LEVEL 4 INTERVIEW QUESTIONS AND ANSWERS

Oracle Database Patching | Exadata Patching | Oracle Database Appliance Patching

Expanded Command-Rich Edition

Target audience	Level 4 / Lead / Principal Oracle DBA
Coverage	Oracle Database, RAC, Data Guard, Exadata and ODA
Content	63 questions, model answers, command examples, validation and rollback
Prepared	July 2026

Important: Commands are illustrative templates. Replace placeholders and follow the patch README and release-specific Oracle documentation. ODA command sequencing changes between appliance releases.

How to Use This Guide

- Answer the question conceptually first, then explain the commands you would use and why.
- For production scenarios use this structure: protect service, capture evidence, determine patch state, choose a supported recovery path, validate, and communicate.
- Never run an example blindly. Confirm the exact patch README, target release, home ownership, cluster state, backup status, and rollback capability.
- Commands containing placeholders such as <target_version>, <db_unique_name>, and <patch_id> must be replaced with environment-specific values.

Contents

- Part I - Oracle Database Patching (Q1-Q18)
- Part II - Exadata Patching (Q19-Q36)
- Part III - Oracle Database Appliance Patching (Q37-Q54)
- Part IV - Governance and Senior Scenarios (Q55-Q63)
- Appendix - Consolidated Command Checklists and References

Part I - Oracle Database Patching

Q1. What is the difference between a Release Update, Monthly Recommended Patch, one-off patch and merge patch?

Answer: A Release Update (RU) is Oracle's cumulative quarterly maintenance bundle for a database release. It normally contains security fixes, regression fixes, optimizer fixes and functional corrections. Monthly Recommended Patches (MRPs) provide additional recommended fixes on top of an RU for supported releases and are also cumulative within their stream. A one-off patch addresses a specific bug and must be checked for conflicts against the installed RU and other one-offs. A merge patch is produced when two or more required patches conflict and Oracle combines them into a compatible patch. A Level 4 DBA should standardise on an approved RU baseline, minimise unmanaged one-offs, and document every exception.

Command examples

```
$ORACLE_HOME/OPatch/opatch lspatches
SELECT patch_id, patch_type, action, status, action_time FROM dba_registry_sqlpatch ORDER BY action_time;
```

Q2. Explain binary patching and SQL patching.

Answer: Binary patching changes files in an Oracle home and is performed with OPatch, OPatchAuto, image-based home deployment or an appliance lifecycle tool. SQL patching changes database dictionary objects and is performed by datapatch after the patched home is active and the database is open. A patch can be present in OPatch inventory while still missing from DBA_REGISTRY_SQLPATCH. Therefore both inventories must be verified before declaring success.

Command examples

```
$ORACLE_HOME/OPatch/opatch apply
sqlplus / as sysdba <<EOF
startup
alter pluggable database all open;
EOF
$ORACLE_HOME/OPatch/datapatch -verbose
```

Q3. What checks do you perform before patching an Oracle Database home?

Answer: Confirm platform, release and patch compatibility; read the patch README; verify support status; compare OPatch version with the minimum required version; run opatch lsinventory; check free space in the Oracle home, central inventory, /tmp and archive destinations; run conflict and prerequisite checks; validate backups and restore tests; check invalid objects and component status; record services, PDB open states and listener configuration; validate RAC, Data Guard and application dependencies; capture AWR baselines and critical SQL plans; and produce a timed rollback plan with named owners.

Command examples

```
$ORACLE_HOME/OPatch/opatch version
$ORACLE_HOME/OPatch/opatch lsinventory -detail
$ORACLE_HOME/OPatch/opatch lspatches
df -h $ORACLE_HOME /tmp
sqlplus / as sysdba
$ORACLE_HOME/OPatch/opatch prereq CheckConflictAgainstOHWithDetail -phBaseDir /stage/PATCH
srvctl status database -d <db_unique_name>
crsctl check cluster -all
```

Q4. How do you check patch conflicts before applying an RU or one-off patch?

Answer: Use the exact commands specified in the patch README. Common checks include `opatch prereq CheckConflictAgainstOHWithDetail -phBaseDir <patch_dir>` for a patch directory and `opatchauto apply <patch_dir> -analyze` for supported GI/RAC system patches. Review conflicts against installed patches, not only the base release. If a required one-off conflicts with the RU, search for an RU-compatible replacement or request a merge patch. Never force an apply merely to bypass a conflict.

Command examples

```
cd /stage/PATCH
$ORACLE_HOME/OPatch/opatch prereq CheckConflictAgainstOHWithDetail -phBaseDir .
$GRID_HOME/OPatch/opatchauto apply /stage/PATCH -analyze
```

Q5. When would you use OPatch and when would you use OPatchAuto?

Answer: OPatch patches an individual Oracle home and is commonly used for database home one-offs or manual out-of- place image construction. OPatchAuto orchestrates system patching for supported Grid Infrastructure and RAC configurations, including sequencing, privilege transitions, service management and rolling execution where supported. OPatchAuto should be run from the Grid home version specified by the patch documentation and normally requires root privileges. The patch README remains authoritative.

Command examples

```
$ORACLE_HOME/OPatch/opatch apply /stage/ONEOFF
$GRID_HOME/OPatch/opatchauto apply /stage/SYSTEM_PATCH -analyze
$GRID_HOME/OPatch/opatchauto apply /stage/SYSTEM_PATCH
```

Q6. Describe in-place versus out-of-place database patching.

Answer: In-place patching modifies the existing Oracle home. It is simple but increases rollback complexity because the home must be restored or the patch rolled back. Out-of-place patching builds a new Oracle home at the target RU, applies approved one-offs, validates it, and then moves databases and listeners to it. Out-of-place patching provides cleaner rollback, reduces home drift, supports parallel preparation and is generally preferred for critical systems. The DBA must still preserve configuration files, inventory registration, ownership, permissions and application library dependencies.

Command examples

```
unzip -q LINUX.X64_193000_db_home.zip -d /u01/app/oracle/product/19c/dbhome_2
$NEW_HOME/OPatch/opatch apply /stage/RU
srvctl modify database -d <db_unique_name> -oraclehome $NEW_HOME
srvctl start database -d <db_unique_name>
```

Q7. How do you patch a two-node RAC database with minimum downtime?

Answer: Use a rolling method only when the patch is certified rolling. Drain or relocate services from node 1, stop the local instance if required, patch and validate the node's Grid and database homes in the documented order, restart resources, verify cluster health and service placement, then repeat on node 2. Maintain application capacity during each step and monitor global cache, connection pools and service failover. Run datapatch only at the correct stage after all required binaries are patched and the database is open, unless the patch documentation states otherwise.

Command examples

```
srvctl relocate service -d <db_unique_name> -s <service> -oldinst <inst1> -newinst <inst2>
```

```
srvctl stop instance -d <db_unique_name> -i <inst1> -stopoption immediate
opatchauto apply /stage/PATCH -oh $GRID_HOME,$ORACLE_HOME
srvctl start instance -d <db_unique_name> -i <inst1>
```

Q8. What is the purpose of datapatch and how do you run it safely?

Answer: Datapatch applies post-patch SQL changes to the CDB root, seed and applicable PDBs and records results in DBA_REGISTRY_SQLPATCH. Run `$ORACLE_HOME/OPatch/datapatch -verbose` from the active patched database home after the database is open in the required mode. Ensure intended PDBs are open, confirm sufficient TEMP and SYSTEM/SYSAUX capacity, preserve the logs under cfgtoollogs, and investigate any `WITH ERRORS` status. Recompile invalid objects after correcting genuine failures, not as a substitute for diagnosis.

Command examples

```
sqlplus / as sysdba <<EOF
show pdbs
alter pluggable database all open;
EOF
$ORACLE_HOME/OPatch/datapatch -sanity_checks
$ORACLE_HOME/OPatch/datapatch -verbose
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q9. How do you validate that a database patch was fully applied?

Answer: Validate the active home and executable, `opatch lspatches`, `opatch lsinventory`, `DBA_REGISTRY_SQLPATCH`, `CDB_REGISTRY` or `DBA_REGISTRY`, invalid objects, alert log, listener and service state, RAC resources, PDB open modes and application smoke tests. Check `datapatch -sanity_checks` where supported. Level 4 Oracle, Exadata and ODA Patching Interview Guide For critical systems, compare performance against the prepatch AWR baseline and verify backup, monitoring and Data Guard transport/apply.

Command examples

```
$ORACLE_HOME/OPatch/opatch lsinventory -detail
SELECT patch_id, patch_uid, action, status, description, action_time FROM dba_registry_sqlpatch ORDER BY action_time;
SELECT comp_id, comp_name, version, status FROM dba_registry ORDER BY comp_id;
utlrp.sql
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q10. What causes datapatch failures?

Answer: Frequent causes include closed PDBs, wrong ORACLE_HOME, insufficient tablespace or TEMP, invalid component state, dictionary locks, failed prerequisite SQL, Java or XML DB component problems, incorrect environment, unsupported patch combinations and interrupted earlier runs. Review sqlpatch logs and DBA_REGISTRY_SQLPATCH details, correct the root cause and rerun datapatch. Do not manually mark a patch successful.

Command examples

```
$ORACLE_HOME/OPatch/datapatch -sanity_checks
$ORACLE_HOME/OPatch/datapatch -verbose
grep -iE "error|ORA-|failed" $ORACLE_BASE/cfgtoollogs/sqlpatch/sqlpatch_*/sqlpatch_invocation.log
```

Q11. How do you rollback a database patch?

Answer: First distinguish binary rollback from SQL rollback. Stop affected resources according to the README, use OPatch or OPatchAuto rollback with the patch identifier, restore the prior home if out-of-place patching was used, start the database from the correct home, and run datapatch so SQL rollback actions are applied where supported. Validate both inventories and application function. Some patches are not fully reversible; the approved recovery route may instead be home restoration, guaranteed restore point, database restore or Data Guard failback.

Command examples

```
$ORACLE_HOME/OPatch/opatch rollback -id <patch_id>
$GRID_HOME/OPatch/opatchauto rollback /stage/PATCH
$ORACLE_HOME/OPatch/datapatch -verbose
```

Q12. How should Data Guard be handled during database patching?

Answer: Use a documented standby-first or primary-first plan based on patch rolling capability and business requirements. Patch the standby binaries, preserve apply requirements, patch the primary during the maintenance phase, and execute datapatch on the primary at the documented point after all necessary homes are patched. SQL changes then flow through redo. Do not run datapatch against a physical standby. Confirm transport, apply, broker status, protection mode and switchover readiness before and after patching.

Command examples

```
dgmgrl / "show configuration"
dgmgrl / "show database verbose <standby_unique_name>"
srvctl stop database -d <standby_unique_name> -stopoption immediate
$ORACLE_HOME/OPatch/opatch apply /stage/PATCH
dgmgrl / "validate database verbose <standby_unique_name>"
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q13. What is the role of the OPatch inventory and how do you troubleshoot inventory errors?

Answer: The local Oracle home inventory and central inventory record installed components and patches. Inventory errors can result from wrong ownership, stale locks, incorrect oraInst.loc, a detached home, filesystem corruption or an incomplete clone. Validate `opatch lsinventory -detail`, central inventory permissions and home registration. Use supported attachHome or inventory repair procedures only after backups and Oracle guidance; never edit inventory XML casually.

Command examples

```
$ORACLE_HOME/OPatch/opatch lsinventory -detail
cat /etc/oraInst.loc
grep -n "HOME NAME" $ORACLE_HOME/inventory/ContentsXML/comps.xml
runInstaller -silent -attachHome ORACLE_HOME=$ORACLE_HOME ORACLE_HOME_NAME=<home_name>
```

Q14. How do you patch a multitenant database?

Answer: Patch the Oracle home once, then use datapatch to update the root, seed and open PDBs. Save and record PDB states before maintenance, open all intended PDBs for SQL patching, and query CDB_REGISTRY_SQLPATCH by container after completion. A PDB that was closed may remain at an older SQL patch level and must be opened and datapatch rerun. Check unplug/plug compatibility when PDBs move between homes or CDBs.

Command examples

```
SELECT name, open_mode FROM v$pdbs ORDER BY con_id;
alter pluggable database all open;
$ORACLE_HOME/OPatch/datapatch -verbose
SELECT con_id, patch_id, action, status FROM cdb_registry_sqlpatch ORDER BY action_time;
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q15. How do you manage patching when several databases share one Oracle home?

Answer: Inventory every database, service, listener and agent dependency using the home. Agree a common outage or migrate databases progressively to a prepared target home. Out-of-place homes reduce the blast radius and allow per- database cutover. Ensure all databases receive the SQL patch, including rarely opened PDBs and standby configurations. Do not patch a shared home based on only one database owner's approval.

Command examples

```
srvctl config database
ps -ef | grep pmon
$ORACLE_HOME/OPatch/opatch lsinventory
srvctl stop database -d <db_unique_name> -stopoption immediate
```

Q16. What post-patch performance checks do you perform?

Answer: Compare database time, top waits, CPU, I/O latency, parse rates, hard parse, library cache activity, cluster waits, redo generation, log file sync, critical SQL elapsed time and execution plans against a representative baseline. Review new optimizer fix controls and patch notes. Use SQL Plan Management for critical statements and avoid immediately disabling fixes without evidence. Confirm that monitoring thresholds were not distorted by the maintenance period.

Command examples

```
SELECT * FROM table(dbms_xplan.display_cursor('<sql_id>',NULL,'ALLSTATS LAST +PEEKED_BINDS'));
@?/rdbms/admin/awrrpt.sql
SELECT event, total_waits, time_waited_micro/1e6 seconds FROM v$system_event ORDER BY time_waited_micro DESC FETCH FIRST 20 ROWS ONLY;
```

Q17. Scenario: OPatch shows the RU, but DBA_REGISTRY_SQLPATCH does not. What do you do?

Answer: This means the binaries are patched but database SQL actions are incomplete. Confirm that the database is running from the patched home, open all intended PDBs, review prior sqlpatch logs, run datapatch with the correct environment, and verify SUCCESS in DBA_REGISTRY_SQLPATCH. If datapatch reports nothing to apply, verify the database's actual executable path and inventory. Treat the database as incompletely patched until both layers agree.

Command examples

```
$ORACLE_HOME/OPatch/opatch lspatches
$ORACLE_HOME/OPatch/datapatch -verbose
SELECT patch_id, action, status, logfile FROM dba_registry_sqlpatch ORDER BY action_time;
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q18. Scenario: a RAC rolling patch fails on node 1 halfway through. What is your response?

Answer: Freeze progression to node 2, keep services on the healthy node, capture OPatchAuto session details and logs, determine whether the failed node is pre-change, partially changed or binary-complete, and use the

documented resume or rollback action. Validate CRS and home inventories before returning the node to service. If consistency cannot be guaranteed, isolate the node and engage Oracle Support rather than attempting arbitrary manual repairs. Level 4 Oracle, Exadata and ODA Patching Interview Guide

Command examples

```
crsctl stat res -t
srvctl status database -d <db_unique_name>
$GRID_HOME/OPatch/opatchauto resume
$GRID_HOME/OPatch/opatchauto rollback /stage/PATCH
```

Part II - Exadata Patching

Q19. What components are normally considered in an Exadata patching programme?

Answer: Database and Grid Infrastructure homes, Exadata database server operating system image, Exadata System Software on storage servers, RDMA Network Fabric switches, management switches where applicable, ILOM and component firmware, AHF and ExaCLI-related tooling. The exact bundle and sequence depend on the target Exadata release and hardware generation. Database RU patching and Exadata infrastructure image patching are related but separate maintenance streams.

Command examples

```
imageinfo -ver
dcli -g ~/dbs_group -l root imageinfo -ver
dcli -g ~/cell_group -l root imageinfo -ver
ibswitches || true
```

Q20. What is patchmgr and what does it patch?

Answer: Patchmgr is Oracle's orchestration utility for Exadata infrastructure updates. Depending on the supplied update package and invocation, it coordinates updates for storage servers, database servers and supported RDMA Network Fabric switches. It performs prerequisite checks, stages files, sequences nodes, validates health and supports rollback or cleanup operations where documented. Use the patchmgr version delivered with or required by the target update.

Command examples

```
./patchmgr --help
./patchmgr --dbnodes dbs_group --precheck --repo /stage/exadata.zip --target_version <target>
./patchmgr -cells cell_group -patch_check_prereq -rolling
```

Q21. Why must you use the patchmgr packaged for the target update?

Answer: Patchmgr logic evolves with image layouts, firmware, prerequisite rules and supported hardware. An older utility may not understand a newer target image, while an unrelated patchmgr can use incompatible workflows. Oracle documentation also distinguishes utilities used for storage-server updates from those used for database-server updates. Always follow the target update README and maintenance guide.

Command examples

```
unzip -q p<PATCH>.zip -d /stage/patchmgr
cd /stage/patchmgr
./patchmgr --help
```

Q22. Describe a safe high-level Exadata patching sequence.

Answer: Establish the supported source-to-target path; run exachk and health checks; resolve hardware, ASM, cluster and network faults; verify backups and rescue plans; update tooling; perform patchmgr prechecks; patch components in Oracle's documented sequence; validate each rolling batch; patch Grid and database homes as a coordinated but separate workstream; run datapatch at the correct stage; and complete exachk, performance, backup, Data Guard and application validation. Never invent a universal sequence independent of the release documentation.

Command examples

```
exachk -profile exadata
```

```
./patchmgr -cells cell_group -patch_check_prereq -rolling
./patchmgr -cells cell_group -patch -rolling
./patchmgr --dbnodes dbs_group --precheck --repo /stage/exadata.zip --target_version <target>
```

Q23. What prechecks are essential before patching Exadata storage servers?

Answer: Confirm all cells are healthy and reachable, no critical alerts exist, griddisks and cell disks are normal, ASM disk groups have adequate redundancy and rebalance capacity, no rebalance or repair operation is active, flash and physical disks are healthy, IORM configuration is backed up, cell services are stable, and cluster/database workload can tolerate rolling cell restarts. Run patchmgr precheck and capture `list cell detail`, alert history and relevant metrics.

Command examples

```
cellcli -e "list cell detail"
cellcli -e "list griddisk attributes name,status,asmModeStatus"
cellcli -e "list physicaldisk attributes name,status"
./patchmgr -cells cell_group -patch_check_prereq -rolling
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q24. How can Exadata storage cells be patched with databases online?

Answer: Supported storage-server updates are commonly rolling. Patchmgr updates one cell or a safe subset at a time while ASM redundancy and database connectivity maintain service. The DBA must confirm disk-group redundancy, partner-disk placement, no degraded disks, and sufficient performance headroom because remaining cells carry more workload. Rolling capability reduces outage but does not mean zero risk or zero performance impact.

Command examples

```
./patchmgr -cells cell_group -patch_check_prereq -rolling
./patchmgr -cells cell_group -patch -rolling
dcli -g cell_group -l root imageinfo -ver
```

Q25. What is Exadata Live Update?

Answer: Exadata Live Update is a database-server update capability supported for eligible Exadata releases and configurations that can reduce or avoid database-server reboot impact for selected updates. Patchmgr determines eligibility and executes the supported workflow. A senior DBA should not promise live update solely from the target version; kernel, source image, configuration and update content must meet the documented criteria, and a conventional rolling reboot plan must remain available.

Command examples

```
imageinfo -ver
./patchmgr -cells cell_group -patch_check_prereq -rolling
tail -f /var/log/cellos/cellos.log
```

Q26. How do you patch Exadata database servers?

Answer: Use the supported Exadata database-server update package and patchmgr or the documented dbnodeupdate workflow. Run prechecks, verify local and remote repositories or staging, drain database services from the target node, preserve quorum and cluster capacity, update nodes sequentially, reboot where required, and

validate OS image, GI, interfaces, multipath/RDMA, CRS and database services after each node. Do not use generic operating-system yum updates outside the supported Exadata process.

Command examples

```
./patchmgr --dbnodes dbs_group --backup --repo /stage/exadata.zip --target_version <target>
./patchmgr --dbnodes dbs_group --precheck --repo /stage/exadata.zip --target_version <target>
./patchmgr --dbnodes dbs_group --upgrade --repo /stage/exadata.zip --target_version <target> --rolling
```

Q27. How do you patch RDMA Network Fabric switches?

Answer: Use the Exadata-supported patchmgr switch workflow and exact target firmware path. Validate redundant fabric paths and switch health, confirm no cabling or port faults, back up switch configuration, update one switch at a time where the design supports it, and verify links, subnet manager state, partitioning and database/cell connectivity before continuing. Network maintenance is a high-risk dependency and should include console access and a recovery procedure.

Command examples

```
./patchmgr --roceswitches roce_group --precheck --repo /stage/switch.zip --target_version <target>
./patchmgr --roceswitches roce_group --upgrade --repo /stage/switch.zip --target_version <target>
show version
```

Q28. What is the difference between Exadata infrastructure image patching and Oracle Database RU patching?

Answer: Infrastructure image patching updates Exadata database-server OS and firmware, storage-server software and sometimes fabric components. Database RU patching updates Grid Infrastructure and Oracle Database homes plus Level 4 Oracle, Exadata and ODA Patching Interview Guide database SQL components. Infrastructure image version does not prove that database homes are current, and an RU inventory does not prove cells or switches are current. Both baselines must be recorded independently.

Command examples

```
imageinfo -ver
$ORACLE_HOME/OPatch/opatch lspatches
SELECT patch_id, description, status FROM dba_registry_sqlpatch ORDER BY action_time;
```

Q29. What tools do you use for Exadata patch health validation?

Answer: Exachk for best-practice and health assessment; patchmgr precheck and status; CellCLI for cell, disk, griddisk, alert and metric state; `crsctl`, `srvctl` and ASM views for cluster resources; `dcli` for controlled read-only checks; ExaWatcher for operating-system and network evidence; AHF/TFA for incident collection; and switch-specific commands for fabric verification. Use dcli carefully and avoid uncontrolled write commands across all nodes.

Command examples

```
exachk -profile exadata
dcli -g cell_group -l root imageinfo -ver
cellcli -e "list alerthistory where severity != clear"
crsctl stat res -t
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q30. What conditions would make you postpone Exadata patching?

Answer: A failed or predictive-failure disk, degraded ASM redundancy, cell or switch alerts, unstable RDMA links, cluster node eviction, unresolved filesystem capacity, invalid inventory, failed patchmgr prechecks, missing rescue media, inadequate backup/standby protection, unsupported source-to-target path, or insufficient workload capacity for rolling maintenance. Patching is not a repair substitute for a pre-existing unhealthy platform.

Command examples

```
cellcli -e "list alerthistory"
asmcmd lsdg
crsctl check cluster -all
dcli -g dbs_group -l root "df -h / /u01 /var"
```

Q31. How do you estimate Exadata patching impact?

Answer: Assess component count, rolling versus non-rolling operations, reboot requirements, current and target image gap, workload capacity after removing one node or cell, ASM rebalance state, application connection behavior, Data Guard role, firmware update duration, and validation time. Use timings from recent comparable systems only as input; precheck, hardware state and site-specific workload determine the actual window.

Command examples

```
iostat -xm 5 12
cellcli -e "list metriccurrent where objectType = CELL"
dcli -g dbs_group -l root uptime
srvctl status service -d <db_unique_name>
```

Q32. What is your rollback strategy for an Exadata infrastructure patch?

Answer: Use only rollback paths supported by the target release. Preserve configuration and diagnostic baselines, retain the required update packages and patchmgr version, confirm rescue and reimage procedures, and know when a component rollback is supported versus when Oracle Support must guide recovery. For rolling failure, stop the rollout, stabilise the patched component, preserve redundancy and avoid creating mixed-version states beyond those explicitly supported.

Command examples

```
./patchmgr -cells cell_group -rollback_check_prereq -rolling
./patchmgr -cells cell_group -rollback -rolling
./patchmgr --dbnodes dbs_group --rollback --repo /stage/exadata.zip --target_version <previous>
```

Q33. How do you coordinate Grid Infrastructure and database patching on Exadata?

Answer: Treat GI/database home RUs as Oracle Database software maintenance and Exadata images as platform maintenance, but coordinate dependencies and outages. Prepare out-of-place homes where possible, ensure target RUs are certified with the Exadata image, patch GI using supported RAC rolling procedures, patch database homes, run datapatch, and then verify Smart Scan, ASM, services and offload behavior. Do not assume patchmgr patches arbitrary database homes.

Command examples

```
crsctl query crs activeversion
crsctl query crs softwareversion -all
$GRID_HOME/OPatch/opatch lspatches
$ORACLE_HOME/OPatch/opatch lspatches
```

Q34. Scenario: patchmgr precheck reports a cell disk problem. What do you do?

Answer: Stop the patching plan. Validate the physical disk, cell disk, griddisk and ASM disk status, review alerts and confirm redundancy. Replace or repair the failed component through the supported Exadata procedure, allow ASM and cell operations to complete, rerun exachk and patchmgr precheck, and proceed only after the platform is healthy. Forcing the update could remove another failure domain and cause data unavailability.

Command examples

```
cellcli -e "list physicaldisk attributes name,status,errMediaCount,errOtherCount"
cellcli -e "list griddisk attributes name,status,asmModeStatus"
asmcmd lsdsk -p
./patchmgr -cells cell_group -patch_check_prereq -rolling
```

Q35. Scenario: performance drops while one storage cell is being patched. How do you respond?

Answer: Check whether the remaining cells are saturated for flash, hard-disk I/O, CPU or network; validate ASM and cell alerts; monitor Smart Scan and database wait events; and determine whether the workload has enough rolling headroom. Pause before updating the next cell. Reschedule heavy jobs, adjust the maintenance batch size only within supported rules, or move workload using services/Data Guard. Do not continue merely because the current cell update succeeded technically.

Command examples

```
cellcli -e "list metriccurrent where objectType = CELL"
cellcli -e "list alerthistory"
asmcmd lsdg
dcli -g cell_group -l root uptime
```

Q36. Scenario: an Exadata database server does not rejoin the cluster after update.

Answer: Use console/ILOM access, verify boot and target image, check network and RDMA interfaces, time synchronisation, storage visibility, GI startup and voting/OCR access. Review patchmgr, OS, CRS and AHF logs. Keep services on surviving nodes and do not patch another node. Use the documented resume/rollback path or Oracle Support with a TFA collection if the node's state is uncertain. Level 4 Oracle, Exadata and ODA Patching Interview Guide

Command examples

```
crsctl check crs
crsctl stat res -t
systemctl --failed
journalctl -b -p err
imageinfo -ver
```

Part III - Oracle Database Appliance Patching

Q37. What components are patched in an ODA lifecycle update?

Answer: ODA lifecycle updates can include DCS administration and service components, ODA server software, operating system, local-disk firmware, ILOM firmware, Grid Infrastructure, shared-storage firmware where applicable, database homes and databases. Commands and component coverage vary by ODA release and model, so the release-specific deployment guide and README are authoritative.

Command examples

```
/opt/oracle/dcs/bin/odacli describe-component
/opt/oracle/dcs/bin/odacli list-jobs
/opt/oracle/dcs/bin/odacli list-dbhomes
```

Q38. Explain the purpose of `odacli update-dcsadmin`, `update-dcscomponents`, `update-server`, `update-storage` and `update-dbhome`.

Answer: `update-dcsadmin` updates the DCS administration component. `update-dcscomponents` updates core DCS lifecycle services and associated metadata components. `update-server` updates the appliance server stack and, depending on release, OS, firmware, ILOM and Grid Infrastructure. `update-storage` updates supported shared-storage firmware. `update-dbhome` patches a database home and databases using it. Exact scope must be confirmed in the release documentation.

Command examples

```
odacli update-dcsadmin -v <target_version>
odacli update-dcscomponents -v <target_version>
odacli update-servercomponents -v <target_version>
odacli update-gihome -v <target_version>
odacli update-dbhome -i <dbhome_id> -v <target_version>
```

Q39. What is the role of the ODA patch repository?

Answer: The repository stores the unpacked ODA software bundles used by lifecycle commands. Before patching, download the correct server and database clone/update bundles from My Oracle Support, validate checksums, copy them to the appliance and update the repository with the documented `odacli update-repository` workflow. Confirm the target version appears in `odacli describe-component` or repository-related output before starting updates.

Command examples

```
odacli update-repository -f /stage/oda_patch_bundle.zip
odacli describe-component
odacli cleanup-patchrepo
```

Q40. What is an ODA prepatch report and why is it important?

Answer: The prepatch report validates readiness for a specific target version and component scope. It checks hardware, services, configuration, space and other prerequisites and records failures or warnings before changes begin. Generate the report for server, storage or database-home scope using the command syntax for the installed release. Resolve failures rather than routinely using force options.

Command examples

```
odacli create-prepatchreport -s -v <target_version>
```

```
odacli list-prepatchreports
odacli describe-prepatchreport -i <report_id>
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q41. Describe the recommended high-level sequence for ODA patching.

Answer: Verify the supported source-to-target path and model; read known issues; take database backups and an ODABR snapshot where supported; update the repository; update DCS administration and DCS components in the documented order; generate and clear prepatch reports; update server components, storage if required, and then database homes/databases; monitor jobs to completion; and validate appliance, GI, databases, backups and application services. Some releases change command behavior, so do not reuse an old runbook unmodified.

Command examples

```
odacli update-repository -f /stage/oda_patch_bundle.zip
odacli create-prepatchreport -s -v <target_version>
odacli update-dcsadmin -v <target_version>
odacli update-dcscomponents -v <target_version>
odacli update-servercomponents -v <target_version>
odacli update-gihome -v <target_version>
```

Q42. What is ODABR and how is it used in patching?

Answer: Oracle Database Appliance Backup and Recovery provides system-level snapshot and recovery capabilities for the appliance's boot and system volumes on supported releases. Taking an ODABR snapshot before patching is a strong best practice because it provides an additional recovery option for server-stack failures. It does not replace database backups, Data Guard or application rollback planning.

Command examples

```
odabr backup -create
odabr backup -list
odabr backup -restore -key <backup_key>
```

Q43. How do you monitor ODA patching jobs?

Answer: Capture the job identifier returned by odacli, use `odacli describe-job -i <job_id>` and, where useful, `odacli list- jobs` to follow tasks and failures. Review DCS logs under the documented locations and preserve the job report before retrying. A parent job can contain multiple subtasks; identify the first real failure rather than focusing only on the final generic error.

Command examples

```
odacli list-jobs
odacli describe-job -i <job_id>
tail -f /opt/oracle/dcs/log/dcs-agent.log
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q44. What is the difference between `odacli update-dbhome` and `odacli update-database`?

Answer: `update-dbhome` patches a selected database home and the databases associated with it according to the ODA lifecycle workflow. `update-database` moves or updates a selected database to a specific later database home

within the supported major release, enabling more granular sequencing. The choice depends on ownership, consolidation, downtime and whether multiple databases share the home.

Command examples

```
odacli list-dbhomes
odacli update-dbhome -i <dbhome_id> -v <target_version>
odacli update-database -i <database_id> -to <destination_dbhome_id>
```

Q45. Can you apply an out-of-cycle database one-off patch on ODA?

Answer: Only patches and methods explicitly supported for the installed ODA release should be used. Oracle documents a process for supported out-of-cycle database patches, typically involving the relevant database home and OPatch, while Grid Infrastructure and appliance-stack patching remain controlled by the ODA lifecycle framework. Record the one-off because it may conflict with the next quarterly ODA bundle and require removal or a merge.

Command examples

```
$ORACLE_HOME/OPatch/opatch prereq CheckConflictAgainstOHWithDetail -phBaseDir /stage/ONEOFF
$ORACLE_HOME/OPatch/opatch apply /stage/ONEOFF
$ORACLE_HOME/OPatch/datapatch -verbose
```

Q46. Why should you not patch ODA Grid Infrastructure independently with a generic RU?

Answer: ODA is an engineered appliance with a tested software stack. Grid Infrastructure is tied to the ODA server update and certification matrix. Applying an arbitrary GI RU can create an unsupported stack and break future lifecycle operations. Use the ODA bundle and odacli workflow unless Oracle provides explicit exception instructions. Level 4 Oracle, Exadata and ODA Patching Interview Guide

Command examples

```
odacli describe-component
odacli create-prepatchreport -g -v <target_version>
odacli update-gihome -v <target_version>
```

Q47. How do you patch an ODA high-availability pair with minimum downtime?

Answer: Confirm both nodes and shared storage are healthy, verify cluster and database service failover, update DCS components as documented, run server prechecks, and allow the ODA workflow to perform its supported rolling node sequence. Monitor service relocation and client reconnection. Patch database homes in a controlled sequence and run functional checks after each phase. Do not manually reboot or relocate resources while an odacli lifecycle job owns the workflow unless directed.

Command examples

```
odacli create-prepatchreport -s -v <target_version>
odacli update-servercomponents -v <target_version>
odacli list-jobs
crsctl stat res -t
```

Q48. What checks do you perform before ODA server patching?

Answer: Check `odacli describe-component`, appliance and job status, DCS service health, cluster resources, database and Data Guard status, filesystem capacity, repository content, hardware alerts, storage health, ILOM reachability, backups and ODABR snapshot, model-specific known issues and prepatch reports. Clear stale failed jobs only through supported methods and verify no other lifecycle task is running.

Command examples

```
odacli describe-component
odacli create-prepatchreport -s -v <target_version>
odacli list-jobs
df -h / /opt /u01
oakcli validate -c storagetopology || true
```

Q49. How do you handle a failed ODA patching job?

Answer: Do not immediately rerun or use force. Describe the job, identify the failed task, review DCS and component logs, check current component versions on both nodes, and determine whether the workflow supports resume or rerun. Correct the prerequisite or environmental problem, then use the documented retry action. If nodes show different component states or metadata is inconsistent, open an Oracle Support request before manual intervention.

Command examples

```
odacli describe-job -i <job_id>
odacli list-jobs
grep -iE "error|failed|exception" /opt/oracle/dcs/log/dcs-agent.log
odacli create-prepatchreport -s -v <target_version>
```

Q50. What validation is required after ODA server patching?

Answer: Confirm all components report the target version, DCS services are healthy, both nodes and ILOM are reachable, GI resources are online, shared storage and ASM are normal, databases and listeners use the expected homes, services are correctly placed, backups run, Data Guard is healthy and monitoring has resumed. Review patch job subtasks and appliance logs for hidden warnings.

Command examples

```
odacli describe-component
odacli list-jobs
crsctl stat res -t
srvctl status database -d <db_unique_name>
ipmitool mc info
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q51. What validation is required after ODA database-home patching?

Answer: Verify the new home version and inventory, database home association, database startup, listener and services, PDB states, datapatch status in DBA_REGISTRY_SQLPATCH, invalid components/objects, alert logs, backups and application tests. For RAC databases, verify both instances are using the same intended home and that no service remains tied to an obsolete home.

Command examples

```
odacli list-dbhomes
odacli list-databases
$ORACLE_HOME/OPatch/opatch lspatches
SELECT patch_id, description, action, status FROM dba_registry_sqlpatch ORDER BY action_time;
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q52. Scenario: ODA prepatch report fails for insufficient space. What do you do?

Answer: Identify the exact filesystem or repository requirement, remove only documented obsolete bundles, logs or old homes after confirming they are unused, and preserve rollback material. Recheck database home dependencies before deleting anything. Rerun the prepatch report. Do not bypass the check, because server and firmware updates can fail catastrophically when staging or boot volumes fill.

Command examples

```
odacli describe-prepatchreport -i <report_id>
df -h / /opt /u01
du -sh /opt/oracle/oak/pkgrepos/* 2>/dev/null
odacli cleanup-patchrepo
```

Q53. Scenario: `update-server` completes on node 1 but fails on node 2.

Answer: Keep the healthy node serving workload, capture the failed job and component versions, check node 2 boot, ILOM, network, DCS and GI status, and follow the documented resume/rerun procedure. Do not initiate database-home patching or another server update while the pair is mixed beyond the supported rolling window. Escalate if metadata and actual versions disagree.

Command examples

```
odacli list-jobs
odacli describe-job -i <job_id>
crsctl stat res -t
tail -200 /opt/oracle/dcs/log/dcs-agent.log
```

Q54. Scenario: `update-dbhome` fails during datapatch.

Answer: Confirm the binary home patch succeeded, identify affected databases and containers, inspect the odacli job and sqlpatch logs, correct closed PDB, space, invalid component or locking problems, and use the ODA-supported retry workflow. Verify DBA_REGISTRY_SQLPATCH for every container. Avoid manually changing ODA metadata or registering success outside the lifecycle process. Level 4 Oracle, Exadata and ODA Patching Interview Guide

Command examples

```
odacli describe-job -i <job_id>
$ORACLE_HOME/OPatch/datapatch -sanity_checks
$ORACLE_HOME/OPatch/datapatch -verbose
SELECT patch_id, status, logfile FROM dba_registry_sqlpatch ORDER BY action_time;
```

Part IV - Architecture, Governance and Senior Scenarios

Q55. How do you build an enterprise quarterly patching standard?

Answer: Define approved RU and appliance baselines, a release calendar, severity-driven exceptions, inventory ownership, non-production soak periods, regression tests, security risk acceptance, standard prechecks, backup and rollback evidence, change templates, maintenance sequencing for primary/standby systems, validation criteria and compliance reporting. Track binary and SQL patch levels separately and prevent undocumented one-offs.

Command examples

```
opatch lsinventory -detail > lsinventory_before.txt
exachk -profile exadata -o pre
odacli create-prepatchreport -s -v <target_version>
sha256sum /stage/*patch*
```

Q56. How do you choose between rolling patching, Data Guard switchover and full outage?

Answer: Use rolling patching when certified and application capacity can tolerate one node or cell being unavailable. Use Data Guard role transition when site-level isolation, home replacement or near-zero primary downtime is required and the standby is validated. Use a controlled outage when patch semantics are non-rolling, capacity is insufficient, or consistency risk outweighs availability. The decision must consider recoverability, not only downtime.

Command examples

```
srvctl status service -d <db_unique_name>
dgmgrl / "show configuration"
srvctl relocate service -d <db_unique_name> -s <service> -oldinst <inst1> -newinst <inst2>
```

Q57. What evidence should be attached to a patching change record?

Answer: Approved patch list and READMEs, support/certification checks, current and target inventories, precheck reports, health checks, backup and restore evidence, Data Guard status, application test plan, command transcript, job IDs, start/end timestamps, rollback triggers, postpatch inventories, DBA_REGISTRY_SQLPATCH output, exachk or ODA validation, incident notes and stakeholder sign-off.

Command examples

```
date; hostname; uname -a
opatch lsinventory -detail
crsctl stat res -t
SELECT patch_id, action, status, action_time FROM dba_registry_sqlpatch ORDER BY action_time;
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Q58. How do you prevent configuration drift across RAC, Exadata and ODA fleets?

Answer: Maintain a configuration management database, discover homes and component versions automatically, compare them to approved baselines, use image-based homes and appliance lifecycle tools, centrally store one-off exceptions, reconcile OPatch and SQL patch inventories, schedule recurring compliance scans, and block new maintenance if the previous change evidence is incomplete.

Command examples

```
dcli -g dbs_group -l root imageinfo -ver
```

```
dcli -g cell_group -l root imageinfo -ver
odacli describe-component
opatch lspatches
```

Q59. How do you manage an emergency security patch outside the quarterly cycle?

Answer: Confirm applicability and exploit exposure, obtain the correct patch or merge, assess rolling capability and conflicts, test against the current RU, prioritise internet-facing and high-value databases, create a short but complete rollback plan, and patch through normal supported tools. Afterward, update the approved baseline and decide whether the emergency one- off must be removed or replaced at the next RU.

Command examples

```
opatch prereq CheckConflictAgainstOHWithDetail -phBaseDir /stage/EMERGENCY
opatch apply /stage/EMERGENCY
datapatch -verbose
opatch rollback -id <patch_id>
```

Q60. Scenario: the business asks you to skip backups because the patch is rolling. How do you answer?

Answer: Rolling describes service availability, not recoverability. A rolling operation can still introduce binary inconsistency, dictionary failure, firmware failure, latent corruption or application regression. I would not proceed without the backup, standby and rollback controls required by policy. I would offer risk-based alternatives such as validated Data Guard, storage snapshot plus database backup, or rescheduling, but I would document and escalate any exception.

Command examples

```
rman target / <<EOF
backup database plus archivelog;
restore database validate;
EOF
odabr backup -create
./patchmgr --dbnodes dbs_group --backup --repo /stage/exadata.zip --target_version <target>
```

Q61. Scenario: all technical steps pass but application response time is worse after patching.

Answer: Preserve the new and old baselines, compare SQL plans, optimizer environment, wait events, CPU and I/O, identify whether regression is broad or statement-specific, and check patch-related optimizer fixes. Stabilise critical SQL using SQL Plan Management or a verified SQL patch/profile where appropriate. If the regression is systemic and cannot be safely mitigated within the window, execute the approved home or role rollback. Do not roll back solely on anecdotal reports without measuring, but do not dismiss user impact because infrastructure checks are green.

Command examples

```
@?/rdbms/admin/awrrpt.sql
SELECT * FROM table(dbms_xplan.display_cursor('<sql_id>',NULL,'ALLSTATS LAST'));
sqlplus / as sysdba <<EOF
select name,value from v$parameter where ismodified <> 'FALSE';
EOF
```

Q62. Scenario: a one-off needed by the application conflicts with the next RU.

Answer: Determine whether the bug fix is included in the RU or superseded by a new patch. If not, obtain an RU-compatible one-off or merge patch from Oracle. Test removal and replacement in a cloned home. Do not apply the RU with a force option or retain an incompatible binary combination. Update the exception register and future baseline.

Command examples

```
opatch prereq CheckConflictAgainstOHWithDetail -phBaseDir /stage/RU
opatch lsinventory -detail
opatch rollback -id <old_oneoff_id>
opatch apply /stage/MERGE_PATCH
```

Q63. What distinguishes a Level 4 DBA during patching?

Answer: A Level 4 DBA understands the internals and commands but also controls risk across database, cluster, storage, network and application layers. They challenge unhealthy prerequisites, choose the right availability strategy, preserve evidence, diagnose partial states without improvising unsupported repairs, communicate decision points clearly, and improve the fleet process after every maintenance event. Level 4 Oracle, Exadata and ODA Patching Interview Guide Selected Command Reference Purpose Example Oracle inventory $ORACLE_HOME/OPatch/opatch lsinventory $ORACLE_HOME/OPatch/opatch lspatches Conflict check $ORACLE_HOME/OPatch/opatch prereq CheckConflictAgainstOHWithDetail -phBaseDir <patch_dir> Datapatch $ORACLE_HOME/OPatch/datapatch -verbose SQL patch verification SELECT con_id, patch_id, action, status, action_time, description FROM cdb_registry_sqlpatch ORDER BY action_time; RAC resources crsctl stat res -t srvctl status database -db <db_unique_name> Exadata health exachk cellcli -e "list cell detail" cellcli -e "list alerthistory" ODA component versions /opt/oracle/dcs/bin/odacli describe-component ODA jobs /opt/oracle/dcs/bin/odacli list-jobs /opt/oracle/dcs/bin/odacli describe-job -i <job_id> ODA prepatch report /opt/oracle/dcs/bin/odacli create-prepatchreport <release-specific options> Level 4 Oracle, Exadata and ODA Patching Interview Guide Oracle Documentation References ⁃ Oracle Database Patch Maintenance, Oracle Database 19c: https://docs.oracle.com/en/database/oracle/oracle- database/19/dbptc/ ⁃ DBA_REGISTRY_SQLPATCH Reference: https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/ DBA_REGISTRY_SQLPATCH.html ⁃ Oracle Exadata Database Machine Maintenance Guide: https://docs.oracle.com/en/engineered-systems/exadata- database-machine/dbmmn/ ⁃ Exadata Patchmgr Update Utility: https://docs.oracle.com/en/engineered-systems/exadata-database-machine/dbmmn/ exadata-patchmgr-update-utility.html ⁃ Oracle Database Appliance 19.31 Patching Guide: https://docs.oracle.com/en/engineered-systems/oracle-database- appliance/19.31/ ⁃ Oracle Database Appliance Command-Line Interface: https://docs.oracle.com/en/engineered-systems/oracle-database-appliance/19.31/daten/oracle-database-appliance-command-line-interface2.html Reference note: The guide is interview-oriented and intentionally summarises operational concepts. Consult My Oracle Support patch READMEs, release notes, known issues and certification data before executing maintenance.

Command examples

```
opatchauto apply /stage/PATCH -analyze
exachk -profile exadata
odacli create-prepatchreport -s -v <target_version>
SELECT patch_id, action, status FROM dba_registry_sqlpatch ORDER BY action_time;
```

Validation emphasis: Do not declare success from the tool return code alone. Confirm cluster/service state, binary inventory, SQL registry, component health, logs, and application-level testing.

Appendix A - Oracle Database Patching Checklist

- export ORACLE_HOME=/u01/app/oracle/product/19c/dbhome_2
- export PATH=$ORACLE_HOME/OPatch:$ORACLE_HOME/bin:$PATH
- opatch version
- opatch lsinventory -detail
- opatch prereq CheckConflictAgainstOHWithDetail -phBaseDir /stage/PATCH
- opatch apply /stage/PATCH
- sqlplus / as sysdba
- alter pluggable database all open;
- datapatch -sanity_checks
- datapatch -verbose
- SELECT patch_id, description, action, status, action_time FROM dba_registry_sqlpatch ORDER BY action_time;

Appendix B - Exadata Patching Checklist

- exachk -profile exadata
- dcli -g dbs_group -l root imageinfo -ver
- dcli -g cell_group -l root imageinfo -ver
- ./patchmgr -cells cell_group -patch_check_prereq -rolling
- ./patchmgr -cells cell_group -patch -rolling
- ./patchmgr --dbnodes dbs_group --precheck --repo /stage/exadata.zip --target_version <target>
- ./patchmgr --dbnodes dbs_group --upgrade --repo /stage/exadata.zip --target_version <target> --rolling
- cellcli -e "list alerthistory"
- crsctl stat res -t

Appendix C - ODA Patching Checklist

- odacli update-repository -f /stage/oda_patch_bundle.zip
- odacli describe-component
- odacli create-prepatchreport -s -v <target_version>
- odacli describe-prepatchreport -i <report_id>
- odacli update-dcsadmin -v <target_version>
- odacli update-dcscomponents -v <target_version>
- odacli update-servercomponents -v <target_version>
- odacli update-gihome -v <target_version>
- odacli update-dbhome -i <dbhome_id> -v <target_version>
- odacli list-jobs
- odacli describe-job -i <job_id>

References

- Oracle Database OPatch and OPatchAuto documentation: https://docs.oracle.com/en/database/
- Oracle Exadata Database Machine Maintenance Guide - patchmgr: https://docs.oracle.com/en/engineered-systems/exadata-database-machine/
- Oracle Database Appliance Deployment and User Guide: https://docs.oracle.com/en/engineered-systems/oracle-database-appliance/
- My Oracle Support patch README files and release-specific known-issues notes remain authoritative.