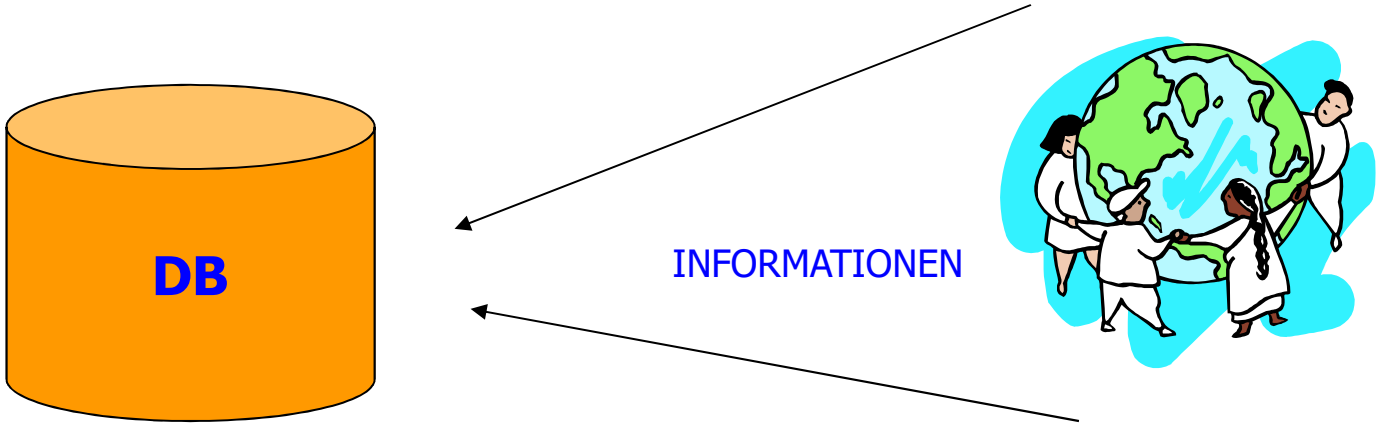
Datenbank-Modellierung

- ➔ Darstellung einer realen Situation in einem Entity-Relationship-Modell?
- ➔ Umwandlung des ER-Modells in das relationale Datenbankmodell?

Vom Konzept zur Datenbank

Eine Datenbank ist ein **Ausschnitt** der **Realität**.



Sie enthält **reale Informationen** (z.B. CD, Interpret, Mitarbeiter, …) in strukturierter Form.

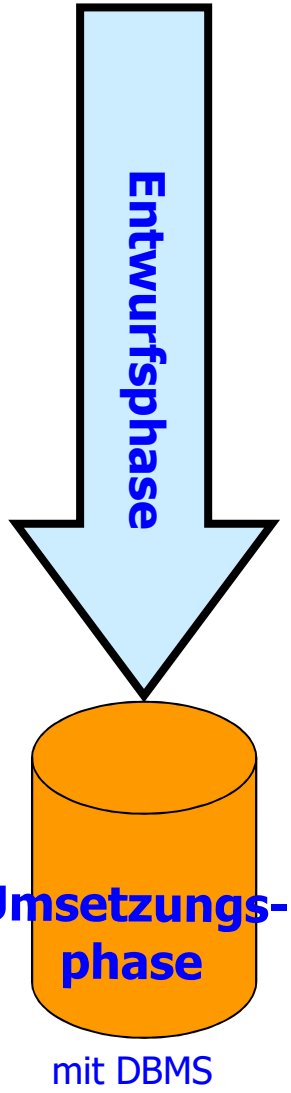
Schritte der DB-Entwicklung

1. Informationsanalyse

2. Beschreibung mit Entity-Relationship-Modell (ERM)

3. Beschreibung mit Hilfe eines Datenmodells
(zB. dem relationalen Datenmodell)

4. Implementierung der Datenbank



Schritte der DB-Entwicklung

Anforderungsanalyse

Sammeln und Analysieren der Anforderungen an die neue DB

Informationsanforderung

- Welche Inhalte (Umweltobjekte) sollen in die DB?
- Welche konkreten Eigenschaften der Objekte sollen gespeichert werden?

Beziehungsanalyse

- In welchen Beziehungen stehen die Umweltobjekte zueinander?

Benutzergruppenanalyse

- Welche Mitarbeiter sollen mit dem Datenbestand arbeiten?
- Können Benutzergruppen (gleiche Lese- bzw. Schreibrechte) gebildet werden

Verarbeitungsanforderungen

- Welche konkrete Einzel-Daten werden von den Benutzern lesend genutzt (Sichten, Berichte, Seriendruck usw.)
- Welche konkreten Einzel-Daten müssen von welchen Benutzern bearbeitet werden (Schreibzugriff über gespeicherte Prozeduren)

Einsicht in vorhandene Dokumentation

Fragebögen und Besprechungen mit Betroffenen des modellierten Umweltausschnitts

Notation des Entity-Relationship-Modells – Objekte (Entities)

Objekte: Dinge der realen Welt – z.B: Personen, CDs, Schüler, Lehrer, Mitarbeiter,…

Objekte der gleichen Art werden zu Objekttypen (Entity) zusammengefasst.

➔ Die CDs ChartShow, Best Of, … gehören zur Entity **„CD"**

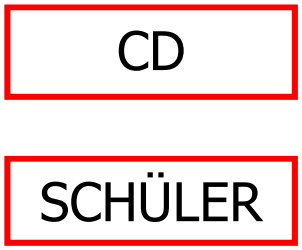
➔ Die Schüler Huber, Meier und Müller gehören zur Entity **„Schüler"**

Darstellung im ERM:

➔ als Rechteck

➔ Mit Großbuchstaben beschrieben

➔ Einzahl



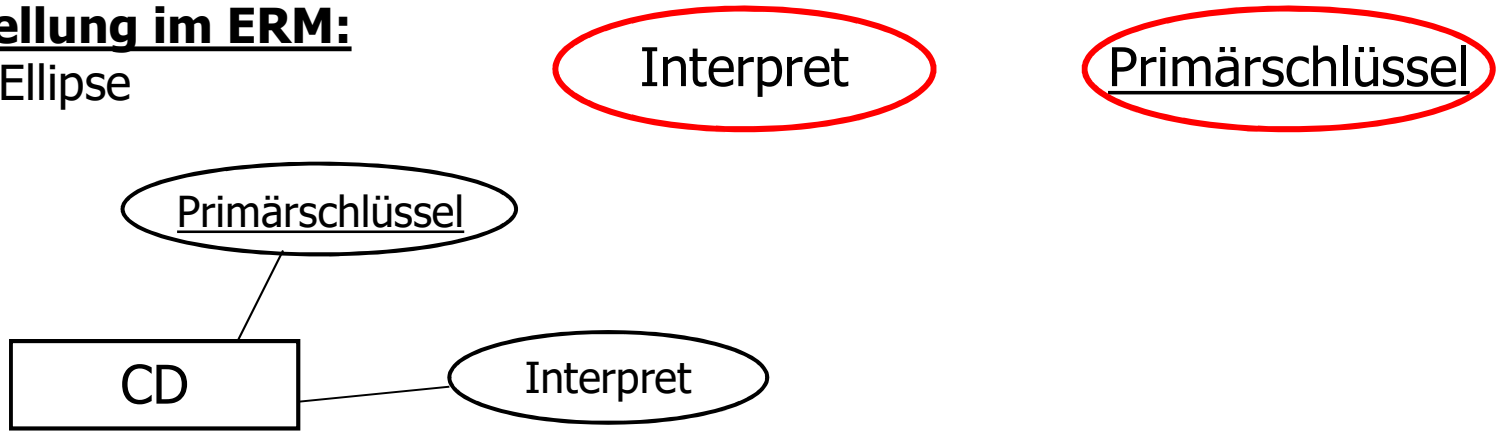
➔ **In der Datenbank werden Entitäten als Tabellen dargestellt**

Notation des Entity-Relationship-Modells – Attribute

Attribut: Objekte (Entities) werden durch Eigenschaften (Attribute) beschrieben

➔ Attribut des Objekts CD: z.B. Interpret, Preis, CD-Nummer, …
➔ Attribut des Objekts Schüler: Vorname, Familienname, S-Nummer, …

Darstellung im ERM:
➔ als Ellipse



➔ **In der Datenbank werden Attribute als Spalten dargestellt.**

Eigenschaften des Primärschlüssels

Primärschlüssel

EINDEUTIG

➔ Der Primärschlüssel dient dazu, einen Datensatz eindeutig zu identifizieren. Je zwei Datensätze einer Tabelle unterscheiden sich zumindest durch den Primärschlüssel.

MINIMAL

➔ Der Primärschlüssel besteht aus einem oder mehreren Attributen

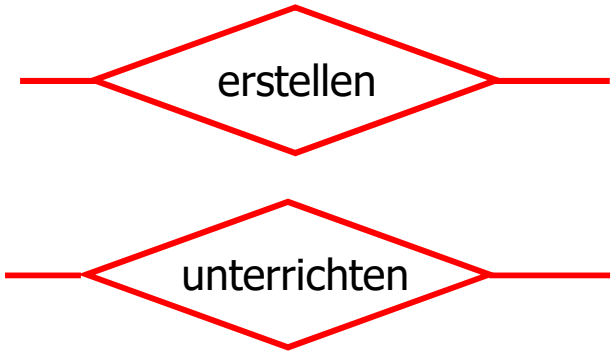
➔ Besteht der Primärschlüssel aus mehreren Attributen so muss diese Attributemenge minimal sein. D.h. durch Weglassen eines Attributes dieser Menge verliert der Primärschlüssel seine Identifikationseigenschaft.

Notation des Entity-Relationship-Modells – Beziehungen (Relationship)

Relationship: Verbindung zwischen zwei oder mehreren Entities.

➔ Beziehung zwischen CD und Interpret: erstellen
➔ Beziehung zwischen Lehrer und Schüler: unterrichten

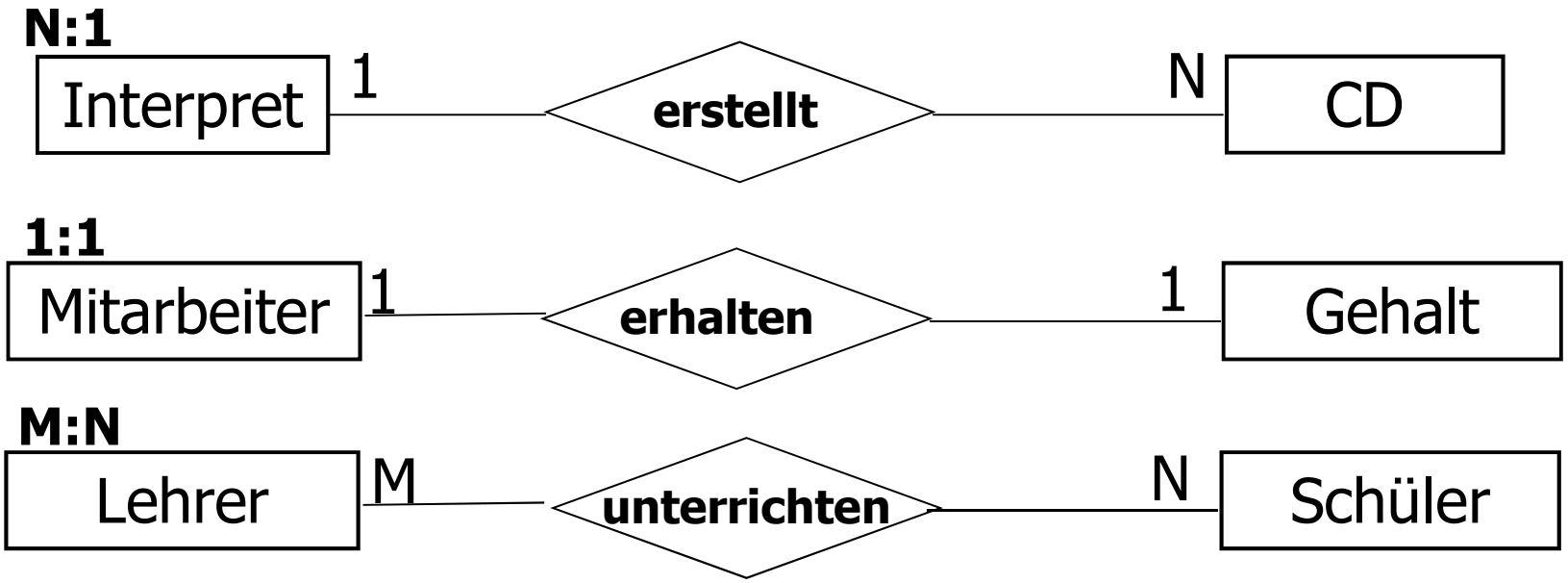
Darstellung im ERM:
➔ als Raute
➔ Mit Verb (Tätigkeitswort) beschrieben



Was im Zuge einer DB-Modellierung als Entity, als Attribut und was als Relationship dargestellt wird, kann nicht generell festgelegt werden, sondern ist von der jeweiligen Aufgabenstellung abhängig.

Mengenmäßige Beziehung zwischen Entities – Kardinalität oder Konnektivität

Die Kardinalität/Konnektivität legt fest, zwischen wie vielen Ausprägungen von Entitäten die Beziehung besteht.



Zwingende und optionale Beziehungen

Zusätzlich zur Kardinalität einer Beziehung kann festgelegt werden, ob eine Beziehung in der Realität bestehen **MUSS** oder lediglich bestehen **KANN**.

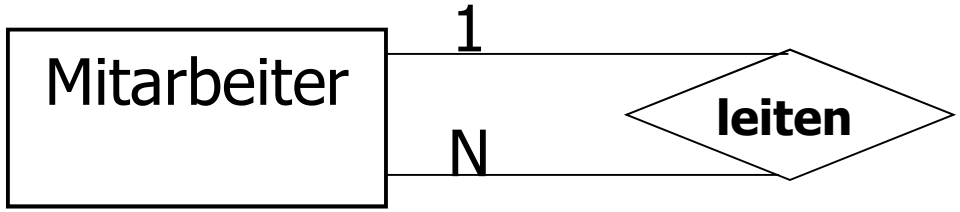
Ein Interpret kann eine CD erstellen, muss aber nicht (**optional**). Eine CD stammt aber immer von einem Interpreten (**zwingend**).

Ein Lehrer der an einer Schule arbeitet, unterrichtet immer Schüler. Ein Schüler wird in einer Schule immer von mind. einem Lehrer unterrichtet. (beides **zwingend**)

Grad einer Beziehung

Der Grad einer Beziehung gibt an, wie viele Entitäten durch eine Beziehung verbunden werden (n>=1, meist binäre Beziehungen)

Unäre Beziehung (bezieht sich nur auf eine Entität)

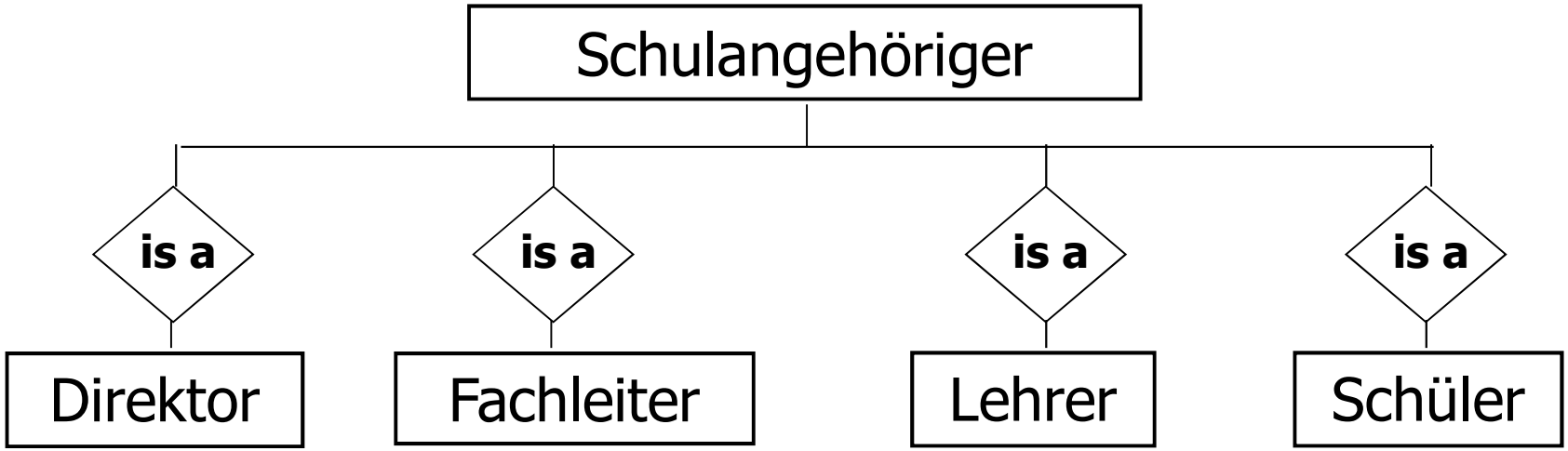


Binäre Beziehung

Mehrwertige Beziehungen (n-är) sind möglich jedoch selten

Teilmengen und Klassen

Eine Entity kann aus mehreren **Teilmengen** oder **Klassen** bestehen.



Teilmengen und Klassen erhalten automatisch alle Attribute der übergeordneten Entität.

Teilmengen: Einzelne Objekte die sich überlappen können ➔ ein Fachleiter kann auch Lehrer sein.

Klassen: Objekte die sich nicht überlappen ➔ Ein Direktor kann kein Schüler sein.

Relationales Datenmodell

Die Entitäten und Beziehungen aus dem ER-Modell werden mit Hilfe des **Relationalen Datenmodells** in **Relationen** umgewandelt.

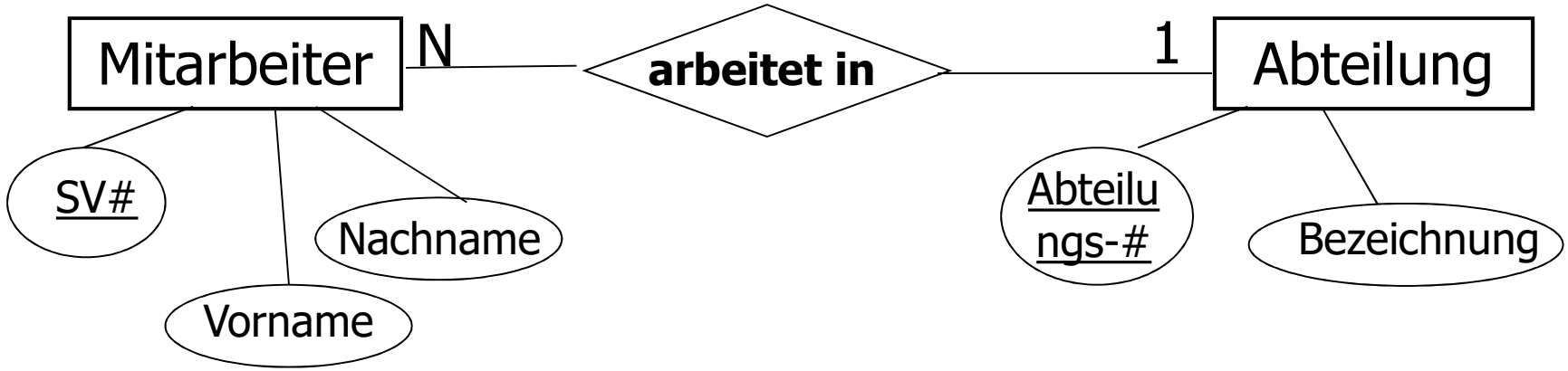
Diese Relationen stellen bereits die Tabellen der Datenbank dar

Notation:

TABELLENNAME (Primärschlüssel, Feld 1, Feld2, Feld3, …,
Sekundärschlüssel)

Tabellenname … Entity oder m:n-Beziehung aus ER-Modell
Feld … Attribut aus ER-Modell

Auflösung von 1:n-Beziehungen

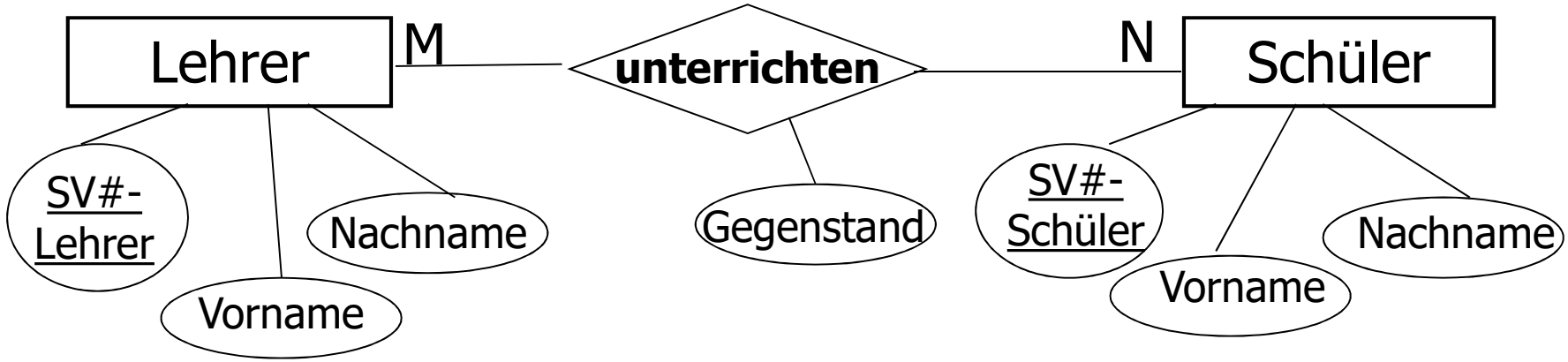


MITARBEITER (SV#, Vorname, Nachname, Abteilung)

Abteilung (Abteilungs#, Bezeichnung)

➔ Bei einer 1:n-Beziehung wird der Primärschlüssel der 1-Tabelle als **Fremd- oder Sekundärschlüssel** in der n-Tabelle gespeichert.

Auflösung von m:n-Beziehungen



LEHRER (SV#-Lehrer, Vorname, Nachname)

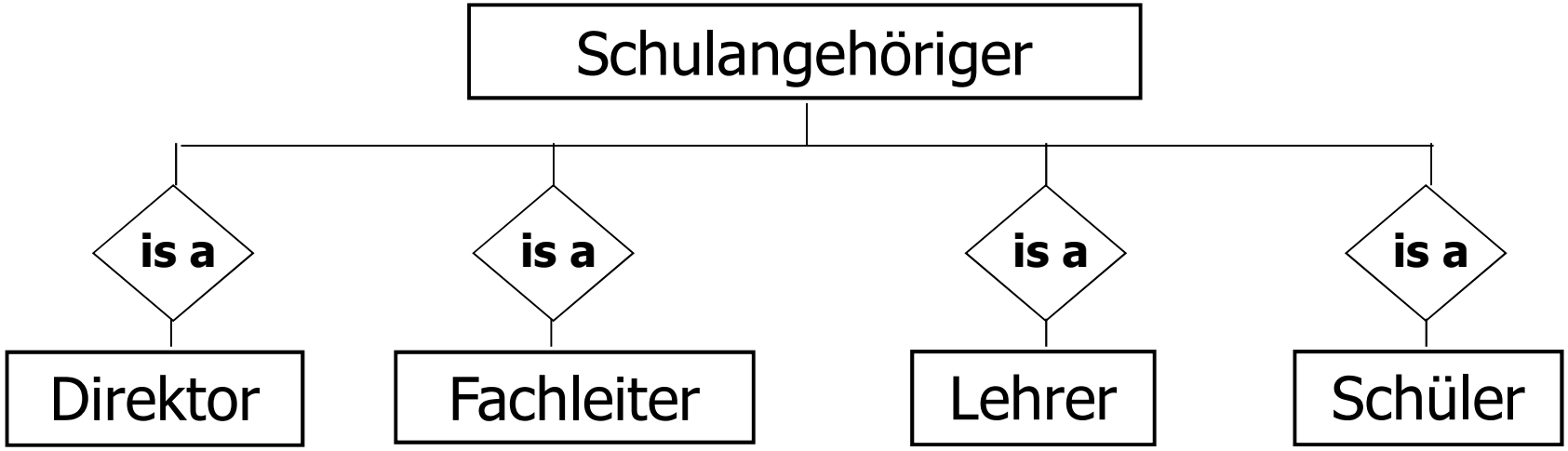
SCHÜLER (SV#-Schüler, Vorname, Nachname)

UNTERRICHT (SV#-Lehrer, SV#-Schüler, Gegenstand)

➔ Bei einer m:n-Beziehung wird die Beziehung als eigene Relation dargestellt, welche die Primärschlüssel der verbundenen Entitäten als Primärschlüssel enthält.

Darstellung von Teilmengen im relationalen Datenmodell

Teilmengen ➔ Überlappung ist möglich



SCHULANGEHÖRIGER (SV#, Vorname, Nachname, …)

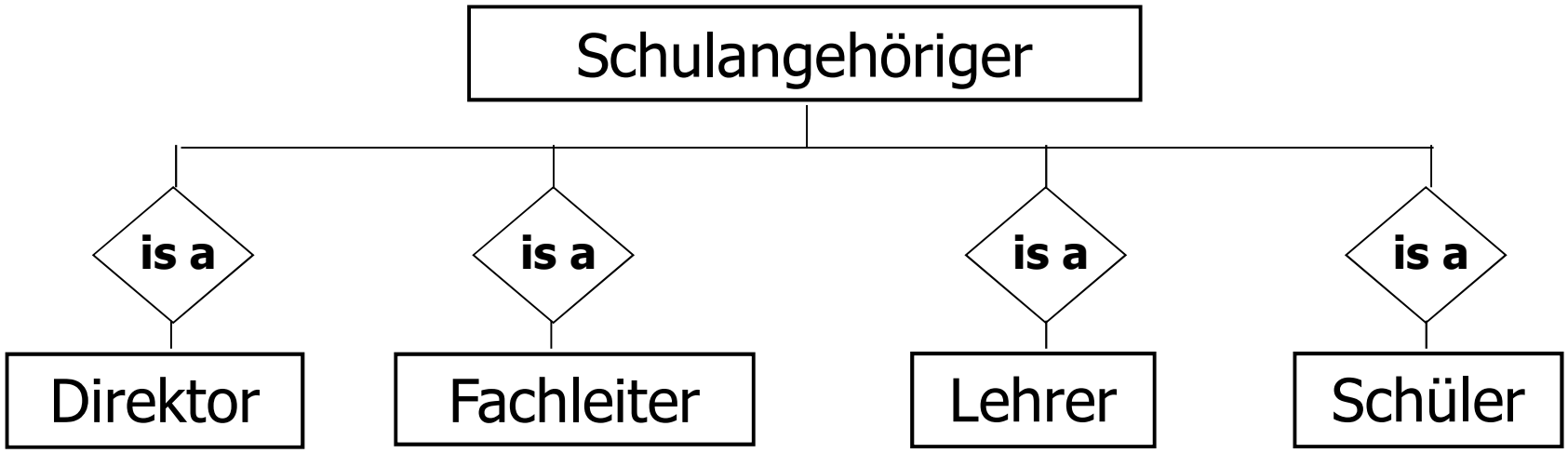
FUNKTION (Funktions#, Bezeichnung)

TÄTIGKEIT (SV#, Funktions#)

➔ Entspricht m:n-Beziehung

Darstellung von Klassen im relationalen Datenmodell

Klassen ➔Überlappung ist **nicht** möglich



SCHULANGEHÖRIGER (SV#, Vorname, Nachname,
Funktion, …)

➔ Entspricht 1:1-Beziehung

➔ **Besser 1:n-Beziehung**

Datennormalisierung I

➔ Um Mehrfachspeicherung (Redundanzen) zu vermeiden

➔ Um Schwierigkeiten im Zusammenhang mit Änderungen der Datenbank (Einfügen, Löschen von Datensätzen, Spalten und Tabellen) zu vermeiden

➔ Um einen logischen Aufbau der Datenbank und Datenunabhängigkeit zu gewährleisten

Datennormalisierung II

Die unnormalisierte Form

In der unnormalisierten Relation sind am Kreuzungspunkt von Zeile (Datensatz) und Spalte (Attribut) mehrere Elemente enthalten.

Die 1. Normalform

Eine Relation befindet sich in 1NF, wenn am Kreuzungspunkt von Zeile und Spalte genau ein Wert steht (ATOMAR).

Datennormalisierung III

Die 2. Normalform

Eine Relation befindet sich in 2NF, wenn sie in 1NF ist und jedes nicht dem Schlüssel angehörende Attribut voll funktional vom Schlüssel abhängig ist.

Die 3. Normalform

Eine Relation befindet sich in 3NF, wenn sie in 2NF ist und es keine funktionale Abhängigkeiten zwischen Nicht-Schlüsselattributen gibt.

Zusammenfassende Regeln des Relationalen Datenmodells

- ➔ Am Kreuzungspunkt einer Zeile und Spalte steht 1 Wert.
- ➔ Die in einem Feld gespeicherten Informationen dürfen nicht weiter zerlegt werden können. Ein Feld erhält die kleinstmögliche Information.
- ➔ In einer Tabelle sollten alle Felder vom Primärschlüssel abhängig sein und nicht von einem versteckten Schlüssel.
- ➔ Bei einer 1:N Beziehung wird der Primärschlüssel der 1-Tabelle als Sekundärschlüssel in der N-Tabelle gespeichert.
- ➔ Bei einer M:N-Beziehung erfolgt die Verknüpfung der beiden Tabellen durch eine dritte Tabelle, in welcher die Primärschlüssel der beiden verbundenen Tabellen gespeichert sind.