Die Anwendung von Index-Dateien

Das private Adreßbuch, die gesamten Bestellungen bei einer Großfirma und das öffentliche Telefonbuch sind Sammlungen von Datensätzen, das heißt, es sind Datenbanken. Obwohl die Inhalte dieser Datenbanken unterschiedlich sind, haben alle drei eines gemeinsam: man will auf Basis eines Teils der Daten passende Datensätze finden.

Die Aufgabe »Suchen aus einer Datenbank« begegnet einem auch außerhalb der EDV. Denken Sie an die Plattensammlung zuhause, die eventuell einige Hunderte Schallplatten umfaßt. Hier gilt es, jene Schallplatte, die man in diesem Moment hören will, umgehend zu finden. Dazu gibt es verschiedene Ansätze. Nicht jeder Mensch ordnet seine Schallplatten. Sucht er aber eine bestimmte Platte, bleibt ihm dann nichts anderes übrig, als die ganze Sammlung zu durchsuchen, bis die gewünschte Platte gefunden ist.

Wird, die Sammlung noch größer, so ist diese Vorgehensweise in jedem Fall zu langsam. Man hat also zuerst Ordnung in die Sammlung zu bringen. Dabei ist es unwichtig, in welcher Reihenfolge die Platten geordnet werden, solange jede Platte genau eine »richtige« Stelle in der Sammlung hat. So kann man Schallplatten nach Titel, Interpret, Art der Musik oder Plattennummer sortieren. Der Nachteil ist natürlich, daß es einige Zeit dauert, Hunderte von Platten zu sortieren. Ist die Sammlung aber einmal fertig sortiert, so fügt man anschließend jede neue Platte stets am Ende der Sammlung ein, was sehr schnell geht. Gelegentlich muß dann die Sammlung neu sortiert werden, das kommt aber relativ selten vor. Selbstverständlich ist das Suchen nach einer Platte sehr schnell geworden: Man kann eine beliebige Platte der Sammlung ansehen und auf Basis der festgelegten Reihenfolge feststellen, ob eine gesuchte Platte nun links oder rechts von der aktuellen Position in der Sammlung liegt. Wählt man die Suchposition sinnvoll, so kann die Suche in wenigen Schritten abgeschlossen sein.

Die Qualität eines Dateiverwaltungsmusters ist an drei Kriterien zu messen

Es könnte aber sein, daß man nicht Hunderte sondern Tausende von Platten besitzt, wenn erstmals die Notwendigkeit einer Ordnung ansteht. In diesem Fall ist es durchaus möglich, daß der Suchaufwand viel zu groß erscheint. Man setzt hier ein anderes Verfahren ein. Statt die Platten selbst zu ordnen, fertigt man eine Liste von allen Platten einer Sammlung an, jeweils mit der Stelle im Regal, wo sich die Platte befindet. Diese Liste wird dann sortiert, wie bei einer kleineren Sammlung die Platten selbst sortiert wurden. Der Vorteil liegt auf der Hand: die Liste ist kleiner und handlicher und daher schneller in Ordnung gebracht. Sucht man jetzt eine Platte, wird die Liste durchsucht (ein sehr schnelles Verfahren) und anschließend kann die Platte direkt aus dem Regal geholt werden.

Am Beispiel der Plattensammlung haben Sie nun drei verschiedene Verfahren, Daten ausfindig zu machen, kennengelernt. Sie sollen jetzt aus der Sicht des Programmierers anhand dreier Kriterien verglichen werden.

1. Wie schwierig ist das Verfahren als Programm zu verwirklichen?
2. Wie viel Zeit muß man in die Pflege der Ordnung investieren?
3. Wie schnell ist (im Durchschnitt) ein gesuchter Datensatz gefunden?

Die Ergebnisse sind in der Tabelle und in den Bildern zusammengefaßt, falls Sie sich einen schnellen Überblick verschaffen möchten. Das erste Verfahren heißt einfach »unsortiert«. Es ist sehr einfach realisiert, da es nämlich nichts zu tun gibt. Daten werden der Reihe nach gespeichert, ohne Vergleiche und ohne Ordnung.

In die Pflege der Ordnung wird ebenfalls keine Zeit investiert, weil keine vorhanden ist. Das Suchen dauert dafür relativ lange, genau gesagt ist die Suchzeit proportional zur Anzahl Datensätze. Das werden Sie einsehen, wenn Sie überlegen, wie viele Datensätze angesehen werden müssen, bevor Sie den gesuchten finden. Wenn es n Datensätze gibt, kann der gesuchte der erste, der zweite, der dritte und so weiter bis zum n-ten sein. Im Durchschnitt wird ein Datensatz an der Position

$$(1+2+3+4+...+n)/n$$

stehen. Diese Formel läßt sich auf

$$n\cdot(n+1)/2n = (n+1)/2$$

reduzieren. Die Suchzeit ist daher linear mit n.

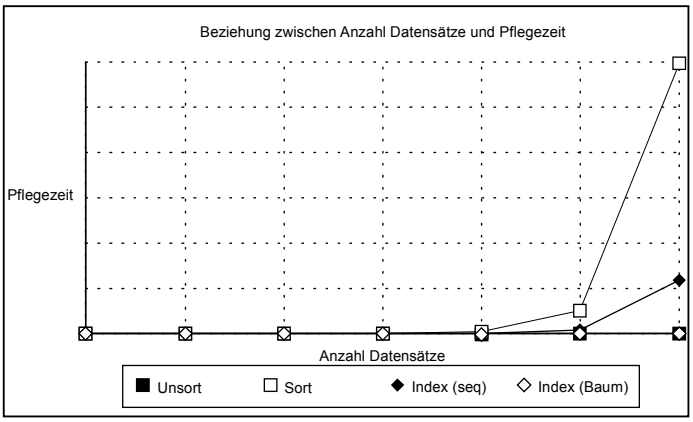

Bild 1: Die Relation zwischen Pflegezeit und der Anzahl der Datensätze für die verschiedenen Verfahren

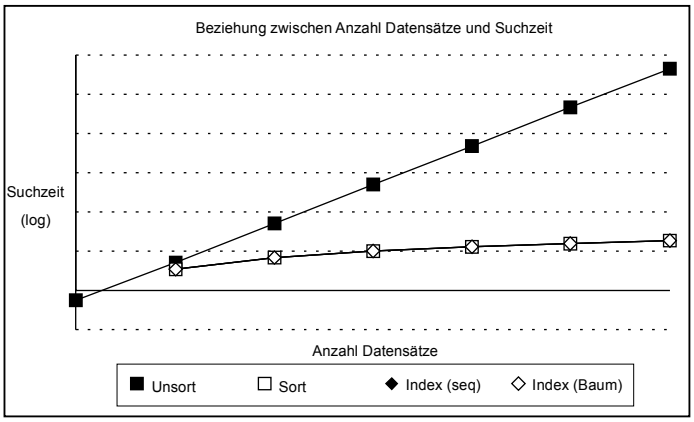

Bild 2: Die Relation zwischen Suchzeit und der Anzahl der Datensätze für die verschiedenen Verfahren

Das zweite Verfahren heißt »sortiert«. Es ist nicht besonders schwer zu verwirklichen, weil die Algorithmen seit langer Zeit bekannt sind und mehrmals veröffentlicht wurden. (Viele PC-Compiler haben inzwischen Sortierroutinen in ihren Bibliotheken.)

Die Pflege der Ordnung erfordert aber einige Zeit, genau gesagt, ist das beste Ergebnis einer Sortierroutine proportional zu $n\cdot\log(n)$, wenn n die Anzahl der Datensätze ist. Sortieren bringt aber wesentliche Vorteile, wenn ein Datensatz gesucht wird. Ein binäres Suchverfahren erfordert eine Zeit, die proportional zu $\log_2(n)$ ist. Wie Bild 2 zeigt, hat dieses Verfahren einen großen Vorteil gegenüber dem unsortierten, und der Vorteil wächst mit dem Datenbestand. Das dritte Verfahren benutzt eine Liste, die sortiert wird. Mit Datensätzen wird dieses Verfahren »indiziert« genannt, weil ein Index (die Liste) angefertigt wird, die wiederum auf den gesuchten Datensatz zeigt. Der Index wird wie zuvor die Datensätze sortiert, dieser Vorgang ist daher nicht schwieriger zu realisieren als das Sortieren von Daten. Das Verfahren insgesamt ist etwas schwieriger als nur Daten zu sortieren, weil hier zwei Dateien und die Verbindungen zwischen den beiden verwaltet werden müssen. Bei der Plattensammlung war der Index von Vorteil, weil es weniger umständlich und schneller ist, die Liste zu sortieren als die Platten. Dies ist bei Datensätzen ebenfalls der Fall. Der Grund dafür ist, daß die Länge eines Indexeintrags meistens nur ein Bruchteil der Länge eines Datensatzes beträgt. Da die Dauer der Vorgänge »Vergleichen«, »Lesen« und »Schreiben« proportional zur Länge der Daten sind, wird Zeit gespart, weil kürzere Datenfelder bearbeitet werden. Alle anderen Faktoren, die die Dauer eines Indexvorgangs bestimmen, sind die gleichen wie beim Sortieren. Die Ersparnisse bei der einfachen Indizierung gegenüber der Sortierung ergeben sich einwandfrei aus der reduzierten Länge der Daten, die bearbeitet werden müssen. Eine weitere Verfeinerung des Index-Verfahrens kann durch Reduzierung der Pflegezeit erzielt werden. In dem gerade beschriebenen Verfahren wird der Index einfach sortiert; dies ist erforderlich, sobald neue Datensätze eingefügt werden. Ein

Verfahren	Pflegezeit proportional zu	Suchzeit proportional zu
unsortiert	0	$g\cdot n/2$
sortiert	$g\cdot(n\cdot\log(n))$	$g\cdot(\log_2(n))$
Index (sequentiell)	$i\cdot(n\cdot\log(n))$	$i\cdot(\log_2(n))$
Index (Baum)	0	$i\cdot(\log_2(n))$

n: Anzahl Datensätze
g: Größe eines Datensatzes
i: Größe eines Indexfelds ($i \le g$)

Die Relationen zwischen der Anzahl der Datensätze und Pflege- und Suchzeiten für verschiedene Verfahren

besseres Verfahren würde zwei Merkmale mit sich bringen. Erstens soll der Index stets aktualisiert werden, um die langwierige Sortierung zu vermeiden. Zweitens soll diese Aktualisierung des Index möglichst schnell erledigt werden, um die Pflegezeit insgesamt zu reduzieren. Das heißt, die laufende Aktualisierung des Index soll viel weniger Zeit in Anspruch nehmen als die Sortierung. Ein entsprechendes Verfahren ist mit Programmen wie dBase bekannt geworden. Hier wird die Erweiterung des Index beschleunigt und die Sortierung des Index erübrigt sich. Das verwirklicht man mit einer sehr speziellen Struktur des Index, die allgemein als »Baum« bekannt ist.

Statt der Datei mit ihren Datensätzen sortiert man eine Liste mit Zeigern (Indizes)

Diese Struktur ermöglicht das sehr schnelle Einfügen und Löschen von Einträgen und gewährt gleichzeitig die kurzen Suchzeiten des einfachen Index-Verfahrens. Damit ist eine optimale Lösung erreicht worden, die aber natürlich ihren Preis hat: Das Verfahren ist relativ schwierig zu programmieren. In Bild 3 sehen Sie einen Index-Baum grafisch abgebildet. Jeder Eintrag im Index enthält die Daten, auf die sich die Indizierung bezieht und einen Zeiger auf den Datensatz, der zu diesem Indexeintrag gehört. Um die Baumstruktur zu realisieren, werden weitere Zeiger benutzt: Ein Zeiger ermöglicht es, »nach oben« im Baum zu klettern während die anderen beiden »nach unten« zeigen. Die beiden nach unten gerichteten Zeiger sind nicht zu verwechseln: Einer zeigt »nach links«, das heißt, in die Richtung der Indexeinträge, die vor dem aktuellen stehen. Der andere Zeiger zeigt »nach rechts«, auf Datensätze, die nach dem aktuellen stehen. Der Trick bei dieser Struktur ist, daß der Baum durch Änderungen an den Zeigern aktualisiert wird. Die Indexeinträge brauchen nicht mehr sortiert werden, da eine Suche durch den Index über die Zeiger erfolgt. Wird ein neuer Datensatz dazugefügt, so sucht man die Stelle, an der dieser Datensatz in der Index-Reihenfolge stehen soll. Anschließend werden lediglich die Zeiger geändert, um den neuen Indexeintrag an dieser Stelle »einzuhängen«.

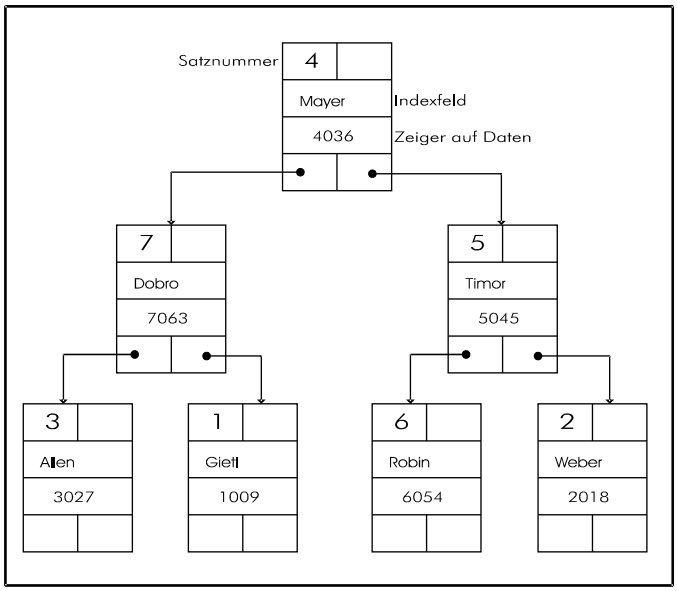

Bild 3: Ein Index in der Form eines Baums

Tatsächlich müssen unter Umständen fast alle Zeiger im Baum nach komplexen Regeln geändert werden, da der Baum in einer speziellen Form gehalten werden muß, um die Suchzeiten zu optimieren. Die meisten Datenbanken verzichten jedoch auf diesen komplizierten Umbau. Statt dessen wird empfohlen, den Index bei Gelegenheit komplett neu zu generieren. Dies geschah nicht aus Trägheit seitens der Entwickler, sondern aus ganz kühner Kalkulation heraus: Warum soll der Umfang der Index-Prozeduren verdoppelt werden, um eine relativ kleine und vor allem nur vorübergehende Verbesserung in den Suchzeiten zu erreichen?

In diesem kurzen Überblick haben Sie nun den Sinn und Zweck der Indizierung kennengelernt. Ein Punkt ist besonders wichtig, wird aber sehr oft vergessen oder übersehen: Indizieren ist nichts anderes als ein Ersatz für Sortieren, der unter manchen Umständen sehr vorteilhaft ist. Die Vorteile resultieren jedoch nicht aus irgendwelchen qualitativen Unterschieden zwischen Indizieren und Sortieren. Vielmehr ist der Vorteil rein praktischer Natur: Indizieren ist in der Regel schneller.

(Thomas Little/db)

aus DOS International 11/90 – Seite 224-226