

The Grid File

An Adaptable, Symmetric Multikey File Structure

Vortrag von Markus Apell im Rahmen des Seminars „Support for Non-Standard Datatypes in DBMS“

Symmetrie und Anpassbarkeit

- **Symmetrie:** Jedes Attribut wird wie ein Primärschlüssel behandelt.
- Datenstrukturen wie Baumstrukturen oder Hash Files sind asymmetrisch, da sie einzelne Attribute bei den Operationen bevorzugen. Beispiel: Telefonbuch
- **Anpassbarkeit** (adaptable): Die Datenstruktur passt seine Form dem Inhalt an.
- Speicherplatzauslastung und Zugriffszeit ist im gesamten File nahezu gleich.
- Bei vielen älteren Strukturen wurden die Behandlung von Einfüge- und Löschoperationen vernachlässigt.
- Wenig elegante Lösungen wie Überlaufbereiche oder elegantere balancierte Bäume wurden entwickelt.
- **Multikey Access:** Zugriff anhand einem oder der Kombination von mehreren Attributwerten.
- Die meisten Multikey-Strukturen sind Weiterentwicklungen von Singlekey-Strukturen.

Die zwei Prinzipien

Neben der Speicherplatzauslastung sind die Zugriffszeiten ein wichtiges Qualitätsmerkmal, diese hängen fast nur von der Geschwindigkeit und der Anzahl der Plattenzugriffe ab.

- **Two-disk-access principle:** Eine voll spezifizierte Anfrage muss einen Datensatz nach mindestens zwei Plattenzugriffen zurückgegeben haben.
- **Efficient range queries with respect to all attributes:** Datensätze, die sich in den Werten ihrer Attribute ähneln, sollen auch im physischen Speicher so nahe wie möglich beieinander liegen.

Diese Prinzipien werden selten erfolgreich umgesetzt.

Das Konzept

Datensatz ist Punkt in einem k-dimensionalen Raum.

Beispiel für $k = 3$:

Datenraum $S = X * Y * Z$

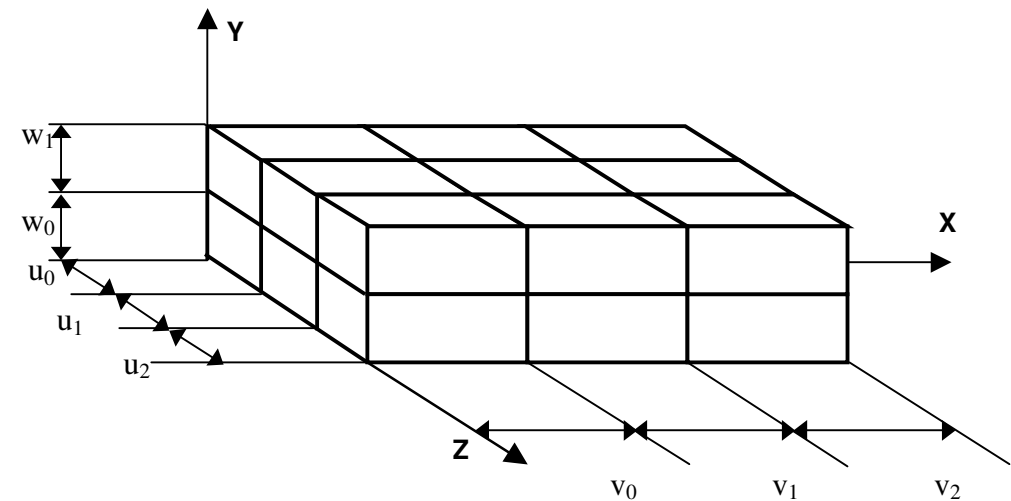
Zellpartition $P = U * V * W$

Mit den Abschnitten

$$U = (u_1, u_2 \dots u_l)$$

$$V = (v_1, v_2 \dots v_m)$$

$$W = (w_1, w_2 \dots w_n)$$



Die Koordinaten sind die Attribute des Datensatzes.

Durch diese Aufteilung entstehen die sogenannten Grid Blocks.

Das Grid File

Besteht aus folgenden Komponenten:

- **Buckets:**

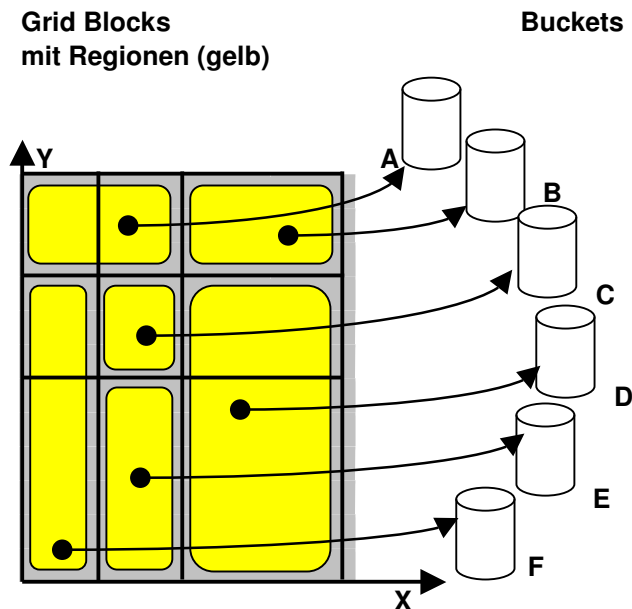
- Lagereinheit, enthält die Datensätze
- festgelegte Kapazität (in der Praxis 10 – 1000 Datensätze)
- Organisation innerhalb eines Buckets eher unwichtig und daher linear
- in einem Bucket können die Datensätze von mehreren Grid Blocks gespeichert sein
- Die Datensätze eines Grid Blocks sind in dem selben Bucket gespeichert

- **Grid Directory:**

- Bucketmanagementsystem des Grid Files
- Aufrechterhaltung der dynamischen Korrespondenz zwischen Grid Blocks und Buckets.
- Aktualisiert konvexe Zuordnungen beim Over- oder Underflow der Buckets.

Das Grid Directory

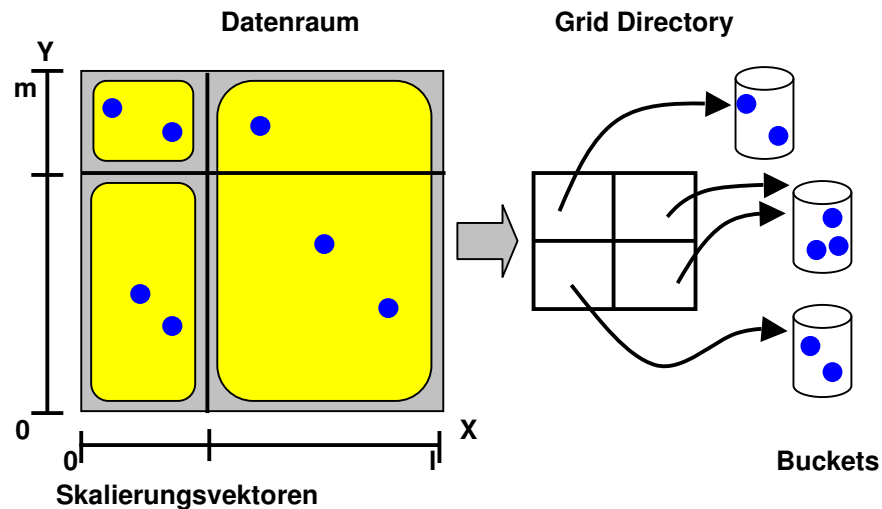
- **Grid Array:** k -dimensionales Array, dessen Inhalt die Zeiger auf die Buckets sind
- **Skalierungsvektoren (linear scales):** k eindimensionale Arrays, teilen die einzelnen Dimensionen ein



Region: Sammlung von Grid Blocks, die sich das selbe Bucket teilen.

Konvex: Grid Blocks, die sich das selbe Bucket teilen und vereinigt wieder ein Rechteck ergeben.

Beziehung zwischen den Komponenten



Beispiel für $k = 2$:

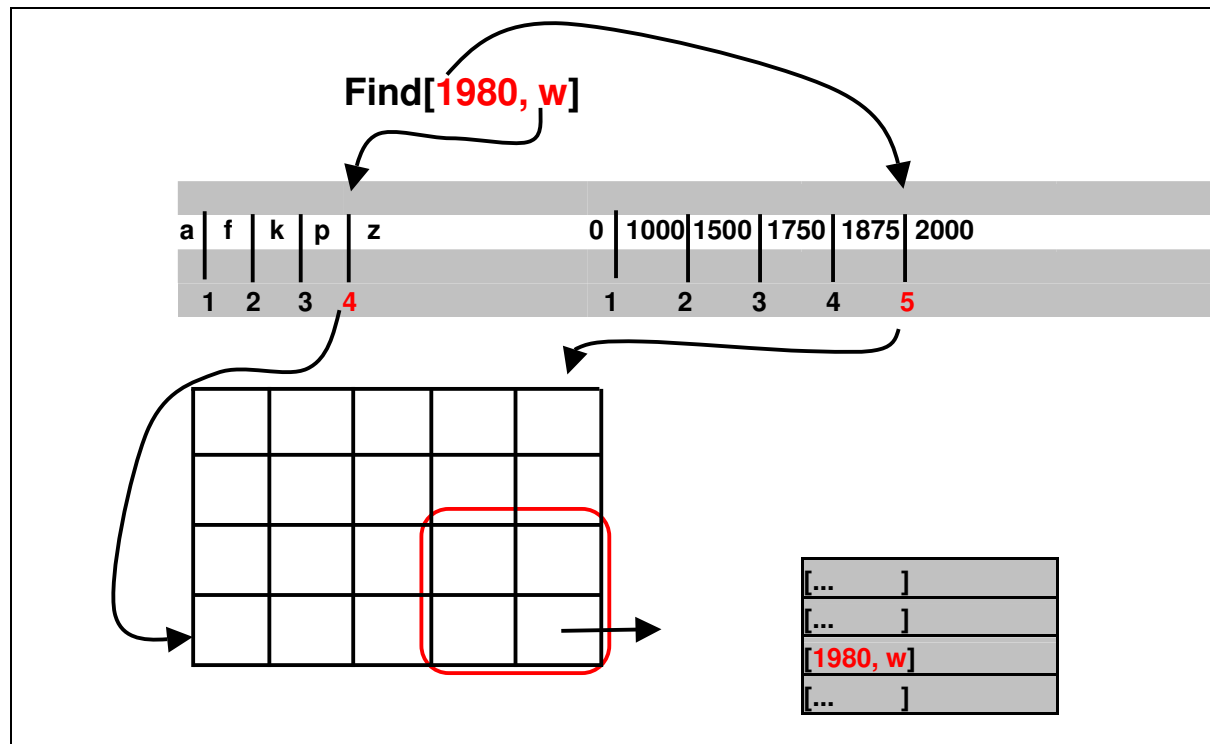
Gegeben: Grid Array G und Skalierungsvektoren X (0...1), Y (0...m).

Direkter Zugriff: $G(x_i, y_j)$

Wenn $G(x', y') = G(x'', y'')$ mit $x' \leq x \leq x''$ und $y' \leq y \leq y''$, dann

$G(x', y') = G(x, y) = G(x'', y'')$

Datensatzzugriff



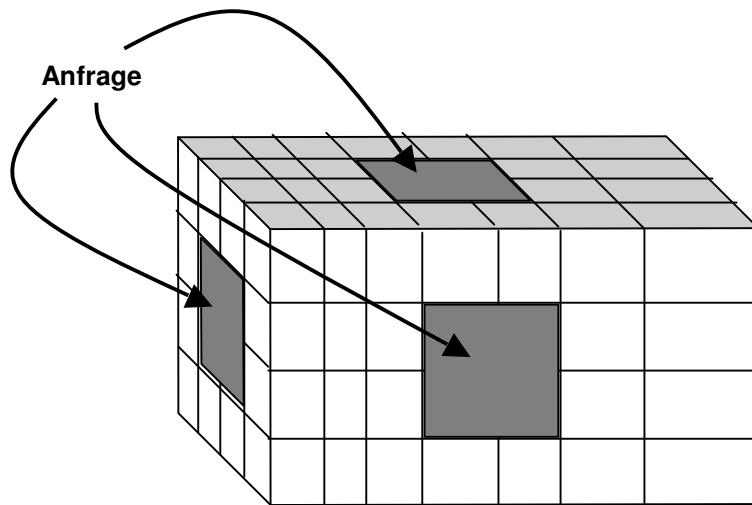
Anfrage: Find[1980, w]

```
SELECT *  
FROM G  
WHERE Year = „1980“  
AND Initial = „w“
```

Beachte: Das **Two-disk-access principle** wird eingehalten.

Bereichsanfrage

- Skalierungswerte jeder Dimension
- Finden der richtigen Grid Directory Einträge
- Laden der referenzierten Buckets



Dynamik des Grid Files

- **Einfügen** neuer Daten: Bucket wird gesucht und geladen. Daten werden eingefügt und das Bucket wieder auf die Festplatte geschrieben.
- **Löschen:** Analog zum Einfügen, nur wird der Datensatz entfernt.

Was passiert, wenn ein Bucket überläuft, oder (fast) leer ist?

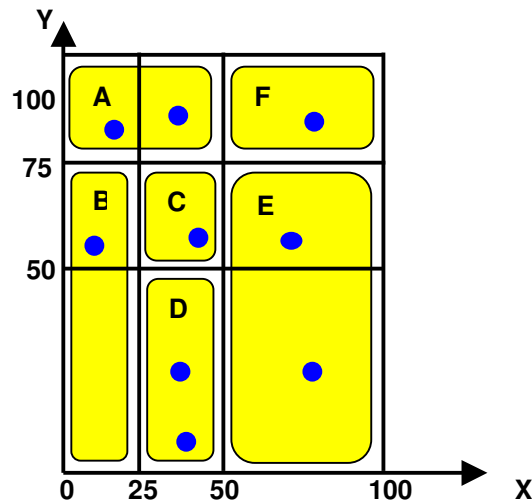
- **Teilung:** Wird die Bucketbelegung zu groß, so wird der Inhalt auf zwei Buckets verteilt. Dabei wird dessen Region und damit der gesamte Datenraum geteilt.
- **Verschmelzung:** Wenn die Bucketbelegung unter einen festgelegten Schwellenwert ist, bietet es sich an dieses Bucket zu löschen und seine Region mit der eines anderen zu verschmelzen.

Teilungsoperationen

Wenn der Datenraum gesplittet wird müssen dabei oft die Skalierungsvektoren neu eingeteilt werden.

Für die Wahl der Dimension die gesplittet wird gibt es mehrere Möglichkeiten:

- Zyklische Reihenfolge
- Die Dimension mit der größten Einteilung
- Die Dimension auf der schon eine geeignete Einteilung vorhanden ist.



Beispiel für Teilungsvorgänge:

Einfügen von Datensätzen bei einer Bucketkapazität von 2:

(90, 10)

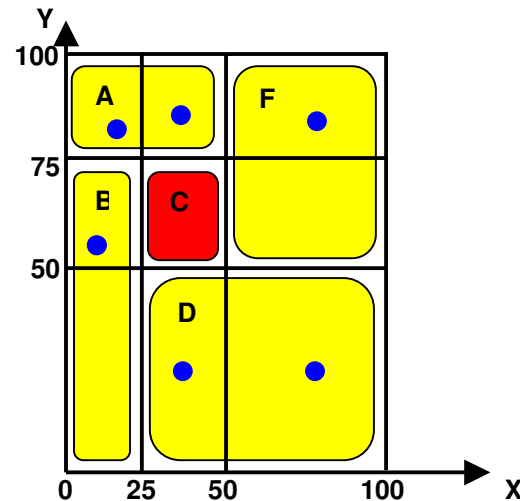
(40, 15)

Verschmelzungsoperationen

Das Verschmelzen von Sektionen des Grid Directory ist wenig rentabel.

Wesentliche Entscheidungen bei der Bucketverschmelzung:

- Welcher Bucket/Region ist Kandidat, so das der entstehende Bereich auch konvex ist?
→ Buddysystem, Nachbarschaftssystem
- Welcher Kandidat hat Priorität (welche Dimension ist vorzuziehen)?
- Was ist ein angemessener Grenzwert?
→ Das resultierende Bucket sollte eine Belegung von 70% haben

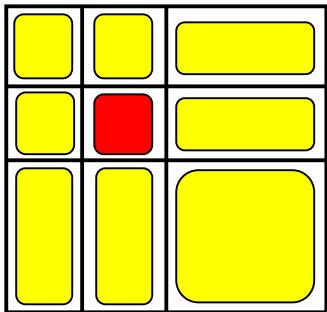


Verschmelzungsoperationen

Buddysystem (Twinsystem): Ein Bucket kann mit genau einem Buddy einer der k Dimensionen verschmelzen von dem es gespalten wurde.

Nachbarschaftssystem: Die zu verschmelzenden Regionen müssen nur benachbart sein und die entstehende Region muss wieder konvex sein.

Welche Möglichkeiten gibt es für den roten Bucket nach dem Buddy- bzw. Nachbarschaftssystem?



Implementierung des Grid Directory

Linked List:

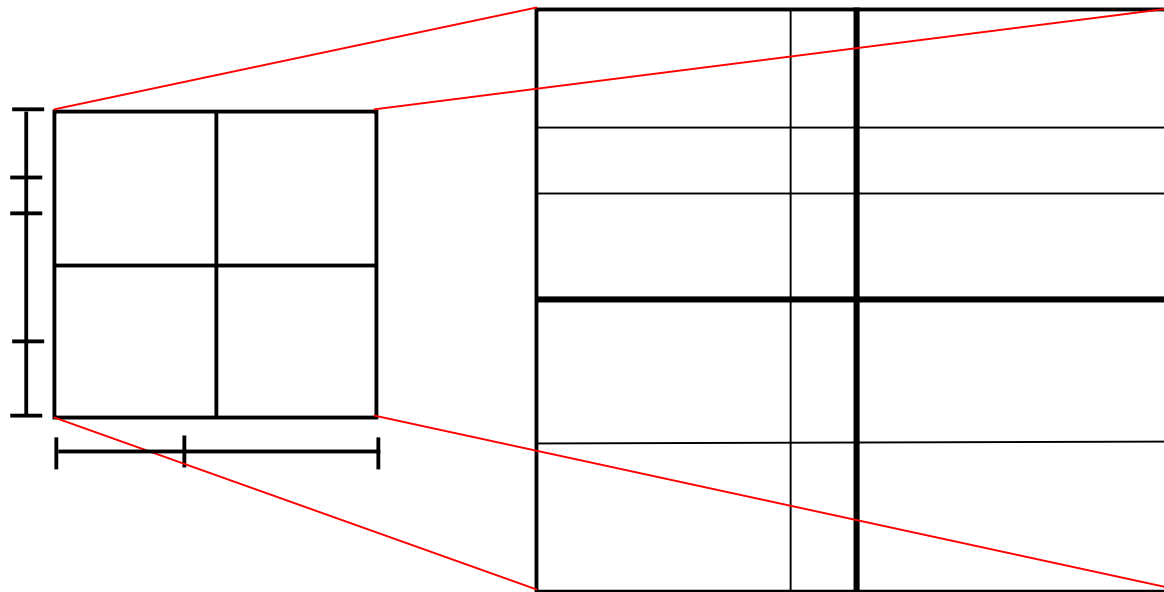
- Gut für Einfüge- und Löschooperationen an beliebigen Stellen.
- Schlecht: schneller Zugriff ist nur möglich, wenn Zeigerketten innerhalb eines Diskblocks sind.
- Nachteil: Es entsteht ein „Space overhead“ an Zeigern, der signifikant ist, verglichen mit dem normalen Inhalt eines Directoryelement.

k-dimensionales Array

- Nachteil: Teilungs- und Löschooperationen brauchen wegen des Umkopierens deutlich länger als bei Linked Lists. (Aber: Das alte Directory bleibt dabei bestehen!)
- Vorteil: Direkter Zugriff → sehr schnell!
- Vorteil: Optimale Memoryauslastung

Erweiterungen

- Resident Grid Directory
- Dynamisches Gewichten der Attribute
- Organisation von gleichzeitigen Zugriff



Ein kleines „Resident Grid Directory“ verwaltet das große Grid Directory auf der Platte.

Simulation

Warum eine Simulation und keine theoretische Analyse des Grid Files?

- Analysemodelle der Single-Key Strukturen können nicht auf Multikey-Strukturen verallgemeinert werden.
- Einige Parameter des Grid File sind zu kompliziert um sie in ein mathematisches Modell unterbringen zu können. (Beispiel: diverse Teilungs- und Verschmelzungsmethoden)

Augenmerk der Simulation

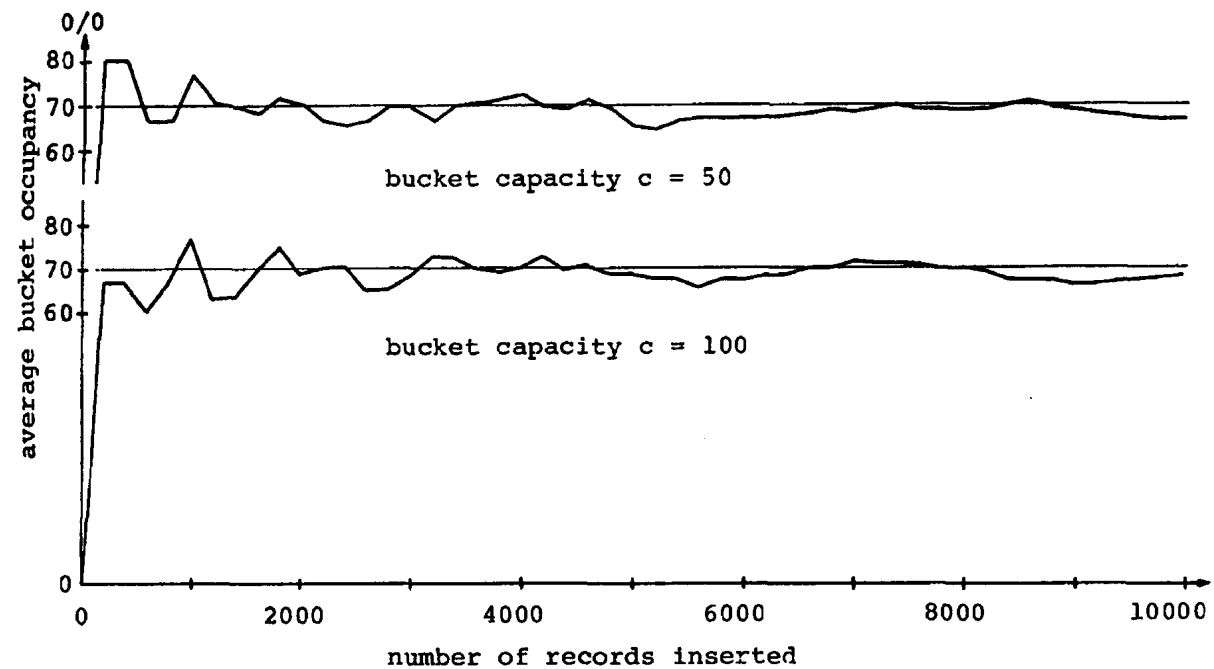
- Abschätzung der Bucketbelegung
- Abschätzung der Directorygröße
- Visualisierung der Geometrie der Bucketregionen
- Bewertung der Teilungs- und Verschmelzungsmethoden

Das wachsende File

Das Grid File wird mit Einträgen überhäuft (10.000 Datensätze).

Die Bucketkapazität ist festgelegt ($c = 50$ oder $c = 100$).

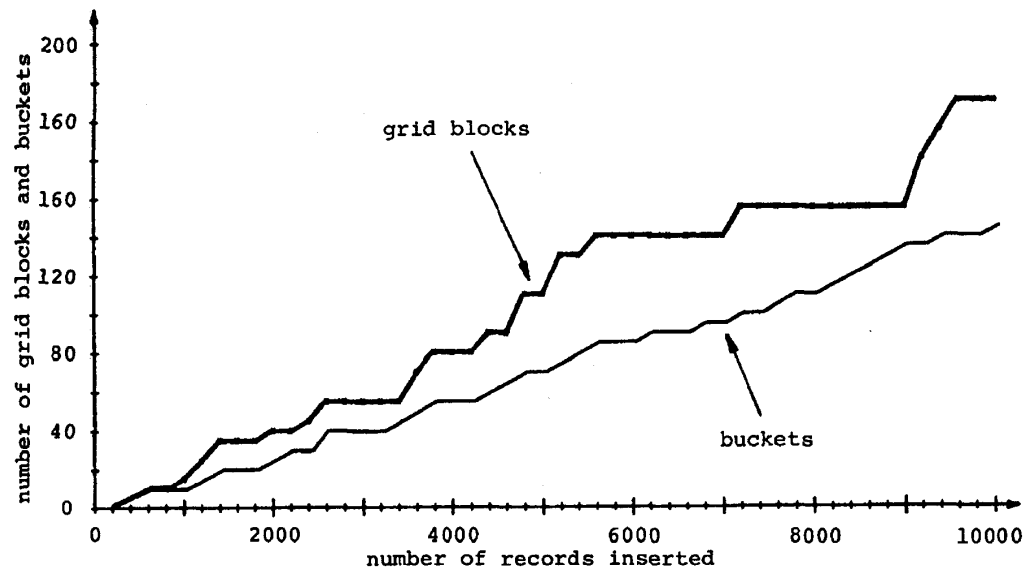
Bucketbelegung:



Das wachsende File

Directorygröße:

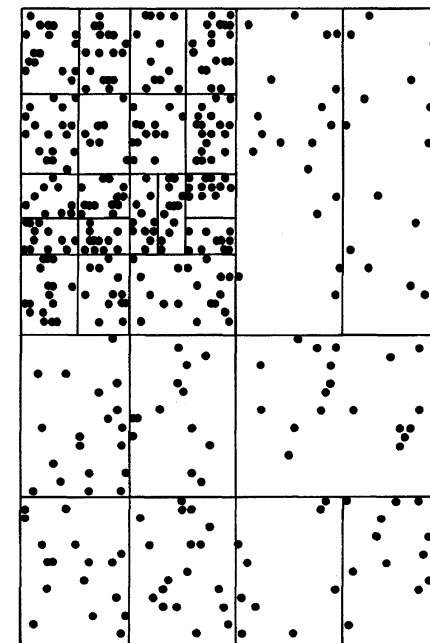
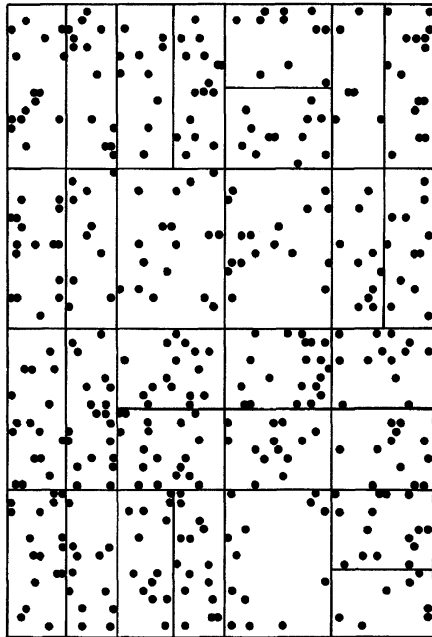
- Da sich mehrere Grid Blocks ein Bucket teilen können, kann man nicht von dem Wachstum der Bucket auf das des Grid Directory schließen.
- Korrelierte Attribute lassen das Grid Directory deutlich schneller wachsen als die Buckets
- Worst-case: Alle Datensätze sind im selben kleinen Bereich. Die Anzahl der Grid Blocks steigt schneller als die der Einträge. (Passiert aber nur bei kleiner Bucketkapazität)



Das wachsende File

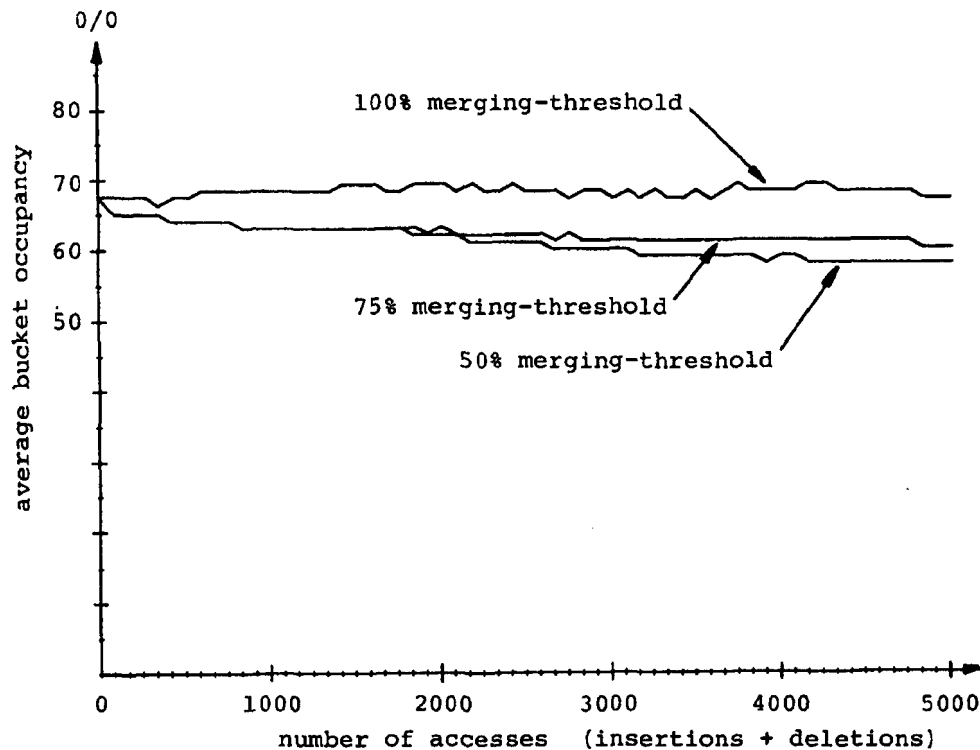
Visualisierung der Geometrie der Bucketregionen:

- Bucketregionen mit gleichverteilten Daten
 - Bucketregionen mit ungleichverteilten Daten
- 400 Einträge, Bucketkapazität von 20



Das gleichgroß bleibende File

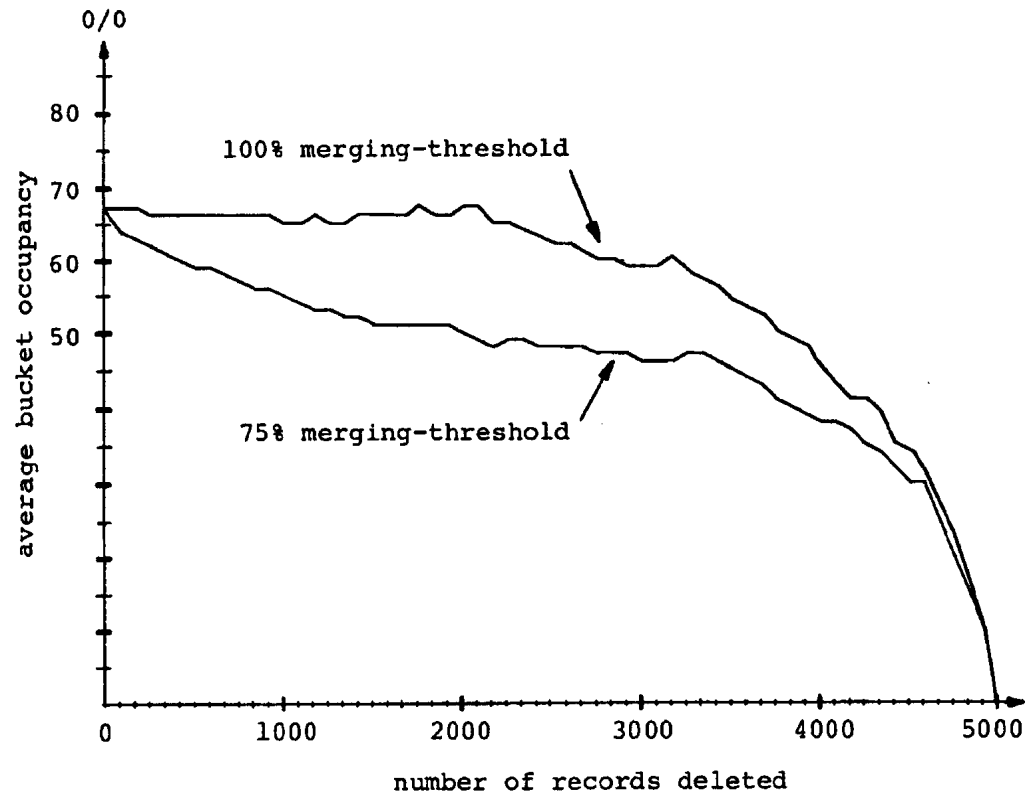
Durch viele Einfüge- und Löschoperationen bleibt die Anzahl der Daten im File konstant.



Um die Anzahl der Operationen gering zu halten, empfiehlt sich ein Verschmelzungsgrenzwert von 70%.

Das schrumpfende File

Aus einem gefüllten Grid File werden kontinuierlich Daten entfernt.



Beobachtung:

Niedrige Bucketauslastung

Bei einem schrumpfenden File ist ein Verschmelzungsgrenzwert von 100% optimal.

Schlussfolgerung

Das Grid File eignet sich für große Datenmengen mit Daten, die weniger als 10 Attribute haben, deren Wertebereich groß und linear geordnet ist.

- Symmetrische, anpassende Multikey-Struktur
- Erfüllt die zwei Prinzipien
- Hat ein Grid Directory zur Verwaltung der Korrespondenz
- Schneller Zugriff auf individuelle Datensätze
- Erlaubt effiziente Bereichsanfragen
- Unempfindlich gegenüber „Datenhaufen“
- Simulation zeigt, dass die Speicherauslastung gut ist (70%)

Quellenangaben

- [a] J. Nievergelt, H. Hinterberger (Institut für Informatik, ETH Zürich) K.C. Sevik (University of Toronto): An Adaptable, Symmetric Multikey File Structure. ACM Transactions on Database Systems, Vol. 9, No.1, March 1984, Pages 38-71.

Bildnachweis

- Ab Seite 17 sind alle Bilder (Simulationsergebnisse) aus [a] entnommen.
- Die übrigen Abbildungen sind selbst erstellt worden.
- Die Grafiken auf Seite 12 und 13 sind an Bildern aus J. Graf: „Das Gridfile“ angelehnt.