

Aufgabe 1 : Pufferverwaltung

a) Beschreiben Sie die Wirkungsweise der in der Vorlesung beschriebenen Ersetzungsstrategien der Pufferverwaltung (Folie 181) anhand des ebenfalls in der Vorlesung beschriebenen Beispiels des (natürlichen) Verbundes zweier Tabellen unter Einsatz des nested loop joins (Folie 184).

FIFO - Strategie:

Jede Seite muss wenigstens einmal geladen werden. Wenn der Puffer voll ist, wird die Seite, welche am längsten im Puffer ist, gelöscht. Weitere Kriterien gibt es nicht.

Vorteile: "Rückwärts-Navigation" ist sehr schnell

Nachteile: Stark frequentierte Seiten müssen periodisch neu geladen werden

LFU Strategie:

Hier spielt nicht das "Alter" eine Rolle, sondern die Anzahl der Zugriffe. Kann von Vorteil sein, aber "jüngere" Seiten haben schlechtere Chancen, eine Prioritätsstellung im Puffer zu bekommen wie "ältere" Seiten.

Vorteile: Speicher muss nur bei häufiger Verwendung von neuen Seiten öfter ausgetauscht werden

Nachteile: Neue Seiten immer alten Seiten gegenüber benachteiligt

LRU Strategie:

Es werden Seiten im Puffer ausgetauscht, die am längsten nicht mehr referenziert wurden. Hier wird also ein Datestamp gesetzt, das für Relevanz sorgt.

Vorteile: Laufend verwendete Seiten bleiben im Puffer

Nachteile: Neue und lange nicht mehr aktivierte Seiten müssen nachgeladen werden

CLOCK Strategie:

Seiten, die in der jüngsten Zeit am seltensten referenziert werden, werden ersetzt. Zwei Kriterien kommen hier zum Tragen. Es werden also die LFU- und LRU-Strategien vereinigt.

Vorteile: Praxisrelevante Seiten bleiben lange im Speicher, Nachteile der anderen Strategien werden weitestgehend ausgeräumt

Nachteile:

b) Wie sieht im Vergleich dazu der Ablauf aus, wenn die MRU (most recently used) Strategie eingesetzt wird, die diejenige Seite im Puffer ersetzt, die zuletzt referenziert wurde?

Betrachten Sie auch die Fälle, dass die Anzahl der Seiten der Tabelle B kleiner oder viel größer ist als die Anzahl der Pufferrahmen, die zur Verfügung stehen.

Stellen wir uns folgendes Szenario vor: Die Anzahl der Seiten ist um eins höher als die Anzahl der Pufferspeicher. Bei den unter a) aufgezählten Strategien wird sich jedes Mal einstellen, daß bei drohender Überfüllung des Puffers ausgerechnet jene Stelle überschrieben wird, welche die Seite, die als nächstes benötigt wird, enthält. Effektiv wird dadurch die Pufferung alles verlangsamen, weil sie bei jeder Inanspruchnahme einen Speicherbereich löschen und neu beschreiben muss.

Bei der MRU-Strategie wird der Speicher "von hinten" gelöscht, somit kann hier die Gefahr nicht auftreten.

Aufgabe 2 Pflichtaufgabe : feste und variable Satz­längen

Betrachten Sie die Tabelle employees des Schemas HR. Nehmen Sie eine Seitengröße von 512 Bytes und (als Vereinfachung) Seitengröße = nutzbarer Speicherbereich an . Oracle speichert numerische Daten in einem Format mit variabler Länge mit einem Byte für den Exponenten und maximal 20 Bytes für die Mantisse. Number(p) erfordert $(p+s)/2 + 1$ Bytes mit $s=0$ für positive und $s=1$ für neg. Zahlen. Der Maximalplatz beträgt 21 Bytes. Das interne Datenformat für Datumsangaben umfasst 7 Bytes (feste Länge)

- a) Nehmen Sie an, jedes Attribut wäre mit einem Datentyp fester (also maximaler) Länge vereinbart. Berechnen Sie die benötigten Satz­längen.

Die Satz­längen der Tabelle employees errechnet man mit:

$$\begin{array}{r} 5 \\ + 20 \\ + 25 \\ + 25 \\ + 20 \\ + 7 \\ + 10 \\ + 6 \\ + 3 \\ + 5 \\ + 4 \\ \hline = 130 \end{array}$$

- b) Berechnen Sie Unter- oder Obergrenzen der benötigten Satz­längen bei variabler Satz­länge.

Wie sind die Speicherauslastung und der Speicherbedarf bei 100000 Datensätzen (fester Länge), wenn man keine Spannsätze zulässt?

Hinweis: Sie können sich anhand der Oracle Dokumentation näher informieren für diesen Bereich bspw. unter Oracle concepts, Built-in datatypes (S:\allgemeines\literatur\Oracle8i\INDEX.HTM)

Die Obergrenze wurde bereits in a) mit 130 ermittelt. Die Untergrenze ist:

$$\begin{array}{r} 5 \\ + 1 \\ + 1 \\ + 1 \\ + 1 \\ + 7 \\ + 1 \\ + 6 \\ + 0 \\ + 0 \\ + 0 \\ \hline = 23 \end{array}$$