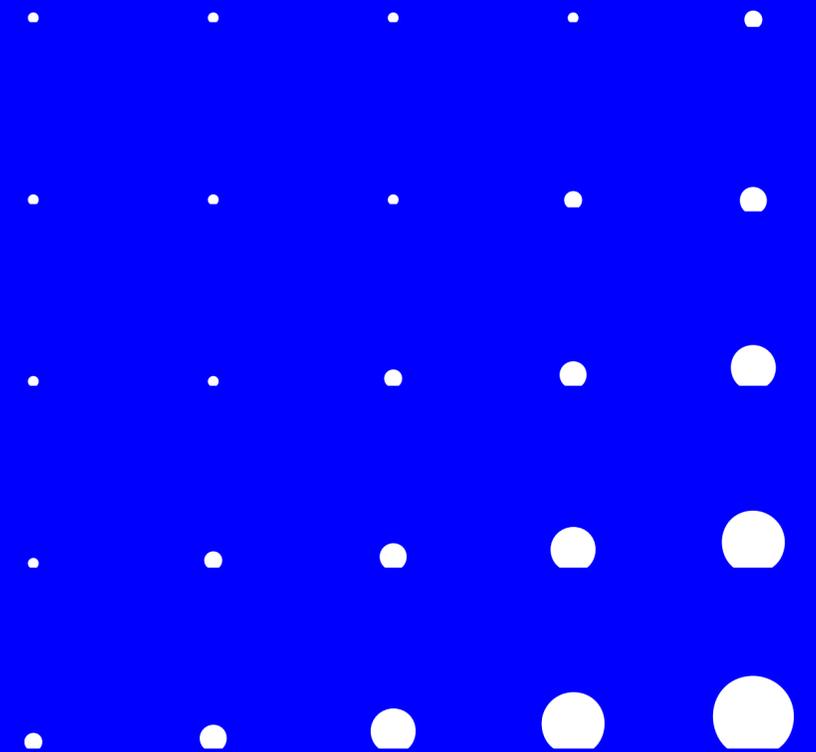


**E/R-Modellierung vs.
UML-Klassenmodellierung**



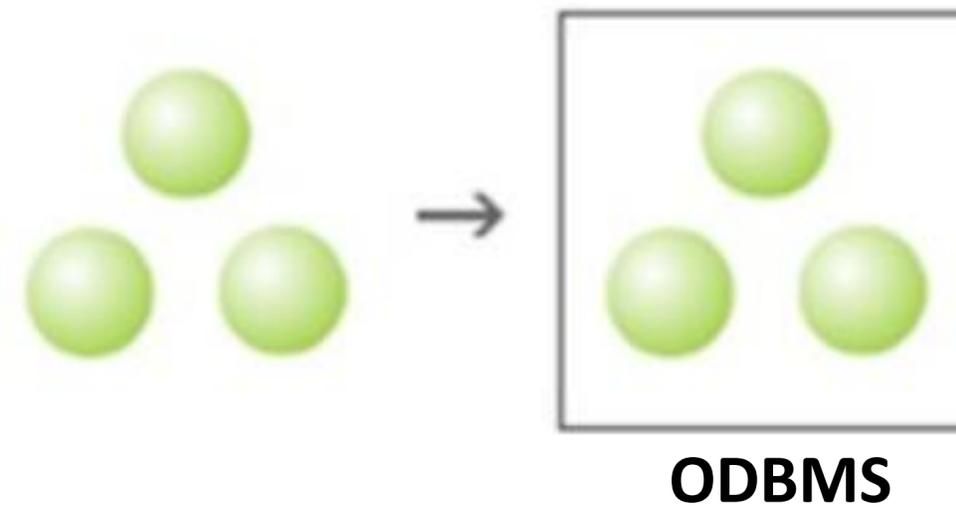
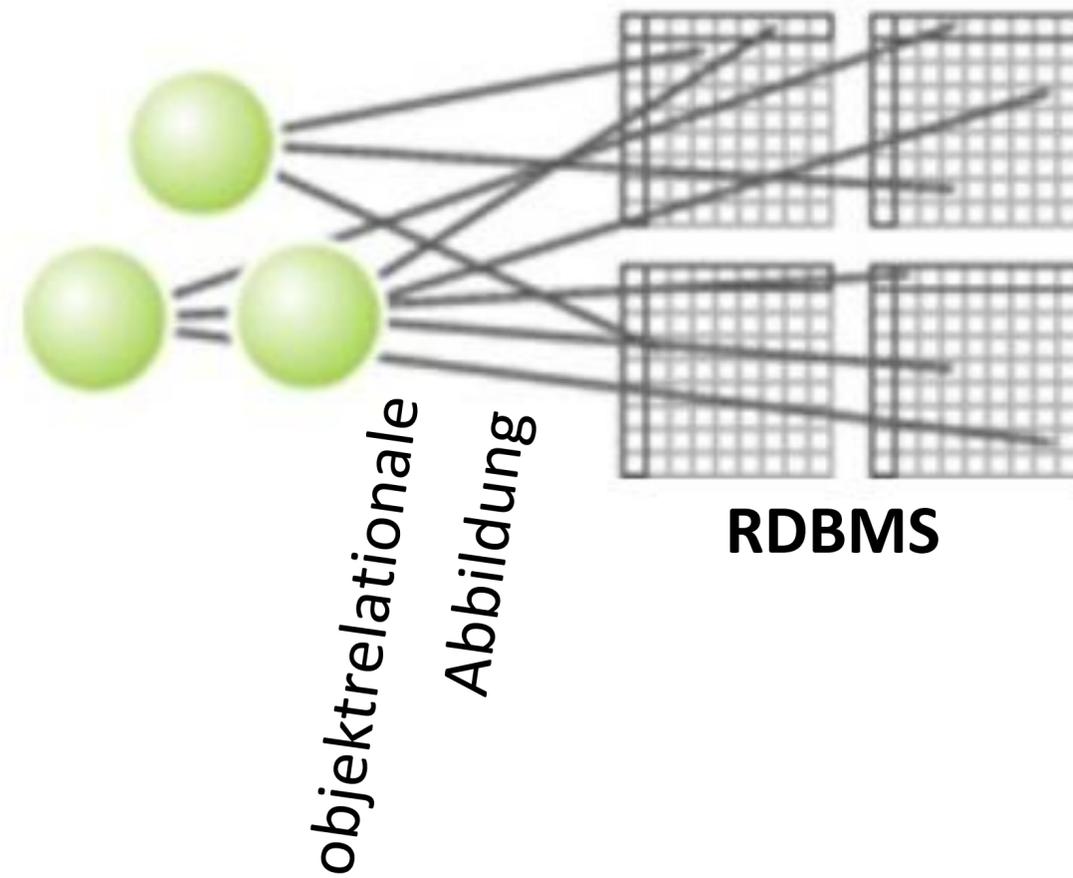
Wieso der Bezug zwischen E/R und UML?

- Die E/R-Modellierung existiert seit 1976, als sie von Peter Chen veröffentlicht wurde.
- Seitdem hatte sie sich insbesondere im Zusammenhang mit der imperativen Programmierung etabliert.
 - EVA-Konzept
 - Datenverarbeitung
- Die damit in Verbindung stehenden relationalen DBMS (RDBMS) sind sehr performant und heutzutage sehr weit verbreitet.
- Beispiele dafür sind
 - MySQL / MariaDB
 - MS SQL
 - Oracle DB

Wieso der Bezug zwischen E/R und UML?

- Die UML hat sich später etabliert, als sich komplexere, objektorientierte Konzepte durchgesetzt hatten.
- Unter diesen entstehenden objektorientierten Anwendungen verbergen sich normalerweise auch heutzutage noch relationale Datenbanken.
- Es gibt zwar auch Objektdatenbankmanagementsysteme (ODBMS).
- Damit in Verbindung wurden
 - die Object Query Language (OQL) sowie die
 - Object Definition Language (ODL) als Datenmanipulationsprache definiert.
- Die Performance solcher Systeme ist jedoch noch problematisch.
- Beispiele für ODBMS sind
 - db4o sowie
 - Zope.

Wieso der Bezug zwischen E/R und UML?



Wieso der Bezug zwischen E/R und UML?

- Mit der Objektorientierung hat sich der gesamte Software-Entwicklungsprozess verlagert:
 - Früher stand die „Denkweise“ der Maschine im Vordergrund, Stichwort: Datenverarbeitung
 - Heute modelliert man einen Ausschnitt aus der realen Welt, vgl. Software-Engineering 1
- Diese Modellierung erfolgt heutzutage unter Verwendung von verschiedenen UML-Diagrammtypen.
 - Use Cases für die Anwendungsfälle bzw. für die Funktionalitäten, die ein Benutzer in seiner Rolle gerne hätte.
 - Aktivitätsdiagramme zur Modellierung von (geschäftlichen) Abläufen für die Fachlogik.
 - Klassendiagramme für Eigenschaften (statt ER-Modellierung) und Funktionen/Methoden von Objekten für die Fachlogik.

Wieso der Bezug zwischen E/R und UML?

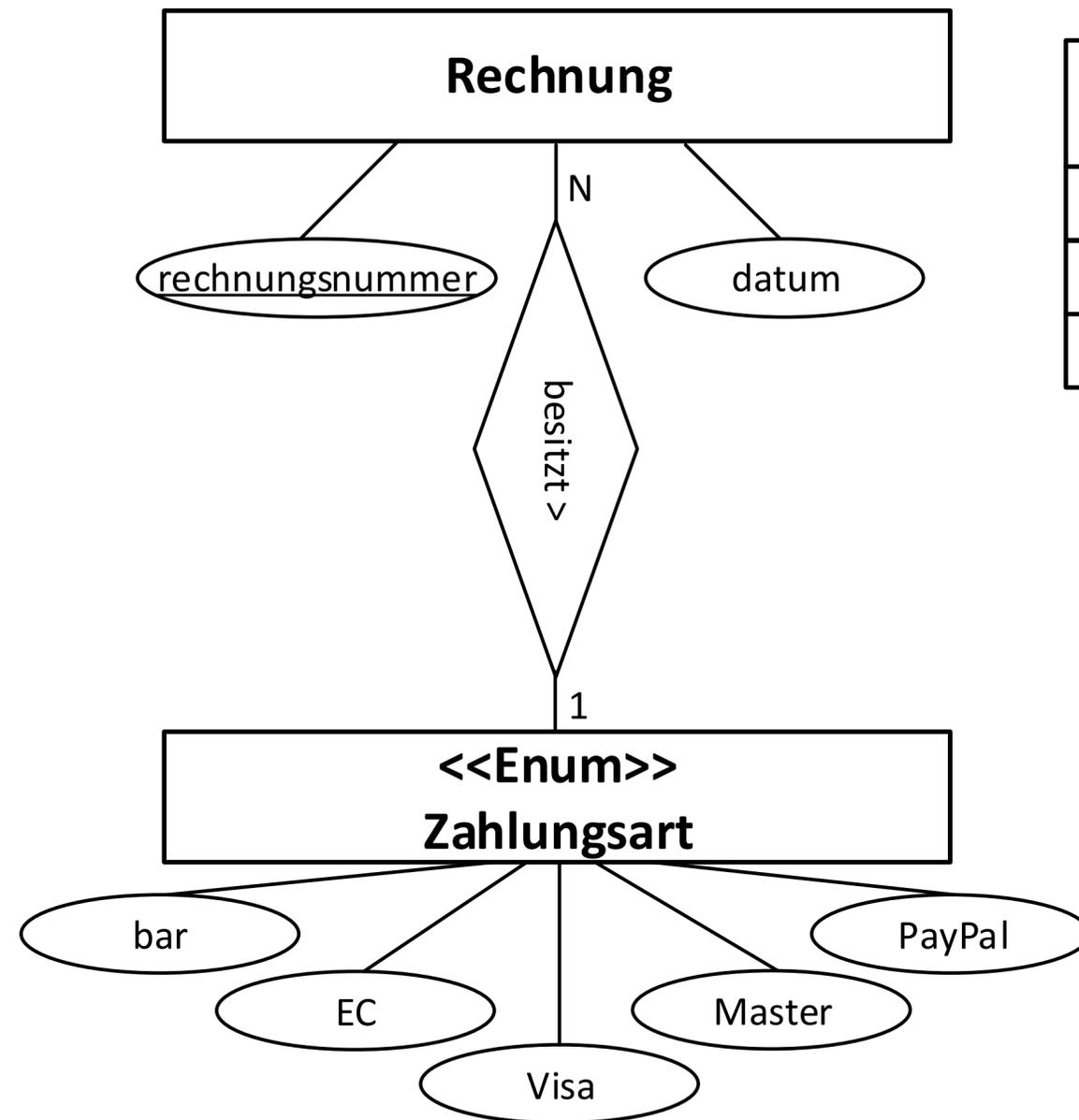
- Während die Aktivitätsdiagramme die dynamische Sicht auf das System darstellen, repräsentieren die Klassendiagramme die statische Sicht auf die Klassen und deren Beziehungen zueinander.
- Mögliche Beziehungen sind
 - Assoziation: „kennt“-Beziehung
 - Aggregation/Komposition: „besteht aus“-Beziehung
 - Vererbung: „ist ein“-Beziehung
- Die Eigenschaften und die Beziehungen aus UML-Klassendiagrammen sind direkt in E/R-Diagramme überführbar und umgekehrt.
- Wieso sind die Funktionen bzw. Methoden der UML-Klassendiagramme nicht überführbar?

Betrachtung einer einzelnen Klasse

- Eine UML Klasse entspricht einer relationalen Tabelle, einer Entitätsklasse bzw. einem Entitätstyp.
- Eine Eigenschaft bzw. ein Attribut einer UML Klasse in Form eines primitiven Typs oder eines Strings, BigDecimal, Date entspricht einem Attribut bzw. einer Spalte der Tabelle.
 - Die Datentypen der Programmiersprache sind dabei in SQL-Datentypen zu überführen.
- Eine Eigenschaft bzw. ein Attribut einer UML Klasse in Form einer Referenz wird im relationalen Datenmodell i.d.R. als Fremdschlüssel realisiert.
- Eine Tabellenzeile in einer relationalen Datenbank ist ein Datensatz. Ein Datensatz entspricht i.d.R. genau einem Objekt dieser Klasse.

Betrachtung einer einzelnen Klasse: Mehrwertige Attribute

- Mehrwertige Attribute können im relationalen Modell durch eine neue Tabelle dargestellt werden...

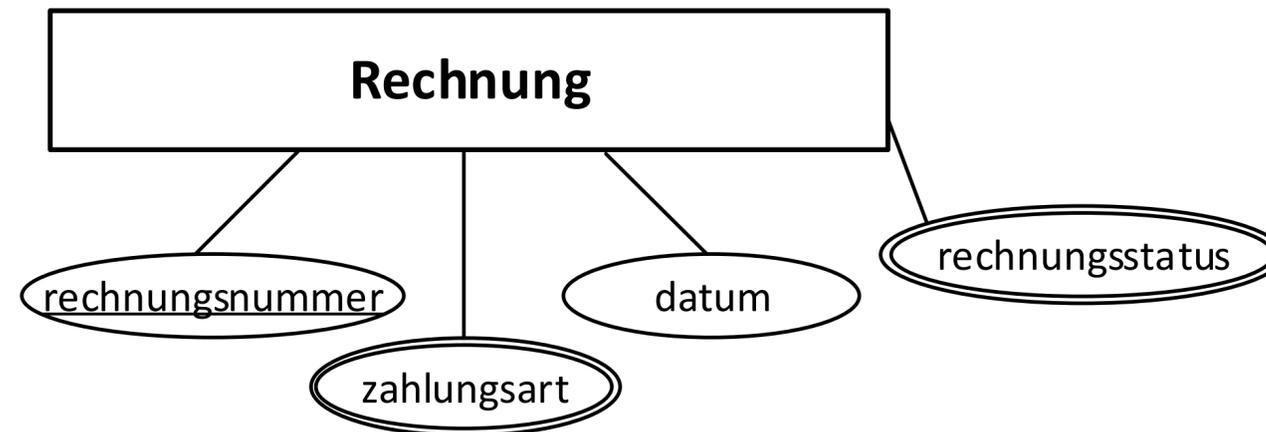


Entitätstyp Rechnung			
r.-nr.	datum	zahlungsart	...
23	23.01.2025	3	...
24	24.01.2025	NULL	...

Entitätstyp Zahlungsart	
ID	Bezeichnung
1	bar
2	EC
3	Visa
4	Master
5	PayPal

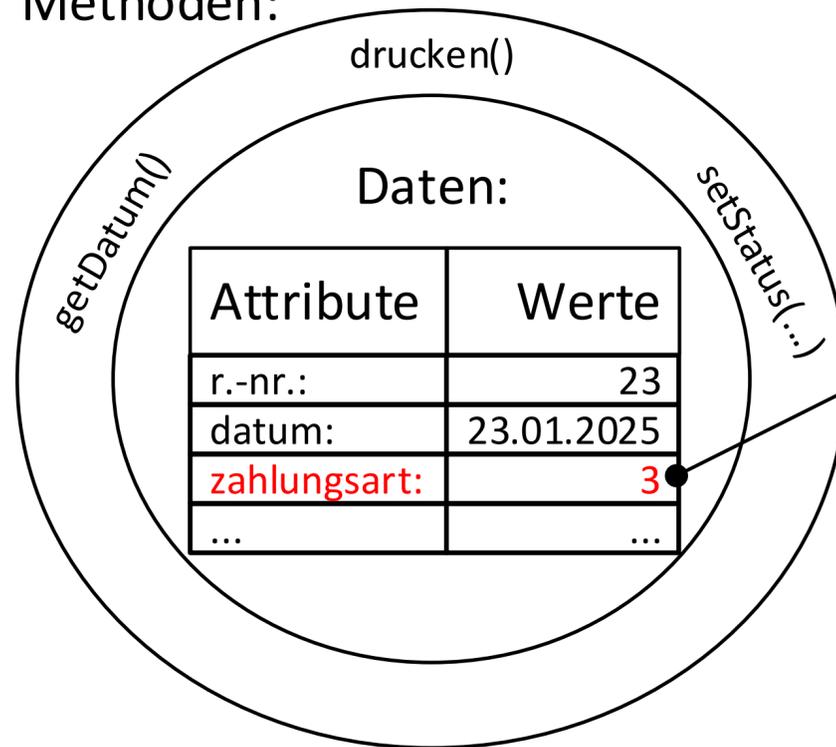
Betrachtung einer einzelnen Klasse: Mehrwertige Attribute

- Alternativ in Java & MySQL/MariaDB: Eigener Datentyp „Enum“!

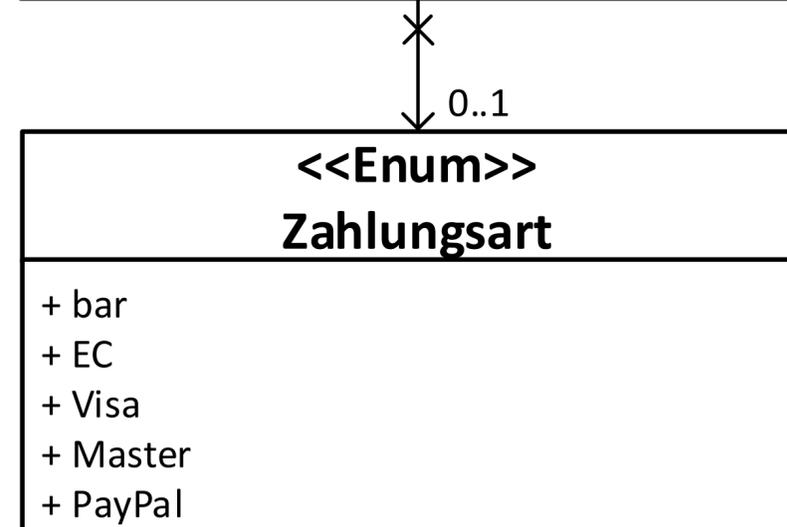
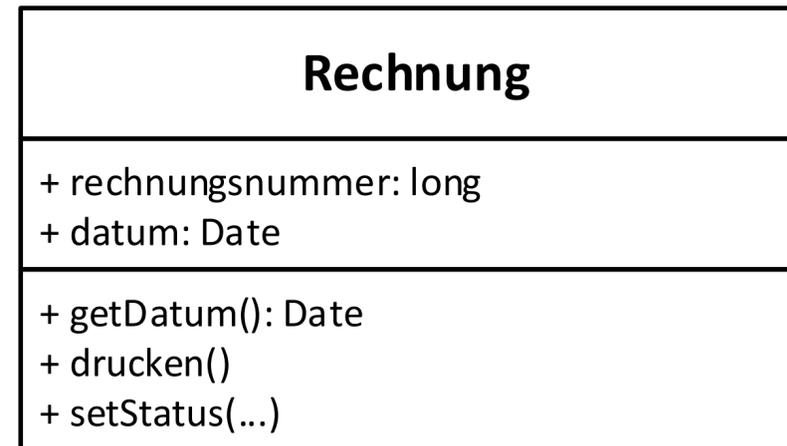


Objekt vs. Datenbank-Tabelle

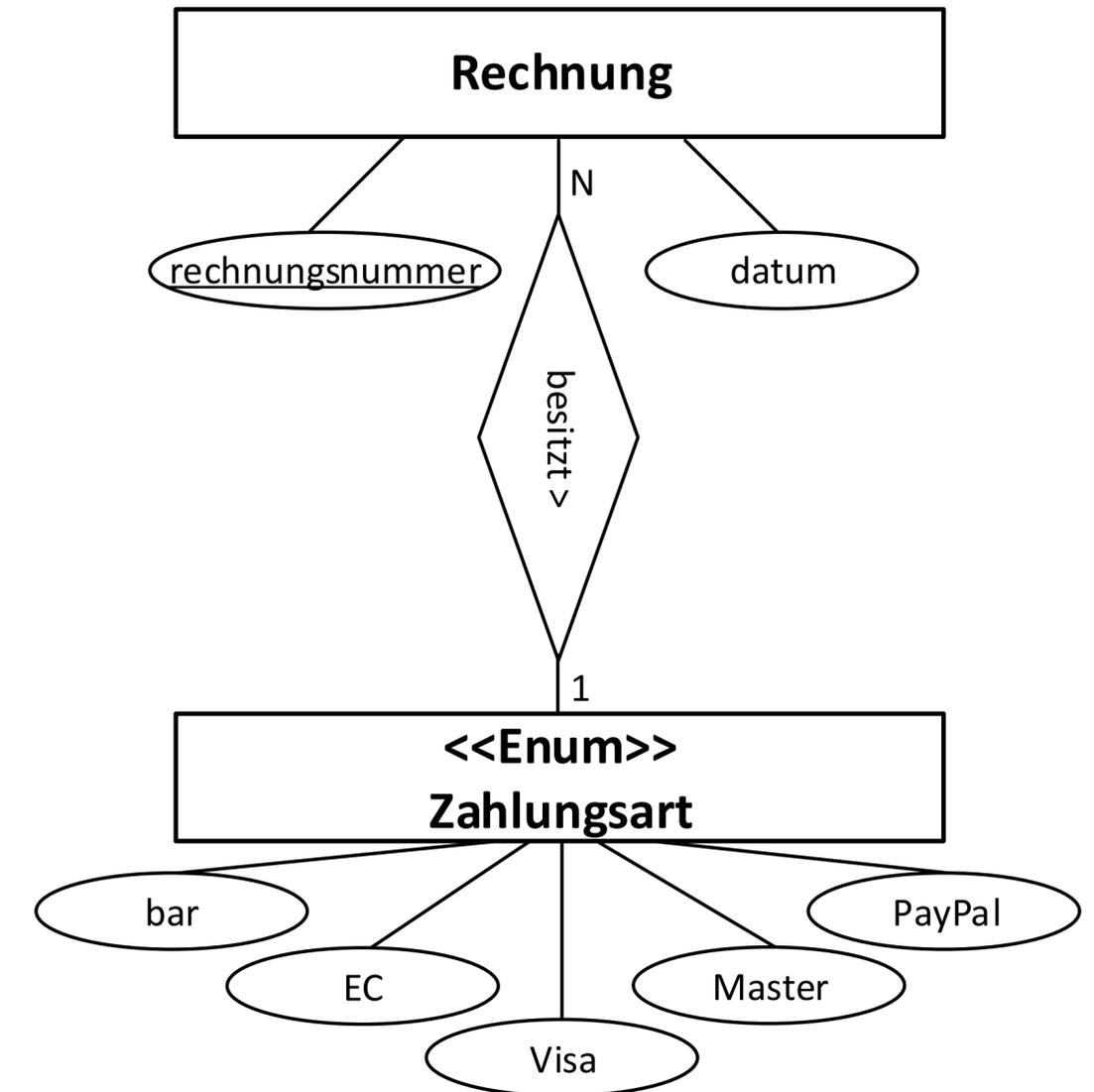
Methoden:



Objektreferenz...



* 0..1



Objekt vs. Datenbank-Tabelle

Methoden:

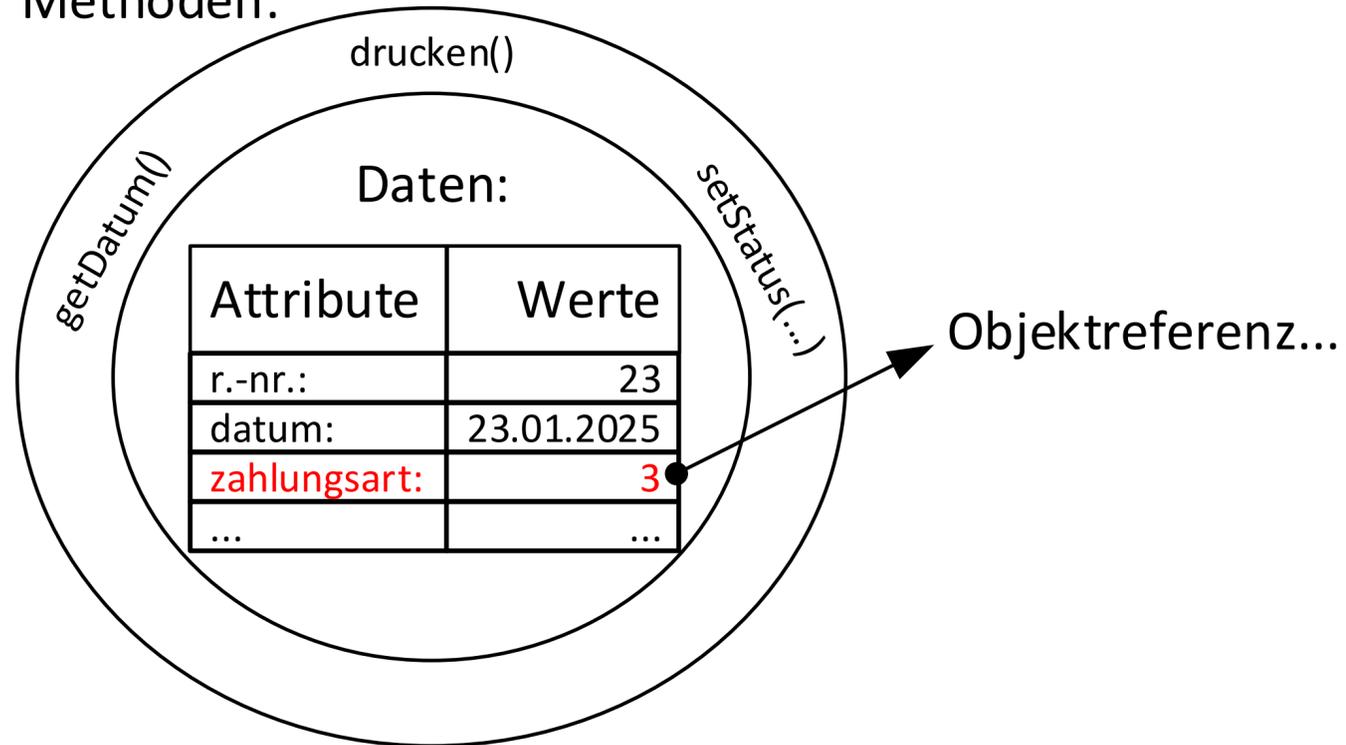


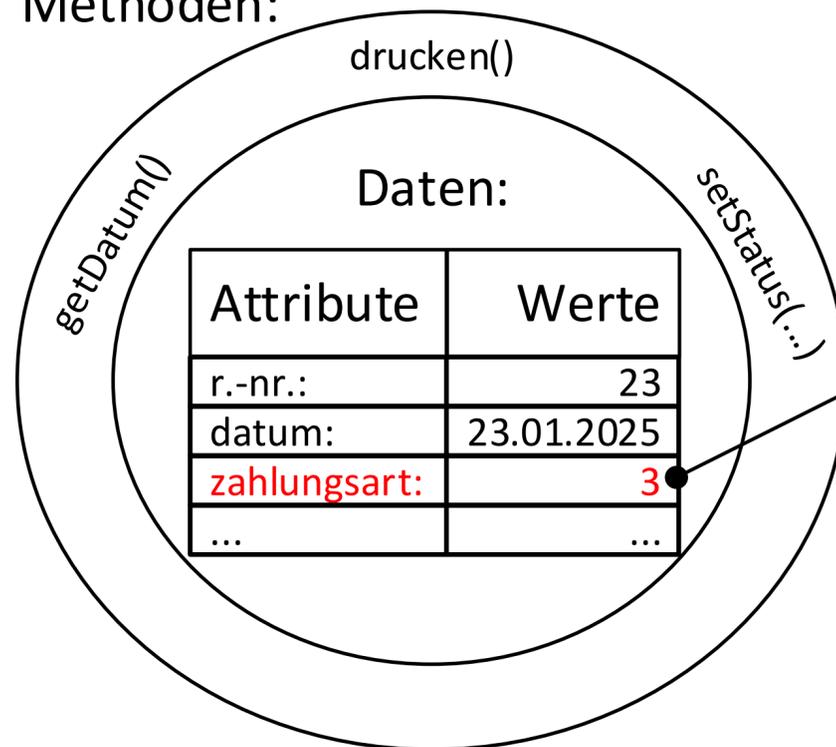
Tabelle Rechnung			
r.-nr.	datum	zahlungsart	...
23	23.01.2025	3	...
24	24.01.2025	NULL	...

Tabelle Zahlungsart	
ID	Bezeichnung
1	bar
2	EC
3	Visa
4	Master
5	PayPal

Umsetzung der Assoziation: „kennt“-Beziehung 1:N

- Im Beispiel ist auch bereits eine Assoziation enthalten:
 - Die Rechnung kennt ihre Zahlungsart.
 - Jede konkrete Zahlungsart ist ein Objekt der Klasse Zahlungsart.

Methoden:



Objektreferenz...

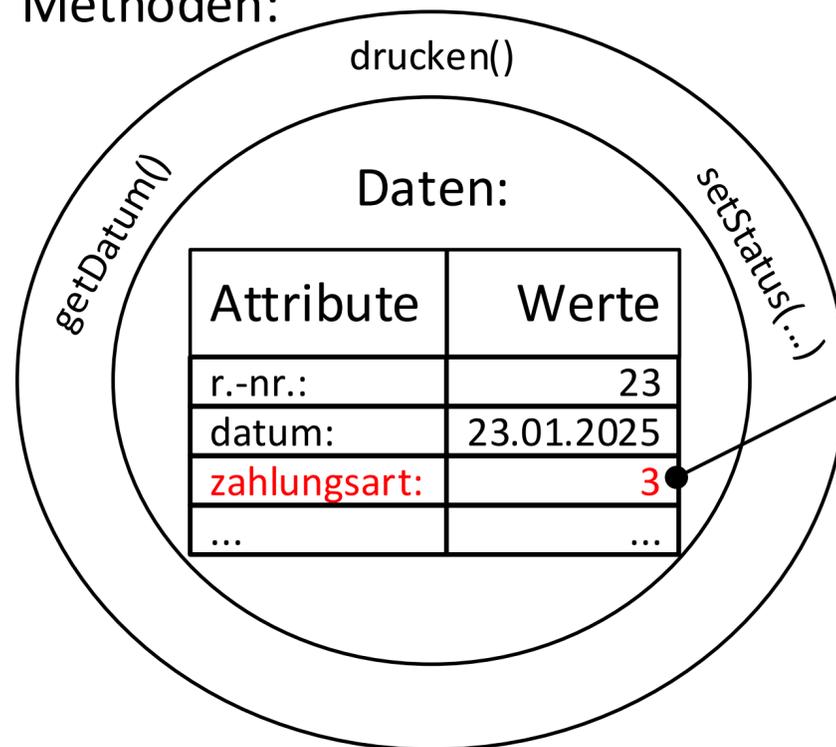
Tabelle Rechnung			
r.-nr.	datum	zahlungsart	...
23	23.01.2025	3	...
24	24.01.2025	NULL	...

Tabelle Zahlungsart	
ID	Bezeichnung
1	bar
2	EC
3	Visa
4	Master
5	PayPal

Umsetzung der Assoziation: „kennt“-Beziehung 1:N

- Aus Sicht des relationalen Modells wird diese Referenz über einen Primärschlüssel und einen Fremdschlüssel realisiert.
 - Was würde passieren, wenn man diese Rechnung löscht?
 - Was würde passieren, wenn man diese Zahlungsart löscht?

Methoden:



Objektreferenz...

Tabelle Rechnung			
r.-nr.	datum	zahlungstyp	...
23	23.01.2025	3	...
24	24.01.2025	NULL	...

Tabelle Zahlungsart	
ID	Bezeichnung
1	bar
2	EC
3	Visa
4	Master
5	PayPal

Primärschlüssel
in Zahlungsart

Fremdschlüssel
in der Rechnung

Umsetzung der Assoziation: „kennt“-Beziehung 1:1

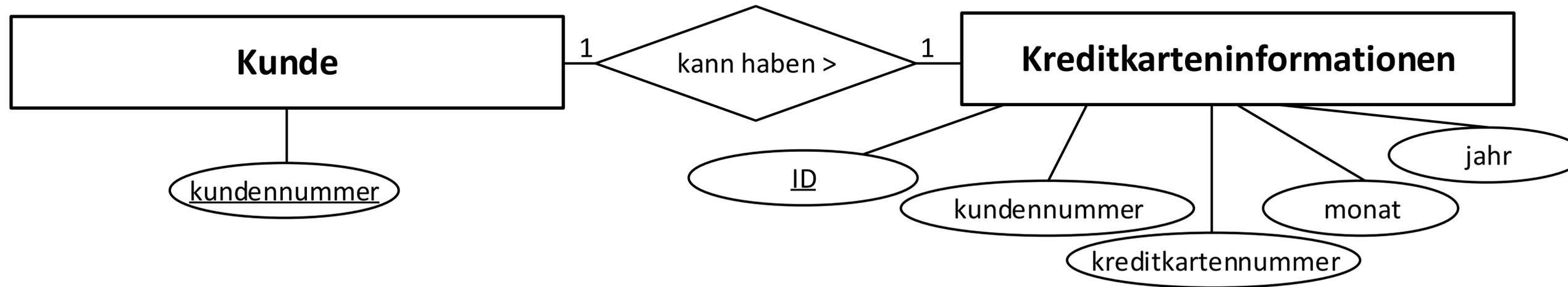


Tabelle Kunde		
<u>kundennummer</u>	...	id_kredit
12	...	NULL
15	...	17

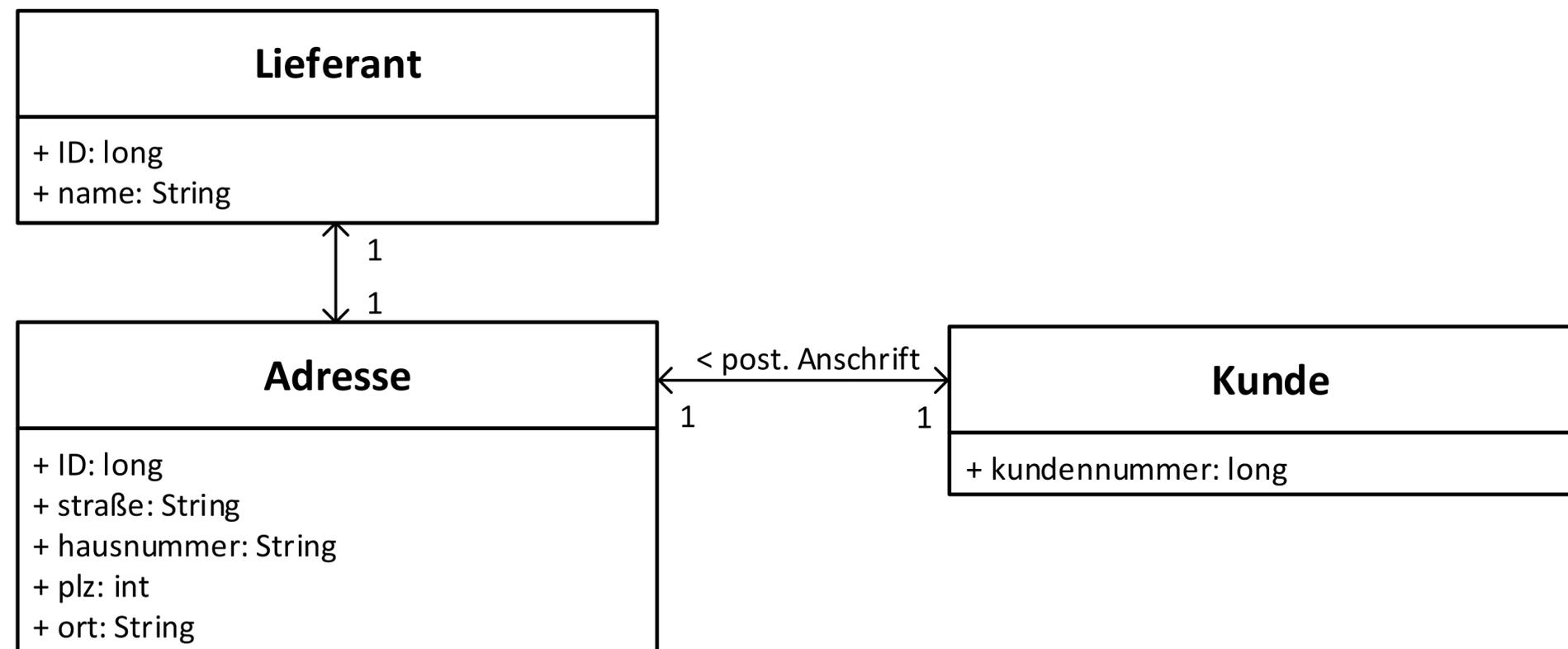
- Kunde:
 - kundennummer ist Primärschlüssel
 - id_kredit ist Fremdschlüssel und Unique

Tabelle Kreditkarteninformationen				
<u>id</u>	kundennummer	kreditkartennummer	monat	jahr
...
17	15

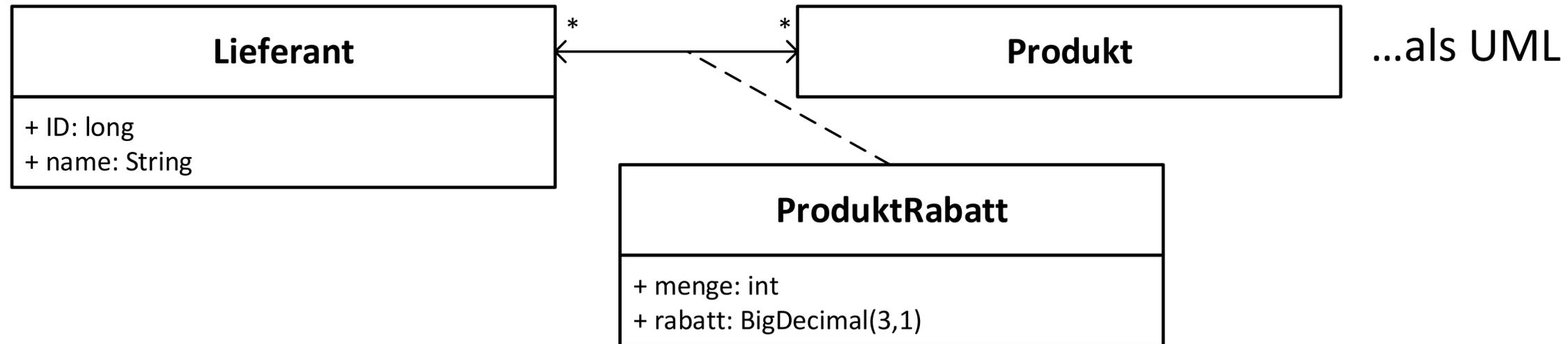
- Kreditkarteninformationen:
 - id ist Primärschlüssel
 - kundennummer ist Fremdschlüssel und Unique
 - kreditkartennummer ist Unique

Umsetzung der Assoziation: „kennt“-Beziehung 1:1 komplexer

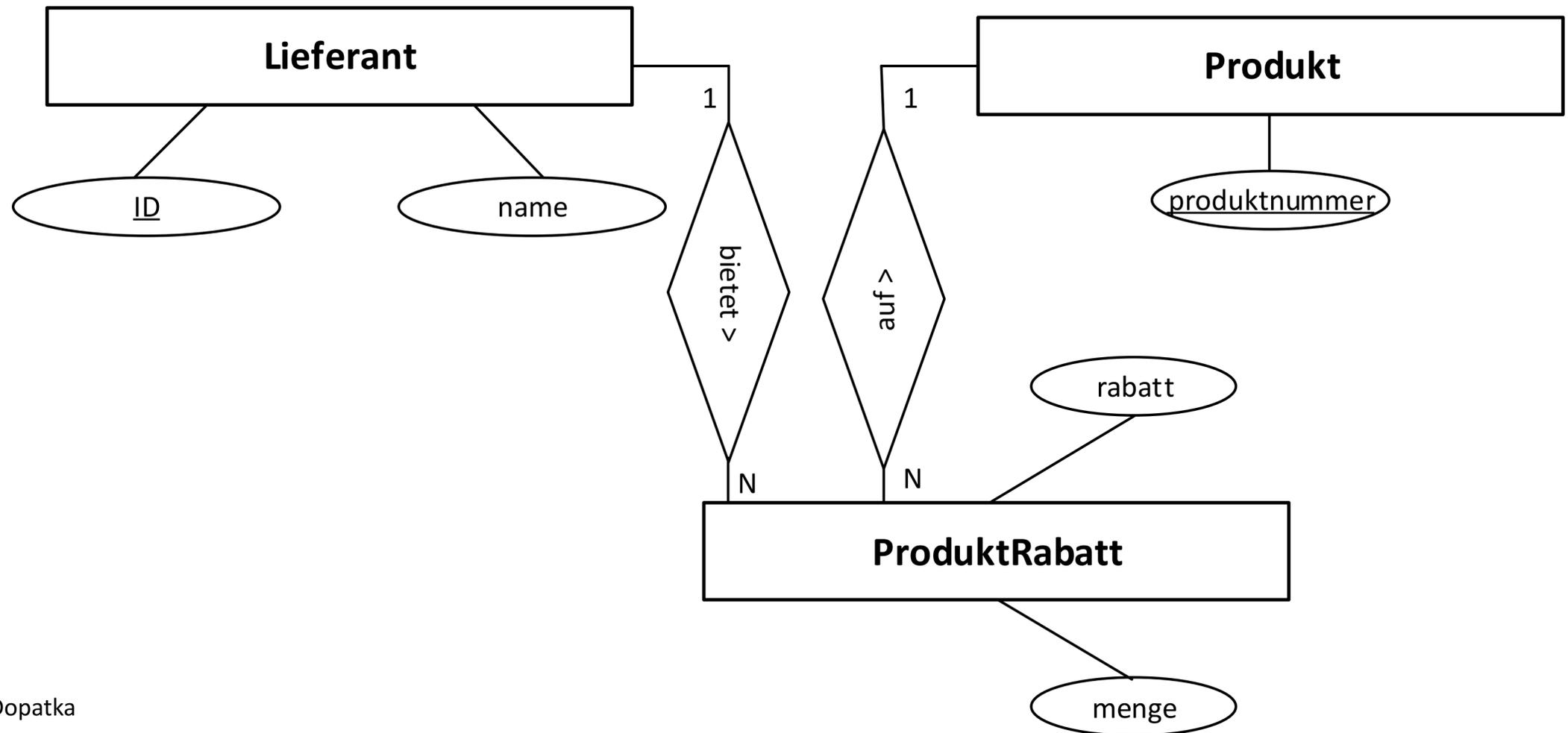
- Der Eintrag (die Zeile, die Entität) in den Tabellen Kunde und Lieferant haben jeweils eine Referenz auf eine Adress-ID.
- Wenn die Adresse auch den zugehörigen Lieferanten oder Kunden kennen soll, so müssen 2 separate Spalten id_kunde und id_lieferant für die Fremdschlüssel-Beziehung erstellt werden, die entweder/oder ausgefüllt werden.
 - Eine passende Logik dazu ist ggf. nur über eine Stored Procedure realisierbar!



Umsetzung der Assoziation: „kennt“-Beziehung N:M



Als ERD mit aufgeschlüsselter
N:M-Beziehung:

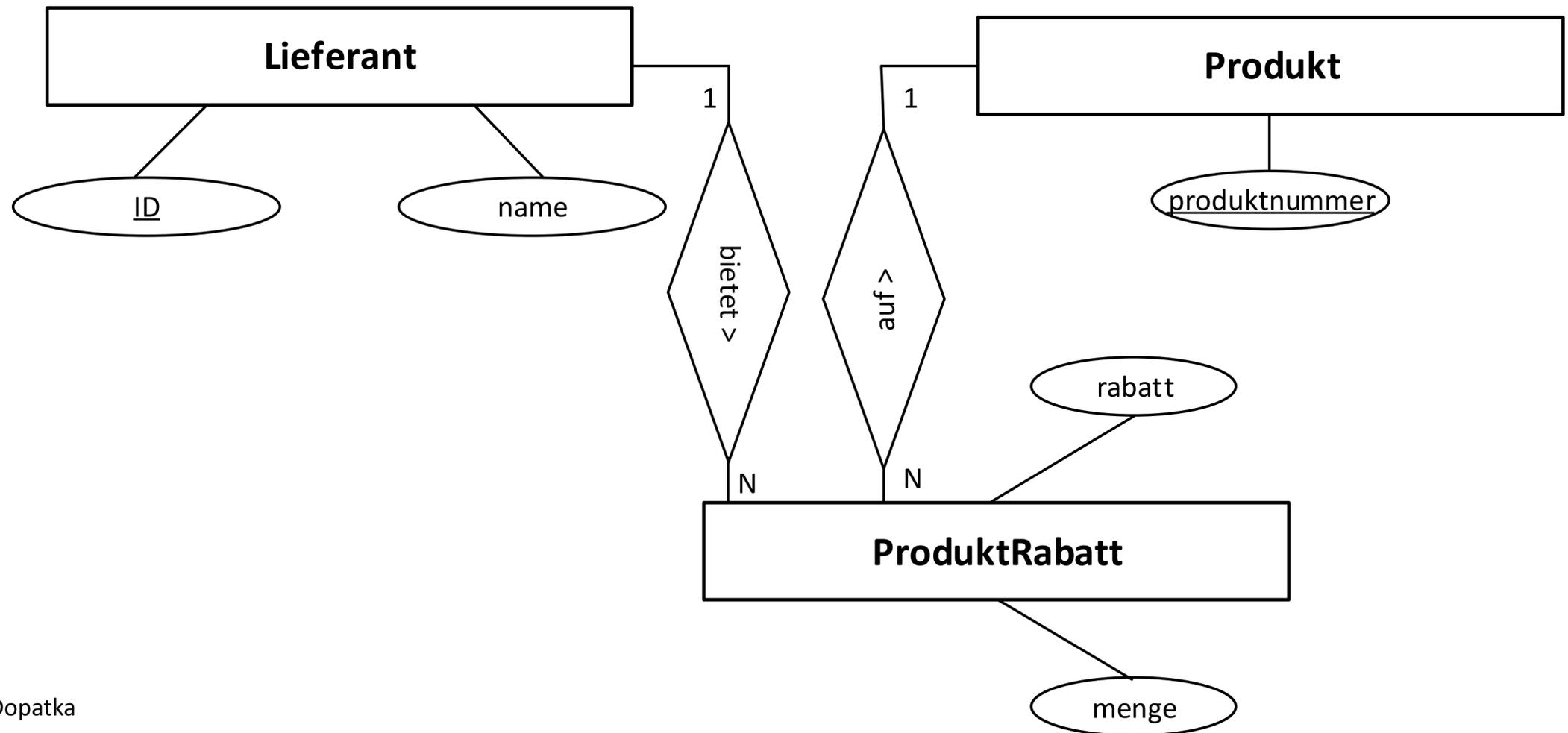


Umsetzung der Assoziation: „kennt“-Beziehung N:M

Tabelle Lieferant	
<u>id</u>	name
453	Schmidt
356	Schulz

Tabelle ProduktRabatt			
<u>id_lieferant</u>	<u>id_produk</u>	menge	rabatt (%)
453	4567	1000	20
356	5644	500	10
356	4567	30	5

Tabelle Produkt	
<u>produktnummer</u>	...
5644	...
4567	...



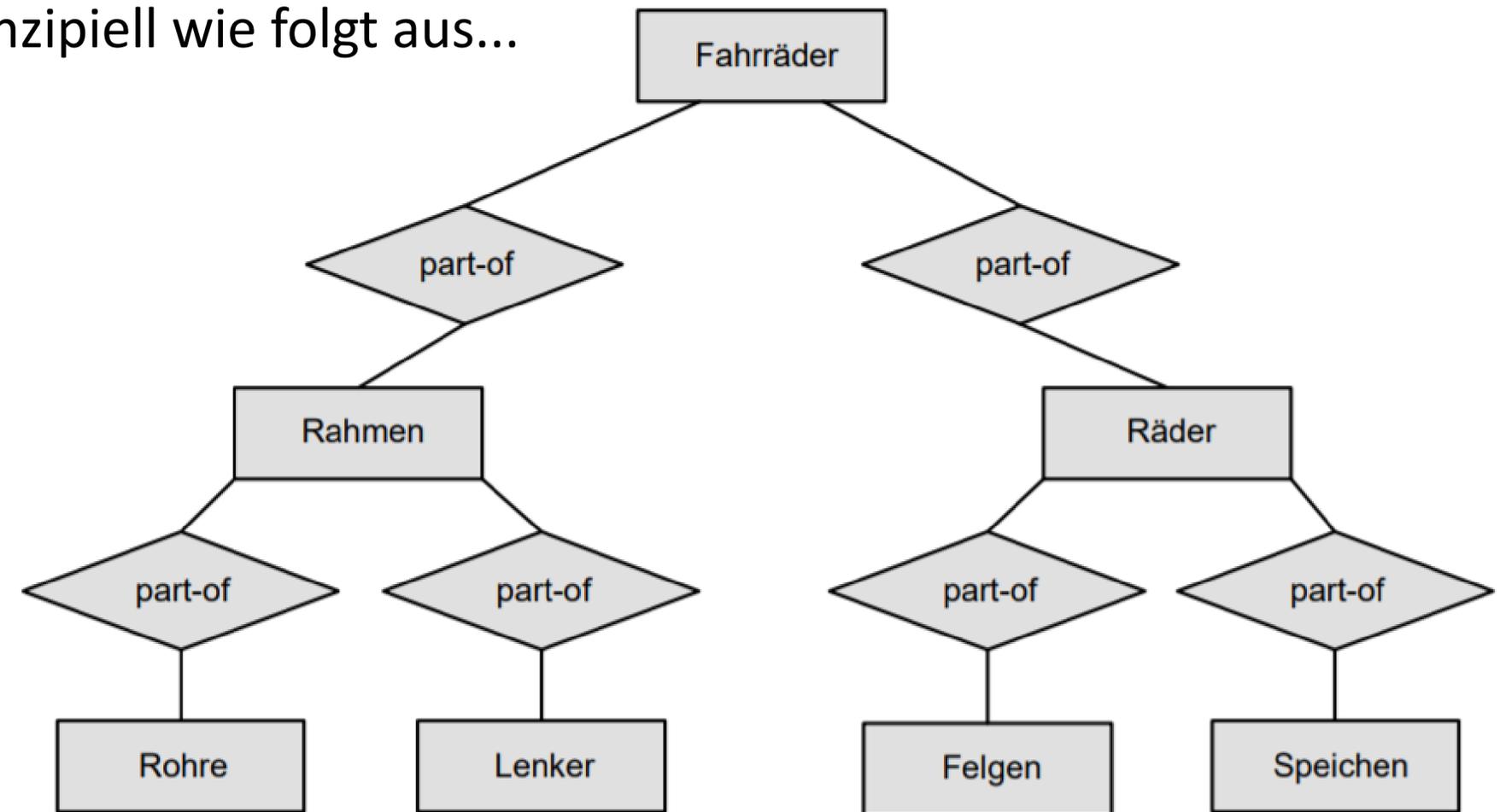
Umsetzung der Aggregation/Komposition: „besteht aus“-Beziehung

- Ein Ganzes besteht aus seinen Teilen...
 - Können die Teile auch unabhängig existieren von einem Ganzen und ggf. zu mehreren Ganzen gehören, so spricht man von einer Aggregation.
 - Kann das Teil nicht selbständig ohne sein Ganzes existieren, so spricht man von einer Komposition.
 - Wie Sie eine Aggregation/Komposition in Java umsetzen, kennen Sie aus der Veranstaltung „Programmierung 2“.
- Eine Aggregation/Komposition sieht im UML Klassendiagramm wie folgt aus:



Umsetzung der Aggregation/Komposition: „besteht aus“-Beziehung

- Nehmen Sie folgende Aufgabenstellung an:
 - Jedes Fahrrad besteht aus einem Rahmen und aus Rädern.
 - Ein Rahmen besteht aus Rohren und aus einem Lenker.
 - Räder bestehen aus Felgen und aus Speichen.
- In einem E/R-Diagramm sieht dies prinzipiell wie folgt aus...



Umsetzung der Komposition: „besteht aus“-Beziehung

- Die Komposition wird in einem relationalen Datenmodell prinzipiell genauso gehandhabt wie eine Aggregation.
- Um die Integrität der Daten sicherzustellen, muss man dafür sorgen, dass alle Teile auch gelöscht werden, wenn das dazugehörige Ganze in der Datenbank gelöscht wird.

Umsetzung der Komposition in unserem Beispiel...

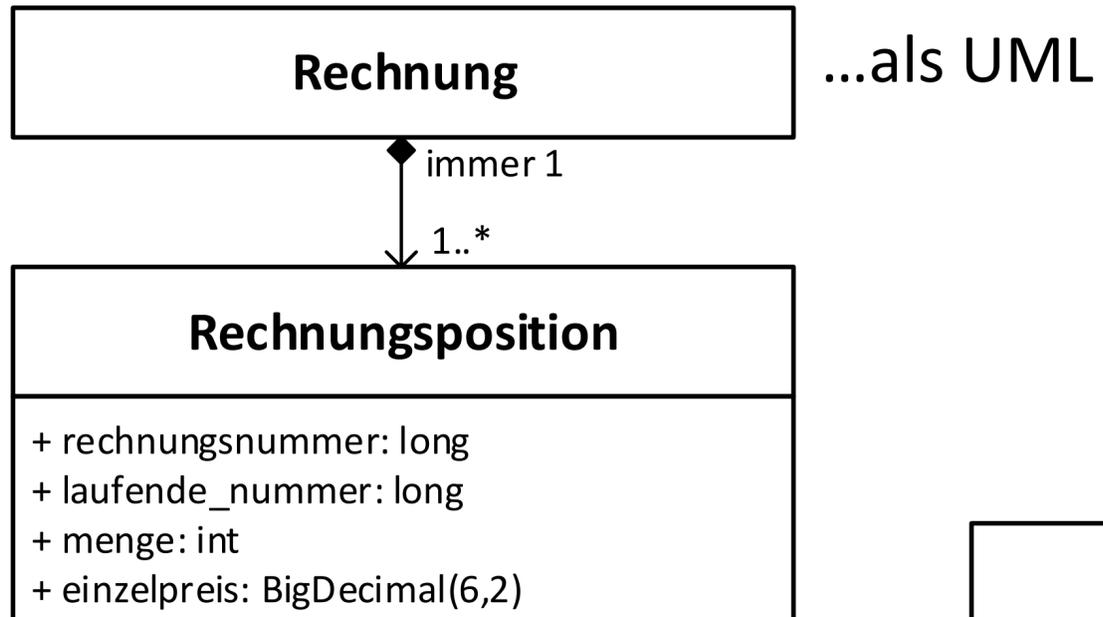


Tabelle Rechnung	
<u>rechnungsnummer</u>	...
56	...
57	...

Als ERD:

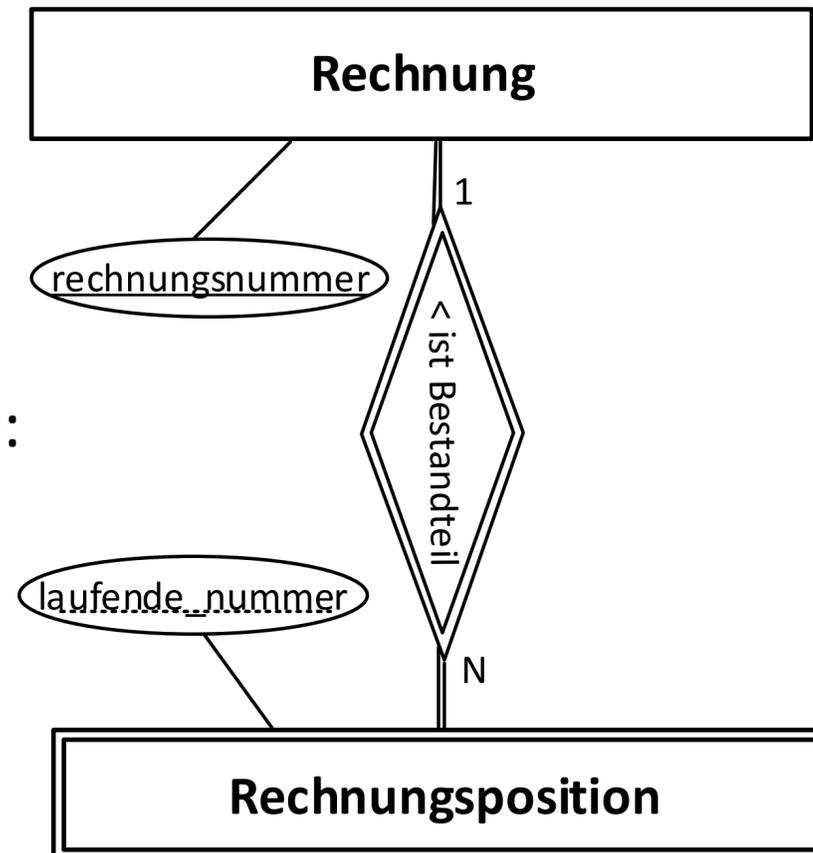


Tabelle Rechnungsposition		
<u>rechnungsnummer</u>	<u>laufende_nummer</u>	...
56	1	...
56	2	...
56	3	...
57	1	...

Umsetzung der Vererbung: „Ist ein“-Beziehung

- „Ist ein“-Beziehungen drücken Spezialisierungen bzw. Generalisierungen aus, eine Vererbung der Objektorientierung!
- Dabei gilt, dass der Spezialentitätstyp alle Attribute des allgemeinen Entitätstyps erbt.
- Aus diesem Grund findet sich bei diesen Beziehungen vom Typ 1:1 an dem speziellen Entitätstyp keine Schlüsselattributangabe.
- „Ist ein“-Beziehungen lassen sich auflösen und in eine Tabelle integrieren.
- Hierzu muss nur der Primärschlüssel des allgemeineren Entitätstyps
 - entweder als Fremdschlüssel (bei 1:n)
 - oder als Primärschlüssel (bei 1:1)in die Tabelle des speziellen Entitätstyps übernommen werden.

Umsetzung der Vererbung in unserem Beispiel...

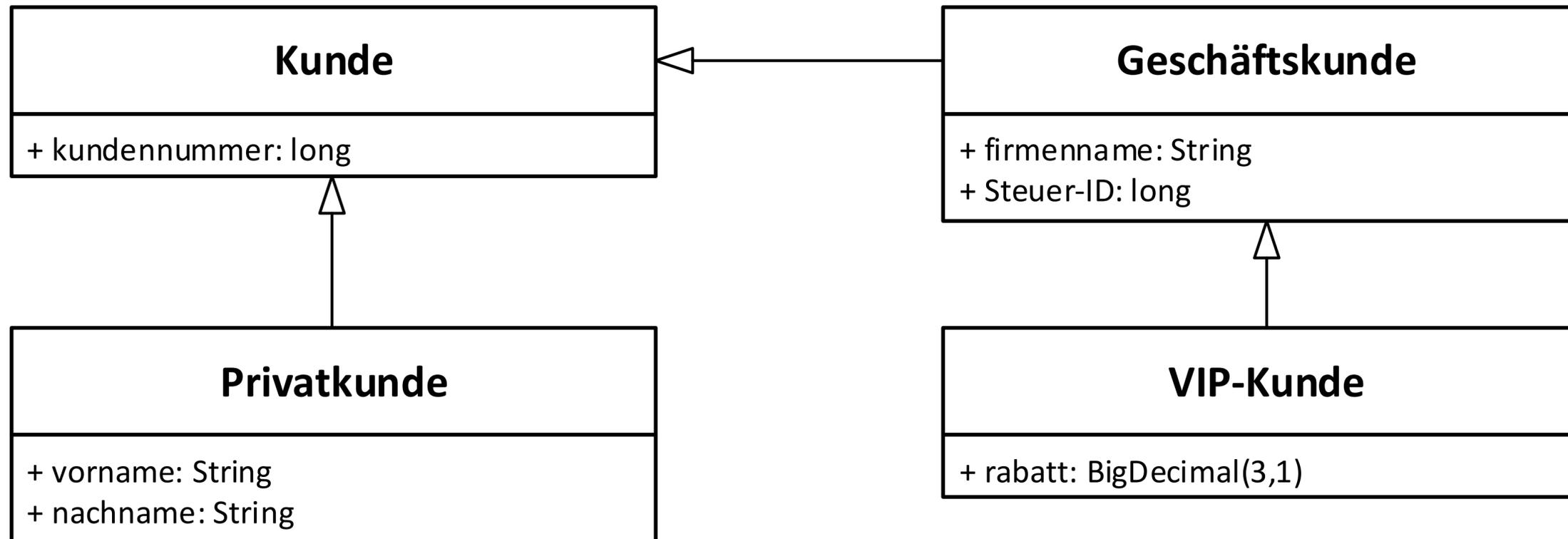


Tabelle Kunde	
<u>kundenummer</u>	...
23	...
24	...
25	...
26	...

Tabelle Privatkunde		
<u>kundenummer</u>	vorname	nachname
23	Hans	Wurst
24	Clair	Grube
25	Anna	Bolika
26	Peter	Silie

Umsetzung der Vererbung in unserem Beispiel...

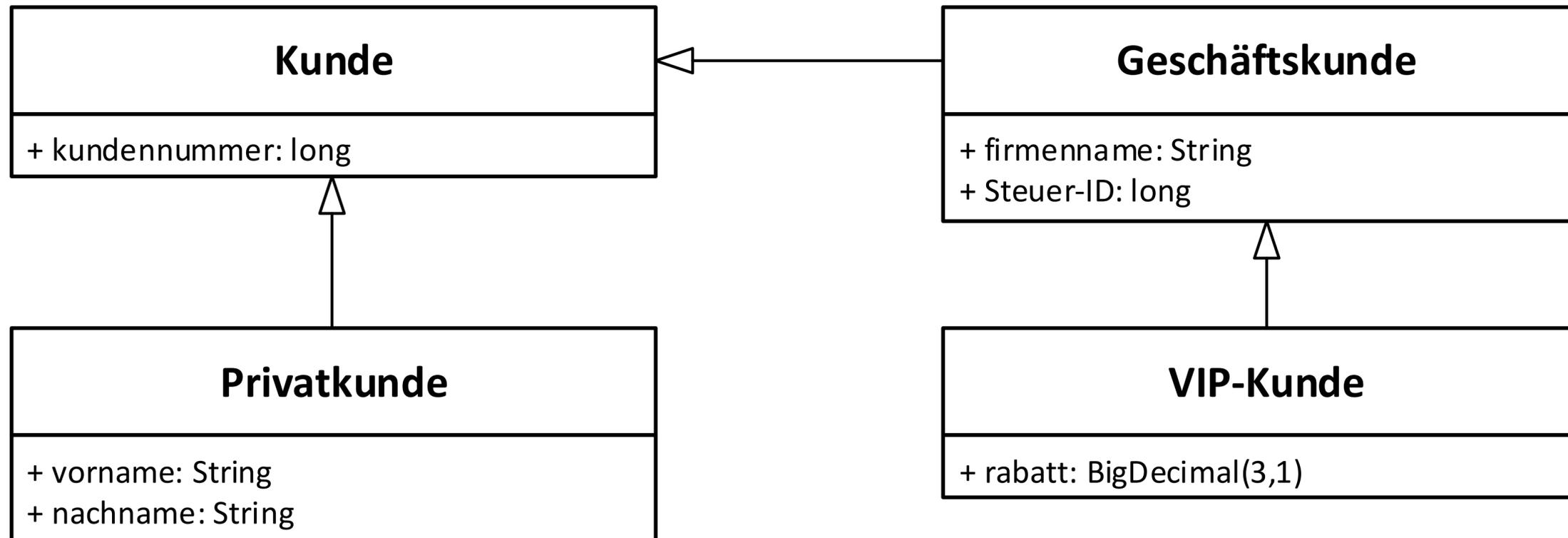
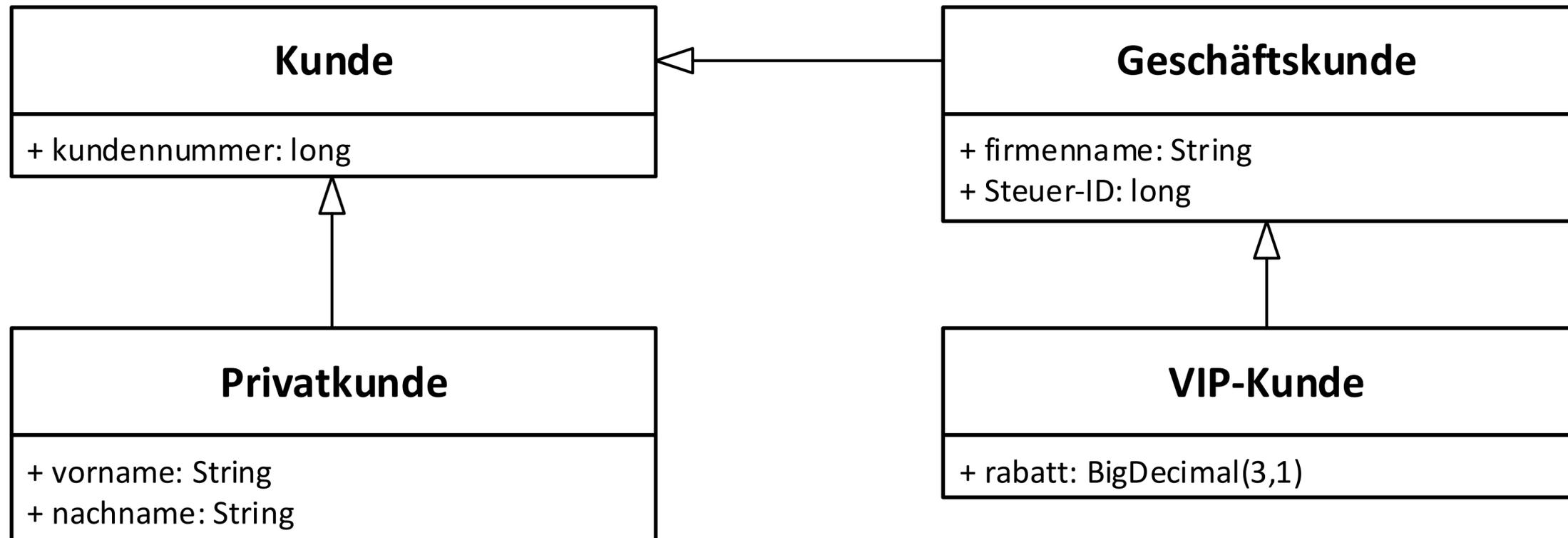


Tabelle Kunde	
<u>kundennummer</u>	...
27	...
28	...
29	...
30	...

Tabelle Geschäftskunde		
<u>kundennummer</u>	firmenname	steuer-id
27	Bäckerei Kotzbäck	54634535532
28	Fleischerei Kill	65466743554
29	Wekrstatt Schaden	45754745445
30	Hairforce One	45745342357

Tabelle VIP-Kunde	
<u>kundennummer</u>	rabatt
29	12,3

Umsetzung der Vererbung in unserem Beispiel...



- Noch sauberer bei Mehrfachvererbung:
 - Geschäftskunde hat neuen zusammengesetzten Primärschlüssel, bestehend aus lfd. Nummer und Kundennummer
 - Geschäftskunde ist schwache Entität gegen Kunde, kann allein nicht existieren
 - VIP-Kunde hat neuen zusammengesetzten Primärschlüssel, bestehend aus einer weiteren lfd. Nummer, der lfd. Nummer des Geschäftskunden und der Kundennummer
 - VIP-Kunde ist schwache Entität gegen Geschäftskunde, kann allein nicht existieren