**ST. XAVIER’S COLLEGE**

**MAITIGHAR, KATHMANDU**

****

**database management system**

**LAb ASSIGNMENT #9**

**Submitted By:**

Abhishek Tamrakar

013BSCIT003

**Submitted To:**

Department of Computer Science

St. Xavier’s College, Maitighar

Lecturer

|  |  |
| --- | --- |
| Er. Sanjay Kumar Yadav | Er. Anil Shah |
|  |  |

**Database Recovery**

1. **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 [1].

To recover a SQL Server database from a failure, a database administrator has to restore a set of SQL Server backups in a logically correct and meaningful restore sequence. SQL Server restore and recovery supports restoring data from backups of a whole database, a data file, or a data page, as follows:

* The database (a complete database restore)

The whole database is restored and recovered, and the database is offline for the duration of the restore and recovery operations.

* The data file (a file restore)

A data file or a set of files is restored and recovered. During a file restore, the filegroups that contain the files are automatically offline for the duration of the restore. Any attempt to access an offline filegroup causes an error.

* The data page (a page restore)

Under the full recovery model or bulk-logged recovery model, you can restore individual databases. Page restores can be performed on any database, regardless of the number of filegroups [2].

1. **Purpose of data recovery**

* To bring the database into the last consistent state, which existed prior to the failure.
* To preserve transaction properties (Atomicity, Consistency, Isolation and Durability).

Example:

If the system crashes before a fund transfer transaction completes its execution, then either one or both accounts may have incorrect value. Thus, the database must be restored to the state before the transaction modified any of the accounts.

1. **Types of failure**

The types of failure in DBMS are:

**3.1 Transaction failures**

When a transaction is failed to execute or it reaches a point after which it cannot be completed successfully it has to abort. This is called transaction failure, where only few transaction or processes are hurt.

* overflow, interrupt, data not available, explicit rollback,
* concurrency enforcement, programming errors
* no memory loss.

Reason for transaction failure could be:

* Logical errors: where a transaction cannot complete because of it has some code error or any internal error condition
* System errors: where the database system itself terminates an active transaction because DBMS is not able to execute it or it has to stop because of some system condition. For example, in case of deadlock or resource unavailability systems aborts an active transaction.

**3.2 System crashes**

There are problems, which are external to the system, which may cause the system to stop abruptly and cause the system to crash. For example interruption in power supply, failure of underlying hardware or software failure. Examples may include operating system errors.

* due to hardware or software errors
* main memory content is lost

**3.3 Media failures**

* problems with disk head, unreadable media surface
* (parts of ) information on secondary storage may be lost
* lose a disk or a disk controller fails
* a head crash
* file corruption
* the overwriting or deletion of a datafile
* sny of the multiplexed control files are deleted or lost
* any datafile belonging to the system or the undo tablespace is deleted or lost.
* sn entire redo log group is lost.

**3.4 Natural disasters**

* fire, flood, earthquakes, theft, etc.
  + - * physical loss of all information on all media

**3.5 Disk failure**

In early days of technology evolution, it was a common problem where hard disk drives or storage drives used to fail frequently.

Disk failures include formation of bad sectors, unreachability to the disk, disk head crash or any other failure, which destroys all or part of disk storage.

**3.6 Statement Failure**

* When a program attempts to enter invalid data into an Oracle table.
* No more room to put the data in.
* Not having the proper privileges to perform a task.

**3.7 User Process Failure**

* A user process may be terminated abruptly (A single database session fails).

**3.9 Network Failure**

* Oracle Net listener.
* The network interface card.
* The network connection has failed.

**3.10 Instance Failure**

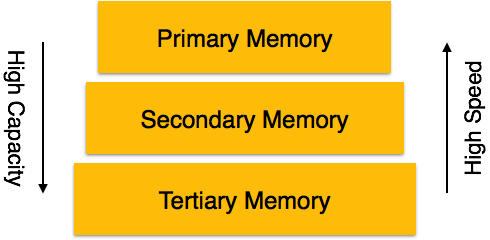
* Hardware failure.
* A power failure.
* An emergency shutdown procedure.
* Oracle background process such as pmon shuts down because of an error condition.

**3.11 User Error**

* Accidentally dropping a table.
* Wrongly modify or delete data from a table [3][4][5].

1. **Storage hierarchy**

Databases are stored in file formats, which contains records. At physical level, actual data is stored in electromagnetic format on some device capable of storing it for a longer amount of time. These storage devices can be broadly categorized in three types:



**4.1 Primary Storage:**

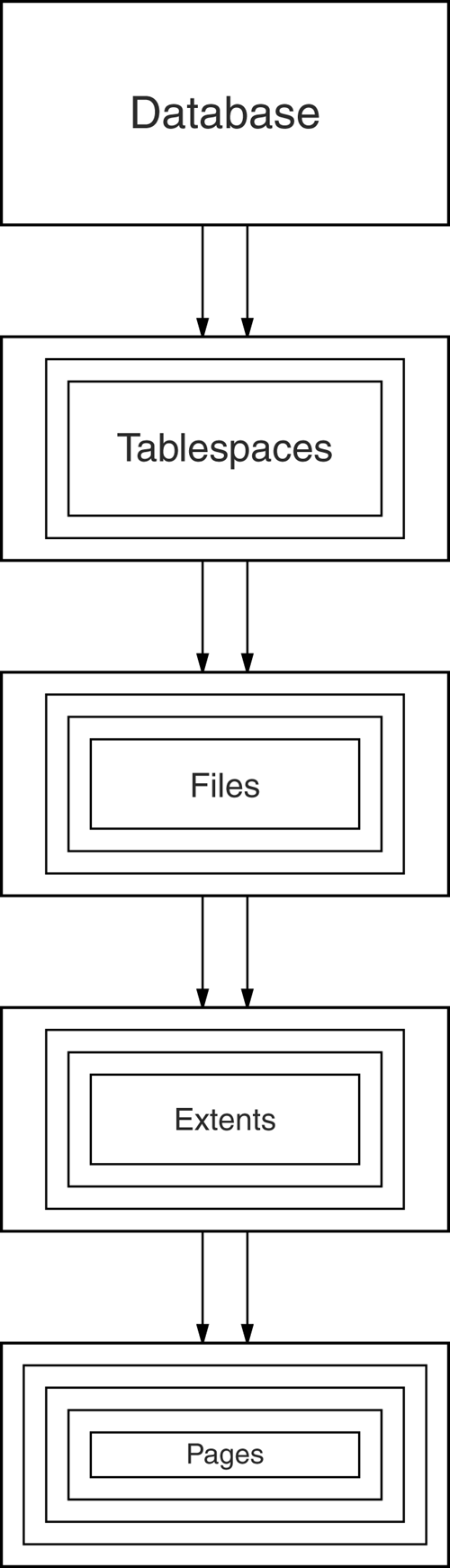
The memory storage, which is directly accessible by the CPU, comes under this category. CPU's internal memory (registers), fast memory (cache) and main memory (RAM) are directly accessible to CPU as they all are placed on the motherboard or CPU chipset. This storage is typically very small, ultra fast and volatile. This storage needs continuous power supply in order to maintain its state, i.e. in case of power failure all data are lost.

**4.2 Secondary Storage:**

The need to store data for longer amount of time and to retain it even after the power supply is interrupted gave birth to secondary data storage. All memory devices, which are not part of CPU chipset or motherboard comes under this category. Broadly, magnetic disks, all optical disks (DVD, CD etc.), flash drives and magnetic tapes are not directly accessible by the CPU. Hard disk drives, which contain the operating system and generally not removed from the computers are, considered secondary storage and all other are called tertiary storage.

**4.3 Tertiary Storage:**

Third level in memory hierarchy is called tertiary storage. This is used to store huge amount of data. Because this storage is external to the computer system, it is the slowest in speed. These storage devices are mostly used to backup the entire system. Optical disk and magnetic tapes are widely used storage devices as tertiary storage [6].

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, tablespaces, 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.

**Pages**

Depending on the DBMS, a page is also called a data block, a block, a blocking unit, a control interval, and a row group.A page is a fixed-size hopper that stores rows of data. Pages have four common characteristics, which are not true by definition but are always true in practice. All pages in a file have the same size. Indeed for some DBMSs, it is true that all pages in all files have the same size, but the usual case is that you have a choice when making a new object.

**Extents**

An extent is a group of contiguous pages. Extents exist to solve the allocation problem. The allocation problem is that, when a file gets full, the DBMS must increase its size. If the file size increases by only one page at a time, waste occurs because:

* The operating system must update the file allocation tables. The amount of updating is about the same whether the addition is one page or eight pages.
* If file A grows, then file B grows, then file A grows again, and so on; the operating system will have to maintain a succession of short (one page) chains: ABABABAB. This is a type of fragmentation. As we've hinted before, fragmentation is bad; we'll discuss it later in this chapter.

**Files**

A file is a group of contiguous extents. Surprisingly, a file is not a physical representation of a table.

**Tablespaces**

A tablespace (also called a dbspace by some DBMSs, e.g., Informix) is a file, or a group of files, that contains data [7].

1. **Buffer management**

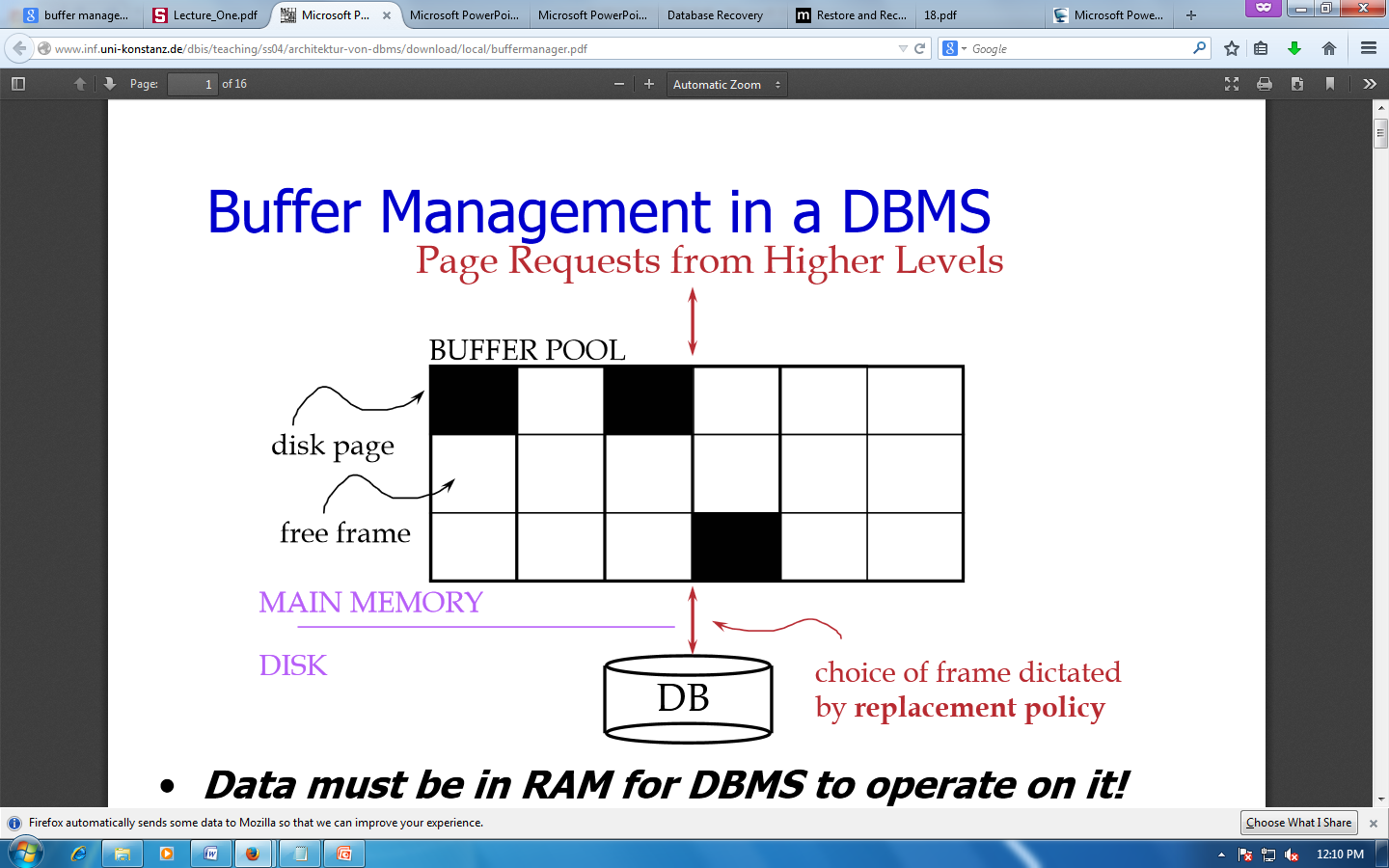


Figure 1Buffer Management in DBMS

Data must be in RAM for DBMS to operate on it. Buffer Mgr hides the fact that not all data is in RAM

* When a Page is Requested
  + Buffer pool information table contains:

<frame#, pageid, pin\_count, dirty>

* If requested page is not in pool:
  + Choose a frame for replacement (only un-pinned pages are candidates)
  + If frame is “dirty”, write it to disk
  + Read requested page into chosen frame
  + Pin the page and return its address.
* Once we choose a page to remove
  + A page is dirty, if its contents have been changed after writing
  + Buffer Manager keeps a dirty bit
  + Say we choose to evict, if P is dirty, we write it to disk [8]

1. **Transaction log**

A DBMS uses a transaction log to keep track of all transactions that update the database. The information stored in this log is used by the DBMS for a recoveryrequirement triggered by a ROLLBACK statement, a program’s abnormal termination, or a system failure such as a network discrepancy or a disk crash. Some RDBMSs use thetransaction log to recover a database forward to a currently consistent state. After a server failure, for example, Oracle automatically rolls back uncommitted transactions and rolls forward transactions that were committed but not yet written to the physical database.While the DBMS executes transactions that modify the database, it also automaticallyupdates the transaction log. The transaction log stores:

* A record for the beginning of the transaction.
* For each transaction component (SQL statement):
* The type of operation being performed (update, delete, insert).
* The names of the objects affected by the transaction (the name of the table).
* The “before” and “after” values for the fields being updated.
* Pointers to the previous and next transaction log entries for the same transaction.
* The ending (COMMIT) of the transaction [9].

1. Data Updates

The UPDATE statement is used to change or modify the existing records in a database table. It is typically used in conjugation with the WHERE clause to apply the changes to only those records that matches specific criteria.

The basic syntax of the UPDATE statement can be given with:

UPDATE table\_name SET column1=value, column2=value2,... WHERE column\_name=some\_value

Let's make a SQL query using the UPDATE statement and WHERE clause, after that we will execute this SQL query through passing it to the mysqli\_query() function to update the tables records. Consider the following "persons" table inside the "demo" database:

+-----------+------------+-----------+----------------------+

| person\_id | first\_name | last\_name | email\_address |

+-----------+------------+-----------+----------------------+

| 1 | Peter | Parker | peterparker@mail.com |

| 2 | John | Rambo | johnrambo@mail.com |

| 3 | Clark | Kent | clarkkent@mail.com |

| 4 | John | Carter | johncarter@mail.com |

| 5 | Harry | Potter | harrypotter@mail.com |

+-----------+------------+-----------+----------------------+

1. Data Cachin
2. Check Pointing, Shadow paging

Shadow paging:

This recovery scheme does not require the use of a log in a single-user environment. In a multiuser environment, a log may be needed for the concurrency control method. Shadow paging considers the database to be made up of a number of fixed-size disk pages (or disk blocks)—say, n—for recovery purposes. A directory with n entries is constructed, where the i th entry points to the i th database page on disk. The directory is kept in main memory if it is not too large, and all references—reads or writes—to database pages on disk go through it. When a transaction begins executing, the current directory—whose entries point to the most recent or current database pages on disk—is copied into a shadow directory. The shadow directory is then saved on disk while the current directory is used by the transaction.

During transaction execution, the shadow directory is never modified. When a write\_item operation is performed, a new copy of the modified database page is created, but the old copy of that page is not overwritten. Instead, the new page is written elsewhere—on some previously unused disk block. The current directory entry is modified to point to the new disk block, whereas the shadow directory is not modified and continues to point to the old unmodified disk block. For pages updated by the transaction, two versions are kept. The old version is referenced by the shadow directory, and the new version by the current directory.

To recover from a failure during transaction execution, it is sufficient to free the modified database pages and to discard the current directory. The state of the database before transaction execution is available through the shadow directory, and that state is recovered by reinstating the shadow directory. The database thus is returned to its state prior to the transaction that was executing when the crash occurred, and any modified pages are discarded. Committing a transaction corresponds to discarding the previous shadow directory. Since recovery involves neither undoing nor redoing data items, this technique can be categorized as a NO-UNDO/NO-REDO technique for recovery.

.

.

**Reference**

1. “Database Recovery”

<http://www.csee.umbc.edu/portal/help/oracle8/server.815/a67781/c28recov.htm>

1. “Restore and recovery overview”

<http://msdn.microsoft.com/en-us/library/ms191253.aspx>

1. “DBMS System”

<http://webdocs.cs.ualberta.ca/~zaiane/courses/cmput391/slides/L9-391-04.pdf>

1. “Types of database failures”

<http://www.saraie.net/index.php?option=com_content&view=article&id=45:types-of-database-failures-&catid=5:recovery-manager-rman&Itemid=19>

1. “Database data recovery”

<http://www.tutorialspoint.com/dbms/dbms_data_recovery.htm>

1. “DBMS storage system”

<http://www.tutorialspoint.com/dbms/dbms_storage_system.htm>

1. “The storage hierarchy”

<http://codeidol.com/community/sql/the-storage-hierarchy/3437/>

1. “Buffer management in a database”

<http://www.inf.uni-konstanz.de/dbis/teaching/ss04/architektur-von-dbms/download/local/buffermanager.pdf>

1. “The transaction log”

<http://www.myreadingroom.co.in/images/stories/docs/dbms/the%20transaction%20log.pdf>