DAB2 – Datenbanken 2

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Datenorganisation, Speicherung

Lehrbuch Kapitel 4.1, 5.1

Zürcher Fachhochschule

Gliederung

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

1 L	**Einführung**
4 L	**Datenorganisation Speicherung**
4 L	**Optimierung**
2 L	**Transaktionen, Recovery**
2 L	**Non-Standard Datenbanken**
1 L	**Repetition, Abschluss**

← "You are here"

Zürcher Fachhochschule

Rückblick

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Einfluss der Speicherhierarchie auf RDBMS kennen, Zugriffslücke und Caching
- Zusammenspiel Betriebssystem ↔ RDBMS
- Aspekte der physischen Speicherung von Datenbanken verstehen:
  - Blöcke, Seiten, Sätze
  - Abbildung: Relation → Dateisystem

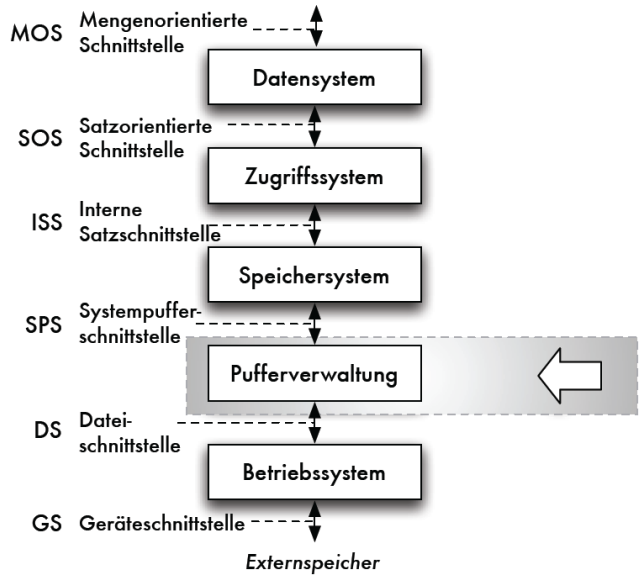
Lernziele heute

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Aufgaben der Pufferverwaltung kennen
- Wissen, was ein Index ist
- Verschiedene Arten von Zugriffsstrukturen kennen

Zürcher Fachhochschule

Pufferverwaltung



Pufferverwaltung

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Klassische Verfahren sind ungeeignet (FIFO, LRU)
- Die Pufferverwaltung muss konform sein mit Transaktionssystem (Sperren, Commit)
- Aufgaben:
  - Suchverfahren
  - Seitenersetzungsstrategien
  - Speicherzuteilung zu Datenbanktransaktionen

6

Zürcher Fachhochschule

Der Puffer ist ein Bereich im RAM. Im SQL Server kann festgelegt werden, wieviel Speicher das DBMS im RAM maximal verwenden darf. Dies begrenzt natürlich auch die maximale Grösse des Pufferbereichs. Ohne Begrenzung nutzt SQL Server mehr und mehr Speicher (welchen er bei Bedarf wieder freigeben sollte). Das kann in unglücklichen Situationen zum Absturz des Server führen. Es ist daher eine gute Idee, den maximal zulässigen Speicher zu begrenzen. Aber klar: Je mehr Speicher, desto effizienter arbeitet das DBMS.

Pufferverwaltung ist analog (aber nicht gleich) zur virtueller Speicherverwaltung; in der virtuellen Speicherverwaltung werden z.B. die Verfahren FIFO (first-in-first-out) oder LRU (least recently used) verwendet. Warum braucht es dann eine eigene Pufferverwaltung? Das hat mehrere Gründe. Zum einen unterstützt die Pufferverwaltung den Ablauf bei der Ausführung von SQL-Anweisungen besser (wir werden auf den nächsten zwei Folie ein Beispiel sehen, bei welchem die klassische Speicherverwaltung komplett versagt), zum anderen werden in einem Datenbanksystem auch parallele Transaktionen (eventuell mit mehreren Usern) ausgeführt – hierbei muss ein Kompromiss der Speicherzuteilung gefunden werden, ansonsten behindern sich die Transaktionen gegenseitig stark.

Daneben muss die Pufferverwaltung mit dem Transaktionssystem abgestimmt sein. So muss z.B. sichergestellt werden, dass Änderungen an Daten die mittels Commit bestätigt wurden, bei einem Verlust des Puffers (z.B. durch Absturz des Servers) nicht verloren gehen. Die Pufferverwaltung muss dann «irgendwie» sicherstellen, dass diese Daten auf dauerhaften Medien gesichert werden.

Aufgaben der Pufferverwaltung:

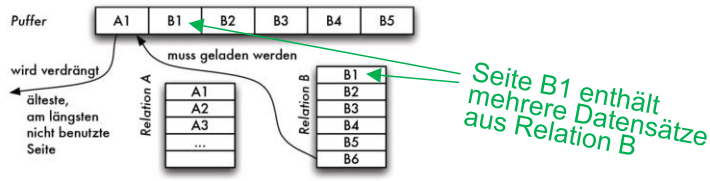
- Effiziente Suchverfahren für angeforderte Seiten (eine bestimmte Seite kann ohne Hilfsmittel durch lineares Durchsuchen ermittelt werden, oder es wird eine Zusatzstruktur wie z.B. Seitenlisten gebildet)
- Adäquate Seitenersetzungsstrategien für gefüllte Puffer (welche Seite wird aus dem Puffer entfernt, wenn der Puffer voll ist?)
- Parallele Datenbanktransaktionen verlangen geschickte Speicherzuteilung im Puffer

Pufferverwaltung

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Beispiel für ein Versagen von Cache (Fortsetzung auf nächster Folie):

- Operation:
  Join von Relationen A und B mittels Nested-Loop-Verfahren: Jedes Tupel in A (äusserer Loop) wird mit jedem Tupel in B (innerer Loop) verbunden.
- Zustand Bild:
  Das erste Tupel von A1 wurde mit den Tupeln auf den Seiten B1 bis B5 verbunden, jetzt sollen die Tupel auf B6 verbunden werden.



7

Zürcher Fachhochschule

Achtung: Im Beispiel beinhaltet jede Seite (A1, A2, A3, B1, B2, …) im Puffer jeweils *mehrere* Tupel einer bestimmten Relation.

Das Beispiel zeigt auf, wie bei einem Join zweier Tabellen (z.B.: `SELECT * FROM A JOIN B ON B.FK_A = A.PK`) bei der Verwendung eines klassischen Verfahrens, die Seitenersetzungsstrategie versagt.

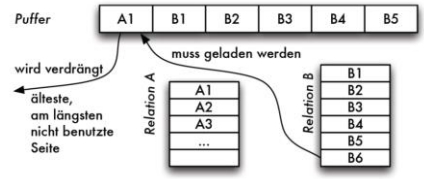
Eine Join-Operation kann mittels verschiedener Algorithmen ausgeführt werden. Das Nested-Loop-Verfahren ist ein solcher Algorithmus. In späteren Lektionen werden wir weitere, effiziente Varianten kennen lernen.

Der Puffer ist im Beispiel sechs Seiten gross.

Pufferverwaltung

Welche Seite wird verdrängt?

- FIFO:
  A_1, da älteste Seite im Puffer
- LRU:
  A_1, da diese Seite nur im ersten Schritt beim Auslesen des ersten Vergleichstupels benötigt wurde



Problem: Im nächsten Schritt wird das zweite Tupel von Seite A_1 benötigt, Seite muss wieder geladen werden (welche Seite wird ersetzt?)

Pufferverwaltung im folgenden nicht weiter betrachtet (→ Kap. 4 Buch)

8

Zürcher Fachhochschule

Die Seite A1 im Puffer wird durch die Seite B6 verdrängt und ist damit nicht mehr im Puffer verfügbar.

Der Join des ersten Tupels von A1 (diese ist noch im 'normalen' RAM, ausserhalb des Puffers verfügbar) mit den Tupeln auf Seite B6 kann zwar noch zu Ende geführt werden, aber wenn das zweite Tupel auf der Seite A1 gelesen werden soll, tritt eine katastrophale Kettenreaktion auf:

- Die Seite A1 ist ja nicht mehr im Speicher, diese muss daher zunächst in den Puffer geladen werden. Hierzu wird B1 ausgelagert (FIFO und LRU).
- Jetzt soll A1 mit dem ersten Tupel in B1 gejoined werden. B1 wurde aber eben ausgelagert. Also muss zunächst wieder B1 geladen werden. Hierzu wird B2 (FIFO und LRU) ausgelagert.
- Sind alle Tupel von B1 verarbeitet ist B2 an der Reihe, dieses wurde aber eben ausgelagert. Also muss zunächst wieder B2 geladen werden. U.s.w.

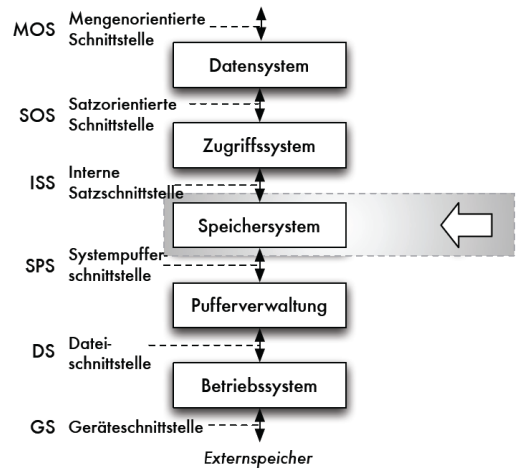
In der Folge ist die Pufferverwaltung völlig wirkungslos, da alle Seiten immer wieder geladen werden müssen. Es muss daher ständig auf externe Medien zugegriffen werden. Der Join dauert um 10er-Potenzen länger als bei geschickter Auslagerungsstrategie nötig wäre.

Speichersystem

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Das Speichersystem realisiert logische Datensätze mit Hilfe von Zeigern, speziellen Indexeinträgen und weiteren Hilfsstrukturen.

Diese Daten werden mittels blocken auf die Seiten abgebildet.



9

Zürcher Fachhochschule

- Das Speichersystem fordert über Systempufferschnittstelle SPS Seiten an und extrahiert die internen Datensätze.
- Interne Realisierung der Datensätze mit Hilfe von Zeigern, speziellen Indexeinträgen und weiteren Hilfsstrukturen.
- Die interne Satzschnittstelle ISS stellt Operationen und Zugriffs-strukturen auf die internen Sätze zur Verfügung.

Zur Erinnerung, in der **letzten Lektion** wurden verschieden Varianten zur satzorientierten Speicherung von Daten betrachtet, d.h. wie ein einzelner Datensatz auf eine Pufferseite abgebildet wird:

- satzorientiert/attributsorientiert
- Nichtspannsätze/Spannsätze
- variable/feste Länge
- fixierter/unfixierter Record (TID-Konzept)

In der **heutigen Lektion** werden wir betrachten, wie die Menge der Datensätze auf die verschiedenen Seiten abgebildet werden und wie diese wieder gefunden werden.

Klassifikation von Speichertechniken

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Vielzahl von Verfahren zur Speicherung und Zugriff auf Daten, Klassifikationskriterien:
  1. Interne Organisation der Relation selbst (Dateiorganisationsform)
  2. Weitere Zugriffsmöglichkeiten (Index, Baum, etc.) (Zugriffspfad)
  3. Attribute die unterstützt werden (Primär- / Sekundärschlüssel)
  4. Art der Zuordnung von gegebenen Attributswerten zu Datensatz-Adressen
  5. Arten von Anfragen, die durch Dateiorganisationsformen und Zugriffspfade effizient unterstützt werden können
  6. …

⇨ Zugriffsstruktur (bzw. Zugriffsverfahren):
Ohne Unterscheidung von Dateiorganisationsform und Zugriffspfad

10

Zürcher Fachhochschule

Es gibt eine Reihe von Klassifikationskriterien, wie die Daten im Speichersystem gespeichert werden können und wie auf diese Daten zugegriffen werden kann. Auf den folgenden Folien wird eine Auswahl dieser Kritieren aufgezeigt.

Achtung: Es herrscht eine Begriffsvielfalt und damit häufig auch Verwirrung!

Diese Kriterien sind für die Praxis aber wichtig, da bei der Definition der Datenstruktur jeweils eine bestimmte Variante ausgewählt werden muss. Dies hat dann direkten, zum Teil gewaltigen Einfluss auf die Performance des Datenbanksystems.

Wenn nicht unterschieden werden soll zwischen Dateiorganisationsform und Zugriffspfad spricht man von Zugriffsstruktur (bzw. Zugriffsverfahren).

Dateiorganisationsform

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Form der Speicherung:

1. Heap-Organisation: Unsortierte Speicherung
2. Sequenzielle Organisation: Sortierte Speicherung
3. Hash-Organisation: Gestreute Speicherung
4. Mehrdimensionale Dateiorganisationsformen: Speicherung in mehrdimensionalen Räumen:

Häufig: Sortierung mittels Primär- oder Fremdschlüssel

11

Zürcher Fachhochschule

Für mehrdimensionalen Dateiorganisationformen existieren verschiedene Algorithmen. Auf diese wird in der Vorlesung nicht eingegangen (vergleiche Literatur).

Sehr häufig wird die Datei nach dem Primärschlüssel oder nach dem Fremdschlüssel sequentiell sortiert (geclustert). Die Sortierung nach Fremdschlüssel macht dann Sinn, wenn auf die Daten häufiger über einen Fremdschlüssel als über den Primärschlüssel zugegriffen wird (z.B. für die Auftragspositionen, denn diese werden vermutlich am häufigsten für einen bestimmten Auftrag gesucht; so liegen die Auftragspositionen eines Auftrages mit etwas Glück alle auf der selben Seite und können zusammen eingelesen werden).

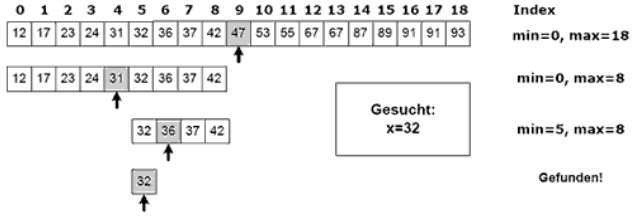
Im SQL Server werden die Daten per Default nach dem Primärschlüssel sortiert gespeichert/geclustert. Wird ein künstlicher Primärschlüssel verwendet, muss auch überlegt werden, ob nicht eine unsortierte Speicherung besser ist.

Ziel der Dateiorganisationform: Performanzoptimierung:

- Schreib-/Lesezugriffe auf Sekundärspeicher minimieren
- Gesuchte Daten rasch finden, einfügen und löschen

Beispiel Suchmöglichkeiten (vergleiche Vorlesung Algorithmen und Datenstrukturen):

- Heap: Sequentielle Suche (Worst Case: O(n), n = Anzahl Datensätze)
- Sequentielle Organisation: Binäre Suche (Worst Case: $O(\log_2 n)$ Zugriffe) (optimale Suche)
  Beispiel zur binären Suche:



Dateiorganisationsform

- Aber: maximal ein Kriterium zur Sortierung
- Idee: Verschiedene 'Verzeichnisse' für unterschiedliche Zugriffe unterhalten → Zugriffspfade (z.B. Index)

Zugriffspfad

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Der Zugriffspfad ist eine über die Dateiorganisationsform hinausgehende, zusätzliche Zugriffsstruktur (z.B. Indexdatei, Bäume).
- Der Zugriffspfad ist eine selbständige Datenstruktur und ist redundant
- Zugriffspfade sind das weitaus wichtigste Hilfsmittel zur Leistungssteigerung
- Abwägung:
  - Nutzen: Zugriffspfade können Abfragen beschleunigen
  - Kosten: Speicherplatz, Indexpflege bei DML, Indexunterhaltsarbeiten, Logging, Locking, etc.

13

Zürcher Fachhochschule

Daten können physisch nur nach maximal einem Kriterium sortiert gespeichert werden. Für weitere, andere Zugriffskriterien müssen daher zusätzliche Hilfsstrukturen/Zugriffspfade geschaffen werden. Diese zusätzlichen Zugriffspfade sind in der Praxis ein extrem wichtiges Mittel, um die Zugriffszeiten auf Daten zu optimieren. Beispiele:

- Wird ein Auftrag nebst der Auftragsnummer (Primärschlüssel) häufig auch mittels Auftragsdatum gesucht, so kann ein Zugriffspfad für das Auftragsdatum die Zugriffe massiv beschleunigen.
- Wird eine Tabelle häufig durch ein bestimmtes Attribut in Joins eingebunden (häufig ein Fremdschlüssel), kann der Join massiv beschleunigt werden. Fremdschlüssel sind daher gute, weitere Kandidaten zur Bildung von Zugriffspfaden.

Natürlich verursachen Zugriffpfade auch ‘Kosten’. Diese Kosten müssen mit dem Nutzen abgewogen werden (SQL Server stellt hierzu Ratgeber/Wizards zur Verfügung welche statische Daten der entsprechenden Datenbank verwenden).

Gewisse Anfragen können allein mit Hilfe eines Zugriffspfades beantwortet werden (d.h. ohne Zugriff auf die gespeicherten Tupel selbst). Diese sind dann extrem effizient. Allenfalls lohnt es sich für häufige Zugriffe auf wenige Daten sogar, speziell hierfür einen Zugriffspfad anzulegen.

Die Begriffe Zugriffspfad und Index werden häufig als Synonyme verwendet, obgleich ein Zugriffspfad z.B. auch als B-Baum implementiert sein kann.

Wenn möglich wird die Dateiorganisationsform durch den Zugriffspfad genutzt (z.B. wenn ein Index und die sequentielle Organisation das selbe Kriterium anwenden = indexsequentielle Dateiorganisationsform).

Zugriffspfad (Demo)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

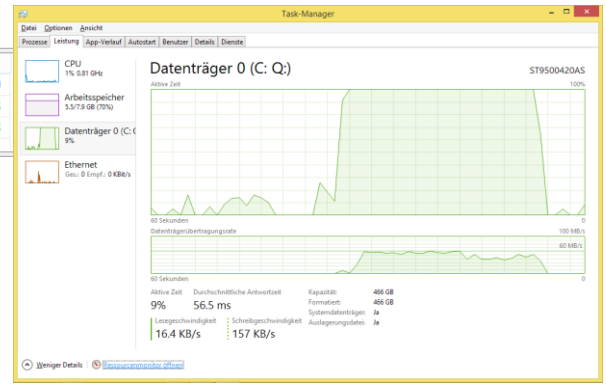
In SQL Server Syntax

```
-- Suche ohne Index
DROP INDEX [Passagier_IDx] ON  [FlughafenDB].[dbo].[Buchung]
DBCC DROPCLEANBUFFERS WITH NO_INFOMSGS
SELECT 'Start ohne Index:', GETDATE()
SELECT [Buchung_ID],[Flug_ID],[Sitzplatz],[Passagier_ID],[Preis]
   FROM [FlughafenDB].[dbo].[Buchung] WHERE Passagier_ID =  22444
SELECT 'Ende ohne Index:', GETDATE()
```

	(Kein Spaltenname)	(Kein Spaltenname)
1	Start ohne Index:	2015-10-01 15:44:12.100

	Buchung_ID	Flug_ID	Sitzplatz	Passagier_ID	Preis
1	54	17821	6A	22444	291.30
2	19172	127549	11A	22444	206.13
3	44631	113522	12E	22444	219.08
4	237639	150124	6A	22444	99.92

	(Kein Spaltenname)	(Kein Spaltenname)
1	Ende ohne Index:	2015-10-01 15:44:38.660



14

Zürcher Fachhochschule

DBCC = *D*ata*b*ase *C*onsole *C*ommand

DBCC DROPCLEANBUFFERS WITH NO_INFOMSGS:
Removes all clean buffers from the buffer pool, and columnstore objects from the columnstore object pool.

Zur Erinnerung: Bei Columnstore werden die Daten attributsweise gespeichert (und nicht nach Tupel).

Zugriffspfad (Demo)

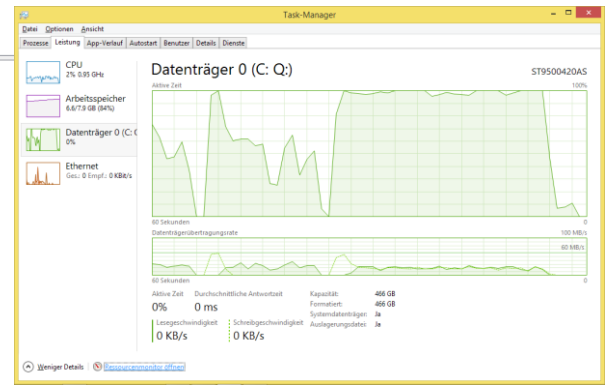
School of Engineering
InIT Institut für angewandte Informationstechnologie

In SQL Server Syntax

```
-- Index erstellen
DROP INDEX [Passagier_IDx] ON  [FlughafenDB].[dbo].[Buchung]
DBCC DROPCLEANBUFFERS WITH NO_INFOMSGS
SELECT 'Start erzeuge Index:', GETDATE()
CREATE NONCLUSTERED INDEX [Passagier_IDx] ON [dbo].[Buchung] ([Passagier_ID] ASC)
   WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
   DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
SELECT 'Ende erzeuge Index:', GETDATE()
```

	(Kein Spaltenname)	(Kein Spaltenname)
1	Start erzeuge Index:	2015-10-01 15:50:59.127

	(Kein Spaltenname)	(Kein Spaltenname)
1	Ende erzeuge Index:	2015-10-01 15:52:32.743



Zugriffspfad (Demo)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

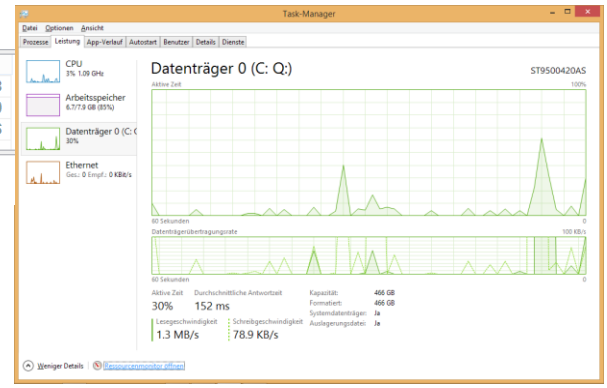
```
-- Suche mit Index
DBCC DROPCLEANBUFFERS WITH NO_INFOMSGS
SELECT 'Start mit Index:', GETDATE()
SELECT [Buchung_ID],[Flug_ID],[Sitzplatz],[Passagier_ID],[Preis]
   FROM [FlughafenDB].[dbo].[Buchung] WHERE Flug_ID = 133530
SELECT 'Ende mit Index:', GETDATE()
```

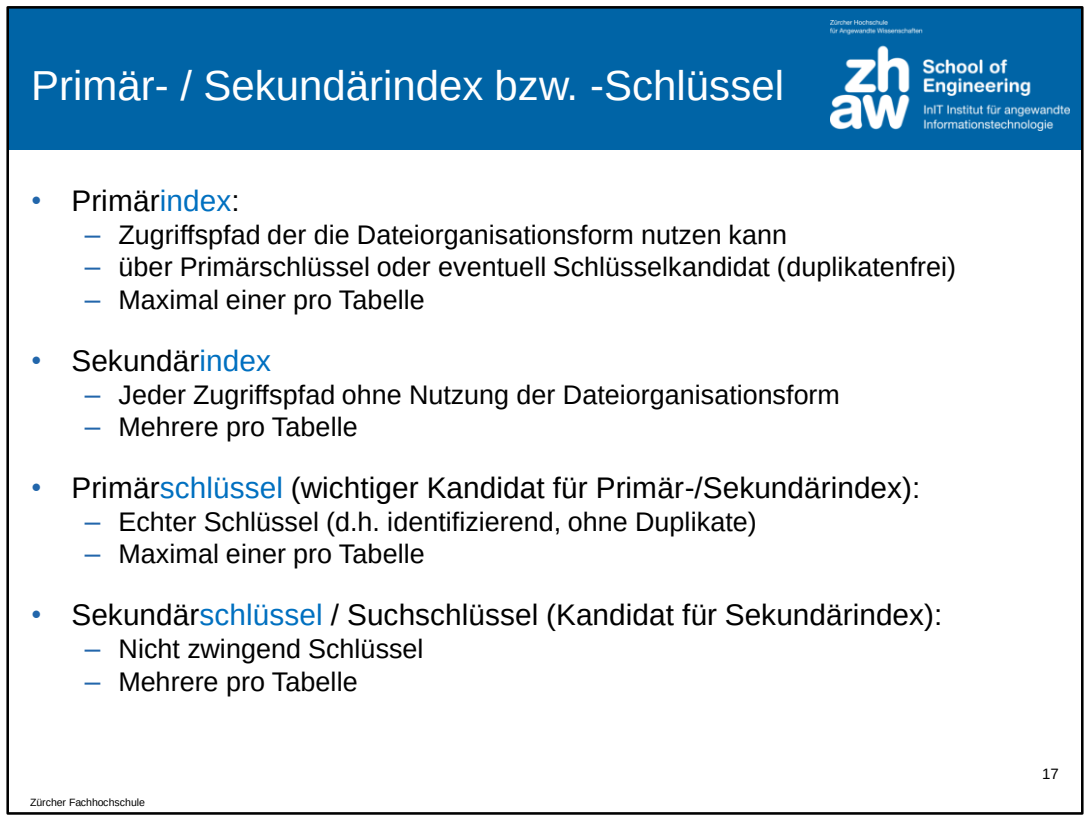
In SQL Server Syntax

	(Kein Spaltenname)	(Kein Spaltenname)
1	Start mit Index:	2015-10-01 15:55:25.007

	Buchung_ID	Flug_ID	Sitzplatz	Passagier_ID	Preis
1	23873827	133530	18F	21743	106.98
2	23873826	133530	19B	7712	197.20
3	23873879	133530	1D	3380	476.36

	(Kein Spaltenname)	(Kein Spaltenname)
1	Ende mit Index:	2015-10-01 15:55:25.073





Die Folie zeigt die Begriffe Primär-, Sekundar-Index und –Schlüssel.

Während Indexe eine physische Hilfsstruktur von Datenbanksystemen bezeichnen, stammt der Begriffe Primärschlüssel aus der ‘relationalen Modellierung’. Der Begriff Sekundärschlüssel stellt einen Suchschlüssel dar, welcher als beliebige Kombination von Attributen als Suchkriterium zum Auffinden von Datensätzen in einem Datenbanksystem verwendet wird (hat eigentlich nichts mit einem Schlüssel zu tun).

In aller Regel wird für Primärschlüssel natürlich ein eindeutiger Primärindex gebildet (damit wird auch dessen Eindeutigkeit garantiert). Bei Sekundärschlüsseln wird nach einer Kosten/Nutzen-Abwägung ein Sekundärindex angelegt (was auch im laufenden Betrieb erfolgen kann).

Leider tritt der Begriff Primärindex in der Praxis in zwei Varianten auf:

1. Index welcher über den Primärschlüssel gebildet wird (die Dateiorganisationsform kann dann unabhängig vom Primärschlüssel sein)
2. Index welcher die Dateiorganisationsform nutzt (und meist über den Primärschlüssel ‘sortiert’ ist)

Zugriffspfad: Index

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Die Indexdatei hat Einträge (K, K↑):

- K der Wert eines Primär- oder Sekundärschlüssels
- K↑ der Datensatz oder Verweis auf Datensatz

K↑ kann von folgenden Formen sein:

- K↑ ist Datensatz selbst: "Zugriffspfad" wird zur Dateiorganisationsform
- K↑ ist Adresse genau eines Datensatz:
  - Primärschlüssel (K kommt nur einmal vor)
  - Sekundärschlüssel mit (K, $K\uparrow_1$) , …, (K, $K\uparrow_n$) für denselben Zugriffsattributswert K
- K↑ ist Liste von Datensatzadressen: Sekundärschlüssel (K, $K\uparrow_1$, …, $K\uparrow_n$)
  Nachteil: variable Länge der Indexeinträge

18

Zürcher Fachhochschule

Statt Datensatzadressen (TID-Konzept) können für K↑ auch Seitenadressen verwendet werden. Dann muss innerhalb der Seite sequentiell gesucht werden.

Index - Beispiel

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Index

PersNr	Speicher-adresse
101	3000
102	3200
106	4800
107	4400
108	4600
109	3400
115	4000

physische Speicheradressen der Tupel

	PersNr	Vor-name	Nach-name	Abt.	Strasse	PLZ	Ort	Eintritt	..	..	..	..	..
3000	101	Urs	Maurer	31	....								
3200	102	Peer	Anders	..									
3400	109	Bea	Stock										
4000	115	Hans	Glück										
4400	107	Martin	Rother										
4600	108	Kevin	Miles										
4800	106	..											

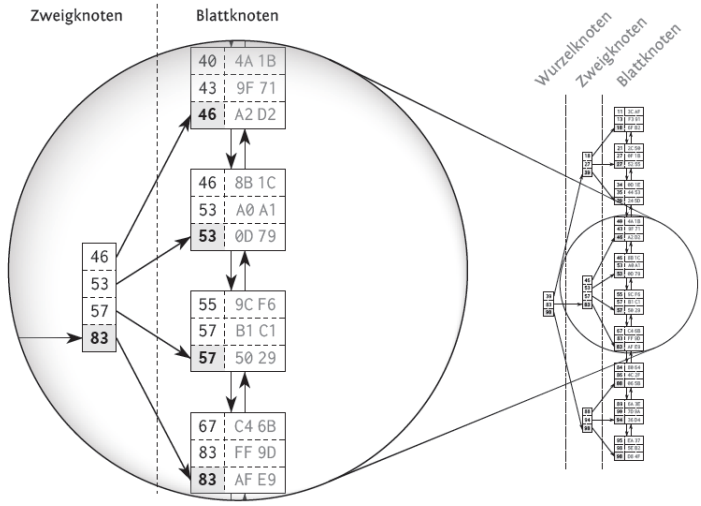
19

Zürcher Fachhochschule

Index - Grundaufbau

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Typischer Grundaufbau eines mehrstufigen Index (Varianten & Details siehe später):



20

Zürcher Fachhochschule

Da ein Index selbst sehr gross werden kann, liegt die Idee nahe, für den Index selbst wieder einen Index anzulegen. Dadurch entsteht ein mehrstufiger Index (rekursiv).

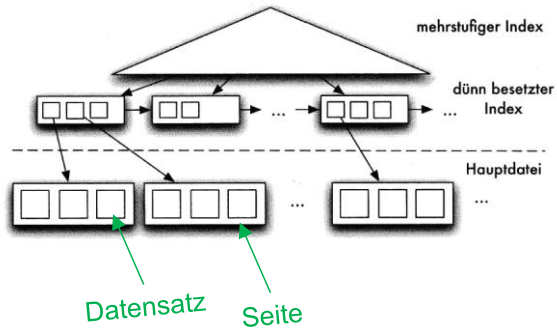
Diese Zugriffsmethoden wurden bereits in den 1960er Jahren entwickelt (ISAM) und wurde für Datenbanksysteme und in COBOL eingesetzt. ISAM-Dateien werden aber auch noch heute eingesetzt, z.B. in der MyISAM-Engine von MySQL.

Die Darstellung erinnert schon etwas an einen B-Baum (werden wir in der nächsten Lektion genauer betrachten). Im Gegensatz zum B-Baum ist aber die Anzahl Ebenen statisch vorgegeben.

Dünn- vs. dichtbesetzte Indexe

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Dünn besetzter Index: Wenn die interne Relation nach den Zugriffsattributen sortiert ist, dann reicht im Index ein Eintrag pro Seite (= dünn).



21

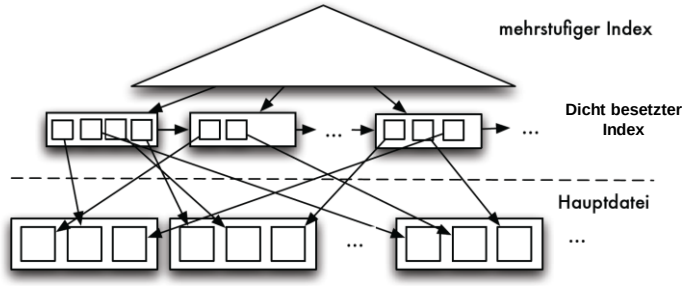
Zürcher Fachhochschule

Der Index verweist mit (K_1, $K_1\uparrow$) auf den Anfang der Seite, der nächste Indexeintrag ist dann (K_2, $K_2\uparrow$) (nächste Seite).

Datensätze mit Zugriffsattributwerten K_x mit $K_1 \leq K_x < K_2$ sind auf Seite von $K_1\uparrow$ zu finden.

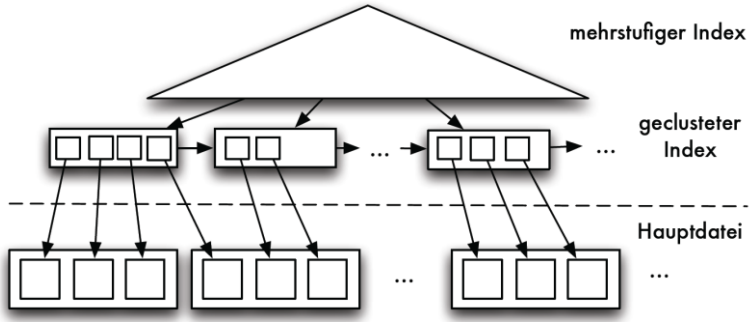
Dünn- vs. dichtbesetzte Indexe

- Dicht besetzter Index: Für jeden Datensatz (d.h. Zugriffsattributwerte) der Relation ist ein Eintrag in der Indexdatei vorhanden
- Sekundärindexe sind immer dicht besetzt!
- Primärindex kann dichtbesetzter Index sein, z.B. wenn die Datei-organisationsform eine Heap-Datei ist.



Geclusterte vs. nicht-geclusterte Indexe

- Geclusterter Index: Der Index ist gleich sortiert wie interne Relation.
- Bsp.: Interne Relation `KUNDEN` nach Kundennummern sortiert
  → Indexdatei über dem Attribut `KNr` üblicherweise geclustert



23

Zürcher Fachhochschule

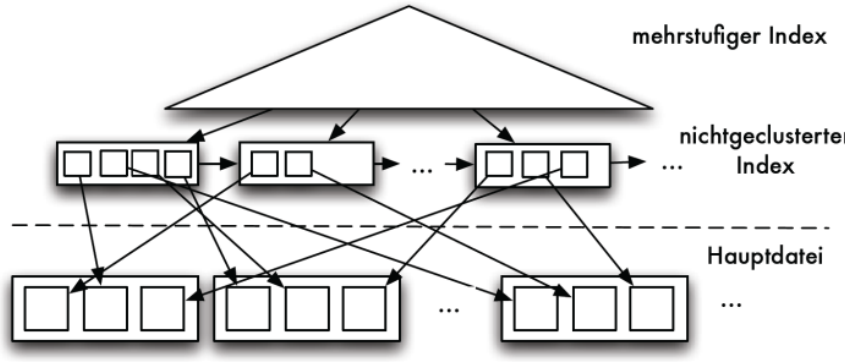
Das Bild zeigt einen dicht-besetzten, geclusterten Index. Jeder dünn-besetzte Index muss auch geclustert sein, aber nicht umgekehrt.

Der geclusterte Index geeignet sich gut für häufige Bereichsanfragen. Z.B. wenn nach Namen innerhalb eines Bereichs gesucht wird (z.B. alle Müller, geclustert nach Name), oder wenn bei Fremdschlüsseln häufig alle abhängigen Tupel gesucht werden (z.B. alle Auftragspositionen des Auftrages 001, geclustert nach Fremdschlüssel).

Geclusterte vs. nicht-geclusterte Indexe

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Nicht-geclusterter Index: anders sortiert als die interne Relation.
- Bsp.: Über `Name` ein Sekundärindex, die Datei selbst nach `KNr` sortiert



24

Zürcher Fachhochschule

Das Bild zeigt einen dicht-besetzten, nicht-geclusterten Index. Sekundärindexe können nur dicht-besetzt und nicht-geclustert sein.

Schlüsselzugriff vs. -transformation

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Zugriffsverfahren kann man auch nach der Art der Umsetzung von Zugriffsattributwerten auf Tupeladressen unterscheiden:

- Schlüsselzugriff: Zuordnung von Werten zu Adressen (z.B. indexsequentielle Organisation, B-Bäume, …).
- Schlüsseltransformation:
  Tupeladresse wird aufgrund einer Formel berechnet (z.B. Hashing).

25

Zürcher Fachhochschule

Während die bisherigen Folien die Klassifikation der Speichertechniken (Folie Seite 10) gemäss Punkt 1 bis 3 aufgezeigt haben, wird in den folgenden Folien auf Punkt 4 eingegangen: Wie wird aus Attributswerten eine Datensatz-Adresse abgeleitet. Diese und die folgenden Folien zeigen die Klassifikation nach:

- Schlüsselzugriff versus Schlüsseltransformation
- Ein-Attribut versus Mehr-Attribut-Indexe

Schlüsselzugriff: Zuordnung von Primär- oder Sekundärschlüssel-Werten zu Adressen in einer Hilfsstruktur (z.B. indexsequentielle Organisation, B-Bäume, …). Attributswerte und Adressen sind ein einer Datei gespeichert.

Schlüsseltransformation: Tupeladresse wird aufgrund einer Formel aus Primär- oder Sekundärschlüsselwerten berechnet (z.B. Hashing). Es wird nur die Berechnungsvorschrift gespeichert (Überlaufbehandlung).

Ein-Attribut vs. Mehr-Attribut-Indexe

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Ein-Attribut-Index (non-composite index):
  Zugriffspfad über einem einzigen Attribut.
  Bsp.: Indexieren von Name
- Mehr-Attribut-Index (composite index):
  Zugriffspfad über mehreren Attributen.
  Bsp.: Indexieren von Name & PLZ.

26

Zürcher Fachhochschule

Die Wahl, ob zwei Ein-Attribut-Indexe (Name, PLZ) oder ein zusammengesetzter Index (Name & PLZ, oder PLZ & Name) verwendet werden soll, hängt von den Abfragen ab.

Anforderungen an Speichertechniken

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Effizienz beim Einzelzugriff (Schlüsselsuche beim Primärindex)
- Effizienz beim Mehrfachzugriff (Schlüsselsuche bei Sekundärindexen)
- Effizienz beim sequentiellem Durchlauf (Sortierung, geclusterter Index)
- Clustering der Datensätze (auch unterschiedlicher Relationen)
- Unterstützung für verschiedene Anfragetypen: exact-match, partial-match, range queries (Bereichsanfragen)
- Dynamisches Verhalten, d.h. Anpassung an sich ändernde Datenmengen

27

Zürcher Fachhochschule

Zum Abschluss wird in den folgenden Folien auf Punkt 5 von Folie 10 eingegangen: Welche Arten von Zugriffen, Dateiorganisationsformen werden wie effizient unterstützt. Die Folie zeigt eine nicht abschliessende Aufzählung von möglichen Arten.

Beim Clustering der Datensätze sind wir bisher davon ausgegangen, dass die Tupel einer einzelnen Tabelle geclustert werden. Es ist beim Clustering in bestimmten DBMS aber auch möglich, Regeln aufzustellen, so das Tupel aus mehreren Tabellen geclustert werden (z.B. so, dass die Tupel der Auftragspositionen beim Tupel des Auftrags zu liegen kommen).

Klassifikation nach Eignung

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Wann eignet sich welches Zugriffsverfahren:

Indexart	Dimensionalität	Verfahrensart	Verfahren
kein Index	eindimensional	direkt	Heap sequenziell
Primärindex	eindimensional	Schlüsselzugriff	indexsequenziell mehrstufig indexsequenziell B-Baum, B^+-Baum
		Schlüsseltransf.	dynamische und statische Hashverfahren
	mehrdimensional	Schlüsselzugriff	KdB-Baum Grid-Files
		Schlüsseltransf.	mehrdimensionales Hashen
Sekundärindex	eindimensional	Schlüsselzugriff	indexiert-nichtsequenziell mehrstufig indexiert-nichtseq. B-Baum, B^+-Baum-Varianten
		Schlüsseltransf.	Hashverfahren
	mehrdimensional	Schlüsselzugriff	KdB-Baum Grid-Files
		Schlüsseltransf.	mehrdimensionales Hashen

28

Zürcher Fachhochschule

Die Folie zeigt, welches Verfahren (Algorithmus) in Abhängigkeit

- der gewünschten Indexart (Primärindex=eindeutiger, duplikatenfreier Schlüssel oder Sekundärindex),
- Der Dimensionalität (erfolgt die Abbildung mittels mehrdimensionaler Transformation? Bitte hierfür Literatur konsultieren) und
- der Verfahrensart (direkt, Schlüssel-Zugriff oder –Transformation)

geeignet ist. Wir werden die wichtigsten dieser Verfahren in den kommenden Lektionen untersuchen.

In konkreten, relationalen Datenbanksystemen stehen verschiedene Verfahren zur Indexierung zur Verfügung. So kennt der SQL Server (Version 2016) 12 verschiedene Indextypen. Darunter z.B. spezielle Indextypen für spezielle Datentypen:

- geografische Daten (spatial Index)
- XML-Daten
- Volltext Suche
- Berechnete Attribute

Oder spezifische Ergänzungen zu Indexen:

- Eindeutiger Index (unique)
- Um Attribute erweiterter Index (ohne dass diese indexiert werden), so dass Queries, die nur diese Attribute verwenden, ausschliesslich auf dem Index ausgeführt werden können
- Memory-optimierte Indexe
- Gefilterte Indexe
- Usw.

Für den praktischen Einsatz ist es wichtig, die jeweiligen Varianten des DBMS zu kennen und den ‘optimalen’ Kompromiss zu suchen. Hierzu stellt das DBMS auch statistische Informationen zur Verfügung, um die Annahmen im laufenden Betrieb zu überprüfen.

Ausblick

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Das nächste Mal: Zugriffsstrukturen
- Stoff dieser Vorlesung: Lehrbuch Kapitel 4 bis 4.1 und 5 bis bis 5.1

Zürcher Fachhochschule