Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

Datenbanken

Vorlesungsskript für das 3. Semester

Studiengang Internationale Medieninformatik

7. Datenmodellierung -

Normalisierung

Dozent: M. Sc. Burak Boyaci

Version: 02.12.2025

Wintersemester 25/26

11 Normalisierung

Kapitelübersicht

Wir haben Datenmodelle bisher vorwiegend auf der Ebene von Tabellen behandelt, angefangen von der Modellierung der Entitätstypen und ihrer Beziehungen untereinander bis zur Ableitung des konkreten Tabellenmodells daraus nach bestimmten Ableitungsregeln. Die Normalisierung nun ist ein standardisiertes Verfahren, mit dem sich auf *Ebene der Attribute* der Tabellen Redundanzen (d.h. die mehrfache Speicherung gleicher Daten) und Anomalien vermeiden lassen. Anomalien sind fatal für Datenbanken, da sie beim Einfügen, Ändern oder Löschen von Datensätzen zu inkonsistenten oder lückenhaften Dateninhalten führen. Redundanzen auf der anderen Seite können zu Anomalien führen, verhindern aber auf jeden Fall eine effiziente Datenspeicherung, was insbesondere bei großen Datenbanken zu ernsten Problemen führen kann.

Bei der Normalisierung unterscheidet man in der Literatur bis zu sechs sogenannten Normalformen, wovon in der Praxis jedoch nur vier berücksichtigt werden. Diese vier Normalformen werden wir in diesem Kapitel behandeln.

11.1 Anomalien

Eine *Anomalie* bezeichnet in der Informatik ein Fehlverhalten der Datenverwaltung, die zu Inkonsistenzen der Daten führen. Es gibt mehrere Arten von Anomalien. Im Allgemeinen für ein Mehrbenutzersystem wichtig sind Anomalien durch mangelhafte Synchronisation der Daten. Da sie vorwiegend durch das Managementsystem einer Datenbank gelöst werden und vor allem bei paralleler Datenspeicherung auftreten, werden wir uns im Weiteren aber damit nicht beschäftigen.[1] Daneben gibt es nämlich noch auch drei für den Einbenutzerbetrieb wichtige Anomalien, die das Einfügen, Ändern und Löschen betreffen. Wir verwenden dazu das Datenbeispiel aus Piepmeyer (2011:S. 143ff). Die Tabelle erscheint auf den ersten Blick recht plausibel. Jeder Datensatz speichert die für ein Album relevanten Daten — also alles klar! Gehen wir nun davon

[1] Siehe dazu auch https://de.wikipedia.org/wiki/Anomalie_(Informatik)

alben

reihe	band	titel	verlag	jahr
Asterix	1	Asterix, der Gallier	Ehapa	1968
Asterix	17	Die Trabantenstadt	Ehapa	1974
Asterix	25	Der große Graben	Ehapa	1980
Tim und Struppi	1	Der geheimnisvolle Stern	Carlsen	1972
Franka	1	Das Kriminalmuseum	Epsilon	1985
Franka	2	Das Meisterwerk	Epsilon	1986

Tabelle 11.1: Beispieldaten der Tabelle alben zur Illustration der Anomalien.

aus, dass Alben einer gegebenen Reihe *immer* im gleichen Verlag erscheinen, so können wir dies nicht mit Integritätsregeln automatisch sicherstellen. Anders sieht es für Wertebereiche von Attributen aus: Wir können mit dem reservierten Wort **CHECK** bei den Attributdeklarationen die Integritätsregeln einfügen, dass die Nummer des Bandes positiv sein muss und das Erscheinungsjahr 1937 oder später sein soll,[2] d.h. wir können das folgende Tabellenschema erstellen:

```
CREATE TABLE alben(
  reihe varchar(30),
  band int CHECK (band > 0),
  title varchar(30) NOT NULL,
  verlag varchar(30) NOT NULL,
  jahr int CHECK (jahr >= 1937),
  PRIMARY KEY (reihe, band)
);
```

Mit der folgenden SQL-Anweisung können wir dann die folgenden Beispieldaten speichern:

```
INSERT INTO alben(reihe, band, titel, verlag, jahr) VALUES
  ('Asterix', 1, 'Asterix der Gallier', 'Ehapa', 1968),
  ('Asterix', 17, 'Die Trabantenstadt' , 'Ehapa',1974),
  ('Asterix', 25, 'Der große Graben', 'Ehapa', 1980), ('Tim und Struppi', 1, 'Der
  geheimnisvolle Stern', 'Carlsen', 1972), ('Franka', 1, 'Das Kriminalmuseum' ,
  'Epsilon', 1985),
  ('Franka', 2, 'Das Meisterwerk' , 'Epsilon', 1986);
```

Die nicht implementierbare Regel „jede Reihe immer im selben Verlag“ können wir bestenfalls als „Programmierrichtlinie“ festlegen, aber ein Anwender kann möglicherweise nichts davon wissen oder sich einfach vertippen. So kann es zu folgenden Anomalien kommen.

11.1.1 Die Einfügeanomalie

Eine *Einfügeanomalie* liegt vor, wenn ein Datensatz nicht in eine Datenbank eingefügt werden kann, da nicht alle Informationen zum Primärschlüssel vorliegen oder bei einer unvollständigen / falschen Eingabe zu Inkonsistenz. Für unsere Beispieldaten würde eine Anomalie etwa durch den folgenden Befehl bewirkt:

```
INSERT INTO alben(reihe, band, titel, verlag, jahr) VALUES
  ('Asterix', 2, 'Asterix und Kleopatra', 'Ehara', 1968);
```

Durch einen Tippfehler haben wir gegen die obige Regel verstoßen und die Reihe Asterix in einem neuen Verlag erscheinen lassen. Unser Datenbestand ist gemäß dieser Regel nun inkonsistent.

---

[2] Im Jahr 1937 erschien das erste Album *(comic book)* der Comicreihe *Detective Comics*; in derselben Rei he erschien im Mai 1939 übrigens erstmals Batman, vgl. https://de.wikipedia.org/wiki/Comic#Heft-–und–Buchformate.

Genauso würde folgender Befehl zu einer Einfügeanomalie führen, da die Werte für den Primärschlüssel nicht vollständig sind:

```
INSERT INTO alben (reihe, band, titel, verlag, jahr) VALUES
('Asterix', 2, 'Asterix und Kleopatra', NULL, 1968);
```

11.1.2 Die Änderungsanomalie

Exkurs UPDATE. Mit UPDATE können wir in SQL bereits bestehende Einträge in Datenbanken verändern. Dies geschieht vor allem dann, wenn sich der Wert einer Ausprägung ändert.

Syntax:

```
UPDATE tabellenname SET spaltenname = 'Eintrag_neu' WHERE spaltenname = 'Eintrag_alt';
```

Eine *Änderungsanomalie* liegt vor, wenn bei Änderung eines Attributwertes nicht alle betroffenen Datensätze geändert werden. Ähnlich wie bei der Einfügeanomalie ergäbe für unsere Beispieldaten der folgenden Änderungsbefehl eine Anomalie:

```
UPDATE alben SET verlag='Springer' WHERE reihe='Asterix' AND band=1;
```

alben

reihe	band	titel	verlag	jahr
Asterix	1	Asterix, der Gallier	Springer	1968
Asterix	17	Die Trabantenstadt	Ehapa	1974
Asterix	25	Der große Graben	Ehapa	1980
Tim und Struppi	1	Der geheimnisvolle Stern	Carlsen	1972
Franka	1	Das Kriminalmuseum	Epsilon	1985
Franka	2	Das Meisterwerk	Epsilon	1986

Das Update-Statement würde in diesem Fall in Zeile 1 der Tabelle *alben* den Verlag von „Ehapa“ zu „Springer“ ändern, jedoch würden die Zeilen 2 und 3 davon unbeeinflusst. Auch hier liegt dann also eine Dateninkonsistenz gemäß unserer Regel vor.

11.1.3 Die Löschanomalie

Exkurs DELETE: Mit DELETE können wir einzelne Datensätze aus unserer Datenbank löschen. Dies geschieht beispielsweise dann, wenn wir Informationen zu Kunden nicht mehr benötigen, weil der Kunde gekündigt hat.

Syntax:

```
DELETE FROM tabellenname WHERE spaltenname = 'Eintrag_neu';
```

Eine *Löschanomalie* liegt vor, wenn durch das Löschen eines Datensatzes mehr Informationen als erforderlich verloren gehen. Löschen wir beispielsweise aus unserem Beispieldaten das Album *Tim und Struppi*:

```
DELETE FROM alben WHERE reihe= 'Tim und Struppi' AND band=1;
```

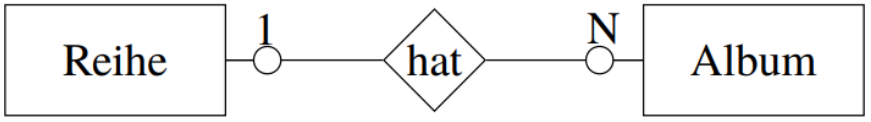
alben

reihe	band	titel	verlag	jahr
Asterix	1	Asterix, der Gallier	Springer	1968
Asterix	17	Die Trabantenstadt	Ehapa	1974
Asterix	25	Der große Graben	Ehapa	1980
Franka	1	Das Kriminalmuseum	Epsilon	1985
Franka	2	Das Meisterwerk	Epsilon	1986

Eigentlich wollten wir doch nur ein einzelnes Album aus unserer Datenbank löschen, z.B. weil wir es aktuell nicht mehr haben. Verloren haben wir jedoch zusätzlich jegliche Information darüber, in welchem Verlag diese Reihe erscheint, obwohl das gar nicht unser Ziel. Die Löschanomalie hat also eine etwas andere Qualität als die anderen beiden Anomalien: Zwar führt sie nicht zu Dateninkonsistenzen gemäß unserer Regel, aber vernichtet mehr Information als gewollt.

11.1.4 Auflösung der Anomalien durch Normalisierung

Anomalien sind für Datenbanken ein ernstes Problem. Sie führen zu Dateninkonsistenzen und zu überflüssigen Informationsverlusten. Nicht alle lassen sich durch Integritätsregeln in SQL verhindern. Müssen wir uns also damit abfinden? Betrachten wir dazu die wesentliche Ursache von Anomalien: Wir haben in einem Datensatz *zu viel* Information gespeichert. Stattdessen wäre es vorteilhaft, die Information auf mehrere Datensätze zu verteilen. So muss ein Album nur „wissen“, zu welcher Reihe es gehört: gemäß unserer Regel „jede Reihe immer im selben Verlag“ muss die Information über den Verlag in einem eigenen Datensatz gespeichert werden. Wir benötigen also *zwei* Tabellen:



Das ergibt die Tabellenschemata

```
CREATE TABLE reihen(
   reihe varchar(30),
   verlag varchar(30) NOT NULL,
   PRIMARY KEY (reihe)
);
CREATE TABLE alben(
   reihe varchar(30),
   band int CHECK (band > 0),
   title varchar(30) NOT NULL,
   jahr int CHECK (jahr >= 1937),
   PRIMARY KEY (reihe, band),
   FOREIGN KEY (reihe) REFERENCES reihen(reihe)
);
```

und die Datenspeicherungen

```
INSERT INTO reihen(reihe, verlag) VALUES
  ('Asterix', 'Ehapa'), ('Tim und Struppi',
  'Carlsen'), ('Franka', 'Epsilon');

INSERT INTO alben(reihe, band, titel, jahr) VALUES
  ('Asterix', 1, 'Asterix, der Gallier', 1968),
  ('Asterix', 17, 'Die Trabantenstadt' , 1974),
  ('Asterix', 25, 'Der große Graben', 1980),
  ('Tim und Struppi' , 1, 'Der geheimnisvolle Stern', 1972),
  ('Franka', 1, 'Das Kriminalmuseum' , 1985),
  ('Franka', 2, 'Das Meisterwerk' , 1986);
```

Damit können die obigen Anomalien nicht mehr passieren! Wir haben also mit einem konsequenten Redesign unseres Tabellenmodells die Informationen so verteilt, dass sie unseren expliziten Regeln entsprechen. Was aber ist hier eigentlich geschehen? In einem tieferen Sinn haben wir aus unserem ursprünglichen Tabellenentwurf Redundanzen entfernt. Um dies systematisch durchzuführen, wird in der Informatik das Verfahren der *Normalisierung* eines Datenmodells durchgeführt. Es liefert mit einer schrittweisen Redundanzsbefreiung in Form von sogenannten Normalformen eine Art Qualitätsverbesserung eines Datenbankentwurfs. Dazu betrachtet es die Beziehungen der Attribute einer Tabelle untereinander. Wie dies geschieht, werden wir im Folgenden detailliert betrachten.

11.2 Die ersten vier Normalformen

Die Normalisierung mit den Normalformen ist ein schrittweiser Prozess, in dem im Grunde Qualitätskriterien an einen einmal entworfenes Datenmodell angewendet werden, um Datenredundanz und Anomalien zu verhindern. Um die Normalformen zu erläutern, verwenden wir das folgende Anwendungsbeispiel. Nehmen wir an, in einem Projekt zur Erstellung einer Datenbank für die Auftragsabwicklung seien durch die Datenmodellierung die folgenden Tabellen ermittelt worden:

Tabelle	Primärschlüssel	Attribute
Kunden	Kundennr	Firma, Ort, AuftragsNr
Aufträge	AuftragsNr	Auftragsdatum, Lieferdatum
Artikel	ArtikelNr	Bezeichnung, LagerNr, Lagerort
Positionen	ArtikelNr, AuftragsNr	Menge, Nettopreis
Rechnungen	RechnungsNr	Datum, AuftragsNr, Nettopreis, Umsatzsteursatz, Bruttopreis

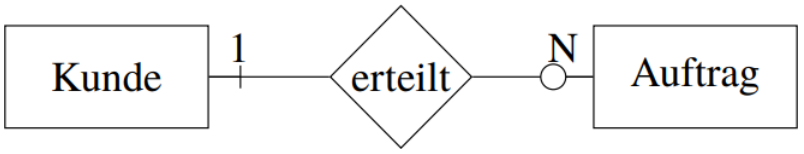
(11.1)

11.2.1 Erste Normalform

Eine Tabelle ist in der *ersten Normalform* (1NF), wenn ihre Attribute nur einfache Werte besitzen und nicht wieder Tabellen sind: Es gibt keine Tabellen in einer Tabelle. Beispielsweise widerspricht die folgende Tabelle *kunden* der 1NF:

kunden(KundenNr, Firma, Ort, AuftragsNr)

Denn falls der Kunde mehrere Aufträge erteilt, enthält die Spalte Auftragsnummer eine Liste von mehreren Werten, was einer Tabelle entspricht. Lösung: Da es sich hier um eine C-CM-Beziehung handelt,



ist sie falsch aufgelöst. Stattdessen muss das Attribut Auftragsnummer aus der Tabelle Kunde gestrichen werden und die Tabelle Aufträge die Kundennummer als Fremdschlüssel erhalten. Mit anderen Worten: Halten wir uns an die Implementierungsregeln aus Abschnitt 10.1, und insbesondere an Regel 3 (bei C-CM-Beziehung wird der Primärschlüssel der 1-Seite zugeordnet, Fremdschlüssel der N-Seite) und Regel 4 (bei 1-CM-Beziehung wird der Primärschlüssel der 1-Seite zugeordnet, Fremdschlüssel der N-Seite), so ist die erste Normalform automatisch erfüllt.

11.2.2 Zweite Normalform

Eine Tabelle ist in der *zweiten Normalform*, wenn die folgenden Bedingungen erfüllt sind:

1. Sie befindet sich in der ersten Normalform.
2. Attribute, die nicht zum Primärschlüssel gehören, sind vollständig abhängig von den Primärschlüsselattributen.

(„Eine Tabelle ist eine logische Einheit von Attributen.“) Die folgende Tabelle Positionen ist zwar in der ersten Normalform, aber nicht in der zweiten:

Positionen(ArtikelNr, AuftragsNr, Menge, Preis),

denn der Preis ist nicht abhängig von der AuftragsNr. Lösung: Das Attribut Preis muss in eine (neue) Tabelle Artikel ausgelagert werden.

11.2.3 Dritte Normalform

Eine Tabelle ist in der *dritten Normalform*, wenn die folgenden Bedingungen erfüllt sind.

1. Sie befindet sich in der zweiten Normalform.
2. Es gibt keine indirekten (transitiven) Abhängigkeiten zwischen zwei Tabellenattributen.

(„Nichtschlüsselattribute sind unabhängig voneinander.“) Die folgende Tabelle ist in der zweiten Normalform, aber nicht in der dritten:

Artikel (ArtikelNr, Preis, LagerNr, Lagerort),

denn der Lagerort hängt von der Lagernummer und daher transitiv von der Artikelnummer ab: ArtikelNr ⇒ LagerNr und LagerNr ⇒ LagerOrt, also ArtikelNr ⇒ LagerOrt.

Lösung: Das Attribut Lagerort muss in eine Tabelle *Lager* mit Primärschlüssel *LagerNr* ausgelagert und dieses Attributes in *Artikel* als Fremdschlüssel aufgenommen werden:

Artikel(ArtikelNr, LagerNr, Preis)

11.2.4 Vierte Normalform

Eine Tabelle ist in der *vierten Normalform*, wenn sie in 3NF ist und keine aus der Datenbank ableitbaren Attribute enthält. („Berechnete Kennzahlen streichen!“) Die folgende Tabelle ist in 3NF, aber nicht in 4NF:

Rechnung(RechnungsNr, Datum, AuftragsNr, Nettopreis, Umsatzsteuersatz, Bruttopreis),

denn der Bruttopreis ergibt sich aus dem Nettopreis und dem Umsatzsteuersatz. Lösung: Das Attribut Bruttopreis muss gestrichen werden.

11.2.5 Beispiel: Tabellen vor und nach der Normalisierung

Zusammenfassend bewirkt die Normalisierung des ursprünglichen Ausgangsmodells (11.1) unserer Datenmodellierung die folgenden Veränderungen.

Tabelle	Primärschlüssel	Attribute
Kunden	Kundennr	Firma, AuftragsNr, Ort
Aufträge	AuftragsNr	Auftragsdatum, Lieferdatum
Artikel	ArtikelNr	Bezeichnung, LagerNr, Lagerort
Positionen	ArtikelNr, AuftragsNr	Menge, Nettopreis
Rechnungen	RechnungsNr	Datum, AuftragsNr, Nettopreis, Umsatzsteuersatz, Bruttopreis

Tabelle	Primärschlüssel	Attribute
Kunden	Kundennr	Firma, Ort
Aufträge	AuftragsNr	KundenNr, Auftragsdatum, Lieferdatum
Artikel	ArtikelNr	Bezeichnung, LagerNr, Nettopreis
Positionen	ArtikelNr, AuftragsNr	Menge
Rechnungen	RechnungsNr	Datum, AuftragsNr, Umsatzsteuersatz
Lager	LagerNr	Ort

11.3 Anwendungsfall: Die Comic-Alben

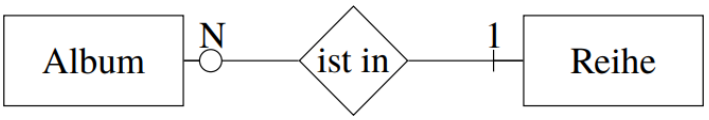
Beispiel 11.1. [3] Wider besseres Wissen haben wir in Beispiel 3.4 (Seite 12 Datenbanken_2_SQL_ Grundlagen) unsere Comicalben in einer einzigen Tabelle abgelegt.

Tabelle	Primärschlüssel	Attribute
Album	Reihe, Band	Titel, Preis, Verlag, Jahr

Sie widerspricht aber insbesondere der zweiten Normalform, denn die Reihe eines Albums ist nicht abhängig von seinem Titel, muss also in eine eigene Tabelle ausgelagert werden:

Tabelle	Primärschlüssel	Attribute
Reihe	Reihe	Verlag
Album	Reihe, Band	Titel, Preis, Jahr

Nach der Normalisierung sieht das ER-Modell also wie folgt aus:



Es ist also eine 1-CM-Beziehung zwischen zwei Entitäten. Da jetzt alle Normalformen erfüllt sind, wie man schnell nachprüft, ist das Datenmodell damit normalisiert. Die Datenbank können wir es daher wie folgt implementieren:

[3] nach Piepmeyer (2011):S. 207ff.

```
CREATE TABLE reihen (
   reihe varchar(30),
   verlag varchar(30) NOT NULL ,
   PRIMARY KEY (reihe)
);
```

```
CREATE TABLE alben (
   reihe   varchar(30),
   band int CHECK (band > 0),
   titel     varchar(30) NOT NULL,
   jahr      int CHECK (jahr >= 1937),
   PRIMARY KEY (reihe, band),
   FOREIGN KEY (reihe) REFERENCES reihen(reihe)
);
```

Angereichert mit den Daten:

```
INSERT INTO reihen (reihe, verlag)
VALUES
  ('Asterix', 'Ehapa'), ('Tim und
  Struppi' , 'Carlsen'),
  ('Franka', 'Epsilon'),
  ('Lucky Luke' , 'Egmont'),
  ('Gespenster Geschichten' ,
  'Bastei'), ('Prinz Eisenherz' ,
  'Carlsen');
```

```
INSERT INTO alben (titel, reihe, band, preis, jahr) VALUES
  ('Asterix, der Gallier' , 'Asterix', 1, 2.80, 1968),
  ('Asterix und Kleopatra' , 'Asterix', 2, 2.80, 1968),
  ('Asterix als Legionär' , 'Asterix', 10, 3.00,NULL),
  ('Die Trabantenstadt' , 'Asterix', 17, 3.80, 1974),
  ('Zarter Schmelz' , 'Lucky  Luke' , 1, 16.00, 2021),
  ('Der große Graben' , 'Asterix', 25, 5.00,1980),
  ('Das Kriminalmuseum' , 'Franka', 1, 8.80,1985),
  ('Das Meisterwerk' , 'Franka', 2, 8.80, 1986), ('Der
  geheimnisvolle Stern' , 'Tim und Struppi' , 1, NULL,
  1972), ('Tim  und der Haifischsee', 'Tim und
  Struppi' , 23, NULL, 1973);
```

Ergibt das die Datenbestände:

reihen

reihe	verlag
Asterix	Ehapa
Tim und Struppi	Carlsen
Lucky Luke	Egmont
Franka	Epsilon
Prinz Eisenherz	Carlsen

alben

titel	reihe	band	preis	jahr
Asterix, der Gallier	Asterix	1	2.80	1968
Asterix und Kleopatra	Asterix	2	2.80	1968
Asterix als Legionär	Asterix	10	3.00	NULL
Die Trabantenstadt	Asterix	17	3.80	1974
Zarter Schmelz	Lucky Luke	1	5.00	2021
Der große Graben	Asterix	25	5.00	1980
Das Kriminalmuseum	Franka	1	8.80	1985
Das Meisterwerk	Franka	2	8.80	1986
Der geheimnisvolle Stern	Tim und Struppi	1	NULL	1972
Tim und der Haifischsee	Tim und Struppi	23	NULL	1973

11.4 Zusammenfassung

Die Normalisierung ist ein standardisiertes Verfahren, um Redundanzen und Anomalien durch ein falsches Tabellenschema zu vermeiden. Es wird nach der Entity-Relationship-Modellierung und der Ableitung des Tabellenmodells durchgeführt. Während diese beiden Modellierungsschritte die wesentlichen Tabellen des Datenmodells liefern, werden bei der Normalisierung die Attribute der einzelnen Tabellen betrachtet.
Schrittweise werden die sogenannten Normalformen 1NF bis 4NF untersucht. 1NF überprüft im Wesentlichen, ob die Regeln 3 und 4 bei der Tabellenableitung aus dem ER-Diagramm eingehalten wurden. 2NF betrachtet das Verhältnis der Attribute zum Primärschlüssel und 3NF das Verhältnis von Nicht-Schlüsselattributen untereinander. 4NF schließlich entfernt berechenbare Attribute.