Vorüberlegungen

Wir versuchen uns, in das folgende Szenario einzudenken:

- Wir sind ein Zwischenhändler, der Zahnärzte mit allem ausstattet, was die brauchen, um ihre Patienten zu foltern.
- Wir müssen Buch führen über unsere Kunden (Doktoren), unsere Lieferanten, angebotene Produkte und unsere Bestellungen.
- Unser sehr unerfahrene Sekretär hat mal was von mysql gehört und die derzeit bekannten Daten in eine Tabelle geschrieben.

Die Ausgangslage

Da unser Sekretär normal nur mit Tabellenkalkulationen arbeitet, hat er alle Infos einfach mal in eine Tabelle gepackt:

Probleme...

Auf den ersten Blick fällt auf, dass in unserer Tabelle etliche Informationen mehrfach gespeichert sind, das kann zu Problemen führen. Im Moment sieht alles noch sehr übersichtlich aus. Aber was passiert, wenn wir 10.000 oder 100.000 Datensätze verwalten müssen? Was, wenn ein Kunde den Händler wechselt oder sich die Adresse eines Herstellers ändert? Wie kann jemand etwas bestellen, wenn nicht irgendwo ersichtlich ist, welche Produkte es überhaupt gibt?

Um diesen Problemen zu begegnen, sollte man Datenbanken "normalisieren". Allerdings müssen wir vorher noch ein paar Begriffe klären.

Redundanzen, Anomalien, Inkonsistenzen

Hinter diesen Begriffen verbirgt sich alles, was den logischen Aufbau unserer Datenbank gefährden könnte. Einen ersten Einblick in die Problematik habe ich euch schon oben gegeben.

Redundanzen, Redundanzfreiheit

Redundanzen sind einfach ausgedrückt, doppelt gespeicherte Informationen. In unserem ersten Entwurf haben wird die zuhauf, bei den Produkten, Preisen, Doktoren oder Adressen. Redundanzfreiheit ist also nur das Gegenteil von Redundanz, also die Vermeidung doppelter Einträge. Die Vorteile liegen auf der Hand.

- Geringerer Speicherbedarf
- Erhöhung der Wartbarkeit
- Vermeidung von Bevor wir uns an ein gutes Design heranwagen, schauen wir uns an, wie man

es nicht macht und warum. Nehmen wir mal folgende Situation an. Wir sind ein Zwischenhändler, der Zahnärzte mit allem ausstattet, was die brauchen, um uns zu foltern.

Erster Entwurf

Man könnte nun auf die Idee kommen, alle Bestellungen, so wie man es normalerweise aus Excel gewöhnt ist, einfach in eine Tabelle zu packen.

```
+--+------------+------+------+------+------------+---------------+---------
----------------------+-----------------------+
|id|produkt     |preis |nummer|anzahl|doktor      |telefon_fax    |adresse
|hersteller             |
+--+------------+------+------+------+------------+---------------+---------
----------------------+-----------------------+
| 1|Schlagbohrer|199.95|1000-1|1     |Blutgesicht |123-550,123-551|Kariesweg
1, 12345 Zahnstein   |Hilti, Adresse...
| 2|Zement      |  5.95|1000-2|5     |Quälstein   |456-777,456-778|Lochpfad
23, 23458 Schmerzstadt|Hoch & Tief, Adresse...|
| 3|Kneifzange  | 19.95|1000-3|3     |Eisenfaust
|789-250,789-251|Zahnwurzel 3, 87454 Dolomostadt|Eisen-Karl, Adresse... |
| 4|Brecheisen  | 49.95|1000-4|7     |Rostzange   |234-100,234-101|Peinweg
5, 74512 Reissheim     |Eisen-Karl, Adresse... |
| 5|Hammer      | 19.95|1000-5|4     |Frankenstein|567-200.567-201|Am Dom 5,
50670 Köln           |Stahl AG, Adresse...   |
| 6|Zement      |  5.95|1000-2|9     |Eisenfaust
|789-250,789-251|Zahnwurzel 3, 87454 Dolomostadt|Hoch & Tief, Adresse...|
| 7|Brecheisen  | 49.95|1000-4|2     |Blutgesicht |123-550,123-551|Kariesweg
1, 12345 Zahnstein   |Eisen-Karl, Adresse... |
+--+------------+------+------+------+------------+---------------+---------
----------------------+-----------------------+
```

Bevor wir uns an ein gutes Design heranwagen, schauen wir uns an, wie man es nicht macht und warum. Nehmen wir mal folgende Situation an. Wir sind ein Zwischenhändler, der Zahnärzte mit allem ausstattet, was die brauchen, um uns zu foltern.

Erster Entwurf

Man könnte nun auf die Idee kommen, alle Bestellungen, so wie man es normalerweise aus Excel gewöhnt ist, einfach in eine Tabelle zu packen.

```
+--+------------+------+------+------+------------+---------------+---------
----------------------+-----------------------+
|id|produkt     |preis |nummer|anzahl|doktor      |telefon_fax    |adresse
|hersteller             |
+--+------------+------+------+------+------------+---------------+---------
----------------------+-----------------------+
```

```
| 1|Schlagbohrer|199.95|1000-1|1     |Blutgesicht |123-550,123-551|Kariesweg
1, 12345 Zahnstein   |Hilti, Adresse...
| 2|Zement      |  5.95|1000-2|5     |Quälstein   |456-777,456-778|Lochpfad
23, 23458 Schmerzstadt|Hoch & Tief, Adresse...|
| 3|Kneifzange  | 19.95|1000-3|3     |Eisenfaust
|789-250,789-251|Zahnwurzel 3, 87454 Dolomostadt|Eisen-Karl, Adresse... |
| 4|Brecheisen  | 49.95|1000-4|7     |Rostzange   |234-100,234-101|Peinweg
5, 74512 Reissheim     |Eisen-Karl, Adresse... |
| 5|Hammer      | 19.95|1000-5|4     |Frankenstein|567-200.567-201|Am Dom 5,
50670 Köln           |Stahl AG, Adresse...   |
| 6|Zement      |  5.95|1000-2|9     |Eisenfaust
|789-250,789-251|Zahnwurzel 3, 87454 Dolomostadt|Hoch & Tief, Adresse...|
| 7|Brecheisen  | 49.95|1000-4|2     |Blutgesicht |123-550,123-551|Kariesweg
1, 12345 Zahnstein   |Eisen-Karl, Adresse... |
+--+------------+------+------+------+------------+--------------+---------
----------------------+-----------------------+
```

Probleme...

Auf den ersten Blick fällt auf, dass in unserer Tabelle etliche Informationen mehrfach gespeichert sind, das kann zu großen Problemen führen. Im Moment sieht alles noch sehr übersichtlich aus. Aber was passiert, wenn wir 10.000 oder 100.000 DAtensätze verwalten müssen?

Angenommen, Dr. Frankenstein geht künftig wieder seiner ursprünglichen Profession nach und baut Menschen zusammen, er fällt als unser Kunde weg, und belegt als Karteileiche in unserer Datenbank unnötig Speicherplatz.

Oder die Einkaufspreise steigen. Oder Eisen-Karl kann nicht mehr liefern, da er pleite ist. Und wo zum Teufel kommen eigentlich die Produkte her? Wie kann jemand etwas bestellen, wenn nicht irgendwo ersichtlich ist, welche Produkte es überhaupt gibt?

Ihr habt also schon gemerkt, dass unser erster Entwurf ziemlicher Müll ist, da wir bei Änderungen immer wieder vor großen Problemen stehen werden. Um diesen Problemen zu begegnen, sollte man Datenbanken "normalisieren". Allerdings müssen wir vorher noch ein paar Begriffe klären.

Redundanzen, Anomalien, Inkonsistenzen

Hinter diesen Begriffen verbirgt sich alles, was den logischen Aufbau unserer Datenbank gefährden könnte. Einen ersten Einblick in die Problematik habe ich euch schon oben gegeben.

Redundanzen, Redundanzfreiheit

Redundanzen sind einfach ausgedrückt, doppelt gespeicherte Informationen. In unserem ersten Entwurf haben wird die zuhauf, bei den Produkten, Preisen, Doktoren oder Adressen. Redundanzfreiheit ist also nur das Gegenteil von Redundanz, also die Vermeidung doppelter Einträge. Die Vorteile liegen auf der Hand.

- Geringerer Speicherbedarf

- Erhöhung der Wartbarkeit
- Vermeidung von Inkonsistenzen und Anomalien

Inkonsistenzen/Anomalien

Sind widersprüchliche Daten oder logische Brüche innerhalb einer Datenbank. Sie treten sehr oft auf, wenn man zum Beispiel Änderungen an den Inhalten vornimmt und Beziehnungen zwischen Informationen nicht berücksichtigt.

Normalisierung

Unter der "**Normalisierung einer Datenbank**" kann man sich eine Sammlung von Regeln vorstellen, die eingehalten werden sollten, um die erläuterten Probleme zu vermeiden. Wir betrachten im Folgenden die ersten drei "Normalformen".

Probleme...

Auf den ersten Blick fällt auf, dass in unserer Tabelle etliche Informationen mehrfach gespeichert sind, das kann zu großen Problemen führen. Im Moment sieht alles noch sehr übersichtlich aus. Aber was passiert, wenn wir 10.000 oder 100.000 DAtensätze verwalten müssen?

Angenommen, Dr. Frankenstein geht künftig wieder seiner ursprünglichen Profession nach und baut Menschen zusammen, er fällt als unser Kunde weg, und belegt als Karteileiche in unserer Datenbank unnötig Speicherplatz.

Oder die Einkaufspreise steigen. Oder Eisen-Karl kann nicht mehr liefern, da er pleite ist. Und wo zum Teufel kommen eigentlich die Produkte her? Wie kann jemand etwas bestellen, wenn nicht irgendwo ersichtlich ist, welche Produkte es überhaupt gibt?

Ihr habt also schon gemerkt, dass unser erster Entwurf ziemlicher Müll ist, da wir bei Änderungen immer wieder vor großen Problemen stehen werden. Um diesen Problemen zu begegnen, sollte man Datenbanken "normalisieren". Allerdings müssen wir vorher noch ein paar Begriffe klären.

Redundanzen, Anomalien, Inkonsistenzen

Hinter diesen Begriffen verbirgt sich alles, was den logischen Aufbau unserer Datenbank gefährden könnte. Einen ersten Einblick in die Problematik habe ich euch schon oben gegeben.

Redundanzen, Redundanzfreiheit

Redundanzen sind einfach ausgedrückt, doppelt gespeicherte Informationen. In unserem ersten Entwurf haben wird die zuhauf, bei den Produkten, Preisen, Doktoren oder Adressen.

Redundanzfreiheit ist also nur das Gegenteil von Redundanz, also die Vermeidung doppelter Einträge. Die Vorteile liegen auf der Hand.

- Geringerer Speicherbedarf
- Erhöhung der Wartbarkeit
- Vermeidung von Inkonsistenzen und Anomalien

Inkonsistenzen/Anomalien

Sind widersprüchliche Daten oder logische Brüche innerhalb einer Datenbank. Sie treten sehr oft auf, wenn man zum Beispiel Änderungen an den Inhalten vornimmt und Beziehnungen zwischen Informationen nicht berücksichtigt.

Normalisierung

Unter der "**Normalisierung einer Datenbank**" kann man sich eine Sammlung von Regeln vorstellen, die eingehalten werden sollten, um die erläuterten Probleme zu vermeiden. Wir betrachten im Folgenden die ersten drei "Normalformen".

Inkonsistenzen und Anomalien

Inkonsistenzen/Anomalien

Sind widersprüchliche Daten oder logische Brüche innerhalb einer Datenbank. Sie treten sehr oft auf, wenn man zum Beispiel Änderungen an den Inhalten vornimmt und Beziehnungen zwischen Informationen nicht berücksichtigt.

Normalisierung

Unter der "**Normalisierung einer Datenbank**" kann man sich eine Sammlung von Regeln vorstellen, die eingehalten werden sollten, um die erläuterten Probleme zu vermeiden. Wir betrachten im Folgenden die ersten drei "Normalformen".