TEMA 7 - Parte 2
Administración de las estructuras físicas de almacenamiento

7.4. ADMINISTRACIÓN DE REDO LOGS.

- Recordando de temas anteriores, las estructuras Redo Log son cruciales para realizar la recuperación de una base de datos.
- Estas estructuras se almacenan en al menos 2 archivos que contienen todos los cambios realizados a la base de datos conforme ocurren.
- Se requieren al menos 2 para garantizar que siempre existirá un archivo disponible para escritura mientras que el otro esté en proceso de sincronización con data files o esté siendo archivado (mod archivelog).

7.4.1. Contenido de un Redo Log.

- Redo log files están formados por *Redo records*, llamados también *Redo entries.*
- Cada *Redo Record* está formado por un grupo de *vectores de cambios.*
- Cada *vector* describe los cambios realizados a un bloque de datos.

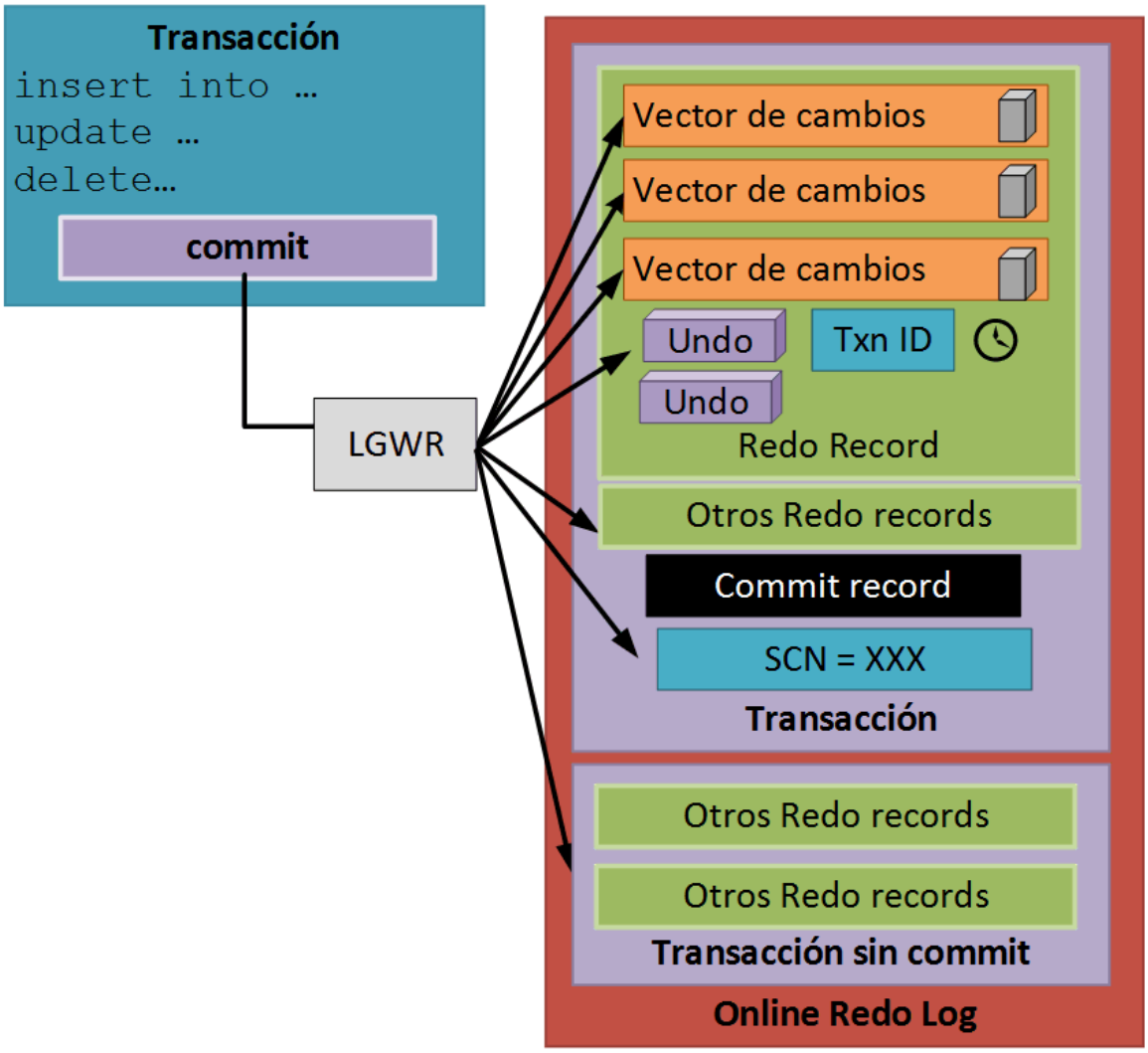
Ejemplo:

Suponer que se realiza un ajuste al campo salario de un empleado en la base de datos.

- Este cambio produce un *Redo Record* que contiene:
  - Vector de cambios que describen el cambio realizado al bloque de datos que pertenece a la tabla.
  - El bloque de datos del segmento *undo* asociado a la tabla (versión anterior del dato).
  - La tabla de transacciones del segmento *undo*. Esta tabla es una estructura de datos que se encuentra en un segmento *undo* y contiene todos los identificadores de las transacciones que están haciendo uso de dicho segmento.
  - Metadatos asociados con el cambio realizado:
    - Time stamp en el que ocurrió el cambio.
    - ID de la transacción que generó el cambio.
    - SCN y time stamp cuando la transacción hace commit (en caso de hacer commit).
    - Tipo de operación que realizó el cambio.
    - Nombre y tipo del segmento modificado.
- En un *redo Record* se guardan los datos necesarios para poder reconstruir todos los cambios realizados a la base de datos incluyendo los segmentos undo, empleados para realizar operaciones de rollback.
- Cuando se realiza la recuperación de una base de datos empleando Redo Logs, se realiza la lectura de los vectores de cambio contenidos en los *Redo Records* y los aplica en los bloques correspondientes.
- Los *redo records* son mantenidos en caché empleando un buffer circular empleando el *Redo Log Buffer* de la SGA.

- Cuando una transacción realiza `commit`, LGWR escribe el contenido del *Redo Log Buffer* hacia un *Redo Log file.* Adicionalmente se asigna un ***System change number (SCN)*** empleado para identificar a los *Redo Records* de una determinada transacción.
  - Cada transacción tiene asociado a un SCN.
  - Los cambios de una transacción comparten el mismo SCN.

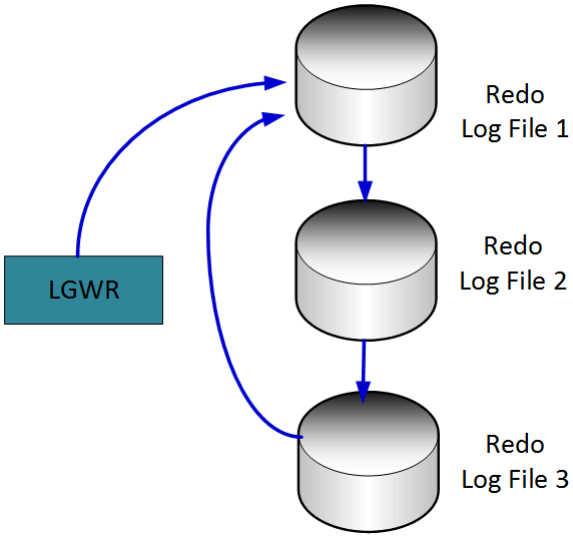
- Únicamente cuando todos los *Redo Records* asociados a una transacción son guardados de manera segura en los Online Redo Logs, el user process notifica que la transacción ha sido confirmada de forma exitosa.
- Recordando de temas anteriores, existen otros eventos que pueden provocar la escritura de Redo Records a los Online Redo Logs. Por ejemplo, cuando el buffer se llena, cuando otra transacción hace `commit`, etc.
- Lo anterior significa que pueden existir Redo Records almacenadas en los Online Redo Logs que pertenecen a transacciones que aún no hacen `commit`.



7.4.2. Algoritmo de escritura en Online Redo Logs.

- La base de datos requiere de al menos 2 Redo Log Files.
- Lo anterior es con el objetivo de garantizar que siempre existirá un archivo Redo Log disponible para realizar escrituras.

- Mientras que uno de los archivos recibe escrituras, el otro puede estar siendo respaldado o *archivado*, siempre y cuando la BD se encuentre en modo `archivelog`. Este concepto se revisará más adelante. Este mecanismo permite proteger posibles pérdidas de los Online Redo Logs.
- LGWR escribe en los Online Redo Logs en un estilo circular.
    - Comienza a escribir en el primer archivo. Cuando se llena, LGWR comienza a escribir en el siguiente Redo Log File disponible.
    - Cuando el último Redo Log File se llena, LGWR comienza a escribir en el primero cerrando así una especie de ciclo circular.



¿En qué momento un Redo Log File lleno estará listo para ser sobrescrito con nuevos cambios?

R: Depende si la base de datos está configurada en modo `archivelog`.

- Si el modo archivado no está habilitado (modo `noarchivelog`), un Redo log lleno estará disponible para ser sobrescrito hasta que su contenido haya sido sincronizado con los data files.
- Si el modo archivado está habilitado (modo `archivelog`) el redo Log estará disponible para sobrescritura cuando el contenido del archivo ha sido sincronizado con los data files, y de forma adicional, el Redo Log ha sido ***archivado.***
    - Los procesos encargados de realizar el archivado de Redo Logs se les conoce como *ARCn*.

7.4.3. Redo Log File Activo e Inactivo.

- En todo momento LGWR siempre escribe el contenido del log buffer cache a un solo Redo log. A este archivo se le conoce como ***Redo Log actual***.
- A los archivos Redo Log requeridos para realizar una recuperación de la base de datos, se les conoce como ***Redo Log Activos.***
- A los archivos Redo Log que ya no se requieren para realizar recuperación de base de datos, se les conoce como ***Redo Log Inactivos.***
- Los Redo Log activos NO pueden ser sobrescritos hasta que estos sean marcados como inactivos.

7.4.4. Log switches, Log sequence numbers

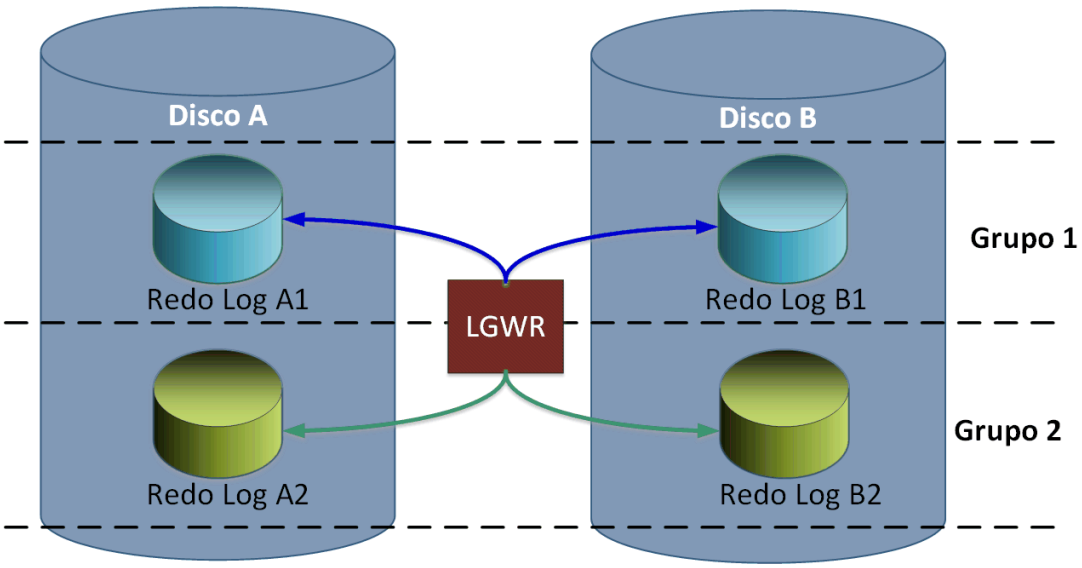
- Un log switch representa el punto en el cual la base de datos detiene la escritura de cambios en un Redo log y comienza a escribir en otro, generalmente ocurre cuando el Redo Log se ha llenado.

- Un Log switch puede programarse a intervalos de tiempo específicos o forzarlo de forma manual.
- Cada vez que ocurre un log switch, a cada Redo Log file donde se comienza a escribir se le asigna un nuevo *log sequence number.*
- Este número es conservado cuando el Redo Log es archivado.
- Por lo anterior, cada ***online redo log*** y cada ***archived redo log*** es únicamente identificado por su *log sequence number.*
- Cuando ocurre una falla, la recuperación de la base de datos aplica los cambios de los Redo Logs en orden ascendente empleando el *log sequence number.*

7.4.5. Multiplexado de Redo Logs.

- Mecanismo que protege fallas ocurridas en los Redo Log Files. ¿Qué pasaría si los Redo Log Files se pierden?
- En esta configuración existen 2 o más copias idénticas de un Redo Log las cuales pueden ser administradas de forma automática en ubicaciones diferentes.
- Para obtener el mayor beneficio, cada copia debe ubicarse en discos separados.
- Cuando el modo de multiplexado es habilitado, LGWR escribirá en cada una de las copias la misma información.
- Esta técnica evita el llamado "único punto de falla".
- Multiplexado de Redo Logs se realiza a través de la definición de ***grupos***.
- Cada grupo consiste de un Redo Log file y sus copias. A cada copia idéntica se le conoce como ***miembro.***
- Cada grupo cuenta con un número: 1,2, etc.
- Se recomienda ampliamente multiplexar Los archivos Redo Logs.
- Si se llegara a requerir una recuperación de base de datos, y si no existieran Redo Logs disponibles, el resultado sería muy serio: ***pérdida de datos.***
- En caso que LGWR no pueda escribir en alguno de los miembros, se genera una alerta y se marca al Redo log como `invalid`.

La siguiente figura muestra una configuración de 2 grupos, cada uno con 2 miembros. Generalmente se recomienda configurar 3 grupos, como mínimo se requieren 2 grupos.



7.4.6. Determinar el número de Redo Log Files.

- La mejor forma de detectar el número adecuado de Redo Logs es a través de realizar pruebas bajo diversos escenarios.
- La configuración más óptima es tener la cantidad mínima de grupos de Redo Logs de tal forma que no se sobrecargue la actividad de LGWR.
- En algunos escenarios 2 grupos es suficiente, en otros pueden requerirse más grupos para garantizar que siempre existirá un grupo disponible para realizar escrituras.
- El análisis de la bitácora del *LGWR* y el *alert log* de la base de datos son fundamentales para detectar si se tiene una mala configuración.
    - Si las bitácoras indican que LGWR tiene que esperar frecuentemente para que un grupo de redo log se libere, conviene agregar un nuevo grupo.
- Parámetros relacionados con el número máximo de redo Logs:
    - `maxlogfiles` empleado en la instrucción `create database`.
    - `maxlogmembers` empleado en la instrucción `create database`.

7.4.7. Planeación del tamaño de los Redo Logs.

- Primero, determinar si se realizará archivado de Redo Logs.
- El tamaño de los archivos debe ser tal que un grupo de archivos lleno debe ser archivado en una sola unidad de almacenamiento (disco o cinta), tratando de dejar la menor cantidad posible de espacio libre en la unidad de almacenamiento, o en su defecto, el tamaño de cada grupo archivado debe ser un múltiplo con respecto al espacio total del dispositivo.
- Todos los miembros del grupo deben tener el mismo tamaño.
- El tamaño mínimo de un Redo Log es de 4mb.
- Dependiendo el sistema operativo se define el tamaño del Redo Log por default.

7.4.8. Planeación del tamaño del bloque de datos para Redo Logs.

- Como se mencionó en otros temas, el tamaño del bloque coincide con el tamaño del sector físico en disco. Típicamente 512 bytes.
- Este es el valor recomendado para evitar el problema de redo wastage (explicado en temas anteriores).

```
select name, value from v$sysstat where name = 'redo wastage';
```

	NAME	VALUE
1	redo wastage	38804

7.4.9. Organización de Online Redo Logs, Data files y Archive Redo Logs.

La organización más recomendada implica las siguientes acciones:

- Data files deben ubicarse en discos diferentes con respecto a los discos donde se ubican los Online Redo logs.
- Online Redo Logs deben ubicarse en sus propios discos para evitar contención. Cada miembro del grupo debe ubicarse en discos físicos independientes.
- Archive Redo Logs, de forma similar, deben ubicarse en un siguiente disco físico e independiente. Esto evitará contención entre LGWR y los procesos ARCn.

7.4.10. Selección del número de archivos por grupo.

- La selección óptima requiere pruebas bajo diferentes escenarios.
- La configuración óptima debe tener el número menor posible de grupos y evitar así saturar al LGRW escribir demasiado en N grupos.
- Una forma de validar el número adecuado es realizando monitoreo de las bitácoras de LGRW.
    - Si se detecta que LGRW continuamente tiene que esperar para obtener un grupo disponible, por ejemplo, el checkpoint aún no termina, o un grupo aún no ha sido archivado, sería adecuado crear nuevos grupos.
- Los siguientes parámetros pueden emplearse para controlar límites:

    - `maxlogfiles`
    - `maxlogmembers`

7.4.11. Creando grupos y miembros.

Ejemplos:

- Agregar un nuevo grupo de Online Redo logs con 2 miembros, el tamaño del bloque es de 512 (no requerido especificarlo de forma explícita al ser el valor por default).

```
alter database
  add logfile
  group 4 ('/u01/logs/jrcbd2/redo04a.log','/u01/logs/orcl/redo04b.log')
  size 100m blocksize 512 reuse;
```

    - En este ejemplo se define el número de grupo el cual puede omitirse.
    - Si se especifica el número, debe ser entre 1 y `maxlogfiles`.
    - Los números de grupo deben ser consecutivos iniciando en 1.

- Agregar un nuevo grupo de Online Redo logs de 100 MB.

```
alter database
add logfile ('/u01/logs/jrcbd2/redo05a.log','/u01/logs/orcl/redo05b.log')
size 100m;
```

- Agregar un nuevo miembro a un grupo existente.

```
alter database
add logfile member '/u01/logs/jrcbd2/redo06c.log' to group 6;
```

- Es posible omitir el número de grupo. En este caso se deben listar todos los miembros:

```
alter database
add logfile member '/u01/logs/jrcbd2/redo06c.log' to
('/u01/logs/jrcbd2/redo06a.log','/u01/logs/orcl/redo06b.log')
```

7.4.12. Renombrar o modificar la ubicación de Online Redo Logs.

- Antes de realizar esta operación se recomienda realizar un respaldo de los data files.
- Posterior a realizar esta operación generar un backup del control file.

El procedimiento para realizar la reubicación se muestra a continuación:

1. Hacer shutdown.
2. Copiar los Redo Log files a su nueva ubicación.
3. Iniciar la instancia en modo `mount` (no abrir la BD)
4. Ejecutar la instrucción SQL correspondiente para realizar la reubicación de los archivos:

```
alter database
rename file
  '/u01/logs/jrcbd2/redo05a.log','/u01/logs/orcl/redo05b.log'
to
  '/u02/logs/jrcbd2/redo05aa.log','/u02/logs/jrcbd2/redo05bb.log';
```

5. Abrir la base de datos.

7.4.13. Eliminar grupos de Online Redo Logs.

- Para eliminar un grupo de Redo logs, este debe estar marcado como inactivo.
- Si se requiere eliminar el grupo de Online Redo log que está activo, se deberá forzar un log switch.
- Si el modo `archivelog` está habilitado, asegurarse que el grupo haya sido archivado.

Para conocer la situación actual del grupo que se desea eliminar, consultar la vista `v$log`.

Ejemplo:

```
select group#,sequence#,bytes, bytes/(1024*1024) size_mb, blocksize,
  members,archived,status
from v$log;
```

GROUP#	SEQUENCE#	BYTES	SIZE_MB	BLOCKSIZE	MEMBERS	ARCHIVED	STATUS
1	10	104857600	100	512	2	NO	CURRENT
2	8	104857600	100	512	2	NO	INACTIVE
3	9	104857600	100	512	2	NO	INACTIVE

Los posibles valores para el campo status son:

Status	Descripción
`unused`	Significa que nunca se ha escrito en este archivo. Esto se debe a que el archivo se acaba de agregar, o se ha ejecutado una instrucción para realizar reset en los logs (`resetlogs`).
`current`	El archivo está activo y es el archivo que actualmente se emplea para guardar los cambios de la BD.
`active`	El archivo está activo pero no es el actual. Esto significa que el archivo es requerido para aplicar instance recovery en caso de falla.
`clearing`	El archivo está siendo re-generado posterior a la ejecución de la sentencia alter database clear logfile. Al terminar la operación, su status se actualiza a UNUSED.
`clearing_current`	El archivo está siendo “limpiado” (su header está siendo actualizado) para iniciar su sobrescritura posterior a un log switch. Si ocurre una falla en este proceso, el archivo permanece con este status

Status	Descripción
inactive	El archivo ya no se requiere para realizar instance recovery, puede ser empleado para realizar media recovery, o puede ser archivado

Una vez que se han realizado las verificaciones anteriores, eliminar el grupo empleando la siguiente instrucción:

```
alter database drop logfile group 3;
```

- La instrucción no elimina a los archivos físicamente. Esta operación se debe realiza directamente a nivel de sistema operativo. Únicamente se actualiza el diccionario de datos y el control file.

7.4.14. Eliminar un miembro a un grupo de Online Redo Logs.

- Es posible eliminar un miembro de un grupo sin importar que los otros grupos permanezcan con un número diferente de grupos (grupos asimétricos).
- Lo anterior no se recomienda. Los grupos deberían tener el mismo número de miembros, por lo menos 2.
- La instancia requiere al menos 2 grupos, y por lo menos con un solo miembro aunque lo recomendable son al menos 2.
- Para poder eliminar un miembro, debe existir al menos otro miembro que esté identificado como activo (`status = null`).
- El miembro se puede eliminar siempre y cuando no forme parte de un grupo de Online Redo Log actual. Si se desea eliminar a un archivo que actualmente se está empleando, se debe forzar un log switch.
- Para mostrar la situación de los miembros de los online Redo Logs, emplear la siguiente consulta:

```
select * from v$logfile;
```

GROUP#	STATUS	TYPE	MEMBER	IS_RECOVERY_DEST_FILE	CON_ID
3	(null)	ONLINE	/u01/app/oracle/oradata/JRCBD2/redo03a.log	NO	0
3	(null)	ONLINE	/u01/app/oracle/oradata/JRCBD2/redo03b.log	NO	0
2	(null)	ONLINE	/u01/app/oracle/oradata/JRCBD2/redo02a.log	NO	0
2	(null)	ONLINE	/u01/app/oracle/oradata/JRCBD2/redo02b.log	NO	0
1	(null)	ONLINE	/u01/app/oracle/oradata/JRCBD2/redo01a.log	NO	0
1	(null)	ONLINE	/u01/app/oracle/oradata/JRCBD2/redo01b.log	NO	0

- `Is_recovery_dest_file` Indica si el archivo fue creado en el fast recovery area.
- Los posibles valores del campo status son:

Status	Descripción
invalid	El archivo es inaccesible
stale	El archivo está corrupto o incompleto
deleted	El archivo ya no se está empleando
null	El archivo está en uso.

Posterior a las verificaciones, ejecutar la siguiente instrucción para realizar la eliminación del miembro:

```
alter database drop logfile member '/u02/logs/jrcbd2/redo05bb.log ';
```

7.4.15. Forzar Log switch.

Se emplea la siguiente instrucción:

```
alter system switch logfile;
```

Generalmente se emplea para realizar las operaciones antes descritas: Eliminar/modificar un grupo de archivos o eliminar un miembro.

7.4.16. Realizar limpieza de archivos Redo Log.

- En algunas situaciones, un miembro de un Redo Log puede volverse corrupto mientras la base de datos está abierta.
- En este caso, eventualmente la base de datos podría detener su operación, en especial si este archivo no se puede archivar debido a que se encuentra corrupto.
- Bajo esta situación, el archivo puede ser reinicializado con la siguiente instrucción:

```
alter database clear logfile group 3;
```

- Si el archivo no ha sido archivado, incluir la cláusula `unarchived`:

```
alter database clear unarchived logfile group 3;
```

- La instrucción anterior reinicializa y evita su archivado.
- OJO: Si se reinicia un archivo que se necesita para realizar recovery de un backup, dicho backup no podrá ser recuperado.
- TIP: Si se realiza el reinicio de un Redo Log no archivado, se debe realizar un ***nuevo respaldo*** de la BD.
- Otro aspecto importante a considerar cuando se intente hacer limpieza de un redo log es la existencia de tablespace en modo offline.
- Recordando temas anteriores, un tablespace offline puede requerir de un ***recovery*** para poner ser puesto en modo online nuevamente.
- Si se desea limpiar un Redo log que es necesario para pasar un tablespace de modo offline a modo online, emplear la cláusula `unrecoverable datafile` dentro de la cláusula `alter database clear logfile`.
- La instrucción anterior tiene un efecto: El tablespace no podrá recuperarse y por lo tanto no podrá pasarse a modo online. La única opción es eliminar el tablespace o realizar un ***incomplete recovery.***

7.4.17. Comportamiento de la BD ante la presencia de fallas en los Redo Logs.

Condición	Acciones del LGWR
LGRW tuvo éxito al escribir en al menos un miembro del grupo	La base de datos sigue operando normalmente, ignora a los miembros no disponibles. Se escribe un mensaje de error en el archivo log del LGWR y se genera una alerta.
LGWR no puede acceder al siguiente grupo durante un Log Switch debido a que el grupo debe ser archivado	La operación de la base de datos se detiene momentáneamente hasta que el archivo log sea archivado.

Condición	Acciones del LGWR
Todos los miembros son inaccesibles debido a una falla de hardware	• Regresa un mensaje de error y se realiza un shutdown. • Este evento puede requerir realizar una recuperación (media recovery). • En algunos casos no es necesario realizar media recovery, bastará con eliminar el grupo de Redo Log inaccesible. • Se emplea `alter database clear logfile unarchived`. Esta instrucción permite deshabilitar el archivado antes que el grupo sea eliminado.
Todos los miembros se vuelven inaccesibles de forma repentina mientras LGRW se encontraba escribiendo	• Regresa un mensaje de error y se realiza shutdown. • Esta situación puede requerir recuperación de datos. • Si la falla fue debido a una falta de corriente eléctrica, pero los archivos siguen estando disponibles, bastará con restaurarlos y aplicar una recuperación automática.

7.4.18. Vistas del diccionario de datos asociadas con Redo Logs.

En secciones anteriores se han mencionado las vistas que muestran datos de los Online Redo logs. En resumen son las siguientes:

Vista	Descripción
`v$log`	Muestra información de los Online Redo Logs contenida en el archivo de control.
`v$logfile`	Identifica a cada uno de los grupos de Redo Logs, a sus miembros y a sus status.
`v$loghistory`	Muestra información histórica de los Online Redo Logs.

Ejercicio 1

Suponer que se ha decidido modificar el tamaño de los Online Redo Logs. Originalmente se cuentan con archivos de 100MB. Por cuestiones de espacio en disco, en nuevo tamaño será de 50MB. En general la estrategia para realizar esta actividad es crear nuevos grupos y eliminar los existentes.

A. A nivel de sistema operativo, ejecutar un ***solo comando*** para mostrar la ruta completa y el tamaño actual de todos los Online Redo logs. Recordar que estos se encuentran en los directorios `/u01, /u02, /u03`. Tip: usar el comando `find`.
B. Generar una consulta que muestre los siguientes datos de los grupos de Redo Logs. Número de grupo, número de secuencia, tamaño en MB, tamaño de su bloque, número de miembros, status del grupo, número SCN menor que contiene el grupo de Redo Log, fecha hasta segundos en el que se guardó el menor SCN, número SCN máximo que contiene el grupo SCN.
C. Con base a la consulta anterior, ¿Qué grupo de Redo Log es el que se está empleando?
D. Generar una consulta SQL para mostrar los datos de los miembros de cada grupo de Online Redo Logs. Número de grupo, status, tipo, ruta absoluta donde se encuentra el archivo. Explicar:

- ¿Por qué razón el status de los miembros es nulo?
- ¿Qué otros valores del campo status pueden existir?

E. Crear 3 nuevos grupos de Online Redo Logs. Su tamaño será de ***50 MB*** con un tamaño de bloque de **512**. Agregar únicamente 2 miembros. Emplear la siguientes ubicaciones ,nombres y número de grupo indicado (4,5 y 6)

Num. grupo	Ubicaciones y nombres
4	/u01/app/oracle/oradata/<ORACLE_SID>/redo01_**A.**log /u01/app/oracle/oradata/<ORACLE_SID>/redo01_**B.**log
5	/u01/app/oracle/oradata/<ORACLE_SID>/redo02_**A.**log /u01/app/oracle/oradata/<ORACLE_SID>/redo02_**B.**log
6	/u01/app/oracle/oradata/<ORACLE_SID>/redo03_**A.**log /u01/app/oracle/oradata/<ORACLE_SID>/redo03_**B.**log

F. Agregar un tercer miembro a cada uno de los grupos anteriores empleando `alter database add logfile member`. Emplear las siguientes ubicaciones y nombres de archivo:

Num. grupo	Ubicaciones y nombres
4	/u01/app/oracle/oradata/<ORACLE_SID>/redo01_**C.**log
5	/u01/app/oracle/oradata/<ORACLE_SID>/redo02_**C.**log
6	/u01/app/oracle/oradata/<ORACLE_SID>/redo03_**C.**log

G. Ejecutar nuevamente la consulta del inciso B, describir las diferencias.

H. Ejecutar nuevamente la consulta del inciso D. Observar los status de los miembros, en especial los agregados en el inciso F. Explicar los valores.

I. Notar que uno de los 3 grupos de Redo Logs anteriores está marcado como el archivo que actualmente se está empleando. Para poder eliminar a los 3 grupos anteriores, el Redo Log actual debe ser alguno de los nuevos grupos. Generar las sentencias necesarias para que el grupo de Redo Log actual sea el grupo 4. Mostrar nuevamente la consulta del inciso G para verificar los resultados.

J. Una vez que el Redo Log actual sea el grupo 4, los grupos 1 a 3 podrían ya eliminarse. Sin embargo, es posible que alguno de estos 3 grupos tenga el status ACTIVE. Recordando su significado, este status indica que el Redo Log file será requerido en caso de recovery. Ejecutar las sentencias SQL necesarias para eliminar esta dependencia de tal forma que los Redo Log Files de los grupos 1 a 3 puedan eliminarse de forma segura. ¿En qué status deberían estar para poder ser eliminados de forma segura?, Ejecutar nuevamente la consulta del inciso anterior para confirmar los cambios.

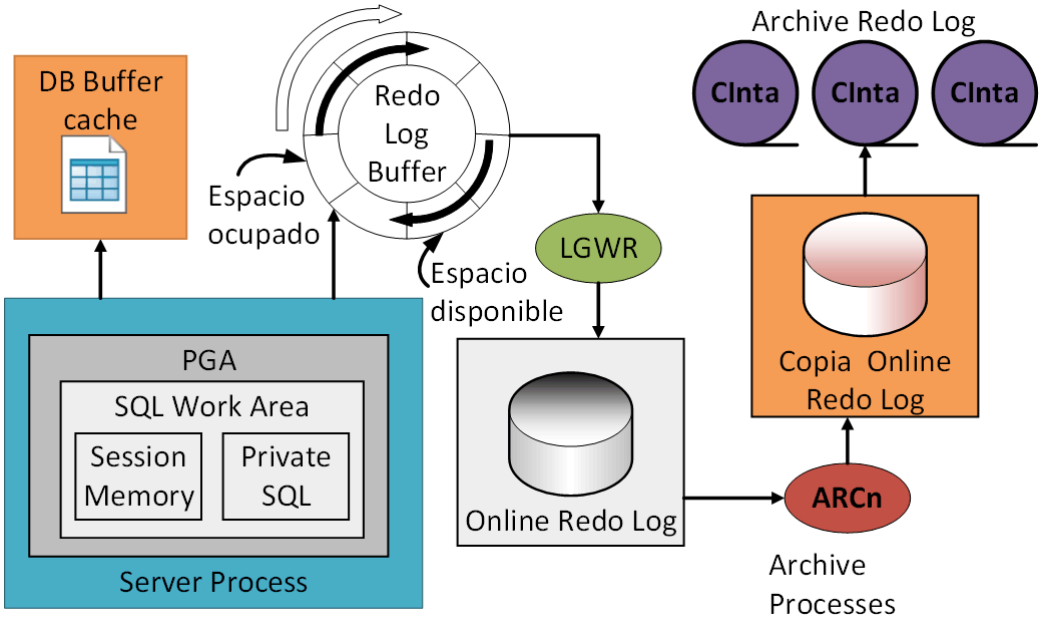
K. Ejecutar las sentencias correspondientes para eliminar los grupos 1 a 3. Importante: Asegurarse que estos grupos tengan el status adecuado determinado en el inciso anterior. Ejecutar nuevamente la consulta anterior para verificar resultados.

L. Notar que la eliminación de estos grupos no implica la eliminación de los archivos del sistema operativo. Por lo tanto, ejecutar los comandos correspondientes en Linux para eliminar los archivos redo Logs anteriores. En una BD productiva, antes de ejecutar este paso se debería realizar un ***backup completo***.

M. Ejecutar nuevamente el comando en sistema operativo del inciso A para revisar los archivos Redo Log existentes posterior a esta actividad.

7.5. Administración de Archived Redo Logs.

- Cuando un grupo de archivos Redo Logs se ha llenado, existe la posibilidad de respaldarlo en algún medio de almacenamiento, generalmente externo.
- A este grupo de archivos respaldado se le conoce como ***Archive Redo Log***.
- Al proceso de respaldar un Redo Log y convertirlo en un archive redo log se le conoce como ***archiving***.
- Este proceso se realiza únicamente si la base de datos se encuentra configurada en modo `archivelog`.
- Existe la posibilidad de realizar archivado manual y automático.
- Un Archive Redo Log es una copia de un Redo Log ***lleno***. La copia contiene todos los *log entries (redo records)* así como el *sequence number* del Redo Log (atributo `sequence#` de `v$log`). Este número permite asociar a un *archive redo log* con su correspondiente *Redo Log*.
- Los procesos ARCn (Archive processes) son los encargados de realizar estas copias.



- La copia se realiza tomando alguno de los miembros del grupo. Si un miembro está corrupto, la copia se realiza empleando otro miembro del grupo.
- Al habilitar el archivado, LGWR ***no podrá*** sobrescribir el grupo de Redo Logs hasta que este haya sido archivado.
- Por lo contrario, si el modo de archivado no está habilitado, el grupo de Redo Log puede ser sobrescrito cuando este adquiere el status de inactivo, (cambios han sido sincronizados con los data files).
- Una vez que el modo `archivelog` es habilitado, en archivo de control guarda la información necesaria para evitar que un Redo Log sea sobrescrito por LGWR hasta que el grupo sea archivado.
- El grupo de Redo Logs estará disponible para ser archivado justo después de la ocurrencia de un log switch.

7.5.1.1. Beneficios de un Archive Redo log:

- Recuperar una base de datos, en especial posterior a la pérdida de un disco, etc.
- Actualizar una base de datos de respaldo que ha estado fuera de sincronía.

- Obtener información histórica de la base de datos empelando alguna herramienta, por ejemplo LogMiner.
- El modo `noarchivelog` protege a la base de datos en caso de una falla a nivel de la instancia ***instance failure***, pero no protege de una falla en los medos de almacenamiento: ***media failure.***
- Únicamente los cambios más recientes aplicados a la BD mismos que se encuentran en los Online Redo Logs están disponibles para realizar *instance recovery.*
- El modo `archivelog`, protege a la base de datos de fallas en los medios de almacenamiento.
- Como buena práctica se recomienda almacenar los Archive Redo Logs y backups en medios de almacenamiento llamados *offline storage media*, por ejemplo, en cintas.

Ejemplo:

¿Qué sucede si la BD se encuentra en modo `noarchivelog` y alguno de los discos donde se encuentran los data files se pierden?

- La única opción es recuperar la base de datos empleando el último backup realizado. Pero, ¿qué pasa con los cambios generados posterior al último respaldo?
- Para estos casos el modo de archivado permite recuperar todos estos cambios.

¿Qué relación existe entre los backups y el modo `archivelog` ?

- En este modo, es posible realizar respaldos de la base de datos sin que esta sea detenida: online backups.

7.5.2. Procedimiento para cambiar una base de datos a modo archivelog.

1. Detener la instancia de forma ordenada para evitar *instance recovery.*
2. Realizar un respaldo completo de la BD (para efectos del curso este paso podría ser omitido ya que aún no se revisa el tema correspondiente).
3. Editar el archivo de parámetros para configurar los parámetros necesarios (ver sección *Configuración de parámetros para habilitar el modo de archivado*).
4. Iniciar la instancia en modo `mount`.
5. Ejecutar las siguientes instrucciones para habilitar el modo `archivelog`.

```
alter database archivelog;
alter database open;
```

6. Detener la instancia.
7. Realizar un backup de la base de datos. Para efectos del curso, este paso se puede omitir ya que aún no se cuentan con respaldos.

Al pasar a la base de datos en modo `archivelog`, se actualiza el control file, y todos los backups realizados en modo `noarchivemode` se vuelven inusables a partir del cambio a modo `archivelog`. De aquí la importancia de realizar un backup completo.

8. Abrir la base de datos.
9. Comprobar que el modo `archivelog` fue habilitado. La forma más sencilla es ejecutando la siguiente instrucción:

```
idle> archive log list
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            /unam-bda/archivelogs/JRCBDA2/disk_b
```

```
Oldest online log sequence     7
Next log sequence to archive   9
Current log sequence           9
```

7.5.3. Configuración de parámetros para habilitar el modo de archivado.

Los siguientes pasos deben realizarse como parte de la actividad 3 listada anteriormente:

- Configurar el(las) ubicación(es) donde se almacenará(n) los *archive redo logs*. Algunas opciones:
  - Sistema de archivos local.
  - Grupo de discos empleando ASM (Automatic Storage Management).
  - Ubicación remota (BD remota tipo standby).
  - Hacer uso de la Fast Recovery Area.
- Es posible especificar más de una ubicación para generar varias copias y evitar posibles problemas de fallas en medios.
- Configurar el número de procesos ARCn que serán empleados para realizar el proceso de archivado.
- Configurar el formato del nombre de los *archive redo logs.*

La siguiente tabla muestra los parámetros a configurar y sus posibles valores.

Parámetro	Descripción
`log_archive_max_processes`	• Empleado para configurar el número de procesos ARCn requeridos para realizar el archivado de Redo logs. • Por default se crean 4 procesos, se pueden configurar hasta 30.
`log_archive_format`	• Indica el formato que será empleado para nombrar a los archivos. Los siguientes patrones pueden ser empleados. o `%t` Es el número de hijo de ejecución (obligatorio) o `%s` es un número secuencial que indica el número de archivo (obligatorio), o `%r` identificador que asegura la generación de valores únicos. o `%d` ID de la base de datos. Ejemplo: `log_archive_format = arch_%t_%s_%r.arc`
`log_archive_trace`	Especifica un número entero para configurar el nivel de detalle que se guardará en las bitácoras que describen el proceso de archivado. Ejemplo: `alter system set log_archive_trace=12;`
Parámetros empleados para configurar las ubicaciones.	
`log_archive_dest`	Permite configurar una sola ubicación. Este parámetro junto con el parámetro `log_archive_duplex_dest` se ha marcado como deprecated, se prefiere hacer uso de `log_archive_dest_n`
`log_archive_dest_n`	• Permite configurar hasta 30 ubicaciones diferentes. N= [1,30]. • El destino puede ser una ubicación o un servicio. • La vista `v$archive_dest` contiene la lista de ubicaciones configuradas por este parámetro. • Existen diversos atributos que pueden configurarse con este parámetro. Uno de ellos es el atributo `mandatory`. Si se incluye, esta copia debe ser generada con éxito para considerar al proceso de archivado como exitoso. Ejemplos:

	`log_archive_dest_1='LOCATION=/disk1/arch MANDATORY'`
	`log_archive_dest_2='LOCATION=/disk2/arch'`
	
	En la configuración anterior se generarán 2 copias. La primera debe ser exitosa (obligatoria) y la segunda puede omitirse si llegara a producirse un error al realizar el proceso de archivado.

`log_archive_min_succeed_dest`	• Indica el número mínimo de copias que deben generarse con éxito para considerar al proceso de archivado como exitoso, y por lo tanto, el Redo log podrá ser sobrescrito por LGWR.

Importante: Garantizar que el destino tenga el suficiente espacio. De otra forma la BD pude dejar de operar.

Ejercicio 2:

Configurar la base de datos para habilitar el modo `archivelog`. Considerar los siguientes requerimientos:

A. Crear las siguientes estructuras de directorios para simular la existencia de 2 discos de respaldo para archive Redo Logs. Asegurarse que solo el usuario `oracle` tengan privilegios de lectura, escritura y ejecución. A nivel de grupo solo se permiten lecturas y permiso de ejecución. Estos permisos deben aplicarse a partir de la carpeta `archivelogs`. Los demás usuarios no deben tener ningún tipo de permiso. El valor de `<ORACLE_SID>` debe aparecer en mayúsculas.

- `/unam-bda/archivelogs/<ORACLE_SID>/disk_a`
- `/unam-bda/archivelogs/<ORACLE_SID>/disk_b`

B. Realizar la configuración en el archivo de parámetros de forma permanente para preparar la activación del modo `archivelog`. Antes de aplicar los cambios al archivo de parámetros, generar un respaldo (generar un `pfile` a partir del `spfile`). Recordando: antes de aplicar cambios al spfile se debe realizar un respaldo. La forma más sencilla, creando un pfile.

- Configurar 5 procesos ARCn encargados de realizar el proceso de archivado.
- Configurar la creación de 2 copias en cada proceso de archivado. Emplear las siguientes 2 rutas para simular 2 discos de respaldos.
  - `/u01/unam-bda/archivelogs/<ORACLE_SID>/disk_a`
  - `/u01/unam-bda/archivelogs/<ORACLE_SID>/disk_b`
- Realizar las configuraciones necesarias para que la copia que se realiza en `disk_a` sea considerada como obligatoria
- Emplear el siguiente formato para nombrar a los archivos: `arch_<oracle_sid>_%t_%s_%r.arc`. Notar que el valor de `<oracle_sid>` debe ir en minúsculas.
- Configurar el parámetro correspondiente para asegurar que al menos una copia se genere con éxito para considerar al proceso de archivado como exitoso.

C. Detener/Iniciar la instancia en el modo correspondiente para activar el modo de archivado

D. Abrir la base de datos. Ejecutar la instrucción `archive log list` para comprobar la habilitación, Incluir la salida del comando.

E. Respaldar nuevamente el spfile empleando un pfile.

- *Incluir instrucciones SQL y resultados de ejecución.*

7.5.4. Archivado Manual

Generalmente el modo archivado automático es el mejor, sin embargo es posible configurar el modo archivado para realizarse de forma manual.

- En este modo, el archivado ***debe*** realizarse con grupos de Redo Logs inactivos y llenos. De otra forma, si se lanza el archivado con grupos activos, la operación de la BD se detendrá mientras se realiza el archivado.
- El procedimiento para habilitar el modo archivado manual es similar a los pasos descritos anteriormente con la diferencia de la sentencia alter database archivelog. Se agrega la opción manual.

```
alter database archivelog manual;
```

- Para realizar el archivado manual se emplea:

```
alter system archive log all;
```

7.5.5. Vistas del diccionario de datos asociadas con Archive Redo Logs.

Vista	Descripción
`v$database`	Indica si la base de datos se encuentra en modo noarchivelog o archivelog así como si el modo de archivado manual está activado.
`v$archived_log`	Contiene información histórica de los procesos de archivado configurados en el archivo de control.
`v$archive_dest`	Describe principalmente las ubicaciones de destino donde se almacenan los redo logs archivados.
`v$archive_processes`	Describe el status de los procesos encargados de realizar el archivado.

Ejercicio 3:

A. Considerar la vista `v$database`. Generar una sentencia que muestre el nombre de la instancia y el atributo correspondiente que confirme el modo `archivelog` configurado en el ejercicio anterior.
B. Generar una consulta SQL que muestre los datos de las ubicaciones donde serán almacenados los Archived Redo Logs. Mostrar identificador de la ubicación, nombre de la ubicación, status, indicar si la generación es opcional u obligatoria, la ruta absoluta donde se guardarán los archivos. Mostrar únicamente los datos de las 2 ubicaciones configuradas en el ejercicio anterior.
C. Generar una consulta que muestre todos los datos de los Redo Logs (`v$log`). Los campos `first_time` y `next_time` deben mostrarse a nivel de segundos. Con base a la información obtenida, contestar las siguientes preguntas:

- ¿Cuál es el número de grupo en el que actualmente se está escribiendo?
- ¿Qué número de secuencia tiene cada grupo?
- ¿Qué valor tiene el número de secuencia menor?

- ¿Qué significado y relación tienen las columnas `first_change# y next_change#` ?

D. Considerar la vista `v$log_history`. Esta vista contiene información histórica de los log switches que se han realizado. A cada log switch se le asocia un número de secuencia. Generar una consulta que muestre todos los datos de esta vista a partir del valor del número de secuencia menor encontrado en el inciso. Los campos `first_time` y `resetlogs_time` deben mostrarse a nivel de segundos. Ordenar la consulta con base al campo `first_time` el cual indica la fecha y hora en la que se escribió por primera vez en el grupo de Redo Logs.

E. Consultar la vista `v$archived_log`. En ella se realiza el registro de todos los Redo Logs archivados. Debido a que el modo *archivelog* fue activado en el ejercicio anterior, muy probablemente esta vista estará vacía. Para provocar que se realice archivado, se deberá provocar log switches. Considerando el usuario creado en temas anteriores, crear una tabla con la siguiente estructura:

```
create table <iniciales>_cadena (
  cadena varchar2(1024)
);
```

Crear un programa que inserte cierta cantidad de registros para provocar un log switch. Recordando de ejercicios anteriores, el tamaño de los Online Redo logs es de 50MB. Se recomienda insertar 20MB aprox. Crear cadenas de 1KB (1024). Emplear la función `dbms_random.string` Esta función permite generar cadenas aleatorias de cierta longitud. Generar cadenas alfanuméricas en mayúsculas. Consultar la documentación de la base de datos para mayores detalles. Hacer `commit` al final. Al terminar de ejecutar el programa, consultar nuevamente la vista para verificar si se ha realizado un proceso de archivado. Ejecutar el programa las veces que sea necesario para provocar que los 3 grupos de Redo Logs sean archivados. Incluir el código del programa.

- Considerando nuevamente la vista `v$archived_log` , Incluir una sentencia SQL y su salida que demuestre el archivado de los 3 grupos. Incluir el id de la copia (record id), la ruta completa donde se almacenó la copia, el número de ubicación, el número de secuencia, la fecha en la que se escribió el primer SCN en el archive Log, status, fecha en la que se completó el archivado. Las fechas deberán estar hasta segundos.

F. ¿Cuántos registros se insertaron en la tabla anterior para provocar el archivado de los 3 grupos de Redo?

G. Ejecutando una sola instrucción a nivel de Sistema operativo, mostrar la lista de archivos Redo Log generados. La lista debe incluir su ruta completa y la fecha en la que se generaron.

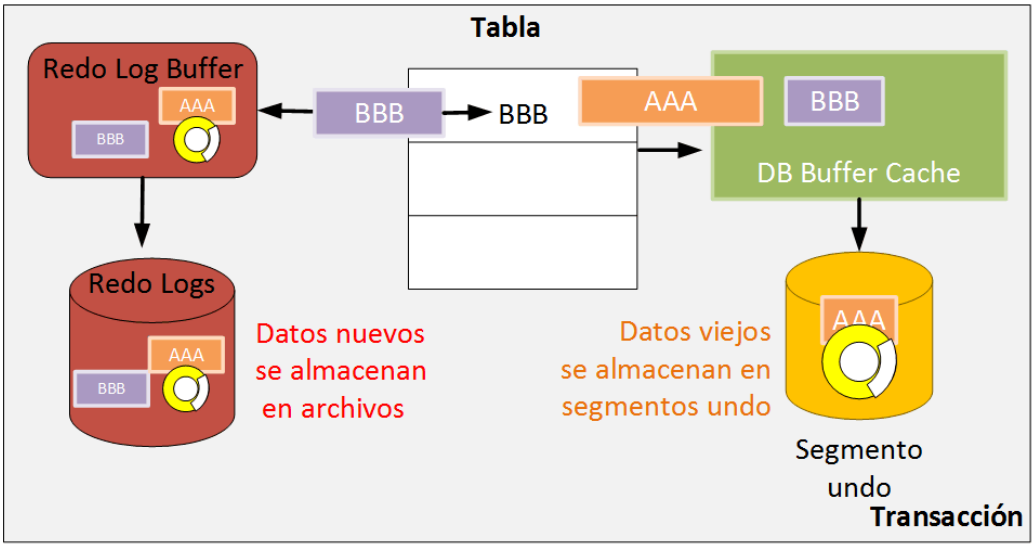
H. Generar una consulta SQL que muestre el número de log swithes que se han realizado a partir del número de secuencia menor obtenido en incisos anteriores.

I. Generar una consulta SQL que muestre el número de log switches que se han realizado el día de hoy.

- *Incluir sentencias SQL, comandos del sistema operativo, resultado de su ejecución.*

7.6. Administración de Datos Undo.

- Cuando una transacción realiza cambios a los datos a través de operaciones DML la base de datos guarda la versión anterior o valores viejos (undo data).
- Estos datos se utilizan para implementar diversas funcionalidades
  - Hacer Rollback de una transacción
    - Empleado para deshacer cambios que fueron actualizados por una transacción.
  - Realizar la recuperación de una base de datos.
    - Empleado para deshacer cambios aplicados de los Redo Logs hacia los data files.
  - Implementar la propiedad ACID de las transacciones: lecturas confirmadas, lecturas repetibles en general, "consistencia de datos (read consistency)".
  - Analizar datos en un instante del tiempo pasado empleando Oracle Flashback Query
  - Realizar recuperaciones de corrupción de datos empleando Oracle flashback.



- En la imagen anterior se ilustra la relación entre datos REDO y UNDO. Suponer que una transacción realiza operaciones update sobre la tabla mostrada. Los cambios realizados junto con datos undo son almacenados en los Redo Logs. Por otro lado, los datos viejos o versión anterior es almacenada empleando segmentos undo.
- Cuando una transacción inicia, se le asigna un único segmento Undo para poder almacenar *datos undo.*
- Un segmento Undo puede ser empleado para varias transacciones.
- Cuando la transacción modifica un dato, la versión vieja se copia a este segmento.
- La vista `v$transaction` contiene entre otros datos, el segmento *undo* que se le asigna a cada transacción.

Ejemplo:

- Suponer que 3 sesiones ejecutan las siguientes instrucciones:

```
--sesión 1
set transaction name 't1';
insert into jrc_cadena values('x');

--sesión 2
set transaction name 't2';
insert into jrc_cadena values('y');
```

```
--sesión 3
set transaction name 't1';
insert into jrc_cadena values('z');
```

Generar una consulta que muestre: El identificador de la transacción, número de segmento undo asignado a la transacción, nombre, status, fecha de inicio, número de bloques empleados del segmento undo, número de registros undo empleados, SCN inicial.

```
select xid,xidusn,name,status,start_time,used_ublk,used_urec,start_scn
from v$transaction;
```

	XID	XIDUSN	NAME	STATUS	START_TIME	USED_UBLK	USED_UREC	START_SCN
1	03002100A3040000	3	t2	ACTIVE	05/09/20 17:21:18	1	1	2442637
2	02000F0097040000	2	t3	ACTIVE	05/09/20 17:21:49	1	1	2442666
3	04001500A3040000	4	t1	ACTIVE	05/09/20 17:20:36	1	1	2442582

- Segmentos undo son similares a los segmentos empleados por tablas permanentes, formados por extensiones, bloques.
- Los segmentos crecen y se reutilizan, lo que produce un algoritmo circular.
- Una transacción puede requerir de varios segmentos undo, tantos como requiera para completar sus operaciones.

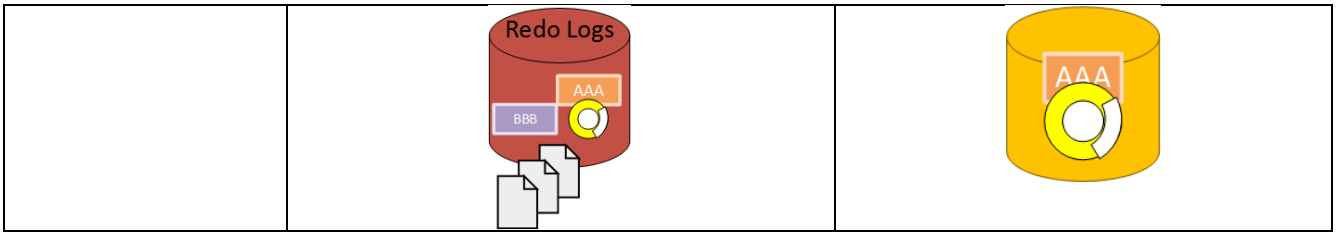
7.6.1. Undo tablespaces.

- Empleados para almacenar segmentos undo.
- Cuentan con mecanismos especiales para realizar recovery.
- Solo se requiere un tablespace por instancia.
- Para sistemas OLTP donde hay una gran cantidad de transacciones concurrentes, un tablespace undo formado por varios data files ofrece mejor desempeño respecto a un big file tablespace principalmente para evitar contención cuando múltiples transacciones acceden al header del data file para actualizar metadatos.
- Debido a que el segmento undo es circular (sus extensiones se pueden reutilizar), este debe contar con al menos 2 extensiones.

7.6.2. Comparación Undo Vs Redo.

La siguiente tabla ilustra las similitudes y diferencias entre los datos Undo y los Datos Redo.

	Undo	**Redo**
Empleado para:	Deshacer un cambio.	Re-hacer un cambio
Usado en:	• Rollback de una txn • En consultas que requieren niveles de aislamiento (lecturas repetibles, lecturas confirmadas) • Operaciones flashback	• Reconstruir cambios cuando se requiere hacer recovery.
Almacenado en	Segmentos Undo	Archivos Redo Log



7.6.3. Administración automática de los datos Undo.

- La BD ofrece un mecanismo para realizar la administración automática de datos Undo.
- La BD realiza la administración de Datos Undo empleando *segmentos Undo* y *Tablespaces Undo*.
- Este mecanismo automático es el default.
- Durante la creación de la base de datos, se crea un tablespace auto extendible llamado `undotbs1` empleando `dbca`.
- Si el tablespace no existe, los datos Undo se escriben en el tablespace `system` (no recomendado).
- En caso de existir varios tablespaces, el parámetro `undo_tablespace` se emplea para indicar cuál de ellos se empleará: `undo_tablespace = undotbs2`.
- Para habilitar o deshabilitar la administración automática se emplea el parámetro `undo_management` con los valores `auto | manual`. Se recomienda ampliamente la administración automática por la complejidad de este tipo de datos.

7.6.4. Undo retention period.

- Una vez que la transacción realiza `commit`, los datos Undo ya no son necesarios para realizar `rollback`.
- Sin embargo, los datos Undo deben conservarse por un cierto periodo de tiempo por las siguientes causas:
  - Continuar garantizando lecturas confirmadas (read consistency) para transacciones con alta duración.
  - Empleados para realizar operaciones Flashback.
- De lo anterior, se considera retener los datos Undo el mayor tiempo posible.
- El parámetro `undo_retention` puede ser empleado para determinar la cantidad de segundos como ***mínimo*** para retener datos Undo.

Ejemplo:

```
alter system set undo_retention = 2400;
```

- La BD calcula de forma automática el valor de este periodo de tiempo basado principalmente en el tamaño del tablespace y la actividad del sistema.
- Este cálculo debería permitir a la consulta más larga y antigua actualmente ejecutándose en la base de datos terminar y al hacer `commit`, los datos Undo ya no serían necesarios.
- En caso de que una consulta o alguna otra funcionalidad que requiera hacer uso de datos undo que ya hayan sido eliminados, se obtendrán errores del tipo ***snapshot too old.***
  - Básicamente el error significa la inexistencia de Datos Undo para garantizar *read consistency*
- Para evitar este problema, se pueden emplear 2 técnicas:
  - Inicializar el parámetro `undo_retention` a un valor que sea mayor a la duración de la consulta más larga
  - Modificar la configuración del tablespace Undo para que tenga un ***tamaño fijo*** (fixed size).

- Si el tablespace Undo es configurado con un tamaño fijo, la BD ***dinámicamente*** actualiza el ***periodo de retención*** de los datos Undo basándose en la actividad del sistema y el tamaño fijo del tablespace.

Ejemplo:

```
alter database datafile '/oracle/dbs/undotbs.dbf' resize 300m;
alter database datafile '/oracle/dbs/undotbs.dbf' autoextend off;
```

- Si se decide hacer el cambio anterior al tablespace, asegurarse de especificar un valor grande del tablespace. De lo contrario, 2 errores pueden ocurrir:
    - Error al ejecutar DML al no encontrar suficiente espacio para generar datos Undo.
    - Operaciones largas pueden caer en el error ***Snapshot too old.*** Esto significa que no existe la cantidad de datos Undo necesarios para ofrecer consistencia de datos (*read consistency).*

7.6.5. Categorías de datos Undo.

Con base a lo mencionado, es posible clasificar a los datos undo en las siguientes categorías:

Categoría	Descripción
Active.	• Corresponden a datos Undo de transacciones que aún no han hecho `commit`. • Estos datos nunca son sobrescritos.
Unexpired	• Corresponden a datos Undo de transacciones que ya han hecho commit, pero que aún están dentro del periodo de retención configurado, por lo que no se pueden sobrescribir aun.
Expired	• Datos undo que pueden ser sobrescritos cuando una transacción activa los requiera.

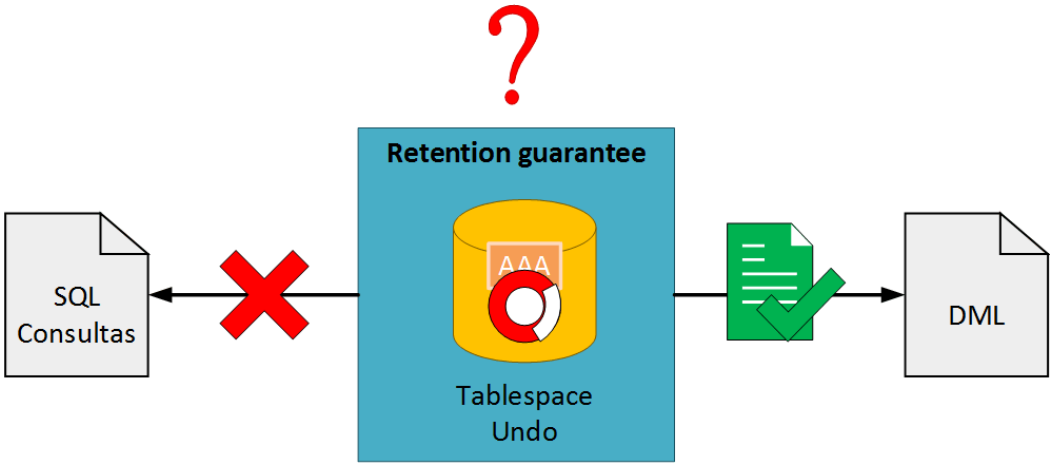
7.6.6. Retention Guarantee

Como se mencionó anteriormente, si el espacio del tablespace undo está próximo a terminarse, puede ocurrir alguna de las 2 fallas mencionadas:

- Error al ejecutar operaciones DML que requieran generar datos undo.
- Error al ejecutar consultas largas con el error ***Snapshoot too old.***

Suponer que el tablespace tiene cierta cantidad de segmentos Undo marcados como ***unexpired.*** Bajo esta situación, la base de datos podría tomar 2 decisiones:

1. Sacrificar estos datos para asegurarse que nuevas transacciones tengan espacio. Al hacer esto, los datos ***unexpired*** serán sobrescritos, lo que provocaría que posibles consultas que requieran leer estos datos fallarían con un ***snapshoot too old.***
2. Respetar el periodo de retención para evitar que consultas fallen bajo el riesgo que transacciones nuevas fallen al no contar con espacio disponible.



- Por default, el comportamiento es tomar la decisión 1.
- Este comportamiento se puede cambiar empleando la cláusula `retention garantee` de la instrucción `create/alter tablespace`.
- Esta opción se debe manejar con precaución ya que esta reserva de datos Undo puede causar que el espacio disponible para datos Undo se termine, y por lo tanto, sentencias DML pueden fallar.
- Para deshabilitar esta opción se emplea la cláusula `retention noguarantee`.
- La vista `dba_tablespaces` permite conocer si esta funcionalidad se encuentra activa o no consultando el atributo `retention`.

Ejemplo:

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

7.6.7. Monitoreando la administración de retención de datos Undo.

- Como se mencionó anteriormente, la BD ajusta el valor del periodo de retención de forma dinámica para permitir una operación adecuada y evitar los 2 problemas antes mencionados a medida de lo posible.
- La vista `v$undostat` puede ser empleada para visualizar la forma en la que la BD realiza la administración del espacio del tablespace undo.
- Es posible visualizar el periodo de retención de datos Undo actual empleando la columna `tuned_undoretention`.
- La vista representa un historial del uso de los datos undo.
- Se genera un registro cada 10 minutos y se conserva durante 4 días. Posterior a este tiempo, los registros se almacenan en `dba_hist_undostat`.

Ejemplo:

```
select begin_time, end_time, undotsn, undoblks, txncount,
maxqueryid,maxquerylen
  activeblks, unexpiredblks, expiredblks, tuned_undoretention
from v$undostat
order by begin_time desc;
```

	BEGIN_TIME	END_TIME	UNDOTSN	UNDOBLKS	TXNCOUNT	MAXQUERYID	MAXQUERYLEN	ACTIVEBLKS	UNEXPIREDBLKS	EXPIREDBLKS	TUNED_UNDORETENTION
1	09/05/2020 23:40:58	09/05/2020 23:48:04	12	3	20	(null)	0	192	304	8	4131
2	09/05/2020 23:30:58	09/05/2020 23:40:58	12	1	63	(null)	0	192	304	3016	3874
3	09/05/2020 23:20:58	09/05/2020 23:30:58	12	0	17	(null)	0	192	304	3016	3441
4	09/05/2020 23:10:58	09/05/2020 23:20:58	12	1	54	(null)	0	192	304	3016	3008
5	09/05/2020 23:00:58	09/05/2020 23:10:58	12	8	356	89w8y2pgn25yd	1749	192	304	3016	2575
6	09/05/2020 22:50:58	09/05/2020 23:00:58	12	226	137	89w8y2pgn25yd	1140	192	0	3520	2146
7	09/05/2020 22:40:58	09/05/2020 22:50:58	12	8	67	f3yfg50ga0r8n	1857	192	0	3520	1821
8	09/05/2020 22:30:58	09/05/2020 22:40:58	12	3	89	f3yfg50ga0r8n	1552	192	3520	0	1408
9	09/05/2020 22:20:58	09/05/2020 22:30:58	12	0	60	f3yfg50ga0r8n	943	192	3520	0	946
10	09/05/2020 22:10:58	09/05/2020 22:20:58	12	3533	98	f3yfg50ga0r8n	934	192	440	2528	483
11	09/05/2020 22:00:58	09/05/2020 22:10:58	12	8	325	89w8y2pgn25yd	1695	192	440	2520	1538
12	09/05/2020 21:50:58	09/05/2020 22:00:58	12	247	142	89w8y2pgn25yd	1085	192	3520	0	1153
13	09/05/2020 21:40:58	09/05/2020 21:50:58	12	8	152	89w8y2pgn25yd	475	192	3520	0	813
14	09/05/2020 21:30:58	09/05/2020 21:40:58	12	3975	1227	(null)	0	944	0	0	444
15	09/05/2020 21:20:58	09/05/2020 21:30:58	12	263	641	(null)	0	192	0	0	28138

En la muestra anterior se puede observar lo siguiente:

- Notar la diferencia de 10 min entre la fecha inicio y la fecha fin de la muestra.
- La columna `undotsn` indica el número del tablespace undo que se está empleando.
- La columna `undoblks` indica el número total de bloques undo consumidos.
- La columna `txncount` indica la cantidad de transacciones ejecutadas en ese periodo.
- La columna `maxqueryid` hace referencia al ID de la sentencia sql (consulta) que tardó la mayor cantidad de tiempo en procesarse durante el periodo. Este ID puede emplearse para consultar el código SQL en la vista `v$sql`.
- La columna `maxquerylen` indica el número de segundos que tardó la consulta en procesarse.
- Las siguientes 3 columnas indican el número de bloques del segmento undo clasificados por su categoría: *activos*, *no expirados* y *expirados*. Por ejemplo, para el registro marcado, existen 3520 bloques no expirados, 0 expirados y 192 activos.
- Finalmente se muestra el valor dinámico y calculado por la base de datos que se usa como periodo de retención. Notar que cambia constantemente. Para el registro marcado, la bd calculó un periodo de retención de 1408 segundos.

7.6.8. Vistas del DD asociadas con datos Undo.

```
v$undostat
v$tempundostat
v$rollstat
v$transaction
dba_undo_extents
dba_hist_undostat
```

Ejercicio 4:

En este ejercicio se realizarán algunas acciones para ilustrar los efectos que produce un tablespace undo sin espacio durante la operación de la base de datos.

A. Generar una sentencia SQL que muestre el tablespace undo que actualmente está en uso.
B. Generar una sentencia SQL que genere un nuevo tablespace undo cuya característica principal es contar con un espacio muy limitado y por lo tanto la base de datos estará en riesgo de generar errores al requerir más espacio undo del disponible. Nombrar al tablespace `undotbs2`, asignarle un solo data file ubicado en `'/u01/app/oracle/oradata/<ORACLE_SID>/undotbs_2.dbf'` con una capacidad únicamente de 30 MB. El data file no deberá auto extenderse. La administración de sus extensiones debe ser local.

C. Generar la sentencia SQL necesaria para que la instancia ahora haga uso del nuevo tablespace en lugar el actual. Aplicar el cambio únicamente mientras la instancia esté iniciada.

D. Crear una nueva sesión con el usuario SYS, ejecutar nuevamente la sentencia del inciso A para verificar que la instancia ahora está haciendo uso del tablespace `undotbs2`.

E. Considerar la vista `v$undostat`. Generar la siguiente consulta. Mostrar la fecha inicio y fin del periodo de la muestra hasta nivel de segundos, el id del tablespace que se está empleando, total de bloques undo empleados, total de transacciones, id de la consulta que tardó el mayor tiempo en ejecutarse, tiempo en segundos de la consulta que más tiempo tardó en ejecutarse durante el periodo de la muestra, número de bloques undo activos, no expirados y expirados; el valor del parámetro `undo_retention` calculado por la instancia. Ordenar los registros con base a la fecha de inicio de forma descendente. Mostrar únicamente los primeros 20 registros.

F. Considerar el periodo de muestreo más reciente de la consulta realizada en el inciso E. ¿Cuantos bloques podrían ser sobrescritos sin causar mayores inconvenientes?, ¿Cuántos bloques NO pueden ser sobrescritos aun?

G. Observar los valores de la columna `undotsn` de la consulta del inciso E. Confirmar que a partir del cambio del tablespace undo, el valor de esta columna cambia. Generar una sentencia SQL que permita mostrar los nombres de los tablespaces asociados a los valores de esta columna. Tip: auxiliarse de `v$tablespace`. En la consulta incluir únicamente el periodo de la muestra, id del tablespace y su nombre.

H. Generar una consulta que muestre las siguientes columnas: Nombre del tablespace undo que fue creado en pasos anteriores, total de bloques que contiene, total de bloques libres, porcentaje de bloques libres. Confirmar que el porcentaje de bloques libres es bajo, menor al 20%. Esta condición pone en riesgo la operación correcta que requieren hacer uso de los datos Undo.

I. Empleando el usuario creado en temas anteriores, generar una tabla con la siguiente estructura:

```
create table <iniciales>_cadena_2 (
  id number constraint jrc_cadena_2_pk primary key,
  cadena varchar2(1024)
) nologging;
```

- Observar la cláusula `nologging` que se agrega al final. La cláusula evita la generación de datos de Redo.
- En esta tabla se estarán realizando diversas operaciones DML para verificar el funcionamiento de los datos undo. Dichas instrucciones provocarán buena cantidad de datos Redo, y por lo tanto su archivado. Para este ejercicio no es necesario conservarlos. Por esta razón, se usa la cláusula `nologging`.
- Generar una secuencia `sec_<iniciales>_cadena_2`.
- Crear un programa PL/SQL que realice la inserción de 50,000 registros. Auxiliarse de la función `dbms_random.string`. Hacer uso de la secuencia para generar los valores de la PK. Ejecutar el programa y hacer `commit;`

J. Replicación del error al ejecutar sentencias DML.

Ejecutar nuevamente la consulta del inciso E. Esperar unos minutos a que el número de bloques undo utilizados sea cercano a cero. Posteriormente, Iniciar con el borrado de registros de los datos insertados. Debido a que se tiene poco espacio Undo, el resultado esperado es un error ya que las sentencias `delete` requieren guardar datos Undo y los bloques disponibles no serán suficientes.

Realizar el proceso de borrado por rangos de 5,000 registros. Posteriormente realizar nuevamente la consulta del inciso E y considerar únicamente al registro más reciente, agregar un renglón a la siguiente tabla. No hacer `commit` ni `rollback` en cada intento. Detener la actividad hasta ocurrir alguno de los 2 eventos: error (incluir el detalle), o al terminar de realizar el borrado. Al terminar sin importar el resultado, realizar `rollback` . Revisar los datos de la tabla y generar una conclusión al respecto.

Registros eliminados	# bloques utilizados	# transacciones ejecutadas	#bloques activos	Retención en segundos calculado.
1-5000				
…				

K. ¿Qué acciones se pueden realizar para corregir el error adicional a aumentar el tamaño del tablespace undo ?

L. Replicación del error ***snapshoot too old.***

En la siguiente actividad se provocará el segundo error: ***Snapshoot too old***. El escenario es el siguiente: Suponer la creación de una transacción T1-RC que tiene un nivel de aislamiento de *lecturas repetibles*. Esto significa que si una sentencia SQL consulta datos a la tabla creada anteriormente, la base de datos deberá mostrar siempre la misma información a pesar de otras transacciones hagan `commit`. Este nivel de aislamiento obliga a que la base de datos haga uso de los datos undo para poder brindar el nivel de aislamiento solicitado (read consistency). Sin embargo, ¿qué sucedería si otras transacciones comienzan a realizar operaciones DML? Debido a que el espacio undo es limitado, la base de datos comenzará a reescribir los segmentos undo para tratar de atender exitosamente a las transacciones DML. La sobrescritura de estos segmentos undo provocará que los datos consultados por T1-RC ya no estén disponibles en el segmento undo, y por lo tanto, la base de datos ya no podrá mostrar los datos con el nivel de aislamiento solicitado, provocando así un error *snapshoot too old.* La forma de entender este error es:

La consulta inició hace cierto tiempo. Al ser una transacción de larga duración, los datos undo que requiere para garantizar consistencia ya fueron reciclados, por lo tanto la "versión (snapshot)" que requiere esta transacción es demasiada vieja y ¡ya no se encontró en los segmentos undo! Las acciones para generar este error son las siguientes:

1. Abrir una sesión en sqlplus con el usuario dueño de la tabla creada anteriormente. Iniciar una transacción con nivel de aislamiento que permita lecturas repetibles:

```
set transaction isolation level serializable name 'T1-RC';
```

2. Ejecutar las siguientes consultas las cuales sirven como referencia. Incluir resultados.

```
select count(*) from <iniciales>_cadena_2;
select count(*)
from <iniciales>_cadena_2
where cadena  like 'A%'
or cadena like 'Z%'
or cadena like 'M%';
```

Mientras esta transacción no haga `commit` o `rollback`, el nivel de aislamiento deberá mostrar exactamente los mismos resultados aunque otras transacciones hagan cambios sobre los registros de la tabla.

3. Abrir otra sesión empleando el mismo usuario. Este usuario intentará borrar los registros de la tabla. La instancia tratará atender exitosamente a esta transacción pero provocará error en la sesión 1. Para ello, repetir las actividades del inciso J. Ejecutar operaciones `delete` en rangos de 5000 registros. Ya no será necesario llenar la tabla. Al terminar de ejecutar la sentencia `delete` hacer `commit` y ejecutar nuevamente las consultas de T1-RC. Observar que se muestran exactamente los mismos resultados a pesar de que otras transacciones han borrado datos y han hecho `commit`. Esto se debe gracias a que la transacción fue configurada con el nivel de aislamiento `serializable` y permite lecturas repetibles. La BD está tomando los datos de los segmentos undo que aún no son sobrescritos. Repetir estas acciones hasta producir el error *snapshoot too old*. Incluir las sentencias y la salida de su ejecución. ¿Cuántas instrucciones `delete` se tuvieron que ejecutar para provocar el error *snapshppt too old*?

M. Provocar ahora el comportamiento inverso: Hacer que la base de datos le de preferencia a consultas que requieran hacer uso de datos undo en lugar de dar preferencia a las sentencias SQL:

1. Incluir la sentencia SQL que permite modificar este comportamiento, ejecutarla.
2. Posterior a su ejecución, nuevamente crear una transacción T1-RC y ejecutar las mismas consultas. De forma similar, en otra sesión intentar eliminar registros de la tabla. Incluir las sentencias SQL empleadas, incluir los errores generados, explicar el comportamiento obtenido. ¿Qué sucede al ejecutar repetidamente las consultas de T1-RC?