**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**DATABASE MANAGEMENT SYSTEM**

THEORY ASSIGNMENT#9

**Submitted by:**

**Shreesha Pokharel**

013BSCCSIT038

**Submitted to:**

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

Lecturer

Department of Computer Science

Date of submission: 2nd October, 2015

**DATABASE RECOVERY:**

Reconstructing the contents of all or part of a database from a backup typically involves two phases: retrieving a copy of the data file from a backup, and reapplying changes to the file since the backup from the archived and online redo logs, to bring the database to a desired SCN since the backup (usually, the present).

To **restore** a data file or control file from backup is to retrieve the file onto disk from a backup location on tape, disk or other media, and make it available to the database server.

To **recover** a data file (also called **performing recovery** on a data file), is to take a restored copy of the data file and apply to it changes recorded in the database's redo logs. To recover a whole database is to perform recovery on each of its data files.

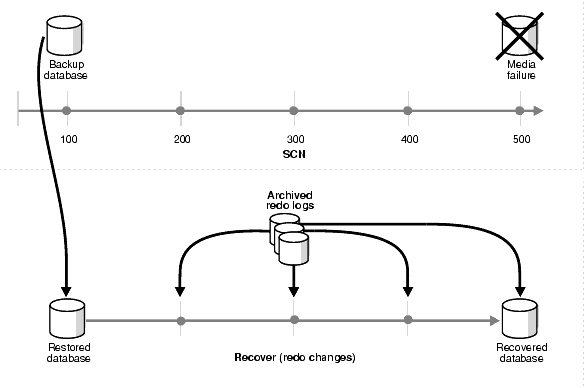


Fig: restoring and recovering a database

1. **Purpose of data recovery**

Recovery—that is, the return to a fully operational environment after a hardware or software failure—is an important process. Moreover, the effects of a system failure on the organization must be curtailed to minimize any substantial financial loss. Actions must be taken to prevent DBMS failures or resolve them quickly if they occur. It is not always cost-effective to implement all possible DBMS controls and use all known review techniques. The choice of whether or not to audit can have a direct impact on the financial consequences caused by these failures. A review of DBMS recovery ensures adherence to appropriate practices and procedures and minimizes business losses. A review further ensures that an organization can recover and return to full operational status following a disaster.

For example, the January 1994 earthquake in the Los Angeles area caused sustained interruption of business in many organizations; those organizations that had established recovery procedures were able to more readily restore operations and minimize losses. Developing, implementing, maintaining, and auditing the DBMS recover controls and processes involve a considerable amount of money and company resources. Costs and Previous benefits must be considered to ensure that company resources are expended efficiently. Systems managers who are either developing or maintaining a DBMS must understand data base structures and participate in the recovery process. This article explains the process and techniques for reviewing DBMS recovery.

1. **Types of failure**

**User Error:**

A database administrator can do little to prevent user errors (for example, accidentally dropping a table). Usually, user error can be reduced by increased training on database and application principles. Furthermore, by planning an effective recovery scheme ahead of time, the administrator can ease the work necessary to recover from many types of user errors.

**Statement Failure:**

Statement failure occurs when there is a logical failure in the handling of a statement in an Oracle program. For example, assume all extents of a table (in other words, the number of extents specified in the MAXEXTENTS parameter of the CREATE TABLE statement) are allocated, and are completely filled with data; the table is absolutely full. A valid INSERT statement cannot insert a row because there is no space available. Therefore, if issued, the statement fails.

**Process Failure:**

A process failure is a failure in a user, server, or background process of a database instance (for example, an abnormal disconnect or process termination). When a process failure occurs, the failed subordinate process cannot continue work, although the other processes of the database instance can continue.

**Network Failure:**

When your system uses networks (for example, local area networks, phone lines, and so on) to connect client workstations to database servers, or to connect several database servers to form a distributed database system, network failures (such as aborted phone connections or network communication software failures) can interrupt the normal operation of a database system.

**Database Instance Failure:**atabase instance failure occurs when a problem arises that prevents an Oracle database instance (SGA and background processes) from continuing to work. An instance failure can result from a hardware problem, such as a power outage, or a software problem, such as an operating system crash. Instance failure also results when you issue a SHUTDOWN ABORT or STARTUP FORCE command.

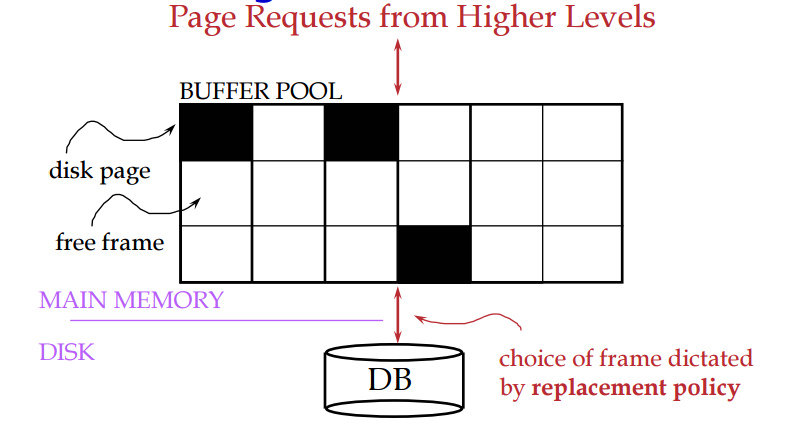
1. **The storage hierarchy**

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.

1. **Buffer management**



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

1. **Transaction management**

Transaction is an event which occurs on the database. Generally a transaction reads a value from the database or writes a value to the database. If you have any concept of Operating Systems, then we can say that a transaction is analogous to processes.

Although a transaction can both read and write on the database, there are some fundamental differences between these two classes of operations. A read operation does not change the image of the database in any way. But a write operation, whether performed with the intention of inserting, updating or deleting data from the database, changes the image of the database. That is, we may say that these transactions bring the database from an image which existed before the transaction occurred (called the**Before Image**or**BFIM**) to an image which exists after the transaction occurred (called the **After Image**or**AFIM**).

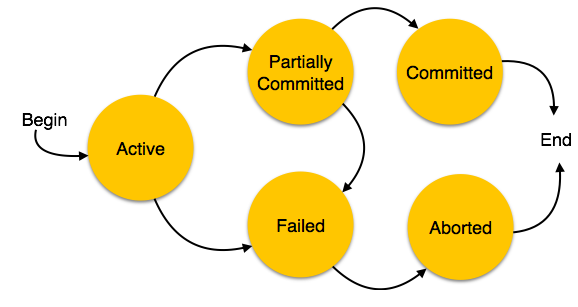


Fig: states in transaction management

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.

In three tier architecture, the application tier and data tier can be in different hosts. Throughput of the application is affected by the network speed. This network overhead will be avoided by having the database at the application tier. As commercial databases are heavy weight, it is not practically feasible to have the application and the database at the same host. There are lots of light-weight databases available on the market, which can be used to cache data from the commercial databases.

1. **Transaction roll back (undo) and roll forward**

**Rollback:** The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.

The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

The syntax for ROLLBACK command is as follows:

ROLLBACK;

**Roll forward:** 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.

1. **Check pointing, shadow paging**

**Check pointing:** 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.

**Shadow paging:** It is inconvenient to maintain logs of all transactions for 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 transaction two page tables are maintained, one called a shadow page table and current page table. When a transaction 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.

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

**Write-ahead logging** (**WAL**) is a family of techniques for providing atomicity and durability in database systems.

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.

The purpose of this can be illustrated by an example. 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 is used, the program can 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.

WAL allows updates of a database to be done in-place. Another way to implement atomic updates is with shadow paging, which is not in-place. The main advantage of doing updates in-place is that it reduces the need to modify indexes and block lists.

1. **Failure with loss of non-volatile storage (General Concepts)**

A volatile storage like RAM stores all the active logs, disk buffers, and related data. In addition, it stores all the transactions that are being currently executed. What happens if such a volatile storage crashes abruptly? It would obviously take away all the logs and active copies of the database. It makes recovery almost impossible, as everything that is required to recover the data is lost.

Following techniques may be adopted in case of loss of volatile storage −

* We can have **checkpoints** at multiple stages so as to save the contents of the database periodically.
* A state of active database in the volatile memory can be periodically **dumped** onto a stable storage, which may also contain logs and active transactions and buffer blocks.
* <dump> can be marked on a log file, whenever the database contents are dumped from a non-volatile memory to a stable one.

1. **Recovery in multi-database**

The Recovery MT strategy consists of a collection of recovery protocols which are distributed among the components of an MDBS. Hence, some of them are performed by the GRM, some by the servers and some are provided by the LDBMSs. We assume that every participating LDBMS provides its own recovery mechanism. Local recovery mechanisms should be able to restore the most recent transaction-consistent state of local databases after local failures. For each type of failure described in Section 3, we propose a specific recovery scheme.