



Inhalt:

- *Einführung*
- *Mengenlehre*
- *ANSI-SPARC-Modell*
- *Anforderungsanalyse*
- *Konzeptueller Entwurf*
- *Logischer Entwurf*



Inhalt:

- Einführung
- Mengenlehre
- ANSI-SPARC-Modell
- Anforderungsanalyse
- Konzeptueller Entwurf
- Logischer Entwurf

- Definitionen
- Modellierung
- Datenhaltung
- Architektur
- Datenbankbenutzer
- Entwicklung
- Arten
- Prinzipien von E.F.Codd



Charakteristik der Informationen im Unternehmen:

- *Informationen bilden Entscheidungsgrundlagen.*
- *Informationen können aus unterschiedlichen Quellen stammen.*
- *Qualität der Information ist von Verfügbarkeit, Korrektheit und Vollständigkeit abhängig.*
- *Erhebung, Speichern und Verarbeiten der Daten erzeugt Aufwände.*
- *Aufgabengebiete im Unternehmen sind durch Informationsbeziehungen miteinander verknüpft.*

Verschieden Aspekte des Datenmanagements:

- *Architektur (Datenmodellierung).*
- *Datentechnik (Hardware, Installation, Reorganisation, Sicherung).*
- *Administration und Datennutzung.*



Anforderungen an Datenverwaltung:

- *zentrale Verwaltung der Daten;*
- *Vermeidung oder Einschränkung der Redundanzen;*
- *Vermeidung von Inkonsistenzen;*
- *gemeinsame Nutzung der Daten durch verschiedene Anwendungen;*
- *Datenschutz und Datensicherung;*
- *Datenintegrität;*
- *Datenunabhängigkeit von Anwendungen.*



*Was sind die Daten und
Datenbanken überhaupt?*

*Daten = Logisch strukturierte
Informationseinheiten*

*Datenbank = Einrichtung
für langfristige sichere
Aufbewahrung von Daten*



Source Tommy Lee Walker / Shutterstock.com

Die Datenbanken sind das Herzstück jeder modernen IT-Infrastruktur.

Die Anfrage an die DB-Spezialisten ist immer noch sehr hoch.

Die DB-Fachleute werden gut bezahlt.

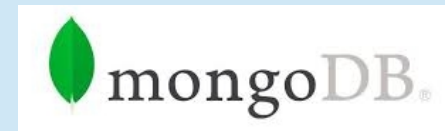
Es lohnt sich damit zu beschäftigen!



417 Systeme im Ranking, Februar 2024

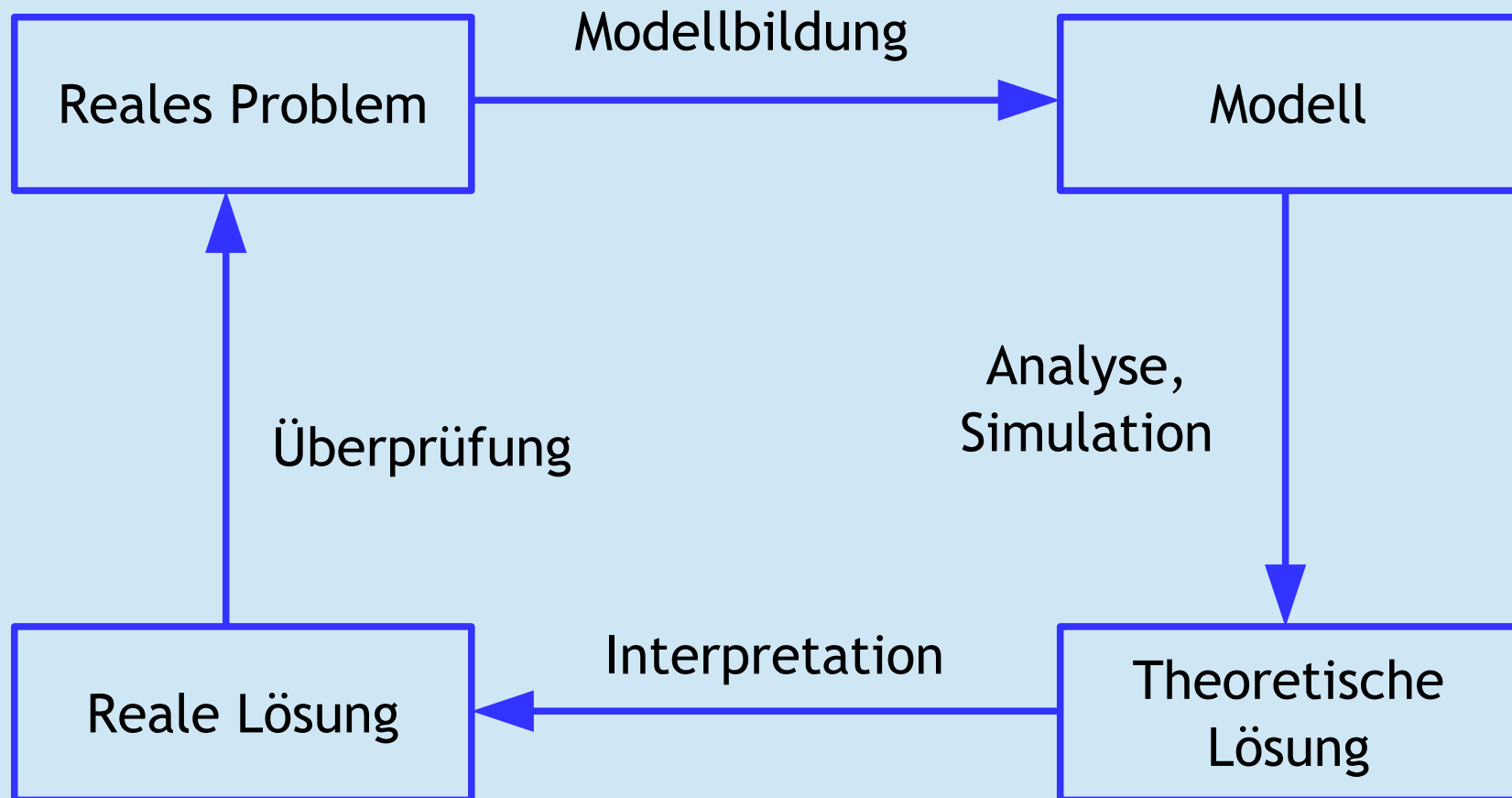
Rang			DBMS	Datenbankmodell	Punkte		
Feb 2024	Jan 2024	Feb 2023			Feb 2024	Jan 2024	Feb 2023
1.	1.	1.	Oracle +	Relational, Multi-Model ⓘ	1241,45	-6,05	-6,08
2.	2.	2.	MySQL +	Relational, Multi-Model ⓘ	1106,67	-16,79	-88,78
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-Model ⓘ	853,57	-23,03	-75,52
4.	4.	4.	PostgreSQL +	Relational, Multi-Model ⓘ	629,41	-19,55	+12,90
5.	5.	5.	MongoDB +	Document, Multi-Model ⓘ	420,36	+2,88	-32,41
6.	6.	6.	Redis +	Key-value, Multi-Model ⓘ	160,71	+1,33	-13,12
7.	7.	↑ 8.	Elasticsearch	Suchmaschine, Multi-Model ⓘ	135,74	-0,33	-2,86
8.	8.	↓ 7.	IBM Db2	Relational, Multi-Model ⓘ	132,23	-0,18	-10,74
9.	9.	↑ 12.	Snowflake +	Relational	127,45	+1,53	+11,80
10.	↑ 11.	↓ 9.	SQLite +	Relational	117,28	+2,08	-15,38

Source <https://db-engines.com/de/ranking>





Modell ist ein Abbild der realen Welt, das man mit Absicht erstellt, bestimmte Probleme zu lösen.





Laut Heinrich Hertz soll ein Modell folgende Kriterien erfüllen:

- Richtigkeit.

Die Richtigkeit von Modellen lässt sich im mathematischen Sinne nicht beweisen, sondern nur an Experimenten überprüfen und bei negativem Ausgang widerlegen. Umgekehrt folgt aus der Übereinstimmung mit experimentellen Beobachtungen allenfalls eine Art der vorläufigen Richtigkeit bis zum Beweis des Gegenteils im nächsten Experiment.

- Zulässigkeit.

Ein Modell ist logisch zulässig, wenn es auf eindeutige Weise formuliert ist und keine Widersprüche enthält.

- Zweckmäßigkeit.

Ein Modell ist zweckmäßig, wenn es keine für das behandelte Problem überflüssigen Anteile enthält, die es unnötig verkomplizieren. Ein Modell sollte so einfach wie möglich und so kompliziert wie nötig sein.



Zum Erstellen eines Modells (Modellierung) bedient man sich verschiedene Modellierungssprachen.

Das Hauptziel der Modellierung ist es, möglichst mehr Semantik aus dem Anwendungsbereich zu erfassen. Entsprechend den Modellierungssprachen gibt es mehrere Modellierungsansätze, die zu verschiedenen semantischen Datenmodellen führen.

Semantische Datenmodelle entstanden vor allem zwischen in 1970er und 1980er Jahren. Praktische Bedeutung haben die Entity Relationship Modellierung und Unified Modeling Language.

Eine gängige Vorgehensweise beim Datenbankdesign besteht darin, dass man zuerst ein ER-Modell erstellt und dann es in ein relationales Modell (relationale Datenbank) konvertiert.



Einer der wichtigsten Aspekte der Datenverarbeitung und überhaupt der ganzen IT-Branche ist die Datenhaltung.

Die Form, in der die Daten gespeichert sind, muss vielseitige Anforderungen erfüllen, weil sie den Einfluss auf die unternehmenswichtigen Prozesse der Datenverwaltung hat:

- Verarbeitung und Auswertung.*
- Sicherheit und Integrität.*
- Präsentation und Statistik.*
- Sicherung, Wiederherstellung und Replikation.*
- Import und Export.*



Grundsätzlich gibt es zwei Formen der Datenhaltung:

- *Herkömmliche Form – Dateien.*
- *Datenbanken.*

Beide Formen haben ihre berechtigte Existenz in der IT-Welt, abhängig von den Situationen, in denen die Daten verarbeitet und gespeichert werden sollen.



Der Hauptunterschied dazwischen besteht in der Datenverwaltung:

- Bei der herkömmlichen Form (Dateien) ist die Datenverwaltung in die Anwendung integriert. Das bedeutet einen enormen Aufwand bei der Software-Entwicklung. Der Datenaustausch zwischen den Anwendungen ist kaum möglich. Die Standards und Normen können dabei nur gering verwendet werden.*
- Bei den Datenbanken gehört die Datenverwaltung unbedingt dazu. Es gilt:*

Datenbank = Datenmanagement + Daten.

Ein Datenbankmanagementsystem (DBMS) sorgt für die Speicherung der Daten, Aufruf, Änderungen, Sortieren, Sicherungen. In diesem Fall werden diese Funktionen in der Anwendung nur aufgerufen und müssen nicht neu geschrieben werden. Die Standardisierung der Funktionen zusammen mit den industriellen Normen führt zu viel kleineren Kosten der Anwendung.



Vorteile und Eigenschaften der Datenbanken in Vergleich zu Dateien:

- *Reduktion der Entwicklungskosten der Anwendungen;*
- *flexible Verarbeitung und Darstellung der Daten;*
- *Vermeidung von Redundanzen, Inkonsistenzen, Datenverlust;*
- *Zugriffskontrolle;*
- *Mehrbenutzerbetrieb;*
- *hohe Verfügbarkeit.*



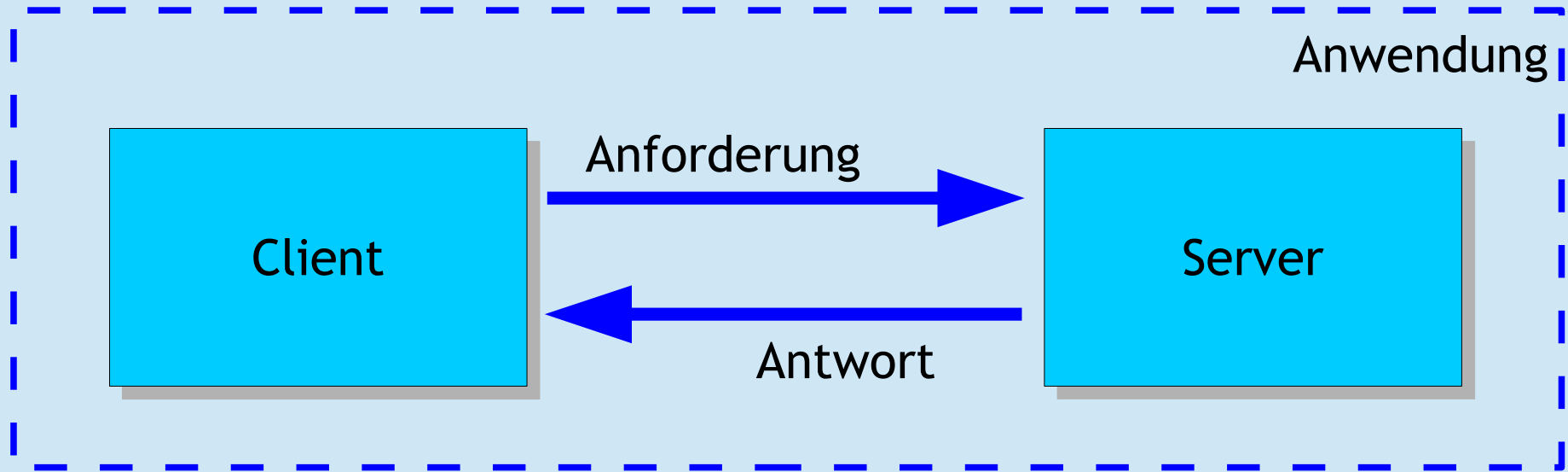
Es gibt nur wenige Situationen, wo es sich empfiehlt, die Dateien statt Datenbanken zu verwenden:

- *sehr kleine, nicht kommerzielle Anwendungen;*
- *systemnahe Anwendungen;*
- *verschiedene Testumgebungen;*
- *grobe konzeptuelle Entwicklungen.*

In allen anderen Fällen, also fast immer, sollte man schon eine passende Datenbank in Betracht ziehen.



Client/Server-Architektur



z.B. Darstellung

z.B. Bearbeitung

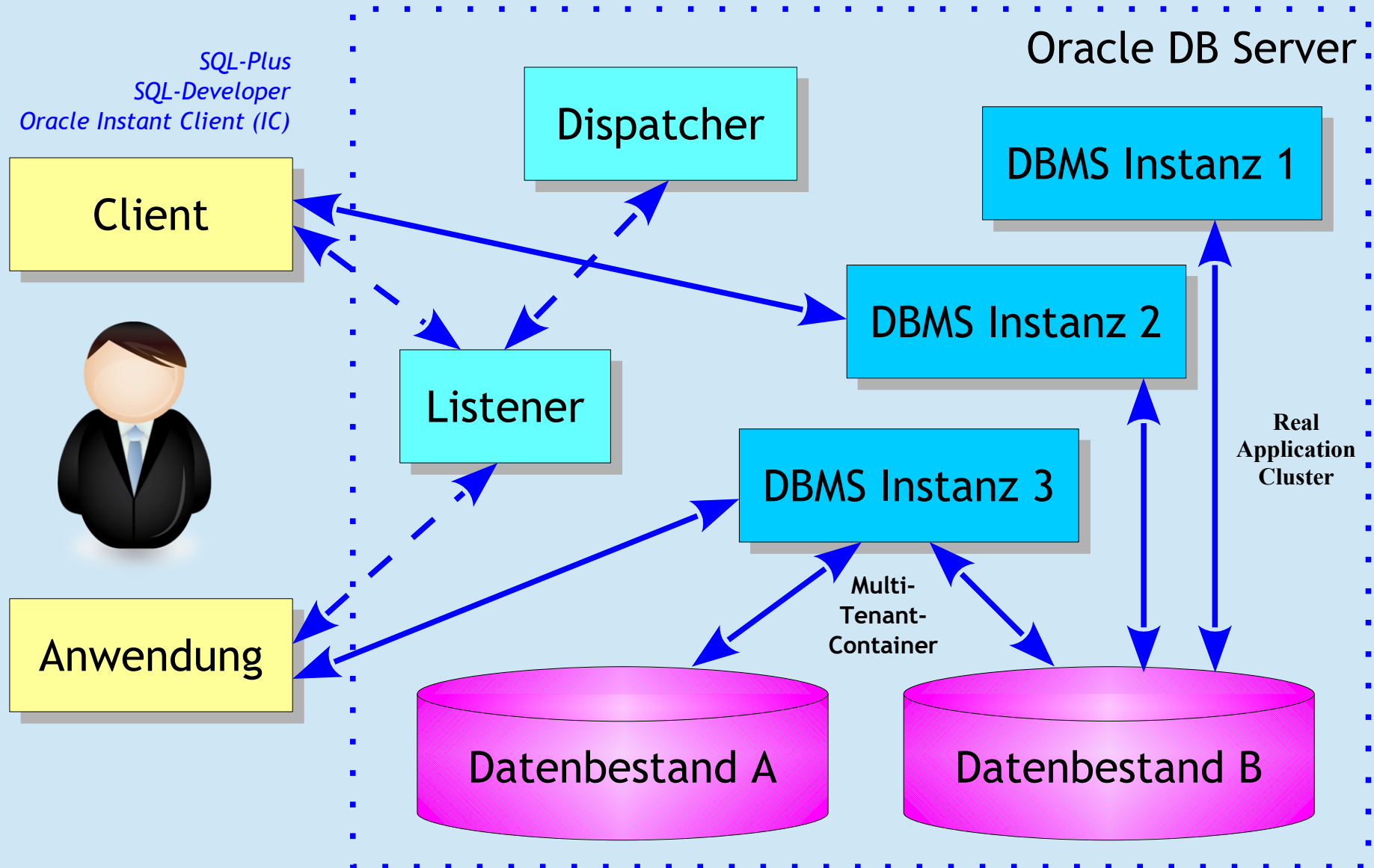
Funktionalitäten

*Client ist ein Programm,
das die Ressourcen in
Anspruch nimmt.*

*Server ist ein Programm,
das die Ressourcen zur
Verfügung stellt.*

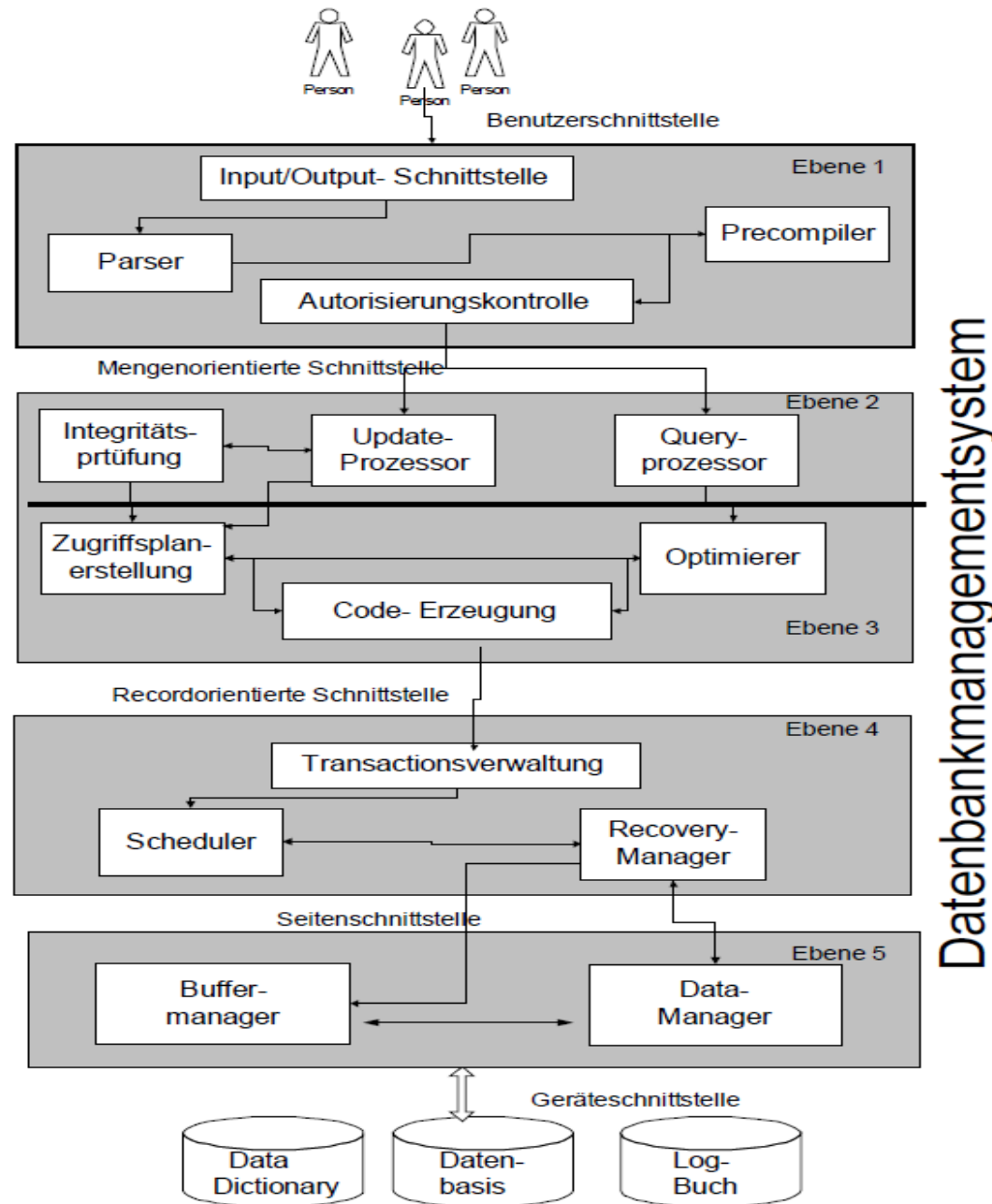


Architektur von Oracle



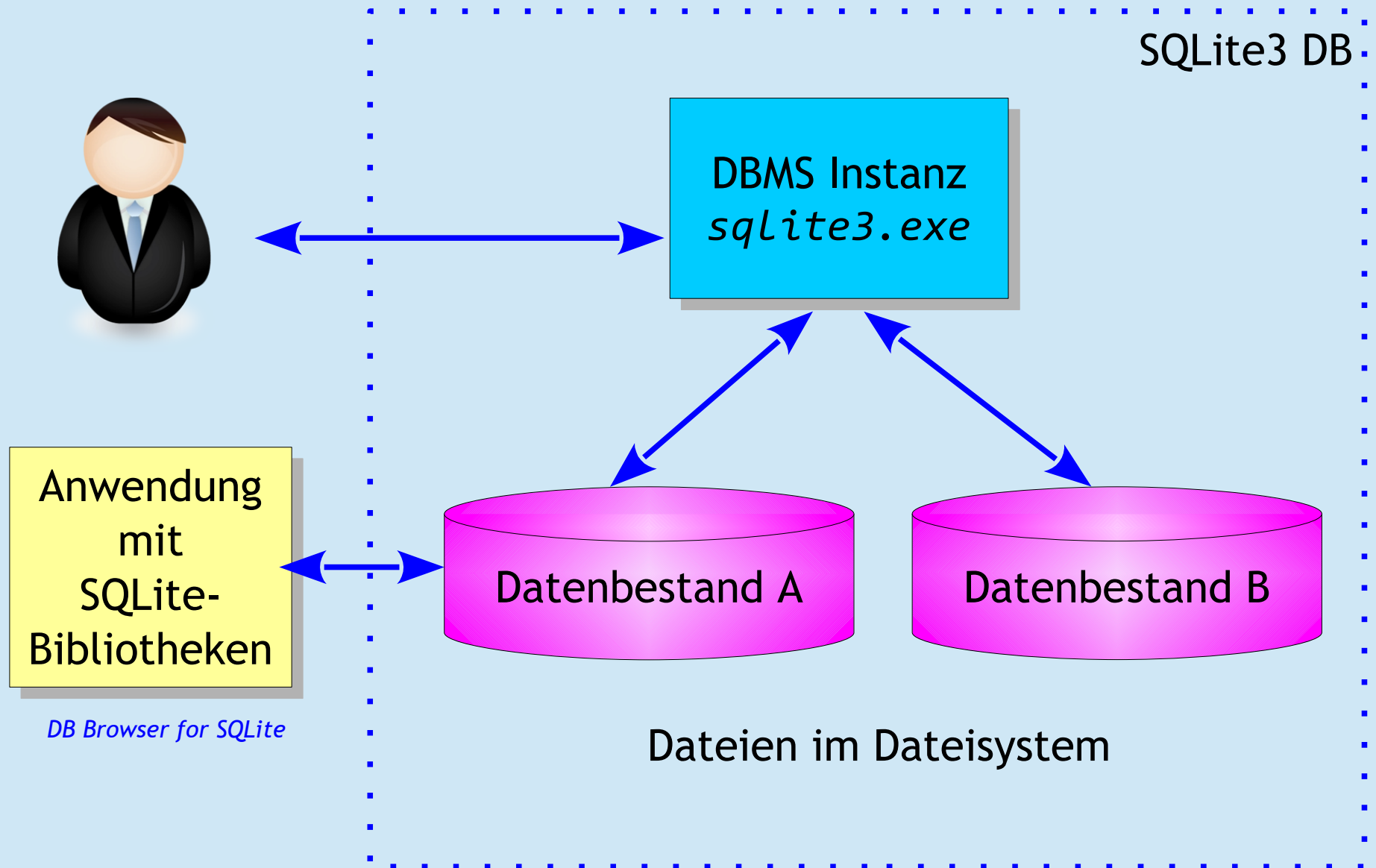


Architektur von Datenbanksystemen





Architektur SQLite (keine Client/Server-Architektur)





Datenbankadministrator (DBA)

- *Datenbankdesign, anlegen der Datenbank.*
- *Softwareinstallation und -wartung.*
- *Speicherplatzverwaltung.*
- *Implementierung von Sicherheitsmechanismen.*
- *Laden von Daten.*
- *Backup und Recovery.*
- *Reorganisation von Datenbeständen.*
- *Systembeobachtung und Tuning.*



Anwendungsentwickler

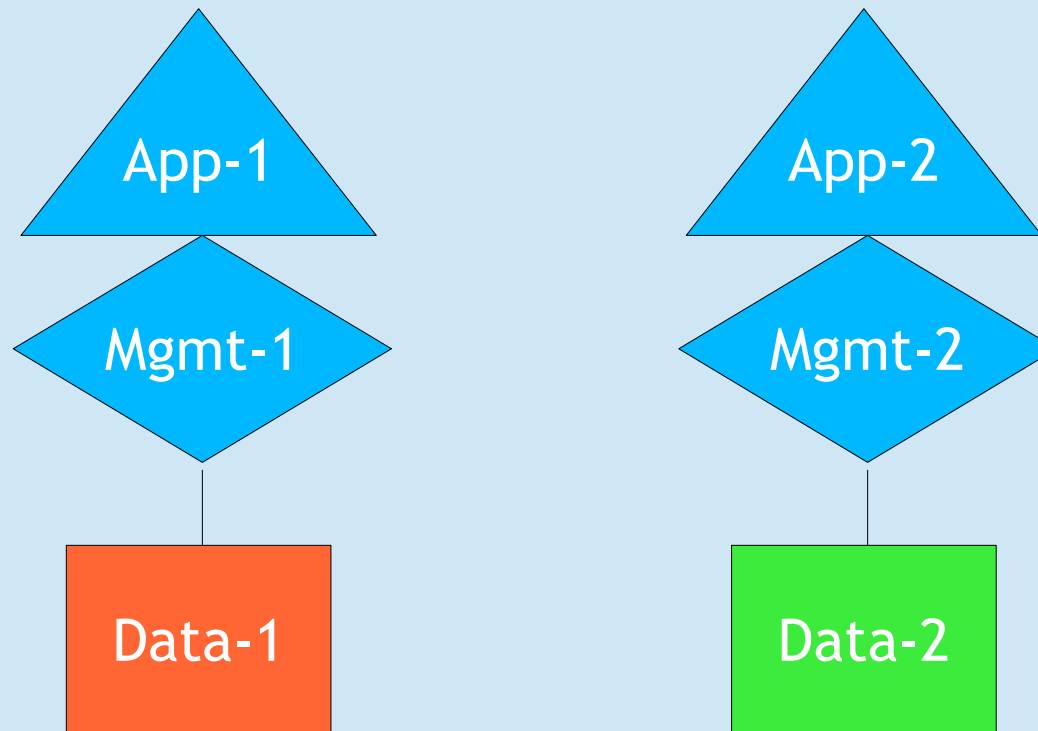
- *Systemanalyse.*
- *SQL.*
- *Fertigung der Standard-Abfragen.*
- *Anwendungsentwicklung.*

Endanwender

- *Benutzung der vom Anwendungsentwickler erstellten Programme.*
- *Benutzung von vorgefertigten und Ad-Hoc-Abfragen.*
- *Benutzung von QBE-Werkzeugen (query by example) zur Ausführung von Ad-Hoc-Abfragen.*

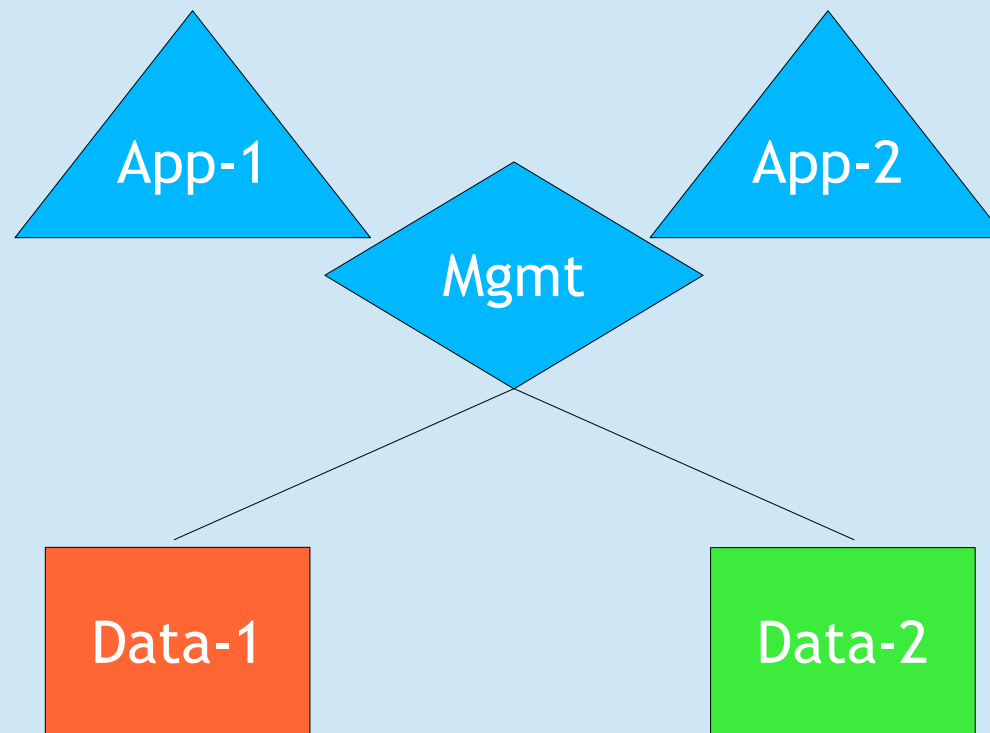


1960er Jahre



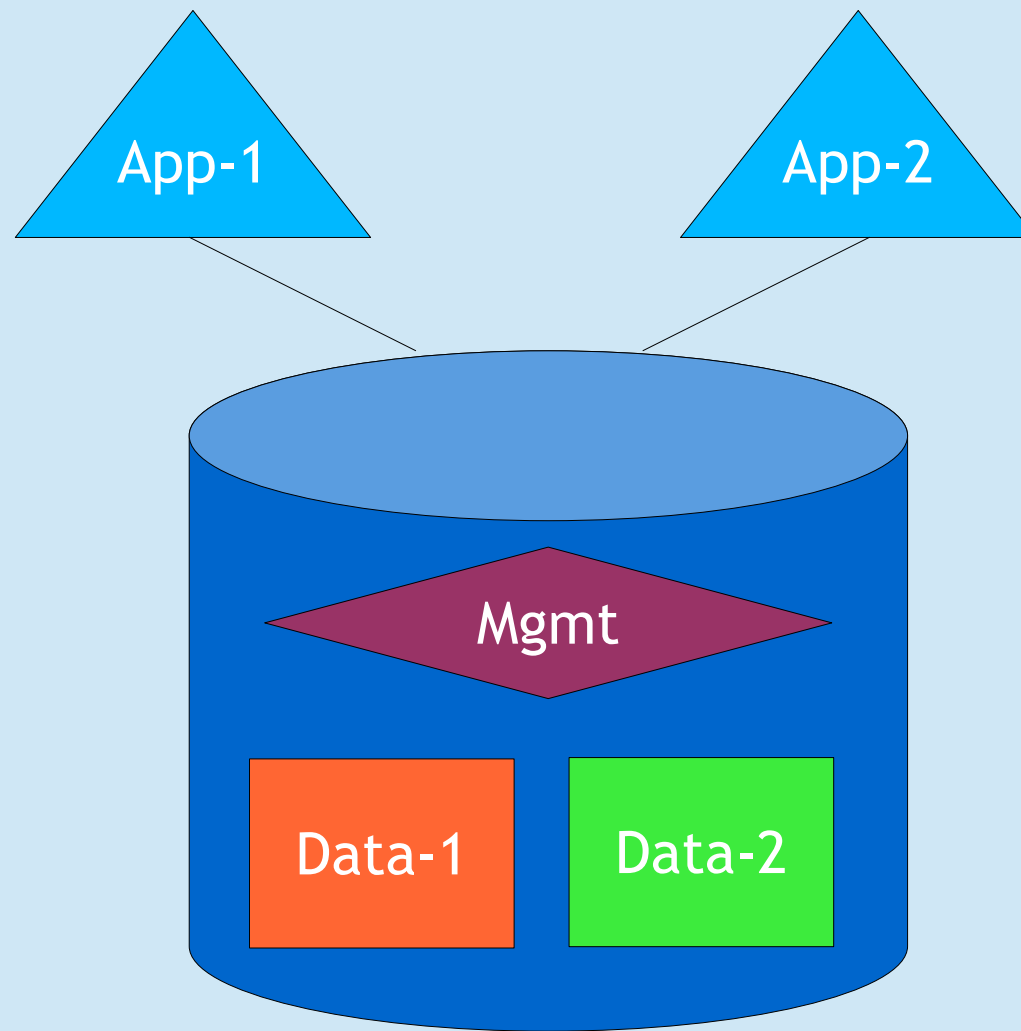


1970er Jahre





Heute





```
. . .  
Connection c = null;  
Statement s = null;  
ResultSet result = null;  
  
. . .  
s = c.createStatement();  
result = s.executeQuery("SELECT * FROM Z"); // Anfrage  
ResultSetMetaData sp = result.getMetaData(); // Spaltennamen  
iAnzSp = sp.getColumnCount(); // Anzahl von Spalten  
for (Zeile="", j=1; j<=iAnzSp; j++)  
    Zeile = Zeile + " " + sp.getColumnName(j);  
System.out.println (Zeile);  
while (result.next())  
{  
    for (Zeile="", j=1; j<=iAnzSp; j++)  
        Zeile = Zeile + " " + result.getString(j);  
    System.out.println(Zeile);  
}
```



Seit 1960er Jahren haben die Datenbanken eine lange Geschichte hinter sich. Im Laufe der Zeit wurden unterschiedliche Typen der Datenbanken entwickelt.

Einen entscheidenden Einfluss darauf hatte einerseits die Entwicklung der mathematischen Theorie der Datenbanken durch E. F. Codd.

Andrerseits die Anforderungen der Wirtschaft und die rasante Kapazitätssteigerung der Hardware haben auch maßgebend zum modernen Stand der Datenbanken beigetragen.

Selbstverständlich, die Art der Daten spielt auch eine wichtige Rolle bei der Auswahl einer Datenbank.



Es gibt folgende Arten der Datenbanken:

- *hierarchische Datenbanken (heute selten verwendet);*
- *netzartige Datenbanken (heute selten verwendet);*
- *dokumentenbasierte Datenbanken (selten verwendet);*
- *Key-Value-Datenbanken (selten verwendet);*
- *objektorientierte Datenbanken;*
- *relationale Datenbanken (heute fast ausschließlich verwendet).*



Hierarchische Datenbank

Binäre Bäume. Ein Vorgänger, mehrere Nachfolger.

Beispiel: Zoo. Linker Nachfolger (Nachbar), rechter Nachfolger (Nachbar).

Verzeichnisbäume in MS DOS, MS Windows, Linux/UNIX, LDAP, FQDN.

Netzartige Datenbank

Mehrere Vorgänger, mehrere Nachfolger. Beispiel: Zoo, die Vorgänger hinzufügen.

Dokumentenorientierte Datenbank

Dokument = Verzeichnis. Dokument enthält die Daten unterschiedlicher Art. Seltene Verwendung. MongoDB.

Schlüssel-Wert-Datenbank

Tabelle mit zwei Spalten. Begrenzte Bedeutung. BerkeleyDB.



Die relationalen Datenbanken haben sich aus folgenden Gründen durchgesetzt:

- Beziehungen und Entitäten werden gleich dargestellt => derselbe Verarbeitungsmechanismus.*
- Relationale Algebra liefert einen sehr gut entwickelten mathematischen Apparat => umfangreiche und fortgeschrittene Operationen.*
- Mit relationalen Datenbanken kann man alle anderen Typen von Datenbanken emulieren.*

Das sind die wichtigsten theoretischen (logischen) Vorteile.



Dr. Codd veröffentlichte 12 Regeln, die eine relationale Datenbank im strengen Sinne definieren:

- 1. Ein relationales DBMS muss in der Lage sein, Datenbanken vollständig über seine relationalen Fähigkeiten zu verwalten.*
- 2. Darstellung von Informationen: Alle Informationen in einer relationalen Datenbank (einschließlich Namen von Tabellen und Spalten) sind explizit als Werte in Tabellen darzustellen.*
- 3. Zugriff auf Daten: Jeder Wert einer relationalen Datenbank muss durch eine Kombination von Tabellennamen, Primärschlüssel und Spaltennamen auffindbar sein.*
- 4. Systematische Behandlung von Nullwerten: Das DBMS behandelt Nullwerte durchgängig gleich als unbekannte oder fehlende Daten und unterscheidet diese von Standardwerten.*



5. *Struktur einer Datenbank: Die Datenbank und ihre Inhalte werden in einem sogenannten Systemkatalog auf derselben logischen Ebene wie die Daten selbst – also in Tabellen – beschrieben. Demzufolge lässt sich der Katalog mit der Datenbanksprache abfragen.*
6. *Abfragesprache: Zu einem relationalen System gehört mindestens eine Abfragesprache mit einem vollständigen Befehlssatz für Datendefinition, Manipulation, Integritätsregeln, Autorisierung und Transaktionen.*
7. *Aktualisieren von Sichten: Alle Sichten, die theoretisch aktualisiert werden können, lassen sich auch vom System aktualisieren.*
8. *Abfragen und Bearbeiten ganzer Tabellen: Das DBMS unterstützt nicht nur Abfragen, sondern auch die Operationen für Einfügen, Aktualisieren und Löschen in Form der Datensätze und ganzer Tabellen.*



9. *Physikalische Datenunabhängigkeit: Der logische Zugriff auf die Daten durch Anwendungen muss unabhängig von den physikalischen Zugriffsmethoden oder den Speicherstrukturen der Daten sein.*
10. *Logische Datenunabhängigkeit: Änderungen der Tabellenstrukturen dürfen keinen Einfluss auf die Logik der Anwendungen haben.*
11. *Unabhängigkeit der Integrität: Integritätsregeln müssen sich in der Datenbanksprache definieren lassen. Die Regeln müssen im Systemkatalog gespeichert werden. Sie dürfen sich durch keine (z.B. Low-Level) Mechanismen umgehen lassen.*
12. *Verteilungsunabhängigkeit: Der logische Zugriff auf die Daten durch Anwendungen darf sich beim Übergang von einer nicht-verteilten zu einer verteilten Datenbank nicht ändern.*



Die neun Codd'schen Anforderungen für Datenbanksysteme

1. **Integration:** einheitliche nichtredundante Datenverwaltung
2. **Operationen:** Speichern, Suchen, Ändern (ad hoc und "fest verdrahtet")
3. **Katalog:** Zugriffe auf Datenbankbeschreibung im "Data Dictionary"
4. **Benutzersichten:** unterschiedliche Sichten auf Datenbankausschnitte
5. **Konsistenzüberwachung:** Gewährleistung der Korrektheit des Datenbankinhaltes gegenüber Schema
6. **Datenschutz:** Ausschluss unauthorisierter Zugriffe
7. **Transaktionen:** mehrere Operationen als Funktionseinheit
8. **Synchronisation:** parallele Transaktionen koordinieren
9. **Datensicherung:** Wiederherstellung von Daten nach Systemfehlern

[Codd, E.: Relational database: A practical foundation for productivity. Communications of the ACM, 25:2, 109-117, 1982]

23



Inhalt:

- Einführung
- Mengenlehre
- ANSI-SPARC-Modell
- Anforderungsanalyse
- Konzeptueller Entwurf
- Logischer Entwurf

- Definition
- Eigenschaften
- Operationen
- Zahlenmengen
- Dimensionen

Die Mengenlehre bildet die mathematische Grundlage der Theorie relationaler Datenbanken.

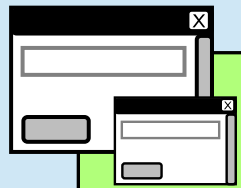
Dafür gibt es ein extra-Skript.



Inhalt:

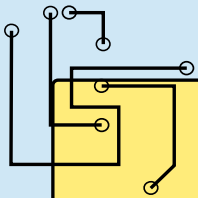
- Einführung
- Mengenlehre
- ANSI-SPARC-Modell
- Anforderungsanalyse
- Konzeptueller Entwurf
- Logischer Entwurf

-
- A light blue speech bubble with a black outline points from the 'ANSI-SPARC-Modell' item in the main list to the right. It contains two sub-points.
- Beschreibung
 - Datenbankentwurf



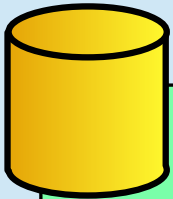
externe Ebene

Benutzeroberflächen, Datensichten, API und Schnittstellen



konzeptionelle Ebene

Beziehungen, Daten



interne Ebene

Art und Form der Speicherung

Quelle <https://de.wikipedia.org/wiki/ANSI-SPARC-Architektur>



Das ANSI-SPARC-Modell ist eine Grundlage für Entwicklung der modernen Datenbanksysteme. Es gewährleistet eine Unabhängigkeit der Datenbank von Programmiersprache und Hardware.

Anwendungs-Ebene, externe Ebene

Die Daten aus der logischen (konzeptionellen) Ebene, die für bestimmte Benutzer auf bestimmte Art und Weise sichtbar sein sollen (Views).

Konzeptionelle Ebene, logische Ebene

Abbildung der realen Welt, Definition und logische Zuordnung der Daten (ERM, Tabellen).

Physische Ebene, interne Ebene

Organisation der Daten auf dem Datenträger (Datendateien, Tablespaces, Segmente, Indexes, Arbeitsspeicher, Prozessor-Prioritäten).



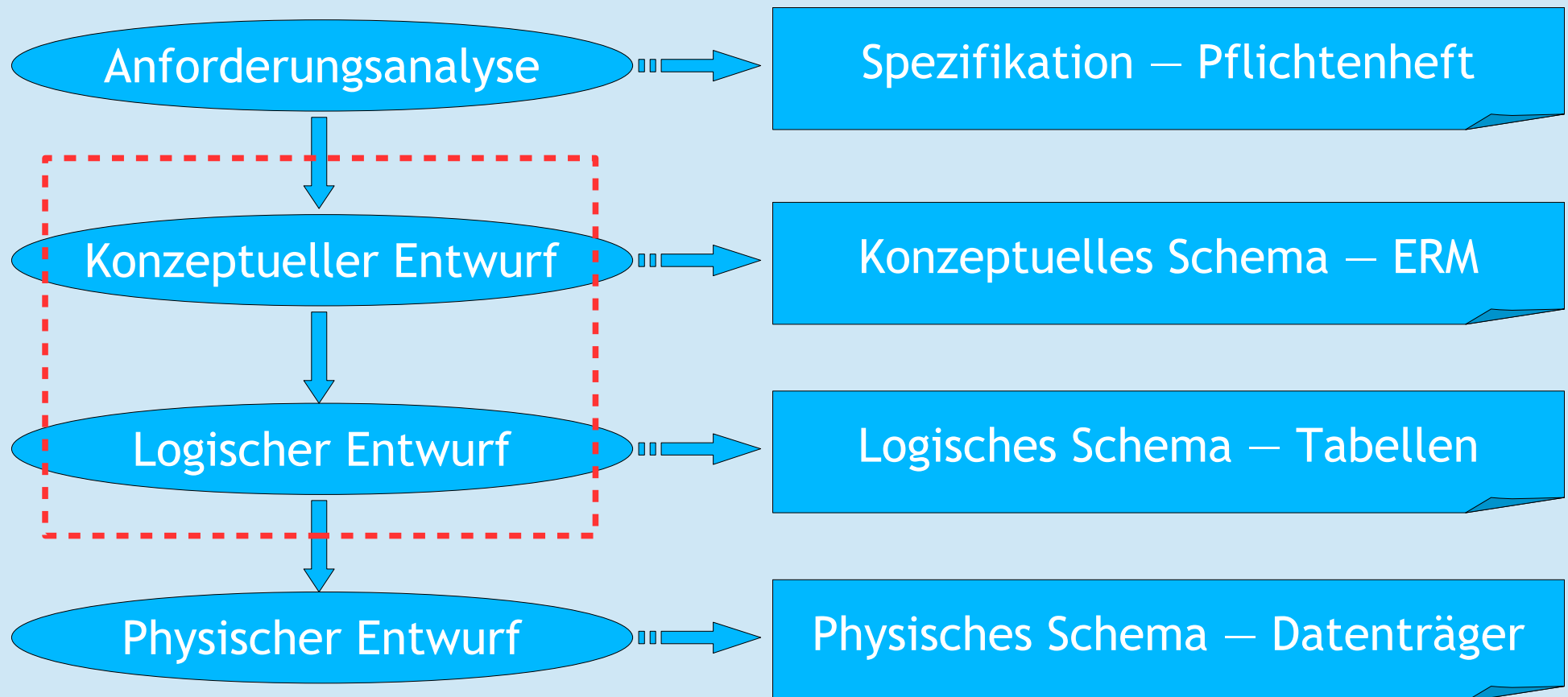
In der externen Ebene sind mehrere Benutzersichten beschrieben. Jede Sicht erlaubt den Anwendern nur einen Teil der Daten der logischen Ebene zu sehen. Der Rest der Daten bleibt verborgen. Es geht hier um grafische Benutzerschnittstellen und Eingabeaufforderung.

In der konzeptionellen Ebene ist die logische Struktur der abgebildeten Welt vertreten. Meistens wird hier das relationale Datenmodell eingesetzt. Hier werden die Daten und deren Beziehungen beschrieben, sowie ERM-Diagramme erstellt und in die Tabellen umgesetzt. Integritätsbedingungen und Zugriffsrechte werden hier festgelegt.

In der internen Ebene werden die physischen Speicherstrukturen (Dateien, Partitionen, Tablespaces) und Zugriffsmechanismen auf Daten beschrieben.



Phasen des Datenbankentwurfs:





Inhalt:

- Einführung
- Mengenlehre
- ANSI-SPARC-Modell
- Anforderungsanalyse
- Konzeptueller Entwurf
- Logischer Entwurf

- Kurzfassung
- Universitätsschema
- Modell
eines Krankenhauses



Die Anforderungsanalyse ist kein Bestandteil dieses Kurses (sondern von Software Engineering), deswegen nur kurz.

Im Laufe der Anforderungsanalyse werden folgende Daten erfasst:

- welche Abteilungen im Unternehmen mit der Datenbank arbeiten;*
- welche Geschäftsprozesse unterstützt werden sollen;*
- welche Daten involviert werden;*
- wie die Daten strukturiert sind;*
- wie die qualitativen und quantitativen Anforderungen an die Datenbank sind.*

In diesem Schritt wird das konkrete Datenbanksystem noch nicht betrachtet.



Schritte der Anforderungsanalyse nach A. Kemper:

- 1. Identifikation von Organisationseinheiten.*
- 2. Identifikation der zu unterstützenden Aufgaben.*
- 3. Anforderungs-Sammelplan: die zu befragenden Personen ermitteln.*
- 4. Anforderungs-Sammlung.*
- 5. Filterung: Verständigkeit und Eindeutigkeit der Daten prüfen.*
- 6. Satzklassifikationen: Informationen den Objekten, Beziehungen, Operationen und Ereignissen zuordnen.*
- 7. Formalisierung/Systematisierung: Informationen ins Pflichtenheft übertragen.*



In Folge der Anforderungsanalyse werden beschrieben:

- *Informationsanforderungen;*
- *Datenverarbeitungsanforderungen.*

Es empfiehlt sich, diese Beschreibungen als strukturierte formularähnliche Dokumente zu gestalten, selbstverständlich in elektronischer Form.

Zu Informationsanforderungen gehören die Beschreibungen von:

- *Objekten und deren Attributen;*
- *Beziehungen und deren Attributen.*

Zu Datenverarbeitungsanforderungen gehören die Beschreibungen von:

- *Prozessen.*



Universitätsschema wird benötigt.

Informationen über Objekte wie Professoren, Studenten, Assistenten, Vorlesungen werden zusammengefasst, sowie deren Attribute und Beziehungen.

Folgende Daten werden in der Anforderungsanalyse gesammelt.



Objektbeschreibung

Objekt: Uni-Angestellte.

Anzahl: 1000.

Attribut	Typ	Länge	Identifi- zierend	Beispiel
PersonalNr	Char	10	ja	1234561234
Gehalt	Dezimal	8.2	nein	9000.11
Rang	String	4	nein	W3



Beziehungsbeschreibung

Beziehung: prüfen.

Anzahl: 1 000 pro Jahr.

*Beteiligte Objekte: Professor als Prüfer,
Student als Prüfling,
Vorlesung als Lehrstoff.*

Attribut	Typ	Länge	Identifizierend	Beispiel
Datum	date	8	nein	31.12.2021
Uhrzeit	time	6	nein	11:12:13
Note	real	3.1	nein	1,3



Prozessbeschreibung

Prozess: Zeugnisausstellung.

Häufigkeit: halbjährlich.

Priorität: hoch.

Benötigte Daten: Prüfungen, Studienordnungen, Studenteninformation.

Datenmenge: 2000 Studenten, 200 Prüfungen, 10 Studienordnungen.



Ein Krankenhaus wird betrachtet.

Nur 3 Objekte werden untersucht: Ärzte, Pflegepersonal und Patienten, sowie die Beziehungen zwischen ihnen und deren Attribute.

Folgende Daten wurden im Laufe der Anforderungsanalysis gesammelt.



Objektbeschreibung

Objekt: Patient.

Anzahl: 2000.

Attribut	Typ	Länge	Identifi- zierend	Beispiel
Name	String	24	ja	Schulz
PatID	Integer	6	nein	120056
Krankheit	String	variabel, 300	nein	Schnupfen

Ist hier alles OK?



Beziehungsbeschreibung

Beziehung: behandelt.

Anzahl: 100.

Beteiligte Objekte: Arzt, Patient.

Attribute der Beziehung: Datum, Uhrzeit.

Attribut	Typ	Länge	Identifi- zierend	Beispiel
Datum	date	8	nein	31.12.2013
Uhrzeit	time	6	nein	11:12:13



Prozessbeschreibung

Prozess: Behandlungsplan erstellen.

Häufigkeit: einmal pro Patient.

Priorität: hoch.

Benötigte Daten: Beschwerden, medizinische Analysen, Ultraschall, Röntgenaufnahmen, u.s.w.

Datenmenge: 100 MiB pro Patient.



Inhalt:

- Einführung
- Mengenlehre
- ANSI-SPARC-Modell
- Anforderungsanalyse
- Konzeptueller Entwurf
- Logischer Entwurf

- Beispiele von Modellen
- ERM
- Relationen
- Spezielle Konzepte

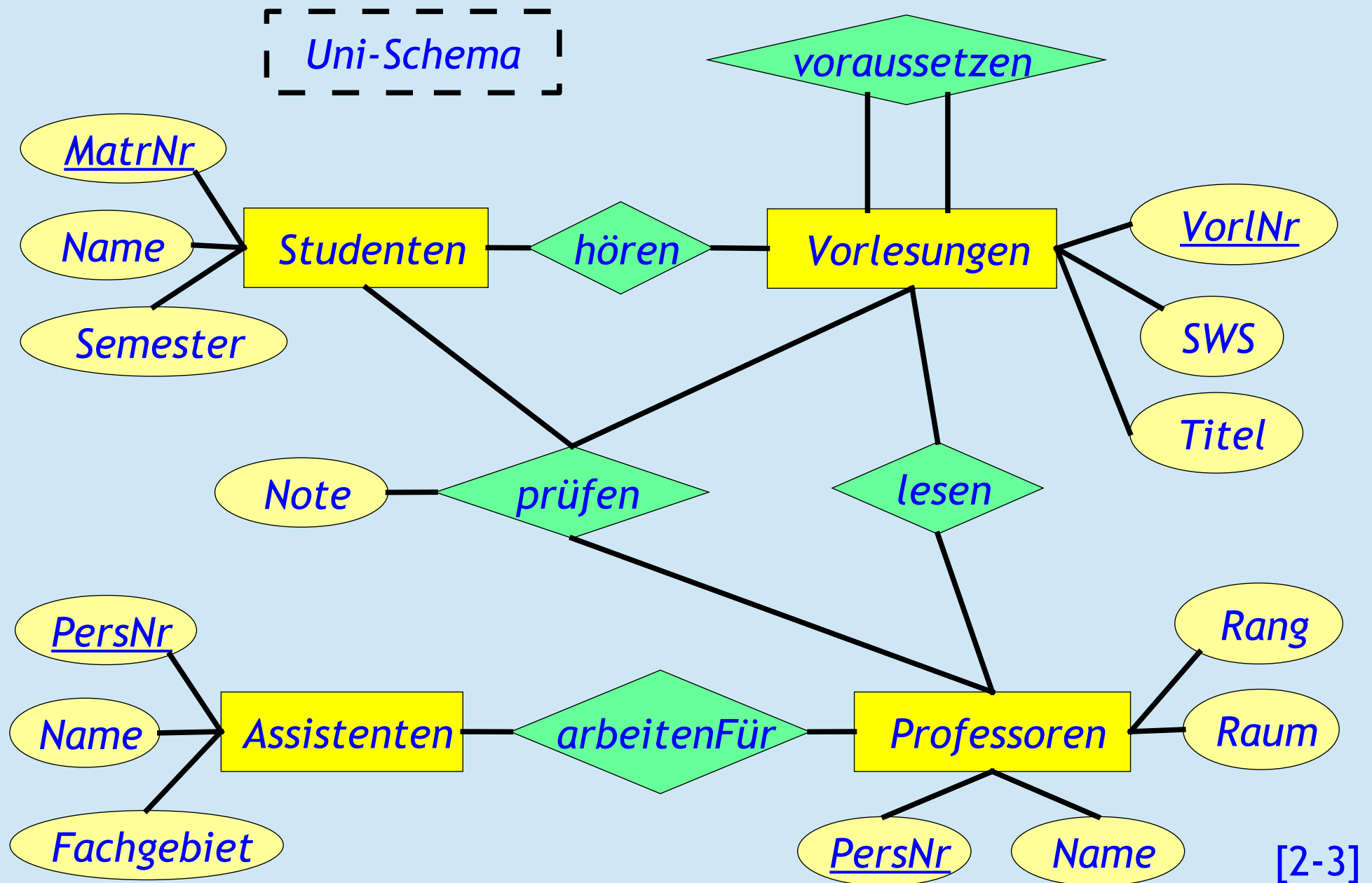


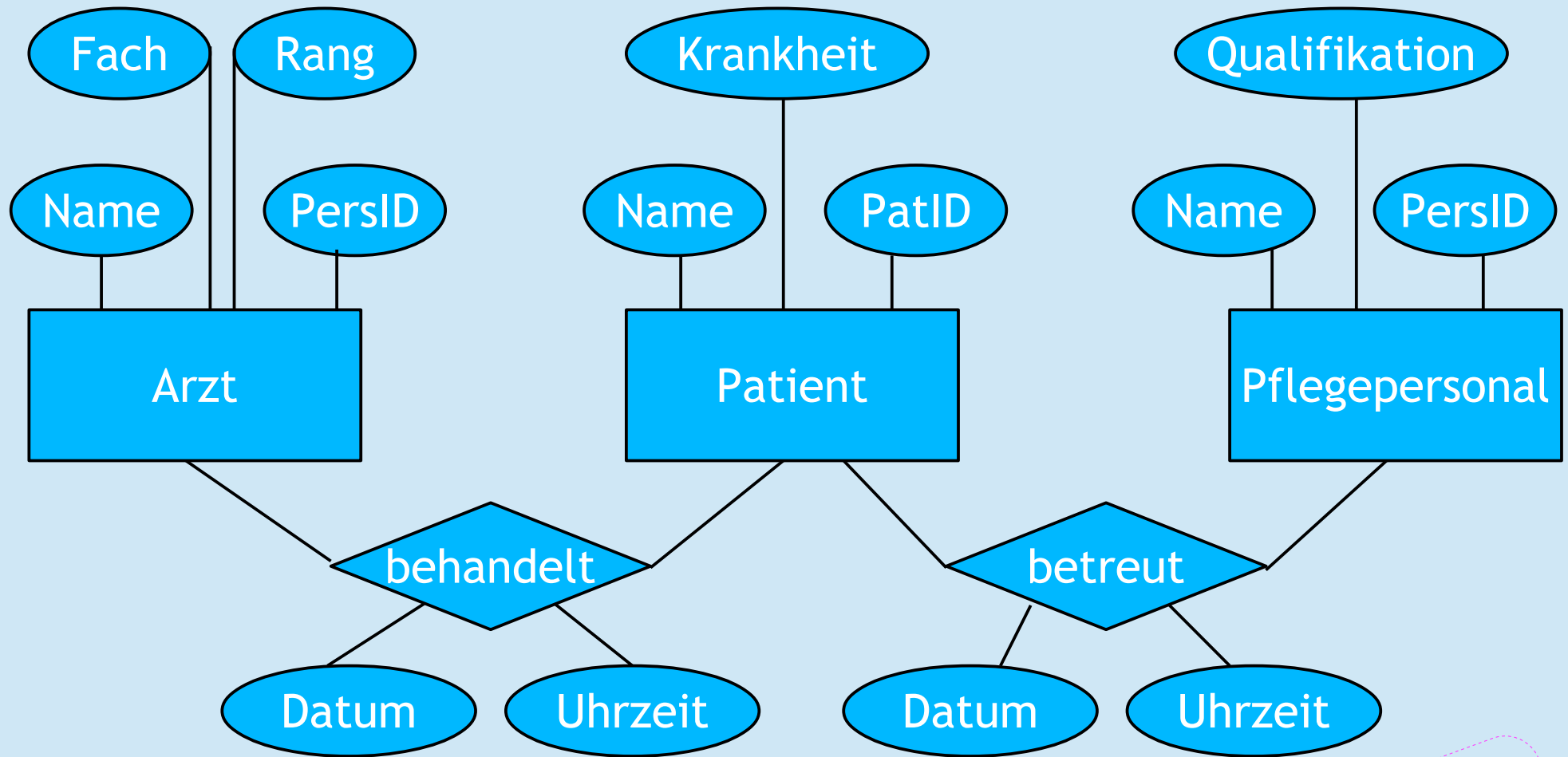
Auf der Basis der in der Anforderungsanalyse erfassten Daten wird ein konzeptueller Entwurf erstellt.

Für die Informationsdarstellungen in dem konzeptuellen Entwurf wird häufig das Entity-Relationship-Modell verwendet.

In diesem Schritt wird das konkrete Datenbanksystem immer noch nicht betrachtet.

Die obigen Beispiele ergeben folgende ER-Modelle für Uni und für Krankenhaus:





Peter-Chen-Notation



Die grundlegenden Modellierungsstrukturen des ERM in Peter-Chen-Notation sind:

- *Entities, Entitäten, Entitätstypen, Gegenstände, Gegenstandstypen;*
- *Relations, Relationships, Beziehungen, Beziehungstypen;*
- *Attribute (Charakteristik der Entitätstypen);*
- *Funktionalitäten (Charakteristik der Beziehungstypen);*
- *Schlüssel, Primärschlüssel (Charakteristik des Modells);*
- *Rollen (Charakteristik der Beziehungstypen).*

Alternative dazu: UML

Modellierungsstrukturen der deutschen Sprache:

- Substantive;
- Adjektive;
- Pronomen;
- Verben;
- Präpositionen;
- Adverbien.



Die Gegenstände sind die Objekte der reellen Welt, die sich von einander physisch oder gedanklich unterscheiden. Sind einige Objekte sehr ähnlich, so werden sie zu Gegenstandstypen abstrahiert:

"Zimmermann", "Merkel", "Meier" ==> "Mensch"

"VW", "Mercedes", "BMW" ==> "Auto"

Eigentlich, man arbeitet ausschließlich mit Gegenstandstypen, also mit abstrakten Objekten (Klassen in C++), und nicht mit einzelnen Instanzen (Gegenständen, Objekte in C++).

Der Gegenstandstyp wird grafisch als Rechteck dargestellt, deren Name drin geschrieben wird.



Die Beziehungen beschreiben das Verhalten von einer Entität in Bezug auf eine andere, also, sie drücken die Bindungen zwischen den Entitäten aus.

Man verwendet ebenso den Begriff Beziehungstypen statt einzelnen Beziehungen, wenn die Rede von den Entitätstypen ist.

Der Beziehungstyp wird grafisch als Raute dargestellt.

Achtung:

In Praxis wird fast immer über die Gegenstände (Entitäten) und die Beziehungen gesprochen, obwohl dabei sind die Gegenstandstypen (Entitätstypen) und die Beziehungstypen fast immer gemeint.

Die Entitätstypen sind untereinander mit Beziehungstypen verbunden. Nicht unbedingt alle – aber ein alleinstehender Entitätstyp ohne jeglichen Beziehungstyp ist schon ziemlich verdächtig.



Die Attribute sind die Charakteristiken von Gegenständen und Beziehungen, sie werden grafisch als Ovale/Kreise dargestellt.

Ein Attribut wird oft als Eigenschaft bezeichnet. Ein Attribut hat einen Namen und kann einen oder mehrere Werte enthalten.

Der zulässige Wertebereich eines Attributes heißt Domäne, z.B. die Domäne des Attributs "Alter" ist:

$$\{ 0, 1, 2, \dots, 130 \}$$

Die Entitätstypen unterscheiden sich durch Anzahl und Namen der Attribute. Die einzelnen Entitäten des selben Entitätstyps unterscheiden sich durch die Werte der Attribute.

Die Beziehungstypen an sich haben keine Attribute, wenn aber beide Entitätstypen (semantisch gesehen) gleiche Attribute haben, dann ist es üblich, diese Attribute dem Beziehungstyp zuzuordnen (attributierte Beziehungstypen).



Name

z.B. Arthur

Betrag

100 €



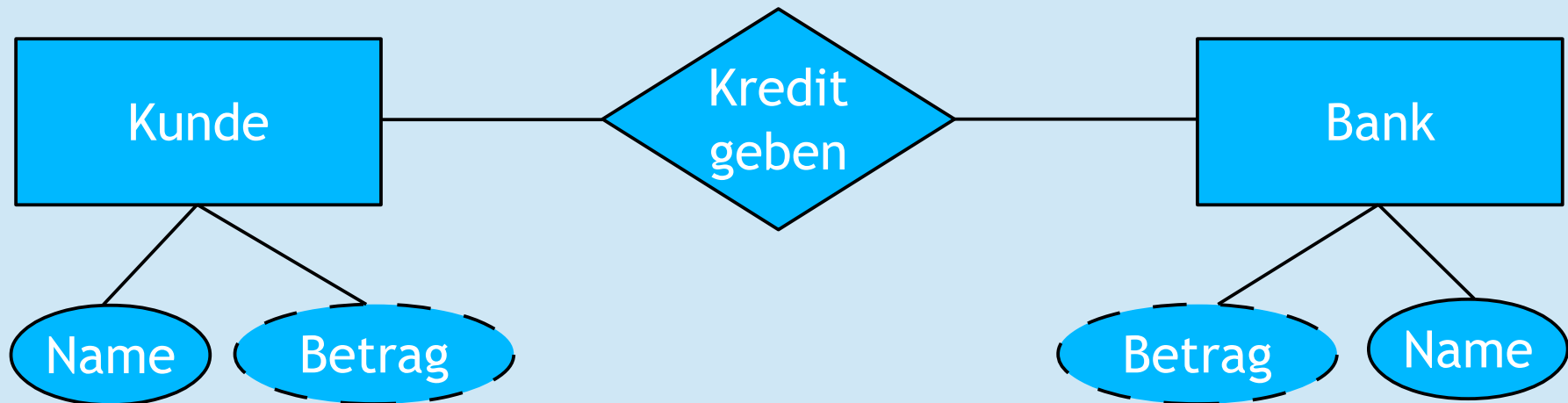
z.B. Credit Generale



Name

100 €

Betrag





Name

z.B. Arthur

Betrag

100 €



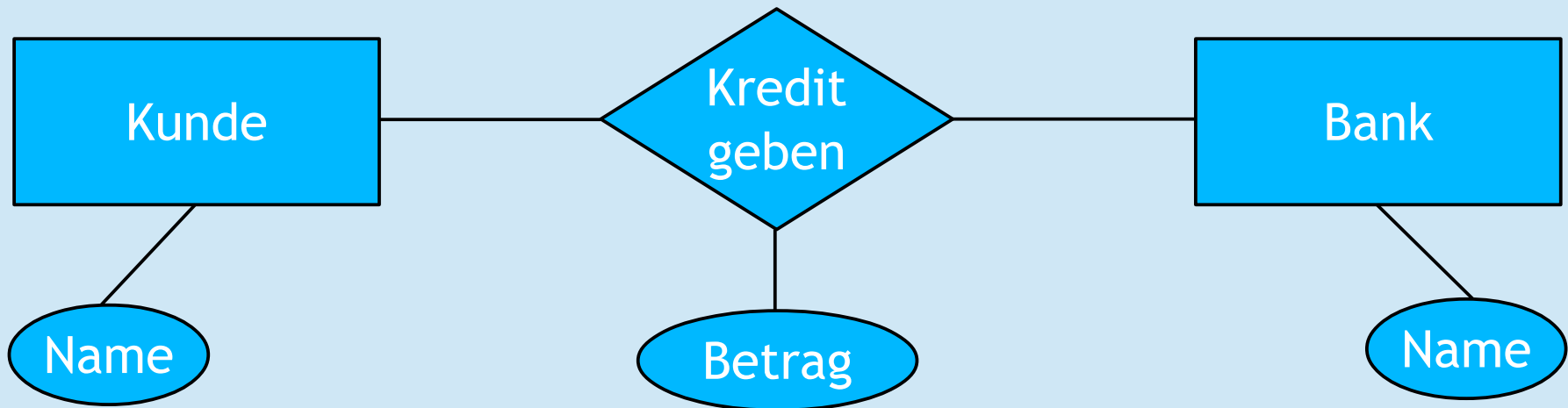
z.B. Credit Generale



Name

100 €

Betrag





Manchmal ist es ziemlich kompliziert, die Attribute von Entitätstypen zu unterscheiden. Bestimmte Objekte können als Entitätstypen und als Attribute gleichzeitig in Erscheinung treten. Grundlage für eine richtige Entscheidung liegt an einem tieferen Verständnis der Mini-Welt.

Grundsätzlich gilt:

- Wenn ein bestimmtes "Etwas" durch ein anderes "Etwas" in der Mini-Welt charakterisiert oder beschrieben wird, dann ist das erste "Etwas" ein Entitätstyp, und das zweite "Etwas" ist (vermutlich) sein Attribut.*
- Kann dagegen ein "Etwas" durch nichts weiteres in der Mini-Welt charakterisiert oder beschrieben werden, dann ist es ein Attribut. Dieses Attribut soll/muss zu irgendeinem Entitätstyp zugeordnet werden.*



Attribute werden in folgende Gruppen unterteilt:

- Einfache Attribute haben zu einem Zeitpunkt nur einen Wert.
- Mehrwertige Attribute haben zu einem Zeitpunkt mehrere Werte, z.B. Ampel kann gleichzeitig rot und gelb sein.
- Zusammengesetzte Attribute bestehen aus mehreren Werten, die als selbständige Attribute betrachtet werden, z.B. Adresse besteht aus PLZ, Ort, Straße, Hausnummer. Allerdings kann die Adresse auch als einfaches Attribut betrachtet werden – abhängig von Miniwelt.
- Abgeleitete (berechnete) Attribute werden aus anderen Attributen berechnet, z.B. Bruttopreis eines Produktes wird aus Nettopreis und Mehrwertsteuersatz berechnet.

Attribute charakterisieren eigentlich nur die Entitätstypen, keine Beziehungstypen. Ein Attribut wird aber einem Beziehungstyp zugeordnet, wenn es zu beiden Entitätstypen gehört, die durch diesen Beziehungstyp verbunden sind.



Definitiv, es gibt keine zwei Entitäten mit absolut gleichen Werten in Attributen (s. Definition von G. Cantor).

Ein Schlüssel ist ein Satz von Attributen, deren Werte die zugeordnete Entität unter allen Entitäten desselben Entitätstyps eindeutig kennzeichnen.

Der Schlüssel kann aus einem oder aus mehreren Attributen bestehen. Im besten Fall besteht der Schlüssel aus einem Attribut. Auf jeden Fall gibt es einen Schlüssel aus allen Attributen.

Mehrere Schlüssel für einen Entitätstyp können existieren (Schlüsselkandidaten).

Keine Fremdschlüssel

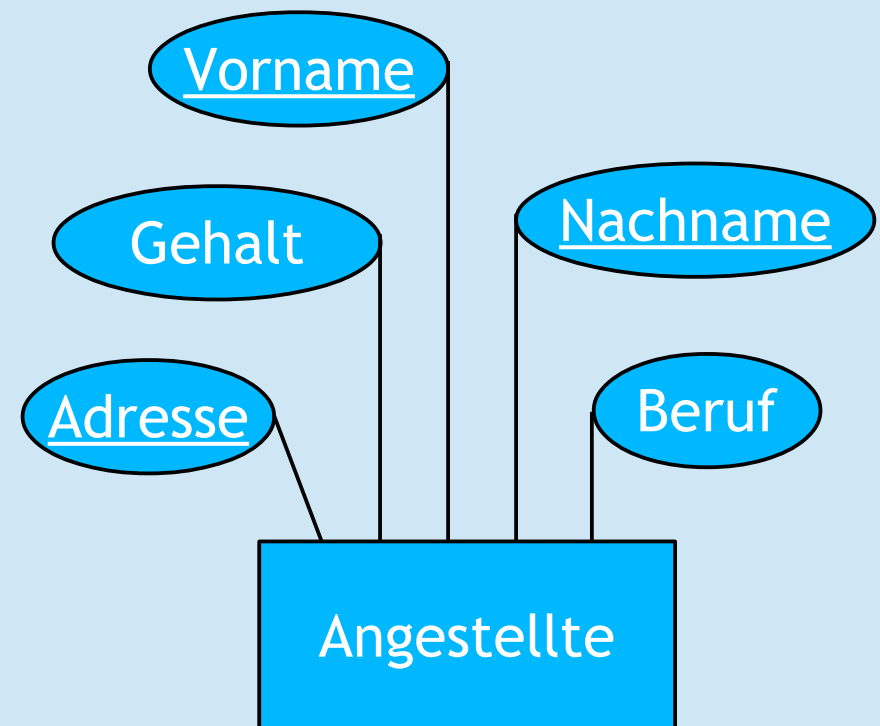
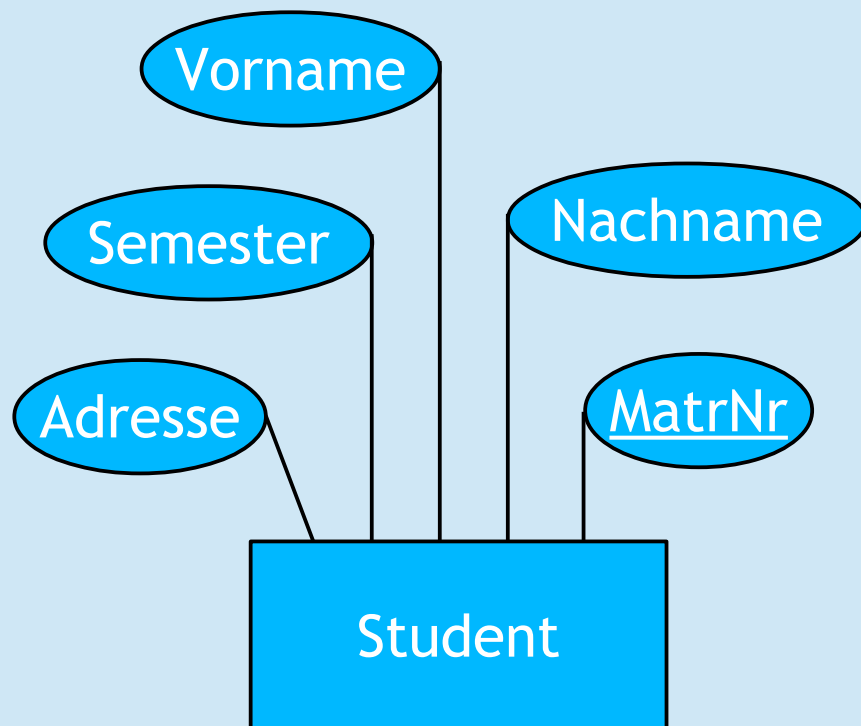
Als Primärschlüssel (Primary Key, PK) wird ein ausgewählt, der minimale Länge unter allen Schlüsselkandidaten hat. Der PK wird in ERM unterstrichen. Der PK ist ein minimaler Schlüssel, das bedeutet, dass keine Untermenge des PK einen Schlüssel bildet.

Wichtig: Wählt man einen anderen PK, so ändert sich das ganze Modell.



Beispiele für Primärschlüssel:

Matrikelnummer für Studenten,
Nachname+Vorname+Adresse für Angestellte.





Beispiele für Primärschlüssel:

Später beim Übergang zum logischen Entwurf sieht es so aus:

Vorname	Nachname	Adresse	Semester	<u>MatrNr</u>
Karl	May	Poststr. 7, 55555 Dresden	1	0356456
Kurt	Wonnegut	12, Michigan street, NY	3	0353453

<u>Vorname</u>	<u>Nachname</u>	<u>Adresse</u>	Gehalt	Beruf
Karl	May	Poststr. 7, 55555 Dresden	5000	Schriftsteller
Kurt	Wonnegut	12, Michigan street, NY	6000	Schriftsteller
Karl	May	Muchacho Plaza 3, Barcelona	7500	Maler



Die Beziehungstypen existieren normalerweise zwischen zwei unterschiedlichen Entitätstypen (binäre Beziehungstypen).

Manchmal gibt es Beziehungen zwischen den Entitäten des selben Entitätstyps – die s.g. rekursiven Beziehungen.

In diesem Fall werden die Rollen den Beziehungen (und entsprechend den Entitäten) zugeschrieben.

Beispiele:

Entitätstyp: "Vorlesungen", Beziehungstyp: "voraussetzen" in dem Universitätsschema.

Entitätstyp: "Softwareprodukt", Beziehungstyp: "Versionsnummer"

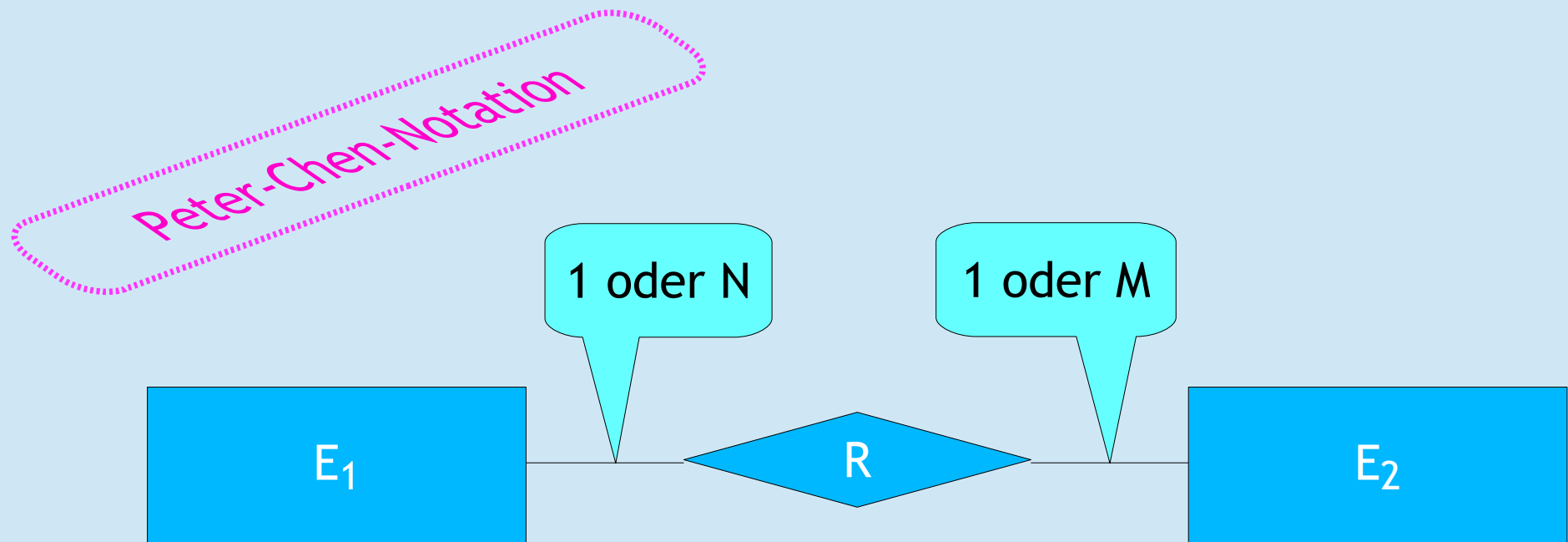
SuSE 10 ==> SuSE 11 ==> SuSE 12

Entitätstyp: "Polizisten", Beziehungstyp: "im Zweierteam patrouillieren".



Es gibt eine Charakteristik der Beziehungen, die sehr wichtig für Modellierung ist. Sie heißt Funktionalität.

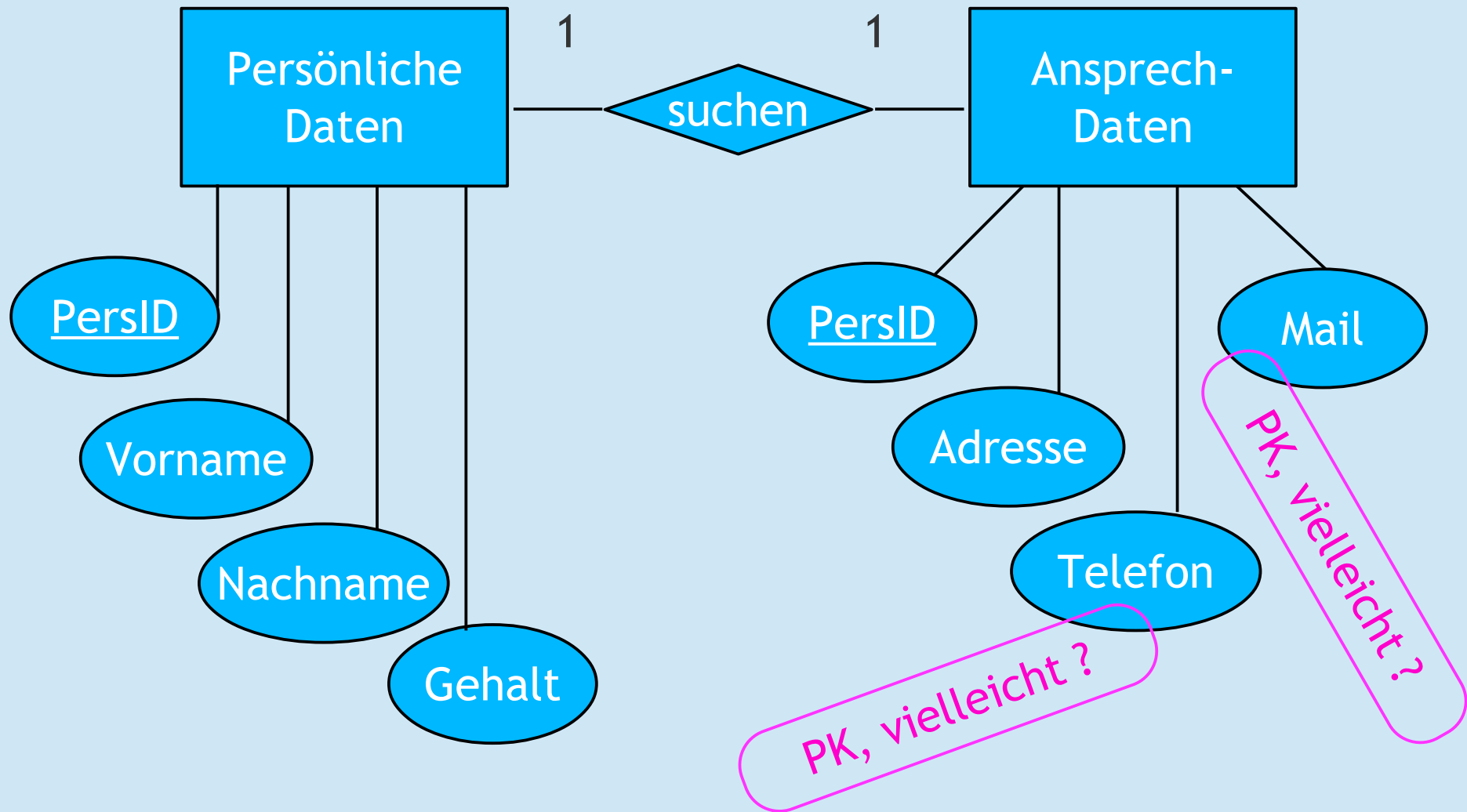
Die Funktionalität eines binären Beziehungstyps gibt an, wie viel Instanzen (Entitäten) von den beiden Entitätstypen in der Beziehung teilnehmen können. Die Funktionalitäten drücken die Gesetzmäßigkeiten der reellen Welt aus, die eigentlich immer gelten müssen, sie stellen eine Art der Integritätsbedingungen dar.

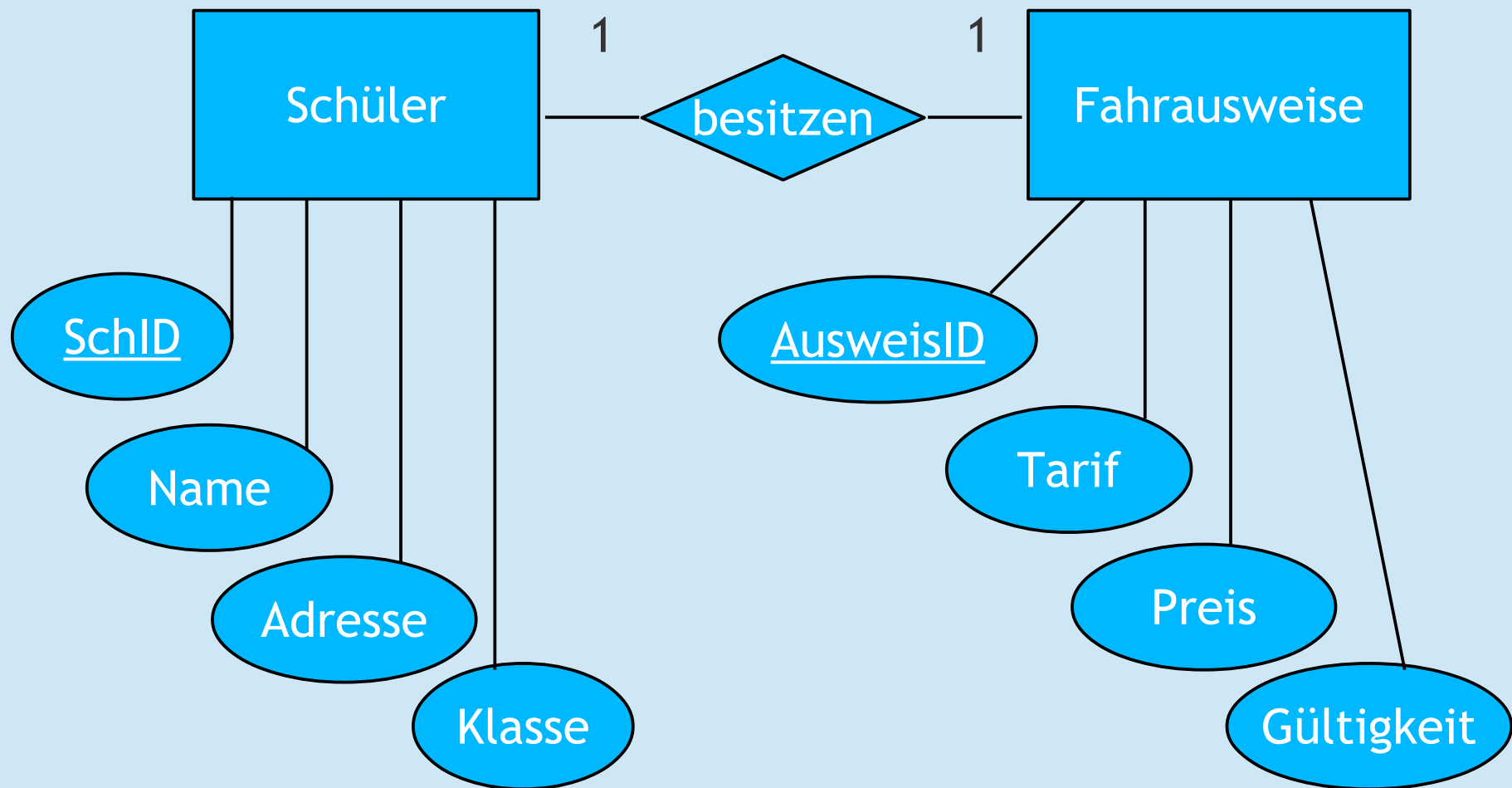


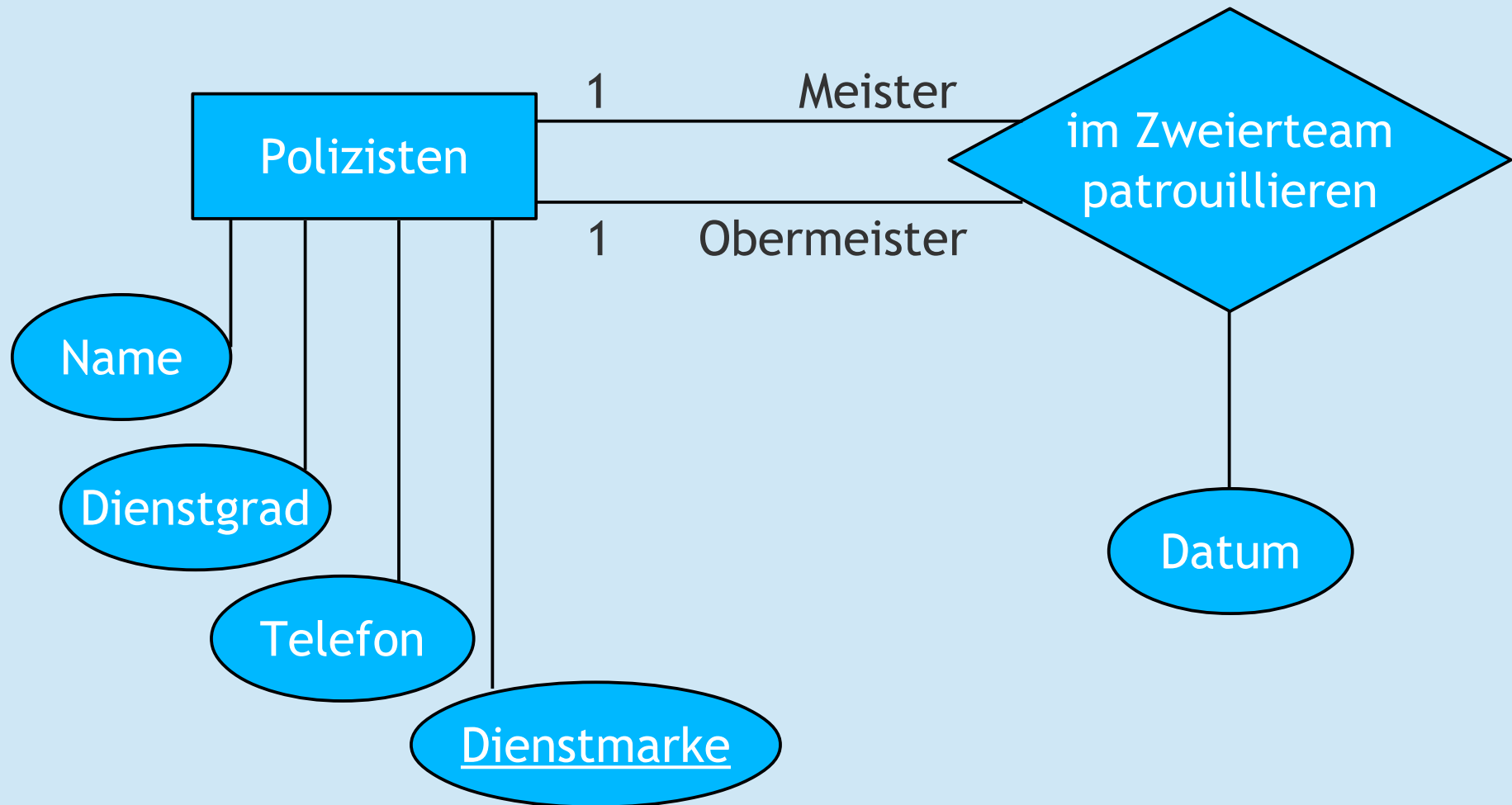


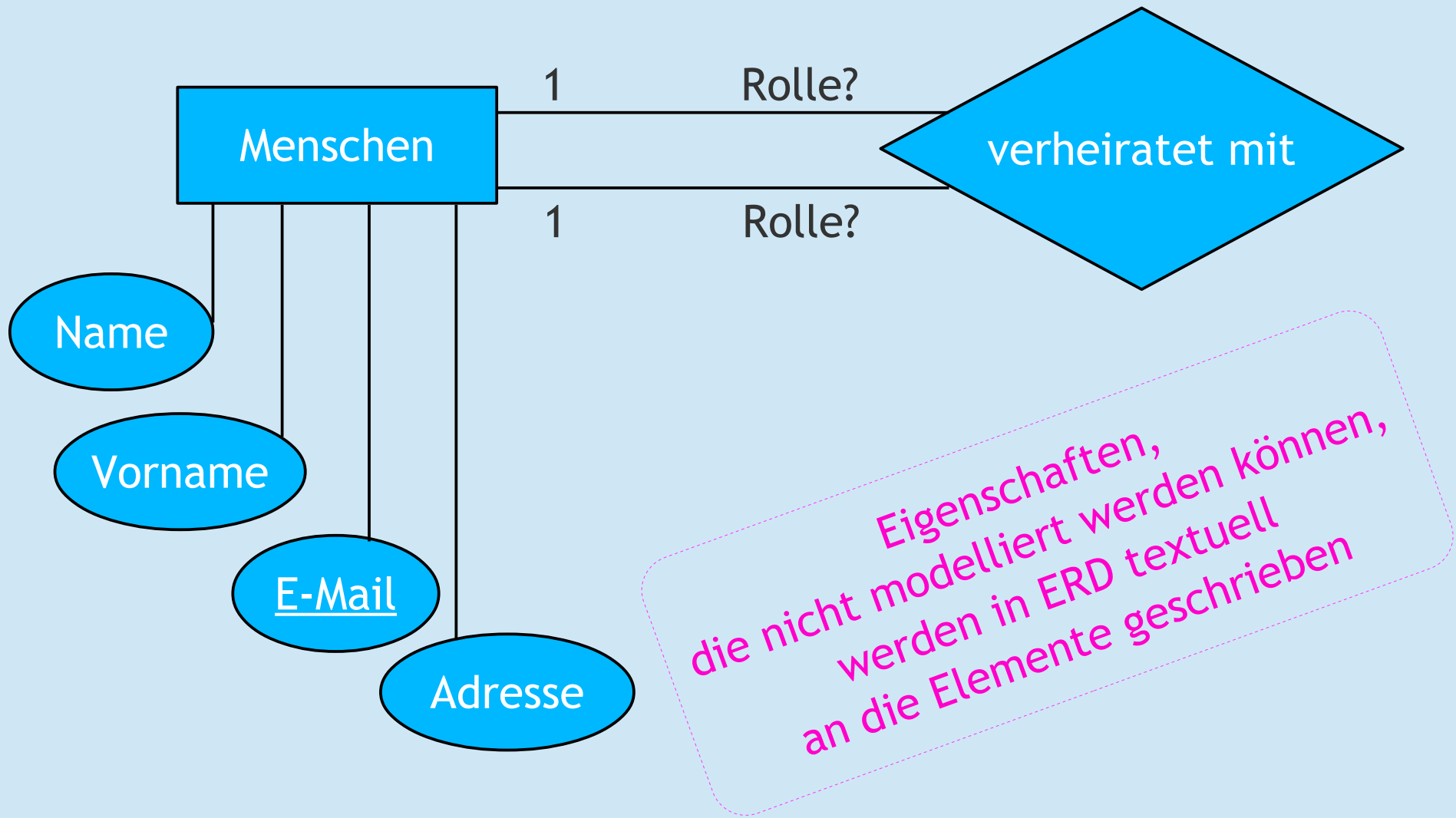
Es gibt:

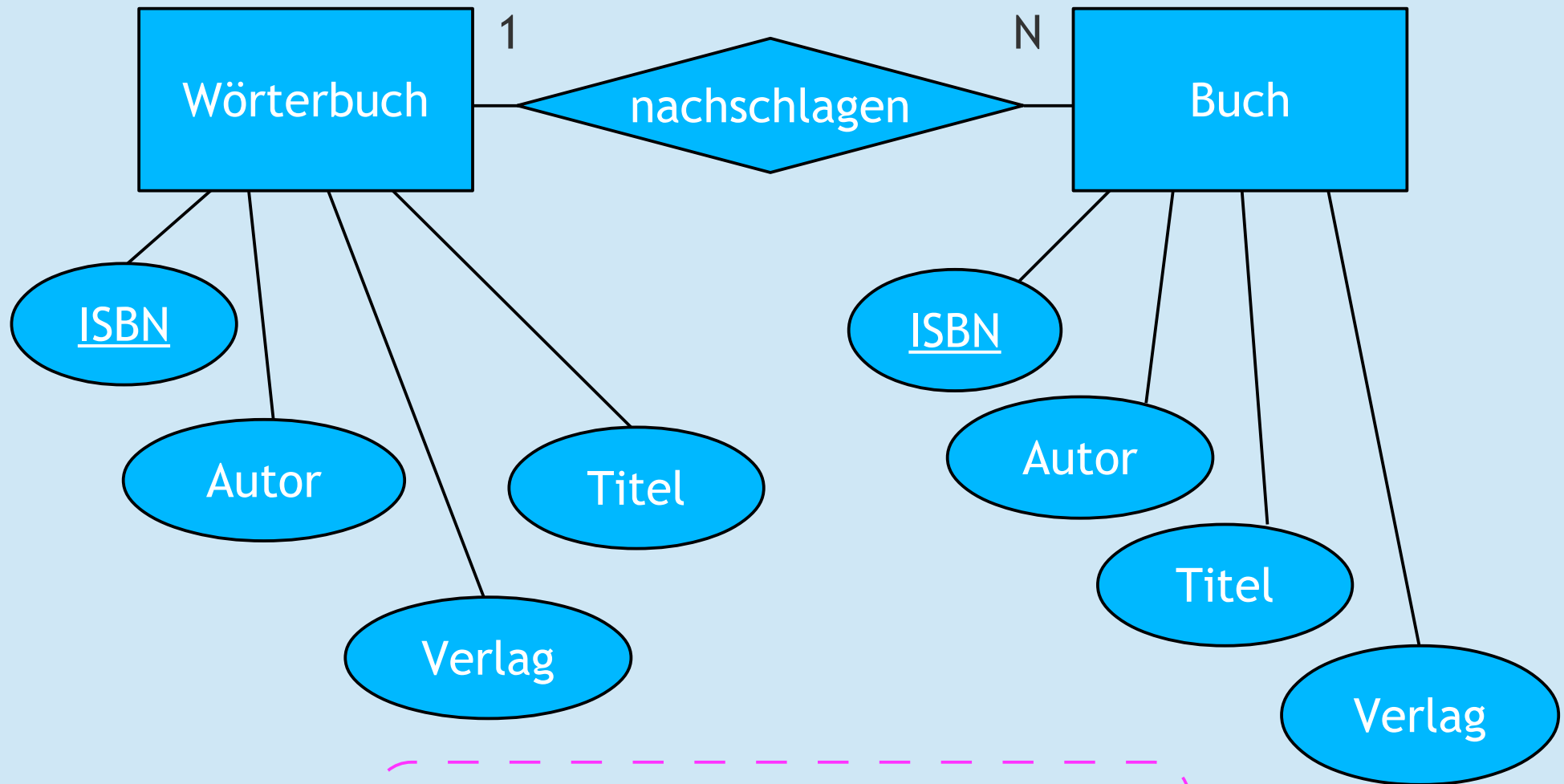
- *1:1-Beziehung. Einer Entität aus E_1 entspricht höchstens eine (oder keine) Entität aus E_2 , und umgekehrt einer Entität aus E_2 entspricht höchstens eine (oder keine) Entität aus E_1 .*
- *1:N-Beziehung. Einer Entität aus E_1 entsprechen mehrere (oder keine) Entitäten aus E_2 , umgekehrt aber einer Entität aus E_2 entspricht höchstens eine (oder keine) Entität aus E_1 .*
- *N:1-Beziehung. Dasselbe wie 1:N-Beziehung.*
- *N:M-Beziehung. Einer Entität aus E_1 entsprechen mehrere (oder keine) Entitäten aus E_2 , umgekehrt auch einer Entität aus E_2 entsprechen mehrere (oder keine) Entitäten aus E_1 .*



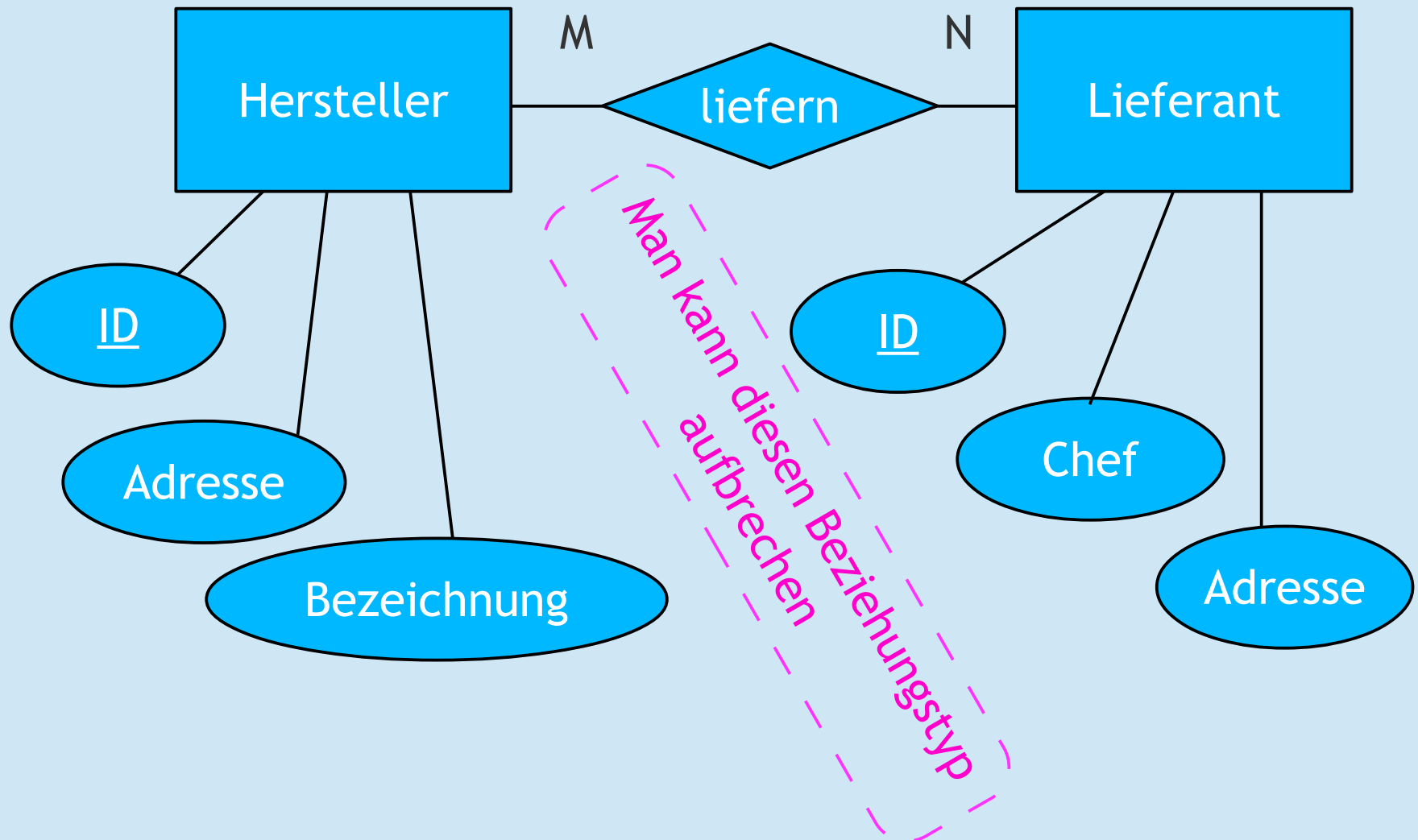






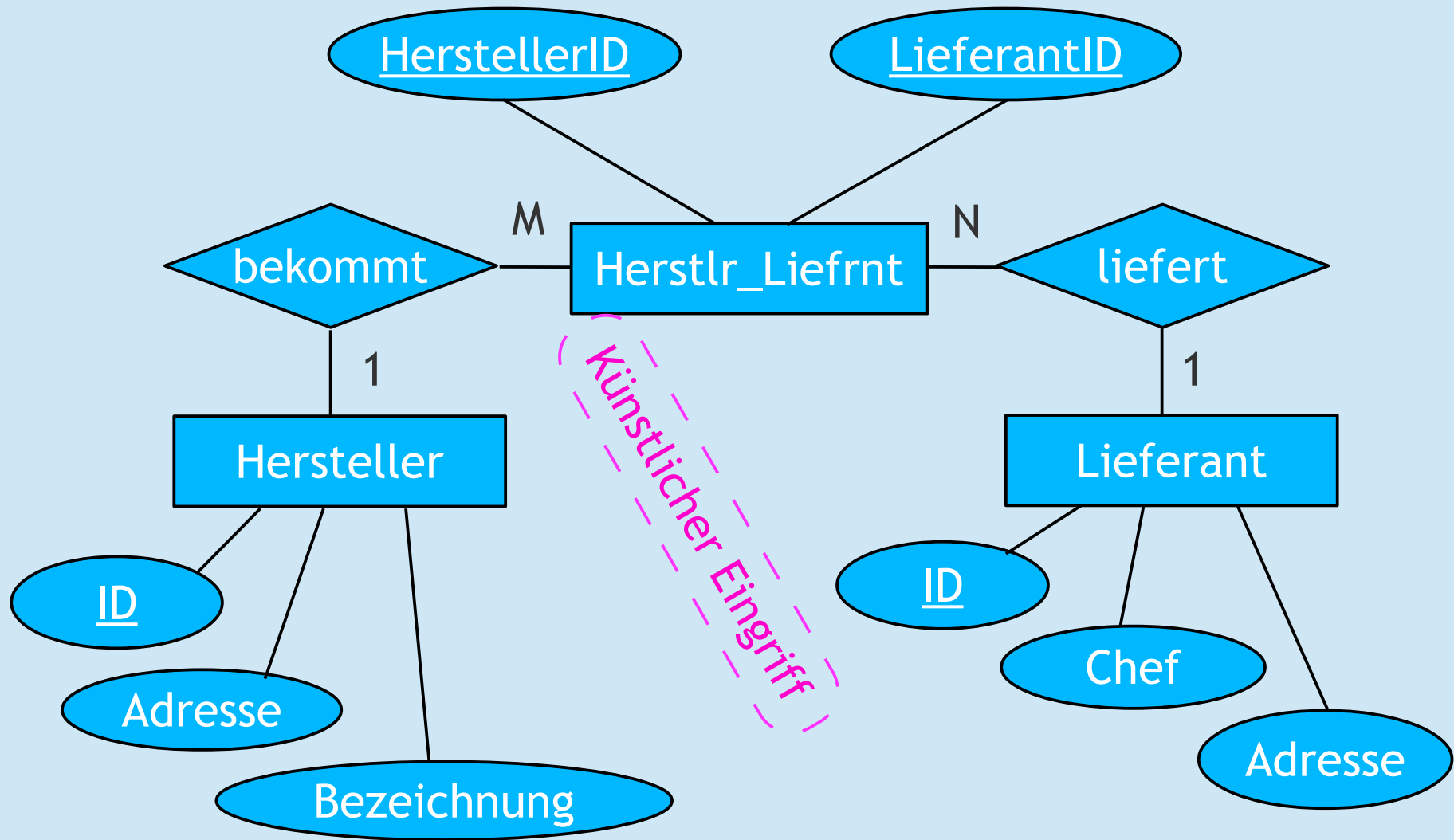


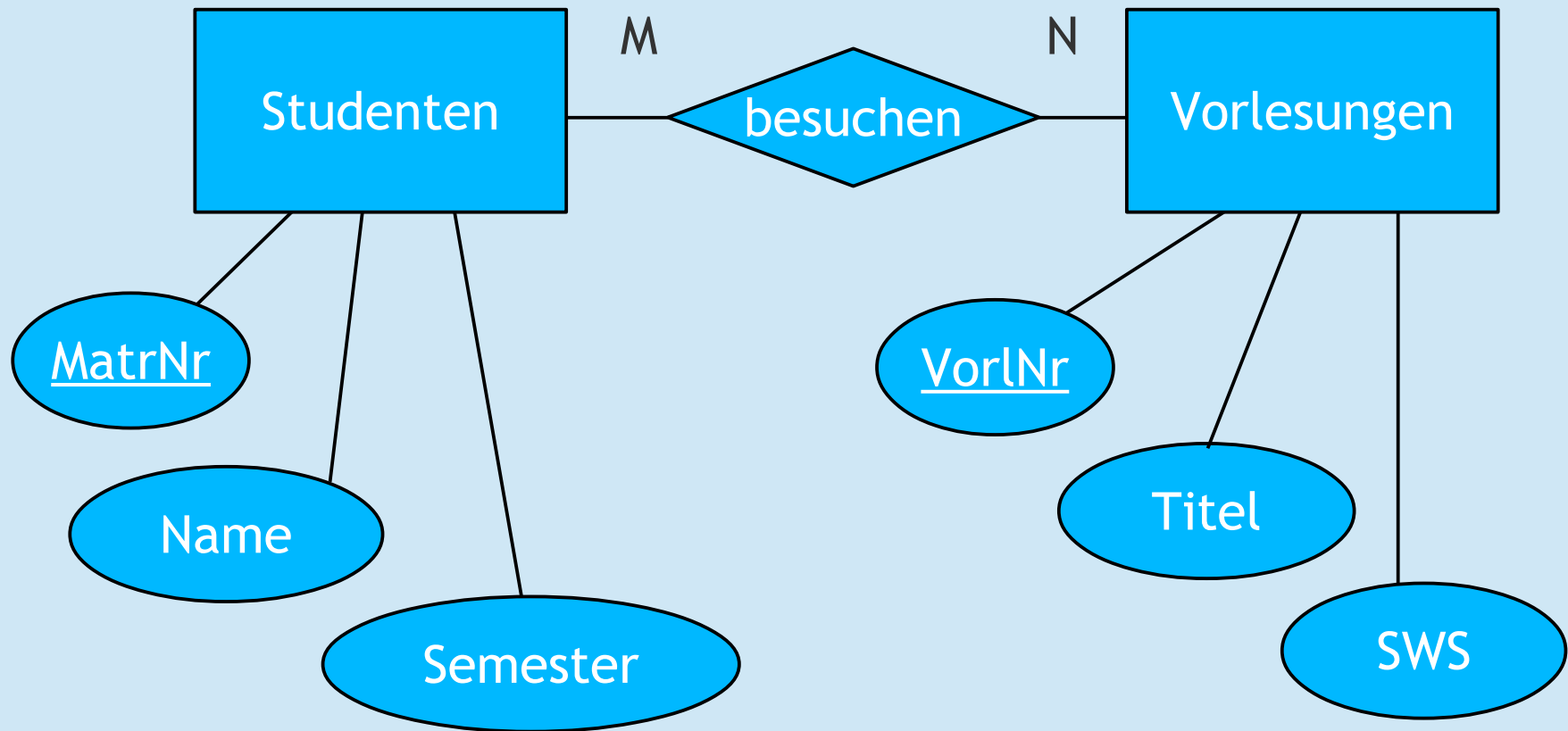
(Sind die Entitätstypen in Ordnung?)





Auflösen (Aufbrechen) einer M:N-Beziehung in zwei 1:N-Beziehungen:







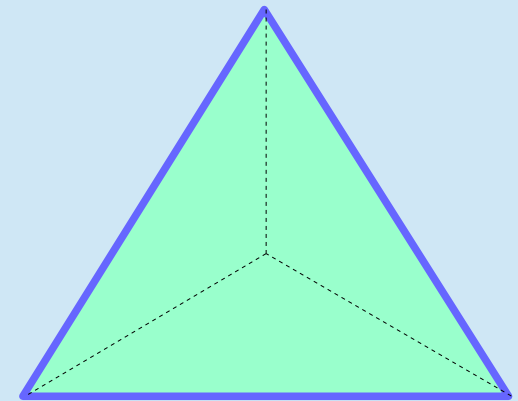
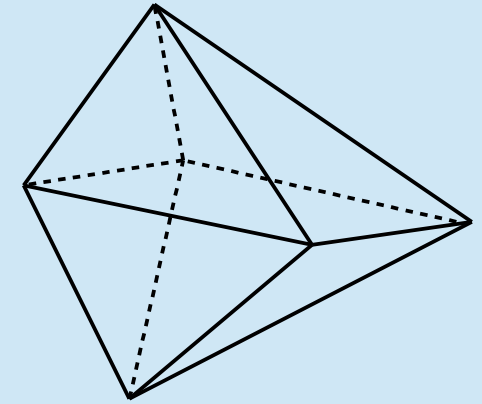
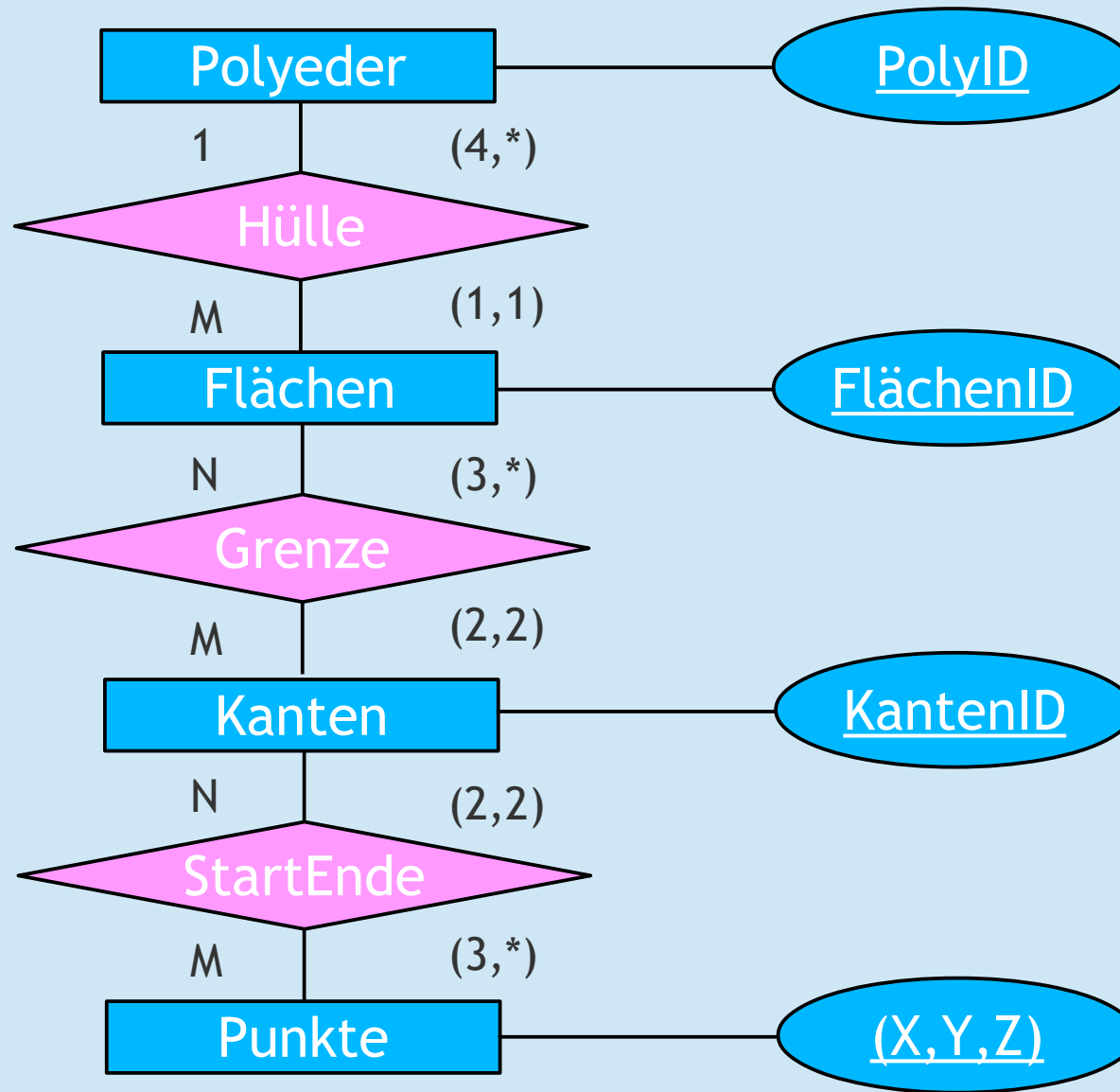
Die Funktionalitäten in der Standard-Form von Peter Chen beschreiben nur ungenaue Anzahl der Beziehungen (Beziehungsinstanzen) – eine oder viel.

Die (min,max)-Notation gibt die genauen Grenzen an: minimale und maximale Anzahl der an den Beziehungen beteiligten Entitäten.

Gibt es die Entitäten, die an keiner Beziehung teilnehmen, dann wird die minimale Anzahl mit 0 belegt.

*Gibt es beliebige Anzahl der Entitäten, die an einer Beziehung teilnehmen, dann wird die maximale Anzahl mit * belegt.*

Diese Zahlen heißen Kardinalitäten.



Tetraeder,
das kleinste Polyeder

[2-3]

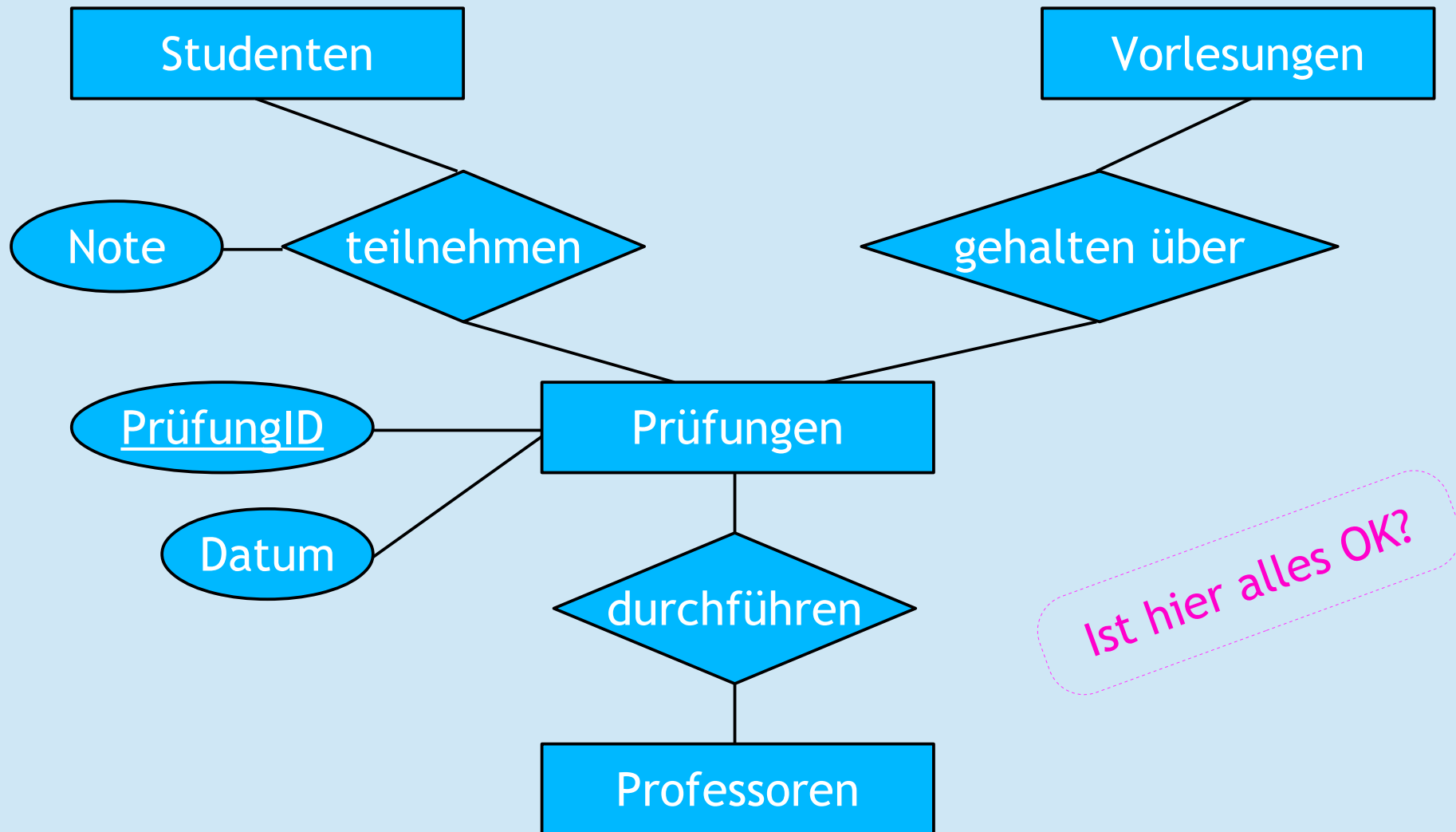


Ein Beziehungstyp zwischen 3 Entitätstypen heißt trinäre oder ternäre, zwischen n Entitätstypen – n -äre.

Beim konzeptuellen Entwurf sollte man versuchen, bei den binären Beziehungstypen zu bleiben.

Die n -ären Beziehungstypen kann man in die binären dadurch umwandeln, dass man die neuen (auch gedanklichen) Entitäts- und Beziehungstypen einführt. Diesen Schritt macht man am Ende des konzeptuellen Entwurfs.

Die Beziehung "prüfen" aus dem Uni-Modell kann man folgenderweise in die binären Beziehungen umwandeln:





Die Relationen werden als Untermenge (Teilmenge) des kartesischen Produktes der beteiligten Entitätstypen betrachtet.

$$R \subseteq E_1 \times E_2 \times \dots \times E_n$$

Wichtig ist – Relation ist nicht obligatorisch das ganze kartesische Produkt, sondern viel mehr eine Untermenge dieses Produktes. Die n-Tupel enthalten die entsprechenden Entitäten.

Das ganze kartesische Produkt hat selten einen Sinn.

Relationen stellen eine andere Ansicht auf die Modellierung der Daten dar.

Funktionalität einer Relation gibt an, wie deren n-Tupel aufgebaut sind, und hat somit einen Einfluss darauf, wie viele n-Tupel in der Relation vorhanden sind.



Es gibt folgende spezielle Konzepte der ERM:

- *Existenzabhängige Entitäten (schwache Entitäten, Komposition);*
- *Generalisierung;*
- *Aggregation.*

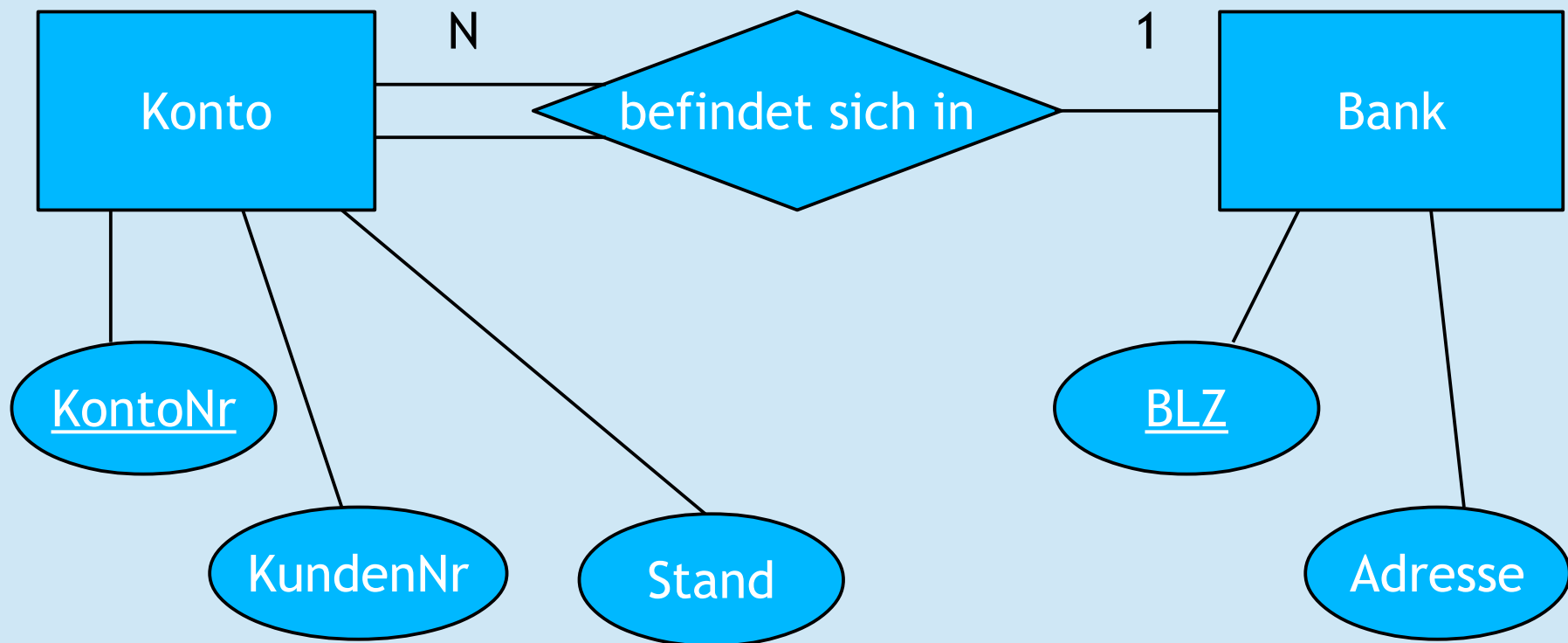
Ihre Eigenschaften:

- *sie dienen der Strukturierung des Anwendungsbereichs;*
- *sie werden von DBMS wenig unterstützt;*
- *sie verwenden die Konzepte der OOP (Vererbung).*



Man betrachtet bei Komposition unterschiedliche Entitätstypen, die aber in ihrer Gemeinsamkeit einen neuen Entitätstyp bilden (has-a).

Wichtig: Existenz eines Entitätstypen ist von der Existenz des anderen Entitätstypen abhängig.





Generalisierung ist eine Abstraktion auf der Ebene der Entitätstypen.

Man betrachtet gleichartige Entitätstypen. Man sucht in den Entitätstypen ähnliche Eigenschaften und fasst sie zu einem neuen Obertyp. Die alten Entitätstypen heißen in diesem Fall Untertypen.

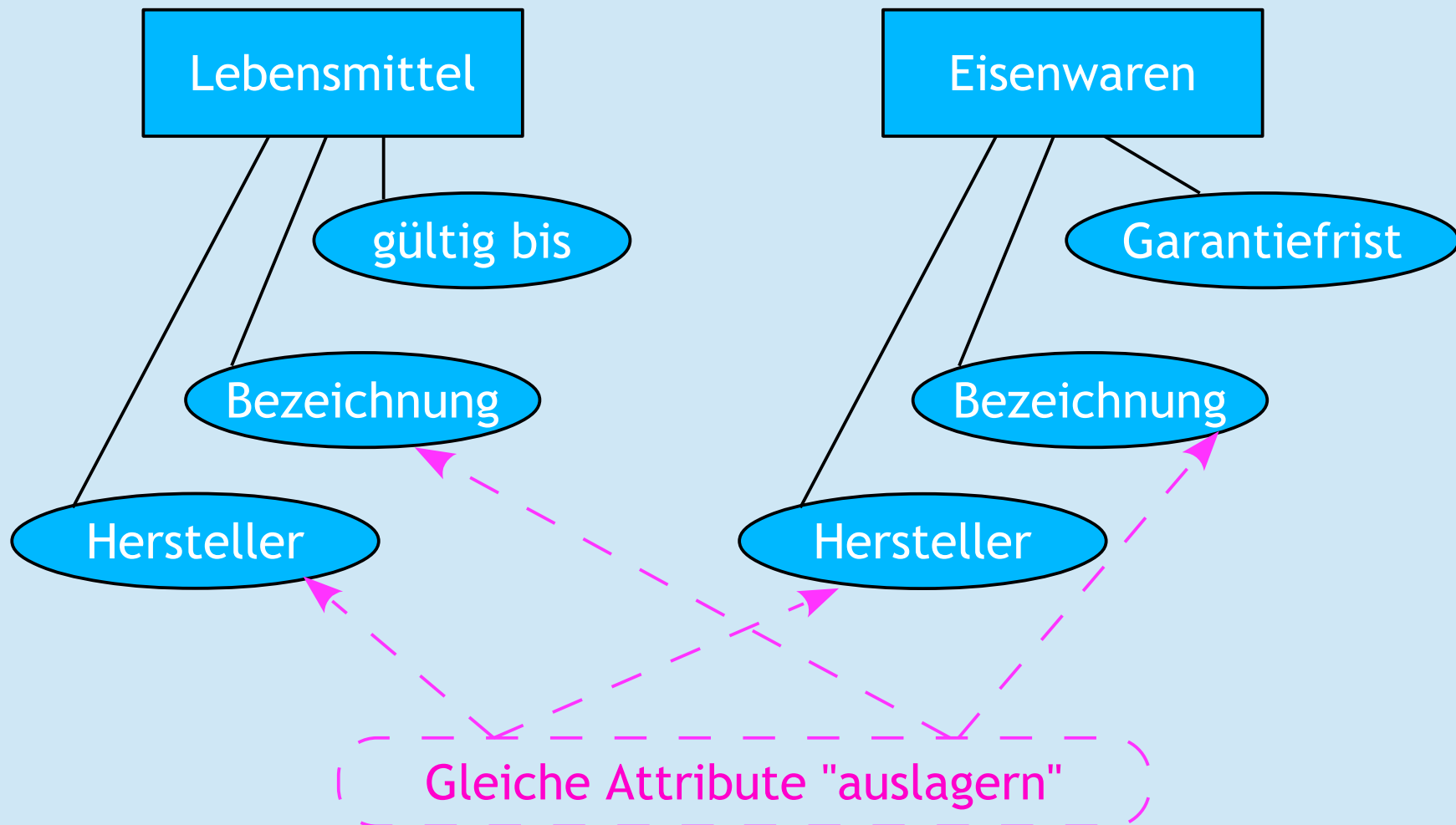
Die Eigenschaften, die sich nicht neu zuordnen lassen, bleiben bei den Untertypen.

Die Untertypen erben die gemeinsamen Eigenschaften von dem Obertyp.

Die Untertypen sind Spezialisierung des Obertyps.

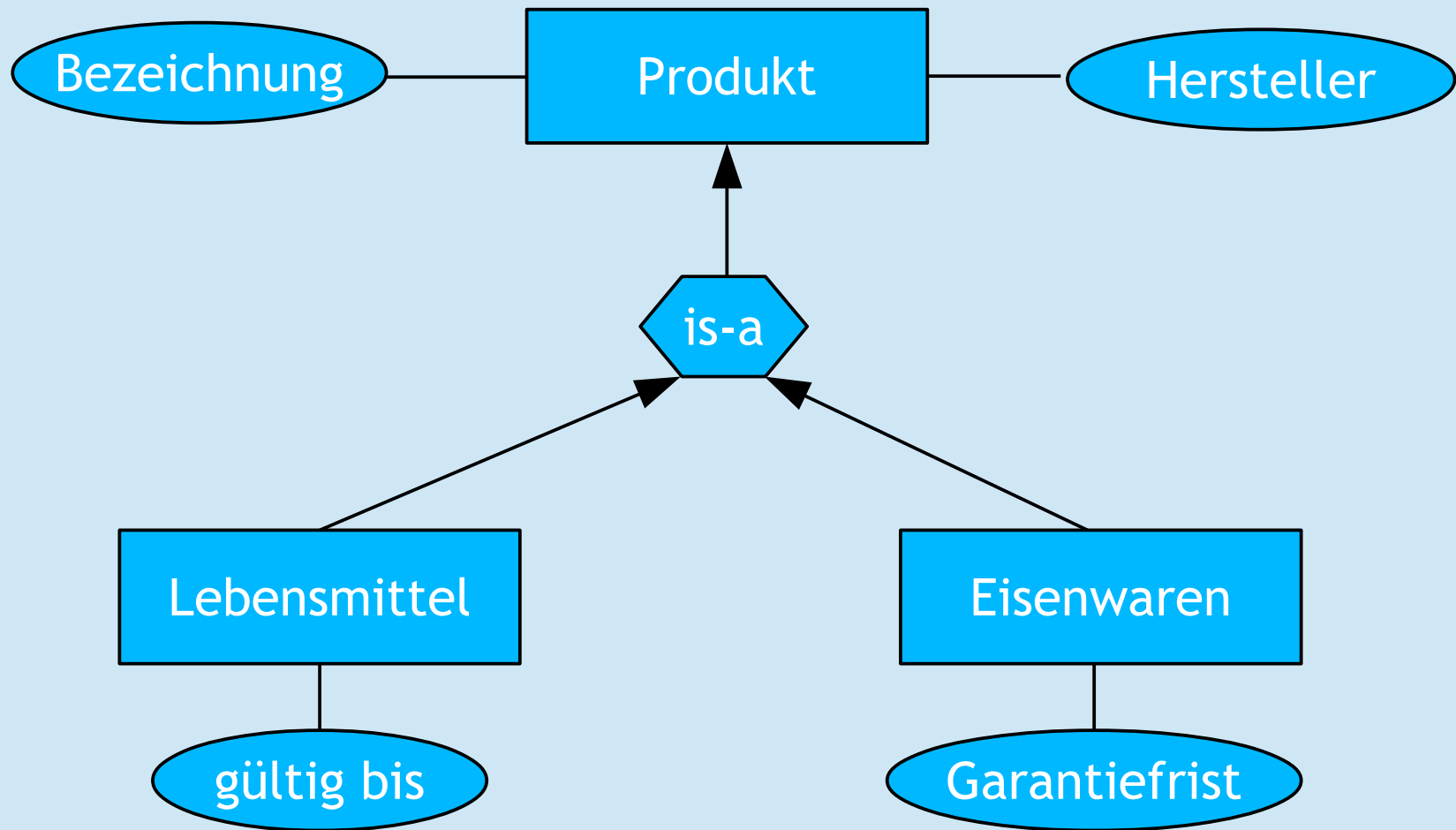


Vor der Generalisierung



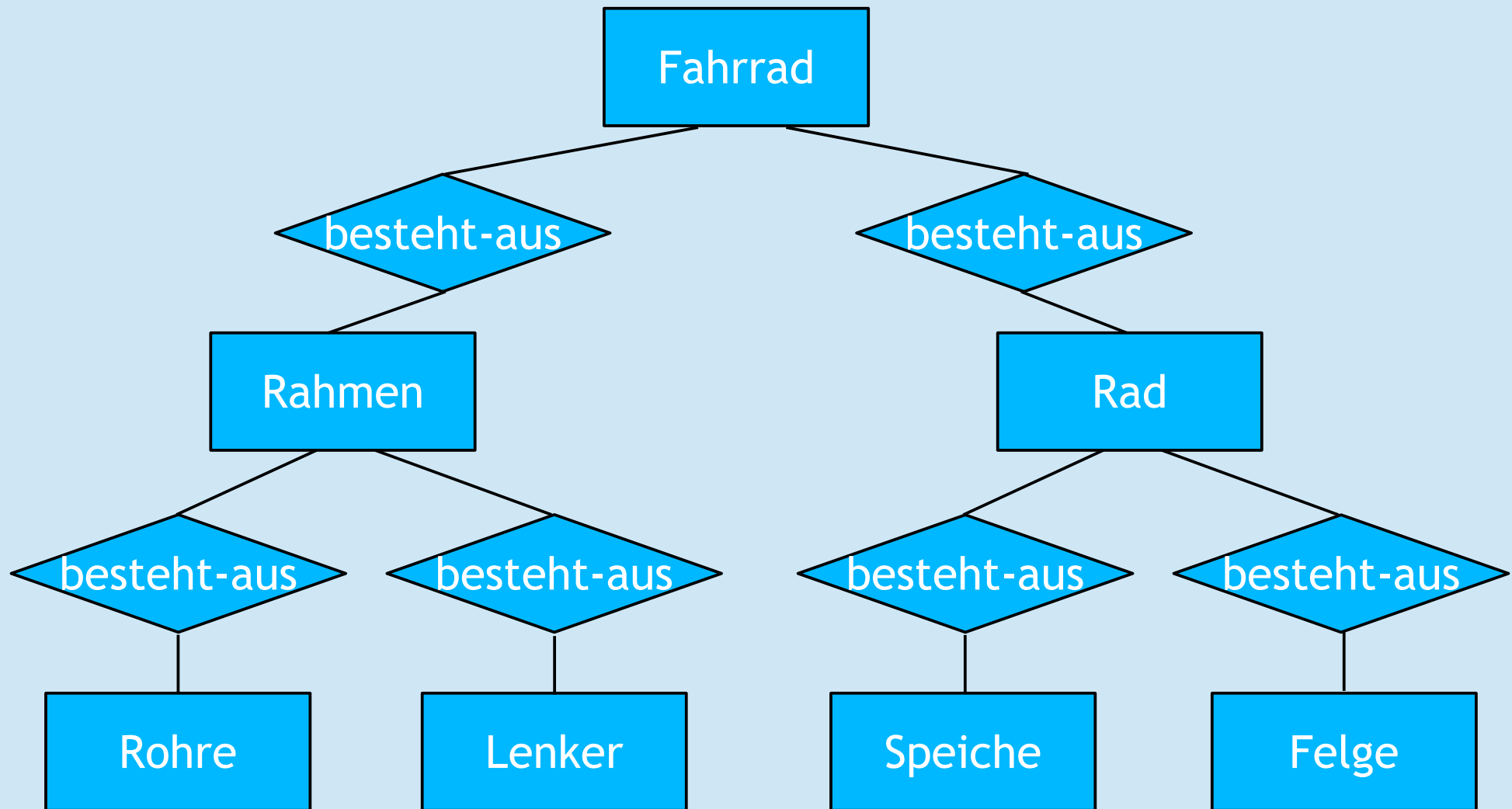


Nach der Generalisierung





Man betrachtet bei Aggregation unterschiedliche Entitätstypen, die aber in ihrer Gemeinsamkeit einen neuen Entitätstyp bilden (has-a).





Inhalt:

- Einführung
- Mengenlehre
- ANSI-SPARC-Modell
- Anforderungsanalyse
- Konzeptueller Entwurf
- Logischer Entwurf

- Definitionen
- Relationale Darstellung
- Vereinfachung der Darstellungen
- Zusammenfassung
- Wichtige Bemerkung



Ein Datentyp stellt die Werte und die darauf definierten Operatoren dar.

Datentyp = Werte + Operationen

Beispiele:

- *Reelle Zahlen (real) [+, -, *, /]*
- *Ganze Zahlen (integer) [+, -, *, /, mod]*
- *Text (string, text) [substr, length, +, pad]*
- *Datum, Uhrzeit (date, time) [-, +]*
- *Unstrukturierte Objekte (video, mem)*
- *Elektrischer Widerstand [R^+ , Reihenschaltung, Parallelschaltung]*



Der Sinn des logischen Entwurfes ist der Übergang vom ERM zu relationalen Darstellungen (Tabellen). Hier geht es immer noch nicht darum, in welcher konkreten Datenbank das Modell realisiert wird.

Definitionen:

- Eine Relation ist eine Untermenge des Kartesischen Produktes von mehreren Datentypen. Relation = Tabelle.*
- Eine Zeile in der Relation heißt Tupel, Record, Datensatz.*
- Eine Spalte in der Relation heißt Feld, Attribut, Eigenschaft. Alle Daten in einer Spalte gehören zu einem Datentyp.*
- Das Schema der Relation ist die Zusammensetzung der Namen aller Felder, deren Datentyp und Länge, sowie die Reihenfolge der Felder.*



Vorteile des relationalen Modells:

Sowohl Gegenstände/Entitäten als auch Beziehungen/Relationen von ERM werden im relationalen Modell als Tabellen (Relationen) dargestellt.

Als Folge werden sowohl Entitäten, als auch Beziehungen mit den selben Operationen verarbeitet.

Man muss in dem relationalen Modell (in der relationalen Datenbank) nur die mathematisch begründeten und von dem E.F.Codd gut entwickelten algebraischen Operationen verwenden.

Diese Operationen stellen die relationale Algebra dar. Sie sind in jeder relationalen Datenbank als Bestandteil des DBMS einprogrammiert.

Der Hauptteil der Operationen ist in jeder Datenbank vorhanden, und nur einige wenige Operationen sind von Datenbank zu Datenbank unterschiedlich.



Neben den Tabellen möglich ist auch diese relationale Darstellung:

RelationsName =

$\{ [\text{Feld1:Datentyp1}, \text{Feld2:Datentyp2}, \text{Feld3:Datentyp3}, \dots] \}$

Hier

$[\dots]$ ist Bezeichnung des Datensatzes;

$\{ \dots \}$ ist Bezeichnung der Menge.

Beispiel:

$\text{Auto} = \{ [\underline{\text{KFZ:string}}, \text{Modell:string}, \text{Gewicht:real}, \text{Baujahr:integer}] \}$

$\text{Auto} = \{ \text{AutoDS} \}$ /* Menge von Datensätzen */

$\text{AutoDS} = [\underline{\text{KFZ:string}}, \text{Modell:string}, \text{Gewicht:real}, \text{Baujahr:integer}]$

Wichtig: Primärschlüssel muss unterstrichen werden.



Die Konvertierung eines ERM in eine relationale Darstellung passiert durch einfache Regeln:

Regeln für die Darstellung der Entitätstypen:

- Man bildet eine neue Tabelle (Entitäts-Tabelle).
- Man inkludiert in diese Tabelle alle Attribute des Entitätstypes.

Regeln für die Darstellung der Beziehungstypen:

- Man bildet eine neue Tabelle (Beziehungs-Tabelle).
- Man inkludiert in diese Tabelle die Primärschlüssel von allen durch diese Beziehung verbundenen Entitätstypen (jeder Entitätstyp hat nur einen Primärschlüssel!) – alle Schlüssel bilden i.d.R. den Primärschlüssel der Beziehungs-Tabelle.
- Man inkludiert in diese Tabelle die Attribute des Beziehungstyps.



In folgenden Fällen kann man die Darstellung für die Entitäten und Beziehungen vereinfachen:

- *1:1-Beziehungen.*
- *1:N-Beziehungen.*
- *Schwache (existenzabhängige) Entitäten, eigentlich auch 1:N-Beziehungen.*

Vereinfachung bedeutet:

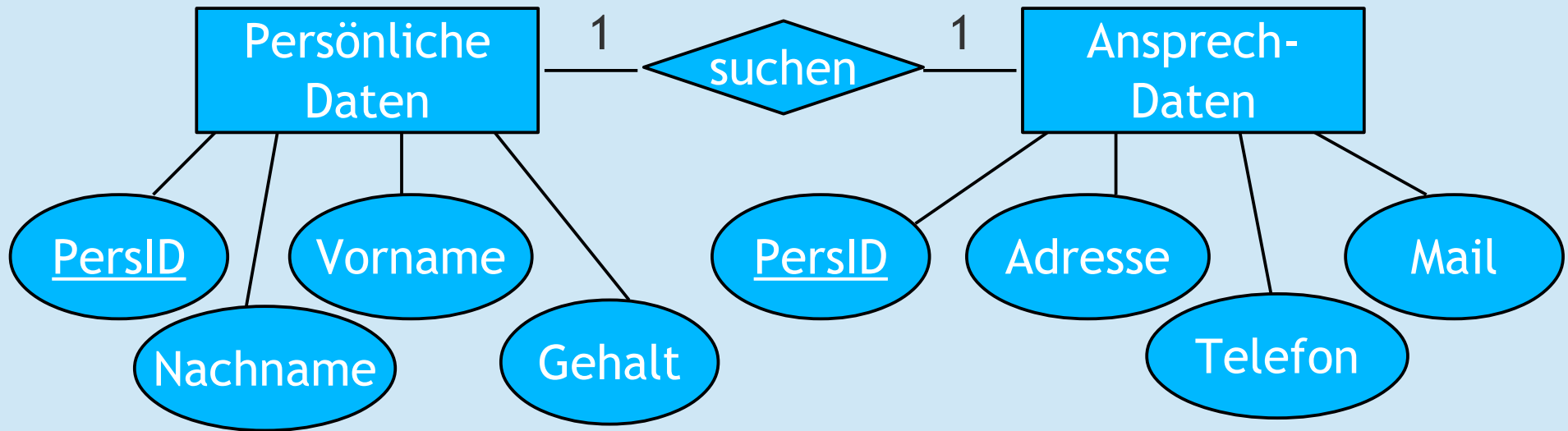
In diesen Fällen kann man auf Beziehungs-Tabellen verzichten, aber dabei werden die Entitäts-Tabellen unbedingt angepasst. Man hat weniger Tabellen im logischen Entwurf, als beim Standard-Verfahren, deswegen wird dieser optimierte Zustand theoretisch bevorzugt.

Man muss diesen Prozess gut überlegen, weil er nicht unbedingt zu der Verbesserung der Leistungen des Datenbank-Entwurfs führt. Man muss berücksichtigen, wie oft man auf die beteiligten Tabellen zugreift.

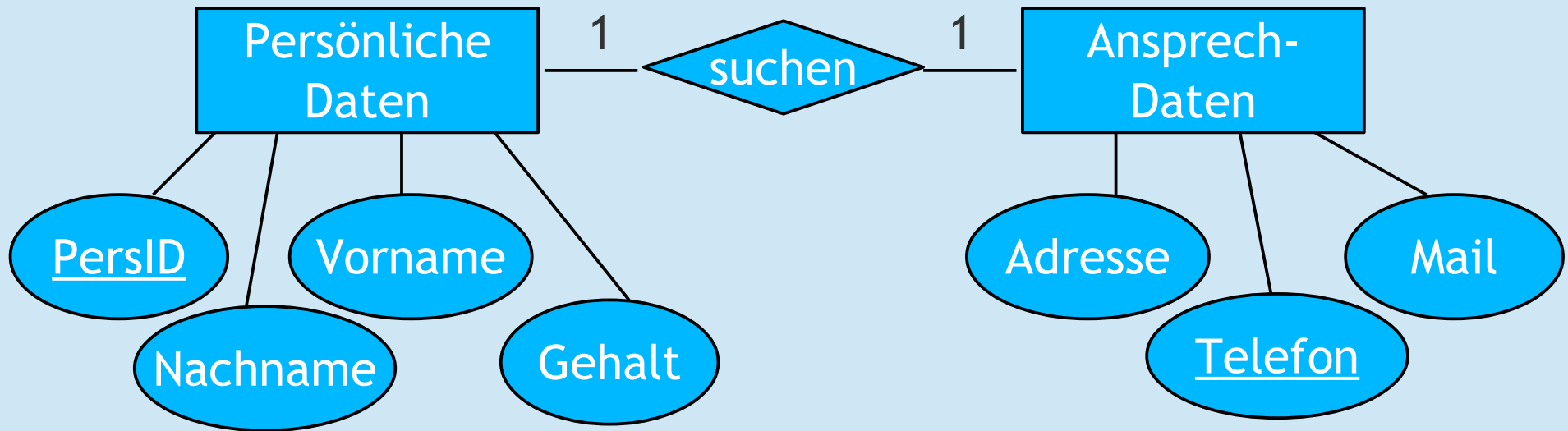


Im Fall der 1:1-Beziehungen folgt man einer der beiden Empfehlungen:

- Die beteiligten Entitäts-Tabellen zu einer Tabelle zusammenführen, und dann ggf. die Attribute der Beziehung hinzufügen. Statt zwei Entitätstypen und einem Beziehungstyp entsteht nur eine (vielleicht ziemlich große) Tabelle. Der Primärschlüssel darf selbstverständlich nur einmal vorkommen, er wird aus einem oder aus anderem Entitätstyp übernommen. Die Attribute aus beiden Entitätstypen, die gleiche Semantik aufweisen, müssen ebenfalls nur einmal vertreten sein. Aber Vorsicht – die Datensätze können zu groß sein, um sie effektiv von dem Datenträger zu laden. Die Attribute der Beziehung werden in diese Tabelle inkludiert.
- Man verzichtet nur auf die Beziehungs-Tabelle. Die beiden Entitäts-Tabellen bleiben, aber in eine wird der Primärschlüssel aus der anderen hinzugefügt, falls die Primärschlüssel in beiden Entitätstypen unterschiedliche Semantik haben. Die Attribute der Beziehung werden in eine der Entitäts-Tabellen inkludiert.



<u>PersID</u>	Nachname	Vorname	Gehalt	Adresse	Telefon	Mail
42	May	Karl	5000	13 Go Home, Texas, USA	001-2345	karl.may@gmx.de



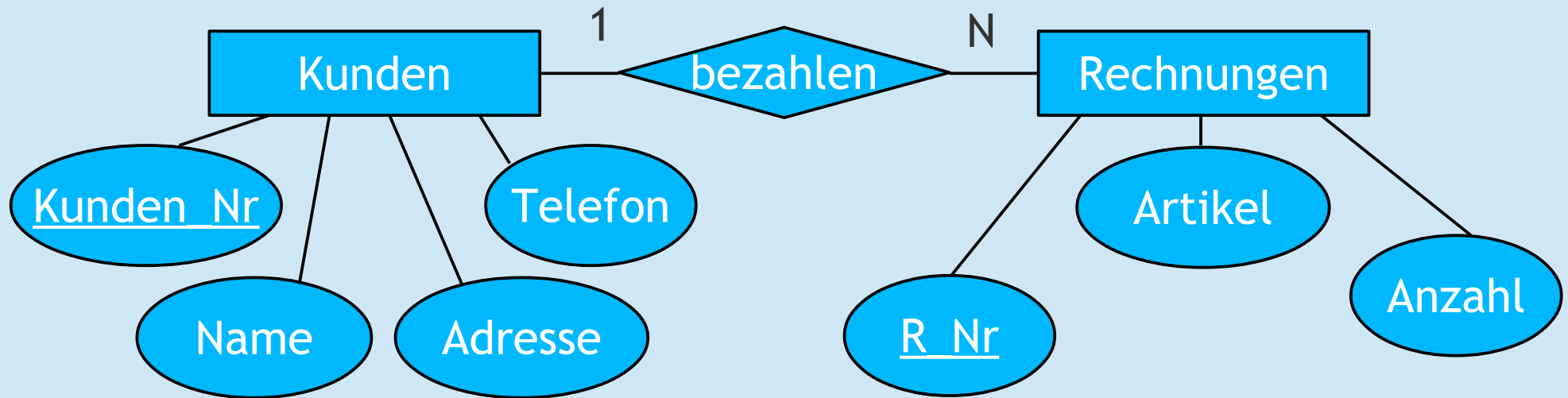
<u>PersID</u>	Nachname	Vorname	Gehalt	Adresse	Telefon	Mail
42	May	Karl	5000	13 Go Home, Texas, USA	001-2345	karl.may@gmx.de



Im Fall der 1:N-Beziehungen kann man die Bildung der Beziehungstabelle ebenfalls vermeiden.

Man fügt in die Tabelle, die an der N-Seite der Beziehung steht, den Primärschlüssel der Tabelle, die an der 1-Seite steht, hinzu. Der eingefügte Schlüssel wird in diesem Fall zum Fremdschlüssel. Der Primärschlüssel der Tabelle, die an der N-Seite der Beziehung steht, bleibt unverändert. Die Attribute der Beziehung werden in die Tabelle an der N-Seite inkludiert.

Im Gegenteil zu dem vorigen Fall führt es nicht zur erheblichen Vergrößerung des Datensatzes.



<u>Kunden_Nr</u>	Name	Adresse	Telefon
42	May	13 Go Home, Texas, USA	001-2345

<u>R_Nr</u>	Kunden_Nr	Artikel	Anzahl
247-2013	42	Schießbogen	3
534-2013	42	Pfeile	120



Im Fall der existenzabhängigen Entitäten kann man die Bildung der Beziehungs-Tabelle ebenfalls vermeiden.

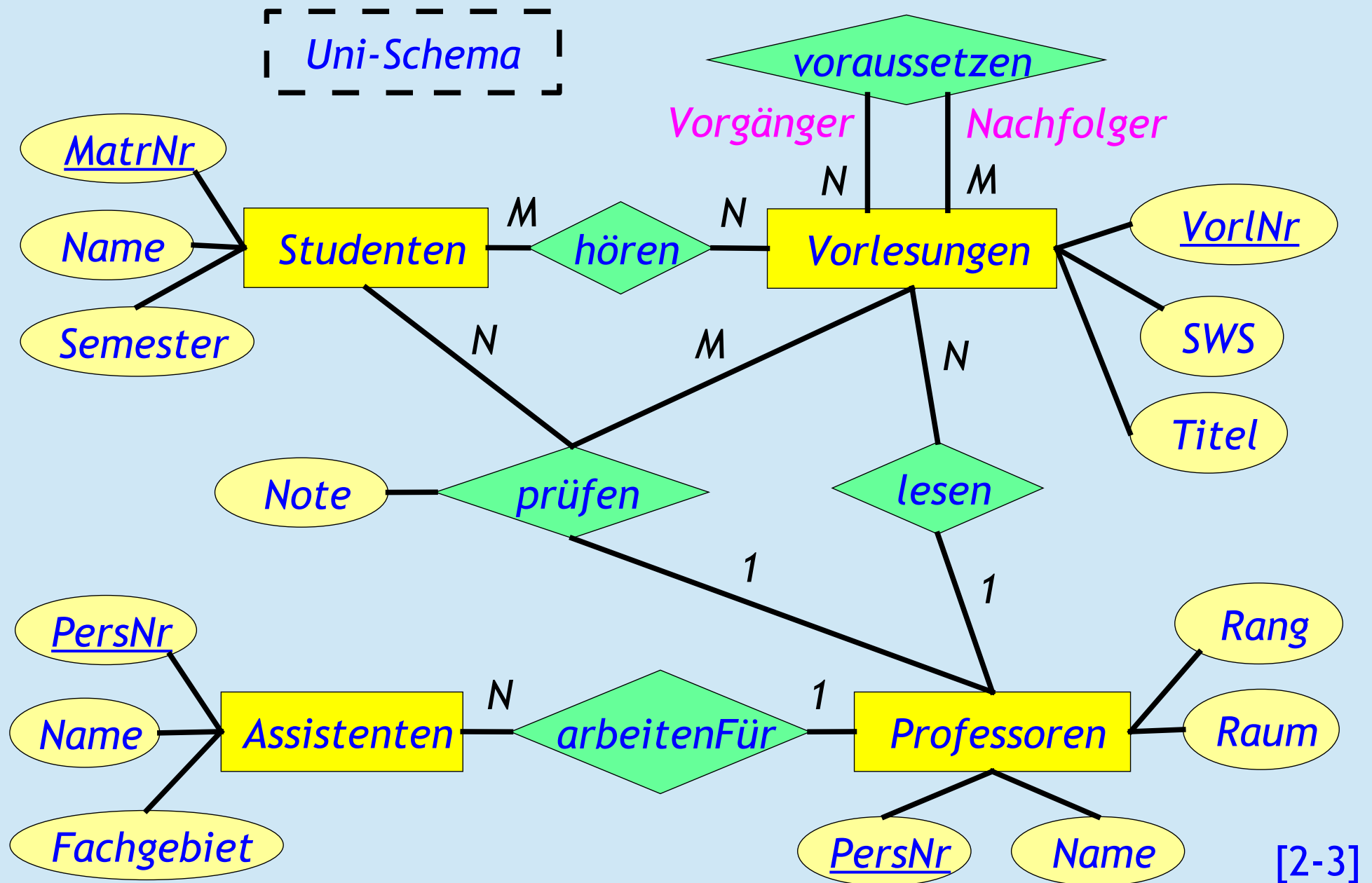
Man bildet (selbstverständlich) eigene Tabelle für die schwache Entität. Man inkludiert in die Tabelle der schwachen Entität den Primärschlüssel der referenzierten (starken) Entität, somit wird dieser Schlüssel zum Teil des Primärschlüssels in der Tabelle der schwachen Entität. Das ist der Unterschied zu den vorher beschriebenen 1:N-Beziehungen.

Es gibt sogar keine andere Möglichkeit, die schwachen Entitäten darzustellen.



<u>KontoNr</u>	KundenNr	Stand	<u>BLZ</u>
11 22 33 44 55 66	42	837,46 €	100 500 00

<u>BLZ</u>	Adresse
100 500 00	Musterstr. 20, 12345 Berlin





Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	SWS	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

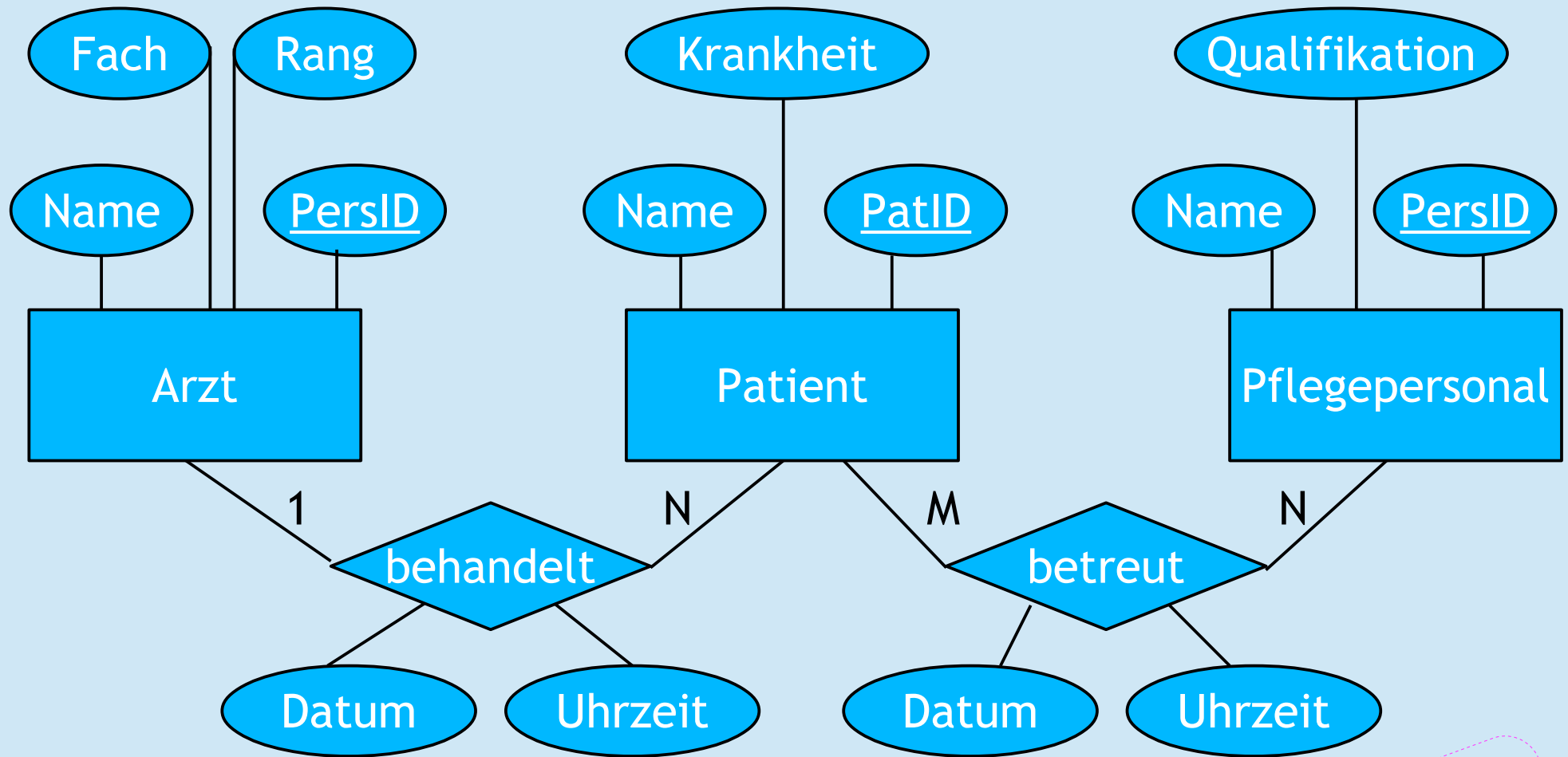
voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Assistenten			
PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

[2-3]



Peter-Chen-Notation



Arzt

Name	Fach	Rang	<u>PersID</u>
Arnold	Orthopädie	R-7	A-1

Patient

<u>ArztPersID</u>	Name	Krankheit	<u>PatID</u>	Datum	Uhrzeit
A-1	Paul	Beinbruch	P-13	01.02.2025	13:00

Mini-Welt:
Der Arzt kommt
nur einmal
zum Patienten

Pflegepersonal

Name	Qualifikation	<u>PersD</u>
Susi	med. Schwester	S-531

betreut

<u>PflegePersID</u>	<u>PatID</u>	<u>Datum</u>	<u>Uhrzeit</u>
S-531	P-13	03.01.2025	12:00



Arzt

Name	Fach	Rang	<u>PersID</u>
Arnold	Orthopädie	R-7	A-1

Patient

Name	Krankheit	<u>PatID</u>
Paul	Beinbruch	P-13

Pflegepersonal

Name	Qualifikation	<u>PersD</u>
Susi	med. Schwester	S-531

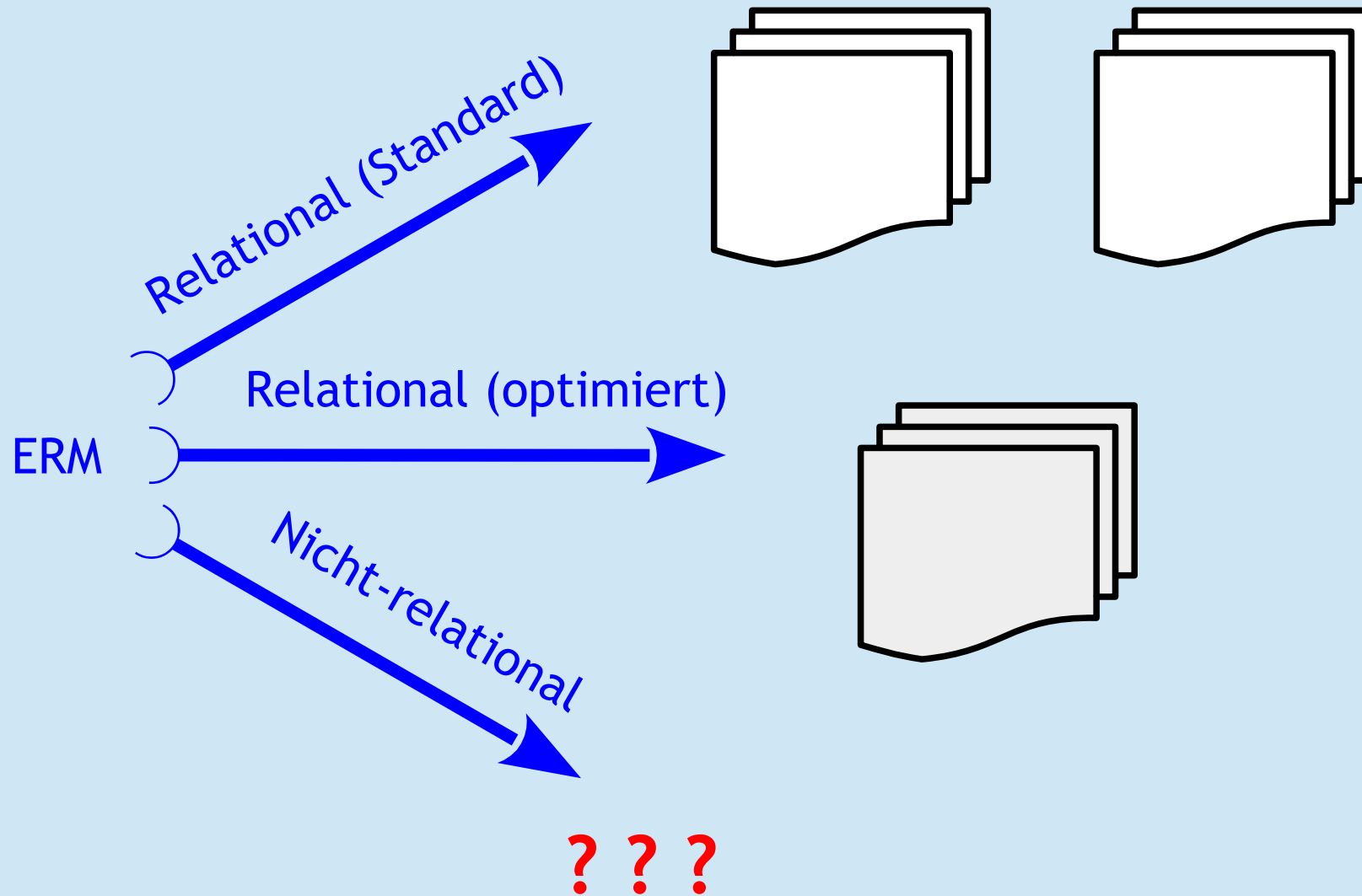
Mini-Welt:
Der Arzt kommt
mehrmals
zum Patienten

behandelt

<u>ArztPersID</u>	<u>PatID</u>	<u>Datum</u>	<u>Uhrzeit</u>
A-1	P-13	01.02.2025	9:00

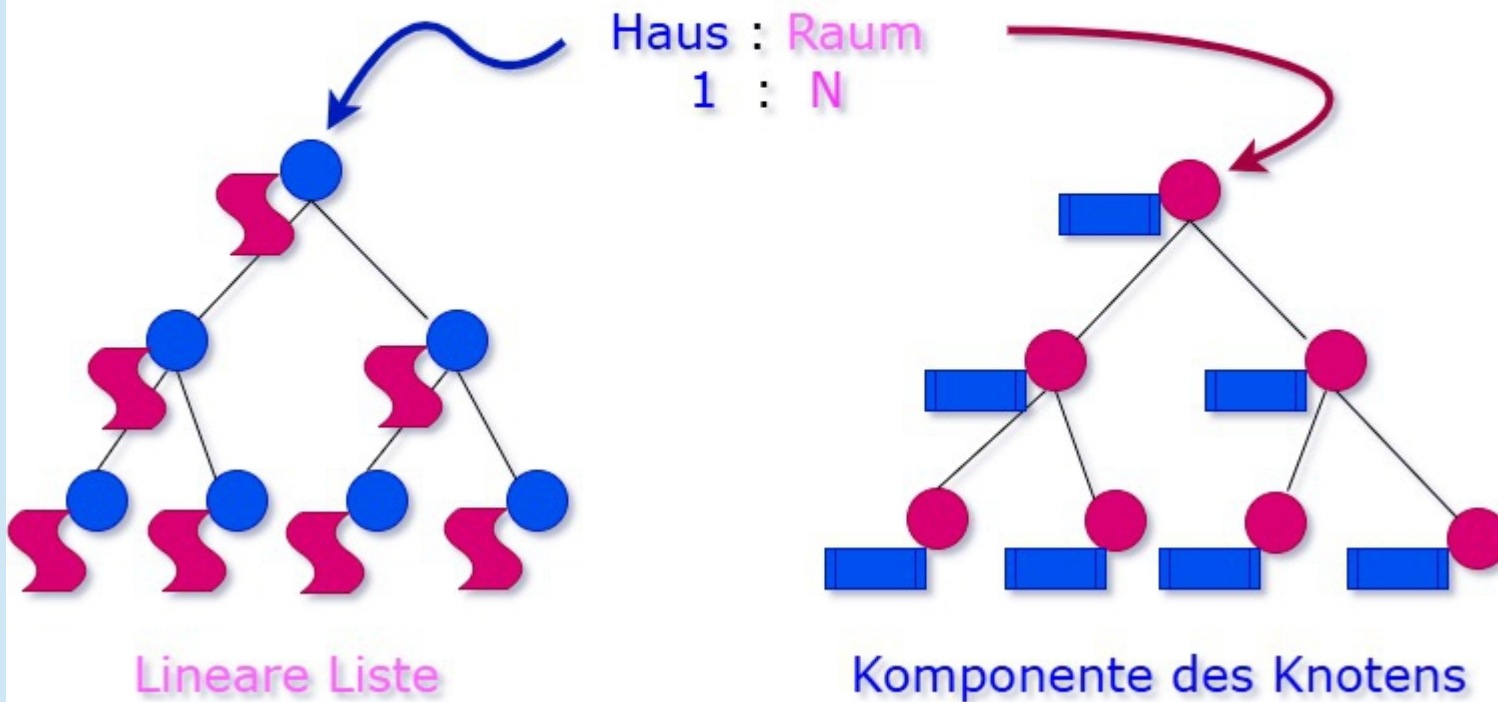
betreut

<u>PflegePersID</u>	<u>PatID</u>	<u>Datum</u>	<u>Uhrzeit</u>
S-531	P-13	03.01.2025	12:00





Alternative nicht-relationale Umsetzung vom ERM





Beim Übergang zum logischen Entwurf dürfen keine Felder/PK/FK willkürlich in Tabellen entstehen! Nur laut den Vereinfachungsregeln!

Ausnahmen:

RowID RowNum

- *ERM-bezogene Gründe.
Beispiele: Prozesse PID, MonsterID, FensterID unter MS Windows.*
- *Auditing, firmeninterne oder gesetzliche Richtlinien.
Beispiele: "Alle Zeilen in JEDER Tabelle müssen eindeutig und fortlaufend nummeriert werden".*
- *Administrative Gründe.
Beispiele: Fusion von großen Datenmengen, Anwendungsfehler.
Dabei können gleiche Datensätze in einer Tabelle entstehen, und diese Datensätze müssen irgendwie gelöscht werden.*
- *Anwendungsbezogene Gründe.
Beispiele: Einige DB-Software (objektorientierte DB) definieren und pflegen die PK selbst, ohne Einmischung der Benutzer.*



Beim Übergang zum logischen Entwurf dürfen keine Felder/PK/FK willkürlich in Tabellen entstehen! Nur laut den Vereinfachungsregeln!

Primärschlüssel

Rechnungen

<u>RID</u>	Datum	Verkaeuer
1	12.12.18	Müller
2	15.12.18	Meier
3	15.12.18	Müller

Rechnungspositionen

<u>RPID</u>	RID	ProdID	Anzahl
1	1	1	2
2	1	2	1
3	1	3	5
4	2	1	3
5	2	4	4
6	3	1	2
7	3	3	3

Produkte

<u>ProdID</u>	Pname	Preis
1	Soft	5
2	Limo	7
3	Cola	2
4	Wasser	1

Dieses Feld wird niemals verwendet, wird aber ständig gepflegt.
Ressourcen werden unnötig verbraucht!



Achtung!

Einige Autoren (siehe [6] Kapitel 5.2) empfehlen ausdrücklich den Gegenteil. Gründe:

- PK soll keinerlei die semantischen Informationen enthalten.
- PK darf sich nicht "plötzlich" mit der Zeit ändern (z.B. Artikelnummer).
- PK soll einfach aufgebaut sein, also nicht aus mehreren Feldern bestehen.

ID-Felder?
Ja, aber mit Bedacht!

Diese Meinung darf selbstverständlich auch existieren, sie ist übrigens in Wirtschaft sehr verbreitet, und viele industrielle Datenbanken sind so aufgebaut (wegen 2NF).

Im Unterricht (Verständnisses halber) wird hier auf diese Meinung aber verzichtet. Und zu jedem Punkt gibt es auch eine Widerrede aus der Sicht des Unterrichts.

