MySQL
Normalisierung

Stefan Maihack Dipl. Ing. (FH)
Datum: 22.04.2018

MySQL – Normalisierung

Allgemeines

- Durch die Normalisierung von Tabellen soll folgendes erreicht werden

  ➔ Redundanzfreie, beziehungsweise redundanzarme Speicherung der Daten.
  ➔ Inkonsistenzen; Vermeidung von Datenbank-Anomalien.
  ➔ Logische Strukturierung der Daten.
  ➔ Vereinfachte Speicherung der Daten.

- Datenbankdesign und Normalisierung sind eng miteinander verzahnte Prozesse.

- Große Tabellen die nicht normalisiert sind, kosten Systemleistung. Die Aufteilung, in sehr viele kleinere Tabellen (extreme Normalisierung) kompliziert die Abfragen, was ebenso die Systemleistung reduziert.

  ➔ Man muss einen Mittelweg finden. Z.B. Tabellen die selten angesprochen werden, nicht weiter normalisieren.

MySQL – Normalisierung

Normalisierungsregeln: 0NF

- Die Nullte Normalform ist dann gegeben, wenn alle Informationen in einer Tabelle vorhanden sind und unnormalisiert vorliegen.
- Die Nullte Normalform liegt in vielen Fällen während der Anforderungsanalyse einer Datenbank vor.
- Die Anforderungsanalyse in der Datenbankentwicklung beginnt mit der Sammlung von unstrukturierten Daten und unsortierten Informationen aus den verschiedenen Fachbereichen oder Informationsquellen.
- Beispiel: Was fällt auf?

R.Nr	Datum	Name	Straße	Ort	Artikel	Anzahl	Preis
187	01.01.2012	Max Mustermann	Musterstraße 1	12345 Musterort	Bleistift	5	1,00 €

MySQL – Normalisierung

Normalisierungsregeln: 1NF

- 1. NF:
  Die 1. NF ist notwendig, ein relationales Datenmodell zu haben. Sie fordert, dass alle Attribute einer Entity in atomarer Form vorliegen. Ein atomarer Wert, wie beispielsweise eine Zahl, eine Zeichenkette oder ein Datum, zusammenhängende Informationen.

  Z.B. in der Tabelle „Personal“ steht in einer Spalte „PLZ“ und der „Ort“. Diese müssen in separaten Spalten stehen.

  ➔ Abhilfe: Alle Informationen müssen in einzelnen Spalten stehen.

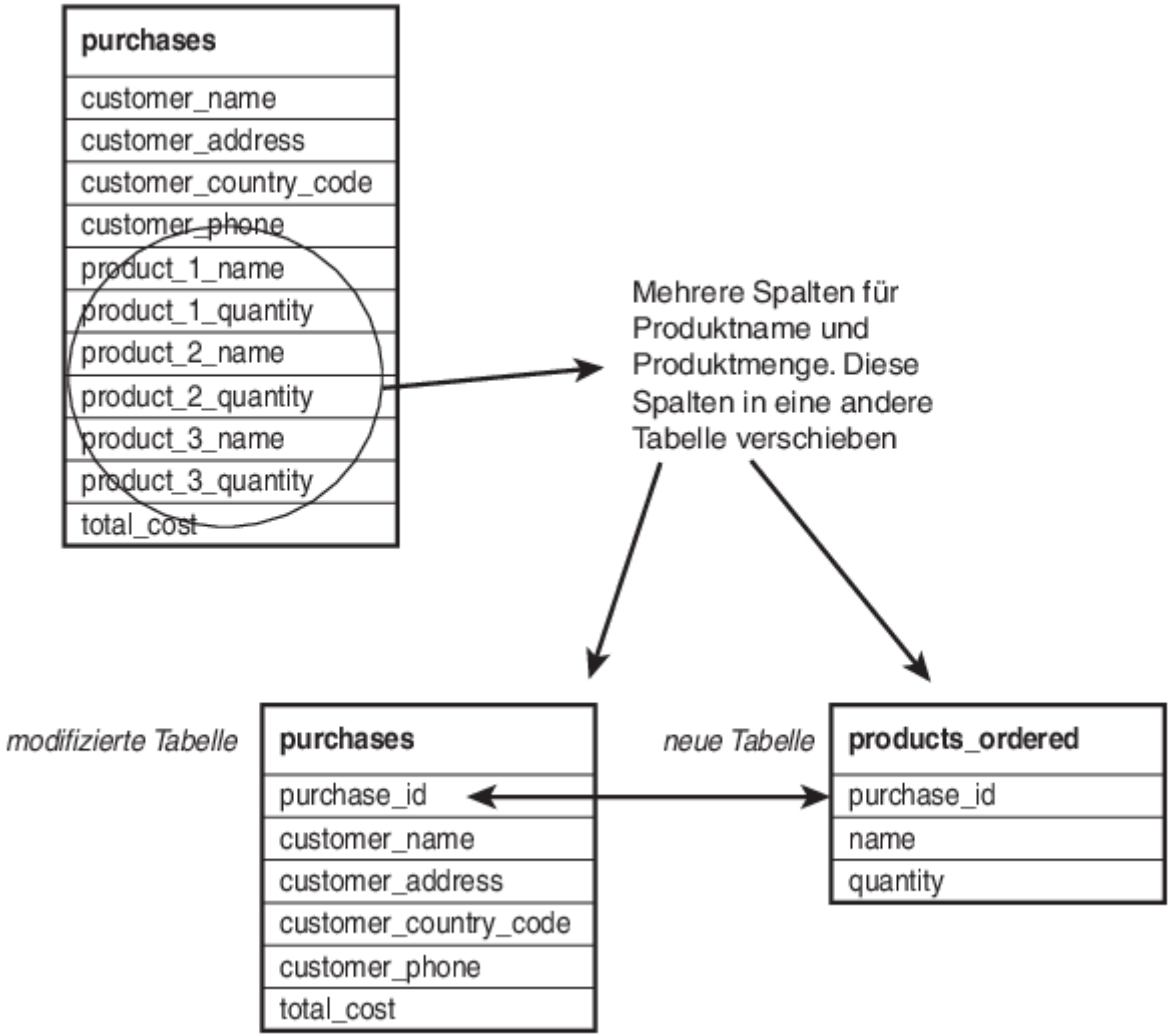
- Beispiel:

R.Nr	Datum	Name	Straße	Ort	Artikel	Anzahl	Preis
187	01.01.2012	Max Mustermann	Musterstraße 1	12345 Musterort	Bleistift	5	1,00 €

R.Nr	Datum	Name	Nachname	Straße	Ort	PLZ	Artikel	Anzahl	Preis
187	01.01.2012	Max	Mustermann	Musterstraße 1	12345	12345	Bleistift	5	1,00 €

MySQL – Normalisierung

Normalisierungsregeln: 1NF



MySQL – Normalisierung

Normalisierungsregeln: 2NF

- 2. NF:
  Relationen müssen sich in der 1. NF befinden. Kein Attribut darf existieren, das nicht zum Schlüssel gehört und nur von einem Teil des Schlüssels voll funktional abhängig ist.

  Oder:

  Jeder Datensatz bildet nur einen Sachverhalt ab. Liegen in einer Tabelle Daten vor, die nicht nur ein Sachverhalt abbilden, werden diese Daten in einzelne thematische Tabellen unterteilt.

  Daraus folgt:

  ➔ Besitzt eine Relation in erster Normalform nur einen einstelligen Schlüssel (Schlüssel besteht nur aus einem Attribut), dann befindet sie sich automatisch auch in der 2. NF.

  ➔ Ein Nichtschlüssel-Attribut steht in einer 1:1 Beziehung zu einem Teil des Schlüssels.

- Die 2. NF hat die Aufgabe, Teilabhängigkeiten zu eliminieren, indem sie aus diesen eigene Relationen erstellt.

MySQL – Normalisierung

Normalisierungsregeln: 2NF

- Beispiel zur 2. NF:

PersNr	**ProjNr**	ProjName	AuftragVom	BearbBeginn
1001	50	HTML-Seiten	10.01.2004	01.02.2004
1005	50	HTML-Seiten	10.01.2004	01.02.2004
1002	51	Datenbank	15.02.2004	01.03.2004
1005	51	Datenbank	15.02.2004	15.03.2004

Primärschlüssel: PersNr und ProjNr

➔ ProjName und AuftragVom sind charakteristische Projekteigenschaften, die unabhängig vom Mitarbeiter sind, der das Projekt bearbeitet. Mit anderen Worten, sie sind funktional abhängig von der Projektnummer (ProjNr) – einem Teil des Schlüssels.

➔ Überführung in die 2. NF - Umwandeln in zwei Relationen (Tabellen).

MySQL – Normalisierung

Normalisierungsregeln: 2NF

- In der 2. Normalform befindliche Tabellen:

  Tabelle „Projektbearbeitung“ in der 2.NF

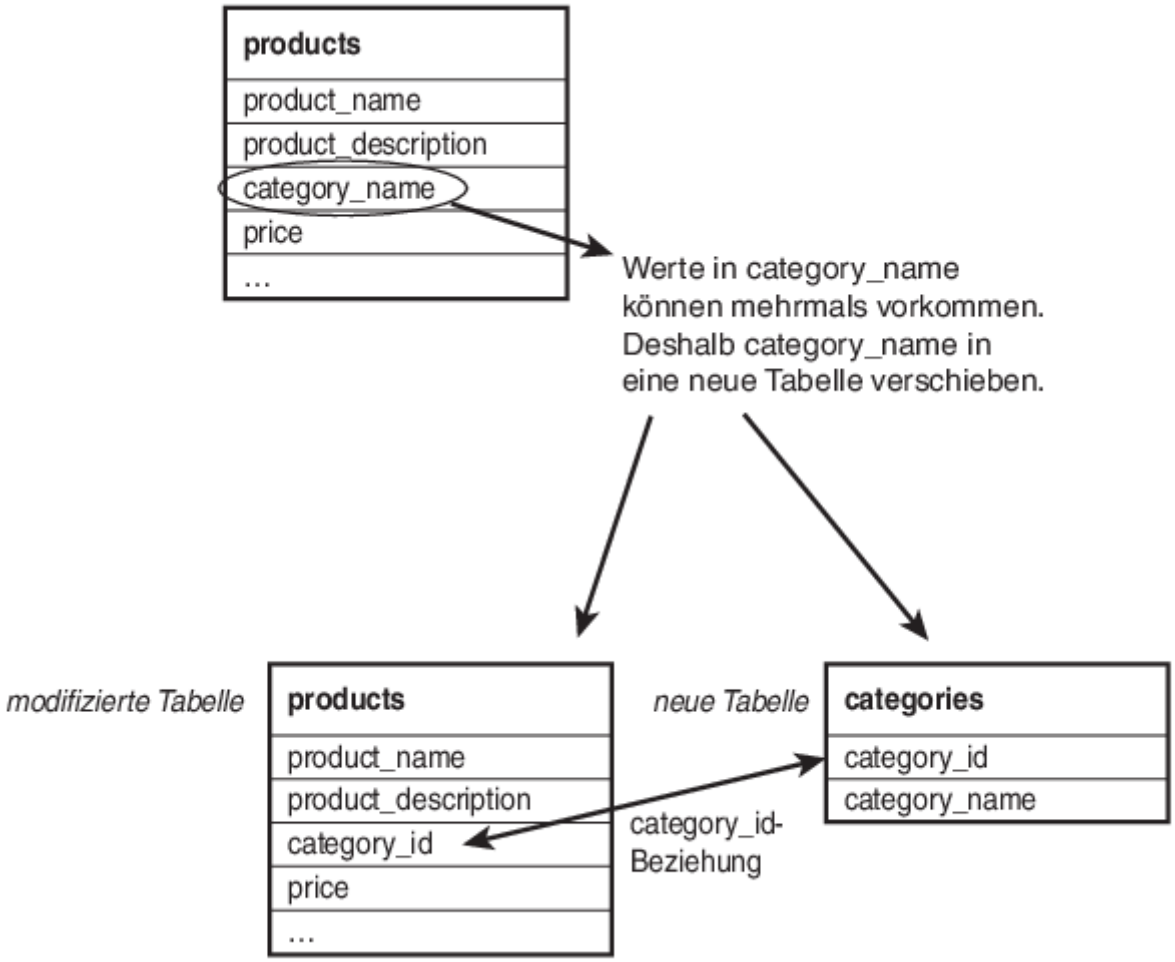
PersNr	ProjNr	BearbBeginn
1001	50	01.02.2004
1005	50	26.01.2004
1002	51	01.03.2004
1005	51	15.03.2004

  Tabelle „Projekte“ in der 2.NF

ProjNr	Projektname	AuftragVom
50	HTML-Seiten	10.01.2004
51	Datenbank	15.02.2004

MySQL – Normalisierung

Normalisierungsregeln: 2NF



MySQL – Normalisierung

Normalisierungsregeln: 2NF

- Ein weiteres Beispiel:

1. NF

R.Nr	Datum	Name	Nachname	Straße	Ort	PLZ	Artikel	Anzahl	Preis
187	01.01.2012	Max	Mustermann	Musterstraße 1	12345	12345	Bleistift	5	1,00 €

2. NF

Rechnung

R.-Nr.	Datum	Knr.
187	01.01.2012	007

Kunde

Knr.	Name	Vorname	Straße	Hnr.	PLZ	Ort
007	Mustermann	Max	Musterstr.	1	12345	Musterort

Rechnungsposition

R.-P.-Nr.	R.-Nr.	Art.-Nr.	Anzahl
1	187	69	5

Artikel

Art.-Nr.	Artikel	Preis
69	Bleistift	1,00

MySQL – Normalisierung

Normalisierungsregeln: 3NF

- Die 3. NF fordert, dass sich eine Relation in der 2. NF befindet und alle Attribute, die nicht zum Schlüssel gehören, nur vom Schlüssel abhängig sind.
  ➔ Mit anderen Worten, es darf keine Abhängigkeiten zwischen Attributen geben.
  ➔ Alle Attribute sind direkt vom Schlüssel abhängig.

- Beispiel: In der Tabelle Bestellungen werden alle Bestellungen eines Buchversenders gespeichert. Diese Relation besitzt die Attribute „BestellNr“, „Datum“, „ISBN“ und „Titel“. Das Attribut „BestellNr“ ist der Primärschlüssel.

„Titel“ ist abhängig von „ISBN“

BestellNr	Datum	ISBN ⟵	Titel
101	10.04.2004	3827291054	Mogel-Power 2004
102	10.04.2004	3827257174	Jetzt lerne ich HTML
103	11.04.2004	3827257174	Jetzt lerne ich HTML
104	11.04.2004	3827291062	Black&White

  Alle Attribute sind zwar charakteristisch für eine Bestellung, aber nicht direkt, denn das Attribut „Titel“ ist abhängig von ISBN.

- Die transitiv abhängigen Spalten werden in eine weitere Untertabelle ausgelagert, da sie nicht direkt vom Schlüsselkandidaten abhängen, sondern nur indirekt.

MySQL – Normalisierung

Normalisierungsregeln: 3NF

- Lösung: Aus „ISBN“ und „Titel“ wird eine weitere Relation erstellt. In der Tabelle „Bestellungen“ bleibt „ISBN“ als Fremdschlüssel .

Tabelle: „Bestellungen“

BestellNr	Datum	ISBN
101	10.04.2004	3827291054
102	10.04.2004	3827257174
103	11.04.2004	3827257174
104	11.04.2004	3827291062

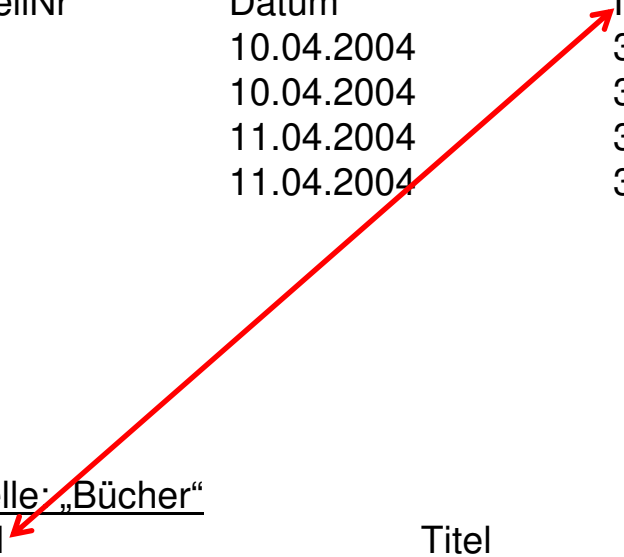
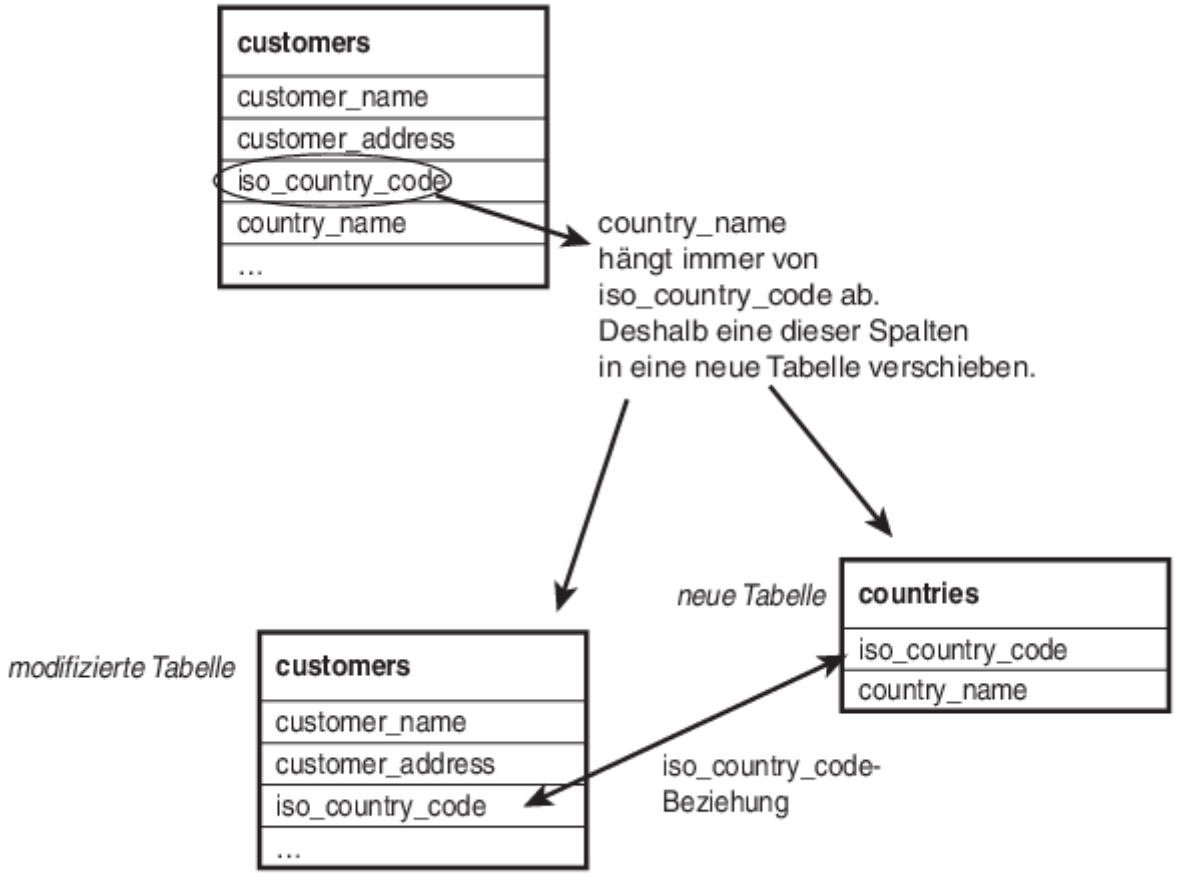


Tabelle: „Bücher“

ISBN	Titel
3827291054	Mogel-Power 2004
3827257174	Jetzt lerne ich HTML
3827291062	Black&White

MySQL – Normalisierung

Normalisierungsregeln: 3NF



MySQL – Normalisierung

Normalisierungsregeln: 3NF

- Die dritte Normalform ist oft ausreichend, um die perfekte Balance aus Redundanzen, Performance und Flexibilität für eine Datenbank zu gewährleisten.

Die **Kundeninformationen** liegen nun in der zweiten Normalform (2NF) vor:

Kunde						
Knr.	Name	Vorname	Straße	Hnr.	PLZ	Ort
007	Mustermann	Max	Musterstr.	1	12345	Musterort

Nach der Anwendung der **Dritten Normalform (3NF)** sieht das Ergebnis folgendermaßen aus:

Kunde					
Knr.	Name	Vorname	Straße	Hnr.	PLZ
007	Mustermann	Max	Musterstr.	1	12345

Postleitzahl	
PLZ	Ort
12345	Musterort

MySQL – Normalisierung

Zusammenfassung

- Fazit der Normalisierung:

  ➔ Redundanzfreie, beziehungsweise redundanzarme Speicherung der Daten

  ➔ Vermeidung von Datenbank-Anomalien

  ➔ Logische Strukturierung der Daten

  ➔ Vereinfachte Speicherung der Daten

- Datenbankdesign und Normalisierung sind eng mit einander verzahnte Prozesse.

MySQL – Normalisierung

In die Praxis gebracht

- 4 Regeln sind zu beachten:

  1. Eliminierung sich wiederholender Gruppen. Für jede Gruppe von Daten, die sich wiederholen, legt man eine eigene Tabelle an.
  Z.B.:

ID	Vorname	Nachname	BestellNr	ArtikelNr	Preis	Artikel

  ➔ Daraus entstehen 3 Tabellen: „Artikel“, „Bestellungen“ und „Adressen“. Jede Tabelle bekommt einen eindeutigen Schlüssel.

  2. Eliminierung redundanter Daten. Wenn eine Eigenschaft mehrere Werte annehmen kann, bringt man diese in eine eigene Tabelle.
  Z.B.: Wenn „Artikel“ mehrere Preise haben kann, bringt man diese in eine eigene Tabelle.

MySQL – Normalisierung

In die Praxis gebracht

3. Eliminieren von Spalten, die von keinem Schlüssel abhängen. Wenn Eigenschaften keinen Zusammenhang mit dem Schlüsselfeld haben, also nicht ebenso eindeutig zugeordnet werden können, überträgt man diese Eigenschaft in eine eigene Tabelle.

   Z.B. Wenn man 2 Adressen pro Kunde hat (Lieferanschrift und Rechnungsanschrift), können die Adressen nicht mehr eindeutig einem Schlüsselfeld Kunden-ID zugeordnet werden.
   ➔ Anschriften der Kunden in weitere Tabellen aufteilen.

4. Isolieren von unabhängigen Beziehungen. Keine Tabelle darf 2 oder mehrere Beziehungen haben, die nicht direkt abhängig sind. Wie viele Tabellen würden sie daraus machen?

ID	Vorname	Nachname	BestellNr	ArtikelNr	Preis	Artikel

➔ Mindestens 3 Tabellen daraus machen.

MySQL – Normalisierung

In die Praxis gebracht

- Große Tabellen die nicht normalisiert sind, kosten Systemleistung. Die Aufteilung, in sehr viel kleinere Tabellen (extreme Normalisierung) kompliziert aber auch die Abfragen, was ebenso die Systemleistung reduziert.

  Versuchen Sie einen Mittelweg zu finden. Z.B. Tabellen die selten angesprochen werden, nicht weiter zu normalisieren.

MySQL – Normalisierung
Übungen

Wann sollte eine Datenbank normalisiert werden?

- Am besten unmittelbar sofort, nachdem heraus gearbeitet wurde, welche Tabellen man benötigt, um die Geschäftsanforderungen zu modellieren.

- Man kann zwar die Datenbank auch später noch normalisieren, nachdem die Anwendungsentwicklung eingesetzt hat und sogar noch nachdem das System funktioniert. Allerdings ist der Aufwand westlich höher. Vor allem, wenn Tabellen bereits mit Daten gefüllt sind.

- Es kann sogar sein, das die Tabellen denormalisiert werden müssen, um das System von der Geschwindigkeit her, zu optimieren.

MySQL – Normalisierung
Übungen

- Eine Sammlung von Musik-CD´s soll durch eine Tabelle „cds“ dargestellt werden. Die Tabelle soll folgende Informationen enthalten: Künstlernamen, CD-Titel, Tracks auf der CD. Stellen Sie sich die aktuelle Tabelle aus den Spalten „artist“, „title“, „name_track1“, „name_track2“ bis „name_track20“ vor. Bilden Sie die erste, zweite und dritte Normalform.

- NF1: Die Spalten „name_track1“, „name_track2“ bis „name_track20“ in eine andere Tabelle verschieben.

- NF2: Die Spalte „artist“ in eine eigene Tabelle verschieben. Da ein Künstler in der Regel mehrere Alben aufgenommen hat. Er würde somit mehrfach in der Tabelle auftauchen.

- NF3: Die Tabellen befinden sich bereits in der 3. Normalform.

Übung 1 – MySQL

Bestimmen sie den Primärschlüssel/ zusammengesetzt oder einzeln

- A) Relation "Kunden"
  wohnort, vorname, nachname, land_id, kunde_id
  (Beispieldatensatz: 'Stuttgart', 'Herbert', 'Maier', 7, 13)

  B) Relation "Lagerbestand"
  artikel_name, regal_nummer, anzahl, artikel_nummer, einkaufspreis
  (Beispieldatensatz: 'Bürste', 287, 3, 1928, 6.85)

  C) Relation "Ausleihvorgaenge"-
  dvd_id, kunde_id, filiale_id, ausleihdatum, rueckgabedatum
  (Beispieldatensatz: 53, 1664, 2, 2010-12-01 11:23:12, 2010-12-03 11:23:12)

  D) Relation "Fahrraeder"
  hersteller, farbe, gangschaltung, fahrrad_id, bremsen
  (Beispieldatensatz: 17, grün, 8, 13, 56)

  E) Relation "Bestellungen"-
  rechnungssumme, kunde_id, bestelldatum
  (Beispieldatensatz: 89.38, 13, 2010-12-01 11:29:12)

  F) Relation "Bücher"-
  buchtitel, erscheinungsjahr, autor_name, preis, seitenzahl
  (Beispieldatensatz: 'Kochen für Anfänger', 1981, 'Herbert Smith', 13.95, 198)

Übung 2 - MySQL

- Aufgabe:
  Es soll eine Datenbank (siehe Studienarbeit) erstellt werden mit der folgende Funktionen verwaltet werden sollen:

  1. Speicherung von Flurstücken (Stadt übergreifend)

  2. Speicherung von Bewohnern (Stadt übergreifend)

  3. Speicherung von Straßen (Stadt übergreifend)

- Erstellen sie das dazugehörige ER-Diagramm und normalisieren sie diese.

Übung 2 - MySQL

- Folgende Attribute soll die Datenbank besitzen:

  ➔ **Vorname** Bewohner
  ➔ **Nachname** Bewohner
  ➔ **Strassenname** Bewohner
  ➔ **PLZ** Bewohner
  ➔ **Stadtname**: Name der Stadt des Bewohners
  ➔ **Hausnummer** des Bewohners
  ➔ **Kinderanz**: Anzahl der Kinder
  ➔ **Alter** des Bewohners
  ➔ **Eigentümer**: Ist der Bewohner ein Eigentümer
  ➔ **Einwohneranzahl**
  ➔ **Flurstücknummern**
  ➔ **Flurstückgröße**: Größe der Flurstücke
  ➔ **Lage** der Flurstücke (X und Y Koordinate)
  ➔ **PLZFLUR** der Flurstücke
  ➔ **EigentümerFLUR**: Eigentümer der Flurstücke
  ➔ **Lagestrassen** der Straßen (X- und Y-Koordinate)
  ➔ **AnzahlHausnr**: Anzahl der Hausnummern der Straßen

Übung 2 – MySQL

mögliche Lösung