ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System Assignment #9**

**Submitted by:**

Bikash Paneru  
013BSCCSIT012

**Submitted to:**

|  |  |
| --- | --- |
| Er. Sanjay Kumar Yadav Lecturer, St. Xavier’s College |  |

**Date of Submission: October 2, 2015**

**PURPOSE OF DATA RECOVERY**

In enterprise information technology (IT), data recovery typically refers to the restoration of data to a desktop, laptop, server, or external storage system from a backup.

The data recovery process may vary, depending on the circumstances of the data loss, the data recovery software used to create the backup, and the backup target media. For example, many desktop and laptop backup software platforms allow end users to restore lost files themselves, while restoration of a corrupted database from a tape backup is a more complicated process that requires IT intervention. Data recovery can also be provided as service. Such services are typically used to retrieve important files that were not backed up and accidentally deleted from a computer's file system but still remain on disk in fragments.

An organization's disaster recovery plan should make known who in the organization is responsible for recovering data, provide a strategy for how data will be recovered and document acceptable recovery point and recovery time objectives.

**TYPES OF FAILURE**

In every database system or affect database I/O operations, the possibility of a system or hardware failure always exists. For some of these problems, crash and instance recovery occur automatically and require no action on the part of the database administrator. The database failures are different types. A database can fail, either entirely or partially because of various reasons.

**The most common types of failure are:**

**1. Statement Failure:** The statement failure occurs when there is a logical failure in the handling of a statement in an Oracle program. When a program attempts to enter invalid data into an Oracle table. When statement failure occurs, any effects of the statement are automatically undone by Oracle and control is returned to the user.

**For example,** all extents of a table are allocated and are completely filled with data. A valid INSERT statement cannot insert a row because no space is available.

**2. User Process Failure:** A user process failure is any failure in a user program accessing an Oracle database. A user process may be terminated abruptly.

**For example,** a user could accidentally drop a table. To enable recovery from user errors and accommodate other unique recovery requirements, Oracle provides exact point-in-time recovery.

**3. Network Failure:**When your system uses networks such as local area networks and phone lines 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.

**4. Instance Failure:**The instance failureoccurs when a problem arises that prevents an instance from continuing work. Instance failure can result from a hardware problem such as a power outage, or a software problem such as an operating system failure. When an instance failure occurs, the data in the buffers of the system global area is not written to the data files.

**Some example of instance failure:**  
1. Hardware failure.  
2. A power failure.  
3. An emergency shutdown procedure.

**5. User Error:** This requires a database to be recovered to a point in time before the error occurred. As an administrator, Oracle DBA can do little to prevent user errors such as accidentally dropping a table and wrongly modify or delete data from a table.

**6. Media and Disk Failure:**Media failure is a physical problem that occurs when a computer unsuccessfully attempts to read from or write to a file necessary to operate the database. An error can occur when trying to write or read a file on disk that is required to operate the database.

**THE STORAGE HIERARCHY**

Data hierarchy refers to the systematic organization of data, often in a hierarchical form. Data organization involves fields, records, files and so on.

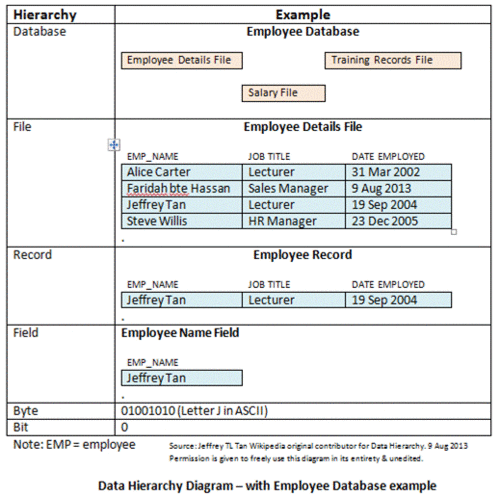
A data field holds a single fact or an attribute of an entity. Consider a date field, e.g. "September 19, 2004". This can be treated as a single date field (e.g. birthdate), or 3 fields, namely, month, day of month and year.

A record is a collection of related fields. An Employee record may contain a name field(s), address fields, birthdate field and so on.

A file is a collection of related records. If there are 100 employees, then each employee would have a record (e.g. called Employee Personal Details record) and the collection of 100 such records would constitute a file (in this case, called Employee Personal Details file).

Files are integrated into a database. This is done using a Database Management System.If there are other facets of employee data that we wish to capture, then other files such as Employee Training History file and Employee Work History file could be created as well.

An illustration of the above description is shown in this diagram below.



With reference to the example in the above diagram:

Data field label = Employee Name or EMP\_NAME

Data field value = Jeffrey Tan

The above description is a view of data as understood by a user e.g. a person working in Human Resource Department. The above structure can be seen in the hierarchical model, which is one way to organize data in a database. In terms of data storage, data fields are made of bytes and these in turn are made up of bits.

**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.

**RECOVERY SCHEMES (WAL)**

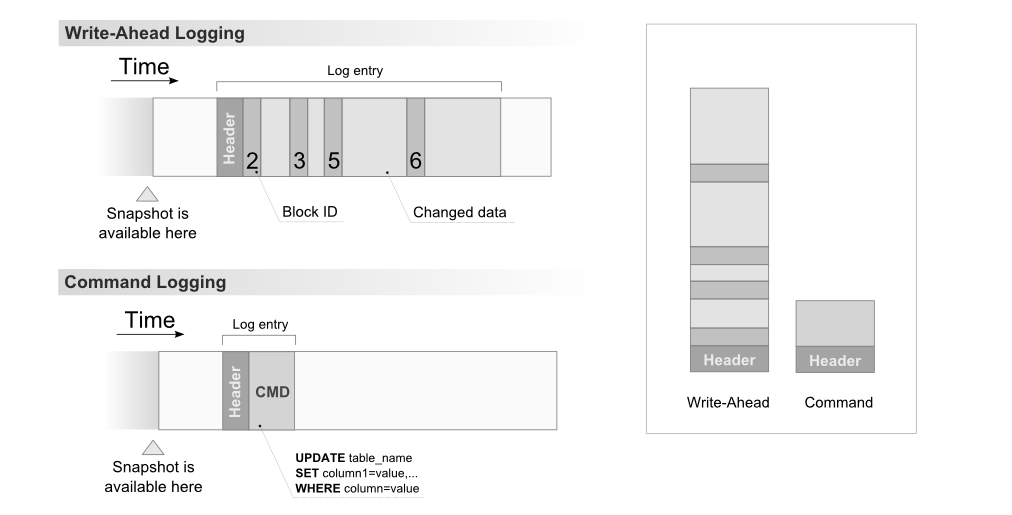
WAL protocol Write-ahead logging (WAL) is a family of techniques for providing atomicity and durability (two of the ACID properties) in database systems.

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.

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.



**FAILURE WITH LOSS OF NON-VOLATILE STORAGE**

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; Here a backup copy of the database is stored at a remote location from where it can be restored in case of a catastrophe.

Alternatively, database backups can be taken on magnetic tapes and stored at a safer place. This backup can later be transferred onto a freshly installed database to bring it to the point of backup.

Grown-up databases are too bulky to be frequently backed up. In such cases, we have techniques where we can restore a database just by looking at its logs. So, all that we need to do here is to take a backup of all the logs at frequent intervals of time. The database can be backed up once a week, and the logs being very small can be backed up every day or as frequently as possible.

**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.

**RECOVERY IN MULTIDATABASE SYSTEM**

To maintain the atomicity of a multidatabase transaction, it is necessary to have a two-level recovery mechanism. A global recovery manager, or coordinator, is needed to maintain information needed for recovery, in addition to the local recovery managers and the information they maintain (log, tables).

The coordinator usually follows a protocol called the two-phase commit protocol, whose two phases can be stated as follows:

**Phase 1:** When all participating databases signal the coordinator that the part of the multidatabase transaction involving each has concluded, the coordinator sends a message "prepare for commit" to each participant to get ready for committing the transaction. Each participating database receiving that message will force-write all log records and needed information for local recovery to disk and then send a "ready to commit" or "OK" signal to the coordinator. If the force-writing to disk fails or the local transaction cannot commit for some reason, the participating database sends a "cannot commit" or "not OK" signal to the coordinator. If the coordinator does not receive a reply from a database within a certain time out interval, it assumes a "not OK" response.

**Phase 2:** If all participating databases reply "OK," and the coordinator’s vote is also "OK," the transaction is successful, and the coordinator sends a "commit" signal for the transaction to the participating databases. Because all the local effects of the transaction and information needed for local recovery have been recorded in the logs of the participating databases, recovery from failure is now possible. Each participating database completes transaction commit by writing a [commit] entry for the transaction in the log and permanently updating the database if needed. On the other hand, if one or more of the participating databases or the coordinator have a "not OK" response, the transaction has failed, and the coordinator sends a message to "roll back" or UNDO the local effect of the transaction to each participating database. This is done by undoing the transaction operations, using the log.

The net effect of the two-phase commit protocol is that either all participating databases commit the effect of the transaction or none of them do. In case any of the participants—or the coordinator—fails, it is always possible to recover to a state where either the transaction is committed or it is rolled back. A failure during or before Phase 1 usually requires the transaction to be rolled back, whereas a failure during Phase 2 means that a successful transaction can recover and commit.