Infrastructure at your Service.

Oracle 12c Flashback Techniques

About me

William Sescu

Consultant

+41 78 674 12 90

william.sescu@dbi-services.com

ORACLE®

Certified Master

Oracle Database 12*c*
Maximum Availability
Architecture

Who we are
dbi services

Experts At Your Service

- Over 50 specialists in IT infrastructure
- Certified, experienced, passionate

Based In Switzerland

- 100% self-financed Swiss company
- Over CHF6 mio. turnover

Leading In Infrastructure Services

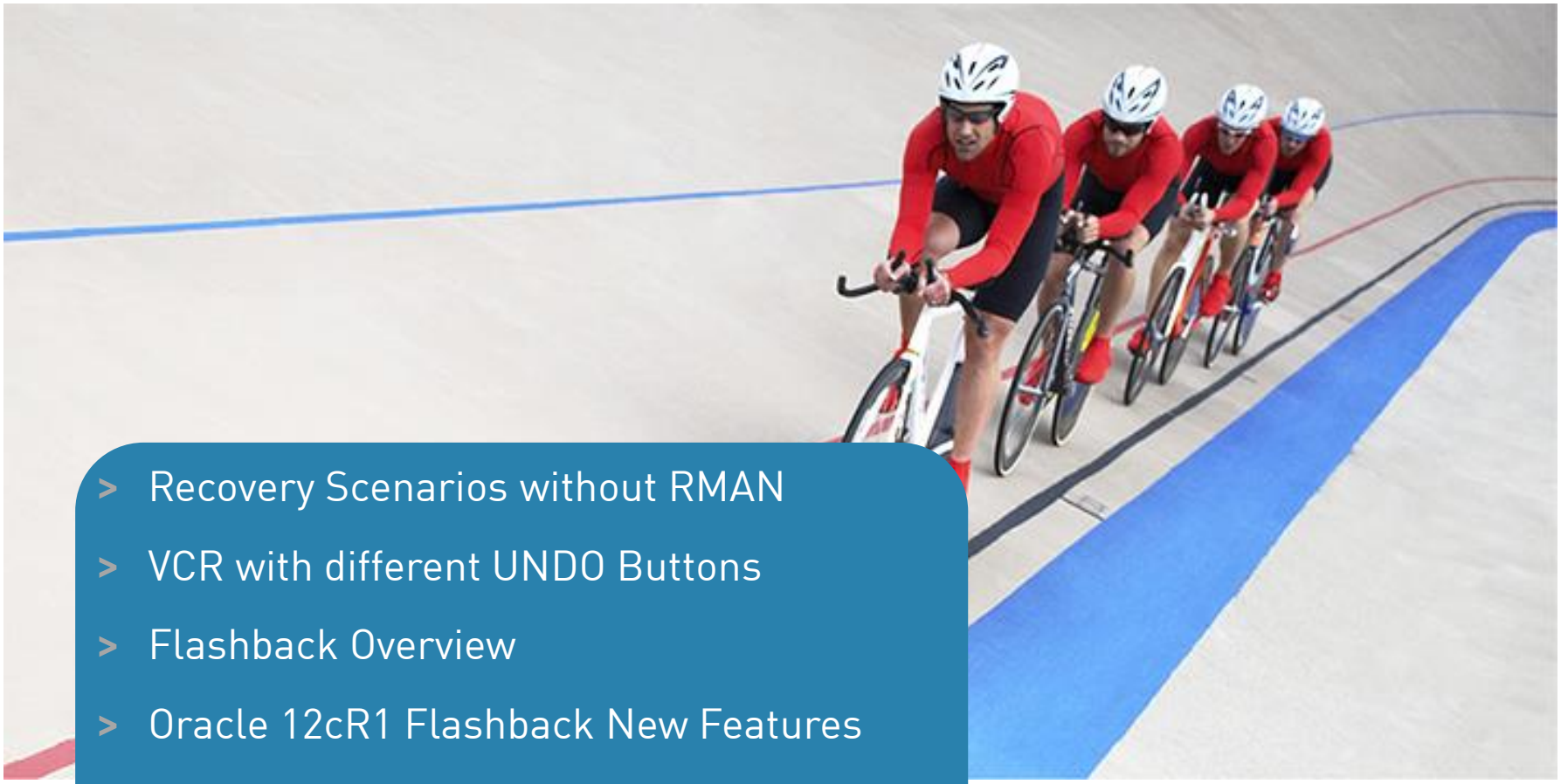
- More than 120 customers in CH, D, & F
- Over 40 SLAs dbi FlexService contracted

Certified Management System SQS ISO/IEC 20000-1

Agenda

1. Introduction
2. Undo Management
3. Flashback Query
4. Core Messages

Introduction



- Recovery Scenarios without RMAN
- VCR with different UNDO Buttons
- Flashback Overview
- Oracle 12cR1 Flashback New Features
- Oracle 12cR1 Flashback Restrictions

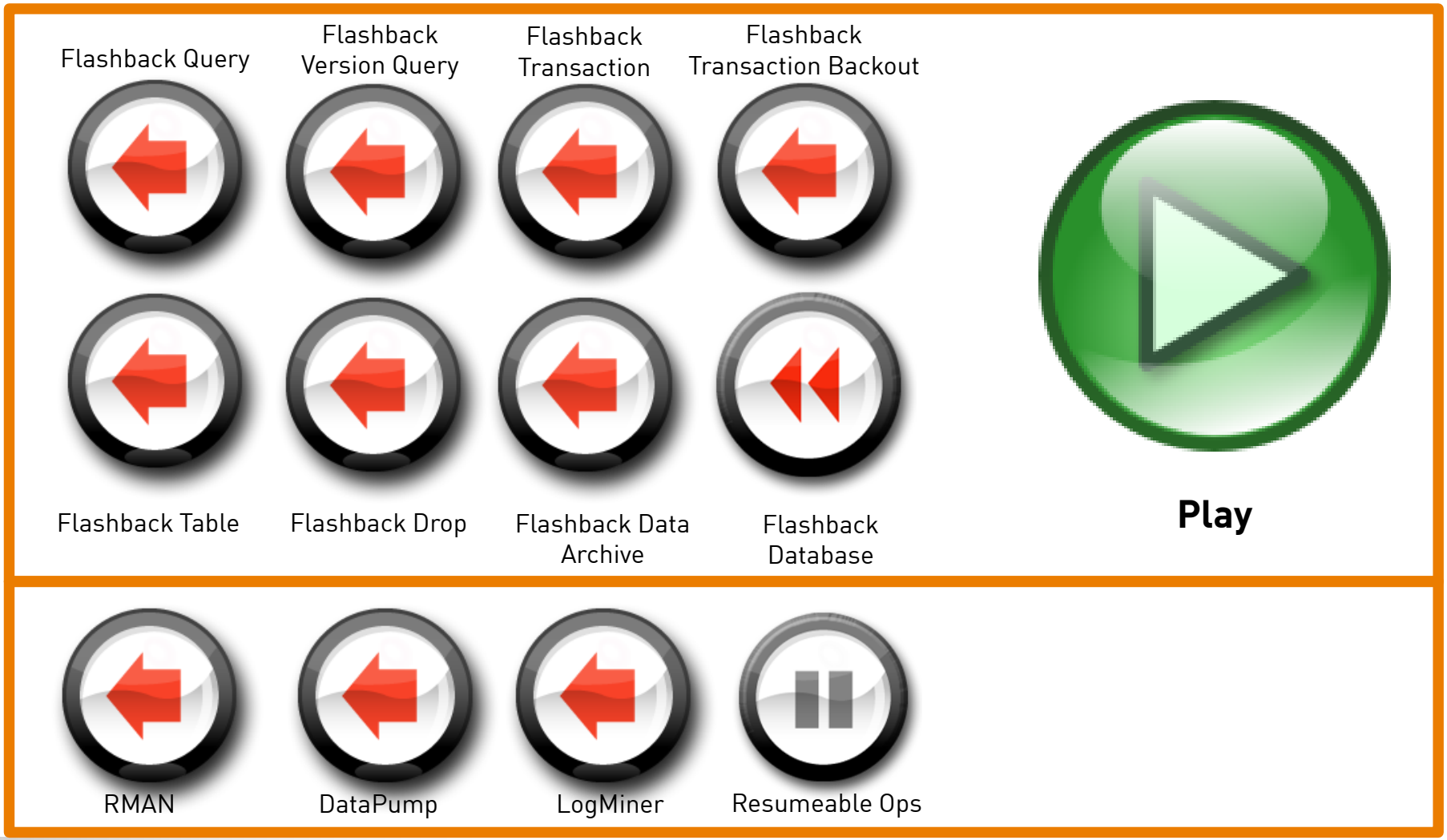
Recovery Scenarios without RMAN

Possible recovery scenarios of logical corruptions or human mistakes

- Recover committed transactions
- Perform queries that return past data
- Perform queries that return metadata that shows a detailed history of changes
- Recover tables or rows to a previous point in time
- Automatically track and archive transactional data changes for 10 years
- Roll back a transaction and its dependent transactions while the database remains online
- Undo a drop table or truncate table command
  - HINT: Recovering from a truncate table is more complicated
- Roll back the database to a time in the past
- Recover a transaction that is running out of space

VCR with different Undo Buttons

Different Flashback Techniques do exist



Flashback Overview

Different Flashback Techniques use different technologies

Technology	Internal	License
Flashback Query	Undo	SE or EE
Flashback Version Query	Undo	SE or EE
Flashback Transaction Query	Undo	EE
Flashback Transaction Backout	Undo	EE
Flashback Data Archive	Undo + FDA + 	EE + Advanced Compression
Flashback Table	Undo + Row Movement	EE
Flashback Drop	Recyclebin	SE or EE
Flashback Database	FB Logs + 	EE

- Before July 2012 Total Recall was a separate Enterprise Feature
  - Starting with July 2012 Total Recall is part of the Advanced Compression Option

Oracle 12cR1 Flashback New Features

Not to many new features regarding Flashback

Temporal Validity Support

- DBMS_FLASHBACK_ARCHIVE.enable_at_valid_time procedure can be used to specify an option for the visibility of table data

Flashback Data Archive in combination with HCC

Limited Flashback Data Archive support for the Oracle SE

- For more details, look into the "Feature Availability by Edition"
- http://docs.oracle.com/database/121/DBLIC/editions.htm#DBLIC109

Flashback Features inside a pluggable database e.g.

- Flashback Query
- Flashback Drop

Oracle 12cR1 Flashback Restrictions

Multitenant Database Restrictions for Oracle Flashback Technology

Flashback Data Archive (FDA) is not supported in a CDB (12.1.0.1)

- For Oracle Database (12.1.0.2), this restriction is removed

Flashback Transaction Query is not supported in a CDB

Flashback Transaction Backout is not supported in a CDB

Flashback Database can not be used to rollback a PDB

Undo Management

- Prerequisites
- Configure your DB for AUM
- Automatic Tuning of AUM
- Important Questions regarding AUM
- Answers regarding AUM
- 12c New Features

Undo Management
Prerequisites

Flashback needs some prerequisites before you can start

Most Flashback features uses Automatic Undo Management (AUM)

- To obtain metadata and historical data for transactions
- e.g. if a user updates the salary from 1200 to 1300, Oracle stores the value 1200 in the undo data

Undo Data is persistent and survives database shutdown

Beside the flashback technologies Undo Data is used

- for rolling back active transactions
- to recover terminated transactions
- to provide read consistency for SQL queries

Undo Management
Configure your DB for AUM

Setup your DB correctly to benefit from the Flashback techniques

Create an Undo Tablespace

- with enough space to keep the required data for flashback operations
- The more updates, and the higher the retention the more space is required

Database initialization parameter

```
UNDO_MANAGEMENT  = AUTO (default AUTO)
RAC1.UNDO_TABLESPACE=UNDOTBS1,RAC2.UNDO_TABLESPACE=UNDOTBS2
UNDO_RETENTION = 3600 (default 900 seconds, max 2*31 -1 seconds)
```

- 3600 = 1h, 2*31-1 = 2147483647 sec. ~ 68 years
- UNDO_RETENTION is a low threshold value of undo retention
- UNDO_RETENTION has different behaviors depending on the creation of the undo tablespace

Undo Management
Configure your DB for AUM

Setup your DB correctly to benefit from the Flashback techniques

More Database initialization parameter

- Max Undo Retention

```
_highthreshold_undoretention = 4294967294 (default)
```

- Min Undo Retention

```
_minimum_db_flashback_retention = 60 (default)
```

- Automatic Auto Tuning

```
_undo_autotune  = TRUE (default)
```

- This underscore parameter is responsible for automatic tuning of the undo retention and it is turned on per default

Automatic Tuning of AUM
Autoextend of Fixed Size Undo TS plays a big role

Automatic Undo Tuning depends on UNDO tablespace configuration

- It behaves different if it is a fixed sized or auto extendable tablespace

AUTOEXTEND Undo Tablespace

- The system retains undo for at least the time specified in the parameter UNDO_RETENTION, and automatically tunes the undo retention period to satisfy the undo requirements of the queries

Fixed Sized Undo Tablespace

- The system automatically tunes **for the maximum possible undo retention period**
- Based on undo tablespace size and usage history, and ignores UNDO_RETENTION unless retention guarantee is enabled

Automatic Tuning of AUM

Autoextend of Fixed Size Undo TS plays a big role

- HINT: Automatic tuning of undo retention is not supported for LOBs. The RETENTION value for LOB columns is set to the value of the UNDO_RETENTION parameter

- HINT: Setup all Undo tablespaces the same way for all RAC instances to avoid different behavior

Undo Management
Automatic Tuning of AUM

A third algorithm comes into play when dealing with AUM

- When below underscore parameter is configured

```
_smu_debug_mode=33554432
```

With this setting

- V$UNDOSTAT.TUNED_UNDORETENTION is not calculated based on a percentage of the fixed size undo tablespace
- Instead it is set to
  - the maximum of (MAXQUERYLEN secs + 300) + UNDO_RETENTION

Automatic Tuning of AUM
Famous ORA-01555

Honoring UNDO_RETENTION

- Only possible if the current undo tablespace has enough space

In case you run out of space for an active transaction

- The system starts reusing unexpired undo space
- This action can potentially cause some queries to fail with the famous ORA-01555 "snapshot too old" message

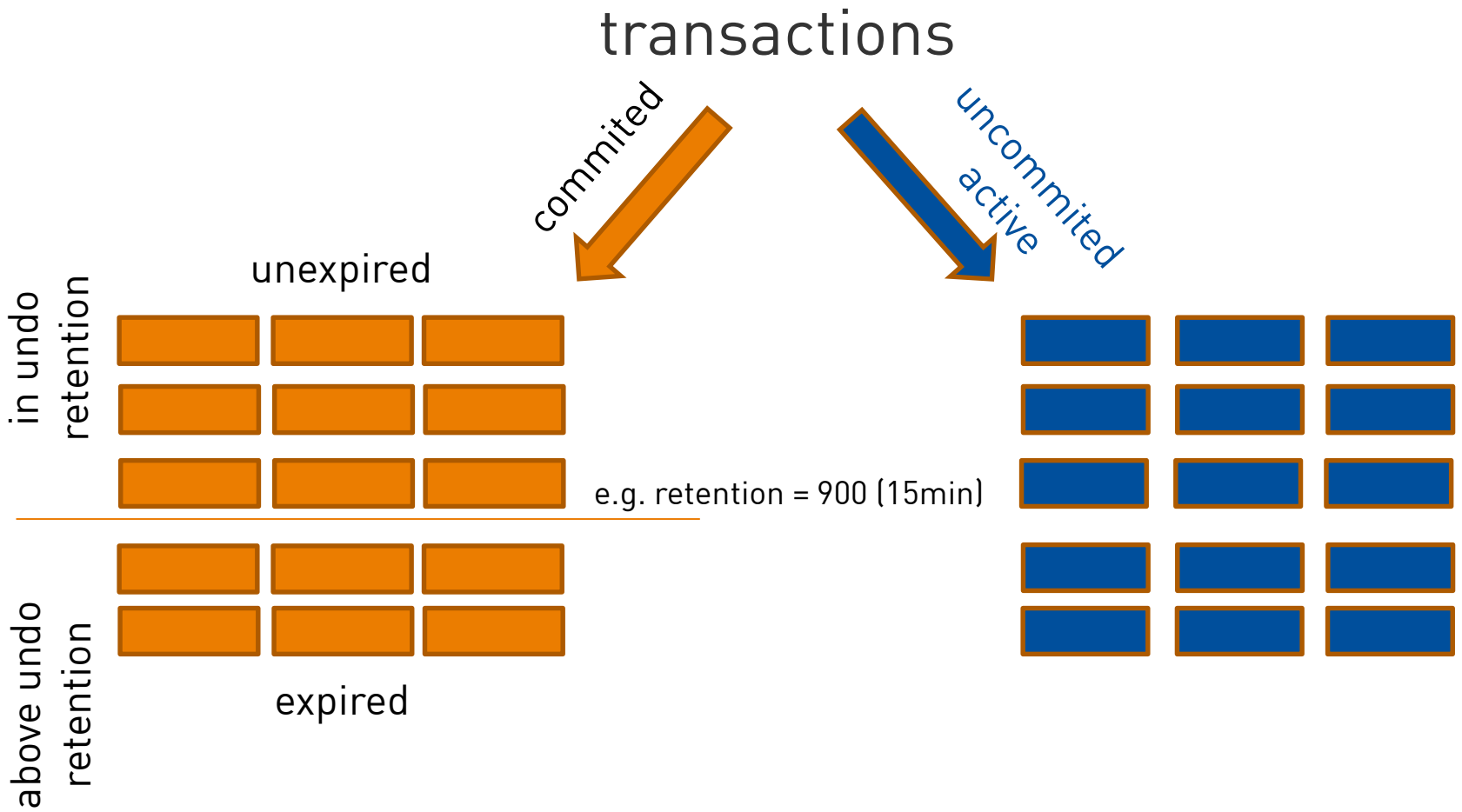
Undo Management

Important questions regarding AUM

- What is the difference between expired and unexpired extents?
- What is the amount of time for which undo is retained?
- Can I guarantee that unexpired undo data is not discarded?
- Can tuned undo retention be less then undo retention configured in the spfile(init.ora)?
- Can I really use flashback features after an instance crash?
- Can I use flashback query on both instances of a RAC cluster?
- Can I use flashback query with DataPump?
- Can I use flashback query with DB Links?
- Can I use flashback query with a join over several tables?
- How long was the longest query, and how big should I make my Undo Tablespace?
- Can I configure User Quotas on the UNDO tablespace?

Answers regarding AUM

What is the difference between expired and unexpired undo extents?



Answers regarding AUM

What is the amount of time for which undo is retained?

Check the v$undostat or gv$undostat view

- e.g. list the tuned undo retention for the last day

```
SELECT BEGIN_TIME, END_TIME, TUNED_UNDORETENTION
FROM v$undostat
WHERE end_time < sysdate -1
ORDER BY end_time;

BEGIN_TIME          END_TIME            TUNED_UNDORETENTION
------------------  ------------------  -------------------
2013:03:19 10:05:54 2013:03:19 10:15:54              129945
2013:03:19 10:15:54 2013:03:19 10:25:54              128411
2013:03:19 10:25:54 2013:03:19 10:35:54              129001
```

- The 10 minute interval is controlled by the parameter

```
_smon_undo_seg_rescan_limit = 10
```

- The v$undostat view contains a total of 576 rows, spanning a 4 day cycle, 576rows*10min/60min/24h = 4 days

Answers regarding AUM

Can I guarantee that unexpired undo data is not discarded?

Yes, by switching to retention guarantee (online operation)

```
select TABLESPACE_NAME, RETENTION from dba_tablespaces
where tablespace_name = 'UNDOTBS1'; -- noguarantee is the the default

TABLESPACE_NAME                RETENTION
------------------------------ -----------
UNDOTBS1                       NOGUARANTEE
```

```
alter tablespace undotbs1 retention guarantee;
```

```
select TABLESPACE_NAME, RETENTION from dba_tablespaces
where tablespace_name = 'UNDOTBS1';

TABLESPACE_NAME                RETENTION
------------------------------ -----------
UNDOTBS1                       GUARANTEE
```

Answers regarding AUM

Can the tuned undo retention be less then undo retention in the spfile?

Can tuned undo retention be less then undo retention in the spfile?

- Yes. UNDO_RETENTION specified in the spfile(init.ora) is a low threshold. Undo retention is automatically tuned to reduce the chances of the famous "snapshot too old" errors during long-running queries

- In the event of any undo space constraints, the system will prioritize DML operations over undo retention. In such situations, the low threshold may not be achieved and tuned_undoretention can go below undo_retention

Answers regarding AUM

Can I really use flashback features after an instance crash?

Like most of the time when you work with Oracle, it depends ☺

```
_in_memory_undo = true (default)
```

- Might have some side effects

If IMU is set to true

- Flashback might not possible after a system crash, or ever worse, it might give you wrong results

If IMU is set to false, Flashback is possible after a system crash

HINT: Oracle does not use IMU for its own dictionary. ;-)

```
_recursive_imu_transactions  =  FALSE
```

Answers regarding AUM

Can I use flashback query on both instances of a RAC cluster?

Yes. Flashback query can be used database wide

In case of RAC

- Every RAC instance has its own undo tablespace
- Read only access to other undo tablespaces is possible
  - and in fact, it is needed to enforce cluster wide consistent reads

Sometimes, RAC instances can write to other RAC instances

- E.g. during transaction recovery
- However, only if that undo tablespace is not currently used by another instance for undo generation or transaction recovery

Answers regarding AUM

Can I use flashback query with DB links?

Yes, as long as the target database is configured correctly

```
SELECT last_name FROM employees@hrlink AS OF SCN 1329996
WHERE last_name = 'Smith';
```

Answers regarding AUM

Can I use flashback query with a join over several tables?

Yes, you can specify the flashback SCN individually per table

- or work with an inline view
- or selecting from a view directly with ‚AS OF SCN' e.g.

```
SELECT e.ename, e.job, d.dname, d.loc
FROM emp e, dept AS OF SCN 693181 d              -- 2 different scn's
WHERE e.deptno = d.deptno AND d.dname = 'SALES';
```

```
SELECT e.ename, e.job, d.dname, d.loc FROM emp AS OF SCN 693143 e,
dept AS OF SCN 693181 d -- 2 different scn's
WHERE e.deptno = d.deptno AND d.dname = 'SALES';
```

```
SELECT * FROM (SELECT e.ename, e.job, d.dname, d.loc
               FROM emp e, dept d
WHERE e.deptno = d.deptno AND d.dname = 'SALES') AS OF SCN 693181;
ENAME      JOB       DNAME          LOC
---------- --------- -------------- -------------
ALLEN      SALESMAN  SALES          CHICAGO
WARD       SALESMAN  SALES          CHICAGO
```

Answers regarding AUM

Can I use flashback query with DataPump?

Yes, you can do it by using the following options

- Get the SCN First

```
SQL> select dbms_flashback.get_system_change_number from dual;
532007133643
```

FLASHBACK_SCN (specifying the SCN directly)

```
$ expdp scott/tiger file=scott.dmp FLASHBACK_SCN=532007133643
```

FLASHBACK_TIME (specifying a time, mapped to the most closely matched SCN)

```
$ expdp scott/tiger file=scott.dmp FLASHBACK_TIME='2013:03:20 13:40:30'
$ expdp scott/tiger file=scott.dmp FLASHBACK_TIME="TO_TIMESTAMP('19-06-
2004 13:24:26', 'DD-MM-YYYY HH24:MI:SS')"
```

HINT: FLASHBACK_SCN and FLASHBACK_TIME are mutually exclusive

HINT: DataPump expdp uses the DBMS_FLASHBACK package behind the scenes, so the user needs the execute privilege to use this feature

Answers regarding AUM

How long was the longest query, and how big should I make my Undo TS?

The Undo Advisor might give you the answers DBMS_UNDO_ADV

```
SQL> SELECT dbms_undo_adv.longest_query(SYSDATE-1, SYSDATE) FROM dual;
DBMS_UNDO_ADV.LONGEST_QUERY(SYSDATE-1,SYSDATE)
----------------------------------------------
                                         22737
```

```
SQL> SELECT dbms_undo_adv.required_undo_size(43200, SYSDATE-1, SYSDATE)
FROM dual;

DBMS_UNDO_ADV.REQUIRED_UNDO_SIZE(43200,SYSDATE-1,SYSDATE)
----------------------------------------------------------
                                                     50555
```

Answers regarding AUM

How long was the longest query, and how big should I make my Undo TS?

Even better is to list the current size, and the required size for your Undo Retention at the same time ☺

```
SQL> SELECT SUM(BYTES)/1024/1024                          CURRENT_MB_SIZE,
            DBMS_UNDO_ADV.REQUIRED_UNDO_SIZE(3600)   REQUIRED_MB_SIZE
       FROM dba_data_files
      WHERE tablespace_name =
            (SELECT value FROM v$parameter
            WHERE name = 'undo_tablespace');

CURRENT_MB_SIZE REQUIRED_MB_SIZE
--------------- ----------------
         122880            17947
```

Answers regarding AUM

Can I configure User Quotas on the UNDO tablespace?

No, not directly

- But you can use the Resource Manager to establish user quotas for the undo space
- It is the Resource Manager directive UNDO_POOL, which allows DBAs to limit the amount of undo space consumed by a group of users (resource consumer group)

12c New Feature

Temporary Undo

Temporary Undo allows you to

- Store undo records for TEMPORARY TABLES in the TEMP tablespace

Activated via the TEMP_UNDO_ENABLED initialization parameter

This feature was mainly designed

- for doing (DML) operations on TEMPORARY TABLES in a physical standby database with the Oracle Active Data Guard
- Temporary undo is enabled by default for a physical standby database with the Oracle Active Data Guard option
- Temporary undo can be enabled only if the compatibility level of the database is 12.0.0 or higher

Conclusion

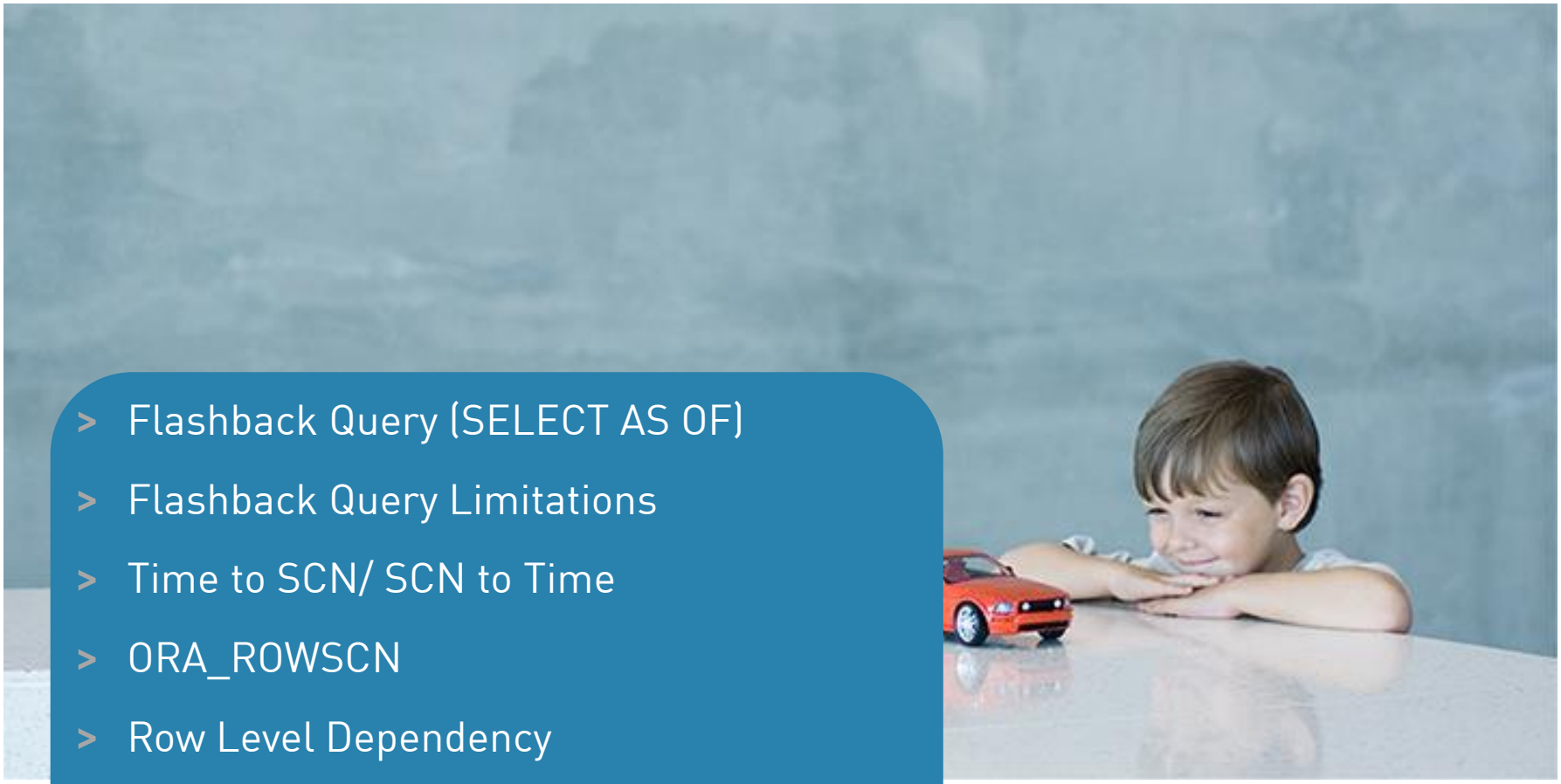
Do and Don't's

UNDO Management is a very important part

You should know how it works before you dive deep into the Flashback technologies

- Take care of the _in_memory_undo and adjust it, if you want to rely on the Flashback technologies, even after a crash
- Make sure that your undo tablespace is big enough to satisfy your retention needs
- In case of RAC, or RAC One Node, make sure that the UNDO tablespaces are of the same size and the same type (autoextend or fixed) on all nodes, if not, you might experience different behavior after a switch/failover
- Use the DBMS_UNDO_ADV to answer your questions (its Free ☺)

Flashback Query



- Flashback Query (SELECT AS OF)
- Flashback Query Limitations
- Time to SCN/ SCN to Time
- ORA_ROWSCN
- Row Level Dependency
- Flashback Query Performance

Flashback Query (SELECT AS OF)

Use Cases of Oracle Flashback Query

Recovering lost data or undoing incorrect, committed changes

Comparing current data with the corresponding data at earlier time

Checking for the state of transactional data at a particular time

Simplifying application design by removing the need to store some kinds of temporal data

Applying packaged applications, such as report generation tools, to past data

Providing self-service error correction for an application, thereby enabling users to undo and correct their errors

Flashback Query (SELECT AS OF)

Example of Examining and Restoring Past Data

Suppose that you discover at 12:30PM that the row for employee „Scott“ was deleted

- However, you know that at 09:30AM the data for „Scott“ was correct

Retrieve a Lost Row

```
SELECT * FROM employees
AS OF TIMESTAMP  -- AS OF SCN would also be possible
TO_TIMESTAMP('2013:03:20 09:30:00', 'YYYY:MM:DD HH24:MI:SS')
WHERE fist_name = 'Scott';
```

Restore a Lost Row

```
INSERT INTO employees (
 SELECT * FROM employees
 AS OF TIMESTAMP -- AS OF SCN would also be possible
 TO_TIMESTAMP('2013:03:20 09:30:00', 'YYYY:MM:DD HH24:MI:SS')
 WHERE fist_name = 'Scott');
```

Flashback Query (SELECT AS OF)

Guidelines for Oracle Flashback Query

- You can use the AS OF clause in queries to perform DML statements such as INSERT and DELETE in the same session as Oracle Flashback Query
- You can create a view that refers to past data by using the AS OF clause in the SELECT statement that defines the view
- If a possible 3-second error (maximum) is important to Flashback Query in your application, use an SCN instead of a time stamp
- You can use the AS OF clause in self joins, or in SET operations such as INTERSECT or MINUS, to extract or compare data from two different times

Flashback Query (SELECT AS OF)

More Flashback Query Examples

Creating a VIEW that shows data how it were 1h ago

```
CREATE VIEW hour_ago AS
 SELECT * FROM employees
    AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '60' MINUTES);
```

Self Join and SET Operations

- This query reinserts into table employees the rows that existed an hour ago

```
INSERT INTO employees (
 SELECT * FROM employees
 AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '60' MINUTES)
MINUS SELECT * FROM employees;
```

SYSTIMESTAMP refers to the time zone of the database host environment

Flashback Query (SELECT AS OF)

Session Level Flashback (with timestamp)

You want to observe data from several queries at a point in time using PL/SQL

Requires the execute privilege on dbms_flashback

```
SQL> GRANT execute on dbms_flashback to SCOTT;
```

```
SQL> exec dbms_flashback.enable_at_time('20-MAR-2013 11:00:00AM');
SQL> SELECT … FROM …;
SQL> SELECT … FROM …; -- No DML or DDL allowed
SQL> SELECT … FROM …;
SQL> exec dbms_flashback.disable;
SQL> -- exec DBMS_FLASHBACK.ENABLE_AT_TIME(sysdate - 5/1440); -- 5 min ago
```

If only a date is specified, time element defaults to 12 midnight

Useful when using 3th party apps and you cannot touch the code

- Simply put the session back in time

Flashback Query Limitations

What you can do and what not?

SYS can use SQL but not session level flashback

Flashback is enabled at nearest SCN to the specified time rounded down to a 3 second interval

- SCNs are mapped to times only once every *3 seconds*
- Need to know SCNs to flashback more accurately

Alterations to tables (and indexes) since flashback time will cause errors

- Avoid flashing back to a time close to a DDL statement, because Oracle might choose a SCN earlier than the DDL statement

Correspondence of SCNs to timestamps are shown in rows generated at 5 minute intervals in the SMON_SCN_TIME table

- But the RAW column, tim_scn_map, holds up to 100 mappings within its raw data
- This allows fine-grained 3 second mapping
- SMON_SCN_TIME holds data to a maximum of 5 days after instance startup
- There are several  related to the SMON_SCN_TIME table

Finding Time to SCN/ SCN to Time
You have different options to map time to scn

Use the SMON_SCN_TIME table or the functions

- First lets get the current SCN

```
SELECT CURRENT_SCN FROM V$DATABASE;
SELECT DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER FROM DUAL;
```

- SCN_TO_TIMESTAMP converts the SCN to a Timestamp with +/- 3 seconds

```
select scn_to_timestamp(532037684777) as timestamp from dual;
TIMESTAMP
---------------------------------
21-MAR-13 10.44.27.000000000 AM
```

- TIMESTAMP_TO_SCN converts the Timestamp to SCN with +/- 3 seconds

```
select timestamp_to_scn(to_timestamp('2013:03:21 10:44:27','YYYY:MM:DD
HH24:MI:SS')) as scn from dual;
                    SCN
-----------------------
           532037684409
```

Finding Time to SCN/ SCN to Time

SCN in the MOUNT state

Be aware that TIMESTAMP_TO_SCN or SCN_TO_TIMESTAMP do not work

- When the database is in the MOUNTED state (SCN will always be zero)
- This is not a bug, this is the expected behavior

```
SQL> SELECT open_mode FROM v$database;
OPEN_MODE
--------------------
MOUNTED

SQL> SELECT CURRENT_SCN FROM V$DATABASE;
CURRENT_SCN
-----------
          0
```

```
SQL> SELECT DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER FROM DUAL;
SELECT DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER FROM DUAL
       *
ERROR at line 1:
ORA-00904: "DBMS_FLASHBACK"."GET_SYSTEM_CHANGE_NUMBER": invalid identifier
```

- HINT: The situation looks different on the Standby Database

ORA_ROWSCN

What is it?

ORA_ROWSCN is a pseudo column of any table that is not fixed or external

It represents the SCN of the most recent change to a given row

- That is, the most recent commit

```
SQL> select ora_rowscn, ename, sal from emp where ename = 'SCOTT';
ORA_ROWSCN ENAME                SAL
---------- ---------- ----------
    962871 SCOTT              3000
```

ORA_ROWSCN is a conservative upper bound of the latest commit time

- The actual commit SCN can be somewhat earlier

ORA_ROWSCN is not absolutely precise

- Because Oracle tracks SCNs by transaction committed for the block in which the row resides
- You can obtain a more fine-grained approximation of the SCN by creating your tables with row-level dependency tracking

So what do we do with the ORA_ROWSCN?

- E.g. we can convert the ORA_ROWSCN to TIMESTAMP to see when the change has happened

RLDT row-level dependency tracking

How to enable it

With the RLDT feature

- Each row in the table has a system change number (SCN)
- The SCN represents a time greater than or equal to the commit time of the last transaction that modified the row

```
create table emp_rdt (empno number, ename varchar2(16)) ROWDEPENDENCIES;
```

You cannot change this setting after table is created

This setting is useful primarily to allow for parallel propagation in replication environments

It increases the size of each row by 6 bytes.

Example with No-RLDT (default)

Without Row Level Dependency Tracking

```
SQL> create table emp_nordt (empno number, ename varchar2(16))
   2 NOROWDEPENDENCIES;  -- this is the default
Table created.
SQL> insert into emp_x values (1, 'William');
1 row created.
SQL> commit;
Commit complete.
SQL> select ora_rowscn, empno, ename from emp_x;
ORA_ROWSCN      EMPNO ENAME
---------- ---------- ----------------
   2165176          1 William

SQL> insert into emp_x values (1, 'Ricardo');
1 row created.
SQL> commit;
Commit complete.
SQL> select ora_rowscn, empno, ename from emp_x;
ORA_ROWSCN      EMPNO ENAME
---------- ---------- ----------------
   2166201          1 William
   2166201          1 Ricardo
```

Example with RLDT

With Row Level Dependency Tracking

```
SQL> insert into emp_rdt values (1, 'William');
1 row created.
SQL> commit;
Commit complete.

SQL> select ora_rowscn, empno, ename from emp_rdt;
ORA_ROWSCN      EMPNO ENAME
---------- ---------- ----------------
   2155592          1 William

SQL> insert into emp_rdt values (1, 'Ricardo');
1 row created.
SQL> commit;
Commit complete.

SQL> select ora_rowscn, empno, ename from emp_rdt;
ORA_ROWSCN      EMPNO ENAME
---------- ---------- ----------------
   2155592          1 William
   2155607          1 Ricardo
```

Flashback Query Performance

How to optimize it?

Generate statistics on all tables involved in a Flashback Query

- Flashback Query always uses the cost-based optimizer

Depends on how much undo data must be accessed

- CPU bound on applying undo data to blocks in the buffer cache

Use queries that access data via indexes

- Try to parallelize full table scans

Flashback Query against a materialized view does not take advantage of query rewrite optimizations

Does not work across a database bounce

- Unless _IN_MEMORY_UNDO is set to false
  - But setting _IN_MEMORY_UNDO to false might impact the performance of your database

Tables with ROWDEPENDENCY add an overhead of 6 bytes to each row

Core Messages

Core messages
Pro & contra

- Flashback Query is very powerful and easy to use
- It comes out of the box with no extra license
- Undo data has to be created anyway, so there is no extra performance overhead when using this feature
- No need for archive logging to be enabled
- No need for supplemental log
- It works on any RAC instance

- Supplemental Logging adds an extra overhead
- Flashback Logs adds and extra overhead
- Might give you wrong results after a crash if _in_memory_undo is true

Let's meet at booth 242

Oracle 12c Flashback Techniques

Any questions? Please do ask

William Sescu

Consultant

+41 78 674 12 90

william.sescu@dbi-services.com

We look forward to working with you!