

# (Single) Table Recovery

**Christian Gohmann**  
nLOUG / OBUG EMEA Tour

## 2 HALLO, GRÜEZI, HI!



### CHRISTIAN GOHMANN

- Tech Architecture Associate Manager, Instructor and Tool Owner of db\*BACKUP
- Working with Oracle since 2006
- Focused on High Availability Solutions, Migration Projects, Backup & Recovery and Cloud Technologies
- Oracle ACE Pro



## **5 AGENDA**

**1. Introduction**

**2. Flashback**

**3. LogMiner**

**4. Recovery Manager (RMAN)**

**5. Summary**

# INTRODUCTION

## 7 CHALLENGE

- Real-life customer request to restore two tables dropped the day before

Two database tables need restoring which have been **dropped accidentally** on the **XX01XXP** database.

Tables names are as follows :

**1. APP\_OWNER.SALES\_DATA**

**2. APP\_OWNER.SALES\_DATA\_ARC**

These are required for Monthend processing hence is very urgent.

Please restore to the point in time **26.01.2023 09:00**.

## 8 DATABASE

- Affected target database was a **PDB in a single-tenancy CDB**
- Version was 19c (19.16)
- All Tablespaces were created as **Bigfile Tablespaces** and encrypted with **Transparent Data Encryption (TDE)**
  - Indexes and Tables were in separate Tablespaces
- Datafiles were located on **ASM (4-node RAC system)**
- Total size of the PDB was 62 TB
  - Size of the Tablespace with the affected tables was 14 TB
  - Nearly 3 TB were occupied by Undo Tablespaces



Create separate Tablespaces if multiple applications are hosted in the database.

## 9 FOREIGN KEY CONSTRAINTS

- The number of tables to restore can increase when foreign key constraint are in place
- Check if ON DELETE clause is used

```
SQL> SELECT c.owner, c.table_name, c.constraint_name, c.r_constraint_name,
           p.owner AS "PARENT_OWNER", p.table_name AS "PARENT_TABLE_NAME",
           c.delete_rule
FROM dba_constraints c
     JOIN dba_constraints p ON (c.r_constraint_name = p.constraint_name)
WHERE p.owner = 'HR'
     AND p.table_name = 'EMPLOYEES';
```

OWNER	TABLE_NAME	CONSTRAINT_NAME	R_CONSTRAINT_NAME	PARENT_OWNER	PARENT_TABLE_NAME	DELETE_RULE
HR	DEPARTMENTS	DEPT_MGR_FK	EMP_EMP_ID_PK	HR	EMPLOYEES	CASCADE
HR	EMPLOYEES	EMP_MANAGER_FK	EMP_EMP_ID_PK	HR	EMPLOYEES	NO ACTION
HR	JOB_HISTORY	JHIST_EMP_FK	EMP_EMP_ID_PK	HR	EMPLOYEES	SET NULL

# 10 TRANSACTIONS

- Keep in mind that **good** transactions might need be to saved after a **bad** transaction happened

```
3.11.982  
05.05.23 09:38:22  
UPDATE employees  
  SET salary = salary * 2;
```

```
7.22.866  
15.05.23 13:11:57  
DELETE FROM employees  
  WHERE last_name = 'Gates';
```

```
4.26.888  
07.05.23 15:51:12  
UPDATE employees SET department_id = 70  
  WHERE employee_id = 203;
```



**FLASHBACK**

## 12 INTRODUCTION

- Flashback was introduced with 9i and enhanced with later versions
- A **set of different data recovery solutions** to reverse human errors or to query historical data
  - Supports recovery on all levels (row, transaction, table, database)
- Depending on the feature either **Undo data** or **Flashback Logs** are used
  - Fast Recovery Area is required to support the storage of Flashback Logs



Most of the Flashback features requires the Enterprise Edition.

## 13 USE CASES

- Query a historical state of a table or show all changes in a historical order
- Rollback of committed transactions
- Restore of a dropped table
- Track changes of a table over a long period (versioning)
- Reset of the database to a previous state with a physical restore
  - For example, to reverse a failed application update

## 14 UNDO RETENTION

- All Flashback features are using the **before images of a row** in the UNDO Tablespace to construct a previous state of a table
  - Except Flashback Drop and Flashback Database
- Database parameter **undo\_retention** controls how long UNDO of committed transactions will be kept (default 900 seconds)

```
SQL> ALTER SYSTEM SET undo_retention = 86400 SCOPE = BOTH;
```

- Per default, the **Undo retention is not guaranteed**
  - Unexpired Undo Extents will be overridden, if not enough space is available

```
SQL> ALTER TABLESPACE UNDOTBS RETENTION GUARANTEE;
```



Automatic Undo Management is required to use Flashback.

## 15 ORA-01555 SNAPSHOT TOO OLD

- If the required UNDO is not available, an ORA-01555 error is raised

```
SQL> SELECT first_name, last_name, salary  
       FROM employees AS OF TIMESTAMP TO_TIMESTAMP('15.05.2023', 'dd.mm.yyyy');
```

ERROR at line 2:

```
ORA-01555: snapshot too old: rollback segment number 343 with name  
"_SYSSMU343_1952840052$" too small
```

- Query column SSOLDERRCNT of **(G)V\$UNDOSTAT** to retrieve the number of occurrences
- PL/SQL package **DBMS\_UNDO\_ADV** provides useful functions/procedures to optimize the Undo Retention/Tablespace



The same error is raised, when you try to create a historical or for a long-running Data Pump export.

## 16 AS OF TIMESTAMP / SCN

- To provide the desired point-in-time a timestamp (**AS OF TIMESTAMP**) or a SCN (**AS OF SCN**) can be provided in a SELECT statement

```
SQL> SELECT first_name, last_name, salary  
       FROM employees AS OF TIMESTAMP TO_TIMESTAMP('15.05.2023', 'dd.mm.yyyy');
```

- If all statements of a session need the same point-in-time, **DBMS\_FLASHBACK** can be used to set it

```
SQL> EXEC DBMS_FLASHBACK.ENABLE_AT_TIME(TO_TIMESTAMP('15.05.2023', 'dd.mm.yyyy'));  
SQL> EXEC DBMS_FLASHBACK.ENABLE_AT_SYSTEM_CHANGE_NUMBER(123456);  
SQL> EXEC DBMS_FLASHBACK.DISABLE;
```



EXECUTE privileges are required to use DBMS\_FLASHBACK.

## 17 FLASHBACK QUERY

- Shows the state of a table for a specific point-in-time in the past
- Example: Show the new and old salary of an employee

```
SQL> UPDATE employees SET salary = salary*2 WHERE last_name = 'Gates';  
SQL> SELECT last_name, salary FROM employees WHERE last_name = 'Gates';
```

FIRST_NAME	LAST_NAME	SALARY
Timothy	Gates	5800

```
SQL> SELECT last_name, salary  
       FROM employees AS OF TIMESTAMP TO_TIMESTAMP('15.05.2023', 'dd.mm.yyyy')  
       WHERE last_name = 'Gates';
```

FIRST_NAME	LAST_NAME	SALARY
Timothy	Gates	2900

## 18 FLASHBACK VERSION QUERY

- Shows the different versions of rows for a **specific time period**

```
SQL> SELECT versions_starttime, versions_endtime, versions_xid,  
           versions_operation, last_name, salary  
FROM employees  
   VERSIONS BETWEEN TIMESTAMP  
           TO_TIMESTAMP('15.05.2023 10:35', 'dd.mm.yyyy HH24:MI') AND  
           TO_TIMESTAMP('15.05.2023 10:47', 'dd.mm.yyyy HH24:MI')  
WHERE last_name = 'Gates';
```

VERSIONS_STARTTIME	VERSIONS_ENDTIME	VERSIONS_XID	V	LAST_NAME	SALARY
15.05.2023 10:45:54		080019005F040000	D	Gates	2900
15.05.2023 10:44:39	15.05.2023 10:45:54	080013005E040000	U	Gates	2900
	15.05.2023 10:44:39			Gates	5800



## 19 FLASHBACK TRANSACTION QUERY

- To use it **Supplemental Logging for Primary Keys and Foreign Keys** is required
- The dynamic view **FLASHBACK\_TRANSACTION\_QUERY** will be used to generate the Undo statement(s) for a transaction

```
SQL> SELECT operation, row_id, undo_sql
       FROM flashback_transaction_query WHERE xid = '04000100B0030000';
```

OPERATION	ROW_ID	UNDO_SQL
DELETE	AAAUEMAAPAAAALdAAy	insert into "HR"."EMPLOYEES"("EMPLOYEE_ID", "FIRST_NAME", "LAST_NAME", "EMAIL", "PHONE_NUMBER", "HIRE_DATE", "JOB_ID", "SALARY", "COMMISSION_PCT", "MANAGER_ID", "DEPARTMENT_ID") values ('150', 'Peter', 'Tucker', 'PTUCKER', '011.44.1344.129268', TO_DATE('30-JAN-05', 'DD-MON-RR'), 'SA_REP', '10000', '.3', '145', '80');

NLS\_DATE\_FORMAT  
is used



Use Flashback Version Query to get the transaction ID.

## 20 FLASHBACK TABLE

- Allows the **rollback of all transactions of a table** in one command
- ROWIDs will change during the Flashback
  - Row movement for a table must be enabled (otherwise ORA-08189)

```
SQL> ALTER TABLE employees ENABLE ROW MOVEMENT;  
SQL> FLASHBACK TABLE hr.employees  
    TO TIMESTAMP TO_TIMESTAMP('15.05.2023 16:58', 'dd.mm.yyyy HH24:MI');  
SQL> ALTER TABLE employees DISABLE ROW MOVEMENT;
```

- As an alternative to SCN and Timestamp, a restore point can be used
  - Requires SELECT ANY DICTIONARY or FLASHBACK ANY TABLE or SELECT\_CATALOG\_ROLE grant

```
SQL> FLASHBACK TABLE hr.employees TO RESTORE POINT before_update;
```



Enabled Triggers are disabled during the Flashback and afterwards reenabled.

## 21 FLASHBACK DROP 1/3

- Restore of a dropped table using the recycle bin of the database
- To activate the feature, database parameter **recyclebin** must be set to on (default)

```
SQL> ALTER SYSTEM SET recyclebin = 'ON' SCOPE = BOTH;
```

- If activated, the table is renamed instead of dropped
  - Dependent objects (indexes, constraints) are also renamed
- How long a dropped table can be restored cannot be set
  - It depends on the data growth in the Tablespace
  - If the Tablespace would need a physical extension, the blocks of an object in the recycle bin will be reused



Use DROP ... PURGE to drop a table directly.

## 22 FLASHBACK DROP 2/3

- Query DBA|USER\_RECYCLEBIN (or the synonym RECYCLEBIN) to check the recycle bin

```
SQL> DROP TABLE job_history;
SQL> SELECT object_name, original_name, type, droptime, can_undrop
       FROM user_recyclebin;
```

OBJECT_NAME	ORIGINAL_NAME	TYPE	DROPTIME	CAN
BIN\$/Dw5W/AeUb/gUwEAAH/pSg==\$0	JHIST_DEPARTMENT_IX	INDEX	2023-05-21:21:56:19	NO
BIN\$/Dw5W/AfUb/gUwEAAH/pSg==\$0	JHIST_EMPLOYEE_IX	INDEX	2023-05-21:21:56:19	NO
BIN\$/Dw5W/AgUb/gUwEAAH/pSg==\$0	JHIST_JOB_IX	INDEX	2023-05-21:21:56:19	NO
BIN\$/Dw5W/AhUb/gUwEAAH/pSg==\$0	JHIST_EMP_ID_ST_DATE_PK	INDEX	2023-05-21:21:56:19	NO
<b>BIN\$/Dw5W/AiUb/gUwEAAH/pSg==\$0</b>	<b>JOB_HISTORY</b>	<b>TABLE</b>	<b>2023-05-21:21:56:19</b>	<b>YES</b>



Although an index has the value NO for CAN\_UNDROP, it will be restored when the table is restored.

## 23 FLASHBACK DROP 3/3

- Restores the table from the recycle bin
  - The original name or the name of the object in the recycle bin are supported

```
SQL> FLASHBACK TABLE job_history TO BEFORE DROP;  
SQL> FLASHBACK TABLE "BIN$/Dw5W/AiUb/gUwEAAH/pSg==$0" TO BEFORE DROP;
```

- Indexes will not be renamed to their original name

```
SQL> SELECT index_name FROM user_indexes WHERE table_name = 'JOB_HISTORY';  
  
INDEX_NAME  
-----  
BIN$/Dw5W/AeUb/gUwEAAH/pSg==$0  
...
```



Generate the ALTER INDEX ... RENAME statements based on the recycle bin before restoring the table.

## 24 FLASHBACK (PLUGGABLE) DATABASE 1/3

- “Rewinds” the database to a point-in-time in the past using Flashback Logs
  - A fast forward is also possible when no OPEN RESETLOGS was executed
- The **Recovery Writer Process (RVWR)** writes Flashback data to the Flashback Logs
- Query column FLASHBACK\_ON of V\$DATABASE to check if it is enabled

```
SQL> SELECT flashback_on FROM v$database;
```

```
FLASHBACK_ON  
-----  
YES
```

- Database parameter **db\_flashback\_retention\_target** controls the upper limit
  - Flashback Logs are deleted, when the FRA is running out of space (except for Guaranteed Restore Points)



Creating a Guaranteed Restore Point activates Flashback Log Mode implicitly.

## 25 FLASHBACK (PLUGGABLE) DATABASE 2/3

### Requirements:

- Archive Log Mode
- Flashback Log Mode

```
SQL> ALTER DATABASE FLASHBACK ON;
```

- Fast Recovery Area

```
SQL> ALTER SYSTEM SET db_recovery_file_dest_size = 1T SCOPE = BOTH;  
SQL> ALTER SYSTEM SET db_recovery_file_dest = '+FRA' SCOPE = BOTH;
```

- Local Undo Mode to support Flashback for a single PDB in a CDB



Starting with 23c a dedicated area for the Flashback Logs can be defined (`db_flashback_file_dest_size` and `db_flashback_file_dest`).

## 26 FLASHBACK (PLUGGABLE) DATABASE 3/4

1. Close the PDB or bring the database to the MOUNTED state

```
SQL> ALTER PLUGGABLE DATABASE PDB1 CLOSE IMMEDIATE INSTANCES = ALL;
```

2. Flashback the PDB/database

```
SQL> FLASHBACK PLUGGABLE DATABASE PDB1  
      TO TIMESTAMP TO_TIMESTAMP('15.05.2023 11:00:00', 'dd.mm.yyy HH24:MI:SS');
```

3. Open the PDB/database in read-only mode

```
SQL> ALTER PLUGGABLE DATABASE PDB1 OPEN READ ONLY;
```



To open the PDB/database after a Flashback directly in read-write mode, you need to use the OPEN RESETLOGS option.



## 27 FLASHBACK (PLUGGABLE) DATABASE 4/4

4. Extract the required data (e.g., SQL Loader or Data Pump export over DB-Link)
5. Recover the PDB/database using RMAN

```
RMAN> RECOVER PLUGGABLE DATABASE PDB1;
```

6. Open the PDB/database read-write

```
SQL> ALTER PLUGGABLE DATABASE PDB OPEN INSTANCES = ALL;
```



You can create a DB-Link from the CDB to the PDB to run the Data Pump export.

# LOGMINER

## 29 GENERAL

- LogMiner was introduced with Oracle 8i
- Allows the **analysis of the contents of (archive) redo logs** for various use cases
  - Detection of logical corruption
  - Recovery of single database objects
  - Auditing of executed DML and DDL statements
  - Enhanced trend analysis and capacity planning
- Internally used by Oracle Data Guard (Logical Standby) and other third-party logical replication tools
- Implemented in PL/SQL
  - [DBMS\\_LOGMNR](#), [DBMS\\_LOGMNR\\_D](#)



LogMiner can be used out-of-the-box, but it has limitations.

## 30 OBJECT IDS

- LogMiner can be used to analyze (archive) redo logs of any database (8.0 and higher)
- Oracle **stores only object IDs** (including column IDs) in the redo stream
- To generate working SQL statements, the **IDs need to be translated** to names
- 3 ways are supported
  - Flat file (backward compatibility)
  - Storing the Data Dictionary in the redo logs  
`DBMS_LOGMNR.DICT_FROM_REDO_LOGS`
  - Usage of the Data Dictionary (preferred method)  
`DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG`



Creating a flat file dump of a PDB is desupported in 21c (ORA-65040).

## 31 SUPPLEMENTAL LOGGING 1/2

- Oracle **stores as little as possible REDO information** in the Redo Logs
- These REDO information can be used to recover the database, but LogMiner needs more information to generate working SQLs (e.g. for chained rows)
- Storing extra REDO information (additional column values) is called **Supplemental Logging**
- It can be activated on CDB or PDB level or for specific tables

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;  
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;  
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;  
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (FOREIGN KEY) COLUMNS;  
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;  
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA FOR PROCEDURAL REPLICATION;  
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA SUBSET DATABASE REPLICATION;
```

For Golden Gate,  
available since 19c



To check Supplemental Logging on PDB level use DBA\_SUPPLEMENTAL\_LOGGING.

## 32 SUPPLEMENTAL LOGGING 2/2

- Impact of Supplemental Logging to the generated SQLs by the LogMiner

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;  
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
```

```
OPERATION SQL_REDO
```

```
-----  
UPDATE      update "HR"."EMPLOYEES"  
            set  
              "SALARY" = 14400  
            where  
              "EMPLOYEE_ID" = 164 and  
              "FIRST_NAME" = 'Mattea' and  
              "LAST_NAME" = 'Marvins' and  
              "EMAIL" = 'MMARVINS' and  
              ...  
              "MANAGER_ID" = 147 and  
              "DEPARTMENT_ID" = 80 and  
            ROWID = 'AAAUEMAAPAAAALeACi';
```

```
SQL_UNDO
```

```
-----  
update "HR"."EMPLOYEES"  
      set  
        "SALARY" = 7200  
      where  
        "EMPLOYEE_ID" = 164 and  
        "FIRST_NAME" = 'Mattea' and  
        "LAST_NAME" = 'Marvins' and  
        "EMAIL" = 'MMARVINS' and  
        ...  
        "MANAGER_ID" = 147 and  
        "DEPARTMENT_ID" = 80 and  
      ROWID = 'AAAUEMAAPAAAALeACi';
```

## 33 IDENTIFY (ARCHIVE) REDO LOGS

- Identify the required (Archive) Redo Logs

```
SQL> SELECT thread#, sequence#, name, deleted
       FROM gv$archived_log
       WHERE first_time BETWEEN
             TO_TIMESTAMP('22.05.2023 07:25', 'dd.mm.yyyy HH24:MI') AND
             TO_TIMESTAMP('22.05.2023 08:40', 'dd.mm.yyyy HH24:MI');
```

THREAD#	SEQUENCE#	NAME	DEL
1	143	/opt/oracle/FREE/archivelog/2023_05_22/o1_mf_1_143_l6pbc7hd_.arc	YES
1	144	/opt/oracle/FREE/archivelog/2023_05_22/o1_mf_1_144_l6pbocy4_.arc	NO

Restore of the  
Archive redo Log  
is required

- Restore already deleted Archive Redo Logs using RMAN

```
RMAN> RESTORE ARCHIVELOG SEQUENCE 143 THREAD 1;
```

## 34 START LOGMINER

- Add the required files and start LogMiner (from CDB\$ROOT)

```
SQL> BEGIN
  DBMS_LOGMNR.ADD_LOGFILE(
    logfilename => '/opt/oracle/FREE/archivelog/2023_05_22/o1_mf_1_143_16pbc7hd_.arc',
    options => DBMS_LOGMNR.NEW
  );
  DBMS_LOGMNR.ADD_LOGFILE(
    logfilename => '/opt/oracle/FREE/archivelog/2023_05_22/o1_mf_1_144_16pbocy4_.arc',
    options => DBMS_LOGMNR.ADDFILE
  );

  DBMS_LOGMNR.START_LOGMNR(
    options => DBMS_LOGMNR.COMMITTED_DATA_ONLY + DBMS_LOGMNR.PRINT_PRETTY_SQL +
              DBMS_LOGMNR.NO_ROWID_IN_STMT + DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG
  );
END;
/
```

Resets the list of  
mined (Archive)  
Redo Logs



## 35 EXTRACT REDO/UNDO SQLS

- Use the dynamic view **V\$LOGMNR\_CONTENTS** to retrieve the required SQLs

```
SQL> SELECT xid, start_timestamp, operation, sql_redo, sql_undo
        FROM v$logmnr_contents WHERE table_name = 'EMPLOYEES' AND operation = 'DELETE';
```

XID	START_TIM	OPERATION	SQL_REDO	SQL_UNDO
04000100B0030000	21-MAY-23	DELETE	delete from "HR"."EMPLOYEES" where "EMPLOYEE_ID" = 150 and "FIRST_NAME" = 'Peter' and "LAST_NAME" = 'Tucker' and "EMAIL" = 'PTUCKER' and "PHONE_NUMBER" = '011...129268' and "HIRE_DATE" = '30-JAN-05' and "JOB_ID" = 'SA_REP' and "SALARY" = 10000 and "COMMISSION_PCT" = .3 and "MANAGER_ID" = 145 and "DEPARTMENT_ID" = 80;	insert into "HR"."EMPLOYEES" values "EMPLOYEE_ID" = 150, "FIRST_NAME" = 'Peter', "LAST_NAME" = 'Tucker', "EMAIL" = 'PTUCKER', "PHONE_NUMBER" = '011...129268', "HIRE_DATE" = '30-JAN-05', "JOB_ID" = 'SA_REP', "SALARY" = 10000, "COMMISSION_PCT" = .3, "MANAGER_ID" = 145, "DEPARTMENT_ID" = 80;



Create a table from the dynamic view V\$LOGMNR\_CONTENTS to run multiple queries without mining the (Archive) Redo Logs again.

# RECOVERY MANAGER (RMAN)

## 37 DUPLICATE DATABASE

- **Creates a (historical) copy of a database** on the same server/cluster or on a different server/cluster
- Tablespace and PDBs can be excluded
  - SYSTEM, SYSAUX and the UNDO Tablespaces are mandatory
- A copy from the active database or from an existing backup can be performed
- Starting with 18c, RMAN supports cloning of a single PDB into an existing CDB using RMAN
  - Has a lot of limitations, not recommended
- Two important terms
  - **TARGET:** Database that will be cloned (`CONNECT TARGET`)
  - **AUXILIARY:** Target instance of the clone operation (`CONNECT AUXILIARY`)



Push and pull clones are supported – a pull clone requires less preparation work.

## 38 AUXILIARY INSTANCE

- To create a clone, RMAN needs a so-called **Auxiliary Instance**
- Create the Auxiliary Instance on the target server/cluster in the target Oracle Home
- Only mandatory parameter for the Auxiliary Instance is **db\_name**
  - But it is better to create a PFILE of the target database and adjust the values

```
SQL> CREATE PFILE = '/tmp/initAUXDB.ora' FROM SPFILE;
```

- Start the Auxiliary Instance

```
SQL> CREATE SPFILE FROM PFILE = '/tmp/initAUXDB.ora';  
SQL> STARTUP NOMOUNT
```



For push clones a static Listener entry is required for the Auxiliary Instance.

## 39 EXAMPLE

- Connect to RMAN Catalog (optional), Target and Auxiliary and create the clone

```
RMAN> CONNECT CATALOG rman/rman@RMANDB
RMAN> CONNECT TARGET sys/manager@XX01XXP
RMAN> CONNECT AUXILIARY /

RUN {
  ALLOCATE CHANNEL disk1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL sbt1 DEVICE TYPE SBT_TAPE ...;
  ALLOCATE AUXILIARY CHANNEL auxd1 DEVICE TYPE DISK;
  ALLOCATE AUXILIARY CHANNEL aux1 DEVICE TYPE SBT_TAPE ...;
  SET UNTIL TIME "TO_TIMESTAMP('22.05.2023 07:25', 'dd.mm.yyyy HH24:MI')";
  DUPLICATE TARGET DATABASE TO AUXDB
    SKIP PLUGGABLE DATABASE HRPDB
    SKIP TABLESPACE CRMPDB:ARCH_DATA;
}
```



It is recommended to allocate disk channels even when all backups are stored on tape.

## 40 RECOVER TABLE

- Introduced with Oracle 12c
- Can be used to **recover single/multiple table (partitions)**
- Creates a minimal historical clone of the database including SYSTEM, SYSAUX, UNDO and the Tablespaces containing the tables (+ Tablespaces of dependent objects)
- An **Automatic Instance** is used for the recovery
- Three options are available for the handling of the tables after the recovery
  - Replace the table(s) in the target database (default)
  - Create a Data Pump dump of the table(s) (**NOTABLEIMPORT**)
  - Create the table with a new name in the target database (**REMAP TABLE**)
- ASM and filesystem are supported locations



If REMAP TABLE is used, named Constraints and Indexes are not imported.

## 4.1 AUTOMATIC INSTANCE 1/2

- An automatic instance is started for the table recovery
- It uses a system generated database unique and instance name
- After the operation completed (with success or failure), the instance is automatically removed

```
Removing automatic instance
shutting down automatic instance
Oracle instance shut down
Automatic instance removed
auxiliary instance file /u01/oradata/C21SFE1_SITE1/controlfile/o1_mf_l6rycb5d_.ctl deleted
```

- To avoid the deletion of the Auxiliary Instance, use the clause **KEEP AUXILIARY**
  - Available since 19c or via one-off patch 22820798 in 12c Release 2
  - <https://christian-gohmann.de/2020/05/08/keep-rman-auxiliary-instance-after-failure/>

## 42 AUTOMATIC INSTANCE 2/2

- Parameters of an Automatic Instance (21c)

Creating automatic instance, with SID='olgd'

initialization parameters used for automatic instance:

```
db_name=C21SFE1
db_unique_name=olgd_pitr_C21SFE1PDB1_C21SFE1
compatible=21.0.0
db_block_size=8192
db_files=200
diagnostic_dest=/u00/app/oracle
_pdb_name_case_sensitive=false
_system_trig_enabled=FALSE
db_domain=goh.trivadis.local
sga_target=2048M
processes=200
db_create_file_dest=/opt/oracle
log_archive_dest_1='location=/opt/oracle'
enable_pluggable_database=true
_clone_one_pdb_recovery=true
#No auxiliary parameter file used
```

Same size as the  
source database



## 44 EXAMPLE

- Connect to RMAN Catalog (optional), Target and start the table recovery

```
RMAN> CONNECT CATALOG rman/rman@RMANDB
RMAN> CONNECT TARGET sys/manager@XX01XXP

RMAN> RUN {
  SET AUXILIARY PARAMETER FILE TO '/tmp/init.ora';
  RECOVER TABLE HR.EMPLOYEES OF PLUGGABLE DATABASE XX01XXP
    UNTIL TIME "TO_DATE('15.05.2023 14:45', 'dd.mm.yyyy HH24:MI')"  
    AUXILIARY DESTINATION '+DATA'  
    DATAPUMP DESTINATION '/dumps/XX01XXP'  
    REMAP TABLE HR.EMPLOYEES:HR.EMPLOYEES_RESTORED  
    KEEP AUXILIARY  
  ;
}
```

Optional



To use a different memory configuration, use an Auxiliary Parameter File (Doc ID 2430319.1).

# SUMMARY

## 46 OPTIMIZATION POTENTIAL

- Optimize the database design
  - Add new Tablespaces for Indexes or archived data
  - Create business-critical tables in a dedicated Tablespace
- Optimize the Undo Tablespace and retention time
- Increase the Flashback retention time
- Activate Flashback Time Travel (also known as Flashback Data Archive)
  - Undo data is saved for a longer period

```
SQL> CREATE FLASHBACK ARCHIVE longterm_arc TABLESPACE arc_data RETENTION 1 YEAR;  
SQL> ALTER TABLE sales_data FLASHBACKCL ARCHIVE longterm_arc;
```

- Create a Physical Standby database with a delayed application of Redo

# FURTHER INFORMATION

## 48 LINKS

- **Using LogMiner to Analyze Redo Log Files**

<https://docs.oracle.com/en/database/oracle/oracle-database/21/sutil/oracle-logminer-utility.html>

- **Using Oracle Flashback Technology**

<https://docs.oracle.com/en/database/oracle/oracle-database/21/adfns/flashback.html>

- **Oracle Database 21c Backup and Recovery User's Guide**

<https://docs.oracle.com/en/database/oracle/oracle-database/21/bradv/>

- **My Oracle Support**

<https://support.oracle.com>

## 49 QUESTIONS AND ANSWERS



# CHRISTIAN GOHMANN

**Tech Architecture Associate Manager**

 [christian.gohmann@accenture.com](mailto:christian.gohmann@accenture.com)

 <https://www.linkedin.com/in/christian-gohmann/>

 @CGohmannDE

 <https://www.christian-gohmann.de>



**trivadis** Part of Accenture