Einführung in Datenbanken

Übung 16: Klausur-Statistik, ER-M., Logischer Entwurf, BCNF

Prof. Dr. Stefan Brass
PD Dr. Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2024/25

http://www.informatik.uni-halle.de/~brass/db24/

Inhalt

Klausur im WS 2023/24 (1)

- Statistik der letzten Klausur (WS 2023/24, 07.03.2024):

Note	Anzahl	Prozent	Kumulativ	2020–2023
1.0	6	10%	10%	16.8%
1.3	7	11%	21%	22.5%
1.7	7	11%	33%	26.8%
2.0	2	3%	36%	35.8%
2.3	5	8%	44%	49.3%
2.7	5	8%	52%	55.8%
3.0	4	7%	59%	65.8%
3.3	8	13%	72%	71.5%
3.7	2	3%	75%	76.4%
4.0	1	2%	77%	78.2%
5.0	14	23%	100%	100.0%

Zusätzlich haben drei Studierende eine 5.0 bekommen wegen „Nicht erschienen“.

Klausur im WS 2023/24 (2)

- Statistik der letzten Klausur (WS 2023/24), Forts.:
  - Es haben 58 Studierende an der Klausur teilgenommen.

    Inklusive nicht erschienen/krank/mündl.: 64. Studienleistung: 69.
  - In StudIP eingetragen waren 123 Studierende.
  - Die Prüfung am 2. Termin (08.07.2024) haben 9 Studierende genutzt, davon sind 5 durchgefallen.
- Maximal erreichte Punktzahl: 50 (als 100% zählten 45).
  - 1. Quartil: 37.0 Punkte (82%, Note: 1.7)
  - 2. Quartil (Median): 28.5 Punkte (63%, Note: knapp 2.7)
  - 3. Quartil: 22.0 Punkte (49%, Note: knapp 4.0)

    Zwei Punkte wurden nachträglich zu Bonuspunkten erklärt.

    Eigentlich gab es $47 + 4$ Punkte. Durchschnitt: 29.5 Punkte (66%, 2.7).

Klausur im WS 2023/24 (3)

- Aufgaben der Klausur:

Nr.	Art	Pkt	Zeit	Abg.	Avg.%
1a	SQL (Selbstverbund)	5	10min	56	66%
1b	SQL (OR, LIKE)	5	10min	56	75%
1c	SQL (GROUP BY)	5	10min	52	43%
1d	SQL (NOT EXISTS)	5	10min	47	46%
1e	SQL (Outer Join)	5	10min	54	68%
2	Logik/Äquivalenzn	5	15min	57	65%
3	Relationale Algebra	4	5min	45	78%
4	ER-Entwurf	7	15min	55	80%
5	Logischer Entwurf	6	15min	55	53%
6	FAen, BCNF (Bonus)	+4	10min	47	49%

Pkt: Maximalpunktzahl, Abg.: Anzahl Abgaben, Avg.%: Durchschnittlich erreichte Punkte in % von Pkt (Durchschnitt nur über Abgaben).

Klausur im WS 2023/24 (4)

- SQL-Aufgabe a) (Selbstverbund):

Punkte	Anzahl	Prozent
5.0	11	20%
4.5	7	13%
4.0	5	9%
3.5	7	13%
3.0	5	9%
2.5	6	11%
2.0	7	13%
1.5	5	9%
1.0	1	2%
0.5	2	4%

Klausur im WS 2023/24 (5)

- Ergebnisse des automatischen Tests für a):

Ergebnis	Anzahl	Prozent
OK	16	29%
WRONG_RESULT	33	59%
NOT_EXECUTABLE	3	5%
NOT_PARSABLE	4	7%

„NOT_PARSABLE“ ist wohl ein Fehler bezüglich der kontextfreien Syntax.
Zu „NOT_EXECUTABLE“ gehören Fehler wie „Die Spalte existiert nicht“ oder „Die Spalte muss in der GROUP BY-Klausel gelistet werden.“ Beide Fehler können beim Testen unmöglich übersehen werden.
„WRONG_RESULT“ können gelegentlich nur kleine Abweichungen im Ergebnis-Spaltennamen sein, aber meist springt der Unterschied zum erwarteten Ergebnis (das in der Klausur gezeigt ist) doch ins Auge.

Klausur im WS 2023/24 (6)

- SQL-Aufgabe b) (`OR` und `LIKE`):

Punkte	Anzahl	Prozent
5.0	14	25%
4.5	11	20%
4.0	10	18%
3.5	3	5%
3.0	7	13%
2.5	2	4%
2.0	4	7%
1.5	4	7%
1.0	0	0%
0.5	1	2%

Klausur im WS 2023/24 (7)

- Ergebnisse des automatischen Tests für b):

Ergebnis	Anzahl	Prozent
OK	11	20%
WRONG_RESULT	42	75%
NOT_EXECUTABLE	1	2%
NOT_PARSABLE	2	4%

- Ergebnisse des automatischen Tests für c):

Ergebnis	Anzahl	Prozent
OK	6	12%
WRONG_RESULT	36	69%
NOT_EXECUTABLE	1	2%
NOT_PARSABLE	9	17%

Klausur im WS 2023/24 (8)

- SQL-Aufgabe c) (Selbstverbund, `GROUP BY`, `HAVING`, `AVG`, Prozentrechnung):

Punkte	Anzahl	Prozent
5.0	3	6%
4.5	2	4%
4.0	4	8%
3.5	4	8%
3.0	5	10%
2.5	9	17%
2.0	5	10%
1.5	4	8%
1.0	3	6%
0.5	5	10%
0.0	8	15%

Klausur im WS 2023/24 (9)

- SQL-Aufgabe d) (`NOT EXISTS`):

Punkte	Anzahl	Prozent
5.5	1	2%
5.0	4	9%
4.0	3	6%
3.5	2	4%
3.0	6	13%
2.5	13	28%
2.0	1	2%
1.5	3	6%
1.0	3	6%
0.5	8	17%
0.0	3	6%

Klausur im WS 2023/24 (10)

- Ergebnisse des automatischen Tests für d):

Ergebnis	Anzahl	Prozent
OK	2	4%
WRONG_RESULT	34	72%
NOT_EXECUTABLE	4	9%
NOT_PARSABLE	7	15%

- Ergebnisse des automatischen Tests für e):

Ergebnis	Anzahl	Prozent
OK	5	9%
WRONG_RESULT	41	76%
NOT_EXECUTABLE	3	6%
NOT_PARSABLE	5	9%

Klausur im WS 2023/24 (11)

- SQL-Aufgabe e) (Outer Join, `GROUP BY`, `COUNT`, `CASE`):

Punkte	Anzahl	Prozent
6.0	1	2%
5.5	3	6%
5.0	6	11%
4.5	4	7%
4.0	11	20%
3.5	9	17%
3.0	5	9%
2.5	2	4%
2.0	5	9%
1.5	3	6%
1.0	1	2%
0.5	2	4%
0.0	2	4%

Klausur im WS 2023/24 (12)

- Größe der Musterlösungen:

Aufgabe	Zeilen	Worte (Token)	Zeichen
a)	15	61	388
b)	8	52	314
c)	13	59	427
d)	10	37	268
e)	10	46	337

- Man sollte also schon auf etwas komplexere SQL-Anfragen vorbereitet sein.

  Alle Anfragen sind ungefähr gleich schwer (unterschiedliche Konstrukte).

- Durchschnitt über alle SQL-Anfragen: 3.0/5 Punkten.

- 38 Mal Fehler „`ORDER BY` fehlt“. Aufgaben genau lesen!

Klausur im WS 2023/24 (13)

- Ankreuzaufgaben zur Logik (Äquivalenz von Anfragen):

Aufgabe	Richtig		Falsch	
a)	41	(72%)	16	(28%)
b)	13	(23%)	44	(77%)
c)	44	(77%)	13	(23%)
d)	47	(82%)	10	(18%)
e)	39	(68%)	18	(32%)

Bei den meisten Aufgaben gab es mehr als zwei Optionen zur Auswahl, aber die Aufgaben waren „Single Choice“ (nur ein Kreuzchen pro Aufgabe). Für jedes richtige Kreuzchen gab es 1 Punkt. Es gab keine Abzüge für falsche Antworten. Man sollte also notfalls raten. Durchschnittlich wurden 0.65 Punkte pro Ankreuzaufgabe erreicht, d.h. 65% der Antworten waren richtig.

Klausur im WS 2023/24 (14)

- Aufgabe zur Relationalen Algebra (max. 4 Punkte):

Punkte	Anzahl	Prozent
4.0	11	24%
3.5	16	36%
3.0	7	16%
2.5	5	11%
2.0	2	4%
1.5	1	2%
1.0	1	2%
0.5	1	2%
0.0	1	2%

Es war eine einfache Anfrage zu schreiben, die z.B. nach dem Muster $\pi\ \sigma\ \bowtie$ gelöst werden konnte. Nur 45/58 Studierenden haben die Aufgabe bearbeitet, diese haben durchschnittlich 3.1/4 Punkte (78%) bekommen.

Klausur im WS 2023/24 (15)

- Aufgabe zum ER-Entwurf (max. 7 Punkte):

Punkte	Anzahl	Prozent
7.00	4	7%
6.75	9	16%
6.50	11	20%
6.25	4	7%
6.00	2	4%
5.75	2	4%
5.50	2	4%
5.25	2	4%

Punkte	Anzahl	Prozent
5.00	5	9%
4.75	2	4%
4.50	6	11%
4.25	1	2%
3.75	1	2%
3.25	2	4%
1.75	2	4%

Diese Aufgabe wurde mit Viertel-Punkten korrigiert, z.B. „Zwei Weak Entity Kennzeichnungen falsch: je -0.25 Punkte“.

Durchschnitt (über 55 abgegebenen Lösungen): 5.6/7 Punkte (80%).

Klausur im WS 2023/24 (16)

- Aufgabe zum Logischen Entwurf (max. 6 Punkte):

Punkte	Anzahl	Prozent
6.0	3	5%
5.5	1	2%
5.0	2	4%
4.5	2	4%
4.0	9	16%
3.5	10	18%
3.0	5	9%
2.5	8	15%
2.0	10	18%
1.5	3	5%
0.5	1	2%
0.0	1	2%

Durchschnitt (über 55 abgegebenen Lösungen): 3.2/6 Punkte (53%).

Klausur im WS 2023/24 (17)

- Bonus-Aufgabe zu FAen/BCNF (max. 4 Punkte):

Punkte	Anzahl	Prozent
4.00	4	9%
3.75	1	2%
3.50	3	6%
3.00	4	9%
2.75	1	2%
2.50	8	17%
2.00	9	19%
1.00	9	19%
0.50	1	2%
0.00	7	15%

Durchschnitt (über 47 abgegebenen Lösungen): 2.0/4 Punkte (50%).

Klausur im WS 2023/24 (18)

Hinweis:

- Natürlich könnte vielleicht auch ein neuer Aufgabentyp (z.B. `CREATE TABLE`) hinzukommen (oder einer wegfallen).
- Die Aufteilung der Punkte ist nicht garantiert.

  Z.B. könnte, selbst wenn es wieder eine Aufgabe zum ER-Entwurf gibt, diese vielleicht eine andere Punktzahl haben als 7 Punkte.
- Es ist auch nicht garantiert, dass die Bewertung wieder genau so gemacht wird.
- Ganz grundsätzliche Änderungen sind nach aktuellem Stand nicht geplant.
- Es gibt alte Klausuren auf der Webseite.

Inhalt

Präsenzaufgabe 13: Logischer Entwurf

- Übersetzen Sie folgendes ER-Diagramm in das relationale Modell:

- Geben Sie das Schema in der Kurznotation (ASCII-Version) ab, z.B.: `STUDENTEN(#SID, VORNAME, NACHNAME, EMAIL?)`

- Falls zusätzliche Bedingungen überwacht werden müssen, um die Äquivalenz zum ER-Diagramm sicherzustellen, schreiben Sie eine SQL-Anfrage, die Fehlermeldungen berechnet.

Lösung der Präsenzaufgabe (1)

- Gegebenes ER-Diagramm:

- Man legt zuerst für jeden Entity-Typ eine Tabelle an und überträgt dabei die Attribute, die Schlüssel und die Festlegung, ob Nullwerte erlaubt sind:

```
A(#B, #C, D)
E(#F, G, H?)
```

```
A(B, C, D)
E(F, G, H°)
```

Lösung der Präsenzaufgabe (2)

- Gegebenes ER-Diagramm:

- Für die eins-zu-viele Beziehung fügt man den Schlüssel der „eins“-Seite als Fremdschlüssel zur „viele“-Seite hinzu:

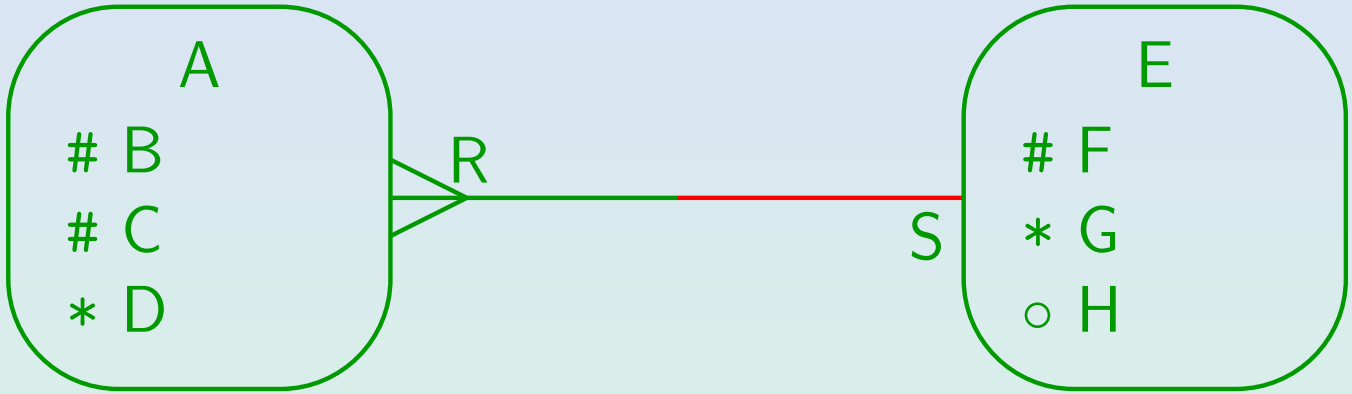
```
E(#F, G, H?)
A(#B, #C, D, F->E)
```

```
E(F, G, H°)
A(B, C, D, F→E)
```

Falls möglich, sollte man die Relationen so anordnen, dass Fremdschlüssel auf vorher definierte Relationen verweisen. Der Fremdschlüssel könnte auch anders heißen, z.B. `R` oder `E_F`. Man sollte aber auf Konsistenz achten.

Lösung der Präsenzaufgabe (3)

- Gegebenes ER-Diagramm:



- Die Übersetzung stellt sicher, dass jedes A-Entity an der Beziehung teilnimmt (der Fremdschlüssel ist nicht Null).
- Die Übersetzung stellt nicht sicher, dass jedes E-Entity auch an der Beziehung teilnimmt.

  Also von einem Fremdschlüssel-Wert referenziert wird.

Lösung der Präsenzaufgabe (4)

- Zusätzlich nötige Integritätsbedingung: „Jeder Wert für den Schlüssel `F` in der Tabelle `E` taucht auch in der Tabelle `A` als Fremdschlüssel-Wert auf.“
- SQL-Anfrage zur Berechnung von Fehlermeldungen bei Verletzung der Integritätsbedingung:

```
SELECT 'E-Entity ohne A-Entity: ' || X.F
FROM   E X
WHERE  NOT EXISTS(SELECT *
                  FROM   A Y
                  WHERE  Y.F = X.F)
```

Wenn das Beispiel nicht so abstrakt wäre, könnte man die Fehlermeldung etwas aussagekräftiger gestalten.

Lösung der Präsenzaufgabe (5)

- Alternative Lösung mit `NOT IN`:

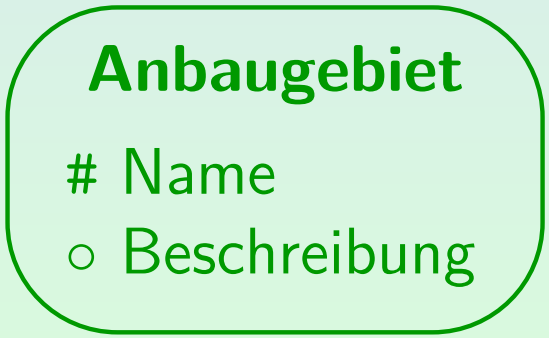
```
SELECT 'E-Entity ohne A-Entity: ' || F
FROM   E
WHERE  F NOT IN (SELECT F
                 FROM   A)
```

Wie bekannt, muss man sich bei `NOT IN`-Anfragen versichern, dass die Unteranfrage keinen Nullwert liefert. Das ist hier gegeben (der Fremdschlüssel ist `NOT NULL`). Wäre die Linie auf der `A`-Seite dagegen gestrichelt, wären Nullwerte für den Fremdschlüssel erlaubt, und man müsste sie in der Unteranfrage explizit ausschließen. Bei der Variante mit `NOT EXISTS` gibt es dieses Problem nicht.

Inhalt

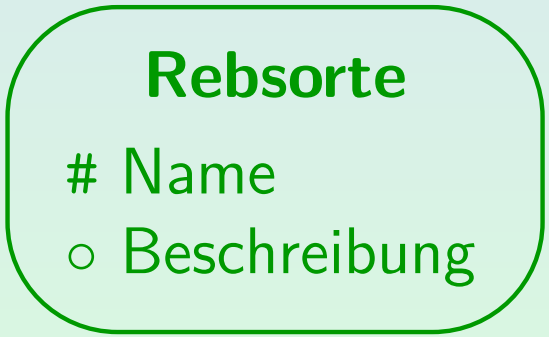
Hausaufgabe 14.1: ER-Entwurf (1)

- Es gibt Weinanbaugebiete, z.B. Saale-Unstrut, Rheinhessen, Pfalz.
- Sie sind durch ihren Namen identifiziert.
- Außerdem soll es möglich sein, optional eine Kurz-Beschreibung zu speichern.



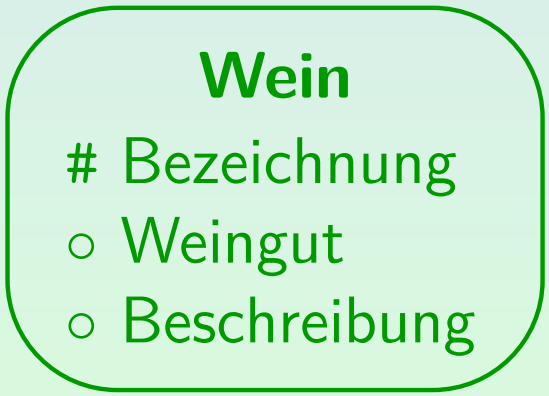
Hausaufgabe 14.1: ER-Entwurf (2)

- Rebsorten (z.B. Riesling, Spätburgunder) sind durch ihren Namen identifiziert.
- Auch hier ist eine optionale Beschreibung zu speichern.



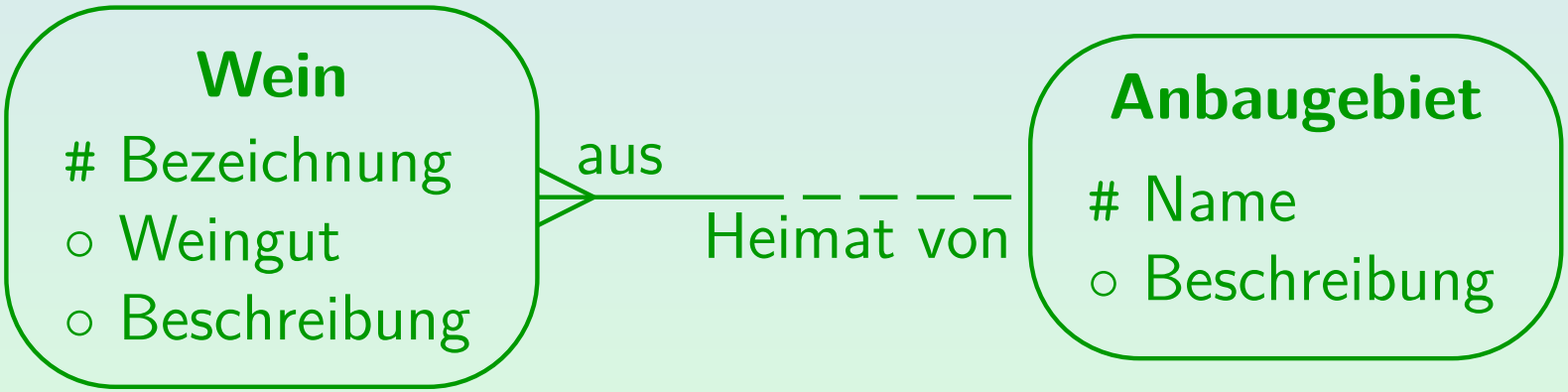
Hausaufgabe 14.1: ER-Entwurf (3)

- Weine sind durch Ihre Bezeichnung identifiziert (dies ist sozusagen der Markenname, der über mehrere Jahrgänge gleich bleibt).
- Optional können der Name des Weinguts sowie eine Beschreibung gespeichert werden.



Hausaufgabe 14.1: ER-Entwurf (4)

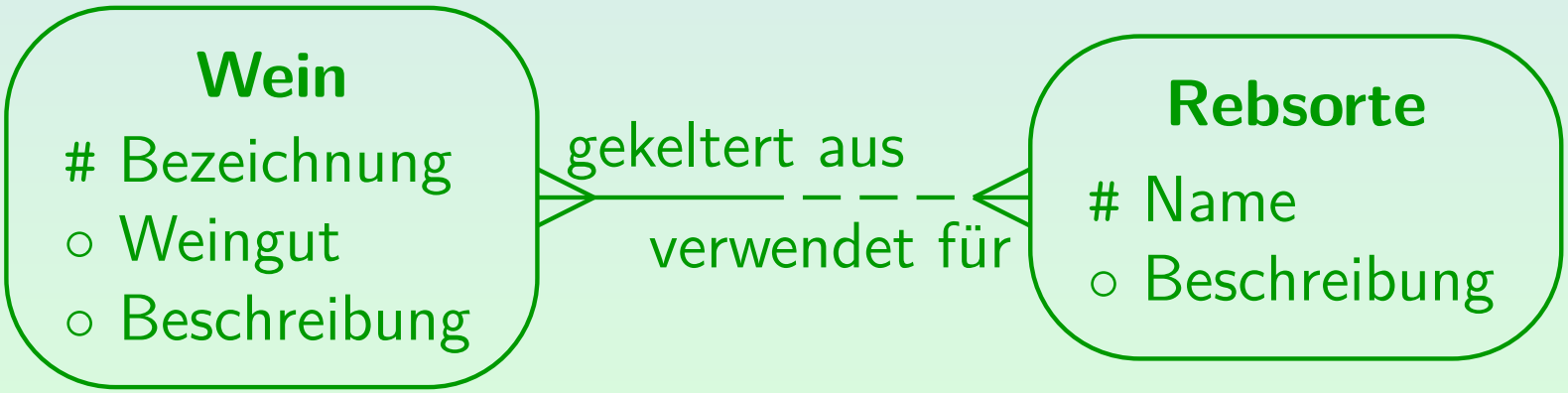
- Weine sind genau einem Weinanbaugebiet zugeordnet (minimal eins und maximal eins).
- Ein Weinanbaugebiet kann beliebig viele Weine enthalten (auch 0).



- Eins-zu-viele Beziehung: Ein Anbaugebiet, viele Weine.

Hausaufgabe 14.1: ER-Entwurf (5)

- Für einen Wein wurden ein oder mehr Rebsorten verwendet.
- Eine Rebsorte kann natürlich in mehreren Weinen enthalten sein.
- Es soll aber auch möglich sein, Rebsorten schon zu erfassen, bevor man einen Wein aus ihr hat. D.h. Rebsorten können auch zu 0 Weinen in Beziehung stehen.



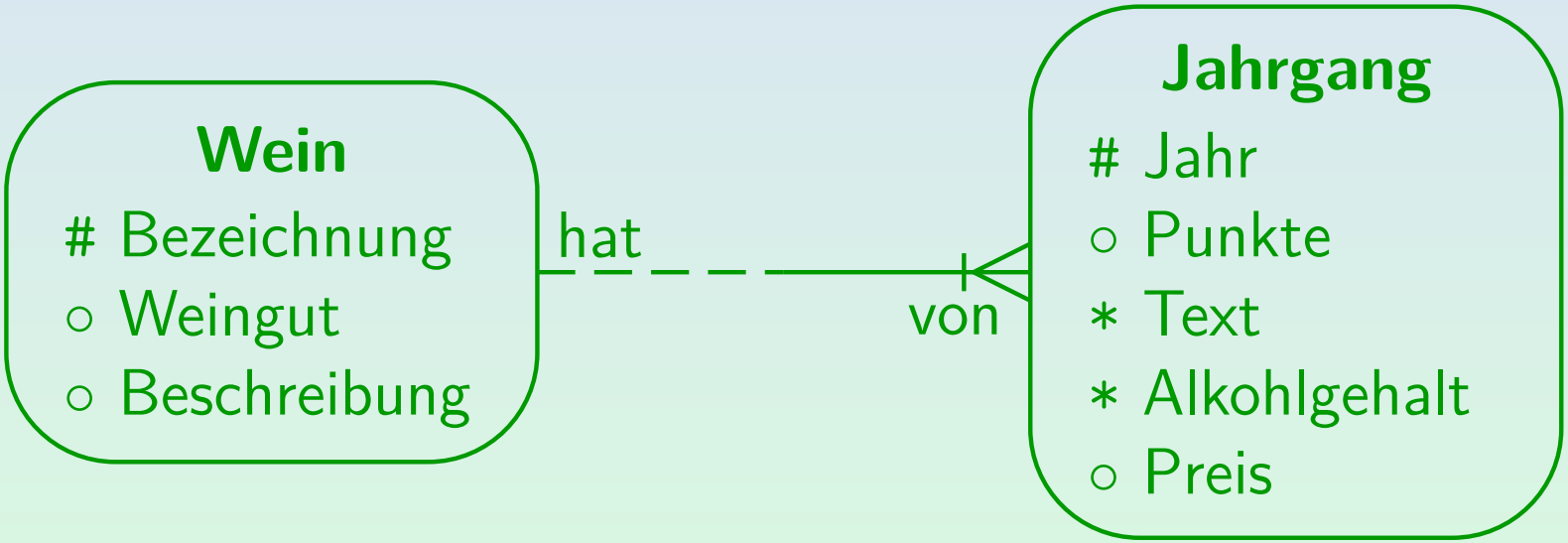
- Viele-zu-viele Beziehung.

Hausaufgabe 14.1: ER-Entwurf (6)

- Ein Jahrgang eines Weins ist durch den Wein und die Jahreszahl identifiziert (der Jahrgang ist, was konkret in Flaschen abgefüllt wird).
- Zum Jahrgang möchte der Weinliebhaber einen persönlichen Punktewert speichern (wie sehr ihm der Wein geschmeckt hat), einen kurzen Text, den Alkoholgehalt, sowie den Preis einer Flasche.
- Der Punktwert ist optional, da die Daten der Weinflasche für den Jahrgang schon erfasst werden sollen, bevor der Wein ausgedrucken wurde.
- Der Preis ist auch optional: Er kann unbekannt sein, wenn es z.B. ein Geschenk war.
- Der Text und der Alkoholgehalt sind nicht optional.

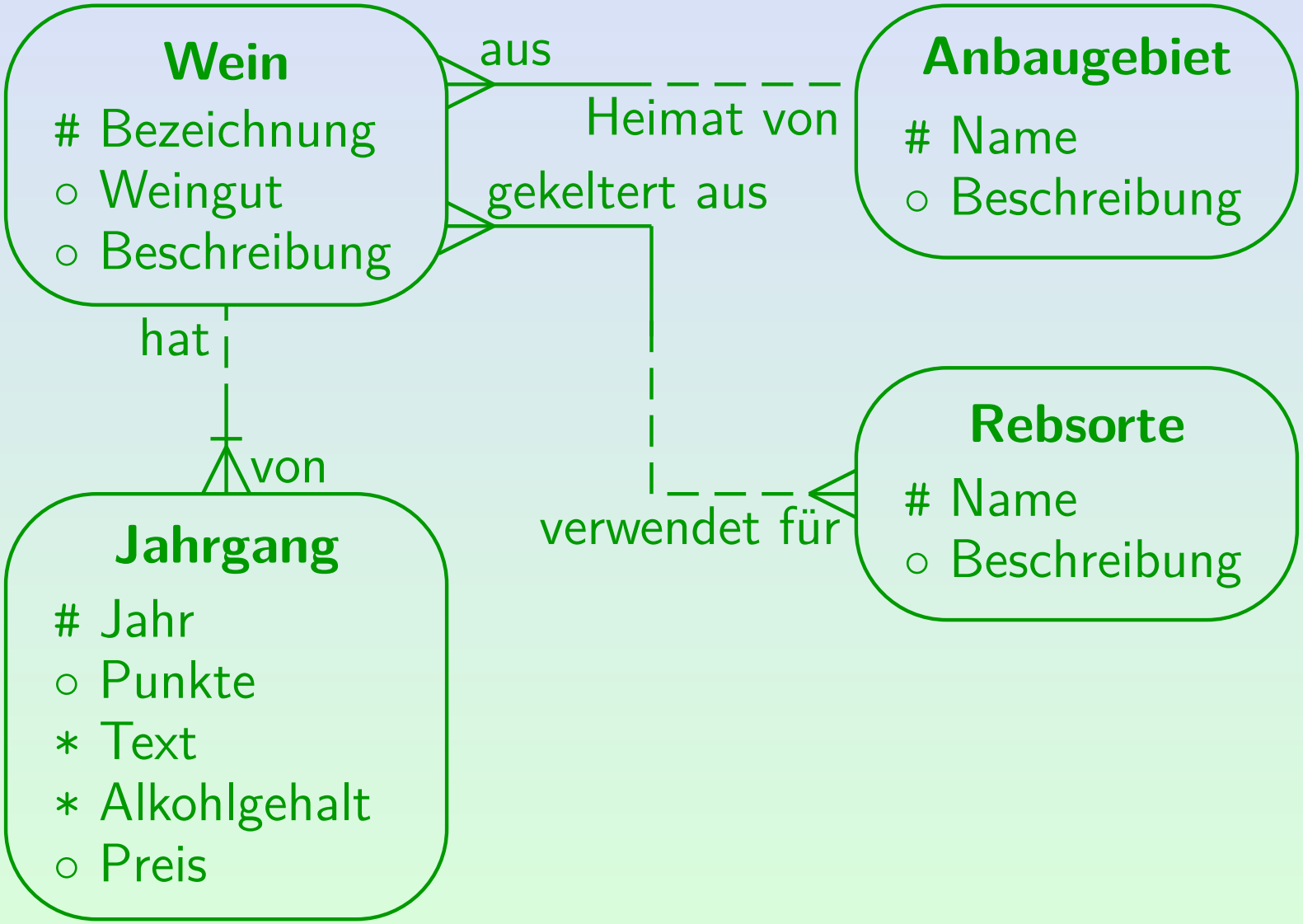
Hausaufgabe 14.1: ER-Entwurf (7)

- Zu einem Wein kann es null oder mehr Jahrgänge geben. Ein Jahrgang (eines Weins) gehört natürlich zu genau einem Wein.



- Jahrgang ist ein schwacher Entity-Typ: Er ist identifiziert über die Beziehung zu Wein und über das Jahr.

Hausaufgabe 14.1: ER-Entwurf (8)



Hausaufgabe 14.1: ER-Entwurf (9)

Häufiger Fehler:

- „Jahrgang“ muss ein schwaches Entity sein, neben dem Kähenfuß an der Beziehung zu „Wein“ muss ein Querstrich über die Beziehungslinie stehen.
- Ein zusätzliches Schlüsselattribut „Wein“ von „Jahrgang“ ist falsch.

  Was sollte es enthalten? Die Bezeichnung des Weines? Man darf die Bezeichnung des Weines nicht hier noch einmal redundant abspeichern. Der zugehörige Wein wird über das Relationship erreicht.
- Die Jahreszahl alleine reicht auch nicht als Schlüssel.

  Natürlich werden in einem Jahr viele verschiedene Weine gekeltert.

Hausaufgabe 14.1: ER-Entwurf (10)

Weitere Anmerkungen:

- Die Notation sieht keine horizontalen Linien in den Entity-Kästen vor.
- Verbessern Sie die Aufgabenstellung nicht.

  Es macht vielleicht nicht besonders viel Sinn, dass die Datenbank auch Weine ohne Jahrgang zulässt, aber das steht explizit in der Aufgabe. Also muss die Beziehungslinie von Wein zu Jahrgang auf der Wein-Seite gestrichelt sein.
- Eine Beziehung von A zu B hat zwei Namen: Einmal für die Richtung von A nach B, dieser Name steht auf der A-Seite. Und einmal für die Richtung von B zu A, dieser Name steht auf der B-Seite.

  Bei der Beziehung von Wein zu Weinanbaugebiet muss „kommt aus“ also auf der Wein-Seite stehen.

Inhalt

Hausaufgabe 14.2: Logischer Entwurf (1)

- Übersetzen Sie das ER-Diagramm auf der nächsten Folie in ein relationales Schema.

  Nutzen Sie die Kurznotation aus der Vorlesung (in der ASCII-Variante) für das relationale Schema:

  Primärschlüssel-Attribute kennzeichnen Sie mit vorangestelltem `#`.

  Fremdschlüssel markieren Sie mit `->` und dem Namen der referenzierten Tabelle, bei zusammengesetzten Fremdschlüsseln verwenden Sie Klammern `(A1,A2)->R`.

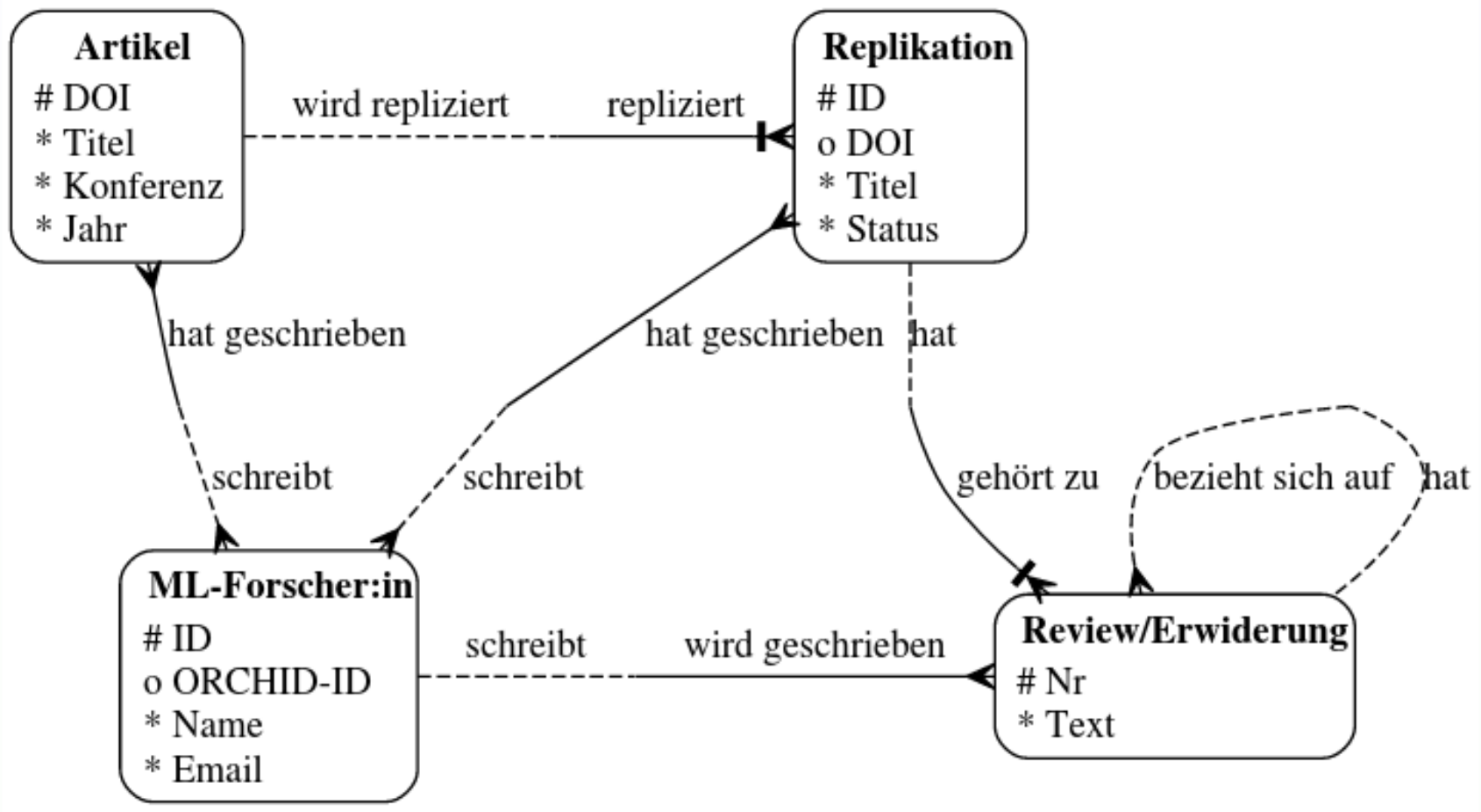
  Hinter Attribute, die Nullwerte erlauben, schreiben Sie „`?`“.

- Bei Bedarf geben Sie auch weitere Integritätsbedingungen an, die notwendig sind, um die Äquivalenz des relationalen Schemas zum ER-Schema sicherzustellen.

  Es reicht dabei eine Formulierung in natürlicher Sprache.

- Das Datenbank-Schema ist etwas vereinfacht. Sie sollen nur das gegebene Schema übersetzen und nicht verbesseren.

Hausaufgabe 14.2: Logischer Entwurf (2)



Hausaufgabe 14.2: Logischer Entwurf (3)

- Der erste Schritt ist, die gegebenen Entity-Typen in entsprechende Tabellen zu übersetzen.
- Für den Schlüssel eines schwachen Entity-Typs braucht man den Schlüssel des übergeordneten Entity-Typs.
- Daher sollte die Reihenfolge der Übersetzung der Hierarchie von schwachen Entity-Typen entsprechen, also hier:
  - Artikel
  - Replikation
  - Review
- Der unabhängige Entity-Typ „`ML-Forscher:in`“ kann jederzeit übersetzt werden.

Hausaufgabe 14.2: Logischer Entwurf (4)

- Übersetzung des Entity-Typs „Artikel“:

  Artikel(DOI, Titel, Konferenz, Jahr)

- Beim schwachen Entity-Typ „Replikation“ muss man den Schlüssel des übergeordneten Entity-Typs als Fremdschlüssel und Teil des Schlüssels einfügen.

  Damit hat man gleichzeitig die viele-zu-eins-Beziehung zum übergeordneten Entity-Typ übersetzt.

- Es ergibt sich das Problem, dass der Entity-Typ „Replikation“ selbst ein Attribut „DOI“ hat.

  Es ist wohl so zu verstehen, dass die Replikation selbst einen „Digital Object Identifier“ haben kann, also selbst ein damit aufzufindener Forschungsartikel ist. Man muss die Anwendung aber nicht verstehen, um die Übersetzung ausführen zu können (siehe Präsenzaufgabe). In der Praxis sollte man natürlich mitdenken, um eventuelle Fehler im ER-Diagramm zu finden.

Hausaufgabe 14.2: Logischer Entwurf (5)

- Eine Tabelle kann keine zwei Spalten mit gleichem Namen haben.
- Man muss den Namenskonflikt also lösen, indem man eine oder beide Spalten umbenennt.

  Man sollte das gut dokumentieren. Es wäre auch darüber nachzudenken, die Umbenennung eventuell schon im ER-Diagramm durchzuführen.
- Hier wurde der Name des Relationships (in Richtung zum übergeordneten Entity) für den Fremdschlüssel benutzt:

  $$\texttt{Replikation}(\underline{\texttt{repliziert}} \rightarrow \texttt{Artikel}, \underline{\texttt{ID}}, \texttt{DOI}^{\circ}, \texttt{Titel}, \texttt{Status})$$

  Eine Alternative wäre z.B. `Artikel_DOI` und `Repli_DOI`.
- Man kann für Tabellen auch die Pluralform verwenden.

Hausaufgabe 14.2: Logischer Entwurf (6)

- Entsprechend werden beim schwachen Entity-Typ Review/Erwiderung die beiden Primärschlüssel-Attribute des übergeordneten Entity-Typs Replikation übernommen:

  Review((repliziert, ID)→Replikation, Nr, Text)

  Der Tabellenname wurde etwas gekürzt. Wenn man wirklich den „/“ im Tabellenname haben will, müsste man in SQL „Delimited Identifier“ verwenden, was bei vielen späteren Anfragen etwas mühsam ist.
  Dies gilt besonders, wenn die SQL-Anfragen in Java-Programmen als String-Konstanten stehen, die dort selbst mit "..." begrenzt werden.
  Eventuell sollte man die Umbenennung schon im ER-Diagramm durchführen. Das ER-Diagramm soll ja auch später noch helfen, im relationalen Schema die Übersicht zu behalten. In Werkzeugen wie dem Oracle Designer kann man Synonyme zu Entity-Typen hinterlegen.

Hausaufgabe 14.2: Logischer Entwurf (7)

- Die Tabelle `Review` ist noch nicht fertig, da der Entity-Typ die „viele“-Seite von zwei eins-zu-viele-Beziehungen ist, und entsprechende Fremdschlüssel zur Tabelle hinzugefügt werden.
- Zunächst wird aber der letzte Entity-Typ „`ML-Forscher:in`“ übersetzt:

  `Forscher(ID, ORCHID-ID°, Name, EMail)`

  Je nach Vorgabe des Arbeitgebers, des Kunden, oder persönlicher Präferenz kann/muss man das „Gendern“ auch auf der Ebene der relationalen Tabellen durchführen. Dies führt objektiv zu Mehrarbeit bei der späteren Verwendung der Tabelle in SQL-Anfragen und anderen Anweisungen. Die entscheidene Frage ist, ob alle Beteiligten die grammatikalisch männliche Form als geschlechtsneutral verstehen können (wie das ja lange Zeit so gehandhabt wurde).

Hausaufgabe 14.2: Logischer Entwurf (8)

- Eins-zu-viele Beziehungen werden in einen Fremdschlüssel auf der „Viele“-Seite übersetzt.
- Die Beziehung „ein/eine ML-Forscher:in“ schreibt viele Reviews/Erwiderungen“ bewirkt also,
  - dass der Schlüssel der Tabelle, in die „ML-Forscher:in“ übersetzt wurde (`Forscher`),
  - als Fremdschlüssel zur Tabelle, in die „Review/Erwiderung“ übersetzt wurde (`Review`), hinzugefügt wird.
- Ergebnis:

  `Review((repliziert, ID)→Replikation, Nr, Text, geschrieben_von→Forscher)`

  Es wird hier auch der Relationship-Name in Leserichtung als Fremdschlüsselname verwendet. Leider war „`wird geschrieben`“ auf dem ER-Diagramm unpassend. Das wurde hier jetzt korrigiert.

Hausaufgabe 14.2: Logischer Entwurf (9)

- Ebenso wird die „rekursive Beziehung“ zwischen zwei Entities des Typs „Review/Erwiderung“ übersetzt:

  Review((repliziert, ID)→Replikation, Nr,
  Text, geschrieben_von→Forscher,
  bezieht_sich_auf→Review°)

- Da sich ein Entity vom Typ Review/Erwiderung nicht unbedingt auf eine anderes Entity dieses Typs beziehen muss (die Linie ist gestrichelt), erlaubt dieser Fremdschlüssel auch Nullwerte.

  Reviews beziehen sich direkt auf eine Replikation, und Erwiderungen nehmen Stellung zu Reviews oder sind Erwiderungen auf Erwiderungen. Die Beziehung sollte wohl keine Zyklen enthalten, aber das ist eine Integritätsbedingung, die bereits beim konzeptuellen Entwurf spezifiziert werden müsste. Der logische Entwurf übersetzt nur.

Hausaufgabe 14.2: Logischer Entwurf (10)

- Nachdem die Entity-Typen und eins-zu-viele Relationships jetzt vollständig übersetzt sind, fehlen noch die viele-zu-viele Beziehungen.
- Sie werden in eigene Tabellen übersetzt, die jeweils Paare von Schlüsselwerten der beiden beteiligten Entity-Typen enthalten.
- Der Name der Tabelle liegt nicht ganz fest, man muss hier beim logischen Entwurf noch Entscheidungen fällen.
- Im Beispiel gibt es wieder einen Namenskonflikt, wenn man z.B. den Relationsship-Namen in der gleichen Richtung benutzt („geschrieben von“ bzw. „hat geschrieben“).

Die Relationship-Namen im gegebenen ER-Diagramm sind leider nicht gut.

Hausaufgabe 14.2: Logischer Entwurf (11)

- Lösung z.B.:
  - `Autor_Artikel(ID→Forscher, DOI→Artikel)`
  - `Autor_Replikation(ID→Forscher, (repliziert,RID)→Replikation)`
- Bei der zweiten Relation gibt es wieder einen Namenskonflikt: Bei naiver Anwendung der Regeln würde man zwei Mal die Spalte „`ID`“ bekommen.

  Wahrscheinlich wäre es angebracht, das Problem schon auf ER-Ebene zu lösen. Wenn das ER-Diagramm nicht geändert werden kann, könnte man auch jetzt noch auf die Idee kommen, die Spalte `ID` in der Tabelle `Replikation` entsprechend in `RID` umzubenennen, um später Joins mit `USING` zu ermöglichen (und die Verbund-Möglichkeiten klar zu machen). Aber das ER-Diagramm verliert an Wert als Dokumentation des relationalen Schemas.

Hausaufgabe 14.2: Logischer Entwurf (12)

- Die obige Übersetzung von viele-zu-viele-Beziehungen kann nur die „minimale Kardinalität 0“ auf beiden Seiten (also die gestrichelte Linie) implementieren.
- Im Beispiel sind die Linien aber teilweise durchgezogen. Das führt zu zusätzlichen Integritätsbedingungen:
  - Jeder Artikel muss mindestens einen Eintrag in der Tabelle `Autor_Artikel` haben.

    Dies entspricht der Spezifikation aus dem ER-Diagramm, dass jeder Artikel von mindestens einem/einer ML-Forscher:in geschrieben sein muss.
  - Zu jeder Zeile in Replikation muss es mindestens eine zugehörige Zeile in der Tabelle `Autor_Replikation` geben.

    „Zugehörig“ heißt mit dem Schlüsselwert als Fremdschlüsselwert.

Hausaufgabe 14.2: Logischer Entwurf (13)

Zusammenfassung:

- Forscher(ID, ORCHID-ID°, Name, EMail)
- Artikel(DOI, Titel, Konferenz, Jahr)
- Replikation(repliziert→Artikel, ID, DOI°, Titel, Status)
- Review((repliziert, ID)→Replikation, Nr, Text, geschrieben_von→Forscher, bezieht_sich_auf→Review°)
- Autor_Artikel(ID→Forscher, DOI→Artikel)
- Autor_Replikation(ID→Forscher, (repliziert,RID)→Replikation)

Hausaufgabe 14.2: Logischer Entwurf (14)

Häufige Fehler:

- Weil „Replikation“ ein schwaches Entity ist, besteht der Schlüssel der entsprechenden Tabelle aus
  - dem „DOI“ des replizierten Artikels (Fremdschlüssel, der Artikel referenziert und gleichzeitig Teil des Schlüssels), und

    Man muss das Attribut umbenennen, weil der Entity-Typ „Replikation“ noch ein eigenes Attribut „DOI“ hat.
  - dem „ID“ der Replikation.
- Es sollte das gegebene ER-Diagramm ins relationale Modell übersetzt werden, und nicht der Entwurf „verbessert“ werden.

  Das ist zu 95% ein rein mechanischer Prozess. Im Ergebnis sollen keine völlig neuen Spaltennamen auftauchen, außer, wenn eine Spalte wegen Namenskonflikten umbenannt werden muss.

Inhalt

Aufgabe zu Normalformen (1)

- Gegeben sei folgende Tabelle mit einigen Daten einer Schule:

SCHULE				
SCHUELER	KLASSE	FACH	LEHRER	RAUM
Karin Koch	7a	Deutsch	Herr Goetz	305
Karin Koch	7a	Englisch	Frau Rowling	305
Karin Koch	7a	Mathematik	Frau Noether	305
Frank Fischer	7a	Deutsch	Herr Goetz	305
Frank Fischer	7a	Englisch	Frau Rowling	305
Frank Fischer	7a	Mathematik	Frau Noether	305
Ronald Richter	10b	Mathematik	Frau Noether	210
Ronald Richter	10b	Englisch	Herr Pratchett	210
Ronald Richter	10b	Pyrotechnik	Herr Shimizu	210
Ronald Richter	10b	Religion	Herr Lewis	210

Die letzte Spalte heißt eigentlich „Klassenraum“. Das ist wichtig, weil die funktionale Abhängigkeit „Klasse bestimmt Klassenraum“ vorausgesetzt wird.

Aufgabe zu Normalformen (2)

- Es seien folgende funktionale Abhängigkeiten gegeben:
  - SCHUELER $\longrightarrow$ KLASSE
  - KLASSE, FACH $\longrightarrow$ LEHRER
  - KLASSE $\longrightarrow$ KLASSENRAUM
  - LEHRER, KLASSENRAUM $\rightarrow$ LEHRER
  - SCHUELER, FACH $\rightarrow$ KLASSENRAUM
- Z.B. bedeutet die erste funktionale Abhängigkeit (FA)

  SCHUELER $\longrightarrow$ KLASSE:

  „Wenn zwei Zeilen den gleichen Wert in der Spalte SCHUELER haben, dann müssen sie auch den gleichen Wert in der Spalte KLASSE haben.“

  Jeder Schüler gehört zu einer eindeutig bestimmten Klasse, diese ist nicht vom Unterrichtsfach abhängig.

Normalformen, Teil a): Weitere FA (1)

Aufgabe:

- Geben Sie eine weitere funktionale Abhängigkeit an,
  - die nicht von den explizit gegebenen Abhängigkeiten impliziert wird,
  - aber im Beispielzustand der Tabelle gilt.
- Es ist nicht verlangt, dass die funktionale Abhängigkeit im realen Leben realistisch ist.

Normalformen, Teil a): Weitere FA (2)

Lösung:

- Mögliche Lösung:

  $$\texttt{KLASSENRAUM} \longrightarrow \texttt{KLASSE}$$

  Dies wurde bei den funktionalen Abhängigkeiten vergessen. Es macht sogar im realen Leben Sinn: Ein Raum kann nur für eine Klasse Klassenraum sein.

- Um zu prüfen, dass

  $$\texttt{KLASSENRAUM} \longrightarrow \texttt{KLASSE}$$

  im Beispiel-Zustand gilt, muss man nachschauen, dass in Gruppen von Zeilen mit gleichem KLASSENRAUM auch überall die gleiche KLASSE steht.

  Man könnte eine SQL-Anfrage dafür schreiben.

Normalformen, Teil a): Weitere FA (3)

Lösung, Forts.:

- Zum Beweis, dass diese FA nicht von den gegebenen FAen impliziert wird, kann man die Attributhülle

$$\{\texttt{KLASSENRAUM}\}^+ = \{\texttt{KLASSENRAUM}\}$$

berechnen (bezüglich der gegebenen FAen).

- Da sie nicht KLASSE enthält, wird

$$\texttt{KLASSENRAUM} \longrightarrow \texttt{KLASSE}$$

nicht impliziert.

Normalformen, Teil a): Weitere FA (4)

Lösung, Forts.:

- Es gibt noch weitere Lösungen, z.B.

$$\texttt{LEHRER} \longrightarrow \texttt{FACH}$$

- Im Beispiel-Zustand unterrichtet jeder Lehrer nur ein Fach.
- Dies wäre in der Realität aber eher nicht so.

  Das kann natürlich von der Schule abhängen. Man muss im Rahmen des Datenbank-Entwurfs welche funktionalen Abhängigkeiten vorausgesetzt werden können.

- Man könnte diese beiden funktionalen Abhängigkeiten auch weiter abschwächen, z.B. wäre auch eine Lösung:

$$\texttt{LEHRER, KLASSENRAUM} \rightarrow \texttt{FACH}$$

Normalformen, Teil b): Attributhülle (1)

Aufgabe:

- Berechnen Sie die Attributhülle $\{\texttt{SCHUELER}\}^+$.

Normalformen, Teil b): Attributhülle (2)

Lösung:

- Lösung: Man startet mit dem gegebenen Attribut:

$$\{\texttt{SCHUELER}\}^+ = \{\texttt{SCHUELER}, \ldots\}.$$

- Aufgrund der funktionalen Abhängigkeit

$$\texttt{SCHUELER} \longrightarrow \texttt{KLASSE}$$

kann man KLASSE hinzufügen:

$$\{\texttt{SCHUELER}\}^+ = \{\texttt{SCHUELER}, \texttt{KLASSE}, \ldots\}.$$

Wenn die linke Seite einer funktionalen Abhängigkeit bereits in der Attributhülle ist, kann man auch die rechte Seite hinzufügen.

- Fortsetzung siehe nächste Folie.

Normalformen, Teil b): Attributhülle (3)

Lösung:

- Da man nun KLASSE hat, kann man die funktionale Abhängigkeit

  $$\texttt{KLASSE} \longrightarrow \texttt{KLASSENRAUM}$$

  anwenden:

  $$\{\texttt{SCHUELER}\}^+ = \{\texttt{SCHUELER}, \texttt{KLASSE}, \texttt{KLASSENRAUM}\}.$$

- Weitere funktionale Abhängigkeiten können nicht angewendet werden:
  - $\texttt{KLASSE, FACH} \longrightarrow \texttt{LEHRER}$
  - $\texttt{LEHRER, KLASSENRAUM} \rightarrow \texttt{LEHRER}$
  - $\texttt{SCHUELER, FACH} \rightarrow \texttt{KLASSENRAUM}$

Normalformen, Teil c): Implizierte FA (1)

Aufgabe:

- Implizieren die gegebenen funktionalen Abhängigkeiten die folgende Abhängigkeit?

$$\texttt{SCHUELER} \longrightarrow \texttt{KLASSENRAUM}$$

Begründen Sie Ihre Aussage.

Normalformen, Teil c): Implizierte FA (2)

Lösung:

- Ja, die FA wird logisch impliziert. Begründung:
  - Wir haben gerade die Attributhülle $\{\texttt{SCHUELER}\}^+$ berechnet.

    Im allgemeinen berechnet man für Aufgaben dieses Typs die Attributhülle der linken Seite der fraglichen funktionalen Abhängigkeit (bezüglich der gegebenen funktionalen Abhängigkeiten).
  - Die Attributhülle

    $\{\texttt{SCHUELER}\}^+ = \{\texttt{SCHUELER}, \texttt{KLASSE}, \texttt{KLASSENRAUM}\}$.

    enthält die rechte Seite der zu prüfenden FA (KLASSENRAUM).

Normalformen, Teil d): Schlüssel (1)

Aufgabe:

- Geben Sie alle minimalen Schlüssel der Tabelle an. Begründen Sie Ihre Aussage, d.h.
  - dass es jeweils ein Schlüssel ist,
  - dass dieser bzw. diese Schlüssel minimal sind, und
  - dass es keine weiteren Schlüssel gibt.

Lösung:

- Wir müssen $\subseteq$-minimale Mengen $\mathcal{A}$ von Attributen finden, so dass $\mathcal{A}^+$ die Menge aller Attribute der Tabelle ist.

  Ein Schlüssel bestimmt alle Attribute der Tabelle funktional.

  Die Attributhülle $\mathcal{A}^+$ ist die Menge alle von $\mathcal{A}$ funktional bestimmten Attribute.

Normalformen, Teil d): Schlüssel (2)

Lösung, Forts.:

- Die gegebenen funktionale Abhängigkeiten sind:
  - SCHUELER $\longrightarrow$ KLASSE
  - KLASSE, FACH $\longrightarrow$ LEHRER
  - KLASSE $\longrightarrow$ KLASSENRAUM
  - LEHRER, KLASSENRAUM $\rightarrow$ LEHRER
  - SCHUELER, FACH $\rightarrow$ KLASSENRAUM
- Die Attribute SCHUELER und FACH kommen auf keiner rechten Seite vor.
- Wenn sie nicht von Anfang an in $\mathcal{A}$ enthalten sind, können sie nicht in $\mathcal{A}^+$ enthalten sein.

Normalformen, Teil d): Schlüssel (3)

Lösung, Forts.:

- Die beiden Attribute SCHUELER und FACH müssen also in jedem Schlüssel enthalten sein.
- Wenn man nun

  $$\{\texttt{SCHUELER, FACH}\}^+$$

  berechnet, stellt man fest, dass es alle Attribute sind:
  - SCHUELER $\longrightarrow$ KLASSE
  - KLASSE, FACH $\longrightarrow$ LEHRER
  - KLASSE $\longrightarrow$ KLASSENRAUM
- Damit haben wir Glück: Es gibt nur einen minimalen Schlüssel, nämlich diese beiden Attribute.

Normalformen, Teil e): BCNF

Aufgabe:

- Ist die Tabelle bezüglich der fünf gegebenen funktionalen Abhängigkeiten in BCNF? Begründen Sie Ihre Aussage.
- Falls die Tabelle nicht in BCNF sein sollte, überführen Sie die Tabelle durch verlustlose Aufspaltung in mehrere Tabellen, die in BCNF sind.

  Alternativ dürfen Sie auch den 3NF Synthesealgorithmus verwenden.

Normalformen, Teil e): BCNF prüfen (1)

Lösung:

- Nein, die drei ersten funktionalen Abhängigkeiten verletzen BCNF, da die linke Seite kein Schlüssel (nicht notwendig minimal) ist, und sie nicht trivial sind:
  - SCHUELER $\longrightarrow$ KLASSE
  - KLASSE, FACH $\longrightarrow$ LEHRER
  - KLASSE $\longrightarrow$ KLASSENRAUM
  - LEHRER, KLASSENRAUM $\rightarrow$ LEHRER
  - SCHUELER, FACH $\rightarrow$ KLASSENRAUM

Normalformen, Teil e): BCNF prüfen (1)

Lösung, Forts.:

- Die vierte funktionale Abhängigkeit verletzt BCNF nicht, da sie trivial ist:

  $$\texttt{LEHRER, KLASSENRAUM} \rightarrow \texttt{LEHRER}$$

  Die rechte Seite (LEHRER) kommt auch links vor.

- Die fünfte funktionale Abhängigkeit verletzt BCNF nicht, da links ein vollständiger Schlüssel steht:

  $$\texttt{SCHUELER, FACH} \rightarrow \texttt{KLASSENRAUM}$$

  Es könnten links auch noch weitere Attribute stehen, das wäre kein Problem. Man kann die BCNF-Bedingung auch ohne Vorab-Berechnung der Schlüssel testen, indem man die Attributhülle der linken Seiten berechnet. Dies muss jeweils die Menge aller Attribute sein (es sei denn, die FA wäre trivial, dann kann man sie sowieso streichen).

Normalformen, Teil e): BCNF herstellen (1)

- Verletzt $\alpha \longrightarrow \beta \in \mathcal{F}$ die BCNF, erzeugt man
  - R_1 mit den Attributen α^+.

    Dies enthält natürlich die Attribute $\alpha \cup \beta$ (wie in der oben gezeigten Dekomposition), könnte aber noch weitere Attribute enthalten (aber nur, wenn noch mehr FAen BCNF verletzen).
  - R_2 mit den Attributen $(\mathcal{A} \setminus \alpha^+) \cup (\alpha \setminus \beta)$.

Normalformen, Teil e): BCNF herstellen (2)

- Wenn nur eine funktionale Abhängigkeit $\alpha \longrightarrow \beta$ BCNF verletzt, erzeugt man einfach eine Relation mit den Attributen $\alpha \cup \beta$ (wobei α Schlüssel ist), und entfernt die rechte Seite β aus der ursprünglichen Relation.

  Voraussetzung dabei ist, dass $\alpha \cap \beta = \emptyset$. Ansonsten entfernt man den „trivialen Anteil“ $\alpha \cap \beta$ einfach von der rechten Seite.

- Im Beispiel gibt es leider mehrere Verletzungen.

  Z.B. wäre das Ergebnis schlecht, wenn man zuerst die Verletzung durch `SCHUELER` $\longrightarrow$ `KLASSE` angeht: Dann würde man das Attribut `KLASSE` aus der ursprünglichen Tabelle entfernen (nach obigem Algorithmus immerhin zusammen mit `KLASSENRAUM`). Anschließend könnte man die funktionale Abhängigkeit `KLASSE, FACH` $\longrightarrow$ `LEHRER` nicht mehr ausdrücken. Man bekommt durch sukzessive Aufspaltung immer Tabellen in BCNF, und die Aufspaltung ist verlustlos, nicht immer das gefühlt beste Ergebnis.

Normalformen, Teil e): BCNF herstellen (3)

- Man sollte bei mehreren Verletzungen die FAen möglichst so anordnen, dass Attribute auf der rechten Seite (die entfernt werden) nicht in einer späteren FA links vorkommen.
- Starten wir also mit:

$$\texttt{KLASSE, FACH} \longrightarrow \texttt{LEHRER}$$

Man könnte genauso gut mit KLASSE $\longrightarrow$ KLASSENRAUM starten, dagegen wäre SCHUELER $\longrightarrow$ KLASSE ungünstig.

- Man legt eine neue Tabelle mit den Attributen der FA an:

LEHRER_EINTEILUNG(KLASSE, FACH, LEHRER)

- Die rechte Seite wird aus der ursprünglichen Tabelle entfernt:

SCHULE1(SCHUELER, KLASSE, FACH, KLASSENRAUM)

Dabei bilden KLASSE und FACH einen Fremdschlüssel zu LEHRER_EINTEILUNG.

Normalformen, Teil e): BCNF herstellen (4)

- Nun verletzt SCHULE1 noch immer BCNF, z.B. wegen

  $$\texttt{KLASSE} \longrightarrow \texttt{KLASSENRAUM}.$$

  Auch wegen SCHUELER $\longrightarrow$ KLASSE, aber das heben wir uns noch auf.

- Also legt man eine neue Tabelle mit den Attributen der FA an:

  KLASSEN(KLASSE, KLASSENRAUM).

  Die linke Seite der FA wird immer Schlüssel. Sie bestimmt nach der Konstruktion ja die übrigen Attribute.

- Wieder entfernen wir die rechte Seite der FA aus der ursprünglichen Tabelle:

  SCHULE2(SCHUELER, KLASSE$\rightarrow$KLASSEN, FACH)

Normalformen, Teil e): BCNF herstellen (5)

- Auch SCHULE2 verletzt noch BCNF wegen

  $$\texttt{SCHUELER} \longrightarrow \texttt{KLASSE},$$

- Also machen wir auch dazu eine Tabelle:

  KLASSEN_EINTEILUNG(SCHUELER, KLASSE→KLASSEN).

- Die rechte Seite wird aus der ursprünglichen Tabelle entfernt, übrig bleibt:

  LERNT(SCHUELER→KLASSEN_EINTEILUNG, FACH)

  Leider wird dabei der ursprüngliche Fremdschlüssel zerstört: Bei SCHULE1 wurde festgelegt, dass KLASSE und FACH zusammen einen Fremdschlüssel bilden, der auf LEHRER_EINTEILUNG verweist. Jetzt werden die beiden Attribute aber auseinander gerissen.

Normalformen, Teil e): BCNF herstellen (6)

- Das Endergebnis ist also:
  - LEHRER_EINTEILUNG(KLASSE, FACH, LEHRER)
  - KLASSEN(KLASSE, KLASSENRAUM).
  - KLASSEN_EINTEILUNG(SCHUELER, KLASSE→KLASSEN).
  - LERNT(SCHUELER→KLASSEN_EINTEILUNG, FACH)

    Da es für diese Tabelle keine FAen mehr gibt, sind beide Attribute zusammen der Schlüssel.
- Im Prinzip hat man aus jeder der interessanten FAen eine Tabelle gemacht, und LERNT ist der Rest.

  Der 3NF-Synthesealgorithmus macht im Prinzip genau das. Im Skript ist bei diesem Algorithmus erklärt, wie man die gegebenen FAen aufbereiten muss, damit das funktioniert. Im Beispiel würden bei der Berechnung der „kanonischen Überdeckung“ die vierte und fünfte FA eliminiert.

Exkurs zu 4NF (1)

Nicht prüfungsrelevant:

- Das Beispiel ist leider komplizierter als beabsichtigt.
- Intuitiv ist die Tabelle

  LERNT(SCHUELER, FACH)

  für diese Anwendung falsch.
- Die Fächer hängen normalerweise nur von der Klasse ab, und nicht vom einzelnen Schüler. Dann würde man eher diese Tabellen wollen:
  - KLASSEN(KLASSE, KLASSENRAUM)
  - FAECHER(KLASSE→KLASSEN, FACH, LEHRER)
  - KLASSEN_EINTEILUNG(SCHUELER, KLASSE→KLASSEN)

Exkurs zu 4NF (2)

- Die gegebenen funktionalen Abhängigkeiten sagen aber nicht aus, dass die Fächer nur von der Klasse abhängen.

  Das kann man beweisen, indem man einen DB-Zustand macht, der alle FAen erfüllt, aber zwei Schüler in der gleichen Klasse enthält, die verschiedene Fächer haben.

- Es kommt ja auch vor, dass Klassen bei bestimmten Fächern aufgespalten werden:
  - Ein Teil der Schüler hat Latein, der andere Teil Französisch.
  - Oder evangelische Religion, katholische Religion, Ethik.
- Wenn die Fächer tatsächlich vom Schüler abhängen und nicht von der Klasse, war der obige Entwurf doch richtig.

Exkurs zu 4NF (3)

- Will man ausdrücken, dass zwei Schüler in der gleichen Klasse immer die gleiche Menge von Fächern haben, braucht man dazu „mehrwertige Abhängigkeiten“:
  - $\texttt{KLASSE} \longrightarrow\!\!\!\!\rightarrow \texttt{SCHUELER}$
  - $\texttt{KLASSE} \longrightarrow\!\!\!\!\rightarrow \texttt{FACH, LEHRER}$
- Diese Art von Integritätsbedingungen bedeutet: „Wenn zwei Zeilen in der linken Seite übereinstimmen, kann man die Werte der rechten Seite vertauschen, und bekommt zwei Zeilen, die auch in der Tabelle enthalten sein müssen.“

  Z.B. kann man für zwei Zeilen mit der gleichen `KLASSE` die `SCHUELER` vertauschen: Die so entstandenen „neuen“ Zeilen müssen bereits in der Tabelle enthalten sein.

Exkurs zu 4NF (4)

- Normalisierung mit mehrwertigen Abhängigkeiten führt zu 4NF.
- Die Reihenfolge der Normalformen ist: 2NF, 3NF, BCNF, 4NF, 5NF.
  - Normalformen später in der Liste implizieren Normalformen früher in der Liste.
  - Z.B. ist eine Tabelle in BCNF automatisch in 3NF.
  - Von diesen Normalformen basieren 2NF, 3NF und BCNF auf funktionalen Abhängigkeiten.

    Wenn man mehr will, braucht man allgemeinere Integritätsbedingungen.
- Dies alles wird ausführlich in der Vorlesung „Datenbank-Entwurf“ im Master behandelt.