**Disaster recovery (DR) script** and logical restore.

This document has 3 use cases.

**CASE 1** : Master Database and other database **dumps are available.**

**CASE 2** : If the **master database dump is missing** but **valid database backups exist for user databases.**

**CASE 3**: Recover the **dropped table in one user database** without affecting other databases.

**CASE 1 : Master Database and other database dumps are available**

* Ensure **regular backups** of:
* **Full database dumps** (dump database)
* **Transaction logs** (dump transaction)
* **Master database** and other system databases (dump database master)
* Backup:
* **RUN\_servername** file (the ASE startup file)
* **Configuration file** ($SYBASE/ASE-\*/bin/\*.cfg)
* **Interfaces file** (listening endpoints, $SYBASE/interfaces)
* **Backup of Sybase license files** ($SYBASE/SYSAM-\*/licenses)
* **Any external scripts (cron jobs, monitoring scripts)**
* **DR Script:**

#!/bin/bash

# Disaster Recovery Script for Sybase ASE

# Author: [Your Name]

# Date: [Date]

### 1. Environment Setup

export SYBASE=/opt/sybase

export SYBASE\_ASE=ASE-16\_0

export SYBASE\_OCS=OCS-16\_0

export PATH=$SYBASE/$SYBASE\_ASE/bin:$PATH

export LD\_LIBRARY\_PATH=$SYBASE/$SYBASE\_ASE/lib:$LD\_LIBRARY\_PATH

export ASE\_SERVER=MY\_SERVER\_NAME

### 2. Start ASE server in minimal mode (if needed)

echo "Starting ASE server in single-user mode...( -m option is for single user mode)"

startserver -f $SYBASE/$SYBASE\_ASE/install/RUN\_$ASE\_SERVER

### 3. Restore the master database

echo "Restoring master database..."

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

shutdown with nowait

go

exit

EOF

# Assuming you have master database dump at /backup/master.dmp

# Dataserver command will rebuild the master database from master.dmp

dataserver -d/backup/master.dmp -i$SYBASE/$SYBASE\_ASE/install/RUN\_$ASE\_SERVER

### 4. Restart server normally

echo "Restarting ASE server normally..."

startserver -f $SYBASE/$SYBASE\_ASE/install/RUN\_$ASE\_SERVER

### 5. Restore system databases if necessary

echo "Restoring model, sybsystemdb, sybsystemprocs..."

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load database model from "/backup/model.dmp"

go

load database sybsystemprocs from "/backup/sybsystemprocs.dmp"

go

load database sybsystemdb from "/backup/sybsystemdb.dmp"

go

online database model

go

online database sybsystemprocs

go

online database sybsystemdb

go

exit

EOF

### 6. Restore user databases

echo "Restoring user databases..."

for db in $(cat /backup/db\_list.txt); do

echo "Restoring database: $db"

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load database $db from "/backup/$db.dmp"

go

online database $db

go

exit

EOF

done

### 7. Apply transaction logs (optional if logs exist)

# For each database, load transaction dumps if any

### 8. Validate Databases

echo "Running dbcc checks..."

for db in $(cat /backup/db\_list.txt); do

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

use $db

go

dbcc checkdb

go

exit

EOF

done

### 9. Final Configurations

# Rebuild any external devices, update configurations if needed.

echo "Disaster Recovery complete for ASE server: $ASE\_SERVER."

**For full database recovery**, meaning:

* Load **full dumps**,
* Load **transaction logs**,
* Bring database **online**.

**For *Point-in-Time Recovery*** (PITR), **you'll need to modify** the script a little bit, because:

* **Load a full dump**,
* Then **sequentially load transaction log dumps**,
* **Stop** *right before* or *at* a specific point (for example, before a user error or corruption),
* **Use** load transaction with stop\_at (or until\_time).

**Key Differences for Point-In-Time Recovery**

| **Full Recovery** | **Point-in-Time Recovery** |
| --- | --- |
| Load full dump, optionally logs | Load full dump + transaction logs carefully |
| Bring database online after all loads | Stop at a specific timestamp inside transaction logs |
| Simple script | Needs control: stop after reaching recovery point |

**In the script for Point-in-Time recovery, you must:**

After loading the full database dump:

load database mydb from "/backup/mydb\_full.dmp"

go

Then load each transaction log **until** the point-in-time you want:

load transaction mydb from "/backup/mydb\_log1.trn"

go

load transaction mydb from "/backup/mydb\_log2.trn"

with stop\_at = "Apr 29, 2025 10:34:00AM"

go

After that Sybase will **truncate** at that time, and **bring the database online** automatically.

**Modified Recovery Script Section (for PITR)**

# Load full database dump

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load database $db from "/backup/$db\_full.dmp"

go

EOF

# Load transaction logs

for logdump in /backup/$db\_log\*.trn; do

echo "Loading transaction log: $logdump"

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load transaction $db from "$logdump"

go

EOF

done

# Final transaction log with STOP\_AT

isql -Usa -P<password> -S$ASE\_SERVER <<EOF

load transaction $db from "/backup/$db\_log\_final.trn"

with stop\_at = "Apr 29, 2025 10:34:00AM"

go

EOF

# No need to ONLINE database manually if stop\_at is used.

**CASE 2 :** If the **master database dump is missing** but **valid database backups exist for user databases.**

**Challenge**: Rebuilding system configuration and metadata if master is lost

### Steps to Recover Sybase ASE Without a Master Database Dump:

#### 1. Rebuild the Master Database (Using dataserver -f)

Rebuild the master database using the -f option.

$ $SYBASE/$SYBASE\_ASE/bin/dataserver -d<device> -s<server\_name> -f

* -f tells ASE to initialize with a new master database.
* Recreate server configuration (devices, databases, logins, etc.).

#### 2. Start the Server with the New Master

Once master is rebuilt, ASE will start with default settings. It won’t know about any user databases yet.

#### 3. Recreate Devices (If Needed)

Use disk init to recreate any user database devices, with the **exact same physical file paths and sizes** as before.

disk init

name = 'user\_device',

physname = '/sybase/data/user\_device.dat',

size = '500M'

#### 4. Load Database Dumps

Once devices are available, you can now use load database to restore the user databases.

load database <dbname>

from '/path/to/backup\_file.dmp'

If the backup was made with headeronly, Sybase knows where to place the data. Otherwise, ensure device layout matches original.

Then run:

online database <dbname>

#### 5. Recreate Logins and Users

Since the master database is fresh, all logins and their SIDs (security IDs) are lost. You'll need to manually recreate them:

sp\_addlogin 'username', 'password'

Then remap users in each database:

use <dbname>

sp\_change\_users\_login 'update\_one', 'username', 'username'

#### 6. Restore Other System Settings (Optional But Important)

You may also need to:

* Recreate replication setup if used
* Reconfigure sp\_configure settings
* Recreate scheduled jobs or alerts
* Apply license keys if custom ones were used

### Backup Master Database: Going forward, regularly back up the master database:

dump database master to '/path/to/master\_backup.dmp'

**CASE 3**: Recover the **dropped table in one user database** without affecting other databases.

1. **Ensure all databases have recent full backups and transaction logs.**
2. **Restore only the affected database** (say, db1) to a **new database** (e.g., db1\_recover) using:  
     
   load database db1\_recover from '/path/db1\_full\_backup.dmp'

load transaction db1\_recover from '/path/db1\_log.trn'

until\_time = 'May 09 2025 16:59:59'

1. **Extract the dropped table** from db1\_recover:  
    Use bcp or select into to export/import the table data.
2. **Restore the table into the original db1**, which is still online and has all transactions from other unaffected tables.
3. Other databases continue to operate normally — no downtime or rollback needed.

This approach uses **point-in-time recovery on a clone** to recover the dropped table while keeping all other data current.