus denen keine Flugzeuge starten.

Land
Deutschland

1.3 1-3 Datenmanagementsysteme

a) XML und HTML basieren sich beide auf **SGML** - eine Metasprache, mit deren Hilfe man verschiedene Markup-sprachen fuer Dokumente definieren kann.

XML ist eine erweiterbare Markup-sprache, die zur Darstellung & Speicherung hierarchisch strukturierter Daten und zur Definition & Entwicklung neuer Markup-sprachen verwendet wird. XML Dokumente haben eine Baumstruktur und bestehen aus **Elemente**, die durch **Tags** Ausgezeichnet werden. XML hat keinen vordifienerten Satz von Tags, wobei die genaue Struktur eines XML-Dokuments durch den **Dokumenttypdefinition** festgelegt werden kann.

HTML beschreibt die semantische Struktur und Formattierung der Inhalte von Webseiten und war urspruenglich eine Anwendung von SGML. Im Gegensatz zu XML hat HTML einen festen Satz von Tags, die fuer die Auszeichnung der Elementen verwendet werden koennen. Streng genommen ist HTML kein XML hat aber im wesentlichen die gleiche Struktur wie ein XML-Dokument. (Hierarchische Baumstruktur, Elemente, Tags, DOM).

Fuer XML gibt es viele standarte Werkzeuge, die XML Dokumente auf Wohlgeformtheit pruefen und porgrammatsich verarbeiten koennen, z.B. wie

- XML-Prozessor/Parser,
- **XQuery**: die standarde XML Abfrage- und Transformationssprache,
- **XPath**: Untersprache von XQuery, die XQuery unterstuetzt,
- **XSLT**: Sprache die speziell dazu geiegnet ist, XML Dokumente in andere Formate umzuwandeln.

Diese Tools stehen in XML Datenbanken zur verfuegung und XML Datenbanken sind fuer die Arbeit mit XML-Dokumenten optimiert. Somit koennen HTML-Dokumente mit den etablierten zahlreichen XML Tools optimal verarbeitet werden, wenn sie in einer XML Datenbank gespeichert werden.

Ein weiterer Vorteil ist, dass eine XML-Datenbank kein oder nur ein vereinfachtes Datenschema (Beziehungsschema/Tabellen) braucht, da die Daten schon durch das Dateiformat strukturiert werden. Bei einer relationael Datenbank muss das Schema explizit definiert werden. D.h. um ein HTML-Dokument in einer RDB zu speichern, oder um ein Dokument aus einer RDB zu exportieren muss jedes mal eine Transformation zwischen der HTML-Darstellung und relationalen Darstellung des Dokuments durchgefuehrt werden. Weiterhin funktioniert die Abbildung zwischen den Dokument-orientierten und relationalen Modellen nicht immer gut und wird als **object-relational impedence mismatch** bezeichnet.

b) **Vorteile**:

- Man benoetigt kein vordefiniertes Schema
- Kommt gut mit vielen Lese- und Schreibzugriffen zurecht.

Nachteile:

- Geringe konsistenzt/Gueltigkeit der Daten.

- Weil es weniger Einschraenkungen gibt, koennen die Abfragen nicht so gut optimiert werden wie bei den relationalen DBen.

1.4 Feedback

```
Rose, Dimitrov, Barthelmes

Aufgabe 1:
    a) Ja, technisch gesehen erfüllen Datenbanken, die nur ein Tier
 ↪  enthalten, die 1 und 4, wäre halt nur besser gewesen, wenn ihr ein
 ↪  "normales" Beispiel mit mindestens 2 Tieren gewählt hättet :D
```

```
9/9 Punkte

Aufgabe 2:
    1. Es wird das MODELL gesucht, nicht der NAME :D Also A380
    5. Nicht ganz, es sind eher alle Personen, die nicht für ein
↪   Flugzeug mit Reichweite >10000 zugelassen sind.
    6. "Ungeeignet"? Habt ihr nie von technischen Zwischenstopps gehört?
↪   :p
    7. Nicht unbedingt Rundfahrten
    8. "... Länder aus, die einen Flughafen haben, aus dem..."

8/11 Punkte

Aufgabe 3:

Habt ihr ChatGPT verwendet? Das sieht sehr nach ChatGPT aus...

4/4 Punkte

Insgesamt 21/24 Punkte
```

2 Blatt 2

2.1 Aufgabe 1

1. $\pi_{\texttt{pid, Name}}(\sigma_{\substack{\texttt{Rolle="Pilot",} \\ \texttt{Reichweite} \geq 15000}}(\texttt{Personal} \bowtie \texttt{Zulassung} \bowtie \texttt{Modell}))$
2. $\pi_{\texttt{Name}}(\sigma_{\texttt{Land='USA'}}(\beta_{\texttt{Code} \leftarrow \texttt{Ziel}}(\texttt{Flug}) \bowtie \texttt{Flughafen} \bowtie \texttt{Flugzeug}))$
3. $\pi_{\texttt{Code, Land}}\Big(\sigma_{\texttt{Name='F. Kohl'}}\Big(\texttt{Flugzeug} \bowtie \Big(\beta_{\texttt{Code} \leftarrow \texttt{Start}}(\pi_{\texttt{Start, fid}}(\texttt{Flug})) \cup \beta_{\texttt{Code} \leftarrow \texttt{Ziel}}(\pi_{\texttt{Ziel, fid}}(\texttt{Flug}))\Big)\Big)\Big)$
4. $\pi_{\texttt{pid, Name}}(\texttt{Personal}) - \pi_{\texttt{pid, Name}}(\texttt{Personal} \bowtie \texttt{Crew} \bowtie \sigma_{\texttt{Datum} < \texttt{07.04.2013}}(\texttt{Flug}))$

2.2 Aufgabe 2

1. **SQL**:

   ```
   select distinct C from R3
   ```

   Ergebniss:

   ```
   {{C: 7},
    {C: 8}}
   ```

2. **SQL**:

   ```
   select distinct * from R2
   where B = rot
   ```

   Ergebniss::

   ```
   {{B: rot, C: 9}}
   {{B: blau, C: 8}}
   ```

3. **SQL**:

```
select distinct * from R2
intersect
select distinct * from R3;
```

Ergebniss:

```
{{B: blau}, {C: 7}}
```

4. **SQL**:

```
select * from R2
union
select * from R3
```

Ergebniss:

```
{{B: blau,  C: 7},
{B: rot,   C: 8},
{B: rot,   C: 9},
{B: gruen, C: 8},
{B: gelb,  C: 7}}
```

5. **SQL**:

```
select * from R3 except (
    select * from R2
);
```

Ergebniss:

```
{{B: gruen, C: 8},
{B: gelb,  C: 7}}
```

6. **SQL**:

```
select distinct * from
R1 natural jo R2
```

Ergebniss:

```
{{A: q, B: rot, C: 8},
{A: q, B: rot, C: 9}}
```

7. **SQL**:

```
select distinct * from
R1, R2
```

Ergebniss:

```
{{A: q, R1.B: rot,   R2.B: blau,  C: 7 },
{A: q, R1.B: rot,   R2.B: gruen, C: 8},
{A: q, R1.B: rot,   R2.B: gelb,  C: 7},
{A: r, R1.B: gruen, R2.B: gelb,  C: 7},
{A: r, R1.B: gruen, R2.B: gruen, C: 8},
{A: r, R1.B: gruen, R2.B: gelb,  C: 7}}
```

2.3 Aufgabe 3

1. Die Anfragen entsprechen sich liefern jedoch nicht das gleiche Ergebniss, da der SQL-Ausdruck Duplikate zulaesst, waehrend bei der relationalen Abfrage die Duplikate entfernt werden.

2. 1. Die SQL-Anfrage liefert die **Bezeichnung** der Modelle, die nach Flughafen 'CDG' fliegen/geflogen haben.
   2. Der relationale Ausdruck liefert die Sitzplatzkapazitaeten der selben Modelle aus der SQL-Anfrage.

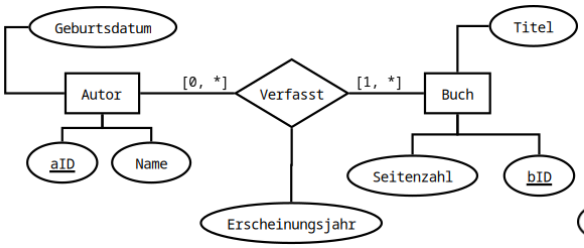
   Somit sind die Ausdrucke nicht Aequivalent und sie entpsrechen sich nicht.

3. 1. Die erste SQL-Anrage gibt die ID's der Co-Pilote und die Bezeichnungen der Modelle aus, dafuer sie zugelassen sind.
   2. Die zweite SQL-Anfrage gibt genau das gleiche Ergebniss wie die erste Anfrage. Man beachte, dass natural join in SQL immer von einem Kreuzprodukt und Selektionsoperationen simuliert werden kann.

   Somit sind die beiden Anfragen Aequivalent

2.4 Aufgabe 4

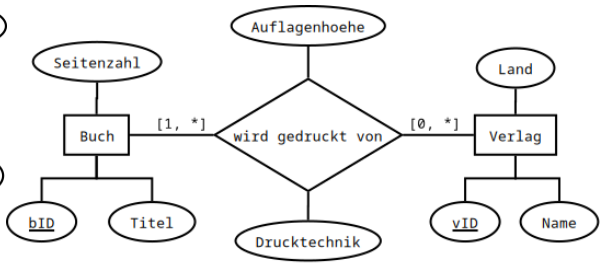
See the diagrams



Author kann beliebig viele Buecher schreiben.
(Moeglicherweise hat ein Autor kein Buch geschrieben)

Ein Buch muss mindestens von einem Author verfasst werden.
Ein buch kann von mehrerenAuthoren mitverfasst werden.

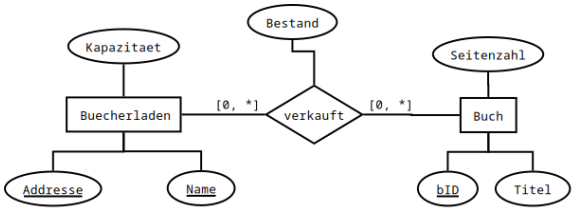
Figure 2.1: Zeile 1



Ein Verlag kann beliebig viel Buecher drucken.
Ein Verlag kann noch keine Buecher gedruckt haben - z.B ein neu gegruendeter Verlag
Ein Verlag hat einen Name und seinen Sitz in einem Land.

Ein Buch wird mindestens von einem Verlag gedruckt,
kann aber von mehreren Verlage gedruckt werden - z.B die Bibel.

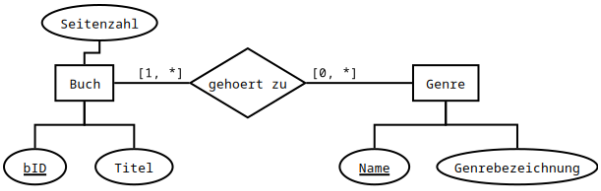
Figure 2.2: Zeile 2



Ein Buecherladen hat einen beliebig grossen Katalog von Buecher, die im Laden verkauft werden.
Der Katalog kann leer sein.
Ein Buecherladen wird Anhand einer Kombination seiner Addresse und seines Names unterschieden.

Ein Buch kann in beliebig vielen oder in keinen Laden verkauft werden.

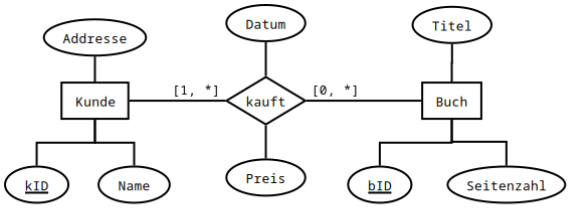
Figure 2.3: Zeile 3



Ein Buch gehoert zu mindestens einer Genre.

Es kann beliebig viele Bucher zu einer Genre geben.

Figure 2.4: Zeile 4



Ein Kunde kann beliebig viele Buecher kaufen, muss aber zumindest ein Buch gekauft haben, um in Datenbank als 'Kunde' eingetragen zu werden.

Ein Buch kann von beliebig vielen Kunden gekauft werden.

Figure 2.5: Zeile 5

2.5 Feedback

```
Zur Aufgabe 1.

1. Richtig, wenn auch > statt >= gemeint

2. Richtig 3. Sollte passen 4. Richtig

Zur Aufgabe 2:

1. Richtig 2. Ergebnis: Wieso B: Blau wenn ihr nach rot selektiert => -
↪  0.25 P.

3.-7. Richtig

Zur Aufgabe 3:

1. Richtig unter der Annahme, dass Tabelle mehr als die abgedruckten
↪  Beispieldaten enthält (Stichwort Distinct)

2. Ebenfalls richtig

3. Dito

Zur Aufgab 4:

1. Richtig 2. Verlage sollen laut ML bitte mindestens ein Buch verlegen
↪  => - 0.25P. 3. Laut ML bitte [1,*] => - 0.25 P.

4. Richtig 5. Ebenfalls
```

3 Blatt 3

3.1 Aufgabe 1

ER-Schema: Stadt

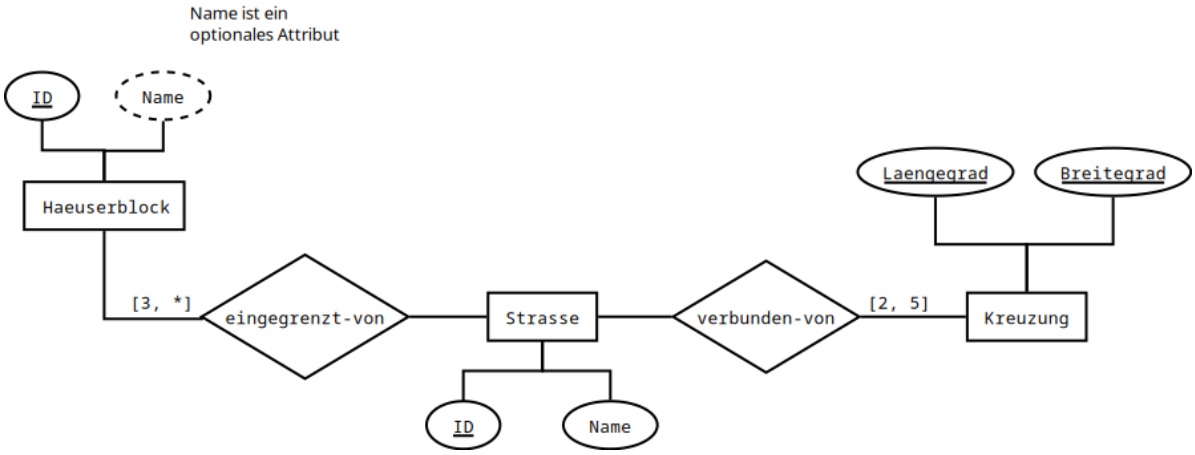


Figure 3.1: ER-Schema: Stadt

3.2 Aufgabe 2

1. ER-Schema Figure 3.2

2. *Integriteatsbedingungen*

    1. i.A. koennen die Wertebereiche der Attribute im ER-Diagram nicht spezifiert werden, z.B. wie
        1. Erscheinungsjahr eines Films darf nicht in der Zukunft liegen oder ein sehr altes Datum wie 1776 sein.
        2. Gage eines Regissuers muss $> 30{,}000$ € sein
        3. Globale Bedingungen wie z.B. **Gesamtgehalt** aus mehreren Filmen darf nie ueber 1000000 € sein koennen auch nicht spezifiziert werden.

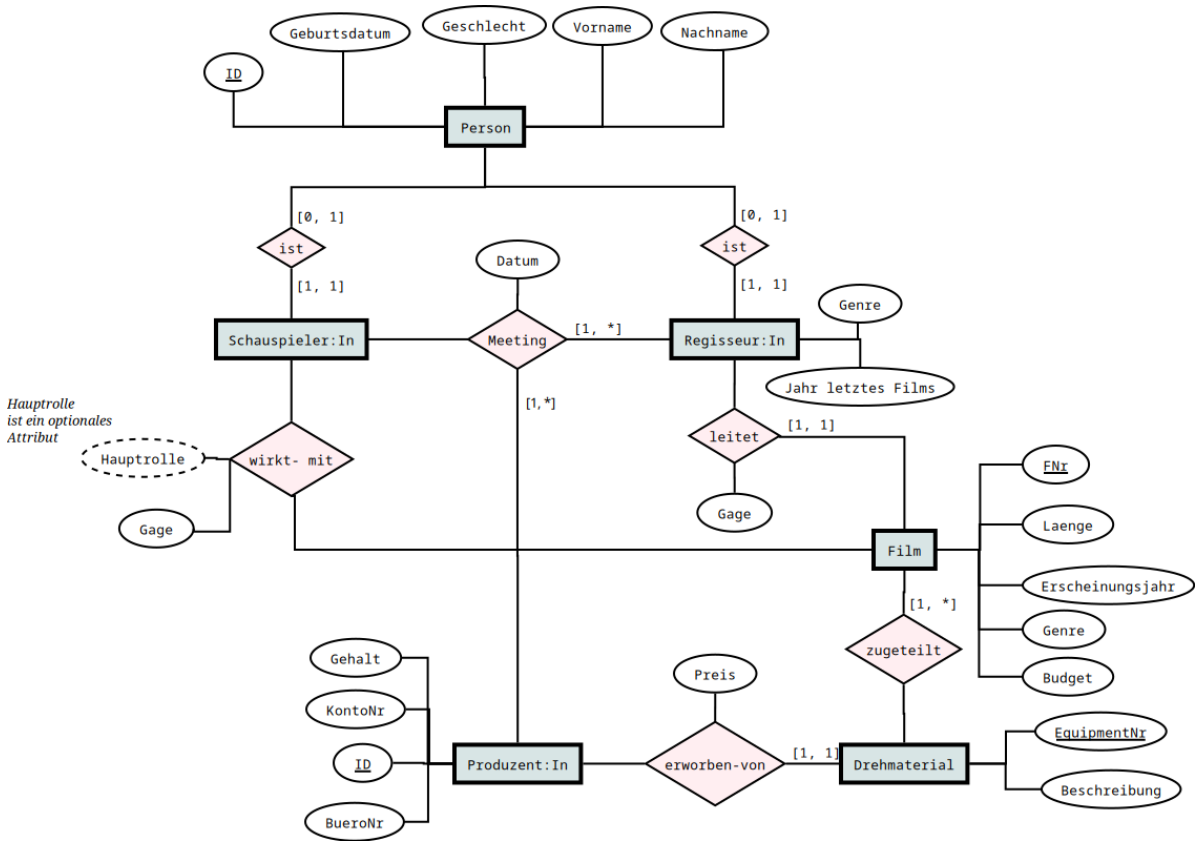
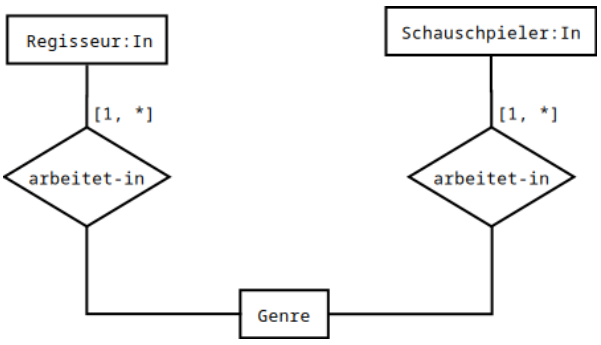


Figure 3.2: ER-Schema: NetMovie DB

2. In der Spezifikation heisst es, dass in jedem Film genau zwei Hauptrollen gibt. In unserem ER-Schema haben wir Hauptrolle als eine optionale Attribute des Beziehungstyps “wirkt-mit” modelliert. Diese Kardinalitaet kann somit nicht in unsrem ER-Schema bestimmt werden.

3. *Alternative Modellierungen*

   a) Gage als Attribute der Entitaet **Film** modellieren.
   b) Eine neue Entitaet “**Genre**” einfuehren, und “arbeitet-in” Beziehungen zwischen Regisseur-Genre, und zwischen Schauspieler-Genre modellieren:



4 Blatt 4

4.1 Aufgabe 1

- Addresse(Ad_ID, PLZ, Stadt, Strasse, Hausnr)
- MusikerIn(M_ID, Name, Geb_Datum, Ad_ID → Addresse)
- Instrument(Name, Stimmung)
- spielt(M_ID→MusikerIn, (Name, Stimmung)→Instrument, bevorzugt)
- Musikstueck(MS_ID, Titel, Laenge, M_ID→MusikerIn)
- Album(A_ID, Titel, Release_Datum, Preis, Tracks, M_ID→MusikerIn)
- erscheint(M_ID→Musikstueck, A_ID→Album, TrackNr)
- spielt_mit(M_ID→MusikerIn, MS_ID→Musikstueck)

4.2 Aufgabe 2

1. *Relationales Schema*

- Personal(Pers_ID, GebDat, Name, Vorname)
- MitarbeiterIn(Pers_ID→Personal, Bonus)
- KundIn(KundenID→Personal, Branche)
- ManagerIn(Pers_ID→MitarbeiterIn, Sektion)
- ProgrammiererIn(Pers_ID→MitarbeiterIn, Abschluss)
- Programmiersprache(ProgSP)
- kann(ProgSP→Programmiersprache, Pers_ID→ProgrammiererIn, level)

2. *Weitere Methoden fuer is-a*:

i)
- **Vorteil**: Vermeidung der Redundanz und moeglichen Inkonsitenzenen, die dadurch enstehen koennen.
- **Nachteil**: Erhoehter Rechenaufwand durch Zugriff auf Attribute der Oberentitaet nur mit Join.

ii)
- **Vorteil**: Vermindertee Rechenaufwand durch direkten Zugriff auf Attribute ueber Tupel einziger Relation.
- **Nachteil**: Redundante Speicherung der gleichen Informationen.

iii)
- **Vorteil**: Vermindernde Komplexitaet des Datenbanks durch kleinere Anzahl von Relationen (eine Relation statt zwei oder drei)
- **Nachteil**: Moegliche Inkonsitenzen durch den vielen Nullwerten, die von dem Nutzer bei Insertoperationen explizit als null gesetzt werden muessen.

5 Blatt 5

5.1 Aufgabe 1

1.

```
select real_name, created_at
from twitter_user tu
where typ = 'lobby' and
date(created_at) <  timestamp '2009-06-30'
order by created_at
```

Table 5.1: A 1.1

real_name	created_at
Sascha Lobo	2007-05-08 21:10:26
netzpolitik	2007-10-24 14:34:50
Ulrich Müller	2009-01-07 14:50:51
Mehr Demokratie e.V.	2009-04-02 19:36:30
CCC Updates	2009-04-16 14:04:59
abgeordnetenwatch.de	2009-04-25 04:14:23
LobbyControl	2009-05-07 14:48:55

2.

```
select twitter_name, like_count
from twitter_user tu , tweet t
where tu.id = t.author_id
and t.like_count between 22000 and 25000;
```

Table 5.2: A 1.2

twitter_name	like_count
MAStrackZi	22713
n_roettgen	22974
SWagenknecht	24656
n_roettgen	24329

3.

```
select distinct h.txt
from
    twitter_user tu ,
    tweet t ,
    hashtag_posting hp ,
    hashtag h
where tu.id = t.author_id  and
    t.id = hp.tweet_id and
    hp.hashtag_id = h.id and
    tu.real_name = 'LobbyControl' and
    t.created_at between '2023-01-01' and '2023-01-15'
order by h.txt
```

Table 5.3: A 1.3

txt
ampel
autogipfel
autolobby
bundestag
eu
exxon
exxonknew
korruptionsskandal
lindner
lobbyregister

4.

```
select *
from hashtag h
where h.id in (
    select hp1.hashtag_id
    from hashtag_posting hp1, hashtag_posting hp2
    where hp1.tweet_id = hp2.tweet_id and
    hp1.hashtag_id = hp2.hashtag_id and not
    hp1.pos_start = hp2.pos_start
)
```

Anzahl der Ergebnisse:

Table 5.4: A 1.4 Anzahl der Ergebnisse

count
437

5.

```
select real_name, follower_count
from twitter_user
where created_at <= '2010-01-01'
and follower_count >= all (
    select follower_count
    from twitter_user
    where created_at <= '2010-01-01'
)
```

Table 5.5: A 1.5

real_name	follower_count
Sascha Lobo	761419

5.2 Aufgabe 2

1.

```
select txt
from tweet
where txt ilike '%openai%'
and retweet_count >= 20
```

Table 5.6: A 2.1

txt
RT @DrScheuch: Die meisten Aerosolforscher haben damals schon sehr starke Zweifel am Sinn von Ausgangssperren geäußert, (#openairstattausgangssperre) wurden aber nicht gehört. @Karl_Lauterbach, die NoCovid Modellierer und @janoschdahmen waren die lautesten Befürworter. https://t.co/bL7QMaoyKP
RT @_SilkeHahn: Guten Morgen: #OpenAI ist doch schon lange #ClosedAI. Seit der ersten Milliarde durch Microsoft 2019 lassen sie sich nicht mehr in die Karten schauen. @netzpolitik_org legt gekonnt den Finger in offene Wunden. Der “Technical Report” schweigt sich auf 98 https://t.co/ix1EYCAOKG… https://t.co/hAW5rbRUtH

2.

```
select txt, char_length(txt) as Laenge
from named_entity
where char_length(txt) >= all (
    select char_length(txt)
    from named_entity
)
```

Table 5.7: A 2.2

txt	laenge
Arbeitsgemeinschaft Sozialdemokratischer Frauen	47

3.

```
select txt
from named_entity
where char_length(txt) >= 4
and txt like reverse(txt)
```

Table 5.8: A 2.3

txt
DAAD
GASAG
CIMIC
ABBA

5.3 Aufgabe 3

1.

```
select *
from named_entity ne
where not exists (
    select *
    from named_entity_posting nep
    where nep.named_entity_id = ne.id
)
```

Anzahl der Ergebnisse:

Table 5.9: A 3.1 Anzahl der Ergebnisse

count
621

2.

```
select txt
from tweet t
where exists (
    select *
    from hashtag_posting hp, hashtag h
    where hp.hashtag_id = h.id
    and t.id = hp.tweet_id
    and h.txt = 'klima'
) and exists (
    select *
```

```
    from named_entity_posting nep, named_entity ne
    where nep.named_entity_id = ne.id
    and t.id = nep.tweet_id
    and ne.txt = 'Berlin'
)
```

Table 5.10: A 3.2

txt
Erinnerung: Bis 31.01.2023 um 23:59 (Europe/Berlin) könnt ihr noch Themenvorschläge zur #nr23 machen. Z.B. zu #Klima-,#Sozial-,#Sport-,#Medizin-, #Lokal-, #Nonprofit- #Datenjournalismus, #crossborder #Diversität #Pressefreiheit etc. Call for Papers: https://t.co/zyyNCsFC0y https://t.co/aDfQOzRG1W
RT @DieLinke_HH: Das #Klima retten, nicht den #Kapitalismus! Heute haben wir mit 12.000 anderen in #Hamburg beim #Klimastreik protestiert.
Klare Ansage an Rot/Grün in Hamburg und die Ampel in Berlin: Der Stillstand bei der Klimapolitik muss aufhören! https://t.co/jbfQrCcQuh | |So sieht die #Verkehrswende von @spdhh und @GRUENE_Hamburg aus
Frei nach Fairy Ultra: Während in Berlin schon das #9EuroTicket anläuft, werden in #Hamburg noch die Preise erhöht
Soziale #Klimapolitik geht anders!
@9euroforever #Klima @LinksfraktionHH https://t.co/Fi0eC8km1M |

3.

```
select tu.twitter_name , tu.real_name
from twitter_user tu
where exists (
    select *
    from tweet t , conversation c
    where t.author_id = tu.id and
    t.id = c.id and
    array_length(c.tweets, 1) >= 70 and
    t.created_at >= '2023-02-15'
)
```

Table 5.11: A 3.3

twitter_name	real_name
salomon_alex	Alexander Salomon
HendrikWuest	Hendrik Wüst

5.4 Aufgabe 4

1.

```
select ne.id, ne.txt, count(*) as Anzahl
from named_entity ne , named_entity_posting nep
where ne.id = nep.named_entity_id
group by ne.id
order by count(*) desc
```

Anzahl der Ergebnisse:

Table 5.12: A 4.1 Anzahl der Ergebnisse

count
15744

2.

```
select tu.real_name, count(*) as anzahl
from twitter_user tu , tweet t
where tu.id = t.author_id and
tu.typ = 'politician' and
t.created_at > '2022-01-01' and
tu.tweet_count > 2000
group by tu.id
order by anzahl desc
```

Anzahl der Ergebnisse:

Table 5.13: A 4.2 Anzahl der Ergebnisse

anzahl_der_ergebnisse
913

3.

```
with erg(id, anzahl) as (
    select nep.tweet_id, count(*) anzahl
    from named_entity_posting nep, tweet t
    where nep.tweet_id = t.id
    group by tweet_id
    having count(*) >= all (
        select count(*)
        from named_entity_posting nep
        group by tweet_id
    )
)
select created_at, txt
from tweet
where id in (
  select id
  from erg
)
```

Table 5.14: A 4.3

created_at	txt
2023-01-06 07:50:05	Adrian, Alexander, Ariturel, Björn, Christian, Christian, Christian, Christopher, Cornelia, Danny, Dirk, Frank, Heiko, Johannes, Kai, Katharina, Kurt, Maik, Martin, Michael, Oliver, Robbin, Roman, Sandra, Scott, Stefanie, Stefan, Stephan, Stephan, Sven.

Verdächtig? @cduberlin |

4.

```
select date(created_at), count(*)
from tweet
group by date(created_at)
```

```
having date(created_at) > '2022-12-31'
and count(*) >= all (
    select count(*)
    from tweet
    group by date(created_at)
    having date(created_at) > '2022-12-31'
)
```

Table 5.15: A 4.4

date	count
2023-01-25	3568

6 Blatt 6

note: html Version der Abgabe (fuer die leichtere Kopierung der Code Blocks)

6.1 Aufgabe 1

1.

```
create table taxonomy(
    id int,
    name varchar,
    primary key(id),
    parent int,
    foreign key (parent) references taxonomy(id)
);

insert into taxonomy
values
    (0, 'animals', null),
    (2, 'chordate', 0),
    (1, 'athropod', 0),
    (6, 'mammals', 2),
    (5, 'reptiles', 2),
    (3, 'insects', 1),
    (4, 'crustacean', 1),
    (9, 'carnivora', 6),
    (8, 'scaled reptiles', 5),
    (7, 'crocodiles', 5),
    (10, 'cats', 9),
    (11, 'pan-serpentes', 8);
```

2.

```
select name
from taxonomy
where parent = 2
  union
select name
from taxonomy t1
where exists (
    select name
    from taxonomy t2
    where t1.parent = t2.id
    and t2.parent = 2
)
```

Table 6.1: A 1.2

name
carnivora
reptiles
crocodiles
scaled reptiles
mammals

3.

```
with recursive subCatOfChordate(id, name) as (
    select id, name
    from taxonomy t
    where t.parent = 2
        union
    select t.id, t.name
    from taxonomy t, subCatOfChordate s
    where t.parent = s.id
)
select id
from subcatofchordate
```

Table 6.2: A 1.2

id
6
5
9
8
7
10
11

6.2 Aufgabe 2

1.

- **rel**: $\pi_{\texttt{real\_name, tweet\_count, follower\_count}}\Big($
  $\sigma_{\texttt{created\_at > 01.01.2019, follower\_count > 8000, tweet\_count > 1000, like\_count > 1000}}$
  $(\beta_{\texttt{author\_id}\leftarrow\texttt{id}}(\texttt{twitter\_user}) \bowtie \beta_{\texttt{ca}\leftarrow\texttt{created\_at}}(\texttt{tweet}))\Big)$
- **sql**:

```sql
select tu.real_name, tu.tweet_count, tu.follower_count
from twitter_user tu
where tu.created_at > '2019-01-01'
and tu.follower_count > 8000
and tu.tweet_count > 1000
and exists (
    select *
    from tweet t
    where t.author_id = tu.id
    and t.like_count > 1000
)
```

Table 6.3: A 2.1

real_name	tweet_count	follower_count
Rote Socke Türk-Nachbaur	16692	21283
Ursula von der Leyen	3675	1295550
Verteidigungsministerium	8923	120387

real_name	tweet_count	follower_count
Carmen Wegge	1029	9355

2.

- **rel**: $\pi_{\texttt{txt, author\_id, created\_at}}(\sigma_{\texttt{like\_count > 1000}}(\texttt{tweet}) - \pi_{\texttt{txt, author\_id, created\_at}}\Big(\sigma_{\texttt{created\_at > ca}}(\sigma_{\texttt{like\_count > 1000}}(\texttt{tweet}) \times \beta_{\texttt{ca} \leftarrow \texttt{created\_at}, \texttt{ai} \leftarrow \texttt{author\_id}, \texttt{t} \leftarrow \texttt{txt}}(\sigma_{\texttt{like\_count > 1000}}(\texttt{tweet})))\Big)$
- **sql**:

```
select t.txt, t.author_id, t.created_at
from tweet t
where t.like_count >= 1000
and t.created_at <= all (
    select created_at
    from tweet
    where like_count >= 1000
)
```

Table 6.4: A 2.3

txt	author_id	created_at
Die Leute haben heute aus Trotz geböllert, oder? Das nahm ja kein Ende. Genial! Danke.	814970546366620237	2023-01-01 00:17:32

3.

- **rel**: $\pi_{\texttt{hi, hashtag\_id}}\Big(\sigma_{\texttt{ti < tweet\_id}}\Big(\beta_{\texttt{ti} \leftarrow \texttt{tweet\_id}}(\sigma_{\texttt{hi < hashtag\_posting}}(\texttt{hashtag\_posting} \bowtie \beta_{\texttt{hi} \leftarrow \texttt{hashtag\_id}}(\texttt{hashtag\_posting}))) \bowtie \sigma_{\texttt{hi < hashtag\_posting}}(\texttt{hashtag\_posting} \bowtie \beta_{\texttt{hi} \leftarrow \texttt{hashtag\_id}}(\texttt{hashtag\_posting}))\Big)\Big)$
- **sql**:

```
with hashtagpairs as (
    select
        hp1.hashtag_id h1_id,
```

```
        h1.txt h1_txt,
        hp2.hashtag_id  h2_id,
        h2.txt h2_txt,
        hp1.tweet_id tid
    from hashtag_posting hp1, hashtag_posting hp2, hashtag h1, hashtag
     ↪  h2
    where hp1.tweet_id  = hp2.tweet_id
    and h1.id = hp1.hashtag_id
    and h2.id = hp2.hashtag_id
    and hp1.hashtag_id  < hp2.hashtag_id
)
select hpr1.h1_txt, hpr1.h2_txt
from hashtagpairs hpr1
where exists (
    select *
    from hashtagpairs hpr2
    where hpr1.h1_id = hpr2.h1_id
    and hpr1.h2_id = hpr2.h2_id
    and hpr1.tid < hpr2.tid
)
```

Table 6.5: A 3.3 Anzahl der Ergebnisse

count
178346

6.3 Aufgabe 3

```
select tu.real_name, regexp_count(t.txt, '\m[[:upper:]]{2,}\M') as cnt,
 ↪  t.txt
from tweet t, twitter_user tu
where tu.typ = 'politician'
and t.author_id  = tu.id
and regexp_count(t.txt, '\m[[:upper:]]{2,}\M') >= all (
    select regexp_count(txt, '\m[[:upper:]]{2,}\M')
    from tweet
)
```

Table 6.6: A 3

real_name	cnt	txt
Udo Hemmelgarn, MdB	41	RT @Georg_Pazderski: BITTE BITTE BITTE BITTE

BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE
BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE
BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE BITTE
https://t.co/snBGvZGABI |

7 Blatt 7

note: html Version der Abgabe fuer leichtere Kopierung der Codeblocks.

7.1 Aufgabe 1

1.

```
select tu.real_name , tu.twitter_name
from twitter_user tu
where tu.typ = 'politician'
and exists (
    select *
    from twitter_user tu2
    where tu2.twitter_name <> tu.twitter_name
    and tu2.real_name = tu.real_name
)
```

Table 7.1: A1.1

real_name	twitter_name
Martin Hagen	_MartinHagen
Martin Hagen	MartinHagenHB

2.

```
select
    tu.real_name real_name,
    tu.twitter_name twitter_name,
    tu.follower_count follower_count,
    tu.tweet_count tweet_count,
    array_length(c.tweets, 1) conversation_length
from tweet t, conversation c, twitter_user tu
where t.id = c.id
```

```
and tu.id = t.author_id
and array_length(c.tweets, 1) >= all (
    select array_length(c2.tweets, 1)
    from conversation c2
)
```

Table 7.2: A1.2

real_name	twitter_name	follower_count	tweet_count	conversation_length
Tom Schreiber	TomSchreiberMdA	5328	48186	86
Christian Lindner	c_lindner	653690	18882	86

3.

```
select ne.txt, ne.id, count(*)
from
    tweet t,
    hashtag_posting hp,
    hashtag h,
    named_entity ne,
    named_entity_posting nep
where t.id = hp.tweet_id
and hp.hashtag_id = h.id
and h.txt ilike 'energie'
and nep.tweet_id = t.id
and nep.named_entity_id = ne.id
group by ne.txt, ne.id
having count(*) >= 4
order by count(*) desc
```

Table 7.3: A1.3

txt	id	count
Deutschland	31	18
Bayern	240	7
Thüringen	526	6
Anschluss	1741	5
Berlin	2	4
Bernhard Stengele	11253	4
CDU	65	4

txt	id	count
Europa	217	4
Bund	655	4

4.

```
select
    ne.id entity_id,
    ne.txt entity_txt,
    date(t.created_at) datum,
    count(*) anzahl
from
    tweet t ,
    named_entity_posting nep ,
    named_entity ne
where t.id = nep.tweet_id
and ne.id =nep.named_entity_id
group by ne.id, ne.txt, date(t.created_at)
order by count(*) desc
limit 5
```

Table 7.4: A1.4

entity_id	entity_txt	datum	anzahl
6	Ukraine	2023-02-24	761
2	Berlin	2023-02-12	427
28	Bundestag	2023-03-17	286
2	Berlin	2023-02-10	283
1425	CSU	2023-03-17	259

7.2 Aufgabe 2

1.

- **umg**: Was sind die echten Namen von allen Twitter Benutzern, die Lobbyisten sind, die einen Tweet mit ueber 2000 Likes veroeffentlicht haben, der die EU oder die USA erwaehnt?
- **tup**:

$$\{\langle \texttt{tu.real\_name} \rangle \mid \texttt{tu} \in \texttt{twitter\_user} \wedge \texttt{tu.typ='lobby'} \wedge \exists \texttt{t} \exists \texttt{ne} \exists \texttt{nep} ($$
$$\texttt{t} \in \texttt{tweet} \wedge$$
$$\texttt{ne} \in \texttt{named\_entity} \wedge$$
$$\texttt{nep} \in \texttt{named\_entity\_posting} \wedge$$
$$\texttt{t.id = nep.tweet\_id} \wedge$$
$$\texttt{ne.id = nep.named\_entity\_id} \wedge$$
$$\texttt{t.like\_count > 2000} \wedge$$
$$\texttt{t.author\_id = tu.id} \wedge$$
$$(\texttt{ne.txt = 'EU'} \vee \texttt{ne.txt = 'USA'}))\}$$

2.

- **umg**: Was sind die IDs aller Authoren, die zwar einen Tweet mit dem Hashtag “openai” verfasst haben aber keinen mit dem Hashtag “chatgpt”.
- **tup**:

$$\{\langle \texttt{t.author\_id} \rangle \mid \texttt{t} \in \texttt{tweet} \wedge \exists \texttt{h} \exists \texttt{hp} ($$
$$\texttt{h} \in \texttt{hashtag} \wedge$$
$$\texttt{hp} \in \texttt{hashtag\_posting} \wedge$$
$$\texttt{h.id = hp.hashtag\_id} \wedge$$
$$\texttt{hp.tweet\_id = t.id} \wedge$$
$$\texttt{h.txt = 'openai'}) \wedge$$
$$\neg \exists \texttt{h} \exists \texttt{hp} ($$
$$\texttt{h} \in \texttt{hashtag} \wedge$$
$$\texttt{hp} \in \texttt{hashtag\_posting} \wedge$$
$$\texttt{h.id = hp.hashtag\_id} \wedge$$
$$\texttt{hp.tweet\_id = t.id} \wedge$$
$$\texttt{h.txt = 'chatgpt'})\}$$

8 Blatt 8

8.1 Aufgabe 1

1. Satzlaenge `twitter_user`

Attribute	Typ	Satzlaenge
`id`	`bigint`	8 byte
`follower_count`	`integer`	4 byte
`tweet_count`	`integer`	4 byte
`typ`	`char(11)`/`char(5)` “politician”/“lobby”	12 byte (1 byte Overhead)
`created_at`	`timestamp`	8 byte
`twitter_name`	`text`	12 byte
`real_name`	`text`	18 byte
—	—	—
$\sum$		**54 byte**

2. Speicherplatz der Header

   - Jede **Page** hat 24 Byte Header
   - Jedes **Tupel** hat 23 Byte Header

   D.h. jedes Tupel hat 54 Byte Nutzdaten + 23 Byte Header = 77 Byte.

3. Groesse der Bloecke im PostgreSQL:

```
select current_setting('block_size');
```

Table 8.2: block size

current_setting
8192

```
select count(*)
from twitter_user
```

Table 8.3: Anzahl der Tupel in der Relation twitter user

count
1825

Anzahl der Tupel pro Seite ca.:

```
round(8192 / 77)
```

```
[1] 106
```

Somit ist die Anzahl der Seiten ungefaehr:

```
round(1825 / 106)
```

```
[1] 17
```

4. Anzahl der Seiten der Relation 'twitter_user':

```
select relname, relpages
from pg_class
where relname = 'twitter_user'
```

Table 8.4: Anzahl der Seiten fuer twitter user

relname	relpages
twitter_user	22

Also in Wirklichkeit werden 22 Seiten gebraucht statt 17 Seiten. D.h. mehr Speicher. Die Gruende dieser Abweichung sind u.a. mehr Speicher fuer:

- Pageheader
- Zeiger auf die Tupel
- Special-/Free Space in Pages
- Optionalen Zusatzelementen wie Null Bitmap in den Tuples

8.2 Aufgabe 2

1. B-Baum 1.1 Figure 8.1

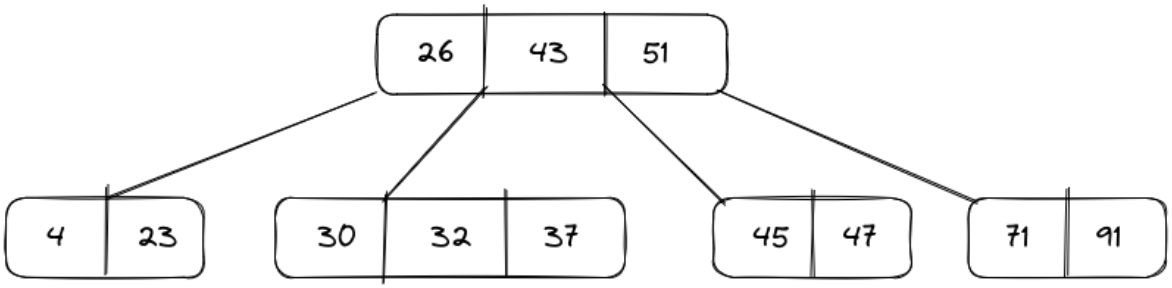


Figure 8.1: B-Baum: A2.1

2. B-Baum 1.2 Figure 8.2

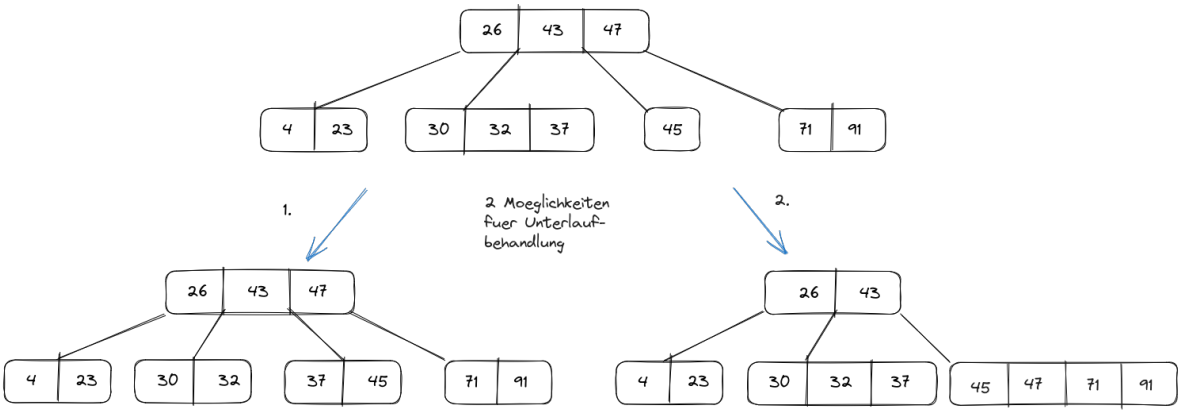


Figure 8.2: B-Baum: A2.1

8.3 Aufgabe 3

1. B+-Baum 3.1 Figure 8.3

2. Die Elemente in der sortierten Reihenfolge in den B+ Baum einfuegen, aber nicht durch ein normales Insert, sondern direkt an das Blatt ganz rechts einfuegen. Dadurch spart man sich die look-up Operation $\mathcal{O}(\log_m(n))$ des insert, die ein groeseres Element als die bisherigen sowieso ganz rechts in Baum ablegen wuerde.

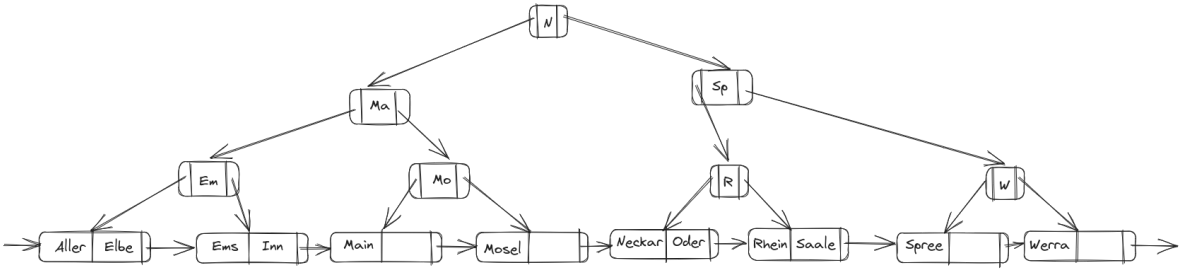


Figure 8.3: B-Baum: A2.1

Part II

Notes