

# Mehr Performance ohne Hardware-Upgrade

Road Map zur Identifizierung und  
Beseitigung der Hauptursachen langer  
Antwortzeiten

**Dr. Martin Rosenbauer**  
(MTI)

**Boris Noack,  
Ulrich Rompel**  
(HZD)





- Rechenzentrum der Hessischen Landesverwaltung
- Lösungen für die Verwaltung
- Wiesbaden und Hünfeld
- 600 Mitarbeiter
- Umsatz 130 Millionen Euro

- Projektleitung für AR System Projekte
- Beratung bei Workflow-Konzeption
- Umsetzung von Workflows
- zeitlich befristete Verstärkung von ARS-Teams
- Schwerpunkte:  
AR System, Perl, Oracle, SQL, UNIX, Schnittstellen

**System  
zu  
LANGSAM**

# Performance

- **Subjektiver Eindruck**
- **Akzeptanzproblem**
  - Service Support
  - Service Delivery
  - Unternehmens-IT  
gesamt

## Übersicht

- I Einführung
  - Performance-Messungen
  - Ursachen schlechter Performance
- II Analyse
  - systematische Versuche,
  - Log-Datei-Analyse
- III Tuning
  - Datenbank
  - Server
  - Workflow
- IV Zusammenfassung
- V Diskussion

„System ist langsamer als gestern...“, „ist unglaublich träge“, ...

- Vortrag gibt kein „Patentrezept“
- Tuning ist Prozess, nicht „*einmalige Maßnahme*“

Teil I des Vortrags: Einführung

1. Schritt: Quantifizierung des Begriffs „zu langsam“
2. Überblick über Architektur und Bottle-Necks



# I Einführung

## Messung von Antwortzeiten

- Manuell (mit Stoppuhr)  
... immer mehrmals messen
- Automatisch (mit Workflow)  
... Statistik!

4

... mit Bleistift und Papier

... oder mit Active Links (z.B. On Submit/Modify mit Execution Level 0  
und „on after

submit“/“after modify“

Auswahl nach versch. Gesichtspunkten:

- Welche Aktionen sind besonders störend für die Benutzer ?
- Welche Aktionen laufen weitgehend/ausschließlich lokal im Client / Browser (Aktive Links, GUI-Operationen etc.)
- Welche Aktionen verlaufen ohne Einfluss des Clients (z.B. Datenbankabfragen, Aufbau von Tabellenfeldern etc.)
- Keine zu kurzen Aktionen messen (>1,5s), sonst zu große Messungenauigkeit/Varianz

## Unser Referenz-Szenario

Aktion	Dauer* (s)	Aktion	Dauer* (s)
Aufruf „e-inventory“-Maske	18,8	Umzugsdialog schließen	2,1
Wechsel Bearbeitungsmodus	7,5	Aktivierung aufrufen	5,6
Wechsel Anzeigemodus	0,8	Aktivierung durchführen	1,5
Erweiterte Suche aufrufen	4,4	Aktivierungsdialog schließen	1,7
Erweiterte Suche abbrechen	1,7	Austausch aufrufen	9,7
Erweiterte Suche, Suche starten	1,8	Austausch abbrechen	2
Speichern im Bearbeitungsmodus	11,8	Austausch durchführen	2,6
Fokus setzen im Bearbeitungsmod	4,9	Austausch schließen	1,1
Eingaben löschen in Schnellsuche	10,1	Ersterfassung aufrufen	5,7
Umzugsdialog aufrufen	8,2	Ersterfassung abbrechen	1,4
Umzug abbrechen	1,6	Berichte aufrufen	1,2
Umzug durchführen	0,9	<b>Summe</b>	<b>107,1</b>

\* Vor der Performance-Optimierung

5

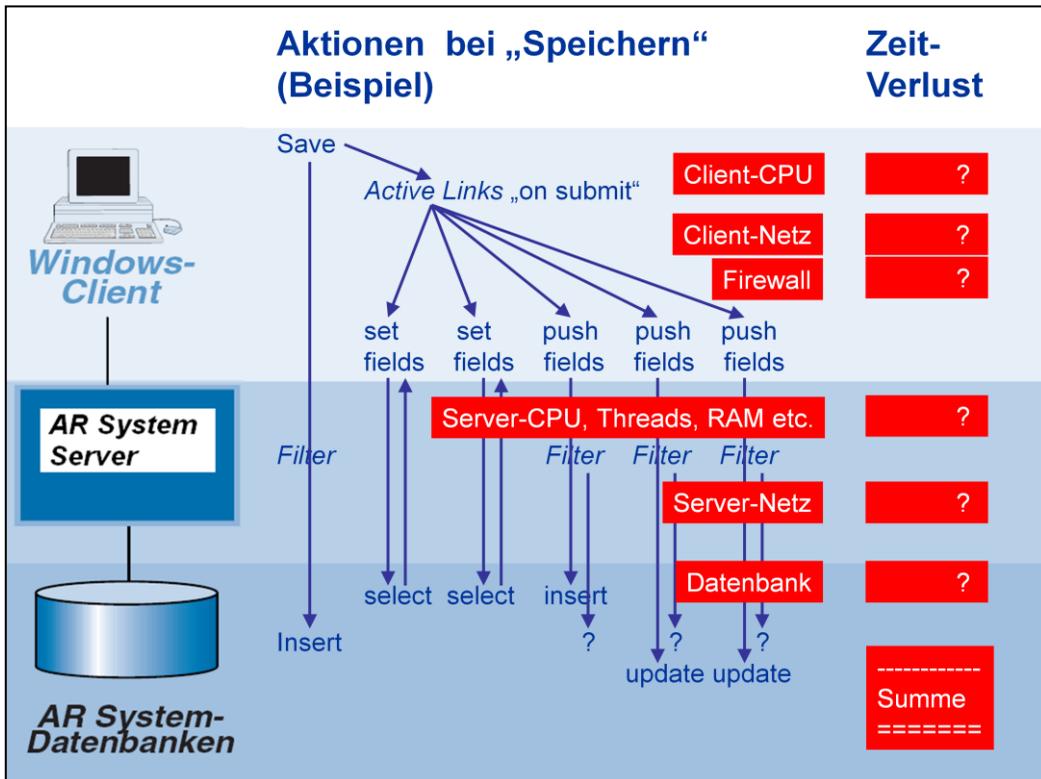
Scenario entspricht normalem Arbeitsvorgang der meisten Kunden

Der Einfachheit halber von Hand mit der Stoppuhr gemessen.

Durchschnittsdauern aus jeweils mehreren Messungen (mindestens jeweils 3 Messungen). Reproduzierbarkeit prüfen!

### Cache-Problematik ! (User Tool)

... woher kommen die langen Antwortzeiten?



<Architektur erklären>

Einführung für Nichtexperten – Wie funktioniert das Action Request System?

Sowohl im Client wird auf Vorgänge reagiert (Active Links) als auch im Server (Filter).

Willkürliches Beispiel: Submit-Vorgang eines Trouble-Tickets. Set-Fields:

- Nachschauen ob Telefonnr und Adresse schon vorhanden und mit Benutzer-Eingaben übereinstimmen (Anfrage von UT an Server, dann Anfrage von Server an Datenbank,
- dann Antwort der Datenbank an den Server, dann Antwort des Servers ans UT, dass beides vorhanden, aber falsch),
- Speichern eines Statistik- oder History-Datensatzes (Filter!). (Insert)
- Submit (Insert) des neuen Trouble Tickets (Filter!).
- Update von Adresse (Filter!)
- Update von Telefonnummer (Filter!)

Jede Aktion (ob Öffnen eines Menüs, Wechsel angezeigter Registerkarten, Öffnen von Dialogform...) ist verschieden.

< Vorteil von Filtern >

Bei langen Antwortzeiten:

- Zunächst nur die „Summe“ bekannt

→ Fischen im Trüben (Viele Parameter)

Prinzipiell 2 Möglichkeiten der Ursachenfindung:



## Übersicht

### I Einführung

Performance-Messungen

Ursachen schlechter Performance

### II Analyse

**systematische Versuche,**

**Log-Datei-Analyse**

### III Tuning

Datenbank

Server

Workflow

### IV Zusammenfassung

### V Diskussion



## II. Performance-Analyse: Vorgehen

### a) Systematische Versuche, z.B.

- i. Variation der Client-CPU
- ii. Variation der Client-Netzbandbreite
- iii. Variation der Anzahl der AR Server-Threads

### b) Log-Datei-Auswertung

8

Teil II des Vortrags: die **Analyse**

a) Systematisches Experimentieren

→ zeigen, dass dies ein **nützliches und wirkungsvolles Mittel** ist.

b) Log-Dateien (mit Zeitstempeln für jede Unteraktion)

- Tools dazu

Aktion	CPU mit 0,9 GHz		Verhältnis
	Dauer (s)	Dauer (s)	
Wechsel Bearbeitungsmodus	8,4	3,4	<b>41%</b>
Erweiterte Suche abbrechen	2,2	0,8	<b>37%</b>
Speichern im Bearbeitungsmodus	8,7	3,0	<b>35%</b>
Fokus setzen im Bearbeitungsmodus	5,5	1,8	<b>32%</b>
Umzug abbrechen	3,8	1,2	<b>32%</b>
Umzugsdialog schließen	3,4	1,0	<b>28%</b>
Aktivierung aufrufen	6,4	2,7	<b>42%</b>
Aktivierungsdialog schließen	3,5	0,9	<b>25%</b>
Austausch abbrechen	3,7	1,4	<b>38%</b>
Austausch schließen	3,7	1,1	<b>29%</b>
Ersterfassung abbrechen	3,0	0,8	<b>28%</b>

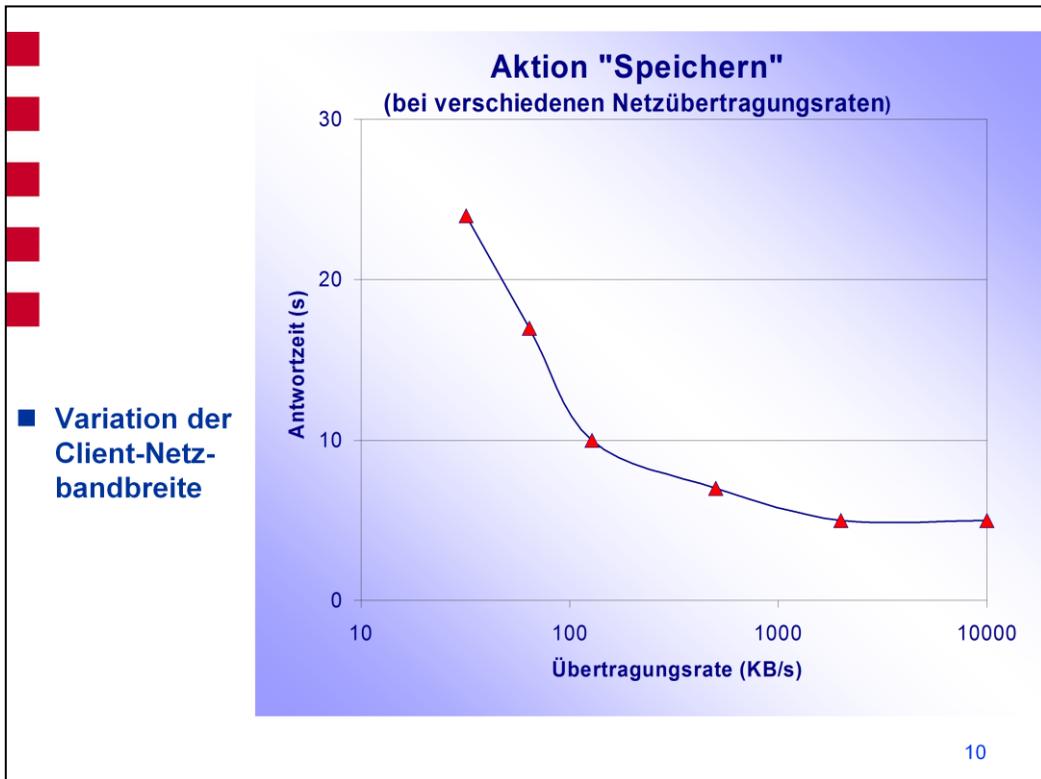
Ausgewählt

- Client-lastige Aktionen mit > 2s Dauer (900 MHz)

Anmerkung:

Rechnerisch:  $900 / 2800 = 32\%$

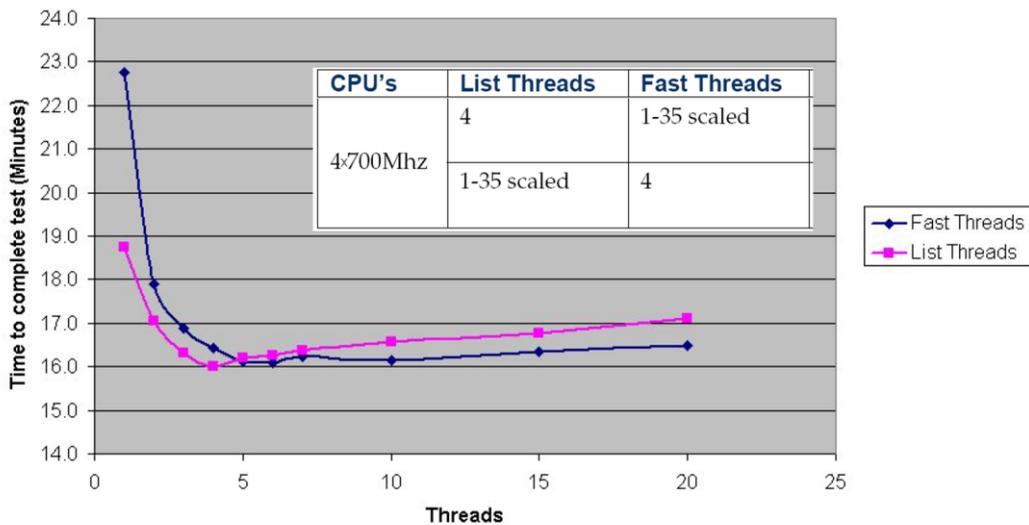
Beim 2,8 GHz-PC sind in der Architektur noch mehr Komponenten schneller (nicht nur die CPU)...



Ab 1024 kB/s in diesem Beispiel keine weitere Verbesserung...

## Variation der Zahl der Server-Threads

### 4 CPU - 400 Test Connections



Aus „Remedy White Paper: Optimizing AR System Performance with Server Thread Settings“

Entnommen aus...

<Server-Thread erklären> „Warteschlange für schnelle Anfragen/aufwändige Anforderungen“

Was wurde gemessen?

- Ein Szenario (ähnlich unserem Referenz-Szenario) wurde entworfen
- Das Szenario wurde automatisiert („Load generation tool“ / „Software Testing Package“), „internal tools“
- Dasselbe Szenario wurde für 400 Benutzer gleichzeitig durchgespielt.
- Die Gesamtdauer wurde gemessen → 20 Minuten

Zunächst: 4 List Threads fix – Messpunkte für verschiedene Anzahlen von Fast-Threads (blau)

Dann: 4 Fast Threads fix – Messpunkte für verschiedene Anzahlen von List-Threads (violett).

→ ca. 7 list und 5 fast threads optimal bei diesem System

→ Eher „etwas zu viel“ als „etwas zu wenig“

→ **Derartige Experimente sind ein geeignetes Mittel**, um ein Optimum zu finden (manchmal das einzige Mittel).

Sie werden auch von Remedy selbst angewandt.

## b) Log-Datei - Auswertung

```
12:20:26.1995 */+GLE ARGetListEntry -- schema User
12:20:26.1999 */SELECT T9.C1,C8 FROM T9 WHERE (T9.C101 = 'rosenbauer') ORDER BY 1 ASC
12:20:26.2201 */COMMIT WORK
12:20:26.2211 */-GLE OK
12:20:26.2245 */+GE ARGetEntry -- schema User entryId 00000000005992
12:20:26.2249 */SELECT C8 FROM T9 WHERE C1 = '00000000005992'
12:20:26.2268 */COMMIT WORK
12:20:26.2275 */Start filter processing -- Operation - GET
        User - 000000000005992
12:20:26.2276 */      End of filter processing (phase 1)
12:20:26.2276 */      Restart of filter processing (phase 3)
12:20:26.2277 */Stop filter processing
12:20:26.2278 */-GE OK
```

- Server-Log von 30 Minuten: > 2 GB, Millionen von Zeilen
- Auswertung von Hand ist mühsam.

12

Log-Datei des Servers

(ähnlich auch Mid-Tier Log, jedoch ohne SQL und Filter)

Zu sehen:

- Bereich < 0,1 s Dauer, 1 einziger Thread
- 2 API-Aufrufe: GLE und GE (nach Ausführen einer Suche im UT)
- Ausführungszeiten haben 0,1 ms Genauigkeit

Wir können analysieren:

GLE

API-Aufruf: 21,6 ms, davon SQL: 20,2 ms (→ AR Server: 1,4 ms)

GE:

API-Aufruf: 3,3 ms, davon SQL: 1,9 ms (→ AR Server: 1,4 ms)

Manuelle Auswertung mühsam.

Bisher:

Erstellen von PERL-Skripten zur Log-Datei-Auswertung (ich habe eine ganze Reihe davon)

Seit 4 Monaten ist das jetzt viel einfacher und komfortabler geworden: ....



## Auswerte-Software (**NEU !**)

### ■ AR System Log Analyzer (arwklga)

- Version 2.11  
12. Juli 2005 (Log-Files AR Version 4.5.2 – 6.3.0)

### ■ Ergebnisse

- Ausführungszeiten von API-Aufrufen
- Ausführungszeiten von SQL-Aufrufen
- Leerlaufzeiten pro Server-Thread

### ■ Zweck

- Unterstützung beim “Performance Tuning”

13

Für die Interessierten hier noch kurz wie's geht....



## arwklga - Kurzanleitung

- Server-Log erzeugen (SQL, API, + z.B. Filter etc.)
- auf PC übertragen
- `arpreparelogs -o arapisql.log *.log`
- `arwklga -i -sMAX -w myanalysis arapisql.log`
- Datei

`myanalysis\index.html`

mit Web-Browser öffnen

14

Filter/Eskalationen mitloggen – sonst Identifizierung der Stellen im Workflow später schwierig.

# arwklga – Output (HTML-Version)

## AR System Log Analysis

generated Wed Sep 21 18:20:35 2005 using arwklga v2.11

General		20 LONGEST RUNNING INDIVIDUAL API CALLS		
EXECUTION TIME	LINE No.	API	FORM	START TIME
1.2711	<a href="#">3324</a>	GLEWF CFG_CI_ConfigItem		Wed Sep 21 2005 17:08:17.8980
1.2519	<a href="#">3882</a>	GLEWF CFG_CI_ConfigItem		Wed Sep 21 2005 17:08:55.7131
1.2382	<a href="#">1404</a>	GLEWF CFG_HIS_ConfigItemHistory		Wed Sep 21 2005 17:08:08.9388
1.2344	<a href="#">2097</a>	GLEWF CFG_HIS_ConfigItemHistory		Wed Sep 21 2005 17:08:10.8194
1.2334	<a href="#">1184</a>	GLEWF CFG_HIS_ConfigItemHistory		Wed Sep 21 2005 17:08:06.0002
1.2296	<a href="#">17247</a>	GLEWF CFG_HIS_ConfigItemHistory		Wed Sep 21 2005 17:22:30.7825
0.8911	<a href="#">3452</a>	GLEWF CFG_CI_ConfigItem		Wed Sep 21 2005 17:08:31.2178
0.8320	<a href="#">3190</a>	GLEWF CFG_CI_ConfigItem		Wed Sep 21 2005 17:08:16.3344
0.8280	<a href="#">2993</a>	EXP		Wed Sep 21 2005 17:08:15.3700
0.8226	<a href="#">3772</a>	GLEWF CFG_CI_ConfigItem		Wed Sep 21 2005 17:08:54.5532
0.5841	<a href="#">12810</a>	SE CFG_BZ_Beziehungen		Wed Sep 21 2005 17:22:27.1868
0.5786	<a href="#">1317</a>	GLEWF CFG_ANCI_CI_AnwenderConfigItemBeziehung_ConfigItem-Join		Wed Sep 21 2005 17:08:07.3592
0.5508	<a href="#">14684</a>	SE CFG_BZ_Beziehungen		Wed Sep 21 2005 17:22:28.7184
0.5452	<a href="#">13747</a>	SE CFG_BZ_Beziehungen		Wed Sep 21 2005 17:22:27.9649
0.5281	<a href="#">15621</a>	SE CFG_BZ_Beziehungen		Wed Sep 21 2005 17:22:29.4649
0.5071	<a href="#">3930</a>	EXP		Wed Sep 21 2005 17:08:14.4251

15

Sehr übersichtliche Darstellung,  
Hier gewählt: „API Aggregates, Top N“

...Genauso geht z.B:  
...„SQL-Aggregates, Top N“...





## Übersicht

### I Einführung

Performance-Messungen  
Ursachen schlechter Performance

### II Analyse

systematische Versuche,  
Log-Datei-Analyse

### **III Tuning**

**Datenbank**

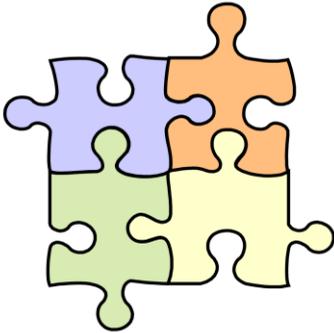
**Server**

**Workflow**

### IV Zusammenfassung

### V Diskussion

### III. Performance-Tuning



- **Tuning ist Prozess, nicht „einmalige Maßnahme“**
- **Teamarbeit:**
  - Datenbank-Tuning-Experten
  - Unix-Experten (mit Analyse-Tools)
  - Evtl. Netzwerk-Experten
  - AR System Experten
  - Anwender / Kunden

18

Je nach Ergebnis der Analyse:

Start z.B. bei der Datenbank



## Oracle Tuning-Maßnahmen



- Setzen von Indices
- Cachen einzelner Tabellen
- Datenbank-Reorganisation
- Shared Pool – Verkleinerung
- Tabellen-Partitionierung

Setzen von Indices...

19

häufig die einfachste und wirkungsvollste Maßnahme: **Index setzen**

## CACHE-Attribut

- SGA = Gesamter Reservierter Hauptspeicher von Oracle (System Global Area)
- Buffer Cache, mit Ordnung:
  - Anfang: MRU („most recently used“) und
  - Ende: LRU („least recently used“)



### Anwendung: CACHE - Attribut

Bei wichtigen, nicht zu großen Tabellen setzen!

... zwingt wichtige Tabellen permanent in den Cache

→ evtl. großer Performance-Gewinn

Gelesene **Oracle-Blöcke** werden im **Buffer Cache** gecached.

Über Index eingelesene Blöcke kommen an das MRU-Ende,  
über „Full Table Scans“ gelesene kommen gleich an das LRU-Ende

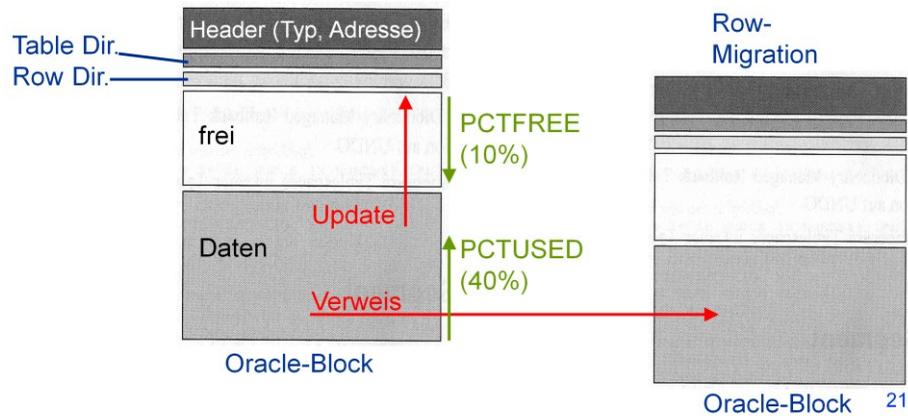
Nicht wiederbenutzte Blöcke wandern vom MRU- in Richtung des LRU-Endes und fallen schließlich wieder aus dem Cache heraus.

**Manuelle Beeinflussung der Oracle-CACHE-Verwaltung: CACHE-Attribut.**

# Reorganisation der Datenbank

2 Ziele:

a) Beseitigen von verketteten Datensätzen („chained rows“)



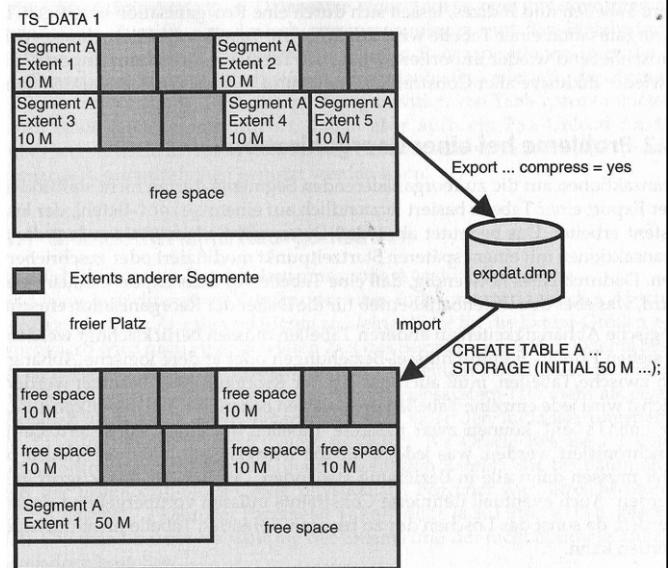
<Block erklären> (Segment=Tabelle, Extents, bestehen aus Blöcken)

Auffüllen eines Blocks mit Inserts (Submit)

PCTFREE-Parameter pro Tabelle setzbar

ardb.conf

b)  
Zusammenlegen  
von Extents



Anmerkung: „Segment“ steht z.B. für „Tabelle“. 22

Zur Reorganisation:

Export (Dump) notwendig.

Anschließender Reimport mit neuen „Insert“-Vorgängen (Auflösung verketteter Blöcke).

Ähnlich wie Disk-Defragmentieren in Windows

# Partitionierung von Tabellen

- Partition-Key
- Verteilung von Tabellenzeilen auf Partitionen
- AR System „sieht“ normale Tabelle
  - weiß nichts von deren Partitionierung
- Abfragen
  - lesen nur spezifizierte Partition
  - „partition elimination“ / „partition pruning“
- Skalierbarkeit der Anwendung:
  - Full Table Scan → Full Partition Scan



Große Tabellen partitionieren z.B. nach

- Status (Incident Management) oder
- Mandant/Kunde (Bestandsführung)

Vorsicht bei Join-Bedingungen mit partition keys!

Bei Partitionierung nach „Kunden“ bleibt die Anwendung skalierbar  
→ Neue Kunden mit „nochmal 150000 Datensätzen“ verlängern die Suchzeiten für Altkunden nicht.

## Vorsicht bei Join-Forms!

Join-Abfragen über mehrere indizierte Felder gehen im Optimalfall zunächst nach dem Index, der die Treffermenge am stärksten einschränkt.

Bei Verwendung eines Partition-Keys in Abfrage und Join-Bedingung entscheidet der Optimizer evtl. zugunsten des Partition-Keys als erstem Index, obwohl dieser die Datensätze evtl. nur auf 150000 „einschränkt“, und joined danach mit der zweiten Tabelle

→ Aus 0,3 s Suchzeit wurden bei uns 12 Minuten Suchzeit !!!

# SQL-Ausführung

## ■ Phasen

Query-Analyse (Parsing)  
Berechnung des  
Ausführungsplans  
Ausführung

## ■ Ausführungsplan

Erstellung durch „Optimizer“  
Speicherung (mit SQL) im  
**Shared Pool**  
Wiedererkennung nur bei  
exakt identischem SQL  
Sonst Voraussetzung zur  
Wiedererkennung: Bind-  
Variablen.

- SQL des ARS benutzt keine Bind-Variablen
  - **Shared Pool** wird für ARS-Anwendung fast immer ohne Ergebnis durchsucht ☹.
- *Shared Pool **klein** machen,  
um nutzlose Suchen abzukürzen*



24

Der **Shared Pool** cached SQL-Anweisungen mit Ausführungsplan des Optimizers.

„rule based“ (RBO)

Costbased (CBO)

## Erfolg der Datenbank-Maßnahmen (\*)

### a) Datenbankverkleinerung von 7,8 GB auf 5,9 GB

b)

Aktion	Vor Reorg (s)	Nach Reorg (s)	Verhältnis
Anmelden	18,8	18,7	99%
Erweiterte Suche aufrufen	4,4	2,5	56%
Umzugsdialog aufrufen	8,2	6,0	74%
Aktivierung aufrufen	5,6	4,3	77%
Austauschdialog aufrufen	9,7	5,3	54%
Austausch durchführen	2,6	2,0	77%
Ersterfassung aufrufen	5,7	4,0	69%
<b>Gesamt</b>	<b>54,9</b>	<b>42,7</b>	<b>78%</b>

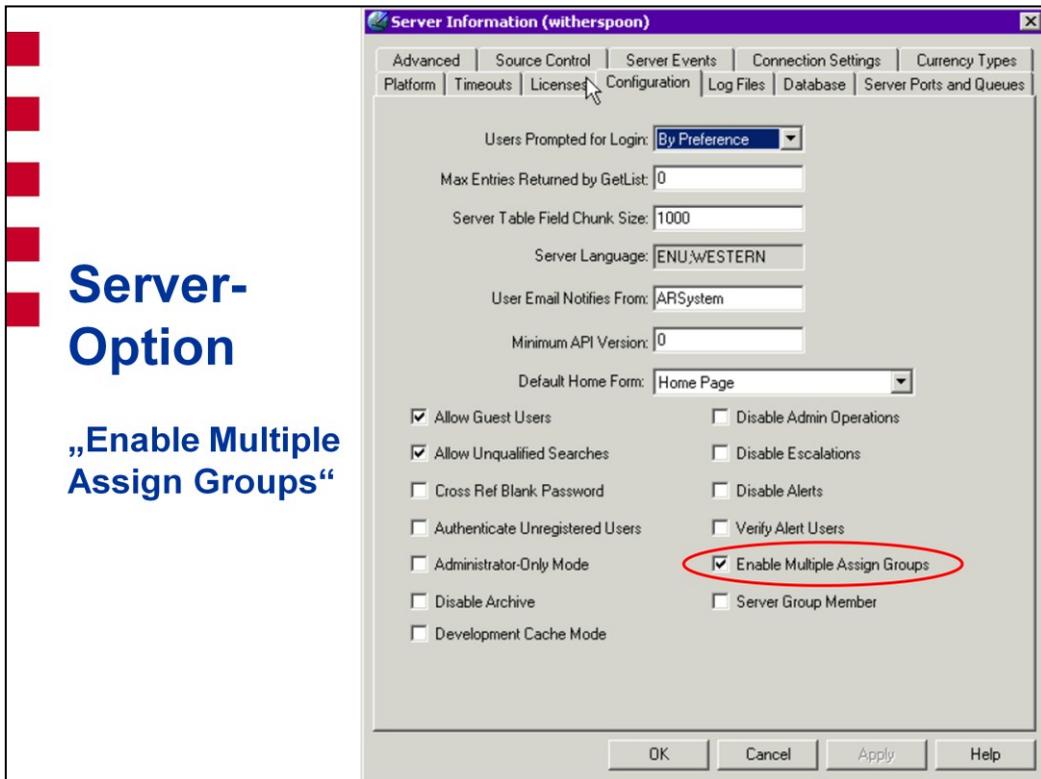
- \*) Gewertet wurden alle Referenz-Szenario-Aktionen, die
- länger als 1,7 s dauerten und
  - *keine* starke Client-CPU-Abhängigkeit haben  
(Verhältnis „t (2,8 GHz) / t (0,9 GHz)“ > 66%)

25



## **AR Server und Workflow Tuning**

- **Multiple Assign Groups**
- **Localize Server - Option**
- **Workflow-Anpassungen**



Bei Zugriffsrecht-Zuteilung auf Datensatz-Ebene („jeder Kunde sieht nur „seine“ Datensätze“):

1 Datensatz kann bequem auch für mehrere Kunden (bzw. Gruppen oder Rollen) sichtbar gemacht werden

```

SELECT C2,C3,C4,C5,C6,C7,C8,0,C112,C800000001,C800000002,C1
FROM T47
WHERE C1 = '0000000000000001'
AND
(T47.C112 IN (';8001;',';8000;',';-8001;',';0;',
              ';''Benutzer1'';'))

```

**...wird mit Option „Enable Multiple Assign Groups“ zu**

```

SELECT C2,C3,C4,C5,C6,C7,C8,0,C112,C800000001,C800000002,C1
FROM T47
WHERE C1 = '0000000000000001'
AND
((T47.C112 LIKE '%''Benutzer1'';%') OR ((T47.C112 LIKE '%;0;%')
OR ((T47.C112 LIKE '%;-8001;%') OR ((T47.C112 LIKE '%;8000;%')
OR (T47.C112 LIKE '%;8001;%')))))
)

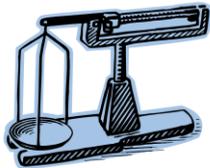
```

**→ Benutzung eines Index damit in vielen Fällen unmöglich !**

- **Ebenso bei „Dynamic Groups“ (60000-60999)**

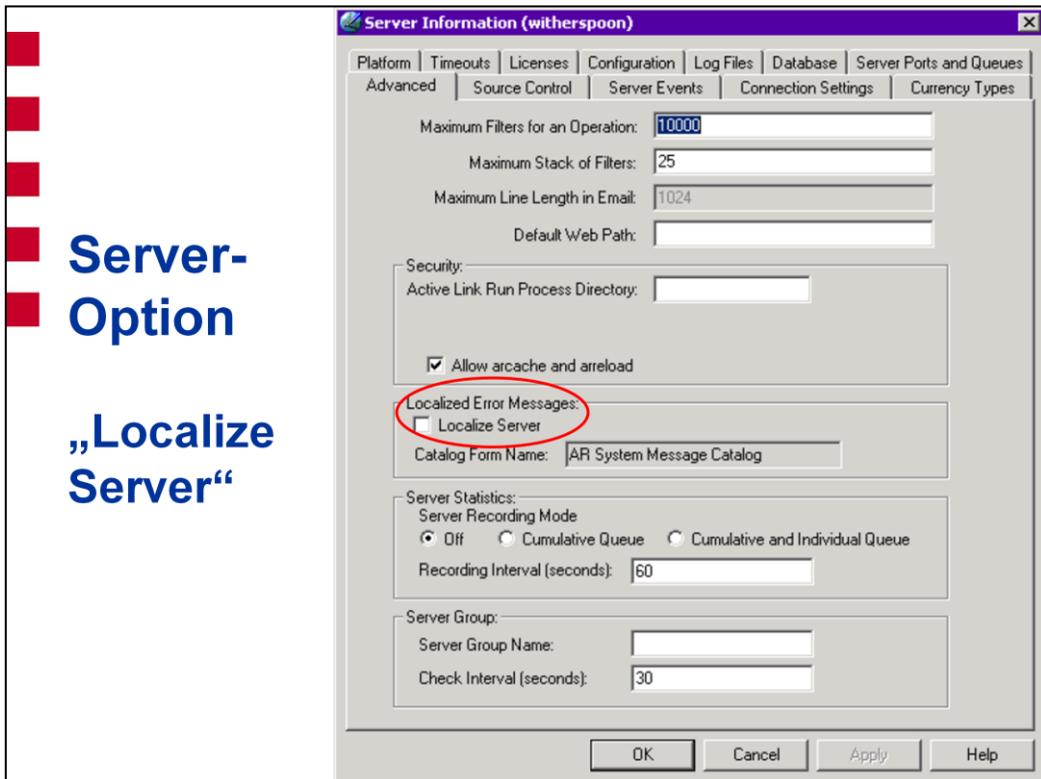
Immer prüfen, ob diese Option verzichtbar ist, z.B.

- bei Altanwendungen  
(geschrieben unter AR System Versionen 4 oder 3)
- bei neuen Anwendungen, durch Gebrauch  
mehrerer „Dynamic Groups“ mit je 1 Eintrag anstelle  
von 1 „Assignee Group“ mit vielen Einträgen



29

Ansonsten bleibt nur die Tabellen-Partitionierung.



Ermöglicht es,

- AR System Fehlermeldungen („no server available“) zu verändern/ zu übersetzen
- Eigene Meldungen („Messages“) in mehreren Sprachen zu hinterlegen

```
12:20:25.1823 *+WFC HRValidateFormCache -- form ROS_TestLoadingIorkfIouElements
12:20:25.1940 *+SELECT charMenuId,name FROM char_menu ORDER BY 1 ASC
12:20:25.2025 *+COMMIT WORK
12:20:25.2081 *+WFC OK
12:20:25.2227 *+EXP AREport -- as user rosenbauer
12:20:25.3336 *+SELECT fieldId,helpText FROM field WHERE schenaId=2605
12:20:25.3413 *+COMMIT WORK
12:20:25.3458 *+SELECT actLinkId FROM actLink_mapping WHERE schenaId=2605 ORDER BY 1 ASC
12:20:25.3475 *+SELECT actLinkId,name,timestamptmp,executemask,controlfieldId,fieldId,enable,numfncs,numfncs,queryShort,queryLong FROM actLink WHERE actLinkId=55834 ORDER BY 1 DESC
12:20:25.3492 *+SELECT actLinkId,name,timestamptmp,executemask,controlfieldId,fieldId,enable,numfncs,numfncs,queryShort,queryLong FROM actLink WHERE actLinkId=55835 ORDER BY 1 DESC
12:20:25.3508 *+SELECT a.actLinkId,actionIndex,macroName,shortText,longText FROM actLink_macro a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.3529 *+SELECT a.actLinkId,actionIndex,a.fieldId,assignShort,assignLong,keywordList,parameterList FROM actLink_set a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC
12:20:25.3959 *+SELECT a.actLinkId,actionIndex,command,keywordList,parameterList FROM actLink_process a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.4057 *+SELECT a.actLinkId,actionIndex,msgType,msgLn,msgText,msgPane FROM actLink_message a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.4070 *+SELECT a.actLinkId,actionIndex,a.fieldId,charMenu,propShort,propLong,focus,accessOpt FROM actLink_set_char a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC
12:20:25.4551 *+SELECT a.actLinkId,actionIndex,serviceName,topic,action,path,command,item FROM actLink_dde a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.4569 *+SELECT a.actLinkId,actionIndex,autoServerName,clsId,isVisible,actionShort,actionLong,CORShort,CORLong FROM actLink_auto a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 OR
12:20:25.4588 *+SELECT a.actLinkId,actionIndex,a.fieldId,assignShort,assignLong FROM actLink_push a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.4689 *+SELECT a.actLinkId,actionIndex,assignShort,assignLong,keywordList,parameterList FROM actLink_sql a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.4910 *+SELECT a.actLinkId,actionIndex,serverName,schenaName,vulLabel,closeBox,assignShort,assignLong,windowMode,nofatchCtrl,pollIntVal,queryShort,queryLong,msgType,msgLn,msgText,msgPane,reportItt
12:20:25.5011 *+SELECT a.actLinkId,actionIndex,closeAll FROM actLink_close a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.5048 *+SELECT a.actLinkId,actionIndex FROM actLink_commit a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.5076 *+SELECT a.actLinkId,actionIndex,serverName,guideName,guideMode,guideTableId FROM actLink_call a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.5176 *+SELECT a.actLinkId,actionIndex,closeAll FROM actLink_exit a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.5193 *+SELECT a.actLinkId,actionIndex,label FROM actLink_goto a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.5207 *+SELECT a.actLinkId,actionIndex,buttonTitle FROM actLink_wait a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.5219 *+SELECT a.actLinkId,actionIndex,tag,fieldId,value FROM actLink_gotoaction a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.5232 *+COMMIT WORK
12:20:25.5306 *+EXP OK
12:20:25.5418 *+GS HRGetSchema -- schena ROS_TestLoadingIorkfIouElements
12:20:25.5424 *+GS OK
12:20:25.5474 *+EXP AREport -- as user rosenbauer
12:20:25.7248 *+SELECT actLinkId FROM actLink_mapping WHERE schenaId=2605 ORDER BY 1 ASC
12:20:25.7276 *+SELECT actLinkId,name,timestamptmp,executemask,controlfieldId,fieldId,enable,numfncs,numfncs,queryShort,queryLong FROM actLink WHERE actLinkId=55834 ORDER BY 1 DESC
12:20:25.7293 *+SELECT actLinkId,name,timestamptmp,executemask,controlfieldId,fieldId,enable,numfncs,numfncs,queryShort,queryLong FROM actLink WHERE actLinkId=55835 ORDER BY 1 DESC
12:20:25.7316 *+SELECT a.actLinkId,actionIndex,macroName,shortText,longText FROM actLink_macro a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.7341 *+SELECT a.actLinkId,actionIndex,a.fieldId,assignShort,assignLong,keywordList,parameterList FROM actLink_set a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC
12:20:25.7741 *+SELECT a.actLinkId,actionIndex,command,keywordList,parameterList FROM actLink_process a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.7827 *+SELECT a.actLinkId,actionIndex,msgType,msgLn,msgText,msgPane FROM actLink_message a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.7916 *+SELECT a.actLinkId,actionIndex,a.fieldId,charMenu,propShort,propLong,focus,accessOpt FROM actLink_set_char a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC
12:20:25.8427 *+SELECT a.actLinkId,actionIndex,serviceName,topic,action,path,command,item FROM actLink_dde a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8453 *+SELECT a.actLinkId,actionIndex,autoServerName,clsId,isVisible,actionShort,actionLong,CORShort,CORLong FROM actLink_auto a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 OR
12:20:25.8476 *+SELECT a.actLinkId,actionIndex,a.fieldId,assignShort,assignLong FROM actLink_push a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8675 *+SELECT a.actLinkId,actionIndex,assignShort,assignLong,keywordList,parameterList FROM actLink_sql a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8691 *+SELECT a.actLinkId,actionIndex,serverName,schenaName,vulLabel,closeBox,assignShort,assignLong,windowMode,nofatchCtrl,pollIntVal,queryShort,queryLong,msgType,msgLn,msgText,msgPane,reportItt
12:20:25.8780 *+SELECT a.actLinkId,actionIndex,closeAll FROM actLink_close a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8868 *+SELECT a.actLinkId,actionIndex FROM actLink_commit a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8823 *+SELECT a.actLinkId,actionIndex,serverName,guideName,guideMode,guideTableId FROM actLink_call a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8894 *+SELECT a.actLinkId,actionIndex,closeAll FROM actLink_exit a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8910 *+SELECT a.actLinkId,actionIndex,label FROM actLink_goto a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8910 *+SELECT a.actLinkId,actionIndex,buttonTitle FROM actLink_wait a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8922 *+SELECT a.actLinkId,actionIndex,tag,fieldId,value FROM actLink_gotoaction a,actLink_mapping b WHERE a.actLinkId=b.actLinkId AND b.schenaId=2605 ORDER BY 1 DESC,2 DESC
12:20:25.8937 *+COMMIT WORK
12:20:26.0332 *+EXP OK
```

Demo-Beispiel: Einfache Form mit 2 Active Links mit jeweils 3 Aktionen

- Mit „Localize Server“: Alle Abfragen zweimal identisch
- Pro Active Link und pro Menü: ca. 1 zusätzliche SQL-Abfrage
- Caching im User Tool ebenfalls zweifach:
  - Einmal allgemein
  - Einmal lokalisiert
- Antwort von Remedy (Ticket-ID ISS01052967, noch offen):

Localized messages are downloaded to the local client as though they were another view (localized) of the form. This view definition then contains the active link, and its particular text message.  
 (...) performance impact, however (...) would be expected to be slight/unnoticeable



- Localize Server – Option: evtl. verzichtbar?

32

Nach Änderungen (update cache):

Bei 1000 Active Link- und Menüobjekte / Anwendung - ca. 1000 unnütze SQL-Abfragen pro Benutzer beim Öffnen einer Form

bei 300 Benutzern damit am nächsten Morgen: 300.000 überflüssige Abfragen zwischen 7.30 Uhr und 9:00 Uhr

→mehr als 50 zusätzliche Abfragen/s an Datenbank.

(alle 18 Sekunden ein neuer Benutzer)



# Workflow-Programmierung

## ■ Verbesserungspotenzial

- Zu große / zu viele Hintergrundbilder / Icons
- Häufige Aktualisierung von Tabellen, sogar von unsichtbaren
- Fehlende Indices / kombinierte Indices
- Set-Fields-Aktionen in Filtern über „\$PROCESS\$ - Aufrufe („Blocking Actions“)
- Überflüssige Temp-Felder, Deaktivierte Active Links
- Vergessene „`\$“ im Namen von Active Links mit Table Loop Guide - Aufrufen
- Push-Field-Aktionen mit Bedingung „1 = 0“ entfernen
- (\$TIMESTAMP\$ - 'Create Date') > 3600 umschreiben
- Feldlängen < 4000 Zeichen verwenden
- Erlauben von „unqualifizierten Suchen“ (Server-Option)



## Änderungen an realem Workflow und GUI

- **Größtes Potenzial zur Performance-Steigerung**
- **Absprache mit Kunden / Benutzern**
  - „Funktionalität contra Performance“

34

z.B. Tabellen-Refresh nur nach Klick auf Tabelle, nicht bei Öffnen des Fensters



## Übersicht

### I Einführung

- Performance-Messungen
- Ursachen schlechter Performance

### II Analyse

- systematische Versuche,
- Log-Datei-Analyse

### III Tuning

- Datenbank
- Server
- Workflow

### **IV Zusammenfassung**

### V Diskussion

## VI Zusammenfassung

### ■ Ursachen für schlechte Performance

- Ungünstige Workflow-Anforderungen
- Client-CPU, Mid-Tier-Server, Netz, Firewall
- AR System Server, Server-Netz, Datenbank

### ■ Tuning

- Ursachenanalyse
- Maßnahmen, z.B. Datenbank  
(Indices, CACHE, Shared Pool, Reorg, Partitioning)

### ■ Performance / Skalierbarkeit

- Von Anfang an im Workflowdesign / im Datenmodell berücksichtigen

### ■ *Performance nicht um jeden Preis !*

36

Abwägen zwischen Funktionalität und Performance !

## V Diskussion / Fragen ?

Boris Noack,  
Uli Rompel

B. Noack@hzd.hessen.de  
Tel: 0611 340 1609  
<http://www.hzd.de>

Dr. Martin Rosenbauer

Martin.Rosenbauer@t-online.de  
Tel: 06222 93 99 51  
<http://remedy.privat.t-online.de>

37

Abstürzende Server-Prozesse  
Blocking Actions – genaueres  
Mid-Tier – Ausführung von Active Links  
Flashboards  
Top und RAM-Angabe

Identifizierung von Anwendungsschritten, die besonders lange dauern,  
Analyse der zugehörigen Workflow-Vorgänge und gezielte Optimierung des  
Workflow-Codes für diese Aktionen

Identifizierung von ungünstigen Datenbank-Abfragen, sowohl mittels Analyse  
von Log-Dateien im ARS und von SQL-Logs in der Datenbank als auch durch  
Auswertung von Statistik-Informationen innerhalb der Oracle-Datenbank

Aufteilung in Table Spaces nach Anwendungen – schafft Übersicht – siehe  
auch ardb.conf

Regelmäßige Reorg

„Related Workflow“ unter 5.1.2 : Datenbanktabellen von Zeit zu Zeit  
„aufräumen“

```
SELECT T539.C1,T539.C1,T539.C1 FROM T539 WHERE (1 = 2) ORDER  
BY 3 ASC
```

Perl-Script zum Auffinden der Active Links