# Backup and Recovery – A Completeness Check

# Agenda

- Raisons d'être
- Types of backups
- Tools at your disposal
- Developing a strategy
- Testing a strategy
- You are what you eat

# Raison d'être

*Back up and take a good look at your backup. The secret about getting your database back is retrieving your backup and bringing your database back up.*

# The players

- A set of tested backup routines leveraging the suite of tools at your disposal
- The closer you are to the vendor software, the more efficient the solution
- Use tools parcelled with the offering
- Have a non-GUI solution

# Are you protected?

- Operator or administrator error
  - the human factor
  - unavoidable
  - stress related
- Loss of any entity
  - complete database
  - data file
  - one or more online redo logs

# Types of backups

- Logical
  - copy as of some point in time
  - vulnerable to transaction loss
  - ideal for static data
- Physical
  - point-in-time image
  - roll-forward capabilities

# Logical Backups

# Export / Import

- Thorough use of all features
- Get a clean copy of the data and *only* the data
  - must be *owner.table* driven
  - no indexes, constraints, triggers, or grants
  - less prone to error
  - uses less temp
  - partitions up rollback requirements

```
set pages 0 lines 999 trimsp on feed off ver off

col a new_value biggest
select max(table_name) a
from dba_tables
where owner = 'PRD';

spool prd_get.parfile

prompt userid=exporter/exporter
prompt triggers=n
prompt statistics=none
prompt file=prd.pipe
prompt log=prd_noicg
prompt direct=true
prompt buffer=20000000
prompt indexes=n
prompt grants=n
prompt constraints=n
prompt tables=

select owner||'.'||table_name||','
from dba_tables
where owner = 'PRD'
and table_name <> '&biggest';

select 'PRD.'||'&biggest'
from dual;
```

- *new_value* takes care of no trailing comma at end of list
- one schema per export run
- exclude *sys* and *system*
- exp_full_database ~~DBA~~
- restrict file sizes to 2Gb
- *gzip* vs. *filesize* trade-offs
- the "direct" conundrum

# Export / Import

- No rows export of the full database
  - use conventional path
  - name appropriately for easy identification
  - self descripting means faster retrieval
- Test the integrity of *all* your export files
- Mismatch in the UNIX environment for NLS_LANG when using direct path

# NLS_LANG

*Fatal on export / informational on import*

```
export NLS_LANG=language_country.charset

export NLS_LANG=american_america.we8iso8859p1

export NLS_LANG=french_france.we8dec

export NLS_LANG=spanish_spain.utf8
```

```
rm prd.pipe > /dev/null 2>&1
mkfifo prd.pipe
gzip < prd.pipe > prd_noicg.dmp.gz &
sleep 2
exp parfile=prd_get.parfile
```

```
userid=exporter/exporter
file=full_APPA_nor
log=full_APP_nor
rows=n
statistics=n
```

```
userid=exporter/exporter
file={prd.pipe|rty.pipe|app.pipe}
indexfile=prd.sql|rty.sql|app.sql
log=prd|rty|app
```

- start each time with a clean pipe
- sleep 2 seconds assists synchronization
- save stats using *dbms_stats.export_schema_stats*
- reset stats using *dbms_stats.import_schema_stats*
- use smart names and match items in the same run

# DBMS_METADATA

- Text output of the DDL to create non-dictionary objects
- Formatting not so pretty
- Take the time to experiment
- Minimal to no system load
- Need certain database privileges to run against others' objects

# What DBMS_METADATA does not do

# DBMS_METADATA

- Seems needlessly cryptic (that's a first!)
- Drawbacks are easily rectified by further experimentation
- Various "cute" options easily adapted to style conventions (e.g. in-line constraint definitions vs. *alter table* syntax)

The PYTHIAN GROUP™

# Oracle Data Pump

- Export/import on steroids
- Just about everything you have ever dreamed of doing with predecessors
- Oracle brags it's 15-45 times faster than export
- Experience has shown more!!
- Somewhat cryptic to get going (*directories*)

# Oracle Data Pump

- Plugs a few holes of export and import
- Wrap yourself around it from day 1
- Can export data, metadata, or both
- More fine-grained inclusion using INCLUDE and EXCLUDE filters
- Export import scripts require little modifications

The PYTHIAN GROUP™

# Logical Backups Summary

- Export
  - Full database no grants / indexes / constraints / triggers
  - Full database no rows
- Import
  - Test the export file with INDEXFILE
- DBMS_METADATA (9 and 10)
  - Run through *every schema*
  - Format the output so it is useful

- Oracle Data Pump
  - Fine-grained set of data and data definitions
  - Do the pre-requisites now
  - Isolate stored object definitions (some of the most pesky to rebuild and transport)
- *alter database backup controlfile to trace;*
  - edit output for *create database* statement

- O/S files
  - listener.ora
  - tnsnames.ora
  - complete contents of
    - $ORACLE_HOME/dbs
    - …/admin/create
    - backup work directories
  - /etc/oratab
  - software owner's $HOME
  - monitoring scripts

# Physical Backups

# The players

- Traditional hot backups
- rman 
- Cold backups
- Physical standby

# Traditional hot backups

- Still the mainstay of many many installations
- Nay's
  - tablespaces in and out of backup mode
  - no native compression
  - privilege rich account
- Yay's
  - in place replacement of database

# Recovery Manager

- Yay's
  - lower level interaction with database software
  - persistent configuration
  - *spfile* inclusion
  - true block level incremental
  - integration with tape vendors EMC legato VERITAS Tivoli. software
  - native compression (even better in 10*g*)
  - huge player in flashback database
  - state of the art (therefore popular with support!)
  - 100% inside the Oracle umbrella

The PYTHIAN GROUP™

# rman

- Nay's
  - no in place switch to production
  - hmmmmmmmmmmm ...
  - adds complexity to upgrades

# Reduced I/O and CPU cycles

# Know your tools

- Backing up the catalog instance
  - standby database
  - full database export in the quiet times
  - nocatalog incremental level 0's
- Understand your levels
  - backs up what has changed since peer or lower level
  - 0 through 4

# Know your tools

- As of 8.1.7.4
  - production backup can be run on the standby server
  - do not register standby database in catalog
  - manual resync catalog after backup
- Standby built nomounted
- Recovery performed mounted

# Close the loop often

- Validate a database backup on a regular basis
  - restore database validate
  - takes just about as long as the real thing
  - no I/O but some CPU
- Build meaningful backup set piece names
  - %d %U
  - imbed the system date and level number

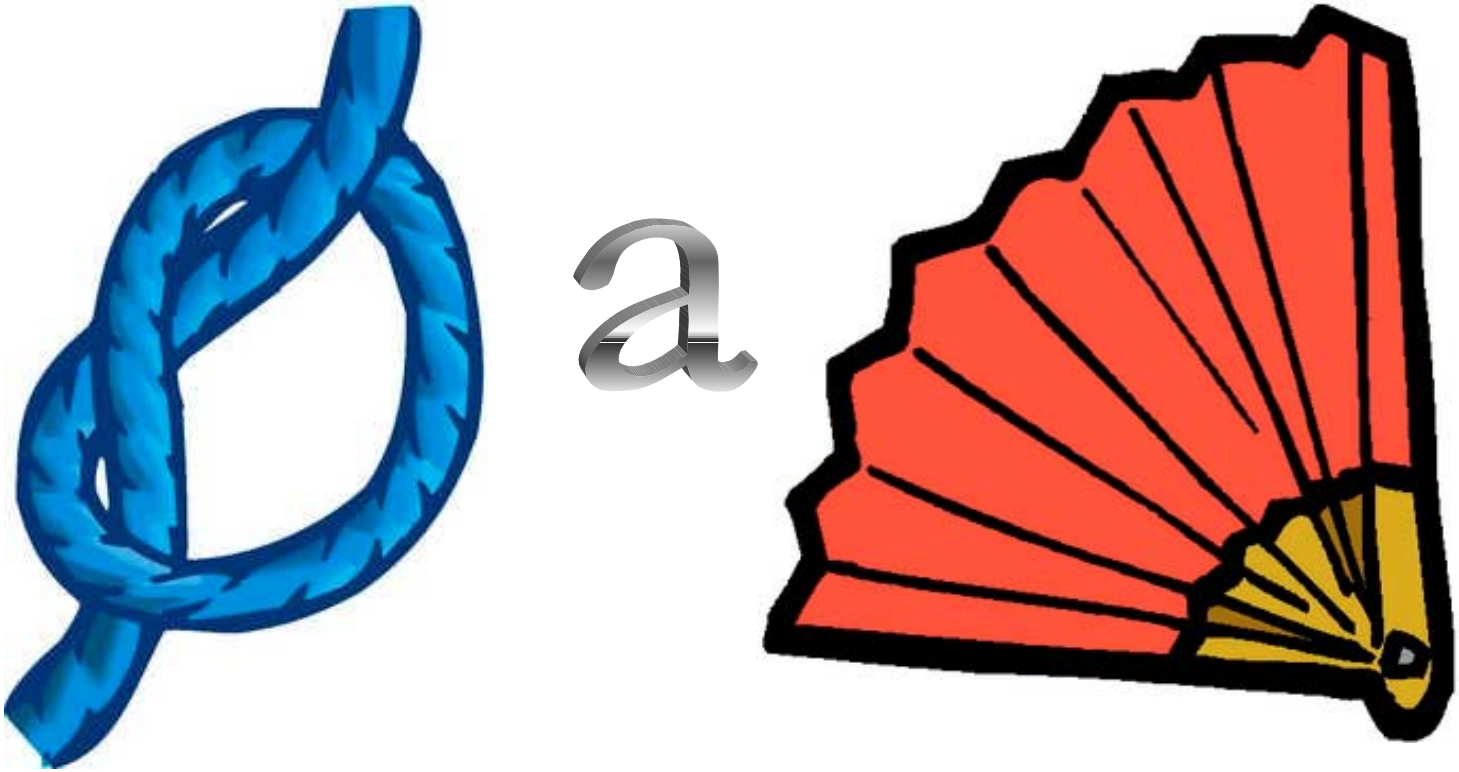
# Maximize throughput

- Keep channel size under 2Gb
  - not a concern when writing to tape
  - 10*g* offers better native compression
- Catalog cleanup
  - determined by your disk/tape retention
  - crosscheck backup completed before 'sysdate – {retention_period}'
  - delete expired backup

# Cold backups

- The whole kit and kaboodle
- Easiest for recovery
- Re-prime the caches when instance re-started
- No more safe than a properly tested and implemented hot backup

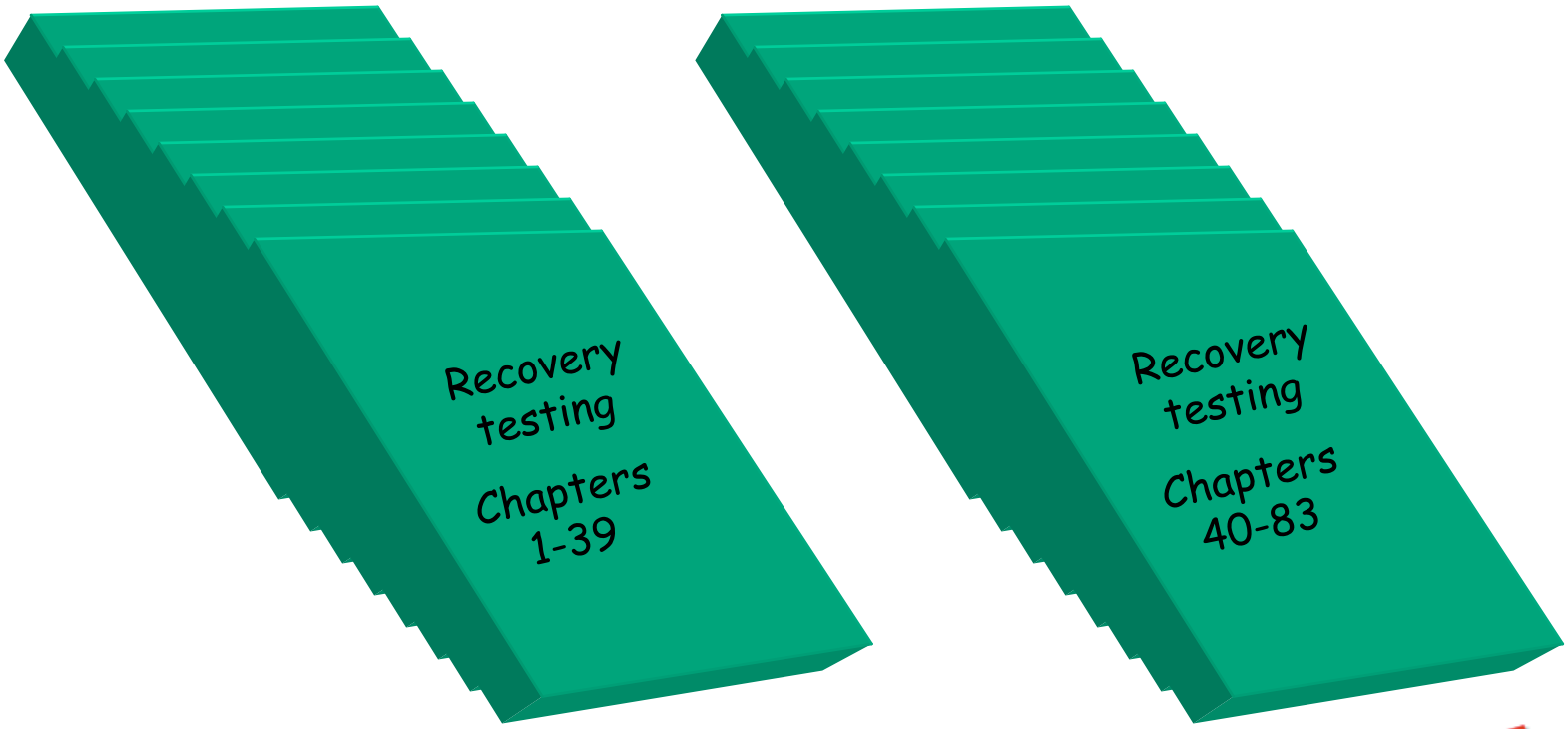
# Cold backups



# Convince your SA

- Suite of recovery exercises with
  - step-by-step outline
  - code
  - log files
- Dispel old spouse tales
  - you will not have problems with the SYSTEM tablespace

# Convince your manager

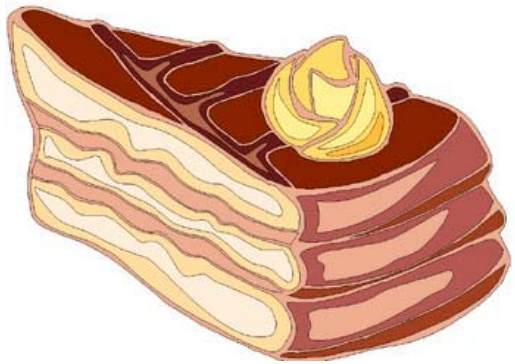


# Physical standby

- An exact replica of production
- One-to-many
- Only difference from production is the *standby controlfile*
- Need a separate licence
- Same O/S and version of Oracle
  - Linux <> Solaris

# The players

- Master
  - log transport services
  - log switching to reduce redo size over network
  - cleanup
- Standby
  - log application services
  - cleanup



# Logical standby

- Portions of production are replicated on standby
- Standby is open and in use
- Mine archived redo logs for selective SQL statements
- LogMiner with a dictionary
  - DDL done properly as of 9*i*

# Developing a strategy

- What are you comfortable with
- How long does each backup type take
- Do you have a tape drive
- Can you write directly to a tape drive
- Is there a window of opportunity
- Is there any downtime

# Developing a strategy

- Glean from experiences of colleagues
- Is the solution in subsequent releases of the product {8 / 9 / 10}
- Means available for testing
- Time available for testing
- Mock disaster recovery
- Run through possible scenarios

# Testing a strategy

- Once backup component is chosen
  - runtime
  - size of outputs
- Rebuild whole or part of a database using the chosen tool and its output(s)
- Fake different types of failurs
- Surf MetaLink for ideas

# Testing a strategy

- Don't you be the single point of failure
  - educate your colleagues
  - learn from them
  - mentor one another
- Why wait for spring, do it now
- Do not be a newbie when disaster strikes
- Document document document

# Proof of concept

- Starting with the basics … a systematic test of potential situations
- Do not cut any coreners … it's only your production data!
- Arm yourself with the tools / knowledge to acclerate disaster recovery
- Help educate one another

# A completeness check

- Object level recovery
- Media recovery with minimal data loss
- Infrastructure mechanism to accelerate getting back up
- Fluent with the create database process outside of the *dbca* (what if no GUI available)
- Protection not offered by RAID