ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**TheoryAssignment #9**

**SUBMITTED BY:**

**Apil Neupane**

**013BSCCSIT008**

**SUBMITTED TO:**

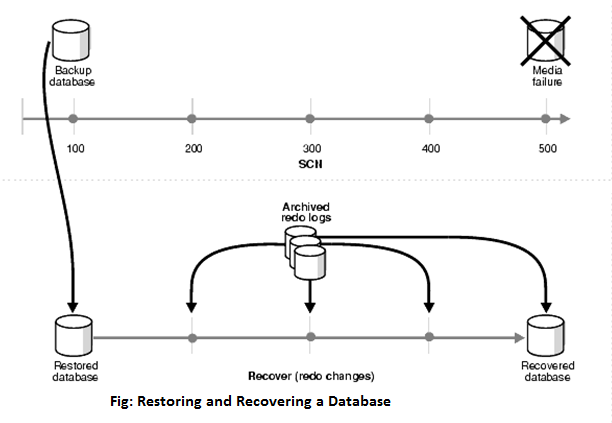
|  |  |
| --- | --- |
| **Er. Sanjay Kr. Yadav**  **Lecturer** |  |
| **Department of Computer Science** | |

Submission Date: 2nd October 2015

**DATABASE RECOVERY:**

Data recovery is the process of restoring data that has been lost, accidentally deleted, corrupted or made inaccessible for any reason.

The data recovery process may vary, depending on the circumstances of the data loss, the data recovery software used to create the backup, and the backup target media. For example, many desktop and laptop backup software platforms allow end users to restore lost files themselves, while restoration of a corrupted database from a [tape backup](http://searchstorage.techtarget.com/definition/tape-backup) is a more complicated process that requires IT intervention. Data recovery can also be provided as service. Such services are typically used to retrieve important files that were not backed up and accidentally deleted from a computer's file system but still remain on disk in fragments.



1. **PURPOSE OF DATA RECOVERY:**

The purpose of this policy is as follows:

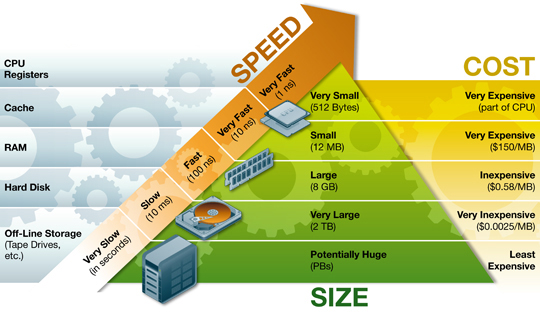
* To provide secure storage for data assets critical to the work flow of official  university business
* To prevent loss of data in the case of accidental deletion / corruption of data, system failure, or disaster
* To permit timely restoration of archived data in the event of a disaster or system failure.

1. **TYPES OF FAILURE:**

* **Transaction failures:** overflow, interrupt, data not available, explicit rollback, concurrency enforcement, programming errors – no memory loss.
* **System Crashes:** due to hardware or software errors – main memory content is lost
* **User Error: caused due the mistake of user.**
* **Carelessness: Caused due to misbehave and neglecting habit of user.**
* **Sabotage: intentional corruption of data**
* **Statement Failure: Caused due to error in code**
* **Application software errors: caused due to problems in application**
* **Network Failure: caused due to failure to connect to network.**
* **Media Failure:** problems with disk head, unreadable media surface – (parts of ) information on secondary storage may be lost
* **Natural Physical Disasters:** fire, flood, earthquakes, theft, etc. – physical loss of all information on all media

1. **THE STORAGE HIERARCHY:**

The range of memory and storage within and attached to a computer system is known as the Storage Hierarchy and to help understand this further can be categorized into 4 segments. As memory and storage devices move down the hierarchy they reduce in performance and cost/MB or GB but tend to rise in capacity through to the last category which includes removable media which in effect has no restriction on capacity a device can store.



* **Primary Storage** is the top level and is made up of CPU registers, CPU cache and memory which are the only components that are directly accessible to the systems CPU. The CPU can continuously read data stored in these areas and execute all instructions as required quickly in a uniform manner. Secondary Storage differs from primary storage in that it is not directly accessible by the CPU. A system uses input/output (I/O) channels to connect to the secondary storage which control the data flow through a system when required and on request
* **Secondary storage** is non-volatile so does not lose data when it is powered down so consequently modern computer systems tend to have a more secondary storage than primary storage. All secondary storage today consist of hard disk drives (HDD), usually set up in a RAID configuration, however older installations also included removable media such us magneto optical or MO
* **Tertiary Storage**is mainly used as backup and archival of data and although based on the slowest devices can be classed as the most important in terms of data protection against a variety of disasters that can affect an IT infrastructure. Most devices in this segment are automated via robotics and software to reduce management costs and risk of human error and consist primarily of disk & tape based back up devices
* **Offline Storage** is the final category and is where removable types of storage media sit such as tape cartridges and optical disc such as CD and DVD. Offline storage is can be used to transfer data between systems but also allow for data to be secured offsite to ensure companies always have a copy of valuable data in the event of a disaster.

1. **BUFFER MANAGEMENT:**

Buffer management is a key component in achieving this efficiency. The buffer management component consists of two mechanisms: thebuffer manager to access and update database pages, and the buffercache (also called the buffer pool), to reduce database file I/O.

The buffer manager interacts primarily with the following components:

* Resource manager to control overall memory usage and, in 32-bit platforms, to control address space usage.
* Database manager and the SQL Server Operating System (SQLOS) for low-level file I/O operations.
* Log manager for write-ahead logging.

1. **TRANSACTION LOG:**

A transaction log is a file – integral part of every SQL Server database. It contains log records produced during the logging process in a SQL Server database. The transaction log is the most important component of a SQL Server database when it comes to the disaster recovery – however, it must be uncorrupted. After each database modification – transaction occurrence, a log record is written to the transaction log. All the changes are written sequentially.

In Conclusion, the transaction log is a critical component of the database and, if there is a system failure, the transaction log might be required to bring your database back to a consistent state. The transaction log should never be deleted or moved unless you fully understand the ramifications of doing this.

1. **DATA UPDATES:**
2. **Immediate Update:** As soon as a data item is modified in cache, the disk copy is updated.
3. **Deferred Update:** All modified data items in the cache is written either after a transaction ends its execution or after a fixed number of transactions have completed their execution.
4. **Shadow update:** The modified version of a data item does not overwrite its disk copy but is written at a separate disk location.

**In-place update:** The disk version of the data item is overwritten by the cache version.

1. **DATA CACHING:**

Many applications today are being developed and deployed on multi-tier environments that involve browser-based clients, web application servers and backend databases. These applications need to generate web pages on-demand by talking to backend databases because of their dynamic nature, making middle-tier database caching an effective approach to achieve high scalability and performance.

**Benefits:**

1. **Scalability**: distribute query workload from backend to multiple cheap front-end systems.
2. **Flexibility**: achieve QoS, where each cache hosts different parts of the backend data, e.g., the data of Platinum customers are cached while that of ordinary customers are not.
3. **Availability**: by continued service for applications that depend only on cached tables even if the backend server is unavailable.
4. **Performance**: by potentially responding fast because of locality of data and smoothing out load peaks by avoiding round-trips between middle-tier and data-tier
5. **TRANSACTION ROLL BACK(UNDO) AND ROLL FRWARD:**

RollBack is the process of undoing changes and reverting to a previous state. This usually occurs either on request when a program detects some logical error and decides the transaction should not take place, or, when the DBMS loses contact with the program before an explicit "COMMIT" has been requested.

RollForward occurs when the database restarts after an abnormal shutdown. It’s a process of going to the log files and applying changes from the log files to the underlying database. In the case where the underlying tables have been restored from an old backup this can involve millions of updates and take several hours.

1. **CHECK POINTING, SHADOW PAGING:**
2. **CHECK POINTING:**

**Checkpointing** is a technique to add [fault tolerance](https://en.wikipedia.org/wiki/Fault_tolerance) into [computing](https://en.wikipedia.org/wiki/Computing) systems. It basically consists of saving a snapshot of the [application](https://en.wikipedia.org/wiki/Application_software)'s state, so that it can restart from that point in case of [failure](https://en.wikipedia.org/wiki/Failure). This is particularly important for long running application that are executed in vulnerable computing system.

Checkpointing is the process of associating a resource with one or more registry keys so that when the resource is moved to a new [node](https://msdn.microsoft.com/en-us/library/aa371745(v=vs.85).aspx)(during [failover](https://msdn.microsoft.com/en-us/library/aa369573(v=vs.85).aspx), for example), the required keys are propagated to the local registry on the new node.

The [Failover Cluster API](https://msdn.microsoft.com/en-us/library/aa372945(v=vs.85).aspx) allows cryptographic key containers to be checkpointed as well as normal registry keys. The following rules apply to checkpoints:

* Whenever anything changes on the check pointed registry tree and the resource is online, the [Cluster service](https://msdn.microsoft.com/en-us/library/aa369163(v=vs.85).aspx) stores a copy of the tree on the [quorum resource](https://msdn.microsoft.com/en-us/library/aa371819(v=vs.85).aspx).
* A change made to a check pointed key while the resource is offline will be overwritten with the check pointed data when the application comes online.
* If the resource moves to another node, the Cluster service restores the registry tree from the quorum resource log file to the registry on the new node before the resource is brought online.
* If the resource is deleted, the checkpoint is deleted.
* Checkpoints are included in backups created by the [**Backup Cluster Database**](https://msdn.microsoft.com/en-us/library/aa367185(v=vs.85).aspx) function.
* Multiple resource instances on different nodes must be handled carefully. Consider the situation where Resource A[0] stores data[0] in checkpoint A on node 0. Resource A[1] stores data[1] in checkpoint A on node 1. If Resource A[1] fails over to node 0, the Cluster service will replace data[0] with data[1] in checkpoint A. If Resource A[0] depends on data[0], it is likely to fail. One solution to this problem is to give the check pointed keys different names on different nodes.

1. **SHADOW PAGING:**

In [computer science](https://en.wikipedia.org/wiki/Computer_science), shadow paging is a technique for providing [atomicity](https://en.wikipedia.org/wiki/Atomic_(computer_science)) and [durability](https://en.wikipedia.org/wiki/Durability_(computer_science)) (two of the [ACID](https://en.wikipedia.org/wiki/ACID) properties) in [database systems](https://en.wikipedia.org/wiki/Database_system). A *page* in this context refers to a unit of physical storage (probably on a [hard disk](https://en.wikipedia.org/wiki/Hard_disk)), typically of the order of 210 to 216 [bytes](https://en.wikipedia.org/wiki/Byte).

Shadow paging is a [copy-on-write](https://en.wikipedia.org/wiki/Copy-on-write) technique for avoiding [in-place](https://en.wikipedia.org/wiki/In-place) updates of pages. Instead, when a page is to be modified, a shadow page is allocated. Since the shadow page has no references (from other pages on disk), it can be modified liberally, without concern for consistency constraints, etc. When the page is ready to become [durable](https://en.wikipedia.org/wiki/Durability_(computer_science)), all pages that referred to the original are updated to refer to the new replacement page instead. Because the page is "activated" only when it is ready, it is [atomic](https://en.wikipedia.org/wiki/Atomic_(computer_science)).

1. **RECOVERY SCHEMES(WAL: WRITE AHEAD LOGGING PROTOCOLS):**

In [computer science](https://en.wikipedia.org/wiki/Computer_science), **write-ahead logging** (**WAL**) is a family of techniques for providing [atomicity](https://en.wikipedia.org/wiki/Atomic_(computer_science)) and [durability](https://en.wikipedia.org/wiki/Durability_(database_systems)) (two of the [ACID](https://en.wikipedia.org/wiki/ACID) properties) in [database systems](https://en.wikipedia.org/wiki/Database_system).

In a system using WAL, all modifications are written to a [log](https://en.wikipedia.org/wiki/Database_log) before they are applied. Usually both redo and undo information is stored in the log.

The purpose of this can be illustrated by an example. Imagine a program that is in the middle of performing some operation when the machine it is running on loses power. Upon restart, that program might well need to know whether the operation it was performing succeeded, half-succeeded, or failed. If a write-ahead log is used, the program can check this log and compare what it was supposed to be doing when it unexpectedly lost power to what was actually done. On the basis of this comparison, the program could decide to undo what it had started, complete what it had started, or keep things as they are.

WAL allows updates of a database to be done [in-place](https://en.wikipedia.org/wiki/In-place_algorithm). Another way to implement atomic updates is with [shadow paging](https://en.wikipedia.org/wiki/Shadow_paging), which is not in-place. The main advantage of doing updates in-place is that it reduces the need to modify indexes and block lists.

[ARIES](https://en.wikipedia.org/wiki/Algorithms_for_Recovery_and_Isolation_Exploiting_Semantics) is a popular algorithm in the WAL family.

[File systems](https://en.wikipedia.org/wiki/File_system) typically use a variant of WAL for at least file system [metadata](https://en.wikipedia.org/wiki/Metadata) called [journaling](https://en.wikipedia.org/wiki/Journaling_file_system).

1. **FAILURE WITH LOSS OF NON-VOLATILE STORAGE(GENERAL CONCEPTS):**

A volatile storage like RAM stores all the active logs, disk buffers, and related data. In addition, it stores all the transactions that are being currently executed. What happens if such a volatile storage crashes abruptly? It would obviously take away all the logs and active copies of the database. It makes recovery almost impossible, as everything that is required to recover the data is lost.

Following techniques may be adopted in case of loss of volatile storage −

* We can have **checkpoints** at multiple stages so as to save the contents of the database periodically.
* A state of active database in the volatile memory can be periodically**dumped** onto a stable storage, which may also contain logs and active transactions and buffer blocks.
* <dump> can be marked on a log file, whenever the database contents are dumped from a non-volatile memory to a stable one.

1. **RECOVERY IN MULTIDATABASE SYSTEM:**

To maintain the atomicity of a multidatabase transaction, it is necessary to have a two-level recovery mechanism. A global recovery manager, or coordinator, is needed to maintain information needed for recovery, in addition to the local recovery managers and the information they maintain (log, tables). The coordinator usually follows a protocol called the two-phase commit protocol, whose two phases can be stated as follows:

* **Phase 1:** When all participating databases signal the coordinator that the part of the multi database transaction involving each has concluded, the coordinator sends a message "prepare for commit" to each participant to get ready for committing the transaction. Each participating database receiving that message will force-write all log records and needed information for local recovery to disk and then send a "ready to commit" or "OK" signal to the coordinator. If the force-writing to disk fails or the local transaction cannot commit for some reason, the participating database sends a "cannot commit" or "not OK" signal to the coordinator. If the coordinator does not receive a reply from a database within a certain time out interval, it assumes a "not OK" response.
* **Phase 2:** If all participating databases reply "OK," and the coordinator’s vote is also "OK," the transaction is successful, and the coordinator sends a "commit" signal for the transaction to the participating databases. Because all the local effects of the transaction and information needed for local recovery have been recorded in the logs of the participating databases, recovery from failure is now possible. Each participating database completes transaction commit by writing a [commit] entry for the transaction in the log and permanently updating the database if needed. On the other hand, if one or more of the participating databases or the coordinator have a "not OK" response, the transaction has failed, and the coordinator sends a message to "roll back" or UNDO the local effect of the transaction to each participating database. This is done by undoing the transaction operations, using the log.