**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**DATABASE MANAGEMENT SYSTEM**

**LAB ASSIGNMENT #9**

**Submitted by:**

Rojesh Tamrakar

013BSCCSIT032

**Submitted to:**

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav** |  |

Lecturer

Department of Computer Science

Date of submission: 2nd October, 2015

**Database Recovery**

In computing, **data recovery** is a process of salvaging inaccessible **data** from corrupted or damaged secondary storage, removable media or files, when the **data** they store cannot be accessed in a normal way.

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, system failure.

1. **Types of failure**

Failures may be

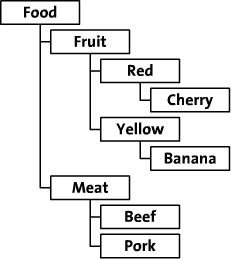
|  |  |
| --- | --- |
| Transaction | Caused by errors within the transaction processes. |
| System | Caused by failure of network or operating system or physical threats to the system as a whole. |
| Media | Failure of hard disk, out of memory errors, out of disk space errors. |

Failure may be caused by a number of things.

|  |  |
| --- | --- |
| A System Crash | A hardware, software or network error causes the transaction to fail. |
| Transaction or System error | Some operation in the transaction may cause the failure or the user may interrupt the transaction. |
| Local Errors or Exceptions | Conditions occur during the transaction that results in transaction cancellation. |
| Concurrency Control Enforcement | Several transactions may be in deadlock so the transaction may be aborted to be restarted later. |
| Disk Failure | Read Write error on the physical disk. |
| Physical Problems | This can be any range of physical problems, such as power failure, mounting wrong disk or tape by operator, wiring problems etc |
| Catastrophe Situations | Large scale threats to the system and the data for example fire, cyclone, security breaches etc. |

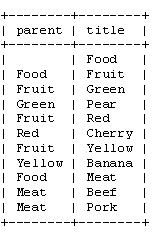
Transaction errors, system errors, system crashes, concurrency problems and local errors or exceptions are the more common causes of system failure.  The system must be able to recover from such failures without loss of data.

1. **The storage Hierarchy**

Storing trees is a common problem, with multiple solutions. There are two major approaches: the adjacency list model, and the modified preorder tree traversal algorithm.

The two methods of saving hierarchical data using the tree from a fictional online food store as an example. This food store organizes its food by category, by colour and by type. The tree looks like this:

It contains a number of code examples that show how to save and retrieve data.



**The Adjacency List Model**

The first, and most elegant, approach we’ll try is called the ‘adjacency list model’ or the ‘recursion method’. It’s an elegant approach because you’ll need just one, simple function to iterate through your tree. In our food store, the table for an adjacency list looks like this:

1. **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. 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''.

* **Forced Output of Blocks**

Sometimes it is necessary to write a block back to disk even though its buffer space is not needed. (Called the forced output of a block.) This is due to the fact that MM contents (and thus the buffer) are lost in a crash, while disk data usually survives.

* **Replacement Strategy –**

When there is no room left in the buffer, some block must be removed to make way for the new one. Typical operating system memory management schemes use a ``least recently used'' (LRU) method. (Simply remove the block least recently referenced.) This can be improved upon for database applications.

1. **Transaction Log**

**Transactions**

* A database is updated by processing *transactions* that result in changes to one or more records.
* A user’s program may carry out many operations on the data retrieved from the database, but the DBMS is only concerned with data read/written from/to the database.
* The DBMS’s abstract view of a user program is a sequence of transactions (reads and writes).

**Log File**

* Log file may be duplexed or triplexed.
* Log file sometimes split into two separate randomaccess files.
* Potential bottleneck; critical in determining overall performance.

1. **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.

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:**

* **Scalability**: distribute query workload from backend to multiple cheap front-end systems.
* **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.
* **Availability**: by continued service for applications that depend only on cached tables even if the backend server is unavailable.
* **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

1. **Transaction Roll back (Undo) & Roll forward**

The transaction log Includes information helpful to the recovery process such as:

A transaction identifier, the date and time, the user running the transaction, before images and after images

Before Image: A copy of the table record (or data item) before it was changed by the transaction.

After Image: A copy of the table record (or data item) after it was changed by the transaction.

* **Rollback:** Undo any partially completed transactions (ones in progress when the crash occurred) by applying the before images to the database.
* **Rollforward:** Redo the transactions by applying the after images to the database. This is done for transactions that were committed before the crash.

1. **Check Pointing, Shadow Paging**

**Check Pointing**

Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery. The following steps defines a checkpoint operation:

* 1. Suspend execution of transactions temporarily.
  2. Force write modified buffer data to disk.
  3. Write a [checkpoint] record to the log, save the log to disk.
  4. Resume normal transaction execution.

During recovery redo or undo is required to transactions appearing after [checkpoint] record.

**Shadow Paging**

Shadow paging is an alternative to log-based recovery; this scheme is useful if transactions execute serially. 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. . Whenever any page is about to be written for the first time. A copy of this page is made onto an unused page. . The current page table is then made to point to the copy.

1. **Recovery Schemes (WAL : Write Ahead Logging Protocol)**

**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

* 1. 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).
  2. 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.

1. **Failure with Loss of Non-volatile storage (General Concepts)**

With the storage device, all the valuable data that is stored inside is lost. We have two different strategies to recover data from such a catastrophic failure −

**Remote Backup**

Remote backup provides a sense of security in case the primary location where the database is located gets destroyed. Remote backup can be offline or real-time or online. In case it is offline, it is maintained manually.

**Remote Data Backup**

Online backup systems are more real-time and lifesavers for database administrators and investors. An online backup system is a mechanism where every bit of the real-time data is backed up simultaneously at two distant places. One of them is directly connected to the system and the other one is kept at a remote place as backup. As soon as the primary database storage fails, the backup system senses the failure and switches the user system to the remote storage. Sometimes this is so instant that the users can’t even realize a failure.

1. **Recovery in Multi-database System**

* A multidatabase system is a special distributed database system where one node may be running relational database system under UNIX, another may be running object-oriented system under Windows and so on.
* A transaction may run in a distributed fashion at multiple nodes.
* In this execution scenario the transaction commits only when all these multiple nodes agree to commit individually the part of the transaction they were executing.
* This commit scheme is referred to as “**two-phase commit**” (**2PC**).
  + If any one of these nodes fails or cannot commit the part of the transaction, then the transaction is aborted.
* Each node recovers the transaction under its own recovery protocol.