

---

# Redo Logs

Informationen soweit der Logminer reicht

Thomas Klughardt  
Senior Systems Consultant

---



# Dell Software



## Data center & cloud management

- + Client management
- + Performance management
- + Virtualization & cloud mgmt
- + Windows server mgmt



## Information management

- + Database management
- + Business intelligence/analytics
- + Application & data integration
- + Big data analytics



## Mobile workforce management

- + Mobile device mgmt
- + Desktop virtualization
- + Application/data access
- + Secure remote access

## Security

- + Identity & access management
- + Network security
- + Endpoint security
- + Email security

## Data protection

- + Enterprise backup & recovery
- + Virtual protection
- + Application protection
- + Disaster recovery



# Redo Logs – Agenda

Informationen soweit der Logminer reicht

- Was sind Redo Logs? Wofür sind sie da?
- Redo Logs und Performance
- Informationen aus Redo Logs
- Was kann man alles mit Redo Logs machen?
- Fazit



# ACID – Persistenz- mechanismen bei Oracle

# Die ACID Eigenschaft

## Persistenzmechanismen bei Oracle

- Oracle Datenbanken erfüllen die ACID Eigenschaften
  - Die anderen "großen" relationalen Datenbanken auch

A – Transaktionen sind atomar. (**A**tomicity)

C – Die Daten sind stets konsistent. (**C**onsistency)

I – Sessions sind voneinander isoliert. (**I**solation)

D – Abgeschlossene Transaktion sind dauerhaft/persistent gespeichert. (**D**urability)



# Die ACID Eigenschaft

## Persistenzmechanismen bei Oracle

- Daten werden in Datendateien gespeichert.
  - Persistent und auf Platte...
    - › ... irgendwann mal
- Datenänderungen erst nur im Speicher
  - Bei Random Access schneller (Memory IO vs. Disk IO)
  - Aber Speicherinhalte sind flüchtig.
- Lösung: Transaktionsprotokolle
  - Sequentielles schreiben, kein Random Access IO



# Die ACID Eigenschaft

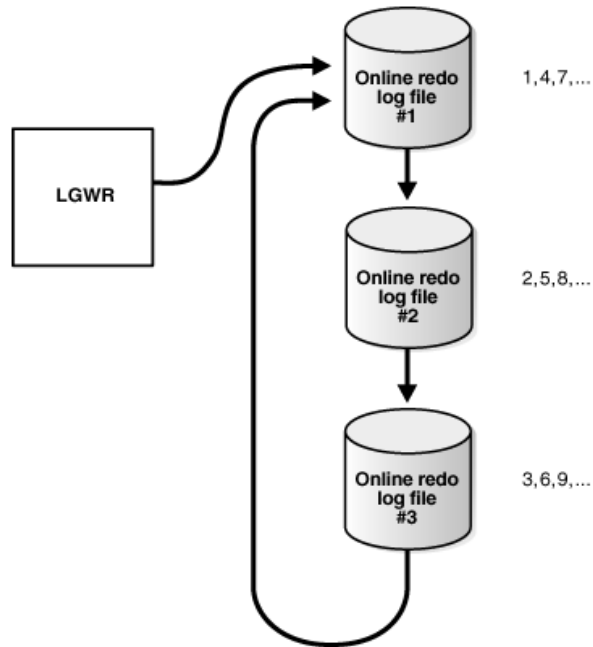
## Persistenzmechanismen bei Oracle

- Database Writer (DBWR) schreibt asynchron und periodisch Daten (Checkpoint).
  - Das kann forciert werden
    - › **ALTER SYSTEM CHECKPOINT;**
    - › Oder nach Logswitches (**ALTER SYSTEM SWITCH LOGFILE;**)
- Redo Log Writer (LGWR) schreibt quasi-synchron
  - Spätestens beim Commit muss Redo Log Buffer auf Platte geschrieben werden, Tx ist erst danach committed.
- Instance Crash?
  - Recovery seit letztem Checkpoint (kann dauern – RTO)

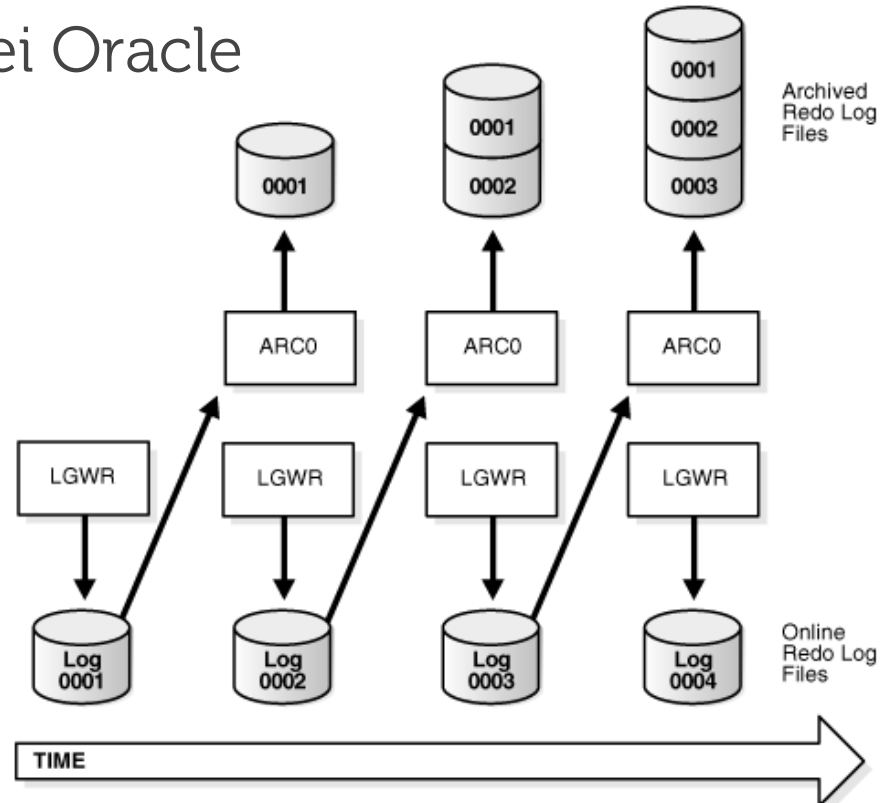


# Die ACID Eigenschaft

## Persistenzmechanismen bei Oracle



Quelle: Oracle 12c Dokumentation



Quelle: Oracle 12c Dokumentation

- Redo Logs werden zyklisch beschrieben.
- Redo Logs werden im ARCHIVELOG Modus archiviert.
  - Wozu? Nach jedem Switch wird doch ein Checkpoint ausgeführt.



# Die ACID Eigenschaft

## Persistenzmechanismen bei Oracle

- Plattencrash, defekter RAID Controller, Verlust des Storage?
  - Dafür gibt es Backups
    - › Full und Incremental (Differential oder Cumulative)
    - › Die enthalten jeweils einen bestimmten Zustand.
  - Archivierte Redo Logs erlauben ein “Vorwärtsrollen” der Datenbank
    - › Ab einem Backup auf jeden Stand, wenn die Redo Logs vorhanden sind.
    - › Lückenlose Archivelog Kette erforderlich
- Ohne Archivierte Redo Logs droht Datenverlust (RPO)!



# Redo Logs und Performance

# Redo Logs und Performance

## Sind Redo Logs ein Flaschenhals?

- Redo Logs werden immer synchron geschrieben.
- Kommt der Redo Log Writer (LGWR) nicht hinterher, wartet die Datenbank.
  - Transaktionen sind solange nicht abgeschlossen.
- Kommt der Archiver (ARCx) nicht hinterher, wartet die Datenbank.
  - Sonst würden Transaktionen überschrieben werden, es gäbe keine Lückenlose Archivelog Kette.
- Redo Logs sind ein notwendiger Flaschenhals.
  - Sie ermöglichen Persistenz bei guter Performance.



# Redo Logs und Performance

## Sind Redo Logs ein Flaschenhals?

- Good Practices um das zu verhindern:
  - Schnelle IO Systeme für Redo Logs
    - › Für alle Redo Log Member
  - Schnelle IO Systeme für Archivierte Redo Logs
  - Mehr Redo Log Gruppen
  - Evtl. Größere Redo Log Dateien.
- Not-so-good Practices, um das zu verhindern (nur in Ausnahmefällen):
  - RAM-Disks für die Redo Logs verwenden
    - › Vorsicht, das umgeht den Persistenz Mechanismus!
  - Tabellen im NOLOGGING Modus und INSERTS mit /\*+APPEND\*/ Hint
    - › Fügt Daten oberhalb der High Water Mark ein und schreibt keine Redo Log Einträge.
    - › Diese Daten können nicht recovert werden, FORCE LOGGING hebt es aus.
- Weniger oft Commits ausführen (wenn es geht).
- Wait Events überwachen und auswerten – nur so bekommt man es überhaupt mit.



# Redo Logs und Performance

## Relevante Wait Events

- **log file sync**
  - Commit oder Rollback, Schreiben des Redo Log Eintrages auf Platte bei Commit oder Rollback noch nicht geschehen.
- **log buffer space**
  - Log Buffer voll und Änderungen kommen schneller, als sie geschrieben werden können.
- **log file switch (archiving needed)**
  - Redo Log kann nicht überschrieben werden, da noch nicht archiviert. Zu langsam oder ARCHIVE\_LOG\_DEST voll?
- **log file switch (checkpoint incomplete)**
  - Redo Log kann nicht überschrieben werden, weil DBWR Daten noch nicht auf Platte geschrieben hat.
- **log file switch completion** [oder **log file switch (private strand flush incomplete)**]
  - Warten darauf, dass Redo Log Writer den Switch an sich durchführt.



Informationen  
aus Redo Logs –  
was steht da  
eigentlich?

# Informationen aus Redo Logs

## Was steht da eigentlich?

- Alles, was recovert werden können soll, muss in den Redo Logs stehen.
  - Jede Änderung an Daten oder der Datenstruktur
- Der Redo Log Eintrag zu jeder Aktion muss alle Informationen enthalten, die zum Recovern dieser Aktion notwendig sind.
  - Tatsächlich ist dort sogar noch mehr enthalten.
- Aber: Das Redo Log Format ist nicht dokumentiert.
  - Es ist ein proprietäres Dateiformat.
- Der Oracle LogMiner kann Redo Logs auslesen.
  - So kommt man an die Informationen aus den Redo Logs.



# Das Logminer Interface

## Auswahl der Redo-Logs und Felder in Toad

The screenshot displays the Logminer interface in Toad, showing three overlapping windows:

- Dictionary Selection:** A window titled "Dictionary" with two radio buttons:  "Use Online Data Dictionary (fastest)" and  "Use Dictionary in Redo Logs".
- 'Next' Button Action:** A window with two radio buttons:  "Build New Dictionary" and  "Use Existing Dictionary".
- Accept or Narrow Date/SCN Range:** A window with two columns: "From :" and "To :".
  - SCN: From 1273623, To 281474976710654
  - Date: From 2013-10-22 15:02:16, To 2013-10-22 15:02:17
- 9i (and newer) LogMiner Options:** A window with three checkboxes:
  - Show Committed Data Only
  - DDL Dictionary Tracking
  - No Dictionary Reset On Select
- Available Logminer Columns:** A dialog box titled "Available Logminer Columns" with an "OK" button. It lists 12 columns with checkboxes:
  - SCN (checked)
  - Timestamp (checked)
  - Log ID (unchecked)
  - Segment Type (checked)
  - Segment Owner (checked)
  - Segment Name (checked)
  - Tablespace (checked)
  - Abs File # (unchecked)
  - Row ID (checked)
  - Session # (checked)
  - Serial # (unchecked)
  - Session Info (unchecked)
  - User Name (checked)
  - Operation (checked)
  - SQL Redo (checked)
  - SQL Undo (checked)
  - Info (unchecked)
  - Status (unchecked)



# Das Logminer Interface

## Darstellung der Redo Log Einträge in Toad

The screenshot displays the Toad for Oracle interface, specifically the LogMiner window. The window title is "Toad for Oracle - [SYSTEM@LOKAL - LogMiner]". The interface includes a menu bar (File, Edit, Search, Editor, Session, Database, Debug, View, Utilities, Runup, Window, Help) and a toolbar. Below the toolbar, there are tabs for "SYSTEM@LOKAL", "Schema Browser", "Editor", "LogMiner", "Data Pump Import", "Import Table Data", "Import Utility Wizard", "Import Watch", "Editor", and "LogMiner".

The main area shows a table of redo log entries. The table has the following columns: SCN, Time Stamp, Segment Name, Username, Operation, and SQL Redo. The entries are as follows:

SCN	Time Stamp	Segment Name	Username	Operation	SQL Redo
1278980	22/10/2013 15:12:34	AUFPOSITIONEN	DEMOLD	INSERT	insert into "DEMOLD"."AUFPOSITIONEN"("AUFNR","POSNR","PRODNR","ANZAHL","EINZELPREIS") values (281,'2','10','3','6,29);
1278980	22/10/2013 15:12:34	AUFPOSITIONEN	DEMOLD	INSERT	insert into "DEMOLD"."AUFPOSITIONEN"("AUFNR","POSNR","PRODNR","ANZAHL","EINZELPREIS") values (258,'1','20','6','88,72);
1278980	22/10/2013 15:12:34	AUFPOSITIONEN	DEMOLD	INSERT	insert into "DEMOLD"."AUFPOSITIONEN"("AUFNR","POSNR","PRODNR","ANZAHL","EINZELPREIS") values (146,'3','18','6','45,21);
1278980	22/10/2013 15:12:34	AUFPOSITIONEN	DEMOLD	INSERT	insert into "DEMOLD"."AUFPOSITIONEN"("AUFNR","POSNR","PRODNR","ANZAHL","EINZELPREIS") values (272,'2','13','8','33,25);
1278980	22/10/2013 15:12:34	AUFPOSITIONEN	DEMOLD	INSERT	insert into "DEMOLD"."AUFPOSITIONEN"("AUFNR","POSNR","PRODNR","ANZAHL","EINZELPREIS") values (32,'2','2','5','78,69);
1278980	22/10/2013 15:12:34	AUFPOSITIONEN	DEMOLD	INSERT	insert into "DEMOLD"."AUFPOSITIONEN"("AUFNR","POSNR","PRODNR","ANZAHL","EINZELPREIS") values (228,'3','8','8','22,32);
1278980	22/10/2013 15:12:34	AUFPOSITIONEN	DEMOLD	INSERT	insert into "DEMOLD"."AUFPOSITIONEN"("AUFNR","POSNR","PRODNR","ANZAHL","EINZELPREIS") values (245,'2','9','5','67,39);
1278980	22/10/2013 15:12:34	AUFPOSITIONEN	DEMOLD	INSERT	insert into "DEMOLD"."AUFPOSITIONEN"("AUFNR","POSNR","PRODNR","ANZAHL","EINZELPREIS") values (125,'2','5','5','58,25);
1278981	22/10/2013 15:12:34		DEMOLD	COMMIT	commit;
1278982	22/10/2013 15:12:34		DEMOLD	START	set transaction read write;
1278983	22/10/2013 15:12:34		DEMOLD		
1278986	22/10/2013 15:12:34	ADRESSEN	DEMOLD	DDL	ALTER TABLE ADRESSEN ...
1278989	22/10/2013 15:12:34	TAB\$	DEMOLD	UPDATE	update "SYS"."TAB\$" set "DATAOBJ#" = '17530', "TS#" = '4', "FILE#" = '4', "BLOCK#" = '13618', "BOBJ#" = NULL, "TAB#" = NULL, "COLS" = '5', "CLUCOLS" = NULL, "PCTFREES" = '10', "PCTUSED#" = '40', "INITRANS" = '1', "MAXTF
1278989	22/10/2013 15:12:34	OBJ\$	DEMOLD	UPDATE	update "SYS"."OBJ\$" set "OBJ#" = '17530', "DATAOBJ#" = '17530', "TYPE#" = '2', "CTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "MTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "STIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "STATUS
1278989	22/10/2013 15:12:34	CDEF\$	DEMOLD	UPDATE	update "SYS"."CDEF\$" set "ENABLED" = '1', "MTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "DEFER" = '4', "SPARE2" = '0', "SPARE3" = '1278985' where "CON#" = '4390' and "OBJ#" = '17530' and "ENABLED" IS NULL and "MTIME" =
1278990	22/10/2013 15:12:34		DEMOLD	COMMIT	commit;
1278991	22/10/2013 15:12:34		DEMOLD	START	set transaction read write;
1278992	22/10/2013 15:12:34		DEMOLD		
1278995	22/10/2013 15:12:34	ADRESSEN	DEMOLD	DDL	ALTER TABLE ADRESSEN ...
1278998	22/10/2013 15:12:34	TAB\$	DEMOLD	UPDATE	update "SYS"."TAB\$" set "DATAOBJ#" = '17530', "TS#" = '4', "FILE#" = '4', "BLOCK#" = '13618', "BOBJ#" = NULL, "TAB#" = NULL, "COLS" = '5', "CLUCOLS" = NULL, "PCTFREES" = '10', "PCTUSED#" = '40', "INITRANS" = '1', "MAXTF
1278998	22/10/2013 15:12:34	OBJ\$	DEMOLD	UPDATE	update "SYS"."OBJ\$" set "OBJ#" = '17530', "DATAOBJ#" = '17530', "TYPE#" = '2', "CTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "MTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "STIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "STATUS
1278998	22/10/2013 15:12:34	CDEF\$	DEMOLD	UPDATE	update "SYS"."CDEF\$" set "ENABLED" = '1', "MTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "DEFER" = '4', "SPARE2" = '0', "SPARE3" = '1278994' where "CON#" = '4391' and "OBJ#" = '17530' and "ENABLED" IS NULL and "MTIME" =
1278999	22/10/2013 15:12:34		DEMOLD	COMMIT	commit;
1279000	22/10/2013 15:12:34		DEMOLD	START	set transaction read write;
1279001	22/10/2013 15:12:34		DEMOLD		
1279004	22/10/2013 15:12:34	AUFPOSITIONEN	DEMOLD	DDL	ALTER TABLE AUFPOSITIONEN ...
1279007	22/10/2013 15:12:34	TAB\$	DEMOLD	UPDATE	update "SYS"."TAB\$" set "DATAOBJ#" = '17540', "TS#" = '4', "FILE#" = '4', "BLOCK#" = '13698', "BOBJ#" = NULL, "TAB#" = NULL, "COLS" = '5', "CLUCOLS" = NULL, "PCTFREES" = '10', "PCTUSED#" = '40', "INITRANS" = '1', "MAXTF
1279007	22/10/2013 15:12:34	CDEF\$	DEMOLD	UPDATE	update "SYS"."CDEF\$" set "ENABLED" = '1', "MTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "DEFER" = '4', "SPARE2" = '0', "SPARE3" = '1279003' where "CON#" = '4401' and "OBJ#" = '17540' and "ENABLED" IS NULL and "MTIME" =
1279007	22/10/2013 15:12:34	OBJ\$	DEMOLD	UPDATE	update "SYS"."OBJ\$" set "OBJ#" = '17540', "DATAOBJ#" = '17540', "TYPE#" = '2', "CTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "MTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "STIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "STATUS
1279008	22/10/2013 15:12:34		DEMOLD	COMMIT	commit;
1279009	22/10/2013 15:12:35		DEMOLD	START	set transaction read write;
1279010	22/10/2013 15:12:35		DEMOLD		
1279013	22/10/2013 15:12:35	AUFPOSITIONEN	DEMOLD	DDL	ALTER TABLE AUFPOSITIONEN ...

At the bottom of the window, there is an "Output" tab and a status bar showing "AutoCommit is OFF" and "CAPS NUM INS".



# Das Logminer Interface

## Auslesen über Toad

- DDLs sind auch nur DMLs

– Im Data Dictionary

•	1278647	22/10/2013 15:12:02	STATUS_PK	DEMOLD	DDL	drop index "DEMOLD"."STATUS_PK";
	1278649	22/10/2013 15:12:02	ICOL\$	DEMOLD	DELETE	delete from "SYS"."ICOL\$" where "OBJ#" = '17533' and "BO#" = '17532' and "COL#" = '1' and "POS#" = '1' and "SEG"
	1278650	22/10/2013 15:12:02		DEMOLD	INTERNAL	
	1278650	22/10/2013 15:12:02		DEMOLD	INTERNAL	
	1278650	22/10/2013 15:12:02	IND\$	DEMOLD	DELETE	delete from "SYS"."IND\$" where "OBJ#" = '17533' and "DATAOBJ#" = '17533' and "TS#" = '4' and "FILE#" = '0' and "B"

– INSERTs, UPDATEs und DELETEs

▷	1278986	22/10/2013 15:12:34	ADRESSEN	DEMOLD	DDL	ALTER TABLE ADRESSEN ...
	1278989	22/10/2013 15:12:34	TAB\$	DEMOLD	UPDATE	update "SYS"."TAB\$" set "DATAOBJ#" = '17530', "TS#" = '4', "FILE#" = '4', "BLOCK#" = '13618', "BOBJ#" = NULL, "TAB#" = NU
	1278989	22/10/2013 15:12:34	OBJ\$	DEMOLD	UPDATE	update "SYS"."OBJ\$" set "OBJ#" = '17530', "DATAOBJ#" = '17530', "TYPE#" = '2', "CTIME" = TO_DATE('22.10.13', 'DD.MM.RR')
	1278989	22/10/2013 15:12:34	CDEF\$	DEMOLD	UPDATE	update "SYS"."CDEF\$" set "ENABLED" = '1', "MTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "DEFER" = '4', "SPARE2" = '0', "SPAR
	1278990	22/10/2013 15:12:34		DEMOLD	COMMIT	commit;
	1278991	22/10/2013 15:12:34		DEMOLD	START	set transaction read write;
	1278992	22/10/2013 15:12:34		DEMOLD		
	1278995	22/10/2013 15:12:34	ADRESSEN	DEMOLD	DDL	ALTER TABLE ADRESSEN ...
	1278998	22/10/2013 15:12:34	TAB\$	DEMOLD	UPDATE	update "SYS"."TAB\$" set "DATAOBJ#" = '17530', "TS#" = '4', "FILE#" = '4', "BLOCK#" = '13618', "BOBJ#" = NULL, "TAB#" = NU
	1278998	22/10/2013 15:12:34	OBJ\$	DEMOLD	UPDATE	update "SYS"."OBJ\$" set "OBJ#" = '17530', "DATAOBJ#" = '17530', "TYPE#" = '2', "CTIME" = TO_DATE('22.10.13', 'DD.MM.RR')
	1278998	22/10/2013 15:12:34	CDEF\$	DEMOLD	UPDATE	update "SYS"."CDEF\$" set "ENABLED" = '1', "MTIME" = TO_DATE('22.10.13', 'DD.MM.RR'), "DEFER" = '4', "SPARE2" = '0', "SPAR
	1278999	22/10/2013 15:12:34		DEMOLD	COMMIT	commit;
	1279000	22/10/2013 15:12:34		DEMOLD	START	set transaction read write;



# Das Logminer Interface

## Auslesen über Toad

- CREATEs sind INSERTs

1278496	22/10/2013 15:12:01	ICOL\$	DEMOLD	INSERT	insert into "SYS"."ICOL\$ "("OBJ#","BO#","COL#","POS#","SEGCOL#","SEGCOLLENGTH","OFFSET","INTCOL#","SPARE1","SPARE2"
1278496	22/10/2013 15:12:01	IND\$	DEMOLD	INSERT	insert into "SYS"."IND\$ "("OBJ#","DATAOBJ#","TS#","FILE#","BLOCK#","BO#","INDMETHOD#","COLS","PCTFREE\$","INITRANS","M
1278496	22/10/2013 15:12:01	TAB\$	DEMOLD	INSERT	insert into "SYS"."TAB\$ "("OBJ#","DATAOBJ#","TS#","FILE#","BLOCK#","BOBJ#","TAB#","COLS","CLUCOLS","PCTFREE\$","PCTUS
1278496	22/10/2013 15:12:01		DEMOLD	INTERNAL	
1278496	22/10/2013 15:12:01	COL\$	DEMOLD	INSERT	insert into "SYS"."COL\$ "("OBJ#","COL#","SEGCOL#","SEGCOLLENGTH","OFFSET","NAME","TYPE#","LENGTH","FIXEDSTORAGE","F
1278496	22/10/2013 15:12:01		DEMOLD	START	set transaction read write;
1278496	22/10/2013 15:12:01	COL\$	DEMOLD	INSERT	insert into "SYS"."COL\$ "("OBJ#","COL#","SEGCOL#","SEGCOLLENGTH","OFFSET","NAME","TYPE#","LENGTH","FIXEDSTORAGE","F
1278497	22/10/2013 15:12:01		DEMOLD	COMMIT	commit;
1278498	22/10/2013 15:12:01	CDEF\$	DEMOLD	INSERT	insert into "SYS"."CDEF\$ "("CON#","OBJ#","COLS","TYPE#","ROBJ#","RCON#","RRULES","MATCH#","REFACT","ENABLED","CONDU
1278498	22/10/2013 15:12:01	CCOL\$	DEMOLD	INSERT	insert into "SYS"."CCOL\$ "("CON#","OBJ#","COL#","POS#","INTCOL#","SPARE1","SPARE2","SPARE3","SPARE4","SPARE5","SPARE
1278499	22/10/2013 15:12:01	AUFSTATUS	DEMOLD	DDL	CREATE TABLE aufstatus (...
1278502	22/10/2013 15:12:01		DEMOLD	COMMIT	commit;

- Manche Operationen sind INTERNAL
  - z.B. Indexpflege
- Manche Operationen sind UNSUPPORTED
  - z.B. CLOB Operationen (hängt von der DB Version ab)



Was kann man  
alles damit  
machen? –  
Lückenlose  
Nachvollziehbar  
keit

# Informationen aus Redo Logs

Was steht da eigentlich?

- Informationsbeschaffung, gezielte Restores aus Backups
- Replikation
  - Physikalisch, Hot Standby System
  - Logisch, Migrationen, Reporting, Datenintegration
- Auditing
  - Wer hat wann was geändert?
  - Pflegen von Staging Umgebungen für ein Data Warehouse



Fazit –  
Was können wir  
mitnehmen?

# Informationen aus Redo Logs

## Was steht da eigentlich?

- Redo Logs sind der Kern der Datenbank
  - Sonst keine Persistenz oder sehr sehr schlechte Performance
- Auch archivierte Redo Logs zu verlieren kann Datenverlust bedeuten.
  - Lückenlose Archivelogkette
- Redo Logs sind für die Performance unheimlich wichtig.
  - Die Datenbank muss auf das Log Processing warten.
- Sie enthalten aber viele Informationen.
  - Die auszulesen lohnt sich.
- Es gibt Tools und Lösungen, die sich das zunutze machen.
  - Es lohnt sich also, sich damit zu beschäftigen.



---

Vielen Dank.  
Welche Fragen haben Sie?

---

