By David Itshak
shaked19@gmail.com

http://www.ildba.co.il/author/cimid/
http://www.sqlserver.co.il/?cat=940

Global Hebrew Virtual PASS Chapter :
https://www.youtube.com/watch?v=x4hGjYGbfkc
https://www.youtube.com/watch?v=eJO8G9if3EY

Sqlsaturday Israel 2016 :
Reference and Credits

**Oracle® Database** Concepts 12c Release 1 (12.1) *E41396-13*
[https://docs.oracle.com/database/121/CNCPT/toc.htm](https://docs.oracle.com/database/121/CNCPT/toc.htm)

**Oracle® Database** SQL Language Reference 12c Release 1 (12.1) *E41329-20*

**Oracle® Database** Backup and Recovery User's Guide 12c Release 1 (12.1) *E50658-07*
[https://docs.oracle.com/database/121/BRADV/toc.htm](https://docs.oracle.com/database/121/BRADV/toc.htm)

Oracle Essentials(Oracle Database 12c), 5th; O'Reilly, 2013

Oracle OCA Oracle Database 12c Administrator Certified Associate Study Guide Exam

Pro Oracle Database 12c Administration, 2 edition ISBN 1430257288 2013

Apress Oracle Database Transactions and Locking Revealed (2014)

Apress RMAN Recipes for Oracle Database 12c A Problem-Solution Approach

[Oracle Learning Library](https://www.oracle.com/education/resources.html)
Reference and Credits

Redo Internals, Julian Dyke
http://www.juliandyke.com/Presentations/Presentations.php
Agenda

• RTO, RPO, SLA, High Availability
• Disaster Recovery, High Availability, and Business Continuity
• Sources of Data loss
• Methods of Data Protection
• Oracle 12C backup and recovery tools
• Recovery
מדדים

(RTO) Recovery Time Objective
הזמן המרבי עד לחזרתה של המשרה לתפקוד מלא עבורי תחוליםتعليمיים שוניות

(RPO) Recovery Point Objective
הזמן המרבי שלגבליו נペット לאבד מידע. קמות המאידעותれます שיאגרון谟ך לאלב? יומייםשלים, מסופר
שעות/דקות
משתנה באסעה סקלה כמי RTO אבל לא ת défini ב (מערכי מסהר RPO אפס אברידול RTO הגדול)

Service Level Agreement

Mean time ) MTTR , (Mean Time Between Failure) MTTF , (To recover

זמין גבורה

(soon availability)
 McCoy תשתיותנגןשת להמראת לחן אגרו אחורונעגנום לכל עת
downtime =

—

—
tàונכית התיאושות מאסון (DRP)

Disaster Recovery Plan

• תוכנית התיאושות הכוללת תהליכים, מדיניות ונהלים המשמשים לתיאושות מאסון המשבית לזמן לא קצרצק את התשתית הטכנולוגית החיה של הארגון.

• כוללת תכנון לחידוש של יישומים, נתונים, חומרה, תקשורת (בוגן) ואלמנטים נוספים של טכנולוגית המידע.

• דוגמאות: שריפה, רעידת אדמה, הצפה, התפוצצות פצצות, קריסת בניין, מחיקת מוטיעת וヂודום

•
Disaster Recovery, High Availability, and Business Continuity

• *Business continuity*
  – *The process of ensuring that day-to-day activities can continue* regardless of the problem. It encompasses both technical and nontechnical disasters, such as a worker strike or a supply-chain issue.

• *High availability*
  – *The process of ensuring that systems remain available as long as possible* no matter what the cause might be for downtime. This includes disasters, but it also includes events such as regular maintenance, patches, and hardware migration.

• *Disaster recovery*
  – *The process of mitigating the likelihood of a disaster and the process of returning the system to a normal state in the event of a disaster.*

The relationship between business continuity, high availability, and disaster recovery.
BCP – Business Continuity Plan

• DRP
• העסקת התוכנית
• פעילות ב organización registrations
• בוחננים ומנהלי רשת
• ברירת מחדשים (ב.chrome הלולם למשל) להנהלת התוכנית התואשת מאסונות
• אפקטיביות
• בין 4%-2% מח嗾ות התוכננים הבוחנים בוחננים.
 BinaryTree

תאומים

יום ה-11 בספטמבר

Pinecones

Pinecone

Pinecones
הפעלה מתאר מרחוק חלופי

תשתיית מענה לערכי מינים למידת ביעט חירום: מערכות

مدير שירות, תכשורת, תחנות עבודה וכולל

[Diagram of network with servers and databases connected]
Step 1: Baseline

SAN or NAS Attached hosts

Immediate Write Acknowledgement

Source

Baseline copy of source volume(s)

LAN/WAN

Target

OR

Time-definite delivery around the world.

Immediate Write Acknowledgement

Step 2: Updates

SAN or NAS Attached hosts

Immediate Write Acknowledgement

Source

Periodic updates of changed blocks

LAN/WAN

Target
Disaster Recovery Process

- Data Centric Environments

"Mean Time To Recovery"-
• צמצום אחרון אירוצים DRP

• שחרור נתונים במוב יכלול למתקףIMIT

Production Site

Disaster Recovery Site

(resync backwards after source restoration)
Oracle Data Guard Architecture

Production Database

Network

Sync or Async Redo Shipping

Physical Standby

Open R/O

Redo Apply

Backup

Redo Shipping
אירוע

• אתר ראשי המסונכר
לאתר משני

קרה אשון באתר ראשי
אירוע数据中心 DRP

סנכרון מופסק וה

סנכרון محمود בו
storage באתר ההופך
ל뿐 פעיל . בסיסי

הנתונים ל-R/W-

השרטים הפוכים לזוית
פעילה

(Primary Site)

(Alternat Site)
Murphy’s laws

• If anything can go wrong, it will

• If anything just cannot go wrong, it will anyway

• If everything seems to be going well, you have obviously overlooked something
Worst Practices

1. Failing to keep offsite backup copies of data
2. Failing to monitor your backup job effectively
3. Using manual or obsolete procedure
4. No written plans
5. Lack of backup integrity (backing up files, not DB)
6. Not testing the backup (no dry run)
7. Doing no backup at all
Why Backups Are So Important ?

JournalSpace Drama: All Data Lost Without Backup, Company Deadpooled

Posted Jan 3, 2009 by Robin Wauters

Blogging platform JournalSpace (which I'd never heard of to date) has ceased to be, following a wipe-out of the main database for which there was no back-up in place. According to the JournalSpace blog, the database was overwritten as a result of a malicious act from a disgruntled ex-employee.

It was the guy handling the IT (and, yes, the same guy who I caught stealing from the company, and who did a slash-and-burn on some servers on his way out) who made the choice to rely on RAID as the only backup mechanism for the SQL server. He had set up automated backups for the HTTP server which contains the PHP code, but, inscrutibly, had no backup system in place for the SQL data. The ironic thing here is that one of his hobbies was telling everybody how smart he was.
Why Backups Are So Important?

Why Backups Are So Important?

Why Backups Are So Important?

Why Backups Are So Important?
Kinds of Failures
Media Failure

• Media failure is a physical problem with a disk that causes a failure of a read from or write to a disk that is required to run the database.

• Any database file can be vulnerable to a media failure.
Corruption

• Sometimes a software or hardware malfunction can corrupt data.

• Physical corruption:
  – Bad header
  – Block is Fractured/Incomplete
  – Block checksum is invalid
  – Block is misplaced

• Logical Corruption
  • Block contains a valid checksum and the structure below the beginning of the block is corrupt.
User Error

UPDATE BANK _TBL SET BALANCE=500
WHERE ACCOUNT_NO=123456
Source Of Data Loss
Sources of Data loss

• Media failure
  – Environmental issue
  – Physical Problems with a disk
  – Remediation: fault tolerance/redundancy.

• User error:
  – Accidental or malicious deletion
  – Remediation: Training, security policy

• Application error
  – Poorly written code can corrupt data blocks
  – A user session can crash
  – SQL statements can fail
  – Remediation: Code Review
Methods of Data Protection

• Backup types
  – Physical backup are copies of physical database files
  – Logical backups are backups for logical data such as tables or stored procedures

• Backup methods:
  – Oracle tools: Rman; EM Cloud control 12C
  – User-managed backup and recovery (OS commands, 3rd party tools)

• Backup-related procedures
  – Data archival: Copying data to long-term storage
  – Data transfer: ETL or OLAP scenarios
Oracle Database 12C backup and Recovery Tools

• Recovery Manager (Rman) command line tool
• Oracle Enterprise Manager Cloud Control 12C
  – GUI front-end to Rman
  – Data Recovery Advisor (DRA) – Automated corruption detection and repair utility (Rman interface)
• Oracle Data Pump:
  – PL/SQL packages to run from the command line
  – Perform logical import and exports
Preliminary Data Protection tasks

• Enable ARCHIVELOG Mode
  – Periodically archives (backs up) the online redo log files
• Backup UNDO tablespace
• Specify the fast recovery area
• Configure Flashback features
• Multplex and back up the control file
• Relocating your data files to different disks: Backup on showers disks
• Back up server parameter file
Recovery

• From which events below can DBMS recover?
  – Wrong data entry
  – Disk failure
  – Fire / earthquake / bankruptcy ...

• **Systems crashes**
  – **Software errors**
  – **Power failures**
## Recovery

<table>
<thead>
<tr>
<th>Type of Crash</th>
<th>Prevention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrong data entry</td>
<td>Constraints and Data cleaning</td>
</tr>
<tr>
<td>Disk crashes</td>
<td>Redundancy: e.g. RAID, archive</td>
</tr>
<tr>
<td>Fire, theft, bankruptcy</td>
<td>Buy insurance, Change jobs…</td>
</tr>
<tr>
<td><strong>Most frequent</strong></td>
<td><strong>DATABASE RECOVERY</strong></td>
</tr>
<tr>
<td>System failures</td>
<td></td>
</tr>
</tbody>
</table>
Oracle Database 12C backup and Recovery Tools

• Recovery Manager (Rman) command line tool

• Oracle Enterprise Manager Cloud Control 12C
  – GUI front-end to Rman
  – Data Recovery Advisor (DRA) – Automated corruption
detection and repair utility (Rman interface)

• Oracle Data Pump:
  – PL/SQL packages to run from the command line
  – Perform logical import and exports
Agenda

• Reminder – Oracle Concepts
• Oracle Database 12c Architecture
• Database Write Process
• Redo Log Buffer
• Online Redo Logs
• Redo and Undo
• Real Application Cluster
Oracle Database 12c Architecture
Oracle Database 12c Architecture:

Major background processes:

- **DBWn**: The database writer writes blocks from the database buffer cache to the data files.
- **CKPT**: The checkpoint process writes checkpoint information to the control files and data file headers.
- **LGWR**: The log writer writes redo information from the log buffer to the online redo logs.
- **ARCn**: The archiver copies the content of online redo logs to archive redo log files.
Database Writer Process (DBWn)

- Write modified **(dirty)** buffer in database buffer cache to disk
- **Asynchronously** while performing other process to advance checkpoint.
Redo Log Buffer

- A **circular** buffer in SGA that stores redo entries describing changes made to DB.

- A redo record is a data structure that contains the information necessary to reconstruct, or redo, changes made to the database by DML or DDL operations.

- DB recovery applies redo entries to data files to reconstruct lost changes.

- DB processes copy redo entries from the user memory space to the redo log buffer in the SGA.

- Redo entries take up continuous, sequential space in the buffer.
Log Writer Process (LGWR)

Writes the redo log buffer to a redo log file on disk writes

- When user commit a transaction
- When the redo log buffer is one-third full
- Before a DBWn process writes modified buffers to disk
- Every 3 seconds
Online Redo Logs (ORLs)

- Crucial to the Oracle database
- Transaction logs for DB
- Two types of redo log files: *online* and *archived*
- Used for recovery purposes in the event of an instance or media failure.
Online Redo Log Switches

- Log writer is the background process responsible for writing transaction information from redo log buffer (in the SGA) to the online redo log files (on disk).
- Log writer flushes the contents of the redo log buffer when any of the following are true:
  - A COMMIT is issued.
  - A log switch occurs.
  - Three seconds go by.
  - Redo log buffer is one-third full.
  - Redo log buffer fills to one megabyte.
At time 1, Block A is read from Data File AA into the buffer cache and modified.

At time 2 the redo-change vector information (how the block changed) is written to the log buffer.

At time 3 the log-writer process writes the Block A change-vector information to online redo log 1.

At time 4 a log switch occurs, and online redo log 2 becomes the current online redo log.

Reuse of Online Redo Log Files
Redo Stream

- Oracle normally only writes change vectors to the redo stream
What is Redo

• Redo log files (ROLs) are crucial to the Oracle database.
  – Transaction logs for the database.

• Oracle maintains two types of redo log files: **online and archived**.
  – *For recovery purposes*;
  – Main Purpose: in event of an instance or media failure.

What is Redo

• **If the power goes off on your database machine**, causing an instance failure, Oracle will use the online redo logs to restore the system to exactly the committed point it was at immediately prior to the power outage.

• **If your disk drive fails (a media failure)**, Oracle will use both archived redo logs and online redo logs to recover a backup of the data that was on that drive to the correct point in time.

• **If you “accidentally” truncate a table or remove some critical information and commit the operation**, you can restore a backup of the affected data and recover it to the point in time immediately prior to the “accident” using online and archived redo log files.
What is Undo?

- *Undo is conceptually the opposite of redo.*
- Undo information is generated by the database as you make modifications to data so that the data can be put back the way it was before the modifications took place.

- This might be done in support of *multiversioning,* or in the event the transaction or statement you are executing fails for any reason, or *if we request it with a ROLLBACK statement.*

- *Whereas redo is used to replay a transaction in the event of failure—to recover the transaction—undo is used to reverse the effects of a statement or set of statements.*

- Undo, unlike redo, is stored internally in the database in a special set of segments known as undo segments.
What is Undo?

• It is a common misconception that undo is used to restore the database physically to the way it was before the statement or transaction executed, but this is not so.

• The database is *logically* restored to the way it was—any changes are logically undone—but the data structures, the database blocks themselves, may well be different after a rollback.

• The reason for this lies in the fact that, in any multiuser system, there will be tens or hundreds or thousands of concurrent transactions. One of the primary functions of a database is to mediate concurrent access to its data.

• The blocks that our transaction modifies are, in general, being modified by many other transactions as well.

• Therefore, we can’t just put a block back exactly the way it was at the start of our transaction—that could undo someone else’s work!
**Undo**

- **Oracle Database uses undo data to:**
  - Roll back an active transaction
  - Recover a terminated transaction
  - Provide read consistency: support of multiversioning. Uncommitted changes cannot be seen by other sessions
  - Perform some logical flashback operations

- **Oracle Database stores undo data inside the database rather than in external logs.**

- **Undo data is stored in blocks that are updated just like data blocks, with changes to these blocks generating redo records.**
Undo Segments and Transactions

- When a transaction starts, DB assigns transaction to an undo segment, and to a transaction table, in current undo tablespace.
- Multiple active transactions can write concurrently to the same undo segment or to different segments.
- EX: transactions T1 and T2 can both write to undo segment U1, or T1 can write to U1 while T2 writes to undo segment U2.
- Undo segment form a ring. Transactions write to one undo extent, and then to the next extent in the ring, and so on in cyclical fashion.
Read Consistency and Undo Segments

- To manage the multiversion read consistency model, the database must create a read consistent set of data when a table is simultaneously queried and updated. Oracle Database achieves this goal through undo data.

- Guarantees that data returned by a single query is committed and consistent for a single point in time.
  - Depends on the transaction isolation level and the query:
    - In the read committed isolation level, point is time at which the statement was opened.
      - EX: if a SELECT statement opens at SCN 1000, then this statement is consistent to SCN 1000.
    - In a serializable or read-only transaction, this point is the time the transaction began.
      - EX: if a transaction begins at SCN 1000, and if multiple SELECT statements occur in this transaction, then each statement is consistent to SCN 1000.
    - In a Flashback Query operation (SELECT ... AS OF), SELECT statement explicitly specifies the point in time.
      - EX: you can query a table as it appeared last Thursday at 2 p.m.
Redo and Undo

• Undo information, stored in undo tablespaces or undo segments, is protected by redo as well.

• **Undo data is treated just like table data or index data**—changes to undo generates some redo, which is logged (to the log buffer and then the redo log file).

• Undo data is added to the undo segment and is cached in the buffer cache, just like any other piece of data would be.
Oracle database server - Real Application Cluster

- More than one instance communicates to a single database.

- If an instance fails, the remaining instances in the RAC pool remain open and active. Connections from failed instances can be failed-over to active instances.

- Oracle manages the connection load balancing and failover automatically.
Oracle RAC architecture and components
Agenda

1. What is Redo?
2. Redo Records
3. Change Vectors
4. Row Operations
What is Redo?

- Redo logs contain a history of all changes made to the database

- Redo log files are used by
  - Recovery (instance and media)
  - Log Miner
  - Oracle Streams
  - Oracle golden Gate
  - Oracle Data Guard

- Every change made to the database is
  - written to the redo log buffer before it is written to the data block buffer
  - written to the redo log file before it is written to the data file

- The redo log buffer is flushed to the redo log file when a COMMIT is issued
What is Redo?

• Redo log files
  – Include all changes made by DML statements
    • INSERT
    • UPDATE
    • DELETE
    • SELECT FOR UPDATE
  – Include all changes made to dictionary objects by DDL statements
  – Include DDL statement text Include all changes made by recursive statements
Buffers and Writers

Redo Log Buffer -> LGWR -> Redo Log File

Redo Log File

Data Block Buffer -> DBWR -> Data File

Data File
Logging and Archiving

LGWR -> Group 1 -> Arch 1
LGWR -> Group 2 -> Arch 2
LGWR -> Group 3 -> Arch 3

Group 1 -> Arch 4
Group 2 -> Arch 5
Group 3 -> Arch 6

Redo Log Files

Archive Log Files
Redo Log Files

- Redo log uses operating system block size
  - usually 512 bytes
  - format dependent on
    - operating system
    - Oracle version
- Each redo log consists of
  - header
  - redo records
- Redo log is written sequentially

<table>
<thead>
<tr>
<th>Block 0</th>
<th>Block 1</th>
<th>Block 2</th>
<th>Block 3</th>
<th>Block 4</th>
<th>...</th>
<th>Block M</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Header</td>
<td>Redo Header</td>
<td>Redo Record 1</td>
<td>Redo Records 2 &amp; 3</td>
<td>Redo Records 3 &amp; 4</td>
<td>...</td>
<td>Redo Record N</td>
</tr>
</tbody>
</table>
Redo Records

- A redo record consists of
  - redo record header
  - one or more change vectors

- Each redo record contains undo and redo for an atomic change

- Some changes do not require undo
Redo Record Header

• Every redo record has a header

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread</td>
<td>Thread Number</td>
</tr>
<tr>
<td>RBA</td>
<td>Redo Byte Address</td>
</tr>
<tr>
<td>LEN</td>
<td>Length of record in bytes</td>
</tr>
<tr>
<td>SCN</td>
<td>System Change Number</td>
</tr>
<tr>
<td>SUBSCN</td>
<td>Date and Time of Change</td>
</tr>
</tbody>
</table>
Redo Byte Address (RBA)

- Every redo record has a Redo Byte Address (RBA) e.g.

  \[
  \text{RBA: 0x003666.000000cf.0010}
  \]

- RBA is 10 bytes in length

- RBA identifies start of redo record
- Fields are
  - Log sequence number (0x3666)
  - Block number within redo log (0xcf)
  - Byte number within block (0x10)
System Change Number (SCN)

- Also called System Commit Number
- Defines committed version of database
- SCN is 6 bytes in length

```
SCN:0x0000.0ac67cc3
```

- Contains
  - Wrap (2 bytes) e.g. 0000
  - Base (4 bytes) e.g. 0ac67cc3

- Base is incremented for each new SCN
- Saved in redo record header

- RAC databases maintain a global SCN
- Distributed transactions use highest SCN
Change Vector

- Describes a change to a single data block

- Can apply to
  - undo headers
  - undo blocks
  - data segment headers
  - data blocks

- Is created in PGA before the data block buffer is modified

- Consists of
  - header
  - array of change record lengths
  - array of change records
Change Vector

For example

- Change Header
- Length Vector: 16
- Change Record 1: 20
- Change Record 2: 48
- Change Record 3: 28
- Change Record 4: 29
- Change Record 5: 2
- Change Record 6: 2
- Change Record 7: 10
Change Vector Header

◆ Every change vector has a header e.g.

CHANGE #2 TYP:0 CLS: 1 AFN:5 DBA:0x0144d023 SCN:0x0000.0ac67cce
SEQ: 4 OP:11.5

◆ Fields include

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE</td>
<td>Change number</td>
</tr>
<tr>
<td>TYP</td>
<td>Change type</td>
</tr>
<tr>
<td>CLS</td>
<td>Class</td>
</tr>
<tr>
<td>AFN</td>
<td>Absolute File Number</td>
</tr>
<tr>
<td>DBA</td>
<td>Relative Database Block Address</td>
</tr>
<tr>
<td>SCN</td>
<td>System Change Number</td>
</tr>
<tr>
<td>SEQ</td>
<td>Sequence Number (relative to SCN)</td>
</tr>
<tr>
<td>OP</td>
<td>Operation Code</td>
</tr>
</tbody>
</table>
## Block Classes

- Class in change header is equivalent to X$BH.CLASS

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data Block</td>
</tr>
<tr>
<td>2</td>
<td>Sort Block</td>
</tr>
<tr>
<td>3</td>
<td>Deferred Undo Segment Blocks</td>
</tr>
<tr>
<td>4</td>
<td>Segment Header Block (Table)</td>
</tr>
<tr>
<td>5</td>
<td>Deferred Undo Segment Header Blocks</td>
</tr>
<tr>
<td>6</td>
<td>Free List Blocks</td>
</tr>
<tr>
<td>7</td>
<td>Extent Map Blocks</td>
</tr>
<tr>
<td>8</td>
<td>Space Management Bitmap Blocks</td>
</tr>
<tr>
<td>9</td>
<td>Space Management Index Blocks</td>
</tr>
<tr>
<td>10</td>
<td>Unused</td>
</tr>
<tr>
<td>11 + 2r</td>
<td>Segment Header for Undo Segment r</td>
</tr>
<tr>
<td>12 + 2r</td>
<td>Data Blocks for Undo Segment r</td>
</tr>
</tbody>
</table>

- e.g. 11 is System Rollback Segment Header
Database Block Address (DBA)

- Every database block has a Database Block Address (DBA)

  e.g. DBA:0x0144d023

- DBA is 4 bytes in length
- Fields are
  - Upper 10 bits represent relative file number
  - Lower 22 bits represent block number
- For example

```sql
DECLARE
  l_dba   NUMBER := TO_NUMBER ('0144D023','XXXXXXXX');
  l_file  NUMBER := DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE (l_dba);
  l_block NUMBER := DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK (l_dba);
BEGIN
  DBMS_OUTPUT.PUT_LINE ('File : '||l_file);
  DBMS_OUTPUT.PUT_LINE ('Block : '||l_block);
END;
```
Operation Codes

- Each change is represented by an operation in the redo log
- There are over 150 different operations
- Each operation has a layer code and a sub code e.g. 11.2

Layers include

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Block Cleanout</td>
</tr>
<tr>
<td>5</td>
<td>Transaction Management</td>
</tr>
<tr>
<td>10</td>
<td>Index Operations</td>
</tr>
<tr>
<td>11</td>
<td>Row Operations</td>
</tr>
<tr>
<td>13</td>
<td>Segment Management</td>
</tr>
<tr>
<td>14</td>
<td>Extent Management</td>
</tr>
<tr>
<td>17</td>
<td>Tablespace Management</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Block Image (Hot backups)</td>
</tr>
<tr>
<td>19</td>
<td>Direct Loader</td>
</tr>
<tr>
<td>20</td>
<td>Compatibility Segment</td>
</tr>
<tr>
<td>22</td>
<td>Locally Managed Tablespaces</td>
</tr>
<tr>
<td>23</td>
<td>Block Writes</td>
</tr>
<tr>
<td>24</td>
<td>DDL Statements</td>
</tr>
</tbody>
</table>
Transactions

- The first DML statement in a session creates a transaction
  - Allocates an undo segment
  - Creates a 5.2 change to update transaction table in undo segment header

```plaintext
CHANGE #1 TYP:0 CLS:25 AFN:3 DBA:0x00c0012e SCN:0x0000.0ac86eb8 SEQ: 1 OP:5.2
ktudh redo: slt: 0x0010 sqn: 0x0000475a flg: 0x0012 siz: 96 fbi: 0
uba: 0x00c04d20.234b.0e    pxid: 0x0000.000.00000000
```

- A commit (or rollback) ends the transaction
- A 5.4 change is created for a commit

```plaintext
CHANGE #1 TYP:0 CLS:25 AFN:3 DBA:0x00c0012e SCN:0x0000.0ac86ebf SEQ: 1 OP:5.4
ktucm redo: slt: 0x0010 sqn: 0x0000475a srt: 0 sta: 9 flg: 0x0
```

- Rollbacks apply all undo for the transaction followed by a commit
Transaction ID (XID)

- Every transaction has an XID

  xid: 0x0004.00e.0000449b

- XID is 8 bytes in length

- Contains
  - Undo segment number (USN) of transaction (0x0004)
  - Undo segment header transaction table slot (0x00e)
  - Sequence number (wrap) (0x0000449b)
Undo Block Address (UBA)

- Address of change in undo block

**uba: 0x00c01f17.2758.04**

- UBA is 7 bytes in length

- Contains
  - DBA of undo block (0x00c01f17)
  - Sequence number (2758)
  - Record number in block (4)
Physiological Logging

- Oracle normally uses physiological logging
- Only changes made to each block are recorded

<table>
<thead>
<tr>
<th>Statement</th>
<th>Undo</th>
<th>Redo</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT INTO t1 VALUES (1, 'ABC');</td>
<td>Delete row 1</td>
<td>Row 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c1 := 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c2 := 'ABC'</td>
</tr>
<tr>
<td>UPDATE t1 SET c2 = 'DEF' WHERE c1 = 1;</td>
<td>Row 1</td>
<td>Row 1</td>
</tr>
<tr>
<td></td>
<td>c2 := 'ABC'</td>
<td>c2 := 'DEF'</td>
</tr>
<tr>
<td>DELETE FROM t1 WHERE c1 = 1;</td>
<td>Row 1</td>
<td>Delete row 1</td>
</tr>
<tr>
<td></td>
<td>c1 := 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c2 := 'DEF'</td>
<td></td>
</tr>
<tr>
<td>SELECT c2 FROM t1 WHERE c1 = 1 FOR UPDATE;</td>
<td>Unlock row 1</td>
<td>Lock row 1</td>
</tr>
</tbody>
</table>

- Note that INSERT statements generate minimal undo
Redo Record Example

◆ This is an example of a redo log dump for a single row update

```sql
UPDATE t1 SET c2 = 20 WHERE c1 = 1;
```

◆ The previous value of column c2 for this row was 10

◆ The redo record starts with a header

```
REDO RECORD - Thread:1 RBA: 0x003e12.00000004.01cc LEN: 0x00f8 VLD: 0x01 SCN: 0x0000.0ac73691 SUBSCN: 1 06/26/2003 14:40:14
```

◆ This redo record is 248 (0xF8) bytes in length
Redo Record Example (Continued)

- Change 1 updates the undo block

CHANGE #1 TYP:0 CLS:26 AFN:3 DBA:0x00c04ab7 SCN:0x0000.0ac73690 SEQ: 2 OP:5.1
ktudb redo: siz: 104 spc: 1860 flg: 0x0022 seq: 0x233f rec: 0x02
    xid: 0x0005.00b.0000460b
ktubu redo: slt: 11 rci: 1 opc: 11.1 objn: 19378 objd: 19378 tsn: 4
Undo type: Regular undo   Undo type: Last buffer split: No
Tablespace Undo: No
    0x00000000
KDO undo record:
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x00c04ab7.233f.01
KDO Op code: URP row dependencies Disabled
    xtype: XA bdba: 0x0144d022 hdba: 0x0144d021
itli: 1 ispac: 0 maxfr: 1177
tabn: 0 slot: 2(0x2) flag: 0x2c lock: 0 ckix: 0
ncol: 2 nnew: 1 size: 0
col 1: [2] c1 0b
Redo Record Example (Continued)

- Change 2 updates the data block

```plaintext
CHANGE #2 TYP:0 CLS: 1 AFN:5 DBA:0x0144d022 SCN:0x0000.0ac73690 SEQ: 1 OP:11.5
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x00c04ab7.233f.02
KDO Op code: URP row dependencies Disabled
  xtype: XA bdba: 0x0144d022 hdba: 0x0144d021
itli: 1 ispac: 0 maxfr: 1177
tabn: 0 slot: 2(0x2) flag: 0x2c lock: 1 ckix: 0
ncol: 2 nnew: 1 size: 0
col 1: [ 2] c1 15
```
Row Operations

- Row operations generate layer 11 redo
- Opcodes include

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.2</td>
<td>IRP</td>
<td>Insert Single Row</td>
</tr>
<tr>
<td>11.3</td>
<td>DRP</td>
<td>Delete Single Row</td>
</tr>
<tr>
<td>11.4</td>
<td>LKR</td>
<td>Lock Row</td>
</tr>
<tr>
<td>11.5</td>
<td>URP</td>
<td>Update Row</td>
</tr>
<tr>
<td>11.6</td>
<td>ORP</td>
<td>Chained Row</td>
</tr>
<tr>
<td>11.9</td>
<td>CKI</td>
<td>Cluster key index</td>
</tr>
<tr>
<td>11.10</td>
<td>SKL</td>
<td>Set cluster key pointers</td>
</tr>
<tr>
<td>11.11</td>
<td>QMI</td>
<td>Insert Multiple Rows</td>
</tr>
<tr>
<td>11.12</td>
<td>QMD</td>
<td>Delete Multiple Rows</td>
</tr>
</tbody>
</table>
### Single Row Insert

<table>
<thead>
<tr>
<th>Statements</th>
<th>Redo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-- Statement #1</strong></td>
<td></td>
</tr>
<tr>
<td>INSERT INTO t1 VALUES (1);</td>
<td>HEADER</td>
</tr>
<tr>
<td></td>
<td>UNDO #1</td>
</tr>
<tr>
<td></td>
<td>REDO #1</td>
</tr>
<tr>
<td><strong>-- Statement #2</strong></td>
<td></td>
</tr>
<tr>
<td>INSERT INTO t1 VALUES (2);</td>
<td>UNDO #2</td>
</tr>
<tr>
<td></td>
<td>REDO #2</td>
</tr>
<tr>
<td><strong>-- Statement #3</strong></td>
<td></td>
</tr>
<tr>
<td>INSERT INTO t1 VALUES (3);</td>
<td>UNDO #3</td>
</tr>
<tr>
<td></td>
<td>REDO #3</td>
</tr>
<tr>
<td>COMMIT;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMMIT</td>
</tr>
</tbody>
</table>
Multi Row Insert

Statements

-- Statement #1
INSERT INTO t1
SELECT * FROM t2;

COMMIT;

Redo

HEADER 5.2
UNDO #1 5.1
REDO #1 11.11
COMMIT 5.4
## Single Row Update

<table>
<thead>
<tr>
<th>Statements</th>
<th>Redo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-- Statement #1</strong>&lt;br&gt;UPDATE t1 SET c2 = c2 + 1 WHERE c1 = 1;</td>
<td>HEADER 5.2, UNDO #1 5.1, REDO #1 11.5</td>
</tr>
<tr>
<td><strong>-- Statement #2</strong>&lt;br&gt;UPDATE t1 SET c2 = c2 + 1 WHERE c1 = 2;</td>
<td>UNDO #2 5.1, REDO #2 11.5</td>
</tr>
<tr>
<td><strong>-- Statement #3</strong>&lt;br&gt;UPDATE t1 SET c2 = c2 + 1 WHERE c1 = 3;</td>
<td>UNDO #3 5.1, REDO #3 11.5</td>
</tr>
<tr>
<td><strong>COMMIT</strong>;</td>
<td>COMMIT 5.4</td>
</tr>
</tbody>
</table>
Multi Row Update

<table>
<thead>
<tr>
<th>Statements</th>
<th>Redo</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- T1 contains 3 rows</td>
<td>HEADER 5.2</td>
</tr>
<tr>
<td>UPDATE t1 SET c2 = c2 + 1;</td>
<td>UNDO #1 5.1</td>
</tr>
<tr>
<td></td>
<td>REDO #1 11.5</td>
</tr>
<tr>
<td></td>
<td>UNDO #2 5.1</td>
</tr>
<tr>
<td></td>
<td>REDO #2 11.5</td>
</tr>
<tr>
<td></td>
<td>UNDO #3 5.1</td>
</tr>
<tr>
<td></td>
<td>REDO #3 11.5</td>
</tr>
<tr>
<td>COMMIT;</td>
<td>COMMIT 5.4</td>
</tr>
<tr>
<td>Statements</td>
<td>Redo</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
</tr>
<tr>
<td>-- Statement #1</td>
<td></td>
</tr>
<tr>
<td>DELETE FROM t1</td>
<td>HEADER</td>
</tr>
<tr>
<td>WHERE c1 = 1;</td>
<td>5.2</td>
</tr>
<tr>
<td></td>
<td>UNDO #1</td>
</tr>
<tr>
<td></td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>REDO #1</td>
</tr>
<tr>
<td></td>
<td>11.3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>-- Statement #2</td>
<td></td>
</tr>
<tr>
<td>DELETE FROM t1</td>
<td>UNDO #2</td>
</tr>
<tr>
<td>WHERE c1 = 2;</td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>REDO #2</td>
</tr>
<tr>
<td></td>
<td>11.3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>-- Statement #3</td>
<td></td>
</tr>
<tr>
<td>DELETE FROM t1</td>
<td>UNDO #3</td>
</tr>
<tr>
<td>WHERE c1 = 3;</td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>REDO #3</td>
</tr>
<tr>
<td></td>
<td>11.3</td>
</tr>
<tr>
<td>COMMIT;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMMIT</td>
</tr>
<tr>
<td></td>
<td>5.4</td>
</tr>
</tbody>
</table>
# Multi Row Delete

## Statements

```
-- T1 contains 3 rows
DELETE FROM t1;
```

## Redo

<table>
<thead>
<tr>
<th>Redo</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEADER</td>
</tr>
<tr>
<td>UNDO #1</td>
</tr>
<tr>
<td>REDO #1</td>
</tr>
<tr>
<td>UNDO #2</td>
</tr>
<tr>
<td>REDO #2</td>
</tr>
<tr>
<td>UNDO #3</td>
</tr>
<tr>
<td>REDO #3</td>
</tr>
</tbody>
</table>

```
COMMIT;
```

```
COMMIT | 5.4 |
```
## Single Row Select For Update

<table>
<thead>
<tr>
<th>Statements</th>
<th>Redo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-- Statement #1</strong>&lt;br&gt;SELECT c2 FROM t1&lt;br&gt;WHERE c1 = 1&lt;br&gt;FOR UPDATE;</td>
<td><strong>HEADER</strong>&lt;br&gt;5.2</td>
</tr>
<tr>
<td><strong>UNDO #1</strong>&lt;br&gt;5.1</td>
<td><strong>REDO #1</strong>&lt;br&gt;11.4</td>
</tr>
<tr>
<td><strong>-- Statement #2</strong>&lt;br&gt;UPDATE t1&lt;br&gt;SET c2 = c2 + 1&lt;br&gt;WHERE c1 = 1;</td>
<td><strong>UNDO #2</strong>&lt;br&gt;5.1</td>
</tr>
<tr>
<td><strong>REDO #2</strong>&lt;br&gt;11.5</td>
<td><strong>COMMIT</strong>&lt;br&gt;5.4</td>
</tr>
<tr>
<td><strong>COMMIT;</strong></td>
<td></td>
</tr>
</tbody>
</table>

---

83
## Multi Row Select For Update

<table>
<thead>
<tr>
<th>Statements</th>
<th>Redo</th>
</tr>
</thead>
</table>
| **-- T1 contains 3 rows**  
**SELECT c2 FROM t1 FOR UPDATE;** | **HEADER** 5.2 |
| | **UNDO #1** 5.1 |
| | **REDO #1** 11.4 |
| | **UNDO #2** 5.1 |
| | **REDO #2** 11.4 |
| | **UNDO #3** 5.1 |
| | **REDO #3** 11.4 |
| **COMMIT;** | **COMMIT** 5.4 |
### Rollback

<table>
<thead>
<tr>
<th>Statements</th>
<th>Redo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statement #1</strong> INSERT INTO t1 VALUES (1);</td>
<td><strong>HEADER</strong> 5.2</td>
</tr>
<tr>
<td></td>
<td><strong>UNDO #1</strong> 5.1</td>
</tr>
<tr>
<td></td>
<td><strong>REDO #1</strong> 11.2</td>
</tr>
<tr>
<td><strong>Statement #2</strong> INSERT INTO t1 VALUES (2);</td>
<td><strong>UNDO #2</strong> 5.1</td>
</tr>
<tr>
<td></td>
<td><strong>REDO #2</strong> 11.2</td>
</tr>
<tr>
<td>ROLLBACK;</td>
<td><strong>UNDO #3</strong> 11.3</td>
</tr>
<tr>
<td></td>
<td><strong>REDO #3</strong> 5.6</td>
</tr>
<tr>
<td></td>
<td><strong>UNDO #4</strong> 11.3</td>
</tr>
<tr>
<td></td>
<td><strong>REDO #4</strong> 5.11</td>
</tr>
<tr>
<td></td>
<td><strong>COMMIT</strong> 5.4</td>
</tr>
</tbody>
</table>
Agenda

• Overview of Checkpoints
• Purpose of Checkpoints
• When Oracle Database Initiates Checkpoints
## Oracle Instance Shutdown Modes

<table>
<thead>
<tr>
<th>Database Behavior</th>
<th>ABORT</th>
<th>IMMEDIATE</th>
<th>TRANSACTIONAL</th>
<th>NORMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permits new user connections</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Wait until current sessions end</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Wait until current transactions end</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Performs a <strong>checkpoint</strong> and closes open files</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Checkpoint Process (CKPT)

- Updates the control file and data file headers with checkpoint information and signals DBW to write blocks to disk.
- Checkpoint information includes the checkpoint position, SCN, location in online redo log to begin recovery, and so on.
Overview of Checkpoints

• A crucial mechanism in consistent database shutdowns, instance recovery, and Oracle Database operation generally.

• The term checkpoint has the following related meanings:

1. A data structure that indicates the checkpoint position, which is the SCN in the redo stream where instance recovery must begin. The checkpoint position is determined by the oldest dirty buffer in the database buffer cache. The checkpoint position acts as a pointer to the redo stream and is stored in the control file and in each data file header.

2. The writing of modified database buffers in the database buffer cache to disk.
Purpose of Checkpoints

• Reduce the time required for recovery in case of an instance or media failure
• Ensure that dirty buffers in the buffer cache are written to disk regularly
• Ensure that all committed data is written to disk during a consistent shutdown
When Oracle Database Initiates Checkpoints

1. Thread checkpoints

- The database writes to disk all buffers modified by redo in a specific thread before a certain target. The set of thread checkpoints on all instances in a database is a database checkpoint.

- Thread checkpoints occur in the following situations:
  - Consistent database shutdown
  - ALTER SYSTEM CHECKPOINT statement
  - Online redo log switch
  - ALTER DATABASE BEGIN BACKUP statement
When Oracle Database Initiates Checkpoints

2. Tablespace and data file checkpoints

Example: making a tablespace read-only or taking it offline normal, shrinking a data file, or executing ALTER TABLESPACE BEGIN BACKUP

3. Incremental checkpoints

An incremental checkpoint is a type of thread checkpoint partly intended to avoid writing large numbers of blocks at online redo log switches. DBWn checks at least every three seconds to determine whether it has work to do. When DBWn writes dirty buffers, it advances the checkpoint position, causing CKPT to write the checkpoint position to the control file, but not to the data file headers.

4. Other types of checkpoints include instance and media recovery checkpoints and checkpoints when schema objects are dropped or truncated.
Agenda

• Instance Recovery
Overview of Instance Recovery

• An instance failure is any kind of failure that prevents the synchronization of the database’s data files and control files before the instance is shut down.
• Oracle automatically recovers from instance failure during instance recovery.

• Instance recovery is initiated by simply starting up the database with the STARTUP command.

• Instance recovery is also known as crash recovery.

• During a STARTUP operation, Oracle first attempts to read the initialization file, and then it mounts the control file and attempts to open the data files referenced in the control files.

• If the data files are not synchronized, instance recovery is initiated.
Overview of Instance Recovery

- Instance recovery is required when you perform database startup after an instance crash or after stopping the database using SHUTDOWN ABORT.

*Instance and Database Startup Sequence*
Instance Recovery phases:

- **Phase 1** Find data files that are out of sync with the control file.

- **Phase 2** Use the online redo log files to restore the data files to the state before instance failure in a roll-forward operation. After the roll-forward, data files have committed and uncommitted data.

- **Phase 3** Open the database. Once the roll-forward operation completes, the database is open to users.

- **Phase 4** Oracle then uses the undo segments to roll back any uncommitted transactions. The rollback operation uses data in the undo tablespace; without a consistent undo tablespace, the rollback operation cannot succeed. After the rollback phase, the data files contain only committed data.
Instance Recovery phases:

Basic Instance Recovery Steps: Rolling Forward and Rolling Back
Overview of Instance Recovery

- **Instance recovery** is the process of applying records in the online redo log to data files to reconstruct changes made after the most recent **checkpoint**.
- Instance recovery occurs automatically when an administrator attempts to open a database that was previously shut down inconsistently.
Purpose of Instance Recovery

- Instance recovery ensures that the database is in a consistent state after an instance failure. The files of a database can be left in an inconsistent state because of how Oracle Database manages database changes.

- A **redo thread** is a record of all of the changes generated by an instance. A single-instance database has one thread of redo, whereas an Oracle RAC database has multiple redo threads, one for each database instance.

- When a **transaction** is committed, **log writer process (LGWR)** writes both the remaining redo entries in memory and the transaction SCN to the **online redo log**. However, the **database writer (DBW)** process writes modified data blocks to the data files whenever it is most efficient. For this reason, uncommitted changes may temporarily exist in the data files while committed changes do not yet exist in the data files.
Purpose of Instance Recovery

• If an instance of an open database fails, either because of a SHUTDOWN ABORT statement or abnormal termination, then the following situations can result:

1. Data blocks committed by a transaction are not written to the data files and appear only in the online redo log. These changes must be reapplied to the data files.

2. The data files contains changes that had not been committed when the instance failed. These changes must be rolled back to ensure transactional consistency.

• Instance recovery uses only online redo log files and current online data files to synchronize the data files and ensure that they are consistent.
When Oracle Database Performs Instance Recovery

• Whether instance recovery is required depends on the state of the redo threads.

• A redo thread is marked open in the control file when a database instance opens in read/write mode, and is marked closed when the instance is shut down consistently.

• If redo threads are marked open in the control file, but no live instances hold the thread enqueues corresponding to these threads, then the database requires instance recovery.
When Oracle Database Performs Instance Recovery

• Oracle Database performs instance recovery automatically in the following situations:

1. The database opens for the first time after the failure of a single-instance database or all instances of an Oracle RAC database. This form of instance recovery is also called **crash recovery**. Oracle Database recovers the online redo threads of the terminated instances together.

2. Some but not all instances of an Oracle RAC database fail. Instance recovery is performed automatically by a surviving instance in the configuration.

• The SMON background process performs instance recovery, applying online redo automatically. **No user intervention is required.**
Importance of Checkpoints for Instance Recover

- Instance recovery uses checkpoints to determine which changes must be applied to the data files.
- The checkpoint position guarantees that every committed change with an SCN lower than the checkpoint SCN is saved to the data files.

![Checkpoint Position in Online Redo Log](image)
Importance of Checkpoints for Instance Recover

- During instance recovery, the database must apply the changes that occur between the checkpoint position and the end of the redo thread.

- Some changes may already have been written to the data files. However, only changes with SCNs lower than the checkpoint position are guaranteed to be on disk.

![Checkpoint Position in Online Redo Log](image)
Instance Recovery Phases

- The first phase of instance recovery is called **cache recovery** or **rolling forward**, and involves reapplying all of the changes recorded in the online redo log to the data files.
- Because rollback data is recorded in the online redo log, rolling forward also regenerates the corresponding undo segments.
Instance Recovery Phases

- Rolling forward proceeds through as many online redo log files as necessary to bring the database forward in time.
- After rolling forward, the data blocks contain all committed changes recorded in the online redo log files. These files could also contain uncommitted changes that were either saved to the data files before the failure, or were recorded in the online redo log and introduced during cache recovery.

Basic Instance Recovery Steps: Rolling Forward and Rolling Back
Instance Recovery Phases

- After the roll forward, any changes that were not committed must be undone. Oracle Database uses the checkpoint position, which guarantees that every committed change with an SCN lower than the checkpoint SCN is saved on disk.
- Oracle Database applies undo blocks to roll back uncommitted changes in data blocks that were written before the failure or introduced during cache recovery.
- This phase is called **rolling back** or **transaction recovery**.

### Basic Instance Recovery Steps: Rolling Forward and Rolling Back
Instance Recovery Phases

- Oracle Database can roll back multiple transactions simultaneously as needed. All transactions that were active at the time of failure are marked as terminated.

- Instead of waiting for the SMON process to roll back terminated transactions, new transactions can roll back individual blocks themselves to obtain the required data.

Basic Instance Recovery Steps:
Rolling Forward and Rolling Back
Agenda

• Example INSERT-UPDATE-DELETE-COMMIT Scenario
Example INSERT-UPDATE-DELETE-COMMIT Scenario

- created a table with an index as follows:
  ```sql
  create table t(x int, y int);
  create index ti on t(x);
  ```
- what might happen with a set of statements like this:
  ```sql
  insert into t (x,y) values (1,1);
  update t set x = x+1 where x = 1;
  delete from t where x = 2;
  ```

Scenarios:

1. What happens if the system fails at various points in the processing of these statements?
2. What happens if the buffer cache fills up?
3. What happens if we ROLLBACK at any point?
4. What happens if we succeed and COMMIT?
The INSERT

- The initial INSERT INTO T statement will generate both redo and undo.
- The undo generated will be enough information to make the INSERT “go away.”
- The redo generated by the INSERT INTO T will be enough information to make the INSERT “happen again.”
Hypothetical Scenario: The System Crashes Right Now

- System crashes before a COMMIT is issued or before the redo entries are written to disk.
- **Everything is OK.**
- SGA is wiped out, but we don’t need anything that was in the SGA.
- It will be as if this transaction never happened when we restart.
- None of the blocks with changes got flushed to disk, and none of the redo got flushed to disk.
- **We have no need of any of this undo or redo to recover from an instance failure.**
Hypothetical Scenario: The Buffer Cache Fills Up Right Now

- DBWn will start by asking LGWR to flush the redo entries that protect these database blocks.
- Before DBWn can write any of the blocks that are changed to disk, LGWR must flush (to disk) the redo information related to these blocks.
- We need to flush the redo log buffers before writing these blocks out so that we can redo all of the changes necessary to get the SGA back into the state it is in right now, so that a rollback can take place.
- At this point, We have generated some modified table and index blocks. These have associated undo segment blocks, and all three types of blocks have generated redo to protect them.
**Hypothetical Scenario: The Buffer Cache Fills Up Right Now**

- The redo log buffer is flushed *at least* every three seconds, when it is one-third full or contains 1MB of buffered data, or whenever a COMMIT or ROLLBACK takes place.
- It is very possible that at some point during our processing, the redo log buffer will be flushed.
- That is, we’ll have modified blocks representing uncommitted changes in the buffer cache and redo for those uncommitted changes on disk. This is a very normal scenario that happens frequently.

---

*State of the system after a redo log buffer flush*
The UPDATE

• The UPDATE will cause much of the same work as the INSERT to take place. This time, the amount of undo will be larger; we have some “before” images to save as a result of the UPDATE.

*State of the system after the UPDATE*

The dark rectangle in the redo log file represents the redo generated by the INSERT, the redo for the UPDATE is still in the SGA and has not yet been written to disk.
The UPDATE

- We have more new undo segment blocks in the block buffer cache. To undo the UPDATE, if necessary, we have modified database table and index blocks in the cache. We have also generated more redo log buffer entries.

*State of the system after the UPDATE*

Let's assume that our redo generated from the INSERT statement is on disk (in the redo log file) and redo generated from the UPDATE is in cache.
Hypothetical Scenario: The System Crashes Right Now

- Upon startup, Oracle would read the redo log files and find some redo log entries for our transaction. Given the state in which we left the system, we have the redo entries generated by the INSERT in the redo log files (which includes redo for undo segments associated with the INSERT).

- However, the redo for the UPDATE was only in the log buffer and never made it to disk (and was wiped out when the system crashed). That’s okay, the transaction was never committed and the data files on disk reflect the state of the system before the UPDATE took place.
Hypothetical Scenario: The System Crashes Right Now

• However, the redo for the INSERT was written to the redo log file. Therefore Oracle would “roll forward” the INSERT. We would end up with modified undo blocks (information on how to undo the INSERT), modified table blocks (right after the INSERT), and modified index blocks (right after the INSERT).

• Oracle will discover that our transaction never committed and will roll it back since the system is doing crash recovery and, of course, our session is no longer connected.

State of the system after the UPDATE

“roll forward” the INSERT after crash
Hypothetical Scenario: The System Crashes Right Now

• To roll back the uncommitted INSERT, Oracle will use the undo it just rolled forward (from the redo and now in the buffer cache) and apply it to the data and index blocks, making them look as they did before the INSERT took place.

• Now everything is back the way it was. The blocks that are on disk may or may not reflect the INSERT (it depends on whether or not our blocks got flushed before the crash).

• If the blocks on disk do reflect the INSERT, then the INSERT will be undone when the blocks are flushed from the buffer cache. If they do not reflect the undone INSERT, so be it—they will be overwritten later anyway.
Hypothetical Scenario: The System Crashes Right Now

• This scenario covers the rudimentary details of a crash recovery. The system performs this as a two-step process.

• First it rolls forward, bringing the system right to the point of failure, and then it proceeds to roll back everything that had not yet committed.

• This action will resynchronize the data files. It replays the work that was in progress and undoes anything that has not yet completed.
The Delete

- Again, undo is generated as a result of the DELETE, blocks are modified, and redo is sent over to the redo log buffer.
- In fact, it is so similar to the UPDATE.

State of the system after the Delete

Let’s assume that our redo generated from the INSERT & Update statements is on disk (in the redo log file) and redo generated from the Delete is in cache.
Hypothetical Scenario: The Application Rolls Back the Transaction

• At this point, Oracle will find the undo information for this transaction either in the cached undo segment blocks (most likely) or on disk if they have been flushed (more likely for very large transactions).

• It will apply the undo information to the data and index blocks in the buffer cache, or if they are no longer in the cache request, they are read from disk into the cache to have the undo applied to them. These blocks will later be flushed to the data files with their original row values restored.
Hypothetical Scenario: The Application Rolls Back the Transaction

• This scenario is much more common than the system crash. It is useful to note that during the rollback process, the redo logs are never involved.

• The only time redo logs are read for recovery purposes is during recovery and archival. This is a key tuning concept: redo logs are written to. Oracle does not read them during normal processing.

• As long as you have sufficient devices so that when ARCn is reading a file, LGWR is writing to a different device, there is no contention for redo logs.

• Many other databases treat the log files as “transaction logs.” They do not have this separation of redo and undo. For those systems, the act of rolling back can be disastrous—the rollback process must read the logs their log writer is trying to write to. They introduce contention into the part of the system that can least stand it. Oracle’s goal is to make it so that redo logs are written sequentially, and no one ever reads them while they are being written.
The Commit

- Oracle will flush the redo log buffer to disk.
- The modified blocks are in the buffer cache; maybe some of them have been flushed to disk. All of the redo necessary to replay this transaction is safely on disk and the changes are now permanent.

- If we were to read the data directly from the data files, we probably would see the blocks as they existed before the transaction took place, as DBWN most likely has not yet written them.
The Commit

- That’s OK—the redo log files can be used to bring those blocks up to date in the event of a failure. The undo information will hang around until the undo segment wraps around and reuses those blocks. Oracle will use that undo to provide for consistent reads of the affected objects for any session that needs them.

State of the system after commit