Editorial

Liebe Leserin, lieber Leser,
nach einer gut gefüllten Mai-Ausgabe fallen wir nun ein wenig in ein Sommerloch. Trotzdem viel Spaß beim Lesen!

Andrea Herrmann

Glossar: Baum (Datenstruktur)

Andrea Herrmann

Bäume kommen in der Natur oft vor: Aus einem Samen wächst ein kleiner Spross senkrecht nach oben, dann zweigen von ihm Äste ab, die sich wiederum verzweigen und so weiter. Auch ein Stammbaum wächst: Die Stammmutter Eva hat eine oder mehrere Töchter, diese ebenfalls und so weiter über viele Generationen hinweg... Da in Software-Systemen meist real existierende Dinge und Zusammenhänge verwaltet werden, sollte es nicht verwundern, dass auch in der Informatik baumförmige Datenstrukturen vorkommen.

Abstrakt betrachtet ist ein Baum ein spezieller Graph. Ein Graph besteht aus Knoten und Kanten, wobei die Kanten jeweils zwei Knoten miteinander verbinden. Die Knoten sind konkrete Datenobjekte, beispielsweise die Mutter und ihre Töchter. Ein Knoten hat mindestens ein Schlüsselattribut, nach dem gesucht oder sortiert werden könnte, und zusätzlich beliebig viele Nutzdaten. Die Kante beschreibt im Stammbaum eine „Ist-Tochter-von“-Beziehung und verbindet zwei Knoten miteinander, also zwei Menschen.

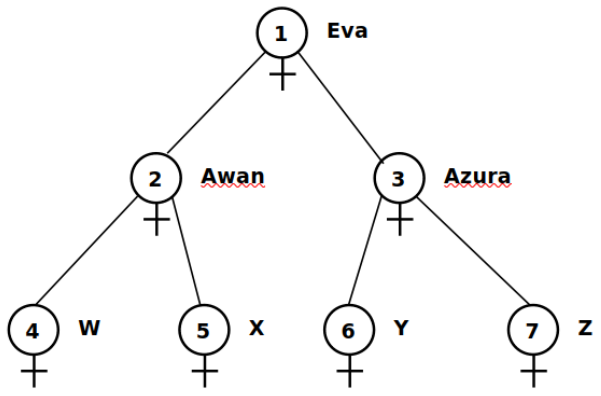

Abbildung 1: Beispiel-Stammbaum

Ein Baum ist ein Graph, der hierarchische Beziehungen darstellt. Es gibt einen einzigen Wurzelknoten (sagen wir mit Namen Eva) ohne Mutterknoten. Alle anderen Knoten sind mit genau einem Mutterknoten verbunden und möglicherweise mit einem oder mehreren Tochterknoten. Knoten ohne Tochterknoten heißen auch Blätter, in Analogie zum biologischen Baum. Wegen seiner hierarchischen Struktur gibt es im Baum genau einen einzigen Pfad von jedem Knoten zum Wurzelknoten und umgekehrt. Ein Baum mit n Knoten enthält genau $n-1$ Kanten. In einem Graphen, wo jeder Knoten mit jedem anderen verbunden sein könnte, gibt es bis zu $n \cdot (n-1) / 2$ Kanten.

Man unterscheidet einige Spezialformen von Bäumen:

- Im Binärbaum hat jede Mutter maximal zwei Töchter.
- In einem geordneten Baum sind alle Tochterelemente eines Mutterknotens nach einem bestimmten Kriterium geordnet abgelegt, z. B. nach dem Schlüsselattribut.
- In einem Suchbaum sind die Unterbäume nach einem Suchkriterium geordnet. Beispielsweise kann man in einem nach Alter sortierten Suchbaum sicher sein, dass alle Tochterknoten links eines Mutterknotens jünger sind als der Mutterknoten, alle älteren Tochterknoten sind rechts vom Mutterknoten angeordnet. Hier bildet der Baum offensichtlich nicht die biologische Mutter-Tochter-Beziehung ab, sondern eine Sortierung nach Alter.
- Bei balancierten Bäumen wird darauf geachtet, dass von einem Mutterknoten aus gesehen der rechte und linke Unterbaum bezüglich einer bestimmten Eigenschaft möglichst gleich sind, beispielsweise bezüglich Anzahl der Knoten oder Anzahl der Ebenen. Beim AVL-Baum weichen beispielsweise die Tiefen (Anzahl der Ebenen) von rechtem und linkem Unterbaum um maximal eine Ebene voneinander ab. Werden nun beispielsweise nur noch junge Frauen links unten hinzugefügt, wird der nach Alter sortierte Suchbaum bald unbalanciert und er muss regelmäßig umsortiert werden, um ihn auszubalancieren. Die-

ser Aufwand lohnt sich, wenn der Baum sehr oft durchsucht wird, weil dies Aufwand für die Suche einspart.

Es gibt außerdem noch 2-3-4-Bäume, B-Bäume, B+-Bäume, B*-Bäume, Bereichsbäume, binäre Suchbäume, BSP-Bäume, Fibonacci-Bäume, Heaps, k-d-Bäume, Max-Heaps, Min-Heaps, Quarternärbäume, R-Bäume, Rot-Schwarz-Bäume, Scapegoat Trees, Spannbäume, Splaybäume, Treaps und Tries. Aber das geht hier zu weit.

Solche Bäume kann man in verschiedenen Datenstrukturen abspeichern. Graphdatenbanken bestehen passenderweise aus Knoten und Kanten. Zusätzlich gibt es noch weitere Möglichkeiten:

- Durch Zeiger verweist ein Knoten auf den anderen.
- In einer Adjazenzliste wird für jeden Knoten die Liste seiner Nachbarn abgespeichert.
- Die Adjazenzmatrix stellt in der i-ten Zeile und j-ten Spalte dar, ob es eine Kante zwischen Knoten i und j gibt.
- Binäre Bäume kann man als Array / Feld speichern, in dem der Wurzelknoten die Nummer 1 erhält. Die weiteren Knoten werden ebenenweise von links nach rechts nummeriert (wie in Abbildung 1). Dann findet man die beiden Tochterknoten des Knoten i bei Index Nummer $2i$ und $2i+1$. Umgekehrt liegt der Mutterknoten des Knoten i bei Nummer $[i/2]$ (abgerundet).

Verschiedenste Algorithmen beschäftigen sich damit, Bäume für eine vorgegebene Datenmenge zu erstellen, sie um neue Knoten zu erweitern oder Knoten zu entfernen, den Baum auszubalancieren und zu optimieren, die Knoten nach ihrem Schlüsselattribut zu sortieren oder zu durchsuchen. Da es oft um sehr große Mengen von Daten geht, spielt die Zeitkomplexität eine Rolle, nämlich die Abhängigkeit der Laufzeit des Algorithmus von der Anzahl n der Knoten. Ist der Baum sortiert und ausbalanciert, wächst der Aufwand für die Suche nach einem bestimmten Datenobjekt nur logarithmisch mit n, während in anderen Datenstrukturen der Aufwand linear wächst.