::ISUG

techcast series

Archive Database Access in ASE

October 16, 2007

SYBASE

::ISUG

techcast series

Archive Database Access in ASE

Mike Harrold,
Executive Director
International Sybase Users Group

SYBASE

::ISUG

techcast series

Archive Database Access in ASE

Graham Ivey,
Senior Staff Engineer
Sybase, Inc.

SYBASE

Motivation – Offline DBCC

- **The ability to run DBCC checks directly on a database dump using some (offline) utility:**
  - Does not impact production
  - Is not constrained by a small maintenance window
  - Does not require additional disk space

Motivation – Object Level Restore

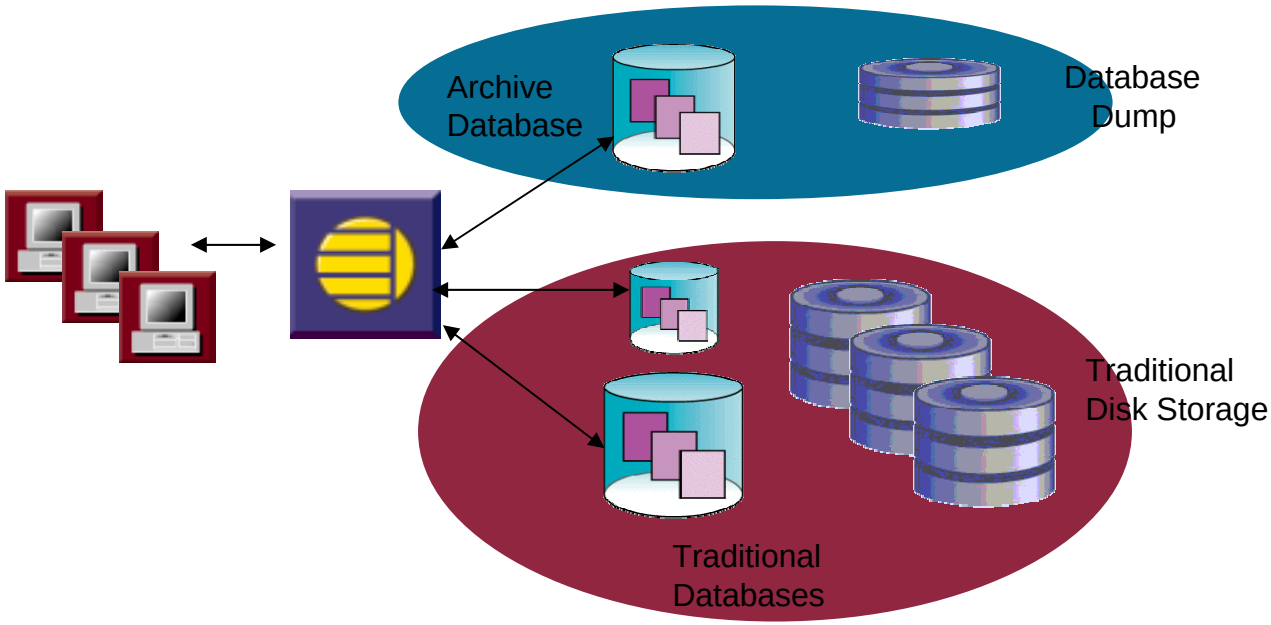
- **The ability to recover a single object (like a table) from a database dump**
  - An unqualified delete or update is accidentally done on a table
  - Data corruption has occurred within a table (never!!)
- **Reloading an entire database dump impacts production because it is time consuming**
  - Particularly if only a small table needs to be restored

Motivation – Load Sequence Verification

- **The ability to know that:**
  - The database dump can be loaded
  - The transaction logs can be successfully applied to the database
- **This will guarantee that the entire load sequence will work if there is a catastrophic data loss**

General Solution

- **Archive Database Access (ADA)**
- **This provides the means of constructing (or virtualising) a database on top of a database dump**
- **The virtualised database is called an archive database**



General Solution (cont)

- **An archive database looks and behaves like a traditional *read-only* database**
- **The *majority* of the database pages are stored within a database dump that has been taken earlier with DUMP DATABASE**
- **Similar to a standby database except**
  - Made available *much* quicker (no data copying by LOAD DATABASE)
  - Consumes little traditional disk storage

Solution for Offline DBCC

- **An archive database looks like any read-only database**
- **DBCC checks may be run against it**
  - The database can be made available on any ASE that has access to the dump file and thus need not affect production in any way
  - The maintenance window can be as big as is needed
- **Caveat**
  - DBCC checks against an archive database are not truly offline
    - **There is no small, standalone utility that validates the contents of the dump**
    - **The utility is ASE itself!**

Solution for Object Level Restore

- **Use SELECT INTO from the archive database into the target database:**

```
select * into <production_db>..<table>
     from <archive_db>..<table>
```

- **Use CIS and SELECT INTO or bcp if the archive database is available on a server other than the target server**
- **Caveat**
  - This is not equivalent to a LOAD TABLE command
  - Index entries are not restored automatically. They are restored by virtue of inserting restored rows into the target table

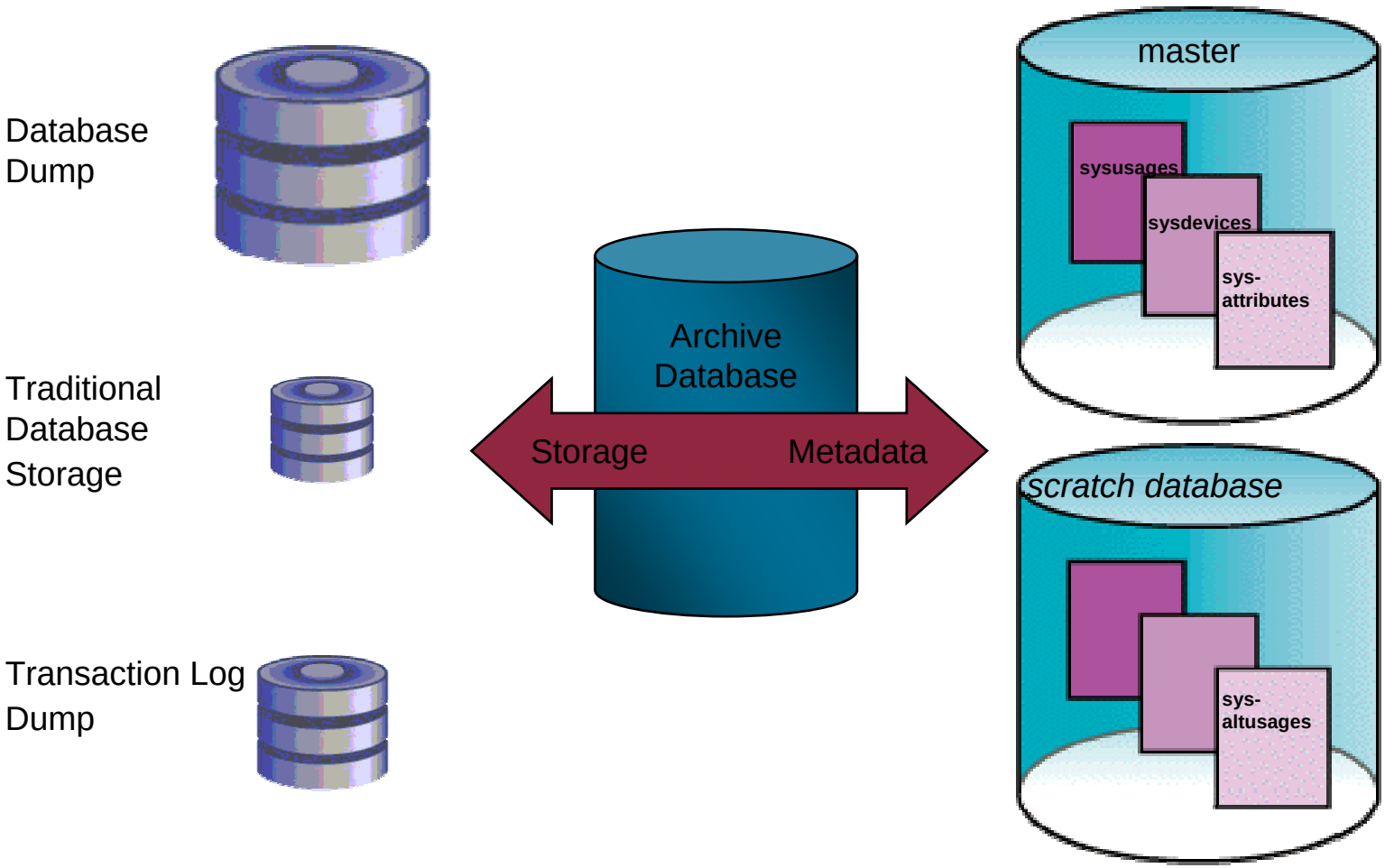
Solution to Load Sequence Verification

- Like a standby database, transaction logs can be applied to an archive database
- Load sequence verification can therefore be done
- The ability to apply transaction logs allows for both DBCC checks and object-level restore to be done using an up-to-date version of the database

Components of an Archive Database

- **An archive database consists of:**
  - A database dump (the archive)
  - A traditional database (known as a *scratch database*) that hosts a new system table called `sysaltusages`
  - Rows in the `sysusages`, `sysdevices` and `sysattributes` tables in the `master` database
  - Some traditional disk storage used to store any modified or newly-allocated pages.This is called the *modified pages section*
  - Optionally, a transaction log dump

Components of an Archive Database



The Database Dump

- **The database dump:**
  - Only stores a subset of pages of the dumped database
  - Is used as a repository for unmodified pages
  - Read-only. Changes cannot be made to the database dump
- **ASE sees the database dump and its stripes as database devices (in `sysdevices`) that are useable by the archive database only**

The Modified Pages Section

- **An archive database is read-only**
- **However, a certain amount of write activity is needed for permissible operations**
  - Database recovery in order to make the archive database consistent
  - DBCC commands that perform fixes so that coherent versions of tables can be restored
- **Modified and newly-allocated pages must be stored somewhere other than the database dump**
- **Every archive database is allocated an amount of traditional database storage for this. This is called the *modified pages section***

The Modified Pages Section (cont)

- **The modified pages section is reflected as rows in the sysusages table in much the same way that space allocated to a traditional database is**
- **The modified pages section:**
  - Is allocated when the archive database is created
  - Can be extended when necessary, to accommodate more modified and newly-allocated pages
- **Together, the database dump and the modified pages section provide the storage for the archive database**

Sysaltusages and the Scratch Database

- **`sysaltusages` is a new data-only locked system table (datarows)**
  - Like `sysusages` for a traditional database, it is used to perform logical-to-virtual page mapping for the archive database
  - Because of the highly fragmented nature of an archive database, it has many more rows than `sysusages`
  - As a result, it is not stored in the `master` database
  - Instead it is stored in a *scratch database*
- **The scratch database is a traditional database that simply hosts `sysaltusages`**

Sysaltusages and the Scratch Database (cont)

- **A database can be designated as a scratch database using:**

```
sp_dboption <dbname>,“scratch database”,“true”
```

- **It is preferable that the scratch database is dedicated to the purpose of hosting `sysaltusages`:**
  - `sysaltusages` is quite volatile
  - Truncate log on checkpoint can be set on the scratch database
  - The scratch database can be dropped and re-created if needed
- **Threshold procedures can be used to manage space within the scratch database**

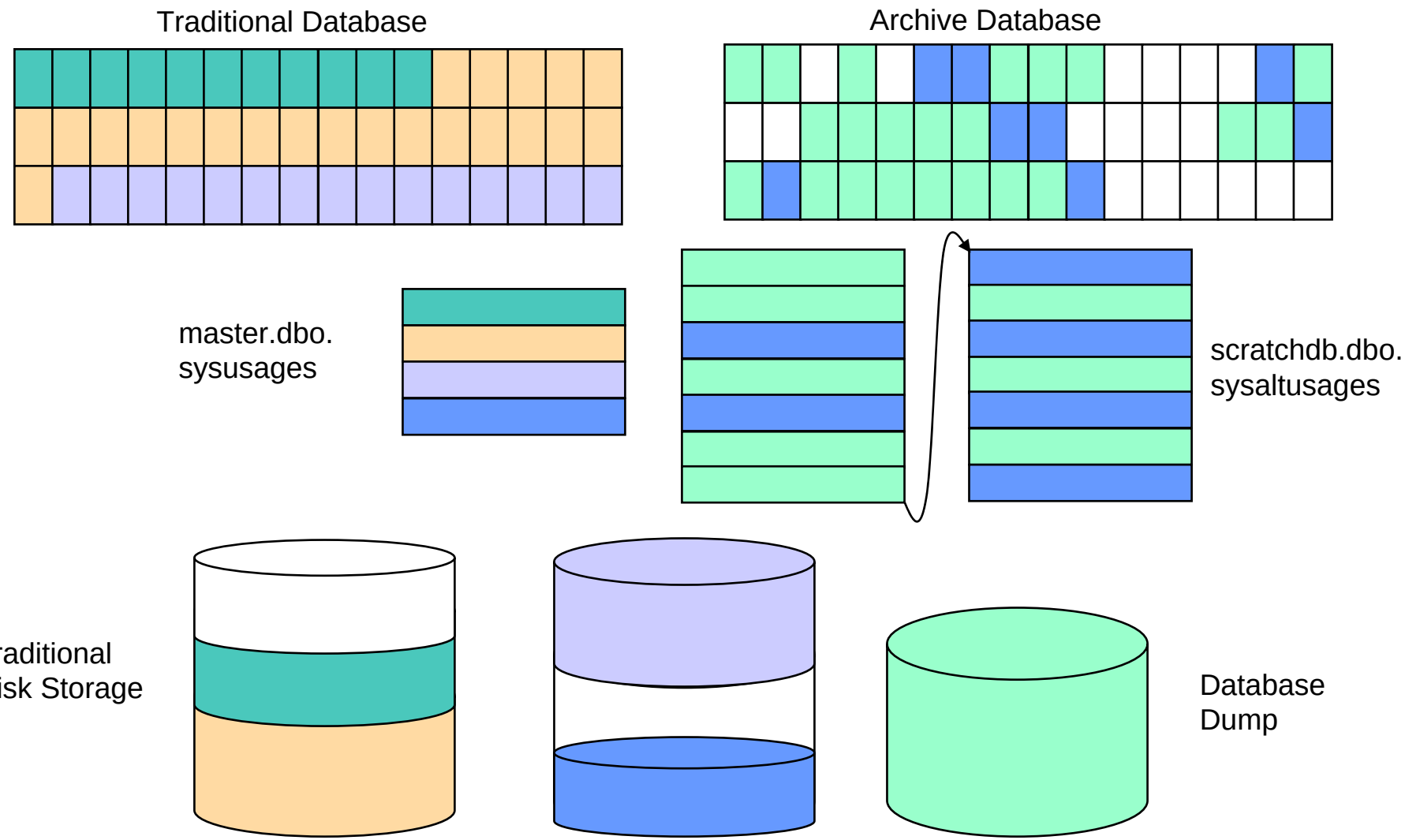
Sysaltusages and the Scratch Database (cont)

- Each archive database can only be assigned a single scratch database
- Multiple archive databases can use the same scratch database
- If you have a large number of archive databases, you may define multiple scratch databases

Sysaltusages Schema

```
Column_name     Type             Length      Nulls
--------------- ---------------- ----------- ----
dbid            smallint                   2     0
location        int                        4     0
lstart          int                        4     0
size            int                        4     0
vstart          int                        4     0
vdevno          int                        4     0
segmap          int                        4     0
index_name           index_description
-------------------- ----------------------------------
csysaltusages        clustered, unique located on system
index keys
----------
dbid, location, lstart
```

Sysusages and Sysaltusages



Creating an Archive Database

- **Create and designate a scratch database:**

```
create database scratchdb on default=20
sp_dboption “scratchdb”,
      “scratch database”,“true”
```

- **Create the archive database with the CREATE ARCHIVE DATABASE command**

```
create archive database <dbname>
      on <device> = <size> [, <device> = <size>]
      with scratch_database = <scratch_dbname>
```

- **Example, an archive database with a 100 MB modified pages section**

```
create archive database archivedb
      on datadev3 = 100
      with scratch_database = scratchdb
```

Populating an Archive Database

- An archive database is populated using LOAD DATABASE
- The syntax is similar to that of loading a traditional LOAD DATABASE:

```
load database <dbname>
        from <dump_device1>
        [stripe on <dump_device2> …]
        [with norecovery]
```

- For example:

```
load database archivedb from “/dumps/060415.dmp”
```

Populating an Archive Database (cont)

- **The stripes of the database dump are inserted into `sysdevices` as if they are database devices**
  - They are marked with a special status to indicate that they are used by an archive database
  - The devices are given automatically generated logical devices names `SYSDEV$__<device number>`

Populating an Archive Database (cont)

- **The data and log pages are not actually loaded. Backup Server is not used. Instead, ASE:**
  - Scans the dump building up a map of physically contiguous fragments of pages and their offsets within the dump
  - Creates the relevant rows in the `sysaltusages` table that reflect the logical-to-virtual page mapping for that contiguous fragment
  - Because the stripe device is seen by ASE as a device within `sysdevices`, the virtual page number of the fragment is like any other virtual page number in `sysusages` (in 12.5.x, an 8 bit device number plus a 24 bit offset, in 15.0.x, a 32 bit device number plus a 32 bit offset)

Loading an Archive Database (Without Recovery)

- **The WITH NORECOVERY option of the LOAD DATABASE command ensures that recovery is not run**
- ***Not* supported for traditional database loads**
- **Allows a database dump to be made visible:**
  - In the quickest possible time (no recovery)
  - Using the least amount of space within the modified pages section (since very few pages are modified)
- **The database is brought online automatically**
- **Because recovery has not been run, the database may be both logically and transactionally inconsistent**
- **Accessible only by a user with sa_role**

Loading an Archive Database (With Recovery)

- More normally, WITH NORECOVERY is not specified
- Recovery (redo pass) is run at the end of LOAD DATABASE
- This will result in modified and newly-allocated pages
- When a page is first modified (or newly-allocated) it is automatically remapped to the modified pages section

Bringing the Archive Database Online

- **ONLINE DATABASE brings the archive database online after the load**

```
online database <dbname>
```

- **Recovery is run (undo pass) as is the case for (normal) ONLINE DATABASE)**
- **This will result in modified and newly-allocated pages**
- **When a page is first modified (or newly-allocated) it is automatically remapped to the modified pages section**

Loading a Transaction Log

- **The syntax for LOAD TRAN is the same as that for a traditional database:**

```
load tran <dbname>
        from <dump_device1>
        [stripe on <dump_device2> …]
```

- **For example:**

```
load tran archivedb from "/dumps/060415_01.dmp"
```

- **Like LOAD DATABASE, the stripes of a transaction log dump are inserted into `sysdevices`**
- **Unused devices from the previous LOAD TRAN are removed**

Loading a Transaction Log (cont)

- **The recovery redo pass is run on the database. Any newly-allocated pages and pages modified for the first time are remapped to the modified pages section**
- **Unlike LOAD TRAN for a traditional database, LOAD TRAN for an archive database can follow an ONLINE DATABASE command**
  - This is to allow the archive database to be useable throughout the load sequence for running DBCC checks and performing object-level restore

Dropping the Archive Database

- **An archive database can be dropped using the DROP DATABASE command:**

```
drop database <dbname>
```

- **Rows are removed from:**
  - master..sysusages
  - master..sysattributes
  - master..sysdevices
  - master..sysdatabases
  - <scratch database>..sysaltusages

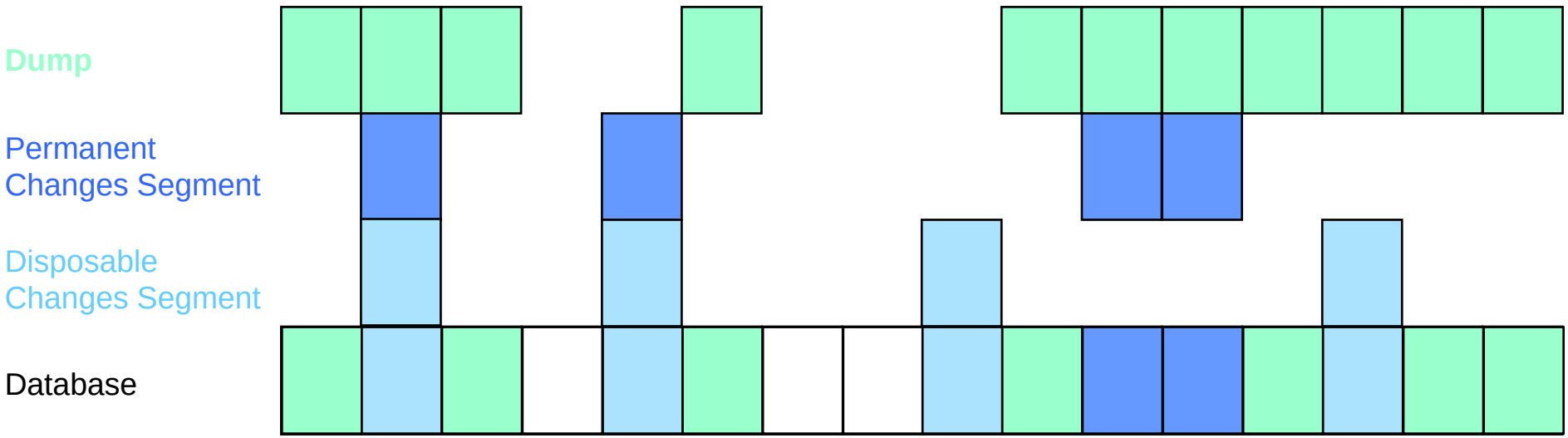
Management of the Modified Pages Section

- **A page is remapped to the modified pages section when it is first modified or newly-allocated**
- **The modified pages section is logically divided into two areas (or segments):**
  - The permanent changes segment
  - The disposable changes segment
- **In *general*:**
  - Any page modified by LOAD DATABASE or LOAD TRAN is mapped to the permanent changes segment
  - Any page modified outside of LOAD DATABASE or LOAD TRAN is mapped to the disposable changes segment

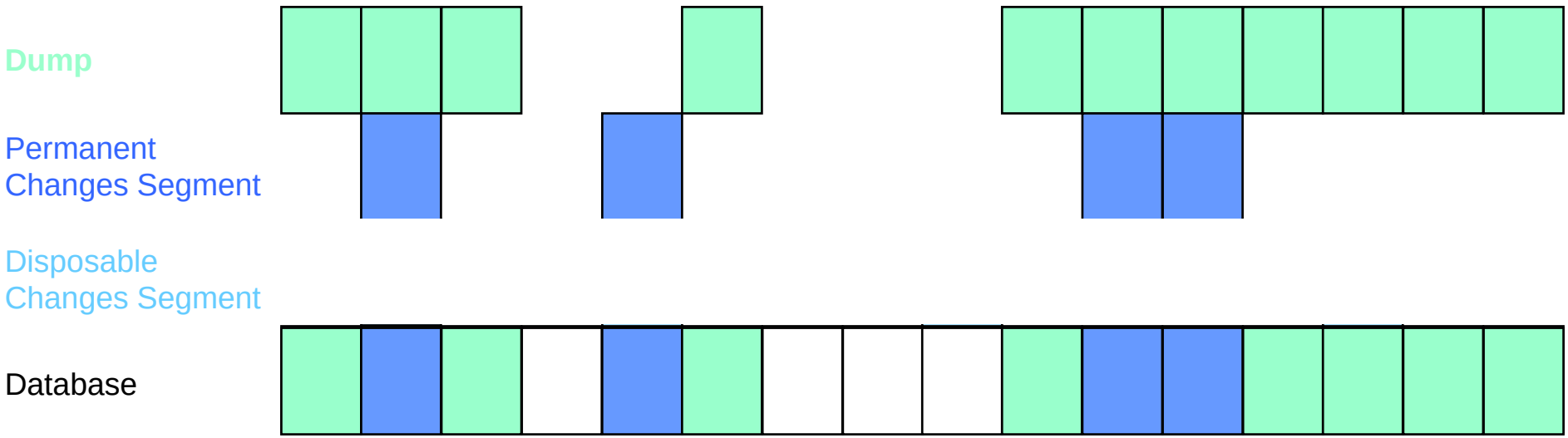
Management of the Modified Pages Section (cont)

- **A maximum of three versions of a page can exist**
  - Page 100 is in the dump
  - Page 100 (the version above) is modified by LOAD DATABASE recovery and the new image is stored in the permanent changes segment of the modified pages section
  - Page 100 (the version above) is modified by ONLINE DATABASE recovery and the new image is stored in the disposable changes segment of the modified pages section
- **LOAD TRAN is able to follow ONLINE DATABASE by disposing of all the pages stored in the disposable changes segment**

Management of the Modified Pages Section (cont)



Management of the Modified Pages Section (cont)



Management of the Modified Pages Section (cont)

- **Allocation of space to the permanent and disposable changes is maintained automatically**
  - Disposable changes segment steals 256 pages at a time from the permanent changes segment
- **Described by rows in `sysusages`**
- **Any change results in an update of `sysusages` at the end of the command**
- **Results in some logging in the `master` database**

Management of the Modified Pages Section (cont)

- **At the start of every LOAD DATABASE and LOAD TRAN, all the space in the modified pages section is reallocated to the permanent changes segment**
- **Use the 15715 trace flag to override this**
  - Can reduce sysusages updates at the expense of wasting some space

Management of the Modified Pages Section (cont)

- **Difficult to size the modified pages section**
  - Depends on the number of (unique) pages modified and newly-allocated by LOAD DATABASE/LOAD TRAN recovery
  - Depends on the number of (unique) pages modified and newly-allocated outside of LOAD DATABASE/LOAD TRAN, even if that page has already been modified by LOAD DATABASE/LOAD TRAN
- **Experience will help in sizing the modified pages section**
  - Start small and the modified pages section can be extended where necessary
  - Dropping and re-loading the database to reduce the size is not nearly as onerous a task as for a traditional database

Extending the Modified Pages Section

- The ALTER DATABASE command can be used to increase the size of the modified pages section:

```
alter database <dbname>
        on <device> = <size> [, <device> = <size>]
```

- The ON clause specifies the size and location of the expansion using traditional database storage
- Can be executed either when the archive database is idle or when it is suspended as a result of running out of space while executing the current command
- In this respect it is much like the suspension that happens when the last chance threshold is hit

DBCC Commands

- **DBCC checks can be run against an archive database in the same way that it is run against any other database**
- **DBCC commands prevent other users from accessing an archive database while they are executing**
  - If you attempt to access an archive database while a DBCC command is being executed, you receive a message saying that the database is in single user mode

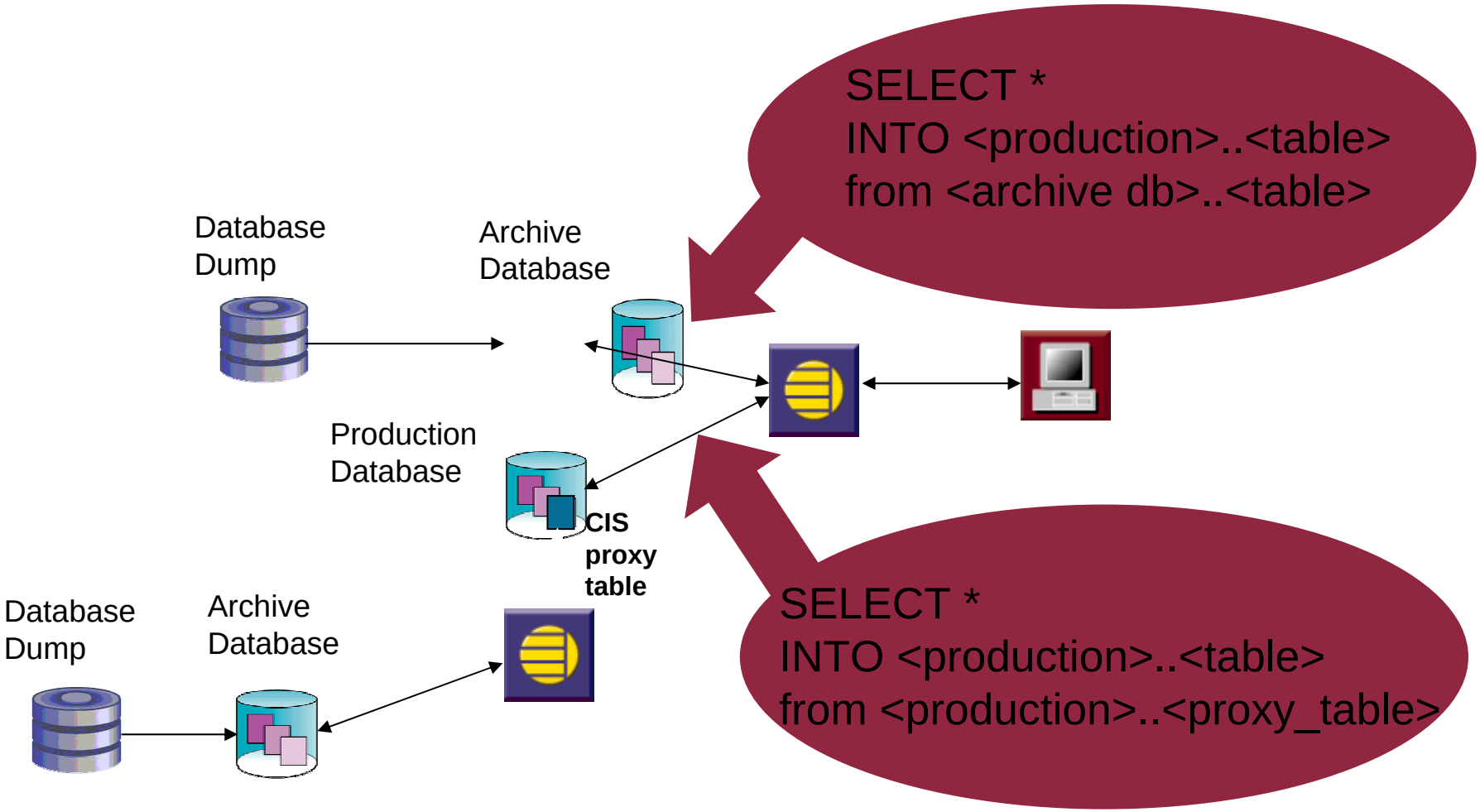
DBCC Commands (cont)

- **The DBCC commands below are allowed in an archive database:**
  - `checkstorage`
  - `checkdb`
  - `checkcatalog` (the `fix` option is not supported)
  - `checktable`
  - `checkindex`
  - `checkalloc`
  - `indexalloc`
  - `tablealloc`
  - `textalloc`
- **The FIX option requires the database to be online**

DBCC Commands (cont)

- **DBCC CHECKALLOC consumes a lot of space in the modified pages section**
  - Every allocation page (1 in 256) has information written to it
  - Modified pages are stored in the modified pages section
  - Therefore, the modified pages section must be at least 1/256th the size of the original database
  - True even if you run it with the NOFIX option

Object Level Restore



Compressed Dumps

- **Only dumps taken using:**
  `DUMP DATABASE… WITH COMPRESSION`
  `DUMP TRAN … WITH COMPRESSION`
  can be loaded into an archive database
- **Dumps taken using the `compress::<n>::<stripe>` syntax are not supported**
  - These dumps can be used by first decompressing the stripes using `gunzip`
- **To load and use a compressed dump a special memory pool must be configured**

Compressed Dumps (cont)

- **When ASE reads a page from a compressed dump, it takes a compressed block from the dump, decompresses it, and extracts the required page. The decompression is done using large buffers from a special memory pool**
- **Each concurrent task uses a maximum of two buffers, one to read the compressed data and another to place the result of the decompression**
  - By default, each buffer is 64 KB
  - Therefore, each concurrent task needs 64 (2K) pages
- **The size of the pool is configured using:**

  `sp_configure "compression memory size", <size>`

  <size> is given in 2 KB pages

Compressed Dumps (cont)

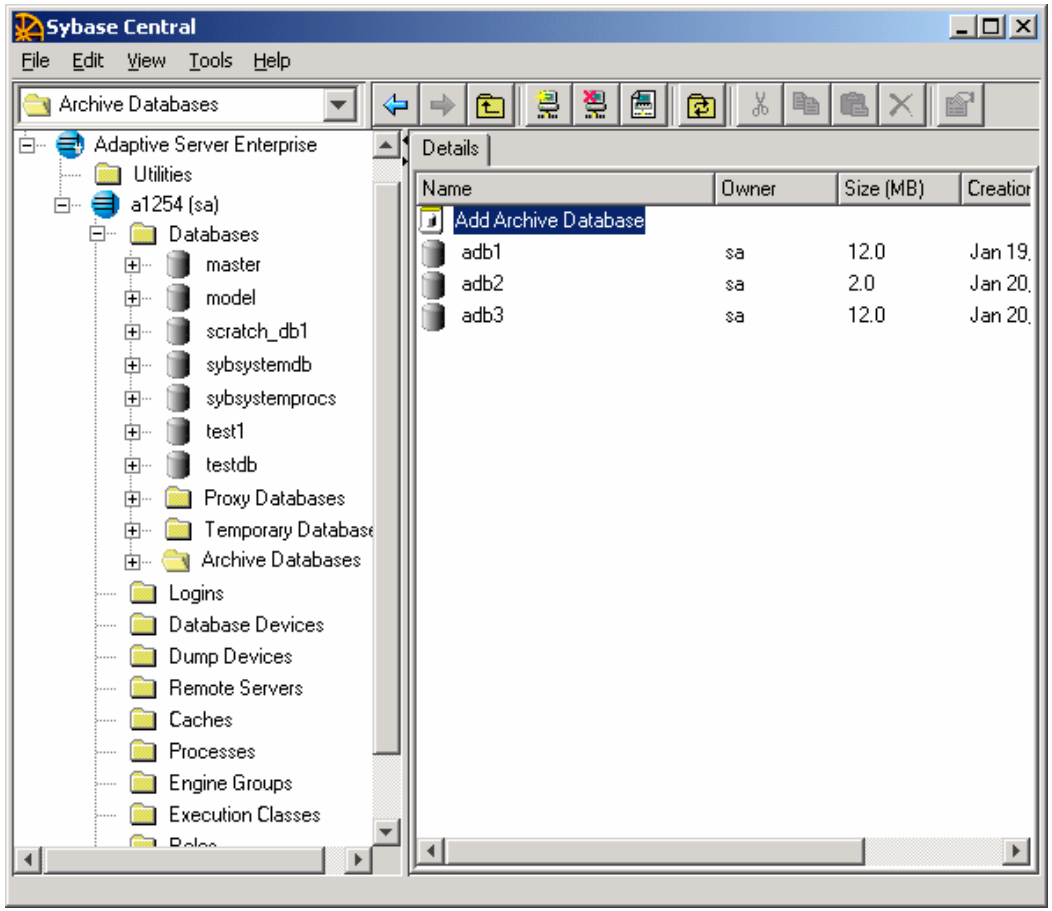
- **The compression format was changed to allow compressed dumps to be loaded**
- **The 12.5.4 Backup Server is the component that writes the new compression format. Therefore:**
  - Use a 12.5.4 Backup Server to load compressed dumps on a 12.5.3 (or earlier) installation if the dumps were created by a 12.5.4 Backup Server
  - Use a 12.5.4 Backup Server to create dumps on a 12.5.3 (or earlier) installation if the dumps are to be loaded into an archive database
- **The 12.5.4 Backup Server understands both 12.5.4 and earlier compression formats**

Compressed Dumps (cont)

- **In 15.x, the 15.0.2 Backup Server is the component that writes the new compression format. Therefore:**
  - Use a 15.0.2 Backup Server to load compressed dumps on a 15.0.1 (or earlier) installation if the dumps were created by a 15.0.2 Backup Server
  - Use a 15.0.2 Backup Server to create dumps on a 15.0.1 (or earlier) installation if the dumps are to be loaded into an archive database
- **The 15.0.2 Backup Server understands both 15.0.2 and earlier compression formats**

Sybase Central

- **Archive databases are supported in Sybase Central**



Availability

- **12.5.4 and 15.0.2**
  - `create archive database`
  - `load database` (with and without recovery)
  - `online database`
  - `alter database`
  - `drop database`
  - `select`, etc
  - dbcc commands (except `checkstorage` and `checkverify`)
  - Compressed dumps
- **12.5.4 ESD 1 and 15.0.2**
  - `dbcc checkstorage` and `checkverify`
- **12.5.4 ESD 3 and 15.0.2**
  - load tran

Performance

- **The “load” portion of LOAD DATABASE and LOAD TRAN is significantly faster for an archive database than a traditional database**
- **There is a small penalty (less than 15%) in accessing an uncompressed archive database**
- **The penalty for accessing a compressed archive database is *much* larger**

Performance (cont)

- **Case 1 (columns 2 and 3 in table)**
  - 7 GB database (~4 GB data and ~3 GB log)
  - Table being selected from had 11 million rows (700 MB)
- **Case 2 (columns 4 and 5 in table)**
  - 13 GB database (~11 GB data and ~2 GB log)
  - Table being selected from had 3 million rows (400 MB)

	Traditional	Archive	Traditional	Archive
load database	13 m	<<1 m	5 m	<<1 m
dbcc checkdb	34 m	33 m	211 m	179 m
dbcc checkalloc	16 m	13 m	50 m	49 m
dbcc checkstorage	11 m	20 m	43 m	40 m
select	5 m	5 m	88 s	99 s
select into	4 m	3 m	279 s	313 s
bcp (out)	14 s	13 s	174 s	199 s

Archive Database Caveats and Limitations

- **Logging of `sysusages` updates in the master database**
- **Large amount of logging of `sysaltusages` updates in the scratch database**
  - Use a database exclusively for the scratch database if possible
  - Use truncate log on checkpoint
- **Increase the "number of devices" parameter to account for dump stripe devices**
- **Archive database is automatically put in single user mode when any command is run that changes the archive database – like DBCC commands**

Archive Database Caveats and Limitations (cont)

- **Changes made by DBCC commands are not permanent**
  - Mapped to the disposable changes segment
  - Lost on next LOAD TRAN
- **Read-only**
- **No use of large buffer pools**
- **Default data cache only**
- **Maximum size of any stripe must be at most 32 GB (12.5.x only)**
- **No upgrade done. Across minor versions this is not generally an issue**

Archive Database Caveats and Limitations (cont)

- No free space thresholds
- Cannot be replicated
- No fail over in a High Availability system
- No DISK REFIT
- No cross-architecture load
- Tape dumps are not supported
- Remote dumps (using the AT syntax) are not supported
- Cannot specify the recovery order with `sp_dbrecovery_order`

Possible Future Additions

- **Allow an archive database that has been used to load an entire load sequence, to be *dumped***
  - No (or little) data copy involved in the dump phase
    - **Dump consists of the original database dump and the modified pages section**
  - The consolidated dump can be loaded into a traditional database
  - No need for a lengthy LOAD TRAN sequence
- **Allow UPDATE STATISTICS to be run**
  - Use *optdiag* to export statistics from archive database into production database