1. Normalform (1NF)

Atomare Attribute. Die Erste Normalform (1NF) ist dann gegeben, wenn alle Informationen in einer Tabelle atomar vorliegen.

Eine Relation befindet sich also dann in der ersten Normalform (1NF), wenn in keinem Feld in der Tabellen mehrere Werte eingetragen sind. Wann genau ein Attribut atomar ist, hängt manchmal auch noch von den Randbedingungen der Miniwelt ab, die man modellieren möchte - für viele Miniwelten reicht ein ISO-Datumsfeld der Form YYYY-MM-DD aus, manche Modelle erfordern aber eventuell die weitere Aufteilung in die Attribute Tag, Monat und Jahr.

(A1)

Wir gehen von unserer Universaltabelle aus:

id	produkt	preis	nummer	anzahl	doktor	telefon_fax	adresse	hersteller
1	Schlagbohrer	199.95	1000-1	1	Blutgesicht, Hans	123-550,123-551	Kariesweg 1, 12345 Zahnstein	Hilti, Bohrmaschinenstr. 58a, 88887 Hilterhausen
2	Zement	5.95	1000-2	5	Quälstein, Bertha	456-777,456-778	Lochpfad 23, 23458 Schmerzstadt	Hoch & Tief, Am Höhenzug 22, 12345 Tiefenbau
3	Kneifzange	19.95	1000-3	3	Eisenfaust, Viktoria	789-250,789-251	Zahnwurzel 3, 87454 Dolomostadt	Eisen-Karl, Karlstrasse 5, 76542 Eisenhafen
4	Brecheisen	49.95	1000-4	7	Rostzange, Eberhart	234-100,234-101	Peinweg 5, 74512 Reissheim	Eisen-Karl, Karlstrasse 5, 76542 Eisenhafen
5	Hammer	19.95	1000-5	4	Frankenstein, Emilia	567-200,567-201	Am Dom 5, 50670 Köln	Stahl AG, Blechweg 102b,89452 Alustetten
6	Zement	5.95	1000-2	9	Eisenfaust, Viktoria	789-250,789-251	Zahnwurzel 3, 87454 Dolomostadt	Hoch & Tief, Am Höhenzug 22, 12345 Tiefenbau
7	Brecheisen	49.95	1000-4	2	Blutgesicht, Hans	123-550,123-551	Kariesweg 1, 12345 Zahnstein	Eisen-Karl, Karlstrasse 5, 76542 Eisenhafen

- Welche Attribute sind nicht atomar?
- Welche Schwierigkeiten ergeben sich daraus, wenn du mit SQL Informationen aus der Datenbank abfragen möchtest? Nenne drei Beispiele, wo Probleme entstehen, weil die Attribute nicht atomar sind.

Importiere die

Universaltabelle

in deine Übungsdatenbank.

(i) Erstelle die Tabellenfelder `name` und `vorname` und überführe die Inhalte des Attributs `doktor` in diese Felder. Dabei kann du die mysql-Funktion `SUBSTRING_INDEX` verwenden[1)]. Mit dieser kann man den Inhalt eines Tabellenfelds an einem Trennzeichen in Teile zerlegen, die man über Ihren Index ansprechen kann:

Probiere die folgenden SQL-Statements aus und mache dir klar, was dabei passiert, du kannst dir auch mal die Dokumentation zu `SUBSTRING_INDEX` durchlesen:
https://mariadb.com/kb/en/substring_index/

- https://www.info-bw.de/

```
SELECT SUBSTRING_INDEX(doktor, ',', 1) FROM `zahnarztbedarf`
SELECT SUBSTRING_INDEX(doktor, ',', -1) FROM `zahnarztbedarf`
```

Jetzt kann man die am Komma aufgesplitteten Werte in die neuen Felder übertragen:

```
UPDATE zahnarztbedarf SET name = SUBSTRING_INDEX(doktor, ',', 1)
UPDATE zahnarztbedarf SET vorname = SUBSTRING_INDEX(doktor, ',', -1)
```

Deine Tabelle sollte jetzt so aussehen:

id	produkt	preis	nummer	anzahl	doktor	name	vorname	telefon_fax	adresse	hersteller
1	Schlagbohrer	199.95	1000-1	1	Blutgesicht, Hans	Blutgesicht	Hans	123-550,123-551	Kariesweg 1, 12345 Zahnstein	Hilti, Bohrmaschinenstr. 58a, 88887 Hilterhausen
2	Zement	5.95	1000-2	5	Quälstein, Bertha	Quälstein	Bertha	456-777,456-778	Lochpfad 23, 23458 Schmerzstadt	Hoch & Tief, Am Höhenzug 22, 12345 Tiefenbau
3	Kneifzange	19.95	1000-3	3	Eisenfaust, Viktoria	Eisenfaust	Viktoria	789-250,789-251	Zahnwurzel 3, 87454 Dolomostadt	Eisen-Karl, Karlstrasse 5, 76542 Eisenhafen
4	Brecheisen	49.95	1000-4	7	Rostzange, Eberhart	Rostzange	Eberhart	234-100,234-101	Peinweg 5, 74512 Reissheim	Eisen-Karl, Karlstrasse 5, 76542 Eisenhafen
5	Hammer	19.95	1000-5	4	Frankenstein, Emilia	Frankenstein	Emilia	567-200,567-201	Am Dom 5, 50670 Köln	Stahl AG, Blechweg 102b,89452 Alustetten
6	Zement	5.95	1000-2	9	Eisenfaust, Viktoria	Eisenfaust	Viktoria	789-250,789-251	Zahnwurzel 3, 87454 Dolomostadt	Hoch & Tief, Am Höhenzug 22, 12345 Tiefenbau
7	Brecheisen	49.95	1000-4	2	Blutgesicht, Hans	Blutgesicht	Hans	123-550,123-551	Kariesweg 1, 12345 Zahnstein	Eisen-Karl, Karlstrasse 5, 76542 Eisenhafen

Nun kannst du die Spalte `doktor` löschen, da die Informationen jetzt atomar in den Attributen `name` und `vorname` vorliegen.

(ii) Zerlege das Feld `telefon_fax` auf dieselbe Weise in die atomaren Attribute `telefon` und `fax`. Erstelle zunächst die neuen Felder, kopiere dann das SQL Statement von oben und passe es entsprechend an. Kontrolliere den Erfolg und lösche dann die Spalte `telefon_fax`.

Lösung

```
UPDATE zahnarztbedarf t1 SET
t1.fax = (SELECT SUBSTRING_INDEX(telefon_fax, ',', -1) FROM `zahnarztbedarf`
t2 WHERE t2.id = t1.id ),
t1.telefon = (SELECT SUBSTRING_INDEX(telefon_fax, ',', 1) FROM
`zahnarztbedarf` t2 WHERE t2.id = t1.id )
WHERE t1.fax=''
```

(iii) Zerlege das Feld `adresse` in die Felder `strasse`, `wohnort` und `plz`. Dabei musst du **zweischrittig** vorgehen, da du den Feldinhalt an verschiedenen Trennzeichen splitten musst: Überführe mit dem Statement von oben zunächst die Strassen in das Feld `strasse` und die Kombination aus PLZ und Wohnort in ein temporäres Feld `plzwo` indem du am Komma teilst.

Überführe dann den Inhalt des temporären Felds nach `plz` und `wohnort` indem du am Leerzeichen splittest. Lösche dann das Feld `adresse` und das Zwischenfeld `plzwo`.

Wichtig: Um das plzwo Feld sauber am Leerzeichen trennen zu können, muss man sicherstellen, dass kein führendes Leerzeichen mehr vorhanden ist, wo zuvor die Zeichenkombination `,<LEER>` war, das kann man mit dem mysql Befehl `TRIM` erreichen: `SELECT SUBSTRING_INDEX(TRIM(plzwo), '`

', 1) FROM `zahnarztbedarf`.

Lösung Schritt 1

```
UPDATE zahnarztbedarf t1 SET
t1.plzwo = (SELECT SUBSTRING_INDEX(adresse, ',', -1) FROM `zahnarztbedarf`
t2 WHERE t2.id = t1.id ),
t1.strasse = (SELECT SUBSTRING_INDEX(adresse, ',', 1) FROM `zahnarztbedarf`
t2 WHERE t2.id = t1.id )
WHERE t1.plzwo = ''
```

Lösung Schritt 2

```
UPDATE zahnarztbedarf t1 SET
t1.wohnort = (SELECT SUBSTRING_INDEX(TRIM(plzwo), ' ', -1) FROM
`zahnarztbedarf` t2 WHERE t2.id = t1.id ),
t1.plz = (SELECT SUBSTRING_INDEX(TRIM(plzwo), ' ', 1) FROM `zahnarztbedarf`
t2 WHERE t2.id = t1.id )
WHERE t1.plz = ''
```

(iv) Jetzt wirds langweilig... Nun muss man das ganze nochmal für das wilde Durcheinander im Feld Hersteller wiederholen - das ist freiwillig und birgt keine wesentlichen neuen Erkenntnisse[2)] mehr, du kannst also auch gleich die Lösung bemühen oder das Ergebnis herunterladen.

Lösung

4 neue Tabellenfelder + temporäres Feld:

```
SELECT SUBSTRING_INDEX(hersteller, ',', 1) FROM `zahnarztbedarf`  -- liefert
die firma
SELECT SUBSTRING_INDEX(hersteller, ',', -1) FROM `zahnarztbedarf`  --
liefert ort + plz
```

Problem. Wie bekommt man das mittlere Feld raus? Probiere mal folgendes aus:

```
SELECT SUBSTRING_INDEX(hersteller, ',', 2) FROM `zahnarztbedarf`  -- liefert
die ersten beiden, durch komma getrennt, also "Eisen-Karl, Karlstrasse 5"
SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(hersteller, ',' , 2), ',' , -1) FROM
`zahnarztbedarf` -- liefert davon den zweiten teil - also die strasse + hnr
```

Nun haben wir alles zusammen:

```
UPDATE zahnarztbedarf t1 SET
t1.firma = (SELECT SUBSTRING_INDEX(hersteller, ',', 1) FROM `zahnarztbedarf`
t2 WHERE t2.id = t1.id ),
t1.f_strasse = (SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(hersteller, ',' , 2),
```

```
',' , -1) FROM `zahnarztbedarf` t2 WHERE t2.id = t1.id ),
t1.temp = (SELECT SUBSTRING_INDEX(hersteller, ',', -1) FROM `zahnarztbedarf`
t2 WHERE t2.id = t1.id )
WHERE t1.firma = ''
```

Jetzt noch analog zu oben PLZ und Ort aufteilen, TRIM nicht vergessen:

```
UPDATE zahnarztbedarf t1 SET
t1.f_ort = (SELECT SUBSTRING_INDEX(TRIM(temp), ' ', -1) FROM
`zahnarztbedarf` t2 WHERE t2.id = t1.id ),
t1.f_plz = (SELECT SUBSTRING_INDEX(TRIM(temp), ' ', 1) FROM `zahnarztbedarf`
t2 WHERE t2.id = t1.id )
WHERE t1.f_plz = ''
```

Jetzt kann man `hersteller` und `temp` löschen.

Ergebnis: Die Universaltabelle in der 1NF

id	produkt	preis	nummer	anzahl	name	vorname	telefon	fax	strasse	wohnort	plz	firma	f_strasse	f_ort	f_plz
1	Schlagbohrer	199.95	1000-1	1	Blutgesicht	Hans	123-550	123-551	Kariesweg 1	Zahnstein	12345	Hilti	Bohrmaschinenstr. 58a	Hilterhausen	88887
2	Zement	5.95	1000-2	5	Quälstein	Bertha	456-777	456-778	Lochpfad 23	Schmerzstadt	23458	Hoch & Tief	Am Höhenzug 22	Tiefenbau	12345
3	Kneifzange	19.95	1000-3	3	Eisenfaust	Viktoria	789-250	789-251	Zahnwurzel 3	Dolomostadt	87454	Eisen-Karl	Karlstrasse 5	Eisenhafen	76542
4	Brecheisen	49.95	1000-4	7	Rostzange	Eberhart	234-100	234-101	Peinweg 5	Reissheim	74512	Eisen-Karl	Karlstrasse 5	Eisenhafen	76542
5	Hammer	19.95	1000-5	4	Frankenstein	Emilia	567-200	567-201	Am Dom 5	Köln	50670	Stahl AG	Blechweg 102b	Alustetten	89452
6	Zement	5.95	1000-2	9	Eisenfaust	Viktoria	789-250	789-251	Zahnwurzel 3	Dolomostadt	87454	Hoch & Tief	Am Höhenzug 22	Tiefenbau	12345
7	Brecheisen	49.95	1000-4	2	Blutgesicht	Hans	123-550	123-551	Kariesweg 1	Zahnstein	12345	Eisen-Karl	Karlstrasse 5	Eisenhafen	76542

Download Zahnarztbedarf 1NF

An diesen Übungen kann man gut erkennen, wie wichtig es ist, bereits beim **Datenbankdesign** an die Normalisierung zu denken - man will sich gar nicht vorstellen, was man für Knoten ins Hirn bekommt, wenn man das hier gezeigte mit einer großen, schlecht designten Datenbank machen muss.

Ausserdem kann man an den Beispielen mit `SUBSTRING_INDEX` und `TRIM` gut erkennen, dass **atomare Attribute** Abfragen vereinfachen und Fehlerquellen eliminieren.

1) https://mariadb.com/kb/en/substring_index/

2) Außer, wie man das mittlere Feld aus einer CSV Liste extrahiert...

From:
https://www.info-bw.de/ -

Permanent link:
https://www.info-bw.de/faecher:informatik:oberstufe:datenbanken:normalisierung:1_normalform:start?rev=1606386987

Last update: **26.11.2020 10:36**