**ST.XAVIER’S COLLEGE**

MAITIGHAR, KATHMANDU



Database Management System

Assignment #9

Submitted By:

Pratik Gautam

013BSCCSIT014

2nd year/ 4th semester

Submitted to:

|  |  |
| --- | --- |
| Er. Sanjay Kumar Yadav  Lecturer  Department of Computer Science |  |

**DATABASE RECOVERY**

**Introduction**

Recovery processes vary depending on the type of failure that occurred, the structures affected, and the type of recovery that you perform. If no files are lost or damaged, recovery may amount to no more than restarting an instance. If data has been lost, recovery requires additional steps

**Purpose of Data Recovery**

Data recovery is the process of restoring data that has been lost, accidentally deleted, corrupted or made inaccessible for any reason. Data recovery typically refers to the restoration of data to a desktop, laptop, server, or external storage system from a backup. The purposes of Data Recovery are:

* Planning and testing responses to different kinds of failures
* Configuring the database environment for backup and recovery
* Setting up a backup schedule
* Monitoring the backup and recovery environment
* Troubleshooting backup problems
* Recovering from data loss if the need arises

## Types of failure

Failures may be

|  |  |
| --- | --- |
| Transaction | Caused by errors within the transaction processes. |
| System | Caused by failure of network or operating system or physical threats to the system as a whole. |
| Media | Failure of hard disk, out of memory errors, out of disk space errors. |

## The Storage Hierarchy

Data are the principal resources of an organization. Data stored in computer systems form a hierarchy extending from a single bit to a database, the major record-keeping entity of a firm. Each higher rung of this hierarchy is organized from the components below it.Data are logically organized into:

Databases are stored in file formats, which contain records. At physical level, the actual data is stored in electromagnetic format on some device. These storage devices can be broadly categorized into three types −



* **Primary Storage** − The memory storage that is directly accessible to the CPU comes under this category. CPU's internal memory (registers), fast memory (cache), and main memory (RAM) are directly accessible to the CPU, as they are all placed on the motherboard or CPU chipset. This storage is typically very small, ultra-fast, and volatile. Primary storage requires continuous power supply in order to maintain its state. In case of a power failure, all its data is lost.
* **Secondary Storage** − Secondary storage devices are used to store data for future use or as backup. Secondary storage includes memory devices that are not a part of the CPU chipset or motherboard, for example, magnetic disks, optical disks (DVD, CD, etc.), hard disks, flash drives, and magnetic tapes.
* **Tertiary Storage** − Tertiary storage is used to store huge volumes of data. Since such storage devices are external to the computer system, they are the slowest in speed. These storage devices are mostly used to take the back up of an entire system. Optical disks and magnetic tapes are widely used as tertiary storage.

**Buffer management**

* The primary purpose of a SQL Server database is to store and retrieve data, so intensive disk I/O is a core characteristic of the Database Engine. And because disk I/O operations can consume many resources and take a relatively long time to finish, SQL Server focuses on making I/O highly efficient. Buffer management is a key component in achieving this efficiency.
* The buffer management component consists of two mechanisms: the buffer manager to access and update database pages, and the buffer cache (also called the buffer pool), to reduce database file I/O.
* The subsystem responsible for the allocation of buffer space is called the buffer manager
* The buffer manager must use some sophisticated techniques in order to provide good service:
* Pinned Blocks: For the database to be able to recover from crashes, we need to restrict times when a block maybe written back to disk. A block not allowed to be written is said to be pinned. Many operating systems do not provide support for pinned blocks, and such a feature is essential if a database is to be ``crash resistant''.

**Transaction Log**

* Every SQL Server database has a transaction log that records all transactions and the database modifications made by each transaction. The transaction log must be truncated on a regular basis to keep it from filling up. However, some factors can delay log truncation, so monitoring log size is important. Some operations can be minimally logged to reduce their impact on transaction log size.
* 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.

**Data Updates**

* Immediate Update: As soon as a data item is modified in cache, the disk copy is updated.
* 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.
* 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.

**Data caching**

A cache, in computing, is a data storing technique that provides the ability to access data or files at a higher speed.

A cache works in both hardware and software to provide similar functionality. In its physical or hardware form, it is a small form factor of internal memory that stores instances of the most frequently executed programs in the main memory to enable faster access when they are requested by the CPU.

A very common example of caching is in a Web browser, where a website's HTML, images, CSS, JavaScript, etc is cached locally so that a page will load faster after its first hit.

**Transaction roll back(Undo)& roll forward**

Roll forward:

The Roll forward is redoing the changes made by a transaction that is after the committed transaction and to over-write the changed value once again to ensure the consistency.

Roll back: The Rollback transaction is a transaction which rolls back the transaction to the beginning of the transaction (Rollback Transaction name). It is possible to use before Commit transaction.

**Check Pointing, Shadow Paging**

Checkpoint:

* Problems in recovery procedure as discussed earlier :

1. searching the entire log is time-consuming
2. we might unnecessarily redo transactions which have already output their updates to the database.

• Streamline recovery procedure by periodically performing check pointing

* + - 1. Output all log records currently residing in main memory onto stable storage.
      2. Output all modified buffer blocks to the disk.
      3. Write a log record onto stable storage

Shadow Paging:

* Alternative to log-based recovery.
* Idea: maintain two page tables during the lifetime of a transaction – the current page table, and the shadow page table.
* Store the shadow page table in nonvolatile storage, such that state of the database prior to transaction execution may be recovered. Shadow page table is never modified during execution.
* To start with, both the page tables are identical. Only current page table is used for data item accesses during execution of the transaction.

1. **Recovery schemes(WAL)**

**Write-Ahead Logging**

When in-place update (immediate or deferred) is used then log is necessary for recovery and it must be available to recovery manager. This is achieved by Write-Ahead Logging (WAL) protocol. WAL states that

* + For Undo: Before a data item’s AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).
  + For Redo: Before a transaction executes its commit operation, all its AFIMs must be written to the log and the log must be saved on a stable store.[9]

1. **Failure with loss of non volatile storage (general concepts** )

Technique similar to checkpointing used to deal with loss of non-volatile storage

* 1. Periodically **dump** the entire content of the database to stable storage
  2. No transaction may be active during the dump procedure; a procedure similar to checkpointing must take place
     1. Output all log records currently residing in main memory onto stable storage.
     2. Output all buffer blocks onto the disk.
     3. Copy the contents of the database to stable storage.
     4. Output a record <**dump**> to log on stable storage.

1. **Recovery in Multidatabase systems**

A multi database transaction require access to multiple databases.

– The DBs may even be stored on different types of DBMS.

• Some DBMS may be relational, whereas others are object oriented, etc.

– Each DBMS involved in the multi database transaction may have its own recovery technique and transaction manager separate from those of the other DBMSs.

• Use a two-level recovery mechanism to maintain the atomicity of a multi database transaction.

– A global recovery manager, or coordinator.

– The local recovery managers.

The coordinator usually follows a two-phase commit protocol.

– Phase 1

– When all participating databases signal the coordinator that the part of the multi database transaction has concluded, the coordinator sends a message «prepare to 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 or «cannot commit» -or not OK- if it fails for some other reasons.

– If the coordinator does not receive a reply from a database

– Phase 2

• If all the participants DB reply «OK» and also the coordinator, the transaction is successful and the coordinator sends a «commit» signal for the transaction to the participant databases.

– Each participant database completes transaction commit by writing a [commit] entry for the transaction in the log and permanently updating the database if needed.

• If one or more participating DBs or the coordinator sends «not OK» message, the transaction fails and the coordinator sends a message to «rollback» -or UNDO- the local effect of the transaction to each participating database.