Einführung in Datenbanken

Übung 13: ER-Diagramme

Prof. Dr. Stefan Brass
PD Dr. Alexander Hinneburg

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2020/21

http://www.informatik.uni-halle.de/~brass/db20/

Lehr-Evaluation

- Nehmen Sie bitte an der offiziellen Lehr-Evaluation der Universität teil, insbesondere auch für diese Vorlesung.
- Sie machen damit deutlich, dass Ihnen die Qualität von Lehrveranstaltungen nicht egal ist.

  Die aktuelle Rücklaufquote sind 10%, gerechnet auf 160 Teilnehmer aus StudIP. Die tatsächlich aktive Teilnehmerzahl liegt bei ca. 100.
- Die Umfrage ist anonym. Deswegen bitte eventuelle Kritikpunkte im Freitext möglichst verständlich und konstruktiv formulieren — Rückfragen sind kaum möglich.
- Ich werde mir das Ergebnis gründlich durchlesen.
- Auch Professoren freuen sich über gute Noten.

  Falls es Ihnen gefallen hat, sollten Sie auch teilnehmen — nicht nur dann, wenn Sie Kritik auf dem Herzen haben. Nur so ergibt sich ein korrektes Bild.

Inhalt

Hausaufgabe 11a (1)

- Zwei Darstellungen der Gesamtleistung für Hausaufgaben, Zwischenklausur, Endklausur:

Resultate_1

STUDENT	H	Z	E
Ann Lloyd	80	90	95
Jim Ford	95	60	75

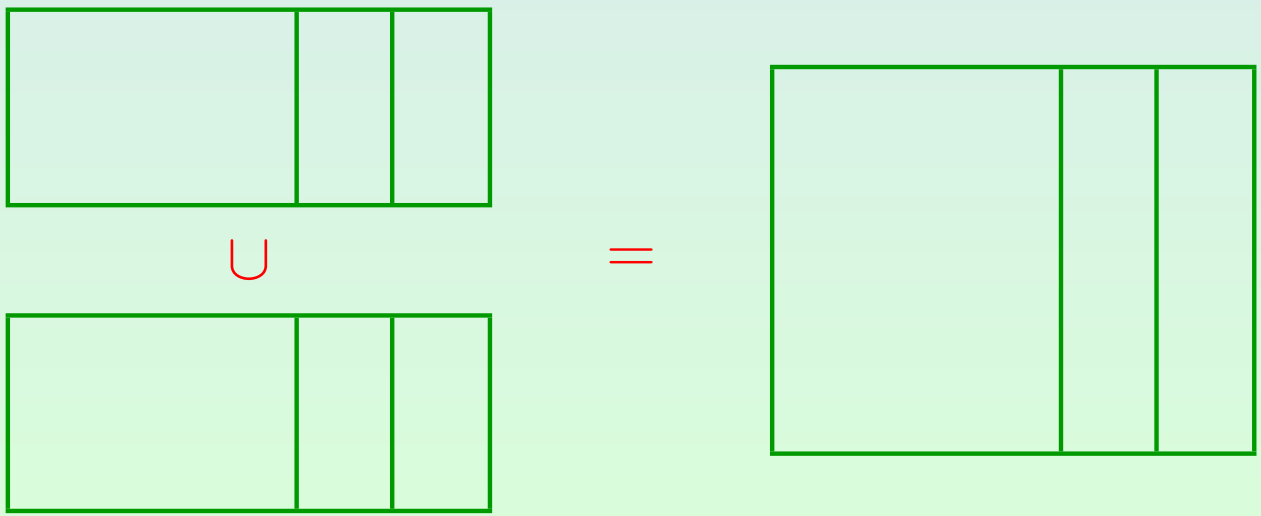
Resultate_2

STUDENT	ATYP	PROZENT
Ann Lloyd	H	80
Ann Lloyd	Z	90
Ann Lloyd	E	95
Jim Ford	H	95
Jim Ford	Z	60
Jim Ford	E	75

- Welchen Vorteil hat `Resultate_1` gegenüber `Resultate_2`?
- Welchen Vorteil hat `Resultate_2` gegenüber `Resultate_1`?

Hausaufgabe 11a (2)

- `Resultate_1(STUDENT, H, Z, E)`
- `Resultate_2(STUDENT, ATYP, PROZENT)`
- a) Umrechnung von `Resultate_1` in `Resultate_2`.
- In der Ausgabe-Spalte `PROZENT` sollen Werte stehen, die in der Eingabe in mehreren verschiedenen Spalten (`H`, `Z`, `E`) stehen. Das fordert `UNION`:



Hausaufgabe 11a (3)

- Berechung von `Resultate_2` aus `Resultate_1`:

```
SELECT STUDENT, 'H' AS ATYP, H AS PROZENT
FROM   Resultate_1
UNION  ALL
SELECT STUDENT, 'Z' AS ATYP, Z AS PROZENT
FROM   Resultate_1
UNION  ALL
SELECT STUDENT, 'E' AS ATYP, E AS PROZENT
FROM   Resultate_1
```

- So werden aus jeder Zeile von `Resultate_1` drei Zeilen im Ergebnis.

  Das passt zur Zeilen-Anzahl im Beispiel-Zustand: `Resultate_1` hat 2 Zeilen, aber `Resultate_2` hat 6 Zeilen.

Hausaufgabe 11a (4)

- Ungewöhnliche Alternative (von Student/Studentin):

```
SELECT r.student, x.atyp, x.prozent
FROM   Resultate_1 r,
       LATERAL (VALUES ('H', r.h),
                       ('Z', r.z),
                       ('E', r.e))
                  AS x(atyp, prozent)
```

- Durch das Schlüsselwort `LATERAL` kann man in der Tabelle rechts auf die aktuelle Zeile links (`r`) zugreifen.
- Nicht besonders portabel.

  Sowohl der `LATERAL` Join, als auch die Spezifikation einer Tabelle mit `VALUES` funktionieren nicht in vielen Systemen. Aber interessante Idee und etwas kürzer.

Hausaufgabe 11b (1)

- Zwei Darstellungen der Gesamtleistung für Hausaufgaben, Zwischenklausur, Endklausur:

Resultate_1

STUDENT	H	Z	E
Ann Lloyd	80	90	95
Jim Ford	95	60	75

Resultate_2

STUDENT	ATYP	PROZENT
Ann Lloyd	H	80
Ann Lloyd	Z	90
Ann Lloyd	E	95
Jim Ford	H	95
Jim Ford	Z	60
Jim Ford	E	75

- Nun die umgekehrte Transformation: `Resultate_2` ist gegeben, und wir wollen `Resultate_1` berechnen.

Hausaufgabe 11b (2)

- `Resultate_1(STUDENT, H, Z, E)`
- `Resultate_2(STUDENT, ATYP, PROZENT)`
- b) Umrechnung von `Resultate_2` in `Resultate_1`.
- Man braucht drei Zeilen von `Resultate_2` (für die drei verschiedenen `ATYP`-Werte), um eine Zeile von `Resultate_1` zu produzieren. Das fordert einen Join:

- Der Join macht die Tabellen breiter (mehr Spalten), die Vereinigung macht sie höher (mehr Zeilen).

Hausaufgabe 11b (3)

- Berechnung von `Resultate_1` aus `Resultate_2`:

```
SELECT H.STUDENT, H.PROZENT AS H,
       Z.PROZENT AS Z, E.PROZENT AS E
FROM   Resultate_2 H,
       Resultate_2 Z,
       Resultate_2 E
WHERE  H.STUDENT = Z.STUDENT
AND    Z.STUDENT = E.STUDENT
AND    H.ATYP = 'H'
AND    Z.ATYP = 'Z'
AND    E.ATYP = 'E'
```

- Also Selbstverbund mit jeweils drei Tupeln von `Resultate_2` (für die drei Aufgabentypen).

Hausaufgabe 11b (4)

- Lösung mit Teilanfragen:

```
WITH
H AS    (SELECT student, prozent as H
         FROM   Resultate_2
         WHERE  atyp = 'H'),
Z AS    (SELECT student, prozent as Z
         FROM   Resultate_2
         WHERE  atyp = 'Z'),
E AS    (SELECT student, prozent as E
         FROM   Resultate_2
         WHERE  atyp = 'E')
SELECT * FROM H NATURAL JOIN Z NATURAL JOIN E
```

Hausaufgabe 11b (5)

- Mit Unteranfragen unter SELECT:

```
SELECT h.student, h.prozent AS H,
       (SELECT z.prozent
        FROM   Resultate_2 z
        WHERE  z.atyp = 'Z'
        AND    z.student = h.student) AS Z,
       (SELECT e.prozent
        FROM   Resultate_2 e
        WHERE  e.atyp = 'E'
        AND    e.student = h.student) AS E
FROM   Resultate_2 h
WHERE  h.atyp = 'H'
```

- Im Beispiel-Zustand ist das Ergebnis gleich, aber ist die Anfrage äquivalent (Ja/Nein-Umfrage)?

Hausaufgabe 11b (6)

- Komische Lösung, Gruppen statt Join:

```
SELECT student,
       MIN(CASE WHEN atyp = 'H' THEN prozent
                ELSE NULL END) AS H,
       MIN(CASE WHEN atyp = 'Z' THEN prozent
                ELSE NULL END) AS Z,
       MIN(CASE WHEN atyp = 'E' THEN prozent
                ELSE NULL END) AS E
FROM   Resultate_2
GROUP  BY student
```

- Auch `MAX`, `SUM`, `AVG` würden funktionieren (in jeder Gruppe gibt es nur einen nicht Null-Wert). „`ELSE NULL`“ optional.
- Im Beispiel-Zustand ist das Ergebnis gleich, aber ist die Anfrage äquivalent zu einer früheren (Ja/Nein-Umfrage)?

Hausaufgabe 11c (1)

- Die folgende Anfrage ist in relationaler Algebra zu schreiben. Sie bezieht sich auf die Beispiel-Datenbank der Vorlesung:
  - STUDENTEN(SID, VORNAME, NACHNAME, EMAIL°)
  - AUFGABEN(ATYP, ANR, THEMA, MAXPT)
  - BEWERTUNGEN(SID→STUDENTEN, (ATYP, ANR)→AUFGABEN, PUNKTE)
- Berechnen Sie Name und Vorname aller Studierenden, die die höchste Punktzahl für Hausaufgabe 2 bekommen haben. Die Forderung ist also:
  - Der oder die Studierende hat überhaupt Hausaufgabe 2 bearbeitet.
  - Es gibt keinen Studierenden mit mehr Punkten für Hausaufgabe 2.

Hausaufgabe 11c (2)

- Die Anfrage verhält sich nicht-monoton. Sie muss also eine der folgenden Operationen enthalten:
  - Mengendifferenz

    Dies ist die einzige nicht-monotone Basisoperation der Relationenalgebra. Letztendlich ist also immer eine Mengendifferenz beteiligt. Sie könnte sich aber in den folgenden abgeleiteten Operationen verstecken.
  - Anti-Join
  - Outer Join
- RelaX enthält einen Gruppierungsoperator mit Aggregationsfunktionen. Dieses Konstrukt würden sich auch nichtmonoton verhalten, ist aber üblicherweise nicht Bestandteil der Relationenalgebra.

  Es war nicht beabsichtigt, dass Sie diesen Operator verwenden.

Hausaufgabe 11c (3)

- Die erwartete Antwort ist (Präfix egal):

STUDENTEN.VORNAME	STUDENTEN.NACHNAME
'Michael'	'Grau'

- Am einfachsten definiert man sich zunächst eine Hilfsrelation mit den Ergebnissen für Hausaufgabe 2:

$$\texttt{H2} := \sigma_{\texttt{ATYP='H'} \wedge \texttt{ANR=2}}(\texttt{BEWERTUNGEN});$$

- Nun berechnet man die nicht besten Studierenden:

$$\texttt{Geschlagen} := \sigma_{\texttt{X.PUNKTE<Y.PUNKTE}}\left(\rho_{\texttt{X}}(\texttt{H2}) \times \rho_{\texttt{Y}}(\texttt{H2})\right)$$

- Dann die SIDs der Studierenden mit bestem Ergebnis:

$$\texttt{Ziel} := \pi_{\texttt{SID}}(\texttt{H2}) \setminus \pi_{\texttt{SID} \leftarrow \texttt{X.SID}}(\texttt{Geschlagen})$$

- Schließlich den Namen:

$$\pi_{\texttt{VORNAME, NACHNAME}}(\texttt{STUDENTEN} \bowtie \texttt{Ziel})$$

Hausaufgabe 11c (4)

- Gleiche Lösung in ASCII (für RelaX):

```
H2 = sigma ATYP = 'H' and ANR = 2 (BEWERTUNGEN)
Geschlagen = sigma X.PUNKTE < Y.PUNKTE
                    (rho X (H2) x rho Y (H2))
Ziel = pi SID (H2) - pi SID <- X.SID (Geschlagen)
pi VORNAME, NACHNAME (STUDENTEN join Ziel)
```

  Man kann das in RelaX testen:
  [http://dbis-uibk.github.io/relax/calc/gist/8dc2652578ee12ae756a234c4cf21b3f]

- Man kann bei RelaX den Relationspräfix so entfernen (war nicht verlangt):

```
pi VORNAME <- STUDENTEN.VORNAME, ... (STUDENTEN)
```

  Abkürzbar als: `pi VORNAME <- VORNAME, ... (STUDENTEN)`

Inhalt

Integritätsüberwachung mit Anfragen (1)

Schreiben Sie folgende Anfrage in SQL (3 Punkte):

- Es soll die Integritätsbedingung überwacht werden, dass kein Angestellter mehr als sein direkter Vorgesetzter verdient.
- `emp(empno, ename, job, mgr°→emp, hiredate, sal, comm°, deptno°→dept)`
- Schreiben Sie eine Anfrage, die Fehlermeldungen folgender Form liefert, wenn die Bedingung verletzt wird:

errmsg
SCOTT verdient 3000, sein Vorgesetzter JONES nur 2975
FORD verdient 3000, sein Vorgesetzter JONES nur 2975

- Falls es keine Interitätsverletzungen gibt, soll die Anfrage eine Zeile mit dem Wert „`ok`“ in der Spalte „`errmsg`“ liefern.

Zum Test können Sie die IB modifizieren: Max. 50 $ mehr erlaubt.

Integritätsüberwachung mit Anfragen (2)

- Lösung (erweiterbar um zusätzliche Bedingungen):

```
WITH   FEHLER1 AS
       (SELECT u.ename || ' verdient ' || u.sal ||
               ', sein Vorgesetzter ' ||
               v.ename || ' nur ' || v.sal
               AS errmsg
        FROM   emp u, emp v
        WHERE  u.mgr = v.empno
        AND    u.sal > v.sal)
SELECT errmsg FROM FEHLER1
UNION  ALL
SELECT 'ok' AS errmsg
WHERE  NOT EXISTS (SELECT * FROM FEHLER1)
```

Integritätsüberwachung mit Anfragen (3)

- In PostgreSQL ist es möglich, eine Anfrage mit `SELECT` und `WHERE`, aber ohne `FROM` zu schreiben:
  - Es gibt dann 0 oder 1 Ausgabezeilen, je nachdem, ob die `WHERE`-Bedingung erfüllt ist.
  - Formal gibt es in diesem Fall also genau eine Variablenbelegung (für die leere Menge von Variablen).
- Bei MySQL könnte man z.B. schreiben:

```
FROM (SELECT 1) X
```

  Was man selektiert, ist nicht wichtig, die „1“ ist nur ein Beispiel. MySQL erlaubt Anfragen nur mit `SELECT`, die dann genau eine Ergebniszeile liefern. Dagegen funktioniert `SELECT ... WHERE ...` (ohne `FROM`) in MySQL nicht.
- Oracle hat eine Systemtabelle `DUAL`, die garantiert genau eine Zeile hat (dort ist `FROM` immer Pflicht).

Inhalt

Hausaufgabe 12a (1)

- Über Komponisten soll der Name und der Vorname sowie Geburts- und Todesjahr und außerdem eine eindeutige Nummer gespeichert werden.

  Das Todesjahr ist möglicherweise unbekannt bzw. existiert nicht (bei noch lebenden Komponisten). Alle anderen Datenwerte sind dagegen für alle Komponisten vorhanden.

Hausaufgabe 12a (2)

- Stücke haben auch eine eindeutige Nummer, außerdem einen Titel, eine Tonart und ggf. eine Opus-Nummer.

Tonart und Opus-Nummer sind nicht immer bekannt bzw. vorhanden.

Hausaufgabe 12a (3)

- Es sei vorausgesetzt, dass jeder Komponist in der Datenbank mindestens ein Stück komponiert hat.

- Durchgezogene Linie auf der Seite des Komponisten: Komponist muss an der Beziehung teilnehmen.

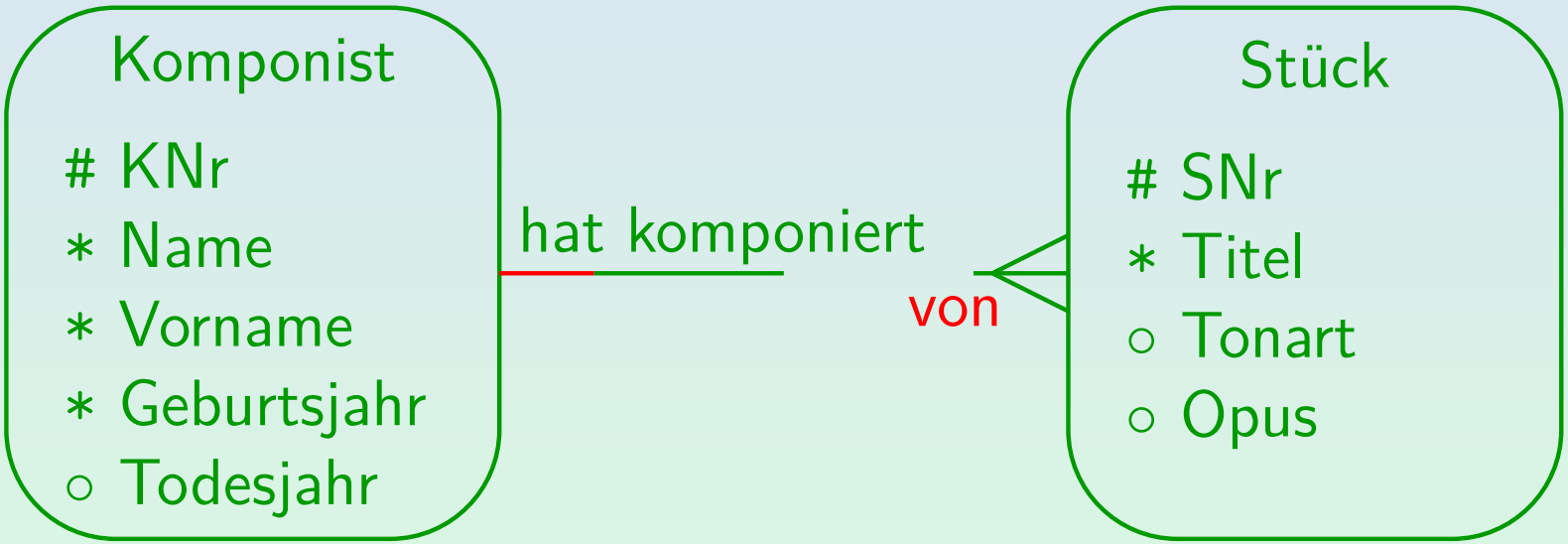
Hausaufgabe 12a (4)

- Er kann natürlich auch mehr als ein Stück geschrieben haben (beliebig viele).

- Krähenfuß auf der Seite des Stücks: Komponist kann zu mehr als einem Stück in Beziehung stehen.

Hausaufgabe 12a (5)

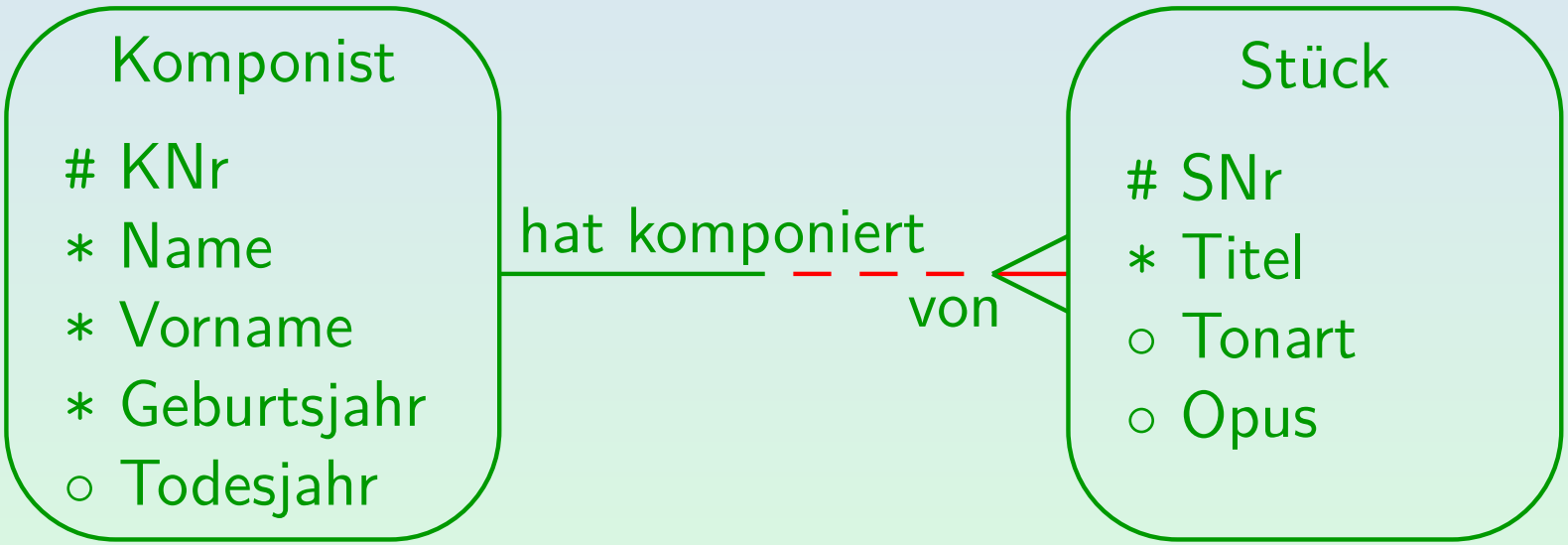
- Stücke in der Datenbank haben immer höchstens einen Komponisten (es geht um klassische Musik).



- Kein Krähenfuß auf der Seite des Komponisten: Jedes Stück ist nur „von“ einem Komponisten.

Hausaufgabe 12a (6)

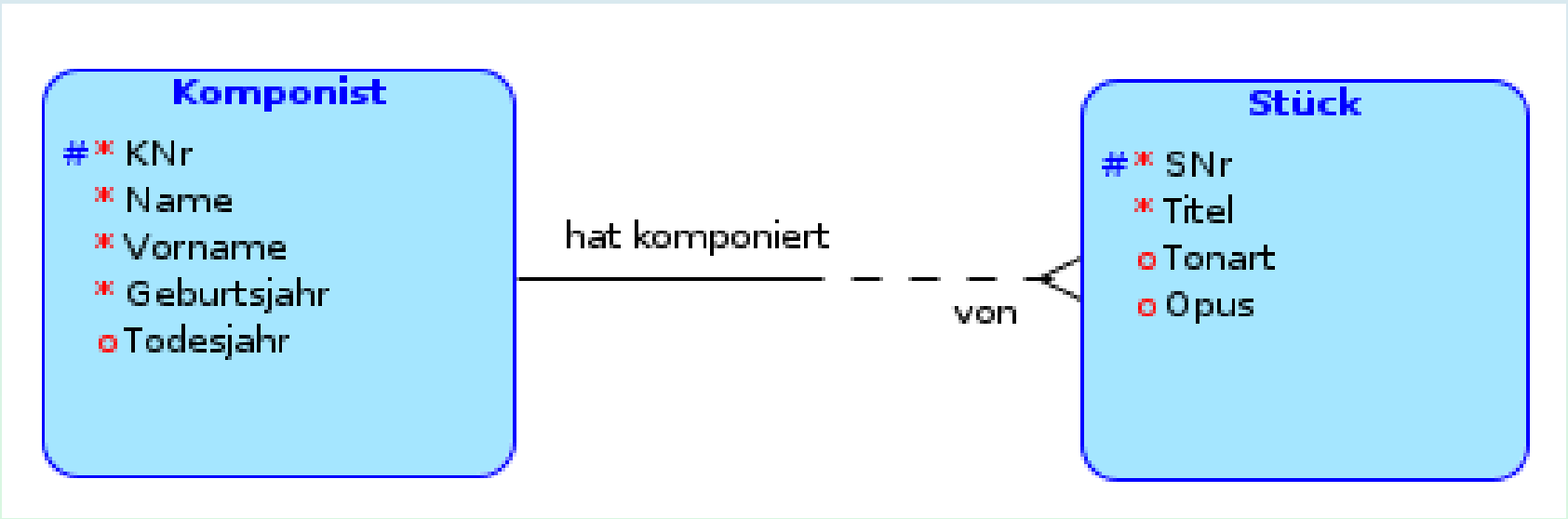
- Es soll aber auch möglich sein, Stücke mit unbekanntem Komponisten (d.h. ohne Komponist) in die Datenbank zu speichern.



- Gestrichelte Linie auf der „Stück“-Seite:
  Stücke müssen nicht unbedingt an der Beziehung teilnehmen.

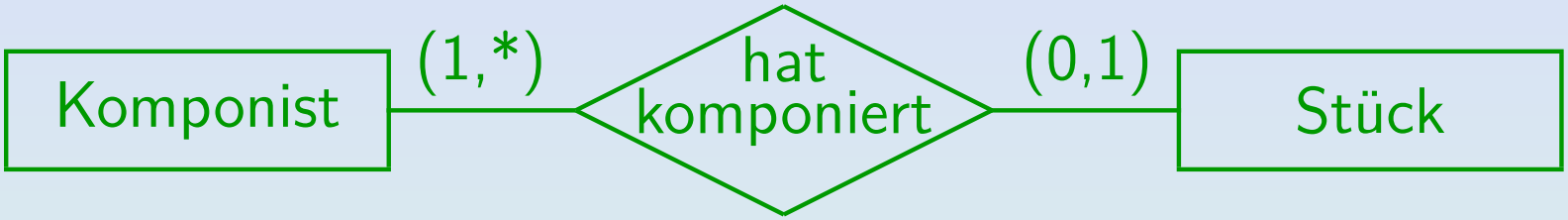
Hausaufgabe 12a (7)

Mit Oracle SQL Developer Data Modeler gezeichnet:



Hausaufgabe 12a (8)

- Alte Notation (Chen-Notation mit (min,max)-Kardinalitäten):



- Die Attribute sind hier weggelassen (wären in Ovalen außerhalb der Rechtecke für die Entity-Typen).
- Die Intervalle spezifizieren, wieviel ausgehende Kanten ein Entity des jeweiligen Typs haben kann. Aber Vorsicht:
  - In UML-Klassendiagrammen genau umgekehrt.
  - In Barker-Notation ist der Krähenfuß für die maximale Kardinalität $*$ auf der anderen Seite.

    Die minimalen Kardinalitäten (erste Komponente) entsprechen dagegen gestrichelter (0) bzw. durchgezogener Linie (1) auf der gleichen Seite.

Hausaufgabe 12a (9)

- Mit handgezeichneten (und eingescannten) Diagrammen können Sie maximal 10 Punkte erreichen (von 11).
  - Zweck dieser Regelung war, dass Sie sich vorbereiten für das Zeichnen von ER-Diagrammen in Ihrer Bachelorarbeit, einem Projekt, oder für kleine Diagramme später im Beruf.

    Für größere Schemata werden Sie spezielle Werkzeuge für den DB-Entwurf nutzen wollen. Siehe Vorlesung „Datenbanken IIA: DB-Entwurf“.
  - All das werden Sie nicht handgezeichnet machen wollen.
- Das Relationship muss wie in der Vorlesung gezeigt beschriftet werden, die Position der beiden Namen kann aber anders sein.
  - Im Notfall reicht auch ein Name in der Mitte.

    Im Oracle SQL Developer Data Modeler werden die „Labels“ der Relationships standardmäßig nicht gezeigt, die Namen werden aber erfasst (und man kann die Anzeige einschalten).

Hausaufgabe 12a (10)

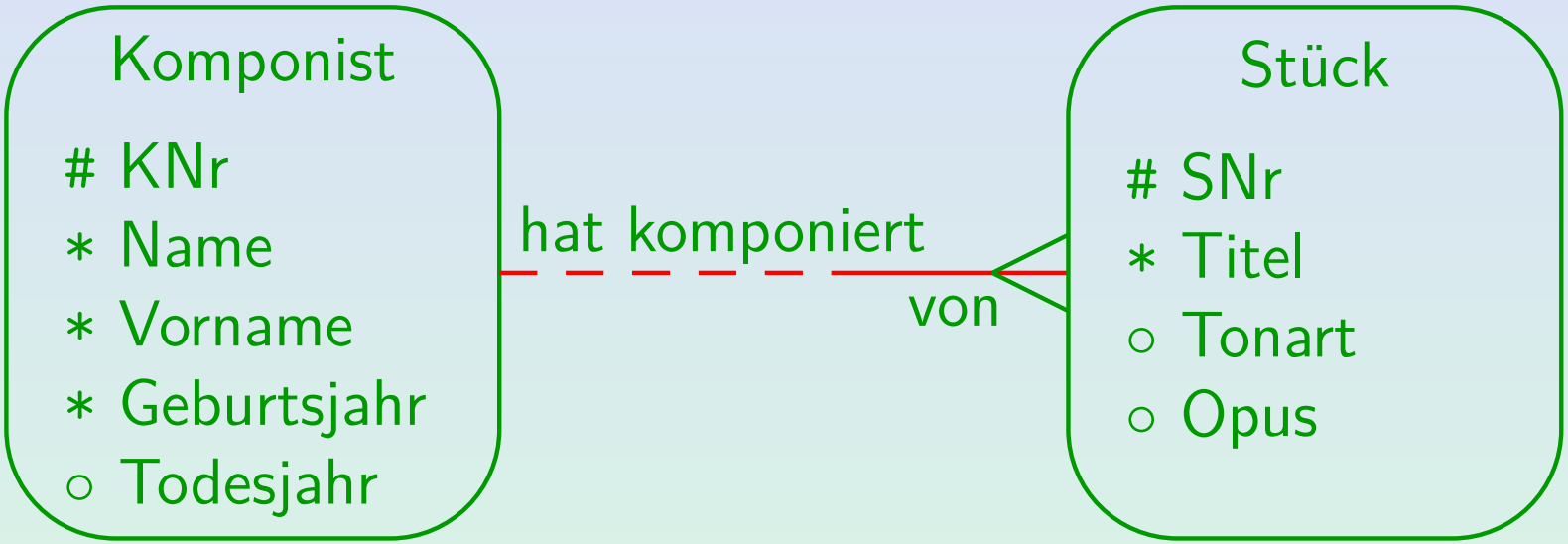
- Womit haben Sie das Diagramm gezeichnet?
  A. Handschriftlich
  B. LaTeX(pur oder mit oder mit Ti*k*Z)
  C. Graphviz
  D. Oracle SQL Developer Data Modeler
  E. Anderes Zeichenprogramm
- Welches andere Zeichenprogramm?
  A. [draw.io] → [https://app.diagrams.net/]
  B. yEd: [https://www.yworks.com/products/yed]
  C. Dia: [http://dia-installer.de/]
  D. Powerpoint, Libre Office, Word, ...
  E. Sonstiges (bitte in Chat schreiben, welches)

Hausaufgabe 12a (11)

- Wieviel Zeit hat das Zeichnen des ER-Diagramms gebraucht?
  - A. 0–15 min.
  - B. 15–30 min
  - C. 30–60 min.
  - D. Mehr als 60 min.
- Können Sie ein Zeichenprogramm besonders empfehlen?
- Möchten Sie von einem Zeichenprogramm abraten?

Hausaufgabe 12a: Beispiele für Fehler (1)

- Gestrichelte Linie und durchgezogene Linie vertauscht:



- Es kann so auch Komponisten in der Datenbank geben, die noch kein einziges Stück komponiert haben.

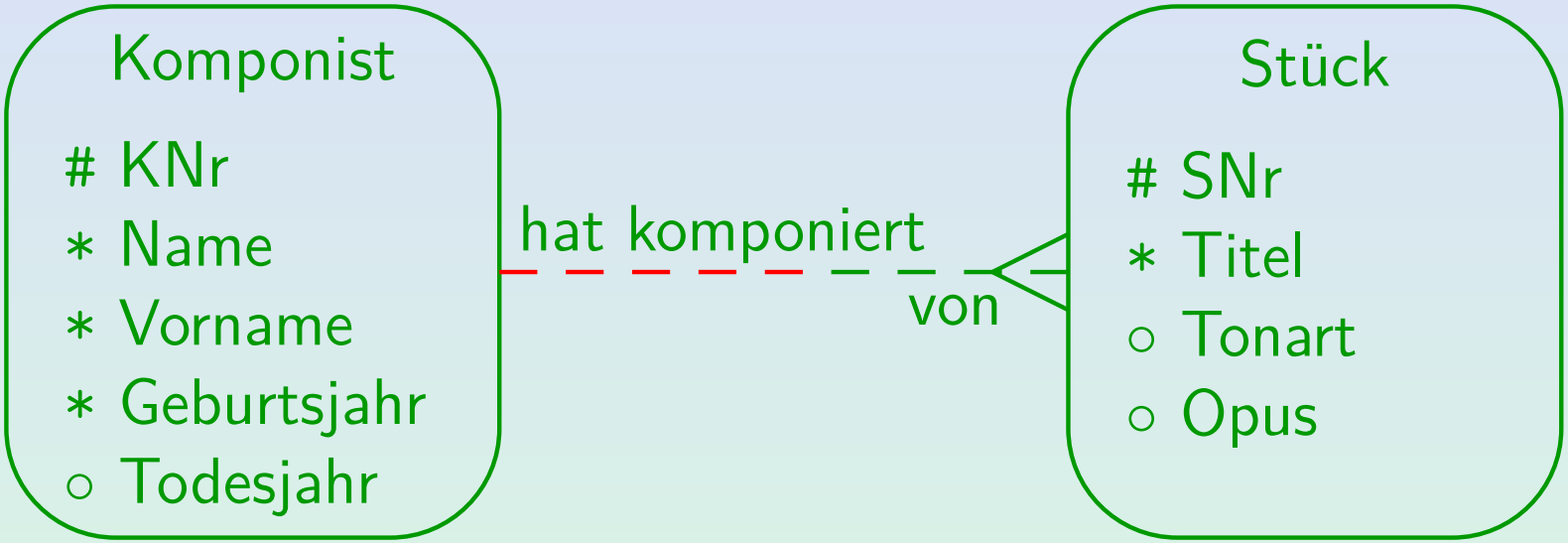
  Das könnte übergangsweise sinnvoll sein, widerspricht aber der Aufgabenstellung.

- Stücke müssen so immer einen Komponisten haben.

  In den Beispiel-Daten gibt es schon 10 Stücke mit einem Nullwert in `knr`.

Hausaufgabe 12a: Beispiele für Fehler (2)

- Line ganz gestrichelt:



- Schema ist zu allgemein:
  - Alle verlangten Zustände können dargestellt werden.
  - Aber auch einige Zustände, die nach der Aufgabenstellung verboten sind (Komponisten ohne Stücke).

    Sie können fragen, ob es nicht vielleicht doch möglich sein sollte, das Komponisten zunächst ohne Stücke in die DB eingefügt werden.

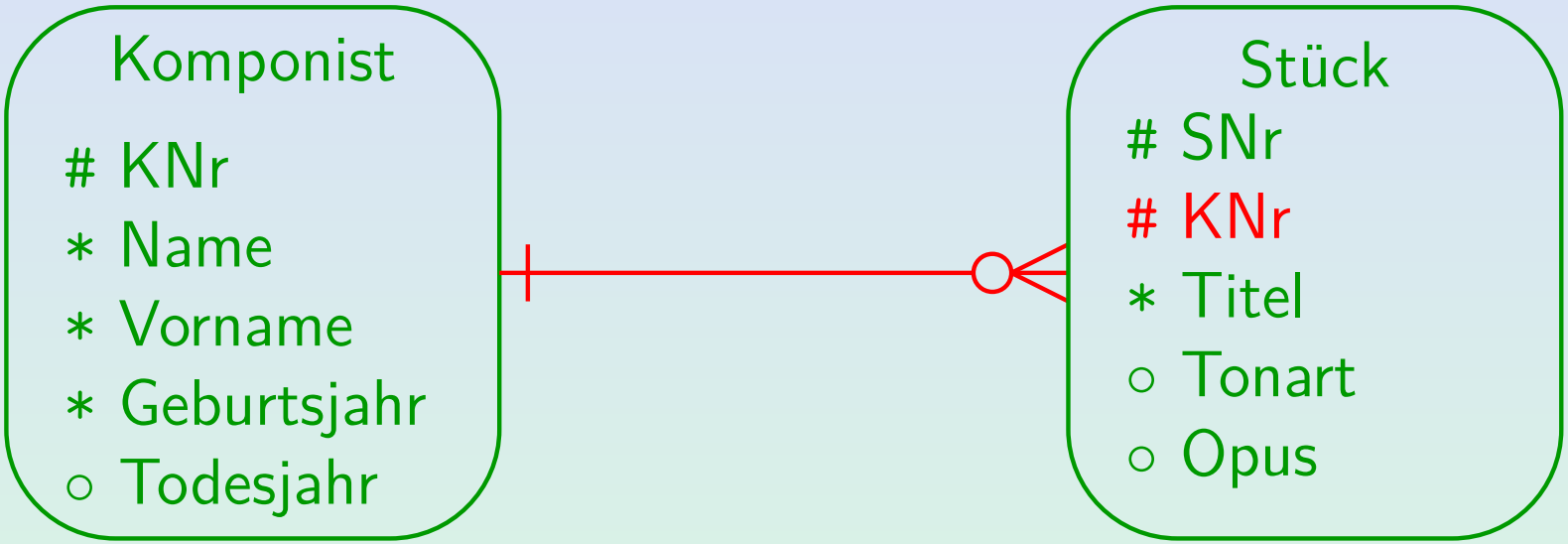
Hausaufgabe 12a: Beispiele für Fehler (3)

- Trennstrich zwischen Entity-Typ-Name und Attributen:

- Sieht ganz nett aus, ist aber formal nicht korrekt.
- Vermutlich von UML Klassendiagrammen inspiriert.

Hausaufgabe 12a: Beispiele für Fehler (4)

- Falsche Notation bzw. falsches Zeichenprogramm:



- Dies ist nicht die Barker-Notation für das Relationship.

  Wenn Sie ein Vorlesungsskript etc. haben, das diese Notation erklärt, würde mich das interessieren.

- Der Fremdschlüssel in Stück ist für ein ER-Diagramm falsch.

  Er ist auch redundant zum dargestellten Relationship.

Hausaufgabe 12a: Beispiele für Fehler (5)

Verschiedenes:

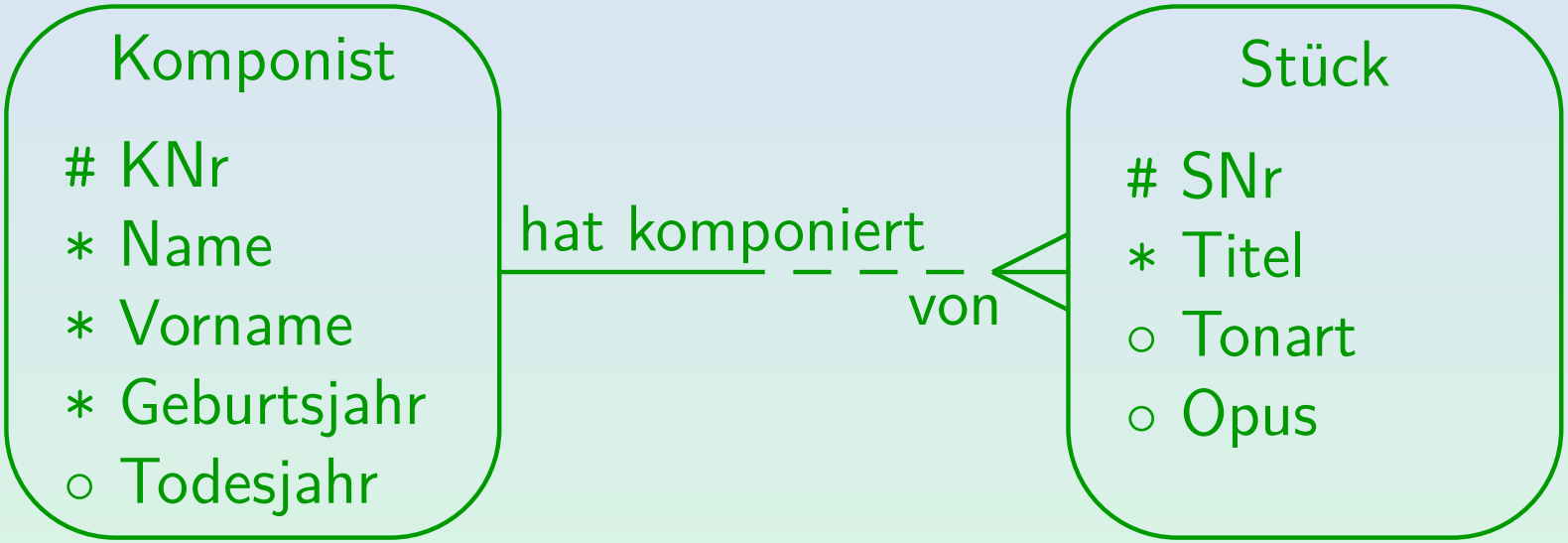
- Ellipsen statt „Softboxes“: Rechtecke mit abgerundeten Ecken.
- Markierung „○“ für optionale Attribute fehlt.
- Symbolspalte und Namensspalte nicht ausgerichtet.

  Die Symbole #, * und ○ müssen übereinander stehen.
- Die zwei Entity-Typen sollten durch eine horizontale Linie verbunden sein (ggf. auch vertikal). Nicht schräg oder abgeknickt.

  Bei komplexeren Diagrammen sind Relationship-Kanten mit einem Knick möglich (Wechsel zwischen horizontal und vertikal). Schräge Kanten sind in seltenen Fällen auch möglich, wenn sie für das Verständnis der Struktur von Vorteil sind. Normalerweise sollte man sich aber auf horizontale und vertikale Linien beschränken. Und Überkreuzungen minimieren.

Logischer Entwurf (1)

- Als Motivation war gesagt, dass das ER-Schema den Tabellen `Komponist` und `Stueck` entsprechen sollte.



- Im „Logischen Entwurf“ (zweite Phase des DB-Entwurfs) wird das ER-Schema ins relationale Modell übersetzt.

  Ziel ist, dass die relationalen Zustände genau den Zuständen zum ER-Schema entsprechen. Wenn der konzeptionelle Entwurf richtig gemacht wurde, entsprechen diese wiederum den Situation des relevanten Welt-Ausschnitts.

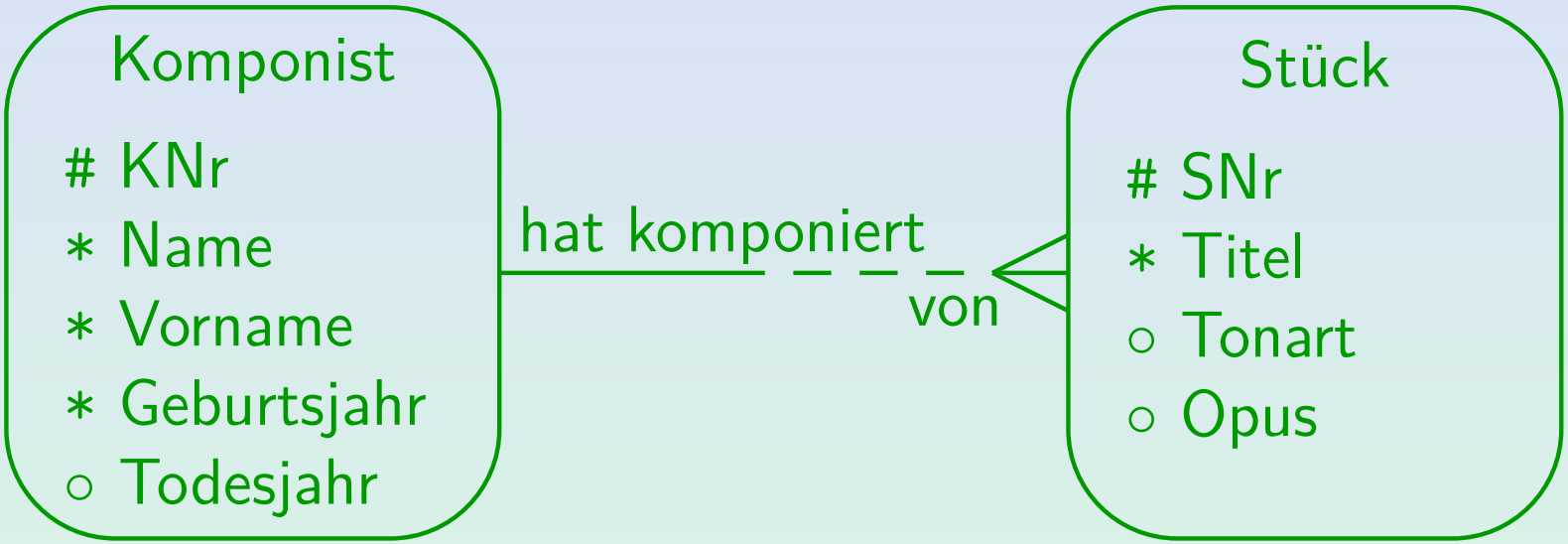
Logischer Entwurf (2)

- ER-Diagramm (Eingabe des logischen Entwurfs):

- Man legt zunächst für jeden Entity-Typ eine entsprechende Tabelle an und übernimmt Attribute, Optionalität, Schlüssel:
  - `Komponist(KNr, Name, Vorname, Geburtsjahr, Todesjahr°)`
  - `Stueck(SNr, Titel, Tonart°, Opus°)`

Logischer Entwurf (3)

- ER-Diagramm:



- Eins-zu-viele (1:n) Beziehungen (Kähenfuß nur auf einer Seite) werden übersetzt, indem man den Schlüssel der „Eins“-Seite als Fremdschlüssel der „Viele“-Seite hinzufügt:
  - `Stueck(SNr, Titel, Tonart°, Opus°`
    `KNr° → Komponist)`

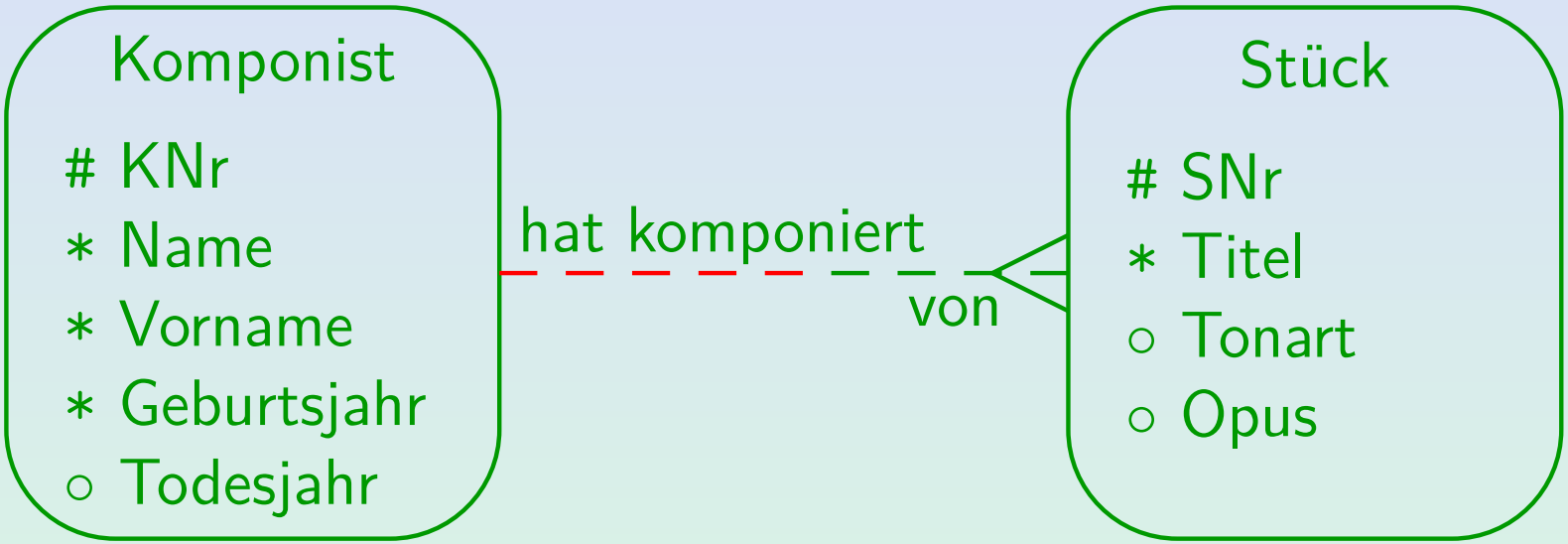
Logischer Entwurf (4)

- ER-Diagramm:

- Der Fremdschlüssel erlaubt Nullwerte, weil die Linie auf der „Stück“-Seite gestrichelt ist (es gibt Stücke ohne Komponist):
  - `Stueck(SNr, Titel, Tonart°, Opus°`
    `KNr° → Komponist)`
- Bei durchgezogener Line wäre der Fremdschlüssel `NOT NULL`.

Logischer Entwurf (5)

- Tatsächlich hat man nur dieses ER-Diagramm übersetzt:



- Beim erzeugten relationalen Schema ist es möglich, Komponisten einzutragen, die von keinem Stück referenziert werden.

  Immerhin können alle vom Original-Diagramm erlaubten Zustände auch als Zustände zum relationalen Schema dargestellt werden. Aber das relationale Schema erlaubt Zustände, die keine Entsprechung im ER-Diagramm haben.

Logischer Entwurf (6)

- Lösung: Zusätzliche allgemeine Integritätsbedingung.
  - In natürlicher Sprache: „Zu jedem Komponisten muss es mindestens ein Stück mit seiner `KNr` geben.“

    Oder: Es darf keine Komponisten ohne Stücke geben.
  - Als logische Formel:

    $\forall$ `Komponist k:` $\exists$ `Stueck s:` `s.KNr` $=$ `k.KNr`
  - Als SQL-Anfrage, die Fehlermeldungen berechnet:

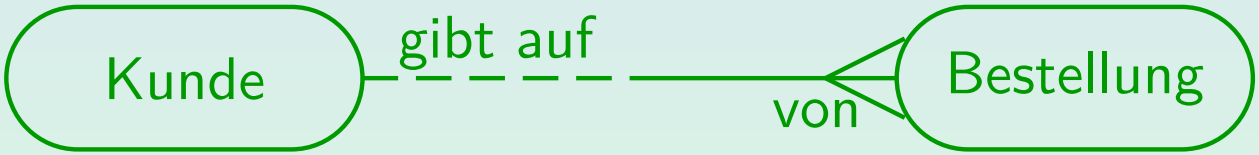
```
SELECT 'Komponist ohne Stücke: ' || Name ||
       ', ' || Vorname AS errmsg
FROM   Komponist k
WHERE  NOT EXISTS (SELECT *
                   FROM   Stueck s
                   WHERE  s.KNr = k.KNr)
```

Aufgaben zu Kardinalitäten (1)

- Legen Sie die Kardinalitäten für eine Beziehung zwischen „Bestellung“ und „Kunde“ fest.

  Hinweis: Neben normalen Kunden enthält die DB auch solche, die noch nichts bestellt haben, sondern nur einen Produktkatalog erhalten haben. Fragen Sie, wenn noch etwas unklar ist. Es ist normal, dass der Datenbank-Entwerfer den Anwendungsexperten fragt.

A.



B.

C.

D.

Aufgaben zu Kardinalitäten (2)

- Legen Sie nun die Kardinalitäten zwischen „Bestellung“ und „Produkt“ fest.

  Eine Bestellung kann mehrere Produkte umfassen. Das gleiche Produkt kann von mehreren Kunden bestellt werden (z.B. bei einem Online-Buchhandel).

A.

B.

C.

D.

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.