Database Administration:

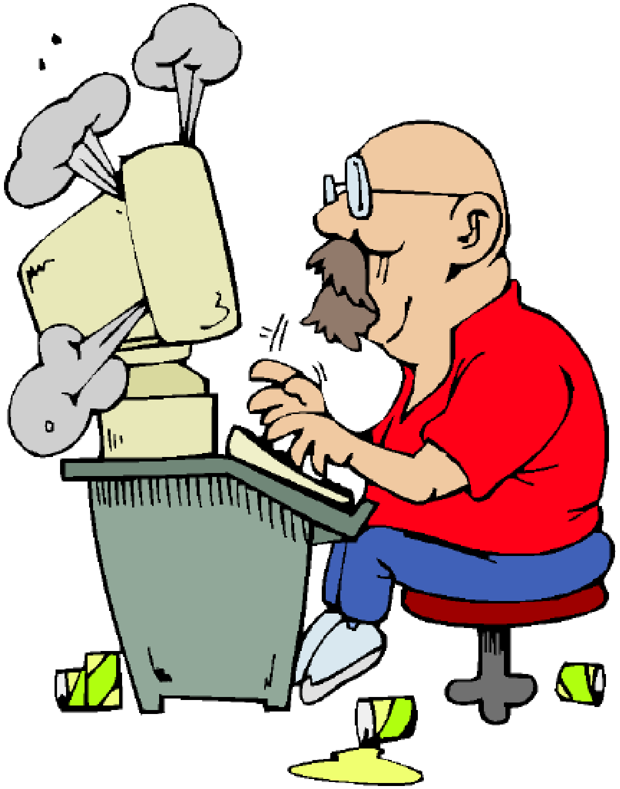
The Complete Guide to Practices and Procedures

Chapter 8

Database Backup and Recovery

Agenda

- The Importance of Backup & Recovery
- Preparing for Problems
- Image Copy Backups
- Recovery
- Alternatives to Backup and Recovery
- Questions



The Importance of Backup & Recovery

- Many DBAs believe that ensuring optimal database and application performance is the most important part of their job…
  - But it is not true.
  - These DBAs are confusing *frequency* with *importance*.
- Recoverability should be at (or near) the very top of the DBA task list, definitely before performance.
  - If you cannot recover your databases after a problem then it won't matter how fast you can access them.
  - Anybody can deliver fast access to the wrong information (or an empty file).

http://www.craigsmullins.com/dbta_076.htm

Preparing For Problems

- *Instance failures*
  - The result of an internal exception within the DBMS, an operating system failure, or other software-related database failure. In some cases, an instance failure can result in corruption of data that requires a recovery, but usually such failures do not damage data, so the DBMS simply needs to be restarted to reestablish normal operations.
- *Application (or transaction) failures*
  - When programs or scripts are run at the wrong time, using the wrong input, or in the wrong order. An application failure usually results in corrupt data that requires a database restore or recovery. The sooner an application failure is identified and corrected, the smaller the amount of damage.
- *Media failure*
  - Includes damage to disk storage devices, file system failures, tape degradation or damage, and deleted data files. Although less common in practice, damaged memory chips also can cause data corruption. After a media failure, the database will likely be in a state where valid data is unreadable, invalid data is readable, or referential integrity is violated.

Backup Plan = Insurance

- It is common for organizations to manage a terabyte or more of data on a single database server.
- A sound backup and recovery plan can be thought of as an insurance policy for your data.
  - You wouldn't go uninsured, would you?

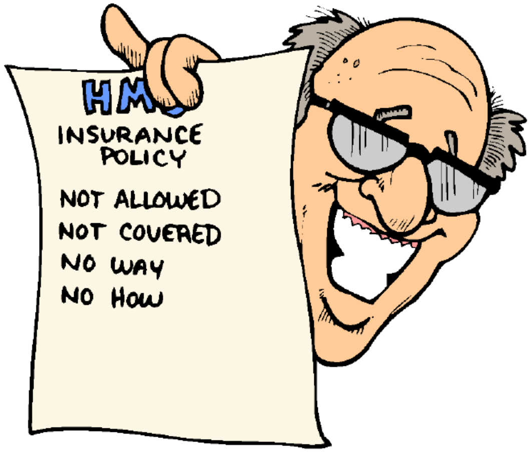


Image Copy Backups

- A fundamental component of a database backup and recovery plan is creating backup copies of data.
- When an error occurs that damages the integrity of the database, a backup copy of the data can be used as the basis to recover or restore the database.
- However, the full story on backing up a database is not quite that simple.

Image Copies

- Backing up databases involves making consistent copies of your data, usually in the form of image copies, which are the output of a COPY utility.
- The name of the copy utility will vary from DBMS to DBMS. Common names for the backup utility include BACKUP, COPY, DUMP, and EXPORT.
- Some DBMSs rely on the native operating system's file system commands for backing up data. However, even if the DBMS supplies an internal backup option, the DBA may choose to use facilities that operate outside the realm of the DBMS.

Assuring Accuracy

- Current and accurate image copies provide the foundation for database recovery.
- The DBA must assure the currency and accuracy of the image copies and base the backup plan on the recovery needs of the applications.
- The DBA will use those recovery requirements to determine how often to take image copy backups and how many backup generations must be kept on hand.
  - The DBA also must make sure that the appropriate log records are available or backed up for recovery purposes.

Factors Influencing Duration of Recovery

- The number of log records that must be processed to recover.
- Whether the log is compacted or compressed
- Whether the image copy backup is encrypted or compressed
- The time it takes an operator to mount and dismount the required tapes
- The time it takes to read the part of the log needed for recovery
- The time needed to reprocess changed pages

Additional Factors

- DBMS Architecture
- Activity During Image Copy Process
  - Read Only
  - Read/Write
- Balancing Duration of Recovery Against the Time Required to Take Image Copy

How Many Backups?

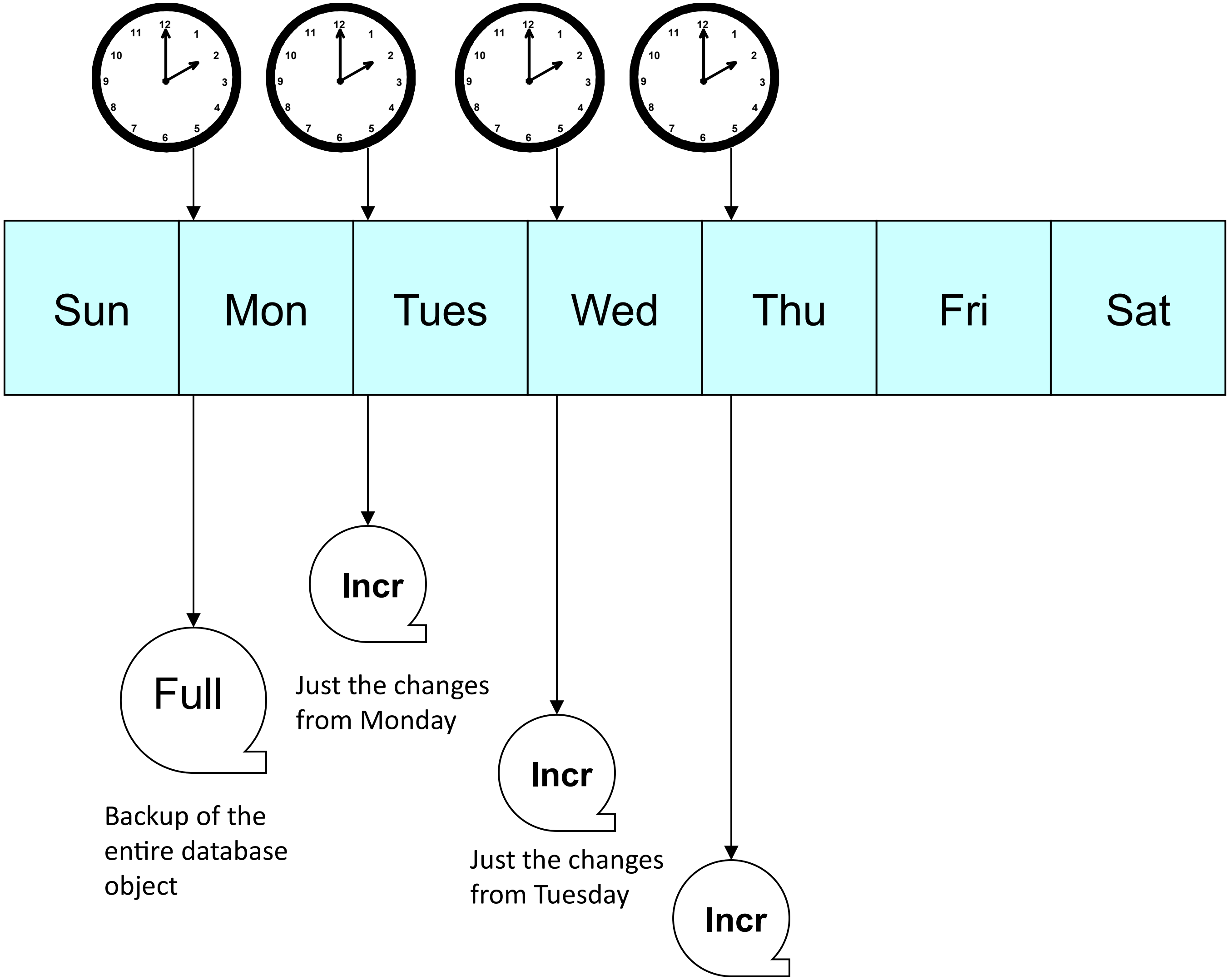
- The DBA must decide how many complete generations of backups (for both database object copies and log copies) to keep.
- Keeping extra generations can help you recover from a media failure during recovery by switching to an older backup.
  - At a minimum, the retention period should be at least two full cycles.
  - The number of copies you decide to keep must be tempered by the number of associated logs that must also be maintained for the backups to remain viable.

General Image Copy Guidelines

- Make at least two local copies of each image copy backup to help avoid an unrecoverable state in the case of a media error
  - For example, a damaged tape.
- Coordinate your local backup strategy with your disaster recovery plans.
- Keep at least two generations of image copy backups for each database object.
- Consider creating image copy backups to disk, and then migrating them to tape (or optical disk, such as CD or DVD), which can speed up the image copy process.
  - When image copy backups are migrated to tape,
- Be sure to include the system catalog database objects in your backup and recovery plans.
- Ensure that the backup process is restartable.
- After the backup has completed, use the DBMS's facilities to verify the correctness of the backup.
  - For example, the DB2 db2ckbkp operation or the Sybase BCP utility.
- Data that is not stored in a database, but is used by database applications, should be backed up at the same time as the database objects.

Full vs. Incremental Backups

- A *full image copy backup* is a complete copy of all the data in the database object at the time the image copy was run.
- An *incremental image copy backup* (aka *differential backup*) contains only the data that has changed since the last full or incremental image copy was made.
  - The advantage of taking an incremental rather than a full backup is that it can sometimes be made more quickly, and requires less space on disk (or tape).
  - The disadvantage is that recovery based on incremental copies can take longer because, in some cases, the same row is updated several times before the last changes are restored.



Sun

Mon

Tues

Wed

Thu

Fri

Sat

Full

Backup of the entire database object

Incr

Just the changes from Monday

Incr

Just the changes from Tuesday

Incr

Just the changes from Wednesday

Incremental Versus Full Image Copy Backups

- Favor full image copies for small database objects.
  - The definition of "small" will vary from site to site and DBMS to DBMS.
- Consider using incremental image copies to reduce the batch processing window for very large database objects that are minimally modified in between image copy backups.
  - The DBA should base the full-versus-incremental decision on the percentage of blocks of data that have been modified, not on the number of rows that have been modified.
- Some scenarios are not compatible with incremental image copy backups.
  - Some DBMSs permit the user to disable logging during some operations and utilities. Whenever an action is taken that adds or changes data without logging, a full image copy is required.

Merging Incremental Image Copies

- A merge utility, sometimes referred to as MERGECOPY, can be used to combine multiple incremental image copy backups into a single incremental copy backup, or to combine a full image copy backup with one or more incremental image copy backups to create a new full backup.
- If your DBMS supports merging incremental copies, consider running the merge utility to create a new full image copy directly after the creation of an incremental copy.

Database Objects and Backups

- Typically, an image copy backup is made at the database, tablespace, or table level.
- The level(s) supported will depend on the DBMS being used.
- In general, though, the idea is to back up the database object or objects that contain the data.
- The more granular control the DBMS provides for backup of database objects, the easier it will be to effectively implement a useful backup and recovery strategy.

Copying Indexes

- Some DBMSs support making backup copies of indexes.
  - Indeed, some DBMSs require indexes to be backed up, whereas index backup is optional for others.
  - Index backup can be optional because the DBMS can rebuild an index from the table data.
- You will need to examine the trade-offs of copying indexes.
- Be sure to perform data and index backups at the same time if you choose to back up rather than rebuild your indexes.
  - Failure to do so can result in indexes that do not match the recovered data

DBMS Control

- The degree of control the DBMS asserts over the backup and recovery process differs from DBMS to DBMS.
  - Some DBMSs record backup and recovery information in the system catalog.
  - That information is then used by the recovery process to determine the logs, log backups, and database backups required for a successful recovery.
- The more information the DBMS maintains about image copy backups, the more the DBMS can control proper usage during recovery.
- If your DBMS does not record backup and recovery information in the system catalog then the DBA must track image copy backup files and assure their proper usage during a recovery.

Concurrent Access Issues

- Concurrent write access allows you to keep the data online during the backup process, but it will slow down any subsequent recovery because the DBMS has to examine the database log to ensure accurate recovery.
- Change accumulation creates an up-to-date image copy backup by merging existing image copies with data from the database logs. This is similar to the merging of incremental image copies.
- Some image copy backup techniques allow only read access to the database object. Backups that allow only read access provide faster recovery than those that allow concurrent read-write because the database log is not needed to ensure a proper recovery.
  - However, they are more disruptive to normal processing.
- Some image copy backup techniques require the database object to be stopped, or completely offline. This type of copy provides fast backup because there is no contention for the tablespace.
  - This is even more disruptive to normal application processing.

Backup Planning Considerations

- The need for concurrent access and modification during the backup process
- The amount of time available for the backup process and the impact of concurrent access on the speed of backing up data
- The speed of the recovery utilities
- The need for access to the database logs
- The difference between a *hot backup* and *cold backup*.

Hot vs. Cold Backup

- A cold backup is accomplished by shutting down the database instance and backing up the relevant database files.
- A hot backup is performed while the database instance remains online, meaning that concurrent access is possible.
- Depending on the capabilities of the DBMS you are using, hot backups can be problematic because:
  - They can be more complex to implement.
  - They can cause additional overhead in the form of higher CPU, additional I/O, and the additional database log archivals.
  - They can require the DBA to create site-specific scripts to perform the hot backup.
  - They require extensive testing to ensure that the backups are viable for recovery.

Backup Consistency

- Be sure your backup plan creates a consistent recovery point for the database object.
  - You need to be aware of all relationships between the database objects being backed up and other database objects including:
    - Application-enforced relationships
    - Referential constraints
    - Triggers
- If you use an image copy backup to recover a database object to a previous point in time, you will need to recover any related database objects to the same point in time.
  - Failure to do so will most likely result in inconsistent data.

Quiesce

- If your DBMS provides a QUIESCE utility:
  - Use it to establish a point of consistency for all related database objects prior to backing them up.
  - QUIESCE halts modification requests to the database objects to ensure consistency and record the point of consistency on the database log.
- If the DBMS does not provide a QUIESCE option:
  - You will need to take other steps to ensure a consistent point for recovery.
  - For example, you can place the database objects into a read-only mode, take the database objects offline, or halt application processes—at least those application processes that update the related database objects.
    - Some recovery options/products can find quiet points without requiring quiesce points during backup.

When to Create a Point of Consistency

- The DBA should create a point of consistency during daily processing.
  - *Before archiving the active log.*
  - *Before copying related database objects.*
  - *Just after creating an image copy backup.*
  - *Just before heavy database modification.*
  - *During quiet times.*

Log Archiving and Backup

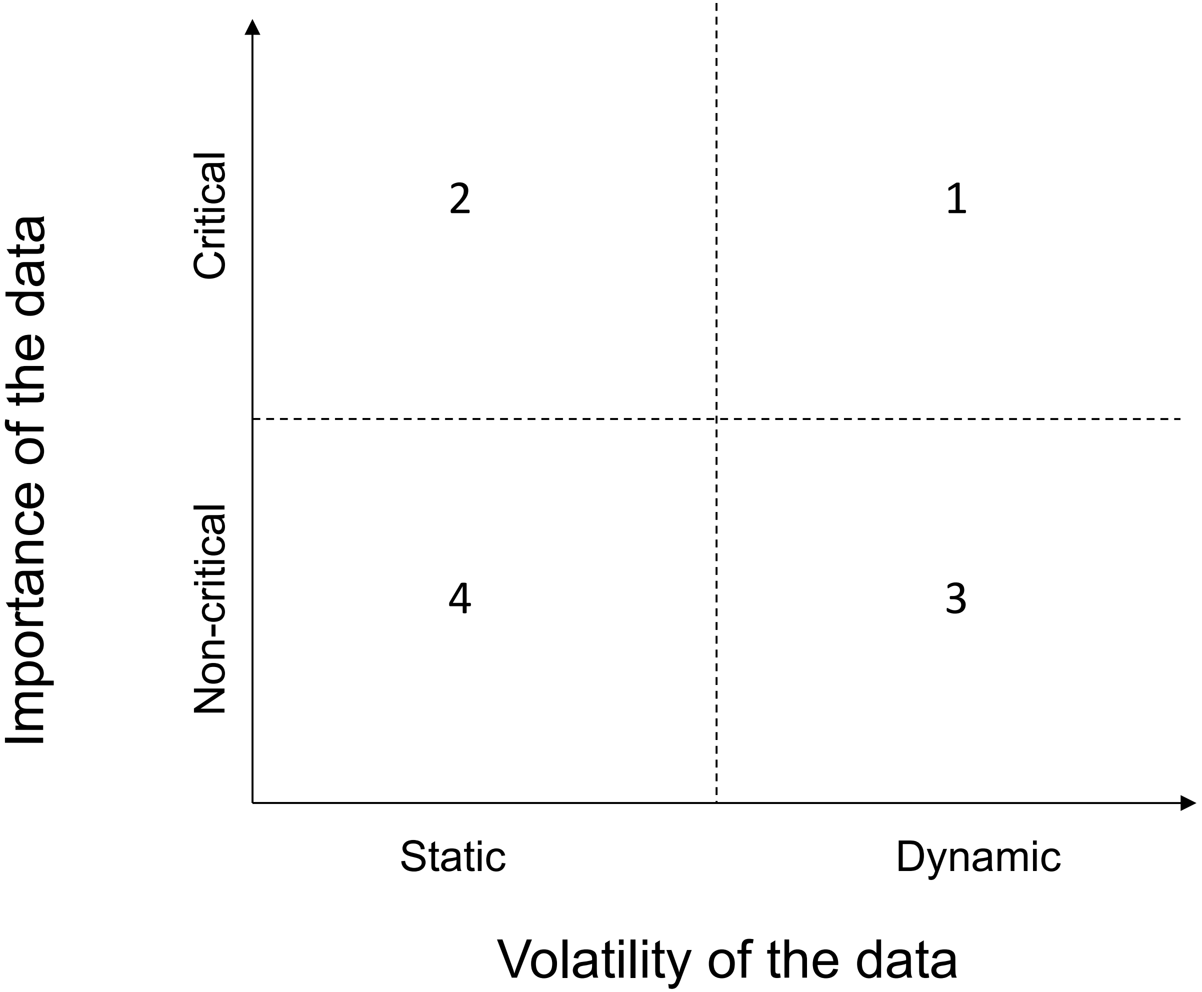
- All database changes are logged by the DBMS to a log file commonly called the *transaction log* or *database log*.
  - Log records are written for every SQL INSERT, UPDATE, and DELETE statement that is successfully executed and committed.
- The database log to which records are currently being written is referred to as the a*ctive log*. As the number of database changes grows, the database log will increase in size.
  - When the active database log is filled, the DBMS invokes a process known as log archival or log offloading.
  - When a database log is archived, the current active log information is moved offline to an archived log file, and the active log is reset.
- The DBA typically controls the frequency of the log archival process by using a DBMS configuration parameter.
  - Most DBMSs also provide a command to allow the DBA to manually request a log archival process.
  - And remember, each DBMS performs log archival and backup differently.

Determining Your Backup Schedule

Not all data is created equal.

- How much daily activity occurs against the data?
- How often does the data change?
- How critical is the data to the business?
- Can the data be recreated easily?
- What kind of access do the users need?
  - Is 24/7 access required?
- What is the cost of *not* having the data available during a recovery?
  - What is the dollar value associated with each minute of downtime?

Criticality and Volatility Grading



Importance of the data

Critical

Non-critical

2

1

4

3

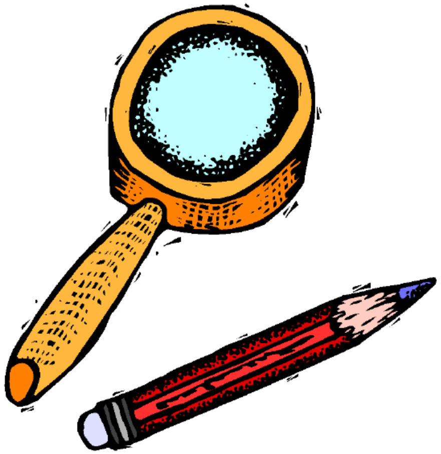
Static

Dynamic

Volatility of the data

DBMS Instance Backup

- In addition to being prepared for failure of individual database objects, the DBA must be prepared to recover from failure of the entire DBMS instance or subsystem.
- Be sure to back up all of the crucial components of the database instance, including:
  - DBMS files
  - System catalog and directory objects
  - Database (archive) logs
  - Configuration and setup files
  - System libraries
  - Tape management libraries
  - Program source libraries and executable libraries.



- Again, each DBMS and platform will have different key components that must be dealt with when planning a recovery strategy for the DBMS instance.

Alternate Approaches to Database Backup

- Using Database Exports to Create Logical Backups
  - Logical backups
- Using Storage Management Software to Make Backup Copies
  - When backing up outside the scope of DBMS control, be sure to disable database write operations for all database objects that are being backed up.
  - Be sure you fully understand both the functionality of the storage management software and the DBMS.
    - For example, some storage management software will not copy open files.

Document Your Backup Strategy

- Thoroughly document and test the backup and recovery strategy, implementation, and procedures.
- Test recovery from:
  - media failure
  - an instance failure
  - and several types of application failures.
- Document the type of backup taken for each database object, along with a schedule of when each is backed up.
- Be sure that all of your databases can be recovered and that all DBAs on-site have firsthand experience at database recovery.
- The DBA group should schedule periodic evaluations of the backup and recovery plans for every production database.

Recovery

- Database recovery can be a very complex task.
- Recovery involves much more than simply restoring an image of the data as it appeared at some earlier point in time.
- A database recovery involves bringing the data back to its state at (or before) the time of the problem.
  - Often a recovery involves restoring databases and then reapplying the correct changes that occurred to that database, in the correct sequence.
- Simply stated, a successful recovery is one where you get the application data to the state you want it—whether that state is how it was last week, yesterday, or just a moment ago.
  - If you planned your backup strategy appropriately, you should be able to recover from just about any type of failure you encounter.

Determining Recovery Options

- What type of failure has occurred: media, transaction, or database instance?
- What is the cause of the failure?
- How did the database go down: abort, crash, normal shutdown?
- Did any operating system errors occur?
- Was the server rebooted?
- Are there any errors in the operating system log?
- Are there any errors in the alert log?
- Was a dump produced?
- Were any trace files generated?
- How critical is the lost data?
- Have you attempted any kind of recovery so far? If so, what steps have already been performed?
- What types of backups exist: full, incremental, both?
- What needs to be recovered: the full database, a tablespace, a single table, an index, or combinations thereof?
- Does your backup strategy support the type of recovery required (recover-to-current vs. point-in-time)?
- If you have cold backups, how was the database shut down when the cold backups were taken?
- Are all of the archived database logs available for recovery?
- Do you have recent logical backup (EXPORT or UNLOAD)?
- What concurrent activities were running when the system crashed?
- Can you bring the DBMS instance up?
- Can you access the database objects?
- What are your system availability requirements?
- How much data must be recovered?
- Are you using raw files?

DBMS Version Migration and Recovery

- DBMS version migration can impact recoverability.
- Sometimes the DBMS vendors change the format of image copy backup files, rendering any backups using the old format unusable. The same could be true for the log file—the format may have changed for a new version, rendering
  - Depending on the DBMS and the particulars of the new version, a backup taken in a prior release may not be usable for recovery after migration.
  - Alternately, a backup taken after migration that is trying to be used after falling back to an older version of the DBMS also may not be usable for recovery.

General Steps for Database Object Recovery

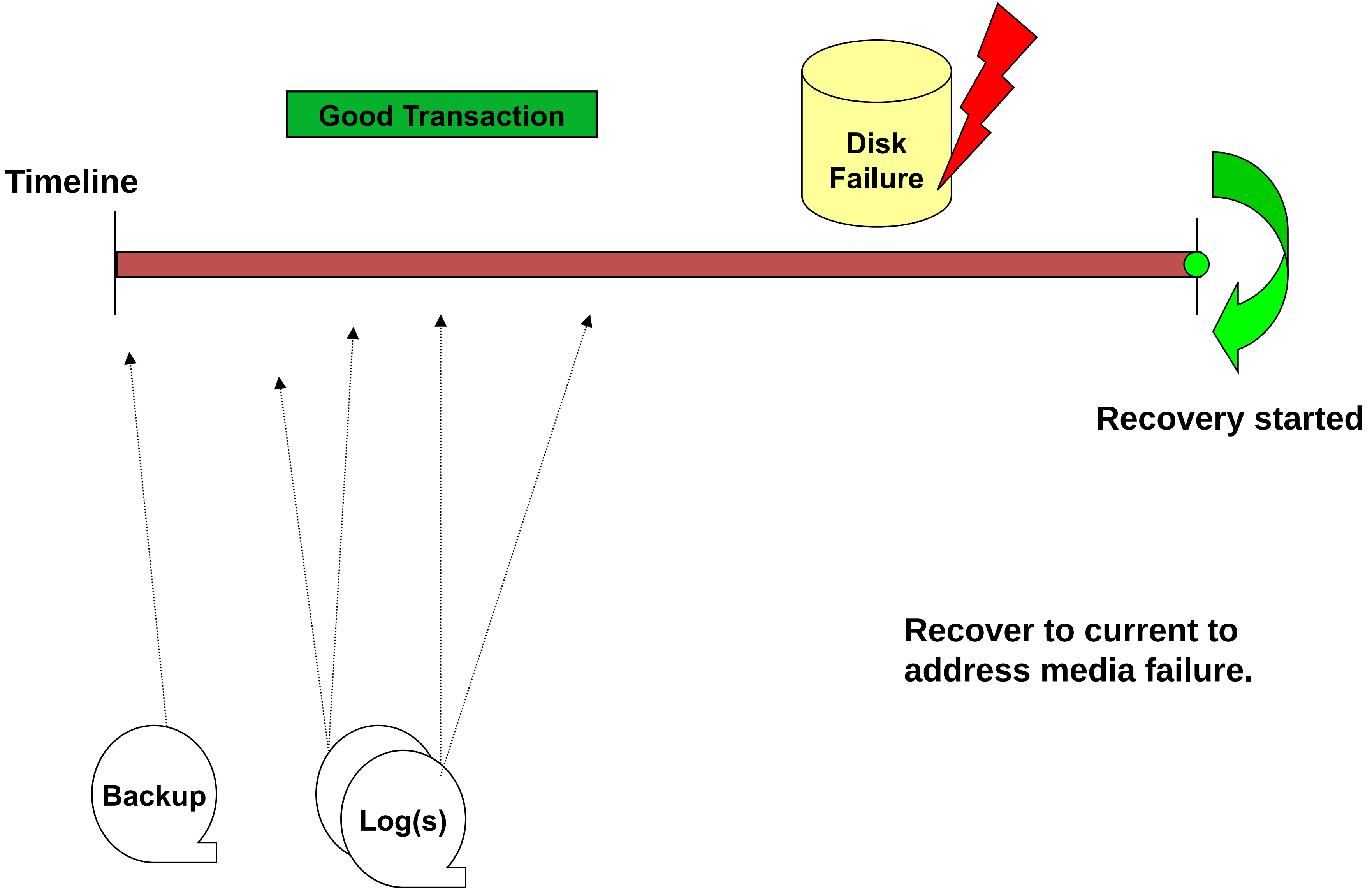
At the very basic level, every database recovery will involve most of these seven steps:

1. *Identify the failure.*
2. *Analyze the situation.*
3. *Determine what needs to be recovered.*
4. *Identify dependencies between the database objects to be recovered.*
5. *Locate the required image copy backup(s).*
6. *Restore the image copy backup(s).*
7. *Roll forward through the database log(s).*

Types of Recovery

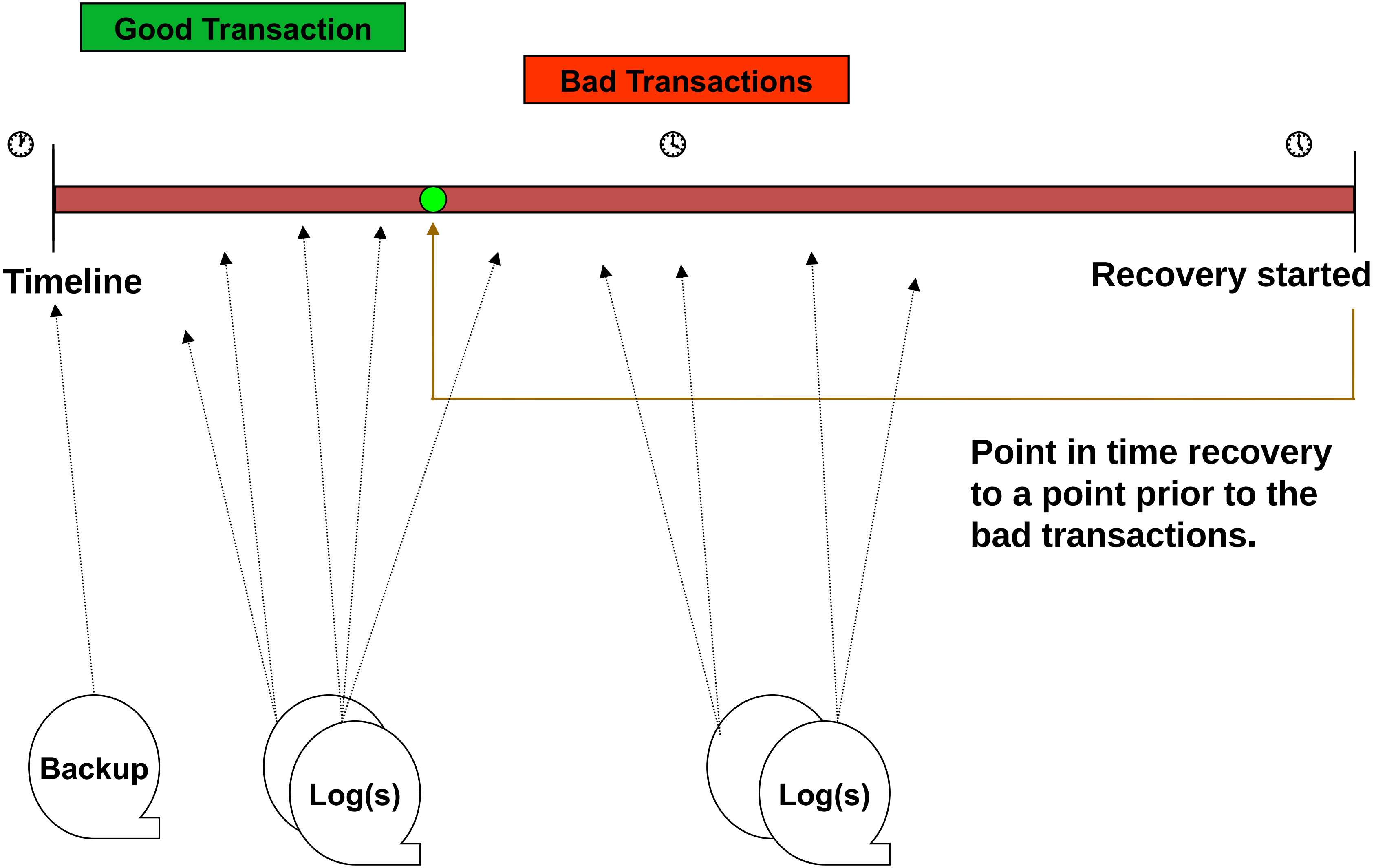
- Recovery to Current
- Point-in-Time (PiT) Recovery
- Transaction Recovery

Recovery to Current



Recover to current to address media failure.

Point-in-Time Recovery



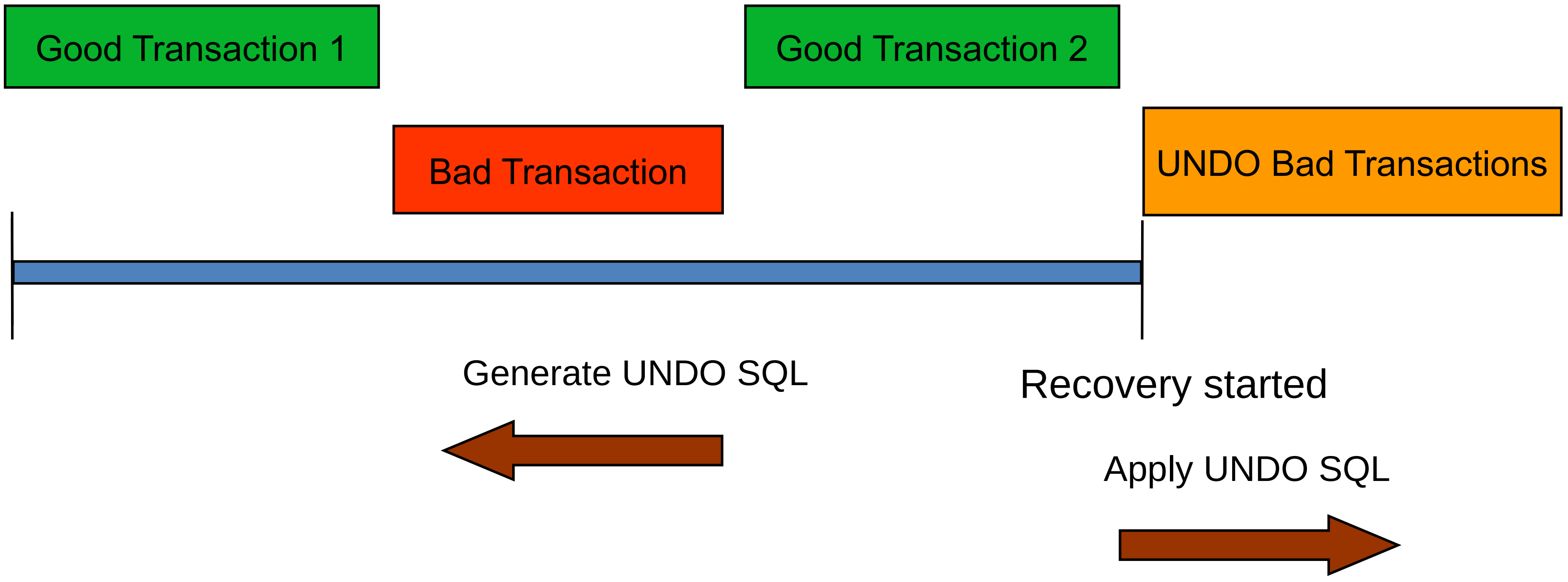
Point in time recovery to a point prior to the bad transactions.

Transaction Recovery

- Tradition recovery to current and PIT recover at the database object level.
- Transaction recovery allows a user to recover a specific portion of the database based on user-defined criteria. This can be at:
  - a transaction or
  - application program level.
- Examples of user-level transaction definitions might be
  - All database updates performed by a userid since last Wednesday at 11:50 A.M.
  - All database deletes performed by the application program named PAYROLL since 8:00 P.M. yesterday.

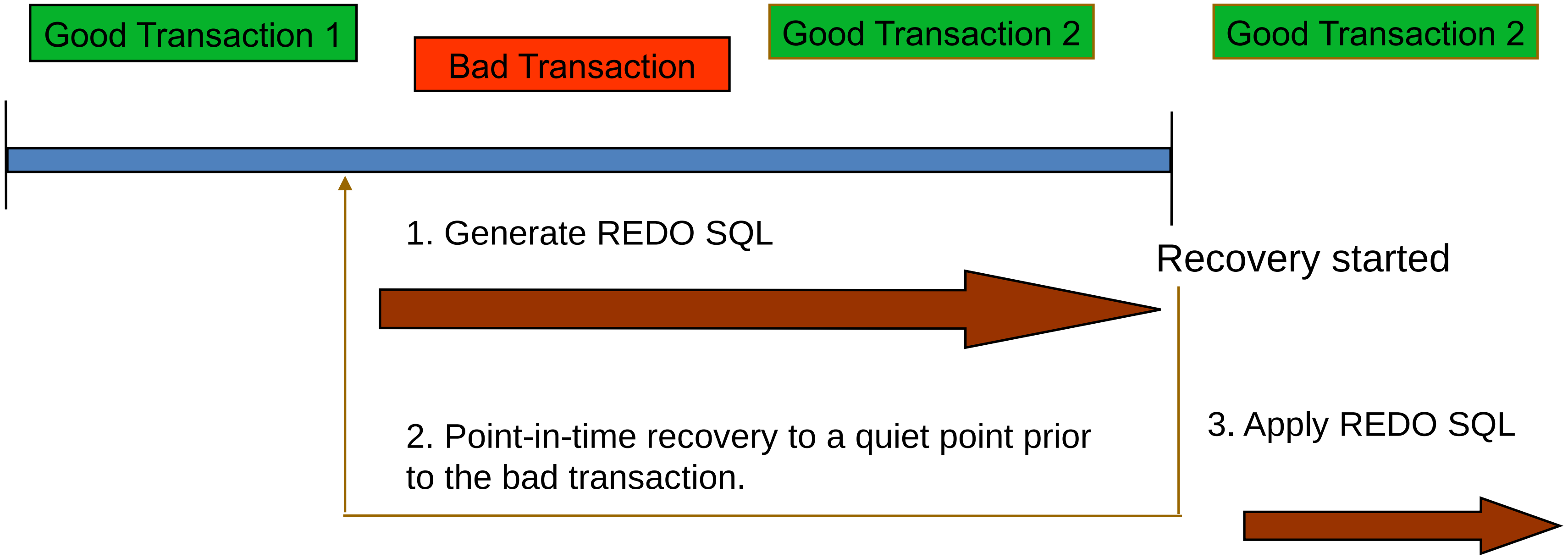
http://findarticles.com/p/articles/mi_m0BRZ/is_4_21/ai_77058262/

UNDO Transaction Recovery



UNDO SQL, generated from the database log, can be used to get rid of bad transactions. And the database can remain online.

REDO Transaction Recovery



You can perform a point in time recovery and then re-apply good transactions using REDO SQL. The database is briefly offline during the PIT recovery, then back online.

Choosing the Optimum Recovery Strategy

- ***Transaction Identification*.** Can all the problem transactions be identified? You must be able to actually identify the transactions that will be removed from the database for transaction recovery to work. Can all the work that was originally done be located and redone?
- ***Data Integrity*.** Has anyone else updated the rows since the problem occurred? If they have, can you still proceed? Is all the data that is required still available? Intervening reorganizations, loads, or mass deletes can require the use of an image copy backup, thereby eliminating UNDO recovery. Will the recovery cause any other data to be lost? If so, can the lost data be identified in some fashion and reapplied?
- ***Speed*.** If multiple techniques are viable, which one is likely to perform the fastest? How many database logs are required to perform the recovery? Can anything be done to reduce the number of logs, such as merging incremental copies?
- ***Availability*.** How soon can the application become available again? Can you afford to go offline?
- ***Invasiveness*.** How invasive was the failure to your database? Were decisions made based on bad data? Can *any* subsequent work be trusted?

Factors Influencing Recovery Duration

- The smaller the size of the components that need to be recovered, the shorter the recovery process will be.
- Recovering at the partition level can lessen recovery duration. Sometimes a failure that would otherwise impact an entire database object can be limited to impacting only a single partition.
- Keeping image copy backups and log archive files on disk (instead of tape or CD/DVD) can speed up the recovery process.
- Test your image copy backups to make sure they are valid.
- Automate your backup and recovery procedures to the greatest extent possible.
- Databases with few dependencies can minimize the duration of a recovery because fewer related database objects may need to be recovered at the same time.
- Be sure that every DBA understands the recovery procedures for each database object under his or her control.

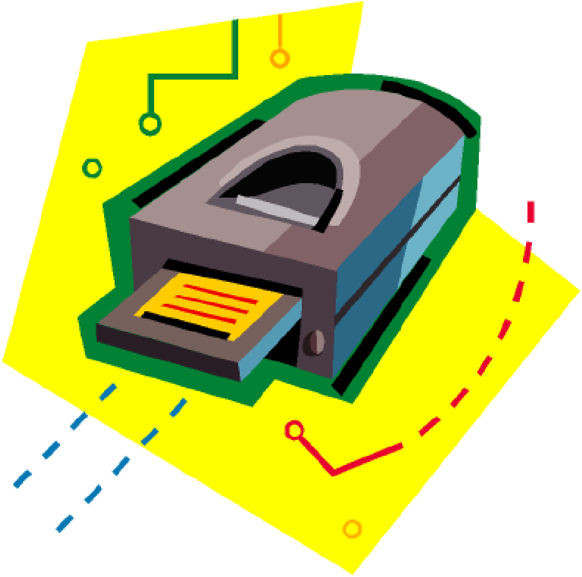
Matching Type of Failure to Type of Recovery

- Match the type of failure to the appropriate type of recovery.
  - Recovering from a *media failure* usually involves a recover to current.
  - Recovering from a *transaction failure* usually involves a point-in-time recovery or a transaction recovery.
  - Recovering from a *database instance or subsystem failure* will most likely involve a recover to current.

Index Recovery

There are two options for index recovery:

- Rebuilding the index from the table data, or
- Recovering the index from a backup copy of the index itself.



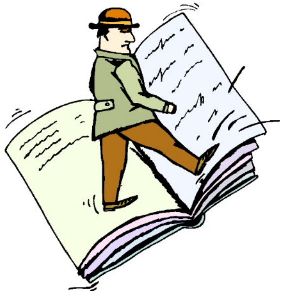
Testing Your Recovery Plan

- You should develop a recovery plan and test it often (ideally no less than twice per year).
- To develop your recovery plan:
  - Write all aspects of the recovery plan out in detail, documenting each step.
  - Include all the scripts required to back up and recover each database object.
  - Review the plan with everyone who may be called on to implement it.
  - Include a contact list with names and phone numbers of everyone who may be involved in the recovery.
  - Keep the recovery plan up-to-date by modifying the plan to include every new database object that is created.

Recovering a Dropped Database Object

- Recovering a dropped object requires extra steps beyond a normal recovery.
- Depending on the DBMS and the tools available, it can sometimes be very complicated.
- Each DBMS identifies the database objects under its control by an internal identifier.
  - When an object is dropped and recreated, the internal identifier for that object usually will change.
  - Therefore, recreating the object using the same DDL and running a recovery using a prior image copy backup usually will not work.
- To recover a dropped database object, the DBA may need to translate the internal identifier of the old database object to the internal identifier of the new database object.

Recovering Broken Blocks and Pages



- A broken block or page is a section of a tablespace or index that contains bad or inconsistent data.
  - Data may be inconsistent due to a broken or orphaned chain, referential constraint violations, a damaged recovery log, a missing or extra index entry, or some other arcane problem.
- To recover an index with a broken page you can simply rebuild the index from the data in the tablespace.
- Tablespaces are a different proposition.
  - Sometimes simply stopping and starting the tablespace or recycling the DBMS instance can fix a broken page.
  - Some DBMSs come with a repair utility that can be used to pinpoint locations within a file based on offsets and replace data at the bit or byte level.
    - Before using any such repair tool, be sure to completely read the DBMS instruction manuals.
    - Repair utilities can be invasive and damaging to the contents of the database.
  - Once you have repaired the information, you may need to recover the tablespace to current.

Alternatives to Backup & Recovery

- Standby Databases
- Replication
  - Snapshot replication
  - Symmetric replication
- Disk Mirroring

These technologies do not completely replace the need to backup and recovery your database objects. They can be used to augment your database recovery planning.

Questions