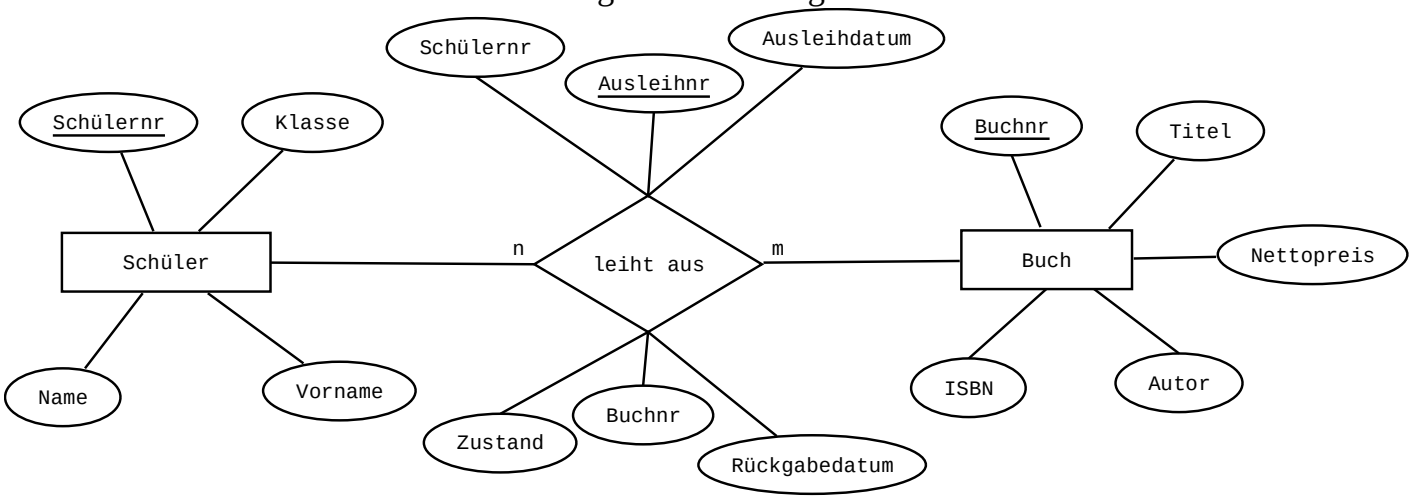
Relationale Datenbanken

Für eine Schülerbibliothek wurde folgendes ER-Diagramm modelliert:



Für eine reale Anwendung sollte man noch weitere Attribute hinzufügen.

Es gibt zwei spezielle Arten von Attribute, auf die bisher nicht eingegangen wurde:

- **Primärschlüssel (Primary Key):** ist ein Attribut, das eine Entität **eindeutig** identifiziert. Primärschlüssel dürfen nicht leer sein. Sie müssen, wie gesagt, eindeutig sein und sollten so wenige Attribute wie möglich enthalten, in der Regel: ein Attribut. Es ist empfehlenswert für den Primärschlüssel ein künstliches Attribut zu verwenden, was in der Modellierung geschehen ist.
- **Fremdschlüssel (Foreign Key):** Wird ein Primärschlüssel einer Entitymenge in einer anderen verwendet, spricht man von einem Fremdschlüssel.

Über die Schlüsselattribute werden die Beziehungen zwischen den Entitymengen hergestellt.

In der Modellierung kommen Fremdschlüssel nur bei der Relationship „leiht aus“ vor, und zwar die „Schülernr“ und die „Buchnr“,

Übertragen eines ER-Modell in ein relationales Modell

Nun sind wir mit der Modellierung fertig und können das Modell in ein Datenbanksystem, üblicherweise eine relationale Datenbank, übertragen. Dabei wird es mit den Begriffen problematisch. Der Begriff relationale Datenbank kommt nicht von Relationship, sondern von Relation. Eine Relation kommt aus der Mathematik und ist in diesem Fall nichts anderes als eine Tabelle.

Folgende Begriffe sind beim **relationalen Modell** (Datenbankschema) wichtig:

- Eine **Tabelle** (Relation) ist eine zweidimensionale Anordnung von Elementen gleicher Struktur.
- Ein **Feld** (Attribut) ist eine Spalte einer Tabelle (Relation). Ein Feld kann verschiedene Werte annehmen und hat einen Datentyp (wie eine Variable).
- **Datensatz** (Tupel): Zeile einer Tabelle (Relation)
- Primärschlüssel und Fremdschlüssel: siehe oben

Zum Übertragen verwendet man folgende Regeln:

1. Jede Entitymenge des ER-Modells wird ohne Änderung in eine Tabelle überführt.
2. Zwei Entitymengen E_1 und E_2 mit der Kardinalität 1:1 werden zu einer einzigen Entitymenge E zusammengefasst und damit in eine einzige Tabelle übertragen. Jede Entity aus E ist durch einen Primärschlüssel eindeutig identifiziert.
   Diesen Schritt muss man vorsichtig angehen und manchmal Entitymengen nicht zusammenführen, damit das DBMS schneller arbeiten kann. (Beispiel: Produkte und ihre Produktdetails)
3. Wenn zwei Entitymengen E_1 und E_2 mit der Kardinalität 1:n in Beziehung stehen, wird der Primärschlüssel der Entitymenge E_1 (also das „1“ der Kardinalität) als Fremdschlüssel in E_2 hinzugefügt.
4. Wenn zwei Entitymengen E_1 und E_2 mit der Kardinalität n:m in Beziehung stehen, wird eine neue Tabelle (Relation) modelliert, die den Primärschlüssel aus E_1 und den Primärschlüssel aus E_2 als Fremdschlüssel enthält.

Regel 1 führt in unserem Beispiel zu den Tabellen „Schüler“ und „Bücher“. Regel 2 und 3 sind hier nicht anwendbar. Regel 4 führt zur Tabelle „Ausleihe“.

Nun können wir die Tabellen in einem DBMS, z.B. OpenOffice /LibreOffice Base oder Access anlegen. Die Attribute werden zu den Feldern. Man muss sich nur jeweils überlegen, was der passende Datentyp ist.

Hier die Tabellen für OO/LO-Base:

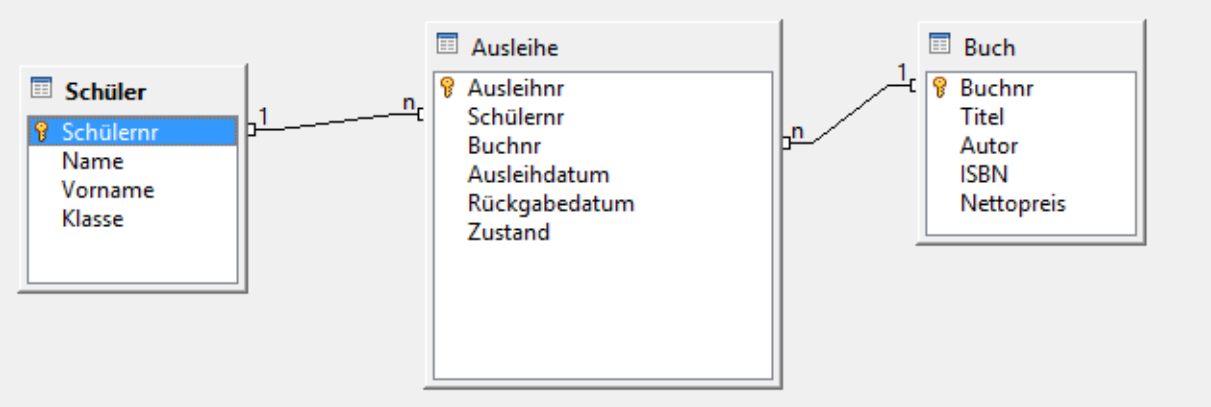
Tabelle Schüler	
Schülernr	Integer (Integer, Auto-Wert)
Name	Text(Varchar[30])
Vorname	Text(Varchar[30])
Klasse	Text(Varchar[3])

Tabelle Buch	
Buchnr	Integer (Integer, Auto-Wert)
Titel	Text(Varchar[60])
Autor	Text(Varchar[50])
ISBN	Text(Varchar[13])
Nettopreis	Dezimal(DECIMAL)

Tabelle Ausleihe	
Ausleihnr	Integer (Integer, Auto-Wert)
Schülernr, Buchnummer	Integer(Integer)
Ausleihdatum, Rückgabedatum	Datum(Date)
Zustand	Text(Varchar[40])

Aus Platzgründen sind in der Tabelle Ausleihe die Felder Schülernr und Buchnr sowie Ausleihdatum und Rückgabedatum in einer Zeile eingetragen. Sie müssten eigentlich jeweils eine eigene Zeile erhalten.
Zustand sollte eigentlich eine Aufzählung sein, mit Zuständen wie „Buch beschädigt“, „Verlängert“, „Buch verloren“ usw.

Die Tabellen stehen in folgender Beziehung miteinander:



Nun kann man beginnen die Tabellen mit Daten zu füllen. Wie man Information aus einer Daten abfragt, wird auf einem weiteren Skript-Blatt „SQL-Abfragen“ erläutert.

Theorie: Normalisieren des Datenbankmodells

Bevor man das relationale Modell wirklich in ein DBMS überträgt, sollte man es noch normalisieren, damit das DBMS besser damit umgehen kann. Es gibt in der Theorie fünf Normalformen, wobei in der Praxis meist nur drei umgesetzt werden.

Erste Normalform:

Eine Tabelle (Relation) ist in 1. Normalform, wenn alle Attribute atomar sind, d. h. nicht strukturiert sind. Beispielsweise sollte der vollständige Name in Name und Vorname vorliegen oder die Adresse als PLZ, Ort, Straße.

Zweite Normalform:

Eine Tabelle (Relation) ist in 2. Normalform, wenn die 1. Normalform erfüllt ist und wenn alle Attribute, die nicht zum Primärschlüssel gehören, von diesem „voll funktional abhängig“ sind.

Beispiel: In einer Zeltlager-Datenbank gibt es eine Tabelle Zelt mit ZeltNr (Primärschlüssel), Anzahl, Organisation. Das Attribut „Organisation“ ist unabhängig von einem bestimmten Zelt, deshalb muss es in eine eigene Tabelle überführt werdet.

Dritte Normalform:

Eine Tabelle (Relation) ist in 3. Normalform, wenn die 2. Normalform erfüllt ist und kein Nichtschlüsselattribut „transitiv“ von einem Schlüsselattribut abhängt.

Beispiel: In einer Kundendatenbank wird Ort und PLZ in der Kundentabelle gespeichert. Jede PLZ erfordert einem bestimmten Ort („transitiv abhängig“). Dadurch entsteht eine Redundanz. Wenn die PLZ geändert werden, müsste jeder Kundendatensatz angefasst werden. Lösung ist eine weitere Tabelle (Relation) mit PLZ und Ort.

Unser Beispiel zur Schülerbibliothek erfüllt schon die 3. Normalform.