Oracle 8I Backup and Recovery
Quick Study Guide

CONTEMPORARY TECHNOLOGIES, INC.
The Oracle Specialist

"A DBA's most important responsibility is to keep the database available to users."

1. Protect the database from failures.
2. Increase Mean-Time-Between-Failures (MTBF).
3. Decrease Mean-Time-To-Recover (MTTR).
4. Minimize Data Loss.

Control Files:

- Binary file that is needed to mount a database and contains the following information: Database name and ID, creation date, file and redo locations, tablespace names, log history, backup info (8I – reusable by RMAN), log sequence number, checkpoint information.
- Are read to mount and open the database.
- Should be backed up any time physical changes are made to the database.

Online Redo Logs:

- Used only for recovery of committed data not yet written to the data files.
- Do not need to be backed up, but can be backed up cold to copy database and avoid having to open new database with RESETLOGS.
- A filled redo log cannot be reused until a checkpoint has occurred and the redo log has been written by the ARCn process.
- Multiplex. Processing can continue as long as 1 member from each group is available.
- Log switches occur when…
  - When online redo log fills.
  - When fast start checkpointing is set.
  - At the log_checkpoint_interval.
  - At the log_checkpoint_timeout.
  - At instance shutdown (except ABORT).
  - When forced by an ALTER SYSTEM CHECKPOINT
  - When a tablespace is taken offline.
  - When an online backup is started.

Archived Redo Logs:

- Redo files that have filled and copied to a backup location (Archive_log_dest).
- Are critical for recovery in a hot backup scheme.

Checkpoints:

- Writes all modifies buffers to the datafiles. Datafiles always have committed data.
- Records SCN in the control file and updates all file headers. (Note: except read only tablespaces).

Fast Start Checkpointing (8I): FAST_START_IO_TARGET=n (blocks).

- Reduces time needed for instance crash recovery by ensuring that roll forward will only require n blocks.
- Can also use LOG_CHECKPOINT_INTERVAL and LOG_CHECKPOINT_TIMEOUT to influence.

Fast Start Recovery: DB is available at the end of the roll forward process, individual user processes perform rollback when the blocks needing rolled back are requested.

Database Synchronization: All online datafiles (except read-only tablespaces), redos, and control files must have the same checkpoint number (SCN) for the database to be opened.

Rollback Segments:

- Store location of data and the data as it existed before being modified for read consistency.
- During recovery, used to undo uncommitted changes applied from redo logs to datafiles.
- Are backed up.

Transportable Tablespaces:

1) Make tablespace Read Only.
2) Export the Metadata. $\boxed{\text{EXP TRANSPORT\_TABLESPACE=Y TABLESPACE ts1}}$
3) Copy data file to target system.
4) Copy export dump to target system.
5) Import the Metadata. $\boxed{\text{IMP TRANSPORT\_TABLESPACE=Y DATAFILES (file1, file2)}}$
6) Bring tablespace online and enable original for Read/Write.

Note: Nested tables, varrays, and bitmap indexes are NOT transportable.

V$DATABASE – Lists status and recovery info (db name, ID, creation date, last checkpoint, etc)
V$DATAFILE – Info on file #, name, creation date, status, checkpoint, etc.
V$BACKUP – Which files are ACTIVE in backup mode.
V$DATAFILE_HEADER – Which file are in backup mode (FUZZY=YES).
V$INSTANCE_RECOVERY – Contains size information of redo logs.
V$RECOVER_FILE - use to determine which file(s) need recovery.
V$RECOVERY_FILE_STATUS – files needing recovery and their recovery status.
V$RECOVERY_STATUS – overall database recovery info.

Database Failure Types:

1) Statement failure – failed SQL is automatically rolled back and an error is returned to user.
2) User Process failure – abnormal disconnect – PMON detects and rolls back and releases locks.
3) User Error (drop table, data) – DBA is required to recover data (import or incomplete recovery)
4) Instance Failure – Abnormal shutdown – Instance simply needs restarted, SMON auto recovers by
    a) Rolling forward changes in the redo log not recorded in the datafiles before Open of database.
    b) Rollbacks can occur after the database is open, when block data is requested.
5) Media Failure – Loss or corruption of files – DBA needs to apply appropriate recovery.

Read Only Tablespaces:
ALTER TABLESPACE ts1 READ ONLY;

- Checkpoint occurs when made read only, SCN is written to headers, and does not change.
- Should be backed up (along with control file) immediately following read only.

Datafiles CAN contain uncommitted data – Data is changed and uncommitted and forced off to disk by the DBWR process when more db buffer space is requested by other transactions.

Setting up Archive Logging:

$\boxed{\text{LOG\_ARCHIVE\_START = TRUE}}$ in init.ora, $\boxed{\text{ALTER DATABASE ARCHIVE LOG;}}$

In Oracle8I you can have multiple archive destinations and multiple archive processes.

$\boxed{\text{LOG\_ARCHIVE\_MAX\_PROCESSES = n}}$ Use to define multiple archiver processes.

For multiple archiver destinations,

$\boxed{\text{LOG\_ARCHIVE\_DUPLEX\_DEST = 'dir'}}$ OR

$\boxed{\text{LOG\_ARCHIVE\_DEST\_1 = 'dir'; … LOG\_ARCHIVE\_DEST\_5 = 'dir';}}$

$\boxed{\text{LOG\_ARCHIVE\_SUCCEED\_DEST = n}}$ (number of destinations for successful log switch).

Example:

LOG_ARCHIVE_DEST_1 = "LOCATION=/archive1 MANDATORY REOPEN"
LOG_ARCHIVE_DEST_2 = "LOCATION=/archive2 OPTIONAL"
LOG_ARCHIVE_DEST_3 = "LOCATION=/archive3 OPTIONAL"
LOG_ARCHIVE_DEST_4 = "SERVICE=standby_db1 MANDATORY REOPEN=600"

If LOG_ARCHIVE_SUCCEED_DEST=1 then no effect since there are 2 mandatory dests defined. If = 3, then at least 1 of the optionals must be written before a redo is reusable.

V$DATABASE or $\boxed{\text{svrmgr>ARCHIVE LOG LIST}}$ – for current archiving state.
V$ARCHIVED_LOG – Archive log information from the control file.
V$ARCHIVE_DEST – describes all the arch destinations (5 rows).
V$LOG_HISTORY – Redo log information from the control file (all logs)
V$RECOVERY_LOG – Redo logs needed for recovery.
V$ARCHIVE_PROCESSES – information on the archival processes (ARC0,1…)

(RMAN)

V$BACKUP_DATAFILE – info from controlfile about backed up files.
V$BACKUP_DEVICE – list of supported 3rd party devices.
V$BACKUP_REDOLOG – info on archive logs in backup sets.
V$BACKUP_SET – info about backup sets.
V$BACKUP_PIECE – info about backup pieces within backup sets.
V$BACKUP_CORRUPTION – info about corrupt blocks in datafiles
V$SESSION_LONGOPS – get status of RMAN operation (recovery or backup).

THE BACKUP

<u>Backup Considerations:</u>

- Archive when…
  1. No data loss is acceptable.
  2. Database is needed online 24 by 7.
  3. Point in time recovery may be needed (roll db forward back to a prior state).
- When structural changes are made…
  1. Archiving ➔ After changes, ALTER DATABASE BACKUP CONTROLFILE TO ‘file’;
  2. No Archive ➔ Backup entire database before and after the change is made.
- Read Only tablespaces only need backed up after they are set to read only or are manually changed.

<u>Physical backups:</u>

Types: Closed = database level, OPEN = tablespace or datafile level.

Get file information from…

V$Tablespace and **V$datafile** (or **DBA_DATA_FILES**).
V$Controlfile, V$Logfile

COLD BACKUPS:

1. Shutdown NORMAL or IMMEDIATE or TRANSACTIONAL (abort is inconsistent).
2. All datafiles, controlfiles, online redo logs (optional) and the init.ora parameter file.
3. Open the database

HOT BACKUPS:

1. ALTER TABLESPACE ts1 BEGIN BACKUP
2. Back up the tablespace datafiles.
3. ALTER TABLESPACE ts1 END BACKUP …
4. Force a log switch – ALTER SYSTEM SWITCH LOGFILE;
5. Backup controlfile – ALTER DATABASE BACKUP CONTROLFILE TO ‘file’ REUSE;

<u>Logical Backups:</u>

Can be accomplished through exports.

<u>EXPORT / IMPORT (LOGICAL):</u>

<u>Export</u> – makes a logical copy of object definitions and data to a binary file (Logical backup).
<u>Conventional</u> (DIRECT=N) uses SQL select to extract data into buffer cache, then into evaluation buffer
<u>Direct Path</u> (DIRECT=Y) Data is read from disk into buffer cache – data in blocks is not reorganized.
COMPRESS=Y – initial extent resized as total current segment size
FEEDBACK=x – A dot is displayed in log file for every x records inserted.
FULL=Y or OWNER=user or TABLES=schema.table
CONSISTENT=Y – entire export as 1 read only transaction.
RECORD=Y – update the information in the SYS tables (INCxxx).
* INCTYPE=
COMPLETE – Export all objects and data, reinitialize dictionary views.
INCREMENTAL – All tables that have changed since last incremental, complete, or cumulative.
CUMULATIVE – All tables that have changed since last complete or cumulative.
<u>Dictionary Tables</u>:
SYS.INCFIL – Tracks all exports in current cycle.
SYS.INCEXP – Tracks which objects are in which export files.
SYS.INCVID – Tracks the last export information.

- Clusters are not exported with the Table mode, they are in User or Full mode.
- SYS owner objects (data dictionary) are never exported.

<u>Import</u> – Reads export files and copies object definitions and data into an Oracle database.
IGNORE=Y – Overlook object creation errors – causes rows to be imported into existing tables.
INDEXFILE=file – Index creation commands written to file (no actual import).
Order of operations: Type defs | Data | B-Tree indexes | Views, procedures, constraints | bitmap indexes.
Before import: Disable Ref Integrity before imports and increase buffer size for better performance
After import: Compile all invalid objects, reenable RI.

<u>STANDBY DATABASES:</u>

- Use standby databases to minimize recovery time.
- Archives from the primary database are applied to the standby.
  - Adding datafiles will propagate to standby (logged activity).
  - Renaming datafiles will not propagate to the standby and must be done manually.
  - Unrecoverable operations will not propagate to the standby.

To create a standby database…
1. ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'file';
2. ALTER SYSTEM ARCHIVE LOG CURRENT; archive the current redo logs.
3. Copy the controlfile, archive logs, and backed up datafiles to standby location.
4. Start the standby in NOMOUNT mode.
5. ALTER DATABASE MOUNT EXCLUSIVE STANDBY DATABASE;

To keep standby current…
1. Copy the archive logs as they are created to the Standby location.
2. RECOVER FROM 'archive location' STANDBY DATABASE;

To activate the standby database:
1. ALTER DATABASE ACTIVATE STANDBY DATABASE – on the current primary.
2. SHUTDOWN the current primary database.

Startup the standby database (resetlogs).

DATABASE RECOVERY SCENARIOS:

FROM COLD BACKUP…

1. Shutdown the database.
2. Recover all datafiles, controlfiles (online redos) from tape.
3. STARTUP MOUNT
4. ALTER DATABASE RENAME FILE if necessary (new file locations).
5. ALTER DATABASE OPEN RESETLOGS (if online redos are not restored);

FROM HOT BACKUP…

Complete / Initially Open / Closed (Mounted) – (System or RBS lost, or majority of datafiles).

1) Close the database if not already closed.
2) Restore the necessary files from backup.
3) STARTUP MOUNT;
4) RECOVER DATABASE or RECOVER DATAFILE 'file';
5) ALTER DATABASE OPEN;

Complete / Initially Open / Open – (File corruption or media failure, database is still open and available)

1) Take tablespaces and/or datafiles to be recovered offline if not already offline.
2) Restore necessary datafiles.
3) RECOVER DATAFILE 'file' or RECOVER TABLESPACE tsname;
4) Bring the tablespace or datafile online
5) ALTER DATABASE DATAFILE 'file' ONLINE or ALTER TABLESPACE tsname ONLINE;

Complete / Initially Closed / **Open** – (media or hardware failure has brought the db down).

1) Mount the database.
2) If files are online, ALTER DATABASE DATAFILE 'file' OFFLINE;
3) Open the database, ALTER DATABASE OPEN;
4) Restore the necessary datafiles.
5) RECOVER DATAFILE 'file' or RECOVER TABLESPACE tsname;
6) Bring the tablespace or datafile online
   ALTER DATABASE DATAFILE 'file' ONLINE or ALTER TABLESPACE tsname ONLINE;

Complete / Initially Open / Open / Loss of file with no backup (must have all archives from point of original creation).

1) Take the file offline if not already.
2) Recreate the file: ALTER DATABASE CREATE DATAFILE 'file1'
3) Apply archive logs to this file – RECOVER TABLESPACE ts1
4) Bring tablespace online – ALTER TABLESPACE ts1 ONLINE;

Incomplete: (To recover from user error or if all archives are not available or loss of all control files).

1. Shutdown and backup entire database.
2. Restore ALL datafiles – all datafiles must be restored to a prior point in time. Do not restore control file, redo logs, password files, or parameter files.
3. Mount the database.
4. RECOVER DATABASE
   UNTIL CANCEL | UNTIL TIME 'YYYY-MM-DD:HH24:MI:SS' | UNTIL CHANGE scn#
      (Query V$LOG_HISTORY to get the SCN number)
   USING BACKUP CONTROLFILE; –
   - If all controlfiles are lost and cannot be recreated.
   - If current controlfile does not match structure of db at point you are recovering to.
   - In a disaster recovery situation (no current control files are available).
5. ALTER DATABASE OPEN RESETLOGS;
6. Backup the database.

To restore to a different location, Mount db, ALTER DATABASE RENAME DATAFILE, Open db.

To enable automatic recovery:
- SET AUTORECOVERY ON
- Enter AUTO when prompted for an archive log
- RECOVER AUTOMATIC …

Parallel Recovery:
- Have several server manager sessions recovering separate tablespaces simultaneously.
- Set init.ora parameter RECOVERY_PARALLELISM (use no more than 2 processes per disk)
- RECOVER DATABASE PARALLEL DEGREE 5 INSTANCES 2 (5 processes on 2 instances=10).

Distributed Recovery:
- If complete recovery is done, no further action is required.
- If an incomplete recovery is done on a database in a distributed environment, a coordinated time or change based recovery should be performed on all dependent databases to ensure global read consistency.
    1. Use time based recovery on the failed database.
    2. Open RESETLOGS.
    3. Check alert file for the message "RESETLOGS after incomplete recovery UNTIL CHANGE scn.
    4. Perform incomplete recoveries on dependent databases to the new SCN.

Clearing a corrupted redo log file (when dropping is not possible and database must be kept open)**:**
- ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP 1;
- Perform a complete backup after this operation.

***** Recover from failure after RESETLOGS (before new backup)**
1) SHUTDOWN IMMEDIATE;
2) Copy current control files to a save location.
3) Restore files and control files from a time before resetlogs (last good backup).
4) STARTUP MOUNT;
5) Get SCN applied at last recovery from the alert log.
6) RECOVER DATABASE UNTIL CHANGE scn USING BACKUP CONTROLFILE;
7) SHUTDOWN NORMAL;
8) Copy current save control files back to original location.
9) STARTUP MOUNT;
10) RECOVER DATABASE (applies all archives from resetlogs to current time);
11) ALTER DATABASE OPEN;

Tablespace Point in Time Recovery:
1) Clone the database
2) Restore backup on clone and perform incomplete recovery to time desired.
3) Export tablespace tables from clone, Import tablespace tables (or use transportable tablespace).

Troubleshooting:

DB_BLOCK_CHECKSUM = TRUE

- A checksum for every block is computed and stored in the block header.
- Verified when the block is read.
- Every log block is also given a checksum before it is written to the redo log.
- Heavy performance overhead.

DB_BLOCK_CHECKING = TRUE

- Check data and index blocks whenever they are changed.
- Minimal performance impact (does not check blocks while reading).

LOG_BLOCK_CHECKSUM = TRUE

- If set to TRUE, Oracle will check log file blocks for corruption at archive time (each log switch).
- Oracle will substitute good blocks from another member for the corrupted blocks.
- If all members are corrupt, archiving will not occur (must be cleared with the ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP n command and back up the database).

DBVERIFY [FILE START END BLOCKSIZE LOGFILE FEEDBACK HELP PARFILE]

- OS utility to verify the integrity of an online or backed up datafile.
- Blocksize default is 2k and must be specified if block size is not 2k.
- Example: $ dbv file=system01.dbf blocksize=8192 logfile=system01.log feedback=100

DBMS_REPAIR

- PL/SQL package to detect and mark block corruptions.
- Created by the **dbmsrpr.sql** and **prctrpr.sql** scripts.

LOGMINER

- Use to track changes to database to pinpoint time or SCN to be used for imcomplete recovery.
- Useful in determining logical corruption times.
- Can be used for granular logical recovery by undoing specific changes.

RECOVERY MANAGER:

- Increased Performance:
  - No extra redo during online backup.
  - Automatic parallelization.
  - Incremental block.
- Compress unused blocks.
- Detect corrupted blocks.
- Verify validity of backups.
- Store backup & recovery scripts in the db.

Recovery Metadata

- **Controlfile**: Stored in the control file of the target database.
  - Use CONTROLFILE_RECORD_KEEP_TIME to limit the # of days to keep this information. [0=no control file expansion]
  - Oracle will create copies of controlfile minus the recovery data (snapshots) when RMAN needs a read consistent view.
- **Recovery Catalog**:
  - A Catalog can be created in a separate database to keep more info (less in the control file)
  - A catalog is also necessary for storing scripts and performing autorecoveries.
  - Catalog can be used to recreate control files.
  - Separate catalog should be kept for each target database (using different schemas).

```
CREATE USER rman_db1 … DEFAULT TABLESPACE rts1 QUOTA UNLIMITED ON rts1;
GRANT RECOVERY_CATALOG_OWNER TO rman_db1;
GRANT CONNECT, RESOURCE TO rman_db1;
$ rman catalog rman_db1/rman_db1@catdb msglog=catalog.log
RMAN> CREATE CATALOG TABLESPACE rts1;
RMAN> REGISTER DATABASE;
```

Connecting:

`$rman target system/manager{ @db1 } nocatalog  @backupall.rcv  log backup1.log` (batch cmdfile)
`$rman target system/manager{ @db1 } catalog rman/rman@catdb` to specify a catalog (interactive).

Media Management Layer (MML): A compatible MML is needed for RMAN to write to tape.

Channel:

- To perform and record backup operations, RMAN requires a link to the target database.
- Can allocate more than 1 channel to enable parallel operations.

`ALLOCATE CHANNEL d1 TYPE DISK | NAME "dev0/tape1";` (type can be tape if MML installed)

Standalone Commands:

CHANGE, CONNECT, CREATE CATALOG, RESYNC CATALOG,
CREATE SCRIPT, DELETE SCRIPT, REPLACE SCRIPT, STARTUP, SHUTDOWN, REPORT.

- Do not need a channel
- Are executed within a RUN block.

RESYNC CATALOG – use to synchronize catalog with changes in control file (done at every backup or copy automatically.
CATALOG – use to record changes done outside of RMAN (ie. OS backup).
    Ex) CATALOG DATAFILECOPY '/disk1/sys01.dbf' TAG 'systs0922';
CHANGE – mark as unavailable, delete, or validate a piece, image copy or archived log.
    Ex) CHANGE ARCHIVELOG '/a/arch1' UNCATALOG   #use UNAVAILABLE if temporary.
    Ex) CHANGE DATAFILECOPY '/disk1/system.bak' CROSSCHECK;
    Ex) CHANGE DATAFILECOPY '/disk1/system.bak' DELETE   #physically delete it.
    Ex) DELETE EXPIRED BACKUP   #removes all expired records in catalog.

<u>REPORT command:</u>
Produces a detailed analysis of the recovery catalog.
- RMAN> REPORT **SCHEMA** – What is the database structure?
- RMAN> REPORT **NEED BACKUP** – Which files need backed up?
  INCREMENTAL: $\boxed{\text{RMAN> REPORT NEED BACKUP INCREMENTAL 3 DATABASE}}$ – Which files need more than 3 incrementals for recovery?
  DAYS: $\boxed{\text{RMAN> REPORT NEED BACKUP DAYS 3 TABLESPACE SYSTEM}}$ – Which files for the system tablespace have not been backed up in 3 or more days?
  REDUNDANCY: $\boxed{\text{RMAN> REPORT NEED BACKUP REDUNDANCY 3}}$ – Which files do not have 3 or more backups?
- RMAN> REPORT **OBSOLETE** REDUNDANCY – Which backups can be deleted?
- RMAN> REPORT **UNRECOVERABLE** DATABASE db1– Which files are not recoverable?

<u>LIST command:</u>
Produces a detailed report listing all information for backup sets, datafile copies, and incarnations.
- RMAN> LIST **COPY OF TABLESPACE** “SYSTEM”;
- RMAN> LIST **BACKUPSET OF DATAFILE** ‘/disk1/file1’;
- RMAN> LIST **INCARNATION OF DATABASE** db1;
- RMAN> LIST **BACKUP OF TABLESPACE** tbs1 **COMPLETED AFTER** ‘May 1 1999 00:00:00’;
- RMAN> LIST **BACKUP OF DATABASE DEVICE TYPE** ‘sbt_tape’;

<u>SCRIPTS:</u>
Example:
```
RMAN>Create script Level0backup (
        Allocate channel d1 type disk;
        Backup Incremental level 0
        format ‘/u01/db01/backup/%d_%s_%p’
        fileperset 5
        (database include current controlfile);
        sql ‘alter database archive log current’;
        release channel d1;)
RMAN> run {execute script Level0backup;}
```

<u>Run command:</u>

• operating system command.	$\boxed{\text{RMAN> run \{ host “ls –l” \}}}$
• SQL command.	$\boxed{\text{RMAN> run \{ sql “alter system switch logfile” \}}}$
• Run script.	$\boxed{\text{RMAN> run \{ execute script nightlybackup;\}}}$

<u>Incarnation:</u>
- A number used to identify the version of a database prior to a log sequence being reset to 0.
- A new incarnation is created when a database is recovered to a point in time or opened with resetlogs.
- After RESETLOGS, the catalog cannot be opened until $\boxed{\textbf{RMAN> RESET DATABASE;}}$ is issued.

other views…
RC_DATABASE – info = dbkey, dbname, incarnation, change#, last resetlogs time.
RC_TABLESPACE – which tablespaces are stored in which backupset.
RC_DATAFILE – which datafiles are stored in which backupset.
RC_STORED_SCRIPT – which scripts exist.
RC_STORED_SCRIPT_LINE – script code lines.

<u>BACKUP options (scope)</u>

• **Whole backup:**	Backup ALL datafiles and control file. Target can be opened or closed.
• **Full backup:**	Backup of 1 or more files, backs up ALL BLOCKS. (not incremental).
• **Incremental backup:**	Dataile blocks that have changed since last incremental. Can be cumulative, must have a level 0 created as a base.
• **Image copy:**	Physical copy of a datafile on disk.

```
BACKUP
[ FULL | INCREMENTAL LEVEL n ]        (0=Full, 1=monthly, 2=weekly, 3=daily,…8)
FILESPERSET n                         How many files to multiplex into backup sets
MAXCORRUPT n                          Max corrupt blocks before process fails.
SKIP [ offline | readonly | inaccessible ]
        (DATABASE | TABLESPACE | DATAFILE | ARCHIVELOG | CURRENT CONTROLFILE)
```

Example:

```
        RMAN> ALLOCATE CHANNEL C1 TYPE DISK;
        RMAN> ALLOCATE CHANNEL C2 TYPE DISK;
        RMAN> ALLOCATE CHANNEL C3 TYPE DISK;
        RMAN> BACKUP INCREMENTAL LEVEL 0
                        FORMAT '/disk1/backup/dfile%S%t.bak'
                                (DATAFILE 1,4,5  CHANNEL C1  TAG copy1)
                                (DATAFILE 2,3,9  CHANNEL C2  TAG copy2)
                                (DATAFILE 6,7,8  CHANNEL C3  TAG copy3);
        RMAN> SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
```

<u>COPY (Image Copies)</u>
Can be used in place of restoring files from tape (image copy exists on disk)

```
RMAN> COPY [ DATAFILE 'f1' | CURRENT CONTROLFILE | ARCHIVELOG ] TO 'f2' {LEVEL n}
```

<u>RECOVER</u>
RMAN uses the recovery catalog or control file to decide which full, incremental, image copies and archived logs to use.

```
RECOVER [ DATABASE | TABLESPACE | DATAFILE ]     { UNTIL [ CANCEL | CHANGE | TIME ]
```

Example:

```
ALLOCATE CHANNEL d1 TYPE NAME 'sbt_tape';
        RESTORE DATABASE;                   #restores all necessary files from tape
        RECOVER DATABASE;
SQL 'ALTER DATABASE OPEN';
```

To relocate a file (target is mounted)…

```
SET NEWNAME FOR   DATAFILE 1 TO '/disk2/sys01.dbf',
                  DATAFILE 2 TO '/disk2/usr01.dbf; …       #specifies where to restore file
RESTORE DATABASE;
SWITCH DATAFILE ALL;               #updates the catalog
RECOVER DATABASE;
```