



Oracle^{19c}_{20c}

Das umfassende Praxis-Handbuch

Inhaltsverzeichnis

	Einleitung	13
	Der Aufbau des Buches	14
	Konventionen	15
	Software und Skripte	15
	Über den Autor	16
Teil I	Installation, Architektur und Infrastruktur	17
1	Installation und Konfiguration	19
1.1	Software-Installation	20
1.1.1	Installation mit dem Setup Wizard	21
1.1.2	RPM-basierende Installation	32
1.2	Eine Oracle-19c-Datenbank erstellen	35
1.3	Eine Oracle-20c-Datenbank erstellen	41
1.4	Beispielschemata	42
1.5	Windows-spezifische Besonderheiten	42
1.6	Erste Administrationsschritte	48
1.7	Administrationswerkzeuge	54
1.7.1	Administration mit der Kommandozeile	55
1.7.2	Administration mit dem Oracle Enterprise Manager	58
1.7.3	Der Oracle SQL Developer	60
1.8	Hilfe und Support	61
1.9	Praktische Tipps zur Datenbankadministration	64
2	Die Oracle-Datenbankarchitektur	81
2.1	Übersicht über die Architektur	81
2.1.1	Die Struktur der Datenbank	81
2.1.2	Die Struktur der Instanz	94
2.1.3	Automatic Memory Management (AMM)	97
2.2	Prozesse und Abläufe	102
2.2.1	Die Oracle-Hintergrundprozesse	103
2.2.2	Lesekonsistenz	105
3	Interne Strukturen und Objekte	113
3.1	Datenbankstrukturen verwalten	113
3.1.1	Tablespaces und Datafiles	113
4	Aufbau einer Datenbankinfrastruktur	123
4.1	Überwachung	123

4.2	Backup and Recovery	134
4.3	Standardisierung.....	135
4.4	Diagnostik	135
4.4.1	Die Komponenten der Fault Diagnostic Infrastructure.....	136
4.4.2	Die Support Workbench.....	139

Teil II Oracle-Datenbankadministration 145

5	Backup and Recovery	147
5.1	Begriffe.....	147
5.2	Backup-and-Recovery-Strategien	151
5.3	Der Recovery Manager (RMAN).....	155
5.3.1	Die RMAN-Architektur	156
5.3.2	Einen Recovery-Katalog erstellen.....	157
5.3.3	Die RMAN-Konfiguration	160
5.3.4	Sicherungen mit RMAN	166
5.3.5	Sicherungen mit dem Oracle Enterprise Manager	172
5.3.6	Einen Media Manager verwenden.....	175
5.4	Die Fast Recovery Area	177
5.4.1	Dateien in die Fast Recovery Area legen.....	179
5.4.2	Die Fast Recovery Area verwalten	180
5.5	Restore und Recovery mit RMAN	181
5.5.1	Verlust eines Datafiles	181
5.5.2	Disaster Recovery	183
5.5.3	Verlust von Online-Redo-Log-Dateien.....	186
5.5.4	Block Media Recovery.....	187
5.5.5	Der Data Recovery Advisor.....	188
5.5.6	Switch zur Datafile-Kopie.....	192
5.5.7	Eine Tabelle mit RMAN zurückspeichern	195
5.6	Wiederherstellung mit Flashback-Operationen	199
5.6.1	Flashback Table.....	199
5.6.2	Flashback Drop	201
5.6.3	Flashback Transaction History	202
5.6.4	Flashback Database.....	203
5.7	Tablespace Point-in-Time Recovery	207
6	Schnittstellen	213
6.1	Data Pump.....	214
6.1.1	Die Data-Pump-Architektur.....	214
6.1.2	Data-Pump-Export	215
6.1.3	Data-Pump-Import	220
6.1.4	Data Pump über Datenbank-Link	222
6.1.5	Full Transportable Export/Import.....	224
6.1.6	Data-Pump-Performance und Monitoring	226

6.1.7	Data Pump für die Migration einsetzen	228
6.1.8	Ein Dumpfile nach Excel konvertieren	233
6.1.9	Data Pump mit dem Enterprise Manager	235
6.2	SQL*Loader	237
6.2.1	Daten laden	237
6.2.2	Externe Tabellen laden	240
6.2.3	Daten entladen	240
7	Job-Verwaltung	243
7.1	Die Architektur des Schedulers	243
7.2	Scheduler-Jobs verwalten	243
7.3	Privilegien	246
7.4	Job-Ketten	247
7.5	Automatisierte Wartungstasks	248
8	Oracle Net Services	251
8.1	Die Oracle-Net-Architektur	252
8.2	Die Server-Konfiguration	256
8.3	Die Shared-Server-Konfiguration	259
8.4	Oracle Net Performance	262
9	Sicherheit und Überwachung	265
9.1	Grundlegende Features und Maßnahmen	267
9.1.1	Benutzer, Rollen und Profile	267
9.1.2	Einfache Maßnahmen zur Erhöhung der Sicherheit	272
9.2	Virtual Private Database (VPD)	277
9.2.1	Application Context	278
9.2.2	Eine VPD konfigurieren	281
9.3	Database Vault	282
10	Der Resource Manager	285
11	Globalization Support	291
11.1	Datentypen mit Zeitzonen	292
11.2	Die Hierarchie der NLS-Parameter	292
11.3	Linguistische Sortierungen und Vergleiche	294
11.4	Der Locale Builder	296
11.5	Der Character Set Scanner	297
11.6	Sprachen im Enterprise Manager	298
12	Die XML-Datenbank	299
12.1	Die Architektur der XML-DB	299
12.2	XML-Dokumente speichern	301
12.2.1	Die dokumentenbasierte Speicherung	303
12.2.2	Die objektrelationale Speicherung	304
12.3	XML-Dokumente abfragen	307

13	Automatic Storage Management (ASM)	311
13.1	Storage-Systeme	312
13.1.1	Network Attached Storage	312
13.1.2	Internet Small Computer System Interface (iSCSI)	313
13.1.3	Storage Area Network (SAN)	314
13.2	Die ASM-Architektur im Überblick	316
13.3	ASM-Disks	318
13.3.1	ASM-Disks auf verschiedenen Plattformen	320
13.3.2	Eine Testumgebung aufsetzen	327
13.4	Die ASM-Instanz	329
13.5	Diskgruppen	333
13.6	ASM überwachen und verwalten	341
13.6.1	ASM-Performance	341
13.6.2	ASMCMD	344
13.6.3	Verwaltung mit dem Enterprise Manager	346
13.7	Das ASM-Cluster-Filesystem	348
13.7.1	General Purpose ACFS-Dateisystem	350
13.8	CRS-Managed-ACFS-Dateisystem	350
13.8.1	ACFS Snapshots	352
13.9	Eine Datenbank nach ASM konvertieren	353
14	Upgrades, Patching und Cloning	357
14.1	Upgrade und Migration	357
14.2	Ein Upgrade umfassend planen	357
14.2.1	Der Upgrade-Pfad	359
14.2.2	Upgrade-Methoden	359
14.2.3	Ein Upgrade vorbereiten	361
14.2.4	Upgrade mit dem DBUA	363
14.2.5	Manuelles Upgrade	366
14.2.6	Upgrade mit dem Oracle Enterprise Manager	369
14.3	Replay Upgrade	372
14.4	Patching	375
14.4.1	Einen RU-Patch einspielen	376
14.4.2	One-Off Patches einspielen	378
14.4.3	Patching mit dem Oracle Enterprise Manager	381
14.5	Ein Read Only Oracle Home klonen	383
14.6	Fleet Patching and Provisioning	385
14.6.1	Vorbereitung	386
14.6.2	Fleet Patching and Provisioning am Beispiel	388
14.6.3	Fazit	398

Teil III	Erweiterte Administration und Spezialthemen	399
15	Multitenant-Datenbanken	401
15.1	Die Multitenant-Architektur	402
15.2	Integrierbare Datenbanken administrieren	408
15.3	Datenbanken klonen und verschieben	415
15.3.1	Klonen einer lokalen Datenbank	416
15.3.2	Klonen einer Remote-Datenbank	419
15.3.3	Refreshable Clone PDBs	420
15.4	Datenbanken ein- und ausstecken	423
15.5	PDB Relocation	426
15.6	Backup and Recovery	428
15.7	Konsolidierung bestehender Datenbanken	433
16	Recovery-Szenarien für Experten	437
16.1	Recovery und Strukturänderungen	438
16.1.1	Szenario 1	438
16.1.2	Lösung 1	440
16.1.3	Szenario 2	441
16.1.4	Lösung 2	443
16.2	Der Trick mit den Redo-Log-Dateien	446
16.3	Der Data Recovery Advisor	450
16.4	Ein unbekanntes Szenario	452
16.5	Ausfall des Recovery-Katalogs	455
16.6	Der Oracle LogMiner	457
17	Erweiterte Sicherheitsthemen	461
17.1	Sicherheit von Netzwerk-Paketen	461
17.2	Netzwerk-Verschlüsselung	464
17.3	Datenbank-Verschlüsselung	471
17.4	Hackerangriffe abwehren	473
17.4.1	Angriffe auf den Authentifizierungsprozess abwehren	474
17.4.2	PL/SQL Injection verhindern	482
17.4.3	Brute-Force-Angriffe	486
17.5	Datenbankaudits	488
17.6	Oracle Data Redaction	490
18	Performance-Tuning	497
18.1	Datenbank-Tuning	498
18.1.1	Vorgehen und Werkzeuge	499
18.1.2	Problemanalyse	500
18.1.3	Instance-Tuning	519
18.1.4	Disk-Tuning	533
18.1.5	Real-Time-ADDM-Analyse	535

18.2	SQL-Tuning	537
18.2.1	Der SQL-Optimizer	538
18.2.2	Optimizer-Statistiken	540
18.2.3	SQL-Performance-Probleme identifizieren	543
18.2.4	Dynamische Features	545
19	Real Application Testing	557
19.1	Workload Capture	559
19.2	Workload Preprocessing	563
19.3	Workload Replay	563
19.4	SQL Performance Analyzer	566
19.4.1	Eine SQL-Anweisung analysieren	567
20	Engineered Systems	575
20.1	Oracle Exadata	575
20.1.1	Modelle	576
20.1.2	Das Storage-System	577
20.1.3	Neue Performance-Features	578
20.2	Oracle Database Appliance (ODA)	579
20.3	ZFS Storage Appliance	580
21	Data Science und Machine Learning	583
21.1	Data Science	584
21.1.1	Python als Werkzeug	585
21.2	Oracle Machine Learning	594
21.2.1	Oracle Machine Learning for SQL (OML4SQL)	594
21.2.2	OML4R	601
Teil IV	Oracle In-Memory	607
22	Die Oracle-In-Memory-Architektur	609
22.1	Das Spalten-Format	610
22.2	Die Prozess-Architektur	615
22.3	Die CPU-Architektur	617
23	Eine In-Memory-Datenbank planen und aufbauen	619
23.1	Eine In-Memory-Datenbank planen	619
23.2	Aufbau einer In-Memory-Datenbank	623
23.2.1	Manuelle Verwaltung von Objekten	625
23.2.2	Automatische Verwaltung von Objekten	635
23.3	In-Memory-Administration mit dem OEM	638
24	Optimierung von SQL-Anweisungen	641
24.1	Joins von In-Memory-Objekten	644
24.2	Summen und Aggregation	648

25	Hochverfügbarkeit für In-Memory	651
25.1	In-Memory und Oracle RAC	651
25.2	In-Memory und Active Data Guard	653
25.3	In-Memory FastStart	655
Teil V	Hochverfügbarkeit und verteilte Architekturen	661
26	Data Guard	663
26.1	Architektur	664
26.2	Physical-Standby-Datenbanken	666
26.2.1	Vorbereitung der Primärdatenbank	668
26.2.2	Vorbereitung der Standby-Datenbank	670
26.2.3	Kopieren der Primärdatenbank	672
26.2.4	Aktivierung von Data Guard	673
26.2.5	Physical-Standby-Datenbanken verwalten	675
26.3	Logical-Standby-Datenbanken	696
27	Global Data Services (GDS)	703
27.1	Architektur und Features	703
27.2	Eine GDS-Umgebung aufsetzen	705
27.2.1	Den GSM installieren	706
27.2.2	Den GDS-Katalog erstellen	708
27.2.3	GSM zum Katalog hinzufügen	708
27.2.4	Pool, Region, Datenbanken und Services hinzufügen	709
27.2.5	GDS-Client-Konfiguration	710
28	Real Application Clusters	713
28.1	Cluster-Architekturen	714
28.2	Cache Fusion	717
28.3	Installation und Konfiguration	722
28.3.1	Die Installation vorbereiten	723
28.3.2	Die Grid Infrastructure installieren	728
28.3.3	Die Datenbank-Software installieren	739
28.3.4	Eine RAC-Datenbank erstellen	742
28.4	Real Application Clusters administrieren	747
28.4.1	ORAchk	747
28.4.2	Die RAC-Datenbank verwalten	750
28.5	RAC-Performance	766
28.6	Backup and Recovery	771
28.7	RAC und Data Guard	772
28.8	Oracle Restart	774

Teil VI	Oracle Cloud Computing	775
29	Der Enterprise Manager Cloud Control	777
29.1	Architekturübersicht	777
29.2	Installation	779
29.3	Das EM CLI	787
30	Verwaltung der Datenbankinfrastruktur	791
30.1	Den Agent ausrollen	792
30.2	Weitere Ziele registrieren	794
30.3	Datenbanken verwalten	799
30.3.1	Monitoring mit dem OEM	800
30.3.2	Klonen und Replikation	804
31	Eine private Cloud aufsetzen	809
31.1	Cloud Computing für Datenbanken	809
31.2	Die Cloud-Management-Infrastruktur bereitstellen	810
31.3	Einen DBaaS-Dienst einrichten	815
31.4	Das Selbstservice-Portal verwenden	816
32	Die Oracle Autonomous Database	819
32.1	Die Oracle-Cloud-Infrastruktur (OCI)	820
32.2	Das Provisioning- und Verbindungsmodul	826
32.3	Migration und Laden von Daten	840
32.3.1	Migration von Datenbanken	840
32.3.2	Daten in die OAD laden	844
32.4	Administration der OAD	855
	Stichwortverzeichnis	867

Einleitung

Seit dem Erscheinen des Buches »Oracle 12c – Das umfassende Handbuch« sind mehr als fünf Jahre vergangen. In der Zwischenzeit hat sich sehr viel in der Datenbankwelt und natürlich auch beim Marktführer Oracle getan. Auch die vielen positiven Kritiken und die große Nachfrage haben Verlag und Autor dazu bewegt, wieder ein aktuelles DBA-Handbuch herauszugeben.

Dabei handelt es sich nicht um eine einfache Überarbeitung des Vorgängers. Das Buch wurde komplett neu strukturiert und reflektiert den Wandel, den die Datenbankanfrastruktur in den letzten Jahren durchlebt hat.

Auch dieses Buch richtet sich wieder an einen breiten Leserkreis wie Datenbankadministratoren, Systemberater und Architekten und nicht zuletzt an Entwickler von Oracle-Applikationen. Es ist gleichermaßen auch für Einsteiger gedacht, die bereits grundlegende Kenntnisse in der IT besitzen und beginnen wollen, sich in die Oracle-Welt einzuarbeiten. Trainer können sich einen Überblick verschaffen und Detailwissen vertiefen. Neben vielen Beispielen und Konfigurationsanleitungen finden Sie Diskussionsschwerpunkte, Erfahrungen und wertvolle Tipps aus dem praktischen Einsatz. Das Buch ist ein Handbuch und Nachschlagewerk für alle, die sich mit Planung, Einsatz und Administration von Oracle-Datenbanken beschäftigen.

Weshalb erscheint das Buch für die Doppelversion 19c/20c? Seit der Version 18c verfolgt Oracle eine neue Release-Strategie. Die bisherige Strategie von zwei Releases pro Version und einem Versionszyklus von vier oder fünf Jahren hat sich als nicht mehr marktgerecht erwiesen. Es gibt nun »Long Term Releases« und »Short Term Releases« (Annual Releases). Das hat den Vorteil, dass mit den Short Term Releases neue Features schneller auf den Markt gebracht werden können.

Oracle 19c ist ein Long Term Release mit einem projizierten Supportende im März 2026. Die Version 20c wird voraussichtlich zum Ende des Jahres 2021 aus dem Support laufen und durch die Version 21c abgelöst werden. Als nächstes Long Term Release ist die Version 22c angekündigt, die im Frühjahr 2022 erscheinen soll. Die Anwender werden die Mehrheit ihrer Datenbanken auf 19c migrieren und die Version 20c für solche Datenbanken verwenden, deren Anwendungen auf die neuen Features angewiesen sind.

Dieses Buch trägt der neuen Release-Strategie ebenfalls Rechnung und berücksichtigt sowohl die Version 19c als auch die Version 20c. Die Version 19c taucht deshalb bei der Mehrheit der Features und Beispiele auf. Überall dort, wo es um neue 20c-Features oder Besonderheiten der Version geht, finden Sie die Version 20c. Damit erfüllt das Buch die Erwartungen der Anwender, die prinzipiell sowohl auf das Long Term Release 19c migrieren als auch die Version 20c wegen der darin enthaltenen neuen Features einsetzen wollen.

Die Veränderungen gegenüber der Version 12c sind erheblich. Insbesondere das Public-Cloud-Angebot hat nennenswerte Fortschritte gemacht und wird von einer zunehmenden

Anzahl von Unternehmen als Option betrachtet, um Datenbanken zu konsolidieren und Kosten zu senken. Der gesamte Teil VI beschäftigt sich mit diesem Thema.

Die Menge der gesammelten Daten und die Notwendigkeit, diese für Geschäftsprozesse und ein effektives Marketing auszuwerten, haben sich erheblich erhöht. Dies ist ohne den Einsatz von In-Memory-Datenbanken nicht mehr möglich. Dabei kommen die klassischen Data-Warehouse-Datenbanken immer seltener zum Einsatz und werden in OLTP-Datenbanken integriert. In Teil IV finden Sie eine umfassende Darstellung für den Einsatz von Oracle In-Memory.

Für die Auswertung der gesammelten Daten spielen die Themen »Data Science« und »Machine Learning« eine zentrale Rolle. Datenbankhersteller liefern für diese Themen fertige Produkte und Features. Davon ist auch Oracle nicht ausgenommen. In Kapitel 21 finden Sie eine Einführung in Features wie *Oracle Machine Learning for SQL* und *Oracle Machine Learning for R*.

Der Oracle-ML-Server spielt auch im Bereich der autonomen Datenbank (OAD) eine zentrale Rolle. Die OAD ist Bestandteil der strategischen Ausrichtung für die Oracle-Datenbank. Mithilfe von ML-Funktionalität sollen die Stabilität erhöht, Kosten gesenkt und der manuelle Administrationsaufwand reduziert werden. Kapitel 32 beschäftigt sich im Detail mit der autonomen Datenbank.

Eine Besonderheit der Version 20c ist, dass die Multitenant-Containerdatenbank die einzig mögliche Architektur ist und Nicht-Container-Datenbanken nicht mehr unterstützt werden. Sicherlich eine richtige Entscheidung aus Sicht des Herstellers, um Komplexität und Anzahl der möglichen Architekturen zu reduzieren. Für den Betrieb entstehen daraus einige Konsequenzen, die teilweise durch das neue Release unterstützt werden. Hinweise dazu finden Sie insbesondere in Kapitel 15, das sich mit integrierbaren Datenbanken beschäftigt, sowie in Kapitel 1, in dem unter anderem Upgrade-Methoden dargestellt sind.

Der Aufbau des Buches

Teil I mit dem Titel »Installation, Architektur und Infrastruktur« bietet Lesern, die wenig oder keine Erfahrung in der Oracle-Welt haben, einen Einstieg. Sie finden darin eine Anleitung zur Installation und lernen die ersten Schritte für die Administration und die Architektur kennen. Für Erfahrene liefert Kapitel 1 Informationen zum Umgang mit einem »Golden Image« und die RPM-basierende Installation. Weiterhin finden Sie im ersten Teil eine Beschreibung von Datenbankarchitektur und -infrastruktur.

In **Teil II** finden Sie die mit den typischen Aufgaben eines Datenbank-Administrators verbundenen Themen wie »Backup and Recovery«, »Resource Manager« oder »Automatic Storage Management«. Kapitel 14 »Upgrades, Patching und Cloning« enthält neben den Standard-Verfahren Anleitungen und Beispiele zum neuen »Replay Upgrade« und zum »Fleet Patching«.

Teil III umfasst eine Sammlung von Spezialthemen, die wichtiger Bestandteil der Oracle-Datenbank sind und mit denen Administratoren, Architekten und Entwickler vertraut sein sollten. Er beginnt mit dem Aufbau und der Administration von Multitenant-Datenbanken, die in der Version 20c zur Pflicht geworden sind. Kapitel 16 »Recovery-Szenarien für Experten« hilft, knifflige Recovery-Aufgaben schnell und ohne Datenverlust zu lösen. Auch das Thema »Sicherheit« gerät immer mehr in den Fokus und wurde in Kapitel 17 vertieft. Neben

dem Dauerbrenner »Performance-Tuning« finden Sie in diesem Teil auch neue Themen wie »Data Science« und »Machine Learning«, die natürlich vor der Oracle-Datenbank nicht haltmachen.

Teil IV widmet sich dem Thema »In-Memory-Datenbank«. Analytische Auswertungen großer Datenmengen sind ohne den Einsatz einer In-Memory-Datenbank nicht mehr performant durchführbar. Hier finden Sie eine Beschreibung der Architektur und der Planung und des Aufbaus einer In-Memory-Datenbank. Informationen zu Optimierung und Hochverfügbarkeit runden das Thema ab. Wie im gesamten Buch gibt es auch hier zahlreiche Beispiele zum Ausprobieren.

Hochverfügbarkeit und die Verwaltung von verteilten Datenbanken sind Optionen, die insbesondere von international agierenden Unternehmen und Konzernen verbreitet eingesetzt werden. Sie sind wichtiger Bestandteil der Datenbankinfrastruktur. In **Teil V** finden Sie eine umfassende Darstellung.

Im Bereich Cloud Computing gibt es signifikante Fortschritte. Insbesondere der Public-Cloud-Sektor der Oracle-Datenbank hat sich wesentlich weiterentwickelt. Viele Unternehmen planen und testen bereits die Migration in die Public Cloud. **Teil VI** enthält eine Darstellung von Private und Public Cloud sowie Anleitungen, wie man beides unter einen Hut bringen kann. Dargestellt sind auch die Besonderheiten der autonomen Datenbank, und was bei einer Migration zu beachten ist.

Konventionen

Begriffe, die in spitzen Klammern dargestellt sind, bezeichnen eine zu ersetzende Variable. So ist zum Beispiel der Ausdruck `<ORACLE_SID>` durch die reale SID der Datenbank zu ersetzen.

Umgebungsvariablen unterscheiden sich für Unix-/Linux- und Windows-Betriebssysteme. Während Unix-Variablen durch ein führendes \$-Zeichen dargestellt werden (zum Beispiel `$ORACLE_HOME`), muss unter Windows ein %-Zeichen an den Anfang und das Ende der Variablen gesetzt werden (`%ORACLE_HOME%`). Im Buch wird vorwiegend die Unix-Schreibweise verwendet. Wenn keine Einschränkungen erwähnt sind, behalten die Aussagen für Windows-Betriebssysteme ihre Gültigkeit. Lesen Sie zu diesem Thema auch den Abschnitt 1.5, »Windows-spezifische Besonderheiten«.

Die Rückmeldungen auf bisherige Bücher haben gezeigt, dass ein großes Interesse daran besteht, Bildschirmmasken und Meldungen möglichst in deutscher Sprache darzustellen. Wir werden deshalb, wo immer es möglich ist, die deutsche Sprache verwenden und gleichzeitig auf die englischen Begriffe hinweisen, um die Verbindung zur offiziellen Dokumentation und den vielen Blogs und White Papers wiederherzustellen.

Software und Skripte

Sie können die Oracle-Software aus dem Internet herunterladen und unter Beachtung der Lizenzbedingungen der Firma Oracle benutzen oder mit einer Cloud-Version der Oracle-Datenbank arbeiten. Aktuell (Stand: Dezember 2020) gibt es ein kostenloses Angebot für die Nutzung einer auf 30 Tage limitierten Version einer Oracle-Datenbank in der Public Cloud: <https://www.oracle.com/de/cloud/free>. Sie können mit einem »Always Free Checker«

arbeiten, um Kosten durch versehentlich aktivierte Features zu vermeiden. Details zu den einzelnen Komponenten finden Sie in den entsprechenden Abschnitten im Buch.

Alle im Buch gedruckten Skripte und Programme können Sie von der Website des Verlags sowie von der Autoren-Website herunterladen:

www.mitp.de/0057

www.lutzfroehlich.de

Über den Autor

Lutz Fröhlich ist Diplommathematiker und Oracle Certified Master sowie erfolgreicher Autor von anderen Fachbüchern und Veröffentlichungen. Fröhlich arbeitet seit 1993 mit Oracle-Datenbanken und ist spezialisiert auf die Themen Performance, Hochverfügbarkeit, Datenreplikation und -Streaming sowie Exadata und arbeitet seit mehreren Jahren in den Bereichen Data Science und Maschinelles Lernen.

Er hält regelmäßig Seminare und Vorträge zu diesen und anderen Themen. Seine praktischen Erfahrungen basieren auf Consulting-Tätigkeiten für über 35 internationale Unternehmen in den USA und Europa.

Autor und Verlag freuen sich über Feedback zum Buch.

Darmstadt, im Dezember 2020

Lutz Fröhlich

lutz@lutzfroehlich.de

Die Oracle-Datenbankarchitektur

Die Kenntnis der Oracle-Datenbankarchitektur ist für den Administrator sehr wichtig. Nur wer den Aufbau, die Zusammenhänge sowie die internen Prozessabläufe kennt, ist in der Lage, Architekturen zu planen und zu implementieren, die täglichen Supportanforderungen zu meistern und Troubleshooting zu betreiben.

Die Architektur der Oracle-Datenbank ist sehr komplex, und es ist eine längere praktische Erfahrung erforderlich, um sie in ihrer Gesamtheit kennenzulernen und zu beherrschen. Je länger und intensiver Sie sich mit dem Thema beschäftigen, desto umfangreicher wird Ihr Wissen und desto plausibler werden die Zusammenhänge.

2.1 Übersicht über die Architektur

Die wohl am häufigsten verwendeten Begriffe sind *Datenbank* und *Instanz*. Die Datenbank ist die Zusammenfassung aller Dateien wie Datafiles, Kontrolldateien, Log-Dateien, SPFILE usw., also alle Objekte, die sich auf der Disk befinden. Dagegen beschreibt die Instanz alle Strukturen, die sich im Hauptspeicher befinden, wie z.B. Buffer Cache, Shared Pool oder die Hintergrundprozesse. Zu jeder Datenbank gehört mindestens eine Instanz. In einer Real-Application-Clusters-Umgebung benutzen mehrere Instanzen dieselbe Datenbank.

2.1.1 Die Struktur der Datenbank

Eine Oracle-Datenbank wird nach logischen und physischen Strukturen unterschieden. Logische Strukturen sind z.B. Schemata oder Tablespaces. Physische Strukturen sind Dateien und Einheiten, die auf Betriebssystemebene sichtbar und greifbar sind. Dazu gehören Datafiles, Kontrolldateien oder Datenblöcke.

Die Dateien der Datenbank können im Dateisystem des Datenbankservers, in einem Cluster-Dateisystem, im Oracle-Automatic-Storage-Management (ASM) oder als Raw Devices gespeichert werden.

Zu den grundlegenden Strukturen gehört Folgendes:

- *Tablespaces* sind Container für Objekte wie Tabellen oder Indexe. Diese Objekte können nach verschiedenen Kriterien wie Applikationslogik oder Performance gezielt in verschiedene Tablespaces gelegt werden. In der SYSTEM-Tablespace befindet sich unter anderem der Datenbankkatalog. Die SYSAUX-Tablespace ist für Repositories oder Tabellen von Werkzeugen vorgesehen. Die TEMPORARY-Tablespace dient der Aufnahme von temporären Segmenten. Eine Tablespace kann aus einem oder mehreren Datafiles bestehen.
- *Datenblöcke* sind die kleinste Speichereinheit einer Oracle-Datenbank. Zulässig in Oracle 19c sind Größen von 2, 4, 8, 16 und 32 KB.
- *Extents* sind Gruppen von Datenblöcken, die zusammenhängend gespeichert werden.

- *Segmente* sind Gruppen von Extents, die eine logische Struktur bilden. Alle Extents einer Tabelle, eines Index oder eines Clusters bilden ein Segment.

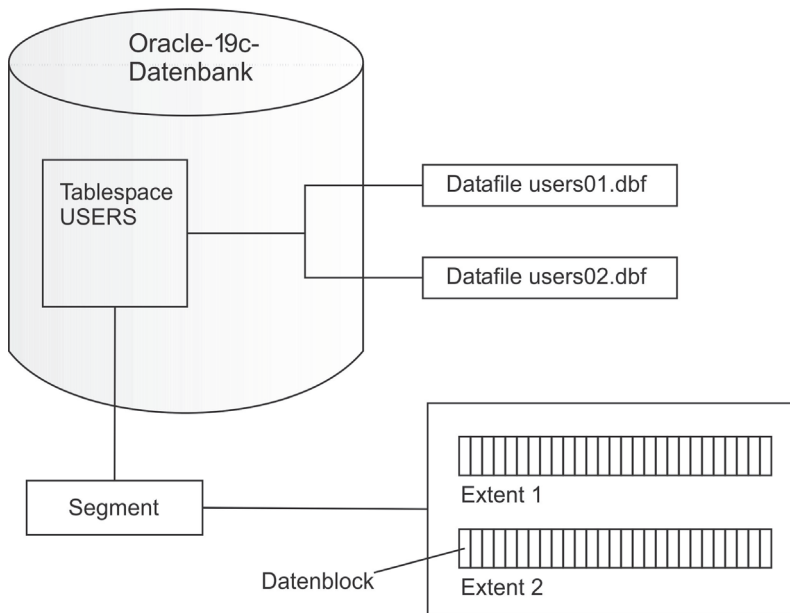


Abb. 2.1: Die Struktur der Oracle-Datenbank

Zu einer Datenbank gehören folgende Dateitypen:

- Datafiles
- Tempfiles
- Kontrolldateien
- Online-Redo-Log-Dateien
- Server-Parameter-File (SPFILE)
- Passwordfile
- Archived-Redo-Log-Dateien (optional)
- Flashback-Log-Dateien (optional)
- Block-Change-Tracking-File (optional)
- Alertlog- und Incident-Dateien

Datafiles sind die physische Umsetzung von Tablespaces. Eine Tablespace besteht aus einem oder mehreren Datafiles. Ein Datafile ist genau einer Tablespace zugeordnet.

Tempfiles sind Dateien, die temporäre Tablespaces verkörpern. Sie enthalten bei geschlossener Datenbank keine relevanten Daten und werden beim Öffnen der Datenbank automatisch neu angelegt, falls sie nicht existieren.

Die *Kontrolldatei* speichert Informationen über die physische Datenbankstruktur. Sie enthält unter anderem den Datenbanknamen, die Datenbankidentifizierungsnummer (DBID) und

Namen, Speicherort sowie Status der Datafiles, Tempfiles und Online-Redo-Log-Dateien. Aus Sicherheitsgründen sollte mindestens ein Spiegel der Kontrolldatei angelegt werden.

Online-Redo-Log-Dateien sind das Transaktionslog der Datenbank. Sie garantieren die Transaktionssicherheit und werden für Recovery-Zwecke benötigt.

Im *Server-Parameter-File* befinden sich die Initialisierungsparameter der Datenbank.

Das *Passwordfile* enthält die Passwörter der privilegierten Benutzer. Es wird benötigt, wenn die Datenbank nicht geöffnet ist.

Archived-Redo-Log-Dateien sind archivierte Online-Redo-Log-Dateien. Sie gewährleisten die Wiederherstellung der Datenbank zu einem beliebigen Zeitpunkt, da Online-Redo-Log-Dateien zyklisch überschrieben werden. Archived-Redo-Log-Dateien werden geschrieben, wenn die Datenbank im ARCHIVELOG-Modus betrieben wird.

Flashback-Log-Dateien sind optional und werden nur erzeugt, wenn das Flashback-Database-Feature aktiviert ist. Mit ihrer Hilfe ist es möglich, die Datenbank auf einen beliebigen Zeitpunkt in der Vergangenheit zurückzusetzen, ohne dass eine Sicherung eingespielt werden muss.

Das *Block-Change-Tracking-File* speichert Informationen über die Datenblöcke, die seit der letzten Vollsicherung geändert wurden, und dient als Index für inkrementelle Sicherungen. Es muss explizit aktiviert werden.

In Oracle 19c gibt es elf Segmenttypen. Der größte Anteil fällt auf Daten- und Indexsegmente sowie UNDO- und TEMP-Segmente. Weiterhin gibt es spezielle Segmente für Large Objects, Cluster und Partitionen. Die Segmenttypen sind:

- CLUSTER
- INDEX
- INDEX PARTITION
- LOB PARTITION
- LOB INDEX
- LOB SEGMENT
- NESTED TABLE
- ROLLBACK SEGMENT
- TABLE
- TABLE PARTITION
- TYPE2 UNDO

Oracle-Datenblöcke

Der Oracle-Datenblock ist die kleinste Einheit in der Hierarchie der Speicherstrukturen. Er enthält die Sätze von Tabellen, Indexeinträge oder Large Objects (LOB). Die Standardgröße der Datenbank wird beim Erstellen festgelegt und kann anschließend nicht mehr geändert werden. Es besteht jedoch die Möglichkeit, Tablespace mit unterschiedlichen Blockgrößen anzulegen. Viele Systeme benutzen eine Blockgröße von 4 KB oder 8 KB. Für große oder Data-Warehouse-Datenbanken ist eine Blockgröße von 16 KB empfohlen, um einen Performance-Gewinn insbesondere beim Lesen der Daten zu erzielen. Die Blockgröße der Datenbank sollte stets ein ganzzahliges Vielfaches der Blockgröße des Betriebssystems sein. Bei

der Wahl der Blockgröße ist außerdem zu beachten, dass es bei einer zu hoch gewählten Größe zu Konkurrenzsituationen und damit zu Wartezeiten kommen kann, da sich die meisten internen Operationen auf Blockebene abspielen.

Ein normaler, unkomprimierter Datenblock besteht aus einem Kopf (Blockheader) und einem Rumpf.

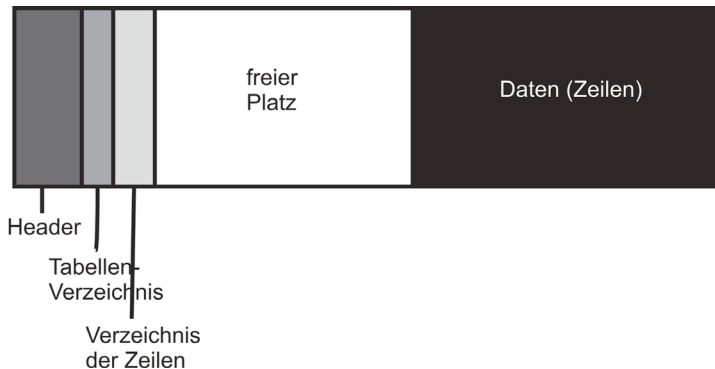


Abb. 2.2: Aufbau eines Datenblocks

Der Header enthält allgemeine Information wie die Speicheradresse auf der Disk und den Blocktyp. Blocktypen, die der Transaktionsverwaltung unterliegen, enthalten zusätzlich historische Informationen zu den durchgeführten Transaktionen. Im Tabellenverzeichnis befinden sich Informationen über die Tabellen der gespeicherten Sätze. Das Zeilenverzeichnis beschreibt die Positionen der Sätze (Zeilen), die im Block gespeichert sind.

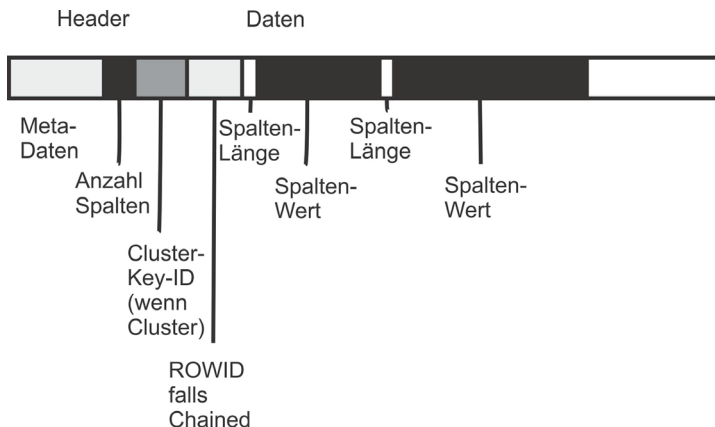


Abb. 2.3: Struktur eines Datensatzes im Block

Passt ein Datensatz nicht komplett in einen Block, dann wird er auf zwei Blöcke aufgeteilt. Man spricht dann von *Chained Rows*. Die ROWID des Blocks, in dem der Datensatz fortgesetzt wird, verweist dann auf den nächsten Block. Chained Rows wirken sich negativ auf die Performance aus und sollten im großen Umfang vermieden werden.

Wenn Sie sich einen Datenblock etwas genauer anschauen möchten, besteht die Möglichkeit, einen Dump zu erzeugen. Im Beispiel in Listing 2.1 wird eine kleine Tabelle mit nur einem Satz verwendet. Die SQL-Abfrage liefert die Blocknummer 52608 zurück. Dort befindet sich der Segment-Header. Der folgende Block ist der erste Datenblock. Die Ausgabe des Dumpfiles erfolgt im Diagnostic-Verzeichnis der Datenbank, da, wo die Alert-Log-Datei liegt.

```
SQL> SELECT file_id, block_id, blocks
  2 FROM dba_extents
  3 WHERE segment_name = 'TEST';
  FILE_ID  BLOCK_ID    BLOCKS
  -----  -
           1      52608         8
SQL> ALTER SYSTEM DUMP DATAFILE 1 BLOCK 52609;
System wurde geändert.
```

Listing 2.1: Dump für einen Datenblock erstellen

Neben vielen interessanten Informationen aus dem Block-Header liefert die Dump-Datei den kompletten Inhalt des Blocks.

```
Dump of memory from 0x00007FA35294EE00 to 0x00007FA352950E00
7FA35294EE00 0000A206 0040CD81 000DB189 06010000
[.....@.....]
7FA35294EE10 0000C4A5 00000001 00005B01 000DB17C
[.....[...]]
7FA35294EE20 00000000 00030002 00000000 00030004
[.....]
7FA35294EE30 000002F8 01416DA1 00210089 00002001
[.....mA...!.. ..]
7FA35294EE40 000DB189 00000000 00000000 00000000
[.....]
7FA35294EE50 00000000 00000000 00000000 00010100
[.....]
7FA35294EE60 0014FFFF 1F5C1F70 00001F5C 1F700001
[...p.\.\....p.]
7FA35294EE70 00000000 00000000 00000000 00000000
[.....]
          Repeat 500 times
7FA352950DC0 00000000 00000000 00000000 0202012C
[.....,...]
7FA352950DD0 412902C1 41414141 41414141 41414141
[..)AAAAAAAAAAAA]
7FA352950DE0 41414141 41414141 41414141 41414141
```

```
[AAAAAAAAAAAAAAAA]
7FA352950DF0 41414141 41414141 41414141 B1890601
[AAAAAAAAAAAAA...]
```

Listing 2.2: Dump eines Datenblocks

Über die ROWID kann jeder Satz in der Datenbank eindeutig identifiziert werden. Sie enthält alle Informationen, die Oracle benötigt, um schnell auf den Satz zugreifen zu können. Sie besitzt das folgende Format:

```
000000 FFF BBBB RRR
```

Dabei ist:

- 000000: Die Objekt-ID der Tabelle oder des Objekts
- FFF: Die Nummer des Datafiles
- BBBB: Die Blocknummer im Datafile
- RRR: Die Zeilennummer im Block

Die ROWID ist eine implizite Spalte und kann mit SQL-Mitteln abgefragt werden:

```
SQL> SELECT rowid,dummy FROM dual;
ROWID          D
-----
AAAACMAABAAAV5AAA X
```

Oracle beginnt mit dem Füllen eines Datenblocks am Ende. Dabei reduziert sich der freie Platz zwischen Daten und Header (siehe Abbildung 2.2). Um Row Chaining und Row Migration zu vermeiden, behält jeder Block einen freien Platz für Updates. Dieser freie Platz wird durch den Storage-Parameter PCTFREE festgesetzt. Hier gilt es, einen Kompromiss zwischen Platzverschwendung und genügend Freiraum für Updates zu finden. Der Standardwert ist 10 Prozent.

Extents

Ein Extent besteht aus einem zusammenhängenden Bereich von Datenblöcken. Neue Extents werden automatisch angelegt, wenn das Segment wächst, so lange, bis eine Grenze erreicht wird. Extents, die von einem Segment belegt wurden, werden nicht automatisch wieder zurückgegeben, auch wenn sie leer sein sollten. Sie werden erst mit dem Löschen des Objekts wieder freigegeben.

Segments

Ein Segment ist eine Zusammenfassung von Extents und repräsentiert ein Objekt in der Datenbank, das mit Daten gefüllt werden kann. So entspricht zum Beispiel eine Tabelle einem Segment.

Ein Segment wird nicht zwangsläufig beim Erstellen eines Objekts angelegt. Für Tabellen, Indexe sowie Partitionen gilt, dass zunächst nur die Metadaten im Datenbankkatalog erstellt werden. Sobald der erste Datensatz eingefügt wird, erfolgt das Anlegen des Segments. Dieses Feature wird als *Deferred Segment Creation* bezeichnet.

Oracle verwaltet den Platz innerhalb eines Segments mithilfe des High Water Mark (HWM). Die Blöcke, die sich oberhalb des HWM befinden, sind unformatiert und wurden noch nicht benutzt. Blöcke unterhalb des HWM können durchaus leer sein, zum Beispiel weil Daten gelöscht wurden.

Tablespaces

Eine Tablespace ist ein logischer Container für Segmente, also Tabellen, Indexe, Cluster, LOBs. Die SYSTEM-Tablespace enthält als wichtigste Komponente den Datenbankkatalog. Eigentümer des Katalogs ist der Benutzer SYS. Die SYSTEM-Tablespace wird mit dem Erstellen der Datenbank automatisch angelegt.

Die Tablespace SYSAUX enthält Schemata von Oracle-Komponenten und kann für weitere Werkzeuge oder Komponenten verwendet werden, die ein Repository benötigen.

In der UNDO-Tablespace werden UNDO-Segmente gespeichert. Diese werden für das Zurückrollen von Transaktionen, die Lesekonsistenz und Flashback-Operationen benötigt.

Eine temporäre Tablespace enthält temporäre Segmente, die für die Gültigkeitsdauer einer Session benötigt werden. Dies ist der Fall bei größeren Sortieroperationen oder beim Anlegen von temporären Tabellen.

```
SQL> SELECT tablespace_name, file_name
       2 FROM dba_data_files
       3 ORDER BY 1,2;
TABLESPACE_NAME  FILE_NAME
-----
SYSAUX           +DATA/mitp/datafile/sysaux.264.818620179
SYSTEM           +DATA/mitp/datafile/system.258.818620149
UNDOTBS1         +DATA/mitp/datafile/undotbs1.261.818620197
USERS            +DATA/mitp/datafile/users.267.818620231
```

Listing 2.3: Permanente Tablespaces und Datafiles anzeigen

Es gibt zwei grundlegende Arten von Tablespaces:

- Locally Managed Tablespaces
- Dictionary Managed Tablespaces

Der Standardtyp ist die Locally Managed Tablespace. Die Verwaltung des freien Platzes erfolgt über ein Bitmap im Header des Datafiles. Segmente innerhalb der Tablespace können automatisch oder manuell verwaltet werden. Das Automatic Segment Space Management (ASSM) ist der Standard, mit Ausnahme der Tablespaces SYSTEM, UNDO und TEMP. ASSM bietet neben einer besseren Performance den Vorteil, dass die manuelle Verwaltung der Storage-Parameter entfällt, was die Administration vereinfacht.

Die Dictionary Managed Tablespace verwendet den Datenbankkatalog zur Verwaltung der Extents. Er war in früheren Versionen der Standardtyp und wurde durch die Locally Managed Tablespace abgelöst, um eine bessere Performance zu erzielen und Konflikte im Katalog zu vermeiden. Dictionary Managed Tablespaces können aus Kompatibilitätsgründen noch verwendet werden.

Mit der Version 10g wurde ein weiterer Tablespace-Typ eingeführt: die Bigfile Tablespace. Dies war vor allem der zunehmenden Größe von Datenbanken geschuldet. Das Datafile einer Smallfile Tablespace (Default) kann aus maximal 4 Millionen Datenblöcken bestehen. Bei einer Blockgröße von 8 KB bedeutet das, dass die maximale Größe eines Datafiles nicht größer als 32 GB werden kann. Das Datafile einer Bigfile Tablespace kann bei einer Blockgröße von 8 KB immerhin 32 TB groß werden. Allerdings darf es nicht mehr als ein Datafile in der Bigfile Tablespace geben.

Hinweis

Weitere Informationen zur Administration und Verwaltung von Speicherstrukturen sowie zum Thema »Fragmentierung« finden Sie in Kapitel 3, »Interne Strukturen und Objekte«.

Redo-Log-Dateien

Die Redo-Log-Dateien bilden das Transaktionslog der Datenbank. Änderungen in der Datenbank landen nach Abschluss einer Transaktion (COMMIT) in der Regel nicht auf der Disk, sondern bleiben bis zum nächsten Checkpoint im Buffer Cache der Datenbank. Sie werden jedoch in die Redo-Log-Dateien geschrieben, sodass bei einem Datenbank-Crash alle abgeschlossenen Transaktionen wiederhergestellt werden können. Online-Redo-Log-Dateien sind also kritisch für die Wiederherstellbarkeit der Datenbank.

Die Online-Redo-Log-Dateien bestehen aus mehreren Gruppen. Jede Gruppe kann mehrere Member (Dateien) enthalten. Die Member einer Gruppe sind identisch gespiegelte Dateien und dienen der Ausfallsicherheit. Ist ein Member einer Gruppe korrupt oder versehentlich gelöscht worden, kann immer noch auf die übrigen Member zurückgegriffen werden. In der Regel verwendet man zwei Member pro Gruppe.

Der Log Writer schreibt immer in genau eine Gruppe, diese besitzt den Status CURRENT. Ist die Gruppe voll, erfolgt ein Log Switch. Dabei wechselt der Log Writer zur nächsten Gruppe und markiert diese als CURRENT. Ist der Log Writer bei der letzten Gruppe angekommen, beginnt er wieder mit der ersten und überschreibt diese. Läuft die Datenbank im Archive-log-Modus, wird die Gruppe vor dem Überschreiben archiviert. Es wird eine Kopie in Form einer Archived-Redo-Log-Datei erstellt. Damit garantiert Oracle durch die vorangegangene Sicherung eine Wiederherstellung zu einem beliebigen Zeitpunkt. Mit jedem Log Switch wird eine neue Sequence-Nummer erstellt.

```
SQL> SELECT group#,sequence#,status,first_time
2 FROM v$log;
GROUP# SEQUENCE# STATUS FIRST_TIME
-----
1 73 CURRENT 23.06.2019 20:00:51
```

2	71 INACTIVE	20.06.2019 21:00:16
3	72 INACTIVE	23.06.2019 18:49:32

Listing 2.4: Informationen über die Online-Redo-Log-Gruppen abfragen

Tipp

Standardmäßig werden vom DBCA drei Log-Gruppen mit einer Größe von 50 MB angelegt. Die Größe ist für viele Datenbanken zu klein und die Anzahl zu gering. Legen Sie Redo-Log-Dateien mit mindestens fünf Gruppen und einer Größe von 250 MB an. Aufgrund ihrer Kritikalität für die Wiederherstellbarkeit der Datenbank sollten die Gruppen mit mindestens zwei Mitgliedern (ein Spiegel) angelegt werden.

Kontrolldateien

In der Kontrolldatei befinden sich u.a. die Informationen über alle Dateien, die unmittelbar zur Datenbank gehören. Das sind Datafiles, Tempfiles und Online-Redo-Log-Dateien. Die Kontrolldatei ist sozusagen die Klammer, die die Datenbank zusammenhält. Damit ist sie sehr kritisch und sollte ebenfalls gespiegelt werden. Der Speicherort der Kontrolldateien wird durch den Datenbankparameter `control_files` festgelegt.

Server-Parameter-File (SPFILE)

Das SPFILE enthält die Werte aller Parameter der Datenbank, die sogenannten *Init-Parameter*. Es ist eine Binär-Datei und sie sollte nicht mit einem Editor bearbeitet werden, da sie beschädigt werden könnte. Ändern Sie Werte im SPFILE nur mit dem ALTER SYSTEM-Befehl oder erstellen Sie eine Datei im Textformat (PFILE) aus dem SPFILE.

Passwordfile

Auch Benutzer mit SYSDBA- oder SYSOPER-Privilegien identifizieren sich mithilfe des verschlüsselten Passworts im Datenbankkatalog. Ist die Datenbank nicht geöffnet, zum Beispiel vor dem Start der Instanz, kann der Katalog zur Prüfung des Passworts nicht herangezogen werden. In diesem Fall und wenn keine Identifizierung über das Betriebssystem erfolgt, wird das Passwordfile herangezogen.

Es befindet sich im Verzeichnis \$ORACLE_HOME/dbs (bzw. %ORACLE_HOME%\database unter Windows) und hat den Namen `orapw<SID>`. Erstellung und Bearbeitung erfolgt mit dem Werkzeug `orapwd`. Eine Änderung des Passworts mit dem ALTER USER-Kommando bewirkt, dass das neue Passwort sowohl im Datenbankkatalog als auch im Passwordfile gespeichert wird. Die SQL-Abfrage in Listing 2.5 liefert alle Benutzer, die im Passwordfile hinterlegt sind.

SQL> SELECT * FROM v\$pwfile_users;							
USERNAME	SYSDB	SYSOP	SYSAS	SYSBA	SYSDB	SYSKM	CON_ID
-----	-----	-----	-----	-----	-----	-----	-----
SYS	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	0
SYSDB	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	0

SYSBACKUP	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	0
SYSKM	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	0

Listing 2.5: Benutzer im Passwordfile abfragen

Archived-Redo-Log-Dateien

Im Archivelog-Modus werden die Online-Redo-Log-Dateien beim Log-Switch in Archived-Redo-Log-Dateien kopiert. Das Zielverzeichnis für die Dateien kann im Parameter `log_archive_dest_n` hinterlegt werden. Der Parameter kann zum Beispiel so aussehen:

```
log_archive_dest_1='LOCATION=/u01/oracle/archive'
```

Ist kein Verzeichnis als Ziel definiert, werden die Dateien in die Fast Recovery Area geschrieben, falls diese definiert ist. Sie werden dann im OMF-Format hinterlegt. Alle Informationen über die Archived-Redo-Log-Dateien finden Sie in der View `V$ARCHIVED_LOG`.

```
SQL> SELECT name,sequence# FROM v$archived_log;
NAME                                     SEQUENCE#
-----
/u01/oracle/fast_recovery_area/MITP/archivelog/201
4_01_06/o1_mf_1_33_9do1d2bn_.arc      33
```

Listing 2.6: Informationen über Archived-Redo-Log-Dateien im Katalog abfragen

Vorsicht

Wird im Verzeichnis für die Archived-Redo-Log-Dateien noch eine Fast Recovery Area definiert, werden die Dateien in das Verzeichnis `$ORACLE_HOME/dbs` (unter Windows `%ORACLE_HOME%\database`) geschrieben, ohne dass ein Fehler gemeldet wird. Die Größe des Dateisystems ist in der Regel nicht dafür ausgelegt. Ist das Dateisystem zu 100 % gefüllt, ist ein regulärer Betrieb der Datenbank nicht mehr möglich.

Flashback-Log-Dateien

Flashback-Log-Dateien dienen dem Zurücksetzen der Datenbank auf einen früheren Zustand. Sie werden mit dem Befehl `FLASHBACK DATABASE` angewandt und funktionieren ähnlich wie die Redo-Log-Dateien, nur in zeitlich umgekehrter Richtung. Sie werden standardmäßig nicht geschrieben und müssen durch den DBA aktiviert werden. Dafür gibt es zwei Optionen:

- Permanente Aktivierung der Flashback-Log-Dateien
- Erstellen eines garantierten Restore Points (GRP). Das Schreiben der Flashback-Log-Dateien wird dynamisch eingeschaltet, und nach dem Löschen des GRP wird abgestellt.

Voraussetzung in beiden Fällen ist, dass die Datenbank im ARCHIVELOG-Modus läuft.

Eine permanente Aktivierung erfolgt durch einen `ALTER DATABASE`-Befehl.

```
SQL> ALTER DATABASE FLASHBACK ON;
Datenbank wurde geändert.
```

Listing 2.7: Das Schreiben von Flashback-Log-Dateien aktivieren

Auf dieselbe Art kann das Schreiben der Log-Dateien deaktiviert werden. Das Ein- und Ausschalten kann dynamisch bei geöffneter Datenbank durchgeführt werden. Flashback-Log-Dateien werden in die Fast Recovery Area geschrieben. In diesem Zusammenhang ist der Parameter `DB_FLASHBACK_RETENTION_TARGET` zu beachten. Er garantiert, dass ältere Log-Dateien nicht vor Erreichen des Zeitraums gelöscht werden. Die Angabe erfolgt in Minuten.

Die zweite Option ist das Setzen eines garantierten Restore Point. In diesem Fall hat der Wert des Parameters `DB_FLASHBACK_RETENTION_TARGET` keinen Einfluss auf das Löschen der Flashback-Log-Dateien. Diese bleiben so lange erhalten, wie der Restore Point aktiv ist. Das Setzen und Löschen eines Restore Point kann ebenfalls dynamisch bei geöffneter Datenbank erfolgen. Mit dem Löschen des Restore Point werden alle zugehörigen Flashback-Log-Dateien gelöscht. Ein Beispiel für das Setzen und das Löschen finden Sie in Listing 2.8.

```
SQL> CREATE RESTORE POINT before_upgrade GUARANTEE FLASHBACK DATABASE;
Restore-Punkt erstellt.
SQL> SELECT name,time FROM v$restore_point;
NAME                                TIME
-----
BEFORE_UPGRADE    05-JAN-19 12.23.40.000000000 PM
SQL> DROP RESTORE POINT before_upgrade;
Restore-Punkt gelöscht.
```

Listing 2.8: Einen garantierten Restore Point setzen und löschen

Block-Change-Tracking-File

Mit einem Block-Change-Tracking-File (BCT-File) können Zeit- und Ressourcenverbrauch für eine inkrementelle Sicherung mit dem Recovery Manager deutlich reduziert werden. Normalerweise liest RMAN bei einer inkrementellen Sicherung jeden Datenblock und prüft, ob er seit der letzten Sicherung geändert wurde. Diese Methode führt zu einem hohen I/O-Aufkommen und einer Laufzeit, die sich nur unwesentlich von einer Vollsicherung unterscheidet.

Das Block-Change-Tracking-File ist ein Index der geänderten Blöcke. Ist es aktiviert, dann benutzt RMAN diesen Index und liest nur die geänderten Datenblöcke. Damit reduzieren sich die Sicherungszeiten auf ca. 10 % bis 20 % und das I/O-Aufkommen wird deutlich verringert. Das BCT-File ist standardmäßig deaktiviert. Die Aktivierung erfolgt durch ein `ALTER DATABASE`-Kommando.

```
SQL> ALTER DATABASE ENABLE BLOCK CHANGE TRACKING
2 USING FILE '/u01/oracle/bct/MITP_bct.bin';
```


Datenbank wurde geändert.

```
SQL> ALTER DATABASE DISABLE BLOCK CHANGE TRACKING;
```

Datenbank wurde geändert.

Listing 2.9: Das Schreiben eines Block-Change-Tracking-Files ein- und ausschalten

Das BCT-File ist eine Binärdatei und bleibt relativ klein. Seine Größe bewegt sich auch für größere Datenbanken im MB-Bereich. Es sollte vor einer Vollsicherung angelegt werden, damit es zur darauf folgenden inkrementellen Sicherung wirksam wird.

Diagnostic-Dateien

Die Funktionalität für die Diagnostik der Datenbank wurde mit der Version 11g wesentlich erweitert. Gleichzeitig wurde eine neue Struktur der zugehörigen Log-, Trace- und Diagnostic-Dateien eingeführt. Das Feature wird auch als *Advanced Diagnostic Repository* (ADR) bezeichnet.

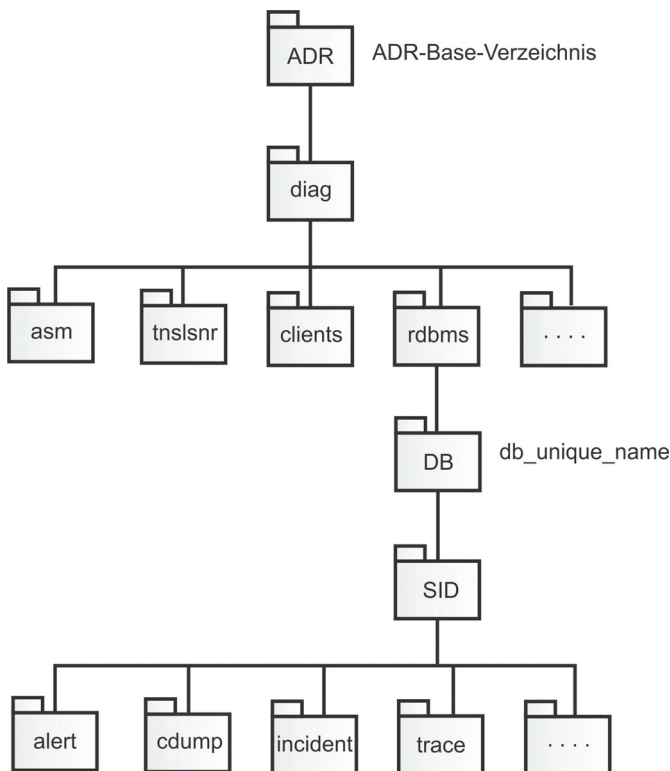


Abb. 2.4: Die ADR-Verzeichnisstruktur

Das ADR-Base-Verzeichnis wird durch den Datenbankparameter `DIAGNOSTIC_DEST` definiert. Die Struktur für die Diagnostic-Verzeichnisse der Datenbank kann mit der SQL-Anweisung in Listing 2.10 abgefragt werden.

```
SQL> SELECT name,value FROM v$diag_info;
NAME                                VALUE
-----
Diag Enabled                        TRUE
ADR Base                           /u01/oracle
ADR Home                           /u01/oracle/diag/rdbms/doag/DOAG
Diag Trace                         /u01/oracle/diag/rdbms/doag/DOAG/trace
Diag Alert                         /u01/oracle/diag/rdbms/doag/DOAG/alert
Diag Incident                      /u01/oracle/diag/rdbms/doag/DOAG/incident
Diag Cdump                        /u01/oracle/diag/rdbms/doag/DOAG/cdump
Health Monitor                    /u01/oracle/diag/rdbms/doag/DOAG/hm
Default Trace File                 /u01/oracle/diag/rdbms/doag/DOAG/trace/DO
AG_ora_13775.trc
Active Problem Count              0
Active Incident Count            0
```

Listing 2.10: ADR-Informationen abfragen

Die Navigation durch die Verzeichnisse ist zeitaufwendiger gegenüber der früheren Struktur. Eine Vereinfachung liefert das Kommandozeilenwerkzeug `adrci`. Das Beispiel in Listing 2.11 zeigt, wie ein schneller Blick in das Alert-Log der Datenbank möglich ist. Für eine laufende Tail-Anzeige können Sie das Kommando `adrci> SHOW ALERT -TAIL -F` verwenden.

```
$ adrci
ADRCI: Release 19.0.0.0.0 - Production on Mi Nov 27 14:32:37 2019
Copyright (c) 1982, 2019, Oracle and/or its affiliates.
All rights reserved.
ADR base = "/u01/oracle"
adrci> SHOW ALERT
Choose the home from which to view the alert log:
1: diag/rdbms/mitp/MITP
2: diag/rdbms/test/TEST
3: diag/tnslsnr/serv7/listener
4: diag/tnslsnr/serv7/listener0
Q: to quit
Please select option: 1
. . .
```

Listing 2.11: Zugriff auf die Alert-Log-Datei mit `adrci`

2.1.2 Die Struktur der Instanz

Das Hochfahren der Instanz ist der erste Schritt beim Starten einer Datenbank. Wenn Sie im `STARTUP`-Befehl die `NOMOUNT`-Option verwenden, wird nur die Instanz gestartet, ohne die Kontrolldatei und die Datafiles zu öffnen. Oracle liest die Initialisierungsparameter aus dem `SPFILE`, initialisiert die Hauptspeicherstrukturen des Memory (SGA und PGA) und startet die Hintergrundprozesse.

Hinweis

Hintergrundprozesse sind unter Unix und Windows unterschiedlich implementiert, bedingt durch die unterschiedliche Architektur der Betriebssysteme. Während unter Unix jeder Hintergrundprozess ein einzelner unabhängiger Prozess ist, wird unter Windows ein Thread unter dem Hauptprogramm `oracle.exe` gebildet.

Der Hauptbestandteil der Instanz ist die *System Global Area* (SGA), die sich im Shared Memory befindet. Die SGA enthält Daten und Buffer, die datenbankweit von allen Sessions (Benutzern) gemeinsam benutzt werden. Ein weiterer Bestandteil des Memory ist die *Program Global Area* (PGA). In ihr befinden sich sitzungsspezifische Informationen, die nicht mit anderen Sessions geteilt werden. Deshalb wird die PGA auch als *Private Memory* bezeichnet.

Hinweis

Traditionell wurde im Betriebssystem zwischen Shared und Private Memory unterschieden und die Bereichsgrößen sind fix definiert. Von diesem Prinzip ist man abgekommen. Es wird nur noch ein Hauptspeicherbereich erstellt, der sowohl vom Private Memory als auch vom Shared Memory der Datenbank benutzt wird. Aus Sicht der Datenbankarchitektur ist die Unterscheidung nach wie vor gerechtfertigt.

Die System Global Area besteht im Wesentlichen aus folgenden Komponenten:

- Database Buffer Cache
- Redo Log Buffer
- Shared Pool (u.a. Library Cache, Dictionary Cache)
- Large Pool
- Java Pool
- Streams Pool

Im *Buffer Cache* werden Kopien der Datenblöcke gespeichert, um einen schnellen Zugriff auf deren Inhalt zu ermöglichen. Im Buffer Cache gibt es drei verschiedene Pools:

- Der *Default Pool* ist der Bereich, in dem alle Datenblöcke standardmäßig gespeichert werden.
- Sie können dem *Keep Pool* Segmente zuweisen, die regelmäßig frequentiert werden. Der Mechanismus im Default Pool würde diese Datenblöcke regelmäßig auslagern.
- Der *Recycle Pool* ist für große Segmente gedacht, die selten angefordert werden. Diese würden im Default Pool häufiger benötigte Segmente verdrängen.

System Global Area

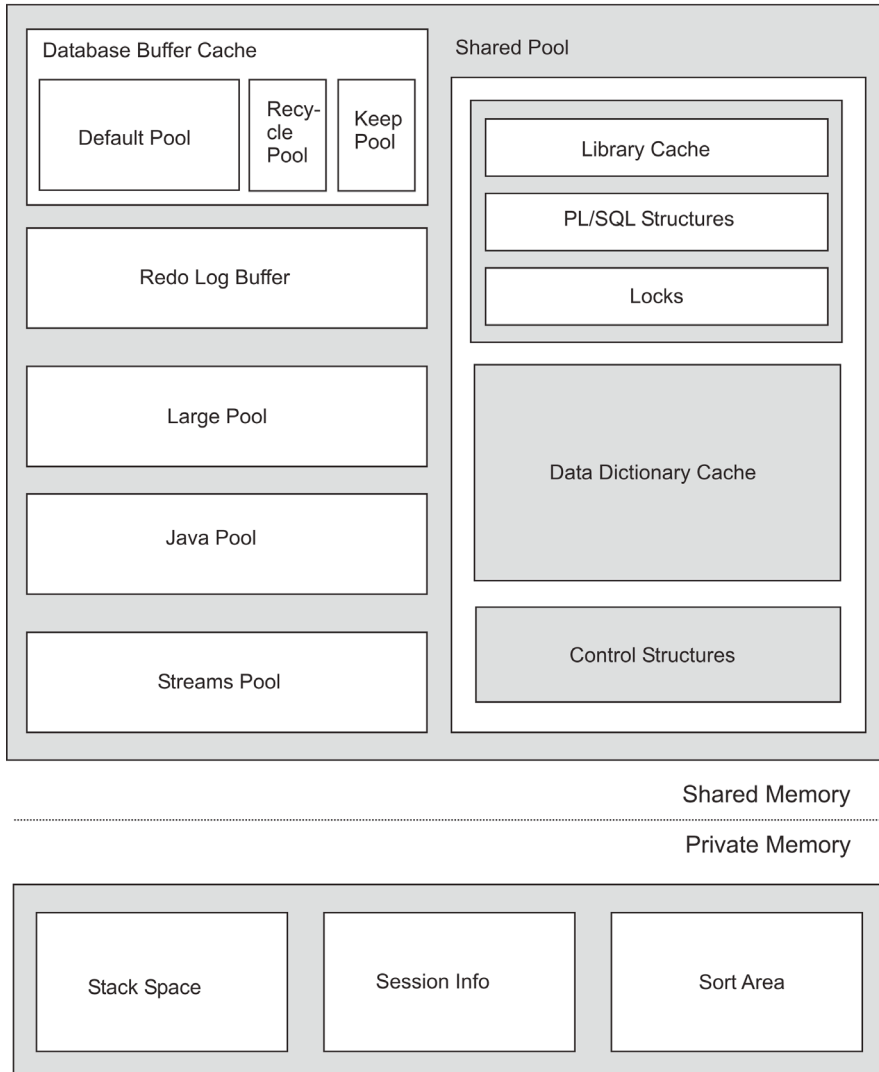


Abb. 2.5: Die Architektur der Oracle-Instanz

Datenblöcke im Buffer Cache können die folgenden drei verschiedenen Charakteristiken annehmen:

- *Dirty Buffer* sind Datenblöcke, die in die Datafiles geschrieben werden müssen, da sie gegenüber dem Original in der Datei verändert wurden.
- *Free Buffer* enthalten keine Daten oder stehen zum Überschreiben zur Verfügung. Sie werden mit Blöcken gefüllt, die von der Disk gelesen werden.
- *Pinned Buffer* sind Blöcke, die gerade benutzt werden oder für zukünftige Benutzung reserviert sind.

Oracle verwendet zwei Listen, um den Buffer Cache zu verwalten. Die *Write List*, auch *Dirty Buffer List* genannt, registriert alle Blöcke, die verändert wurden und auf Disk geschrieben werden müssen. Die *LRU-Liste* (Least Recently Used List) enthält *Free Buffer*, *Pinned Buffer* und *Dirty Buffer*, die noch nicht auf der Write List stehen. In der LRU-Liste stehen die zuletzt am häufigsten benutzten Blöcke vorn.

Wenn ein Prozess auf einen Datenblock zugreift, wird er an den Anfang der LRU-Liste gesetzt. Gleichzeitig wird ein Block am Ende der Liste hinausgeschoben. Mit diesem Mechanismus wird garantiert, dass sich die am häufigsten benutzten Datenblöcke im Buffer Cache befinden.

Hinweis

Es gibt eine Ausnahme von dieser Vorgehensweise. Wird ein Full Table Scan durchgeführt, werden die gelesenen Blöcke nicht an den Anfang, sondern an das Ende der LRU-Liste geschrieben. Damit wird verhindert, dass andere häufig verwendete Blöcke durch Full Table Scans von der LRU-Liste verdrängt werden.

Wenn ein Datenblock von einem Oracle-Prozess angefordert wird, wird zuerst geprüft, ob sich der Block im Buffer Cache befindet. Ist der Block im Cache vorhanden, dann nennt man diese Situation einen *Cache Hit*. Andernfalls muss er von der Disk gelesen werden. Dies nennt man einen *Cache Miss*.

Bevor Oracle einen Datenblock von der Disk in den Buffer Cache laden kann, muss ein freier Buffer gefunden werden. Der Prozess sucht, bis er einen freien Buffer gefunden hat oder bis ein Schwellenwert erreicht ist. Findet der Prozess während des Suchlaufs einen Dirty Buffer, entfernt er diesen von der LRU-Liste und trägt ihn in die Write-Liste ein. Anschließend sucht er weiter. Wird ein freier Buffer gefunden, lädt der Prozess den Datenblock von der Disk und trägt ihn in die LRU-Liste ein.

Hat der Prozess bei Erreichen des Schwellenwerts keinen freien Buffer gefunden, dann hört er auf zu suchen und signalisiert die Situation dem *Database-Writer-Prozess*. Der Database Writer fängt an, Dirty Buffer auf Disk zu schreiben, womit wieder freie Buffer entstehen.

Hinweis

Seit der Version 10g ist es möglich, verschiedene Blockgrößen innerhalb einer Datenbank zu verwalten. Für jede Blockgröße muss ein eigener Buffer Cache bereitgestellt werden. Die Blockgröße der SYSTEM-Tablespace bestimmt die Standardblockgröße der Datenbank.

Im *Redo Log Buffer* befinden sich die Daten, die durch den *Log-Writer-Prozess* (LGWR) in die Online-Redo-Log-Dateien geschrieben werden. Redo-Log-Daten werden zur Wiederherstellung von Transaktionen verwendet.

Der *Library Cache* enthält SQL-Anweisungen, PL/SQL-Prozeduren und Kontrollstrukturen wie Locking-Informationen. Befindet sich eine SQL-Anweisung in der Shared SQL Area, wurde sie bereits geparkt und kann von anderen Sessions direkt ausgeführt werden. Oracle behandelt PL/SQL-Strukturen ähnlich wie SQL-Anweisungen. Sie werden von der PL/SQL Engine ausgeführt.

Der *Large Pool* ist ein optionaler Bereich der SGA und erfüllt eine ähnliche Funktion wie der Shared Pool mit der Einschränkung, dass nur bestimmte Typen und Größen von Shared Memory zugewiesen werden können. Der Hauptspeicher für den Large Pool wird direkt aus der SGA zugewiesen. Der Large Pool wird unter anderem für folgende Operationen verwendet:

- Session-Informationen des Shared Server
- I/O-Server-Prozesse
- Backup- und Restore-Operationen
- Message Buffer für parallele Abfragen

Der *Streams Pool* wurde in Oracle 10g eingeführt. Er dient der Speicherung von *Buffered Queues*. Buffered Queues werden vorwiegend von Oracle Streams benutzt und besitzen signifikante Performance-Vorteile gegenüber herkömmlichen Queues.

Die *Program Global Area* (PGA) und die *User Global Area* (UGA) befinden sich im Private Memory. Andere Sitzungen haben keinen Zugriff darauf. Während die PGA im Wesentlichen die Sort Area und den Stack Space enthält, werden in der UGA die Statusinformationen der Session gespeichert.

2.1.3 Automatic Memory Management (AMM)

Das Automatic Memory Management wurde in Oracle 10g eingeführt. Es war da noch auf den Shared Memory beschränkt. Seit Oracle 11g ist es möglich, auch den Private Memory einzubinden. Damit ist Oracle nicht nur in der Lage, sowohl den Shared als auch den Private Memory dynamisch zu verwalten, sondern auch Speicher zwischen beiden Bereichen dynamisch auszutauschen.

AMM wird von den Plattformen AIX, Solaris, HP-UX, Linux und Windows unterstützt. Für das AMM wurden zwei neue Init-Parameter eingeführt: `MEMORY_TARGET` und `MEMORY_MAX_TARGET`. Der Wert von `MEMORY_TARGET` kann bis zur Grenze `MEMORY_MAX_TARGET` dynamisch verändert werden. Um die Datenbank auf AMM einzustellen, ist es ausreichend, den Parameter `MEMORY_TARGET` auf den gewünschten Wert und die übrigen AMM-Parameter auf »null« zu setzen. Für die Parameter untereinander existieren die folgenden Abhängigkeiten:

- `MEMORY_TARGET` ist auf einen Wert ungleich »null« gesetzt (AMM eingeschaltet):
 - Wenn gleichzeitig `SGA_TARGET` und `PGA_AGGREGATE_TARGET` gesetzt sind, werden diese als die Minimalwerte für diese Bereiche angesehen.
 - Falls `SGA_TARGET` gesetzt und `PGA_AGGREGATE_TARGET` nicht gesetzt ist, werden beide Parameter durch AMM gesetzt. Initial wird `PGA_AGGREGATE_TARGET` auf die Differenz zwischen `MEMORY_TARGET` und `SGA_TARGET` gesetzt.
 - Ist `PGA_AGGREGATE_TARGET` gesetzt und `SGA_TARGET` nicht gesetzt, dann werden beide Parameter durch AMM getunt. Die initiale Größe für `SGA_TARGET` ist die Differenz zwischen `MEMORY_TARGET` und `PGA_AGGREGATE_TARGET`.
 - Sind weder `SGA_TARGET` noch `PGA_AGGREGATE_TARGET` gesetzt, dann werden beide Bereiche durch AMM ohne Minimalwert verwaltet. Beim Start der Instanz werden 60 % an die SGA und 40 % an die PGA vergeben.

- MEMORY_TARGET ist nicht oder auf »null« gesetzt.
 - Wenn SGA_TARGET gesetzt ist, werden die Pools der SGA durch AMM verwaltet, so wie das aus Oracle 10g bekannt ist. Die PGA wird durch AMM verwaltet, unabhängig davon, ob der Parameter PGA_AGGREGATE_TARGET gesetzt ist oder nicht.
 - Sind weder SGA_TARGET noch PGA_AGGREGATE_TARGET gesetzt, dann wird die PGA durch AMM verwaltet, die SGA jedoch nicht.
 - Ist nur MEMORY_MAX_TARGET gesetzt, wird MEMORY_TARGET auf »null« gesetzt, und das automatische Tuning für SGA und PGA ist ausgeschaltet.
 - Wenn SGA_MAX_SIZE nicht gesetzt ist, wird der Parameter intern auf MEMORY_MAX_TARGET gestellt.

Hinweis

Wenn Sie in einer Init-Parameterdatei die Zeile für MEMORY_MAX_TARGET weglassen und MEMORY_TARGET auf einen Wert größer »null« setzen, wird MEMORY_MAX_TARGET automatisch auf den Wert von MEMORY_TARGET gesetzt.

Mit den folgenden Schritten schalten Sie AMM für eine Oracle-19c-Datenbank ein:

1. Ermitteln Sie die aktuellen Werte für die Initialisierungsparameter SGA_TARGET und PGA_AGGREGATE_TARGET.

```
SQL> SHOW PARAMETER sga_target
NAME                                TYPE        VALUE
-----
sga_target                          big integer 268435456
SQL> SHOW PARAMETER pga_aggregate_target
NAME                                TYPE        VALUE
-----
pga_aggregate_target               big integer 536870912
```

2. Fragen Sie mit der folgenden SQL-Anweisung den Maximalwert ab, den die PGA seit dem Start der Datenbank benötigt hat.

```
SQL> SELECT value FROM v$pgastat
2  WHERE name = 'maximum PGA allocated';
VALUE
-----
98786304
```

3. Legen Sie den Wert für MEMORY_TARGET auf Basis der ermittelten Größen fest.
4. Setzen Sie die Parameter für das AMM.

```
SQL> ALTER SYSTEM SET memory_target=1328M SCOPE=spfile;
System altered.
```

```
SQL> ALTER SYSTEM SET pga_aggregate_target=0 SCOPE=spfile;
System altered.
SQL> ALTER SYSTEM SET sga_target=0 SCOPE=spfile;
System altered.
```

5. Führen Sie einen Neustart der Datenbank durch.

Nach der Aktivierung von AMM passt Oracle die Größen der Hauptspeicherbereiche jeweils dem aktuellen Workload auf der Datenbank an. Für den Datenbankadministrator stehen folgende Views zur Verfügung:

- **V\$MEMORY_DYNAMIC_COMPONENTS**: Enthält zusammengefasste Informationen über die Änderungen von Speichergrößen seit dem Start der Instanz.
- **V\$MEMORY_RESIZE_OPS**: Liefert eine Historie aller Operationen zur Änderung von Speichergrößen seit dem Start der Instanz.
- **V\$MEMORY_TARGET_ADVICE**: Erstellt eine Schätzung der Verbesserung der Datenbank-Performance in Abhängigkeit von der Größe des Gesamtspeichers für Oracle.

```
SQL> SELECT component, current_size, min_size, max_size
2 FROM v$memory_dynamic_components;
```

COMPONENT	CURRENT_SIZE	MIN_SIZE	MAX_SIZE
shared pool	234881024	234881024	234881024
large pool	16777216	16777216	16777216
java pool	16777216	16777216	16777216
streams pool	0	0	0
SGA Target	1056964608	1056964608	1056964608
DEFAULT buffer cache	771751936	771751936	771751936
KEEP buffer cache	0	0	0
RECYCLE buffer cache	0	0	0
DEFAULT 2K buffer cache	0	0	0
DEFAULT 4K buffer cache	0	0	0
DEFAULT 8K buffer cache	0	0	0
DEFAULT 16K buffer cache	0	0	0
DEFAULT 32K buffer cache	0	0	0
Shared IO Pool	0	0	0
PGA Target	335544320	335544320	335544320
ASM Buffer Cache	0	0	0

Listing 2.12: Zusammenfassung der Werte des AMM

```
SQL> SELECT parameter, initial_size, target_size, status,
2 start_time, end_time
3 FROM v$memory_resize_ops;
```


PARAMETER	INITIAL_	TARGET_SIZE	STATUS	START_TI	END_TIME
shared_pool_size	0	234881024	COMPLETE	12:13:01	12:13:01
db_cache_size	51936	771751936	COMPLETE	12:13:01	12:13:02
java_pool_size	0	16777216	COMPLETE	12:13:01	12:13:01
streams_pool_size	0	0	COMPLETE	12:13:01	12:13:01
sga_target	0	1056964608	COMPLETE	12:13:01	12:13:01
db_cache_size	0	771751936	COMPLETE	12:13:01	12:13:01
db_keep_cache_size	0	0	COMPLETE	12:13:01	12:13:01
db_recycle_cache_size	0	0	COMPLETE	12:13:01	12:13:01
db_2k_cache_size	0	0	COMPLETE	12:13:01	12:13:01
db_4k_cache_size	0	0	COMPLETE	12:13:01	12:13:01
db_8k_cache_size	0	0	COMPLETE	12:13:01	12:13:01
db_16k_cache_size	0	0	COMPLETE	12:13:01	12:13:01
db_32k_cache_size	0	0	COMPLETE	12:13:01	12:13:01
pga_aggregate_target	0	335544320	COMPLETE	12:13:01	12:13:01
db_cache_size	0	0	COMPLETE	12:13:01	12:13:01
large_pool_size	0	16777216	COMPLETE	12:13:01	12:13:01

Listing 2.13: Historie der durch den AMM vorgenommenen Veränderungen

```
SQL> SELECT * FROM v$memory_target_advice
2 ORDER BY memory_size;
```

MEMORY_SIZE	MEMORY_SIZE_FACTOR	ESTD_DB_TIME	ESTD_DB_TIME_FA	VERSION
664	.5	1007	1	0
996	.75	1007	1	0
1328	1	1007	1	0
1660	1.25	1007	1	0
1992	1.5	1007	1	0
2324	1.75	1007	1	0
2656	2	1007	1	0

Listing 2.14: Die Werte des AMM Advisor

Es stellt sich die Frage, wie Oracle den Hauptspeicher verwaltet und Speicherbereiche zwischen PGA und SGA austauscht. Wie ist der Hauptspeicher, hier am Beispiel eines Linux-Betriebssystems, bei gestarteter Instanz mit eingeschaltetem AMM konfiguriert? Schauen wir uns die Shared-Memory-Segmente an:

```
$ ipcs -m
```

----- Shared Memory Segments -----						
key	shmid	owner	perms	bytes	nattch	status
0xa5d6936c	229378	oracle	660	4096	0	

Die Shared-Memory-Segmente weisen nur eine Größe von 4 KB auf. Wie schafft es Oracle dann, ein zeitnahes Resizing der Bereiche vorzunehmen? Wie sieht das *Mapped Memory* des Database Writer aus?

```
$ pmap 'pgrep -f dbw'
8021:  ora_dbw0_MITP
. . .
20001000 16380K rwxS- /dev/shm/ora_MITP_229378_0
21000000 16384K rwxS- /dev/shm/ora_MITP_229378_1
22000000 16384K rwxS- /dev/shm/ora_MITP_229378_2
23000000 16384K rwxS- /dev/shm/ora_MITP_229378_3
24000000 16384K rwxS- /dev/shm/ora_MITP_229378_4
. . .
```

Listing 2.15: Der Mapped Memory des Database Writer

Hier wird offensichtlich, dass Oracle /dev/shm für die Implementierung von Shared Memory verwendet und dafür Segmente in der Größe von 16 MB verwendet. Oracle benutzt eine Segmentgröße von 4 MB, wenn MEMORY_MAX_TARGET kleiner als 1024 MB ist, sonst 16 MB.

Die Konfiguration der Instanz sieht wie folgt aus:

```
SQL> show parameter target
NAME                                TYPE        VALUE
-----
. . .
memory_max_target                   big integer 1328M
memory_target                       big integer 1328M
pga_aggregate_target               big integer 0
sga_target                         big integer 0
SQL> SELECT component, current_size
2 FROM v$memory_dynamic_components;
COMPONENT          CURRENT_SIZE
-----
shared pool        234881024
large pool         16777216
java pool          16777216
streams pool       0
SGA Target         822083584
DEFAULT buffer cache 536870912
PGA Target         335544320
. . .
```

AMM hat also ein SGA Target von 784 MB und ein PGA Target von 320 MB gesetzt. Der Wert für das SGA Target wird bestätigt durch den von Oracle aktuell benutzten Shared Memory:

```
$ df -k /dev/shm
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
none	1683404	802100	881304	48%	/dev/shm

Jetzt wird der Wert für die PGA auf 900 MB erhöht. Offensichtlich gibt Oracle den Speicher aus dem Shared-Memory-Bereich und verwendet ihn für die PGA als *Private Memory*.

```
SQL> ALTER SYSTEM SET PGA_AGGREGATE_TARGET=900M;
System altered.
$ df -k /dev/shm
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
none	1683404	195152	1488252	12%	/dev/shm

Listing 2.16: Vergrößerung der PGA auf Kosten von Shared Memory

Damit ist das Prinzip klar, wie Oracle die Speicherbereiche dynamisch zuweist und auch Hauptspeicher zwischen Shared Memory und Private Memory verschiebt.

Hinweis

Stellen Sie sicher, dass ein hinreichend großes temporäres Dateisystem auf `/dev/shm` gemountet ist. Andernfalls können Sie AMM nicht einsetzen und erhalten beim Start der Instanz die Fehlermeldung ORA-00845.

```
# umount /dev/shm
SQL> startup
ORA-00845: MEMORY_TARGET not supported on this system
```

Listing 2.17: Starten der Instanz ohne tmpfs

Sobald genügend tmpfs auf `/dev/shm` gemountet ist, lässt sich die Instanz mit eingeschaltetem AMM wieder normal starten.

```
# mount -t tmpfs shmfs -o size=1600m /dev/shm
SQL> startup
ORACLE instance started.
Total System Global Area 1389391872 bytes
. . .
```

Listing 2.18: tmpfs auf `/dev/shm` zuweisen

2.2 Prozesse und Abläufe

In den vorhergehenden Abschnitten haben Sie die Datenbankarchitektur unter den Blickwinkeln Datenbank und Instanz kennengelernt. Dahinter verbergen sich natürlich eine

ganze Reihe von komplexen Prozessen und Abläufen. Mit den wichtigsten wollen wir uns in diesem Abschnitt beschäftigen.

2.2.1 Die Oracle-Hintergrundprozesse

Die Hintergrundprozesse stellen die Verbindung zwischen den Dateien der Datenbank und den Hauptspeicherstrukturen der Instanz her. Sie sind das Gehirn des Datenbanksystems und steuern alle Abläufe. Auch wenn sie voneinander unabhängig laufen, findet eine Kommunikation zwischen den Prozessen statt. Mit dem Hochfahren der Instanz werden die Kernprozesse gestartet. Bestimmte Prozesse werden nur dann gestartet, wenn das zugehörige Feature aktiviert ist.

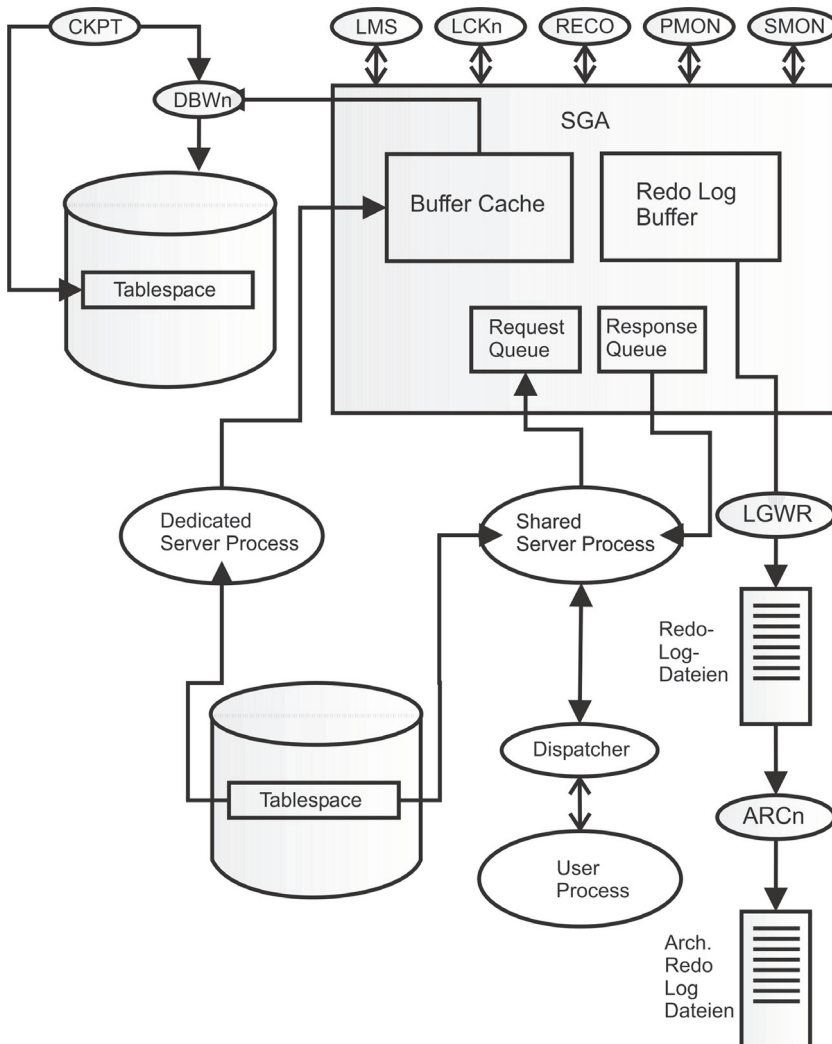


Abb. 2.6: Hintergrundprozesse der Oracle-Datenbank

In einer Client-Server-Umgebung wird für jeden Client, der eine Verbindung zur Datenbank herstellt, ein Client-Server-Prozess auf dem Datenbankserver gestartet. Diese Prozesse sind keine Hintergrundprozesse der Datenbank.

Jeder Hintergrundprozess übernimmt bestimmte Aufgaben im gesamten Prozessablauf der Datenbank. Nur wenn alle notwendigen Prozesse fehlerfrei laufen, ist die Datenbank voll funktionsfähig. Die wichtigsten Prozesse sind in der folgenden Liste beschrieben:

- Der *System Monitor* (SMON) führt beim Start der Instanz, falls erforderlich, ein Instance oder Crash Recovery durch. Weiterhin ist der SMON für das Löschen von temporären Segmenten, die nicht mehr benutzt werden, verantwortlich. Der SMON wacht regelmäßig auf und kontrolliert, ob er gebraucht wird. Andere Prozesse rufen ihn auf, wenn sie feststellen, dass er eine Aufgabe ausführen muss.
- Der *Process Monitor* (PMON) entfernt Benutzerprozesse, die nicht normal beendet wurden. Gleichzeitig gibt er alle von den Prozessen benutzten Ressourcen frei. So wie der SMON wacht der PMON regelmäßig auf und überprüft, ob Aufgaben zu erledigen sind.
- Der *Database Writer* (DBWn) ist verantwortlich für die Verwaltung des Buffer Cache. Seine Hauptaufgabe ist es, geänderte Datenblöcke aus dem Buffer Cache auf Disk zu schreiben. Standardmäßig schreibt er die Blöcke zuerst, die am Ende der LRU-Liste stehen. Zur Verbesserung der Performance können mehrere Database-Writer-Prozesse gestartet werden. Neben einer periodischen Überprüfung wird der Database Writer aus folgenden Situationen heraus aktiviert:
 - Ein Client-Server-Prozess kann keinen freien Buffer finden und erreicht den Schwellenwert für das Timeout. Daraufhin benachrichtigt er den Database Writer.
 - Wenn ein Checkpoint ausgeführt wird.
- Checkpoints zwingen den Database-Writer-Prozess, alle Dirty Buffer auf Disk zu schreiben. Der *Checkpoint Process* (CKPT) ändert die Header aller Datafiles und der Kontrolldateien und trägt die Checkpoint SCN ein. Ein Checkpoint wird durch folgende Situationen ausgelöst:
 - Jeder *Log Switch* der Online-Redo-Log-Dateien löst einen Checkpoint aus.
 - Die Initialisierungsparameter LOG_CHECKPOINT_INTERVAL, LOG_CHECKPOINT_TIMEOUT und FAST_START_MTTR_TARGET haben Einfluss auf die Häufigkeit von Checkpoints.
- Der *Log Writer* (LGWR) ist verantwortlich für die Verwaltung des Inhalts des Redo Log Buffer. Er schreibt die Redo Log Buffer in die Online-Redo-Log-Dateien auf Disk. Der Log Writer wird in den folgenden Situationen aktiviert:
 - Durch eine COMMIT-Anweisung in einer Session
 - Alle drei Sekunden
 - Wenn der Redo Log Buffer zu einem Drittel gefüllt ist
 - Wenn der Database Writer Datenblöcke auf die Disk schreibt
- Der *Recovery Process* (RECO) löst Fehler auf, die im Zusammenhang mit verteilten Transaktionen stehen. Er verbindet sich zu den Datenbanken, die in die Transaktionen eingebunden sind.
- Der *Archiver Process* (ARCn) erstellt Kopien der Online-Redo-Log-Dateien in Form von Archived-Redo-Log-Dateien. Dieser Prozess ist nur gestartet, wenn die Datenbank im ARCHIVELOG-Modus läuft. In einer Instanz können bis zu zehn ArchiveLog-Prozesse

(ARC0 bis ARC9) gestartet werden. Der Log Writer startet weitere Prozesse, wenn die aktuelle Anzahl nicht ausreicht.

- Der *Recovery Writer Process* (RVWR) schreibt die Flashback-Log-Dateien in der Flash Recovery Area. Er wird nur gestartet, wenn das Flashback-Database-Feature aktiviert ist.
- Der *Queue Monitor Process* (QMNn) ist beim Einsatz von Advanced Queuing ein optionaler Prozess. Er überwacht die Nachrichtenwarteschlangen. Es können bis zu zehn QMN-Prozesse gestartet werden.
- Der *Dispatcher Process* (Dnnn) ist Teil der Shared-Server-Architektur. Er ist für die Kommunikation mit dem Client verantwortlich.
- Der *Shared Server Process* (Snnn) bedient in einer Shared-Server-Umgebung mehrere Clients.

Wenn Sie die Parallel-Query-Option aktiviert haben, wird die Abfrage auf mehrere Prozesse verteilt. Neben dem Masterprozess werden *Parallel-Server-Prozesse* (Pnnn) gestartet.

2.2.2 Lesekonsistenz

An den Anfang des Themas wollen wir eine Frage stellen. In einer Datenbank wird eine lang laufende SELECT-Anweisung auf einer großen Tabelle ausgeführt. Die Anweisung wird um 10:00 Uhr gestartet und führt einen Full Table Scan aus, liest also Block für Block. In der Zwischenzeit ändert eine andere Session einen Datensatz in dieser Tabelle und führt ein COMMIT aus, sodass die Änderungen datenbankweit sichtbar werden. Dies passiert um 10:05 Uhr. Um 10:07 liest die SELECT-Anweisung diesen Datenblock. Was wird im Ergebnis der SELECT-Anweisung erscheinen: der ursprüngliche oder der geänderte Datensatz? Was denken Sie?

Bevor wir die Frage beantworten, werden noch einige Begriffe und Abläufe geklärt.

Die System Change Number (SCN)

Die *System Change Number* (SCN) charakterisiert den eindeutigen Zustand einer Datenbank zu einem bestimmten Zeitpunkt. Sie wird in jedem Datenblock-, Segment- und Datafile-Header gespeichert. Damit ist es möglich, den Zustand eines jeden Datenblocks zuzuordnen. Sie kann als logischer, interner Zeitstempel betrachtet werden.

Die SCN wird kontinuierlich bei bestimmten Ereignissen erhöht. So besitzt zum Beispiel jede Transaktion in der Datenbank eine SCN. Wird eine COMMIT-Anweisung ausgeführt, dann wird gleichzeitig eine SCN an diese Transaktion gebunden. COMMIT-Anweisungen werden als COMMIT-Record in die Redo-Log-Dateien geschrieben. In diesem COMMIT-Record befindet sich die SCN, die die Transaktion eindeutig identifiziert.

Transaktionen

Die Gewährleistung der Transaktionssicherheit ist eine der wichtigsten Aufgaben einer relationalen Datenbank. Eine bestimmte Anzahl von SQL-Anweisungen wird als Transaktion zusammengefasst und gemeinsam auf die Datenbank angewandt oder gemeinsam zurückgerollt. Es ist nicht wie in anderen Datenbanksystemen erforderlich, den Anfang einer Transaktion zu propagieren. Dies erfolgt implizit. Oracle hält die folgenden drei Anweisungen zur Transaktionssteuerung bereit:

- **COMMIT:** Trägt alle Änderungen der Transaktion als permanent in der Datenbank ein und macht sie für alle Sessions sichtbar. Die Transaktion wird abgeschlossen, und implizit wird eine neue geöffnet.
- **ROLLBACK:** Rollt alle durch die Transaktion gemachten Änderungen zurück. Aus dem Blickwinkel anderer Sessions haben diese Änderungen niemals stattgefunden. Die Transaktion wird abgeschlossen, und implizit wird eine neue geöffnet.
- **SAVEPOINT:** Definiert einen Punkt innerhalb einer Transaktion, auf den separat zurückgerollt werden kann.

Hinweis

Für die DDL-Anweisung erfolgt ein impliziter Abschluss der Transaktion. COMMIT- oder ROLLBACK-Anweisung sind nicht erforderlich und unwirksam.

Oracle verwendet ein sogenanntes *optimistisches Locking*. Es wird davon ausgegangen, dass die überwiegende Mehrheit der Transaktionen mit COMMIT abgeschlossen wird. Aus diesem Grund werden COMMIT-Anweisungen sehr schnell abgeschlossen, wogegen ROLLBACK-Anweisungen lange laufen können.

Eine Transaktion ist aktiv, solange noch keine COMMIT- oder ROLLBACK-Anweisung abgesetzt wurde. Alle in einer nicht abgeschlossenen Transaktion gemachten Änderungen sind als temporär zu betrachten. Dabei werden die folgenden Prozesse ausgelöst:

- Es werden UNDO-Daten generiert, die ein Zurückrollen der Transaktion ermöglichen. Die UNDO-Daten enthalten die originalen, unveränderten Werte.
- Die Änderungen werden über den Redo Log Buffer in die Online-Redo-Log-Dateien in Form von sogenannten *Redo Records* geschrieben. Es werden sowohl die Änderungen der Daten- als auch der UNDO-Blöcke gespeichert.
- Die Änderungen werden im Buffer Cache der Datenbank gespeichert. Eine COMMIT-Anweisung löst nicht das Speichern der geänderten Datenblöcke auf die Disk aus. Dies geschieht in der Regel mit dem nächsten Checkpoint oder eines anderweitig geforderten Speicherns der Datenblöcke.
- Zeilen einer Tabelle, die von den Änderungen betroffen sind, werden bis zum Abschluss der Transaktion gesperrt. Andere Sessions können Änderungen durch nicht abgeschlossene Transaktionen nicht sehen.

Hinweis

Obwohl der interne Buffer-Cache-Mechanismus auf Blockebene arbeitet, erfolgt das Sperren von Datenstrukturen auf Zeilenebene. Dies ist nicht selbstverständlich und wird in anderen Datenbanksystemen anders gehandhabt. Der Vorteil liegt in einer deutlichen Verringerung der Konkurrenz durch Sperren auf Daten.

Eine COMMIT-Anweisung beendet die Transaktion und weist Oracle an, alle darin vorgenommenen Änderungen als persistent zu betrachten und datenbankweit zur Verfügung zu stellen. Dabei werden folgende Prozesse ausgelöst:

- Es wird eine SCN für die Transaktion generiert.
- Der Log Writer schreibt alle Redo Records der Transaktion in die Online-Redo-Log-Dateien.
- Alle Sperren auf den Datensätzen werden aufgehoben.
- Alle Savepoints werden gelöscht.
- Die Transaktion wird beendet.

Nach Abschluss der Transaktion werden die Änderungen für alle anderen Sessions der Datenbank sichtbar.

Eine ROLLBACK-Anweisung macht alle Änderungen, die innerhalb der Transaktion ausgeführt wurden, rückgängig. Dabei werden folgende Aktionen ausgeführt:

- Alle in den Datenblöcken vorgenommenen Änderungen werden zurückgerollt. Grundlage sind die gespeicherten UNDO-Daten.
- Alle Sperren auf den Datensätzen werden aufgehoben.
- Die Transaktion wird nicht gespeichert, so als hätte sie nie stattgefunden.
- Eine neue Transaktion wird implizit gestartet.

Oracle bietet die Möglichkeit, mit autonomen Transaktionen zu arbeiten. Sie werden von einer anderen Transaktion gestartet und sind nicht davon abhängig, ob die Basistransaktion mit COMMIT oder mit ROLLBACK abgeschlossen wird. Eine autonome Transaktion besitzt folgende Eigenschaften:

- Nicht abgeschlossene Änderungen der Haupttransaktion sind in der autonomen Transaktion nicht sichtbar.
- Sie können weitere autonome Transaktionen starten.

Alle aktiven Transaktionen finden Sie in der View V\$TRANSACTION. Die SQL-Abfrage in Listing 2.19 gibt die Transaktionen aus. Über die Startzeit können die zugehörigen Sessions und SQL-Anweisungen identifiziert werden.

```
SQL> SELECT b.sid, b.serial#,b.username,b.status,
2  c.sql_text,e.object_name,a.start_time
3  FROM v$transaction a,v$session b,v$sql c,
   v$locked_object d,all_objects e
4  WHERE a.ses_addr = b.SADDR
5  AND b.prev_sql_addr=c.address(+) AND
   b.prev_hash_value=c.hash_value(+)
6  AND b.prev_child_number = c.child_number(+) AND
   b.prev_sql_id=c.sql_id
7  AND d.object_id=e.object_id AND d.session_id=b.sid(+);
SID USER STATUS SQL_TEXT OBJECT_N START_TIME
-----
237 SYS INACTIVE UPDATE kb SET text=' KB 01/06/19 16:35:33
```



```
New Headline' WHERE  
id = 1
```

Listing 2.19: Offene Transaktionen abfragen

Checkpoint

Ein Checkpoint schreibt alle geänderten Datenblöcke aus dem Buffer Cache in die Tablespaces auf die Disk. Mit einem Checkpoint soll Folgendes erreicht werden:

- Regelmäßiges Schreiben von Änderungen aus dem Buffer Cache in die Tablespaces
- Die Zeit für den Recovery-Prozess im Fehlerfall reduzieren
- Wird für bestimmte Operationen benötigt (z.B. *shutdown immediate*)

Ein Checkpoint wird in folgenden Situationen angefordert:

- Beim Herunterfahren der Instanz mit den Optionen `normal`, `transactional` oder `immediate`
- Vorbereitung einer Online-Sicherung mit dem Befehl `ALTER {DATABASE|TABLESPACE} BEGIN BACKUP`
- Setzen einer Tablespace in den Status `OFFLINE`
- Automatischer Wechsel einer Online-Redo-Log-Gruppe
- Manuelle Auslösung mit dem Befehl `ALTER SYSTEM CHECKPOINT`

Folgende Typen eines Checkpoints können ausgelöst werden:

- *Thread Checkpoint*: Es werden alle geänderten Buffer der Instanz auf die Disk geschrieben.
- *Tablespace Checkpoint*: Es werden die geänderten Buffer einer Tablespace auf die Disk geschrieben.
- *Datafile Checkpoint*: Es werden die geänderten Buffer eines Datafiles gespeichert.
- *Incremental Checkpoint*: Hat das Ziel, einen Teil der geänderten Buffer auf die Disk zu schreiben, um große Checkpoint-Operationen zu vermeiden oder freie Buffer zu schaffen.

Tipp

Vermeiden Sie zu kleine Intervalle zwischen den Checkpoints durch eine hinreichende Größe der Online-Redo-Log-Dateien. Bei jedem Log Switch wird ein Checkpoint angefordert.

Die Häufigkeit von Checkpoints lässt sich auch mit den folgenden Parametern steuern:

- `log_checkpoint_timeout`: Maximale Zeit in Sekunden zwischen zwei Checkpoints. Ist sinnvoll für Systeme mit geringem Transaktionsaufkommen.
- `log_checkpoint_interval`: Anzahl von 512-KB-Blöcken, nachdem ein Checkpoint ausgelöst wird.
- `fast_start_mttr_target`: Anzahl von Sekunden, die die Datenbank für das Crash Recovery benötigt.

Je häufiger ein Checkpoint ausgeführt wird, desto geringer ist die Zeit für das Crash Recovery. Andererseits belastet jeder Checkpoint die Ressourcen des Systems. Es gilt also, einen Kompromiss zwischen Recovery-Zeit und Checkpoint-Häufigkeit zu finden.

Crash Recovery (Instance Recovery)

Beim Crash Recovery werden die Daten aus den Online-Redo-Log-Dateien angewandt, um die Konsistenz der Datenbank wiederherzustellen. In der Regel ist eine Fehlersituation oder Situation vorausgegangen, die zu einem inkonsistenten Zustand der Datenbank geführt hat. Dies kann zum Beispiel ein *shutdown abort* sein. Mit dem Crash Recovery wird die Datenbank wieder in einen konsistenten Zustand versetzt.

Hinweis

Für ein Crash Recovery sind keine Backup-Dateien erforderlich. Es kann komplett mit den Online-Redo-Log-Dateien durchgeführt werden. Archived-Redo-Log-Dateien werden ebenfalls nicht benötigt.

Oracle erkennt automatisch, ob ein Instance Recovery erforderlich ist. Wird ein Crash Recovery durchgeführt, dann erfolgt ein Eintrag in der Alert-Log-Datei.

```
ALTER DATABASE OPEN
Mon Jan 06 18:37:18 2019
Beginning crash recovery of 1 threads
  parallel recovery started with 2 processes
. . .
Completed crash recovery at
  Thread 1: logseq 1160, block 47258, scn 5200809
  38 data blocks read, 38 data blocks written, 44 redo k-bytes read
```

Listing 2.20: Eintrag des Crash Recovery in der Alert-Log-Datei

Ob ein Crash Recovery erforderlich ist, wird am Status der Online-Redo-Log-Dateien festgestellt. Verantwortlich dafür ist der SMON-Prozess. Das Recovery findet in zwei Phasen statt:

1. *Roll Forward:* Es werden alle Änderungen zwischen dem letzten Checkpoint und dem Ende der aktuellen Online-Redo-Log-Datei angewandt. Die Anwendung erfolgt dabei auch auf die UNDO-Segmente. Am Ende enthalten die Datenblöcke auch alle Änderungen von Transaktionen, die nicht mit COMMIT abgeschlossen wurden. Um die Datenbank auf einen konsistenten Stand zu bringen, müssen diese Änderungen zurückgerollt werden.
2. *Rollback:* Das Rollback der nicht abgeschlossenen Transaktionen erfolgt mit den Informationen aus den UNDO-Segmenten. Es wird so lange zurückgerollt, bis die SCN der Checkpoint-Position erreicht wird. Die Checkpoint-Position garantiert, dass alle Änderungen mit niedrigerer SCN in die Datafiles geschrieben wurden.

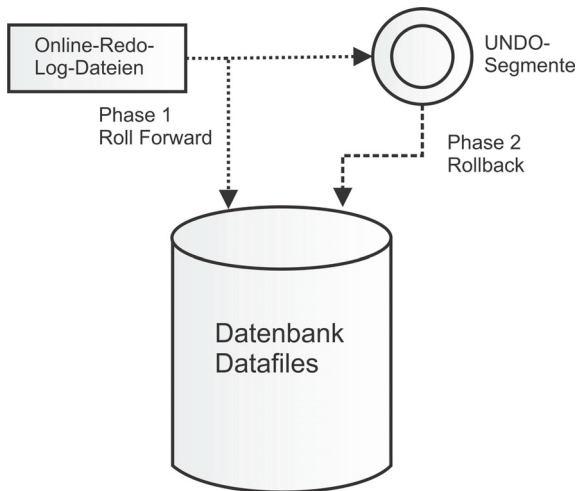


Abb. 2.7: Phasen im Crash Recovery

Nach diesem Prinzip sind nach dem Recovery alle zum Crash-Zeitpunkt abgeschlossenen Transaktionen in der Datenbank gespeichert. Alle Änderungen nicht abgeschlossener Transaktionen wurden zurückgerollt.

Isolation Level

Die Oracle-Datenbank kennt die folgenden drei Isolation-Level:

- *Read Committed (Standard)*: Jede SQL-Abfrage sieht nur die Daten, die vor Beginn der SQL-Anweisung (nicht der Transaktion!) durch abgeschlossene Transaktionen (COMMIT) anderer Sessions eingegangen sind. Das Prinzip wird »Lesekonsistenz« genannt. Ein Schreibkonflikt entsteht, wenn eine andere Session eine Änderung auf dieselbe Ressource durchführen will. Die Ressource bleibt so lange gesperrt, bis die Transaktion abgeschlossen ist.
- *Serializable*: Jede SQL-Abfrage sieht nur die Daten, die vor Beginn der Transaktion (nicht der SQL-Anweisung!) durch abgeschlossene Transaktionen (COMMIT) anderer Sessions eingegangen sind. Das Isolation Level funktioniert in einer Art und Weise, als ob keine anderen Benutzer auf der Datenbank wären.
- *Read Only*: Das Isolation-Level ist vergleichbar mit »Serializable«. Zusätzlich erlauben Read-only-Transaktionen nicht, dass die Daten durch andere Sessions verändert werden.

In der Praxis sollte der Standard beibehalten werden. Andere Isolation Level machen nur in wenigen Situationen Sinn.

Lesekonsistenz

Erinnern Sie sich noch an die Frage am Anfang des Abschnitts im Zusammenhang mit der lang laufenden SQL-Abfrage? Sie sollte spätestens jetzt beantwortet sein. Oracle garantiert im Isolation-Level *Read Committed* Lesekonsistenz. Alle Änderungen, die nach Beginn der SQL-Anweisung eingegangen sind, werden für das Ergebnis nicht berücksichtigt.

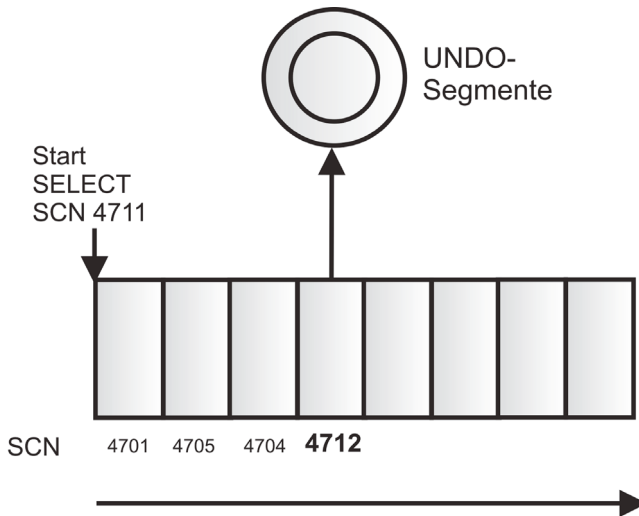


Abb. 2.8: Umsetzung der Lesekonsistenz über UNDO-Segmente

In diesem Zusammenhang tritt auch der Fehler »ORA-01555 Snapshot too old« auf. UNDO-Segmente werden regelmäßig freigegeben, um Platz für neue Transaktionen zu schaffen. Der Fehler tritt auf, wenn ein UNDO-Segment, das für die Lesekonsistenz benötigt wird, überschrieben wurde. Die SELECT-Anweisung bricht ab. Zwar gibt es einen Datenbankparameter `UNDO_RETENTION`, allerdings ist die Retention nicht garantiert. Werden dringend freie UNDO-Segmente benötigt, werden sie überschrieben, auch wenn die Aufbewahrungszeit noch nicht erreicht ist.

Der Fehler tritt naturgemäß dann auf, wenn lang laufende SELECT-Anweisungen auf ein hohes Transaktionsvolumen treffen.

Tipp

Um den Fehler »Snapshot too old« zu vermeiden, sollte als erste Maßnahme der UNDO-Tablespace vergrößert werden. Damit sinkt die Wahrscheinlichkeit, dass UNDO-Segmente zu schnell überschrieben werden. Prüfen Sie auch, ob SQL-Optimierung möglich ist und damit die Laufzeit der SQL-Anweisung verkürzt werden kann.

Stichwortverzeichnis

A

Abfrage
 analytische 610
Access Control List 462
ACFS 348
ACFS-Treiber 348
Active Data Guard 653, 703
Active Session History 516
Active Workload Repository 499
Active-Database-Duplication-Feature 670, 672
Active-Session-Pool 285
Adaptive Vereinigungsmethode 550
Adaptiver Ausführungsplan 549
Adaptives Cursor Sharing 545, 546
Ad-hoc-Abfrage 653
Administrationsaufwand 135
Administrator-managed-Datenbank 755
ADRCI 136
ADRCI-Utility 75
Agent
 installieren 792
AIX 320
AL32UTF8 38
Alertlog 54, 74, 127, 422
Allocation Unit 317
ALTER SYSTEM 53, 64
Analytische Abfrage 610
Anonymisierung 491
Application Container 403
Application Context 278, 280
Application Continuity 762, 766
Applikation
 Skalierbarkeit 404
Applikationsprofil 147
Apply Service 665, 676
Archive Gap 678
Archived-Redo-Log-Datei 82, 90, 148, 151, 162, 180, 447, 771
Archivelog-Backup 441
Archivelog-Modus 90, 148, 416, 421
Archiver-Prozess 104, 448, 532
ASM 317
ASM Fast Mirror Resync 339
ASM Fast Resync 716
ASM Preferred Read 340
ASM_DISKSTRING 318
ASM_POWER_LIMIT 344

ASMCMD 55, 341, 344
ASM-Disk 320, 323, 731
ASM-Diskgruppe 316, 332
ASM-Fehlergruppe 334
ASM-Flex-Diskgruppe 311
ASM-Instanz 316, 329, 728
ASM-Konfigurationsassistent 329, 742
ASM-Label 325
ASMLIB 318, 322
ASM-Redundanz 334
asmtool 325
ASS 87
AU_SIZE 338
Auditing 273, 457
Aufbewahrungsregel 163
Ausfallsicherheit 318, 716
Ausführungsplan 518, 546, 648
 adaptiver 549
Authentifizierung 270
Authentifizierungsprozess 253, 474
Authentifizierungstoken 846
Autobackup 438
Automatic Database Diagnostic Advisor 518
Automatic Diagnostic Repository 136
Automatic Memory Management 22, 97
Automatic Segment Space Management 87
Automatic Storage Management 150, 311, 713
Automatisierung 822
Autonomous Transaction Processing 827
Auxiliary 672
Auxiliary-Datenbank 195, 196, 198
Availability Domain 821
AWR-Repository 519
AWR-Snapshot 518
AWR-Statistik 559

B

Backup
 Consistent 149
 Inconsistent 149
 inkrementelles 148
 logical 147
 physical 147
 RAC-Datenbank 771
Backup and Recovery 147
 RAC-Datenbanken 771
Backup Piece 161, 162, 165, 166, 455

Backup Retention Policy 177
 Backup-and-Recovery-Strategie 134
 Backup-Job 174
 Backup-Medium 177
 Backup-Optimierung 164
 Backup-Set 165
 Backup-Strategie 452
 Backup-Zeit 164
 Bad-File 237
 Beispielschemata 42
 Benennungsmethode 253
 Benutzerverwaltung 406
 Betriebskosten 821
 Betriebssystemauthentifizierung 270
 BI Publisher 779
 Bigfile Tablespace 88, 114
 Bind Peeking 546, 547
 Bind Sensitive 547
 Bindevariable 546, 549
 Blackout 394, 803
 Block Change Tracking 149, 162
 Block Media Recovery 150, 187
 Block-Change-Tracking-File 82, 83, 91
 Blockgröße 84
 Block-Header 85
 Brute-Force-Angriff 486
 Brute-Force-Methode 474
 Buffer Cache 94, 96, 525, 717
 Bundle Patch 375

C

Cache Coherency 717
 Cache Fusion 713
 Cache-Fusion-Feature 713
 Calibration-Tool 558
 Capture-Datei 560, 563
 Capture-Prozess 557, 560
 Chained Row 84
 Change-Management 123, 557
 Channel 156, 161
 Character Set Scanner 297
 Chargeback-Modul 817
 Checkpoint 104, 108, 149
 Client-Wallet 465
 Cloud-Datenbank 228, 791, 840, 860
 Cloud-Dienst 20, 810, 826
 Cloud-Infrastruktur 856
 CLUSTER 83
 Cluster Health Monitor 758
 Cluster Interconnect 713
 Cluster-Cache-Kohärenz 770
 Cluster-Dateisystem 348, 713
 Cluster-Datenbank 728
 Clusterware 713
 Codiersatz 291
 Column Store 579, 610, 615, 620, 641
 COMMIT 88, 106

Common Role 415
 Common User 406, 414
 Complete Recovery 447
 Compression Unit 632
 Concurrent I/O 320
 Connect Time Load Balancing 705, 766
 Connection Pooling 586
 Connect-String 764
 Consistent Backup 149
 Consistent Read 718
 Containerdatenbank 36, 41, 372, 401, 408
 Backup 430
 CONTROLFILE AUTOBACKUP 160
 Conventional Path Load 237
 CPU-Ressource 285, 577
 Crash Recovery 50, 109, 149, 186, 721
 Crash-Zeitpunkt 152, 453
 Critical Patch Update 375
 CRS Daemon 737
 CSS Daemon 737
 CSV-Datei 213, 238, 590, 846
 CSV-Format 241
 Current-Block 718
 Cursor 545
 Cursor-Cache 547
 cx_Oracle 585

D

Data Guard 135, 663
 Data Guard Broker 666, 675, 682, 687, 688, 703
 Data Pump 213
 PARALLEL 227
 Data Recovery Advisor 136, 188, 450
 Data Science 583, 584
 Data Scientist 584
 Database Buffer Cache 94
 Database Character Set 291
 Database Lifecycle Management Pack 386
 Database Point-in-Time Recovery 199, 206
 Database Replay 557, 558, 559
 Database Vault 282
 Database Writer 104, 533
 DATABASE_PROPERTIES 292
 Database-Writer-Prozess 96
 Datafile
 Kopie 192
 kopieren 214
 Verlust 181
 Datafile-Header 105
 Data-Guard-Architektur 675
 Data-Guard-Konfiguration 674, 689
 Data-Guard-Prozess 677
 Data-Pump-Export 154, 195, 196, 215
 Data-Warehouse-Funktionalität 620
 Datenbank 81
 autonome 21, 357, 822, 851
 Containerdatenbank 36, 372, 401, 408

- integrierbare 372, 401, 406, 810
 - klonen 415
 - Monitoring der 800
 - Datenbankadministration 58
 - Datenbankaudit 488
 - Datenbank-Crash 452
 - Datenbank-Domäne 36
 - Datenbank-Gruppen 27
 - Datenbankidentifikationsnummer 154
 - Datenbankidentifizierungsnummer 82
 - Datenbankkatalog 87, 366, 373, 405, 458
 - Datenbank-Konfigurationsassistent 35
 - Datenbankkopie 670
 - Datenbank-Link 222, 233, 421
 - Datenbank-Listener 39
 - Datenbanksicherheit 265
 - Datenbank-Software 34
 - Datenbank-Trigger 413
 - Datenbank-Tuning 497
 - Datenblock 81, 96
 - Oracle 83
 - unkomprimierter 84
 - Datenorientiertes XML 302
 - Datenredundanz 406
 - Datensätze
 - doppelte 70
 - Datenübertragung 680
 - Datenverlust 446
 - Datumsformat 291
 - DB2 Shared Database 714
 - DB_FLASHBACK_RETENTION_POLICY 178
 - DB_FLASHBACK_RETENTION_TARGET 91, 178
 - DB_LOST_WRITE_PROTECT 679
 - DB_UNIQUE_NAME 668
 - DBA_REGISTRY_SQLPATCH 72
 - DBA-Recht 485
 - DBCA 35, 408, 781
 - DBID 82
 - DBMS_BACKUP_RESTORE 455
 - DBMS_CLOUD 845, 849
 - DBMS_JOB 243
 - DBMS_QOPATCH 72
 - DBMS_SCHEDULER 244
 - DBMS_SHARED_POOL 523
 - DBUA 359, 363, 369, 396
 - Debugger 60
 - Default-Plan 549
 - Deferred Segment Creation 87
 - Definer Rights 482
 - Defragmentierung 113
 - Deletion Policy 165
 - Denial of Service 506
 - Desaster-Recovery-Lösung 123
 - Desaster-Recovery-System 135
 - DGMGRL 55, 666, 688
 - Diagnose 136
 - Diagnostic Infrastructure 136, 144
 - Diagnostic-Datei 92
 - Diagnostik 135
 - Diagnostikdaten 140
 - Dictionary Cache 522
 - Dictionary Managed Tablespace 88
 - Dienst
 - globaler 703
 - Direct Path Load 237
 - Directory-Objekt 563
 - Direct-Path-Methode 214
 - Direct-Path-Modus 560
 - Dirty Buffer 95
 - Disaster Recovery 183, 184, 438, 440, 663
 - Discard-File 237
 - Discovery-Pfad 328, 731
 - Discovery-Prozess 325, 794
 - DISK_REPAIR_TIME 338
 - Disk-Layout 319
 - Dokumentorientiertes XML 302, 303
 - Domain Account 27
 - Domain Naming Service 724
 - Downtime 357
 - Dumpfile 76, 213
 - Duplicate Database 560
 - Dynamische Statistik 554
- E**
- Easy Connect 253
 - EM CLI 787, 810
 - emcli-Client 389, 419
 - Endian-Format 225
 - Engineered Systems 575
 - Enterprise Edition 20
 - Enterprise Manager 235
 - Enterprise Manager Cloud Control 124, 172, 704, 777
 - Enterprise Manager Console 777
 - Enterprise Manager Express 54
 - Enterprise Manager Grid Control 688
 - Entladen
 - Daten 240
 - Event-Klasse 515
 - Eviction 758
 - EVM Daemon 737
 - EVM Logger 737
 - Exadata 575
 - Exalogic Elastic Cloud 575
 - Extended Cluster 340
 - Extent 81, 86
 - External Table Load 237
 - External-Table-Methode 214
 - Externe Tabelle 240
- F**
- FAILED_LOGIN_ATTEMPTS 486
 - Failover 666, 684
 - automatisches 714
 - Failover-Art 762

Failover-Funktionalität 704, 714
 FAL-Prozess 667
 Fast Connection Failover 762, 764
 Fast Recovery Area 150, 152, 177, 203, 455, 743, 805
 FAST_START_MTTR_TARGET 155, 531
 fast_start_mttr_target 108
 FastConnect 798, 825
 FastStart 656
 Fast-Start Failover 663, 666, 675, 681, 684, 694
 Fast-Start-Failover-Feature 688
 Fault Domain 821
 Fehlergruppe 317
 Fibre-Channel-Netzwerk 314
 FINISH-Option 684
 Fixup-Skript 368
 Flash Recovery Area 356
 FLASHBACK DATABASE 90
 Flashback Database 150, 177, 203, 686
 Flashback Drop 201
 Flashback Table 199
 Flashback Transaction History 202
 Flashback-Database-Feature 677
 Flashback-Log-Datei 82, 83, 90
 Flashback-Operation 432
 Flashback-Technologie 199
 Flash-Disk 577
 Fleet Patching 385, 386, 388, 396
 Forced-Logging-Modus 152, 668
 Fragmentierung 113, 117
 Free Buffer 95
 Full Table Scan 504, 529, 547, 577
 Full-Backup 441, 452

G

GDS 703
 GDS Region 703
 gdsctl 704
 GDS-Katalog 704, 708
 GDS-Konfiguration 703
 GDS-Pool 703
 GES *siehe* Global Enqueue Service
 Global Cache 767
 Global Cache Service 715, 718
 Global Data Services 703
 Global Enqueue Service 718
 Global Resource Directory 718
 Global Service Manager 704
 Globaler Dienst 703
 Globalization 291
 Golden Gate 703
 Golden Image 25, 29, 388, 391
 Golden Master 420
 GRD *siehe* Global Resource Directory
 Grid Home 379
 Grid Infrastructure 728
 Grid Infrastructure Management Repository 728
 GSM-Listener 711

GSM-Server 710
 GSM-Service-Name 711
 Guaranteed Restore Point 205
 GV\$CACHE_TRANSFER 768
 GV\$RESOURCE_LIMIT 769

H

Hackerangriff 266, 473
 Hard-Parsing 539, 545
 Hardware
 Voraussetzungen 723
 Hash Join 645
 Hash-Code 273
 Havariesituation 446
 Health-Check 140
 Health-Check-Werkzeug 747
 Herunterfahren
 Datenbank 50
 High Availability Service 737
 High Water Mark 87, 120, 511
 Hintergrundprozess 81, 103, 243
 Histogramm 540
 Hochverfügbarkeit 703, 704, 821
 Hochverfügbarkeitslösung 713
 HWM 87
 Hybrid Cloud Management 777

I

I/O-Aktivität 319, 534
 IBM 713
 Image Copy 166
 Image-Kopie 194
 IMCU 612, 645
 Incident 123, 140
 Incident Package 143
 Incident Packaging Service 136
 Incidents 61
 Incomplete Recovery 187, 440
 Inconsistent Backup 149
 INDEX 83
 INDEX PARTITION 83
 Indizierung
 automatische 545
 InfiniBand-Netzwerk 576
 Infrastruktur 123, 135
 Init-Parameterdatei 68
 Inkrementelle Sicherung 162
 Inkrementelles Backup 148
 In-Memory Area 611
 In-Memory Compression Unit 611
 In-Memory Dynamic Scan 616
 In-Memory Expression Unit 611, 615
 In-Memory-Datenbank 619, 648
 In-Memory-Join 644
 In-Memory-Objekt 641
 In-Memory-Option 653
 In-Memory-Pool 658

In-Memory-Segment 636
 In-Memory-SQL 641
 Installation
 manuell 35
 RPM-basierend 32
 Installationsmethoden 21
 Instance-Tuning 519
 Instant Client 585
 Instanz 81
 Integrierbare Datenbank 810
 Inter-database Service Failover 705
 Intrusion Detection 267
 Inventar 25
 Invoker Rights 482
 iSCSI-Protokoll 313
 Isolation-Level 110
 ISO-Zeichensatz 291
 IT-Sicherheit 265

J

Java Pool 94
 Job-Kette 247
 Job-Tabelle 243
 Join-Gruppe 645
 JSON-Datei 844
 JSON-Datentyp 213

K

Kernel-Parameter 22, 723
 Klon 804
 Klon-Datenbank 420
 Klonen 357
 Knowledge Base 62
 Kompatibilitätsprüfung 374, 424
 Komplettsicherung 437, 438
 Kompressionsmethode 167, 620
 Kompressionsstufe 638
 Konsolidierungsplattform 580
 Kontrolldatei 82, 89, 162, 447
 Kopie
 Datafiles 192

L

Langläufer 65
 Large Object 83, 588
 Large Pool 94, 97, 524
 Lernen
 maschinelles 583, 594
 Lesekonsistenz 105, 110, 718
 Library Cache 96, 521
 Listener 50, 253, 256, 379, 394
 listener.ora 39, 257
 LNS-Prozess 667
 Load Balancing 311, 703, 704
 LOB INDEX 83
 LOB PARTITION 83
 LOB SEGMENT 83

Local Area Network 704
 Locally Managed Tablespace 87
 Locking
 optimistisches 106
 Locking-Verhalten 510
 Locks 67
 Log Switch 676
 Log Transport Service 675, 700
 Log Writer 88, 531
 log_checkpoint_interval 108
 log_checkpoint_timeout 108
 LOG_FILE_NAME_CONVERT 671
 Logical Backup 147
 Logical Change Records 664, 696
 Logical-Standby-Datenbank 664, 696, 701
 LogMiner 457, 459
 LogMiner Dictionary 699
 LogMiner-Sitzung 458
 LogMiner-Technologie 664
 Logon-Trigger 280
 Log-Writer-Prozess 96
 Lost-Write-Fehler 679
 LRU-Mechanismus 522
 lsnrctl 40

M

Managed-Recovery-Modus 677
 Managed-Recovery-Prozess 667
 Managed-Recovery-Status 677
 Management Agent 777
 Maschinelles Lernen 583, 594
 Maskierung 491
 Master Key 472
 Materialized View 631
 Maximum-Availability-Modus 664, 665, 694
 Maximum-Performance-Modus 664, 665, 694
 Maximum-Protection-Modus 664, 666
 Media Failure 150
 Media Recovery 150, 207
 Media-Management-Library 156
 Media-Management-Software 175
 Memory Advisor 530
 MEMORY_MAX_TARGET 97, 101
 MEMORY_TARGET 97, 98
 Metrik 800
 Migration 228, 809, 820, 840
 Migrationsaufwand 19
 Migrationsschritt
 testen 843
 Migrieren
 nach ASM 353
 Mirroring 316
 Monitoring 126
 Mount-Status 50
 MRP *siehe* Managed-Recovery-Prozess
 Multipathing 311, 318
 Multiplexing 167

Multisection Backup 168
 Multitenant-Architektur 401, 402
 Multi-Version-Consistency-Modell 718

N

Namensauflösung 253
 NAS-Architektur 313
 National Character Set 291
 NESTED TABLE 83
 Net Services 251
 NETCA 35, 52
 Network Attached Storage 312
 Network File System 313
 Netzwerkport 255
 Netzwerkprotokoll 252
 NLS_LANG 46, 293
 NLS_SORT 295
 Noarchivelog-Modus 148
 NOMOUNT-Option 94

O

Object Store 266, 844, 846
 Objekt-ID 67
 Objektprivileg 275
 Objektrelationale Speicherungsform 304
 Observer 694
 OBSOLETE 163
 OCI-Konsole 846, 855
 OFA 36
 Offline-Backup 148
 Offloading 577
 OLTP-Schema 261
 OML4R 604
 OMS-Server 784, 798, 811
 One-Off Patch 375, 378
 Online-Backup 148
 Online-Katalog 458
 Online-Redo-Log-Gruppe 441
 OPATCH_XML_INV 72
 OPatch-Utility 72, 377
 Operation
 länderspezifisch 291
 Optimierungsaufgabe 497
 Optimierungsziel 521
 Optimistisches Locking 106
 Optimizer 518, 547
 optimizer_dynamic_sampling 551
 Optimizer-Plan 641
 Optimizer-Statistik 502, 557, 822
 Optimizer-Trace 539, 642
 ORAchK 747
 Oracle Autonomous Database 819
 Oracle Base 25
 Oracle Call Interface 252
 Oracle Cloud 228, 797, 819
 Oracle Cloud Infrastructure 819
 Oracle Clusterware 728

Oracle Data Redaction 490
 Oracle Database Appliance 575
 Oracle Database Machine 575
 Oracle Database Setup Wizard 25
 Oracle Database Vault 38
 Oracle Enterprise Manager 58, 369, 417
 Oracle Flexible Architecture 36
 Oracle Home
 Read Only 30
 Oracle In-Memory 609
 Oracle Label Security 38
 Oracle Machine Learning 594
 Oracle Managed File 113, 316
 Oracle Restart 331
 Oracle-Active-Data-Guard-Option 685
 Oracle-Datenbankarchitektur 81
 Oracle-Datenblock 83
 Oracle-Edition 55
 Oracle-Home-Verzeichnis 25
 Oracle-Support-Website 749
 oradim 43, 367
 oraenv 48
 orapki 466
 OSDBA 48
 Out-of-place-Upgrade 359, 392

P

Parallele Vereinigungsmethode 553
 Parallelitätsgrad 285
 Parameter
 obsolete 66
 Parsing-Prozess 521
 Passwordfile 82, 83, 89
 Passwort 272
 Passwortdatei 311, 681
 Passwort-Hash 475
 Passwort-Policy 267, 273
 Past Image 719
 Past-Image-Konzept 719
 Patch 557
 Patch Set Update 375
 Patching 375
 Patchlevel 72
 Patch-Nummer 382
 Patch-Verfahren 357
 PBKDF2-Algorithmus 475
 PCTFREE 86
 PDB Snapshot 423
 Pending Area 286
 Performance 497
 Performance-Killer 504
 Performance-Monitoring 863
 Performance-Problem 499, 538, 557
 Performance-Risiko 358
 Performance-Steigerung 713
 Performance-Tuning 497
 Performance-Übersicht 501

Performance-Werkzeug 499
 Persistent Memory 579
 PGA 94
 PGA_AGGREGATE_LIMIT 506
 PGA_AGGREGATE_TARGET 97
 PGA-Verbrauch 508
 Physical Backup 147
 Physical-Standby-Datenbank 663, 664, 666
 Pinned Buffer 95
 PL/SQL 60
 PL/SQL Injection 482
 PL/SQL-API 455
 PL/SQL-Injektion 277
 Plattform
 für RAC 722
 Plattformunabhängigkeit 713
 Pluggable Database 36
 PMON-Prozess 127
 Policy-Funktion 278
 Policy-managed-Datenbank 755
 Pre Upgrade Check 397
 Preferred Mirror Read 716
 Preprocessing 558, 563
 Pre-Upgrade Check 364
 Pre-Upgrade Information Tool 361
 Primärdatenbank 664
 Primärrolle 669
 Priorität 638
 Private Cloud 777
 Private Cloud Appliance 575
 Private Interconnect 715, 724
 Process Monitor 104
 Profil 271
 Program Global Area 94, 97
 Protection-Modus 687
 Provisioning 580, 809, 826
 Proxy PDB 403
 Public Cloud 777, 797, 809
 Public Interface 724
 PVID 320
 Python 585
 Python-Bibliothek 586

Q

Quell-PDB 415

R

RAC-Datenbank 651, 739, 747, 750
 RAID-System 318
 Raw Device 318
 Read Committed 110
 Read Only 110
 Read Only Oracle Home 48, 357, 383
 Read-only-Modus 677
 Real Application Clusters 713
 Real Application Testing 357, 360, 557, 566
 Real Time Apply 663, 676

Real Time Monitoring 543
 Real Time Query 685
 Real-time SQL Monitoring 65
 Real-Time-ADDM-Analyse 535
 Real-Time-Apply-Konfiguration 681
 Recovery Manager 147, 155, 354, 452
 Recovery Window 163
 Recovery-Katalog 156, 157, 170, 172, 444, 455
 Recovery-Prozess 104, 151, 445, 447, 722
 Recovery-Strategie 438
 Recovery-Szenario 193, 437
 Recovery-Zeit 154, 771
 Recycle Pool 94, 529
 Redaction Policy 492
 redo buffer allocation retries 531
 Redo Log Buffer 94, 96, 531
 Redo Transport Service 665, 680
 Redo-Log-Dateien 69, 88
 Redundancy 317
 Redundancy Set 153
 Refreshable Clone PDB 420
 Regressionsanalyse 598
 Regressionsmodell 593
 Release
 Long Term 19
 Short Term 19
 Release Update 375
 Release Update Revision 375
 Release-Strategie 19
 Release-Update 376
 Remote-Datenbank
 klonen 419
 Remote-File-Server-Prozess 667
 Remote-Installation 725
 Reoptimization 554
 Replay Upgrade 372, 402
 Replay-Client 558, 563
 Replay-Prozess 563
 RESETLOGS-Option 185, 445
 Resetlogs-Option 441
 Resource Consumer Group 285
 Resource Manager 285, 401
 Resource Plan 285
 Resource Plan Directive 285
 Response-Datei 389
 Ressourcen-Auslastung 659
 Ressourcenverwaltung 703
 Restore Point 91, 364
 Restore und Recovery 147
 Restore-Zeit 134
 RESYNC CATALOG 159
 Retention Policy 457
 RFS *siehe* Remote-File-Server-Prozess
 RMAN 55, 147
 RMAN-Backup 437, 447
 RMAN-Client 158, 172
 RMAN-Katalog 156, 165
 RMAN-Konfiguration 160

RMAN-Skript 174, 198
 Role Management Service 665
 ROLLBACK 106
 ROLLBACK SEGMENT 83
 Rollentausch 669, 682
 Rolling Upgrade 664, 675
 Rolling-Upgrade-Feature 687
 Root Container 403, 408
 Row Chaining 512
 Row Migration 86, 512
 Row Store 615
 ROWID 71, 84, 652
 RPM-Paket 32
 RSA-Verschlüsselung 465
 Runaway Query Management 865
 Run-Time Load Balancing 705

S

SAN-Architektur 314
 SAN-Infrastruktur 123
 SAVEPOINT 106
 SCAN-Listener 708, 728, 761
 SCAN-Name 729
 SCAN-Operation 577
 Scheduler 243
 Schemaregistrierung 304
 Schwellenwert 290, 694, 803
 SCN 105
 SCSI-System 312
 SDU-Parameter 262
 Seed PDB 403
 Segment 82, 86
 Selbstservice-Portal 816
 SELECT FAILOVER 763
 Semantik-Check 539
 Sequence Number 454
 Serializable 110
 Server-Pool 755
 Service Level Agreement 151
 Service Request 62, 140
 Service-Namen 410
 Session
 abbrechen 64
 Session Data Unit 262
 SESSION FAILOVER 763
 Setup Wizard 25, 30
 SGA 94
 SHA256-Hash-Algorithmus 465
 SHA2-Verschlüsselung 475
 Sharding-Architektur 404
 Shared Pool 94, 520, 545
 Shared Pool Reserved Area 523
 Shared Server 97
 Shared Server Process 105
 Shared-Everything-Architektur 715
 Shared-Nothing-Architektur 714
 Shared-Server-Architektur 259

Shared-Server-Verbindung 560
 Shared-Storage-Architektur 715
 Sicherheitsfeature 265
 Sicherheitslücke 266, 274, 375, 486
 Sicherung
 inkrementelle 162
 SID 36
 Single Point of Failure 134, 716
 Skalierbarkeit 260, 704, 713, 716
 Smart Aggregation 579
 Smart Flash Log Write-Back 579
 Smart Scan 577
 smca 742
 Snapshot 352
 Snapshot Metadata Unit 611
 Snapshot Standby Database 560
 Snapshot Standby-Datenbank 685
 Snapshot-Karussell 423
 Soft-Parsing 545
 Software-Library 389, 784, 811
 Solaris 320
 Sortierung 294
 linguistische 294
 Spalten-Format 610
 Speicherungsform
 objektrelationale 304
 SPFILE 53, 66, 68, 81, 89, 153, 162, 184, 261, 332,
 354, 361, 622, 751
 Spiegelung 317
 SQL Access Advisor 518
 SQL Apply 664
 SQL Developer 60
 SQL Performance Analyzer 557, 558, 566
 SQL Plan Baselines 558, 567
 SQL Tuning Advisor 59, 518, 567
 SQL Tuning Set 557, 566
 SQL*Loader 213, 237
 SQL*Plus 52, 56
 SQL-Anweisung
 I/O-intensive 568
 SQL-Apply-Architektur 696
 SQL-Ausführungsplan 557
 SQLc 56
 SQL-Monitoring 865
 sqlnet.ora 253
 SQL-Optimizer 497, 538, 546, 549, 641
 SQL-Tuning 497, 537, 822
 SQL-Überwachung 862
 srvctl 750
 SSB-Schema 623
 SSH Keys 725
 SSL-Zertifikat 266
 Staging-Tabelle 570
 Standardisierung 135
 Standby-Datenbank 152, 360, 663
 Standby-Redo-Log-Datei 669
 Standby-Rolle 693
 Startmodus 44

Statistics Feedback 554
 Statistik 540
 dynamische 554
 sammeln 503
 Storage Area Network 314
 Storage-Architektur 312
 Storage-Array 333
 Storage-Index 578
 Storage-Server 577
 Storage-Subsystem 318
 Storage-System 577
 Storage-Typ 727
 Stored Outlines 547
 Streams Pool 94, 97
 Striping 316, 317, 343
 Strukturänderung 438, 441
 STS *siehe* SQL Tuning Set
 SuperCluster 575
 Supplemental Logging 457, 697
 Support 62
 Support Workbench 136, 139
 Switchover 666, 687, 692
 Switchover-Befehl 692
 Switchover-Prozess 681, 682, 693
 Switchover-Status 682
 Syntax-Check 539
 SYS_CONTEXT 279
 SYSASM-Rolle 333
 SYSAUX 87
 SYSAUX-Tablespace 81
 SYSDBA 23, 48, 89, 333, 432
 SysInternals 47
 SYSLOG 266
 SYSOPER 23, 89, 333
 System Change Number 105, 149, 679
 System Global Area 94, 611
 System Monitor 104
 SYSTEM-Account 52
 System-Container 403
 SYSTEM-Tablespace 81

T

Tabelle
 externe 240
 verschlüsselte 579
 Tablespace 87, 113, 116, 158
 UNDO 87
 Tablespace Point-in-Time Recovery 148, 195, 207
 Tablespace 81
 Tape Library 175
 Tempfiles 82
 TEMPORARY-Tablespace 81
 Time-Modell 500, 514
 Timezone Upgrade 364
 Time-Zone-Version 366
 Tivoli Storage Manager 175
 TLS-Verbindung 469

TLS-Verschlüsselung 464
 TNS-Layer 252
 Top Sessions 59, 544
 Top-SQL-Anweisung 861
 Trace-Dateien 75
 Transaction Guard 762, 765
 Transaktion 106, 457, 558
 Transaktionen zurückgerollt 65
 Transaktionslog 88
 Transaktionssicherheit 105
 Transparent Application Failover 762
 Transparent Data Encryption 471
 Transportable Tablespace 213
 TSPITR 148, 207
 Tuning-Methode 499
 Tuning-Roboter 822
 TYPE2 UNDO 83

U

Übertragung
 verschlüsselte 464
 Überwachung 123, 125, 800
 Übungsdatenbank 19
 UNDO_RETENTION 111
 UNDO-Segment 111
 UNDO-Tablespace 87, 115, 199
 Unified Auditing 266
 Universal Installer 739
 Unix-Betriebssystem 45
 Upgrade 357, 557
 automatisiertes 363
 manuelles 366
 Upgrade-Methode 363
 Usage-Statistik 644
 User Managed Backup 148
 USER_RECYCLEBIN 201

V

V\$ACTIVE_SESSION_HISTORY 543
 V\$ARCHIVE_GAP 684
 V\$ARCHIVED_LOG 90, 185
 V\$ASM_DISKGROUP 334
 V\$BGPROCESS 46
 V\$BUFFER_POOL_STATISTICS 526
 V\$DATABASE_BLOCK_CORRUPTION 188
 V\$DB_CACHE_ADVICE 528
 V\$DIAG_INFO 138
 V\$FLASH_RECOVERY_AREA_USAGE 180
 V\$FLASHBACK_DATABASE_LOG 178
 V\$INSTANCE_RECOVERY 155
 V\$LIBRARYCACHE 521
 V\$LOGMNR_CONTENTS 458
 V\$MEMORY_DYNAMIC_COMPONENTS 99
 V\$MEMORY_RESIZE
 OPS 99
 V\$MEMORY_TARGET_ADVICE 99
 V\$MTTR_TARGET_ADVICE 531

V\$PROCESS 65
 V\$SESSION_LONGOPS 65
 V\$SHARED_POOL_ADVICE 521
 V\$SQL_MONITOR 65
 V\$SQLAREA 516
 V\$SQLSTATS 544
 V\$STANDBY_LOG 678
 V\$TRANSPORTABLE_PLATFORM 225
 Vereinigungsmethode 539
 adaptive 550
 parallele 553
 Veritas 713
 Verschlüsselung 471
 TLS 464
 Version 20c 41
 Versionszyklus 19
 Virtual Interface 724
 Virtual Private Database 277

W

WAIT_FOR_GAP 678
 WAIT_FOR_LOG 677
 Wallet 472
 Wallet Manager 831
 Warteklasse 501
 Wartungstask
 automatisiert 248
 WebLogic Active GridLink 765
 WebLogic GridLink 762
 Wiederherstellbarkeit 134
 Wiederherstellung 181, 455
 Wiederherstellungsprozess 437
 Wiederherstellungsstrategie 181, 437
 Wiederherstellungszeit 151
 Wiederherstellungszeitpunkt 148

Windows-Betriebssystem 42
 Windows-Dienst 44
 Workload 557, 558, 560
 repräsentativer 567
 Workload-Capture-Prozess 558
 Workload-Client 565
 Workload-Replay-Prozess 558
 Workload-Verteilung 705
 wrcl-Utility 563

X

XML
 datenorientiertes 302
 dokumentorientiertes 302, 303
 XML DB Repository 300
 XML-Datei 424
 XML-Datenbank 299
 XML-DB 299
 XML-Dokument 301, 304, 307, 308
 XML-Schema 304
 XMLTYPE 301, 303

Z

ZDLRA 575
 Zeichensatz 38
 umwandeln 297
 Zeitzone-Upgrade 397
 Zellserver 577
 Zertifikat
 self-signed 465
 Zertifizierungsmatrix 780
 ZFS 580
 ZFS Storage Appliance 575