**ST.XAVIER,S COLLEGE**

**Maitighar, Kathmandu**



DataBase Management System Assignment #6

**.**

Yub Raj Basnet

013BScCSIT048 (4th Semester)

**Submitted to**

|  |  |
| --- | --- |
| Er. Sanjya Kumar Yadav  (Lecturer, St.Xavier’s College ) |  |

1. **Database Recovery**
   1. **Purpose of Data recovery.**

As a backup administrator, your principal duty is to devise, implement, and manage a backup and recovery strategy. In general, the purpose of a backup and recovery strategy is to protect the database against data loss and reconstruct the database after data loss. Typically, backup administration tasks include the following:

* 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
  1. **Types of failure**
     1. Transaction System
* Caused by errors within the transaction processes.
* Caused by failure of network or operating system or physical threats to the system as a whole.
  + 1. Media
* Failure of hard disk, out of memory errors, out of disk space errors.
  1. **The Storage Hierarchy**

You can doubtless think of many examples of storage hierarchies in ordinary life. For example, people live in neighborhoods, which are in towns, which are in regions, countries, continents, and so on up the line. The relations are generally many-to-one, although there are occasional one-to-one correspondences (e.g., Australia is both a country and a continent), and occasional exceptions (e.g., a person can straddle a city boundary).

Figure shows the storage hierarchy—the physical constructs of a database. The hierarchy of physical objects suggests that—with occasional one-to-one correspondences or exceptions—data rows live in pages, which are in extents, which are in files, table spaces, and databases. There is a reason for each level of grouping. To see what the reason is, we'll go through each of those objects in order, up the line.

**BUFFER MANAGEMENT**

A buffer is an 8-KB page in memory, the same size as a data or index page. Thus, the buffer cache is divided into 8-KB pages. The buffer manager manages the functions for reading data or index pages from the database disk files into the buffer cache and writing modified pages back to disk. A page remains in the buffer cache until the buffer manager needs the buffer area to read in more data. Data is written back to disk only if it is modified. Data in the buffer cache can be modified multiple times before being written back to disk.

A hash table is used to figure out what page frame a given disk page (i.e., with a given pageId) occupies. A buffer descriptor object is associated with every page frame in the buffer pool. It contains a dirty bit, the page number, and the pin count for the page occupying that frame.

When a page is requested, the buffer manager brings it in and pins it. The buffer manager does not keep track of all the pages that have been pinned by a transaction. It is up to the various components (that call the buffer manager) to make sure that all pinned pages are subsequently unpinned.

**TRANSACTION LOG**

A transaction log (also transaction journal, database log, binary log or audit trail) is a history of actions executed by a database management system to guarantee ACID properties over crashes or hardware failures. Physically, a log is a file listing changes to the database, stored in a stable storage format.

If, after a start, the database is found in an inconsistent state or not been shut down properly, the database management system reviews the database logs for uncommitted transactions and rolls back the changes made by these transactions. Additionally, all transactions that are already committed but whose changes were not yet materialized in the database are re-applied. Both are done to ensure atomicity and durability of transactions.

The database can be modified using two approaches −

* **Deferred database modification** − All logs are written on to the stable storage and the database is updated when a transaction commits.
* **Immediate database modification** − Each log follows an actual database modification. That is, the database is modified immediately after every operation.

**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**

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

**TRANSACTION ROLL BACK AND 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.

**CHECK POINTING, SHADOW PAGING**

Keeping and maintaining logs in real time and in real environment may fill out all the memory space available in the system. As time passes, the log file may grow too big to be handled at all. Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.

It is inconvienient to maintain logs of all transactions fro the purposes of recovery. An alternative is to use a system of shadow paging. This is where the database is divided into pages that may be stored in any order on the disk. In order to identify the location of any given page, we use something called a page table.

During the life of a tranasacation two page tables are maintained, one called a shadow page table and current page table. When a tranasaction begins both of these page tables point to the same locations (are identical). During the lifetime of a transaction the shadow page table doesn't change at all. However during the lifetime of a transaction changes may be made update values etc. So whenever we update a page in the database we always write the updated page to a new location. This means that when we then update our current page table to reflect the changes that have been made.