STORAGE FILE MAPPING IN DATABASES

D. Patel
April 22, 2003

VERITAS ARCHITECT NETWORK

TABLE OF CONTENTS

Introduction

Storage Area Networks (SAN) are blurring the relationship between a DBMS and the physical storage infrastructure. Datafiles, created on a file system or raw device, no longer map directly to the physical storage environment. Instead, these files have become abstract views into the complex hierarchy of a storage infrastructure. This makes it difficult to tie a logical database entity directly to a physical file on disk.

Providing an abstracted view of the physical world to applications has enormous benefits for storage administrators. The fluid nature of views allows the administrator tremendous flexibility in managing storage with minimal impact on business applications and databases.

For the database administrator (DBA), however, this layer of abstraction presents a real problem. New storage technologies place a barrier between logical database entities and physical files. This presents significant problems when a DBA evaluates the I/O performance of database entities.

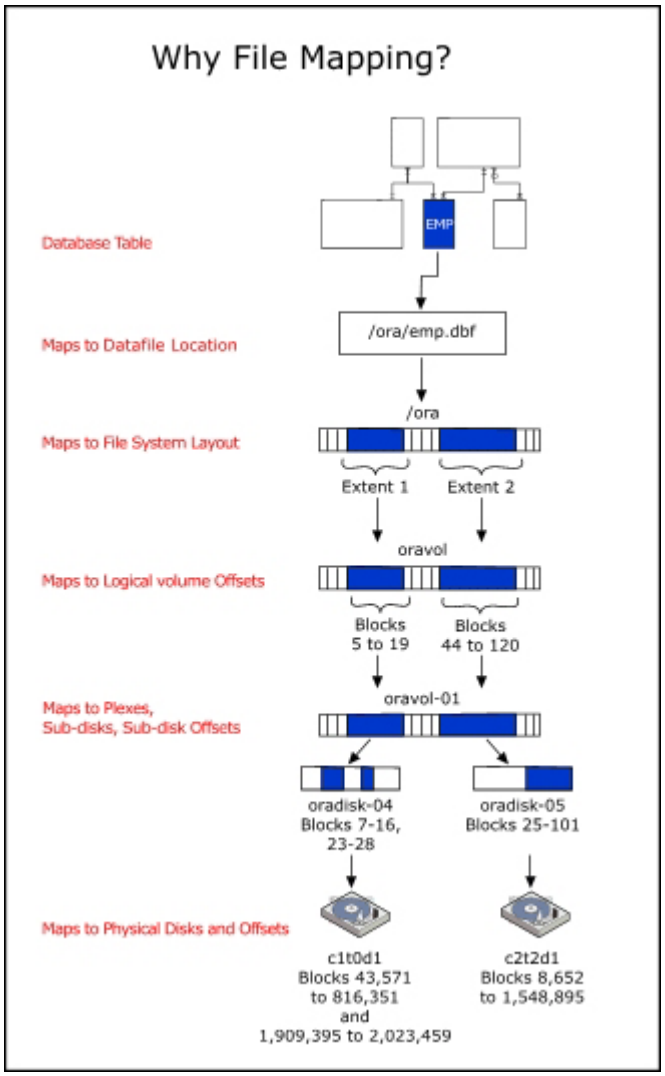

Figure 1: Why File Mapping?

The Evolution of Storage Complexity

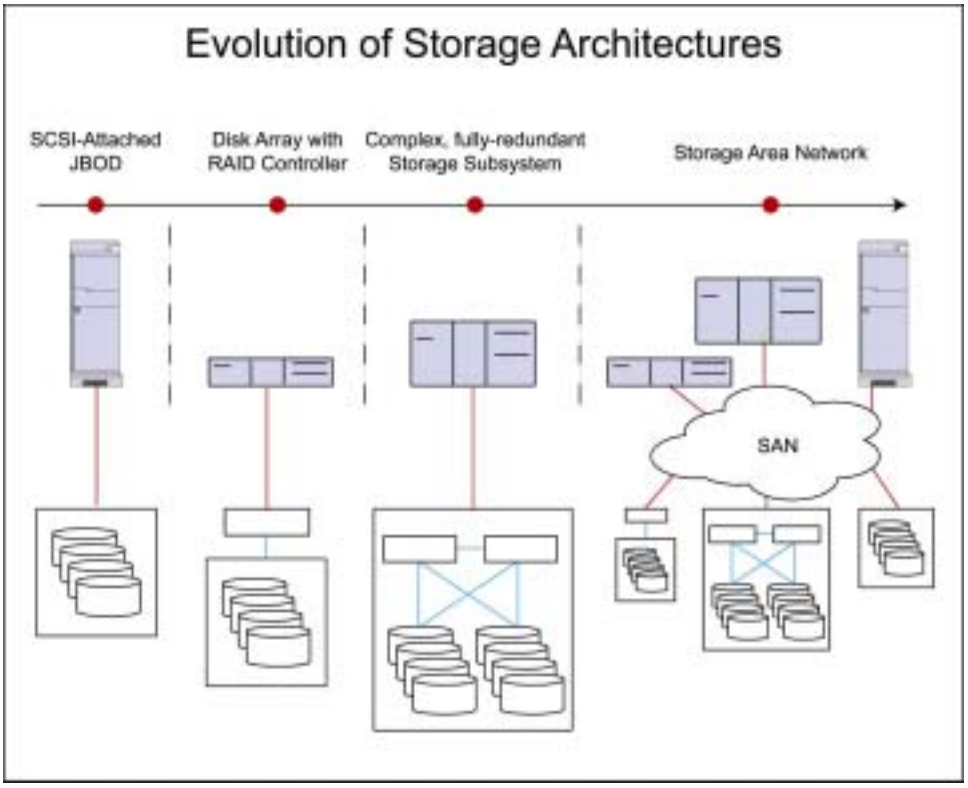

Figure 2: Evolution of Storage Architectures

The problem of mapping logical database entities to physical storage components did not suddenly arise with the deployment of SAN architectures. Host-based logical volume management software and sophisticated storage subsystems providing RAID features have always presented challenges for the DBA. However, the relative simplicity of these configurations, as compared to a modern SAN, make the task of searching for file to device mapping information a doable, if arduous, task. The heterogeneous combination of storage and networking components deployed in a SAN has significantly upped the ante.

The I/O stack for an enterprise database table now includes a combination of files, file systems, system extents, volumes, plexes, subdisks, subdisk offsets, LUNs, and physical disks. Introducing storage virtualization technology to this list adds even more complexity.

Each evolutionary turn for storage has delivered innumerable benefits to storage users and administrators, including better throughput, higher-availability, and greater capacity. But for each technological advance a veil has been lowered between the application and the physical device. And while this may not be important for many software applications, for database systems it is crucial.

The DBA's Dilemma

If every advance in technology has a cost, then the dramatic increase in storage infrastructure complexity has left the DBA holding the check. Although application databases benefit enormously from advances in speed, availability, and size offered by the latest storage technologies, the downside for the DBA managing these databases is considerable.

Understanding how a logical database structure relates to the physical world of storage is a fundamental prerequisite for a DBA. Designing high-performance databases, debugging I/O related performance problems, and communicating effectively with storage administrators about the needs of a database requires a basic understanding of the storage infrastructure. Even common-sense data placement tasks, like ensuring index files are not placed on the same disk as the data files that they hold pointers to, cannot be effectively performed.

Recognizing the difficulty new storage technologies pose for the DBA, DBMS vendors developed the file mapping interface. The file mapping interface gives storage vendors access to APIs enabling them to communicate information about the I/O stack hierarchy to the DBMS. Storage vendors provide mapping libraries that describe their area of responsibility in the I/O stack, and the DBMS queries these libraries to build a true picture of the physical structure of a database. Once built, a DBA need only query the file mapping information, using simple SQL statements, to determine exactly which physical devices house logical database data.

The Oracle File Mapping Interface

When a DBA creates database files on a local file system or raw device it is very easy to determine the mapping between physical disk and database elements. In fact, Oracle provides mapping views, such as `DBA_TABLESPACES`, `DBA_DATA_FILES`, and `V$DATAFILE` that give details of the file to device mapping. Querying these views provides the DBA with the physical device mapping that can be matched to device statistics to create a complete I/O picture of a database.

Once a layer of abstraction is introduced, however, such as with VERITAS Volume Manager or any storage virtualization product, the database catalog no longer reflects an accurate picture of the location of the physical files. Storage management vendors solve this problem by exploiting the file mapping API provided by Oracle

Using the file mapping interface APIs storage vendors create mapping libraries that illustrate the I/O stack. Oracle accesses the mapping files to build accurate physical location information into the data dictionary for each logical database entity.

As with Oracle's simple file mapping views, the file mapping interface provides DBAs with a set of dynamic performance views that can be queried to show mapping information from database object to a physical file. These views give DBAs access to the exact disk location of a specific block of data, and can tie that back to a logical database entity, providing the DBA, once again, with full picture of a databases physical I/O configuration.

How File Mapping Works

If the Oracle FILE_MAPPING variable is set to TRUE, Oracle will initialize the file mapping interface at instance startup by spawning the FMON process. FMON builds mapping information and stores it in the instance System Global Area (SGA), and, if commanded by the DBA, will save the information to file structures in the Oracle data dictionary, maintaining data persistence across instance startup and shutdown. Oracle provides special file mapping structures to house I/O stack mapping data and FMON refreshes these structures with mapping data whenever changes occur to physical file components.

```
FILE_MAPPING=TRUE
or
SQL> alter system set file_mapping=true;
```

Figure 3: Setting the Oracle File Mapping Variable

File mapping begins with vendor-supplied file mapping libraries. These libraries exist for each layer of the I/O stack, provided the storage vendor has created the libraries. The mapping information can also be consolidated into a single library, as with the VERITAS Federated Mapping Service, VxMS.

Vendor-supplied file mapping libraries communicate with FMON through a non-Oracle process called FMPUTL. This process is spawned by FMON and loads all vendor mapping libraries specified in the `filemap.ora` file, sending detailed mapping information about the I/O stack back to the FMON process.

Mapping Structures

The Oracle mapping structures files, file system extents, elements, and subelements capture information from the I/O stack and are stored in the Oracle instance SGA. A visual example of file structures can be seen in Figure 4 below.

The file mapping structure provides attributes related to the logical database entity file, including file size, number of file system extents making up the file, and the file type.

The file system extent structure describes physical contiguous blocks of data written to a device managed by the file system. These structures are distinct from Oracle's logical database extents. The file system extent structure contains the attributes device offset, extent size, file offset, type (data, parity, or data and parity), and the name of the element – the mapping structure describing a physical storage component - on which the extent resides.

The element mapping structure provides the building blocks for the file mapping interface. Each element describes a physical component within the I/O stack, for example mirrors, stripes, partitions, RAID5, concatenated elements, and disks.

The Subelement mapping structures describe the connection between different element structures in the I/O stack using the attributes subelement number, size, name of the element where the subelement exists, and the element offset.

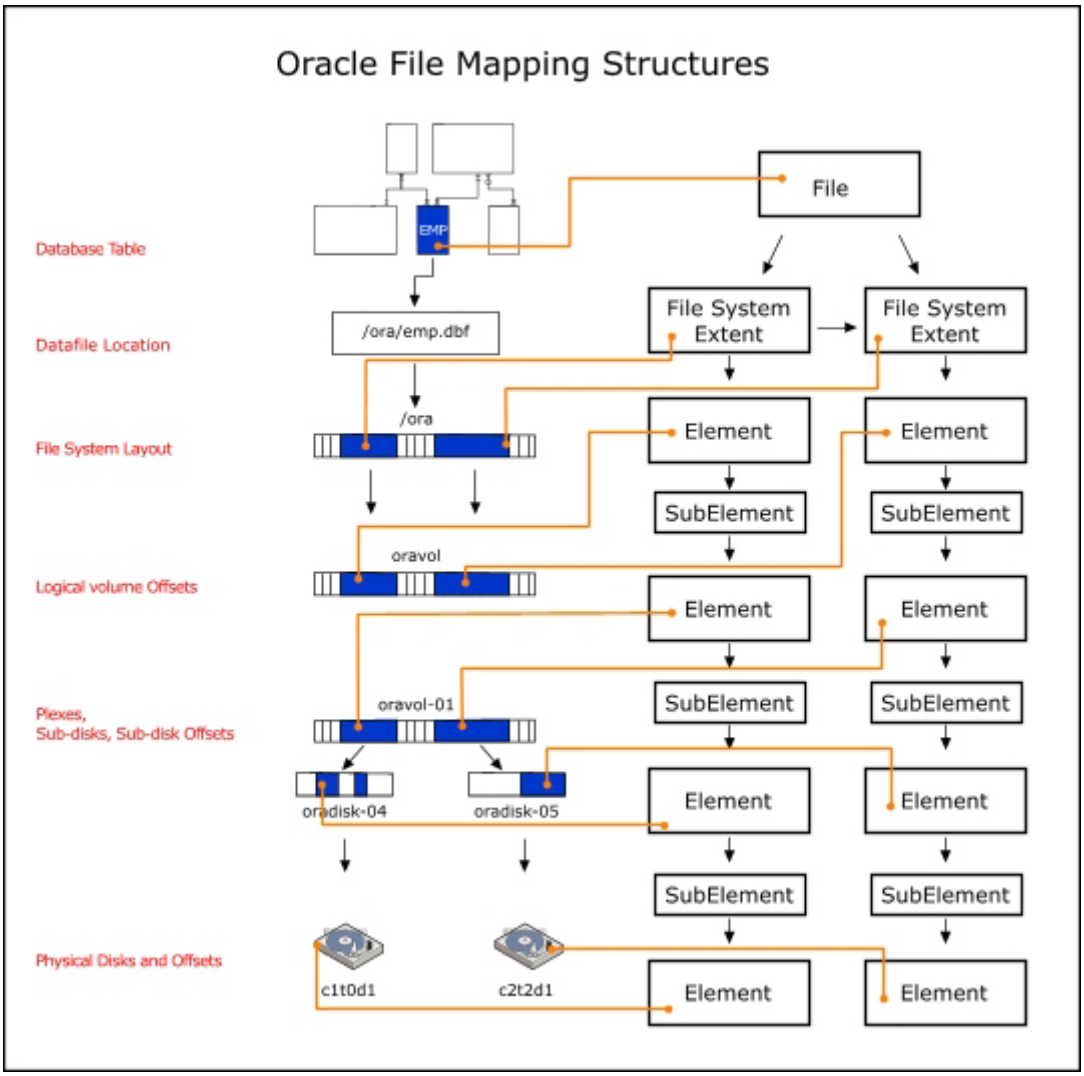

Figure 4: Oracle File Mapping Structures

Configuration Identifier

Guaranteeing accurate file structure data requires a versioning mechanism to alert Oracle when changes in the I/O stack occur. The file mapping interface uses a configuration identifier, generated by the storage vendor mapping library, to track changes in the datafile size, the addition or deletion of datafiles, and the I/O stack configuration - although these happen less frequently.

Oracle supports two types of configuration identifier, persistent and non-persistent, with the storage vendor determining which type is used by a mapping library. If persistent, only the specific changes prompting an update to the configuration identifier need to be refreshed the next time the Oracle instance starts. Non-persistent configuration identifiers require a time-consuming and I/O intensive full refresh of all mapping information at instance startup.

Implementing the Oracle File Mapping Interface

The special Oracle file `filemap.ora` in `$ORACLE_HOME/rdbms/filemap/etc` contains path information for vendor-supplied mapping libraries. Each entry in the file points to the library containing mapping information for a particular level of the I/O stack. The order in which the libraries are listed is very important as Oracle searches the libraries in the sequence specified.

Vendor mapping library entries in `filemap.ora use the` following format :

```
lib=vendor_name:mapping_library_including_path
```

For example :

```
lib=VERITAS:/opt/VRTSdbed/lib/libvxoramap_32.so
```

Figure 5: Maping Library Format in filemap.ora

Generating File Mapping Data

File mapping operations are controlled by the Oracle PL/SQL package DBMS_STORAGE_MAP. The package contains stored procedures that build and drop mapping information, and save, restore, lock, and unlock mapping data in the SGA. The table in Figure 6 shows details for each procedure in the package.

Procedure	Function
MAP_OBJECT	Build the mapping information for the Oracle object identified by object name, owner, and type
MAP_ELEMENT	Build mapping information for the specified element
MAP_FILE	Build mapping information for the specified filename
MAP_ALL	Build entire mapping information for all types of Oracle files (excluding archive logs)
DROP_ELEMENT	Drop the mapping information for a specified element
DROP_FILE	Drop the file mapping information for the specified filename
DROP_ALL	Drop all mapping information in the SGA for this instance
SAVE	Save into the data dictionary the required information needed to regenerate the entire mapping
RESTORE	Load the entire mapping information from the data dictionary into the shared memory of the instance
LOCK_MAP	Lock the mapping information in the SGA for this instance
UNLOCK_MAP	Unlock the mapping information in the SGA for this instance

Figure 6: Stored Procedures in the Oracle DBMS_STORAGE_MAP Package.

Accessing File Mapping Data

The information captured by the file mapping interface can be viewed graphically through the Oracle Enterprise Manager (OEM) and through dynamic performance views, accessed using SQL statements. The table in Figure 7 details the contents of various dynamic performance views provided by Oracle.

View	Description
V$MAP_LIBRARY	Contains a list of all mapping libraries that have been dynamically loaded by FMPUTL
V$MAP_FILE	Contains a list of all file mapping structures in the SGA
V$MAP_FILE_EXTENT	Contains a list of all file system extent mapping structures in the SGA
V$MAP_ELEMENT	Contains a list of all element mapping structures in the SGA
V$MAP_EXT_ELEMENT	Contains supplementary information for all element mapping
V$MAP_SUBELEMENT	Contains a list of all subelement mapping structures in the SGA
V$MAP_COMP_LIST	Contains supplementary information for all element mapping structures
V$MAP_FILE_IO_STACK	Contains the hierarchical arrangement of storage containers for the file. This information is displayed as a series of rows. Each row represents a level in the hierarchy

Figure 7: Oracle Dynamic Performance Views

Examples

The following example shows the SQL statements used to display elements within the Quick I/O stack for a specific file.

```
SQL> set lines 200;
SQL> column elem_name format a28;
SQL> set numwidth 8;
SQL> set echo on;
SQL> select elem_name, depth, cu_size cusz, stride, num_cu ncu,
2 file_offset FLofst, elem_offset ELofst, id, parent_id pid
3 from v$map_file_io_stack, v$map_element
4 where v$map_file_io_stack.elem_idx = v$map_element.elem_idx
5 order by file_offset;
```

Figure 8: Example SQL Statements.

```
ELEM_NAME                    DEPTH   CUSZ    STRIDE  NCU     FLOFST  ELOFST          ID      PID
---------------------------  ------ -------- ------ -------- ------  ------          ------   --
/dev/vx/rdsk/raghu_dg/ora92  0       20488   0       1       0       7733248         6       0
vxvm:raghu_dg/ora92-01       1       20488   0       1       0       7733248         7       6
/dev/vx/rdmp/c2t5d0s4        2       20488   0       1       0       7733248         8       7
/dev/rdsk/c2t5d0s4           3       20488   0       1       0       7733248         9       8
c2t5d0                       4       20488   0       1       0       7740430         10      9
```

Figure 9: The Results of Running The SQL Query in Figure 8

Conclusion

The Oracle file mapping interface, and the vendor supplied mapping libraries that plug into the API, lift the veil obscuring a DBA's view of complex storage subsystems. The information generated is essential if a DBA is to fully understand the interaction between logical database entities and the physical world of storage. Maintaining accurate file mapping information is the first step a DBA needs to take in improving diagnostic capabilities for debugging performance problems.

VERITAS Software Corporation

Corporate Headquarters
350 Ellis Street
Mountain View, CA 94043
650-527-8000 or 866-837-4827

For additional information about VERITAS Software, its products, VERITAS Architect Network, or the location of an office near you, please call our corporate headquarters or visit our Web site at www.veritas.com.