Datenanomalien und andere Datenbankprobleme

Die durch widersprüchliche Daten auftretenden Fehler nennt man Anomalien. Sie entstehen, wenn dieselbe Information mehrfach, also redundant, gespeichert wird und es somit zu nicht regelkonformen Datenstrukturen kommt.

Redundanz

Sie ist vorhanden, wenn man eine Informationseinheit ohne Informationsverlust weglassen kann. Das wäre in unserem Modell der RC Wildbach Datenbank der Fall, wenn beispielsweise die Information über die Anzahl der vorhandenen Sitze eines Bootes an mehr als einer Stelle abgelegt wären.

Dateninkonsistenz

Es liegen widersprüchliche Daten vor. Etwa, wenn für ein und dasselbe Boot unterschiedliche Sitzplätze gespeichert wären.

Änderungsanomalie

Sie tritt auf, wenn nicht alle redundanten Einträge zugleich korrigiert werden.

Einfügeanomalie

Ein neuer Datensatz kann nicht problemlos in eine Tabelle eingetragen werden. Das wäre beispielsweise der Fall, wenn man ein neues Boot nicht hinzufügen könnte, ohne dass es gleich gebucht werden muss.

Löschanomalie

Sie entsteht, wenn durch das Entfernen eines Datensatzes mehr Informationen als gewünscht verloren gingen. Falls beispielsweise ein Kunde aus der Datenbank gelöscht werden möchte, ginge mit dieser Löschung die Informationen über die Boote verloren.

Die oben beschriebenen Datenbankprobleme sind die Folge eines schlechten Datenbankdesigns. Unser Modell ist jedoch frei von Anomalien.

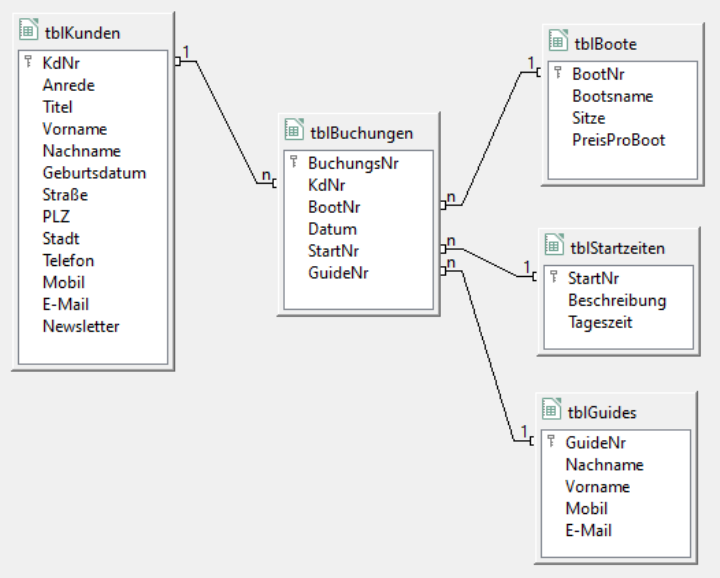

Abb. 01: Relationales Modell zurDatenbank RC Wildbach

Zusammenfassung - Vermeidung von Datenbankanomalien

Will man Anomalien beim Erstellen einer Datenbank vermeiden, so hat sich nach dem Erstellen eines ersten Entwurfes folgende Vorgehensweise als sehr vorteilhaft erwiesen.

1.	**Zerlege (atomarisiere) Felder mit zusammengesetzten Inhalten in ihre einzelnen Bestandteile** Beispiel: Ein Feld *Adresse* lässt sich in die Felder *Straße, Hausnummer, Postleitzahl, Stadt* und *Land* zerlegen.

2.	**Definiere ein Schlüsselfeld für jede Tabelle** Jeder Datensatz muss eindeutig identifizierbar sein. Entweder ist dazu ein neues Feld *ID* anzulegen oder man verwendet einen zusammengesetzten Schlüssel aus bereits vorhandenen Datenfeldern.

3.	**Beseitige Gruppen mit sich wiederholenden Feldern** Beispiel: Wenn eine der Tabellen die Felder *Hobby1, Hobby2* und vielleicht auch noch *Hobby3* enthält, dann sind diese Datenfelder zu löschen und eine neue Tabelle mit einem Datenfeld *Hobby* und einem Schlüsselfeld zu erzeugen.

4.	**Jedes Nicht-Schlüsselfeld einer Tabelle muss vom Gesamtschlüssel abhängen** Ist das nicht der Fall, so legt man eine neue Tabelle für das betreffende Feld an und benutzt dann die Schlüsselfelder, um beide Tabellen miteinander zu verbinden.

5.	**Alle Datenfelder sollten ausschließlich von den Schlüsselfeldern abhängen** Sie sollten nicht voneinander abhängig sein. Beispiel: Enthalten Felder Daten, die aus anderen Feldern bestimmt oder berechnet werden können, wie sich etwa die *Kursdauer* aus *Kursbeginn* und *Kursende* ergeben würde? Falls ja, dann solltest du das Attribut *Kursdauer* nicht verwenden. (In unserem Beispiel mussten wir uns um dieses Problem nicht kümmern.)

Normalisierung der Datenbank

Die wissenschaftliche Methode, einen Modellentwurf von Datenanomalien zu befreien, nennt man **Normalisierung**. Die oben aufgelisteten fünf Schritte zur Vermeidung von Datenanomalien sind die wichtigsten Kriterien der ersten **drei Normalformen (NF)** von **E. F. Codd**, dem Erfinder des relationalen Datenbankmodells. Die Normalformen bauen aufeinander auf. Tabellen in der dritten Normalform sind weitgehend frei von Redundanzen.

Erste Normalform (1NF)

Eine Relation (Tabelle) ist in der ersten Normalform, wenn sie **frei von Wiederholungsgruppen** ist **und** alle **Datenfelder** mit zusammengesetzten Inhalten in ihre einzelnen Bestandteile zerlegt (**atomisiert**) sind.

Zweite Normalform (2NF)

Eine Relation ist in der **zweiten Normalform**, wenn die erste Normalform vorliegt **und** alle Nichtschlüsselfelder einer Tabelle von **jedem** Schlüsselfeld **voll funktional abhängig** sind.

Dritte Normalform (3NF)

Eine Relation ist in der **dritten Normalform**, wenn die zweite Normalform vorliegt **und** alle Datenfelder, die **nicht** Teil des Schlüssels sind, untereinander in **keiner Abhängigkeit**[1] stehen.

Unser Datenmodell für die RC Wildbach Datenbank (Abb. 01) liegt in normalisierter Form vor, und ist somit weitgehend frei von Redundanzen.

Funktionale Abhängigkeit von Attributen bedeutet, dass sich die Werte eines abhängigen Datenfeldes ändern, wenn sich Attributwerte eines anderen Feldes ändern.

Beispiel in Kurzschreibweise: *tblMeineCDs.Interpret* → *tblMeineCDs.CD-ID*

Ändert man den Interpret, dann muss man auch die CD-ID in dem Datensatz anpassen, weil ein anderer Interpret ja auch eine andere CD gemacht haben muss.

Mathematisch exakter unterscheidet man:

1. **Funktionale Abhängigkeit**: Ein Attribut A ist von einem Attribut B funktional abhängig, wenn zu jedem Wert von B eindeutig der Wert von A bestimmt werden kann.
2. **Volle funktionale Abhängigkeit**: Von voller funktionaler Abhängigkeit spricht man, wenn ein Attribut A von einer Attributkombination B komplett funktional abhängig ist, und nicht nur von einem Teil dieser Attributkombination.
3. **Transitive Abhängigkeit**: Transitive Abhängigkeit zwischen zwei Attributen A und C liegt vor, wenn ein Attribut A von einem Attribut B eindeutig bestimmt wird, das Attribut B aber wiederum von einem Attribut C eindeutig bestimmt wird.

[1] Oder anders gesagt: … jedes Nichtschlüsselattribut von keinem Schlüsselfeld transitiv (also „um die Ecke“ über ein anderes Nichtschlüsselfeld) abhängt.