Prof. Dr.-Ing. Dr. h. c. T. Härder
Fachbereich Informatik
AG Datenbanken und Informationssysteme
Universität Kaiserslautern

Übungsblatt 6

Unterlagen zur Vorlesung: „wwwdvs.informatik.uni-kl.de/courses/DBSREAL/“

Aufgabe 1: Darstellung von Sätzen - Extremlösung

Es soll eine Speicherungsstruktur für eine Satzmenge mit ≥ 200 Feldern/Satz entwickelt werden, von denen ggf. viele NULL-Werte besitzen können. Da der Software-Hersteller seine Anwendungssoftware weltweit vertreibt und den Kunden eine freie Wahl aus k ≤ 6 relationalen DBMS gestattet, muss die Struktur in allen DBMS gleichermaßen abbildbar sein. Jeder Satz oder mindestens jedes Satzfragment muss in einer Seite speicherbar sein, d.h. $S_l \leq L_S - L_{SK}$. Außerdem sollen sich die Felder indexieren lassen. Beliebige dynamische Erweiterungen mit neuen Feldern sind wünschenswert.

Als Beispiel wählen wir die Tabelle Pers mit den herkömmlichen Spalten (Attributen) Pnr, Vorname, Name, Beruf, Alter usw. Auf die einzelnen Felder wird mit unterschiedlicher Häufigkeit zugegriffen. Auf der DB-Schemaebene kann Pers natürlich vertikal partitioniert werden. Dann entstehen aber mehrere Tabellen, die bei DB-Operationen explizit angesprochen werden müssen.

a) Es werde eine Tabellenstruktur für die Sätze von Pers gewählt. Welche Speicherungsoptionen sind sinnvoll, wenn auch darauf geachtet werden muss, dass ein Satz (Satzfragment) immer in eine Seite passt?

b) Wenn eine Tabellenstruktur gewählt wurde, wie kann man dann auf die Programmierer zugreifen, die älter als 50 Jahre sind und den Vornamen Xaver haben?

c) Welche Systemunterstützung ist möglich, um diesen Anfragetyp so schnell wie möglich auswerten zu können?

d) Da Erweiterbarkeit, NULL-Werte und Abbildbarkeit auf k DBMS ein großes Problem bleiben, untersuchen Sie die extreme Form einer Speicherungsstruktur mit AOW (Attribut von Objekt ist Wert), wobei in einer AOW-Tabelle prinzipiell nur solchermaßen aufgebaute Tripel gespeichert werden. Die O-Spalte enthalte systemweite eindeutige OIDs als Werte.
Wie sieht die Speicherungsstruktur für einen Satz dieser Tabelle aus?

e) Formulieren Sie die vereinfachte obige Anfrage auf der AOW-Tabelle, wobei nur das Prädikat (Vorname = ‘Xaver’) verwendet wird. Die Ausgabe kann zunächst als Sub-Tabelle von AOW

spezifiziert werden.

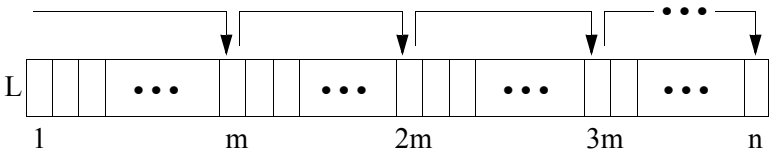
f) Wie lautet die Anfrage auf AOW mit (Vorname = ‘Xaver’ And Beruf = ‘Programmierer’ And Alter > 50), die das äquivalente Ergebnis zur Anfrage auf Pers erzielt? Da das in SQL sehr aufwendig ist, schränken Sie die Ausgabe auf die explizit aufgelisteten Attribute Pnr, Vorname, Name, Beruf und Alter ein.

g) Wie wird dynamische Erweiterbarkeit gelöst?

h) Die bisherige Darstellung der W-Spalte war stark vereinfacht, da ja Werte verschiedenen Typs in dieser Spalte auftreten. Das wird u. a. auch zum Problem, wenn man auf den Werten (vom gleichen Typ) Indexstrukturen einführen will. Wie kann dieses Problem gelöst werden?

Aufgabe 2: Suche in einer Liste - Sprungsuche

Gegeben ist eine Liste L mit n Einträgen fester Länge. Die Sprungsuche soll auf Liste L optimiert werden, wobei folgendes Schema angenommen wird:



a) Bei der einfachen Sprungsuche erfolgen konstante Sprünge zu den Positionen m, 2m, 3m usw. Sobald der Abschnitt, in den der gesuchte Schlüssel fällt, lokalisiert ist, wird sequentiell gesucht. Welche mittleren Suchkosten $C_{avg}(n)$ ergeben sich, wenn ein Sprung a und ein sequentieller Vergleich b Einheiten kostet?

b) Was sind die Kosten $C_{avg}(n)$ bei optimaler Sprungweite m?

c) Bei der Zwei-Ebenen-Sprungsuche wird statt sequentieller Suche im lokalisierten Abschnitt wiederum eine Quadratwurzel-Sprungsuche angewendet, bevor dann sequentiell gesucht wird. Welches Cavg(n) ergibt sich mit

a = Kosten eines Sprungs auf der ersten Ebene,
b = Kosten eines Sprungs auf der zweiten Ebene,
c = Kosten für einen sequentiellen Vergleich?

d) Welche Verbesserungen ergeben sich durch optimale Abstimmung der Sprungweiten m_1 und m_2 der beiden Ebenen, wenn a = b = c gesetzt wird?

e) Lässt sich die Effizienz der Suche steigern, wenn das Verfahren zu einem n-Ebenen-Verfahren verallgemeinert wird?

Aufgabe 3: Aufbau eines TRIE

a) Erstellen Sie einen Trie mit variablem Knotenformat für die Schlüsselfolge
OTTO, HUGO, OTTI, OLLI, EDDI, ELLI, HASI, HANS, ELSE, ULLI, ILSE.

b) Ergänzen Sie den Trie aus a) mit den Schlüsseln
OTT, HU, OTTILIE, OTTOKAR, ILSEBILL.

Aufgabe 4: PATRICIA-Baum

Erstellen Sie einen PATRICIA-Baum für die Schlüsselfolge aus Aufgabe 3a. Verwenden Sie folgende Kodierung der Buchstaben (ASCII):

A =	0100 0001	D =	0100 0100	E =	0100 0101
G =	0100 0111	H =	0100 1000	I =	0100 1001
L =	0100 1100	N =	0100 1110	O =	0100 1111
S =	0101 0011	T =	0101 0100	U =	0101 0101

Aufgabe 5: Analyse einer Hash-Funktion

Die Schlüssel zur Identifikation der Datensätze einer Anwendung seien als CHAR(8) im EBCDIC-Code definiert. Sie werden typischerweise als Zahlen in festen Abständen, also 10, 20, 30, ..., vergeben. Vor Verwendung in einer Hash-Funktion werden diese Schlüssel erst nach folgendem Schema gefaltet und dabei mit EXOR verknüpft. In hexadezimaler Darstellung (im EBCDIC-Code) ergibt sich beispielsweise nach Faltung für den Schlüssel S_1 mit 0000 0010 der Wert von K_1 mit

$$\begin{array}{rcl} & \text{F 0 F 0 F 0 F 0} & \\ & \text{F 0 F 0 F 1 F 0} & \oplus \\ \hline K_1 = & \text{0 0 0 0 0 1 0 0}_{16} & \triangleq \quad 256_{10} \end{array}$$

Danach werde das Divisionsrest-Verfahren mit

$$h(K_i) = K_i \bmod n$$

angewendet.

Analysieren Sie das Kollisionsverhalten dieser Funktion bei obiger Schlüsselvergabe und einem Hash-Bereich von n = 576 Buckets.

a) Nach welchen Abständen von j vergebenen Schlüsseln, die sich jeweils mit einem Inkrement von Δ k (was dem Wert von K_1 entspricht) erhöhen, treten Kollisionen auf?

b) Was ist eine effektive Abhilfemaßnahme?

Aufgabe 6: Erweiterbares Hashing 18

Die Sätze mit den Schlüsseln K10, J84, K35, A12, E77, K12, B22, K08 sollen in der angegebenen Reihenfolge mit dem Verfahren des Erweiterbaren Hashing in Buckets abgespeichert werden, die jeweils 2 Sätze aufnehmen können. Den Pseudoschlüssel des jeweiligen Satzes erhält man durch Verknüpfung der in EBCDIC codierten Zeichen mittels XOR. Zeichnen Sie die Belegung der Buckets und das Directory nach jeder Einfügung.