

Transformation

- Selektionsprädikte (σ) aufspalten
 - Selektionen "nach unten" schieben
 - Zusammenlegen von σ und \times zu Δ
 - Optimierung der Δ Reihenfolge
 - Kommutativität und Assoziativität ausnutzen
- Möglichkeiten Umsetzung Operatoren:**
- σ : IndexScan P
 - σ : RelationalScan P
 - \times : NestedLoop
 - \times : HashJoin
 - \times : MergeJoin

Selektivität

• Selektivität einer Anfrage, die Schlüsselattribut einer Relation R definiert ist. Falls ein Attribut A spezifiziert wird, für das nur verschiedene Werte existieren, ist Selektivität $= \frac{1}{\text{Val}(A)}$.

Wann lohnt sich ein Index?

Bsp.: Kosten Baumtiefe: $\text{lev}_1(RCA)$

$$(\text{lev}_1(\text{rcet})) + 1 \cdot n < b$$

$$\frac{n}{b} < 1 / (\text{lev}_1(\text{rcet})) + 1 \cdot \frac{1}{b}$$

$$\frac{n}{b} < 1 / (3 + 0 \cdot 6) = \frac{1}{24} = 4.1\%$$

Abfrage muss weniger als 4.1% Tabelle liefern damit sich Index lohnt!

4. Achtung! Überindexierung kostet Zeit, Speicherplatz und kann Optimizer in die Irre führen.

Bsp.: SELECT DISTINCT s.Semester

```
FROM studenten S, Hören H, Vorlesung V,
Professoren P WHERE p.Name = 'sokl'
AND v.gelesenVon = p.PersNr
AND v.VorlesNr = h.VorlesNr
AND h.MatrNr = s.Matr.Nr;
```

II S.semester

$\sigma_p(\text{PersNr}) = v.\text{gelesenVon}$

$\sigma_v(\text{VorlesNr}) = h.\text{VorlesNr}$

$\sigma_h(\text{MatrNr}) = s.\text{Matr.Nr}$

$\sigma_v(\text{VorlesNr}) = h.\text{VorlesNr}$

$\sigma_s(\text{MatrNr}) = h.\text{Matr.Nr}$

$\sigma_v(\text{VorlesNr}) = h.\text{VorlesNr}$

$\sigma_p(\text{Name}) = 'sokl'$

$\sigma_v(\text{gelesenVon})$

$\sigma_h(\text{MatrNr})$

$\sigma_v(\text{VorlesNr})$

$\sigma_h(\text{VorlesNr})$

$\sigma_p(\text{PersNr})$

$\sigma_v(\text{gelesenVon})$

$\sigma_p(\text{Name})$

$\sigma_v(\text{VorlesNr})$

$\sigma_h(\text{MatrNr})$

$\sigma_v(\text{VorlesNr})$

$\sigma_h(\text{VorlesNr})$

$\sigma_p(\text{Name})$

$\sigma_v(\text{gelesenVon})$

$\sigma_h(\text{MatrNr})$

$\sigma_v(\text{VorlesNr})$

$\sigma_h(\text{VorlesNr})$



Optimierung: **Index-Scan:** nutzt Index zum Auslesen der Tupel (Externes) Sortieren = Merge-Sort in Sortierreihenfolge (Ausnahme: Hash-Index)

Relationen-Scan (full-table scan)

Bsp.: SELECT # FROM kunde WHERE Nachname BETWEEN 'Heuer' AND 'Panik'

1. currScanID ← open-rel-scan (KUNDE-RelationID)

2. currTID ← nextTID (currScanID)

while end-of-scan (currScanID) do

currRec ← fetch-tuple (KUNDE-RelationID, currTID)

if currRec.Nachname ≥ 'Heuer' AND currRec.Nachname < 'Panik' then put (currRec),

end

currTID ← nextTID (currScanID)

end

close-scan (currScanID)

Entschachtelung von Anfragen

• Erkennen von gemeinsamen Teilanfragen • Jede Anfrage in kanonische n-Relationen Es müssen alle Seiten geladen werden! □ Anfrage umformen

Besser durch Ausnutzung der Blockierung, ist aber unsortiert

n-Relationen mit n-1 Verbundprädikaten (geschachtelte

Iteration durch effizientere Verbundoperationen ersetzen

Typ-A-Schachtelung (Aggregation):

1. inner Block Q enthält kein Verbundprädikat, dass die äussere Relation referenziert

2. Q berechnet Aggregat

Bsp.: SELECT BestNr FROM Bestellung WHERE ProdNr = (SELECT MAX(ProdNr) FROM Produkt WHERE Preis < 15)

Ausführung: 1. Innere Anfrage + Aggregat berechnen dann 2. Ergebnis in äussere Anfrage einsetzen und berechnen

Typ-N-Schachtelung (Nested):

1. wie Typ-A. 2. Q berechnet kein Aggregat

Bsp.: Ref. Bsp Typ A, but instead "use IN"

Ausführung: Vor A: Innere Anfrage berechnen (Tabelle) und in Äussere einsetzen. Vor B: Entschachtelung

Anfrage Entschachtelt: SELECT BestNr FROM

Bestellung B JOIN Produkt P ON B.ProdNr =

P.ProdNr AND P.Preis < 15

Typ-N-Anfragen mit 'IN' Prädikaten der tiefen n-1 können semantische n-Rel. anfragen transformiert werden.

Typ-J-Schachtelung (Join)

1. Inner Block Q referenziert im Verbundprädikat die Relation des äusseren Blocks ('korrelierte Unterabfrage')

2. Q liefert keinen aggregationswert

Bsp.: SELECT BestNr FROM Bestellung B

COHERE B.ProdNr IN (SELECT ProdNr

FROM Lieferung L WHERE L.LiefNr = Q

B.LiefNr AND L.Datum = current_date)

Ausführung: Entschachteln

Anfrage Entschachtelt: SELECT BestNr

FROM Bestellung B JOIN Lieferung L

ON B.ProdNr = L.ProdNr AND L.Datum

= current_date

□ Forme x-operation, die von or gefolgt wird, wenn möglich um

□ Regel 8: Blattknoten so vertauschen, das kleinste Zwischenergebnis zuerst ausgewertet wird

□ Regel 3, 4, 7, 10: II-operationen so weit wie möglich nach unten

□ Regel 1: Konjunkturen or in Kaskaden von or zerlegt

□ Regel 5, 6, 9: or so weit nach unten wie möglich

□ Regel 11: M und/or S: Morgan's Regeln

□ Regel 12: Kartesisches Produkt gefolgt von or

kann in R gefolgt werden: or(R × S) = R M S

Anwendung der Alg. Regeln

a) Regel 1 → Konjunkturen or in Kaskaden von or

zerlegt + b) Regeln 4, 6, 9: or so weit nach unten wie möglich

c) Regel 8: Blattknoten so vertauschen, das kleinste Zwischenergebnis zuerst ausgewertet wird

d) Regel 3, 4, 7, 10: II-operationen so weit wie möglich nach unten

e) Regel 11: M und S: Morgan's Regeln

f) Regel 12: Kartesisches Produkt gefolgt von or

g) Regel 1: Konjunkturen or in Kaskaden von or

h) Regel 10: II-operationen so weit wie möglich nach unten

i) Regel 11: M und S: Morgan's Regeln

j) Regel 12: Kartesisches Produkt gefolgt von or

k) Regel 1: Konjunkturen or in Kaskaden von or

l) Regel 10: II-operationen so weit wie möglich nach unten

m) Regel 11: M und S: Morgan's Regeln

n) Regel 12: Kartesisches Produkt gefolgt von or

o) Regel 1: Konjunkturen or in Kaskaden von or

zurück

p) Regel 11: M und S: Morgan's Regeln

q) Regel 12: Kartesisches Produkt gefolgt von or

r) Regel 1: Konjunkturen or in Kaskaden von or

zurück

s) Regel 11: M und S: Morgan's Regeln

t) Regel 12: Kartesisches Produkt gefolgt von or

u) Regel 1: Konjunkturen or in Kaskaden von or

zurück

v) Regel 11: M und S: Morgan's Regeln

w) Regel 12: Kartesisches Produkt gefolgt von or

x) Regel 1: Konjunkturen or in Kaskaden von or

zurück

y) Regel 11: M und S: Morgan's Regeln

z) Regel 12: Kartesisches Produkt gefolgt von or

a) Regel 1: Konjunkturen or in Kaskaden von or

zurück

b) Regel 11: M und S: Morgan's Regeln

c) Regel 12: Kartesisches Produkt gefolgt von or

d) Regel 1: Konjunkturen or in Kaskaden von or

zurück

e) Regel 11: M und S: Morgan's Regeln

f) Regel 12: Kartesisches Produkt gefolgt von or

g) Regel 1: Konjunkturen or in Kaskaden von or

zurück

h) Regel 11: M und S: Morgan's Regeln

i) Regel 12: Kartesisches Produkt gefolgt von or

j) Regel 1: Konjunkturen or in Kaskaden von or

zurück

k) Regel 11: M und S: Morgan's Regeln

l) Regel 12: Kartesisches Produkt gefolgt von or

m) Regel 1: Konjunkturen or in Kaskaden von or

zurück

n) Regel 11: M und S: Morgan's Regeln

o) Regel 12: Kartesisches Produkt gefolgt von or

p) Regel 1: Konjunkturen or in Kaskaden von or

zurück

q) Regel 11: M und S: Morgan's Regeln

r) Regel 12: Kartesisches Produkt gefolgt von or

s) Regel 1: Konjunkturen or in Kaskaden von or

zurück

t) Regel 11: M und S: Morgan's Regeln

u) Regel 12: Kartesisches Produkt gefolgt von or

v) Regel 1: Konjunkturen or in Kaskaden von or

zurück

w) Regel 11: M und S: Morgan's Regeln

x) Regel 12: Kartesisches Produkt gefolgt von or

y) Regel 1: Konjunkturen or in Kaskaden von or

zurück

z) Regel 11: M und S: Morgan's Regeln

aa) Regel 12: Kartesisches Produkt gefolgt von or

bb) Regel 1: Konjunkturen or in Kaskaden von or

zurück

cc) Regel 11: M und S: Morgan's Regeln

dd) Regel 12: Kartesisches Produkt gefolgt von or

ee) Regel 1: Konjunkturen or in Kaskaden von or

zurück

ff) Regel 11: M und S: Morgan's Regeln

gg) Regel 12: Kartesisches Produkt gefolgt von or

hh) Regel 1: Konjunkturen or in Kaskaden von or

zurück

ii) Regel 11: M und S: Morgan's Regeln

jj) Regel 12: Kartesisches Produkt gefolgt von or

kk) Regel 1: Konjunkturen or in Kaskaden von or

zurück

ll) Regel 11: M und S: Morgan's Regeln

mm) Regel 12: Kartesisches Produkt gefolgt von or

nn) Regel 1: Konjunkturen or in Kaskaden von or

zurück

oo) Regel 11: M und S: Morgan's Regeln

pp) Regel 12: Kartesisches Produkt gefolgt von or

qq) Regel 1: Konjunkturen or in Kaskaden von or

zurück

rr) Regel 11: M und S: Morgan's Regeln

ss) Regel 12: Kartesisches Produkt gefolgt von or

tt) Regel 1: Konjunkturen or in Kaskaden von or

zurück

uu) Regel 11: M und S: Morgan's Regeln

vv) Regel 12: Kartesisches Produkt gefolgt von or

ww) Regel 1: Konjunkturen or in Kaskaden von or

zurück

xx) Regel 11: M und S: Morgan's Regeln

yy) Regel 12: Kartesisches Produkt gefolgt von or

zz) Regel 1: Konjunkturen or in Kaskaden von or

zurück

aa) Regel 11: M und S: Morgan's Regeln

bb) Regel 12: Kartesisches Produkt gefolgt von or

cc) Regel 1: Konjunkturen or in Kaskaden von or

zurück

dd) Regel 11: M und S: Morgan's Regeln

ee) Regel 12: Kartesisches Produkt gefolgt von or

ff) Regel 1: Konjunkturen or in Kaskaden von or

zurück

gg) Regel 11: M und S: Morgan's Regeln

hh) Regel 12: Kartesisches Produkt gefolgt von or

ii) Regel 1: Konjunkturen or in Kaskaden von or

zurück

jj) Regel 11: M und S: Morgan's Regeln

kk) Regel 12: Kartesisches Produkt gefolgt von or

ll) Regel 1: Konjunkturen or in Kaskaden von or

zurück

mm) Regel 11: M und S: Morgan's Regeln

nn) Regel 12: Kartesisches Produkt gefolgt von or

oo) Regel 1: Konjunkturen or in Kaskaden von or

zurück

pp) Regel 11: M und S: Morgan's Regeln

qq) Regel 12: Kartesisches Produkt gefolgt von or

rr) Regel 1: Konjunkturen or in Kaskaden von or

zurück

ss) Regel 11: M und S: Morgan's Regeln

tt) Regel 12: Kartesisches Produkt gefolgt von or

uu) Regel 1: Konjunkturen or in Kaskaden von or

zurück

vv) Regel 11: M und S: Morgan's Regeln

ww) Regel 12: Kartesisches Produkt gefolgt von or

xx) Regel 1: Konjunkturen or in Kaskaden von or

zurück

yy) Regel 11: M und S: Morgan's Regeln

zz) Regel 12: Kartesisches Produkt gefolgt von or

aa) Regel 1: Konjunkturen or in Kaskaden von or

zurück

bb) Regel 11: M und S: Morgan's Regeln

cc) Regel 12: Kartesisches Produkt gefolgt von or

dd) Regel 1: Konjunkturen or in Kaskaden von or

zurück

ee) Regel 11: M und S: Morgan's Regeln

ff) Regel 12: Kartesisches Produkt gefolgt von or

gg) Regel 1: Konjunkturen or in Kaskaden von or

zurück

hh) Regel 11: M und S: Morgan's Regeln

ii) Regel 12: Kartesisches Produkt gefolgt von or

jj) Regel 1: Konjunkturen or in Kaskaden von or

zurück

kk) Regel 11: M und S: Morgan's Regeln

ll) Regel 12: Kartesisches Produkt gefolgt von or

mm) Regel 1: Konjunkturen or in Kaskaden von or

zurück

nn) Regel 11: M und S: Morgan's Regeln

oo) Regel 12: Kartesisches Produkt gefolgt von or

pp) Regel 1: Konjunkturen or in Kaskaden von or

zurück

qq) Regel 11: M und S: Morgan's Regeln

rr) Regel 12: Kartesisches Produkt gefolgt von or

ss) Regel 1: Konjunkturen or in Kaskaden von or

zurück

tt) Regel 11: M und S: Morgan's Regeln

uu) Regel 12: Kartesisches Produkt gefolgt von or

vv) Regel 1: Konjunkturen or in Kaskaden von or

zurück

ww) Regel 11: M und S: Morgan's Regeln

xx) Regel 12: Kartesisches Produkt gefolgt von or

yy) Regel 1: Konjunkturen or in Kaskaden von or

zurück

zz) Regel 11: M und S: Morgan's Regeln

aa) Regel 12: Kartesisches Produkt gefolgt von or

bb) Regel 1: Konjunkturen or in Kaskaden von or

zurück

cc) Regel 11: M und S: Morgan's Regeln

dd) Regel 12: Kartesisches Produkt gefolgt von or

ee) Regel 1: Konjunkturen or in Kaskaden von or

zurück

ff) Regel 11: M und S: Morgan's Regeln

gg) Regel 12: Kartesisches Produkt gefolgt von or

hh) Regel 1: Konjunkturen or in Kaskaden von or

zurück

ii) Regel 11: M und S: Morgan's Regeln

jj) Regel 12: Kartesisches Produkt gefolgt von or

kk) Regel 1: Konjunkturen or in Kaskaden von or

zurück

ll) Regel 11: M und S: Morgan's Regeln

mm) Regel 12: Kartesisches Produkt gefolgt von or

nn) Regel 1: Konjunkturen or in Kaskaden von or

zurück

oo) Regel 11: M und S: Morgan's Regeln

<p

Fehlerklassifikation: (durch Recovery verursacht)

- Transaktionsfehler:** • Abbruch der jeweiligen Transa. • kein Einfluss aufs restliche System (lokaler Fehler)
- Typisch:** Anwendungsprogramm (z.B. divide by 0)
- Transa. abbruch durch Benutzer (z.B. Time-out, invalid entry)
- Transa. abbruch durch System (z.B. Deadlock, ACID, zuviel Lösung: isolates Zurücksetzen aller Änderungen von abgebrochenen Transa. (Atomicity needs "redo")

- Systemfehler:** Folge: Zerstörung der Daten im Hauptspeicher (flüchtige DB) • NICHT BETROFFEN: Hintergrundspeicher (permanente DB)
- Typisch:** • RDBMS-Fehler (config file Fehlerhaft) • Betriebssystemfehler (Seitenfehler, ungültiger Befehl) • Hardware Fehler (Stromausfall, Fehler im Memory)
- Lösung:** Nachvollziehen der Änderungen, die von abgeschlossenen Transa. nicht in die DB eingebroht wurden (REDO) • Zurücksetzen von Änderungen von nicht abgeschlossenen Transa. (undo)

Mediafehler: • Verlust der Daten der stabilen DB

Typisch: • "Head-Crashes", Controller-Fehler • Naturgewalten (Feuer, Erdbeben) • Operator-Fehler

Lösung: Voraussetzung: Sicherungskopien (DB-Backup, Log-Backup) auf sep. Medium • Einspielen vom BackUp • Anwendung aller Transactionlogs, die seit dem Anlegen vom BackUp erzeugt worden sind • DB muss durch UNDO und REDO in einen konsistenten Zustand gebracht werden!

DB-Backup: 1x Tag/Woche
Log-Backup: 1x 30 Minuten!

Operative Daten (OLTP)	
Ziel	• Abwicklung der Geschäftsprozesse
Ausrichtung	• Detaillierte, granulare Geschäftsvorfall daten
Zeitbezug	• Aktuell • Zeitpunktbezogen • Auf die Transa. ausgerichtet
Modellierung	• Altbestände oft nicht modelliert (funktionsorientiert)
Zustand	• Häufig redundant • Inkonsistenzen
Update	• Langsam und konkurrenzfrei
Quellen	• Strukturiert • Meist statisch im Programmcode

Sicherungspunkte

Problem bei Recovery: Alter der ältesten Seite im Puffer nicht bekannt. Gesamtes Log-Buch (seit Letztem DB-Backup) muss eingespielt werden → **Hoher Zeitaufwand!**

Lösung: Sicherungspunkte (SP) (Checkpoints). Aufbau vom SP:
1. Geänderte Seiten (dirty Pages) vom Puffer (flüchtige DB) in die Persistente DB geschrieben.
2. Puffer vom Log auf stabilen Speicher. **3.** SP im Log registriert
Arten von SP:
1. Transaction-konsistente SP. **2.** Aktionskonsist. SP (einzelner SQL-Befehl)
3. Unscharfe SP

Transaktionskonsistente SP:

- Alle Änderungen werden in einem Moment, in dem keine Transa. aktiv sind, vom Puffer in die DB geschrieben

Ablauf: 1. SP anmelden

- Neu ankommende Transa. müssen warten.
 - Aktive Transa. zuende führen.
 - Geänderte Seiten auf Platte schreiben.
 - SP im Log-Buch registrieren.
- ⊕ Späteres Recovery keine Veränderung vor diesem Punkt berücksichtigen
- ⊕ Lösigt Benutzer unter Umständen lange warten (unscharfe SP)

Non-Standard Datenbanken:

Data-Warehouse:

Operative Daten: Generiert von Admin-, Plan-, Abrechnungssystemen
• Große Teile von OLTP-Systemen erzeugt
OLTP (Online Transaction Processing - Taxis geschäft)
• Mehrere Benutzer greifen das selbe System und DB zu (Auskunft, Bücher, etc.)

Dispositive Daten: • Unterstützen das Management (OLAP-Systeme). Direkter Durchgriff von Mgmt.-Unterstützenden Systemen nicht zielführend. • Werden aus operativen Daten erzeugt.

OLAP (Online Analytical Processing)
↳ Entscheidungsunterstützend

Begriffsvielfalt: OLAP sind Softwaresysteme, die für menschliche Entscheidungsträger relevante Informationen für operative und strategische Aufgaben ermittelt, aufbereitet, übersichtlich zusammenstellt und bei der Auswertung hilft - Wikipedia

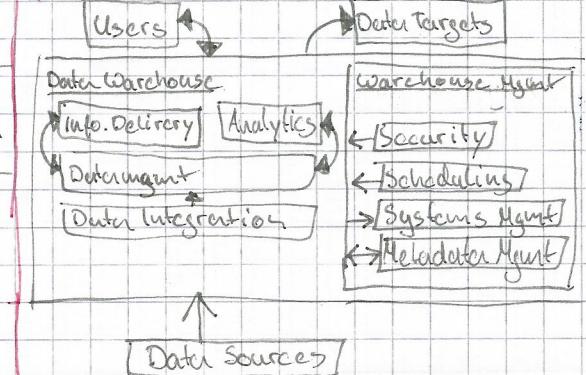
System	Einführung	Fokus
Decision Support System (DSS)	Anfang 1970	Modellfokus
Executive Information Systems	Ende 1980	Präsentationsfokus
DataWarehouse (DW)	Anfang 1990	Datenfokus
OLAP	Anfang 1990	Modellfokus
BI	Anfang 1990	Präz.-Fokus
Advanced Analytics	Ab ca. 2011	Analysefokus

Data Warehouse:

"Themenorientierte, zeitorientierte, integrierte und unverzerrte Datensammlung, deren Daten sich für Mgmt.-entscheidungen auswerten lassen" innen 1994

Merkmale: • strikte Trennung von operativen und dispositiven Daten • Integration vielfältiger (auch externer) Datenquellen • Langfristige Datenaufbewahrung, Historisierung • Mehrdimensionale Auswertung • Detailliert und verdichtete Daten • hohe Datenvolumen (keine Löschung) • immer unterschneidet/ anwenderspezifisch angebaut.

DW/H Referenzarchitektur



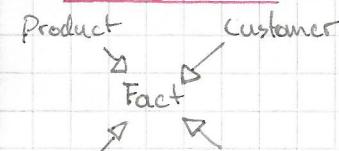
Data-Ware

- Langfristig gehaltener Datenbestand innerhalb eines DW/H (Kopie vom Raum, Zeit, Organisation...)
- ↳ Geschaffen für bestimmte Anwendung
- Organisationsbereich • Viele Architekturvarianten, Übergänge oft nicht scharf.
- "Analysedatenbanken"
- ↳ können auch Excel-File sein

Drill-Down • Hierarchie von oben nach unten traversieren • "Verfeinerung" entlang einer Dimension

Roll-up • Hierarchie von unten nach oben traversieren • "Aggregation" entlang einer Dimension

Starschema



Faktentabelle: Enthält Fakten und Fremdschlüssel auf Dimensionstabellen

Dimensionstabelle: Enthält Dimension und Primärschlüssel, die mit Faktentabelle "verknüpft" ist

No-SQL (Not only) • nicht relational • schemafrei • einfache API • Auf horizontale Skalierung ausgerichtet • einfache Datenreplication • Konsistenzmödell BASE und Eventual Consistency

Core NoSQL:

- Key / Value Stores
- Wide Column Stores
- Document Stores
- Graphdatenbank

Soft NoSQL:

- Obj. DB
- Grid/Cloud DBAnw.

Multidimensionalität

- Unterscheidung von Fakten (gemessene Werte) und Dimensionen (Raum, Zeit, Organisation...)
- Hierarchische Struktur von Daten.

- Fakten (Measures):**
- Eigentliche Obj. der Analyse
 - Ansammlung von Einzelmesswerten / Kennzahlen, verknüpft mit Kontext (Dimension)
 - Jedes Faktum stellt einen Businesswert, eine Transaktion oder ein Ereignis dar.
 - I.d.R. numerische Werte (Verkaufszahlen, Lagerbestand, Umsatz, Gewinn etc.)
 - Analyse der Fakten soll es dem Nutz. ermöglichen die Unternehmensleistung zu überwachen/steuern
 - Numerische Werte lassen sich auf verschiedene Arten verdichten / aggregieren.

Dimensionen:

- Kontext der Daten
- Eindeutige Strukturierung des Datenraums
- Hoffentlich orthogonal (unabhängig) • Verfügen über Attrib., welche geordnet werden können
- Anzahl nicht beschränkt • 'Zeit' ist häufig

No-Standard DB-Systeme (alle die nicht stand)

- Quasi Echtzeit • Riesiges Datenvolumen (Petabytes)
- Lange Transa. (z.B. Konstruktionsbereich)
- Abbildung obj.-orientierter Konstrukte (z.B. Vererbung, Polymorphismus) und Speicherung von Objekten

No-SQL: CAP

Consistency: Alle Knoten sehen zu jeder Zeit die selben Daten. ACID hat nur innere Konsistenz

Availability: Ausgefallene Knoten beeinflussen die Verfügbarkeit anderer nicht

Partition Tolerance: System arbeitet trotz Verlust von Nachrichten, einzelner Knoten (Partitionen) vom Netz weiter

Anwendung: • Klassische vert. Systeme ist 'C' am wichtigsten → C.U.A eingeschränkt • Web-Anwendungen können Fokus auf A + P, dabei Einschränkung von 'C'

Bsp.: DVS → AP, RDBMS → CA

Geldautomat → CP

Eventual Consistency (BAS E):

- Konsistenz erst nach definiertem Zeitraum erreicht, nicht nach jeder Transa.

Big Data (5-V)

Volume: Tera - bis Petabyte

Velocity: Hohe Stream/Verarbeitungsgeschwindigkeit • Viele Datensätze pro Tag/Stunde/Minute

Variety: Unterschiedliche Datenquellen (Text, Bilder, Videos, Blogs etc.)

Veracity: Schwankende Datenqualität

Value: Wert des Daten

Herausforderung: Erfassung, Speicherung, Auswertung und Visualisierung von Daten

"Standard - DB - Systeme" Anforderungen

- Anwendungen kommen aus dem Betriebswirtschaftlichen Umfeld.
- Transaktionssicherheit • Hohe Performance bei Search/Update • Verwaltung 'einfacher', stark strukturierter Daten
- i.d.R. zentraler DB-Server
- Relationale DB-Systeme

↳ Grenzen: • Fokus auf strukturierte Daten • Begrenzte Skalierbarkeit • Schemaänderungen sind komplex

Skalierung

Vertikal: Leistungssteigerung durch hinzufügen von Ressourcen (z.B. +CPU, +RAM, +HDD, etc) • Unabhängig von der Implementierung möglich. Begrenzt durch Stand der Technik und Kosten • Transparent für DBMS

• Admin. aufwand ist konstant.

• **Hardwarekosten** ◉ Skalierung nur in größeren Stufen möglich

↳ höhere Kosten für ungenutzte Leistung

Horizontal: • Hinzufügen von Rechnern/Knoten. • System muss für Parallelisierbarkeit (Anwendung) ausgerichtet sein ◉ Kostengünstig ◉ Skalierung in kleinen Stufen

Last/Datenverteilung notwendig

◉ ggf. verteilte Protokolle

◉ Höhere Fehlerrate ◉ Höherer Administrationsaufwand

DAB 2 Zusammenfassung:

- Trigger:
- Code der vom DBMS ausgeführt wird
 - Ausführung auf Grund von sich ändernden Datenbankzuständen
- Prinzip: ECA
- Event - condition
- Action
- beeignet für
 - Realisierung von Integritätsbedingungen
 - Berechnungen (z.B. nachführen von Summen)
 - Protokollieren von Datenänderungen
 - + Entlastung der Anwendungsprogramme \circlearrowleft Fehlerbehandlung
 - + effiziente Ausführung möglich \circlearrowleft Testen, Debuggen
 - + einfacher Rollback \circlearrowleft Ergebnis event. abhängig von Aufruf-Reihenfolge \circlearrowleft Terminierung von geschachtelten Triggern
- SQL: CREATE / DROP / ALTER Trigger...
- \hookrightarrow Lösung: BEFORE / AFTER / INSTEAD OF

- INSTEAD OF:
- + Verhindern unerlaubte DML-Operationen
 - + SQL-constraints nicht ausreichend
 - + Nicht änderbare Sichten "änderbar" machen

- AFTER:
- + weitere operationen auslösen
 - + Historisierung

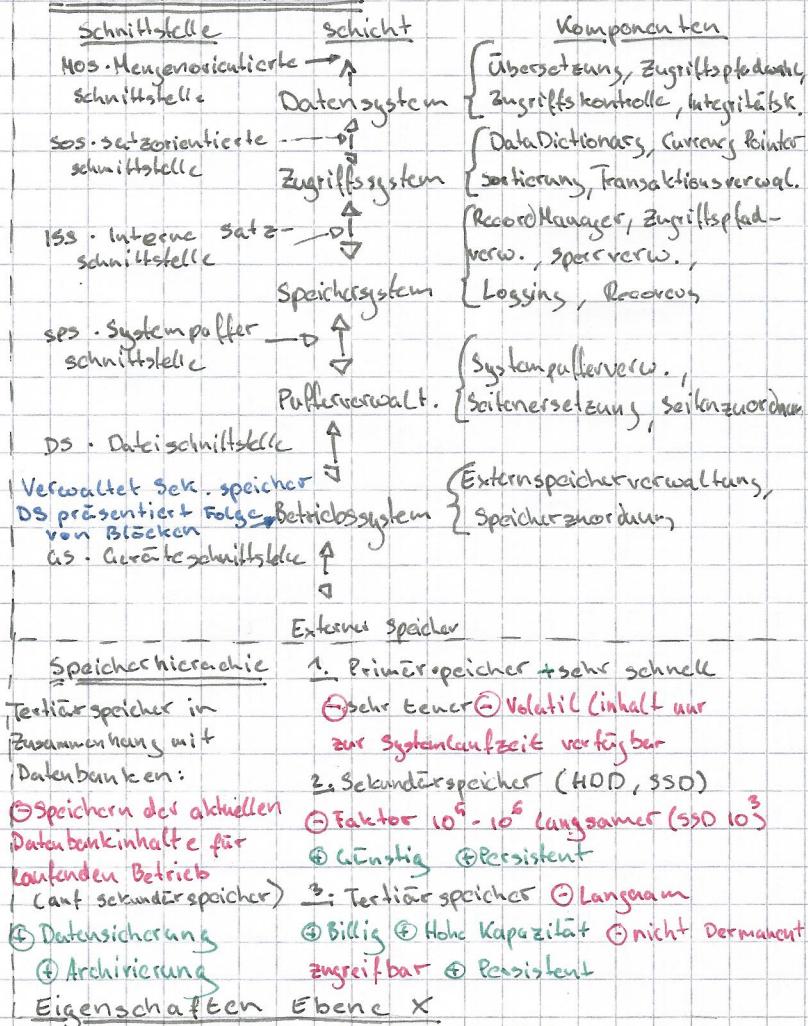
SQL Example:

```

CREATE TRIGGER NewPODetail
ON Purchasing.PurchaseOrderDetail
AFTER INSERT Table to watch
AS
IF @@ROWCOUNT = 1
BEGIN
    UPDATE Purchasing.PurchaseOrderHeader
    SET Subtotal = Subtotal + LineTotal
    FROM inserted
    WHERE PurchaseOrderHeader.PurchaseOrderID =
        inserted.PurchaseOrderID
END;

```

5-Schichten-Architektur:



Speicher- und Zugriffszeit: $X < X+1$

Kosten pro Speicherplatz: $X > X+1$

Kapazität: $X < X+1$

Lebensdauer: $X < X+1$

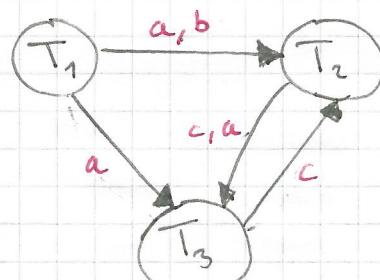
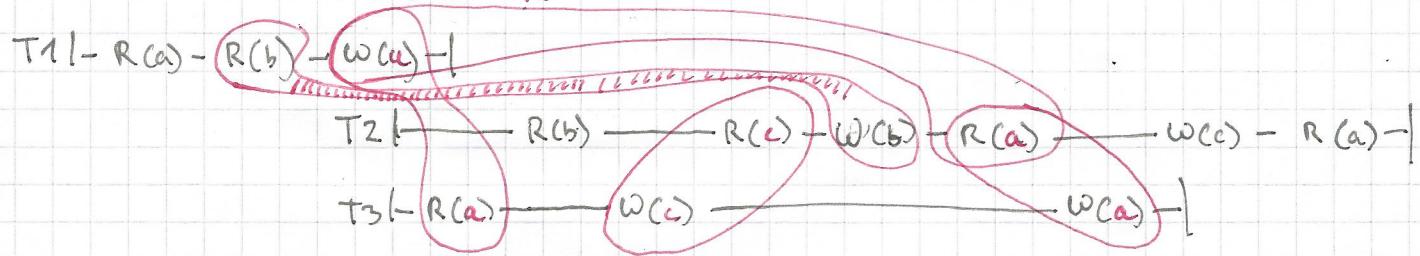
Problem: Zugriffslücke zwischen Primärspeicher und Sekundärspeicher ca. 10^5

Caching kann helfen wenn 1) Zeitliche Lokalität (in kurzer Zeit auf gleiche Daten zugreifen) und Räumliche Lokalität (Data Cluster: Zusammen angefragte Daten sind auf dem Sekundärspeicher eng zusammen)

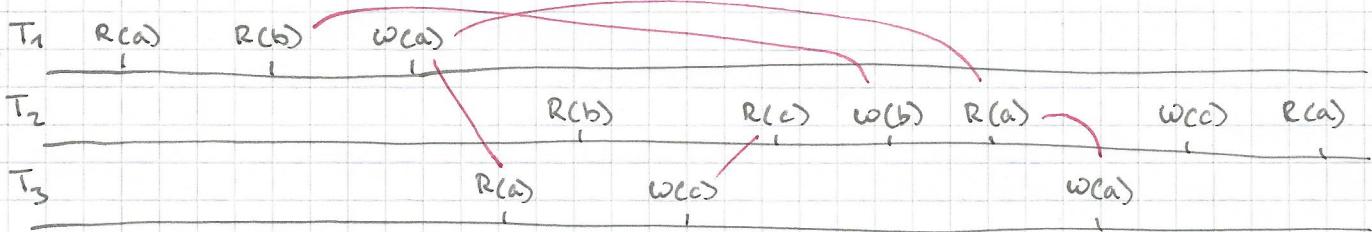
DAB 2 Praktikum 6

1.

Schreib/Lese Konf.:

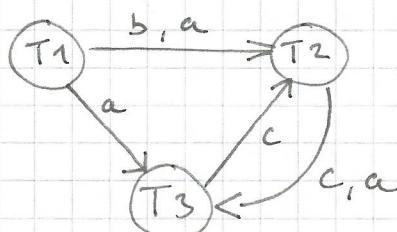


Nicht Schreibbar da Zyklus existiert!



Timingdiagramm ↗

Abhängigkeitsgraph
gemäßes Timingdiagramm →



Nicht Schreibbar weil Zyklus existiert

DAB2 Queries

```
CREATE TABLE Tname (  
    column_name type,  
    "           NVARCHAR(n),  
    CONSTRAINT PK_Tname PRIMARY KEY (column_name)
```

UPDATE schema.tName SET cName = ? WHERE condition;
JOIN → SELECT * FROM sName.tName JOIN sName.tName2 TABLES
ON sName.tName.cName = sName.tName2.cName2; MATCH columns
WHERE condition;

ALTER TABLE Tname.
 ALTER COLUMN cName Type //change datatype of cName
 ADD " " "
 DROP " "
 ADD CONSTRAINT constraint-name FOREIGN KEY (cName) } create new FF
 REFERENCES schema-name.table-name (column-name) }

```
INSERT INTO tName (cName1, cName2) VALUES (v1, v2);
```