

PROP

Performance- und Ressourcen-optimierte Programmierung

cps4it

consulting, projektmanagement und seminare für die informationstechnologie

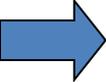
Ralf Seidler, Stromberger Straße 36A, 55411 Bingen

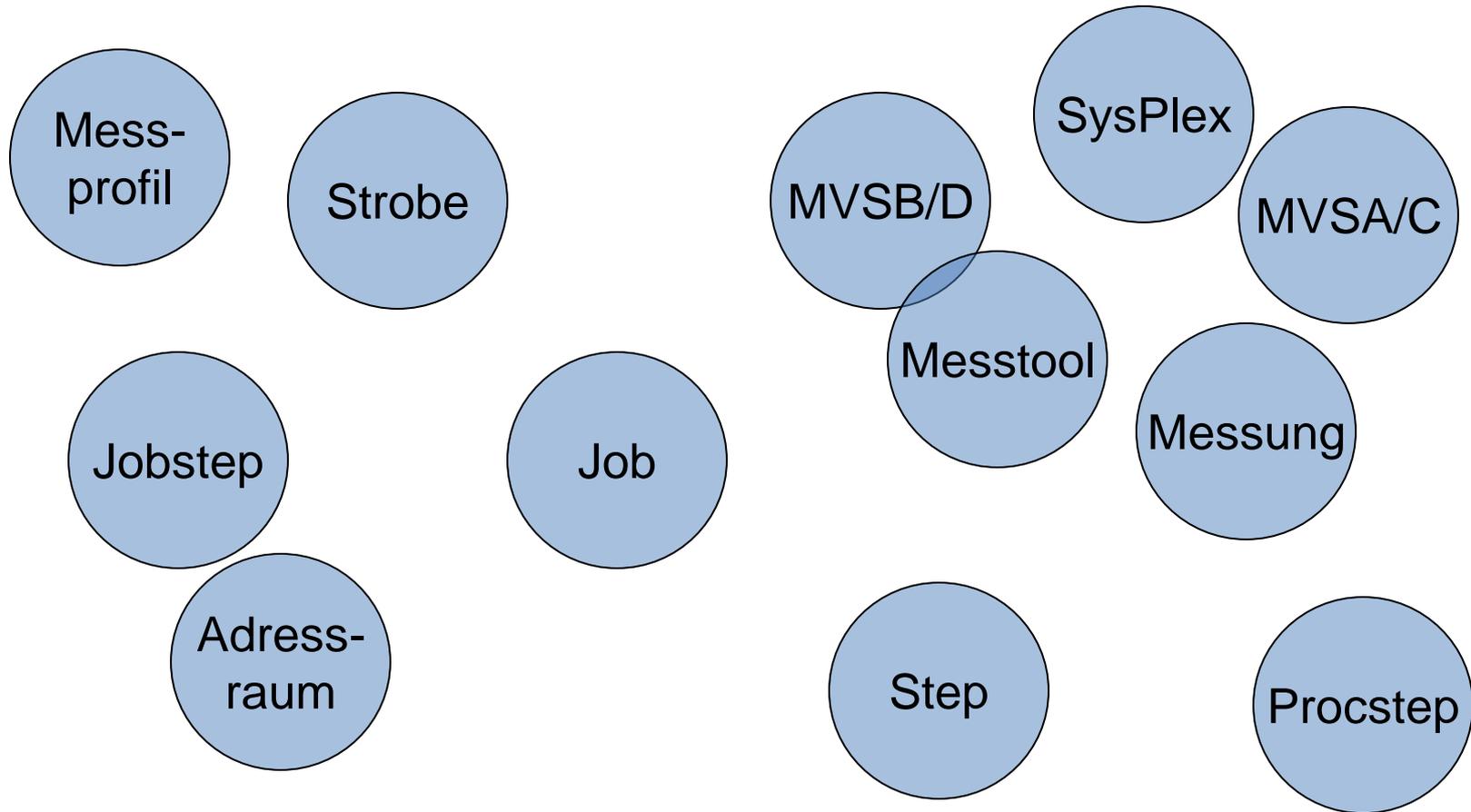
Fon: +49-6721-992611, Fax: +49-6721-992613, Mail: ralf.seidler@cps4it.de

Internet: <http://www.cps4it.de>

- Möglichkeiten, performante Anwendungen in COBOL zu erstellen und zu warten, kennen lernen
 - COBOL-Code
 - Compile-Optionen und LE-Optionen
 - DB2-SQLs und DB2-Umgebung
- Grundbegriffe eines Messtools und seinen Umgang kennen lernen
 - einfache Messungen analysieren

-
- Seite 5: Vorstellung und Einführung
 - Seite 17: Optimierungen – Beispiele / Potential
 - Seite 33: Richtlinien
 - Seite 41: Modellierung und DB2-Zugriffe
 - Seite 89: COBOL–Felder – COBOL-Befehle
 - Seite 105: Optionen – COBOL – LE
 - Seite 137: Informationen und Tools bei Firma X
 - Seite 149: Strobe – Handling und Interpretation

- 
- A blue arrow pointing to the right, highlighting the first item in the list.
- Vorstellung und Einführung
 - Optimierungen – Beispiele und Potential
 - Richtlinien
 - Modellierung und DB2-Zugriffe
 - COBOL–Felder – COBOL-Befehle
 - Auswirkungen von Optionen – COBOL – LE
 - Informationen und Tools bei Firma X
 - Strobe – Handling und Interpretation
 - Diskussion - Austausch



- PMA
 - Performance Management für Anwendungen
- PROP
 - Performance- und Ressourcenorientierte Programmierung
- APM
 - Application Performance Management

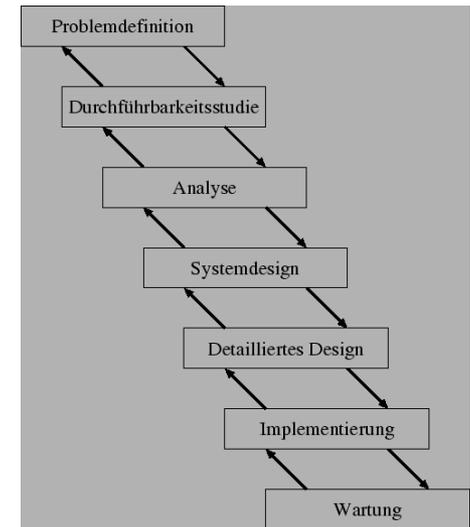
- **Optimierungen - Potential allgemein**
 - <http://www-1.ibm.com/support/docview.wss?uid=swg27001475&aid=1> (Perf.Paper von IBM)
- **COBOL Compile Options**
 - Enterprise COBOL for z/OS and OS/390 Programming Guide Version 3 Release 4: Kapitel 2.4 und Kapitel 7
- **LE Options**
 - Language Environment for OS/390 & VM Programming Reference: Kapitel 2
- **COBOL–Code**
 - Enterprise COBOL for z/OS and OS/390 Programming Guide Version 3 Release 4: Kapitel 7
 - <http://www-1.ibm.com/support/docview.wss?uid=swg27001475&aid=1> (Perf.Paper von IBM)
- **DB2**
 - DB2 Version 9.1 for z/OS Performance Monitoring and Tuning Guide (SC18-9851)
<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db29.doc.perf/dsnpfk16.pdf?noframes=true>
 - DB2 Version 9.1 for z/OS Monitoring and tuning DB2 performance
http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db29.doc.perf/db2z_perf.htm

- Strobe – Handling
 - <http://frontline.compuware.com/>
nur für registrierte Benutzer
 - STROBE MVS User's Guide
 - STROBE for CICS / for DB2 / for IMS
 - Bookmanager im Hause
- Strobe – Interpretation
 - !! Es gibt keine Literatur vom Hersteller, wie Strobe-Profile interpretiert werden !!
- PMA bei der Firma X
 - vielleicht bald im Intranet ;-)



Lifecycle der Anwendungen (Wasserfallmodell / V-Modell)

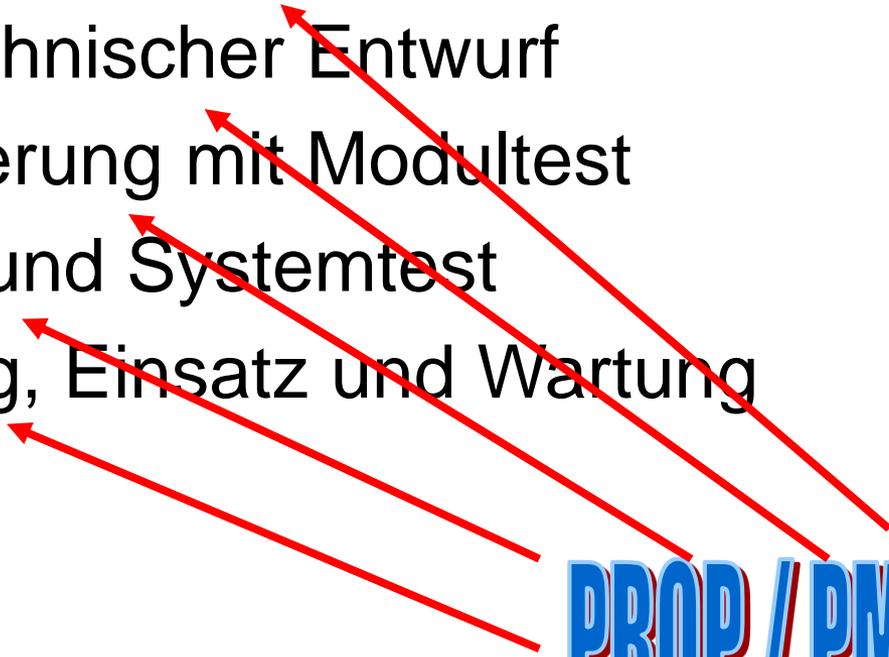
- Planung / Grobentwurf
- Analyse / Fachentwurf
- Design / technischer Entwurf
- Programmierung mit Modultest
- Integration und Systemtest
- Auslieferung, Einsatz und Wartung



Lifecycle der Anwendungen (Wasserfallmodell / V-Modell)

- Planung / Grobentwurf
- Analyse / Fachentwurf
- Design / technischer Entwurf
- Programmierung mit Modultest
- Integration und Systemtest
- Auslieferung, Einsatz und Wartung

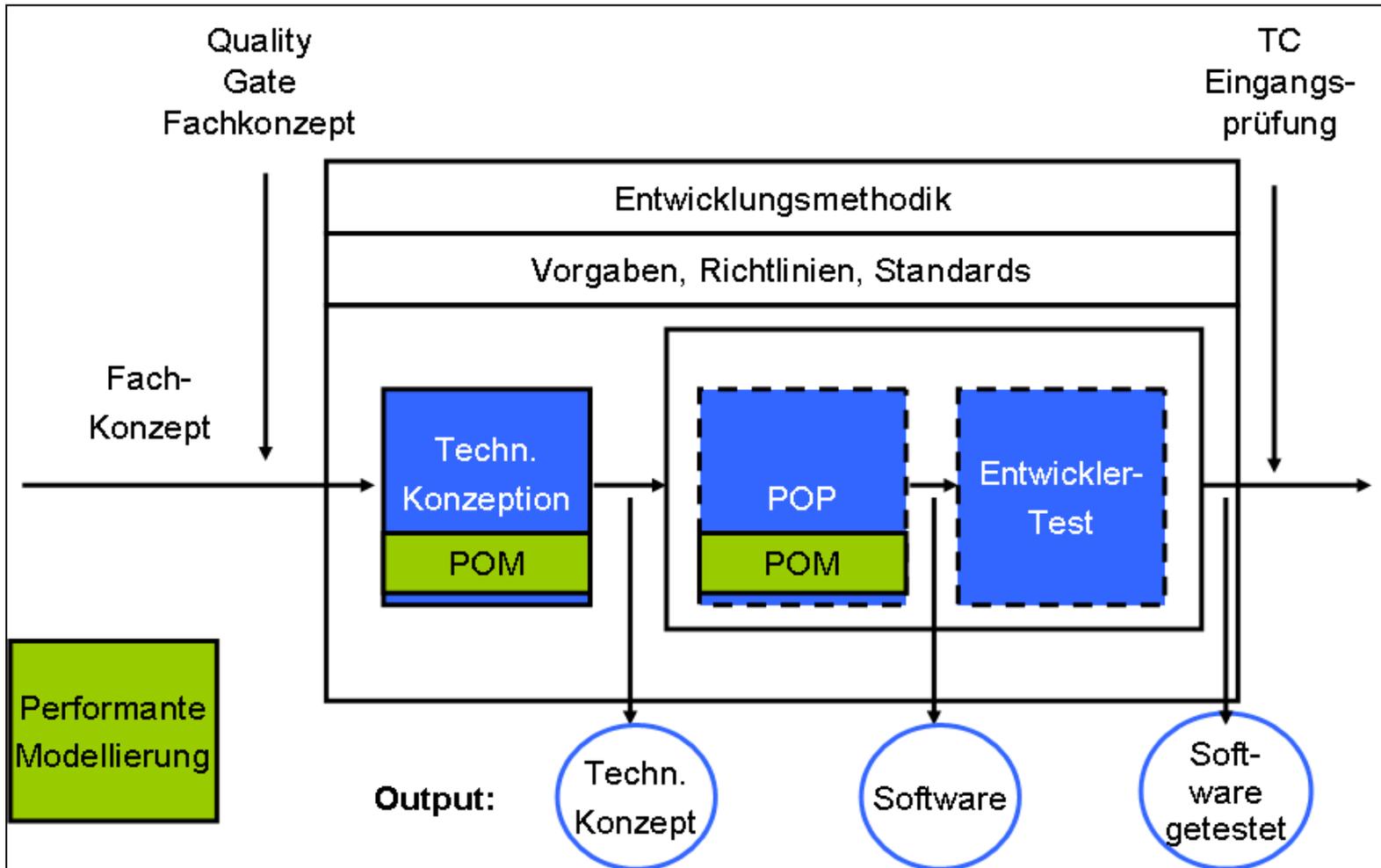
PROP / PMA / APM

Five red arrows originate from the text 'PROP / PMA / APM' and point to the following list items: 'Design / technischer Entwurf', 'Programmierung mit Modultest', 'Integration und Systemtest', 'Auslieferung, Einsatz und Wartung', and 'Auslieferung, Einsatz und Wartung'.

Lifecycle der Anwendungen (allgemein)

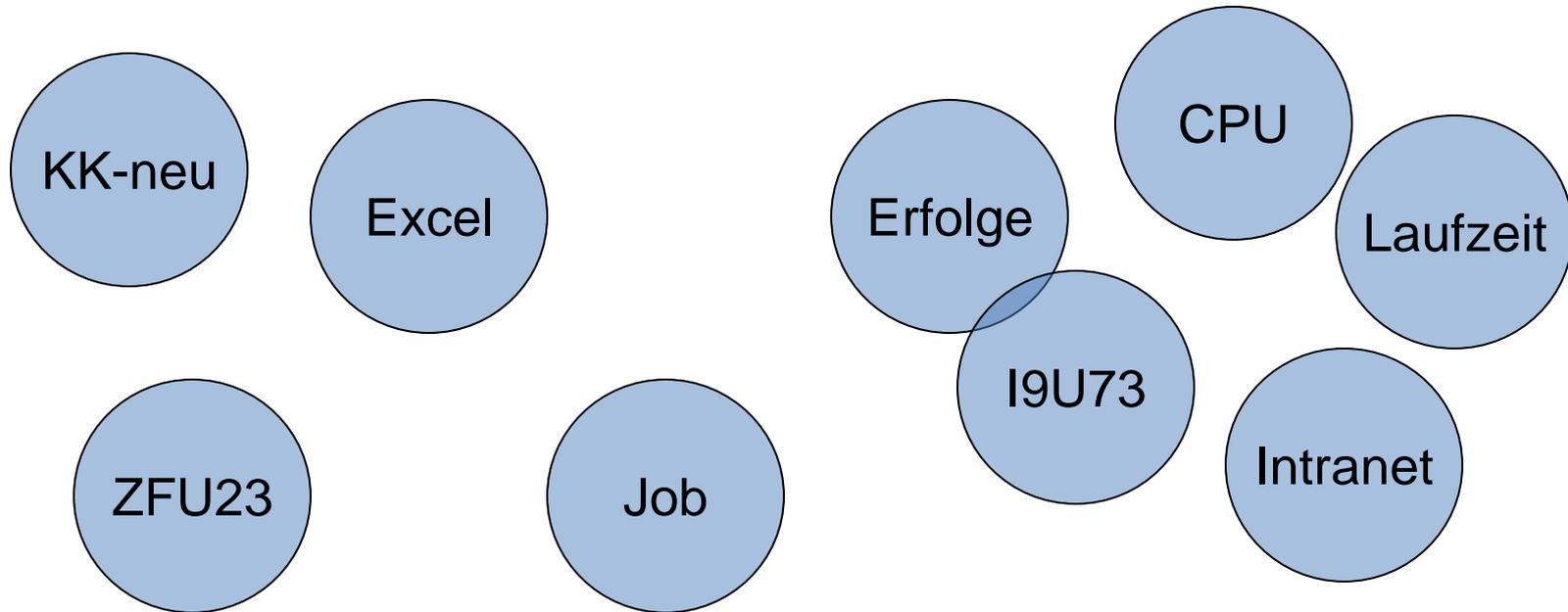
- Anwendungsentwicklung
 - Fachlicher Entwurf
 - Technische Konzeption
 - Programmierung / Umwandlung
 - Modultest / Massentest
 - Systemtest / Regressionstest / Massentest
- Produktion
 - Nachbereitung Einführung
 - Überwachung

Firma X-Schaubild



-
- Vorstellung und Einführung
 - ➔ • Optimierungen – Beispiele und Potential
 - Richtlinien
 - Modellierung und DB2-Zugriffe
 - COBOL–Felder – COBOL-Befehle
 - Auswirkungen von Optionen – COBOL – LE
 - Informationen und Tools bei Firma X
 - Strobe – Handling und Interpretation
 - Diskussion - Austausch

Begriffe

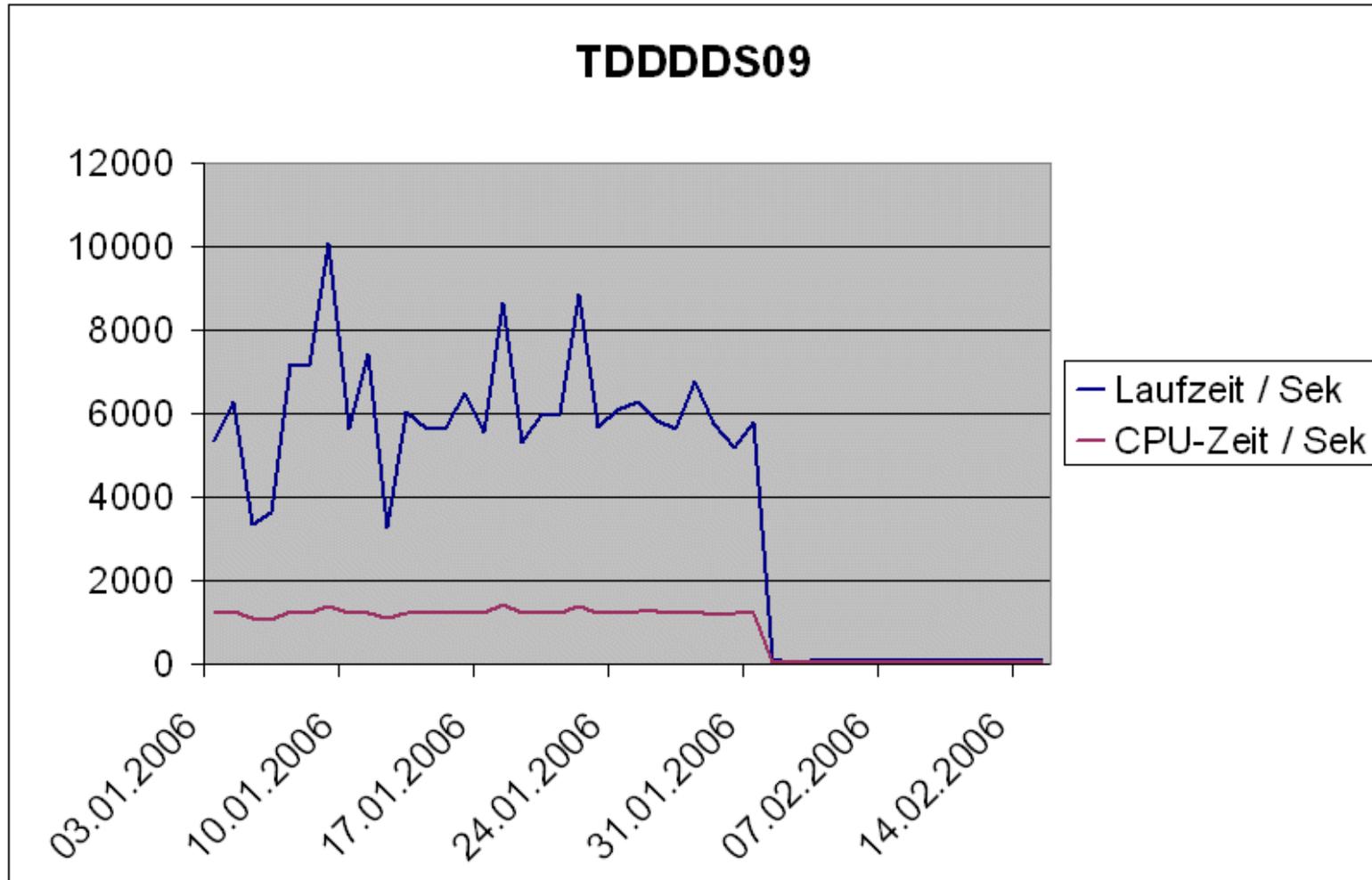


Optimierungen – Beispiele und Potential

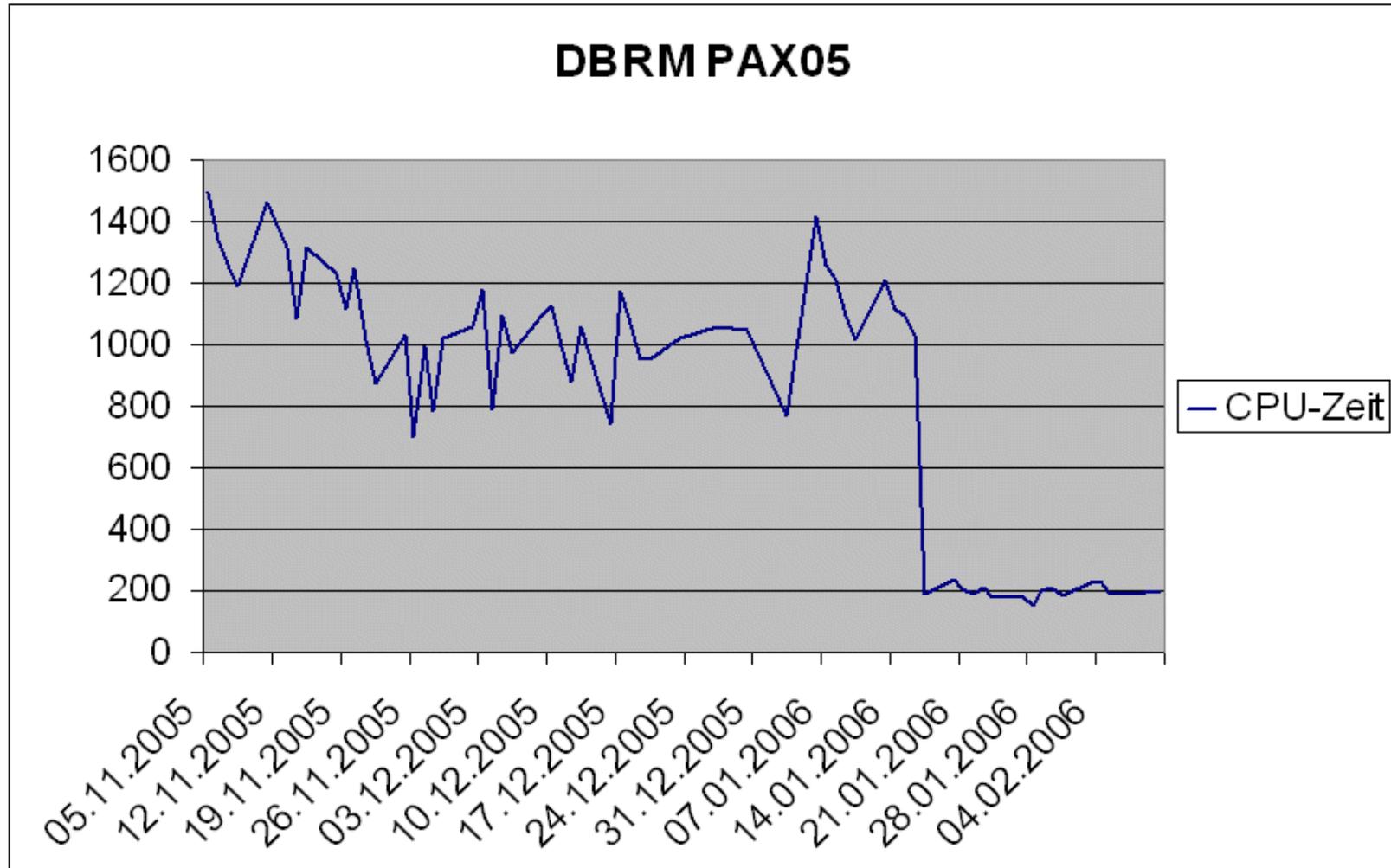
Beispiel 1 – Aufrufhäufigkeit zentrales Modul

Programm / Job	CPU Ersparnis (hochgerechnet auf 1 Jahr)	Laufzeit Ersparnis (hochgerechnet auf 1 Jahr)	Art der Optimierung
I9L55	1000 Stunden	(Online)	versch.
GTE71	300 Stunden	(Online)	Index eingeführt
DO692/PDODO692	700 Stunden	850 Stunden	Reorganisation DB2-Table
I9S61 / IMS	240 Stunden	(Online)	Loop (Code).
N1451 / TN1451*	270 Stunden	336 Stunden	Aufrufhäufigkeit N2U73
GT500 / MGT500%1	--- ⁽³⁾	--- ⁽³⁾	RUNSTATS, REBIND
BF001 / MBFBF001	240 Stunden	310 Stunden	DB2-Optimierung
N2735 / TN2735*	625 Stunden	667 Stunden	Aufrufhäufigkeit I9U73
Posy / TPOPO001	---	30 Stunden	BUFNO=16
IK019 / TIKIK019	40 Stunden	120 Stunden	Aufrufhäufigkeit I9U73
	3.317.559,12 €⁽¹⁾		

Beispiel 2 – SQL-Änderung Tagesjob



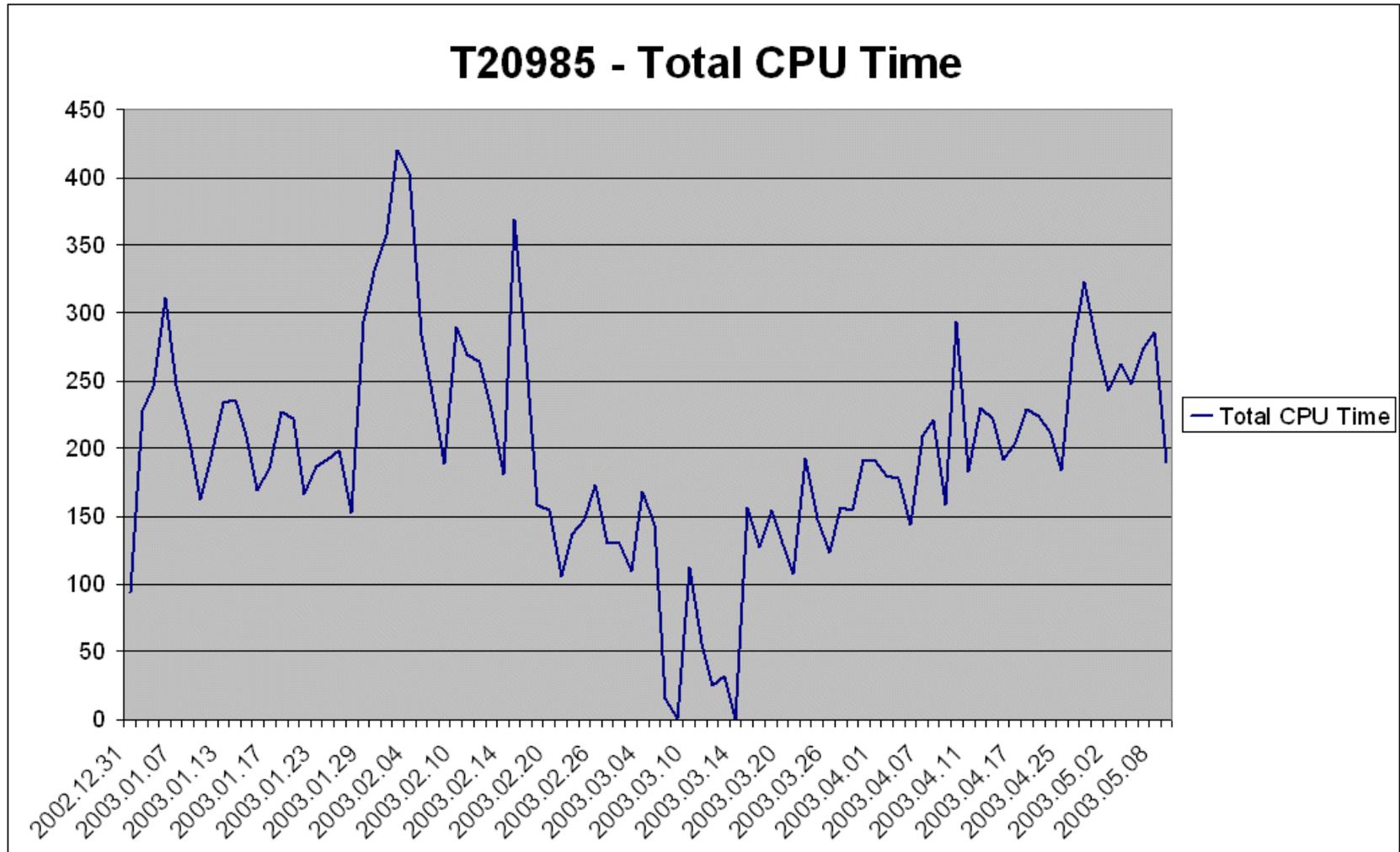
Beispiel 3 – SQL-Änderung zentrales Modul



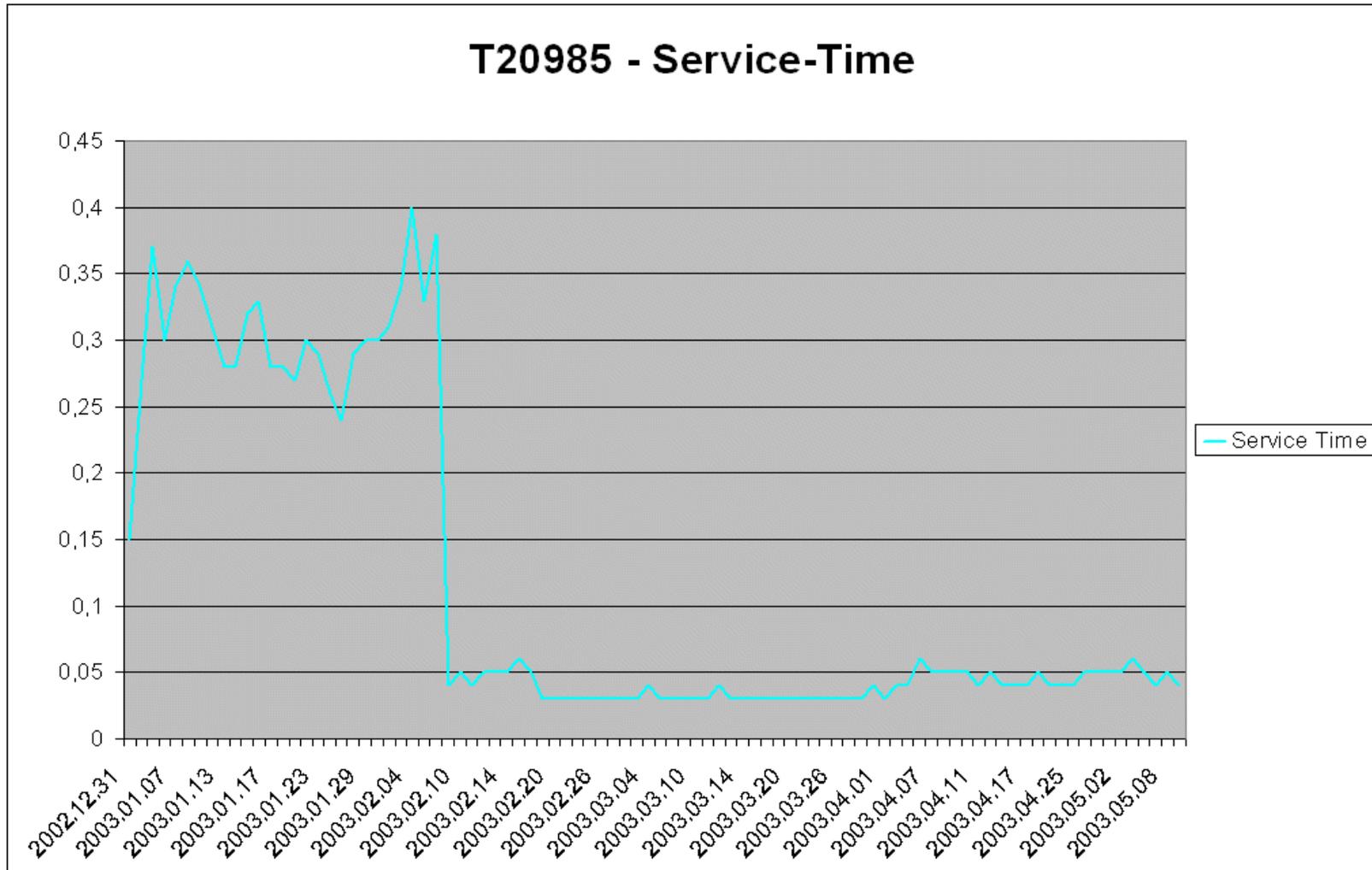
Beispiel 4 – Datumsroutine unter C / LE

- „zufällig“ Anfrage wegen Routine
- mehr als 5 Mio. Aufrufe pro Tag
 - Auswertung(en) für Vorstand ☹
- GETMAIN / FREEMAIN
- LE-Enclave für C aufgebaut / abgebaut
- Optimierungsversuche (ca. ½ Jahr)
- Umschreiben auf COBOL brachte Erfolg
- „Einsparung“ ca. 3.500 CPU-Stunden p.a.
 - ca. 1,5 Prozessoren

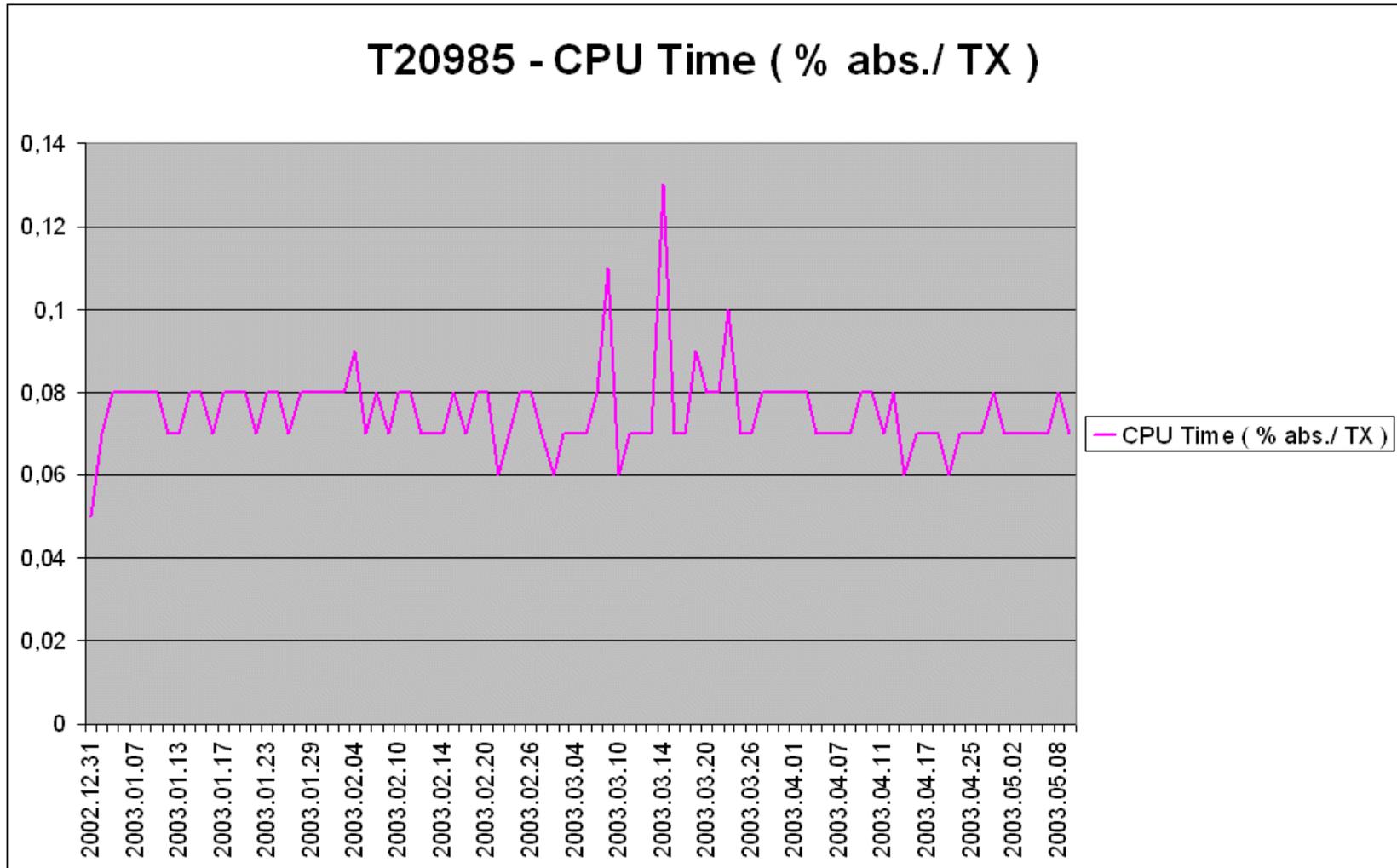
Beispiel 5 – kein Handlungsbedarf – 1



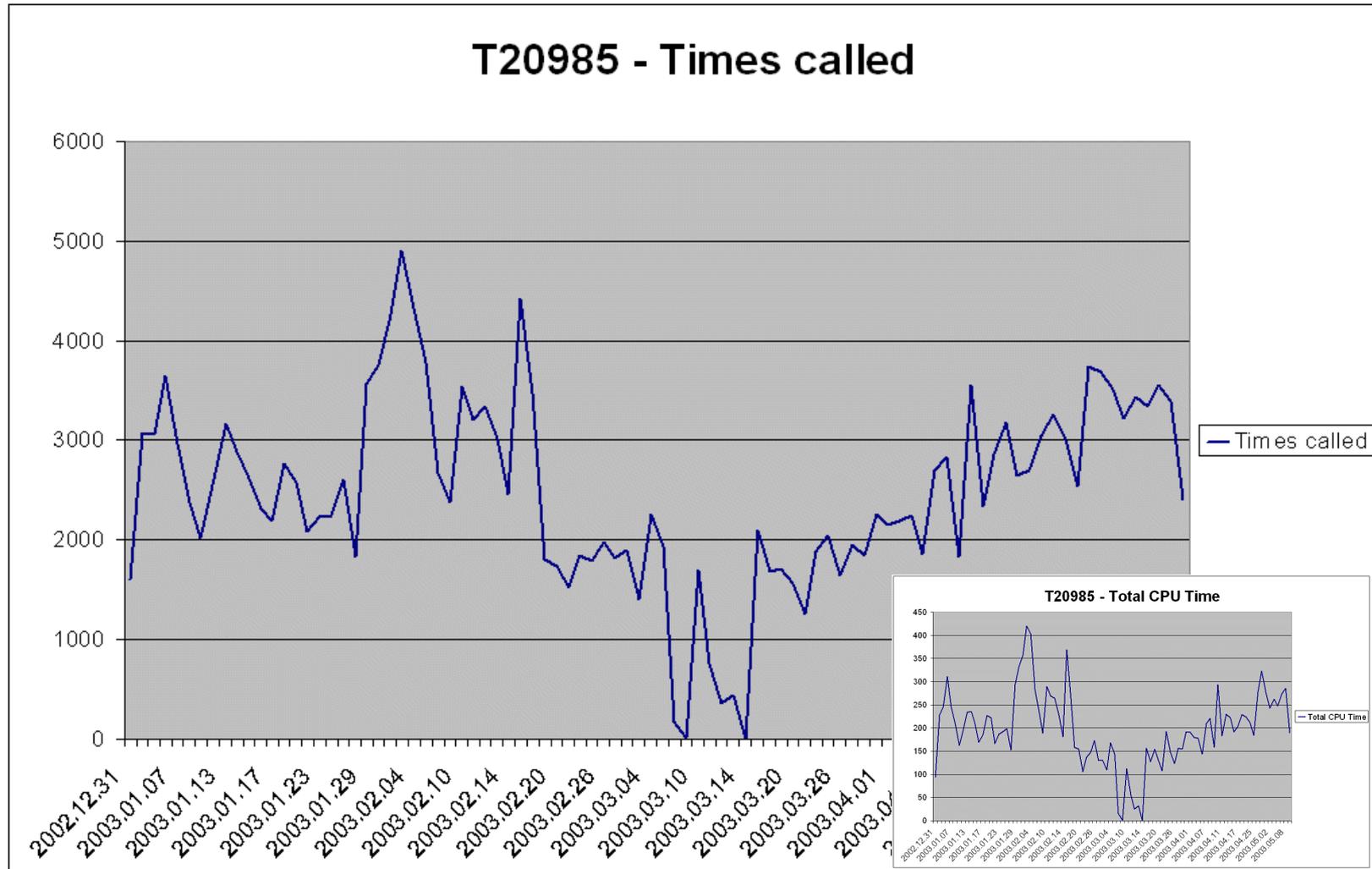
Beispiel 5 – kein Handlungsbedarf – 2



Beispiel 5 – kein Handlungsbedarf – 3



Beispiel 5 – kein Handlungsbedarf – 4



Optimierungen – Beispiele und Potential

Beispiele – mögliche Erfolge – 1

Datum	Programm / Job	CPU Ersparnis (hochgerechnet auf 1 Jahr)	Laufzeit Ersparnis (hochgerechnet auf 1 Jahr)	Art der Optimierung
19.12.2002	I9L55	1000 Stunden	(Online)	versch.
03.12.2002	GTE71	300 Stunden	(Online)	Index eingeführt
05.11.2002	GTF83 / TGTF83*	1000 Stunden	3000 Stunden	DB2-Zugriff umkodiert
05.11.2002	ZFU23 / *	350 Stunden ⁽⁴⁾	nicht berechnet	COBOL-Layer
05.11.2002	ZZH75 / TZZH75*	250 Stunden ⁽⁵⁾	14 Stunden	Index eingeführt
30.10.2002	DHI70/TDHI70*	270 Stunden	380 Stunden	Aufrufhäufigkeit ZFU23
23.10.2002	DW215/TDWDW215	250 Stunden	300 Stunden	SQL
30.10.2002	KK-Abschluss Detailinformation	nicht messbar ^{(2) (3)}	nicht messbar	versch.
12.09.2002	DHJ20/TDHIJ20*	170 Stunden	400 Stunden	fachliche Änderungen
10.09.2002	DHN91/TDHN91*	100 Stunden	100 Stunden	fachliche Änderungen
02.09.2002	DO692/PDODO692	700 Stunden	850 Stunden	Reorganisation DB2-Table
22.08.2002	I9S61 / IMS	240 Stunden	(Online)	Loop (Code).
15.08.2002	N1451 / TN1451*	270 Stunden	336 Stunden	Aufrufhäufigkeit N2U73
15.08.2002	ZZH75 / TZZH75*	800 Stunden ⁽⁵⁾	0 Stunden	Parallelisierung / Fachlichkeit
14.08.2002	DM238 / TDM238*	400 Stunden	400 Stunden	Einsatz internes Array
08.08.2002	U6W89 / TU6W89*	900 Stunden	900 Stunden	Index eingeführt
07.08.2002	PK110 / PPKPK110	300 Stunden ⁽²⁾	400 Stunden	Aufruf ZFU23
11.07.2002	GTF83 / TGTF83*	--- (3)	--- (3)	Runstats/Rebind
11.07.2002	KIA99 / IMS	55 Stunden	(Online)	Code-Optimierung
20.06.2002	HAA45 / THAHA23*	270 Stunden	450 Stunden	Index eingeführt
28.05.2002	DHV08 / TDHDHV08	90 Stunden	800 Stunden	SQL-Zugriff
29.04.2002	GTK00 / versch.	750 Stunden	1875 Stunden	SQL-Zugriff, RUNST. etc.
29.04.2002	ZFU88 / versch.	140 Stunden	200 Stunden	Felddefinitionen
23.04.2002	DHI25 / TDHDHI25	120 Stunden	200 Stunden	Initialize, DBRM etc.
27.03.2002	GTL01 / TGTL0151	420 Stunden	nicht berechnet	Inspect / Initialize
11.03.2002	HH720 / xHH72001	38 Stunden ⁽⁵⁾	30 Stunden	COBOL-Felder (Stufe 1)
05.03.2002	HAA24 / THAHA24W	400 Stunden	400 Stunden	SQL-Zugriff
17.02.2002	TD1D1B30	---	40 Stunden	BUFNO=16
05.02.2002	GT500 / MGT500%1	---	---	RUNSTATS, REBIND
05.02.2002	BF001 / MBFBF001	240 Stunden	310 Stunden	DB2-Optimierung
28.01.2002	N2735 / TN2735*	625 Stunden	667 Stunden	Aufrufhäufigkeit I9U73
25.01.2002	Posy / TPOPO001	---	30 Stunden	BUFNO=16
22.01.2002	IK019 / TIKIK019	40 Stunden	120 Stunden	Aufrufhäufigkeit I9U73
		3.317.559,12 €⁽¹⁾		

Beispiele – mögliche Erfolge – 2

Datum	Programm / Job	CPU Ersparnis (hochgerechnet auf 1 Jahr)	Laufzeit Ersparnis (hochgerechnet auf 1 Jahr)	Art der Optimierung
14.05.2003	ZFF01 / *	315 Stunden ⁽⁴⁾	Online	COBOL-Tuning / Pgmlogik
21.03.2003	I9S76 / *	290 Stunden ⁽⁴⁾	Online	DB2-Optimierung
05.03.2003	ZFU23 / *	400 Stunden ⁽³⁾	nicht berechnet	COBOL
17.02.2003	IK008 / TIKIK008	50 Stunden	60 Stunden	Sort, Programmlogik
11.02.2003	PAX05 / T33*, T08*	940 Stunden	Online	DB2-Optimierung
11.02.2003	I9L55 / T21574	1000 Stunden	Online	DB2-Optimierung
11.02.2003	N3A02 / TN2A02*	295 Stunden ⁽²⁾	1040 Stunden	Ausbau überflüssiger Code
11.02.2003	DDS09 / TDDDDS09	80 Stunden	400 Stunden	SQL optimiert
14.01.2003	DH6*	70 Stunden	100 Stunden	Aufrufhäufigkeit I9U73
		930.038,40 € ⁽¹⁾		



Potential allgemein - Compile Options (*)

• AWO NOAWO	0%	bis	10%	/ -
• NUMPROC(PFD NOPFD)	1%	bis	20%	/ 3%
• NOOPT OPT(STD)	1%	bis	12%	/ 3%
• OPT(STD FULL)	0%	bis	80%	/ 1%
• NOSSRANGE SSRANGE	1%	bis	27%	/ -
• TEST NOTEST	20%	bis	200%	/ -
• TRUNC(BIN STD) (**)	15%	bis	78%	/ 40%
• TRUNC(OPT STD)	6%	bis	65%	/ -

* IBM-Zahlen aus IBM Enterprise COBOL Version 3 Release 1 Performance Tuning, January 16, 2002 u.a.

** IBM will TRUNC(BIN) optimieren ab/seit COBOL for OS/390 & VM V2R2: bisher wenig überzeugend

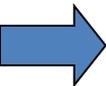


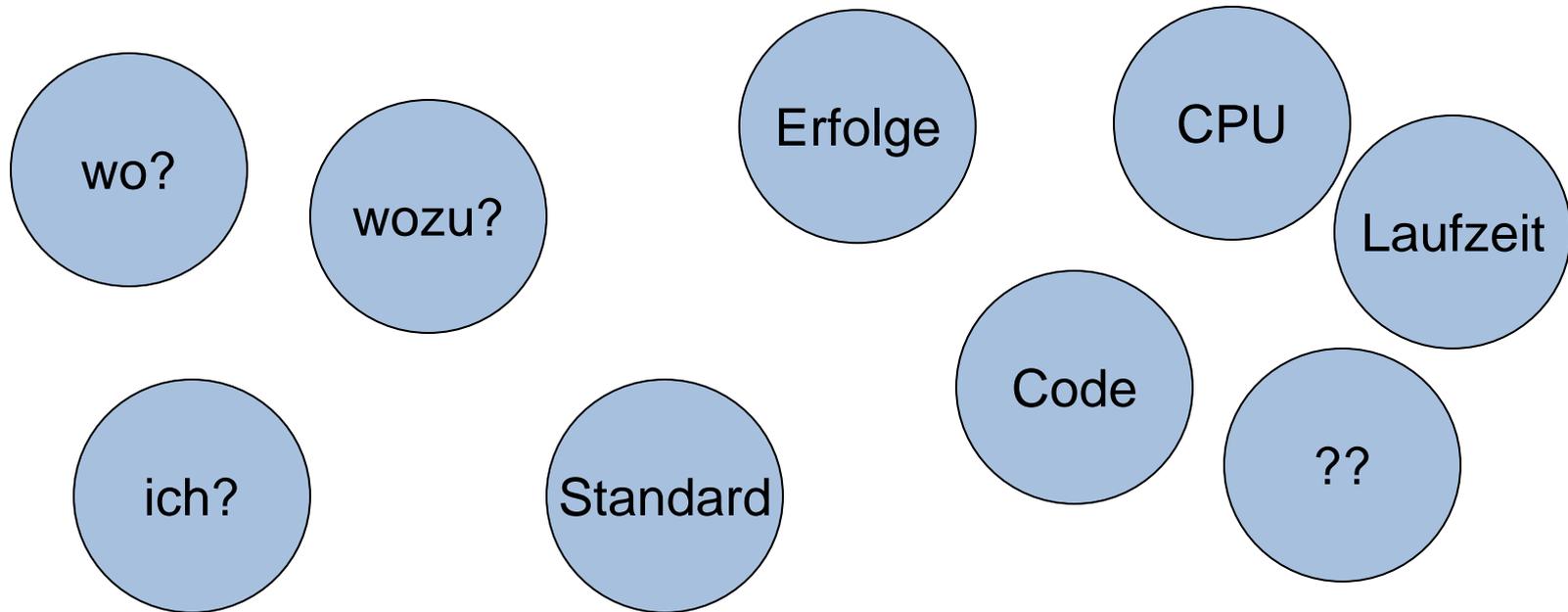
Basis bei Firma X

- Informationen zu DBRMs / Packages
- Top-100-Liste CICS (07:00 bis 17:00 Uhr)
- Performancebilder von Transaktionen

- laufende Beobachtung
- kein echtes Reporting über Erfolge



-
- Vorstellung und Einführung
 - Optimierungen – Beispiele und Potential
 -  • Richtlinien
 - Modellierung und DB2-Zugriffe
 - COBOL–Felder – COBOL-Befehle
 - Auswirkungen von Optionen – COBOL – LE
 - Informationen und Tools bei Firma X
 - Strobe – Handling und Interpretation
 - Diskussion - Austausch



Ideen

- Was bringen sie prinzipiell?
- Was bringen sie *mir*?
- Welche Schnittstellen gibt es?
- Chancen und Grenzen von Richtlinien
- Wo sind sie zu finden?

Vorteile – für mich

- eine Linie, an der ich mich (aus)richten kann
- Rahmen
- Orientierung
- Beispiele
- Tipps und Tricks
- lessons learned

Murphys Gesetze

1. Die Dinge sind komplexer als sie scheinen!
2. Die Dinge brauchen länger als erwartet!
3. Die Dinge kosten mehr als vorgesehen!
4. Wenn etwas schief gehen kann,
so geschieht es!

Anmerkung: Murphy war ein Optimist !

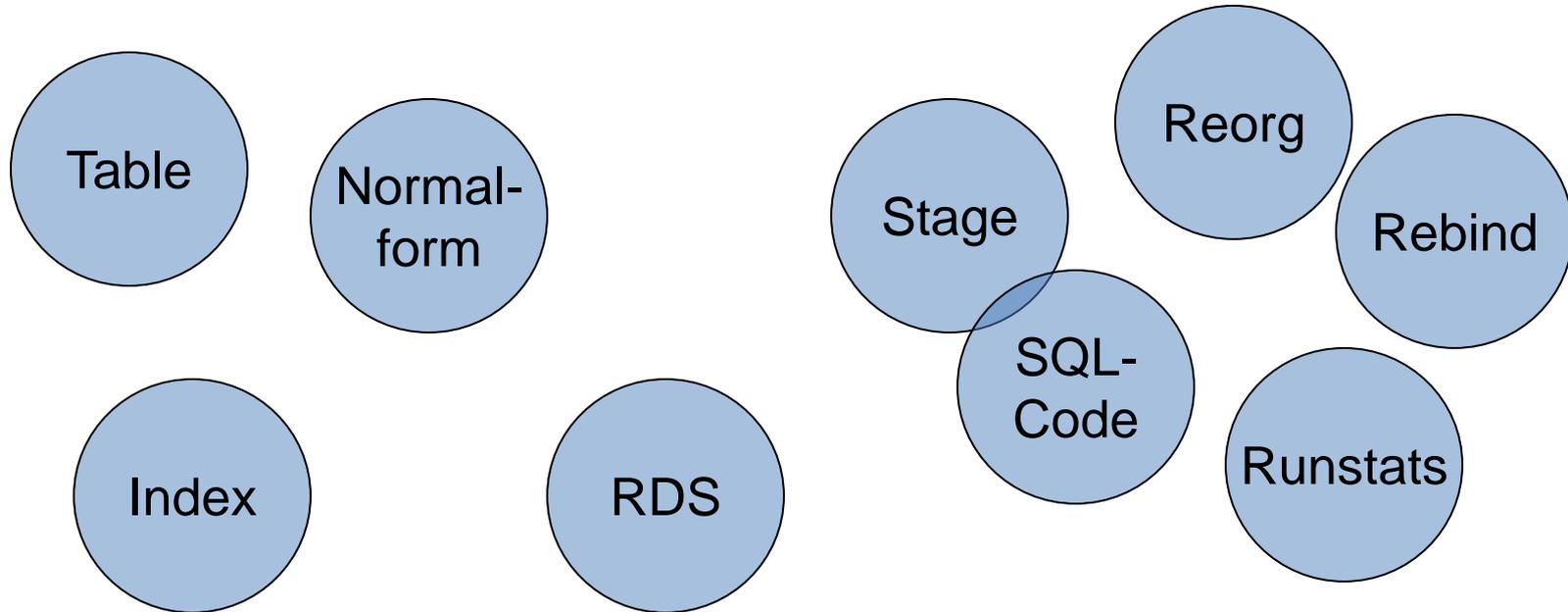
Ziele

- Performance Management für Anwendungen (PMA) ist ein wichtiger Prozess
- vorhandene Informationen zusammenfassen
- vorhandene Informationen zentral bereit stellen
- weitere Ideen
 - Tipps und Tricks
 - lessons learned
- siehe Wiki bei Firma X



-
- Vorstellung und Einführung
 - Optimierungen – Beispiele und Potential
 - Richtlinien
 - ➔ • Modellierung und DB2-Zugriffe
 - COBOL–Felder – COBOL-Befehle
 - Auswirkungen von Optionen – COBOL – LE
 - Informationen und Tools bei Firma X
 - Strobe – Handling und Interpretation
 - Diskussion - Austausch

Begriffe



Komplexität des DB2

- DB2 in sich sehr komplex
- mehrere verschiedene Buffer Pools
 - BP2-Pool für Daten
 - BP3-Pool für Indexes
- viele DB-Objekte wie
 - Tablespace, Table, View, Index ...
- Umgang damit – KISS ist (lebens)notwendig
- Zitat Einstein: “Alles sollte so einfach wie möglich sein, aber nicht noch einfacher.”

Ziel von DB2 und SQL

- kodieren des WAS nicht des WIE 😊
- Aber:
Modellierung⁽¹⁾, Wartung^(1,2,3) und Zugriff⁽²⁾ haben großen Einfluss auf das WIE. ☹️

⁽¹⁾ Datenmodell, Aufbau Tabellen, Aufbau Indizes

⁽²⁾ SQL

⁽³⁾ Änderung von Datenmengen, Art der Daten, Art der Abfragen ...

- Modellierung der Tabellen
- passende Nutzung der Runstats
- geeignete Nutzung von Reorgs
- angemessene Nutzung der Indexe
- richtiges Kodieren der SQLs



- Um einfache Relationen zu erhalten, wurde formalisierter Zerlegungsprozess für die Daten entwickelt.
- Es werden verschiedene Stufen für die Abhängigkeit der Daten untereinander definiert:
 - 1. Normalform
 - 2. Normalform
 - 3. Normalform

Normalisierung – 1. Normalform

- Eine Relation ist in der 1. NF, wenn alle Attribute direkt (funktional) vom Primärschlüssel abhängig sind.
oder:
- Jedes Attribut kann nur einen Wert annehmen. Wiederholgruppen sind nicht erlaubt.
- 1970, Codd
 - A relational R is in 1NF if and only if all underlying domains contain atomic values only.

Normalisierung – 2. Normalform

- Eine Relation in der 1. NF ist automatisch in der 2. NF, wenn der Primärschlüssel nicht aus mehreren Attributen zusammen gesetzt ist.
oder:
- Bei zusammen gesetzten Primärschlüsseln muss jedes Attribut vom gesamten Primärschlüssel direkt abhängig sein.
- 1971, Codd
 - A relational R is in 2NF if it is in 1NF and every non-key attribute is fully dependant on the primary key. (Any relation in 1NF and not in 2NF must have a composite key.)

Normalisierung – 3. Normalform

- Die 3. NF ist erfüllt, wenn die 2. NF erfüllt ist und alle Attribute, die nicht zum Primärschlüssel gehören, voneinander unabhängig sind.
- 1971, Codd
 - A relational R is in 3NF if it is in 2NF and every non-key attribute is non transitively dependant on the primary key.



Normalisierung – 4. Normalform

- Die 4. NF ist erfüllt, wenn die 3. NF erfüllt ist und keine paarweisen, mehrwertigen Abhängigkeiten zwischen Attributen bestehen.
- 1977, Fagin
 - A normalized relational R is in 4NF if and only if whenever there exists a multivalued dependency in R, say of attribute B on attribute A, all attributes of R are also functionally dependant on A.

Normalisierung – 5. Normalform

- Die 5. NF ist erfüllt, wenn sie notwendig ist, Daten der 4.NF ohne Informationsverlust über einen Join zusammen zu führen.
- 1979, Fagin
 - A relational R is in 5NF if and only if every join dependency in R is a consequence of keys of R.



Normalisierung – Fragen

- Ist das denn noch normal?
- Das kann doch keiner mehr verstehen!
- Ist der ganze Quatsch denn notwendig?

- Normalisierungsprozess
 - ist aufwändig
 - liefert die Basis für stabile Datenstrukturen
 - Daten in 1. NF sind nicht sinnvoll verwaltbar
 - Daten in 2. NF sind schwierig verwaltbar
 - (mindestens) bis 3. NF durchführen
 - 5. NF „garantiert“ stabile Ergebnisse zur Laufzeit
- Denormalisierung für Physik immer möglich!

Normalisierung – wie wäre es mit ...

- ...
every entity depends

on the key,

the whole key,

and nothing but the key

Formulierung: Danke an Gerhard Heiß.



Normalisierung – Beispiel 1

- Umsatz pro Produkt und Monat

Produkt	Jahr	Jan	Feb	Mar	...
P1	2007	10,7	11,3	9,5	
P2	2007	6,8	4,3	5,5	

- Auswertung für 1. Quartal:
SELECT PRODUKT, JAN+FEB+MAR
WHERE JAHR= :HV1

1. Normalform
und nicht
2. Normalform

- Auswertung für 1. Halbjahr:
SELECT PRODUKT, JAN+FEB+MAR+APR+MAI+JUN
WHERE JAHR= :HV1

Normalisierung – Beispiel 2

- Umsatz pro Produkt und Monat

Produkt	von_dat	bis_dat	Umsatz
P1	01.01.2007	31.01.2007	10,7
P1	01.02.2007	28.02.2007	11,3
P1	01.03.2007	31.03.2007	9,5
P2	01.01.2007	31.01.2007	6,8
P2	01.02.2007	28.02.2007	4,3
P2	01.03.2007	31.03.2007	5,5

- Auswertung für alles Mögliche:

```
SELECT PRODUKT, SUM(UMSATZ)
WHERE VON_DAT >=:HV2
AND    BIS_DAT <=:HV3
GROUP BY PRODUKT
```



Denormalisierung

- ... ist erlaubt aus Gründen der Performance und der Flexibilität
- Beispiel Partner-Modell
 - Ein Partner kann mehrere Anschriften haben.
 - 1:n-Relation partner ->> adresse
 - in Praxis fast immer: 1:1-Relation
 - “Hauptadresse” wird in Table partner aufgenommen mit Hinweis auf zusätzliche Adresse
 - Einsparung: 800€/Tag nur in CICS

Separierung

- Trennung von häufig benutzten Daten von wenig benutzten Daten innerhalb einer Table
- wichtig bei großen Tabellen
- Ergebnis:
 - Ausfallsicherheit erhöht
 - regelmäßige Reorgs möglich
 - Recovery deutlich schneller
- Beispiel EDM
 - Q98T27H: 803 Mio. / Q98T270: 212.000

Partitionierung

- Gründe für physische Partitionierung
 - Tablespace mit 64GB-Grenze (früher 4GB)
 - Parallelisierung von Prozessen
- Beispiel KFZ (Firma X):
 - Folgeinkasso in 5 parallelen Jobs
 - ohne Parallelisierung nicht durchführbar
 - Bildung von Nummernkreisen
- Beispiel Kontokorrent (Bank):
 - tägliche Verarbeitung 48-fach parallel



Runstats

- Statistik zu einer Tabelle
- Beispiel:
- Anzahl der Zeilen
 - letzter Runstats
 - Anzahl pages
 - Anzahl indexpages
 - etc.
 - also alles, was ein Optimizer für seinen Zugriff braucht.

Runstats – Aktualität

- Es wird zum Zeitpunkt des Bind auf die Runstat-Informationen zugegriffen und dabei der Zugriffspfad festgelegt!
 - Achtung: statischer vs. dynamischer SQL
- Folgerung:
 - regelmäßig Runstats (mit Rebind?) durchführen
- Tipp:
 - Es gibt Tools, die die Runstats-Informationen interpretieren können bzgl. der Inhalte.

Reorganisation einer Tabelle

- Reorg heißt u.a.
 - Neuaufbau der Tabelle
 - Neuaufbau des Index (Clustering)
- Ziel (denke an VSAM ;-)
 - leere Bereiche füllen
 - Überlaufbereiche neu anlegen
 - etc.
- Folgerung:
regelmäßig Reorg durchführen
... spätestens wenn Clusterratio <95%



Indexdesign

- Zugriff muss durch Index unterstützt werden
 - Ausnahme: Minitabellen
- Ergebnis:
 - Tablespacescan wird vermieden
 - Non-matching Indexscan wird vermieden
 - oft werden interne Sorts nicht mehr benötigt
 - ascending / descending – ab V8 automatisch
- wichtigsten Index clustern
 - also nicht immer den primary index!

Sortierung der Tabelle

- Ist das wirklich wichtig?
- Beispiel:
 - Briefträger ist ein INSERT-Operator
 - Straße ist die Tabelle
 - Briefkästen sind die Pages der Tabelle, in die eingefügt werden soll
 - Sortierung nach Name ... ☹
 - Sortierung nach Straße und Hausnummer ... ☺

allgemein

- Es lohnt sich, von Zeit zu Zeit einen Blick auf die Anwendung und die zugehörigen Tabellen zu werfen.
- Frage: Passt das Design der Tabelle zur Implementierung der Anwendung?
- Frage: Hat sich das Verhalten (Zugriffsarten) der Anwendung gegenüber “damals” verändert?
- Prinzip: schaue nach PK, dann auf Indexe, die Predicates unterstützen (nicht zu viele)



DB2-Internas in aller Kürze

- Es gibt 5 Ebenen für die Zugriffe:
 - Stage 1
 - Data Manager mit einfachen Predicates
 - Indexmanager mit “matching index scan”
 - Stage 2
 - Daten laufen über RDS (Relational Data System)
 - Stage 3
 - virtuelle Predicates / set current timestamp
 - Stage 4
 - Alles andere, das bisher nicht abgedeckt ist
 - wie substr, timestamp auf Tabelle



- Index sorgt für Eindeutigkeit.
- Index sorgt für Geschwindigkeit.
- Indexaufbau
 - 1. Primary Key (wenn möglich *keine* UID!!!)
 - 2. weitere Keys nur für GeschwindigkeitWie greife ich auf Tabelle zu?
 - where-clauses passend zum Index
- Clustering-Index
 - Reihenfolge wie im Tablespace
(Denke an Briefträger! Reihenfolge nach Namen oder Hausnummer)

- Matchcols möglichst hoch
 - Beispiel Telefonbuch
 - Nachname bekannt: Matchcol=1
 - Vorname zusätzlich bekannt: Matchcol=2
 - Straße zusätzlich bekannt: Matchcol=3
 - Hausnummer zusätzlich bekannt: Matchcol=4
- Matchcol=0 (non-matching indexscan) so schlecht wie Tablespacescan
 - Beispiel: Telefonbuch nach Straßennamen sortiert

- Tablespace scan ist okay wenn
 - Batchverarbeitung (fast) alles lesen muss
 - kleine Tabellen (z.B. wenige 100 Rows)
- Sort möglichst vermeiden – ORDER BY genau dann, wenn durch Index unterstützt
 - denn: open cursor muss bei order by ohne Index-Unterstützung erst die gesamte Ergebnismenge lesen!
- Split von Index-Pages problematisch; dann Freespace erhöhen (lassen)

Analyse 1

Entscheidung 1

Analyse und Entscheidung 2

Analyse 3

Strobe-Report 4

Analyse und Umsetzung 4

Ergebnis



- Eine Bemerkung vorab:

Es gibt unterschiedliche Top-Ten-Listen für das Kodieren von SQLs; daher kann es je nach Autor leicht unterschiedliche Sichtweisen geben. Aus diesem Grund sind die nachfolgenden 10 Gebote als *eine* von verschiedenen Sichtweisen zu sehen

10 Gebote für das Schreiben eines SQL – 1

1. SELECT nur die benötigten Felder (Columns)
 - SELECT * ist „verboten“.
2. SELECT nur die benötigten Zeilen (Rows)
 - Nicht das Programm auswählen lassen.
3. SELECT nur mit „unbekannten“ Werten
 - SELECT FIRMA FROM ... (FIRMA ist immer = 1) ist eine „unsinnige“ Abfrage

10 Gebote für das Schreiben eines SQL – 2

- 4. Versuche, Predicates auf Stage-1 zu bringen
 - WHERE COL BETWEEN :x1 AND :x2 ist Stage-2
WHERE COL >= :x1 AND COL <= :x2 ist Stage-1

 - Achtung: Das gilt genau dann, wenn *kein* Index benutzt werden kann; wenn COL im Index enthalten ist, dann besser mit BETWEEN arbeiten!

 - COL NOT IN (:w1, :w2, :w3) ist Stage-2
COL IN (:a1, :a2, :a3) ist Stage-1

5. WHERE clause mit AND oder OR

a. AND: Kodiere where-clause so, dass die größte Einschränkung am Anfang steht.

- WHERE X1 = „weiblich“ AND x2 = „Physiker“
besser:
- WHERE X2 = „ Physiker“ AND X1 = „weiblich“

b. OR: Kodiere where-clause so, dass die größte Menge am Anfang steht.

- WHERE X2 = „ Physiker“ OR X1 = „weiblich“
besser:
- WHERE X1 = „weiblich“ OR x2 = „Physiker“

6. Filtern von Daten *vor* einem Join nicht während eines Join.
7. Versuche statt einer Arithmetik innerhalb einer where-clause feste Werte zu verwenden. Wenn nicht vermeidbar ...
 - ☺ WHERE SALARY > 50000/(1 + :hv1)
 - ☹ WHERE SALARY + (:hv1 * SALARY) > 50000
8. Vermeide sortieren von Daten
 - ORDER BY und GROUP BY möglichst nur auf dem Clustering Index

10 Gebote für das Schreiben eines SQL – 5

9. Wenn 1 Zeile erwartet wird, nutze einen einfachen SELECT statt einer Cursor-Verarbeitung.
 - FETCH FIRST ROW ONLY auch bei SELECT!!!
10. Ändere nur die veränderten Rows.

11. Vermeide arithmetische Ausdrücke.
12. Nutze `NOT EXISTS (SELECT ...)`
statt `NOT IN (SELECT ...)`.
13. `>=` ist indexable, `>` ist nicht indexable
14. Nutze aktuellen Runstats.
15. Nutze multi-row-fetch
etc.

Übrigens: Die SQL-Reference von IBM ist mehr
als 20 MB groß! ☹ ☹ ☹



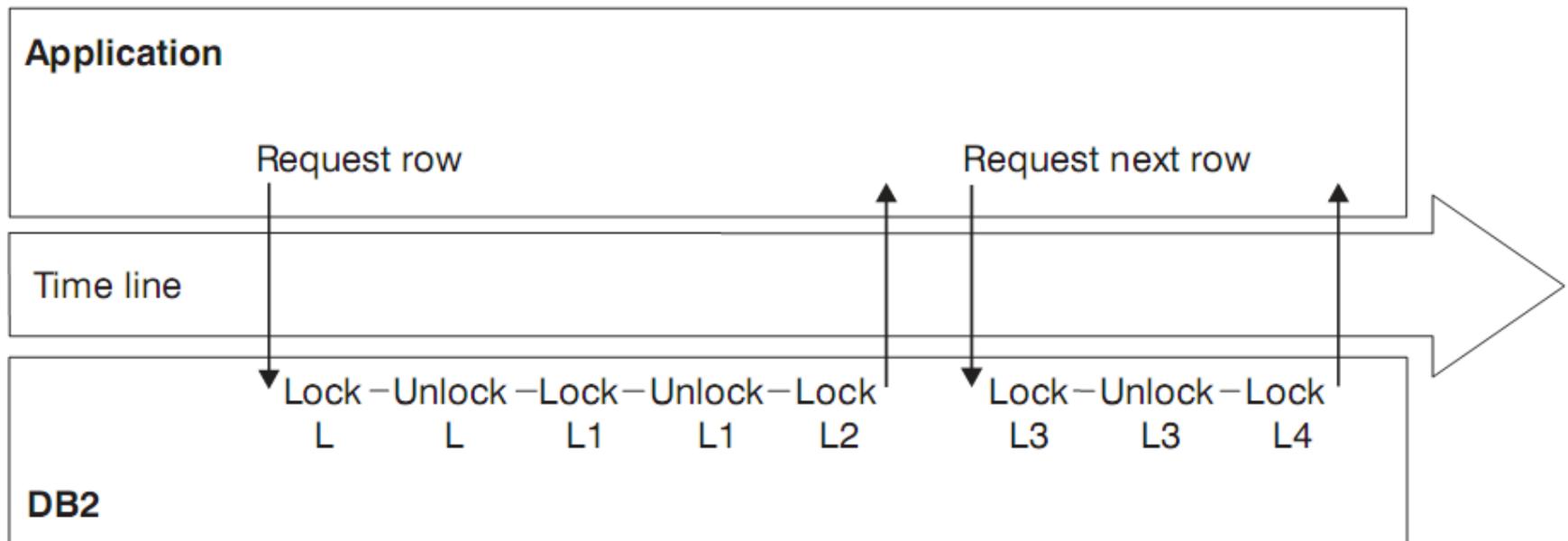
Isolation Level – RR

- RR – Repeatable Read
 - mehrfaches Lesen von Rows oder Pages
 - Jede benutzte Page wird gelockt, selbst wenn sie *nicht* den Predicates genügt.
 - *kein* paralleler Update erlaubt

- RS – Read Stability
 - mehrfaches Lesen von Rows oder Pages
 - Jede benutzte Page wird gelockt, selbst wenn sie *nicht* den Predicates genügt.
 - *paralleler Update teilweise erlaubt
 - Gelockt werden Rows bzw. Pages, die Stage 1 und Stage 2 erfüllen (und keine anderen).

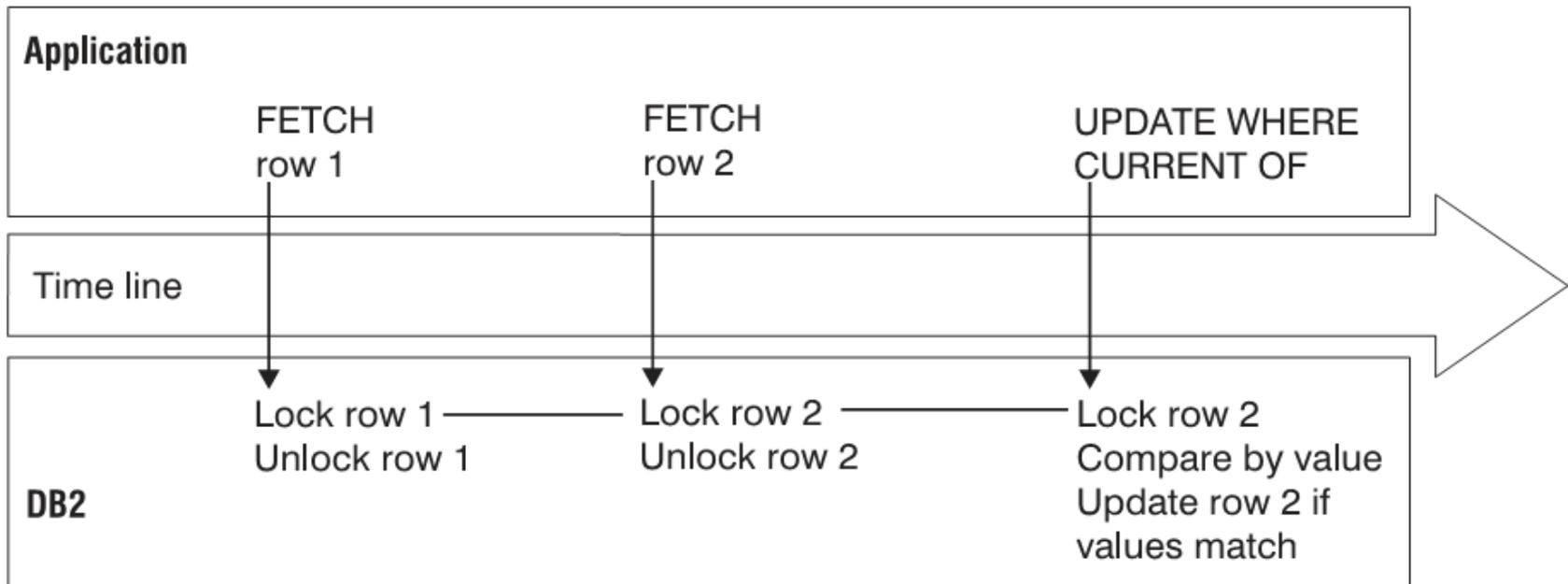
Isolation Level – RS – 2

- RS – Read Stability – Beispiel
 - L2 und L4 erfüllen die Predicates



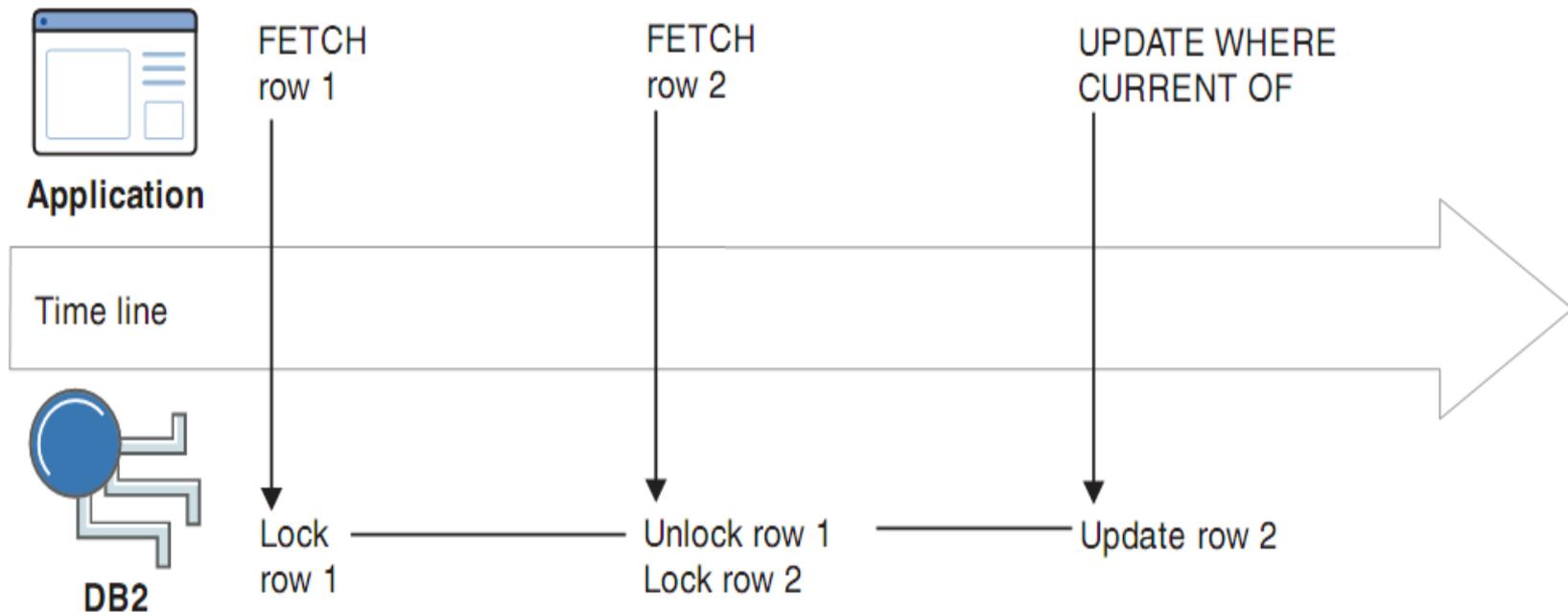
Isolation Level – CS – 1

- CS – Cursor Stability – höchste Datenintegrität mit „optimistic currency control“



Isolation Level – CS – 2

- CS – Cursor Stability – höchste Datenintegrität ohne „optimistic currency control“ bei „dynamic scrollable cursors“



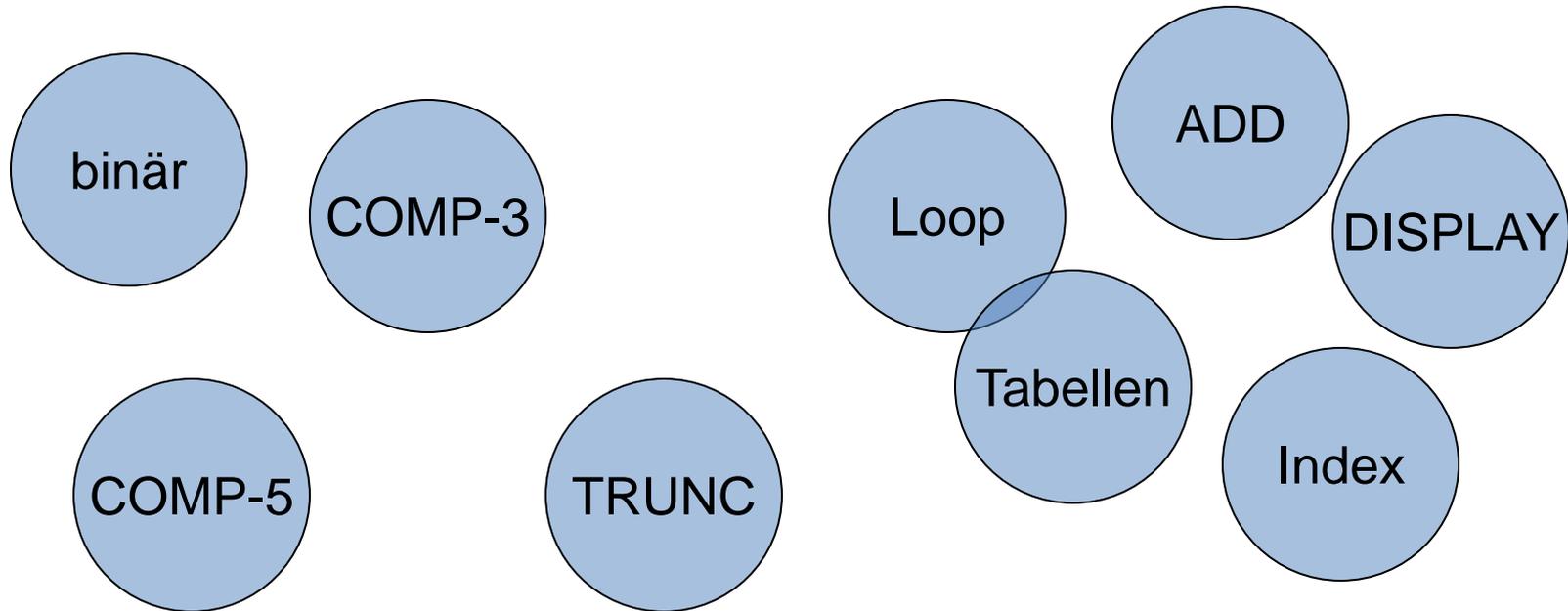
Isolation Level – UR

- UR – Uncommitted Read
 - auch „dirty read“ genannt
 - geht nicht bei
DELETE, UPDATE, INSERT, MERGE
CURSOR ... FOR UPDATE
 - Sollte immer als Möglichkeit in Betracht gezogen werden. Denn: Kann es denn wirklich sein, dass parallel, also genau zur gleichen Zeit, exakt an diesem einen Objekt etwas getan wird?



-
- Vorstellung und Einführung
 - Optimierungen – Beispiele und Potential
 - Richtlinien
 - Modellierung und DB2-Zugriffe
 - ➔ • COBOL–Felder – COBOL-Befehle
 - Auswirkungen von Optionen – COBOL – LE
 - Informationen und Tools bei Firma X
 - Strobe – Handling und Interpretation
 - Diskussion - Austausch

Begriffe



- **Binärfelder – BINARY**

- Halbwort S9(04) oder Vollwort S9(08) mit Vorzeichen
- Compile Option TRUNC beachten (später detailliert)
- Doppelwort (z.Z.) sehr inperformant
- bei intensiver Nutzung: SYNC benutzen

DB2: INTEGER / SMALLINT
CICS: EIBCALEN

- **gepackte Felder – PACKED-DECIMAL**

- auf Bytegrenzen achten (S9(n) mit n ungerade ≤ 15)

- **“normale Felder” – USAGE DISPLAY**

- nicht für Rechenoperationen verwenden
- auch hier: ungerade Anzahl Digits ist schneller
- Anzahl Digits ≤ 15 wählen

- **COBOL-Option ARITH(EXTEND) – bis 31 Ziffern**

Felddefinitionen – 2

- Loop-Verarbeitung (ohne Tabellen)
 - COMP-3: bis zu 280% langsamer als binär (*)
 - DISPLAY: bis zu 575% langsamer als binär (*)
 - wenn oft benutzt: besser ADD 1 TO ... statt varying
- ADD / SUBTRACT mit numerischen Feldern
 - es gibt je nach Einstellung TRUNC und Länge der Felder verschieden performantes Verhalten
 - meist Operationen mit binären Felder am schnellsten
 - je nach Anzahl Digits aber display-Felder schneller

(*) Quelle: IBM

- Tabellen
 - nur mit Indizes (INDEXED BY)
 - Ausnahme Binärfelder (mit TRUNC(OPT/STD))
 - S9(08) COMP 30% langsamer
 - niemals andere numerische Felder benutzen
 - denn COMP-3: 300% langsamer
 - DISPLAY: 450% langsamer
 - möglichst 1-dimensional
 - ODO möglichst nicht nutzen (ca. 140% langsamer)
 - wenn ODO notwendig: ODO-Feld muss binär sein
 - mehr-dim im Loop: ganz rechts schnellster Subscript

Felddefinitionen – 3 – Index – Beispiel 1

- 01 TAB OCCURS 20
INDEXED BY IND PIC X(88).
- Anzeige in DUMP: B0
- B0 = 176 (dezimal)
- Berechnung:
 $(176 / 88) + 1 = 3$
Der Index hat also den Wert 3!

Felddefinitionen – 3 – Index – Beispiel 2

- 01 TABX OCCURS 20
INDEXED BY INDX PIC X(27).
- Anzeige in DUMP: 6C
- 6C = 108 (dezimal)
- Berechnung:
 $(108 / 27) + 1 = 5$
Der Index hat also den Wert 5!

- INITIALIZE

- jedes einzelne Feld wird auf Anfangswert gesetzt
- jedes einzelne Feld wird auf Anfangswert gesetzt
- innerhalb Schleifen möglichst unterlassen
- Hilfsfelder nutzen
- jedes schwierige Beispiel muss separat beurteilt werden, daher kein “Kochrezept” möglich

außer FILLER

- STRING/UNSTRING/INSPECT/SEARCH

- zieht hohen CPU-Verbrauch nach sich
- ab V4R1 wird es schneller – NO ☹ ☹ ☹ noch nicht

- **PERFORM VARYING**
 - Schleifenzähler binär definieren / gepackt
 - Begrenzer binär definieren / gepackt
 - bei Tabellenverarbeitung nur mit INDEX arbeiten
 - jederzeit auf Formatgleichheit achten
 - wenn oft benutzt:
besser ADD 1 TO ... statt varying
- **EVALUATE**
 - (leider wieder) häufigsten Fall zu Beginn codieren
- **Stufe 88**
 - sehr schnelle Verarbeitung

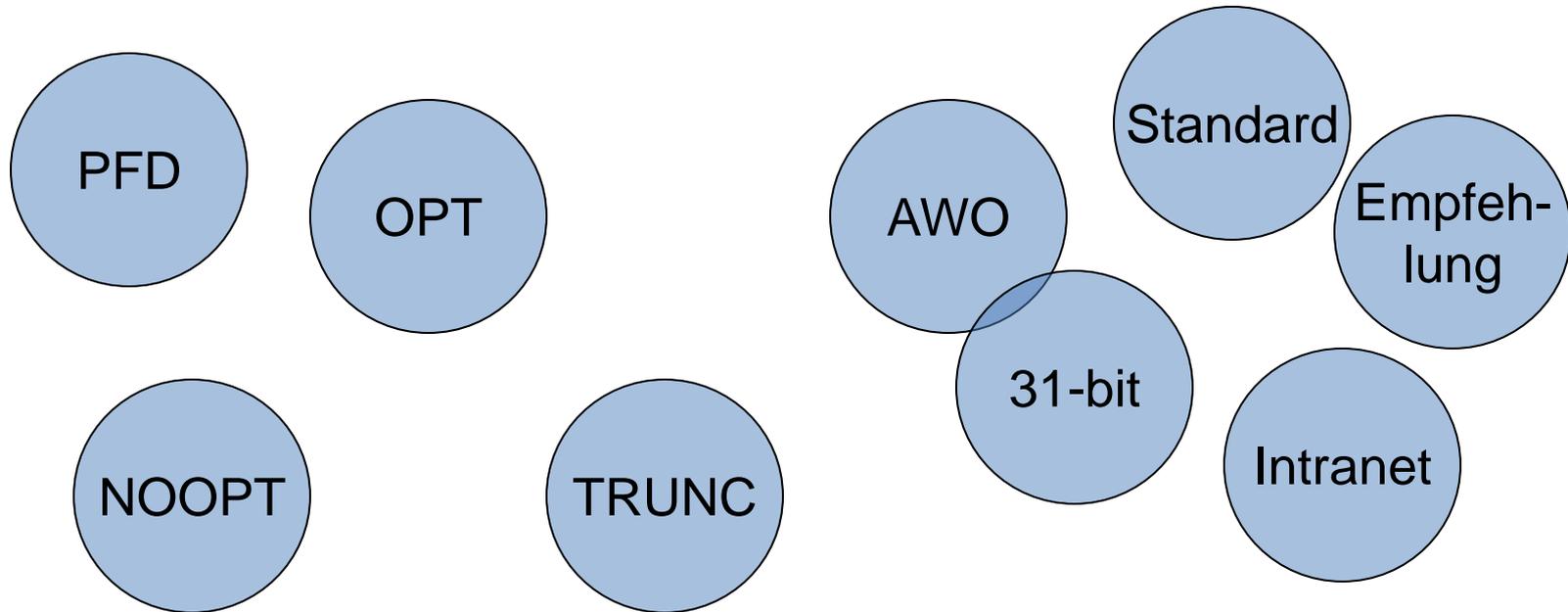
- Rechenoperationen
 - beteiligte Felder mit gleichen Längen
 - beteiligte Felder mit gleichem Format
- Vergleichsoperationen
 - beteiligte Felder mit gleichen Längen
 - beteiligte Felder mit gleichem Format
- Substr-Move
 - besser: `MOVE FELD-A(2:5) TO FELD-B (-> MVC)`
 - nicht: `MOVE FELD-A(2:N) TO FELD-B (-> MVCL)`
 - erste Zahl darf Variable sein

- MOVE
 - MVC ist schnell / MVCL ist langsam
 - MVC kann nur bis 256 Byte übertragen
 - bis Länge 768 werden MVCs generiert (aber nicht bei move spaces to ziefeld!!)
 - MVC / MVCL abhängig von **Ziefeld**



-
- Vorstellung und Einführung
 - Optimierungen – Beispiele und Potential
 - Richtlinien
 - Modellierung und DB2-Zugriffe
 - COBOL–Felder – COBOL-Befehle
 - ➔ • Auswirkungen von Optionen – COBOL – LE
 - Informationen und Tools bei Firma X
 - Strobe – Handling und Interpretation
 - Diskussion - Austausch

Begriffe



- Vorteile:

Notation: Standard *Empfehlung*

- Unnötige interne Programmverzweigungen werden eliminiert
- Out-of-Line PERFORM Statements werden, wenn möglich In-Line dargestellt. Die Verzweigung wird eingespart.
- Nicht erreichbarer Programmcode wird eliminiert und damit die Größe des Lademoduls reduziert.
- Optimierte Subscript Verarbeitung
- Redundante Rechenoperationen werden eliminiert.
- Rechenoperationen für Konstanten werden eliminiert.
- Einzelne, fortlaufende MOVE Statements werden teilweise als single MOVE aggregiert

Anmerkung: Gleichzeitig muss die Option LIST gesetzt werden. Diese wird benötigt, damit der Abend-Aid Postprozessor in Verbindung mit OPTIMIZE ohne Fehler durchläuft. Ohne LIST kann Abend-Aid bei einem Abbruch zwar die Offset-Adresse ermitteln, nicht aber das zugehörige COBOL-Statement

Notation: Standard *Empfehlung*

- Nachteile:
 - Einzelne, fortlaufende MOVE Statements werden teilweise als single MOVE aggregiert. Dazu mögliche fachliche Auswirkungen berücksichtigen.
 - COMPILE-Zeit länger
 - DEBUGGING evtl. erschwert
- Beispiel -> [LINK](#)
 - d.h.: numerische Felder werden evtl. als CHAR übertragen!
- Beispiel -> [LINK\(Pgm\)](#) / [Link\(Dump\)](#)
 - d.h.: Eyecatcher sind weg
- Konsequenzen beachten bei Fehlersuche

Notation: Standard *Empfehlung*

- Auswirkungen:

- Der Parameter bezieht sich auf geblockte sequentielle Dateien mit variabler Satzlänge, die im Output Modus verarbeitet werden.
- COBOL prüft bei AWO, ob der zu schreibende Satz noch in den zur Verfügung gestellten Buffer passt. Wenn dies der Fall ist, wird der Buffer noch nicht weggeschrieben.
- Bei NOAWO (Compilerdefault) geschieht diese Prüfung nicht sondern der Buffer wird weggeschrieben, wenn der längste, im Programm definierte Satz nicht mehr in den Buffer paßt.
- Mit AWO kann hier CPU und Laufzeit eingespart werden. Abhängig von den Satzdefinitionen können die Einsparungen über 50% erreichen.

bei Firma X wohl nicht relevant

FASTSRT | *NOFASTSRT*

Notation: Standard *Empfehlung*

- Auswirkungen:
 - Das I/O-Handling für internen Sort wird von DFSORT (o.V.) gemacht.
- Vorteil:
 - Die Option eliminiert den Overhead, der nach jedem Record zu COBOL zurückkehrt.
- Nachteil:
 - keine Mischung von “PROCEDURE” und “USING” möglich.
- persönliche Meinung:
 - keinen internen Sort verwenden

bei Firma X wohl nicht relevant

TRUNC(OPT) | TRUNC(BIN) | TRUNC(STD)

- Auswirkungen:

Notation: Standard *Empfehlung*

TRUNC ist bei allen Rechen- und Vergleichsoperationen mit binär definierten Feldern aktiv. Die empfohlene Einstellung ist der Compilerdefault. Für Änderungen von binären Feldern wird bei TRUNC(BIN) zusätzlicher Code zum Überprüfen generiert.

Der maximale Wertebereich von Binärfeldern (COMP) ist bei TRUNC(STD) durch die Anzahl der definierten Digits vorgegeben. Prüfungen, ob Überläufe stattfinden, werden nicht durchgeführt, was den CPU-Overhead reduziert.

- TRUNC(BIN) und TRUNC(OPT) sind IBM-Erweiterungen des Compilers.
- TRUNC(STD) hat einen minimalen Performanceverlust gegenüber TRUNC(OPT).

- **Nutze COMP-5 statt TRUNC(BIN) !!!**

DB2: INTEGER / SMALLINT
CICS: EIBCALEN

Felddefinitionen – Comparing Data Types – 1

- DISPLAY compared to packed decimal (COMP-3)
 - using 1 to 6 digits: DISPLAY is 100% slower than packed decimal
 - using 7 to 16 digits: DISPLAY is 40% to 70% slower than packed decimal
 - using 17 to 18 digits: DISPLAY is 150% to 200% slower than packed decimal
- DISPLAY compared to binary (COMP or COMP-4) with TRUNC(STD)
 - using 1 to 8 digits: DISPLAY is 150% slower than binary
 - using 9 digits: DISPLAY is 125% slower than binary
 - using 10 to 16 digits: DISPLAY is 20% faster than binary
 - using 17 digits: DISPLAY is 8% slower than binary
 - using 18 digits: DISPLAY is 25% faster than binary
- DISPLAY compared to binary (COMP or COMP-4) with TRUNC(OPT)
 - using 1 to 8 digits: DISPLAY is 350% slower than binary
 - using 9 digits: DISPLAY is 225% slower than binary
 - using 10 to 16 digits: DISPLAY is 380% slower than binary
 - using 17 digits: DISPLAY is 580% slower than binary
 - using 18 digits: DISPLAY is 35% faster than binary
- DISPLAY compared to binary (COMP or COMP-4) with TRUNC(BIN) or COMP-5
 - using 1 to 4 digits: DISPLAY is 400% to 440% slower than binary
 - using 5 to 9 digits: DISPLAY is 240% to 280% slower than binary
 - using 10 to 18 digits: DISPLAY is 70% to 80% faster than binary

Felddefinitionen – Comparing Data Types – 2

- Packed decimal (COMP-3) compared to binary (COMP or COMP-4) with TRUNC(STD)
 - using 1 to 9 digits: packed decimal is 30% to 60% slower than binary
 - using 10 to 17 digits: packed decimal is 55% to 65% faster than binary
 - using 18 digits: packed decimal is 74% faster than binary
- Packed decimal (COMP-3) compared to binary (COMP or COMP-4) with TRUNC(OPT)
 - using 1 to 8 digits: packed decimal is 160% to 200% slower than binary
 - using 9 digits: packed decimal is 60% slower than binary
 - using 10 to 17 digits: packed decimal is 150% to 180% slower than binary
 - using 18 digits: packed decimal is 74% faster than binary
- Packed decimal (COMP-3) compared to binary (COMP or COMP-4) with TRUNC(BIN) or COMP-5
 - using 1 to 8 digits: packed decimal is 130% to 200% slower than binary
 - using 9 digits: packed decimal is 85% slower than binary
 - using 10 to 18 digits: packed decimal is 88% faster than binary
- Quelle: Share-Tagung 2002, Tom Ross, IBM, Santa Teresa

NUMPROC(PFD) | *NUMPROC(NOPFD)*

Notation: Standard *Empfehlung*

- Auswirkungen:
 - NUMPROC(NOPFD) führt implizit Vorzeichenprüfungen für packed decimal und usage display Felder durch. Bei Einsatz von NUMPROC(PFD), geht der Compiler davon aus, dass die numerischen Felder das richtige Vorzeichen haben. Prüfungen, die das Vorzeichen verifizieren, finden nicht statt.
 - Rechen- und Vergleichsoperationen benötigen weniger CPU während der Ausführung. **PFD = preferred sign**
- möglicher Nachteil:
 - bei unsicheren Datenquellen könnten erst später zur Laufzeit Fehler auftreten.

Notation: Standard *Empfehlung*

- Auswirkungen:
 - Programm kann 24- oder 31-bit-Adressen benutzen
- Vorteil:
 - 2GB vs. 16MB
- Nachteil:
 - keiner bekannt

DATA(31) (mit RENT) | DATA(24)

Notation: Standard *Empfehlung*

- Auswirkungen:
 - Die QSAM-Buffer und die Working Storage werden above-the-line angelegt.
 - Das Programm wird bei RENT in die LPA/ELPA geladen.
- Vorteil:
 - schnellere I/O-Behandlung; bessere Speicherausnutzung
- Nachteil:
 - bei RENT wird zum Programmmanfang minimal mehr Code generiert, der RENT prüft.

RMODE(AUTO) | RMODE(24)

Notation: Standard *Empfehlung*

- Auswirkungen:
 - Programm wird dort hin geladen, wo Platz ist.
- Vorteil:
 - Das System sucht optimalen Platz für das Programm.
- Nachteil:
 - keiner bekannt

- Linkoption: RMODE(24|ANY)

31-bit-adressing

- Der Weg in Richtung 64-bit-Adressierung muss u.a. wegen der wachsenden Datenmengen konsequent verfolgt werden!
- LE-Option ALL31(ON) spart laut IBM ca. 3% der gesamten CPU-Last .
Diese Option kann nicht gesetzt werden, so lange noch Anwendungsprogramme below-the-line laufen müssen.



COBOL Compile Options

NUMPROC(NOPFD) und NOOPT

1PP 5655-G53 IBM Enterprise COBOL for z/OS 3.4.1
0Invocation parameters:

Date 01/17

0PROCESS(CBL) statements:

000001 CBL TRUNC(OPT) NOSSRANGE NOOPT LIST NOOFFSET

00000113

0Options in effect:

NUMPROC(NOPFD)

```
000022      000023*-----*
000023      000024* 1. Anfang Felddefinitionen *
000024      000025*- *
000025      000026 01 DATEN.
000026      000027      05 STRUK-1.
000027      000028          10 S1-AG          PIC S9(02).
000028      000029          10 S1-VSNR       PIC S9(09) PACKED-DECIMAL.
000029      000030      05 STRUK-2.
000030      000031          10 S2-AG          PIC S9(02).
000031      000032          10 S2-VSNR       PIC S9(09) PACKED-DECIMAL.
000032      000051*- *
000033      000052* 1. Ende   Felddefinitionen *
000034      000053*-----*
000063      000084*- *
000064      000085 V00-VERARBEITUNG SECTION.
000065      000086      MOVE S1-AG      TO S2-AG
000066      000087      MOVE S1-VSNR   TO S2-VSNR
000067      000098      CONTINUE.
```

000064 *V00-VERARBEITUNG

000065 MOVE

0003FA F211 D148 8000

PACK 328(2,13),0(2,8)

TS2=0

000400 F811 D148 D148

ZAP 328(2,13),328(2,13)

TS2=0

000406 F311 8007 D148

UNPK 7(2,8),328(2,13)

S2-AG

000066 MOVE

00040C F844 8009 8002

ZAP 9(5,8),2(5,8)

S2-VSNR

000067 CONTINUE

COBOL Compile Options

NUMPROC(PFD) und NOOPT

1PP 5655-G53 IBM Enterprise COBOL for z/OS 3.4.1
0Invocation parameters:

Date 01/

0PROCESS(CBL) statements:

000001 CBL TRUNC(OPT) NOSSRANGE NOOPT LIST NOOFFSET NUMPROC(PFD) 00000114
0Options in effect:

```
000022      000023*-----*
000023      000024* 1. Anfang Felddefinitionen *
000024      000025*- **
000025      000026 01 DATEN.
000026      000027      05 STRUK-1.
000027      000028          10 S1-AG          PIC S9(02).
000028      000029          10 S1-VSNR         PIC S9(09) PACKED-DECIMAL.
000029      000030      05 STRUK-2.
000030      000031          10 S2-AG          PIC S9(02).
000031      000032          10 S2-VSNR         PIC S9(09) PACKED-DECIMAL.
000032      000051*- **
000033      000052* 1. Ende   Felddefinitionen *
000034      000053*-----*
000035      000054*- **
000064      000085 V00-VERARBEITUNG SECTION.
000065      000086          MOVE S1-AG      TO S2-AG
000066      000087          MOVE S1-VSNR     TO S2-VSNR
000067      000098          CONTINUE.

000064 *V00-VERARBEITUNG
000065 MOVE
      0003FA D201 8007 8000          MVC  7(2,8),0(8)          S2-AG
000066 MOVE
      000400 D204 8009 8002          MVC  9(5,8),2(8)          S2-VSNR
000067 CONTINUE
```

COBOL Compile Options

NUMPROC(NOPFD) und OPT(FULL)

IPP 5655-G53 IBM Enterprise COBOL for z/OS 3.4.1
0Invocation parameters:

Date 01/1

0PROCESS(CBL) statements:

000001 CBL TRUNC(OPT) NOSSRANGE OPT(FULL) LIST NOOFFSET
0Options in effect:

00000112

NUMPROC(NOPFD)

```
000022      000023*-----*
000023      000024* 1. Anfang Feldefinitionen *
000024      000025*- *
000025      000026 01 DATEN. *
000026      000027      05 STRUK-1. *
000027      000028          10 S1-AG          PIC S9(02). *
000028      000029          10 S1-VSNR        PIC S9(09) PACKED-DECIMAL. *
000029      000030      05 STRUK-2. *
000030      000031          10 S2-AG          PIC S9(02). *
000031      000032          10 S2-VSNR        PIC S9(09) PACKED-DECIMAL. *
000032      000051*- *
000033      000052* 1. Ende Feldefinitionen *
000034      000053*-----*

000063      000084*- *
000064      000085 V00-VERARBEITUNG SECTION. *
000065      000086      MOVE S1-AG          TO S2-AG *
000066      000087      MOVE S1-VSNR        TO S2-VSNR *
000067      000098      CONTINUE. *

000064 *V00-VERARBEITUNG
000065 MOVE
0002E4 D201 3007 A0A1          MVC 7(2,3),161(10)      (BLW=0)+7
000066 MOVE
0002EA D204 3009 A09C          MVC 9(5,3),156(10)      (BLW=0)+9
000067 CONTINUE
000046 PERFORM
000072 *Z99-ENDE
000074 DISPLAY
0002F0 58F0 202C              L 15,44(0,2)          V(IGZCDSP )
0002F4 4110 A0A3              LA 1,163(0,10)       PGMLIT AT +147
0002F8 05EF                    BALR 14,15
000075 CONTINUE
000048 CONTINUE
000049 GOBACK
```

SSRANGE – Beschreibung

- Prüfen Subscripte
- Prüfen Indexe
- Prüfen var-Felder
- jeweils *vor* Ausführung des Codes

Use SSRANGE to generate code that checks whether subscripts (including ALL subscripts) or indexes try to reference an area outside the region of the table. Each subscript or index is not individually checked for validity; rather, the effective address is checked to ensure that it does not cause a reference outside the region of the table.

Variable-length items are also checked to ensure that the reference is within their maximum defined length.

SSRANGE – mögliches Ergebnis

```
IEF375I JOB/RZSRGEN /START 2006062.2301
IEF376I JOB/RZSRGEN /STOP 2006062.2301 CPU      0MIN 00.36SEC SRB      0MIN 00.01SEC
```

```
* ANF. TES39 *
* ----- *
```

```
* ANF. TES47 *
* ----- *
```

```
* UEBERTRAGEN INDEX :00000001 OK. FELDHINHALTE: 00000002Ü000000002Ü
* UEBERTRAGEN INDEX :00000002 OK. FELDHINHALTE: 00000100Ü000000100Ü
* UEBERTRAGEN INDEX :00000003 OK. FELDHINHALTE: 00000001Ü000000001Ü
* UEBERTRAGEN INDEX :00000004 OK. FELDHINHALTE: 00000001Ü000000001Ü
* UEBERTRAGEN INDEX :00000005 OK. FELDHINHALTE: 00000001Ü000000001Ü
```

```
IGZ0006S The reference to table TABELLE by verb number 01 on line 000100
addressed an area outside the region of the table.
From compile unit TES47 at entry point TES47 at compile unit
offset +00000614 at entry offset +00000614 at address 2F7170A4.
```

```
<> LEAID ENTERED (LEVEL 05/09/2005 AT 11.27)
```

```
<> LEAID PROCESSING COMPLETE. RC=0
```

```
ICEE3DMP V1 R6.0: Condition processing resulted in the unhandled condition.
```

Information for enclave TES39

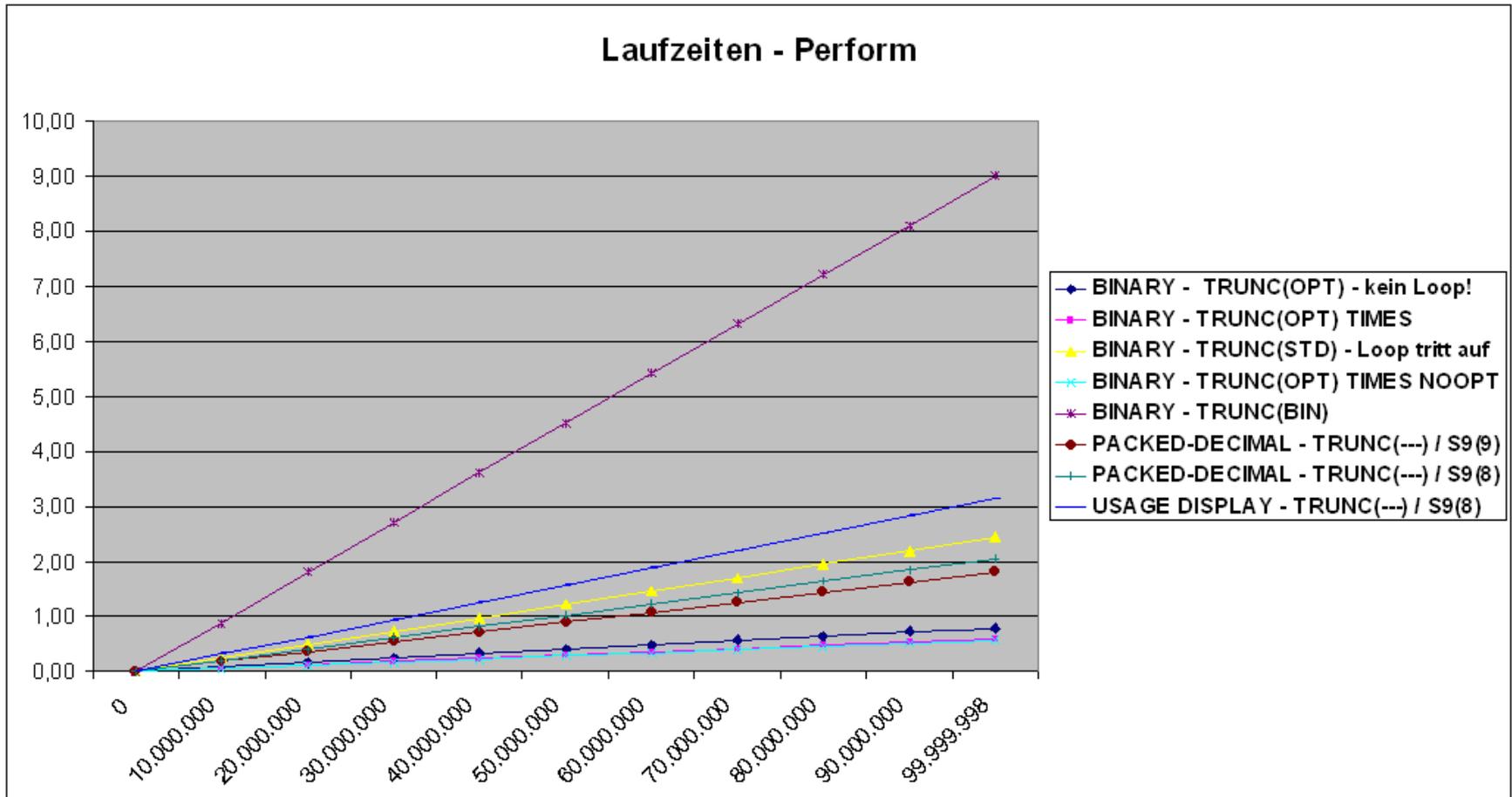
TRUNC – Felddefinitionen – explizite Tests – V3R4

- einfacher Perform -> [Programm](#)

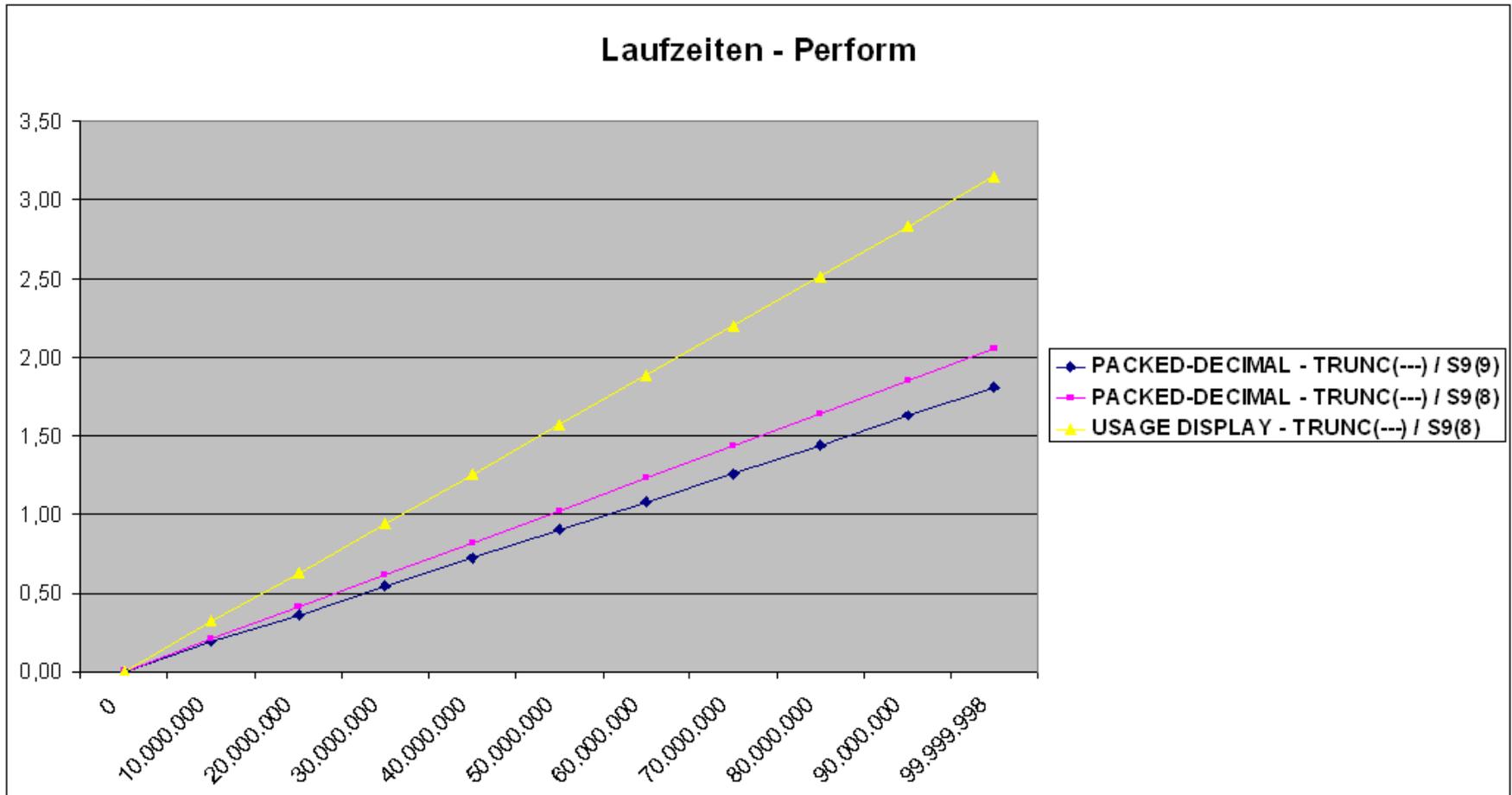
Anzahl	0	10.000.000	20.000.000	30.000.000	40.000.000	50.000.000	60.000.000	70.000.000	80.000.000	90.000.000	99.999.998
BINARY - TRUNC(OPT) - kein Loop!	0,00	0,09	0,16	0,24	0,32	0,41	0,48	0,56	0,64	0,73	0,79
BINARY - TRUNC(OPT) TIMES	0,00	0,07	0,12	0,18	0,24	0,30	0,35	0,41	0,47	0,54	0,59
BINARY - TRUNC(STD) - Loop tritt auf	0,00	0,25	0,49	0,73	0,98	1,22	1,46	1,71	1,95	2,20	2,45
BINARY - TRUNC(OPT) TIMES NOOPT	0,00	0,06	0,11	0,17	0,23	0,29	0,34	0,39	0,45	0,51	0,57
BINARY - TRUNC(BIN)	0,00	0,87	1,81	2,71	3,62	4,52	5,42	6,33	7,23	8,10	9,01
PACKED-DECIMAL - TRUNC(---) / S9(9)	0,00	0,19	0,36	0,54	0,72	0,90	1,08	1,26	1,44	1,63	1,81
PACKED-DECIMAL - TRUNC(---) / S9(8)	0,00	0,21	0,41	0,62	0,82	1,03	1,23	1,44	1,64	1,85	2,05
USAGE DISPLAY - TRUNC(---) / S9(8)	0,00	0,32	0,63	0,94	1,26	1,57	1,88	2,20	2,51	2,83	3,15

- Ergebnisse -> [Chart – alle](#)
- Ergebnisse -> [Chart – dec/dis](#)
- Ergebnisse -> [Chart – binary](#)

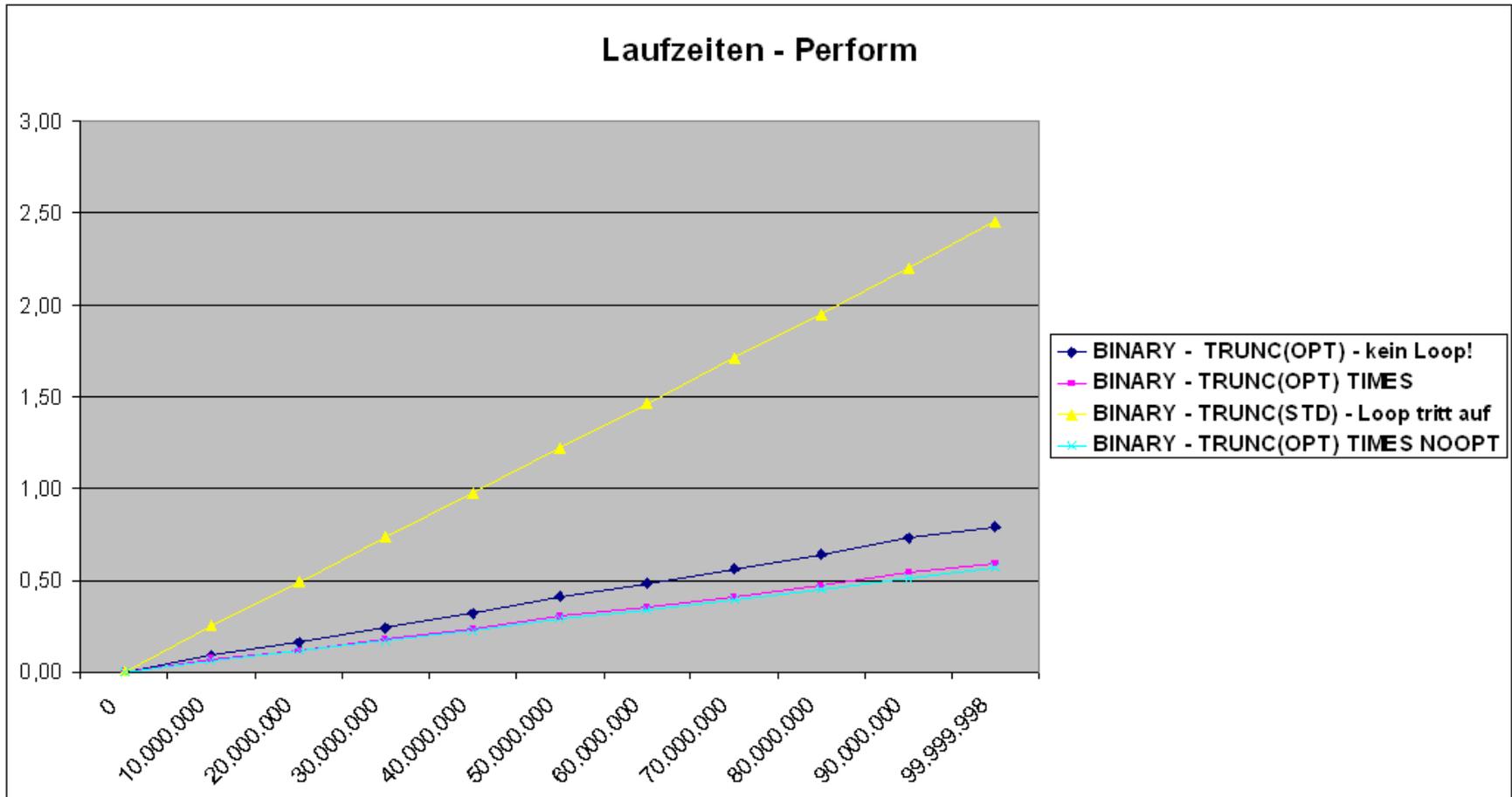
- Ergebnisse -> [Chart – alle](#)



- Ergebnisse -> [Chart – dec/dis](#)



- Ergebnisse -> [Chart – binary](#)



COBOL Compile Options / Code



Beispielprogramm – BINARY – SYNC – 1

```
000022      000023*-----*
000024      000025*-
000025      000026 01  DATEN.
000026      000027      05  filler                PIC  X(01).
000027      000028      05  BIN-FELD-1           PIC  S9(04) BINARY.
000028      000029      05  filler                PIC  X(02).
000029      000030      05  BIN-FELD-1-SYNC       PIC  S9(04) BINARY SYNC.
000030      000031      05  filler                PIC  X(02).
000031      000032      05  BIN-FELD-2           PIC  S9(08) BINARY.
000032      000033      05  filler                PIC  X(02).
000033      000034      05  BIN-FELD-2-SYNC       PIC  S9(08) BINARY SYNC.
000034      000035*-
000036      000037*-----*
000054      000055*-
000056      000057      MOVE ZEROES TO  BIN-FELD-1
000057      000058      MOVE ZEROES TO  BIN-FELD-1-SYNC
000058      000059      MOVE ZEROES TO  BIN-FELD-2
000059      000060      MOVE ZEROES TO  BIN-FELD-2-SYNC
000060      000061      DISPLAY          BIN-FELD-1
000061      000062      DISPLAY          BIN-FELD-1-SYNC
000062      000063      DISPLAY          BIN-FELD-2
000063      000064      DISPLAY          BIN-FELD-2-SYNC

000071      000072      ADD  1          TO  BIN-FELD-1
000072      000073      ADD  1          TO  BIN-FELD-1-SYNC
000073      000074      ADD  1          TO  BIN-FELD-2
000074      000075      ADD  1          TO  BIN-FELD-2-SYNC
```

COBOL Compile Options / Code

Beispielprogramm – BINARY – SYNC – 2

```
000055  -ADD-INIT
000056  MOVE
    000280  4120 0000
    000284  5830 912C
    000288  4020 3001
000057  MOVE
    00028C  4020 3006
000058  MOVE
    000290  5020 300A
000059  MOVE
    000294  5020 3010
000060  DISPLAY
    000298  5820 905C
    00029C  58F0 202C
    0002A0  4110 A08B
    0002A4  05EF
000061  DISPLAY
    0002A6  58F0 202C
    0002AA  4110 A07F
    0002AE  05EF
000062  DISPLAY
    0002B0  58F0 202C
    0002B4  4110 A073
    0002B8  05EF
000063  DISPLAY
    0002BA  58F0 202C
    0002BE  4110 A067
    0002C2  05EF
000064  CONTINUE
000047  PERFORM
000070  *V00-VERARBEITUNG
000071  ADD
    0002C4  4140 0001
    0002C8  4040 3001
000072  ADD
    0002CC  4040 3006
000073  ADD
    0002D0  5040 300A
000074  ADD
    0002D4  5040 3010
000075  CONTINUE
```

LA	2,0(0,0)	
L	3,300(0,9)	BLW=0
STH	2,1(0,3)	(BLW=0)+1
STH	2,6(0,3)	(BLW=0)+6
ST	2,10(0,3)	(BLW=0)+10
ST	2,16(0,3)	(BLW=0)+16
L	2,92(0,9)	TGTFIXD+92
L	15,44(0,2)	V(IGZCDSP)
LA	1,139(0,10)	PGMLIT AT +127
BALR	14,15	
L	15,44(0,2)	V(IGZCDSP)
LA	1,127(0,10)	PGMLIT AT +115
BALR	14,15	
L	15,44(0,2)	V(IGZCDSP)
LA	1,115(0,10)	PGMLIT AT +103
BALR	14,15	
L	15,44(0,2)	V(IGZCDSP)
LA	1,103(0,10)	PGMLIT AT +91
BALR	14,15	
LA	4,1(0,0)	
STH	4,1(0,3)	(BLW=0)+1
STH	4,6(0,3)	(BLW=0)+6
ST	4,10(0,3)	(BLW=0)+10
ST	4,16(0,3)	(BLW=0)+16

Zusammenfassung

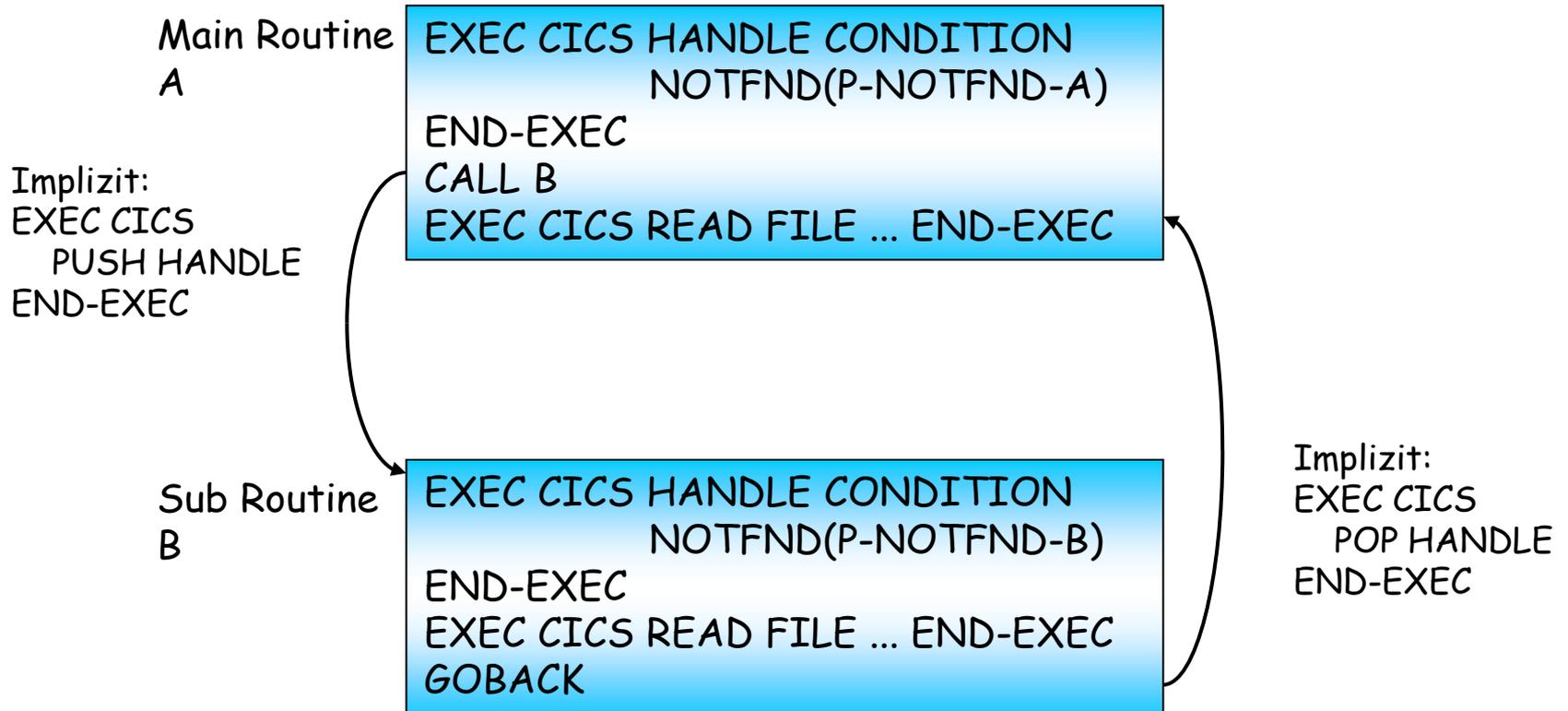
1. Überlegen, welche Option welche Auswirkungen hat.
 - Umgebung, Typ des Programms beachten
2. Hin und wieder auf Basis Assembler Listing entscheiden, was Sinn macht.
3. Nicht optimieren, weil es Spaß macht, sondern optimieren, weil/wo es Sinn macht.
4. Die fachlichen Hintergründe sind ein wesentlicher Maßstab zu entscheiden, wann welche Option eingesetzt wird.
5. COBOL schüttelt man nicht aus dem Ärmel.



- Der Weg in Richtung 64-bit-Adressierung muss u.a. wegen der wachsenden Datenmengen konsequent verfolgt werden!
- LE-Option ALL31(ON) spart laut IBM ca. 3% der gesamten CPU-Last. Diese Option kann nicht gesetzt werden, so lange noch Anwendungsprogramme below-the-line laufen müssen.

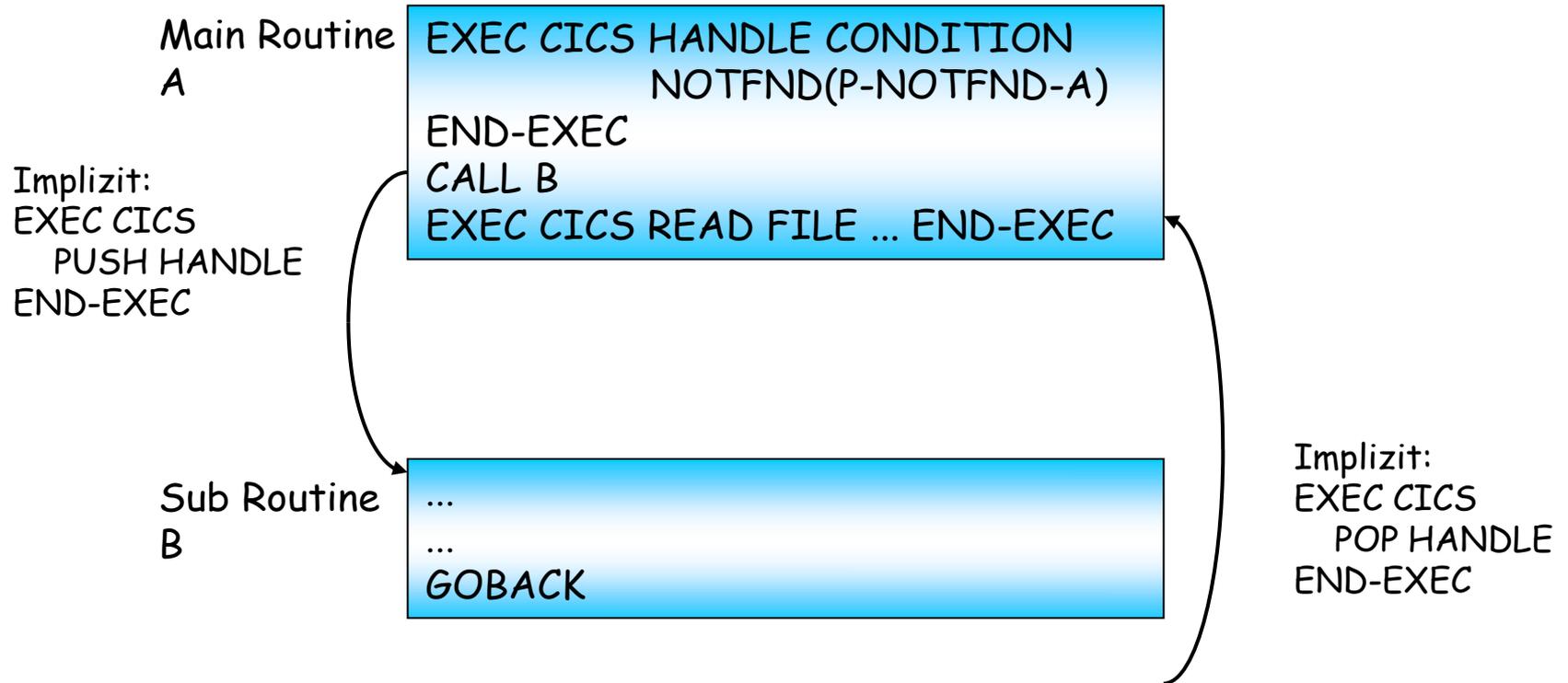
LE Options

CBLPUSHPOP(ON) – 1



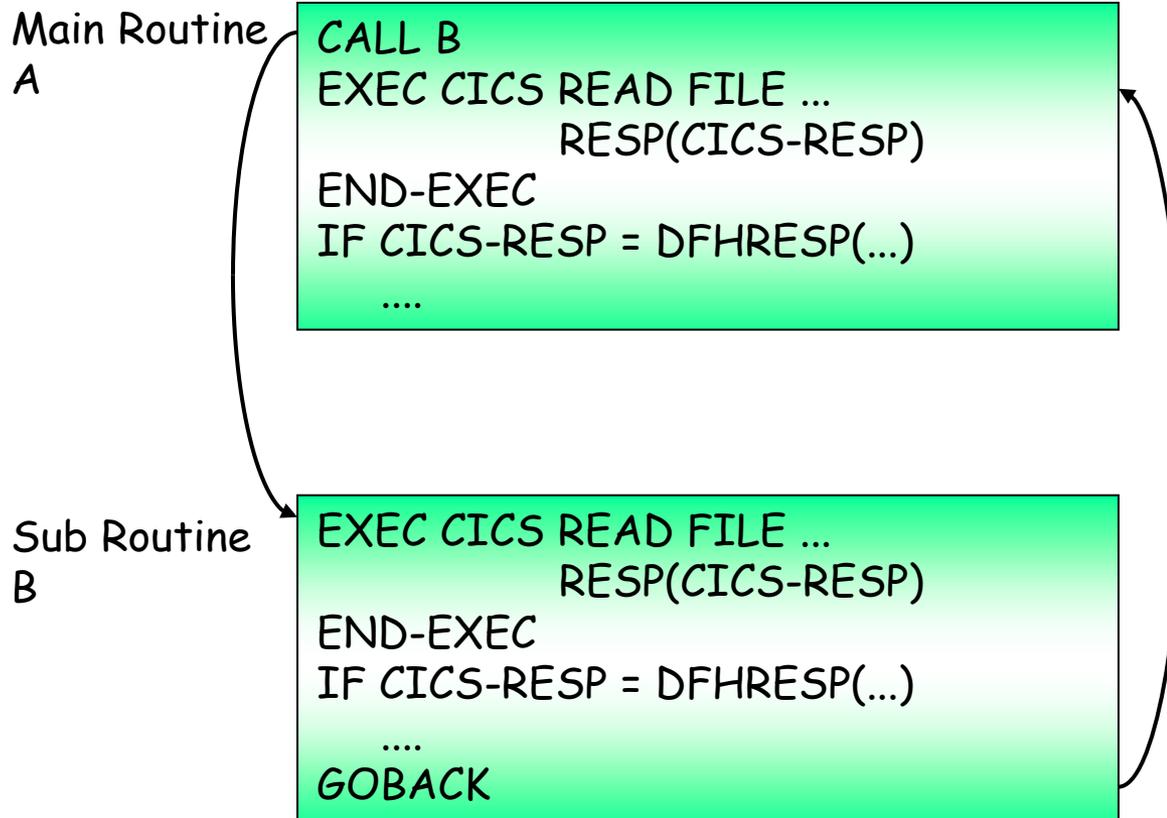
LE Options

CBLPUSHPOP(ON) – 2



LE Options

CBLPUSHPOP(OFF)



Voraussetzung:

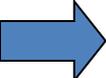
- kein Handle Condition
- kein Handle Abend
- kein Handle AID

Empfehlung:

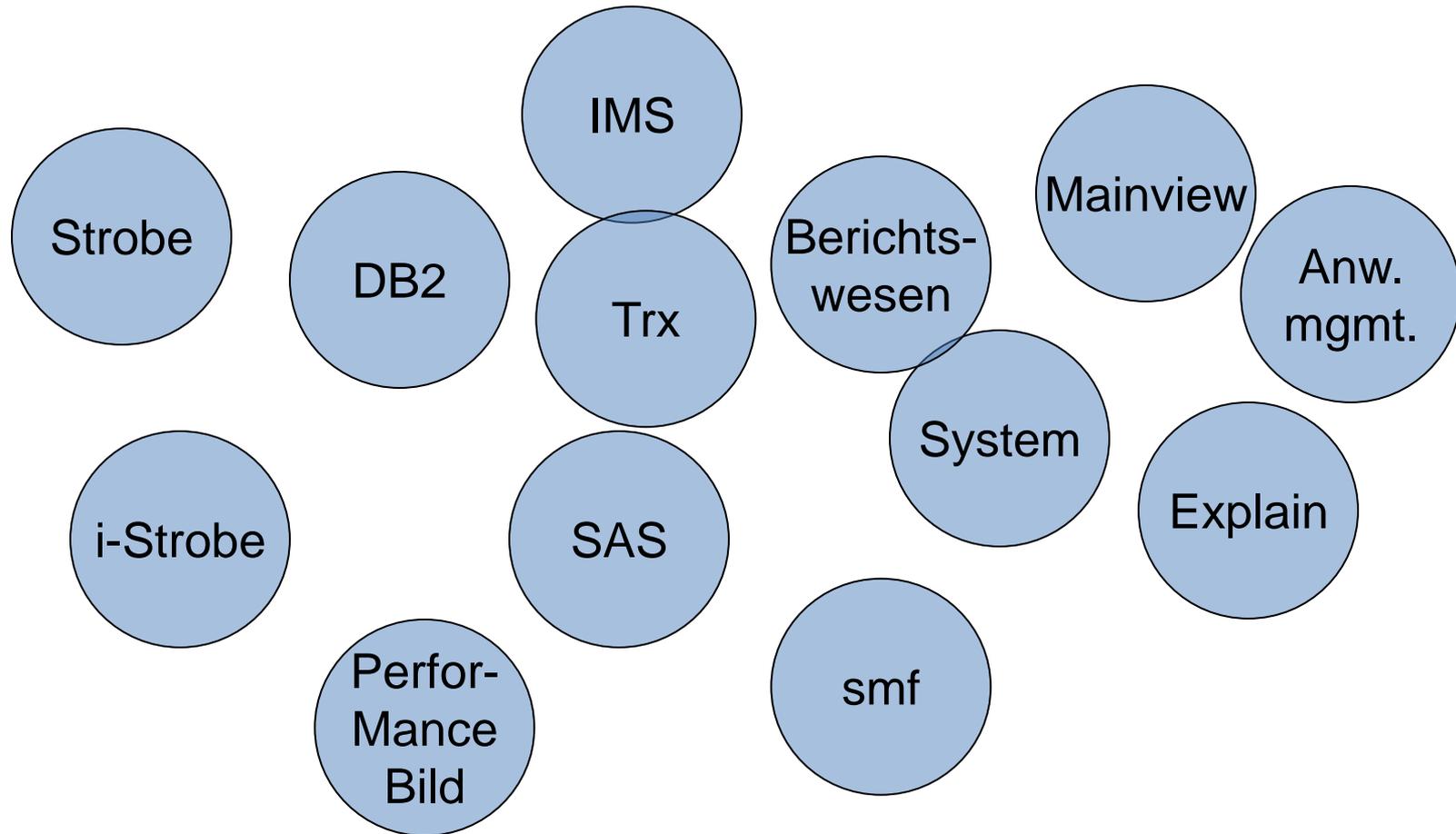
- CICS-Commands mit Resp-Option
- LE-Condition-Handling

CBLPUSHPOP: Danke an Rita Backstein



-
- Vorstellung und Einführung
 - Optimierungen – Beispiele und Potential
 - Richtlinien
 - Modellierung und DB2-Zugriffe
 - COBOL–Felder – COBOL-Befehle
 - Auswirkungen von Optionen – COBOL – LE
 -  • Informationen und Tools bei Firma X
 - Strobe – Handling und Interpretation
 - Diskussion - Austausch

Begriffe



Informationen

- Performancebilder von Transaktionen (Basis: smf mit SAS)
- DB2-Auswertungen (Basis: Mainview)
- Explain-Daten (bei Freigabe)
- Mainview
- Strobe / i-Strobe
- ...

Performancebilder

- konsolidierte Informationen für jede CICS-Trx aus S-Test und Produktion pro Tag
 - CPU-Verbrauch CICS, DB2, IMS
 - Antwortzeiten CICS, DB2, IMS
 - TOP-DB2-Packages (S: 10, P:4)
 - IMS-DBen mit maximalen DB-Calls
 - Speicherung als Hostdateien seit Okt 2005
 - gleiche Quelle wie Monatsbericht

Performancebilder – Beispiele

- Transaktions-Hitliste Produktion
- Transaktions-Hitliste S-Test
- SAS-Dateien
 - Dateinamen: T76MVS.SAS.PBSSTxxP
 - Lieferung als Excel möglich (AP: Herr Globisch)



Performancebilder – Bewertung

- Daten sind vorhanden, aber keiner kümmert sich so richtig
- “Profis” erhalten keinen Auftrag
- AE fehlt Know-how und/oder Anleitung für Interpretation der Daten

DB2-Auswertungen / Explain-Daten

- regelmäßige halbautomatische Auswertungen
 - Basis: Explain-Daten
 - Basis: Mainview (läuft immer mit)
- bei Problemen erfolgt Kontaktaufnahme mit verantwortlichen Stellen

- Kontrolle ist personenbezogen
- Zugang zu Daten ist personenbezogen
- Interpretation der gesammelten Daten relativ einfach
- Interpretation der Explain-Daten muss gelernt werden

Mainview / Strobe / i-Strobe

- TSO %MAINVIEW
 - Plex Management
 - Leitfaden offen
- TSO %STROBE
 - Leitfaden
- i-Strobe
 - Profilerstellung
 - ftp
 - Zugang: `http:\\mvss\istrobe\TestRuV`
 - Klein-/Großbuchstaben beachten

[Link](#)

[Link](#)

[Link](#)

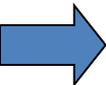
[Link](#)

[Link](#)

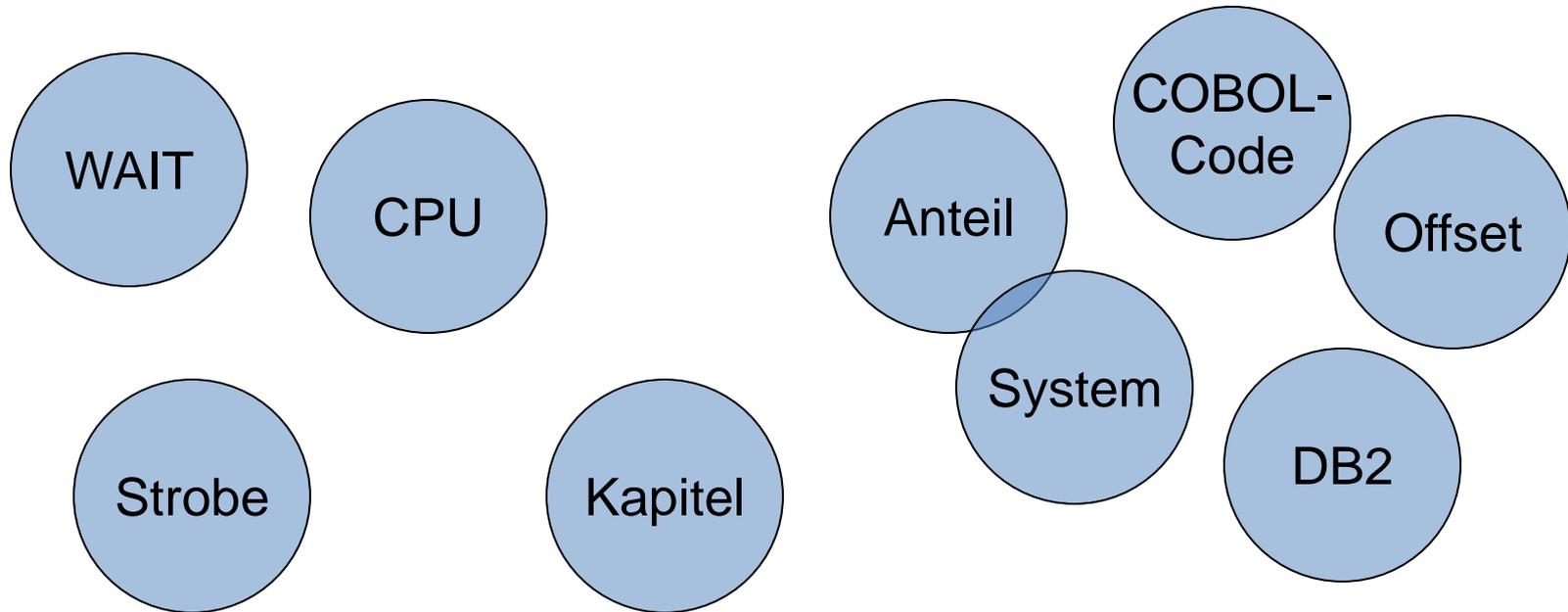
Umgebungen – Firma X

- Produktion – MVSA, MVSC
 - Verantwortung: Anwendungsmanagement
 - jeder kann Messungen aufsetzen, sollte aber nicht
- R-Test
 - Verantwortung: Anwendungsmanagement
 - jeder kann Messungen aufsetzen, sollte aber nicht
- Testumgebungen (T,S,B)
 - Verantwortung AE
 - Supportstelle: nicht festgelegt



-
- Vorstellung und Einführung
 - Optimierungen – Beispiele und Potential
 - Richtlinien
 - Modellierung und DB2-Zugriffe
 - COBOL–Felder – COBOL-Befehle
 - Auswirkungen von Optionen – COBOL – LE
 - Informationen und Tools bei Firma X
 -  • Strobe – Handling und Interpretation
 - Diskussion - Austausch

Begriffe



Ziel des Kapitels

- Das vorliegende Kapitel will versuchen, an Hand von konkreten Beispielen den Weg der Analyse zu beschreiben. Ziel ist es, so genannte „Eye Catcher“, d.h. offensichtliche Fehler, zu beleuchten. Diese treten in gut 90% aller Fälle auf. Für spezielle Analysen sollten stets Spezialisten hinzu gezogen werden.
- Das Kapitel beinhaltet Auszüge aus Messungen. Es wurden nur die relevanten Kapitel bzw. Kapitelteile aus den Messungen aufgenommen. Hinweise sind mit einem → gekennzeichnet und umrahmt.

Datei – 1

Date: 2003.07.06 Job: WN281409 N2814 IKJEFT01

Chapter : #MSD

```
----- JOB ENVIRONMENT -----      ----- MEASUREMENT STATISTICS -----
PROGRAM MEASURED   -   IKJEFT1B      CPS TIME PERCENT   -   11.18
JOB NAME           -   WN281409      WAIT TIME PERCENT  -   88.82
JOB NUMBER         -   JOB18936      RUN MARGIN OF ERROR PCT -   .94
STEP NAME          -   N2814.IKJEFT01 CPU MARGIN OF ERROR PCT -   2.81
DATE OF SESSION    -   07/06/2003    TOTAL SAMPLES TAKEN -   20,877
TIME OF SESSION    -   17:52:06      TOTAL SAMPLES PROCESSED - 10,876
CONDITION CODE     -   C-0000        INITIAL SAMPLING RATE- 16.67/SEC
                                           FINAL SAMPLING RATE  -   8.33/SEC

SYSTEM -           z/OS   01.03.00
DFSMS              -           1.3.0   SESSION TIME -   21 MIN 47.72 SEC
SUBSYSTEM          -           DB2 7.1.0 CPU TIME -           2 MIN 9.89 SEC
DB2 SUBSYSTEM ID  -           D205    WAIT TIME -           17 MIN 11.85 SEC
DB2 APPLICATION   -           N2814   STRETCH TIME -           2 MIN 25.97 SEC
CPU MODEL          -           2064-116
SYSTEM ID          -           P005    SRB TIME -           0 MIN 4.54 SEC
LPAR               -           P005    SERVICE UNITS-           1,054,317
```

→ WAIT-Time ist überproportional hoch

→ WAIT-Analyse erforderlich

Datei – 2

Chapter : #TDA

TASK OR DDNAME	RESOURCE	N*10 = % FULL UTILIZATION;	* IS > 95%;	- IS < 5%								
DSNECP10	CPU	.-	-132333333323221223-	-111112222211.								
DSN	CPU	.		.								
END	CPU	.		.								
N2UMSO2	3490	.	24666767898999989995	2889998899***.								
N2CONI4	3490	.	27553364333533264236	2332222332111.								
N2UMSI2	3490	.	24566656455443423442	1334333343322.								
N2CONO5	3490	.	14232243122321132123	-422111221-1-.								
. FILEMGT		.	-	- - -.								
N2TXTO3	3490	.	1-21111-111111--112	-11112112----.								
N2TXTI3	3490	.	--11-11-1111--1---1	- -1-1-11- - .								
N2AUSO4	3490	.	--- - - - -	-- ---- .								
		0----	1----	2----	3----	4----	5----	6----	7----	8----	9----	*.
		0----	0----	0----	0----	0----	0----	0----	0----	0----	0----	*.
		START RUN	% ALLOCATED RUN TIME	END RUN								

- ➔ 30% der Laufzeit zu Beginn wird nichts getan
- ➔ 5% der Laufzeit ab 70% wird nichts getan
- ➔ Verursacher?

WAIT-Analyse

Datei – 4

Chapter : #WTM

MODULE	SECTION	COMPRESSED	FUNCTION	% RUN	TIME	MARGIN	OF ERROR	.94%
NAME	NAME			PAGE	TOTAL	00	19.00	38.00
.DB2	DSNVSR		SUSP/RES/CANCE	.00	2.15	++		
.IOCS	IGG019AQ		QSAM GET NEXT	.00	15.05	+++++++		
.IOCS	IGG019AR		QSAM PUT NEXT	.00	33.70	+++++		
.IOCS	IGG019CW		SAM EOB CHAIN.	.00	.32			
				----	-----			
.IOCS	TOTALS		DATA MANAGEMENT	.00	49.07			
.NUCLEUS	IEAVESLL		SUSPEND LOCK S	.00	.07			
.SVC	SVC 001		WAIT	.00	37.50	+++++		
.SVC	SVC 006		PROGRAM MANAGE	.00	.01			
.SVC	SVC 119		TESTAUTH	.00	.01			
				----	-----			
.SVC	TOTALS		SUPERVISOR CONTR	.00	37.52			

- ➔ Verursacher der WAITs:
Datei lesen, Datei schreiben und expliziter WAIT
- ➔ expliziter WAIT passt zu 30% am Beginn des Jobs
- ➔ Joblog-Analyse ergibt in diesem Fall:
das war ein WAIT auf Kasette

Datei – 6

Chapter : #SWS

SQL	SQL	STMT-EXECUTION TIME/CNT		% RUN TIME	MARGIN OF ERROR	.94%
TYPE	NAME	CNT	AVG-TIME	PAGE	TOTAL	00 4.00 8.00
DBRM	N2814	32,816	.0010	13:15:40	1.08 1.84	****+
				-----	-----	
TOTAL WAIT ACTIVITY				1.31	2.13	

→ Nichts Außergewöhnliches

Datei – 7

Chapter : #WBS

DBRM - N2J36

CREATED - 11/23/1999 11:06:38

STATIC, NON-CURSOR SQL

```
3138 SELECT MANDANTK, ZAUFRID, ZVUMLFNR, STUMLBEA, KTOINHNR, KTONR, ISOWHRCD, BUCHDA
      T, UABUKTOU, UMSUMART, UMSSPTS
      INTO :H, :H FROM RWBUUIVI WHERE MANDANTK=:H AN
      D KTOINHNR=:H AND KTONR=:H AND ISOWHRCD=:H AND BUCHDAT=:H AND UMSSPTS=:H
```

STMT NUMBER	STATEMENT TEXT	STMT-EXECUTION CNT	AVG-TIME	% RUN TIME PAGE	TIME TOTAL	MARGIN OF ERROR 00	1.00	.94% 2.00
3138	SELECT			.09	.13	**		
		19,128	.0001					
		-----	-----	-----	-----			
	TOTALS	19,128	.0001	.09	.13			

→ Nichts Außergewöhnliches

Datei – 8

Chapter : #ACW

```
-----INVOKED BY-----          -----VIA-----          -WAIT TIME%-
XACTION  MODULE  SECTION  RETURN  LINE  MODULE  SECTION  PAGE  TOTAL
.DB2     DSNVSR                SUSP/RES/CANCEL SYNCHRO

XACTION  QUERY NAME          TIME          TEXT          STMT  TEXT          PAGE  TOTAL
N2814    N2814                13:15:40      5387 OPEN          .00  1.24
-----  -----
                          .00  2.15

.SVC     SVC 001                WAIT

          N2814    N2814    007962          IGZEQOC          .00  22.85
          N2814    N2814    007B8E          IGZEQOC          .00  2.50
          N2814    N2814    007D9E          IGZEQOC          .00  3.74
N2814    .IOCS    IGG019AQ          QSAM G    SVC 055          .00  8.10
```

→ Adresse 007962 in N2814 zeigt auf einen OPEN

→ Gesamtergebnis:

- WAIT-Zeiten gehen auf die Dateiverarbeitung zurück
- Programm-technisch keine Optimierung möglich
- Buffer überprüfen gegen Empfehlungen von verantwortlichen Stellen (gibt es schon Empfehlungen??)

DB2 – Index – 1

Date: 2002.11.14 Job: TN3A5K07 N3A56 IKJEFT01

----- JOB ENVIRONMENT -----			----- MEASUREMENT STATISTICS -----		
PROGRAM MEASURED	-	IKJEFT1B	CPS TIME PERCENT	-	95.21
JOB NAME	-	TN3A5K07	WAIT TIME PERCENT	-	4.79
JOB NUMBER	-	JOB22610	RUN MARGIN OF ERROR PCT	-	.94
STEP NAME	-	N3A56.IKJEFT01	CPU MARGIN OF ERROR PCT	-	.96
DATE OF SESSION	-	11/14/2002	TOTAL SAMPLES TAKEN	-	50,844
TIME OF SESSION	-	00:44:54	TOTAL SAMPLES PROCESSED	-	10,843
CONDITION CODE	-	C-0000	INITIAL SAMPLING RATE	-	11.11/SEC
			FINAL SAMPLING RATE	-	0.69/SEC
SYSTEM	-	z/OS 01.01.00	SESSION TIME	-	259 MIN 53.57 SEC
DFSMS	-	2.10.0	CPU TIME	-	202 MIN 1.73 SEC
SUBSYSTEM	-	DB2 7.1.0	WAIT TIME	-	10 MIN 9.37 SEC
DB2 SUBSYSTEM ID	-	D203	STRETCH TIME	-	47 MIN 42.46 SEC
DB2 APPLICATION	-	N3A56			

→ CPU-Analyse erforderlich

DB2 – Index – 2

Achtung: Nur Beschreibung des DB2-Moduls DSNK2DM; dies hat nichts mit dem SQL Fetch zu tun!

#IEP

MODULE NAME	SECTION NAME	LINE NUMBER	PROCEDURE NAME	START LOC	% CPU TIME SOLO TOTAL
.DB2	DSNK2DM		DSNKFTCH		66.69 66.70
.DB2	DSNK2DM		DSNKNXT2		22.55 22.60
.DB2	DSNBBM		DSNB1GET		6.20 6.21
.DB2	DSNBBM		DSNB1REL		2.41 2.41
.DB2	DSNBBM		DSNB1CPF		.53 .53
.DB2	DSNVSR		SUSP/RES/CANCEL		.33 .33
.DB2	DSNXGRDS		RDS ACCESS MODULE		.30 .30
.DB2	DSNIDM		DATA MANAGEMENT		.28 .28
.NUCLEUS	IEAVESLK		SUSPEND LOCK SERVICE		.07 .07
N3A56				01BD00	.04 .04

→ schlechte Index-Nutzung

→ Gesamtergebnis

- Hohe %-Zahl bei dem Text “FETCH TYPE 2 IDX ...” weist auf schlechte Index-Nutzung hin
- Index fehlt oder wird nicht / kaum benutzt
- In Kapitel #SUS findet man den DBRM
- In Kapitel #CSS findet man den SQL
- auch ein fehlender Run-Stats kann die Ursache sein
- Kontaktaufnahme mit DBA

DB2 – Table-Space-Scan – 1

Date: 2003.06.06 Job: MN5952Q1 N5952 IKJEFT01

→ CPU-Analyse war bei diesem Job erforderlich

#IEP

Achtung: Nur Beschreibung des DB2-Moduls DSNK2DM; dies hat nichts mit dem SQL Fetch zu tun!

MODULE NAME	SECTION NAME	LINE NUMBER	PROCEDURE NAME	START LOC	% CPU SOLO	TIME TOTAL
.DB2	DSNIDM		DSNIRNXT FETCH NEXT ROW TO PROG		68.53	68.53
.DB2	DSNK2DM		DSNKFTCH FETCH TYPE 2 IDX ENTRY		5.93	5.93
.DB2	DSNBBM		DSNB1GET RETRIEVE REQUESTED PAGE		4.49	4.49
.DB2	DSNXGRDS		DSNXSINE RETR/BLD BLK OF SRT RECS		3.60	3.60
.DB2	DSNXGRDS		DSNXSTSE RDS TREE SORT MODULE		2.73	2.73
.DB2	DSNXGRDS		DSNXSMRE RDS MERGE MOD		1.50	1.50
.DB2	DSNBBM		DSNB1REL PAGE RELEASE ROUTINE		1.40	1.40
.DB2	DSNXGRDS		DSNXRRP RTIME RESIDUAL PRED EXEC		1.27	1.27
.DB2	DSNIDM		DSNIONX2 NEXT ON CUB ON IXED FAN		1.02	1.02
.DB2	DSNK2DM		DSNKNXT2 FETCH TYPE 2 IDX ENTRY		1.01	1.01

→ Hinweis auf Table-Space-Scan
→ Wer ist Verursacher?

DB2 – Table-Space-Scan – 2

Chapter : #SUS

SQL	SQL	STMT-EXECUTION	TIME/CNT	% CPU	TIME	MARGIN OF ERROR	.77%	
TYPE	NAME	CNT	AVG TIME	SOLO	TOTAL	00	29.00	58.00
DBRM	N5B18	943	.4409	14:42:13	20.00	20.00	*****	
DBRM	N5J00	4,004	.0163	13:39:10	2.11	2.11	*	
DBRM	N5J08	1,501	.1341	10:37:59	10.27	10.27	****	
DBRM	N5J22	1,489	.0839	10:43:08	6.34	6.34	***	
DBRM	N5X78	1,501	.7503	07:59:40	56.97	56.97	*****	
TOTAL SQL CPUUSAGE				97.79	97.79			

→ es könnte 3-4 Verursacher geben, daher

→ Packages genauer prüfen

→ hier nicht aufgeführt, aber ...

→ Kapitel #ACE gibt Hinweise, wer der Aufrufer des häufig benutzten DB2-Moduls „FETCH NEXT ROW...” ist

→ hier Konzentration auf N5X78

DB2 – Table-Space-Scan – 3

#CSS

DBRM - N5X78

CREATED - 10/17/2002 07:59:40

LOCATION: DECOM_DB2N

STATIC, NON-CURSOR SQL

7078 DELETE FROM RWAEZUVI WHERE MANDANTK=:H AND KTOINHNR=:H AND RWKTONR=:H AND
ISOWHRCD=:H

STMT NUMBER	STATEMENT TEXT	STMT-EXECUTION CNT	AVG-TIME	% CPU TIME SOLO	MARGIN OF ERROR TOTAL	00	29.00	.77% 58.00
7078	DELETE	1,501	.7503	56.97	56.97	*****		
	TOTALS	1,501	.7503	56.97	56.97			

→ Es ist der DELETE

→ Gesamtergebnis

- genau der betrachtete DELETE ist der Verursacher
- Kontaktaufnahme mit DBA erforderlich

DB2 – Aufrufzahlen – 1

Date: 2003.06.28 Job: WN6402J2 N6402 IKJEFT01

#MSD

```
----- JOB ENVIRONMENT -----      ----- MEASUREMENT STATISTICS -----
PROGRAM MEASURED   -   IKJEFT1B      CPS TIME PERCENT   -   94.55
JOB NAME           -   WN6402J2      WAIT TIME PERCENT  -   5.45
JOB NUMBER         -   JOB18314      RUN MARGIN OF ERROR PCT -   .83
STEP NAME         -   N6402.IKJEFT01  CPU MARGIN OF ERROR PCT -   .85
DATE OF SESSION    -   06/28/2003      TOTAL SAMPLES TAKEN -   24,044
TIME OF SESSION    -   02:32:27      TOTAL SAMPLES PROCESSED - 14,043
CONDITION CODE    -   C-0000         INITIAL SAMPLING RATE- 1.68/SEC
                                                           FINAL SAMPLING RATE  - 0.84/SEC

SYSTEM -          z/OS   01.03.00
DFSMS             -          1.3.0   SESSION TIME - 278 MIN 8.60 SEC
SUBSYSTEM         -          DB2 7.1.0 CPU TIME - 216 MIN 27.68 SEC
DB2 SUBSYSTEM ID  -          DB2N    WAIT TIME - 12 MIN 28.27 SEC
DB2 APPLICATION   -          N6402   STRETCH TIME - 49 MIN 12.64 SEC
```

→ sehr hoher CPU-Verbrauch im Vergleich zur WAIT-Zeit
→ CPU-Analyse erforderlich

DB2 – Aufrufzahlen – 2

#IEP

MODULE	SECTION	LINE	PROCEDURE	START	% CPU TIME	
NAME	NAME	NUMBER	NAME	LOC	SOLO	TOTAL
.NUCLEUS	IEAVSTA1		COMM TASK ESTAE		6.93	6.93
.DB2	DSNK2DM		DSNKFTCH FETCH TYPE 2 IDX ENTRY		6.12	6.12
.DB2	DSNXGRDS		DSNXRTIM RDS ACCESS MODULE GENER		5.71	5.71
.DB2	DSNXGRDS		DSNXERD TOPMOST RDS CSECT		5.68	5.68
.DB2	DSNIDM		DSNISFX2 SET FUNC TYPE 2 IDX SCAN		4.87	4.87
.PRIVATE			PRIVATE AREA		4.47	4.47
.DB2	DSNIDM		DSNISRID SET CUB BY LST OF RIDS		3.80	3.80
.DB2	DSN3EPX		DSNAPRHX PGM REQUEST APPL INTERFC		3.50	3.50
.DB2	DSNBBM		DSNB1GET RETRIEVE REQUESTED PAGE		2.61	2.61
.DB2	DSNXGRDS		DSNXERT APPLICATION CALL ROUTINE		2.43	2.43

→ kein eindeutiger Verursacher außer DB2
→ ist es (im) DB2?

DB2 – Aufrufzahlen – 3

#PSU

MODULE	SECTION	16M	FUNCTION	% CPU TIME		MARGIN OF ERROR	.85%
NAME	NAME	<,>		SOLO	TOTAL	00	35.00 70.00
.SYSTEM	.COBLIB		COBOL LIBRARY SUBROUTI	.89	.89		
.SYSTEM	.DB2		DB2 SYSTEM SERVICES	69.63	69.63	*****	
.SYSTEM	.NUCLEUS		MVS NUCLEUS	10.25	10.25	***	
.SYSTEM	.PRIVATE		PRIVATE AREA	4.47	4.47	**	
.SYSTEM	.SMS		SYSTEM MANAGER STORAGE	.64	.64		
				-----	-----		
.SYSTEM	TOTALS		SYSTEM SERVICES	87.11	87.11		
N2X20	N2X20			2.09	2.09	*	
				-----	-----		
N2X20	TOTALS	>		2.11	2.11		
XXA08	XXA08	<		7.40	7.40	***	
ZFU23		<		1.48	1.48		
				-----	-----		
PROGRAM IKJEFT1B	TOTALS			100.00	100.00		

→ Der Verbrauch liegt im DB2

DB2 – Aufrufzahlen – 4

#SUS

SQL	SQL	STMT-EXECUTION	TIME/CNT	% CPU	TIME	MARGIN OF ERROR	.85%
TYPE	NAME	CNT	AVG-TIME	SOLO	TOTAL	00	28.00 56.00
DBRM	N1X85	141,498	.0001	07:48:23	2.08	2.08	*
DBRM	N2J90	126,784	.0000	13:39:43	1.17	1.17	
DBRM	N2X20	8,189,558	.0000	11:04:42	55.17	55.17	*****
DBRM	N6K90	403,801	.0003	06:54:01	10.46	10.46	****
TOTAL SQL CPUUSAGE				68.88	68.88		

→ hohe Zahlen des Package machen die Last

→ Gesamtergebnis

→ Topverbraucher ist das DBRM N2X20

→ die Aufrufzahlen sind zu plausibilisieren

COBOL-Befehle – 1

Date: 2003.06.02 Job: TI9I9G08 I9G08 IMSBMPP

```
----- JOB ENVIRONMENT -----          ----- MEASUREMENT STATISTICS -----
PROGRAM MEASURED   -   DFSRRC00          CPS TIME PERCENT   -           85.70
JOB NAME           -   TI9I9G08          WAIT TIME PERCENT  -           14.30
JOB NUMBER         -   JOB26220          RUN MARGIN OF ERROR PCT -    .77
STEP NAME         -   I9G08.IMSBMPP      CPU MARGIN OF ERROR PCT -    .84
DATE OF SESSION   -   06/02/2003        TOTAL SAMPLES TAKEN -   16,064
TIME OF SESSION   -   17:10:37          TOTAL SAMPLES PROCESSED - 16,064
CONDITION CODE    -   C-0000            INITIAL SAMPLING RATE-  7.58/SEC
                                           FINAL SAMPLING RATE  -  7.58/SEC

SYSTEM -          z/OS   01.03.00
DFSMS             -          1.3.0
SUBSYSTEM        -   IMS BMP 6.1 L=S     SESSION TIME -    35 MIN 18.30 SEC
                                           CPU TIME -      24 MIN 39.71 SEC
                                           WAIT TIME  -    4 MIN 6.88 SEC
                                           STRETCH TIME -  6 MIN 31.70 SEC
                                           DB2 7.1.0
DB2 SUBSYSTEM ID -          DB2Q
```

→ CPU-Analyse erforderlich

COBOL-Befehle – 2

#IEP

MODULE	SECTION	LINE	PROCEDURE	START	% CPU TIME	
NAME	NAME	NUMBER	NAME	LOC	SOLO	TOTAL
.COBLIB	IGZCPAC		IGZCIN1 (V3) INSPECT		22.69	22.69
.COBLIB	IGZCPAC		IGZCUST UNSTRING		15.18	15.18
.DB2	DSNK2DM		DSNKFTCH FETCH TYPE 2 IDX ENTRY		4.10	4.10
I9G10	I9G10			013280	3.66	3.66
.DB2	DSNIDM		DSNIOST2 SET ON CUB DEF ON IX FAN		3.20	3.20
.COMMON	.COMMONX		EXTENDED COMMON AREA		2.85	2.85
.DB2	DSNIDM		DSNIONX2 NEXT ON CUB ON IXED FAN		2.66	2.66
I9G10	I9G10			030A40	2.27	2.27
I9G10	I9G10			030A00	1.78	1.78
.DB2	DSNBBM		DSNB1GET RETRIEVE REQUESTED PAGE		1.69	1.69

→ Verursacher ist klar

→ Weitere Anmerkungen

- das Kapitel #ACE zeigt genau die Adressen, wo die Befehle INSPECT und UNSTRING aufgerufen werden; die Adresse (Adressumgebung) muss in der Umwandlungsliste gesucht werden
- es muss darauf geachtet werden, dass die Umwandlungsliste zum Laufzeitpunkt passt

→ Gesamtergebnis

- Topverbraucher ist das Programm
- Es ist zu prüfen, ob ohne großen Aufwand die CPU-Last auf INSPECT / UNSTRING verringert werden kann

COBOL-Code – 1

Date: 2003.06.03 Job: MDEDEB12 DEB12 IKJEFT01

#MSD

```
----- JOB ENVIRONMENT -----          ----- MEASUREMENT STATISTICS -----
PROGRAM MEASURED   -   IKJEFT1B          CPS TIME PERCENT   -   90.27
JOB NAME           -   MDEDEB12          WAIT TIME PERCENT  -   9.73
JOB NUMBER         -   JOB30383          RUN MARGIN OF ERROR PCT -   .97
STEP NAME         -   DEB12.IKJEFT01     CPU MARGIN OF ERROR PCT -   1.03
DATE OF SESSION    -   06/03/2003        TOTAL SAMPLES TAKEN -   20,109
TIME OF SESSION    -   01:00:12          TOTAL SAMPLES PROCESSED - 10,108
CONDITION CODE     -   C-0000            INITIAL SAMPLING RATE - 1.68/SEC
                                                           FINAL SAMPLING RATE  - 0.84/SEC

SYSTEM -           z/OS   01.03.00
DFSMS              -     1.3.0           SESSION TIME - 200 MIN 5.84 SEC
SUBSYSTEM          -     DB2 7.1.0       CPU TIME - 156 MIN 28.26 SEC
DB2 SUBSYSTEM ID  -           DB2N       WAIT TIME - 16 MIN 52.49 SEC
DB2 APPLICATION   -     DEB12           STRETCH TIME - 26 MIN 45.08 SEC
```

→ CPU-Analyse folgt

COBOL-Code – 2

#IEP

MODULE	SECTION	LINE	PROCEDURE	START	% CPU	TIME
NAME	NAME	NUMBER	NAME	LOC	SOLO	TOTAL
DEU64	DEU64			000EC0	23.48	23.48
DEU56	DEU56			0023C0	17.79	17.79
.NUCLEUS	IEAVSTA1		COMM TASK ESTAE		4.22	4.22
.DB2	DSNXGRDS		DSNXERD TOPMOST RDS CSECT		4.04	4.04
.PRIVATE			PRIVATE AREA		3.65	3.65
.DB2	DSNIDM		DSNIOST2 SET ON CUB DEF ON IX FAN		2.51	2.51
.DB2	DSN3EPX		DSNAPRHX PGM REQUEST APPL INTERFC		2.22	2.22
.DB2	DSNXGRDS		DSNXERT APPLICATION CALL ROUTINE		2.19	2.19
.DB2	DSNXGRDS		DSNXECP COPY APPLCTN STRUCTURES		1.56	1.56
.DB2	DSNK2DM		DSNKFTCH FETCH TYPE 2 IDX ENTRY		1.49	1.49

→ Verursacher ist klar

→ Weitere Anmerkungen

- die Adressen (Adressumgebung) müssen in den Umwandlungslisten gesucht werden
- es muss darauf geachtet werden, dass die Umwandlungsliste zum Laufzeitpunkt passt

→ Gesamtergebnis

- Es ist zu prüfen, ob ohne großen Aufwand die CPU-Last auf dem entsprechenden Code verringert werden kann.
- Hinweise geben die veröffentlichten Empfehlungen (sofern vorhanden).



- Vorstellung und Einführung
- Optimierungen – Beispiele und Potential
- Richtlinien
- Modellierung und DB2-Zugriffe
- COBOL–Felder – COBOL-Befehle
- Auswirkungen von Optionen – COBOL – LE
- Informationen und Tools bei Firma X
- Strobe – Handling und Interpretation
- ➔ • Diskussion - Austausch

