todo extract deck name

Silas Hoffmann, inf103088

November 24, 2019

Contents

1 Generelles

Erläutern Sie die Motivation hinter der Datenbanktheorie. Gehen Sie insbesondere auf die **Probleme** ein, und beschreiben Sie was es heißt wenn ein Datum **integer** ist.

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Motivation

- Daten sollen dauerhaft gespeichert sein und jederzeit schnell verfügbar sein
- Integrität der Daten muss zu jedem Zeitpunkt gewährleistet sein (Richtigkeit d. Daten)
- Mehrbenutzerbetrieb mit stark unterschiedlichen Nutzungsszenarien (Endanwender, Programmierer, Administrator)
- Probleme hierbei:
  - Konflikte um Ressourcen -> Anfrageoptimierung notwendig
  - Inkonsistente Zustände -> Deadlocks
  - Transaktionsverwaltung -> 2 Phasen-Sperrprotokoll
  - Datenschutz/Autorisierung -> Grant/Revoke

Um diese Aufgaben zu erleichtern und einzelne Bereiche eines DBMS unabhängig vom anderen zu machen, entstanden mehrstufige Architekturmodelle

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 3

Erläutern Sie das **ANSI-SPARC** Schema. Welches Ziel wird hierbei verfolgt.

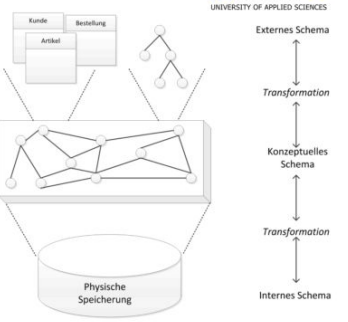
fhwedel
UNIVERSITY OF APPLIED SCIENCES

Architektur von Datenbanken

ANSI/SPARC 3 Schema-Architektur (1975)

Unterscheidung von drei Ebenen

- **Externe Schema** (Benutzersicht)
- **Konzeptuelles Schema** (Referenzschema)
- **Internes Schema** (physische Realisierung)



Ziel:
Trennung der einzelnen Schichten, um Modularität und Austauschbarkeit zu gewährleisten (einzelne Schichten müssen nur Schnittstellen kennen).

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 4

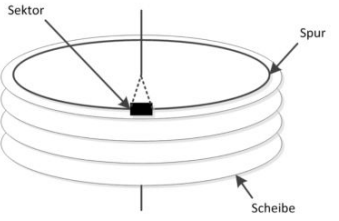
Wie werden Daten auf einer Magnetplatte *adressiert?* Nennen Sie sowohl den Überbegriff als auch die einzelnen Komponenten aus denen sich dieser zusammensetzt.

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Internes Schema - physische Speicherung

Magnetplatten adressieren ihre Daten über:

- Sektor
- Spur
- Scheibe



Ein so bestimmter Bereich ist ein Datenblock
kurz : **Block**
Früher ein Block = 512 Byte. Mit wachsender Festplattengröße mittlerweile Blöcke mit 4 kByte (= Mindestgröße aktueller Betriebssysteme)
↳ Aufgeteilte Datenblöcke liegen oft nicht nah beieinander → zeitaufwendige Lesevorgänge
Was ist das Problem bei kleinen Datensätzen?
↳ Viel ungenutzter Speicherplatz

Wägen Sie ab warum es sinnvoll sein kann bei der Speicherung von Daten auf einer Magnetplatte auf größere Datenblöcke zu setzen.

Bei zu kleinen Datenblöcken müssen die zu speichernden Daten öfter aufgeteilt werden. Da diese im Zweifelsfall über die gesamte Platte verteilt sein können wird viel Zeit für diverse Schreib- und Lesevorgänge benötigt. Bei zu großen Datenblöcken kann es vorkommen, dass eine kleines Datum (z.B. eine Zahl) nur einen geringen Teil der verfügbaren Blockgröße verwendet.

Erläutern Sie grob wie ein Raid-System funktioniert. Was versteht man unter dem sog. **Striping** sowie dem **Mirroring** ?

fhwedel
UNIVERSITY OF APPLIED SCIENCES

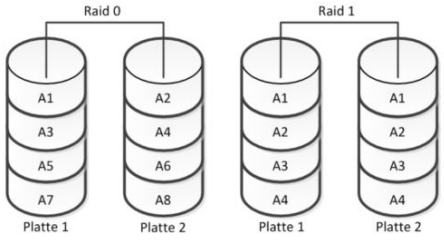
Physische Speicherung - Raid

Festplatten können zu sogenannten RAIDs (**R**edundant **A**rray of **I**ndependent **D**isks) zusammengeschalten werden, um die Ausfallsicherheit oder die Zugriffsgeschwindigkeit zu erhöhen.

Wie funktioniert ein Raid?

RAID 0 - Striping (max. Performance)
↳ Daten werden auf zwei Platten aufgeteilt

RAID 1 - Mirroring
↳ Daten werden kopiert & auf zwei Platten gespeichert

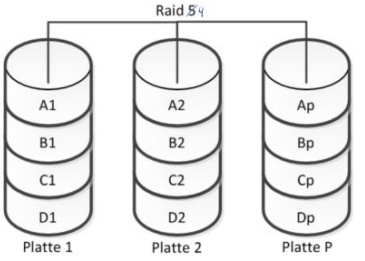


Erläutern Sie was man unter einem **Raid-4 System** versteht. Wie wird hierbei die Parität ermittelt?

fhwedel UNIVERSITY OF APPLIED SCIENCES

RAID 5 - Parität auf mehreren Platten

- Vorteile von RAID 0 und RAID 1 kombiniert
- Parity Information auf Platte P erlaubt bei Ausfall einer Platte die Wiederherstellung der Daten der ausgefallenen Platte.
- RAID 5 ersetzt kein Backup!!
- Werden Daten gelöscht, können diese auch mit der Parity-Information nicht wiederhergestellt werden



fhwedel UNIVERSITY OF APPLIED SCIENCES

Beispiel Paritätsberechnung mittels XOR

Platte 1: `0101 1110...`

Platte 2: `1110 0011...`

Paritätsplatte: `1011 1101...`

Wie vorgehen bei mehr als 2+1 Platten?

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

Platte 1: 1111 0111

Platte 2: 1011 1100

Platte 3: 0111 0011

a) Berechnen Sie die Parity Information der Platte P.
b) Nach erfolgreicher Parity-Berechnung fällt Platte 2 aus. Rekonstruieren Sie diese auf einer Platte 2'. Wichtig ist hierbei der Rechenweg.

---

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Übungsaufgabe

Gegeben sind folgende 4 Platten:

Platte 1: 1111 0111
Platte 2: 1011 1100
Platte 3: 0111 0011

Pl.1	1 1 1 1	0 1 1 1
Pl.3	0 1 1 1	0 0 1 1
Par.	0 0 1 1	1 0 0 0
Pl.2	1 0 1 1	1 1 0 0

Platte P: 0011 1000

a) Berechnen Sie die Parity Information der Platte P.
b) Nach erfolgreicher Parity-Berechnung fällt Platte 2 aus. Rekonstruieren Sie diese auf einer Platte 2'. Wichtig ist hierbei der Rechenweg.

Warum erhöht ein Raid 4 die Ausfallsicherheit, ersetzt aber kein Backup?

Manuelles Löschen führt weiterhin zum Datenverlust.

Weswegen wird in der Datenbanktheorie eine möglichst effiziente Pufferverwaltung angestrebt? Beschreiben Sie grob wie so eine aussehen könnte.

Motivation

Wenn Daten von Festplatte gelesen werden ist dies sehr viel langsamer als wenn diese im Arbeitsspeicher liegen. Die Zugriffszeit gibt dabei an, wie schnell Daten bereitgestellt werden können. Typische Zugriffszeiten heutiger Speichermedien:

- Festplatten: 9 ms
- SSD: 250 µs
- Arbeitsspeicher 60-70 ns
  (Quelle: https://de.wikipedia.org/wiki/Zugriffszeit)

-> Daten sollten komplett im Arbeitsspeicher gehalten werden. Dies ist aufgrund hoher Kosten meist nicht komplett möglich.

-> Intelligente Auslagerung der am häufigsten/dringendsten benötigten Daten in den Arbeitsspeicher

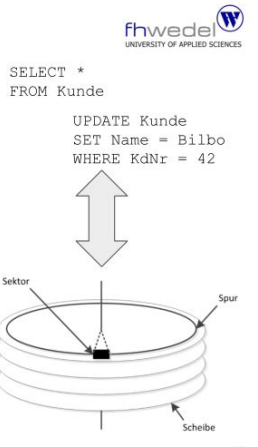
Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 12

Was versteht man unter dem **Mapping** von Speicheradressen?

Zuordnung von Speicheradressen auf Scheiben, Spuren und Sektoren.

Speicherverwaltung

- Verbindung der Daten mit dem Festplatten-Speicher (Mapping von Speicheradressen auf Scheiben, Spuren und Sektoren)
- Abstraktion von konkreter Hardware auf der Speicherverwaltung realisiert
  -> Technologieunabhängigkeit
- Effizientes Auslesen der angeforderten Daten in den Arbeitsspeicher
- Schnelles Auffinden freier Datenblöcke



Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 13

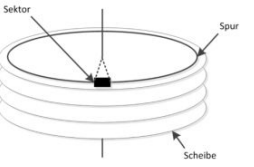
Beschreiben Sie generell den Begriff der Speicherverwaltung.

Speicherverwaltung

- Verbindung der Daten mit dem Festplatten-Speicher (Mapping von Speicheradressen auf Scheiben, Spuren und Sektoren)
- Abstraktion von konkreter Hardware auf der Speicherverwaltung realisiert
  -> Technologieunabhängigkeit
- Effizientes Auslesen der angeforderten Daten in den Arbeitsspeicher
- Schnelles Auffinden freier Datenblöcke

```
SELECT *
FROM Kunde
```

```
UPDATE Kunde
SET Name = Bilbo
WHERE KdNr = 42
```

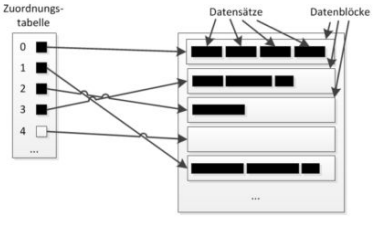


Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 13

Beschreiben Sie grob den Begriff der Speicherzuordnung. Mit welchem Hilfsmittel ist es hierbei möglich die Datenblöcke zu lokalisieren? Müssen die Datenblöcke stets die gleich Länge aufweisen?

Speicherzuordnung

- **Zuordnungstabelle** als Inhaltsverzeichnis für Datenblöcke
- Datenblöcke enthalten mehrere **Datensätze** unterschiedlicher Länge
- Eine Datei kann mehrere Einträge in der Zuordnungstabelle beanspruchen
- Freie Datenblöcke können leicht über Zuordnungstabelle gefunden werden



Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 14

Was sind die Ziele der **Pufferverwaltung** ?

Pufferverwaltung

Ziel Pufferverwaltung:

- Reduktion von physischer Ein-/Ausgabe
- Häufig genutzte Daten werden in **Seiten** im Hauptspeicher vorgehalten
- **Verdrängung** alter Daten im Hauptspeicher durch intelligente Verfahren

-> Weniger Zugriffe auf die Festplatte und damit schnellere Verarbeitung der Anfragen möglich

Pufferverwaltung in Datenbanken hat viele Ähnlichkeiten zum Vorgehen von Cache-mechanismen bei Betriebssystemen.

Wo werden die Pufferdaten zwischengelagert?

In sog. **Seiten** im Hauptspeicher (RAM).

Pufferverwaltung

Ziel Pufferverwaltung:

- Reduktion von physischer Ein-/Ausgabe
- Häufig genutzte Daten werden in **Seiten** im Hauptspeicher vorgehalten
- **Verdrängung** alter Daten im Hauptspeicher durch intelligente Verfahren

-> Weniger Zugriffe auf die Festplatte und damit schnellere Verarbeitung der Anfragen möglich

Pufferverwaltung in Datenbanken hat viele Ähnlichkeiten zum Vorgehen von Cache-mechanismen bei Betriebssystemen.

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 15

Beschreiben Sie anhand der Datenbanken vom Hersteller **Oracle** was man unter einem *Segment* sowie einem *Extent* versteht. Wie kann es sein, dass die Größe des *Extents* von den kleinsten logischen Einheiten einer Festplatte / SSD abweichen kann?

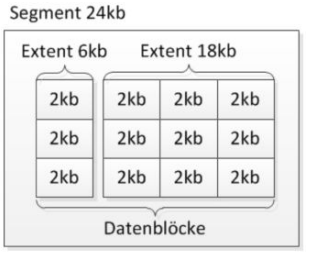
fhwedel
UNIVERSITY OF APPLIED SCIENCES

Beispiel Oracle

- **Segmente**: logische Einheit
- **Extent**: kleinste logische Speichereinheit in Oracle

Segmente besteht aus einem oder mehreren Extents, die die Informationen zu einem Objekt gespeichert haben (z.B. Relation).

Reicht Speicherplatz für Segment nicht aus, werden weitere Speicherbereiche im Rahmen des Tablespaces allokiert.



Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 17

Da eine Datenbank i.d.R. ein eigenes Filesystem anlegt ist es komplett losgelöst von sämtlichen Restriktionen von Festspeicher / Betriebssystem.

Was ermöglicht das Konzept eines **Segments** in der Datenbanktheorie?

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Segmente

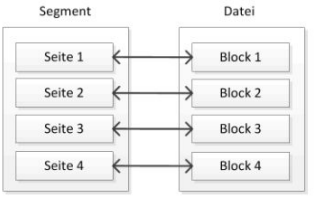
- Abstraktionsebene, die Blöcke und Dateien verbirgt.
- Konzept des Segments ermöglicht:
  - verschiedene Einbringungsstrategien (direktes Einbringen/update-in-place, indirektes Einbringen)
  - Segmente als kleinste Einheit
    - des Sperrens für Transaktionen
    - der Wiederherstellung bei Gerätefehlern
    - der Zugriffskontrolle
- Behälter für Relationen, Zugriffspfade, ...
- Man unterscheidet:
  - Öffentliche Segmente (gemeinsam genutzte Datensätze, erlaubter konkurrierender Zugriff)
  - Private Segmente (z.B. Speicherung von Log-Informationen)
  - Temporäre Segmente (z.B. Sortieren, Speichern von Zwischenergebnissen)

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 18

Beschreiben Sie das Prinzip der **direkten** Seitenadressierung an einem Bild.

Seitenzuordnung/ direkte Seitenadressierung

- Managed **Segmente** im Arbeitsspeicher, die aus Seiten (pages) bestehen.
- Größe der Seiten wird so gewählt, dass sie jeweils einem Block der Festplatte entspricht (egal ob voll oder leer).
  -> je 1 Zugriff je Seite/Block
- Bei **direkter Seitenadressierung** werden Segmente analog zu dem Bild rechts zugeordnet.



direkte Seitenadressierung

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 19

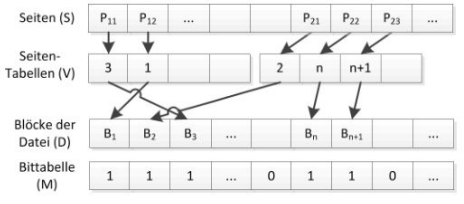
Beschreiben Sie das Prinzip der **indirekten** Speicheradressierung. Gehen Sie insbesondere auf die beiden verwendeten Hilfsstrukturen ein.

Indirekte Seitenadressierung

- erhöhte Zugriffskosten mit besserer Speicherauslastung
- maximale Flexibilität bei Blockzuordnung
- arbeitet mit 2 Hilfsstrukturen
- dynamisch änderbar

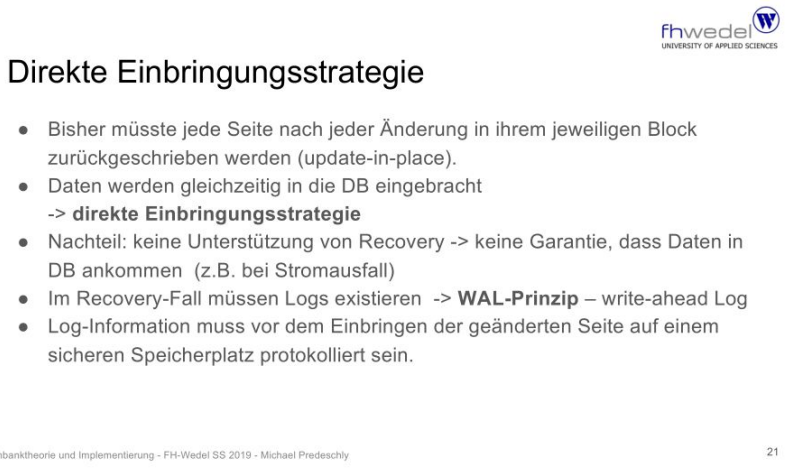
Hilfsstrukturen:

- eine **Seitentabelle** (V) je Segment
- **Bittabelle** (**Map** - M) zur Freispeicherverwaltung



Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 20

Beschreiben Sie das Prinzip der **direkten** *Einbringunsstrategie*. Welche Risiken können hierbei auftreten und wie kann man diesen entgegenwirken?



WAL-Prinzip: Man schreibt einen Log über die Dinge die als nächstes getan werden sollen (wie eine Todo-Liste). Falls hierbei ein Stromausfall dazwischen kommt weiß man genau welche Daten noch nicht kopiert wurden / noch gelöscht werden müssen um die Daten konsistent zu halten.

Beschreiben Sie den Begriff der **verzögerten Einbringungsstrategie** am Beispiel des *Schattenspeicherkonzepts.*

Verzögerte Einbringungsstrategie

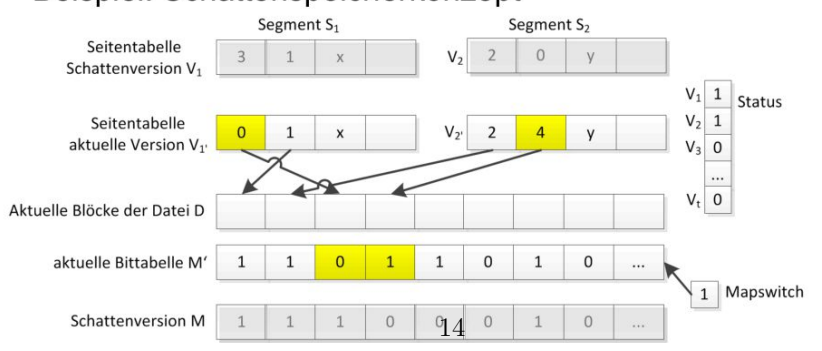
- **Verzögerte Einbringungsstrategien** unterscheiden zwischen Zurückschreiben von Seiten und ihrem Einbringen.
- Hierbei kann Logging reduziert werden oder sogar ganz entfallen.
- Ein Beispiel für ein solches Verfahren ist das **Schattenspeicherkonzept** aus IBMs System R (1977).

Idee:
Alle belegten Segmente, Seitentabellen und Bittabellen werden in **Sicherungspunkten** gespeichert. Sicherungspunkte sind dabei „eingefrorene" Zustände auf einem nichtflüchtigen externen Speicher. Seitenänderungen führen dann zu neuen Seiten (in freien Bereichen), während alte Inhalte unverändert in **Schattenseiten** abgelegt bleiben.

Beispiel: Schattenspeicherkonzept

- Vorteile:
  - Unterstützung von Recovery
  - WAL wird vermieden -> flexibleres Schreiben der Logfiles
- Nachteile:
  - Erhöhte Zugriffszeiten durch verlorene Clustereigenschaft von Blöcken (Daten liegen verteilter)
  - Seitentabellen und Bittabellen sehr aufwendig, d.h. passen nicht mehr in den Hauptspeicher (nur bei großen Datenbanken ein Problem)
- Zu Beginn einer Änderung werden für alle Segmente S_x, die geändert werden sollen, Schattenversionen der Seitentabelle V_x angelegt.
- Zusätzlich wird eine Schattenkopie der Bittabelle M angelegt (Mapswitch zeigt aktuelle Version an).
- Über den Status ist ersichtlich, welches Segment gerade geändert wird.

Beispiel: Schattenspeicherkonzept

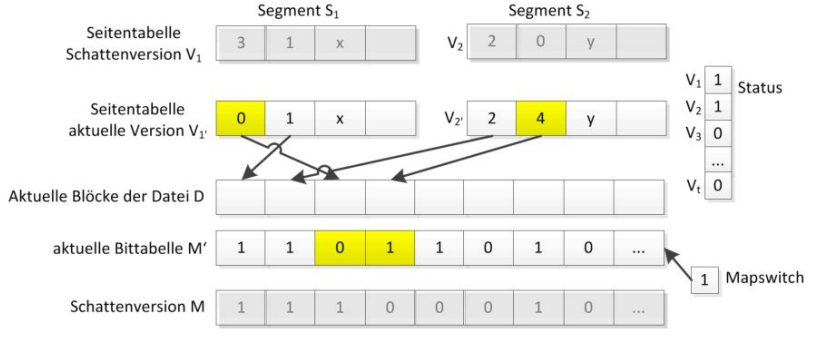


Erläutern Sie das Konzept des **Schattenspeicherkonzepts.**

Beispiel: Schattenspeicherkonzept

- Vorteile:
  - Unterstützung von Recovery
  - WAL wird vermieden -> flexibleres Schreiben der Logfiles
- Nachteile:
  - Erhöhte Zugriffszeiten durch verlorene Clustereigenschaft von Blöcken (Daten liegen verteilter)
  - Seitentabellen und Bittabellen sehr aufwendig, d.h. passen nicht mehr in den Hauptspeicher (nur bei großen Datenbanken ein Problem)
- Zu Beginn einer Änderung werden für alle Segmente S_x, die geändert werden sollen, Schattenversionen der Seitentabelle V_x angelegt.
- Zusätzlich wird eine Schattenkopie der Bittabelle M angelegt (Mapswitch zeigt aktuelle Version an).
- Über den Status ist ersichtlich, welches Segment gerade geändert wird.

Beispiel: Schattenspeicherkonzept



Erläutern Sie wie die Verwaltung des Puffers funktioniert? Gehen Sie insbesondere auf die Funktion des Pufferrahmens sowie des Pufferkontrollblocks ein.

Verwaltung des Puffers

- Alle Lese- und Schreiboperationen aller (parallelen Transaktionen) der DB werden über den Puffer abgewickelt.
  -> GB-Bereich
  -> Puffer kann nur Bruchteil einer DB aufnehmen
- Puffer besteht aus N **Pufferrahmen** für N Datenbankseiten
- **Pufferkontrollblock** (PKB) enthält Verwaltungsdaten wie z.B. Änderungsvermerke
- Es werden Kontroll- und Zuteilungsaufgaben notwendig, um den Puffer zu verwalten.

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 25

Welche Kontroll- und Zuteilungsaufgaben besitzt die Speicherverwaltung einer Datenbank?

Kontroll- und Zuteilungsaufgaben

- Speicherverwaltung zur Zuteilung von Pufferrahmen, sowie zur Suche und Ersetzung von Seiten
- Lastkontrolle zur Anpassung der Anzahl der aktiven Transaktionen an das momentane Optimum, das vom jeweiligen Betriebszustand abhängt
- Ziel ist die Reduktion der Input/Output Zugriffe.

- Vorgehen:
  - Seitennummer ermitteln
  - Zugehörige Seite anfordern (**logische Seitenreferenz**)
  - Festlegen, ob Seite: gelesen oder geändert werden soll
- **FIX-Operation** legt fest, dass Seite im Pufferrahmen bleibt (**UNFIX = Freigabe**)

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 26

Gehen Sie auf die verschieden Szenarien ein welche beim Arbeiten mit einem Puffer auftreten können.

Logische Seitenreferenz

a) **Seite im Puffer**

-> geringer Aufwand (ca. 100 CPU Instruktionen)

- Seite finden
- FIX-Operation ausführen
- Wartungsarbeiten im PKB ausführen
- Pufferadresse an rufende Komponente übergeben

b) **Seite nicht im Puffer**

-> Logische Seitenreferenz führt zu einer physischen Seitenreferenz

- erfolglose Suche im Puffer
- 2 Input/Output – Vorgänge (je I/O etwa 2500 CPU Instruktionen pro Vorgang + 9 ms für HD)
  - Auswahl einer Seite zur Verdrängung
  - Herausschreiben der Seite bei Änderungsvermerk
  - Neue Seite lesen

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 27

Geben Sie eine grobe Übersicht über die verschiedenen Suchstrategien der Pufferverwaltung.

Auffinden einer Seite

- **Forderung:** Hocheffiziente Suchstrategie, da der Vorgang extrem häufig vorkommt.

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 29

Erläutern Sie nun noch kurz, wozu bei einem Router (oder einer Arbeitsstation) eine Wegwahl- bzw. Routingtabelle eigentlich dient. Was ist das bzw. was wird hier prinzipielle gespeichert?

In paketvermittelten Netzen werden Ende-zu-Ende-Verbindungen zwischen zwei Endgeräten dadurch hergestellt, dass Datenpakete von einer sendenden Station zu einer Empfangsstation übertragen werden. Die Stationen müssen über ihre Netzwerkadressen eindeutig identifizierbar sein. Zwischen den beiden Stationen liegen Router, die die Datenpakete gemäß ihrer Routingtabellen von der Sendestation über den ersten Router zum zweiten, zum dritten usw. weiterleiten, bis der letzte Router auf dem Weg durch das Datenpaketnetz die Datenpakete an die Empfängerstation ausliefert. Routingtabellen werden von den Routing-Protokollen für die Pfadermittlung erstellt und können je nach Routing-Protokoll unterschiedlich sein. Beim statischen Routing ist der Routingpfad fest vorgegeben. Beim dynamischen Routing werden die Routingtabellen durch aktuelle Routinginformationen aktualisiert. Die Aktualisierung übernehmen Routing-Protokolle, die die Informationen eintragen. Das können Information über die Verknüpfungen der Netzwerke, bzw. der Endgeräte und der Hops mit den dazu gehörenden IP-Adressen sein, oder die Routing-Metriken mit der die Route festgelegt wurde.

Was versteht man unter einem **logischen** Seitenreferenzstring?

Seitenreferenzstrings

- **Jede Datenanforderung ist eine *logische Seitenreferenz***
- **Aufgabe der DB-Pufferverwaltung:**
  Minimierung der *physischen Seitenreferenzen*
- **Referenzstring** $R = \langle r_1, r_2, \ldots r_i, \ldots r_n \rangle$
  mit $r_i = (T_i, D_i, S_i)$

  T_i zugreifende Transaktion
  D_i referenzierte DB-Partition
  S_i referenzierte DB-Seite

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Logischer Seitenreferenzstring

- zeitliche Folge der logischen Seitenanforderungen aller parallelen Transaktionen innerhalb eines Zeitabschnittes
- beschreibt aktuelles Zugriffsverhalten des DBS
- **Problem:** Erzeugung eines optimalen physischen Seitenreferenzstrings (= minimale Zugriffe) Wesentlich beeinflusst durch:
  - Größe des Systempuffers und Wahl der Seitenersetzungsstrategie
  - Ein logischer Seitenreferenzstring kann unterschiedliche physische Referenzstrings zur Folge haben.
- Die Optimierung der Seitenersetzungsstrategie im Systempuffer ist von entscheidender Bedeutung.

Geben Sie eine kurze Aufschlüsselung über die möglichen Suchstrategien bei der *Pufferverwaltung*.

Erläutern Sie die **Vorgehensweise** bei der direkten Suche im Pufferrahmen, wie viele Rahmen werden durchschnittlich durchsucht und was ist der wesentliche Nachteil dieser Technik?

Direkte Suche im Pufferrahmen

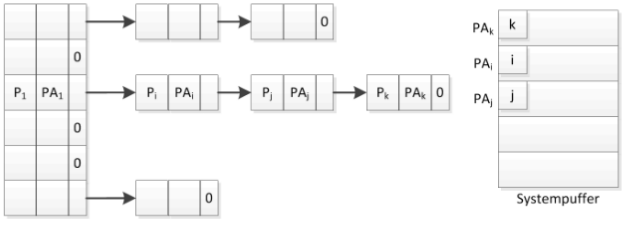
- Im **Seitenkopf** (page header) wird geprüft, ob die im Rahmen vorhandene Seitennummer mit der angeforderten übereinstimmt.
- Im Erfolgsfall (bei N Rahmen) sind durchschnittlich N/2, bei Misserfolg N Rahmen aufzusuchen.
- Hoher Aufwand besonders bei virtueller Systemumgebung (verborgener Zusatzaufwand durch Paging).

Erläutern Sie wie eine indirekten Suche mittels einer Hashfunktion funktioniert. Gehen Sie hierbei insbesondere auf eine sog. Überlaufkette ein.

Indirekte Suche über Hilfsstrukturen – Hash

- Hash Funktion $f(P_i)$ mit P = Page ergibt Position in der Hash-Tabelle.
- Zuordnung **Seitennummer** P_i – **Pufferadresse** PA_i von jeder Seite im Puffer wird gehasht und in der Hash-Tabelle abgelegt.
- Bei Kollisionen der Hash-Funktion ($f(P_j) = f(P_k) = f(P_l) = f(P_m)$) entsteht eine Überlaufkette.
- Anzahl der Überläufer ist vom gewählten Hash-Verfahren/Funktion und der Größe der Hash-Tabelle abhängig.
- Wahl so, dass die n Einträge, die bei einer logischen Seitenreferenz durchsucht werden müssen, im Mittel auf $1 < n < 1.2$ begrenzt sind.

Hash-Funktion mit Überlaufliste

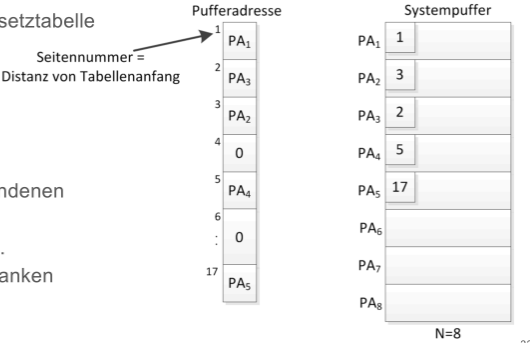


Ein/Austragen einer Seite impliziert Ein-/Austragen eines entsprechenden Verweises.

Wie funktioniert die Suche nach den im Puffer über eine *direkt adressierte* **Umsetzungstabelle** ? Welchen wesentlichen Nachteil bringt dieses Verfahren mit sich?

Indirekte Suche über Hilfsstrukturen – Tabelle

a) Direkt adressierte Umsetztabelle



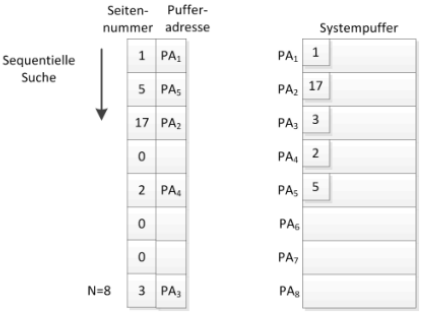
Nachteil:
Alle nicht im Puffer vorhandenen Seiten müssen ständig Tabelleneinträge belegen.
-> nur bei kleinen Datenbanken

Wie funktioniert die indirekte Suche nach einer Seite im Pufferrahmen mittels einer *unsortierten* Liste?Gehen Sie insbesondere auf die Vor-/Nachteile sowie die Laufzeit des Verfahrens ein.

Indirekte Suche über Hilfsstrukturen – Tabelle

b) Unsortierte Tabelle

Im Erfolgsfall durchschnittlich N/2 Einträge, bei Misserfolg (d.h. Seite nicht im Puffer) N Einträge durchsuchen.



Wie funktioniert die indirekte Suche nach einer Seite im Pufferrahmen mittels einer *sortierten* Liste? Gehen Sie insbesondere auf die Vor-/Nachteile sowie die Laufzeit des Verfahrens ein.

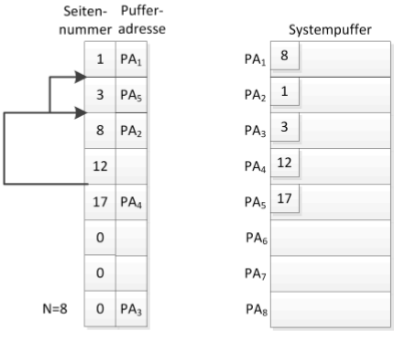
Indirekte Suche über Hilfsstrukturen – Tabelle

c) Sortierte Tabelle

Vorteil:
Bei Misserfolg sind max. log(n) Einträge zu durchsuchen

binäre Suche

Nachteil:
Einfüge- und Löschoperationen sehr aufwendig



Wie funktioniert die indirekte Suche nach einer Seite im Pufferrahmen mittels einer *sortierten mit verketteten Einträgen* Liste? Gehen Sie insbesondere auf die Vorteile des Verfahrens ein.

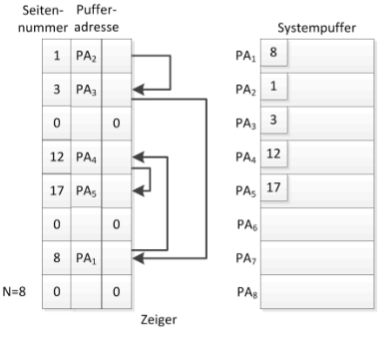
Indirekte Suche über Hilfsstrukturen – Tabelle

d) Tabelle mit verketteten Einträgen

Wie sortierte Tabelle, nur aufeinander folgende Einträge nicht physisch benachbart.

Gute Einfüge- und Löscheigenschaften.

Suche entlang einer Zeigerkette



Warum sollte man Zeiger verwenden? : Wenn ein neues Element eigefügt / gelöscht wird muss man nicht die komplette Restliste umkopieren um Platz für eine neues Element zu machen / löschen Man setzt einfach 2 Zeiger neu.

Welche wesentlichen Aufrufe einer Datenbank muss eine Pufferverwaltung bewerkstelligen können?

Zusammenstellung Aufrufe an die Pufferverwaltung

Aufruf	Funktion
Bereitstellen (logische Referenz)	Bereitstellen der angeforderten Seite im Puffer, erfordert gegebenenfalls Verdrängen einer Seite (vgl. Ersetzungsstrategie) und physisches Einlesen der neuen Seite.
FIX	Festhalten einer Seite im Puffer, damit ist die Adressierbarkeit des Seiteninhaltes durch Maschineninstruktionen gewährleistet.
UNFIX	Aufheben eines FIX. Die Seite ist für Ersetzung frei.
Änderungsvermerk	• Eintragen eines Vermerks in die Seite, dass beim Verdrängen aus dem Puffer physisch geschrieben werden muss. • evtl. Sicherstellen eines Before-Image, falls dies die Einbringstrategie erfordert
Schreiben	Sofortiges Ausschreiben der Seite aus dem Puffer in die DB

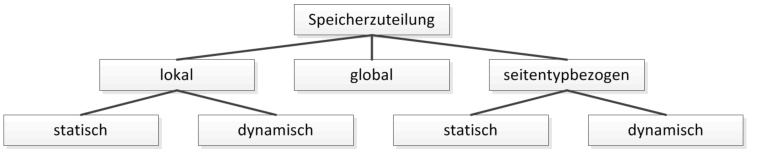
Welches Problem wird mit der Speicherzuteilung im Puffer gelöst. Welche Strategie wird hierbei mit welchen Eigenschaften verwendet?

Speicherzuteilung im Puffer

- Problem: Beschränkte Anzahl von Rahmen für alle Nutzer
  -> Faire Speicherzuteilung
- Speicherzuteilungsstrategie
  - Bestimmt für jede aktive Transaktion eine Menge von Rahmen zur Aufnahme der Seiten.
  - Besitzt Pool von freien bzw. freigegebenen Rahmen.
- Eigenschaften:
  - DB-Seiten können im Puffer durch mehrere Benutzer genutzt werden.
  - Lokalität der DB- Zugriffe kommt durch das Zugriffsverhalten aller Benutzer (Transaktionen) zustande.
  - Auf Grund der Zugriffspfad- und Speicherungsstrukturen des DBS ist eine Vorhersage der Zugriffswahrscheinlichkeit möglich.

Geben Sie eine grobe Klassifikation der **Speicherzuteilungsstrategien** , zwischen welchen wesentlichen Strategien gilt es hierbei zu unterscheiden?

Klassifikation der Speicherzuteilungsstrategien



- lokal:
  - nur das aktuelle Referenzverhalten einer Transaktion wird berücksichtigt
  - jeder Prozess erhält für die benötigten Seiten eine Menge reservierter Hauptspeicher - Rahmen
- global:
  - gesamter Puffer steht allen aktiven Transaktionen gemeinsam zur Verfügung
  - Speicherzuteilung wird vollständig durch Ersetzungsstrategien bestimmt.

Was versteht man unter einer *statischen / dynamischen / seitenbezogenen* Speicherzuteilungsstrategie?

Klassifikation der Speicherzuteilungsstrategien

- statisch:
  - eine erforderliche Menge von Rahmen (Partitionen) muss frei sein, bevor die Transaktion gestartet wird (preclaiming)
  - im DBS kaum verwendbar
- dynamisch:
  - Partitionsgrößen variieren in Abhängigkeit vom aktuellen Bedarf an Seiten (d. h. Seiten werden zwischen Partitionen ausgetauscht)
- seitentypbezogen:
  - Teile des Systempuffers für bestimmte Seitentypen (Datenseiten, Zugriffspfadseiten, Systemseiten) reserviert.
  - Zuteilungs- und Ersetzungsentscheidungen jeweils auf die betreffende Partition bezogen.

Grenzen Sie das **Demand-Paging-** vom **Prepaging-Verfahren** ab.

Klassifikation der Ersetzungsstrategien

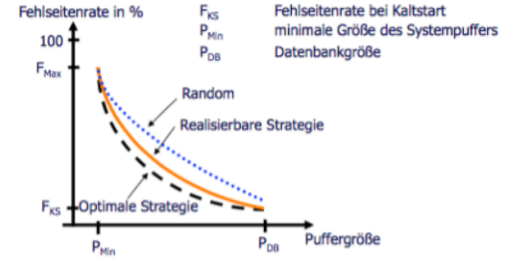
- **Demand - Paging - Verfahren**: Bei Auftreten einer Fehlseitenbedingung wird genau eine Seite im Puffer durch die angeforderte Seite ersetzt.
- **Prepaging - Verfahren**: Neben der angeforderten Seite werden noch weitere im Sinne eines **Look-ahead** in den Puffer gebracht (nützlich bei ausgeprägter Sequentialität).

- **Demand - Paging - Verfahren:**
  - realisierbare Verfahren: Die Ersetzungsstrategie verlangt keine Kenntnis über das zukünftige Referenzverhalten
  - nicht realisierbare Verfahren
    Verlangt Kenntnisse über zukünftige Referenzen
    - theoretisches Interesse
    - für Abschätzung der unteren Schranke für Fehlseitenrate

Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 42

Demand-Paging-Verfahren

- Berühmt ist das Belady - Optimalitätsprinzip: Ersetze die Seite im Puffer, deren Abstand bis zur nächsten Referenz maximal ist.

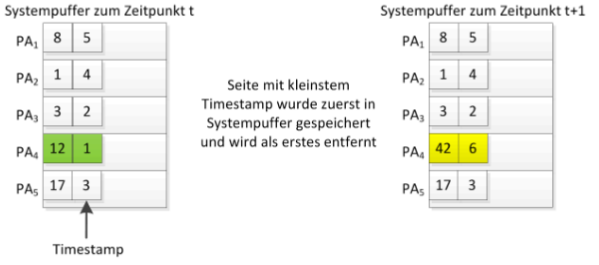


Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 43

Erläutern Sie das **FIFO** Ersetzungsverfahren, für welchen Anwendungsfall eigent sich dieses Verfahren am besten?

FIFO (First In First Out)

- Ersetze die Seite, die am längsten im Puffer ist.
- Eignet sich beim sequentiellen Zugriffsverhalten.



Erläutern Sie das **LFU** Ersetzungsverfahren. Welchen wesentlichen Nachteil bringt dieses Verfahren mit sich und wie kann man dem entgegenwirken?

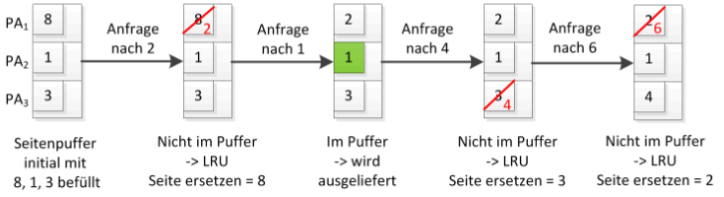
LFU (Least Frequent Used)

- Ersetze die Seite mit geringster Referenzhäufigkeit
- Explizite Angabe eines Refrenzzählers (RZ) zu jeder Seite im Puffer
- Nachteil:
  Seiten, die punktuell sehr intensiv genutzt wurden, sind praktisch nicht mehr zu verdrängen
- Abhilfe:
  Herabsetzen der Referenzzählerwerte in bestimmten Zeitabständen

Erläutern Sie das **LRU** Verfahren. Welche Spezialisierungen / genauen Ausprägungen dieses Verfahrens gibt es?

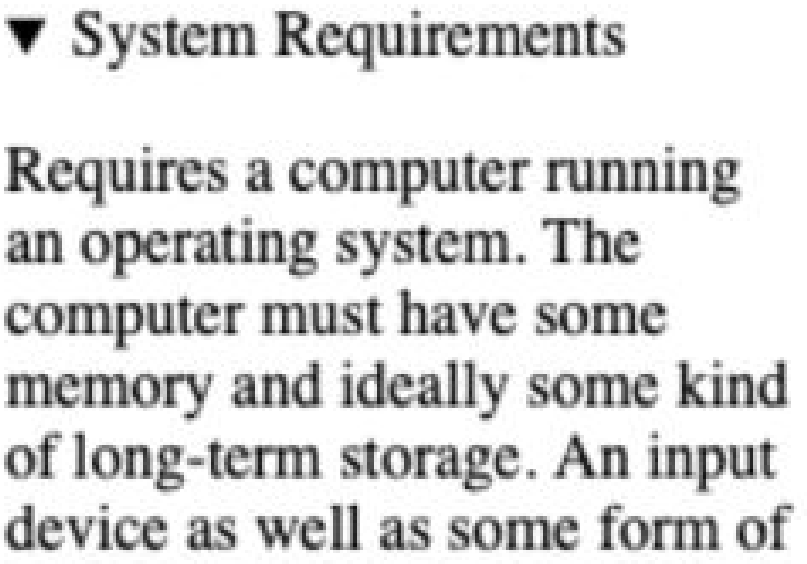
LRU (Least Recently Used)

- Ersetze die Seite, die am längsten nicht mehr verwendet wurde
- Seiten können als LRU - Stack verwaltet werden
- Seite kommt bei jeder Referenz in die oberste Position



- Nach der Interpretation von „USED“ unterscheidet man **LRR** (Least Recently Referenced) **LRUN** (Least Recently Unfixed)

Wie lässt sich ein Dropdown Text mit HTML implementieren, beschreiben Sie die Syntax am folgenden Beispiel:



fhwedel
UNIVERSITY OF APPLIED SCIENCES

Verbergen von Details

Manche Informationen sind nicht für alle Anwender relevant
-> Anzeige nur auf Aufforderung

```
<details>
  <summary>System Requirements</summary>
  <p>Requires a computer running an operating
      system. The computer must have some memory
      and ideally some kind of long-term storage. An
      input device as well as some form of output
      device is recommended.
  </p>
</details>
```

▼ System Requirements

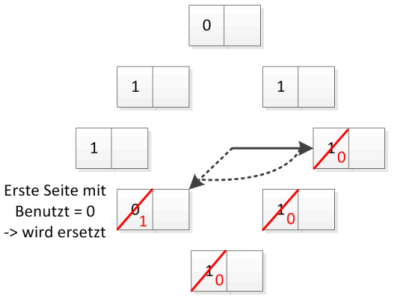
Requires a computer running an operating system. The computer must have some memory and ideally some kind of long-term storage. An input device as well as some form of output device is recommended.

Erläutern Sie die **CLOCK / Second Chance** Ersetzungsstrategie.

fhwedel
UNIVERSITY OF APPLIED SCIENCES

CLOCK / Second Chance

- jede Seite erhält ein Benutzt - Bit
- Benutzt - Bit wird bei jeder Seitenreferenz auf 1 gesetzt
- Bei Fehlseitenbedingung beginnt zyklische Suche mit dem Auswahlzeiger
- Falls Bit=1, wird es auf 0 gesetzt
- Zeiger um eins weitergerückt
- Falls Bit=0, wird Seite ersetzt
- jede Seite überlebt mindestens 2 Zeigerumläufe



Übung: Seitenersetzungsstrategien

Die Funktion und die Operation eines Datenbankpuffers soll anhand eines Beispiels verdeutlicht werden. Gegeben sei ein Referenzstring, d.h. eine Folge von B- (Bereitstellen, fix), F- (Freigeben, unfix) Aufrufen an die Pufferverwaltung des DBMS:

B(2) B(4) B(7) B(8) F(4) B(3) B(6) F(2) F(7) B(2)
F(8) F(3) B(5) B(3) B(1) F(1) B(9) B(10)

Die Puffergröße beträgt 5 Seiten. Führen Sie für den gegebenen Referenzstring die Pufferersetzung nach den folgenden Strategien durch:

A) FIFO (First in - First out)
B) LRU (Least recently used)

FIFO

Op	Puffer 1	2	3	4	5	Kommentar
B(2)	2					
B(4)	2	4				
B(7)	2	4	7			
B(8)	2	4	7	8		
F(4)	2	4	7	8		
B(3)	2	4	7	8	3	
B(6)	2	7	8	3	6	
F(2)	2	7	8	3	6	
F(7)	2	7	8	3	6	FIFO auch für auf Unfix ges. Elemente → Wenn mehrere unfix-Elemente vert wird d. mit zuerst ges. zuerst rausgeworfen.
B(2)	2	7	8	3	6	
F(8)	2	7	8	3	6	
F(3)	2	7	8	3	6	
B(5)	2	8	3	6	5	
B(3)	2	8	3	6	5	
B(1)	2	3	6	5	1	
F(1)	2	3	6	5	1	
B(9)	2	3	6	5	9	
B(10)	2	3	6	5	9	Puffer voll die 6 wäre d. nächste Zahl die vorraussichtl bei B(10))

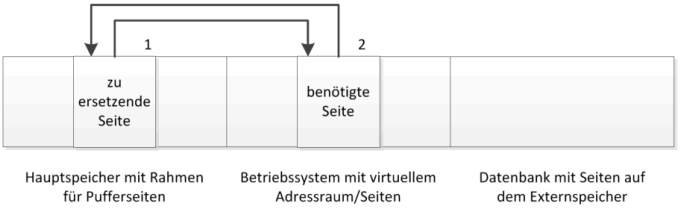
Was versteht man unter einem *Page Fault?*

Ein Seitenfehler (engl. page fault) tritt bei Betriebssystemen mit Virtueller Speicherverwaltung und Paging auf, wenn ein Programm auf einen Speicherbereich zugreift, der sich gerade nicht im Hauptspeicher befindet, sondern beispielsweise auf die Festplatte ausgelagert wurde oder wenn zu der betreffenden Adresse gerade kein Beschreibungseintrag in der MMU verfügbar ist. Als unmittelbare Folge des Seitenfehlers kommt es zu einer synchronen Programmunterbrechung (engl.: synchronous exception (fault)). Das Betriebssystem sorgt nun dafür, dass der angeforderte Speicherbereich wieder in den Hauptspeicher geladen wird oder der fehlende MMU-Eintrag nachgeladen wird, damit das Programm darauf zugreifen kann. Ein Seitenfehler ist daher kein Fehler im eigentlichen Sinne. Der Anwender spürt von diesem Vorgang nichts, maximal eine Verlangsamung des Programms, das den Seitenfehler verursachte, da das Laden der Seite oder das Bearbeiten des Vorgangs einen kurzen Augenblick benötigt. Andere Programme oder Prozesse sind davon nicht betroffen.

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Page Fault

- Benötigte Seite im Systempuffer, aber ausgelagert Betriebssystem muss die referenzierte Seite vom Seitenwechselspeicher einlesen.

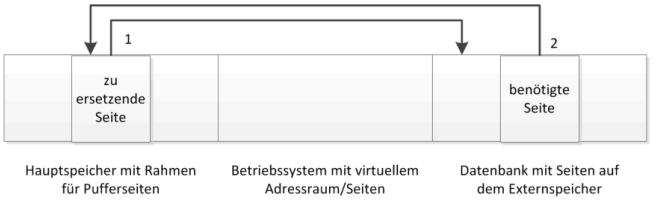


Datenbanktheorie und Implementierung - FH-Wedel SS 2019 - Michael Predeschly 51

Erläutern Sie den *Database Fault*

Database Fault

- Benötigte Seite ist nicht im Systempuffer, zu ersetzende Seite ist im Hauptspeicher, die zu ersetzende Seite kann zurück geschrieben und die angeforderte Seite eingelesen werden.



Erläutern Sie was man unter einem *Double page fault* versteht.

Double Page Fault

- Double Page Fault:
  - benötigte Seite ist nicht im Systempuffer,
  - zu ersetzende Seite ist nicht im HS,
  - zu ersetzende Seite muss in den HS eingelesen werden,
  - dann erfolgt Rückschreiben in DB