# Managing Undo Data

By Ahmed Baraka

# Objectives

In this lecture, you will learn how to perform the following:

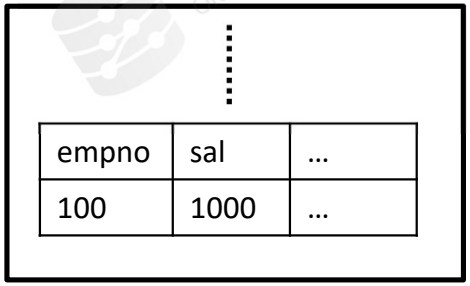
- Describe undo benefits
- Describe undo space management tasks
- Set undo management parameters
- Set undo retention
- Tune undo data
- Use undo advisor
- Enable undo guarantee
- Enable temporary undo
- Enable shared or local undo modes

## User Issuing DMLs

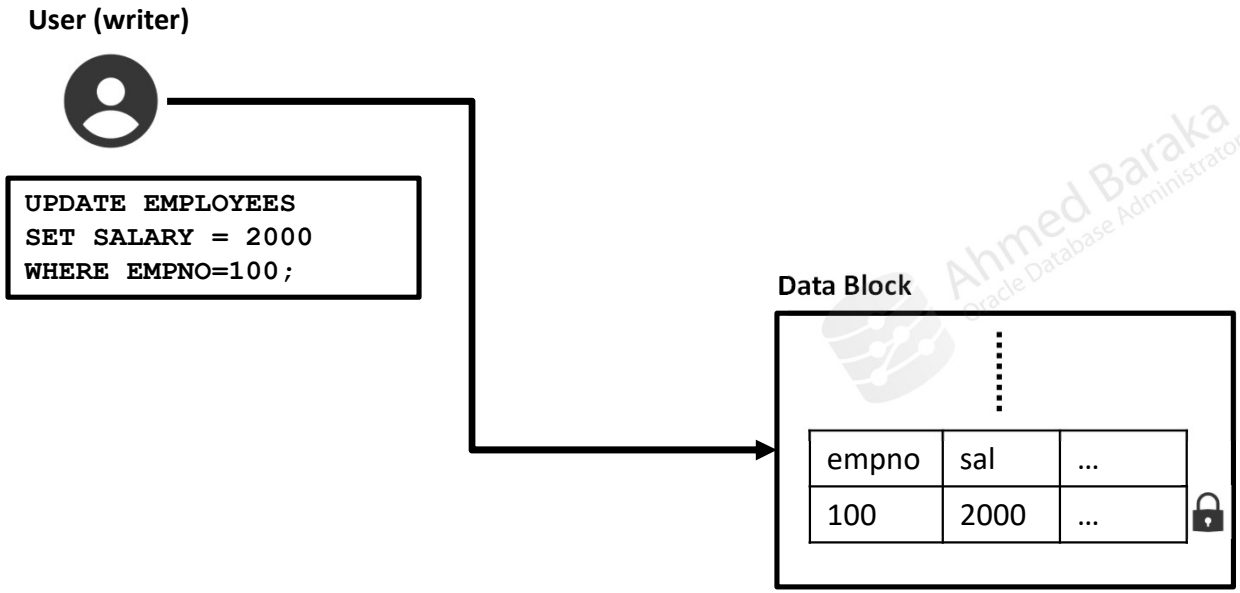
User (writer)

```
UPDATE EMPLOYEES
SET SALARY = 2000
WHERE EMPNO=100;
```

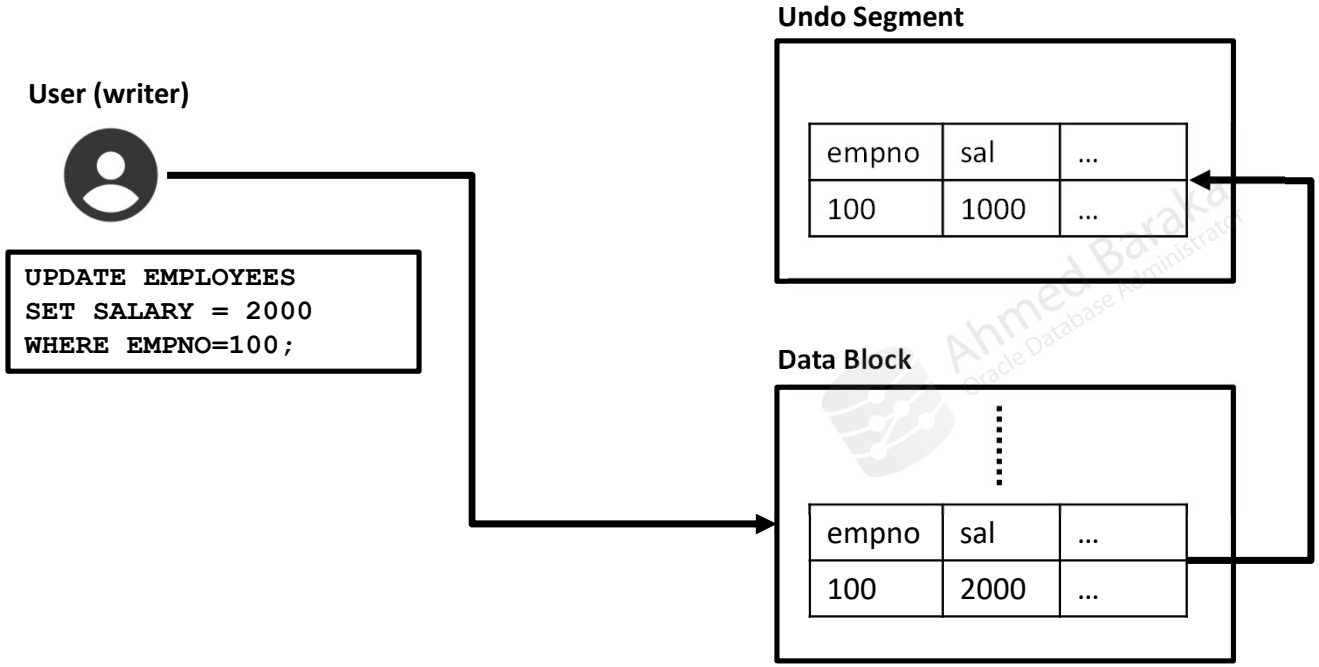
Data Block



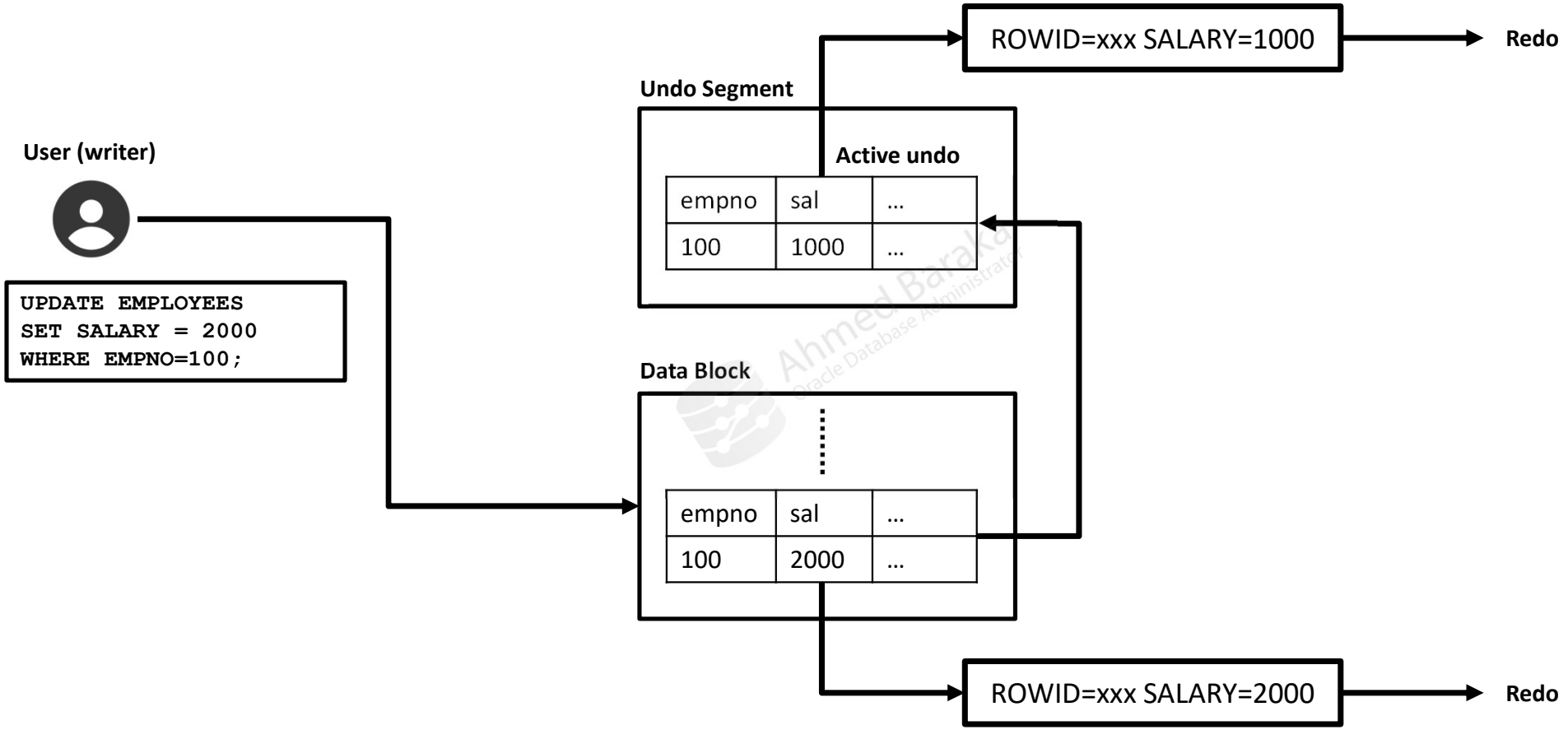
## User Issuing DMLs



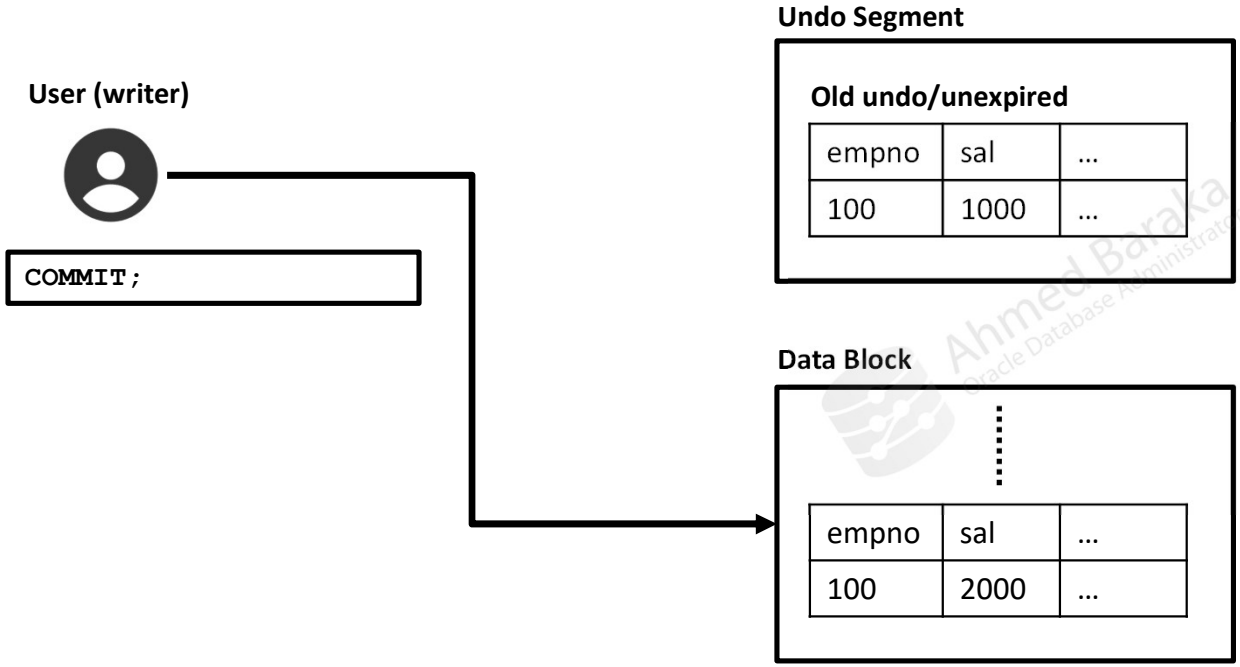
## User Issuing DMLs



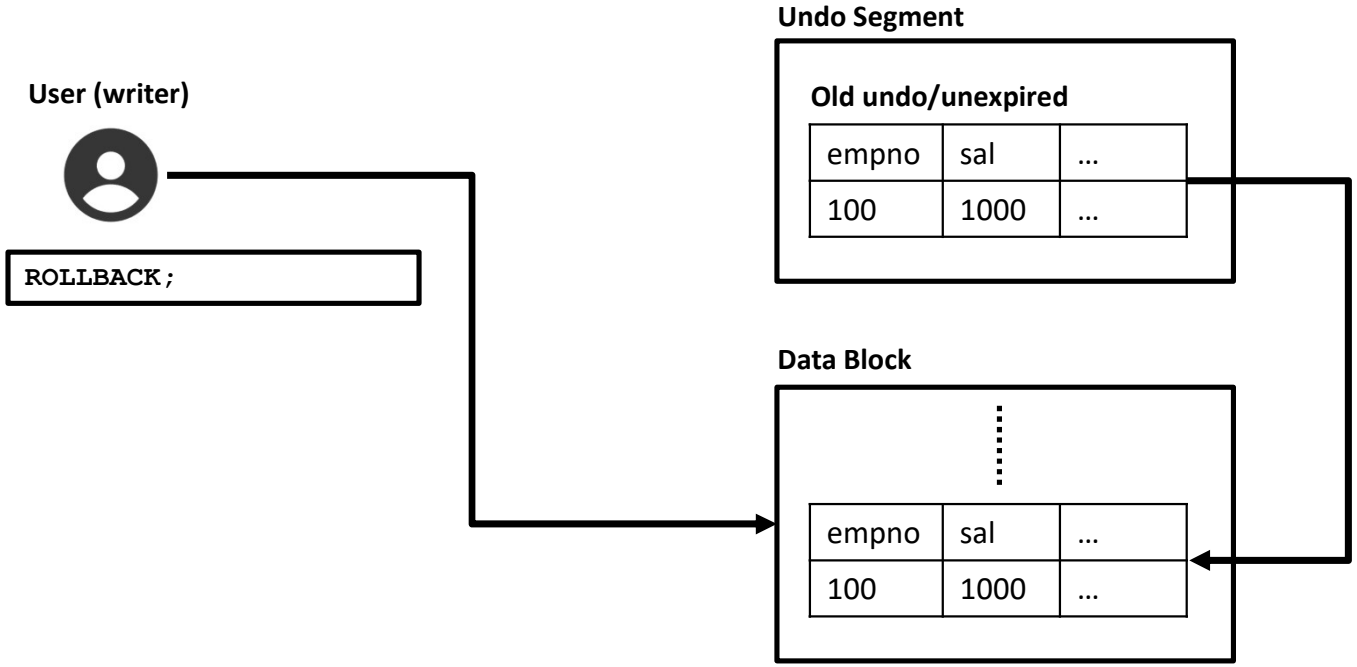
## User Issuing DMLs



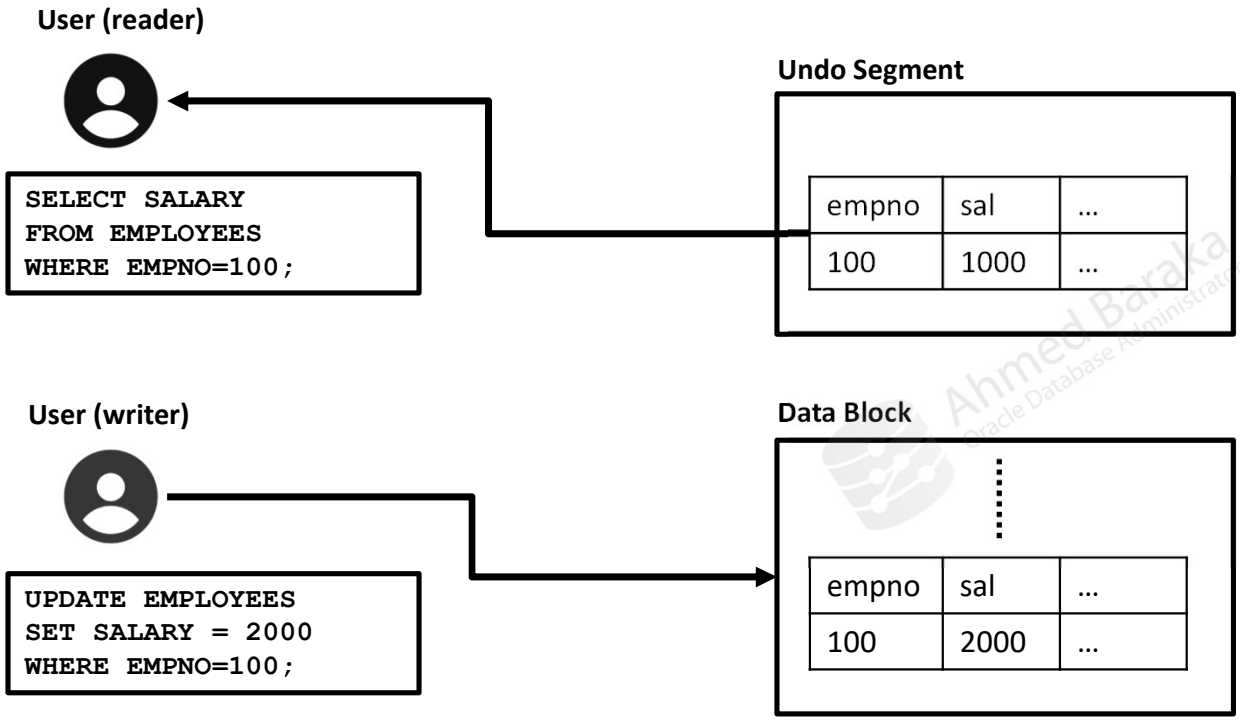
## User Issuing DMLs



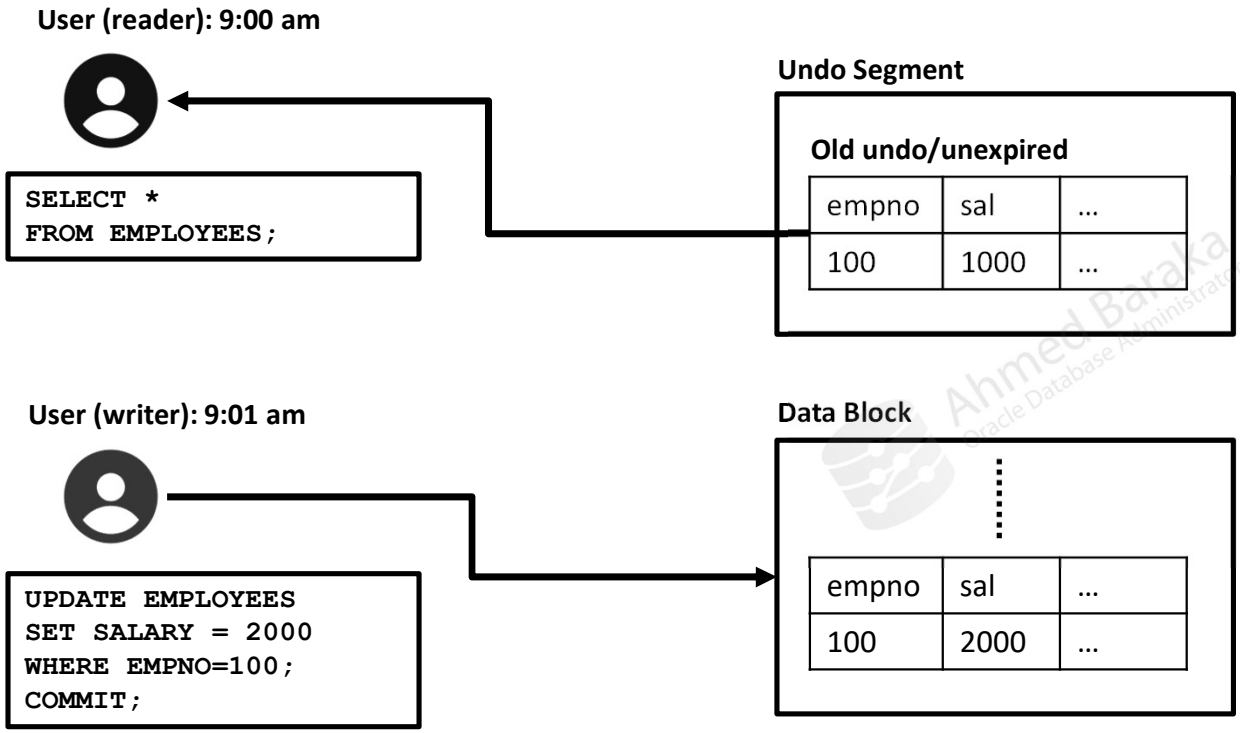
## User Issuing ROLLBACK



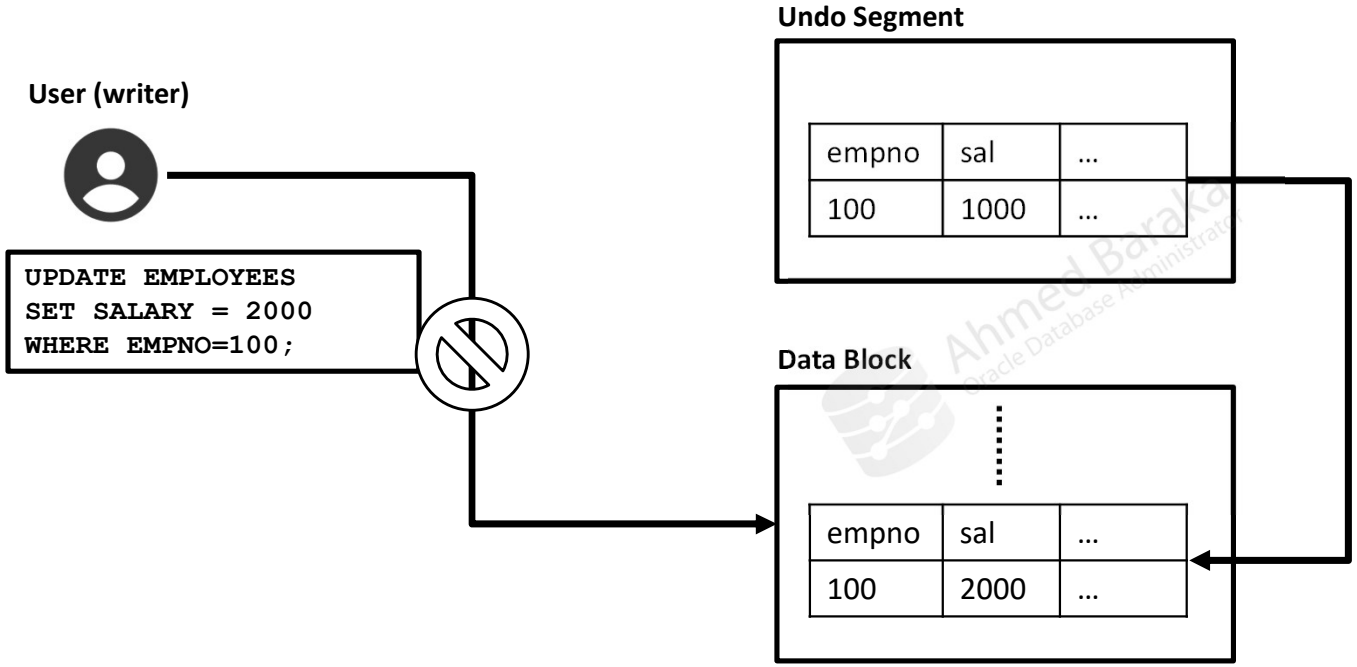
## Query before COMMIT/ROLLBACK



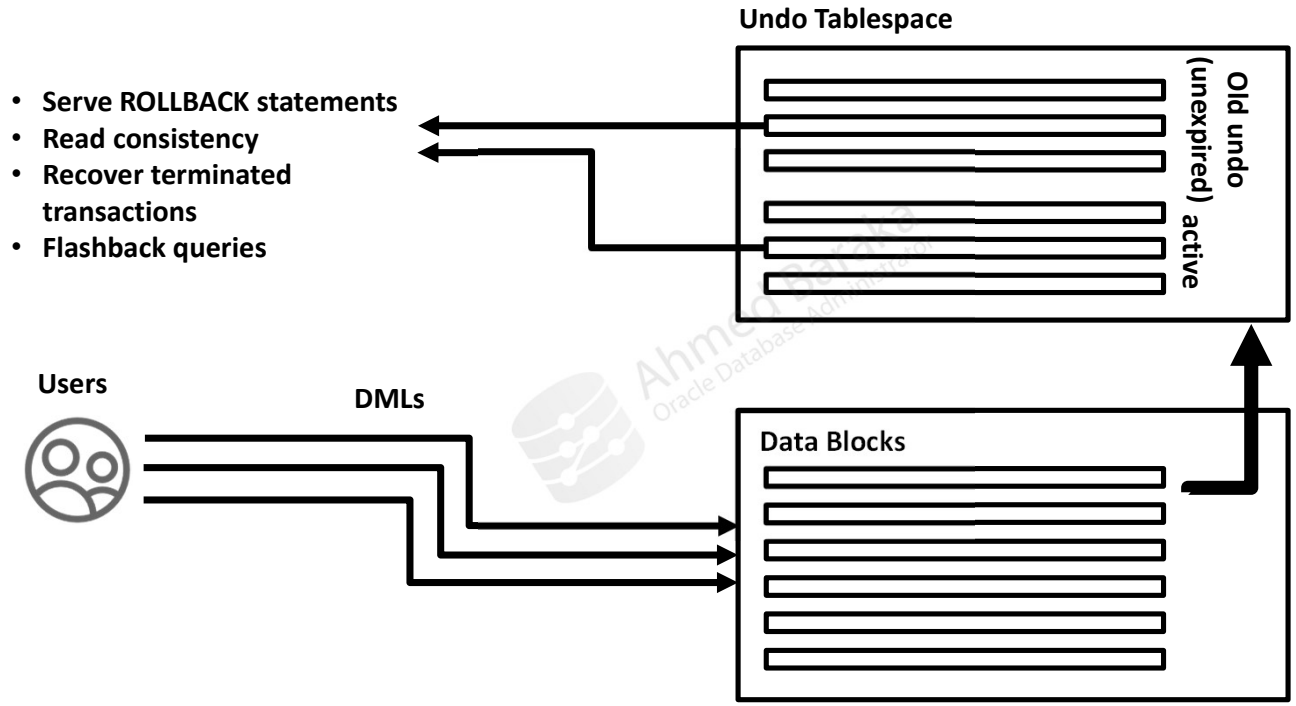
## Long Running Query



## Recover from Failed Transaction



## Undo Contents



# Undo Benefits

- Serve rollback statements
- Read consistency:
  - Readers before commit/rollback
  - Long running queries
- Recovery from failed transactions
- Flashback queries

# Undo Space Management

- Setting the following parameters:
  - **UNDO_MANAGEMENT**
  - **UNDO_TABLESPACE**
- Avoiding space and “snapshot too old” errors
- Configuring the minimum undo retention
- Enabling undo guarantee
- Enabling temporary undo
- Enable shared or local undo modes

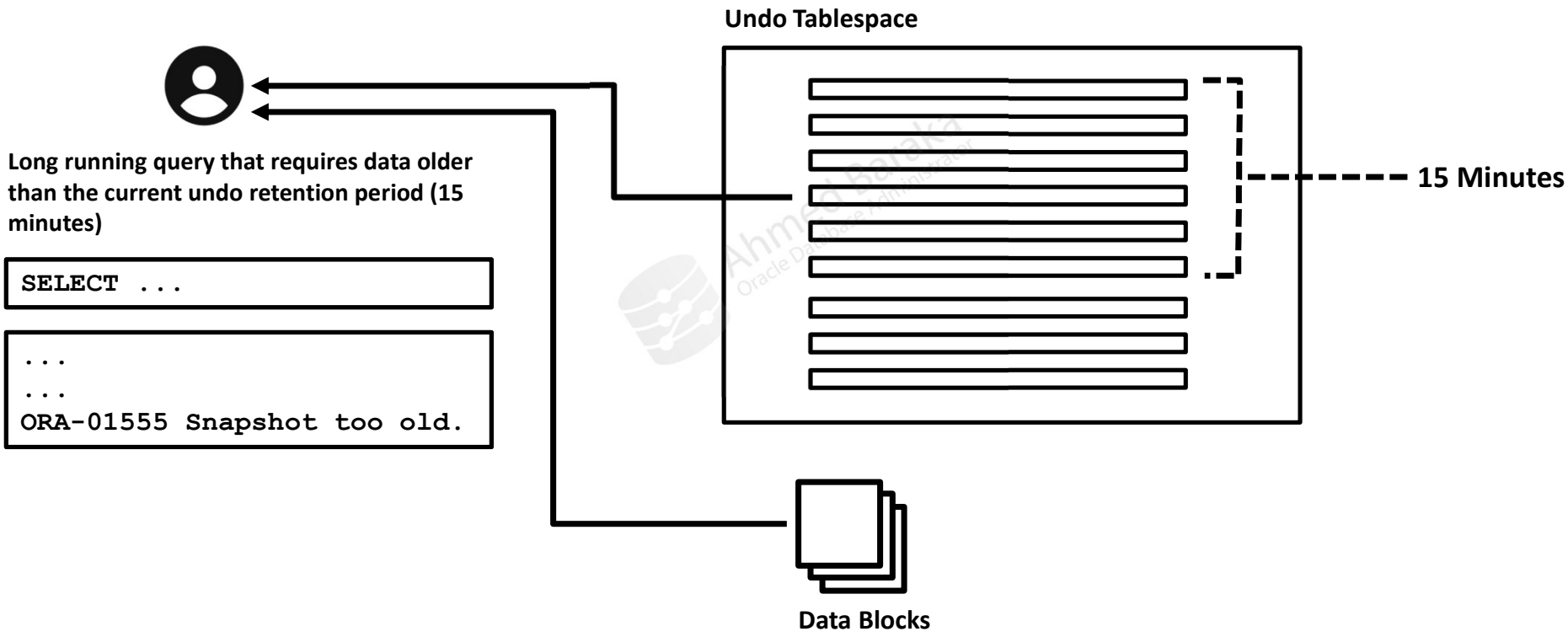
# Setting Undo Parameters

- **UNDO_MANAGEMENT** sets the undo management modes:
  - Automatic (**AUTO**)
  - Manual (**MANUAL**) (not supported in CDB, not recommended)
- One undo tablespace is automatically created by **dbca**
- You can create multiple undo tablespaces, but only one of them can be active at any one time

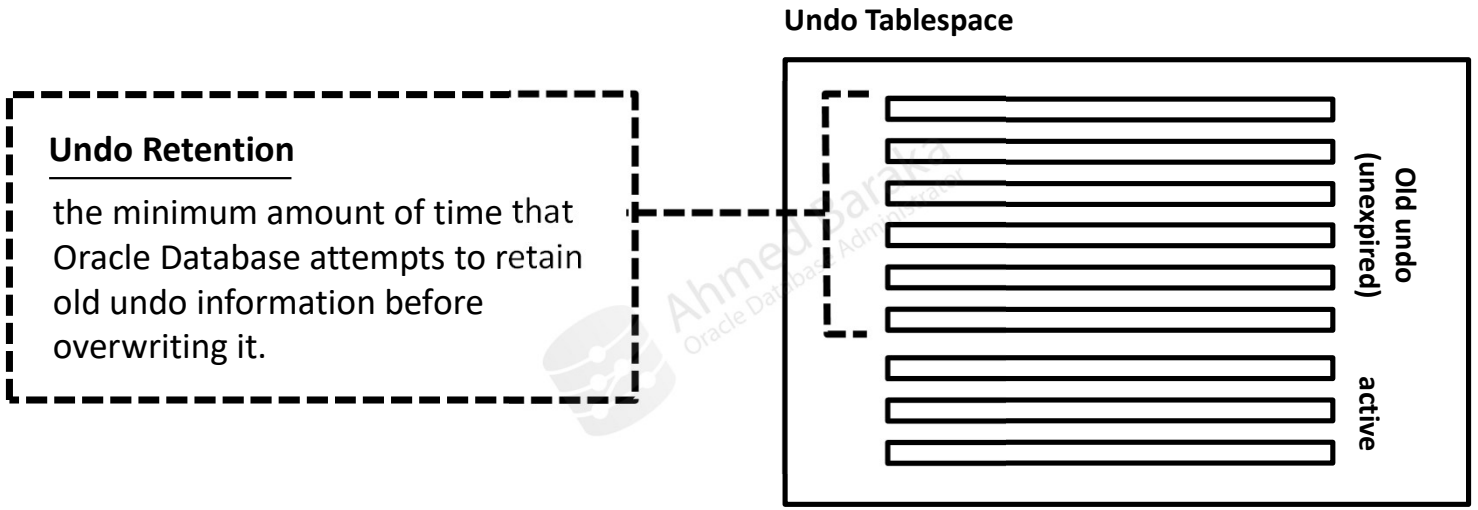
```
CREATE UNDO TABLESPACE undotbs2
DATAFILE '/u01/oracle/rbdb1/undo2.dbf' SIZE 2M REUSE AUTOEXTEND ON;
ALTER SYSTEM SET UNDO_TABLESPACE = undotbs2;
```

- No user object can be created in undo tablespaces

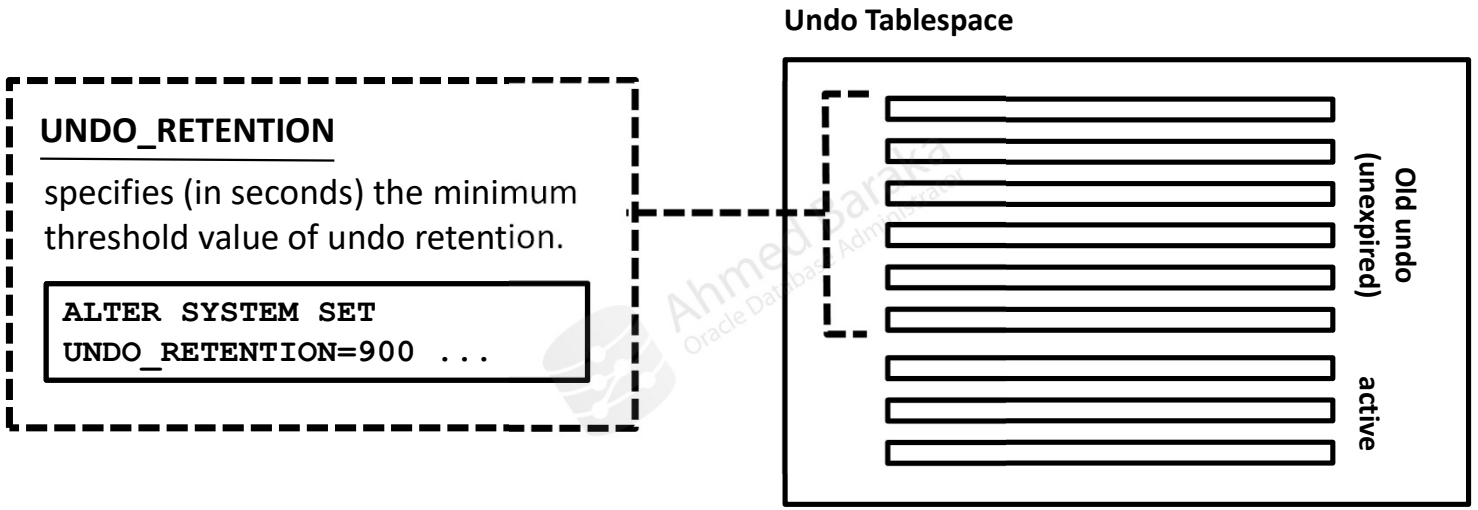
## “Snapshot too old” Error



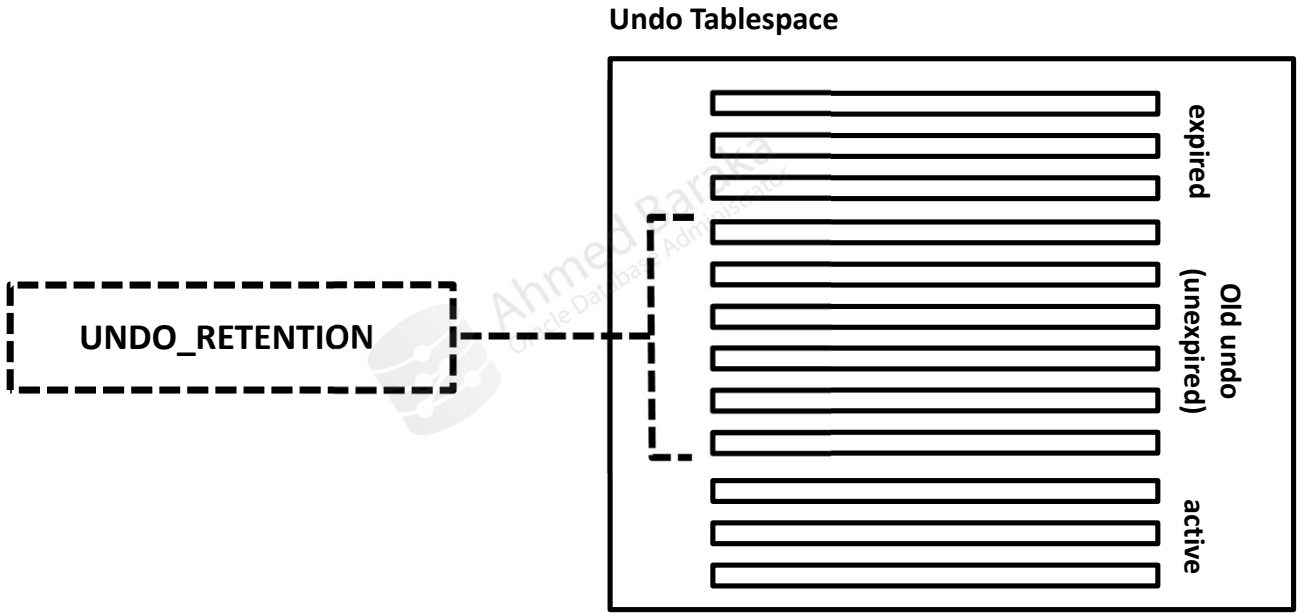
## Undo Retention



## Setting UNDO_RETENTION



## Expired Undo



# Setting the Undo Retention

## UNDO_RETENTION

`UNDO_RETENTION` specifies (in seconds) the low threshold value of undo retention.

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `900` |
| **Modifiable** | `ALTER SYSTEM` |
| **Modifiable in a PDB** | Yes |
| **Range of values** | `0` to $2^{31} - 1$ |
| **Basic** | No |
| **Oracle RAC** | Oracle recommends that multiple instances have the same value. |

# Using V$UNDOSTAT to Tune Undo Data

- Use **V$UNDOSTAT** to tune the undo tablespace: size, retention period

| Column | Description |
|---|---|
| BEGIN_TIME | The beginning point in time of the time interval |
| END_TIME | The end of the point in time of the time interval |
| UNDOBLKS | The total number of undo blocks consumed. |
| MAXQUERYLEN | The length of the longest query (in seconds) executed during the period |
| MAXQUERYID | SQL identifier of the longest running SQL statement in the period |
| SSOLDERRCNT | The number of times the error ORA-01555 occurred |
| UNXPBLKRELCNT | Number of unexpired blocks removed from certain undo segments |
| ACTIVEBLKS | Total number of blocks in the active extents of the undo tablespace |
| TUNED_UNDORETENTION | Amount of time (in seconds) for which undo will not be recycled from the time it was committed |

# Sizing Undo Tablespace: Manual Method

- Use the following formula:
  - Tbs size = Required Retention period * undo block per second * block size
- Undo block per second can be obtained by:

```
SELECT MAX(UNDOBLKS/((END_TIME-BEGIN_TIME)*3600*24))
FROM V$UNDOSTAT;
```

# About using the Undo Advisor

- Can be used to estimate the undo tablespace size
  - It assumes that the required retention period is correctly set
- It relies on the AWR statistics
  - We must provide the AWR snapshot IDs (begin and end) that cover the period to analyze
  - To know the periods covered by which AWR snapshot ID:

```
SELECT SNAP_ID, BEGIN_INTERVAL_TIME, END_INTERVAL_TIME
FROM DBA_HIST_SNAPSHOT
ORDER BY SNAP_ID DESC;
```

- To use the advisor, create a task for it, execute the task, then check its findings and recommendations

# Sizing Undo Tablespace: Using Undo Advisor

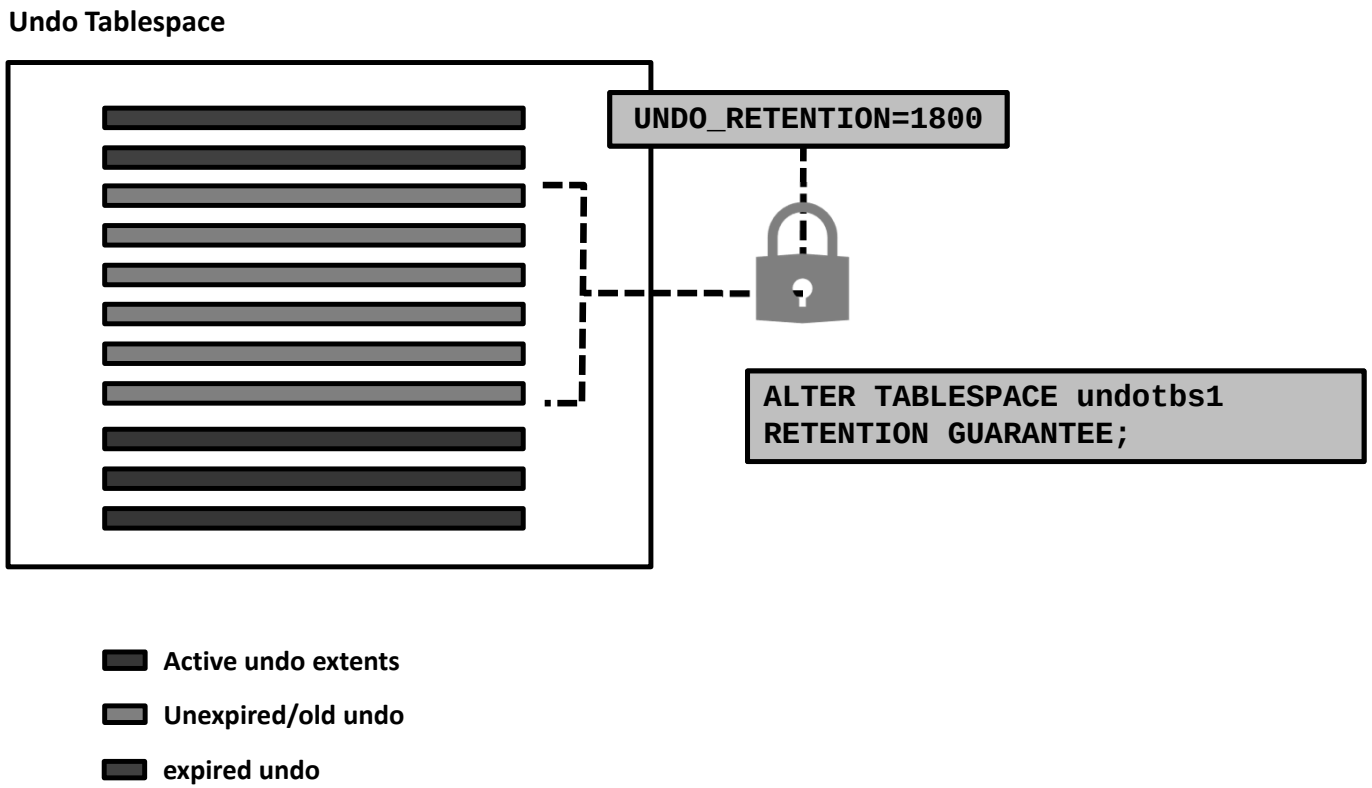
1. To creates an Undo Advisor task to evaluate the undo tablespace:

```
DECLARE
 tid NUMBER;
 tname VARCHAR2(30);
 oid NUMBER;
BEGIN
 DBMS_ADVISOR.CREATE_TASK('Undo Advisor', tid, tname, 'Undo Advisor
Task');
 DBMS_ADVISOR.CREATE_OBJECT(tname, 'UNDO_TBS', null, null, null,
'null', null, oid);
 DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'TARGET_OBJECTS', oid);
 DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'START_SNAPSHOT', 1);
 DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'END_SNAPSHOT', 2);
 DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'INSTANCE', 1);
 DBMS_ADVISOR.EXECUTE_TASK(tname);
END;
```

# Sizing Undo Tablespace: Using Undo Advisor

2. Query the advisor framework views:
   - **DBA_ADVISOR_TASKS, DBA_ADVISOR_OBJECTS, DBA_ADVISOR_FINDINGS, DBA_ADVISOR_RECOMMENDATIONS**

## Guaranteeing Undo Retention



# About Undo Retention Guarantee

- Used to guarantee the success of long-running queries or flashback operations
- When enabled, the database never overwrites unexpired undo data
  - Transactions may fail due to lack of space in the undo
- Can be enabled by setting the **RETENTION GUARANTEE** clause:

```
ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
CREATE UNDO TABLESPACE undotbs2 ... RETENTION GUARANTEE
```

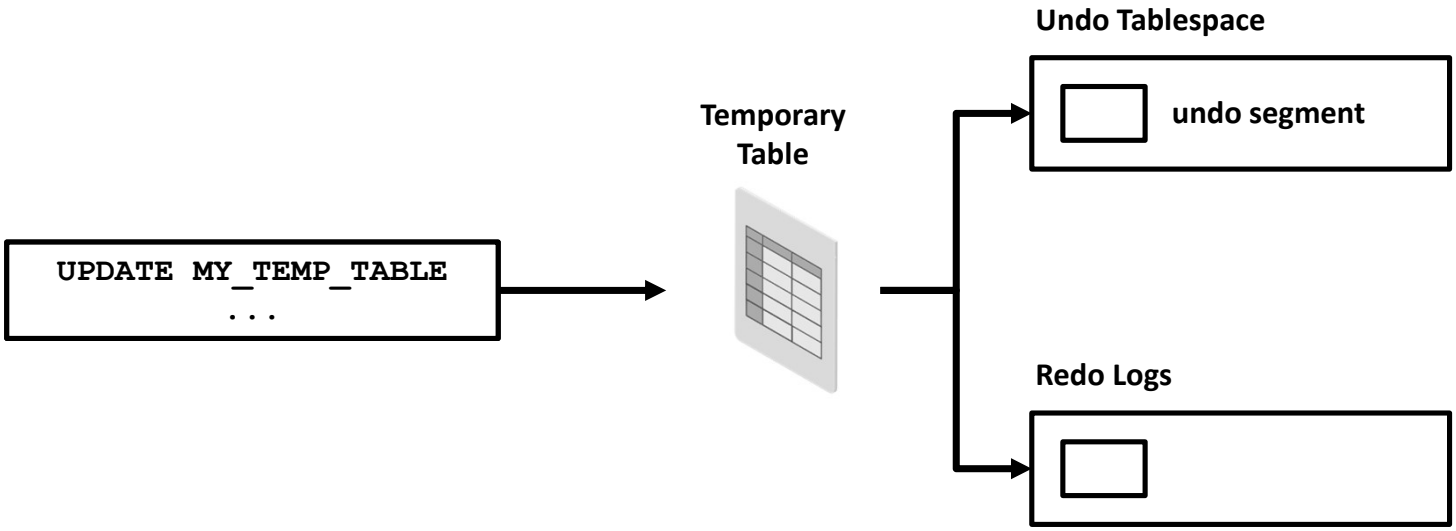
- Can be disabled:

```
ALTER TABLESPACE undotbs1 RETENTION NOGUARANTEE;
```

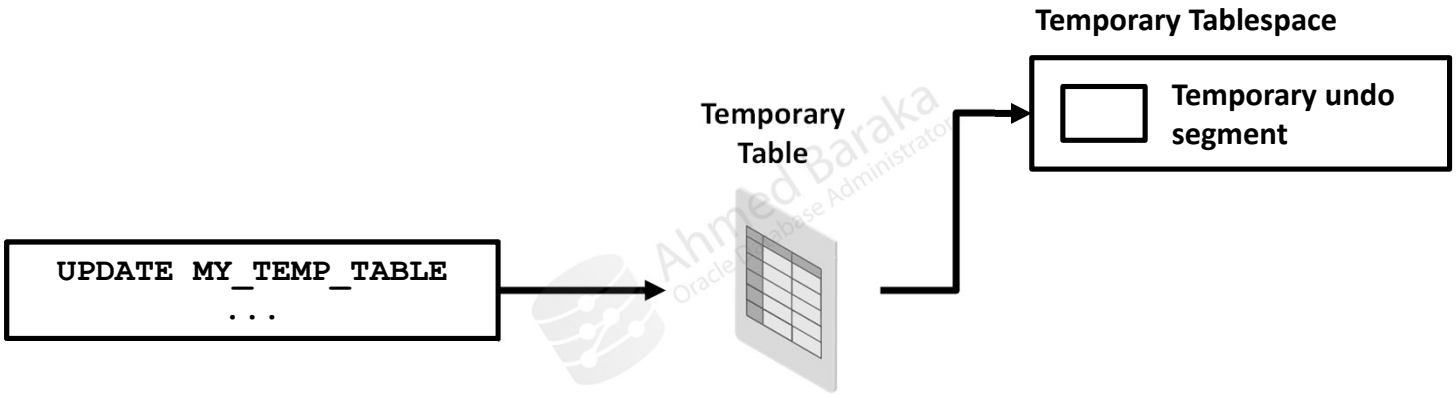
- Can be queried:

```
SELECT RETENTION FROM DBA_TABLESPACES WHERE TABLESPACE_NAME='..';
```

## Undo and Redo Generation when Temporary Undo is Disabled



## Temporary Undo is Enabled



# About Temporary Undo

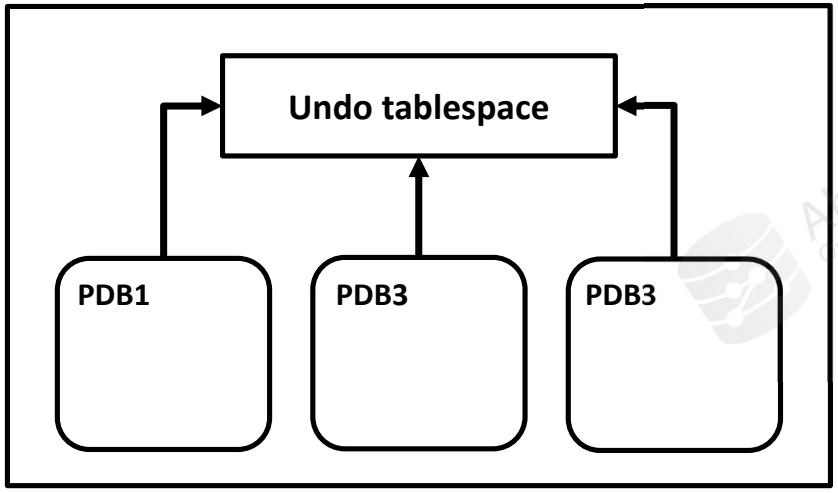
- By default, updating a temporary table generates undo segments in the undo tablespace and redo entries
- When the **temporary undo** is enabled, undo records for changes to temporary tables are saved in **temporary undo segments** in the temporary tablespace
- Available in 12c onwards
- Benefits:
  - Reduces the amount of undo stored in the undo tablespaces
  - Reduces the size of the generated redo logs

# Enabling Temporary Undo

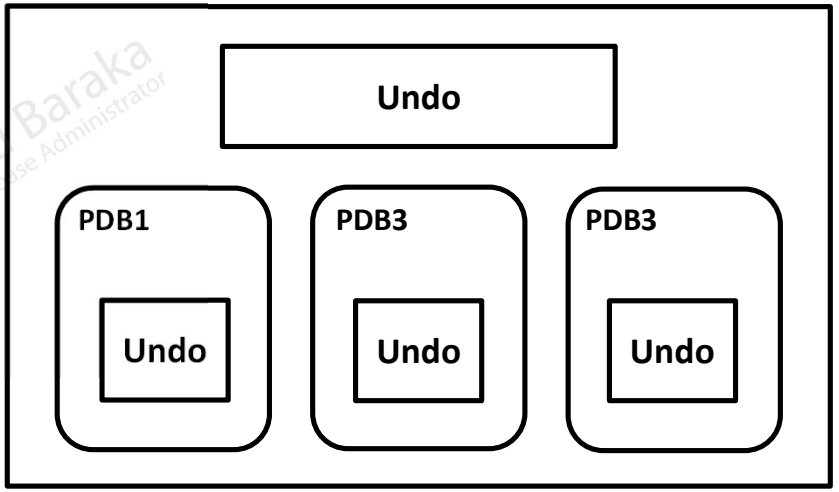
- Temporary undo is enabled by setting the parameter **`TEMP_UNDO_ENABLED`** to **`TRUE`** (default **`FALSE`**)
- Can be set at the system level and at the session level
- When a session uses temporary objects for the first time, the current value of the **`TEMP_UNDO_ENABLED`** initialization parameter is set for the rest of the session
- When enabled, the existing applications that make use of temporary objects run as is without any change

## Shared and Local Undo Modes

### CDB with Shared Undo




### CDB with Local Undo



# About CDB Undo Mode

- Possible modes: **Shared** or **Local**
  - **Shared:** an undo tablespace exists in the CDB only and all the PDBs use it
  - **Local:** every PDB has its own local undo tablespace
- You can change the undo mode of a CDB after it is created by issuing an **`ALTER DATABASE`** statement and restarting the CDB

```
STARTUP UPGRADE
ALTER DATABASE LOCAL UNDO ON | OFF
-- restart
```

- To know the current mode setting:

```
SELECT PROPERTY_NAME, PROPERTY_VALUE
FROM DATABASE_PROPERTIES
WHERE PROPERTY_NAME = 'LOCAL_UNDO_ENABLED';
```

# About Shared Undo Mode

- Only one active undo tablespace exists for a single-instance CDB
- Undo tablespaces can be created only in the CDB root and by a common user
- Visibility in undo dictionary views:
  - Current container is root: data dictionary + dynamic performance views (v$)
  - Current container is pdb: only dynamic performance views (v$)

# About Local Undo Mode

- Each container has its own undo tablespace for every instance
- Any user who has the appropriate privileges for the current container can create an undo tablespace for the container.
- Undo tablespaces are visible in data dictionary views and dynamic performance (V$) views in every container
- When a CDB is changed from shared undo mode to local undo mode, Oracle Database creates the required undo tablespaces automatically
- The preferred mode:
  - Increase efficiency of unplugging PDBs and PITR
  - A requirement for some operations, such as PDB relocation and cloning PDBs in read/write mode

# Summary

In this lecture, you will should have learnt how to perform the following:

- Describe undo benefits
- Describe undo space management tasks
- Set undo management parameters
- Set undo retention
- Tune undo data
- Use undo advisor
- Enable undo guarantee
- Enable temporary undo
- Enable shared or local undo modes