Multitenant

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

CONTAINER SCOPE

- **CDB root** — Common scope
- **PDB** — Local scope
- **Application root** — Shared definitions
- **Application PDB** — Sync changes

01 Multitenant architecture and administration

Container boundaries and open states

- **CDB$ROOT** holds the common dictionary; **PDB$SEED** is the read-only template for regular PDBs. A PDB has its own SYSTEM, SYSAUX and temporary storage and appears as a database to its applications.
- **Instance resources** such as background processes, the SGA, control files and online redo are shared across containers. PDBs do not each have their own LGWR or PMON.
- **Opening the CDB** opens the root; ordinary PDBs need their own open operation. **SAVE STATE** preserves a PDB's open state across restarts; the seed normally opens read-only.
- **Services** select a container for client connections. **ALTER SESSION SET CONTAINER** changes the current container for an authorized common user; finish transactions in their originating container before switching.

Common users, privileges and isolation

- **Common users and roles** span their root's scope; **local users and roles** belong to one PDB. CDB common names normally use **COMMON_USER_PREFIX**, whose default is C##; application common users follow application-container rules.
- **CONTAINER=ALL** makes a supported grant common; **CONTAINER=CURRENT** limits its scope. A common user's existence across PDBs does not imply identical privileges or permission to log in everywhere.
- **SYSBACKUP** delegates backup and recovery administration without general access to application data. Separate it from **SYSDBA**, **SYSASM**, **SYSDG** and **SYSKM** responsibilities.
- **Lockdown profiles** restrict selected statements, options and features in PDBs. Define them in the CDB root or application root and assign them with **PDB_LOCKDOWN**; a dynamically based profile can inherit later base-profile changes.
- **Schema-only accounts** use NO AUTHENTICATION to own objects without direct password login. **Privilege Analysis** records exercised privileges; **unified auditing** records configured actions, including distinctions between direct and recursive activity.

Multitenant

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

CONTAINER SCOPE

- **CDB root** — Common scope
- **PDB** — Local scope
- **Application root** — Shared definitions
- **Application PDB** — Sync changes

PDB creation, unplugging and cloning

- **CREATE PLUGGABLE DATABASE** can use the seed, another PDB or an unplugged PDB description. A newly created PDB is initially mounted; open it read-write to complete integration. A default PDB service is created.
- **OMF**, **CREATE_FILE_DEST**, **FILE_NAME_CONVERT** and **PDB_FILE_NAME_CONVERT** provide file-placement choices. **SOURCE_FILE_NAME_CONVERT** locates existing source files when their paths differ from the manifest.
- **XML manifests** describe a PDB but leave its data files separate. A **.pdb archive** packages the manifest and compressed data files. Check plug compatibility and inspect **PDB_PLUG_IN_VIOLATIONS** before normal use.
- **COPY** creates destination files; **NOCOPY** uses files in place; **MOVE** moves them. Remove the unplugged PDB's old catalog entry before reusing its identity in that CDB. **USER_TABLESPACES** selects user tablespaces for supported plugging operations.
- **Remote cloning** uses a database link and suitable source privileges. **SERVICE_NAME_CONVERT** renames copied user-defined services in the clone; it does not rename source services. Online relocation requires local undo in both CDBs and ARCHIVELOG on the source.

Undo and snapshot choices

- **Local undo** gives each container its own undo; **shared undo** uses root undo for transactions throughout the CDB. Read **LOCAL_UNDO_ENABLED** from database properties to identify the active mode.
- **ALTER DATABASE LOCAL UNDO ON/OFF** is a CDB-level change performed from the root in upgrade mode, followed by restart. Switching to shared undo does not require manually deleting every PDB undo tablespace first.
- A **PDB snapshot** is a point-in-time source for later clones and may be full or sparse according to storage support. A **snapshot copy PDB** is a sparse clone dependent on supported underlying snapshot storage; preserve that dependency until materialized.

Application containers and shared objects

- An **application container** has an application root, application PDBs and optionally an application seed. It can host multiple applications; an application PDB belongs to one application container.
- **Metadata-linked objects** share definitions while keeping data container-local. **Data-linked objects** share root-held data. **Extended data-linked objects** combine shared root data with PDB-local data.
- Record common application changes inside **BEGIN/END INSTALL**, **PATCH** or **UPGRADE** operations. **APPLICATION ... SYNC** applies those changes to selected application PDBs; changes are not automatically applied to every PDB or seed.

Multitenant

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

CONTAINER SCOPE

- **CDB root** — Common scope
- **PDB** — Local scope
- **Application root** — Shared definitions
- **Application PDB** — Sync changes

- An **application seed** is an optional template. Synchronize it with the application root before cloning current application PDBs. Existing application PDBs can synchronize directly with their root.
- An **application upgrade** records application names and versions and creates an application root clone to support PDBs still using the earlier version. Upgrade the application in its application root, then synchronize PDBs deliberately.

Cross-container access and security

- **CONTAINERS()** queries corresponding objects across eligible containers and exposes **CON_ID**. Container visibility and privileges still apply; connecting to the root alone does not grant access to every application's data.
- A **container map** associates key ranges or values with application PDBs so supported queries can prune containers. A **proxy PDB** represents a remote PDB and enables supported cross-CDB application access.
- **VPD**, **FGA**, **OLS**, **TSDP** and unified audit policies have different scoping rules. Treat application-common policy creation and synchronization as explicit lifecycle operations; do not assume all policies propagate identically.

Resource and memory controls

- A **CDB resource plan** allocates CPU and parallel-execution resources among PDBs. **Shares** determine relative allocation under contention; unused capacity can be borrowed. **Utilization limits** impose ceilings.
- A **PDB resource plan** manages consumer groups within that PDB, with controls such as session resource use and uncommitted undo. CDB directives and PDB consumer-group directives act at different levels.
- **PDB SGA_TARGET** limits PDB SGA use when its prerequisites are met. PDB **DB_CACHE_SIZE** and **SHARED_POOL_SIZE** can reserve minimum component memory; distinguish reservations from upper bounds.
- **PDB PGA_AGGREGATE_TARGET** guides work-area sizing; **PGA_AGGREGATE_LIMIT** limits aggregate PGA. Validate root settings and Resource Manager prerequisites before treating a PDB setting as enforced isolation.

RMAN Backups

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

BACKUP PIPELINE

Read Data files → **Transform** Compress & encrypt → **Write** Disk or SBT → **Catalog** Record metadata

02 RMAN backup architecture and administration

Backup forms and consistency

- An **image copy** is a disk-resident copy of a data file, control file or archived log. A **backup set** contains RMAN-formatted data in one or more **backup pieces**, written to disk or through SBT.
- A **whole database backup** covers all data files and can be consistent or inconsistent. Control file, SPFILE and archived-redo inclusion depend on the command and configuration; the term alone does not promise every recovery artifact.
- **ARCHIVELOG** permits online backups of changing data files, recovered using redo. **NOARCHIVELOG** backups require a consistent database after clean shutdown; RMAN backs it up mounted, while user-managed cold copies can be made with it shut down.
- **Full backups** and **level 0 incrementals** can contain similar data, but only level 0 establishes an incremental strategy's base. Level 1 **differential** backs up changes since the latest level 0 or 1; **cumulative** uses the latest level 0.
- **User-managed copies** of eligible data files can be cataloged as image copies. For an online operating-system backup, follow the supported backup-mode procedure and preserve redo; an ordinary file copy of changing files is insufficient.

Sets, channels and throughput

- A **channel** is an RMAN server session that reads or writes backup data. Device **PARALLELISM** permits multiple channels; align SBT concurrency with available drives and media-manager capability.
- **Multiplexing** interleaves blocks from multiple input files in a backup piece. **Multisection backups**, using SECTION SIZE, divide a large data file into sections that available channels can process concurrently.
- **Duplexing** produces multiple identical copies of each backup piece using **COPIES** or persistent backup-copy settings. It is distinct from striping one piece across devices.
- A **media manager** implements SBT access and may support proxy copies, where it controls data movement. Disk backup sets can be backed up to SBT; do not assume backup-of-backup-set operations support every device direction.

Compression, encryption and secrets

- **Binary compression** reduces backup-set size at a CPU cost and supports disk or SBT output. **AS COPY** image copies cannot use RMAN binary compression; unused-block omission and binary compression are different mechanisms.
- **Tape compression** is supplied by the media system. Combining it with already compressed RMAN data often adds overhead with little space benefit; measure the full pipeline.

RMAN Backups

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

BACKUP PIPELINE

Read Data files → **Transform** Compress & encrypt → **Write** Disk or SBT → **Catalog** Record metadata

- **Transparent encryption** uses the database keystore; **password encryption** uses a supplied password; **dual-mode encryption** permits either recovery method. Preserve and independently back up required keystores and credentials.
- **SET ENCRYPTION** overrides applicable **CONFIGURE ENCRYPTION** settings for the job. RMAN encryption protects backup sets; it does not turn image copies into encrypted backup sets or automatically back up an external TDE keystore.

Repository, catalogs and access

- The **target control file** holds RMAN metadata and persistent configuration. **SHOW ALL** includes default settings; **V$RMAN_CONFIGURATION** exposes explicitly configured values. A recovery catalog preserves a longer, centralized history.
- Create a **catalog owner** with quota on a suitable tablespace and **RECOVERY_CATALOG_OWNER**, then run **CREATE CATALOG**. The catalog can live in an existing database; protect it independently of the databases it describes.
- **REGISTER DATABASE** and subsequent resynchronization copy target metadata into the catalog. Structural changes can trigger full resynchronization; frequent resync avoids losing aged-out control-file records.
- A **recovery catalog** enables stored RMAN scripts, **KEEP FOREVER** backups and historical schema reporting. **CONTROL_FILE_RECORD_KEEP_TIME** governs reusable control-file records; it is not a backup retention policy.
- A **virtual private catalog** restricts a catalog user's visible backup metadata. The base owner enables and delegates access; use the supported PDB registration and VPC workflow for catalog-backed PDB-target administration.

Retention, validation and recovery-area space

- **REDUNDANCY** retains a specified backup count; a **RECOVERY WINDOW** retains backups needed to reach points within a time interval. **REPORT OBSOLETE** evaluates policy; **DELETE OBSOLETE** removes eligible backups.
- **CROSSCHECK** tests recorded backups against storage and marks missing ones **EXPIRED**. **LIST EXPIRED** displays them; **DELETE EXPIRED** removes repository records. Expired means missing, while obsolete means no longer required by retention policy.
- The **FRA** is governed by **DB_RECOVERY_FILE_DEST** and **DB_RECOVERY_FILE_DEST_SIZE**. Eligible files can be reclaimed under space pressure; retention and archived-log deletion policies determine eligibility, not unlimited capacity.

RMAN Backups / Recovery & Flashback

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

BACKUP PIPELINE

Read Data files → **Transform** Compress & encrypt → **Write** Disk or SBT → **Catalog** Record metadata

- **BACKUP RECOVERY AREA** targets supported recovery files in current and previous recovery areas. **BACKUP RECOVERY FILES** also considers eligible recovery files outside them. These commands use SBT and exclude online redo and flashback logs; **FORCE** overrides their already-backed-up filtering.
- **KEEP UNTIL/FOREVER** creates archival backups outside normal obsolescence rules and preserves the redo needed for a consistent restore point. Keep archival backups outside the FRA; **KEEP FOREVER** requires a catalog.

Control files and container backup scope

- **CONFIGURE CONTROLFILE AUTOBACKUP ON** protects the control file and the SPFILE when one is in use. Autobackups follow qualifying backup or structural-change events; preserve the **DBID** and autobackup location for disaster recovery.
- **BACKUP CURRENT CONTROLFILE** or SQL binary backup creates a restorable control-file copy. **BACKUP CONTROLFILE TO TRACE** instead writes SQL for re-creation; it does not preserve all RMAN repository history like a binary copy.
- Connected to **CDB$ROOT**, RMAN can back up the whole CDB or selected PDBs and PDB tablespaces. Qualify repeated tablespace names with their PDB context.
- Connected directly to a **PDB**, **BACKUP DATABASE** covers that PDB. CDB control files, SPFILE and archived redo are root-level backup resources; online redo logs are not RMAN backup inputs.

03 Recovery, flashback and duplication

Instance recovery versus media recovery

- **Instance recovery** follows an abnormal shutdown and uses online redo from the checkpoint position to roll forward changes, then rolls back uncommitted work. Transaction rollback can continue after the database opens.
- **FAST_START_MTTR_TARGET** guides checkpoint activity toward a recovery-time target. A smaller target generally increases steady-state write work to reduce redo processing after a crash.
- **Media recovery** repairs lost or damaged files by restoring suitable backups and applying redo. **RESTORE** retrieves files; **RECOVER** advances their contents. Complete recovery needs an unbroken redo chain from each restored file's checkpoint.
- Loss of an inactive log group, a current unarchived log, a data file and a control file are different failures. Losing required current redo can force incomplete recovery; an older missing archived log may be irrelevant to a sufficiently recent usable backup.

Recovery & Flashback

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

CHOOSE RECOVERY SCOPE

- **Instance** — Online redo
- **Media** — Restore & recover
- **Flashback** — Rewind history
- **PITR** — Target & RESETLOGS

Recovery scope and point-in-time recovery

- **Database PITR** restores and recovers the database to a chosen time, SCN or log sequence while mounted, then opens **RESETLOGS**. It needs suitable backups and redo, not prior activation of Flashback Database.
- **RESETLOGS** starts a new database incarnation. Retain incarnation metadata and establish a tested backup strategy after recovery; avoid treating backup age alone as proof that a restore path is valid.
- **PDB media recovery** can leave unrelated containers open while the affected PDB or file is unavailable. The required open state and connection scope depend on the damaged file and local versus shared undo.
- **PDB PITR** isolates recovery to one PDB. Shared-undo recovery can require an auxiliary environment to recover undo dependencies; local undo simplifies isolation. Perform CDB-wide operations from the root.

Flashback features and prerequisites

- **Flashback Query** uses AS OF for an earlier row image; **Flashback Version Query** uses VERSIONS BETWEEN for row history. Both depend on available undo. **Flashback Transaction Query** exposes transaction metadata and undo SQL.
- **Flashback Table** reverses eligible table data changes using undo and requires row movement plus appropriate flashback and DML privileges. It cannot cross structural barriers such as TRUNCATE merely because undo retention is guaranteed.
- **Flashback Drop** retrieves eligible dropped tables and dependent objects from the **recycle bin**. Purged objects and dropped tablespaces are outside its protection; it does not use undo to reconstruct dropped segments.
- **Flashback Data Archive** retains tracked table history in archive storage. It serves longer-term historical queries rather than relying only on ordinary undo retention.
- **Flashback Database** uses flashback logs to rewind physical blocks and redo to reach the exact target. It requires ARCHIVELOG, an FRA and adequate flashback history; it can reverse logical damage that table-level flashback cannot cross.

Restore points and flashback storage

- A **normal restore point** names an SCN; it does not guarantee that recovery data remains available. A **guaranteed restore point** preserves required flashback logs and can consume enough FRA space to stop database progress.
- **DB_FLASHBACK_RETENTION_TARGET** expresses desired history, not an unconditional recovery guarantee. Space pressure and guaranteed restore points affect log retention; obsolete flashback logs can also be reclaimed proactively.

Recovery & Flashback

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

CHOOSE RECOVERY SCOPE

- **Instance** — Online redo
- **Media** — Restore & recover
- **Flashback** — Rewind history
- **PITR** — Target & RESETLOGS

- **PDB restore points** identify recovery positions for an individual PDB. A **clean PDB restore point** has no outstanding transactions in that PDB and is useful for flashback with shared undo.
- **FLASHBACK DATABASE** changes the CDB's physical history; **FLASHBACK PLUGGABLE DATABASE** targets one PDB under its prerequisites. Undo retention guarantee does not replace flashback logs or protect against physical file loss.

Corruption, table recovery and diagnostics

- **VALIDATE** and **BACKUP VALIDATE** check file readability and corruption without creating a backup. **CHECK LOGICAL** adds supported logical block checks; inspect **V$DATABASE_BLOCK_CORRUPTION** for recorded corrupt ranges.
- **Block media recovery** repairs selected physically corrupt blocks while leaving the data file online; affected blocks remain unavailable during repair. It does not repair every form of logical corruption or require Flashback Database to be enabled.
- **Automatic block repair** can obtain a good block from an eligible Active Data Guard partner. It requires the supported primary/physical-standby configuration; enabling block checking alone does not create a repair source.
- **RMAN RECOVER TABLE** uses an auxiliary database and Data Pump to recover eligible tables to an earlier point. It can handle structural changes that prevent Flashback Table, but excludes SYS-owned tables and tables in SYSTEM or SYSAUX.
- The **ADR** stores diagnostic files in product-instance homes; **ADRCI** inspects them and packages incidents. **Data Recovery Advisor** uses failure metadata with LIST, ADVISE and REPAIR workflows where supported.
- Preserve the complete **RMAN error stack**, including command, channel and underlying ORA/media-manager errors. **LOG** directs RMAN output to a file; use **SPOOL LOG** when terminal output is also needed.

Duplication and recovery essentials

- **DUPLICATE** requires an **auxiliary instance**. Backup-based duplication can work without a live target connection when backups and the required metadata are available; active duplication transfers data from a running source.
- **Active duplication** can push image copies or pull backup sets, depending on the method. Prepare auxiliary startup, authentication, network access and file names; do not assume RMAN can infer every prerequisite.
- An ordinary **duplicate** receives a new DBID; a **standby duplicate** preserves the primary's DBID. PDB-to-new-CDB duplication and cloning into an existing CDB have different scope and command restrictions.
- A **NOARCHIVELOG** duplicate needs a consistent backup and **NOREDO**. For parameter-file loss, create a PFILE or SPFILE **FROM MEMORY** while the instance survives; re-create a password file with **orapwd** when necessary.

Install & Upgrade

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

CONTROLLED CHANGE

- **Assess** — Prerequisites
- → **Protect** — Recovery path
- → **Change** — Home & dictionary
- → **Verify** — Services & applications

04 Installation, upgrades and data movement

Linux installation and software homes

- **ORACLE_BASE** identifies the installation owner's base; **ORACLE_HOME** identifies a software home; **ORACLE_SID** selects a local instance. **TNS_ADMIN** locates Net configuration, while **TMP/TMPDIR** identify installation scratch space.
- The **Oracle preinstallation RPM** installs prerequisite packages and configures common kernel settings, limits, the oracle user and standard groups. It does not finish ASM storage configuration or replace every installation prerequisite check.
- **oinstall** owns central inventory access; **OSDBA** grants database administrative authentication; **OSASM** grants ASM administration. Software-owner names are configurable, and role separation can use different database and Grid owners.
- **Image-based installation** extracts software into the intended home, then runs the setup wizard. An **RPM-based installation** installs software through the package manager; its configuration script creates the database separately.
- **Root scripts** such as orainstRoot.sh and root.sh perform privileged setup. The installer can automate supported execution using root or sudo credentials; configure the method and sudo path without storing privileged passwords in a response file.
- A **read-only Oracle home** separates binaries from writable configuration and logs. Enable it with **roohctl -enable** after software-only installation and before database creation; it does not require RHP.

DBCA, Grid Infrastructure and Restart

- **DBCA** creates databases from templates, with interactive and silent workflows. Templates can describe structure only or include data-file content. Silent workflows also support remote CDB duplication and remote PDB cloning or relocation under their prerequisites.
- **Character sets** determine representable characters and byte requirements. **AL32UTF8** supports Unicode multilingual data; database and national character sets serve different SQL data types. DBCA is not a general existing-database character-set conversion tool.
- **Grid Infrastructure for a standalone server** includes **Oracle Restart** and **ASM**. Restart monitors registered local resources and starts dependencies in order; use **SRVCTL** for managed resources rather than bypassing dependency management.

Install & Upgrade

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

CONTROLLED CHANGE

Assess Prerequisites → **Protect** Recovery path → **Change** Home & dictionary → **Verify** Services & applications

- **DBCA/ASMCA** integrate supported created resources with Restart; resources created manually can require **srvctl add** registration. Creating a database service only with DBMS_SERVICE or SERVICE_NAMES is not equivalent to a managed service resource.
- **ASM** mounts disk groups and manages ASM file space and allocation units. Database processes normally perform database-file I/O directly; ASM does not allocate the table and index extents inside data files.

Patching and software provisioning

- **OPatch** applies supported interim binary patches. **OPatchAuto** orchestrates supported Grid and database-home patching, including required stops and restarts; follow patch-specific root, rolling and prerequisite instructions.
- A **patch set** can introduce functionality and require database upgrade processing. Distinguish it from an interim patch or release update, and use the installation procedure supplied for that patch type.
- **datapatch** applies required SQL changes after binary patching. Validate inventory and SQL patch status; successful file replacement alone does not prove dictionary patching completed.
- **Rapid Home Provisioning** manages gold images and working copies for supported database, Grid, application and middleware software. Provisioning a new home can occur before the running database is switched to it.
- **rhpctl move database** moves a database between suitable working-copy homes, such as a patched home or a previous home for a supported rollback. Moving binaries, applying patches and upgrading the database dictionary are distinct operations.
- **Grid Infrastructure upgrades** use a separate new home and supported source-to-target paths. Existing Oracle base directories can be reused; do not assume the installer supplies every mandatory patch or that every patch permits rolling operation.

Database upgrade workflow

- Start **DBUA** from the target home. Run preupgrade checks, resolve reported issues and plan tested restore/rollback protection. Upgrade eligibility depends on the source release; platform or character-set migration needs a suitable migration method.
- The **preupgrade tool** reports component, parameter and space issues and generates fixups. Ensure intended PDBs are available for assessment; a closed PDB must not silently escape the upgrade plan.
- **dbupgrade/catctl.pl** controls manual upgrade execution and parallelism. Upgrade the root and seed before user PDBs; priority lists control selected PDB order and inclusion, so distinguish explicitly listed PDBs from omitted ones.
- After upgrade, inspect component status, errors and invalid objects; **utlrp.sql** recompiles invalid PL/SQL. Verify services, time-zone handling and application compatibility, and upgrade a separate **RMAN catalog** when required.

Install & Upgrade / Performance

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

CONTROLLED CHANGE

Assess Prerequisites → **Protect** Recovery path → **Change** Home & dictionary → **Verify** Services & applications

Logical migration and cross-platform transport

- **Data Pump** moves schemas and data between non-CDBs and PDBs or between PDBs. Connect to the destination PDB's service; import does not create a CDB or automatically provision the destination PDB.
- **Logical migration** using export/import, CTAS or supported tools changes database contents. An operating-system copy alone cannot migrate selected heap tables into an unrelated database; DBUA upgrades a dictionary rather than moving table rows.
- **Transportable tablespaces** move data files plus object metadata. Check self-containment, character-set compatibility and platform format; the conventional method makes source tablespaces read-only during transport.
- **V$TRANSPORTABLE_PLATFORM** identifies endianness. Tablespace data files can be converted on the source or destination with the supported RMAN procedure. **RMAN CONVERT DATABASE** requires source and destination to have the same endianness.
- **Cross-platform backup sets** can combine RMAN data-file transport with Data Pump metadata through supported **DATAPUMP** clauses. Keep data files and metadata synchronized; backup-set transport and manual EXPDP/IMPDP procedures have different steps.
- **Non-CDB to PDB migration** can use logical or transportable methods; direct plugging requires a supported compatible source and conversion procedure. A release too old for direct plugging may still be a source for a supported logical migration.

05 Performance diagnosis and tuning

Measure workload impact first

- Begin with **service-level goals**, representative workload and general OS health. Check CPU saturation, memory pressure and storage latency, then use the **database time model** to identify where database work spends time.
- **DB time** combines foreground CPU and non-idle waits; **DB CPU** measures foreground CPU consumption. Compare an affected interval with a useful baseline rather than optimizing isolated counters.
- Tune the dominant workload cause, validate the user-visible improvement and stop when agreed goals are met. Poor SQL and long-held transactions can limit concurrency; changing one statement can also harm competing work.

Waits, metrics and alerts

- **V$SESSION** and **V$SESSION_WAIT** show current or most recent wait information; interpret STATE and event fields together. **V$SESSION_EVENT** accumulates events for an existing session rather than preserving every past session indefinitely.
- **V$SESSION_WAIT_CLASS** aggregates a session's waits by class; **V$SERVICE_EVENT** aggregates by service. Compare wait time, count and workload context before calling an event a bottleneck.

Performance

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

TUNING CYCLE

Measure Goals & baseline → **Diagnose** DB time & waits → **Tune** Dominant cause → **Validate** Workload response

- **Metrics** measure statistics over intervals. **Server-generated alerts** compare supported metrics with thresholds; TYPICAL statistics collection is sufficient for normal manageability features, without requiring STATISTICS_LEVEL=ALL.
- **Stateful alerts** remain outstanding until the condition clears and then appear in **DBA_ALERT_HISTORY**. **Stateless alerts** record one-time events in history; they have no continuing threshold state to clear.

AWR, ADDM and workload comparison

- **AWR** stores performance history in SYSAUX. Defaults are hourly snapshots and eight-day retention; settings and baselines can change retention. **STATISTICS_LEVEL=TYPICAL/ALL** supports normal automatic collection; BASIC disables much of it.
- **MMON** coordinates automatic snapshots and manageability tasks. **ADDM** analyzes the interval between snapshots and can also run on request, reporting findings and recommendations rather than merely writing an alert-log message.
- **ADDM impacts** may overlap, so adding finding percentages can exceed 100% of DB time. Evaluate recommendations in workload context, including memory, SQL, I/O and contention.
- **CDB and PDB AWR** have distinct collection scopes. PDB-local snapshots and reports are supported with appropriate configuration; do not assume every container's history resides only in root SYSAUX.

Optimizer statistics and SQL advisors

- **DBMS_STATS** gathers table, column, index and system statistics. Row counts, selectivity, index height and clustering influence estimated access costs; stale statistics can still be used and do not represent live row-by-row facts.
- **Automatic optimizer statistics collection** operates through maintenance tasks. In multitenant databases, gather object statistics in the intended container; a root DBMS_STATS call is not an instruction to gather every PDB's objects.
- **Optimizer Statistics Advisor** evaluates statistics-collection practices and recommends improvements through DBMS_STATS. The optimizer chooses plans using available statistics, predicates and valid hints; it does not obey every hint or gather all missing statistics itself.
- **SQL Tuning Advisor** analyzes individual statements and can recommend statistics, indexes, SQL restructuring and **SQL profiles**. A profile supplies auxiliary information to improve estimates; it does not freeze one execution plan.
- **SQL Access Advisor** considers workload-level access structures. **SQL Performance Analyzer** compares SQL performance before and after a change and identifies regressions; use SQL Tuning Advisor for affected statements. **Database Replay** tests workload concurrency more directly.

Performance

Oracle Database Administration II

KNOWLEDGE CHEATSHEET

TUNING CYCLE

Measure Goals & baseline → **Diagnose** DB time & waits → **Tune** Dominant cause → **Validate** Workload response

Memory models and advisors

- **AMM**, controlled by **MEMORY_TARGET**, can balance memory between SGA and PGA. **ASMM**, controlled by **SGA_TARGET**, redistributes eligible SGA components; PGA remains separately managed.
- With manual SGA sizing, configure components individually. **PGA_AGGREGATE_TARGET** guides automatic SQL work areas; **PGA_AGGREGATE_LIMIT** is an aggregate limit, not a per-session allocation.
- **Sorts and hash joins** use work areas that can spill to temporary storage. Dedicated-server session state is normally in the PGA; shared-server **UGA** resides in the SGA, typically the large pool when configured.
- Use the **shared-pool advisor** for hard-parse/library-cache pressure, **buffer-cache advisor** for cache sizing and **PGA advisor** for work-area spills. SGA and memory-target advisors match ASMM and AMM; select the advisor for the active memory model.

RMAN performance diagnosis

- A backup pipeline has **read**, **copy/transform** and **write** phases. Compression and encryption can make CPU the limit even when storage is fast; locate the limiting phase before adding channels.
- Compare **BACKUP VALIDATE** with an actual backup: similar elapsed time points toward input or processing cost; a much faster validation suggests output-side cost. Inspect **V$BACKUP_ASYNC_IO**, **V$BACKUP_SYNC_IO** and session waits.
- **Asynchronous disk I/O**, balanced file placement, suitable multiplexing and multisection backups can improve input concurrency. More channels do not help once disks, CPUs or tape drives are saturated.
- **I/O slaves** can simulate asynchronous behavior on supported synchronous-I/O configurations. Buffer-cache enlargement and FORCE LOGGING changes do not directly solve an RMAN data-file read bottleneck.