# Assignment

## Advance Database Systems

Course Code: IT-5101


*Submitted to:*
Dr. Rashed Mazumder
Assistant Professor
Institute of Information Technology
Jahangirnagar University

*Submitted by:*
Tajim Md. Niamat Ullah Akhund
Roll: 1120 (MSc)

## A. Why reliability and availability are important for Distributed Database Management System?

**Ans:**

A distributed database (DDB) is a collection of multiple, logically inter related databases distributed over a computer network. A distributed database management system (D–DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users.

Distributed database system (DDBS) = DDB + D–DBMS.

**Reliability and availability are important for Distributed Database Management System:**

Reliability is an attribute of any computer-related component (software, or hardware, or a network, for example) that consistently performs according to its specifications. It refers to the probability that the system under consideration does not experience any failures in a given time interval.

$$R(t) = Pr\{0 \text{ failure in time } [0,t] \text{ no failures at } t = 0\}$$

Where R(t) : reliability of the system

Availability is the degree to which a system is operational and accessible when required for use. It refers to the probability that the system is operational according to its specification at a given point in time t.

$$A = \mu / \zeta + \mu$$

Where $\zeta$ is a failure rate

$\mu$ is a mean repair time

Availability is a liveness guarantee, while reliability is a safety guarantee. Systems that provide both reliability and availability are often said to be fault-tolerant.

A distributed database has multiple nodes (computers) and if one fails than others are required to be available to do the job. Reliability and availability properties do that for DDBMS respectively by running the system efficiently most of the time and making the system continuously usable or accessible during a time interval, like a failure does not make the entire system inoperable.

## B. Explain. MTBF and MTTR for Distributed Database Management System.

**Ans:**

**MTBF:**

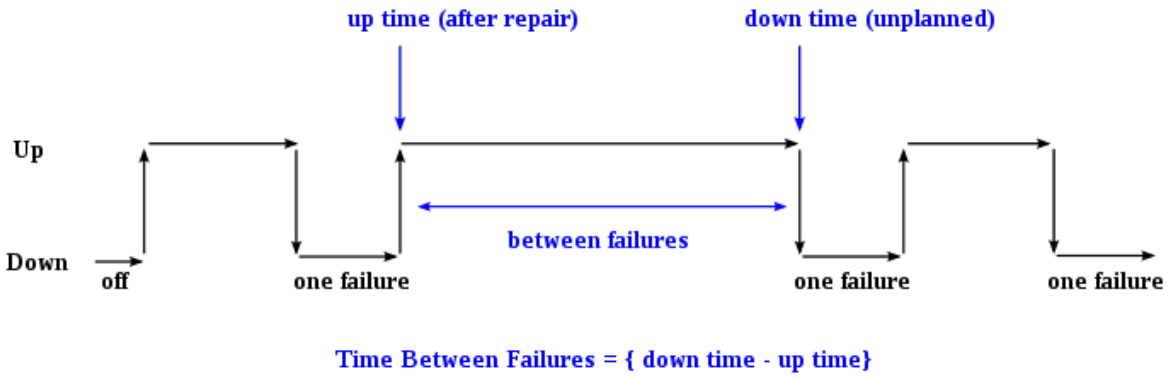
**Mean time between failures** (**MTBF**) is the predicted elapsed time between inherent failures of a mechanical or electronic system, during normal system operation. MTBF can be calculated as the arithmetic mean (average) time between failures of a system. The term is used for repairable systems, while **mean time to failure** (**MTTF**) denotes the expected time to failure for a non-repairable system. The higher the MTBF, the longer a system is likely to work before failing.

MTBF is defined by the arithmetic mean value of the reliability function $R(t)$, which can be expressed as the expected value of the density function $f(t)$ of time until failure:

$$\mathrm{MTBF} = \int_0^\infty R(t)\,dt = \int_0^\infty t f(t)\,dt$$

Mean time between failures (MTBF) describes the expected time between two failures for a repairable system. For example, three identical systems starting to function properly at time 0 are working until all of them fail. The first system failed at 100 hours, the second failed at 120 hours and the third failed at 130 hours. The MTBF of the system is the average of the three failure times, which is 116.667 hours. If the systems are non-repairable, then their MTTF would be 116.667 hours.

In general, MTBF is the "up-time" between two failure states of a repairable system during operation as outlined here:




**MTTR:**

**Mean time to repair** (**MTTR**) is a basic measure of the maintainability of repairable items. It represents the average time required to repair a failed component or device. Expressed

mathematically, it is the total corrective maintenance time for failures divided by the total number of corrective maintenance actions for failures during a given period of time. It generally does not include lead time for parts not readily available or other Administrative or Logistic Downtime (ALDT)MTTR is often part of a maintenance contract, where a system whose MTTR is 24 hours is generally more valuable than for one of 7 days if mean time between failures is equal, because its Operational Availability is higher.

Steady State availability of a system with exponential failure and repair rates can be specified as,

$$A = MTBF/(MTBF + MTTR)$$

## C. What are the purposes of data replication for Distributed Database management system?

**Ans:**

**Data Replication:**

**Data Replication** is the process of storing data in more than one site or node. It is useful in improving the availability of data. It is simply copying data from a database from one server to another server so that all the users can share the same data without any inconsistency.

There can be full replication, in which the whole database is stored at every site. There can also be partial replication, in which some frequently used fragment of the database are replicated and others are not replicated.

**Purposes of data replication for Distributed Database management system:**

**1.Increased reliability and availability –** there are many copies of same data in several different locations (usually different geographical locations). Hence, failure of any sites (servers) will not affect the transactions.

**2. Queries requesting replicated copies of data are always faster (especially read queries) –** Distributed database ensures the availability of data where it is needed much. In case of replication, this is one step ahead. Yes, the complete table itself loaded locally. Hence, those queries can be answered quickly from the local site where they are initiated.

**3. Less communication overhead –** When more number of read queries is generated in a site, all of them can be answered locally. Only the queries involving different table or the queries try to write something need to use the communication links to contact other sites.

**4.Scalability -** As systems grow geographically and in terms of the number of data access points (consequently, in terms of the number of access requests), replication allows for a way to support this growth with acceptable response times.

**5.Less Data Movement over Network –**

The more replicas of, a relation are there, the greater are the chances that the required data is found where the transaction is executing. Hence, data replication reduces movement of data among sites and increases speed of processing.

## D. Discuss Mutual Consistency of Replicated Database.

**Ans:**

**Mutual consistency** refers to the replicas converging to the same value. When global transactions update copies of a data item at different sites, the values of these copies may be different at a given point in time. A replicated database is said to be in a **mutually consistent** state if all the replicas of each of its data items have identical values. What differentiates different mutual consistency criteria is how tightly synchronized replicas have to be; some ensure that replicas are mutually consistent when an update trans-action commits (thus, they are usually called strong consistency criteria), while others take a more relaxed approach (and are referred to as weak consistency criteria).

Strong mutual consistency criteria require that all copies of a data item have the same value at the end of the execution of an update transaction. This is achieved by a variety of means, but the execution of 2PC at the commit point of an update transaction is a common way to achieve strong mutual consistency.

Weak mutual consistency criteria do not require the values of replicas of a data item to be identical when an update transaction terminates. What is required is that, if the update activity ceases for some time, the values eventually become identical. This is commonly referred to as eventual consistency, which refers to the fact that replica values may diverge, but will eventually converge.

**Example:**

Consider three sites (A, B, and C) and three data items (x, y, z) that are distributed as follows: Site A hosts x, Site B hosts x, y, Site C hosts x, y, z. We will use site identifiers as subscripts on the data items to refer to a particular replica.

Now consider the following three transactions:

| T1: x ← 20 | T2: Read(x) | T3: Read(x) |
|---|---|---|
| Write(x) | y ← x + y | Read(y) |
| Commit | Write(y) | z ← (x * y)/100 |
| | Commit | Write(z) |
| | | Commit |

Note that T1's Write has to be executed at all three sites (since x that is replicated at all three sites), T2's Write has to be executed at B and C, and T3's Write has to be executed only at C. We are assuming a transaction execution model where transactions can read their local replicas, but have to update all of the replicas.

Assume that the following three local histories are generated at the sites:

HA = {W1(xA), C1}
HB = {W1(xB), C1, R2(xB), W2(yB), C2}
HC = {W2(yC), C2, R3(xC), R3(yC), W3(zC), C3, W1(xC), C1}

The serialization order in HB is T1 -> T2 while in HC it is T2 -> T3 -> T1. Therefore, the global history is not serializable. However, the database is mutually consistent. Assume, for example, that initially xA = xB = xC= 10, yB = yC = 15, and zC = 7. With the above histories, the final values will be xA = xB = xC= 20, yB = yC = 35, zC = 3.5. All the physical copies (replicas) have indeed converged to the same value.