ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Theory Assignment**

**SUBMITTED BY:**

Ronak Agrawal

013BSCCSIT033

**SUBMITTED TO**

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

**Database Recovery**

Data recovery is the process of restoring data that has been lost, accidentally deleted, corrupted or made inaccessible for any reason [1].

**Purpose of Data Recovery**

* To bring the database into the last consistent state, which existed prior to the failure [5].
* To preserve transaction properties:
  + Atomicity
  + Consistency
  + Isolation
  + Durability

**Types of Database Failures:**

There are six general categories for database-related failures [3].

1. **Statement Failures**

Statement failures occur when a single database operation fails, such as a single INSERT statement or the creation of a table. In the list that follows are a few of the most common problems and their solutions when a statement fails.

Although granting user privileges or additional quotas within a table space solves many of these problems, also consider whether there are any gaps in the user education process that might lead to some of these problems in the first place.

1. **User Process Failures**

The abnormal termination of a user session is categorized as a user process failure; any uncommitted transaction must be cleaned up. The PMON (process monitor) background process periodically checks all user processes to ensure that the session is still connected. If the PMON finds a disconnected session, it rolls back the uncommitted transaction and releases all locks held by the disconnected process. Causes for user process failures typically fall into one of these categories:

* A user closes their SQL\*Plus window without logging out.
* The workstation reboots suddenly before the application can be closed.
* The application program causes an exception and closes before the application can be terminated normally.
* A user process times out and database disconnects the session.

1. **Network Failures**

Depending on the locations of your workstation and your server, getting from your workstation to the server over the network might involve a number of hops: you might traverse several local switches and WAN routers to get to the database. From a network perspective, this configuration provides a number of points where failure can occur. These types of failures are called network failures.

In addition to hardware failures between the server and client, a listener process on the Oracle server can fail or the network card on the server itself can fail. To guard against these kinds of failures, you can provide redundant network paths from your clients to the server, as well as additional listener connections on the Oracle server and redundant network cards on the server.

1. **User Error Failures**

Even if all your redundant hardware is at peak performance, and your users have been trained to disconnect from their database sessions properly, users can still inadvertently delete or modify data in tables or drop an index. This is known as a user error failure. Although these operations succeed from a statement point of view, they might not be logically correct: the DROP TABLE command worked fine, but you really didn’t want to drop that table!

If data was inadvertently deleted from a table, and not yet committed, a ROLLBACK statement will undo the damage. If a COMMIT has already been performed, you have a number of options at your disposal, such as using data in the undo table space for a Flashback Query or using data in the archived and online redo logs with the Log Miner utility, available as a command-line or GUI interface.

1. **Instance Failures**

An instance failure occurs when the instance shuts down without synchronizing all the database files to the same system change number (SCN), requiring a recovery operation the next time the instance is started. Many of the reasons for an instance failure are out of your direct control; in these situations, you can minimize the impact of these failures by tuning instance recovery.

A few causes for instance failure:

* A power outage
* A server hardware failure
* Failure of a database background process
* Emergency shutdown procedures (intentional power outage or SHUTDOWN ABORT)

1. **Media Failures**

Another type of failure that is somewhat out of your control is media failure. A media failure is any type of failure that results in the loss of one or more database files: data files, control files, or redo log files. Although the loss of other database-related files such as an init.ora file or a server parameter file (SPFILE) is of great concern, Oracle Corporation does not consider it a media failure. The database file can be lost or corrupted for a number of reasons:

* Failure of a disk drive
* Failure of a disk controller
* Inadvertent deletion or corruption of a database file

Following the best practices by adequately mirroring control files, redo log files, and ensuring that full backups and their subsequent archived redo log files are available will keep you prepared for any type of media failure.

|  |  |
| --- | --- |
| Attempts to access tables without the appropriate privileges | Provide the appropriate privileges or create views on the tables and grant privileges on the view. |
| Running out of space | Add space to the tablespace, increase the user’s quota on the tablespace, or enable resumable space allocation. |
| Logic errors in applications | Work with developers to correct program errors or provide additional logic in the application to recover gracefully from unavoidable errors. |

**Transactions roll back and roll forward**

**Rollback:** The Rollback transaction is a transaction which rolls back the transaction to the beginning of the transaction. The transaction can be rolled back completely by specifying the transaction name in the Rollback statement or to cancel any changes to a database during current transaction. It is permissible to use before Commit transaction [4].

**Roll forward:** Recovering a database by applying different transactions that recorded in the database log files. It is nothing but re-doing the changes made by a transaction i.e. after the committed transaction and to over write the changed value again to ensure consistency [4].

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

WAL protocol Write-ahead logging (WAL) is a family of techniques for providing atomicity and durability (two of the ACID properties) in database systems [9].

In a system using WAL, all modifications are written to a log before they are applied. Usually both redo and undo information is stored in the log.

Before a block of data in main memory can be output to the database (in nonvolatile storage), all log records pertaining to data in that block must have been output to stable storage. This rule is called the WAL rule. Strictly speaking, the WAL rule requires only that the undo information in the log have been output to stable storage, and permits the redo information to be written later. The difference is relevant in systems where undo information and redo information are stored in separate log records.

Write-ahead logging is employed to flush log records to the persistent log file before data pages are written or at commit time.

,  
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 were used, the program could 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.

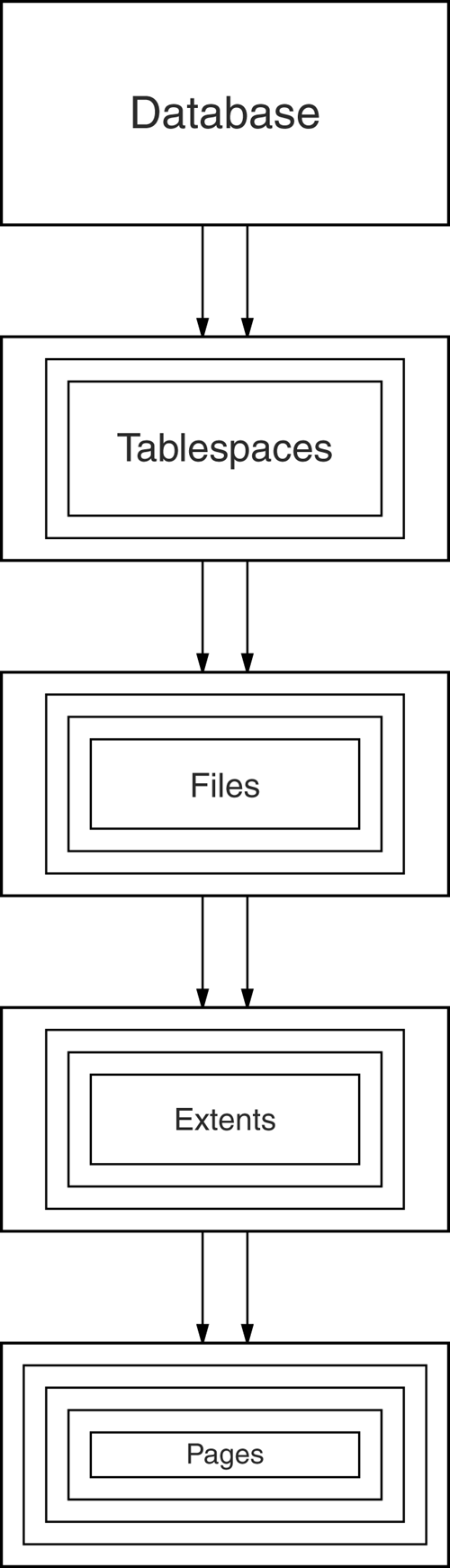
**The Storage Hierarchy:**

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.

Pages

Depending on the DBMS, a page is also called a data block, a [block](http://codeidol.com/community/sql/the-storage-hierarchy/3437/app02.html#gloss19), a [blocking unit](http://codeidol.com/community/sql/the-storage-hierarchy/3437/app02.html#gloss21), a [control interval](http://codeidol.com/community/sql/the-storage-hierarchy/3437/app02.html#gloss48), and a [row group](http://codeidol.com/community/sql/the-storage-hierarchy/3437/app02.html#gloss197).

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. They are:

* 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.
* The choice of page sizes is restricted to certain multiples of 1024 (1KB), in a range between 1024 and 65536—that is, between 1KB and 64KB.
* The optimum page size is related to the disk system's attributes. Smaller page sizes like 2KB were once the norm, but disks' capacity tends to increase over time, so now 8KB is reasonable, while 16KB is what we'll upgrade to soon.
* Pages contain an integral number of rows. Even for the rare DBMSs that allow large rows to overflow into later pages, the very strong recommendation is that you should avoid overflow.

**Buffer Management**

**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.
* If requests can be predicted (e.g., sequential scans), pages can be pre-fetched several pages at a time.

**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 recovery requirement 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 the transaction 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 automatically updates 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.

**Data Updates:**

The SQL UPDATE statement is used to update existing records in the tables.

**Syntax:**

The syntax for the SQL UPDATE statement when updating one table is:

UPDATE table

SET column1 = expression1,

column2 = expression2,

...

WHERE conditions;

The syntax for the SQL UPDATE statement when updating one table with data from another table is:

UPDATE table1

SET column1 = (SELECT expression1

FROM table2

WHERE conditions)

WHERE conditions;

**Checkpoints**

When a system failure occurs, we must consult the log to determine those transactions that need to be redone and those that need to be undone. Rather than reprocessing the entire log, which is time-consuming and much of it unnecessary, we can use *checkpoints*:

1. Output onto stable storage all the log records currently residing in main memory.
2. Output to the disk all modified buffer blocks.
3. Output onto stable storage a log record, **<checkpoint>**.

Now recovery will be to only process log records since the last checkpoint record [7].

**Shadow Paging:**

Shadow paging is an alternative to log-based recovery techniques, which has both advantages and disadvantages. It may require fewer disk accesses, but it is hard to extend paging to allow multiple concurrent transactions. The paging is very similar to paging schemes used by the operating system for memory management.

The idea is to maintain two page tables during the life of a transaction: the current page table and the shadow page table. When the transaction starts, both tables are identical. The shadow page is never changed during the life of the transaction. The current page is updated with each**write**operation. Each table entry points to a page on the disk. When the transaction is committed, the shadow page entry becomes a copy of the current page table entry and the disk block with the old data is released. If the shadow is stored in nonvolatile memory and a system crash occurs, then the shadow page table is copied to the current page table. This guarantees that the shadow page table will point to the database pages corresponding to the state of the database prior to any transaction that was active at the time of the crash, making aborts automatic [7].

There are drawbacks to the shadow-page technique:

* **Commit overhead.** The commit of a single transaction using shadow paging requires multiple blocks to be output -- the current page table, the actual data and the disk address of the current page table. Log-based schemes need to output only the log records.
* **Data fragmentation.** Shadow paging causes database pages to change locations (therefore, no longer contiguous.
* **Garbage collection.** Each time that a transaction commits, the database pages containing the old version of data changed by the transactions must become inaccessible. Such pages are considered to begarbage since they are not part of the free space and do not contain any usable information. Periodically it is necessary to find all of the garbage pages and add them to the list of free pages. This process is called garbage collection and imposes additional overhead and complexity on the system.

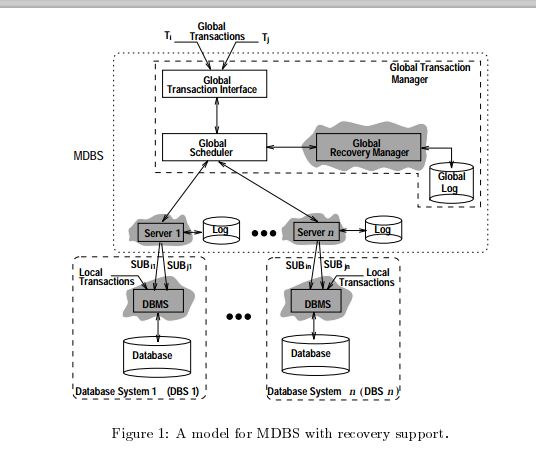
**Failure with loss of Non- volatile storage:**

The basic scheme is to *dump* the entire content of the database to stable memory periodically. No transaction can be active during the dump procedure.

To recover from the loss of nonvolatile memory, we restore the database from the archive and all the transactions that have been committed since the most recent dump are redone.

This is also known as an *archival dump*. Dumps of the database and checkpointing are very similar [7].

**Recovery in Multi database system:**



The architecture of an MDBS basically consists of the Global Transaction Manager (GTM), a set of Interface Servers (servers, for short), and multiple LDBSs. To each LDBS, there is an associated server. An LDBS consists of a DBMS and at least one database. The GTM comprises three modules: Global Transaction Interface (GTI), Global Scheduler (GS), and Global Recovery Manager (GRM). An MDBS architecture is depicted in Figure 1.

Users interact with the local DBMS (LDBMS) by means of transactions. Two classes of transactions are supported in a multi database environment [8]:

* Local transactions which are transactions executed by an LDBMS outside the control of the MDBS and
* Global transactions which comprise transactions submitted by the MDBS to LDBMSs. Global transactions may be executed in more than one local system. Thus, we define a global transaction Gi as a set of subsequences fsubi;1, subi;2, subi;3, :::, subi;mg where each subi;k is executed at the local system LDBSk as an ordinary local transaction.

**References:**

1. http://searchdisasterrecovery.techtarget.com/definition/data-recovery
2. http://codeidol.com/community/sql/the-storage-hierarchy/3437/
3. http://ora10g.com/8\_1\_failures.html
4. http://riaddba.blogspot.com/2012/11/definition-of-rollback-and-roll-forward.html
5. people.cis.ksu.edu/~hankley/d560/Topics/Finger\_**Recovery**.ppt
6. <http://www.myreadingroom.co.in/images/stories/docs/dbms/the%20transaction%20log.pdf>
7. http://www.csee.umbc.edu/courses/461/current/burt/lectures/lec25/recovery.html
8. http://www.inf.ufsc.br/sbbd99/anais/SBBD-Completo/09.pdf
9. http://www.hostitwise.com/hosting-blog/dbms-wal-protocol/