Pythia lüftet ihr Geheimnis

Der Entwurf einer Datenstruktur im relationalen Datenbankmodell
Wolfgang Korte

I. Datenbankmodelle

Relationales, hierarchisches und Codasyl-Datenbankmodell

Eine Datenbank ist eine sinnvoll organisierte Informationssammlung, die dem Benutzer auf Grund ihrer Organisation einen schnellen und möglichst unkomplizierten Zugriff auf die von ihm gewünschten Informationen bietet.
Man hat unterschiedliche Organisationsformen gefunden: z.B. das relationale Datenbankmodell, das Codasyldatenbankmodell und das hierarchische Modell.
Auf PC's wird am häufigsten das relationale Datenbankmodell angewandt, das unter dem Namen dBASE (Ashton-Tate, Inc.) wohl einen seiner bekanntesten Vertreter gefunden hat.

II. Das relationale Datenbankmodell

Qualitätsentscheidend: Das Datenbank-Design

Bevor Sie eine relationale Datenbank aufbauen können, müssen Sie die Struktur der Daten festlegen (Datenbankdesign). Die Struktur bestimmt entscheidend Qualität und Schnelligkeit eines Systems und ist deshalb von großer Bedeutung.

Relation und Tabelle

Eine relationale Datenbank besteht aus einer oder mehreren Tabellen (Relationen), die Sie sich wie ein Stundenplanschema vorstellen dürfen. Jede Zeile einer Tabelle bildet einen Datensatz. Die Spalten der Tabelle nennt man Felder. Ein Feld, dessen Inhalt einen Datensatz eindeutig, d.h. unterscheidbar von allen anderen Eintragungen in der Tabelle, bestimmen läßt, nennt man Schlüssel (Key). Ein Schlüssel kann aus einem Feld, das zu diesem Zweck ausdrücklich zugewiesen worden ist (Primärschlüssel), oder mehreren Feldern bestehen (zusammengesetzter Schlüssel). Alle Zeilen (Datensätze) einer Tabelle müssen sich unterscheiden. Die Reihenfolge der Zeilen ist ohne Bedeutung. Lassen Sie mich dieses am Beispiel des in Abb. 1 dargestellten einfachen Telefonregisters, bestehend aus Namen, Ort, Straße und Telefonnummer, veranschaulichen:

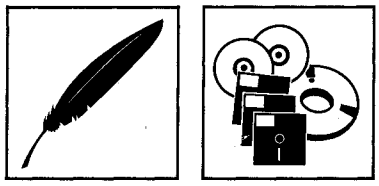
Feld = Tabellenspalte

Datensatz = Tabellenzeile

Schlüssel	Feld	Feld	Feld	Feld
Nr.	Name	Ort	Strasse	Telefon
1	Meyer	Berlin	Lessingstr.	1234567
2	Müller	Hanburg	Schillerstr.	7654321
3	Schulze	München	Goethestr.	4567891

Abb. 1: Einfaches Telefonregister

Wolfgang Korte ist Richter am LG Bielefeld.



Die „offene" Architektur einer Tabelle

Während die Tabelle durch Anzahl und Breite der Felder in ihrer horizontalen Ausdehnung festgelegt ist, ist sie nach unten hin offen, d.h., es können Datensätze hinzugefügt oder gelöscht werden. Zugriff auf einen Datensatz kann man über den Inhalt eines beliebigen Feldes nehmen, indem man das System z.B. bittet, in dem Feld „Name" nach „Meyer" zu suchen und für den Fall, daß es den Eintrag „Meyer" gefunden hat, den kompletten Datensatz bestehend aus Name, Anschrift und Telefonnummer, anzuzeigen. Interessant ist dabei, daß die Felder in einer Tabelle ein Eigenleben führen können, indem die Einträge eines Feldes nach bestimmten Kriterien (z.B. alphabetisch in aufsteigender oder absteigender Reihenfolge) sortiert werden.

Indexdateien für Feld-Sortierungen

Hierbei wird nicht die gesamte Tabelle entsprechend sortiert, sondern nur der Inhalt des Feldes. Das Ergebnis kann in eine neue Datei (Indexdatei) geschrieben werden. In dieser Indexdatei findet sich ein Hinweis auf die entsprechende Datensatznummer der Tabelle. Wenn Sie in Ihrem Telefonverzeichnis die Nummer des Herrn Meyer suchen, könnte das System also zunächst in der Indexdatei nach dem Namen Meyer suchen, dort einen Hinweis auf die Datensatznummer der Tabelle finden und sodann anhand der Datensatznummer den gesamten Datensatz aus der Tabelle anzeigen. Eine Suche in einer sortierten Liste (Indexdatei) hat im übrigen den Vorteil, daß sie erheblich schneller durchgeführt werden kann, als es bei unsortierten Feldinhalten der Fall wäre.

Es ist selbstverständlich auch möglich, dem jeweiligen Benutzer nur eine bestimmte Sicht (View) auf die Datenbank zu ermöglichen. So dürfte in einem großen Betrieb die Personalbuchhaltung andere Daten (z.B. Steuerklasse, Anzahl der Kinder) interessieren als es z.B. bei dem Organisationsleiter der Fall sein könnte (z.B. Ausbildungsstand, Fähigkeiten).

Das Schlüsselfeld: Eindeutige Beziehungen knüpfen

Das Feld „Nummer" ist das Schlüsselfeld, durch das sich jeder Datensatz eindeutig bestimmen läßt (Meyers kann es ja mehrere geben). Selbstverständlich muß die Nummer dem Benutzer der Datenbank nicht offenbart werden – wie er überhaupt nicht merken muß, daß die Informationen in Tabellenform abgespeichert sind. Das ist lediglich eine Frage der internen Organisation, die mit der Darstellung nach außen hin nichts zu tun haben muß.

Über ein Schlüsselfeld kann eine Beziehung zu einer weiteren Tabelle hergestellt werden. So könnte die einfache Telefondatei eine Kundendatei werden, indem in einer zweiten Tabelle in einem Feld offene Rechnungen und in einem weiteren Feld die „Kundennummer" abgespeichert sind. Über die Kundennummer kann so festgestellt werden, ob Herr Meyer noch eine Rechnung offen hat.

III. Die Datenstruktur des „Orakel von Bielefeld"

Wenn die unter II. gewonnenen Erkenntnisse auf eine Rechtsprechungs-Datenbank zur Verwaltung von Texten übertragen werden sollen, muß man sich zunächst Gedanken darüber machen, welche Informationen denn benötigt werden. Dies sind neben den reinen Texten das Entscheidungsdatum, das Gericht (Verfasser), das Aktenzeichen (Fundstelle) und eine unbekannte Anzahl von Suchbegriffen, anhand derer der Text später wiedergefunden werden kann.

Eine Datenbank für Texte könnte danach so aufgebaut sein (Abb. 2):

Nr.	Gericht	Datun	Aktenz	Text	Suchwort1	Suchwort2	Suchwort3
1	LG Bielef	1.1.88	1 S 1/88	Text1	Kauf	BGB § 462	Handlung
2	LG Bielef	1.1.89	1 S 2/89	Text2	Kauf	BGB § 459	
3	AG Bielef	1.1.90	1 C 1/90	Text3	Vertrag	Anfechtung	

Abb. 2: Erster Strukturentwurf – nicht normalisiert

Wenn Sie sich diese nicht „normalisierte" Tabelle anschauen, werden Sie schnell mehrere Nachteile feststellen:
(1) Die Tabelle ist umständlich auszuwerten. Das Auffinden eines Suchbegriffs dauert zu lange, da im Zweifel drei Felder (Suchbegriff 1, Suchbegriff 2 und Suchbegriff 3) durchsucht werden müssen.
(2) Pro Text können nur drei Suchbegriffe abgespeichert werden.
(3) Werden weniger als drei Suchbegriffe einem Text zugeordnet, wird Platz verschwendet. Deshalb müssen die Daten streng strukturiert („normalisiert") werden.

E.F. Codd: Die Lehre von der Normalform der Daten

E.F. Codd, Begründer des relationalen Datenbankmodells (Codd veröffentlichte 1970 einen Aufsatz: „A relational model of data for large shared data banks"), entwickelte die Lehre von der Normalform. Bei der Normalisierung der Daten sind drei **R**egeln (1., 2. und 3. Normalform) zu beachten.

Erste Normalform

Regel 1

Gruppen und mehrfach vorkommende Felder innerhalb einer Tabelle sind verboten. Alle Sätze müssen dieselbe Anzahl von Feldern haben.
In der Tabelle (Abb. 2) haben die Felder Suchwort 1 bis Suchwort 3 zwar unterschiedliche Namen, die Inhalte sind aber jeweils gleich (Suchbegriffe). Das ist verboten. Um zu erreichen, daß nicht mehrere Felder Suchbegriffe zum Inhalt haben, werden die Suchbegriffe für einen Text (Datensatz) nicht mehr nebeneinander, sondern untereinander geschrieben. Dies zeigt Abb. 3 (wobei das Feld „Gericht" auch als Feld zur Aufnahme eines normalen Suchbegriffs behandelt worden ist):

Nr.	Datum	Aktenz	Text	Suchwort
1	1.1.88	1 S 1/88	Text1	Kauf
1	1.1.88	1 S 1/88	Text1	BGB § 462
1	1.1.88	1 S 1/88	Text1	Handlung
1	1.1.88	1 S 1/88	Text1	LG Bielefeld
2	1.1.89	1 S 1/89	Text2	Kauf
2	1.1.89	1 S 1/89	Text2	BGB § 459
2	1.1.89	1 S 1/89	Text2	LG Bielefeld

Aus Platzgründen ist diese Tabelle nicht vollständig!

Abb. 3: Erste Normalform

Die Daten befinden sich in der ersten Normalform. Eben genannte Schwächen sind beseitigt. Suchbegriffe befinden sich nur noch in einem Feld. Da die Tabelle nach unten hin offen ist, können jedem Text beliebig viele Suchbegriffe zugeordnet werden.

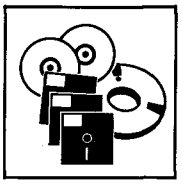
Datenredundanz

Leider sind nunmehr in den Feldern „Nr.", „Aktenzeichen", „Datum" und Text einige Einträge mehrfach, nämlich so oft, wie Suchbegriffe zugeordnet worden sind, vorhanden. Man spricht von Datenredundanz. Sie bedeutet nicht nur eine Verschwendung von Speicherplatz, sondern kann auch den Programmablauf erheblich verlangsamen (Verarbeitungsredundanz). Hier hilft die **R**egel 2.

Regel 2

Zweite Normalform

Daten, die vom Gesamtschlüssel nicht funktionell abhängig sind, müssen entfernt und in einer eigenen Tabelle gespeichert werden zusammen mit dem Teilschlüssel, von dem sie abhängig sind.



Während in Abb. 2 durch das Feld „Nr." noch jeder Datensatz eindeutig bestimmt werden konnte (Schlüsselfeld), ist dies in Abb. 3 nicht mehr der Fall, weil sich dort dieselben Einträge wiederholen. Zur eindeutigen Bestimmung eines Datensatzes muß neben dem Feld „Nr." noch ein anderes (Suchwort) herangezogen werden. Wie bereits erwähnt, spricht man dann von einem Gesamtschlüssel. Datum, Aktenzeichen und Text sind aber nur von dem Feld „Nr.", nicht aber auch von dem Feld „Suchwort" abhängig. Diese Felder müssen also entfernt und in einer neuen Tabelle vereinigt werden (Abb. 4): Um Ihnen die dritte Regel besser veranschaulichen zu können, habe ich in der Tabelle der Suchbegriffe noch das Feld „Wortcode" angefügt, das einen für jedes Wort nur einmal zu vergebenden Zahlencode enthält.
Während die Texttabelle nun gelungen erscheint, treten bei der Worttabelle noch Redundanzen auf. Wenn das Wort „Vertrag" 1000 Texten zuzuordnen ist, wird es so auch 1000 Mal in der Tabelle „Suchwort" auftauchen, was überflüssig ist.

Die Dritte Normalform

Regel 3

Zwischen Nicht-Schlüsselfeldern darf keine Abhängigkeit bestehen, andernfalls sind diese in getrennten Tabellen zu speichern.
Auffällig ist in Abb. 4, daß der Wortcode nur vom Suchwort, nicht aber von der Text-Nr. abhängig ist. Deshalb sind auch diese beiden Felder (Suchwort und Wortcode) in einer gesonderten Tabelle zu speichern. Nun werden Sie mir zu Recht entgegenhalten, daß die Regel 3 auf diesen Fall deshalb nicht völlig zutrifft, weil zumindest eines der beiden auszusondernden Felder (Suchwort, Wortcode) Teil des Gesamtschlüssels (Nummer, Suchwort oder Nummer, Wortcode) ist. Genaugenommen habe ich die Regel 2 nämlich ein weiteres Mal angewandt, um die letzten Redundanzen zu beseitigen. Die drei Regeln besagen aber gerade nicht, daß die Datennormalisierung in drei Schritten zu erfolgen hat, sondern lediglich, daß alle drei Regeln erfüllt sein müssen. Das kann bereits nach einmaliger Umstellung der Tabelle der Fall sein.
Die Datenstruktur stellt sich nun wie in Abb. 5 dar:

Texttabelle

Nr.	Datum	Aktenzeichen	Text
1	1.1.88	1 S 1/88	Text1
2	1.1.89	1 S 1/85	Text2
3	1.1.96	1 C 1/96	Text3

Worttabelle

Nr.	Suchwort	Wortcode
1	Kauf	1
1	BGB § 462	2
1	Wandlung	3
1	LG Bielefeld	4
2	Kauf	1
2	BGB § 459	5
2	LG Bielefeld	4

Aus Platzgründen ist diese Tabelle nicht vollständig!

Abb. 4: Zweite Normalform

Worttabelle

Suchwort	Wortcode	Anzahl
Kauf	1	2
BGB § 462	2	1
Wandlung	3	1
LG Bielefeld	4	2
BGB § 459	5	1
Vertrag	6	1
Anfechtung	7	1

Wortzuordnungstabelle

Nr.	Worttode
1	1
1	2
1	3
1	4
2	1
2	5
2	4
3	6

Aus Platzgründen ist diese Tabelle unvollständig!

Abb. 5: Dritte Normalform

Optimierung der Suchzeit

In der Worttabelle ist noch ein weiteres Feld (Anzahl = Anzahl der Texte, denen das betreffende Wort zugeordnet ist) eingefügt worden. Das erleichtert den späteren Programmablauf. Die in Abb. 5 dargelegte Datenstruktur erfüllt alle drei Regeln der Codd'schen Normalformlehre: Datenredundanzen und Mehrfachfelder sind vermieden worden; jeder Datensatz ist durch einen eindeutigen Schlüssel bestimmbar. Die Anzahl der einem Text zuzuordnenden Suchbegriffe ist theoretisch unbegrenzt. Gerade der dritte Schritt (Aufteilung der Tabelle „Wort" in die Tabellen „Wort" und „Wortzuordnung") ist für einen schnellen Programmablauf extrem wichtig. Bei der logischen Verknüpfung größerer Textmengen miteinander müssen Vergleiche durchgeführt werden,

weshalb das System in der Tabelle „Wortzuordnung“ so viele Zeilen lesen muß, wie Texte einem Begriff zugeordnet worden sind. Muß nun das System anstelle eines kurzen Codes ein langes Wort lesen (das evtl. mehrere 100 Mal), wie es beim in Abb. 4 dargestellten Aufbau der Fall wäre, so dauert dies entsprechend länger.
Zuletzt müssen die Felder einer Tabelle noch nach Datentyp (und Länge) festgelegt werden. Folgende Daten

Datentypen

typen werden unterschieden:

```
a (alphanumerisch) = beliebige Zeichenkette
n                  = numerisch
memo               = Code zur (Lang-)Textspeicherung
Datum              = Datumsfeld
logisch            = „true/wahr“ oder „false/falsch“
```

Das ergibt das in Abb. 6 dargestellte Schema:

Tabelle	Feld / Type	Feld / Type	Feld / Type	Feld / Type
Text	Nr. a	Datum date	Aktenz a	Text memo
Wort	Suchwo. a	Wartcode a	Anzahl n	
Wortzuordnu.	Nr. a	Wortcode a		

Abb. 6: Felder und Datentypen

Obwohl die Feldinhalte „Nr.“ und „Wort-Code“ eigentlich nur aus Zahlen bestehen, ist es günstiger, diese mit Daten vom Typ „alphanumerisch“ als mit dem Typ „numerisch“ zu füllen. Zeichenketten sind innerhalb eines Programms nämlich flexibler und leichter zu manipulieren als Zahlen.
Nunmehr ist die Hauptarbeit für die Einrichtung einer Datenbank getan. Der Ablauf eines etwa zu erstellenden Programms folgt zwanglos der vorgegebenen Datenstruktur.