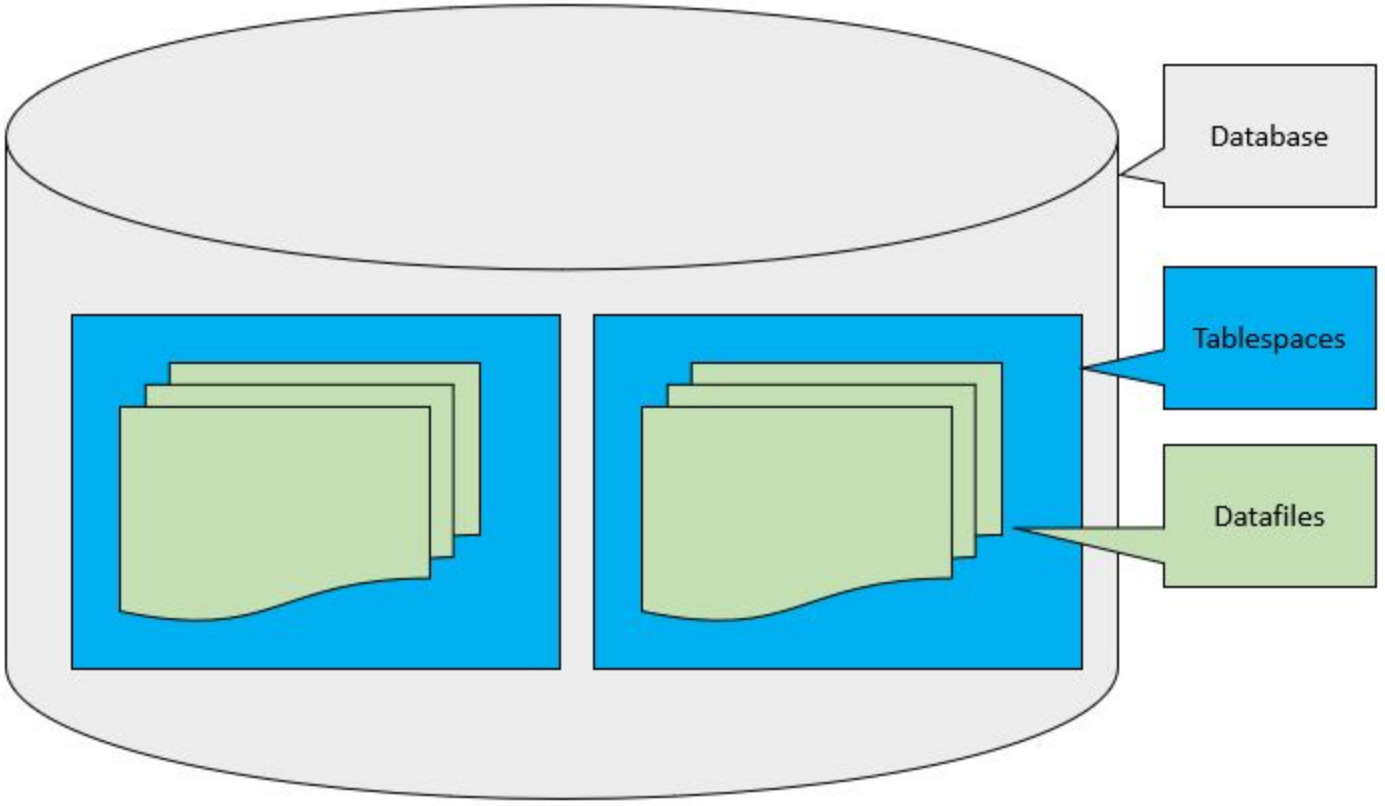
2. Tablespace and Storage Management

- Working with Tablespaces and Data Files, Creating and adding tablespace and datafiles,
- Managing Control Files, Online Redo Logs and Archive logs; Multiplexing;

Tablespaces



Tablespaces

- In Oracle Database, a tablespace is a logical storage container that is used to manage the storage of database objects such as tables, indexes, and other data structures. Tablespaces provide a way to allocate and organize storage resources within the database, and they play a crucial role in database administration, storage management, and performance optimization. Here are some key points about tablespaces in Oracle:
- Logical Storage Container: A tablespace is a logical container for database objects, and it is not tied directly to physical storage devices. This means that a tablespace can span multiple physical data files, and the database administrator can control the allocation and distribution of storage within tablespaces.

Tablespaces

- Types of Tablespaces: Oracle supports several types of tablespaces, including:
  - System Tablespace: This is the default tablespace that stores the database's data dictionary and system-related information. It contains metadata about the database structure.
  - User Tablespaces: These tablespaces are used to store user data, such as tables, indexes, and other objects created by database users.
  - Temporary Tablespaces: Temporary tablespaces are used for sorting and temporary storage of data during SQL operations like sorting and joining. They are essential for query processing.
  - Undo Tablespaces: Undo tablespaces store transaction undo information, allowing for rollback of changes made during transactions. They are crucial for maintaining data consistency.
  - Temporary Undo Tablespaces: These tablespaces store temporary undo data, primarily used in Oracle Real Application Clusters (RAC) environments.
  - Read-Only and Read-Write Tablespaces: Tablespaces can be set as read-only, preventing any modifications to the data stored in them.

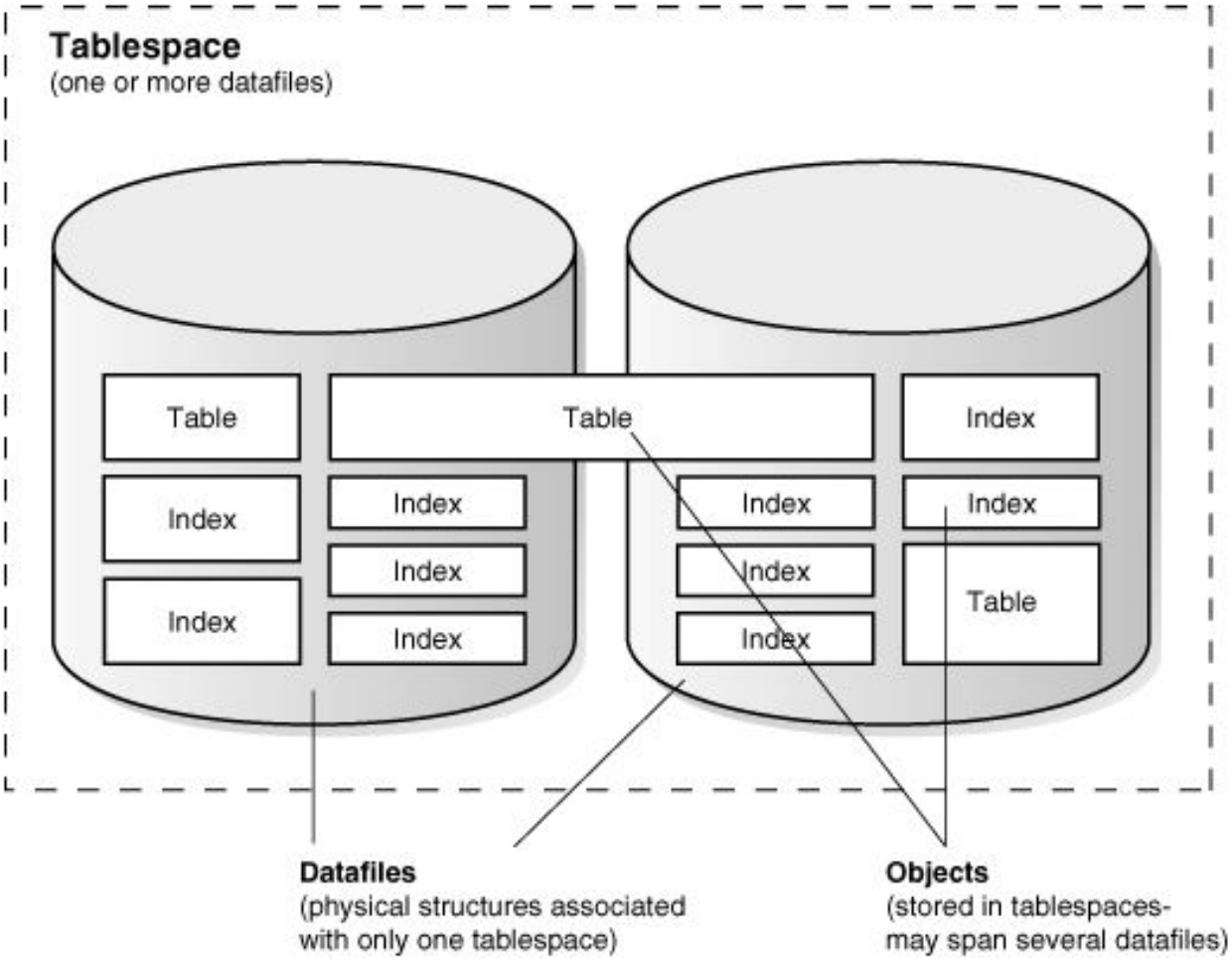
Tablespaces

- Tablespace Management: Oracle provides SQL commands and tools to create, alter, and drop tablespaces. Administrators can also move or reorganize segments within tablespaces to optimize database performance.
- Tablespace Quotas: User accounts can be assigned quotas on tablespaces to limit the amount of space they can use within a specific tablespace.
- Tablespaces in Oracle are a fundamental concept in database administration, providing a logical organization for data storage and enabling efficient management of database storage resources. Understanding how to create, manage, and allocate tablespaces is essential for Oracle database administrators.

Tablespaces

- Data Files: Each tablespace is composed of one or more data files, which are physical files on disk where the actual data is stored. Data files are associated with a specific tablespace.
- Segments: Tablespaces are further divided into segments, which correspond to specific types of database objects. Segments include tables, indexes, and other structures. Each segment is stored within one or more data files.
- Storage Management: Database administrators can manage storage allocation, sizing, and placement by adding or resizing data files within tablespaces. This flexibility allows for efficient storage management and optimization of performance.

Datafiles


Tablespace
(one or more datafiles)
Table
Table
Index
Index
Index
Index
Index
Index
Index
Index
Index
Table
Datafiles
(physical structures associated
with only one tablespace)
Objects
(stored in tablespaces-
may span several datafiles)

Datafiles

- In Oracle Database, datafiles are physical files that store the actual data and database objects, such as tables, indexes, and tablespaces. Datafiles are a critical component of the database's physical storage structure. Here are some key points to understand about datafiles in Oracle:
- Datafile Types:
  - Data Datafiles: These files store the actual data for database tables and other user-defined objects.
  - Index Datafiles: These files store indexes created on database tables to optimize query performance.
  - Temporary Datafiles: Temporary datafiles are used for temporary storage during operations like sorting and joining data. They are part of temporary tablespaces and are used to support database operations.

Datafiles

- Here are some key points to understand about datafiles in Oracle:
  - Physical Storage: Each datafile corresponds to a specific tablespace within the database. The datafile is the physical representation of that tablespace on the underlying storage media, such as hard drives or solid-state drives.
  - Attributes: Datafiles have attributes such as file name, file size, and autoextend settings. Autoextend allows a datafile to automatically grow when it reaches its allocated size limit.
  - Datafile Locations: Datafiles can be stored on different types of storage media, including local disk drives, network-attached storage (NAS), or storage area networks (SAN). The location of datafiles can impact performance and availability.
  - Read-Only and Read-Write Datafiles: Datafiles can be set as read-only or read-write. Read-only datafiles are used in scenarios where you want to prevent any changes to the data stored in them.

Datafiles

- Here are some key points to understand about datafiles in Oracle:
  - Datafile Management: Database administrators can manage datafiles by adding new datafiles to tablespaces, resizing existing datafiles, or taking datafiles offline for maintenance.
  - Online and Offline Datafiles: Datafiles can be online or offline. Online datafiles are accessible for read and write operations, while offline datafiles are not accessible and typically used during maintenance or recovery operations.

Datafiles

- Here are some key points to understand about datafiles in Oracle:
  - Recovery: Datafiles are crucial for database recovery. Oracle maintains a record of changes made to datafiles in the online redo log files, allowing for recovery in case of media failures or other issues.
  - Backup and Restore: Datafiles are often included in database backup and restore procedures. Backup copies of datafiles are used to recover the database to a specific point in time.
  - Multiple Datafiles per Tablespace: A single tablespace can consist of multiple datafiles, which allows for spreading the data across multiple physical storage devices and improving performance and availability.
  - Managing datafiles effectively is a fundamental aspect of Oracle database administration. DBAs need to monitor datafile sizes, ensure appropriate backup and recovery strategies, and optimize datafile placement for optimal database performance and availability.

Control Files

- In Oracle Database, control files are critical components of the database's structure and play a vital role in the management and operation of the database. Control files are used to keep track of the database's physical structure, including datafiles, redo log files, and other structural information. Here are key points about control files in Oracle:
- Purpose of Control Files:
  - Control files serve as the "brain" of the Oracle database. They store metadata about the database's physical structure, such as datafile and redo log file information, checkpoints, database name, and time stamps.

Control Files

- Contents of Control Files: Control files contain information about the following:
  - The database's unique name (DB_NAME) and unique identifier (DBID).
  - The names and locations of datafiles and redo log files.
  - The current online redo log group.
  - Database checkpoints, which indicate the consistent state of the database.
  - Archive log history, which records information about archived redo log files.
  - Backup history, including details about database backups.
  - Control file metadata is critical for database recovery and management.

Control Files

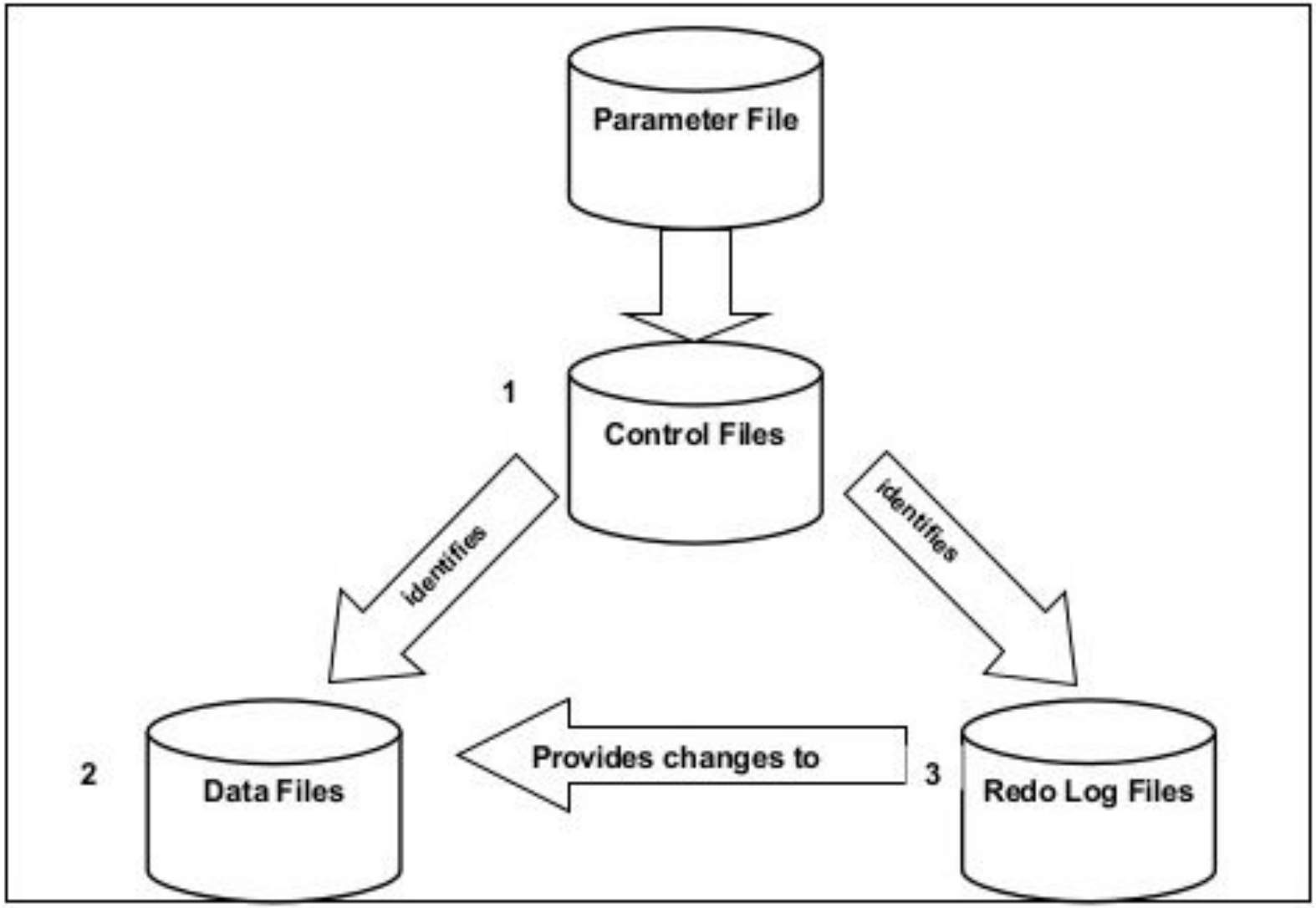
- Multiple Control Files:
  - Oracle recommends having multiple control files (typically three) for redundancy and fault tolerance. If one control file becomes inaccessible or corrupt, the database can still function using the remaining control files.
- Location of Control Files:
  - Control files are usually placed in separate, non-system tablespaces for security and manageability. The paths to control files are specified in the initialization parameter file (init.ora or SPFILE).
- Control File Backups:
  - It is essential to regularly back up control files to ensure database recoverability. Control file backups can be taken using Oracle's RMAN (Recovery Manager) or through manual methods.

Control Files

- Creating or Recreating Control Files:
  - If control files are lost, damaged, or need to be recreated, you can do so using RMAN or by creating a new control file script. However, recreating control files should be done with extreme caution, as it can be a complex and risky operation.
- Control File Multiplexing:
  - Multiplexing control files means having multiple copies of control files in different locations. This provides redundancy and fault tolerance. Control files can be multiplexed when creating a new database or added later.
- Control File Management:
  - Database administrators can manage control files using SQL*Plus, Enterprise Manager, or RMAN. Common operations include adding, dropping, or renaming control files.

Control Files

- Control File Size:
  - The size of control files depends on the number of datafiles, redo log files, and other structural elements in the database. Care should be taken to ensure that control files have adequate space.
- Backup and Recovery:
  - Control files are essential for database backup and recovery operations. Without a valid control file, it is challenging to restore and recover a database to a consistent state.
- Control files in Oracle Database are crucial for database management, recovery, and operation. They store critical metadata about the database's physical structure, and having multiple copies of control files is a best practice to ensure the database's availability and reliability.


Parameter File
1
Control Files
identifies
identifies
2
Data Files
Provides changes to
3
Redo Log Files

Control Files

- Control file must be available for writing by the Oracle Database server whenever the database is open. Without control file, the database cannot be mounted and recovery is difficult.
- It is created at the same time as database. By default, at least one copy of control file is created but we should create two or more copies of control file during database creation and also if we lose a control file or want to change particular settings in the control files.

Managing Control Files: Managing Size of Control files:

- The main determinants of the size of a control file are the values set for the
  - MAXDATAFILES,
  - MAXLOGFILES,
  - MAXLOGMEMBERS,
  - MAXLOGHISTORY, and
  - MAXINSTANCES

Multiplexing Control Files

- Since the control file is critical for the database operation, Oracle recommends a minimum of two control files. You duplicate the control file on different disks either by using the multiplexing feature of Oracle or by using the mirroring feature of your operating system.
- Multiplexing control files refers to a technique used in database management systems, particularly in the context of Oracle Database. Control files are critical components of a database that store metadata and information necessary for the database's operation. Multiplexing control files involves creating and maintaining multiple copies of these control files to ensure data redundancy and recoverability in case of failures.

Multiplexing Control Files

- Here's an overview of multiplexing control files:
  - Control Files: Control files contain vital information about the database, including the database name, data file and log file names, timestamps, and the database's structural information. These files are essential for database recovery, startup, and normal operation.
  - Multiplexing: Multiplexing control files means creating multiple copies of the control file and storing them in different locations. This redundancy is important for data reliability and disaster recovery.

Multiplexing Control Files

- Benefits:
  - Fault Tolerance: Multiplexing control files helps in safeguarding against control file failures. If one control file becomes corrupted or inaccessible due to hardware or software issues, the database can still operate using the copies.
  - Data Recovery: In the event of a control file failure, Oracle can use the remaining copies to recover the database's structural information, which is crucial for data consistency.
  - Reduced Downtime: With multiplexed control files, you can quickly replace a failed control file without significant downtime or data loss.

Multiplexing Control Files

- Multiplexing Strategies:
  - Two or More Copies: It's common to have at least two copies of the control file, but some organizations may choose to maintain three or more copies for added redundancy.
  - Different Disks: Store copies of control files on different physical disks or storage devices to ensure that a single disk failure doesn't impact all copies simultaneously.
  - Different Locations: It's a best practice to keep control file copies in separate physical locations or data centers to protect against disasters like fires, floods, or earthquakes.
- Managing Multiplexed Control Files: Oracle Database provides tools and commands to create, maintain, and manage multiplexed control files. You can add, drop, or relocate control file copies using SQL commands or tools like Oracle Enterprise Manager.
- Multiplexing control files is a critical part of database administration to ensure data integrity and recoverability. It's an essential practice for maintaining the availability and reliability of an Oracle Database.

Creating Control Files:

1. Creating initial control files
2. Creating additional copies, renaming and relocating control files
3. Creating new control files

- **Creating Initial control file**
  - Initial control file is created when you issue the CREATE DATABASE statement. The names of control files are specified by CONTROL_FILES parameter in the initialization parameter file during database creation.
  - If file with same name already exist, you must specify CONTROL FILE REUSE clause in the CREATE DATABASE statement or else error will occur. If size of old control file differ from size parameter of new one, you cannot use REUSE clause.

Creating additional copies, renaming and relocating control files

- Steps for multiplexing or renaming a control file:
  1. Shut down the database
  2. Copy existing control file from old location to new location using operating system commands
  3. Edit the CONTROL_FILES parameter in database initialization parameter file to add the new control file name, or renaming control filename, or specifying new location for multiplexing
  4. Restart the database

Creating new control file

- You can create a new control file for database using CREAE CONTROLFILE command. This is recommended in the following situations:
  - All control files for the database have been permanently damaged and you do not have a control file backup.
  - You want to change one of the permanent database settings originally specified in the CREATE DATABASE statement, including the database's name, MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, and MAXINSTANCES.

Creating new control file

- Example
- CREATE CONTROLFILE

```
SET DATABASE prod
LOGFILE GROUP 1 ('logfile1A', 'logfile1B') SIZE 50K,
        GROUP 2 ('logfile2A', 'logfile2B') SIZE 50K
  NORESETLOGS DATAFILE 'datafile1' SIZE 3M,
  'datafile2' SIZE 5M
MAXLOGFILES 50
MAXLOGMEMBERS 3
MAXDATAFILES 200
MAXINSTANCES 6
ARCHIVELOG;
```

Steps for creating new control files

- Make a list of all datafiles and redo log files of the database.
  - SELECT MEMBER FROM V$LOGFILE;
  - SELECT NAME FROM V$DATAFILE;
  - SELECT VALUE FROM V$PARAMETER WHERE NAME = 'control_files';
- Shut down the database. If the database is open, shut down the database normally if possible. Use the IMMEDIATE or ABORT clauses only as a last resort.
- Back up all datafiles and redo log files of the database.
- Start up a new instance, but do not mount or open the database: STARTUP NOMOUNT

Steps for creating new control files…

- Create a new control file for the database using the CREATE CONTROLFILE statement.
- Store a backup of the new control file on an offline storage device.
- Edit the CONTROL_FILES initialization parameter for the database to indicate all of the control files now part of your database as
- Recover the database if necessary. If you are not recovering the database, skip to
- Open the database using one of the following methods:
  ALTER DATABASE OPEN;

Backing up Control Files:

- Backing up control files is needed every time you change the physical structure of your database. Such structural changes include:
  - Adding, dropping, or renaming datafiles.
  - Adding or dropping a tablespace, or altering the read/write state of the tablespace.
  - Adding or dropping redo log files or groups.

- Use the ALTER DATABASE BACKUP CONTROLFILE statement to back up your control files. You have two options:
  - Back up the control file to a binary file (duplicate of existing control file) using the following statement:

    - ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/backup/control.bkp';
- Produce SQL statements that can later be used to re-create your control file:

Recovering a control file using a current copy

- Recovering from Control File Corruption Using a Control File Copy.
- Recovering from Permanent Media Failure Using a Control File Copy.

- **Recovering from Control File Corruption Using a Control File copy**
- This procedure assumes that one of the control files specified in the CONTROL_FILES parameter is corrupted, the control file directory is still accessible, and that you have a multiplexed copy of the control file.
  - With the instance shut down, use an operating system command to overwrite the bad control file with a good copy
  - Start SQL*Plus and open the database:
- SQL> STARTUP

Recovering from Permanent Media Failure Using a Control File Copy.

- This procedure assumes that one of the control files specified in the CONTROL_FILES parameter is inaccessible due to a permanent media failure and that you have a multiplexed copy of the control file.
  - With the instance shut down, use an operating system command to copy the current copy of the control file to a new, accessible location:
  - Edit the CONTROL_FILES parameter in the initialization parameter file to replace the bad location with the new location:
  - Start SQL*Plus and open the database:
- SQL> STARTUP

Dropping Control Files

- When we want to drop control files from the database, for example, if the location of a control file is no longer appropriate .Remember that the database should have at least two control files at all times.
  - Shut down the database.
  - Edit the CONTROL_FILES parameter in the database initialization parameter file to delete the old control file name.
  - Restart the database.
- Note: This operation does not physically delete the unwanted control file from the disk. Use operating system commands to delete the unnecessary file after you have dropped the control file from the database.

REDO LOG FILES

- Redo log files, often referred to as simply "redo logs," are a critical component of a database management system's architecture, such as in Oracle Database. They play a pivotal role in ensuring data consistency, durability, and recoverability in a database system. Here's an overview of redo log files:
- Purpose:
  - Record Changes: Redo log files record all changes made to data in the database, including INSERT, UPDATE, DELETE, and DDL (Data Definition Language) statements.
  - Transaction Logs: They act as transaction logs, capturing a chronological sequence of database changes. This information is crucial for database recovery and rollback operations.

REDO LOG FILES

- Components:
  - Redo Log Groups: Redo logs are organized into groups, typically consisting of multiple individual redo log files.
  - Individual Redo Log Files: Each group contains one or more individual redo log files. These files are used in a cyclical fashion, with new changes being written to the current redo log file.
- Cyclical Usage:
  - Redo log files are used in a cyclical manner. When the current redo log file is filled, the system switches to the next available redo log file in the group.
  - This cyclical usage ensures that a complete history of changes is maintained and that older redo log files can be archived or reused.

REDO LOG FILES

- Benefits:
  - Data Recovery: Redo logs are crucial for database recovery in the event of a crash or failure. They contain the information needed to replay changes made since the last checkpoint, ensuring that the database can be restored to a consistent state.
  - Rollback and Rollforward: Redo logs facilitate transaction rollback (undoing uncommitted changes) and transaction rollforward (applying committed changes) during recovery processes.
  - Performance: Redo logs can improve database performance by allowing certain database changes to be asynchronously written to disk. This reduces the need for frequent disk I/O operations, which can be a bottleneck.

REDO LOG FILES

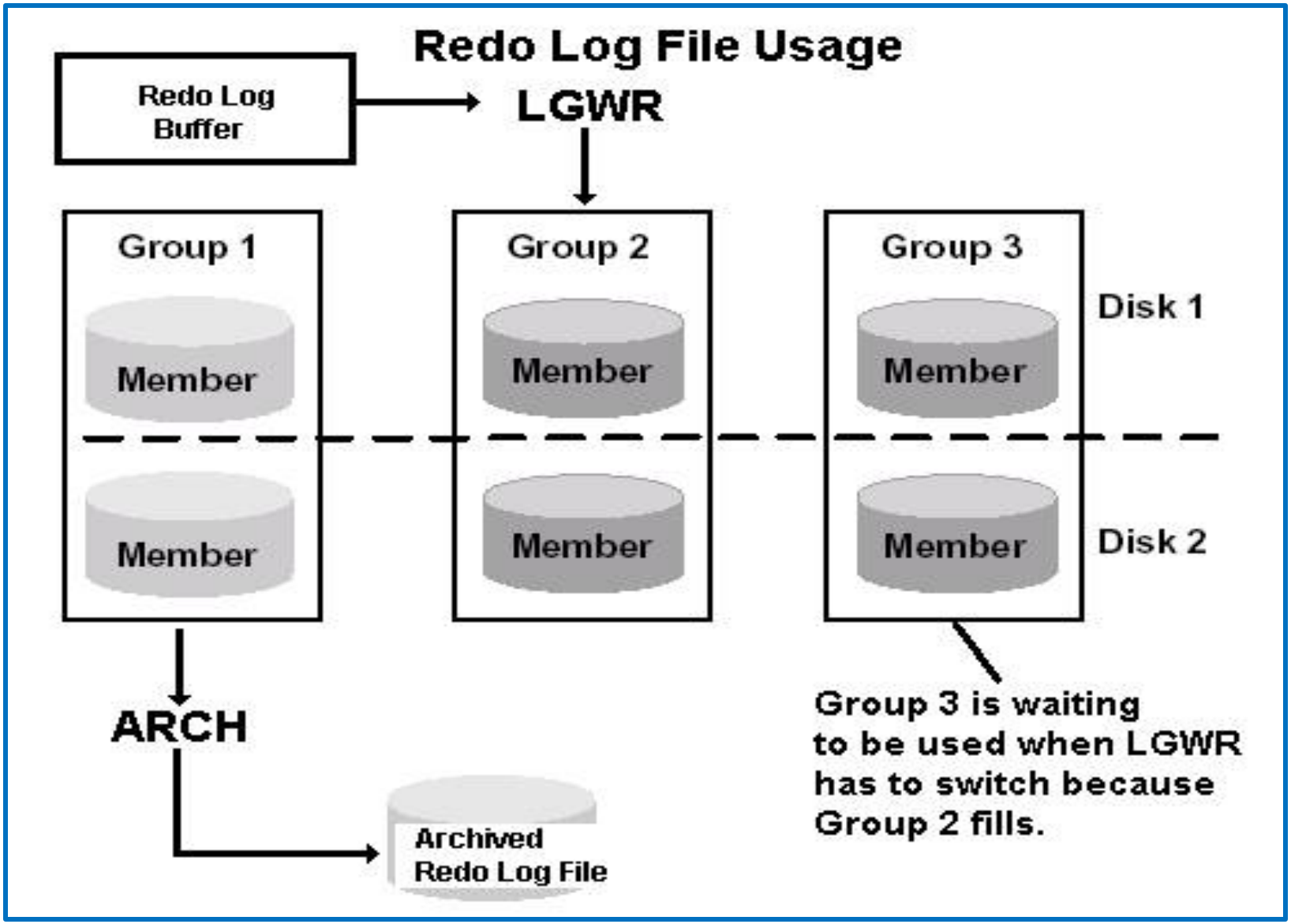
- Types of Redo Log Files:
  - Online Redo Logs: These are active redo log files that the database writes changes to as transactions occur. Online redo logs are essential for the real-time operation of the database.
  - Archived Redo Logs: Archived redo logs are copies of filled online redo log files that are archived to long-term storage. They are used for recovery purposes, especially in the case of media recovery.
- Sizing and Configuration:
  - Database administrators need to properly size and configure redo log files based on the database workload. Sufficient redo log space is essential to avoid issues like "log file switch" and "log buffer space" errors.

REDO LOG FILES

- Management:
  - Database administrators are responsible for managing and monitoring redo logs. This includes ensuring that there are an adequate number of redo log groups and files, monitoring their status, and archiving them as needed.
- Redo log files are a vital component of a database's architecture, providing transaction durability and recovery capabilities. Understanding how to configure and manage redo logs is essential for database administrators to maintain the integrity and availability of a database system.

REDO LOG FILES

- Redo log files are operating system files used by Oracle to maintain logs of all transactions performed against the database.
- The primary purpose of these log files is to allow Oracle to recover changes made to the database in the case of a failure.
- An Oracle database must have at least two redo log files, and most databases have more than two.
- These files are written by the LGWR (**Log Writer**)process in a circular fashion; that is, when the last log file is filled, the first log file is reused.
- For example, if a database has three redo log files, blocks will be written to file1 until it is filled; then that file is closed, and LGWR begins writing to file2 (this is called a *log switch*). When file2 is filled, LGWR switches to file3. When file3 is filled, file1 is reused, and so on.


Redo Log File Usage
Redo Log Buffer
LGWR
Group 1
Group 2
Group 3
Member
Member
Member
Member
Member
Member
Disk 1
Disk 2
ARCH
Archived Redo Log File
Group 3 is waiting to be used when LGWR has to switch because Group 2 fills.

- There are **two types of redo log files:**
  - Online Redo Logs
  - Archived Redo logs
- Oracle Database uses only one redo log files at a time to store redo records written from the redo log buffer. The redo log file that LGWR is actively writing to is called the **current redo log file.**
- Redo log files that are required for instance recovery are called **active redo log files.**
- Redo log files that are no longer required for instance recovery are called **inactive redo log files.**

LOG SWITCHES AND LOG SEQUENCE NUMBER

- A **log switch** is the point at which the database stops writing to one redo log file and begins writing to another.
- Normally, a log switch occurs when the current redo log file is completely filled and writing must continue to the next redo log file. However, you can configure log switches to occur at regular intervals, regardless of whether the current redo log file is completely filled. You can also force log switches manually.
- Oracle Database assigns each redo log file a new **log sequence number** every time a log switch occurs and LGWR begins writing to it. When the database archives redo log files, the archived log retains its log sequence number. A redo log file that is cycled back for use is given the next available log sequence number.
- Each online or archived redo log file is uniquely identified by its log sequence number. During crash, instance, or media recovery, the database properly applies redo log files in ascending order by using the log sequence number of the necessary archived and redo log files.

MAINTAINING AND MONITORING REDO LOG FILES

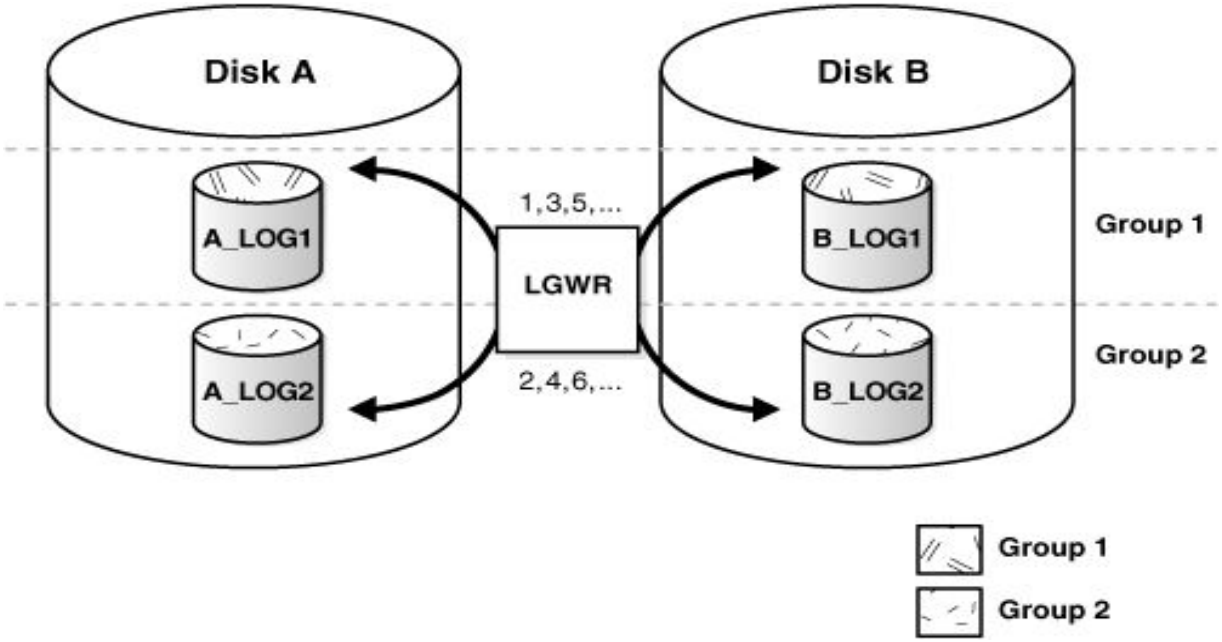
- Multiplexing Redo Log Files.
- Placing Redo Log Members on Different Disks.
- Setting the Size of Redo Log Members.
- Choosing the Number of Redo Log Files.
- Controlling Archive Lag.

Multiplexing Redo Log Files

- Redo log files must be multiplexed in separate location (better in separate disks) in order to protect it against the failure involving the redo log itself. The redundancy of redo log file can help protect against I/O errors, file corruption and so on.
- Multiplexing is implemented by creating group that consists of redo log file and its multiplexed copies where each group is defined by a number, such as group 1, group 2, etc. accessible to LGWR, so the instance can continue to function.

Placing Redo log Members on Different Disks

- When setting up a multiplexed redo log, place members of a group on different physical disks. If a single disk fails, then only one member of a group becomes unavailable to LGWR and other members remain accessible to LGWR, so the instance can continue to function.



Setting size of Redo log members

- Size of Redo log file should be set in such a way that a filled group can be archived to a single unit of storage media with least amount of space on the medium left unused. All members of the same multiplexed redo log group must be of same size. The minimum size permitted for a redo log file is 4 MB.

Choosing number of Redo Log files

- The MAXLOGFILES parameter used in CREATE DATABASE statement determines the maximum number of groups of redo log files for each database. The group value can range from 1 to MAXLOGFILES. If MAXLOGFILES is not specified for the CREATE DATABASE statement, then the database uses an operating system specific default value.

Controlling Archive Lag

- In primary/standby database configuration, changes are made available to standby database by archiving redo logs at primary site and then shipping them to standby database. The changes that are being applied to standby database can lag the changes that are occurring on primary database as the standby database must wait for the changes in primary database redo log to be archived and then shipped to it.
- To limit this lag, you can set ARCHIVE_LAG_TARGET initialization parameter and specify in seconds how long that lag can be.
- eg: ARCHIVE_LAG_TARGET=1800

- **Creating redo log group**
  - To create a new group of redo log files, use the SQL statement ALTER DATABASE with the ADD LOGFILE clause.
  - **ALTER DATABASE ADD LOGFILE ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') SIZE 4M;**
  - You can also specify the number that identifies the group using the GROUP clause:
  - **ALTER DATABASE ADD LOGFILE GROUP 10 ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') SIZE 4M;**
- **Creating Redo log members:**
  - In some cases, it might not be necessary to create a complete group of redo log files. A group could already exist, but not be complete because one or more members of the group were dropped (for example, because of a disk failure). In this case, you can add new members to an existing group.
  - To create new redo log members for an existing group
  - **ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2b.rdo' TO GROUP 2;**

- **RELOCATING AND RENAMING REDO LOG MEMBERS**
  - You can use operating system commands to relocate redo logs, then use the ALTER DATABASE statement to make their new names (locations) known to the database.
  - To rename redo log members, you must have the ALTER DATABASE system privilege.
  - Before relocating your redo logs, or making any other structural changes to the database, completely back up the database in case you experience problems while performing the operation. As a precaution, after renaming or relocating a set of redo log files, immediately back up the database control file.

1. Shut down the database.
   - SQL>SHUTDOWN
2. Copy the redo log files to the new location.
3. Startup the database, mount, but do not open it.
   - SQL> CONNECT / as SYSDBA
   - SQL>STARTUP MOUNT
4. Rename the redo log members.

Use the ALTER DATABASE statement with the RENAME FILE clause to rename the database redo log files.

- ALTER DATABASE RENAME FILE '/diska/logs/log1a.rdo', '/diska/logs/log2a.rdo' TO '/diskc/logs/log1c.rdo', '/diskc/logs/log2c.rdo';

5. Open the database for normal operation. The redo log alterations take effect when the database is opened.

- ALTER DATABASE OPEN;

DROPPING REDO LOG GROUP

- To drop a redo log group, you must have the ALTER DATABASE system privilege.
- Before dropping a redo log group, consider the following restrictions and precautions:
- An instance requires at least two groups of redo log files, regardless of the number of members in the groups. (A group comprises one or more members.)
- You can drop a redo log group only if it is inactive. If you need to drop the current group, first force a log switch to occur.
- Make sure a redo log group is archived (if archiving is enabled) before dropping it.
- Drop a redo log group with the SQL statement ALTER DATABASE with the DROP LOGFILE clause.
- E.g. ALTER DATABASE DROP LOGFILE GROUP 3;

DROPPING REDO LOG MEMBERS

- To drop a redo log member, you must have the ALTER DATABASE system privilege.
- You can drop a redo log member only if it is not part of an active or current group. If you want to drop a member of an active group, first force a log switch to occur.
- To drop specific inactive redo log members, use the ALTER DATABASE statement with the
- DROP LOGFILE MEMBER clause.
- The following statement drops the redo log /oracle/dbs/log3c.rdo: ALTER DATABASE DROP LOGFILE MEMBER '/oracle/dbs/log3c.rdo';

FORCING LOG SWITCHES

- A log switch occurs when LGWR stops writing to one redo log group and starts writing to another. By default, a log switch occurs automatically when the current redo log file group fills.
- You can force a log switch to make the currently active group inactive and available for redo log maintenance operations.
- For example, you want to drop the currently active group, but are not able to do so until the group is inactive.
- To force a log switch, you must have the ALTER SYSTEM privilege. Use the ALTER SYSTEM statement with the SWITCH LOGFILE clause.
- The following statement forces a log switch: ALTER SYSTEM SWITCH LOGFILE database backups at regular, frequent intervals.

MANAGING ARCHIVED REDO LOG FILES

- Oracle Database lets you save filled groups of redo log files to one or more offline destinations, known collectively as the archived redo log.
- The process of turning redo log files into archived redo log files is called archiving. This process is only possible if the database is running in ARCHIVELOG mode.
- You can choose automatic or manual archiving.
- **Use of Archived Redo Log Files**
  - Recover a database.
  - Update a standby database.
  - Get information about the history of a database using the Log-Miner utility.

- **Running Database in NONARCHIVELOG mode**
  - Running database in NOARCHIVELOG mode, disables archiving of the redo log. When a filled group becomes inactive after a log switch, the group is available for reuse by LGWR.
  - NOARCHIVELOG mode protects a database from instance failure but not from media failure.
  - In NOARCHIVELOG mode you cannot perform online tablespace backups, nor can you use online tablespace backups taken earlier while the database was in ARCHIVELOG mode.
  - To restore a database operating in NOARCHIVELOG mode, you can use only whole database backups taken while the database is closed. Therefore, if you decide to operate a database in NOARCHIVELOG mode, take whole database backups at regular, frequent intervals.

Running Database in ARCHIVELOG mode

- Running database in ARCHIVELOG mode, enables the archiving of the redo log. A filled group becomes available for archiving immediately after a redo log switch occurs and can be reused by LGWR once they are finished archiving.
- **Advantages**
  - A database backup, together with online and archived redo log files, guarantees that you can recover all committed transactions in the event of an operating system or disk failure.
  - If you keep an archived log, you can use a backup taken while the database is open and in normal system use.
  - You can keep a standby database current with its original database by continuously applying the original archived redo logs to the standby.
  - You can configure an instance to archive filled redo log files automatically, or you can archive manually. For convenience and efficiency, automatic archiving is usually best.

Steps for switching from NONARCHIVELOG mode to ARCHIVELOG mode

1. Shut down the database instance. SHUTDOWN
2. Back up the database:
   - Before making any major change to a database, always back up the database to protect against any problems. This will be your final backup of the database in NOARCHIVELOG mode and can be used if something goes wrong during the change to ARCHIVELOG mode.
3. Edit the initialization parameter file to include the initialization parameters that specify the destinations for the archived redo log files.
4. Start a new instance and mount, but do not open, the database.
   - STARTUP MOUNT
     - To enable or disable archiving, the database must be mounted but not open.

5. Change the database archiving mode. Then open the database for normal operations.
   - ALTER DATABASE ARCHIVELOG;
   - ALTER DATABASE OPEN;
6. Shut down the database.
   - SHUTDOWN IMMEDIATE
7. Back up the database:
   - Changing the database archiving mode updates the control file. After changing the database archiving mode, you must back up all of your database files and control file. Any previous backup is no longer usable because it was taken in NOARCHIVELOG mode.

Tablespaces

- Oracle divides a database into one or more logical storage units called tablespaces.
- Each tablespace consists of one or more files called datafiles. A datafile physically stores the data objects of the database such as tables and indexes on disk.
- In other words, Oracle **logically** stores data in the tablespaces and **physically** stores data in datafiles associated with the corresponding tablespaces.

Tablespaces

- By using tablespaces, you can perform the following operations:
  - Control the storage size allocated for the database data.
  - Grant specific space quotas to the database users.
  - Control the availability of data by taking tablespaces online or offline.
  - Improve the performance of the database by allocating data storage across devices.
  - Perform partial database backup or recovery.