BKO-Unterrichtsinhalte

NORMALISIERUNG – INFO

Normalisierung (Datenbank)

aus: https://de.wikipedia.org/wiki/Normalisierung_(Datenbank)

Was ist Normalisierung?

Unter **Normalisierung** eines relationalen Datenschemas (Tabellenstruktur) versteht man die Aufteilung von Attributen (Tabellenspalten) in mehrere Relationen (Tabellen) gemäß den Normalisierungsregeln (s. u.), so dass eine Form entsteht, die keine vermeidbaren Redundanzen mehr enthält.

Der Begriff der **Redundanz** beschreibt … diejenigen Informationen, die in einer Informationsquelle mehrfach vorhanden sind. Eine Informationseinheit ist dann redundant, wenn sie ohne Informationsverlust weggelassen werden kann.

Warum Normalisierung?

Ein konzeptionelles Schema, das Datenredundanzen enthält, kann dazu führen, dass bei Änderungen der damit realisierten Datenbank die mehrfach enthaltenen Daten nicht konsistent, sondern nur teilweise und unvollständig geändert werden, womit sie obsolet oder widersprüchlich werden können. Man spricht von auftretenden Anomalien. Zudem belegt mehrfache Speicherung derselben Daten unnötig Speicherplatz. Um Redundanz zu verhindern, normalisiert man solche Tabellen.

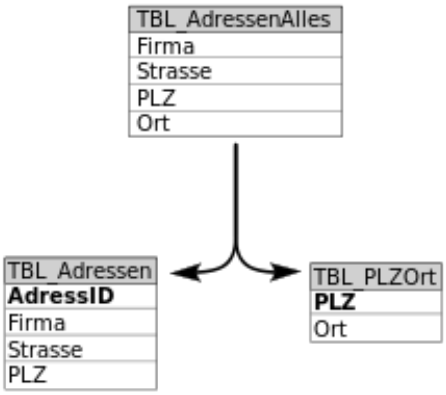
Die Normalisierung hat den Zweck, Redundanzen (mehrfaches Festhalten des gleichen Sachverhalts) zu verringern und dadurch verursachte Anomalien (z. B. infolge Änderung an nicht allen Stellen) zu verhindern, um so die Aktualisierung einer Datenbank zu vereinfachen (Änderungen lediglich an einer Stelle) sowie die Konsistenz der Daten zu gewährleisten.

1., 2. oder 3. NF (Normalform)?

Es gibt verschiedene Ausmaße, in denen ein Datenbankschema gegen Anomalien gefeit sein kann. Je nachdem spricht man davon, dass es in erster, zweiter, dritter usw. **Normalform** vorliege. Diese Normalformen sind durch bestimmte formale Anforderungen an das Schema definiert.

Schrittweise Normalisierung

Man bringt ein relationales Datenschema in eine Normalform, indem man fortschreitend anhand für sie geltender funktionaler Abhängigkeiten seine Relationen in einfachere zerlegt, bis keine weitere Zerlegung mehr möglich ist. Dabei dürfen jedoch auf keinen Fall Daten verloren gehen. (…)



—> mit jeder weiteren NF werden die Tabellen in weitere Tabellen verlegt.
Aufspaltung der Tabelle TBL_AdressenAlles

Bei der Normalisierung in diesen Bereichen werden zunächst Spalten (synonyme Begriffe: *Felder*, *Attribute*) von Tabellen innerhalb dieser Bereiche (der Datenschemata) in neue Spalten aufgeteilt, z. B. Adressen in Postleitzahl, Ort und Strasse. Anschließend werden Tabellen aufgeteilt, zum Beispiel eine Tabelle
tbl_AdressenAlles mit den Feldern Firma, Strasse, PLZ und Ort in diese Tabellen:

- tbl_Adressen mit den Feldern AdressID, Firma, Strasse und PLZ
- tbl_PLZOrt mit den Feldern PLZ und Ort

Siehe Bild *Aufspaltung der Tabelle tbl_AdressenAlles* – wobei die Tabelle tbl_Adressen noch den eindeutigen Primärschlüssel AdressID erhält. In diesem Beispiel wird angenommen, dass es zu jeder Postleitzahl nur jeweils einen Ortsnamen gibt, was in Deutschland jedoch nicht immer zutrifft.

Ein Beispiel dazu: Eine Datenbank enthält Kunden und deren Adressen sowie Aufträge, die den Kunden zugeordnet sind. Da es mehrere Aufträge vom selben Kunden geben kann, würde eine Erfassung der Kundendaten (womöglich mit Adressdaten) in der Auftragstabelle dazu führen, dass sie dort mehrfach vorkommen, obwohl der Kunde immer nur einen Satz gültiger Daten hat (Redundanz). Beispielsweise kann es dazu kommen, dass in einem Auftrag fehlerhafte Adressdaten zum Kunden eingegeben werden, im nächsten Auftrag werden die korrekten Daten erfasst. So kann es – in dieser Tabelle oder auch gegenüber anderen Tabellen – zu widersprüchlichen Daten kommen. Die Daten wären dann nicht konsistent, man wüsste nicht, welche Daten korrekt sind. Womöglich sind sogar beide Adressen nicht korrekt, weil der Kunde umgezogen ist *(Lösung siehe unten)*.

Bei einer normalisierten Datenbank gibt es für die Kundendaten nur einen einzigen Eintrag in der Kundentabelle, mit der jeder Auftrag dieses Kunden verknüpft wird (üblicherweise über die Kundennummer). Im Falle des Umzugs eines Kunden (ein anderes Beispiel ist die Änderung der Mehrwertsteuer) gäbe es zwar mehrere Einträge in der entsprechenden Tabelle, die aber zusätzlich durch die Angabe eines *Gültigkeitszeitraums*unterscheidbar sind und im obigen Kundenbeispiel über die Kombination *Auftragsdatum/Kundennummer* eindeutig angesprochen werden können.

Ein weiterer Vorteil von Redundanzfreiheit, der bei Millionen Datensätzen einer Datenbank auch heute noch eine wichtige Rolle spielt, ist der geringere Speicherbedarf, wenn der Datensatz einer Tabelle zum Beispiel tbl_Auftrag auf einen Datensatz einer anderen Tabelle z. B. tbl_Kunde verweist, anstatt diese Daten selbst zu enthalten.

Aufspaltung von Tabellen zur Normalisierung

Dieses sind die Empfehlungen, die ausgehend von der Theorie der Normalisierung bei der Datenbankentwicklung gegeben werden, um vor allem Konsistenz der Daten und eine eindeutige Selektion von Daten zu gewährleisten. Die hierzu angestrebte Redundanzfreiheit steht allerdings in speziellen Anwendungsfällen in Konkurrenz zur Verarbeitungsgeschwindigkeit oder zu anderen Zielen. Es kann daher sinnvoll sein, auf eine Normalisierung zu verzichten oder diese durch eine Denormalisierung rückgängig zu machen, um

- die Verarbeitungsgeschwindigkeit (Performanz) zu erhöhen oder
- Anfragen zu vereinfachen und damit die Fehleranfälligkeit zu verringern oder
- Besonderheiten von Prozessen (zum Beispiel Geschäftsprozessen) abzubilden.

In diesen Fällen sollten regelmäßig automatische Abgleichroutinen implementiert werden, um Inkonsistenzen zu vermeiden. Alternativ können die betreffenden Daten auch für Änderungen gesperrt werden.

Zurzeit gebräuchliche Normalformen sind:

- ***1. Normalform (1NF)***
- ***2. Normalform (2NF)***
- ***3. Normalform (3NF)***
- *(Boyce–Codd-Normalform (BCNF)*
- *4. Normalform (4NF)*
- *5. Normalform (5NF))*

1. Normalform

Jedes Attribut der Relation muss einen atomaren Wertebereich haben, und die Relation muss frei von Wiederholungsgruppen sein. (Anm.: statt „atomar“ wird auch die Bezeichnung „atomisch“ verwendet.)

Atomar heißt, dass zusammengesetzte, mengenwertige oder geschachtelte Wertebereiche (also relationenwertige Attributwertebereiche) nicht erlaubt sind. Der Wertebereich keines Attributs einer Relation in 1NF kann in weitere (sinnvolle) Teilbereiche aufgespaltet werden.

Beispiel: Die Adresse darf nicht als einzelnes Attribut verwendet werden, sondern muss – sofern es der zugrunde liegende Prozess erfordert – in PLZ, Ort, Straße, Hausnummer etc. aufgeteilt werden.

Frei von Wiederholungsgruppen bedeutet, dass Attribute, die *gleiche* oder gleich*artige* Information enthalten, in eine andere Relation ausgelagert werden müssen.

Ein Beispiel für eine Wiederholungsgruppe wäre eine Spalte { Telefon }, die *mehrere* Telefonnummern enthält oder auch eine Spaltengruppe { Telefon1, Telefon2, Telefon3 }, wobei im letzteren Fall anzumerken ist, dass es sich dabei nicht notwendigerweise um eine Wiederholungsgruppe handeln muss (siehe Alternative Formulierungen).

Praktischer Nutzen
Abfragen der Datenbank werden durch die 1NF erleichtert bzw. überhaupt erst ermöglicht, wenn die Attributwertebereiche *atomar* sind. So ist es beispielsweise in einem Feld, das einen ganzen

Namensstring aus Titel, Vorname und Nachname enthält, schwierig bis unmöglich, nach Nachnamen zu sortieren.

(…)

CD_Lied

CD_ID	ALBUM	ERSCHEINUNGS-JAHR	TITELLISTE
4711	Anastacia – Not That Kind	2000	{1. Not That Kind, 2. I'm Outta Love, 3. Cowboys & Kisses}
4712	Pink Floyd – Wish You Were Here	1975	{1. Shine On You Crazy Diamond}
4713	Anastacia – Freak of Nature	2001	{1. Paid my Dues}

- Das Feld **Album** enthält die Attributwertebereiche Interpret und Albumtitel.
- Das Feld **Titelliste** enthält eine Menge von Titeln.

Dadurch hat man ohne Aufspaltung folgende Probleme bei Abfragen:

- Zur Sortierung nach Albumtitel muss das Feld **Album** in Interpret und Albumtitel aufgeteilt werden.
- Die Titel können (mit einfachen Mitteln) nur alle gleichzeitig als **Titelliste** oder gar nicht dargestellt werden.

CD_Lied

CD_ID	ALBUMTITEL	INTERPRET	ERSCHEINUNGSJAHR	*TRACK*	TITEL
4711	Not That Kind	Anastacia	2000	1	Not That Kind
4711	Not That Kind	Anastacia	2000	2	I'm Outta Love
4711	Not That Kind	Anastacia	2000	3	Cowboys & Kisses
4712	Wish You Were Here	Pink Floyd	1975	1	Shine On You Crazy Diamond
4713	Freak of Nature	Anastacia	2001	1	Paid my Dues

Die Attributwertebereiche werden in atomare Attributwertebereiche aufgespalten:

- Das Feld **Album** wird in die Felder *Albumtitel* und *Interpret* gespalten.
- Das Feld **Titelliste** wird in die Felder *Track* und *Titel* gespalten sowie auf mehrere Datensätze aufgeteilt.

Da jetzt jeder Attributwertebereich atomar ist sowie die Tabelle einen eindeutigen Primärschlüssel (Verbundschlüssel aus den Spalten *CD_ID* und *Track*) besitzt, befindet sich die Relation in 1NF.

2. Normalform

Eine Relation ist in der zweiten Normalform genau dann, wenn die erste Normalform vorliegt und jedes Nichtprimärattribut (nicht Teil eines Schlüssels) ist jeweils von allen **ganzen** Schlüsseln abhängig, nicht nur von einem Teil eines Schlüssels. Wichtig ist hierbei, dass die Nichtschlüsselattribute wirklich von *allen* Schlüsseln vollständig abhängen.

Eine Relation ist genau dann in zweiter Normalform, wenn sie

Die 2NF eliminiert alle *partiellen* funktionalen Abhängigkeiten, d. h. kein Nichtschlüsselattribut ist funktional abhängig von Teilen des Schlüsselkandidaten.

Praktischer Nutzen
Die 2NF erzwingt wesentlich „monothematische“ Relationen im Schema: jede Relation modelliert nur **einen** Sachverhalt.

Dadurch werden Redundanz und die damit einhergehende Gefahr von Inkonsistenzen reduziert. Nur noch logisch/sachlich zusammengehörige Informationen finden sich in einer Relation. Dadurch fällt das Verständnis der Datenstrukturen leichter.

CD_Lied

CD_ID	ALBUMTITEL	INTERPRET	ERSCHEINUNGSJAHR	*TRACK*	TITEL
4711	Not That Kind	Anastacia	2000	1	Not That Kind
4711	Not That Kind	Anastacia	2000	2	I’m Outta Love
4711	Not That Kind	Anastacia	2000	3	Cowboys & Kisses
4712	Wish You Were Here	Pink Floyd	1975	1	Shine On You Crazy Diamond
4713	Freak of Nature	Anastacia	2001	1	Paid my Dues

- Der Primärschlüssel der Relation ist aus den Feldern *CD_ID* und *Track* zusammengesetzt. (Grundsätzlich darf ein Primärschlüssel aus mehreren Attributen bestehen, jedoch entsteht daraus im genannten Beispiel ein Konflikt.)
- Die Felder *Albumtitel*, *Interpret* und *Erscheinungsjahr* sind vom Feld *CD_ID* abhängig, aber nicht vom Feld *Track*. Dieser (Punkt 2) verletzt die 2. Normalform, da die drei nicht-primären Attribute nicht nur von einem Teil des Schlüssels (hier *CD_ID*) abhängen dürfen. Wäre der Schlüssel nicht zusammengesetzt (siehe Punkt 1), so könnte dies nicht passieren.

Probleme, die sich daraus ergeben:
Die Informationen aus diesen drei Feldern sind, wie am Beispiel der CD *Not That Kind* zu erkennen, mehrfach vorhanden, d. h. redundant. Dadurch besteht die Gefahr, dass die *Integrität der Daten* verletzt wird. So könnte man den Albumtitel für das Lied *Not That Kind* in *I Don't Mind* ändern, ohne jedoch die entsprechenden Einträge für die Titel *I'm Outta Love* und *Cowboys & Kisses* zu ändern (Update-Anomalie).

CD_Lied (inkonsistent)

CD_ID	ALBUMTITEL	INTERPRET	ERSCHEINUNGSJAHR	*TRACK*	TITEL
4711	**I Don't Mind**	Anastacia	2000	1	Not That Kind
4711	Not That Kind	Anastacia	2000	2	I'm Outta Love
4711	Not That Kind	Anastacia	2000	3	Cowboys & Kisses
4712	Wish You Were Here	Pink Floyd	1975	1	Shine On You Crazy Diamond
4713	Freak of Nature	Anastacia	2001	1	Paid my Dues

In diesem Fall wäre ein Zustand erreicht, den man als Dateninkonsistenz bezeichnet. Über die komplette Tabelle betrachtet, „passen" die Daten nicht mehr zusammen.

Die Daten in der Tabelle werden in zwei Tabellen aufgeteilt: *CD* und *Lied*. Die Tabelle CD enthält nur noch Felder, die voll funktional von *CD_ID* abhängen, hat also *CD_ID* als Primärschlüssel. Auch der Albumtitel allein sei eindeutig, also ein Schlüsselkandidat. Da keine weiteren (zusammengesetzten) Schlüsselkandidaten existieren, liegt die Tabelle damit automatisch in der 2. Normalform vor. Die Tabelle „Lied" enthält schließlich nur noch Felder, die voll funktional von *CD_ID* und *Track* abhängen, liegt also auch in der 2. Normalform vor. Mit Hilfe dieser verlustfreien Zerlegung sind auch die genannten Redundanzen der Daten beseitigt.

CD

Lied

CD_ID	ALBUMTITEL	INTER-PRET	ERSCHEINUNGS-JAHR
4711	Not That Kind	Anastacia	2000
4712	Wish You Were Here	Pink Floyd	1975
4713	Freak of Nature	Anastacia	2001

CD_ID	*TRACK*	TITEL
4711	1	Not That Kind
4711	2	I'm Outta Love
4711	3	Cowboys & Kisses
4712	1	Shine On You Crazy Diamond
4713	1	Paid my Dues

Das Attribut *CD_ID* aus der Tabelle *Lied* bezeichnet man als Fremdschlüssel, der auf den Primärschlüssel der Tabelle *CD* verweist. Zugleich stellen die Attribute *CD_ID* und *Track* den zusammengesetzten Primärschlüssel der Tabelle Lied dar.

3. Normalform

Die dritte Normalform ist genau dann erreicht, wenn sich das Relationenschema in der 2NF befindet, und ***kein*** *Nichtschlüsselattribut (hellgraue Zellen in der Tabelle) von einem Schlüsselkandidaten* ***transitiv*** *abhängt.*

(…)

Praktischer Nutzen
Transitive Abhängigkeiten sind sofort ersichtlich, ohne dass man die Zusammenhänge der Daten kennen muss. Sie sind durch die Struktur der Relationen wiedergegeben.

Außerdem werden verbliebene thematische Durchmischungen in der Relation behoben: nach der 3NF sind die Relationen des Schemas zuverlässig monothematisch.

Oder: Die dritte Normalform ist erreicht, wenn sich das Relationenschema in 2NF befindet, und kein Nichtschlüsselattribut (hellgraue Zellen in der Tabelle) von einem anderen Nichtschlüsselattribut funktional abhängig ist.

CD

CD_ID	ALBUMTITEL	INTERPRET	GRÜNDUNGSJAHR
4711	Not That Kind	Anastacia	1999
4712	Wish You Were Here	Pink Floyd	1965

4713	Freak of Nature	Anastacia	1999

Offensichtlich lässt sich der *Interpret* einer CD aus der *CD_ID* bestimmen, das *Gründungsjahr* der Band/Interpreten hängt wiederum vom *Interpreten* und damit transitiv von der *CD_ID* ab.

Das Problem ist hierbei wieder Datenredundanz. Wird zum Beispiel eine neue CD mit einem existierenden *Interpreten* eingeführt, so wird das *Gründungsjahr* redundant gespeichert.

CD

CD_ID	**ALBUM-TITEL**	**INTERPRET_ID**
4711	Not That Kind	311
4712	Wish You Were Here	312
4713	Freak of Nature	311

Künstler

INTERPRET_ID	**INTERPRET**	**GRÜNDUNGSJAHR**
311	Anastacia	1999
312	Pink Floyd	1965

Lied

CD_ID	***TRACK***	**TITEL**
4711	1	Not That Kind
4711	2	I'm Outta Love
4711	3	Cowboys & Kisses
4712	1	Shine On You Crazy Diamond
4713	1	Paid my Dues

Die Relation wird aufgeteilt, wobei die beiden voneinander abhängigen Daten in eine eigene Tabelle ausgelagert werden. Der Schlüssel der neuen Tabelle muss als Fremdschlüssel in der alten Tabelle erhalten bleiben.

An der Tabelle „Lied" wurden keine Änderungen bei der Übertragung in die 3. Normalform vorgenommen. Sie ist hier nur der Vollständigkeit halber gelistet.

(…)
Das bedeutet:

- alle (impliziert: atomaren) Werte beziehen sich auf den **Schlüssel** – 1. NF
- bei zusammengesetzten Schlüsseln beziehen sie sich jeweils auf den **gesamten Schlüssel** – 2. NF
- die Werte hängen **nur vom Schlüssel** ab, und nicht von Nichtschlüsselattributen – 3. NF

(…)

1. Ist die Relation in 1. Normalform und besteht der Primärschlüssel aus nur einem Attribut, so liegt automatisch die 2. Normalform vor.
2. Ist eine Relation in 2. Normalform und besitzt sie außer dem Primärschlüssel höchstens ein weiteres Attribut, so liegt die Tabelle in 3. Normalform vor.
3. Die 3. Normalform ist die letzte und muss die 1. und 2. Bedingungen erledigen.

BEARBEITEN