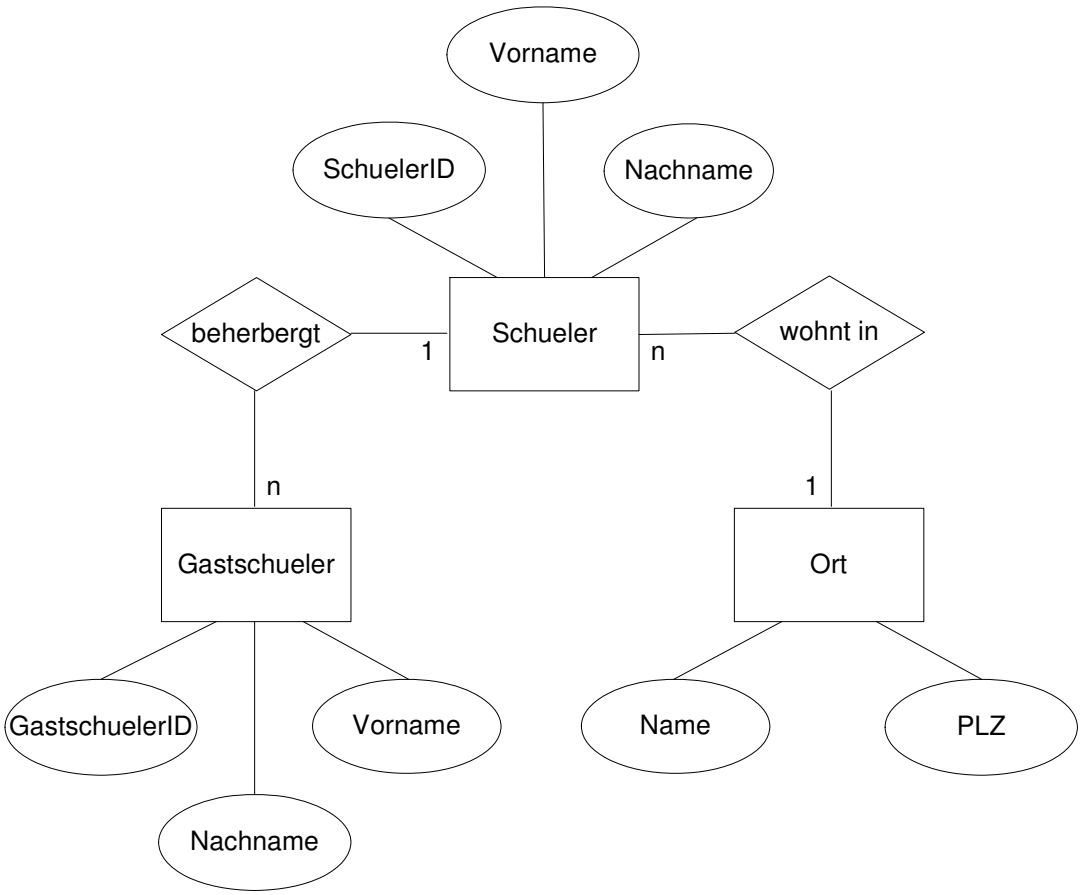
5. Datenbanken

Entity-Relationship-Modell

Eine Entität ist ein gedanklich abgegrenzter Gegenstand. Gleichartige Entitäten fasst man zu Entitätsmengen zusammen. Diese erhalten Rechtecke als Symbole.
Zwischen Entitäten kann es Beziehungen geben, wobei man gleichartige Beziehungen zwischen Entitäten zweier Entitätsmengen zu Beziehungsmengen zusammen fasst. Diese erhalten Rauten als Symbole. Eine Raute wird durch Linien mit den Rechtecken der zugehörigen Entitätsmengen verbunden. Diese Linien werden mit der Kardinalität (1:1, 1:n, n:1, n:m) beschriftet.
Attribute sind Eigenschaften aller Elemente einer Entitäts- bzw. Beziehungsmenge. Sie erhalten Ellipsen als Symbole und diese werden durch eine Linie mit dem Rechteck bzw. der Raute der entsprechenden Entitäts- bzw. Beziehungsmenge verbunden.

Beispiel:
In einer Datenbank sollen Informationen über Schüler sowie deren Gastschüler verwaltet werden.



Ein Schüler hat nur einen Wohnort, in einem Ort können aber mehrere Schüler wohnen, also Kardinalität n : 1.
Ein Schüler kann mehrere Gastschüler beherbergen, ein Gastschüler wohnt aber bei nur einem Schüler, also Kardinalität 1 : n.

Datenbankschema und Schlüssel
In einem Datenbankschema werden zeilenweise die Namen der Tabellen einer Datenbank aufgelistet. Dem Tabellennamen folgen jeweils in Klammern die Namen der einzelnen Spalten der entsprechenden Tabelle.
Durch einen Primärschlüssel ist eine Entität eindeutig festgelegt. Ein Fremdschlüssel verweist auf eine andere Entität. Beide Schlüssel können aus einem oder mehreren Attributen bestehen.
Im Datenbankschema werden Primärschlüssel unterstrichen und Sekundärschlüssel mit einem Pfeil versehen.

Beispiel:
Für obiges Beispiel sieht dies dann folgendermaßen aus:
`Schueler (SchuelerID, Vorname, Nachname, ↑PLZ)`
`Gastschueler (GastschuelerID, ↑SchuelerID, Vorname, Nachname)`
`Ort (PLZ, Name)`

Normalisierung
Ein Datenbankschema ist in der **1. Normalform**, wenn alle Attribute einen atomaren Wertebereich haben.

Ein Datenbankschema ist in der **2. Normalform**, wenn es in der 1. Normalform ist und zusätzlich jedes Attribut, das nicht selbst zum Schlüssel gehört, nur von allen Schlüsselattributen funktional abhängig ist und nicht bereits von einem Teil der Schlüsselattribute.

Ein Datenbankschema ist in der **3. Normalform**, wenn es in der 2. Normalform ist und es zusätzlich kein Nichtschlüsselattribut gibt, das transitiv von einem Schlüsselattribut abhängig ist. Es darf also keine funktionalen Abhängigkeiten von Attributen geben, die selbst nicht zum Schlüssel gehören.

Beispiel:
Wir gehen von folgendem Datenbankschema aus:

```
Schuelerkontakt (SchuelerID, VorNachname, PLZOrt, GastschuelerID,
                                              GastschuelerVorNachname)
```

SchuelerID	*VorNachname*	*PLZOrt*	*GastschuelerID*	*Gastschueler-VorNachname*
1	Peter Meier	12321 Infostadt	8	Marc Pierre
1	Peter Meier	12321 Infostadt	13	Bob Thomson
2	Eva Müller	11111 Infodorf	10	Chantal Paris
3	Maria Dorf	12321 Infostadt	NULL	NULL

Das Datenbankschema ist nicht in erster Normalform, da die Attribute `VorNachname`, `PLZOrt` und `GastschuelerVorNachname` nicht atomar sind. Eine Überführung in die erste Normalform führt zu folgendem Datenbankschema:

```
Schuelerkontakt (SchuelerID, Vorname, Nachname, PLZ, Ort,
        GastschuelerID, GastschuelerVorname, GastschuelerNachname)
```

SchuelerID	*Vorname*	*Nachname*	*PLZ*	*Ort*	*GastschuelerID*	*GastschuelerVorname*	*GastschuelerNachname*
1	Peter	Meier	12321	Infostadt	8	Marc	Pierre
1	Peter	Meier	12321	Infostadt	13	Bob	Thomson
2	Eva	Müller	11111	Infodorf	10	Chantal	Paris
3	Maria	Dorf	12321	Infostadt	NULL	NULL	NULL

Der Primärschlüssel ist aus `SchuelerID` und `GastschuelerID` zusammengesetzt. Dieses Datenbankschema ist nicht in zweiter Normalform, da z.B. `GastschuelerVorname` und `GastschuelerNachname` nur von `GastschuelerID` und nicht vom gesamten Schlüssel abhängen. Eine Überführung in die zweite Normalform führt zu folgendem Datenbankschema:

`Schueler (SchuelerID, Vorname, Nachname, PLZ, Ort)`
`Gastschueler (GastschuelerID, ↑SchuelerID, Vorname, Nachname)`

SchuelerID	*Vorname*	*Nachname*	*PLZ*	*Ort*
1	Peter	Meier	12321	Infostadt
2	Eva	Müller	11111	Infodorf
3	Maria	Dorf	12321	Infostadt

GastschuelerID	*SchuelerID*	*Vorname*	*Nachname*
8	1	Marc	Pierre
13	1	Bob	Thomson
10	2	Chantal	Paris

Dieses Datenbankschema ist nicht in dritter Normalform, da `Ort` abhängig ist von `PLZ` und somit transitiv abhängig ist von `SchuelerID`. Eine Überführung in die dritte Normalform führt zu folgendem Datenbankschema:

`Schueler (SchuelerID, Vorname, Nachname, ↑PLZ)`
`Gastschueler (GastschuelerID, ↑SchuelerID, Vorname, Nachname)`
`Ort (PLZ, Name)`

SchuelerID	*Vorname*	*Nachname*	*PLZ*
1	Peter	Meier	12321
2	Eva	Müller	11111
3	Maria	Dorf	12321

GastschuelerID	*SchuelerID*	*Vorname*	*Nachname*
8	1	Marc	Pierre
13	1	Bob	Thomson
10	2	Chantal	Paris

PLZ	*Name*
12321	Infostadt
11111	Infodorf

Sprachelemente

Folgende SQL-Sprachelemente werden vorausgesetzt:

```
SELECT (DISTINCT) … FROM
WHERE
GROUP BY
ORDER BY
ASC, DESC
(LEFT / RIGHT) JOIN … ON
UNION
AS
NULL
```

Vergleichsoperatoren: `=, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL`

Arithmetische Operatoren: `+, -, *, /, (…)`

Logische Verknüpfungen: `AND, OR, NOT`

Funktionen: `COUNT, SUM, MAX, MIN`

Es werden SQL-Abfragen über eine und mehrere verknüpfte Tabellen vorausgesetzt.
Es können auch verschachtelte SQL-Ausdrücke vorkommen.

Relationenalgebra

Es werden Selektion, Projektion, Vereinigung, Differenz, kartesisches Produkt, Umbenennung und Join vorausgesetzt.

Beispiel:
Wir gehen von folgendem Datenbankschema aus:
Lehrer (ID, Vorname, Nachname)
Lehrerin (ID, Vorname, Nachname)
Schulleitung (ID, Vorname, Nachname)
Klassenleitungsteam (↑LehrerID, ↑LehrerinID)

Lehrer:

ID	*Vorname*	*Nachname*
Me	Peter	Meier
Sz	Peter	Schulz
Bm	Hans	Baum

Lehrerin:

ID	*Vorname*	*Nachname*
Be	Petra	Blume
Sr	Clara	Sommer
Kr	Helga	Kremer

Klassenleitungsteam:

LehrerID	*LehrerinID*
Bm	Be
Me	Kr
Me	Sr

Schulleitung:

ID	*Vorname*	*Nachname*
Bm	Hans	Baum
Kr	Helga	Kremer

Selektion:
Es werden die Zeilen ausgewählt, die eine bestimmte Bedingung erfüllen.

```
SELECT * FROM Lehrer WHERE Vorname = “Peter“
```

ID	*Vorname*	*Nachname*
Me	Peter	Meier
Sz	Peter	Schulz

Projektion:
Es werden nur bestimmte Spalten ausgewählt. Doppelte Zeilen werden entfernt.

```
SELECT DISTINCT Vorname FROM Lehrer
```

Vorname
Peter
Hans

Vereinigung:
Zwei Tabellen mit gleichen Attributen werden zu einer vereinigt. Doppelte Zeilen werden entfernt.

```
SELECT * FROM Lehrer UNION SELECT * FROM Lehrerin
```

ID	*Vorname*	*Nachname*
Me	Peter	Meier
Sz	Peter	Schulz
Bm	Hans	Baum
Be	Petra	Blume
Sr	Clara	Sommer
Kr	Helga	Kremer

Differenz:
Es werden die Zeilen einer Tabelle ausgewählt, die in einer zweiten Tabelle nicht enthalten sind.

```
SELECT * FROM Lehrer WHERE Lehrer.ID
     NOT IN (SELECT ID FROM Schulleitung)
```

ID	*Vorname*	*Nachname*
Me	Peter	Meier
Sz	Peter	Schulz

Kartesisches Produkt:
Es werden alle Zeilen einer Tabelle mit allen Zeilen einer zweiten Tabelle verknüpft.

```
SELECT * FROM Lehrer, Lehrerin
```

ID	*Vorname*	*Nachname*	*ID*	*Vorname*	*Nachname*
Me	Peter	Meier	Be	Petra	Blume
Sz	Peter	Schulz	Be	Petra	Blume
Bm	Hans	Baum	Be	Petra	Blume
Me	Peter	Meier	Sr	Clara	Sommer
Sz	Peter	Schulz	Sr	Clara	Sommer
Bm	Hans	Baum	Sr	Clara	Sommer
Me	Peter	Meier	Kr	Helga	Kremer
Sz	Peter	Schulz	Kr	Helga	Kremer
Bm	Hans	Baum	Kr	Helga	Kremer

Umbenennung:
Ein Attribut wird umbenannt.

```
SELECT ID AS Kuerzel, Vorname, Nachname FROM Lehrerin
```

Kuerzel	*Vorname*	*Nachname*
Be	Petra	Blume
Sr	Clara	Sommer
Kr	Helga	Kremer

Join:
Ein Join ist die Bildung eines karthesischen Produktes gefolgt von einer Selektion.

```
SELECT * FROM Lehrer JOIN Klassenleitungsteam
     ON Lehrer.ID = Klassenleitungsteam.LehrerID
```

ID	*Vorname*	*Nachname*	*LehrerID*	*LehrerinID*
Bm	Hans	Baum	Bm	Be
Me	Peter	Meier	Me	Kr
Me	Peter	Meier	Me	Sr

Beim Left-Join werden auch die Zeilen aus der ersten Tabelle aufgeführt, die keinen Partner in der zweiten Tabelle haben. Es wird mit „NULL“ aufgefüllt.

```
SELECT * FROM Lehrer LEFT JOIN Klassenleitungsteam
     ON Lehrer.ID = Klassenleitungsteam.LehrerID
```

ID	*Vorname*	*Nachname*	*LehrerID*	*LehrerinID*
Me	Peter	Meier	Me	Kr
Me	Peter	Meier	Me	Sr
Sz	Peter	Schulz	NULL	NULL
Bm	Hans	Baum	Bm	Be

Beim Right-Join werden auch die Zeilen aus der zweiten Tabelle aufgeführt, die keinen Partner in der ersten Tabelle haben. Es wird mit „NULL“ aufgefüllt.

```
SELECT * FROM Klassenleitungsteam RIGHT JOIN Lehrer
     ON Lehrer.ID = Klassenleitungsteam.LehrerID
```

LehrerID	LehrerinID	ID	Vorname	Nachname
Me	Kr	Me	Peter	Meier
Me	Sr	Me	Peter	Meier
NULL	NULL	Sz	Peter	Schulz
Bm	Be	Bm	Hans	Baum

SQL-Abfrage über mehrere verknüpfte Tabellen:

```
SELECT Lehrer.Nachname AS Klassenlehrer,
     Lehrerin.Nachname AS Klassenlehrerin
     FROM Lehrer, Lehrerin, Klassenleitungsteam
     WHERE Lehrer.ID = Klassenleitungsteam.LehrerID
     AND Klassenleitungsteam.LehrerinID = Lehrerin.ID
```

Klassenlehrer	*Klassenlehrerin*
Baum	Blume
Meier	Kremer
Meier	Sommer