Telekurs Services

Best Practices für die Entwicklung von Applikationen mit Datenbanken aus der Sicht DBA/Operations

Zühlke AG, 9. Juni 2004

von Oli Sennhauser, Database-Engineer,

Telekurs Services Ltd.

Telekurs Services

Inhalt

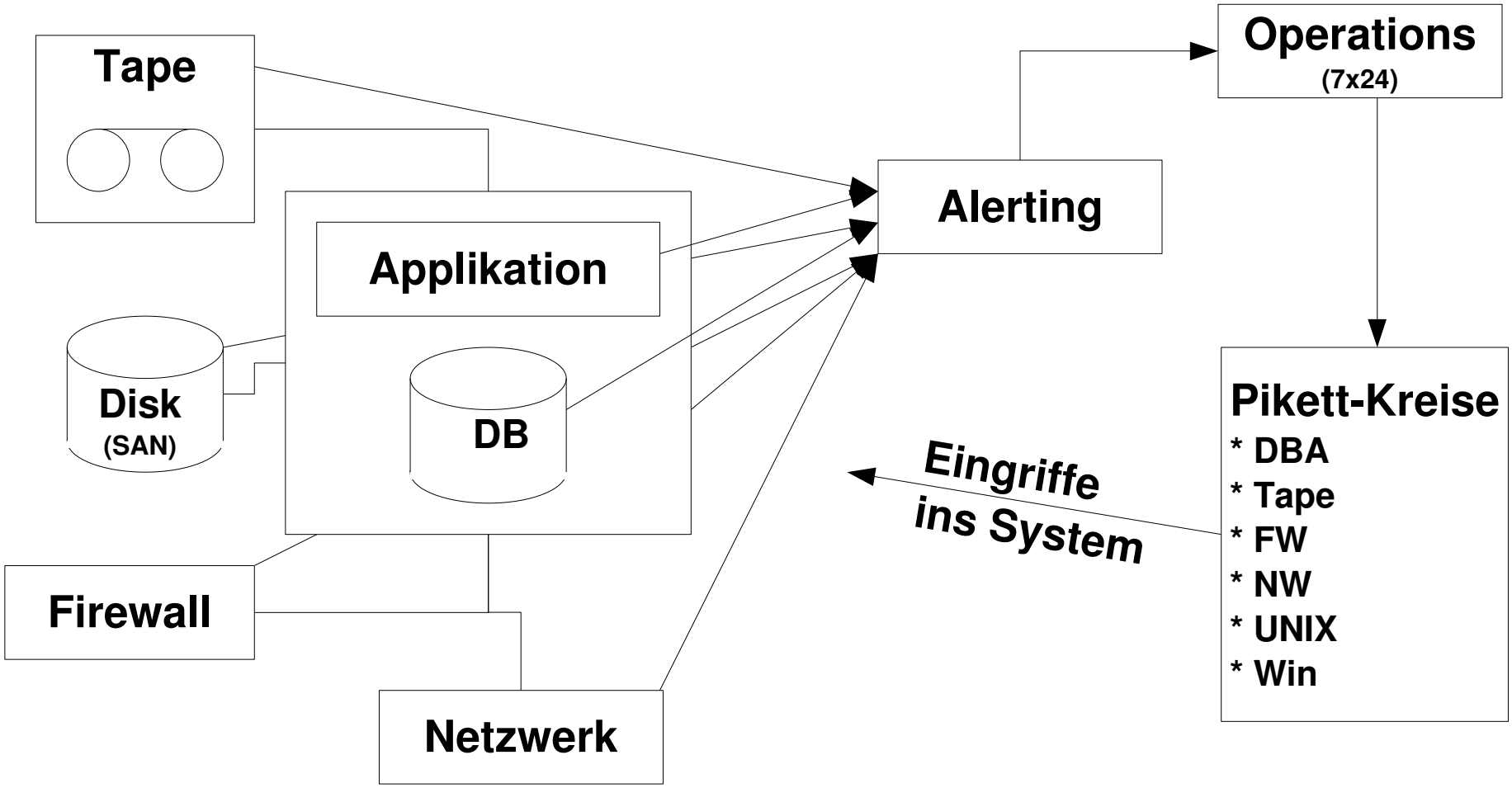
- Datenbanken in der Produktion
- Anforderungen an SW-Architektur
- Zusammenarbeit Entwicklung – DBA's

Datenbanken in der Produktion

- Organisation
- Aufgaben
- typische Tätigkeiten eines DBA
- Standards
- Stages

Telekurs Services

Organisation (2000 MA)



Aufgaben eines DBA

- Abhängig von Firmengrösse, Organisation, ...
- Grundsätzlich:
  - Betrieb der DB's (freier Platz, überwachen, ...)
  - Administration (DB's erstellen, umkonfigurieren, ...)
  - Engineering (neue Features testen, Betriebs- und Administrations-Abläufe organisieren und optimieren, Architekturen evaluieren, ...)
  - Support (Tuning-Support, allgemeine Fragen, ...)

Typische Tätigkeiten eines DBA

- Ein Tag im Leben des DBA Oli S. (Mi, 28. 4.)
  - 07:30 - Mail-Checks
  - 07:30 - SQL*Loader-Problem
  - 08:08 - SCC/check_ts/Partitionen
  - 08:20 - Kaputte CPU/Überwachung/Mail-Checks/Murphy's-Law
  - 09:30 - Server-Überlastung/Connect nicht möglich
  - 10:40 - Welche DB/DB-Übersicht
  - 11:30 - Ctrl-M Linux vs. Solaris etc.
  - 11:45 - Rollenkonzepte
  - 13:40 - Shared vs. Dedicated Server --> der DBA Dein Freund und Helfer
  - 15:50 - User-Account entsperren.

Standards I

- Warum Standards?
  - Vereinheitlichung und Vereinfachung
  - Beschleunigung von Abläufen
  - Sicherheit bei Problemen!!!
- aber Balance Standards vs Flexibilität!
- Standards einhalten, sonst: Nochmal machen!
- Bsp. Kosten eines zusätzlichen RDBMS-Produkts (mySQL)

Standards II

- Anpassen, Einhalten (den es gibt OFT :-) Gründe)
- falsche Kreativität:
  - Umgehung von Standards vs. Optimierung von Std. und Aufbau auf Std.
- Beispiele:
  - Oracle Baseline-Security
  - OFA
  - Codierungs-/Skripting-Richtlinien

Telekurs Services

Stages

Entwickler

Entwicklung
(D...)

Test
(T...)
- Modultest
- Integrationstest
- Massentest

Acceptance
(A...)

Produktion
(P...)

DBA's

Betrieb

Telekurs Services

Anforderungen an SW-Architektur

- Namenskonventionen
- OFA
- Logging
- Skripting Interfaces
- Rollenkonzepte
- Handling von Passwörtern

Telekurs Services

Namenskonventionen 1

- "globales" Firmen-Datenmodell?
  - KIVTR_Z (OLTP) ⇒ K_INH_VERT_ZUST (DWH)
- Namen entweder D oder E aber nicht mischen (D obsolet).
- Abkürzungen bringen nicht wirklich eine Zeitersparnis: KIVTR_Z
- Gefährliche Namen vermeiden: *.log ⇒ *.dbf (essentielle Files)

Telekurs Services

Namenskonventionen 2

- Konsistentes Namensverhalten:
  - Variablennamen

    Bsp. APPL_BASE > LogDirectory > ret > i
  - DB-Objektnamen nnn_T, nnn_I, nnn_V

    Bsp. T_PASS..., I_PASS... aber PASS...T, PASS_...I :-(
  - Einzahl/Mehrzahl (z.B. Tabellennamen)

    Bsp. dba_tablespaces, v$tablespace
  - Underscore “_”

Namenskonventionen 3

- generierbare Namen: z.B. Datafile-Name: <tablespace><nn><SID>.dbf
- Links “wichtigere” Information, rechts “weniger wichtige” Information.
  - 04_2004_PASS_DAT, PASS_2004_04_IND
- Es gibt keine Absolute Richtigkeit nur mehr oder weniger schlaue Konventionen.
- Wichtig: Einheitlichkeit, anpassen an Bestehendes

OFA I

Hintergrund: mehrere DB's mehrere Releases, Apache und ev. Middle-Tier:

- **Flexible Organisation von komplexen Softwarepaketen ⇒ Verminderung von Seiteneffekten**
- **Optimal Flexible Architecture (von Oracle)**
- **Trennung von Applikationen, Datenbank-, Configurations-Files und Binaries (Impact)**

Telekurs Services

OFA II

- Standardisierte Directory Struktur
- Vereinheitlichung der Umgebungen
- Zentralisierung und Automatisierung möglich
- Wartungspersonen sind einfacher austauschbar
- Bsp. Applikation trennen von DB Mountpoints

Telekurs Services

OFA III

- Grundsätzliche Directory-Struktur

```
${APPL_BASE}/
${APPL_BASE}/log
${APPL_BASE}/etc
${APPL_BASE}/bin
${APPL_BASE}/...
```

- Bsp. OFA-4

```
/u00/app/oracle/admin/<SID>
/u00/app/oracle/admin/<SID>/log
/u00/app/oracle/admin/<SID>/etc
/u00/app/oracle/local/dba/log
/u00/app/oracle/local/dba/etc
/u00/app/oracle/local/dba/bin
/u00/app/oracle/product/9.2.0/...
/u00/app/oracle/product/10.0.3/...
/u01/oradata/<SID>/...
/u02/oradata/<SID>/...
```

Logging I

Logging ist zentral für Problembehebung

- **Zentrales vs. dezentrales Logging**
- **Logging pro Applikation (OFA) vs. Logging alles an einem Ort (Praxis)**
- **Global-/Detail-Log Konzept vs. All-In-One**
- **Trace-Level (ein/aus, Level 1-...)**
- **⇒ Logging-Konzept, Operations als Systembenutzer in Analyse einbeziehen.**

Logging II

- JEDE Zeile mit Timestamp ⇒ Regelmässigkeiten sind erkennbar
- Einfache, klare wichtige Infos, “human readable” ausgeben.
- Eindeutige Marker verwenden (bck_inc0, bck_arc, bck_inc1c, bck_inc1d), uneindeutige vermeiden: OK, NOK; Err, ERR, Error, ERROR
- “Schlaue” Fehlermeldungen wie “Allgemeiner Fehler” sind zu vermeiden.

Telekurs Services

Logging III

– **Infos: Was habe ich wo gesucht, was habe ich gefunden und was habe ich erwartet.**

```
2004-04-14_21:47:39  PID:20605      DBLIAV3: bck_inc0:OK:
2004-04-14_22:02:22  PID:26770      DBLIHMI1: bck_inc0:OK:
2004-04-14_23:04:22  PID:23140      ABLIAV1: bck_inc0:OK:
2004-04-14_23:19:27  PID:26524      TBLIAV1: bck_inc0:ERR:
2004-04-14_23:22:41  PID:28364      SBLIAV2: bck_inc0:OK:
2004-04-14_23:35:23  PID:1029       TBLIAV2: bck_inc0:OK:
2004-04-14_23:53:06  PID:28920      TBLIAV3: bck_inc0:OK:

2004-04-14_23:53:06 BEGIN of add_datafile.ksh v 1.17
2004-04-14_23:53:07 WARNING: Configuration file
                          /u00/app/oracle/local/dba/etc/rman.conf
                          NOT found.
2004-04-14_23:53:07       Assuming target = /.
2004-04-14_23:53:06 END of add_datafile.ksh v 1.17 with returncode 0
```

Telekurs Services

Skripting Interfaces

- GUI's sind oft sexy! Aber...
- Interaktive Eingabe vs. Bulk-Load/Automatisierung

  Bsp: Drucker erfassen.
- Maintenance-/Cleanup-Jobs (Bsp: alle ... löschen, welche älter als ... sind)
- Analyse ⇒ Skripting Interface

Rollenkonzepte I

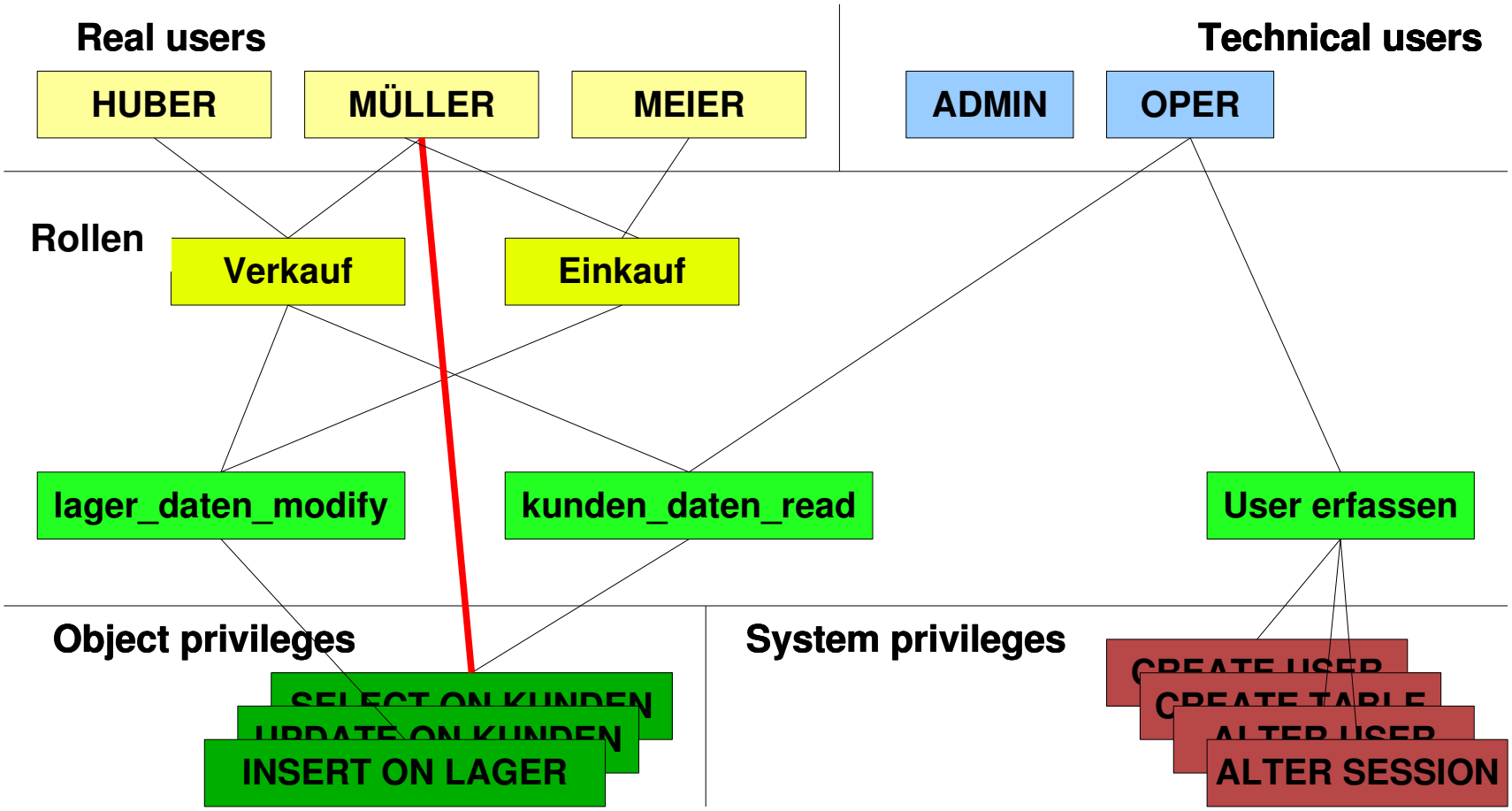
Sicherheit, Wartbarkeit

- Erstellen vor Beginn (Analyse, Ableitbar aus Usecase-Modell).
- Nachher ist es zu spät: Zeit, Lust, Druck von Management.
- Grund: warum kein Adminuser an Entwickler auch auf Dev/Test.
- Schema-Owner sperren, über Rollen an Applikations-User granten.
- Aber: Oracle hat Problem mit gewissen Privilegien :-(

Rollenkonzepte II

- User (technische User: pass_prod, k_read_only, reelle User: tksnv)
- Rollen (Businessrollen: verkauf, einkauf, mangement, ... technische Rollen: kundendaten_read, kundendaten_admin...)
- Privilegien (Systemprivilegien: create user, create table , ..., Objektprivilegien: select on kunden, update on lohn, etc.)

Rollenkonzepte III



Telekurs Services

Handling von Passwörtern I

- Persönliche User vs. Shared User (middle Tier) ⇒ Nachvollziehbarkeit, Identität
- Gute Passwörter Tpio4tZe! ⇒ ”This password is only for the Zühlke event!”
- Passwortfiles:

```
$ chmod 600 password.conf
-rw------- password.conf
```

Telekurs Services

Handling von Passwörtern II

- Achtung bei Tools:

```
$ sqlplus system/manager@TOLI1
...
$ ps -ef | grep sqlplus
oracle  3317  3316  0  18:14  sqlplus system/manager
```

```
$ sqlplus /nolog << _EOI
  connect ${username}/${password}@${ORACLE_SID}
  ...
_EOI
```

- Externally authenticated users (Server)
- Middle-Tier: REMOTE_OS_AUTH DO NOT!!!

Zusammenarbeit Entwickler – DBA's

⇒ DB kann Entwicklungsarbeit verringern

⇒ Entwickler kann Betrieb viel unnötige Arbeit verursachen

- Nutzen von DB-Features (Bsp. Partitionierung)
- Trendbrüche
- Performance-Tuning
- Vorgehen beim Debugging
- Vermeiden von Problemen in der Produktion

Nutzen von DB-Features I

- Produktunabhänges vs produktbezogenes Entwickeln (z.B. Oracle)

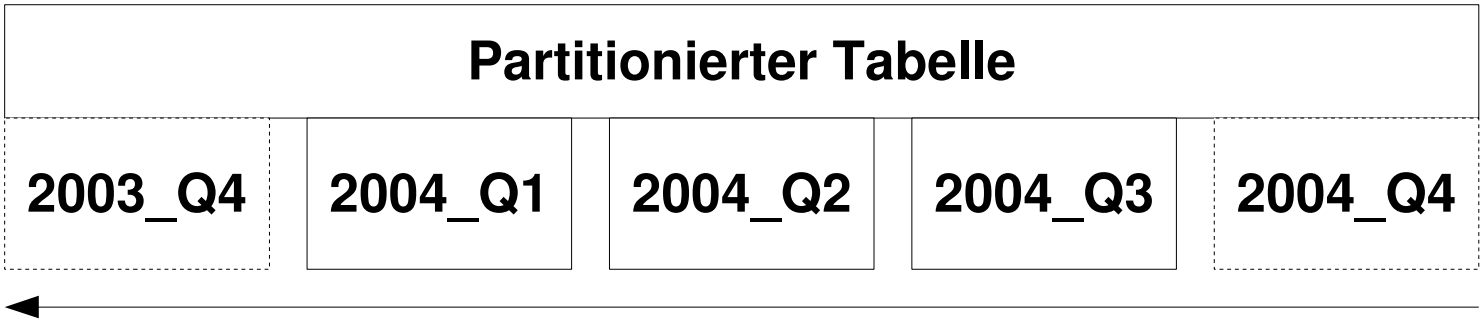
  ⇒ Features nutzen

- Partitionierung, Transportable Tablespaces, Materialized Views, Optimizer, Standby-DB's etc.
- ⇒ Setzt Euch mit Euren DBA's frühzeitig zusammen.

DB-Features (Bsp. Partitionen) II

- z.B. Kreditkarten-Transaktionen
- Jede Nacht Batch-Load
- 7-25 Monate Datenhaltung
- Cleanup-Jobs
- Sehr grosses Datenvolumen (100e von GB!)
- Schneller Zugriff auf Daten nötig
- ⇒ Partitionen

DB-Features (Bsp. Partitionen) III



- Was ist eine partitionierte Tabelle?
- Vorteile/Nutzen von Partitionen:
  - Historisierung oder Entfernung alter Daten (out-ageing)
- Gefahren:
  - Partitionierungsschlüssel für Zuordnung muss stimmen
  - Transaction muss geschlossen sein wenn abgehängt wird (Trx_closed_date != Unendlich!!! :-( )
- Nachteile: Komplexität im Betrieb

Trendbrüche I

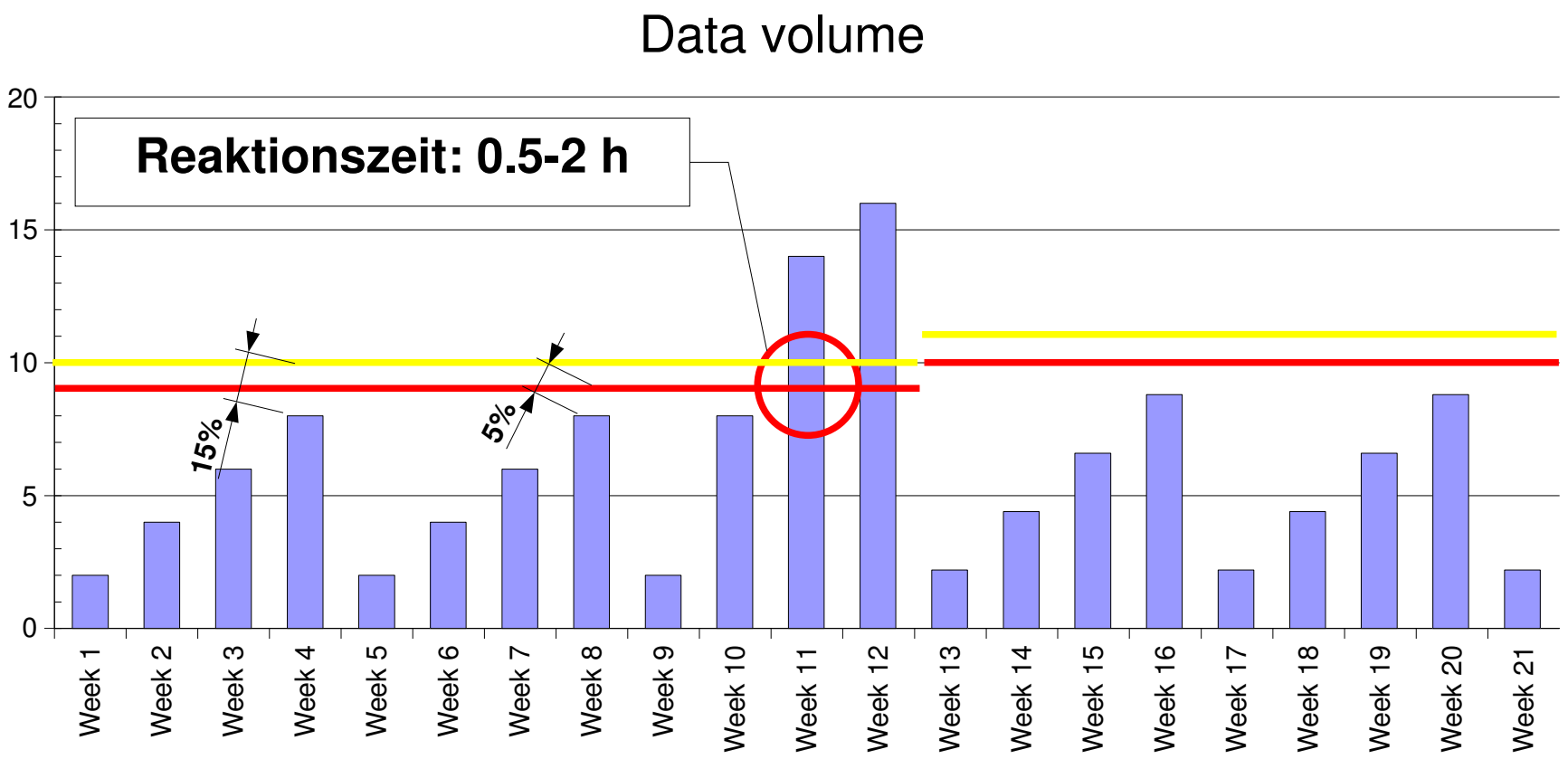
- Wachstum der Daten in der Regel linear (d/W/M/J).
- Faustregel für Wachstum: 1 Monat = 30 d ⇒ 3.3% pro Tag (partitionierte Tabelle) oder z.B. Durchschnittlich 250 MB/Tag.
- Kritische Phasen (Ende W/M/Q/Y).
- Auslegen/Bestellen von Diskplatz (knapp!)
- Einstellen der Grenzwerte für Überwachung der DB (15% free, oder Disk z.B. 3 GB free).

Trendbrüche II

Probleme:

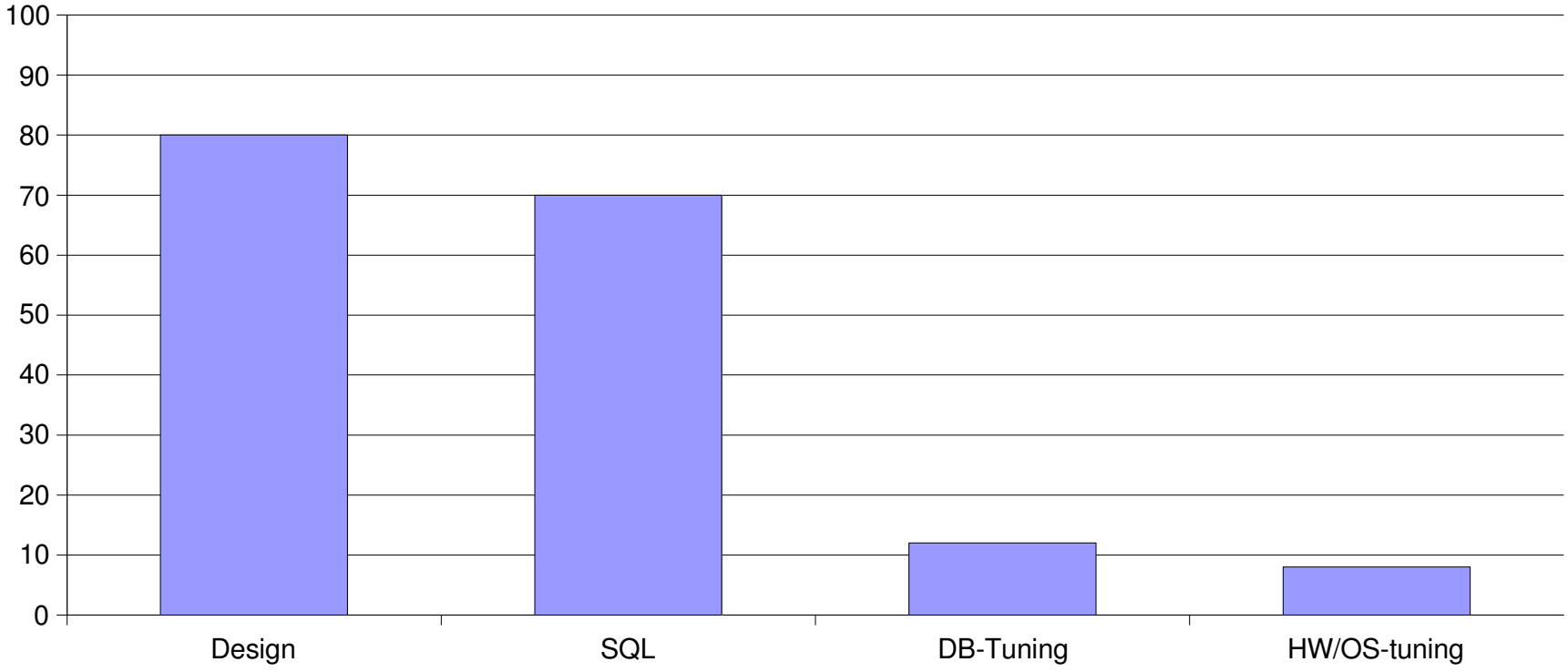
- **Schwankungen im Datenwachstum:**
  - **Wegfall eines Kunden**
    neue Kunden
- **Änderungen in der Applikation(slogik):**
  - **neu benötigte Daten**
  - **nicht mehr benötigte Daten**
  - **Beheben/Entfernen von Bugs, etc.)**
- **Analyse: Mengengerüst**
- **Änderung: Koordinieren mit DBA's**

Trendbrüche im Datenwachstum



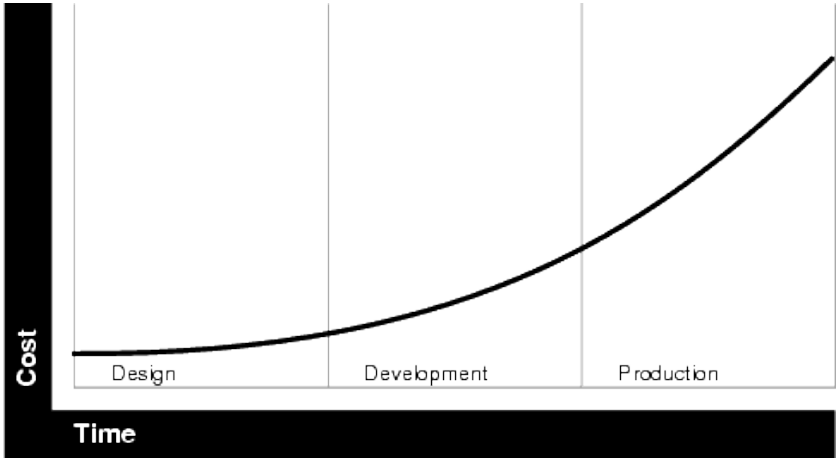
Perfomance-Tuning I

Relative Auswirkung auf Performance (Quelle KPN, NL):

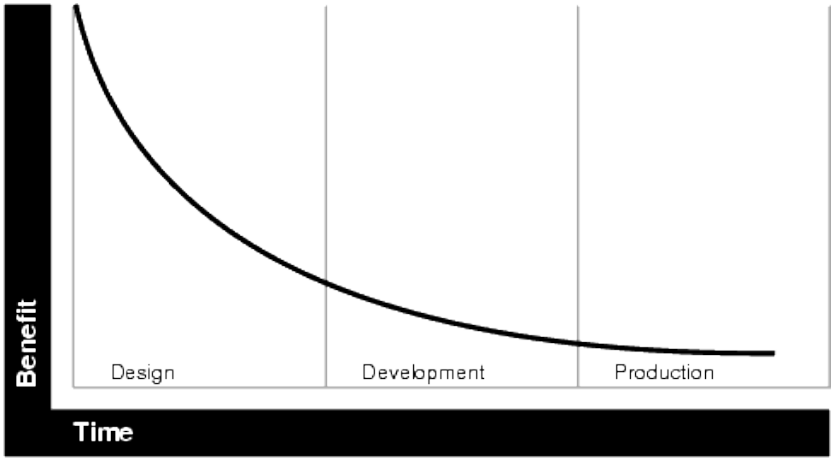


Perfomance-Tuning II

- googeln nach: server.817/a76992/ch2_meth.htm#9247
- Oracle Manual Designing and Tuning for Performance / Performance Tuning Methods



Cost of Tuning During the Life of an Application



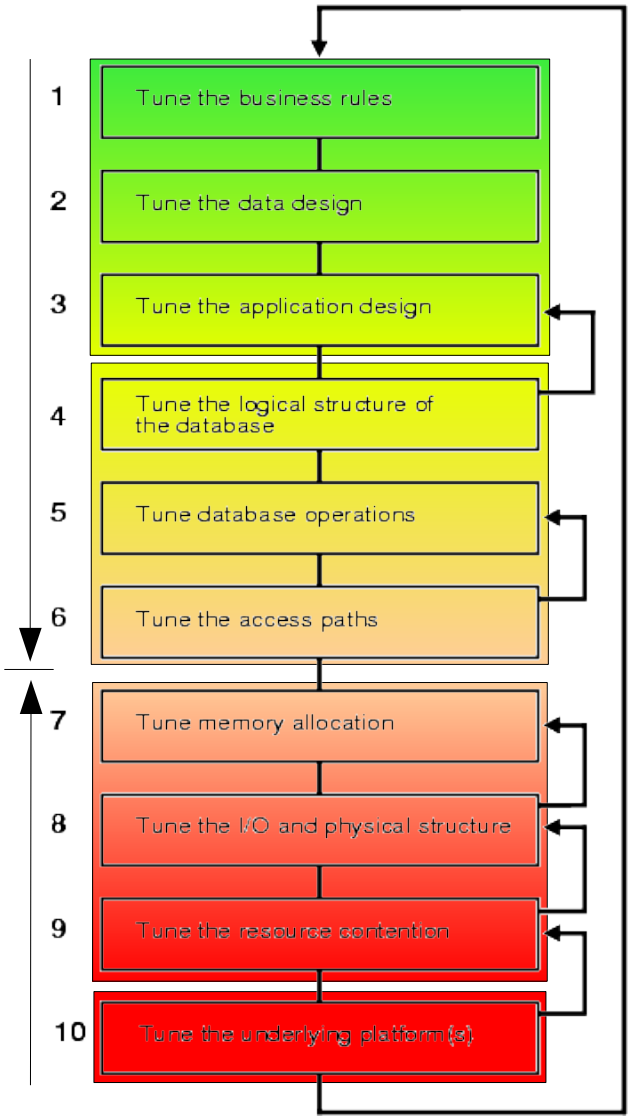
Benefit of Tuning During the Life of an Application

The Tuning Method

80-90%

- **Architekt/Designer**
  Business Rules, Data Design, Application Design
- **Applikationsentwickler**
  Logical Structure, DB-Operations, Access-Paths
- **DBA**
  Memory Allocation, I/O, physical Structure, Ressource Contention
- **SysAd**
  Tune the underlying platform(s)

10-20%



Tuning IV

- Step 1: Tune the Business Rules (realistic expectations for the number of concurrent users, the transaction response time, and the number of records stored).
- Step 2: Tune the Data Design (normalisation/denormalization, hot-spots, partitions)
- Step 3: Tune the Application Design (caching data)
- Step 4: Tune the Logical Structure of the Database (indexing over/underindexing)
- Step 5: Tune Database Operations (optimizer, pl/sql, sql-trace)
- Step 6: Tune the Access Paths (drop/create indexes, collect statistics)

Tuning V

- Step 7: Tune Memory Allocation (sizing and distribution of several different oracle memory types (sga), block buffers, sort area size, hash area size, etc.)
- Step 8: Tune I/O and Physical Structure (io-distribution, internal DB parameters)
- Step 9: Tune Resource Contention (block, shared pool, latch, lock)
- Step 10: Tune the Underlying Platform(s) (UNIX-Buffer, LVM, Memory, etc)

Tuning VI

- Post-Mortem-Analyse schwierig:
- ⇒ SQL-Trace (SQL-Statements)
- ⇒ Statspack (Systemstatistiken)
- SQL vs. PL/SQL (Proc/OO-Denken vs. mengenorientiertes Denken)
- Optimizer
- Explain Plan bevor DB still ststeht :-)

Telekurs Services

Tuning VII

```
EXPLAIN
SELECT cus_company_name, src_source
FROM customer, source
WHERE cus_src_id = src_id
  AND cus_id between 200 and 210;

                              QUERY PLAN
-----------------------------------------------------------------
  -
 Hash Join  (cost=5.78..7.00 rows=1 width=116)
   Hash Cond: ("outer".src_id = "inner".cus_src_id)
   ->  Seq Scan on source  (cost=0.00..1.14 rows=14 width=38)
   ->  Hash  (cost=5.78..5.78 rows=1 width=86)
         ->  Index Scan using customer_pkey on customer
               (cost=0.00..5.78 rows=1 width=86)
               Index Cond: ((cus_id >= 200) AND (cus_id <=
                            210))
```

Telekurs Services

Deadlock

- ORA-00060: Kommentar Oracle: Applikatorisches Problem
- Wie kommt es dazu:

```
BEGIN TRANSACTION
UPDATE KUNDEN
SET PLZ = '8610'
WHERE id = 4711;
...
UPDATE KUNDEN_MA
SET MITARBEITER = 'MEIER'
WHERE kunden_id = 4711;
```

```
BEGIN TRANSACTION
UPDATE KUNDEN_MA
SET ANZ_MA = 2
WHERE kunden_id = 4711;
...
UPDATE KUNDEN
SET STATUS = 'OK'
STATUS != 'OK;
```

Telekurs Services

Deadlock

- LOCK Table: DO NOT!

```
BEGIN TRANSACTION
LOCK TABLE KUNDEN;


...
UPDATE KUNDEN_MA
SET MITARBEITER = 'MEIER'
WHERE kunden_id = 4711;
...
```

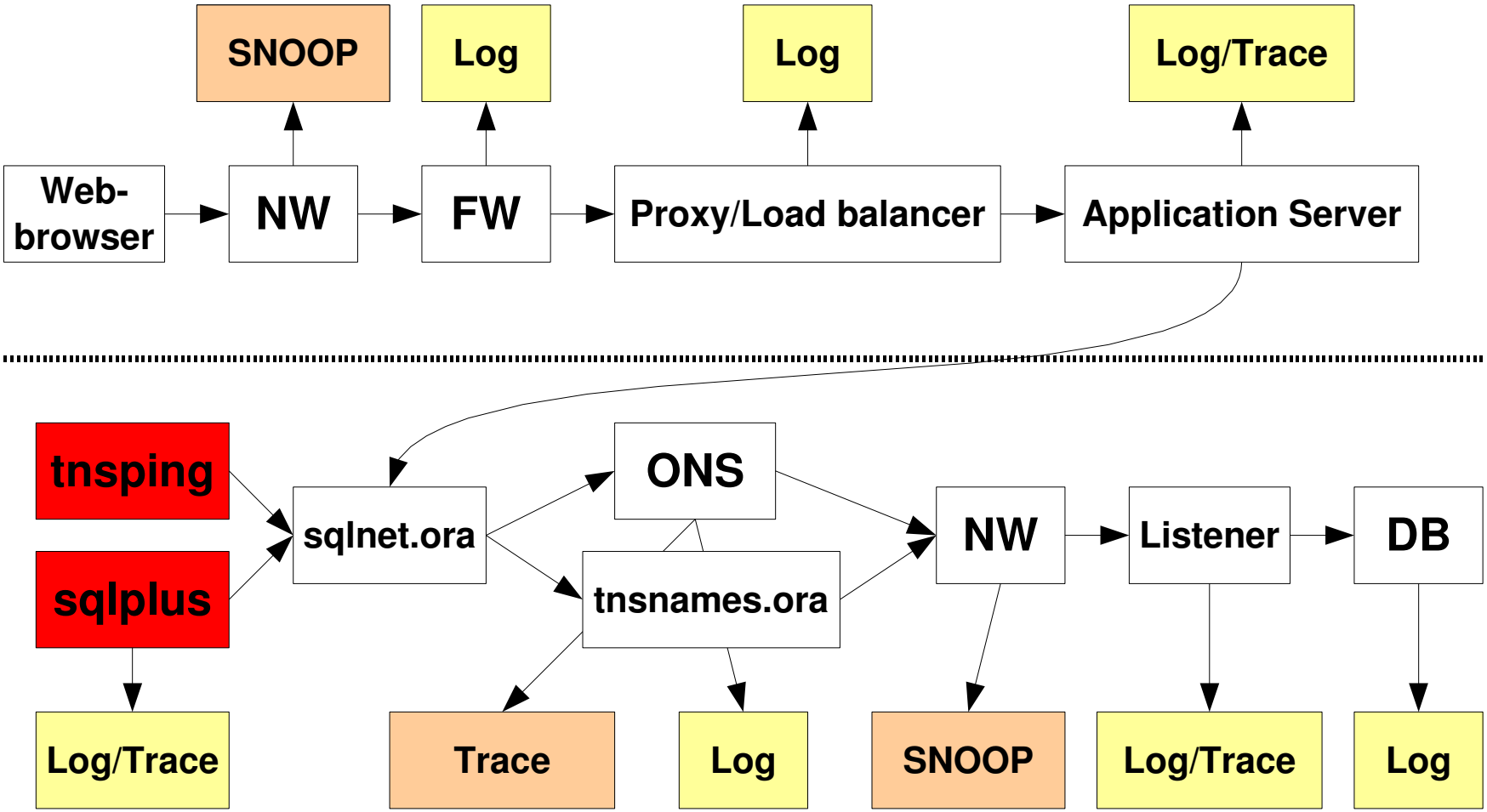
```
BEGIN TRANSACTION

LOCK TABLE KUNDEN_MA;
UPDATE KUNDEN
SET STATUS = 'OK'
STATUS != 'OK;
...
```

- Bsp: 8 CPU-Maschine: 1 CPU wartet 14h pro Tag nur auf die Auflösung von Locks!

Vorgehen beim Debugging I

- Normalfall: Telefon ⇒ DB läuft nicht mehr, DB rebooten? :-(
- Logs, Trace-Mode ⇒ Fehlermeldungen und Symptome, was wurde gecheckt?
- Postmortem Analyse schwierig bis unmöglich!
- Bei welchem Layer happerts?

Vorgehen beim Debugging II



Problemvermeidung in der Produktion

- Sauberes Testen/Stages einhalten
- Keine Entwickler auf der Produktion :-)
- Absprechen, Mitdenken, Zusammenarbeiten (Changes, Wartungsfenster), etc.
- Entkoppeln (z.B. Applikation und DB)

Finally

- Wie Entwickler gibt es auch mässig begabte DBA's :-(
- IT kann ziehmlich komplex sein. Also: helfen wir einander...
- Gedankenaustausch, Kontakt, voneinander lernen...