CHAPTER 6: LAYERED VOLUMES

by Volker Herminghaus

6.1 Overview

In the previous chapter you learned about volumes and the merits of the various volume layouts. You learned how easy it is to specify storage allocations and other features. The relayout feature was discussed, and we made an advance mention of the term **layered volume**. This chapter will talk about what at first looks like just two more volume layouts: **`concat-mirror`** and **`stripe-mirror`**. In the more technical sections, especially the technical deep dive, you will see what the idea behind these so-called **layered volumes** is, and how it enables such features as relayout and RAID-5 mirroring.

6.1.1 Why Use Layered Volumes?

The VxVM volume layouts that we have discussed so far all work pretty well and cover the needs of most administrators. But everything can be improved, and so volume layouts were improved, too. If we take a deep look we will find that **in the case of a disk failure**, there are a few minor issues with, for instance, a mirrored stripe. It is in this case, and only in this case, that something can be gained from using a layered volume layout instead of a standard volume layout. As long as all disks are OK, there is no noticeable advantage to either layout except for improved readability and simplicity on the part of the non-layered layouts.

In order to understand what can be improved, we need to know how a conventional layout volume behaves in case of a disk failure. Look at the layout of a conventional mir-

V. Herminghaus and A. Sriba, *Storage Management in Data Centers*,
DOI: 10.1007/978-3-540-85023-6_6, © Springer-Verlag Berlin Heidelberg 2009

rored stripe, i.e. a volume that was created with the layout specifier **mirror-stripe**:

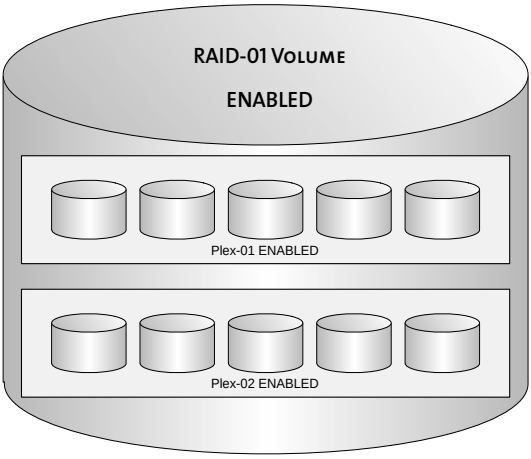

Figure 6-1: A RAID-01 volume in normal operation. Both plexes are enabled, the volume is enabled and active.

You see that the volume consists of two data plexes, i.e. two containers for identical volume data. Now the normal behavior of this volume when undergoing read-I/O is the following: The **select** volume read policy uses the **round** read method to read all plexes in a round-robin fashion. The specific details of this read method vary from array to array as the DDL (dynamic device layer) uses libraries that are specifically tuned for each storage type. But in general, first major read I/O will be satisfied from one plex, the next major read I/O will be satisfied from the next plex, and so on. In effect, this reduces the number of I/Os on each LUN and thus the queue length on the LUNs. Performance is gained by this act of load balancing, especially for the infamous **scattered read** I/O type.

But consider what happens when any one of the LUNs or disks fails: The whole plex will be detached.

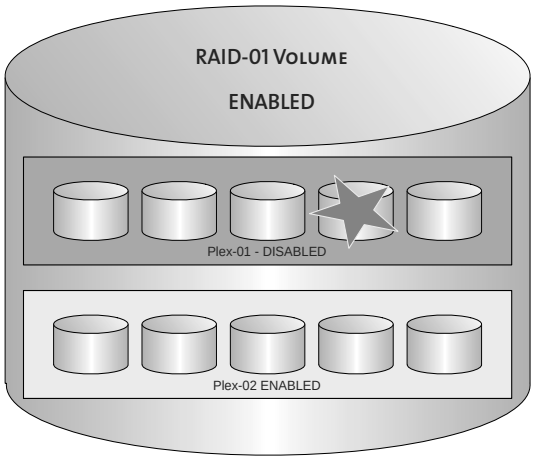

Figure 6-2: A RAID-01 volume after a single disk has failed. One plex has been disabled, the volume is still enabled and active.

This means that even if only one out of the, say, five LUNs making up a striped plex fails, all five disks will stop doing any work for us! They will just idle; nobody asks them for data any more. And that is for good reason, because the plex is now detached and its data is becoming rapidly out of sync with the active plex. You don't want stale data, do you?

This is the first problem that will be solved by layered volumes.

Now consider a second disk failing. The probability of the other plex being affected is fifty percent (actually it is a little higher, since the failed disk cannot fail again). That means the other plex is now detached, too, and the volume has become unusable. We might have valid data on four out of five columns of one plex, and also have valid data on some other four out of five columns of the other plex, but still the volume is unusable. Does that make any sense?

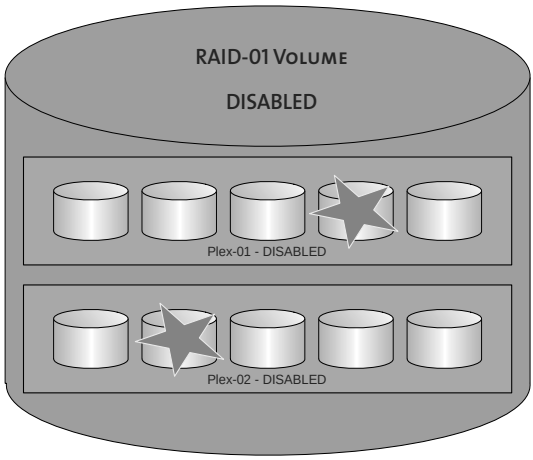

Figure 6-3: A RAID-01 volume after a second disk has failed, on the other plex. Both plexes have been disabled, the volume is disabled.

Well, it does, because that is how VxVM was designed, and the way it was designed (and BTW the way most high-availability software was designed) is to compensate for any **one** fault, but not for double-faults. Why do we not compensate for two faults? Because compensating for just one fault alone requires a lot of design the software, architecting the solution, and operating it. There are dozens of possible single points of failure that we need to guard against. If you had to plan for double faults, that would be not two times dozens, but dozens times dozens! It is simply too complex to even try. Therefore it has been almost universally agreed that a double-fault is nothing that a high-availability solution needs to worry about. Not because it cannot happen, but because it would be impossible to catch reliably anyway.

Losing a second disk without losing the volume was, although this feature did not have to be implemented for "official" reasons, the second problem that was solved with layered volumes.

Going back to our single-fault situation, however, we find another thing that does not work as well as it could: Imagine if you replace the failed disk, and that was one out of five disks that made up the detached plex. What needs to be done? Volume manager needs to resynchronize the whole plex! Remember that as soon as the plex was detached due to the disk failure, no more read or write I/O was performed to it. So at the very minimum all those regions that have been written to must now be copied to the fixed plex before it can be read again. But wait! So far we have not learned of any way to track where changes have been made, which regions have been written to. That means that indeed we must resynchronize the whole plex. Five times as much as would have been necessary if we had kept those other four disks in sync. But that was impossible because they belong to the same plex, and VxVM cannot detach just part of a plex, but only a whole plex.

So, to sum it up, there are three things wrong with volumes **in case of a disk failure**:

1) Load balancing is lost due to the whole plex being detached.
2) For the same reason, the whole plex needs to be resynchronised after disk repair.
3) While the plex is detached, failure of any disk in the other plex renders the volume unusable (if we only have two plexes - we could have more than two and the volume would still be online, of course).

More information on disk and plex failures and how to cope with them can be found in the troubleshooting chapter beginning on page 349.

OK, let's get started and make some layered volumes and look at them in the Easy Sailing chapter!

6.2 Introducing Layered Volumes

The idea of layering comes from the following fact, that eventually hit the designers of VxVM:

In conventional volumes, space is allocated from physical disks. Physical disks have some limitations as to their size and reliability etc. So the subdisks that reside on those physical disks share their limitations.

Why can we not use volumes – instead of physical disks or LUNs - as the basis for subdisks? If we could use volumes instead of disks as the basis for subdisk allocation, then the subdisks could be made redundant or arbitrarily sized on their own, freeing us from some limitations such as the ones we see when a failed disks leads to a whole detached plex.

If the subdisks were allocated from – usually redundant – volumes, then a failing disk would not make the subdisk fail (it is redundant and therefore survives a single failure). Instead, the subdisk would continue to run fine, and the volume would be unaffected. Only one layer deeper, inside the volume that the subdisk was allocated from, would the corresponding plex be detached. The subvolume would be degraded because the plex that contains the failed disk is no longer active. But the subvolume itself as well as the volume that builds on top of it would be unaffected. See the following graphics and compare them to the ones about the RAID-01 volume.

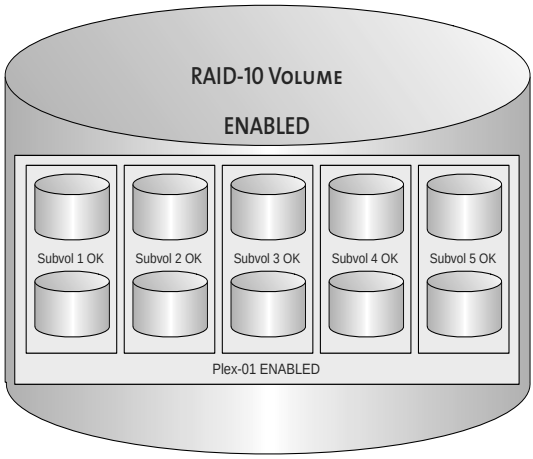

Figure 6-4: A volume in RAID-10 layout. Note how the user data is structured in exactly the same way as in the case of RAID-01. Only the metadata is rearranged to group the subdisks differently, yielding higher resilience.

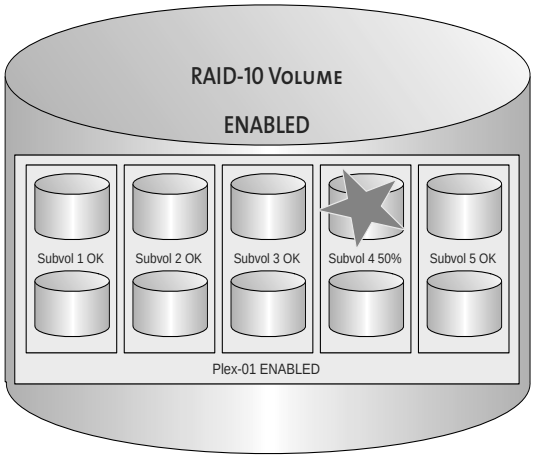

Figure 6-5: A volume in RAID-10 layout after a single disk has failed. The volume is only slightly degraded, as load balancing still takes place on most of it. Also, recovery would only require resynchronisation of a small portion of the total volume. The unaffected parts are still updated and need not be synchronized after replacing the failed disk (only the mirrored subvolume need to be resynchronised).

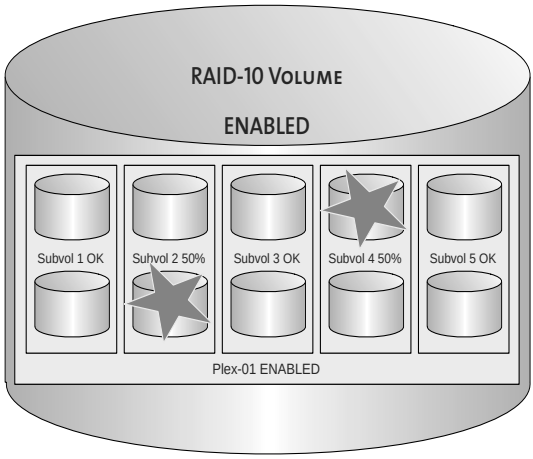

Figure 6-6: **A volume in RAID-10 layout after another disk failure. The volume is still accessible. In fact, it could bear three more disk failures, provided they all happen to different columns.**

Being so clearly superior to RAID-01, the concept of layering (RAID-10) was added to VxVM. Layering is implemented transparently, which means that vxassist will build for you a base layer of (redundant) volumes and then allocate subdisks from that base layer to form a volume on a higher layer without you having to do anything differently from before. Let's try it, it's really simple.

6.2.1 CONCAT-MIRROR

To create a layered mirror you just use a different volume layout specification: **concat-mirror** instead of **mirror-concat** or instead of specifying the individual attribute as a comma-separated list (**concat,mirror**).

Synopsis for creating a layered volume with a concat-mirror layout:

```
# vxassist make avol 1g layout=concat-mirror init=active
```

But look at what this created and try to understand it:

```
# vxprint -qvhtgadg
v  avol         -            ENABLED  ACTIVE   2097152  SELECT    -        fsgen
pl avol-03      avol         ENABLED  ACTIVE   2097152  CONCAT    -        RW
sv avol-S01     avol-03      avol-L01 1        2097152  0         2/2      ENA

v  avol-L01     -            ENABLED  ACTIVE   2097152  SELECT    -        fsgen
pl avol-P01     avol-L01     ENABLED  ACTIVE   2097152  CONCAT    -        RW
sd adg01-03     avol-P01     adg01    204800   2097152  0         c0t2d0   ENA
```

```
pl avol-P02     avol-L01     ENABLED  ACTIVE   2097152  CONCAT    -        RW
sd adg02-03     avol-P02     adg02    204800   2097152  0         c0t3d0   ENA
```

That is the one major drawback: a layered volume consists of more objects and therefore its layout is harder to understand by looking at the output of **vxprint.** It looks like two volumes were created (which is true). If you look closely at the highlighted words in the third line of output you will see that it is not a subdisk (**sd**), but a subvolume (**sv**), i.e. a subdisk that is based on a volume instead of a disk. You will also see where the subvolume was allocated from: **avol-L01**. And if you look at the next line you see that the second volume that was created is actually called **avol-L01**. In other words, the **vxassist** command created a mirrored volume, then used that volume as the basis for allocating a subdisk for another volume. That is why this is called a layered volume: one volume lies on top of the other.

We will give you some help in understanding the layout very soon; it's actually not as complicated as it first looks.

6.2.2 STRIPE-MIRROR

Creating a layered striped mirror is just as easy as creating a layered concat mirror. Of course the layout parameter is different: **stripe-mirror** instead of **mirror-stripe** or **mirror,stripe**.

Synopsis for creating a layered volume with a stripe-mirror layout:

```
# vxassist make avol 1g layout=stripe-mirror init=active
# vxprint -qvhtgadg
v  avol         -            ENABLED  ACTIVE   2097152  SELECT    avol-03  fsgen
pl avol-03      avol         ENABLED  ACTIVE   2097408  STRIPE    3/128    RW
sv avol-S01     avol-03      avol-L01 1        699136   0/0       2/2      ENA
sv avol-S02     avol-03      avol-L02 1        699136   1/0       2/2      ENA
sv avol-S03     avol-03      avol-L03 1        699136   2/0       2/2      ENA

v  avol-L01     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
pl avol-P01     avol-L01     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg01-03     avol-P01     adg01    204800   699136   0         c0t2d0   ENA
pl avol-P02     avol-L01     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg04-03     avol-P02     adg04    204800   699136   0         c0t10d0  ENA

v  avol-L02     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
pl avol-P03     avol-L02     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg02-03     avol-P03     adg02    204800   699136   0         c0t3d0   ENA
pl avol-P04     avol-L02     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg05-03     avol-P04     adg05    204800   699136   0         c0t11d0  ENA

v  avol-L03     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
pl avol-P05     avol-L03     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg03-03     avol-P05     adg03    204800   699136   0         c0t4d0   ENA
```

```
pl avol-P06     avol-L03     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg06-03     avol-P06     adg06    204800   699136   0         c0t12d0  ENA
```

6.2.3 Understanding vxprint Output for Layered Volumes

So far, so good. We have just created our first layered volumes. Now how could we more easily understand the **vxprint** output for a layered volume? It's actually not so hard. Once you know the basics the secrets reveal themselves.

Let's use three different output formats for **vxprint** to try and combine understanding of the volume layout with readability in your day-to-day job. The highlighted parts are what you need to inspect more closely; they contain the layout of the volume and the layout of the subvolume. The other parts are redundant because they show only the layouts of the other subvolumes.. When created with vxassist, all subvolumes share the same layout so they do not need further attention unless a volume has a problem.

Now here are the formats:

1. Easy but maybe too easy, and confusing us with extra volumes: **vxprint -ht**

```
# vxprint -ht -g adg
<...>
v  avol         -            ENABLED  ACTIVE   2097152  SELECT    avol-03  fsgen
pl avol-03      avol         ENABLED  ACTIVE   2097408  STRIPE    3/128    RW
sv avol-S01     avol-03      avol-L01 1        699136   0/0       2/2      ENA
sv avol-S02     avol-03      avol-L02 1        699136   1/0       2/2      ENA
sv avol-S03     avol-03      avol-L03 1        699136   2/0       2/2      ENA

v  avol-L01     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
pl avol-P01     avol-L01     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg01-03     avol-P01     adg01    204800   699136   0         c0t2d0   ENA
pl avol-P02     avol-L01     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg04-03     avol-P02     adg04    204800   699136   0         c0t10d0  ENA

v  avol-L02     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
pl avol-P03     avol-L02     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg02-03     avol-P03     adg02    204800   699136   0         c0t3d0   ENA
pl avol-P04     avol-L02     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg05-03     avol-P04     adg05    204800   699136   0         c0t11d0  ENA

v  avol-L03     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
pl avol-P05     avol-L03     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg03-03     avol-P05     adg03    204800   699136   0         c0t4d0   ENA
pl avol-P06     avol-L03     ENABLED  ACTIVE   699136   CONCAT    -        RW
sd adg06-03     avol-P06     adg06    204800   699136   0         c0t12d0  ENA
```

The output of this command consists of several parts, separated by blank lines. First you see the high-level volume with its solitary plex and the subdisks. But as showed in the

initial example they are now based not on disks but on volumes and are therefore called not subdisks but subvolumes (they are listed as type **sv**). OK, so subvolumes are in the plex instead of subdisks. Where are the volumes that the subdisks were allocated from? They are right below the volume, listed as normal volumes (which they are **not**, in fact. They do not have a device driver in /dev/vx/*dsk/... to write to them). If we want to look at the top layer volume's layout we inspect the plex line of that volume. In the layout column it will either say **STRIPE** or **CONCAT** meaning it will be either a **stripe-mirror** or **concat-mirror**, respectively. Next, to find out if the base volumes are mirrored once, twice, or more often, we will look at the layout of one of them (their layout is always identical if they were created with vxassist). If we see a base volume with two data plexes, that's simple mirroring. If the base volume shows three data plexes, that means the layered volume is a three-way mirror etc. It's actually very simple. The main problem with this output format is that it is not immediately obvious where the individual subvolumes belong in the layered volume. Besides, they look just like normal volumes. A shell script might confuse them with normal volumes and try to start them, create snapshots from them etc., which would fail because they really are not independent volumes but subvolumes belonging to a higher level virtual object.

2. Compact but certainly not easy: **vxprint -rt**

```
# vxprint -rt -g adg
<...>
v  avol         -            ENABLED  ACTIVE   2097152  SELECT    avol-03  fsgen
pl avol-03      avol         ENABLED  ACTIVE   2097408  STRIPE    3/128    RW
sv avol-S01     avol-03      avol-L01 1        699136   0/0       2/2      ENA
v2 avol-L01     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
p2 avol-P01     avol-L01     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg01-03     avol-P01     adg01    204800   699136   0         c0t2d0   ENA
p2 avol-P02     avol-L01     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg04-03     avol-P02     adg04    204800   699136   0         c0t10d0  ENA
sv avol-S02     avol-03      avol-L02 1        699136   1/0       2/2      ENA
v2 avol-L02     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
p2 avol-P03     avol-L02     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg02-03     avol-P03     adg02    204800   699136   0         c0t3d0   ENA
p2 avol-P04     avol-L02     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg05-03     avol-P04     adg05    204800   699136   0         c0t11d0  ENA
sv avol-S03     avol-03      avol-L03 1        699136   2/0       2/2      ENA
v2 avol-L03     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
p2 avol-P05     avol-L03     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg03-03     avol-P05     adg03    204800   699136   0         c0t4d0   ENA
p2 avol-P06     avol-L03     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg06-03     avol-P06     adg06    204800   699136   0         c0t12d0  ENA
```

This output format is for the experienced administrator who has seen a lot of layered volumes already and can parse them almost instantly. What you see is an apparently unstructured lot of output lines with no helpful separation at all. But in exchange for the separation you get some more valuable information: the subvolumes are now listed with their layer number in the type field: a volume on layer 2 (i.e. one down from the top layer,

the regular base layer) will not be shown as a **v** but as a **v2**; a volume on layer 2. Likewise, a plex inside such a volume will not be a **pl** object but a **p2** object, and a subdisk not an **sd** object but an **s2** object. That makes this output format better palatable for scripts and long-time VxVM hackers, who prefer it for its concise look.

3. Less compact and relatively easy: **vxprint -rtL**

```
# vxprint -rtL -g adg
<...>
v  avol         -            ENABLED  ACTIVE   2097152  SELECT    avol-03  fsgen
pl avol-03      avol         ENABLED  ACTIVE   2097408  STRIPE    3/128    RW

sv avol-S01     avol-03      avol-L01 1        699136   0/0       2/2      ENA
v2 avol-L01     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
p2 avol-P01     avol-L01     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg01-03     avol-P01     adg01    204800   699136   0         c0t2d0   ENA
p2 avol-P02     avol-L01     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg04-03     avol-P02     adg04    204800   699136   0         c0t10d0  ENA

sv avol-S02     avol-03      avol-L02 1        699136   1/0       2/2      ENA
v2 avol-L02     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
p2 avol-P03     avol-L02     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg02-03     avol-P03     adg02    204800   699136   0         c0t3d0   ENA
p2 avol-P04     avol-L02     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg05-03     avol-P04     adg05    204800   699136   0         c0t11d0  ENA

sv avol-S03     avol-03      avol-L03 1        699136   2/0       2/2      ENA
v2 avol-L03     -            ENABLED  ACTIVE   699136   SELECT    -        fsgen
p2 avol-P05     avol-L03     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg03-03     avol-P05     adg03    204800   699136   0         c0t4d0   ENA
p2 avol-P06     avol-L03     ENABLED  ACTIVE   699136   CONCAT    -        RW
s2 adg06-03     avol-P06     adg06    204800   699136   0         c0t12d0  ENA
```

The last output format may be the one best suited to sophisticated newcomers. It looks just like the previous, more demanding format, but inserts blank lines between the subvolumes so it is easier to parse to the untrained eye. You may even want to stick with this output format forever.

6.3 Understanding Layered Volumes

6.3.1 Manually Creating a Layered Volume

As always we would like to walk you through the individual virtual objects for this topic. We will now create a layered volume, a three-column stripe-mirror of 300MB to be exact, all by ourselves using only our bare hands and a UNIX shell! Actually we will use **vxassist**, too, but only for the first steps.

```
# export VXVM_DEFAULTDG=adg
# cd /dev/vx/dsk/adg
# ls -l
total 0
# vxprint -qvrtg adg
<nothing>
```

We start with an empty disk group. First we create some mirrored and therefore redundant volumes that we can later use to allocate subdisks from.

```
# vxassist make col0vol 100m layout=mirror init=active
# vxassist make col1vol 100m layout=mirror init=active
# vxassist make col2vol 100m layout=mirror init=active
```

Now let's try to allocate a subdisk from the volumes:

```
# vxmake sd col0sd disk=col0vol offset=0 len=100m
VxVM vxmake ERROR V-5-1-10127 creating subdisk col0sd:
        Volume does not have the storage attribute
```

OK, that doesn't seem to work. Volume manager tells us the volume "**does not have the storage attribute**", i.e. it is not storage that is usable for VxVM. Why is that? Easy, check this out:

```
# ls -l
total 0
brw-------   1 root     root     270, 3000 Jun  8 01:42 col0vol
brw-------   1 root     root     270, 3001 Jun  8 01:42 col1vol
brw-------   1 root     root     270, 3002 Jun  8 01:42 col2vol
```

If the device nodes are visible, they could be used for file system, database and raw

device access. Do we really want to use these publicly accessible devices as parts of a volume? Certainly not, since there would be no protection against somebody unknowingly writing to the devices, thus wrecking the contents of the aggregate volume. Besides, if the subvolumes keep existing as a separate entity we could start and stop it individually etc. In short, it would be a real mess trying to coordinate or rather separate accesses to the volume as a standalone entity or as part of a layered volume. That is why VxVM says the volume "**does not have the storage attribute**". For VxVM, this volume is not storage like disk or LUN space. It is a virtual object that exists purely for the user's benefit. There would be nothing to stop VxVM from using a volume as the basis for storage allocation, because volumes so closely resemble actual disks (well, partitions, but in for the purpose of allocation it is really the same). But it refuses to do so because the user might use a different access path to the same data: the volume device driver in **/dev/vx/*dsk/<DG>/<Volume>**. So a volume needs to be explicitly turned into "storage" first.

How **do** we turn a volume into storage that can be used by VxVM for internal allocation? There's a VxVM attribute that we can set to do so. It is called the **layered** attribute. If we set it to on, the volume will magically be turned into storage usable by VxVM:

```
# vxedit set layered=on col0vol
# ls -l
    total 0
brw-------   1 root     root     270, 3001 Jun  8 01:42 col1vol
brw-------   1 root     root     270, 3002 Jun  8 01:42 col2vol
```

Aha! The volume on which we set the **layered** attribute to **on** has had its device driver removed! With this trick VxVM is now able to allocate subdisks from this volume just like it would from a physical disk. There is no longer any danger of uncoordinated parallel access any more, since the volume is no longer accessible by the user. So let's try allocating a subdisk again on col0vol:

```
# vxmake sd col0sd disk=col0vol offset=0 len=100m
# vxprint -qrtg adg
<...>
SD NAME         PLEX         DISK     DISKOFFS LENGTH   [COL/]OFF DEVICE   MODE
<...>
sd col0sd       -            col0vol  0        204800   -         -        ENA
v2 col0vol      -            ENABLED  ACTIVE   204800   SELECT    -        fsgen
p2 col0vol-01   col0vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg01-01     col0vol-01   adg01    0        204800   0         c0t2d0   ENA
p2 col0vol-02   col0vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg02-01     col0vol-02   adg02    0        204800   0         c0t3d0   ENA
<...>
```

This actually seems to work! Look at how the objects are defined: The type in the left most column show that the first element is an sd record, i.e. a subdisk. In column four of the subdisk, where there used to be names like adg01, adg02 etc., there is the name of the volume: col0vol! That is actually rather obvious and straightforward when you think of it.

Of course, for our convenience the actual volume is appended to the subdisk line as

a **v2** record (volume on layer 2, with plexes being **p2** and subdisks being **s2**), so that the internal layout characteristics of the subvolume can be easily determined (number of mirrors and logs etc.).

To finish manufacturing a layered volume we need to repeat the above process accordingly with the other volumes:

```
# vxedit set layered=on col1vol col2vol
# ls -l
total 0
# vxmake sd col1sd disk=col1vol offset=0 len=100m
# vxmake sd col2sd disk=col2vol offset=0 len=100m
# vxprint -qrtg adg
<...>
sd col0sd       -            col0vol  0        204800   -        -        ENA
v2 col0vol      -            ENABLED  ACTIVE   204800   SELECT   -        fsgen
p2 col0vol-01   col0vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg01-01     col0vol-01   adg01    0        204800   0        c0t2d0   ENA
p2 col0vol-02   col0vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg02-01     col0vol-02   adg02    0        204800   0        c0t3d0   ENA

sd col1sd       -            col1vol  0        204800   -        -        ENA
v2 col1vol      -            ENABLED  ACTIVE   204800   SELECT   -        fsgen
p2 col1vol-01   col1vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg03-01     col1vol-01   adg03    0        204800   0        c0t4d0   ENA
p2 col1vol-02   col1vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg04-01     col1vol-02   adg04    0        204800   0        c0t10d0  ENA

sd col2sd       -            col2vol  0        204800   -        -        ENA
v2 col2vol      -            ENABLED  ACTIVE   204800   SELECT   -        fsgen
p2 col2vol-01   col2vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg05-01     col2vol-01   adg05    0        204800   0        c0t11d0  ENA
p2 col2vol-02   col2vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg06-01     col2vol-02   adg06    0        204800   0        c0t12d0  ENA
```

So we got all three of our subdisks together. All we need to do now is actually make a stripe from them. That's not so hard, just put them into a plex together, and throw the plex into a volume, then wrap a volume around it with the appropriate usage type (**fsgen** for file system generic). Creating a striped plex manually using **vxmake** requires the use of both the ncol and the **stwidth** parameter, so we'll pick something reasonable like **ncol=3** and 1 MB stripesize.

```
# vxmake plex manualvol-01 layout=stripe ncol=3 \
        sd=col0sd,col1sd,col2sd stwidth=1m
# vxprint -qrtg adg
<...>
pl manualvol-01 -            DISABLED -        614400   STRIPE   3/2048   RW
sv col0sd       manualvol    col0vol  1        204800   0/0      2/2      ENA
```

```
v2 col0vol      -            ENABLED  ACTIVE   204800   SELECT    -        fsgen
p2 col0vol-01   col0vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg01-01     col0vol-01   adg01    0        204800   0         c0t2d0   ENA
p2 col0vol-02   col0vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg02-01     col0vol-02   adg02    0        204800   0         c0t3d0   ENA
sv col1sd       manualvol    col1vol  1        204800   1/0       2/2      ENA
v2 col1vol      -            ENABLED  ACTIVE   204800   SELECT    -        fsgen
p2 col1vol-01   col1vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg03-01     col1vol-01   adg03    0        204800   0         c0t4d0   ENA
p2 col1vol-02   col1vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg04-01     col1vol-02   adg04    0        204800   0         c0t10d0  ENA
sv col2sd       manualvol    col2vol  1        204800   2/0       2/2      ENA
v2 col2vol      -            ENABLED  ACTIVE   204800   SELECT    -        fsgen
p2 col2vol-01   col2vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg05-01     col2vol-01   adg05    0        204800   0         c0t11d0  ENA
p2 col2vol-02   col2vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg06-01     col2vol-02   adg06    0        204800   0         c0t12d0  ENA
```

Note how the subdisk virtual objects (**sd**) have turned into subvolume virtual objects (**sv**) to denote that their base storage is not a disk, but a volume.

```
# vxmake vol manualvol usetype=fsgen plex=manualvol-01
# vxprint -qrtg adg
<...>
v  manualvol    -            DISABLED EMPTY    614400   ROUND     -        fsgen
pl manualvol-01 manualvol    DISABLED EMPTY    614400   STRIPE    3/2048   RW
sv col0sd       manualvol-01 col0vol  1        204800   0/0       2/2      ENA
v2 col0vol      -            ENABLED  ACTIVE   204800   SELECT    -        fsgen
p2 col0vol-01   col0vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg01-01     col0vol-01   adg01    0        204800   0         c0t2d0   ENA
p2 col0vol-02   col0vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg02-01     col0vol-02   adg02    0        204800   0         c0t3d0   ENA
sv col1sd       manualvol-01 col1vol  1        204800   1/0       2/2      ENA
v2 col1vol      -            ENABLED  ACTIVE   204800   SELECT    -        fsgen
p2 col1vol-01   col1vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg03-01     col1vol-01   adg03    0        204800   0         c0t4d0   ENA
p2 col1vol-02   col1vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg04-01     col1vol-02   adg04    0        204800   0         c0t10d0  ENA
sv col2sd       manualvol-01 col2vol  1        204800   2/0       2/2      ENA
v2 col2vol      -            ENABLED  ACTIVE   204800   SELECT    -        fsgen
p2 col2vol-01   col2vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg05-01     col2vol-01   adg05    0        204800   0         c0t11d0  ENA
p2 col2vol-02   col2vol      ENABLED  ACTIVE   204800   CONCAT    -        RW
s2 adg06-01     col2vol-02   adg06    0        204800   0         c0t12d0  ENA
```

All we have left to do is to start the volume now:

```
# vxvol start manualvol
VxVM vxvol INFO V-5-1-12459 Volume col0vol of diskgroup adg is already started
VxVM vxvol INFO V-5-1-12459 Volume col1vol of diskgroup adg is already started
VxVM vxvol INFO V-5-1-12459 Volume col2vol of diskgroup adg is already started
```

Ignore the INFO messages; VxVM just tells us that it did not need to start the internal volumes because they were running already (the **vxassist make** command started them automatically). When we check the volume now, we find a perfectly good layered volume ready to serve us:

```
# vxprint -qrtg adg
<...>
v  manualvol    -            ENABLED  ACTIVE   614400   ROUND    -        fsgen
pl manualvol-01 manualvol    ENABLED  ACTIVE   614400   STRIPE   3/2048   RW
sv col0sd       manualvol-01 col0vol  1        204800   0/0      2/2      ENA
v2 col0vol      -            ENABLED  ACTIVE   204800   SELECT   -        fsgen
p2 col0vol-01   col0vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg01-01     col0vol-01   adg01    0        204800   0        c0t2d0   ENA
p2 col0vol-02   col0vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg02-01     col0vol-02   adg02    0        204800   0        c0t3d0   ENA
sv col1sd       manualvol-01 col1vol  1        204800   1/0      2/2      ENA
v2 col1vol      -            ENABLED  ACTIVE   204800   SELECT   -        fsgen
p2 col1vol-01   col1vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg03-01     col1vol-01   adg03    0        204800   0        c0t4d0   ENA
p2 col1vol-02   col1vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg04-01     col1vol-02   adg04    0        204800   0        c0t10d0  ENA
sv col2sd       manualvol-01 col2vol  1        204800   2/0      2/2      ENA
v2 col2vol      -            ENABLED  ACTIVE   204800   SELECT   -        fsgen
p2 col2vol-01   col2vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg05-01     col2vol-01   adg05    0        204800   0        c0t11d0  ENA
p2 col2vol-02   col2vol      ENABLED  ACTIVE   204800   CONCAT   -        RW
s2 adg06-01     col2vol-02   adg06    0        204800   0        c0t12d0  ENA
```

6.3.2 Mirroring RAID-5 Volumes

We certainly are not great fans of software RAID-5, but we are fans of using whatever possibilities a software offers us, even if it is just to double-check if we got everything right so far. So let's try and trick VxVM into doing something it does not normally do: mirroring a RAID-5 plex. Look what happens normally when you try to do that (we use the **nolog** keyword here purely because it makes parsing the vxprint output a little easier. We also use init=zero to speed up synchronisation of parity data):

```
# vxassist make raid5vol 100m layout=raid,nolog ncol=3 init=zero
# vxprint -qrtg adg
v  raid5vol     -            ENABLED  ACTIVE   204800   RAID     -        raid5
```

```
pl raid5vol-01  raid5vol     ENABLED  ACTIVE   204800   RAID      3/32     RW
sd adg01-01     raid5vol-01  adg01    0        102400   0/0       c0t2d0   ENA
sd adg02-01     raid5vol-01  adg02    0        102400   1/0       c0t3d0   ENA
sd adg03-01     raid5vol-01  adg03    0        102400   2/0       c0t4d0   ENA
# vxassist mirror raid5vol
VxVM vxassist ERROR V-5-1-344 avol: RAID-5 volumes cannot be mirrored
```

Well, we're not going to believe that. But we have to hide the fact that the layout is RAID-5 from VxVM. So we set the "storage attribute", allocate a subdisk from what has now become VxVM–usable storage, wrap a plex and a volume around it and end up with a layered RAID-5 volume.

```
# vxedit set layered=on raid5vol
# vxmake sd raid5sd disk=raid5vol len=100m offset=0
# vxprint -rtqgadg
<...>
sd raid5sd      -            raid5vol 0        204800   -         -        ENA
v2 raid5vol     -            ENABLED  ACTIVE   204800   RAID      -        raid5
p2 raid5vol-01  raid5vol     ENABLED  ACTIVE   204800   RAID      3/32     RW
s2 adg01-01     raid5vol-01  adg01    0        102400   0/0       c0t2d0   ENA
s2 adg02-01     raid5vol-01  adg02    0        102400   1/0       c0t3d0   ENA
s2 adg03-01     raid5vol-01  adg03    0        102400   2/0       c0t4d0   ENA
# vxmake plex layraid5vol-01 sd=raid5sd
# vxmake vol layraid5vol usetype=fsgen plex=layraid5vol-01
# vxvol start layraid5vol
# vxprint -rtqgadg
<...>
v  layraid5vol  -            ENABLED  ACTIVE   204800   ROUND     -        fsgen
pl layraid5vol-01 layraid5vol ENABLED ACTIVE   204800   CONCAT    -        RW
sv raid5sd      layraid5vol-01 raid5vol 1      204800   0         1/1      ENA
v2 raid5vol     -            ENABLED  ACTIVE   204800   RAID      -        raid5
p2 raid5vol-01  raid5vol     ENABLED  ACTIVE   204800   RAID      3/32     RW
s2 adg01-01     raid5vol-01  adg01    0        102400   0/0       c0t2d0   ENA
s2 adg02-01     raid5vol-01  adg02    0        102400   1/0       c0t3d0   ENA
s2 adg03-01     raid5vol-01  adg03    0        102400   2/0       c0t4d0   ENA
```

Here we are, the proud owners of a layered RAID-5 volume. Now we can mirror it easily because VxVM does not check the conditions inside layered storage the same as it does otherwise.

```
# vxassist mirror layraid5vol
# vxprint -rtqgadg
<...>
v  layraid5vol  -            ENABLED  ACTIVE   204800   ROUND     -        fsgen
pl layraid5vol-01 layraid5vol ENABLED ACTIVE   204800   CONCAT    -        RW
sv raid5sd      layraid5vol-01 raid5vol 1      204800   0         2/2      ENA
v2 raid5vol     -            ENABLED  ACTIVE   204800   RAID      -        raid5
```

```
p2 raid5vol-01  raid5vol     ENABLED  ACTIVE   204800   RAID      3/32     RW
s2 adg01-01     raid5vol-01  adg01    0        102400   0/0       c0t2d0   ENA
s2 adg02-01     raid5vol-01  adg02    0        102400   1/0       c0t3d0   ENA
s2 adg03-01     raid5vol-01  adg03    0        102400   2/0       c0t4d0   ENA
p2 layraid5vol-P01 raid5vol  ENABLED  LOG      205056   STRIPE    3/128    RW
s2 adg04-01     layraid5vol-P01 adg04 0        68352    0/0       c0t10d0  ENA
s2 adg05-01     layraid5vol-P01 adg05 0        68352    1/0       c0t11d0  ENA
s2 adg06-01     layraid5vol-P01 adg06 0        68352    2/0       c0t12d0  ENA
```

The new plex even has the same number of columns! But it's a stripe, not a parity-stripe, therefore the subdisks in the second plex are smaller (they don't need to hold the additional parity data).

With a little more work we could, of course, have built another RAID-5 volume, layered that one, too, and put it into the volume to create a pure RAID-5 to RAID-5 mirror, but we are sure that with the knowledge you just gained you are able to do it yourself if you are so inclined.