



CHAPTER 1

Oracle and Availability: Illustrated Downtime Scenarios

4 Oracle Database 10g High Availability



As you have already discovered, or perhaps are about to discover, maintaining the high-availability database is a tricky prospect. There are hundreds of possible sources of downtime hiding in the crooks and crannies of your enterprise. Many outage situations cascade into each other, where an attempt to resolve a problem creates another outage and quickly spirals further and further out of control. But almost all downtime scenarios can be prevented with careful planning or swift corrective action.

Throughout this book, we discuss numerous technologies that prevent or drastically reduce different database outages. But what are these downtime scenarios? Which possible outages can you protect yourself against? In order to illustrate the various types of situations that threaten the availability of your database, we will illustrate various database problems that can be prevented using the technologies outlined in this book. We have typically provided a worst-case scenario, but we wanted to place the high-availability technologies in Chapters 2 through 11 in a real-world context prior to exploring the configuration and administration of these technologies.

After each downtime scenario, a text box will describe which piece of technology could be employed to prevent the outage, or to recover from it quickly. If you are not interested in reading through the situations, you can skim through this chapter and just look for the boxes. The boxes provide a road map to the available technologies that are discussed in the rest of this book.

Horatio's Woodscrews

For the situations in this chapter, and for the workshops and examples throughout the book, we will use the database from the company Horatio's Woodscrews, Inc. This is a fictitious company that sells...well...woodscrews. This company has a primary database that holds millions of records concerning woodscrews, woodscrew inventory, and woodscrew orders. The following is the code to create the application owner in the oracle database, along with the three primary tables: Woodscrew, woodscrew_inventory, and woodscrew_orders. These tables will be used throughout the book for all examples, so you might want to make note of this information. After building the tables, a few rows are added so we can manipulate them for labs, like this:

```
create tablespace ws_app_data datafile
'/u01/product/oracle/oradata/orcl/ws_app_data01.dbf' size 100m;
create tablespace ws_app_idx datafile
'/u01/product/oracle/oradata/orcl/ws_app_idx01.dbf' size 100m;

create user ws_app identified by ws_app
default tablespace ws_app_data
temporary tablespace temp;

grant connect, resource to ws_app;
connect ws_app/ws_app;
```

```

create table woodscrew (
scr_id          number not null,
manufactr_id   varchar2(20) not null,
scr_type       varchar2(20),
thread_cnt     number,
length        number,
head_config    varchar2(20),
constraint pk_woodscrew primary key (scr_id, manufactr_id)
using index tablespace ws_app_idx);

create index woodscrew_identity
  on woodscrew(scr_type, thread_cnt, length, head_config)
tablespace ws_app_idx;

create table woodscrew_inventory (
scr_id          number not null,
manufactr_id   varchar2(20) not null,
warehouse_id   number not null,
region         varchar2(20),
count          number,
lot_price      number);

create table woodscrew_orders (
ord_id         number not null,
ord_date       date,
cust_id        number not null,
scr_id         number not null,
ord_cnt        number,
warehouse_id   number not null,
region         varchar2(20),
constraint pk_wdscr_orders primary key (ord_id, ord_date)
using index tablespace ws_app_idx);

---- Now, add rows to the tables.

insert into woodscrew values (
1000, 'Tommy Hardware', 'Finish', 30, 1.5, 'Phillips');
insert into woodscrew values (
1000, 'Balaji Parts, Inc.', 'Finish', 30, 1.5, 'Phillips');
insert into woodscrew values (
1001, 'Tommy Hardware', 'Finish', 30, 1, 'Phillips');
insert into woodscrew values (
1001, 'Balaji Parts, Inc.', 'Finish', 30, 1, 'Phillips');
insert into woodscrew values (
1002, 'Tommy Hardware', 'Finish', 20, 1.5, 'Phillips');
insert into woodscrew values (
1002, 'Balaji Parts, Inc.', 'Finish', 20, 1.5, 'Phillips');
insert into woodscrew values (
1003, 'Tommy Hardware', 'Finish', 20, 1, 'Phillips');

```

6 Oracle Database 10g High Availability

```
insert into woodscrew values (
1003, 'Balaji Parts, Inc.', 'Finish', 20, 1, 'Phillips');
insert into woodscrew values (
1004, 'Tommy Hardware', 'Finish', 30, 2, 'Phillips');
insert into woodscrew values (
1004, 'Balaji Parts, Inc.', 'Finish', 30, 2, 'Phillips');
insert into woodscrew values (
1005, 'Tommy Hardware', 'Finish', 20, 2, 'Phillips');
insert into woodscrew values (
1005, 'Balaji Parts, Inc.', 'Finish', 20, 2, 'Phillips');

insert into woodscrew_inventory values (
1000, 'Tommy Hardware', 200, 'NORTHEAST', 3000000, .01);
insert into woodscrew_inventory values (
1000, 'Tommy Hardware', 350, 'SOUTHWEST', 1000000, .01);
insert into woodscrew_inventory values (
1000, 'Balaji Parts, Inc.', 450, 'NORTHWEST', 1500000, .015);
insert into woodscrew_inventory values (
1005, 'Balaji Parts, Inc.', 450, 'NORTHWEST', 1700000, .017);

insert into woodscrew_orders values (
20202, '2003-09-22 00:02:02', 2001, 1000, 20000, 64114, 'NORTHEAST');
insert into woodscrew_orders values (
20203, '2003-09-22 00:02:04', 2001, 1001, 10000, 64114, 'NORTHEAST');
insert into woodscrew_orders values (
20204, '2003-09-22 00:02:06', 2002, 1002, 10000, 64114, 'NORTHWEST');
insert into woodscrew_orders values (
20205, '2003-09-22 00:02:08', 2002, 1003, 30000, 64114, 'NORTHWEST');
insert into woodscrew_orders values (
20206, '2003-10-04 00:02:12', 2002, 1004, 10000, 80903, 'SOUTHWEST');
insert into woodscrew_orders values (
20207, '2003-10-04 00:02:14', 2001, 1003, 20000, 80903, 'SOUTHWEST');
insert into woodscrew_orders values (
20208, '2003-10-04 00:02:16', 2002, 1002, 30000, 64114, 'SOUTHWEST');
insert into woodscrew_orders values (
20209, '2003-10-04 00:02:18', 2003, 1001, 40000, 90210, 'NORTHWEST');
insert into woodscrew_orders values (
20210, '2003-11-04 00:02:20', 2005, 1000, 10000, 83401, 'SOUTHEAST');
insert into woodscrew_orders values (
20211, '2003-11-04 00:02:22', 2002, 1005, 10000, 83401, 'SOUTHEAST');
insert into woodscrew_orders values (
20212, '2003-11-04 00:02:24', 2001, 1004, 10000, 64114, 'NORTHEAST');
insert into woodscrew_orders values (
20213, '2003-11-04 00:02:26', 2003, 1003, 10000, 64114, 'NORTHEAST');
insert into woodscrew_orders values (
20214, '2003-12-04 00:02:28', 2002, 1001, 20000, 64114, 'SOUTHEAST');
insert into woodscrew_orders values (
20215, '2003-12-04 00:02:30', 2001, 1000, 10000, 80903, 'NORTHWEST');
insert into woodscrew_orders values (
```

```
20216, '2003-12-04 00:02:32', 2005, 1001, 50000, 80903, 'SOUTHWEST');
insert into woodscrew_orders values (
20217, '2003-12-04 00:02:34', 2003, 1003, 70000, 90210, 'SOUTHWEST');
commit;
```

User-Defined Availability

Sometimes database availability is not defined by the database administrator. Even if the DBA can connect to the database and select against tables, users may not be having the same experience from their application.

Take the example of Horatio's Woodscrews. Due to massive transaction processing that occurs over the course of the day against the woodscrew_orders table, thousands of new rows could be added over the course of a day. When the CEO goes to check on his reports which show how many woodscrews are being sold, and to which customers, the report is creating an ad hoc SQL query against the entire table. Because of the amount of data, and the simplistic data design, these queries might not return in what the CEO envisions as a reasonable amount of time. The DBA of the company is inundated with uncomfortable calls from impatient executives wondering why the database is "down."

The CEO has a button on his desktop application, see, that says nothing more than woodscrew_orders_by_customer. He has no interest in hearing about how long-running queries against massive amounts of data might take a little longer than his expectations. He just wants the report to be available.

Test and Development Availability

Just as with CEOs demanding faster reports, sometimes availability is defined more by a human resource issue than actual data availability. At Horatio's Woodscrews, they run all their production systems on Solaris, but are investigating Linux as an alternative. The first step of the investigation is to set up the test and development shop with Linux servers, and then move a copy of production data to the new Linux servers for application testing.

Partitioning, mviews, and Index-Organized-Tables

Sometimes availability has as much to do with user perception as it does with database reality. When upper management complains, even brand new DBAs must take measures to improve performance for them. In this situation, the HA DBA will be best served by looking to reformulate some of the tables used by his boss, or by creating materialized views to subset the data appropriately. This is covered in Chapter 2, in the section "Materialized Views."

Cross-Platform Transportable Tablespaces

Availability isn't always about the production database, and an outage may simply be defined by "idle developers twiddling their thumbs." But a move of database information to a new platform has always been a huge undertaking involving the tried and true export and import utilities to logically extract data. However, this always came with huge resource issues at the export database.

Starting in Oracle Database 10g, Oracle has built in a means of using the Transportable Tablespace feature of the database and allowing it to transfer tablespaces across platforms. This is discussed in Chapter 2, in the section "Transportable Tablespaces."

The Linux servers arrived late in the week, and were configured with the Oracle software image over the weekend. On Monday, the development lead needs to get the data moved from the production Woodscrew database to the Linux servers. So the development lead contacts the senior DBA for the database and asks that they transition the data to the new Linux servers.

The dilemma for the DBA is that getting a read-consistent view of the Woodscrew tables is nearly impossible for long enough to get an export to complete. The only time there is a large enough window for such an operation would be the following weekend. The backups cannot be used, because they are backups of Solaris datafiles. But if they wait for the weekend, the development team will sit idle all week on a project that has imminent deadlines. How can they get the data moved across platforms in an expedient way?

Cyclical Database Resource Requirements

User needs often cycle over the course of time, based on business needs, such that some users will not find the database to be inaccessible until there is a massive amount of activity in a certain area, and then suddenly there are not enough resources.

Accounts Receivable, at Horatio's Woodscrews, has this problem. As the end of each month comes to a close, they have to close out the previous month's accounts and run massive reports on the status of all opening and closing accounts. This month-end processing sends the database processing needs of the AR department through the roof—but only for a week or so, before it settles back into a more routine usage pattern.

Use Services to Allocate Workloads to Higher Priority Applications

The concept of using different services for different applications in Oracle Database 10g can allow the DBA to set up differing resource limits and thresholds for different applications. Those thresholds can be modified easily using the Resource Manager, allowing different limits to apply to applications at different times. So, at the start of each month, the reporting group can get more resources than at other times during the month, the paycheck group gets month-end, and the orders group gets an annual bump in resource allocation. By combining services and real application clusters, different applications can be allowed access to a differing number of nodes, so that higher priority applications can scale up faster when needed. Services are discussed in more detail in Chapter 6.

At the same time, Human Resources typically finds its peak at the lead-up to the end of the month, as it processes employee hours, salaries, and payments. They will need the most resources, then, at a different time than the AR department.

The reporting of these two groups always affects the Order Entry group, as they have a relatively steady database resource usage pattern over the entire course of the month. However, as the summer approaches, the hardware stores begin to increase the stock of woodscrews, and so orders will steadily increase as the summer approaches, and then steadily decrease as weather cools.

What Reports Were Those, Exactly?

The problem with attempting to allocate the correct resources to the correct applications at the correct time of month or year is that often it is nearly impossible to get trended data on usage over time.

The DBA at Horatio's Woodscrews is faced with this dilemma. The different application groups have been complaining about the performance of their reports at different times, but the DBA has not been able to get accurate data on which problem reports or application batch processes are truly in need of more resources. He's been reading up on his performance tuning techniques, but the month is coming to an end again, and he can already hear his pager beginning to beep.

Use AWR and ADDM to Quickly Identify and Resolve Bottlenecks

With Oracle Database 10g, the Automatic Workload Repository (AWR) collects performance information every hour, and stores it in the Workload Repository for a week so that the HA DBA can go back and review historical information and past performance issues. In addition, baselines can be maintained indefinitely, so that the HA DBA can compare problem periods to points in the past when things were running smoothly. The Automatic Database Diagnostic Monitor, or ADDM, runs at the end of every AWR report period and proactively reports on any discovered problems, as well as recommending solutions or running additional tuning advisors. We discuss this in more detail in Chapter 3.

Out of Space in the Woodscrew Tablespace

Availability can be significantly hampered by routine database management tasks. Take something as straightforward as datafile management. Where the files reside, how big they are, and on what disks, can lead to significant availability problems.

Horatio's Woodscrew database recently ran out of disk space while autoextending a datafile in the WS_APP_DATA tablespace. They had turned autoextend on so that they would not have to be faced with disk management. But business has been booming, and the orders and customer tables are growing dramatically. In addition, new reports are being generated by upper management looking to mine the data for more edge in the marketplace. In a word, the single datafile on a single disk volume has begun to fall short of their needs. They recently had disk problems, and the system administrator is reporting excessive I/O on WS_APP_DATA datafiles. A disk burnout is imminent.

However, reorganizing the datafiles across new disk volumes not only means an explicit investigation into which tables, exactly, are getting hit the hardest, but also the outage that would be required while the tablespace is reorganized across new datafiles on new volumes. Such a massive undertaking could take days.

Use ASM to Simplify Disk and File Management

In Oracle Database 10g, automatic storage management, or ASM, greatly simplifies file management by removing the burden of laying out files to avoid hot spots, and also by simplifying file creation and sizing. ASM is a volume manager for Oracle files, which gives the ability to stripe and mirror files with very little effort on the part of the DBA or the sysadmin. In addition, ASM is constantly monitoring for I/O hot spots, and it will automatically direct reads of allocation units or segments to disks that are least heavily used, to avoid the occurrence of hot spots and maintain I/O performance. Aside from this, if a disk fails, or if a new disk is added to an ASM disk group, ASM will automatically rebalance the existing files, to maintain the distribution of I/O. ASM will be discussed in more detail in Chapter 3.

Downtime for Hardware Fixes

Database availability can be taken out of the hands of the DBA when an outage is related to a hardware problem on the database server. At Horatio's Woodscrews, they have had users complaining about losing their connection to the database intermittently. The DBA could find nothing wrong with their setup, but the system administrator ran some diagnostics that pointed to a flaky network interface card. The solution is to replace the network card as soon as possible. Doing so means taking the database down, shutting down the entire server, and replacing the card. With month-end reports running around the clock in order to complete on time, shutting down the database will mean that the reports will have to wait until after the database comes back up.

RAC Clusters Mask Problems Involving a Single Node

With Real Application Clusters, multiple nodes are accessing the database at the same time. As such, if a node fails due to hardware or operating system problems, that node can be taken offline and repaired while users are still accessing the database through the remaining nodes in the cluster. Once repaired, the node can be restarted and will rejoin the cluster automatically, making the instance again available to users. We discuss the setup and configuration of a RAC cluster in Chapter 4.

Transparent Application Failover (TAF) Allows Queries to Be Failed Over to Another Node and Restarted Automatically

With Real Application Clusters and transparent application failover, should an instance on one node crash, it is possible for users connected to that instance to automatically fail over to one of the remaining nodes in the cluster, and have queries be restarted—and then to continue on uninterrupted. This is discussed in Chapter 11.

Restarting Long-Running Transactions

In the event of some kind of instance outage, there will inevitably be transactions that will stop, and then have to be restarted manually. This can lead to significant shortcomings in time-sensitive reporting and processing.

At Horatio's Woodscrews, the developers that were sitting idle due to the Linux servers having no data decided to try and get a little work done by connecting to the production database to review some of the database structure. However, one of the developers had installed a beta version of the ODBC drivers. When this ODBC driver connected to the production database, it caused a heap corruption that resulted in an ORA-600. The ORA-600 caused the database to crash.

Just like that, all the long-running reports that had been generated by the Accounts Receivable teams began to receive the error ORA-3113 "End-of-File on Communication." The DBA's pager started to beep, and beep, and beep. He was able to restart the database, but all of the long-running transactions had to be restarted.

Slow Crash Recovery

Even after the DBA restarted the database, it seemed to be hung at the startup command. There were no errors, but when he checked the alert log, he noted that the last line indicated the database was performing instance recovery (also known as crash recovery). This recovery must be done to recover any changes in the redo logs that have not been committed to the datafiles. Depending on the size of the redo logs, and the frequency of checkpoints, this can take a considerable amount of time. When there is a lot of pressure to make a database available as quickly as possible, waiting on a poorly tuned crash recovery session can be excruciating.

RAC Handles Instance Recovery from Surviving Nodes

With Real Application Clusters, should an instance that is part of the cluster crash, the instance recovery is handled immediately by one of the surviving instances in the cluster. We discuss this in Chapter 5. In addition, Oracle10g has, through Enterprise Manager, a new feature called the Redo Log Advisor, which will allow you to automatically tune redo logs so that checkpoints occur at a rate that allows for faster instance recovery in the event of a crash. This is discussed in Chapter 3.

Dealing with Block Corruption (ORA 1578)

At Horatio's Woodscrews, the system administrators finally installed the new network card. With the system down, they took the opportunity to install new drivers for their storage area network (SAN) device. Then they contacted the DBA to let him know he could start the database.

The DBA starts the database, and begins to run through the system checks to make sure all the applications will start up correctly. Then he initiates all the reports that failed because of the shutdown. Everything seems to be going smoothly, until he notices that one of the reports has an error:

```
ORA-1578: ORACLE data block corrupted (file # 121, block # 68)
```

As the DBA began to investigate the problem, the system administrators called him back: the new SAN driver is causing problems, and he should shut down the database immediately. But the damage has already been done—database corruption.

But there is only one corrupt block, in the `WS_APP_DATA01.DBF` file. Just one block out of thousands of perfectly good blocks. With datablock corruption, the solution to the problem is to restore the entire datafile from backup, and then apply archivelogs to recover the file. Because the datafile contains parts of the `Woodscrew`, `woodscrew_orders`, and `woodscrew_inventory` tables, these objects will be offline until the file can be restored and recovered. The brief outage for the hardware fix now has been extended to include the restore of a file from backup, and then the application of the archivelogs to that file. The tablespace will not be available until recovery completes.

Recovery Manager (RMAN) Provides Media Recovery of Data Blocks

When RMAN is utilized for backups, you can use those backups to restore a single block from the last good backup, and then perform media recovery on that block (apply archive log changes, if there are any). You can do it for a single block or for a list of corrupt blocks, and the tablespace stays online while you do recovery. In fact, the corrupt table stays online, too. We discuss this in Chapter 8.

Waiting for the File to Restore from Tape

For our poor DBA at Horatio's Woodscrews, it only gets worse. He needs to restore the file `WS_APP_DATA01.DBF` from backup and then perform media recovery using archivelogs. He contacts the backup administrator and asks that the file be restored from tape.

After some shuffling around, the backup administrator finds the last backup and begins the tape restore. Because of the size of the file, this will take a few hours to get the file queued over the network from the tape jukebox to the database server. The DBA asks if there are any backups of the file on disk, knowing that there is plenty of unused space on the new SAN that could be used for disk backup of important files. The backup administrator just runs the tape servers, though, and knows nothing about the SAN. That's a different department.

Avoid Long Tape Restores with RMAN Flashback Recovery Area

Using new RMAN functionality, you can create backup jobs that always store at least the last full copy of the database on a disk location called the flashback recovery area. Then, when you take the next backup, RMAN will automatically age out the last backup, or you can set it up to move the old backups to tape from the disk area. When you go to initiate the restore from RMAN, it knows where the last best copy is. The restore can be instantaneous, as we can switch out the bad file with the new file and begin applying archivelogs immediately. We discuss this in Chapter 8.

Protect Against Node Failure with a Robust Archive Strategy

While RAC does provide a higher degree of protection against outages, you have to be careful how you configure the database so that you don't invent single points of failure. In this case, the archivelogs from Node2 are required for recovery, but Node2 is off the network temporarily. For help with archive strategies with RAC, see Chapter 5, in the section "Redo Logs and Media Recovery." Also see Chapter 8, particularly Figure 8-8.

RAC and the Single Point of Failure

The DBA for Horatio's Woodscrews is no fool. He knows that he needs to transition the data to a RAC cluster. As part of the preparation, he has configured the Sales Department's smaller database on a RAC cluster with two nodes. He set it up quickly, with minimal configuration, and it's been running smoothly for a few weeks.

However, the network configuration issues that affected the production database had also affected the Sales system. One of the nodes in the cluster was not available on the network. The DBA wasn't worried, however, because the other node was still available.

But one of the files was asking for media recovery, so the DBA issued the recovery command and it asked for an archivelog that was required for recovery. The DBA pressed ENTER to accept the default archivelog. But the media recovery session failed. He looked a little closer, and noted that the requested archivelog was on the unavailable node. Suddenly, his RAC cluster was not operational as he waited for the other node to become available so he could perform recovery.

Rewinding the Database

Perhaps the most difficult outage situations are those tricky logical errors introduced by the users themselves—when a user updates the wrong table, or updates the wrong values. These types of errors are tough to overcome because they are not perceived by the database as errors, but just another transaction. Typically, user errors do not occur in a vacuum; an erroneous update can occur alongside hundreds of correct updates. Pretty soon, the bad data is buried by thousands of additional updates. How can you fish just one of those transactions out? Can you "rewind" the database back to a previous point in time?

Use Flashback Table to Restore a Table to a Previous State

In Oracle Database 10g, Oracle introduced the ability to rewind a table to a previous state without performing point-in-time recovery. This is called Flashback Table, and it's part of the Flashback Technologies discussed in Chapter 9.

You can also use LogMiner to review transactions in the archived redo logs in order to determine where exactly the bad data was entered, as well as to retrieve the good transactions entered after the bad transaction. LogMiner is discussed in Chapter 2.

At Horatio's Woodscrews, the problem was reported by the Accounting team. As they went through their month-end processing, they began to realize that the data seemed incorrect. All the roll-up values seemed to be impossibly high, even during a good month. They could not figure out what had happened, but the data had been incorrectly entered at some point in the morning. Now, in the afternoon, they came to the DBA and asked that he "start over" all of their tables as they had looked in the morning.

The options for restoring just a few tables are limited, without rolling the entire database back to a previous point-in-time. Oracle has provided for a tablespace point-in-time recovery (TSPITR), where just a single tablespace is restored to a previous point. But that is labor-intensive, and the smallest unit that can be restored is the entire tablespace. The Accounting group does not want to redo all their work for the day, just the work in a few tables.

The Dropped Table

Like an incorrect update to the table, an inadvertently dropped table can be catastrophic. Unlike an inadvertent DML statement (insert, update, or delete), a drop cannot be explored and fixed manually. Once dropped, the table must be restored from a backup of some sort.

The DBA at Horatio's Woodscrews did this one to himself: he was trying to clean up unused objects in the production database. There were some leftover tables that had been used for testing purposes in a now-abandoned user's schema. The DBA was reviewing the objects and then dropping them to free up the extents. However, for the Woodscrew table, he put the wrong username in the DROP statement by accident, and he knew it immediately: `ws_app.woodscrew` had just been dropped.

Use Flashback Drop to Restore Dropped Objects

In Oracle Database 10g, as part of the Flashback Technologies, Oracle introduced Flashback Drop. Now, when an object is dropped, it is placed in a Recycle Bin, where it is stored until there is space pressure in the tablespace. Until it ages out, the object can be “undropped” in only a few moments. For more information, see Chapter 9.

The Truncated Table

Another deadly user error can be the use of TRUNCATE to remove rows of an incorrect table. With no undo generated, a truncate is permanent in a way that even a Flashback Transaction or LogMiner operation cannot assist with. Once truncated, the data is missing, and nothing can be done except to restore from a backup and then cancel recovery prior to the truncate.

The DBA was still trying to figure out how to restore the Woodscrew table when he was interrupted by a page. Someone in the Sales group, impatient because their cluster was down, had logged into the production database with a borrowed password. The salesperson had been trying to delete rows from a small sales table, but it was taking too long so he used TRUNCATE. But the production table he truncated had more than just rows for his region, and suddenly the woodscrew_inventory table was empty:

```
select count(*) from WS_APP.WOODSCREW_INVENTORY;
no rows selected.
```

When There Is an Unrecoverable Operation, Use Flashback Database

A TRUNCATE operation can be deadly to a database, as it is not a DML operation that gets a “before image” stored in the undo segments. A Flashback Table won’t be of any use. Typically, a truncate done in error requires a point-in-time recovery. In Oracle Database 10g, we can use the Flashback Database, which quickly rewinds the database back in time in a manner that does not require a media restore operation—so no waiting for all the files to come from tape. Our flashback can occur in minutes, instead of hours. See Chapter 9 for more information.

Use Oracle Streams to Replicate Data to Unique Databases on Different Operating Systems

If you've researched Oracle Advanced Replication in the past, you may have discovered that it provides a good way to share data among multiple, independent databases. But the performance could slow data processing to some degree. With Oracle Streams, replication has improved its speed dramatically and provides a way to integrate a heterogeneous OS environment into a shared data/high-availability model. For more on Streams, see Chapter 10.

Connecting Online, Identical Databases

The DBA at Horatio's Woodscrews finally found the time to move the production data from the Solaris production database to the Linux servers run by the Test and Development team. The development lead was very excited about the performance they were getting from the Linux boxes, and the application code ported with very little trouble.

In a few weeks, however, the Solaris box was simply overworked. New orders were coming in faster than ever, and the new warehouses were coming online in four new distribution areas. With all the new data, the database began to bog down. The Chief Information Officer approached the DBA and asked a simple question: how can we leverage the Linux servers for our production system? Can we start connecting the Order Entry group to the Linux database, for instance, and just keep the internal groups running against the Solaris system?

Complete and Total Disaster

Disaster struck Horatio's Woodscrews on a particularly wet spring day. Rain had been falling steadily for weeks in the city. The DBA was sleeping soundly, dreaming of data protection systems, when his pager began to beep. He called the number groggily, looking at the clock. It was the middle of the night.

A water main had busted near the Woodscrew building, and the basement had flooded. The basement held all of the database systems—the production Solaris box, the Sales RAC cluster, the test and dev servers, everything. The utility company was marking the area as a disaster area, and not letting anyone near the building.

All the servers were lost. The DBA had to think fast. Did they move the tape backups off-site, as was proposed a few months back? He could not remember, so he called the backup administrator. Did they have those old Solaris boxes they'd retired last year someplace? The DBA dialed the system administrator.

A conference call was put together in the middle of the night. Where were the archived tape backups? Were they moved to a different building? When was the last time the tapes were moved to the other location? How much data would be lost in the flood?

The questions flew around to the system administrator—Where could they find new systems to install Oracle? Would there be enough disk space without the SAN device? Should they contact the vendors to get emergency hardware shipped in?

Then the questions came to the DBA: Could we rebuild the data once the drives were salvaged from the flood? Could we hobble along on one of the older servers at the other office? How fast could he get Oracle installed, patched, and ready to start the file restore from the archive tapes?

And, finally, the ultimate question: Who's going to call the CEO and tell him what happened?

Where to Go from Here

The downtime scenarios that we have tried to illustrate in this chapter are but the tip of the iceberg. We wanted to show you that there are common, everyday situations that can be fixed using new functionality in Oracle Database 10g. Often, this is functionality that you already have at your fingertips and can leverage right now, with just a few steps that we outline in the following chapters.

We also wanted to show you some of the more disruptive events that can immobilize a database: user errors, such as a truncated table, or a hardware failure. These types of events happen less frequently, but the outage can be extremely long without careful planning that accounts for the possibility. From RAC to Flashback, careful planning and forethought will guide you through these disruptions.

Protect Yourself Against Disaster with Data Guard

Oracle Data Guard offers the most effective protection against complete site failure, with real-time data push from the primary database to the standby database. Data Guard allows you to activate an exact copy of the primary database in just a few minutes (or seconds), so that the outage is almost unnoticeable. It can be configured in many different ways to meet the business demands you may have. The standby database can be used as a reporting database, for taking backups of the production database, or even as a failover site during massive system reconfiguration at the primary site. For a complete rundown of the flexibility and usability of Oracle Data Guard, see Chapter 7.

20 Oracle Database 10g High Availability

Even less predictable, and less frequent, is the complete loss of a system due to natural disaster or site network outage. In situations where the entire database is lost and cannot be quickly restored, there are important questions that begin to circulate, and some of them have to do with who's at fault for the lack of a disaster plan. Oracle Data Guard offers one of the most advanced disaster-proofing solutions on the market.

So, now that you've explored just a few of the situations that might cause an outage, its time to explore the technologies that can prevent (or significantly reduce) those disruptive downtimes.