DAB2 – Datenbanken 2

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

Lektion 3: Datenorganisation, Speicherung

Lehrbuch Kapitel 3.1, 3.4, 3.5

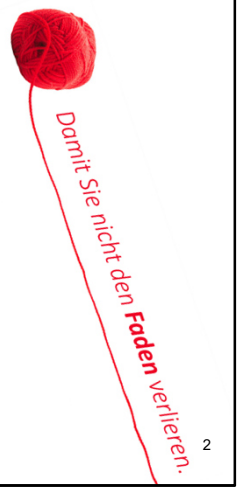
Zürcher Fachhochschule

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"

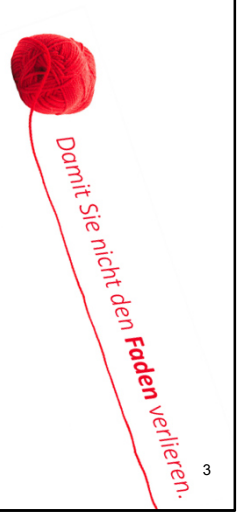


2

Zürcher Fachhochschule

Rückblick

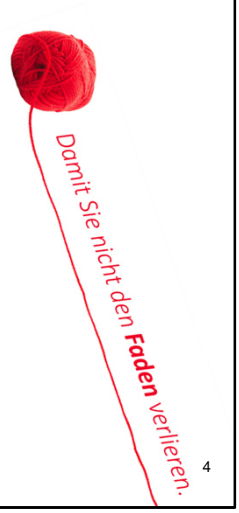
- 5-Ebenen Systemarchitektur:
  - Datensystem
  - Zugriffssystem
  - Speichersystem
  - Pufferverwaltung
  - Betriebssystem
- Speicherhierarchie:
  - Primärspeicher
  - Sekundärspeicher
  - Tertiärspeicher



Lernziele heute

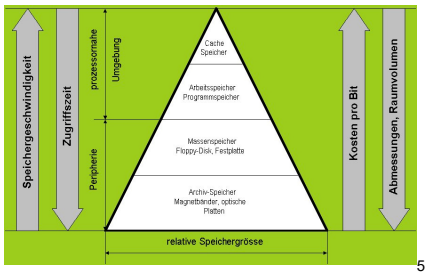
Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

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



Speicherhierarchie – Eigenschaften

- Ebene x (z.B. Hauptspeicher) hat wesentlich schnellere Speicher- und Zugriffszeit als Ebene x + 1 (z.B. Sekundärspeicher)
- Kosten pro Speicherplatz auf Ebene x weitaus höher als auf Ebene x + 1
- Kapazität auf Ebene x weitaus geringer als auf Ebene x + 1
- Lebensdauer der Daten auf Ebene x + 1 weitaus grösser als auf Ebene x

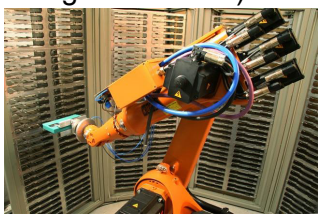


5

Speicherhierarchie – verfeinert

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

1. Prozessor mit Registern (1 cycle)
2. Cache (L1: 1-5 cycles, L2: 5-20 cycles, L3… SRAM: 5 – 10 ns)
3. Hauptspeicher (40 – 100 cyles, DRAM: 40 – 60ns)
4. Sekundärspeicher:
   Harddisk (20'000'000 cycles, 8 – 10 ms)
   SSD (200'000 cycles , 0.1 – 0.2 ms)
5. Nearline-Tertiärspeicher (automatische Bereitstellung der Medien) (Sekunden)
6. Offline-Tertiärspeicher (*manuelle* Bereitstellung der Medien, per Hand)



6

Zürcher Fachhochschule

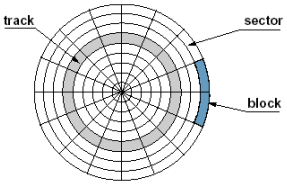
Die Angegebenen Zeiten beziehen sich auf die Zugriffzeiten – nicht auf den Datendurchsatz!

Der SQL Server hat auch eine Erweiterung, in welche SSD als Teil des Puffers verwendet werden können, dadurch kann der Zugriff auf Harddisk weiter reduziert werden.

Speicherhierarchie – Magnetplatten

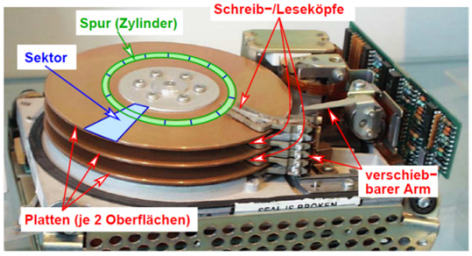
Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Dominantes Speichermedium für Datenbanken



Begriffe:

- Platte, Plattenstapel (Disk)
- Spur (Track)
- Zylinder (übereinander liegende Spuren)
- Sektor (kleinster Teil einer Spur = Spur-Sektor, oder Platten-Sektor)
- Block (Cluster)
  (mehrere Sektoren einer Spur hintereinander)



Aufgrund mechanischer Komponenten sind Magnetplatten «langsam»

7

Zürcher Fachhochschule

Aufgrund der geringen Kosten von Magnetplatten gegenüber SSD und RAM werden diese noch bevorzugt eingesetzt. Die sinkenden Kosten von RAM und SSD ermöglichen es aber, neue Wege zu gehen. So sind in den letzten Jahren z.B. vermehrt In-Memory-Datenbanken auf dem Markt beworben worden. Die bestehenden Datenbanksysteme wurden natürlich auch entsprechend erweitert, so dass diese auch als In-Memory-Datenbanken betrieben werden können.

Der Begriff des Sektors wird leider in zwei verschiedenen Zusammenhängen verwendet. Er bezieht sich entweder auf einen Ausschnitt eines Sektors (Spur-Sektor) oder auf einen Ausschnitt einer ganzen Platte (Platten-Sektor).

Speicherhierarchie – Speichermedien

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

Grössenordnungen:

Speicherart	Typische Zugriffszeit	Typische Kapazität
Cache (L1, L2, …)	5 - 10 ns	256 kByte (L2), 8 MByte (L3)
Hauptspeicher	40 - 60 ns	Bis 512 GByte
	Zugriffslücke ~10^4 bis ~10^5	
SSD	0.1 - 0.2 ms	Mehrere TByte
Magnetplatten	8 – 10 ms	Bis 14 TByte
Platten-Array	10 ms	TByte- bis PByte-Bereich

Für persistente Datenspeicherung wichtigstes Medium: Harddisk
(zunehmend auch SSD, sind aber noch immer teurer)

8

Zürcher Fachhochschule

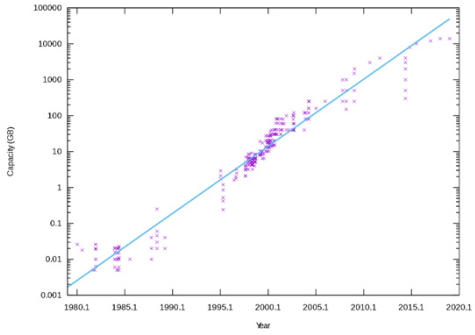
Da Daten von Datenbanksystemen in aller Regel persistent gespeichert werden sollen, müssen diese früher oder später auf Magnetplatten/SSD abgelegt werden, respektive von diesen gelesen werden. Dieser Zugriff dauert aber um 10er-Potenzen länger, als der Zugriff auf flüchtigen Speicher. Es ist für die Performance von Datenbanken daher von absolut zentraler Bedeutung, diesen Zugriff so effizient wie nur möglich zu gestalten. Dieses Problem wird mit dem Begriff «Zugriffslücke» beschrieben.

Speicherhierarchie – Zugriffslücke

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

- Beobachtungen:
  - Geschwindigkeit Magnetplatten: pro Jahr ca. 7% schneller
  - Geschwindigkeit Prozessor: pro Jahr ca. 70% schneller
  - Zugriffslücke zwischen Hauptspeicher und Magnetplattenspeicher beträgt 10^5 (diese Zugriffslücke war in den letzten Jahren im Wesentlichen konstant)
  - Neuere Speichermedien (SSD) sind dabei, diese Lücke um einen Faktor 10 bis 100 zu verringern

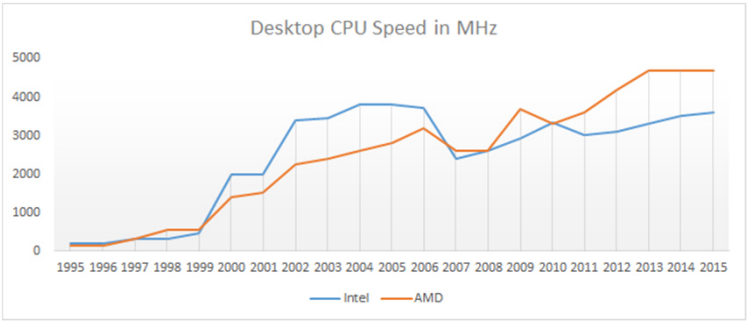
→ Die Zugriffslücke ist und bleibt aber ein ernstes Problem!



9

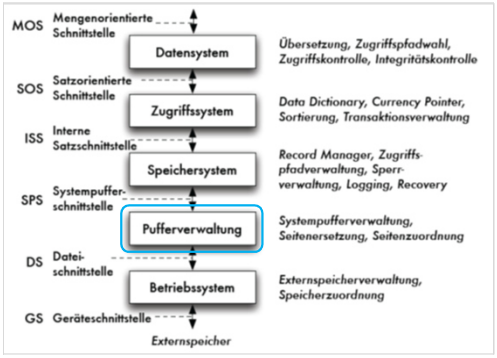
Zürcher Fachhochschule

Seit 2004 ist die Taktfrequenz von CPUs mehr oder weniger konstant! Dafür hat die Anzahl Cores je CPU zugenommen. Da Datenbanksysteme diese Cores gut nutzen können, hat deren Leistung trotz "konstanter" CPU-Taktfrequenz zugenommen. Leider sind die Preismodelle bestimmter Hersteller abhängig von der Anzahl verwendeter Cores.



Speicherhierarchie – Zugriffslücke

- Für RDBMS wichtig:
  - Primärspeicher (SRAM und DRAM)
  - UND schneller Sekundärspeicher (für Sicherstellung der Persistenz)
  - denn der Speicherbedarf und die verarbeiteten Datenmengen sind gross und wachsen weiter rasch an
- Zwingend: Konzept des Caches zur Überwindung der Zugriffslücke (Pufferverwaltung); hilft aber nur, wenn zeitliche und/oder örtliche Lokalität gegeben ist (siehe übernächste Folie)



Speicherhierarchie – Caching

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

- Mit dem Begriff Cache (kæʃ) wird üblicherweise der Speicher gemeint, der mittels schneller Halbleiterspeichertechnologie Hauptspeicherinhalte für den Prozessor bereitstellt, also zwischen RAM und Prozessor angesiedelt ist (kann auch mehrstufig ausgeführt sein, sog. L1-, L2-, L3-Cache, …).
- Insbesondere für RDBMS auch wichtig: Plattenspeicher-Cache. Bereich im Hauptspeicher, in dem Ausschnitte des Sekundärspeichers zwischengespeichert werden. Im Folgenden als Puffer bezeichnet (implementiert in der Pufferverwaltung).
- Daneben gibt es weitere Cache-Speicher, beispielsweise als Bestandteil eines Plattenkontrollers.

Caching wird natürlich nicht nur im Pufferverwalter eingesetzt, sondern wird auf allen Ebenen des 5-Schichtenmodells eingesetzt. Entscheidend für das DBMS ist aber vor allem der Puffer zwischen Primär- und Sekundärspeicher. Dieser Puffer muss an die spezifischen Bedürfnisse des DBMS angepasst sein, da das normale Caching des Betriebssystems ungeeignet ist.

Übrigens: Cache und Cash wird identisch ausgesprochen.

Speicherhierarchie – Caching

- Caching (egal auf welcher Ebene) ist nutzlos, wenn bei jeder Anforderung von Daten immer neue Daten benötigt werden. In vielen Anwendungs-fällen wird aber mehrmals auf dieselben oder 'benachbarte' Daten zugegriffen → Lokalität
- Zeitliche Lokalität: In kurzer Zeit wird wiederholt auf die gleichen Daten zugegriffen
- Räumliche Lokalität (Data Cluster/Clustering): Zusammen angefragte Daten sind auf dem Sekundärspeicher räumlich eng zusammen gespeichert (→ Reduktion der Ein-/Ausgabeoperationen oder zumindest der Positionierungszeit bei Magnetplatten). Die räumliche Nähe wird durch die Nähe im physischen 'Adressraum' repräsentiert.

12

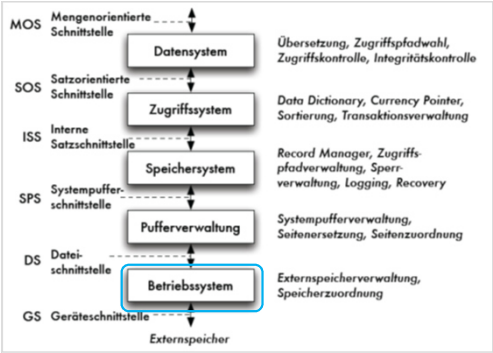
Zürcher Fachhochschule

Während man intuitiv versteht, dass das Betriebssystem die räumliche Lokalität nicht nutzen kann – es weiss ja nicht, welche Daten des Datenbanksystems häufig zusammen angefragt werden (was man im übrigen dem Datenbanksystem auch explizit mitteilen muss = Clustering) – hat man zunächst den Eindruck, dass die zeitliche Lokalität auch durch das Betriebssystem genutzt werden kann. Das Datenbanksystem hat aber auch hier vertiefte Kenntnisse wann welche Daten benötigt werden. So weiss es zum Beispiel bei einem SQL-Join, dass über die Daten einer bestimmten Tabelle mehrfach iteriert wird und diese Daten daher von Vorteil im Puffer liegen bleiben.

Betriebssystem

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

- Unterste Ebene in der 5-Ebenen-Architektur; verwaltet den Sekundärspeicher (Harddisk, SSD)
- Betriebssystem abstrahiert von Speichermedien
- Dateischnittstelle präsentiert eine Folge von Blöcken



Auf den nächsten Folien wird das Zusammenspiel des Betriebssystems mit dem RDBMS betrachtet.

Betriebssystem – Aufgaben

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

- Zuordnung oder Freigabe von Speicherplatz. Holen oder Speichern von Seiteninhalten
- Zuordnung möglichst so, dass logisch aufeinanderfolgende Datenbereiche (etwa einer Relation) auch möglichst in aufeinanderfolgenden Blöcken der Platte gespeichert werden
- Nach vielen Update-Operationen: → Reorganisation nötig
- Wie werden Relationen/Tupel/Attribute in diese Blöcke abgebildet?

Konz. Ebene	Dateisystem
Relationen	Phys. Dateien
Tupel	Seiten / Blöcke
Attributwerte	Bytes

14

Zürcher Fachhochschule

Bei Oracle und SQL-Server werden aufeinanderfolgende Datenbereiche in Extents (Seiten) abgelegt. Dabei werden mehrere Blöcke zusammengefasst. Im SQL-Server werden z.B. immer 8 physikalische, aufeinanderfolgende Blöcke zu einem Extent zusammengefasst (64 KB). Dadurch dass die Blöcke hintereinander liegen, kann eine grössere Datenmenge schnell gelesen/geschrieben werden.

Abbildung auf Dateisystem

«Arbeitsteilung» zwischen RDBMS und Betriebssystem nötig. Grundsätzliche **Varianten** der Arbeitsteilung:

1. RDBMS speichert jede Relation oder jeden Zugriffspfad in genau einer Betriebssystem-Datei (z.B. MySQL)
2. Eine oder mehrere Betriebssystem-Dateien; RDBMS verwaltet Relationen und Zugriffspfade selbst innerhalb dieser Dateien (z.B. SQL Server, Oracle)
3. RDBMS steuert selbst die Magnetplatten an und arbeitet mit den Blöcken in ihrer Ursprungsform (sog. 'raw device', eigenes Dateisystem) oder stellt einen 'schlanken' Volume-Manager zur Verfügung (z.B. Oracle ASM). Gründe:
   - Betriebssystemunabhängigkeit
   - 32-Bit-Betriebssysteme: Dateigrösse maximal 4 GByte
   - Betriebssystemseitige Pufferverwaltung von Blöcken des Sekundärspeichers im Hauptspeicher genügt den Anforderungen des Datenbanksystems nicht
   - …

Abbildung auf Dateisystem

Die Abbildung kann noch über eine zusätzliche interne Ebene erfolgen (Zwischenschritt der Abbildung / logische Dateien):

- Relationen und Zugriffpfade je in separaten logischen Dateien (übliche Form)
- Relation und zugehöriger Zugriffpfad in einer gemeinsamen logischen Datei
- Mehrere Relationen in einer logischen Datei (Cluster-Speicherung)

Konz. Ebene	Interne Ebene	Dateisystem
Relationen	Log. Dateien	Phys. Dateien
Tupel	Datensätze	Seiten / Blöcke
Attributwerte	Felder	Bytes

Die logische Dateien können dann wieder in eine oder mehrere physische Dateien abgebildet werden.

16

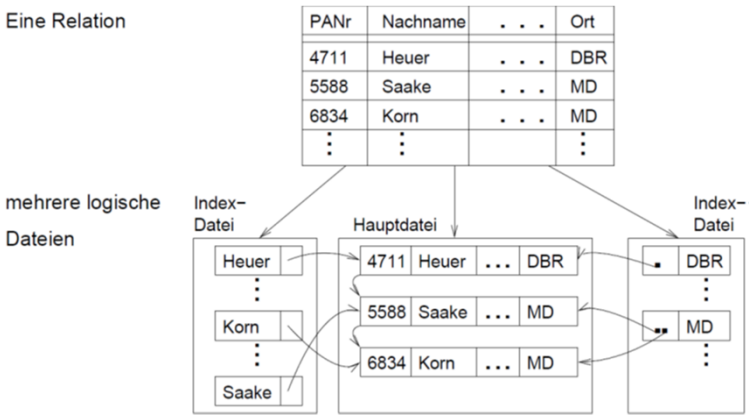
Zürcher Fachhochschule

Die hier durch die interne Ebene beschriebene Funktionalität wird durch das Datensystem oder das Zugriffssystem (5-Schichten-Architektur) wahrgenommen.

Abbildung auf Dateisystem

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

Übliche Form (eine Relation → mehrere logische Dateien):



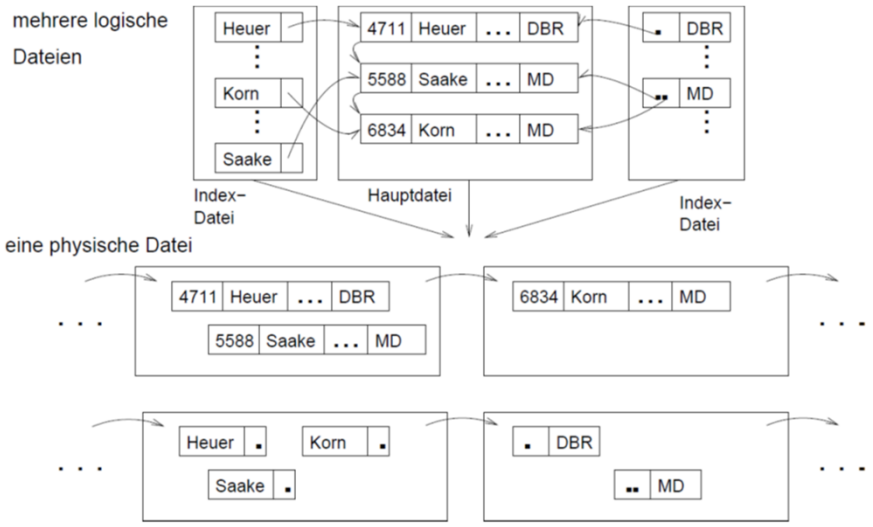
17

Zürcher Fachhochschule

Das Bild zeigt die Abbildung der konzeptionellen Ebene auf die interne Ebene. Im Beispiel werden für eine Relation drei logische Dateien angelegt.

Abbildung auf Dateisystem

- Übliche Form (mehrere logische Dateien → eine physische Datei):



Das Bild zeigt die Abbildung der interne Ebene auf das Dateisystem. Im Beispiel werden für die drei logischen Dateien eine einzige physische Datei angelegt.

Im Beispiel (Bilder) wurde genau eine Relation (vorherige Folie) über Zwischenschritte (interne Ebene) in genau eine physische Datei abgebildet. Es ist aber auch üblich, dass die logischen Dateien mehrerer Relationen zu einer physischen Datei zusammengelegt werden, so dass alle Relationen einer Datenbank in einer einzigen physischen Datei gespeichert werden können (Oracle und SQL Server).

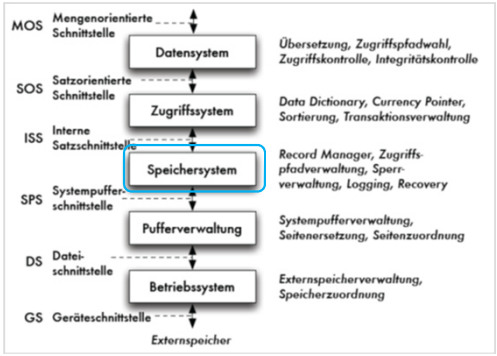
Im SQL Server ist es ausserdem möglich und zur Performance-Verbesserung sinnvoll, die physische Datei auf mehrere physische Dateien aufzuteilen. Diese Dateien sollten dann auf mehrere Disks verteilt werden. Hier ein Auszug aus der Dokumentation: «Put different tables used in the same join queries in different filegroups. This will improve performance, because of parallel disk I/O searching for joined data» und «To maximize performance, create files or filegroups on as many different available local physical disks as possible. Put objects that compete heavily for space in different filegroups».

Und wichtig, die Log-Datei, welche für das Recovery verwendet wird, sollte nicht auf demselben physischen Medium liegen: «Do not put the transaction log file or files on the same physical disk that has the other files and filegroups».

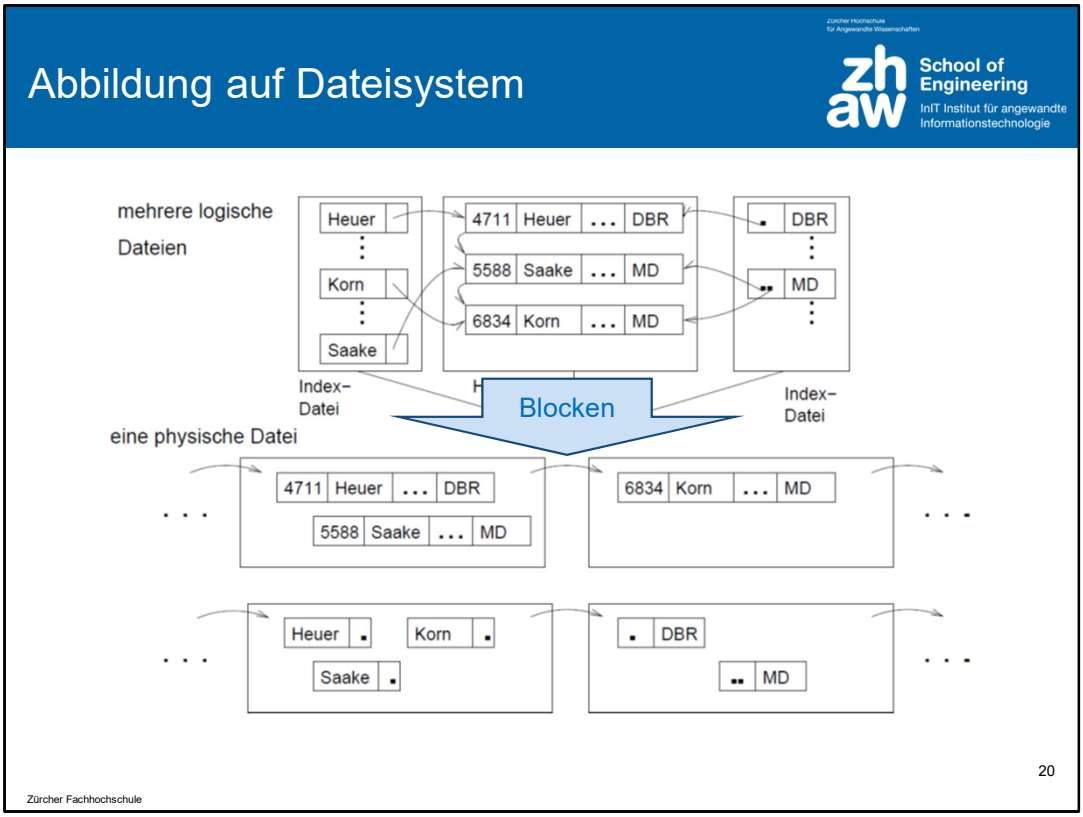
Abbildung auf Dateisystem

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

- Datensätze der logischen Dateien bestehen aus Feldern mit bestimmtem Typ und fester oder variabler Länge
- Diese Datensätze müssen in Seiten (der physischen Dateien) mit fester Grösse 'eingepasst' werden
- Blocken: Abbildung logische Datei(en) → physische Datei



Wir haben zunächst grundsätzlich gezeigt, wie die Tupel in Seiten abgebildet werden, aber wir haben noch nicht überlegt, wie diese Tupel innerhalb der Seiten platziert/verwaltet werden. Das soll auf den nächsten Folien gezeigt werden.



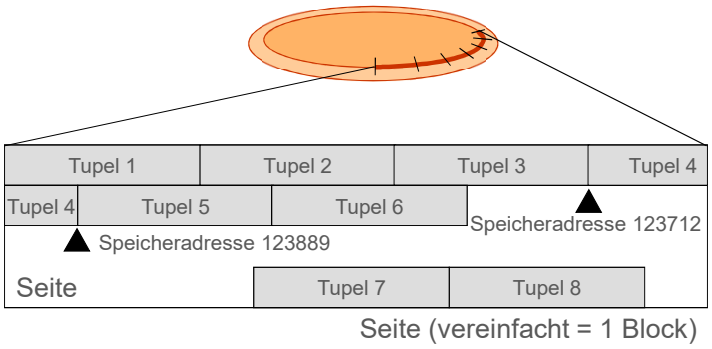
Blocken: Abbildung von internen Datensätzen (eventuell mit variabler Länge) auf Blöcke mit einer fest vorgegebenen Anzahl von Bytes (Aufgabe des Speichersystems).

Abbildung auf Dateisystem

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

Zwischenbemerkung: Unterschied zwischen Block und Seite

- Daten werden blockweise gelesen. Mehrere Blöcke (1, 2, 4, 8,…) werden zu sogenannten Seiten (Pages) zusammengefasst (durch Betriebssystem oder DBMS). Zwischen Hauptspeicher und Magnetspeicher werden immer mehrere Blöcke (= 1 Seite) übertragen.
- Zur Vereinfachung nehmen wir in DAB2 immer an: 1 Block = 1 Seite



21

Zürcher Fachhochschule

Wie schon erwähnt, im SQL-Server werden 8 Blöcke (je 8KB) zu einem Extent (64KB) zusammengefasst, um das Management der Daten zu vereinfachen.

Blocken: N-ary Storage Model

- Das klassische Verfahren des Blocken ist die satzorientierte Speicherung: N-ary Storage Model (NSM) (man könnte aber z.B. auch attributsorientiert speichern)
- Man unterscheidet dabei zwei Varianten:
  - Nichtspannsätze: jeder Datensatz in maximal einem Block.
  - Spannsätze: Datensatz eventuell in mehreren Blöcken. Ist ein Datensatz zu lang für aktuellen Block, dann den noch passenden Anteil des Datensatzes in diesem Block und den Rest in einem neu anzufordernden Block speichern.
- Nichtspannsatz (Normalfall): Spannsatz (z.B. bei BLOBs):

22

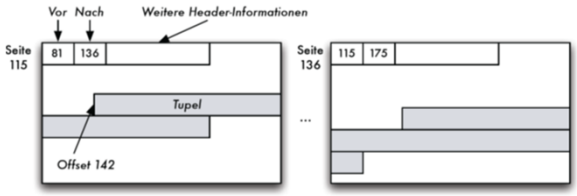
Zürcher Fachhochschule

Bei der attributsorientierten Speicherung werden die Werte der Attribut in den Seiten gruppiert. So werden z.B. die Namen der Kunden einer Tabelle in einer Seite abgelegt. Dies hat für bestimmte Suchen und Zugriffe Vorteile (z.B. für ein Dataware-House). Hoch entwickelte Datenbanksysteme können natürlich wahlweise beide Speichermethoden unterstützen.

Spannsatz: Ist ein Datensatz zu lang für den aktuellen Block (im Beispiel haben nur noch ‘A’ und ‘B’ auf dieser Seite Platz), dann wird der noch passenden Anteil des Datensatzes in diesem Block und der Rest (hier ‘C’ und ‘D’) in einem neu anzufordernden Block gespeichert.

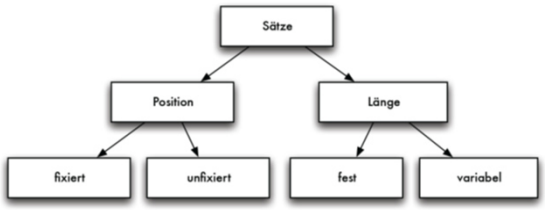
Blocken: N-ary Storage Model

- Interner Aufbau der Seiten:
  - Header
    - Informationen über Vorgänger- und Nachfolger-Seite
    - Eventuell auch Nummer der Seite selbst
    - Informationen über Typ der Sätze
    - Freier Platz
  - Datensätze
  - Unbelegte Bytes
- Organisation der Seiten:
  - Doppelt verkettete Liste
  - Freispeicherverwaltung
  - Datensatzadressen: Seitennummer + Offset



Satztypen – Klassifikation

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



- Variable oder feste Satzlänge:
  - Datensätze mit variabler Satzlänge:
  - Datensätze mit fester Satzlänge
- Fixierter oder unfixierter Datensatz:
  - Fixierter Datensatz / Pinned Record: physischer Zeiger (feste Position)
  - Unfixierter Datensatz / Unpinned Record: logischer Zeiger (z.B. via Index)

Die Begriffe werden auf den folgenden Folien erläutert.

Achtung: Nicht verwechseln mit Nichtspannsätzen und Spannsätzen, diese Klassifikation hier hat damit nichts zu tun.

Satztypen – fixe Satzlänge

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

Aufbau der Datensätze, falls alle Datenfelder feste Länge haben:

- Verwaltungsblock mit Typ und Länge eines Satzes (wenn unterschiedliche Satztypen auf einer Seite möglich), Löschbit
- Freiraum damit Nutzdaten immer an selber Position liegen (Spaces)
- Nutzdaten des Datensatzes

Nachteil: Höherer Speicheraufwand.

Bsp. SQL: char(n); SQL Server: nchar(n);

25

Zürcher Fachhochschule

Mit char(n) wird angegeben, dass das Datenbanksystem n Buchstaben fest reservieren soll. Dies erfolgt unabhängig davon, wie viele Zeichen effektiv ‘gefüllt’ sind. Bei langen Zeichenfolgen werden dann häufig viele Zeichen leer sein (Space), damit wird Platz ‘frei’ gelassen, verschwendet.

Satztypen – variable Satzlänge

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

Aufbau der Datensätze, falls variable Datenfelder vorhanden:

- Verwaltungsblock: Satzlänge l, um die Länge des Nutzdaten-Bereichs d zu kennen
- Nutzdaten des Datensatzes

Nachteil: Höherer Verwaltungsaufwand beim Lesen und Schreiben, Satzlänge muss immer wieder neu ermittelt werden.

Bsp. SQL: varchar(n); SQL Server nvarchar(n);

26

Zürcher Fachhochschule

Varchar(n) wird dann empfohlen, wenn die Grössen der Spaltendateneinträge wahrscheinlich stark variieren (z.B. «varchar(5000)»).

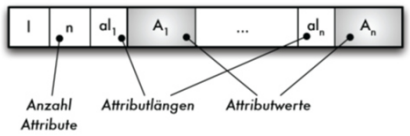
Mit varchar(max) (Speichergrösse von 2^31-1 Byte, 2 GB) wird im SQL-Server der längstmögliche String angegeben.

Satztypen – variable Satzlänge

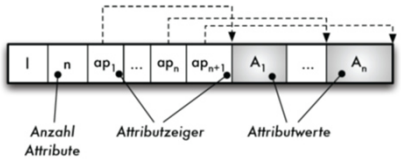
Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Die Nutzdaten selbst werden mit zwei Strategien abgebildet

- Strategie a: Jedes Datenfeld variabler Länge A_i beginnt mit einem Längenzeiger al_i, der angibt, wie lang das folgende Datenfeld ist



- Strategie b: Am Beginn des Satzes wird nach dem Satz-Längenzeiger l und der Anzahl der Attribute ein Zeigerfeld ap_1, …, ap_n für alle variabel langen Datenfelder eingerichtet (Vorteil: leichtere Navigation innerhalb des Satzes)



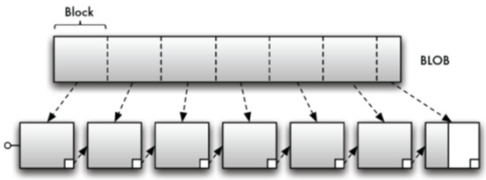
27

Zürcher Fachhochschule

Satztypen – variable Satzlänge

School of Engineering
InIT Institut für angewandte Informationstechnologie

- SQL-Datentypen für sehr grosse, unstrukturierte Datensätze:
  - Z.B. Binary Large Objects (BLOBs): Byte-Folgen wie Bilder, Audio- und Videos
  - Z.B. Character Large Objects (CLOBs): Folgen von Zeichen
- Lange Felder überschreiten i.a. die Grenzen einer Seite, deshalb werden nur die Nicht-BLOB-Felder auf der Originalseite gespeichert
- Auf der Originalseite ist entweder ein einfacher Zeiger (zeigt auf Beginn einer Seitenliste) oder aber ein 'Directory' das weitere Verwaltungs-informationen enthält (z.B. Länge, alle Zeiger zu Seiten, etc.).



28

Zürcher Fachhochschule

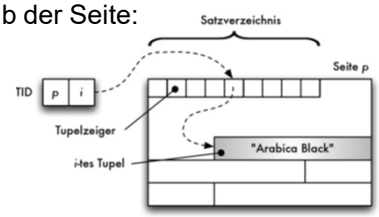
Die Zeichnung zeigt ein BLOB, welches mehrere Blöcke gross ist (oberer Teil) und daher auf mehrere verkettete Blöcke (unterer Teil) aufgeteilt werden muss.

Moderne Datenbanksysteme bieten auch Mechanismen, welche Operationen mit grossen BLOBs unterstützen. Soll z.B. in die Mitte eines zweistündigen Filmes gesprungen werden, möchte man nicht durch die gesamte verkettete Liste navigieren, sondern direkt auf den gewünschten Block springen.

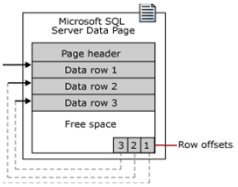
Der SQL Server kennt drei Typen von BLOB-Objekten, welche unterschiedliche Vor- und Nachteile bieten: Filestream, Filetable und Remote Blob Store. In der letzten Variante werden die BLOB-Daten ausserhalb der SQl Server Dateien gespeichert.

Satztypen – fixiert / unfixiert

- Fixiert (pinned): Adressierung über Seitennummer und Offset (Seitennummer → Medium, Zylinder, Spur,…) ist problematisch bei Änderungen (wenn Verschiebung innerhalb der Seite nötig).
- Besser unfixiert (unpinned): TID-Konzept (*T*upel*id*entifikator, Record-ID, Row-ID, RID):
  - TID ist Datensatz-Adresse bestehend aus Seitennummer *p* und Index *i* in das Satzverzeichnis
  - Das Satzverzeichnis ist eine Liste von Tupelzeigern, das am Anfang/Ende der Seite steht. Die i-te Zeigereintrag gibt den Byte-Offset des i-ten Tupels an
  - TIDs bleiben unverändert bei Verschiebung innerhalb der Seite:



SQL Server:



8KB per Page

Zürcher Fachhochschule

29

Wird bei fixierten Satztypen z.B. eine Verschiebung innerhalb einer Seite nötig, müssen alle Referenzen auf diesen Datensatz angepasst werden (z.B. Referenzen in Indexen, Fremdschlüssel-Referenzen).

Beim TID-Konzept ändert sich der Identifikator des Datensatzes nicht, auch wenn dieser verschoben werden muss. Es muss lediglich der Eintrag im Satzverzeichnis angepasst werden. Dieses liegt ja auf der selben Seite und erfordert daher keinen extra Zugriff auf die Daten.

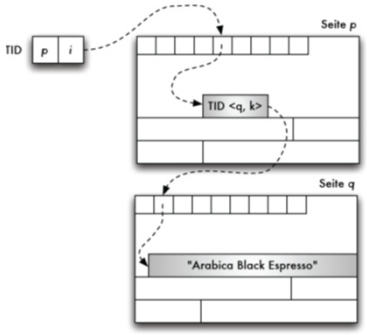
As Bild rechts zeigt das TID-Konzept das im SQL Server verwendet wird (der Index wird Row Offset bezeichnet). Das Satzverzeichnis ist am Ende der Seite platziert, damit sowohl der Datenbereich selbst, wie auch die Daten wachsen können ohne sich gegenseitig zu behindern.

Das TID-Konzept ist bei Datenbanksystemen natürlich weit verbreitet.

Satztypen – unfixiert

Wird ein unfixierter Datensatz auf eine andere Seite verschoben (weil er nach einer Änderung nicht mehr in die Seite passt) wird anstelle des Satzes der TID-Zeiger des migrierten Satzes gespeichert

Dadurch ist der Erhalt des ursprünglichen TIDs gewährleistet → Indexe



Diese zweistufige Referenz ist aber aus Effizienzgründen nicht wünschenswert → Reorganisation in regelmässigen Abständen

Auch noch wichtig (Selbststudium)

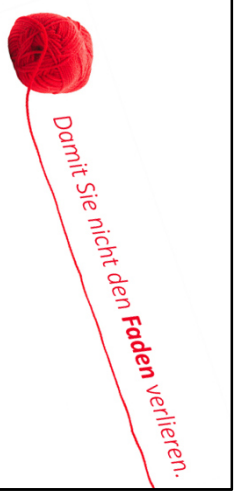
Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Alternative Speichermodelle (Siehe Lehrbuch Abschnitt 3.5.4):
  - "Spaltenorientierte" Speicherung, sog. "Column Stores"
  - Hilfreich z.B. bei häufigen Aggregatabfragen (→ data warehouses, BI)
  - …
- Datenkompression (Siehe Lehrbuch Abschnitt 3.6):
  - Trotz günstigem Speicherplatz sinnvoll, warum?
- Festlegung der physischen Dateien (Siehe Lehrbuch Abschnitt 3.7):
  - Verteilung von Daten, Effizienz
  - Begriffe wie Tablespace, Partition, … (je nach Produkt unterschiedlich) (SQL Server: Filegroups)

Ausblick

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

- Das nächste Mal: Pufferverwaltung, Dateiorganisation & Zugriffsstrukturen



Zürcher Fachhochschule