# AG Datenbanken und Informationssysteme

**Wintersemester 2006 / 2007**

Prof. Dr.-Ing. Dr. h. c. Theo Härder
Fachbereich Informatik
Technische Universität Kaiserslautern

**http://wwwdvs.informatik.uni-kl.de**

# *5. Übungsblatt*

Für die Übung am Donnerstag, **30. November 2006**,
von 15:30 bis 17:00 Uhr in 13/222.

## Aufgabe 1: Zugriffskostenvergleich mit und ohne Clusterung

In einer DB sei ein Segment mit M Seiten gegeben, in dem mehrere Relationen und die zugehörigen Indexstrukturen gespeichert sind. Unter anderem befindet sich in diesem Segment folgende Relation:

```
PERSONAL (PNR, NAME, ADRESSE, GEHALT, MNR, ... )
```

mit Attributen für die Personalnummer, den Namen des Angestellten, die Adresse, das Gehalt sowie den Manager, bei dem der Mitarbeiter beschäftigt ist. Daneben sei noch weitere Information enthalten. Die Relation `PERSONAL` enthalte P Datensätze. Bei Clusterbildung seien in den zum Cluster gehörenden Seiten durchschnittlich C Datensätze zu finden. Weiterhin habe ein Manager im Mittel Q Mitarbeiter.
Wie viele Seitenzugriffe sind notwendig, um folgende Anfragen zu beantworten:

`SELECT * FROM PERSONAL WHERE MNR = X`, wobei `X` eine Personalnummer ist.

```
SELECT * FROM PERSONAL ORDER BY MNR
```

jeweils unter Berücksichtigung nachfolgender Situationen:

Es existieren keinerlei Indexstrukturen auf der Relation `PERSONAL`.

Es gibt eine Indexstruktur auf `PNR` von `PERSONAL`.

Es gibt eine Indexstruktur mit Clusterbildung auf `PNR` von `PERSONAL`.

Es gibt eine Indexstruktur auf `MNR` von `PERSONAL`.

Es gibt eine Indexstruktur mit Clusterbildung auf `MNR` von `PERSONAL`.

Man benutze dabei einen B*-Baum als Indexstruktur, bei der nur Zeiger auf Seiten, in denen sich die zugehörigen Datensätze befinden, in Blättern gespeichert sind. Anschließend berechnen Sie die jeweiligen Kosten für folgende Größen:

M = $10^6$ Seiten, P = $10^5$ Datensätze, C = 5 Datensätze, Q = 100 Datensätze.

Die Seitengröße $S_L$ beträgt 4 KB.

Die Eintraglänge im B*-Baum $E_L$ beträgt 10 Bytes.

Was passiert bei Änderungen, wenn eine Indexstruktur mit Clusterbildung vorliegt?

**Lösung:**

Für die allgemeine Berechnung der Zugriffskosten sind folgende Daten gegeben:

- M ist die Seitenanzahl im Segment.
- P ist die Datensatzanzahl von PERSONAL.
- C ist die Datensatzanzahl pro Seite bei Clusterbildung.
- Q ist die Durchschnittsanzahl der Mitarbeiter pro Manager.
- $h_B$ sei die B*-Baumhöhe.
- $N_B$ sei die Anzahl der Blattseiten in einem B*-Baum.
- $N_{BQ}$ sei die Anzahl der Blattseiten in einem B*-Baum, die Zeiger auf Datenseiten enthalten, in denen Datensätze von `PERSONAL` mit `MNR = X` sich befinden.
- Die Suchkosten eines Datensatzes in einer Seite sei ignoriert.

a) `SELECT * FROM PERSONAL WHERE MNR = X`, wobei `X` eine Personalnummer ist.

(1) Es existieren keinerlei Indexstrukturen auf der Relation `PERSONAL`.
Alle M Seiten müssen zugegriffen werden.
=> M Seitenzugriffe

(2) Es gibt eine Indexstruktur auf `PNR` von `PERSONAL`.
Mittels der existierenden Indexstruktur greift man auf den ersten Datensatz von `PERSONAL` zu und durchsucht die weiteren Datensätze über die Blattseiten des Baumes. Im schlechtesten Falle sind die Datensätze so verteilt, daß nur ein Datensatz von `PERSONAL` in einer Seite existiert und man damit P Seitenzugriffe zum Durchsuchen braucht.
=> $(h_B + N_B + P)$ Seitenzugriffe

(3) Es gibt eine Indexstruktur mit Clusterbildung auf `PNR` von `PERSONAL`.
Mittels der existierenden Indexstruktur greift man wiederum auf den ersten Datensatz von `PERSONAL` zu und durchsucht die weiteren Datensätze über die Blattseiten des Baumes. Durch die Clusterbildung findet man bei einem Seitenzugriff P Datensätze. Damit benötigt man (P / C) Seitenzugriffe zum Durchsuchen.
=> $(h_B + N_B + (P / C))$ Seitenzugriffe

(4) Es gibt eine Indexstruktur auf `MNR` von `PERSONAL`.
Mittels der existierenden Indexstruktur auf `MNR` greift man auf den ersten Datensatz von `PERSONAL` zu, der die Bedingung `MNR = X` genügt. Über die Blattseiten bekommt man dann die weiteren qualifizierten Datensätze. Daher braucht man laut Annahme Q Seitenzugriffe.
=> $(h_B + N_{BQ} + Q)$ Seitenzugriffe

(5) Es gibt eine Indexstruktur mit Clusterbildung auf `MNR` von `PERSONAL`.
Mittels der existierenden Indexstruktur auf `MNR` greift man auf den ersten Datensatz von `PERSONAL` zu, der die Bedingung `MNR = X` genügt. Über die Blattseiten bekommt man dann die weiteren qualifizierten Datensätze. Durch die Clusterbildung findet man bei einem Seitenzugriff P Datensätze. Damit benötigt man laut Annahme (Q / C) Seitenzugriffe zum Durchsuchen.
=> $(h_B + N_{BQ} + (Q / C))$ Seitenzugriffe

b) `SELECT * FROM PERSONAL ORDER BY MNR`

(1) Es existieren keinerlei Indexstrukturen auf der Relation `PERSONAL`.
=> M Seitenzugriffe + Sortierkosten

(2) Es gibt eine Indexstruktur auf `PNR` von `PERSONAL`.
=> ($h_B$ + $N_B$ + P) Seitenzugriffe + Sortierkosten

(3) Es gibt eine Indexstruktur mit Clusterbildung auf `PNR` von `PERSONAL`.
=> ($h_B$ + $N_B$ + (P / C)) Seitenzugriffe + Sortierkosten

(4) Es gibt eine Indexstruktur auf `MNR` von `PERSONAL`.
Die Sortierung der Datensätze liegt bereits durch die Indexstruktur vor.
=> ($h_B$ + $N_{BQ}$ + Q) Seitenzugriffe

(5) Es gibt eine Indexstruktur mit Clusterbildung auf `MNR` von `PERSONAL`.
Die Sortierung der Datensätze liegt bereits durch die Indexstruktur vor.
=> ($h_B$ + $N_{BQ}$ + (Q / C)) Seitenzugriffe

Kostenberechnung mit folgenden Größen:

- M = $10^6$ Seiten, P = $10^5$ Datensätze, C = 5 Datensätze, Q = 100 Datensätze
- Die Seitengröße $S_L$ = 4 KB
- Die Eintraglänge im B*-Baum $E_L$ = 10 Bytes

Bem.: Siehe Vorlesungsskript, 3. Kapitel, Seite 9-13.

Die max. Anzahl der Einträge pro Seite:

ES = $S_L$ / $E_L$ = 4000 Bytes / 10 Bytes = 400

Die Höhe des B*-Baumes läßt sich wie folgt abschätzen:

- Bei minimaler Belegung des B*-Baums hat man aus $N_T = 2 * (ES / 2)^{(hB - 1)}$ die max. Höhe:

$$h_{Bmax} = \lceil \log ((N_T / 2) - (ES / 2)) + 1 \rceil$$
$$= \lceil \log ((100.000 / 2) / \log (400 / 2)) + 1 \rceil$$
$$= \lceil (\log 50.000 / \log 200) + 1 \rceil = \lceil 2{,}04 + 1 \rceil = 4$$

- Bei maximaler Belegung des B*-Baums hat man aus $N_T = ES^{hB}$ die min. Höhe:

$$h_{Bmin} = \lceil \log N_T - ES \rceil = \lceil \log 100.000 / \log 400 \rceil$$
$$= \lceil 1{,}92 \rceil = 2$$

Die Anzahl der Blattseiten im B*-Baum $N_B$ läßt sich wie folgt abschätzen:

- $N_{Bmin}$ liegt vor, wenn alle Blattseiten voll sind, und $N_{Bmax}$ wenn alle Blattseiten halb so voll sind.

$$N_{Bmin} = (N_T / ES) = (10^5 / 400) = 250 \leq N_B \leq N_{Bmax} = (N_T / ES / 2) = (10^5 / 200) = 500$$

Für die beiden Fälle (4) und (5) hat man:

$N_{BQ}$ = 1, da Q < (ES/2)

Somit hat man:

a) `SELECT * FROM PERSONAL WHERE MNR = X`, wobei `X` eine Personalnummer ist.

| | Kein Index | Index auf PNR | Index auf PNR mit Clusterung | Index auf MNR | Index auf MNR mit Clusterung |
|---|---|---|---|---|---|
| $N_{Bmin}$ , $h_{Bmin}$ | **1.000.000** | 2 + 250 + $10^5$ = **100.252** | 2+250+($10^5$/5) = **20.252** | 2 + 1 + 100 = **103** | 2+1+(100/5) = **23** |
| $N_{Bmax}$ , $h_{Bmax}$ | wie oben | 4 + 500 + $10^5$ = **100.504** | 4+500+($10^5$/5) = **20.504** | 4 + 1 + 100 = **105** | 4+1+(100/5) = **25** |

b) `SELECT * FROM PERSONAL ORDER BY MNR`

| | Kein Index | Index auf PNR | Index auf PNR mit Clusterung | Index auf MNR | Index auf MNR mit Clusterung |
|---|---|---|---|---|---|
| $N_{Bmin}$ , $h_{Bmin}$ | **1.000.000** + Sortierkosten | 2 + 250 + $10^5$ = **100.252** + Sortierkosten | 2+250+($10^5$/5) = **20.252** + Sortierkosten | 2 + 1 + 100 = **103** | 2+1+(100/5) = **23** |
| $N_{Bmax}$ , $h_{Bmax}$ | wie oben | 4 + 500 + $10^5$ = **100.504** + Sortierkosten | 4+500+($10^5$/5) = **20.504** + Sortierkosten | 4 + 1 + 100 = **105** | 4+1+(100/5) = **25** |

Beim Vorliegen einer Indexstruktur mit Clusterbildung sind Änderungen sehr teuer, da die betroffenen Datensätze in den Datenseiten entsprechend verschoben werden müssen, um die Clustereigenschaft zu gewährleisten.

## Aufgabe 2: Kostenmodelle für die Selektionsoperation

Gegeben sei eine Tabelle R mit den Attributen A1, A2, A3, ..., An, die zusammenhängend in den Seiten des Segments S gespeichert ist.

R ( A1, A2, A3, ..., An )

Das Segment S habe $M_S=10^4$ Seiten. Die Tabelle R habe $N_R=10^5$ Sätze und ggf. (für die entsprechenden Aufgabenstellungen) einen Cluster-Faktor $c_R=50$.

Weiterhin seien die Indizes $I_R(A1)$ mit $j_{A1}=100$ und $I_R(A2)$ mit $j_{A2}=10$ für die Attribute A1 und A2 angelegt. Bei Indizes sind jeweils als B*-Bäume mit der Höhe $h_B=2$ und $N_B=100$ Blattseiten realisiert.

a) Wie teuer (Anzahl der Seitenzugriffe) ist die Auswertung der SQL-Anfrage

```
SELECT   *
FROM     R
WHERE    A3='x'
```

   (1) bei einem Tabellen-Scan?

   (2) bei Nutzung des Indexes $I_R(A1)$?

   (3) bei Nutzung des Indexes $I_R(A2)$ mit Cluster-Bildung?

   (4) wenn die Tabelle als Hash-Struktur mit A3 als Primärschlüssel angelegt ist?

b) A1 habe 100 Werte, die von 1 bis 100 gleichverteilt vorkommen ($j_{A1}=100$). Wie teuer ist die Auswertung der SQL-Anfrage

```
SELECT   *
FROM     R
WHERE    A1>50
```

   (1) bei Nutzung des Indexes $I_R(A1)$?

   (2) bei Annahme einer Cluster-Bildung bei $I_R(A1)$?

   (3) ohne Indexnutzung?

c) Welche Kosten verursacht die SQL-Anfrage

```
SELECT   *
FROM     R
WHERE    A1=50 AND A2=10
```

   (1) bei Nutzung von $I_R(A1)$ und $I_R(A2)$ jeweils ohne Cluster-Bildung?

   (2) bei gemeinsamer Nutzung von $I_R(A1)$ und $I_R(A2)$ mit Cluster-Bildung?

   (3) bei Zugriff nur über $I_R(A2)$ mit Cluster-Bildung?

   (4) bei Zugriff nur über $I_R(A1)$ mit Cluster-Bildung?

**Lösung:**

a) Wie teuer (Anzahl der Seitenzugriffe) ist die Auswertung der SQL-Anfrage

```
SELECT   *
FROM     R
WHERE    A3='x'
```

(1) bei einem Tabellen-Scan?
$C_{a1} = M_S = 10^4$ (Seiten)

(2) bei Nutzung des Indexes $I_R(A1)$?
$C_{a2} = h_B + N_B - 1 + N_R = 2 + 99 + 10^5$ (Seiten)

(3) bei Nutzung des Indexes $I_R(A2)$ mit Cluster-Bildung?
$C_{a3} = h_B + N_B - 1 + N_R / c_R = 2 + 99 + 2 \cdot 10^3$ (Seiten)

(4) wenn die Tabelle als Hash-Struktur mit A3 als Primärschlüssel angelegt ist?
$C_{a4} = 1$

b) A1 habe 100 Werte, die von 1 bis 100 gleichverteilt vorkommen ($j_{A1}$=100). Wie teuer ist die Auswertung der SQL-Anfrage

```
SELECT   *
FROM     R
WHERE    A1>50
```

(1) bei Nutzung des Indexes $I_R(A1)$?
$C_{b1} = h_B + N_B / 2 + (N_R / j_{A1}) \cdot j_{A1} / 2 = 2 + 50 + 10^5/2$ ($N_B$-1 vernachlässigt wg. / 2)

(2) bei Annahme einer Cluster-Bildung bei $I_R(A1)$?
$C_{b2} = h_B + N_B / 2 + (N_R / j_{A1}) \cdot j_{A1} / (2 \cdot c_R) = 2 + 50 + 10^3$

(3) ohne Indexnutzung?
$C_{b3} = M_S = 10^4$

c) Welche Kosten verursacht die SQL-Anfrage

```
SELECT   *
FROM     R
WHERE    A1=50 AND A2=10
```

(1) bei Nutzung von $I_R(A1)$ und $I_R(A2)$ jeweils ohne Cluster-Bildung?
$C_{c1} = h_B + N_B / j_{A1} + h_B + N_B / j_{A2} + N_R / (j_{A1} \cdot j_{A2}) = 2 + 1 + 2 + 10 + 10^5 / 10^3 = 115$

(2) bei gemeinsamer Nutzung von $I_R(A1)$ und $I_R(A2)$ mit Cluster-Bildung?
$C_{c2} = h_B + N_B / j_{A1} + h_B + N_B / j_{A2} + N_R / (j_{A1} \cdot j_{A2} \cdot c_R) = 2 + 1 + 2 + 10 + 10^5 / (10^3 \cdot 50) = 17$

(3) bei Zugriff nur über $I_R(A2)$ mit Cluster-Bildung?
$C_{c3} = h_B + N_B / j_{A2} + N_R / (j_{A2} \cdot c_R) = 12 + 10^5 / (5 \cdot 10^2) = 212$

(4) bei Zugriff nur über $I_R(A1)$ mit Cluster-Bildung?
$C_{c4} = h_B + N_B / j_{A1} + N_R / (j_{A1} \cdot c_R) = 3 + 20 = 23$

## Aufgabe 3: Allgemeines zum Sicht-Konzept

Welche Maßnahmen erfordert die Definition einer speziellen Sicht, wenn als Sichtsemantik

a) eine Kopie des aktuellen Sichtinhalts

b) ein dynamisches Fenster auf die Basisrelation

unterstellt wird?

Diskutieren Sie verschiedene Fälle von Manipulationen an der Basisrelation `ANGESTELLTE` für das folgende Beispiel:

```
CREATE VIEW UNTERBEZAHLT AS
SELECT PNR, NAME, GEHALT
FROM   ANGESTELLTE
WHERE  BERUF='PROGRAMMIERER' AND GEHALT<20000
```

**Lösung:**

a) Nur bei Änderungsoperationen wie INSERT, UPDATE und DELETE muss die Kopie eventuell aktualisiert werden, d. h. wenn es von den Änderungsoperationen betroffenen Tupel gibt, welche die in der Sicht definierten Bedingungen erfüllen.
Annahme:
`ANGESTELLTE (PNR, NAME, BERUF, GEHALT)`, `PNR` und `GEHALT` sind vom Typ `INTEGER`, `NAME` und `BERUF` sind vom Typ `CHAR(30)`.
Folgende Fälle sind zu beachten:

(1) SELECT
Eine SELECT-Anweisung auf ANGESTELLTE beeinflusst den Inhalt der materialisierten Sicht nicht, da sie lesend ist. Es sind also keine speziellen Maßnahmen nötig.

(2) INSERT
Falls eine INSERT-Anweisung durchgeführt wird, welche die View-Bedingung erfüllt, dann muss die materialisierte Sicht ebenfalls entsprechend aktualisiert werden.
Bsp.:
```
INSERT INTO ANGESTELLTE
VALUES (4711, 'MUSTERMANN', 'PROGRAMMIERER', 9000);
```

(3) DELETE
Falls eine DELETE-Anweisung durchgeführt wird, und dabei Tupel gelöscht werden, welche die View-Bedingung erfüllen, dann muss die materialisierte Sicht ebenfalls entsprechend aktualisiert werden.
Bsp.:
```
DELETE FROM ANGESTELLTE WHERE PNR = 4711;
```

(4) UPDATE
Falls eine UPDATE-Anweisung durchgeführt wird, die Tupel aktualisiert, welche die View-Bedingung erfüllen, dann muss die materialisierte Sicht ebenfalls entsprechend aktualisiert werden.
Bsp.:
```
UPDATE ANGESTELLTE SET GEHALT = GEHALT * 2 WHERE PNR = 4711;
```

b) Hier sind keine zusätzlichen Maßnahmen erforderlich, da die Sicht erst bei ihrer Benutzung aus der Basisrelation und der Sichtdefinition dynamisch erzeugt wird. Die Sicht enthält somit immer die aktuellsten Daten.

## Aufgabe 4: CHECK OPTION bei Sichten in SQL

Gegeben seien folgende SQL-Anweisungen:

```
CREATE TABLE T (S1 INT, S2 INT, S3 INT, S4 INT, S5 INT);

CREATE VIEW V1 AS
       SELECT * FROM T WHERE S1=1;

CREATE VIEW V2 AS
       SELECT * FROM V1 WHERE S2=2 WITH LOCAL CHECK OPTION;

CREATE VIEW V3 AS
       SELECT * FROM V2 WHERE S3=3;

CREATE VIEW V4 AS
       SELECT * FROM V3 WHERE S4=4 WITH CASCADED CHECK OPTION;

CREATE VIEW V5 AS
       SELECT * FROM V4 WHERE S5=5;
```

Ist die Ausführung der nachfolgenden `INSERT`-Anweisungen erfolgreich? Geben Sie jeweils eine kurze Begründung an.

a) `INSERT INTO V1 VALUES (2, 1, 3, 2, 5);`

b) `INSERT INTO V2 VALUES (2, 1, 3, 2, 5);`

c) `INSERT INTO V2 VALUES (2, 2, 3, 2, 5);`

d) `INSERT INTO V3 VALUES (2, 2, 4, 2, 5);`

e) `INSERT INTO V3 VALUES (1, 3, 3, 2, 5);`

f) `INSERT INTO V4 VALUES (2, 2, 3, 2, 5);`

g) `INSERT INTO V4 VALUES (2, 1, 3, 4, 5);`

h) `INSERT INTO V4 VALUES (1, 2, 2, 4, 5);`

i) `INSERT INTO V4 VALUES (2, 2, 3, 4, 5);`

j) `INSERT INTO V5 VALUES (1, 2, 3, 4, 6);`

k) `INSERT INTO V5 VALUES (1, 2, 4, 4, 5);`

**Lösung:**

a) `INSERT INTO V1 VALUES (2, 1, 3, 2, 5);`
Erfolgreich, da CHECK OPTION in V1 nicht spezifiziert ist.

b) `INSERT INTO V2 VALUES (2, 1, 3, 2, 5);`
Nicht erfolgreich aufgrund der CHECK OPTION in V2 (S2=2).

c) `INSERT INTO V2 VALUES (2, 2, 3, 2, 5);`
Erfolgreich aufgrund der LOCAL CHECK OPTION in V2 (S2=2), die Einschränkung in V1 wird nicht berücksichtigt.

d) `INSERT INTO V3 VALUES (2, 2, 4, 2, 5);`
Erfolgreich, da CHECK OPTION in V3 nicht definiert ist, die Bedingung in V2 erfüllt ist und die Bedingung in V1 aufgrund der CHECK OPTION in V2 nicht berücksichtigt wird.

e) `INSERT INTO V3 VALUES (1, 3, 3, 2, 5);`
Nicht erfolgreich, da die Bedingung in V2 nicht erfüllt ist.

f) `INSERT INTO V4 VALUES (2, 2, 3, 2, 5);`
Nicht erfolgreich, da die Bedingung in V4 nicht erfüllt ist und CHECK OPTION definiert ist.

g) `INSERT INTO V4 VALUES (2, 1, 3, 4, 5);`
Nicht erfolgreich, da die Bedingung in V2 aufgrund der CASCADED CHECK OPTION in V4 nicht erfüllt ist.

h) `INSERT INTO V4 VALUES (1, 2, 2, 4, 5);`
Nicht erfolgreich, da die Bedingung in V3 aufgrund der CASCADED CHECK OPTION in V4 nicht erfüllt ist.

i) `INSERT INTO V4 VALUES (2, 2, 3, 4, 5);`
Nicht erfolgreich, da die Bedingung in V1 aufgrund der CASCADED CHECK OPTION in V4 nicht erfüllt ist.

j) `INSERT INTO V5 VALUES (1, 2, 3, 4, 6);`
Erfolgreich, da CHECK OPTION in V5 nicht definiert ist und alle Bedingungen in V4, V3, V2, und V1 erfüllt sind.

k) `INSERT INTO V5 VALUES (1, 2, 4, 4, 5);`
Nicht erfolgreich, da die Bedingung in V3 aufgrund der CASCADED CHECK OPTION in V4 nicht erfüllt ist.