International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 4, August 2023, pp. 4234~4245
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i4.pp4234-4245

Assimilating sense into disaster recovery databases and judgement framing proceedings for the fastest recovery

Vaheedbasha Shaik, Natarajan Kalyanasundaram
Department of Computer Science and Engineering, School of Engineering and Technology, CHRIST (Deemed to be University), Bangalore, India

Article Info

Article history:

Received Mar 22, 2022
Revised Oct 27, 2022
Accepted Nov 6, 2022

Keywords:

Cloud and on-premises database integrations
Cloud database recovery
Database disaster recovery
Intelligence to standby
Oracle on-premises database recovery

ABSTRACT

The replication between the primary and secondary (standby) databases can be configured in either synchronous or asynchronous mode. It is referred to as out-of-sync in either mode if there is any lag between the primary and standby databases. In the previous research, the advantages of the asynchronous method were demonstrated over the synchronous method on highly transactional databases. The asynchronous method requires human intervention and a great deal of manual effort to configure disaster recovery database setups. Moreover, in existing setups there was no accurate calculation process for estimating the lag between the primary and standby databases in terms of sequences and time factors with intelligence. To address these research gaps, the current work has implemented a self-image looping database link process and provided decision-making capabilities at standby databases. Those decisions from standby are always in favor of selecting the most efficient data retrieval method and being in sync with the primary database. The purpose of this paper is to add intelligence and automation to the standby database to begin taking decisions based on the rate of concurrency in transactions at primary and out-of-sync status at standby.

Corresponding Author:

Vaheedbasha Shaik
Department of Computer Science and Engineering, School of Engineering and Technology Christ (Deemed to be University)
Bangalore, India
Email: vaheedbasha.shaik@res.christuniversity.in

1. INTRODUCTION

The disaster recovery (DR) setups are replicas of the primary databases [1]. In different terminology, these replicas are named secondary (standby) databases, secondary databases, and slave databases [2]. The replicas rely on data retrieval and application methods to maintain consistency with the primary database [3]. When compared to its business competitors, Oracle technology is heading forward at a rapid pace in these data retrieval methods [4]. In the recent Oracle 21c database release, many new features have been added to DR setups [5]. Even so, there is a need for more research on adding intelligence and decision-making capabilities to the DR setups. Replication between primary and redundant standby databases can be configured in either synchronous or asynchronous mode [6]–[8]. An advantage of the asynchronous method was demonstrated over the synchronous method in the previous articles [9], [10]. The asynchronous methods of DR setups require a lot of manual work and human intervention [11]. Furthermore, it was not possible to calculate the time and sequence lag between the primary and standby databases accurately with self-intelligence. The current work aims to address these research gaps by implementing a self-image looping database link implementation and decision-making capability at standby databases. The proposed decision-making capabilities of standby are always in favor of selecting the best data retrieval method to be in synchronization with the primary database. By adding more intelligence and automation, the standby begins making decisions based on the rate of concurrency of primary transactions and the out-of-sync status at standby.

The relational database management systems (RDBMS) technologies have evolved in recent years to offer the maximum level of support in DR environments [12]–[14]. As a result, the retrieval process and application of the changes on the DR site have reached a new level. In the research and proposal model, The Oracle RDBMS was chosen to limit the scope of the paper. In the past, server resources allocated to DR environments sat idle until they turned on to support live business support [15], [16]. Nowadays, this is not the case. Instead of sitting idle, the DR databases evolved to provide read-only queries access. In addition to ensuring business continuity during catastrophic and disaster events, DR sites also started supporting real-time analytics and reports based on read-only access to DR databases [17], [18]. However, these DR features are only possible if there is effective data retrieval and synchronization between the primary and DR sites.

In earlier research work, synchronized real-time data can be viewed on DR sites using a real-time retrieval and apply method [19]. Nevertheless, this method would not be effective for highly fluctuating online transactional databases since the network bandwidth between the primary and DR sites would need to be accounted for [10]. A high-bandwidth network link cannot be set up all the time. This might result in financial strain when the network is idle. To deal with such issues, an improvised intelligence to DR databases at DR sites is required to determine how to act. According to the literature review, DR databases currently have limited decision-making capabilities [10], [19], [20]. Particularly regarding the selection of the right data retrieval option on the primary side, either using synchronization or asynchronization. In highly transactional databases, the synchronization method is not very effective, whereas asynchronization can produce very impressive results when it comes to being in sync with the primary database.

The goal of this paper is to introduce a hybrid method that can work with the proposed intelligence on DR sites. This proposed methodology can work with the default bandwidth network speed of cloud providers without purchasing additional high-speed network links. In the results section of this paper, it is demonstrated how the DR sites can take the right decisions independently to be in sync with primary sites. The proposed methodology can also be implemented on traditional on-premises environments since it is not intended only for the cloud rather it provides better utilization of existing resources.

Throughout this paper, various sections have been described. In the second section, related works outline the current functionalities of disaster databases and provide a detailed analysis of the features and limitations. An analysis was completed in the third section to know more about the network bandwidth of the cloud environment and a new self-image looping database link will be implemented. This self-image looping database link offers the most accurate view of the primary database transaction state. In addition, a new intelligent decision-making logic will be added to the standby database. As part of the fourth section of results and discussion, the results from section three will be demonstrated with proof and the codes successfully run. The fifth section concluded with a summary of the paper and explains its future scope.

2. RELATED WORKS

For every live critical database operation and management, the DR site must exist [21]. The DR sites are designed to restore business operations quickly in any domain in the scenarios of catastrophic failures or outages, with little or no downtime [20]. Asynchronous data retrieval and replication between primary and standby may take a little bit longer to catch up with the primary and It is also possible for data to be lost in asynchronous mode [22]. Data retrieval and replication work differently in the synchronous approach and the data in the DR database is kept in the same image as the primary database [23]. Furthermore, if the DR database is not configured in maximum protection mode, even the synchronous method can result in data loss [24].

2.1. Insights from the existing functionalities of disaster databases

The maximum protection (MPT) model is one of the offerings of 3 protection modes [24]. The MPT model has some limitations that slow down the transaction commit on the primary database [25]. Consequently, the transactional latency might result in the timeout error during peak business hours [25]. Despite having limitations and problems, the literature review demonstrates that the one and only MPT method can provide zero data loss on the standby side [25]. In real-time implementations, the MPT model is the least preferred option by database architects because automatic database role switching can happen at any moment which may cause an outage to the business. The other two protection modes, maximum performance (MPF) and maximum availability (MAA) are very popular and often used for disaster recovery databases. By incorporating logical intelligence into the DR databases, the proposed methodology is aimed to leverage the capabilities of MPF and MAA modes more.

The DR databases being in the MPF or MAA modes can handle transactions lag among them and primary DB by themselves with the aid of archive log recovery. Despite this, the DR is not capable enough to decide whether to use asynchronous or synchronous techniques to resolve the lag on their own. An earlier study concluded that the asynchronous method was more capable of resolving lag in the fastest time frame [10]. The synchronous method failed to keep up with the primary changes at peak business hours whereas it got impacted directly with the huge count of archive logs generations. The Oracle corporation has also developed a data guard (DG) Broker that can offer an array of automated capabilities. Still, there was no rationale for choosing the fastest method to resolve the lag [26], [27]. The contribution of the DG to the switchover and failover tasks has been exceptional and wonderful, these tasks have been made easier in the implementation with accuracy [28], [29].

2.2. Current limitations in oracle RDBMS versions and DR setups in cloud and on-premises

This paper has been narrated upon achieving the intended results from the most recent version of Oracle 21c. The narration of this section explains the transition from older versions to recent versions of Oracle technology. From Oracle 12C versions onward, the use of the name of the service has been implemented to rebuild the DR and to restore it if necessary. However, more manual work remains to be handled by database administrators (DBA). Especially for the missing data files, the DBA needs to perform a series of steps like cataloging the data files, restore process of the control files from the server of the primary database, clear process of standby archive logs, initialization of the database, and commence the process of recovery manager [30].

The Oracle 18C version exhibits impressive automation. The manual processes like restoring missed datafiles, cataloguing the datafiles, and restoring the control files from primary have been automated [31]. This same scenario was tested with Oracle 21c and it was found that manual work is still needed to clear the standby archive logs to start the database, and for activation of the recovery manager. So, the leftover automation for all the manual work will be addressed and has been automated in the implementation section of this paper.

The DG Broker or Data Guard command-line interface (DGMGRL) is very useful to get details about the lag between primary and DR databases in a time format [32]. In earlier work, it was noted that there was no way to accurately identify the lag between the primary and DR databases in the format of sequence and time format in a single representation. There were some structured query language (SQL) queries that can be used to verify the received and applying state of the archived logs. However, if there were no recent archive logs in the received status, then the results from those queries will not be accurate. Therefore, a new method of communication between the primary and DR databases was needed.

All supported earlier Oracle DB versions setup the disaster databases to resolve the lag or sequence gap by themselves with the primary databases [33]. Nevertheless, there was a void in the research area to provide more intelligence to DR databases. In a scenario, if the archive log is missed at primary, then the DR databases simply just wait for it to get it restored in the primary. Apart from it, there would be no other action from the DR databases. In another scenario, If the database is highly transactional and unpredictable with the business behavior then there will be a lag between primary and DR databases. Such lag will be caused by the lack of network speed and processing speed at the DR database. The high bandwidth network link and more central processing units (CPUs) cannot always be provided at the DR site due to limited infra budget costs. The asynchronous method has already demonstrated that the DR can be brought up in sync with the primary without additional resources [10]. Therefore, the develop of intelligence or a new logic for the DR databases is required to respond in a very cost-effective manner to deal with the lag issues. As a result of implementing the proposed method, DR databases will be able to take the necessary decisions based on the load or unresolved sequence gap scenarios. Based on the circumstances, the DR selects either sync or async as the appropriate synchronization method.

2.3. Contribution analysis: existing versus proposed methods

According to earlier research, asynchronous methods outperform synchronous methods on highly transactional databases. However, it requires a lot of manual effort to set up DR using asynchronous methods. Moreover, there was no accurate calculation process for estimating the lag between the primary and standby databases in terms of sequence and time. This research aimed to address these research gaps by implementing a self-image looping database link at standby databases and by developing a decision-making capability. The proposed method automates the asynchronous method completely and derives very accurate lag details in terms of both sequence and time at the same time.

The existing models in the disaster recovery (DR) database have limited decision-making capabilities. If there was any sudden spike in the rate of concurrent transactions in the primary database, there would not be any adoptive actions from DR databases adjusting to the load. The existing DR databases can only attempt to fetch the archive logs that have been missed. So, there will be no corrective actions in response to a lag when it occurs. In the proposed method, DR databases begin taking decisions when concurrency or predefined threshold values are reached. The outcomes of the decisions will change the characteristics of DR from synchronous to asynchronous and vice versa. These decision capabilities are not available in any of the existing DR models.

3. RESEARCH METHOD

The DR databases allow the businesses to keep running with zero or very little downtime when the disaster strikes. In DR databases, the objective is to create a replica of the data and keep the data safe from catastrophic or server failure events. This paper describes three types of research queries and leads in accordance with the proposed methodology.

- As a first step, the research work leads to identifying the answers for, what can be a better way to establish the connection to the primary database? and how can the DR database capture the lag details not only in time but also sequence-wise at the same time?
- The second step demonstrates, how accurately lag details can be collected and compared between primary and disaster recovery databases?
- The third analysis examines, how the DR databases will respond according to the situation and how the solution-driven method of making decisions is possible?

3.1. Cloud network bandwidth comparison among continents and countries

The speed of network connectivity among continents and within countries varies. this paper's analysis is being performed on oracle-based databases, So Oracle cloud infrastructure (OCI) environment has been used for better and more accurate results. In this test, the set of files has been transferred without regard to the irrespective claims made by the cloud provider to determine the actual data transfer rate per second. A total of 20 series of files have been used in the test scenario.

As illustrated in Figure 1, A series of 20 database-related files have been used to test the data transfer speed among different OCI regions of different continents and countries. The regions involved are Hyderabad and Mumbai from India and Sydney from Australia.

- Scenario 1: The data transfer speed was captured while transferring the set of 20 series of files from one fault domain (FD) to another FD within the same geographic region Sydney within the same country. According to the calculation, the average data transfer rate is 63.785 MB/s.
- Scenario 2: In this case, data transfer speeds were measured while moving files between two different regions within the same country-one is in Mumbai and the other in Hyderabad. As a result, the average data transmission rate was calculated as 5.695 MB/s.
- Scenario 3: The data transfer speed was captured while transferring 20 files from one region of Hyderabad in India to another region of Sydney in Australia as one continent to another. It was determined that the average data transfer rate is 5.575 MB/s.

The total amount of data transferred $(D_{\mathcal{s}})$ to any point within a period is directly proportional to the mean of network bandwidth $(\mu(N_b))$ can be represented as (1).

$$D_{\mathcal{s}} \propto \mu(N_b) \tag{1}$$

Hence, the product of the time duration (t) as a constant of proportionality and mean of network bandwidth $(\mu(N_b))$. derives the almost equal value for the total amount of data transferred $(D_{\mathcal{s}})$ can defined as mentioned in (2).

$$\exists t, D_{\mathcal{s}} \approx t \times \mu(N_b) \tag{2}$$

According to the Table 1 results, the disaster recovery (DR) database falls behind and lag will be created if the archive log sizes exceeded the total data transfer size at the primary database. The highly transactional or periodic aggressive nature of the transactions in primary databases makes it very difficult for architects to predict the infrastructure resources they will need. If server resources are over-allocated without proper prediction, the company will be forced to pay money for unused resources, and it directly impacts cost management. Thus, this analysis confirms the implementation of the necessity of hybrid algorithms in DR databases.

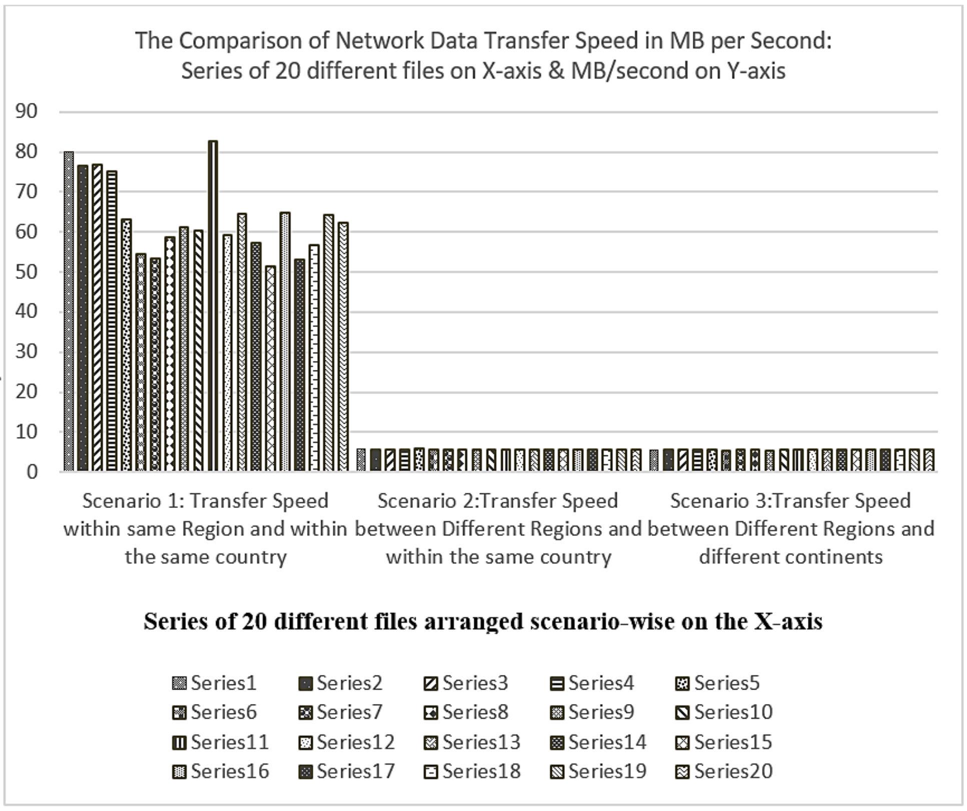


Figure 1. The comparison of network data transfer speed/second

Table 1. The expected size of data

Data Transfer Scenarios	The almost equal total sizes of transferred data
1	3,827.1 MB
2	341.7 MB
3	334.5 MB

3.2. The proposal and implementation of self-image looping database link

To pull the data from different databases across the network, a secure connection can be established among them using the database link (DB Link) [34], [35]. In this paper, the DB link is created in self-image looping mode in the primary database [36], [37]. The proposed methodology performs a comparison between primary and standby databases transaction states. The calculated lag will be the input parameter to the algorithm to take appropriate steps by itself. In the earlier research work, the lag was calculated by considering the current position and lag only at the standby side without considering the real-time details of the primary transaction state. When the standby did not receive the latest update from the primary on time, the old transaction state was considered as the current one and led to the wrong conclusions. The algorithm in this study cannot rely on ambiguous details. Hence, there is a need for a new method implementation to obtain 100% accurate details from the primary database at any time in the combined format of time lag and sequence gap.

As can be seen in Figure 2 a self-image looping database link has been implemented. There are circled numbers in the figure to denote each step and its order of description. As part of step 1, the transparent network substrate (TNS) option is used to create the DB link [36], [38]. As far as the Oracle primary database is concerned, this is a committed transaction. Step 2 entails tracking and transporting the committed transactions from the primary DB via a network. The transaction details will be picked from the online logs or archive logs in an encrypted format to the standby site. In step 3, the encrypted transaction will be logged in standby redo logs on the standby side. In step 4, the logged transaction at standby will be applied to the standby database and TNS entry of the file '*tnsnames.ora*' in Oracle home will be modified and routed towards the primary. In steps 5 and 6, the self-image looping DB link is tested for connectivity and query results. The algorithmic pseudo-code of the self-image looping of DB link is as algorithm 1:

Algorithm 1. The implementation of self-image looping database link (DB Link)

```
Input: Create Database Link at Primary
Output: The Database Link will be formulated to work towards upstream environment
Step 1: Start
Step 2: In Secondary DB: Open Database in read-only mode;
Step 3: In Secondary DB: Start (Media Recovery);
Step 4: In Primary DB: Create (Database Link: db_link);
Step 5: In Secondary DB: Verify DB Link in view (DBA_DB_LINKS);
Step 6: In Secondary DB: Open file (tnsnames.ora) in Path: $ORACLE_HOME/network/admin;
Step 7: In Secondary DB: Add (TNS entry) of Primary DB;
Step 8: In Secondary DB: Verify SQL and Result;
Step 9: end;
```

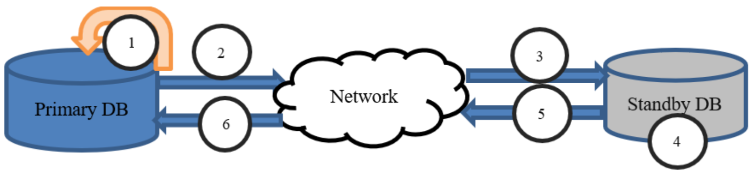
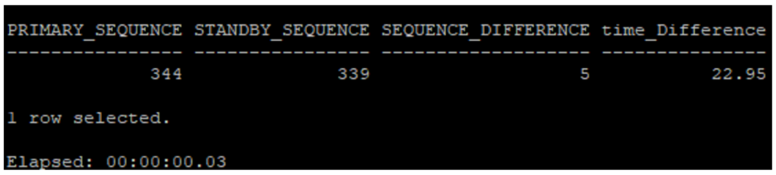


Figure 2. The self-image looping DB link

After execution of Step 8 from the above algorithm the derived results are as shown in Figure 3. The result provides a detailed comparison of Lag between primary and standby databases. The SQL query retrieves accurate transaction state of primary database through DB link '*db_link*' and displays the delay with sequence numbers as well as in time.

```
SQL> SELECT
  p.SEQUENCE # "PRIMARY_SEQUENCE" , s.SEQUENCE# "STANDBY_SEQUENCE",
(p.SEQUENCE# - s.SEQUENCE#) "SEQUENCE_DIFFERENCE",(p.COMPLETION_TIME -
s.COMPLETION_TIME)*24*60 "time_Difference"
from
   (
    SELECT
      THREAD #, SEQUENCE#, COMPLETION_TIME FROM V$ARCHIVED_LOG@db_link
WHERE DEST_ID=1 and (THREAD#, SEQUENCE#) IN (SELECT THREAD#,
MAX(SEQUENCE#) FROM V$ARCHIVED_LOG@db_link GROUP BY THREAD#)) p,
      (
        SELECT
          THREAD #, SEQUENCE#, COMPLETION_TIME FROM V$ARCHIVED_LOG WHERE
(THREAD#, SEQUENCE#) IN (SELECT THREAD#, MAX(SEQUENCE#) FROM
V$ARCHIVED_LOG WHERE APPLIED='YES' GROUP BY THREAD#)) s
        where
          p.THREAD #=s.THREAD#;
```



```
PRIMARY_SEQUENCE STANDBY_SEQUENCE SEQUENCE_DIFFERENCE time_Difference
---------------- ---------------- ------------------- ---------------
             344              339                   5           22.95

1 row selected.

Elapsed: 00:00:00.03
```

Figure 3. The execution of self-image looping DB link with SQL query and the details of lag

3.3. The proposal and implementation of disaster database intelligence (DDI) algorithm

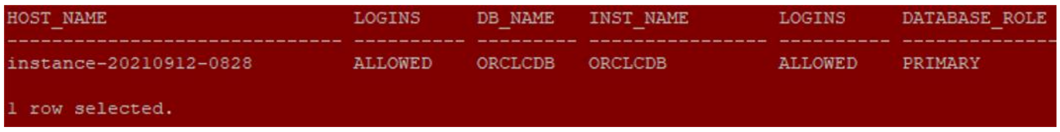
The standby databases can be synchronized either synchronously (sync) or asynchronously (async). During previous research, the author demonstrated that the sync method experienced some challenges when dealing with highly transactional databases, whereas the async method addressed them effectively. As a result, it shows that the combination of sync and async methods is necessary. these both methods should be incorporated together in standby behavior so that the standby gets the capability to choose the right method according to the transaction concurrency on the primary DB. Even in the very latest version of Oracle 21c, the standby databases have been configured with limited intelligence [39]. They can just resolve the missing or corrupted archive logs by themselves. There is still a significant research gap in adding more intelligence. In this paper, the author enhances the standby database's intelligence by implementing decision-making capabilities to choose the best synchronization method based on transaction load, sequence gap, and calculated time lag between the primary and standby databases. In the elaborated format, the algorithmic pseudo-code for disaster database intelligence (DDI) is as algorithm 2:

Algorithm 2. Disaster database intelligence (DDI)

```
Input: Add Decision making logic to the DB
Output: The Secondary (Standby) DB start changing
The roles whenever conditions met.
Step 1: Start
Step 2: In Secondary DB: export ORACLE_SID; export ORACLE_HOME;
Step 3: In Secondary DB:
     if [! -f "RECOVERY_TRIGGERED.lck”]
            then
               if (lag_time > 30 minutes) && ( Sequecne gap > 4)
               then collect(sequence gap);
            create file(RECOVERY_TRIGGERED.lck);
Step 4: In Secondary DB: Trigger (Recovery);
Step 5: In Secondary DB: restart DB mount;clear(Standby logs)
Step 6: In Primary DB: Execute SQL (ALTER SYSTEM ARCHIVE LOG CURRENT;);
Step 7: In Secondary DB: restart DB; open (all pluggable databases);
Step 8: In Secondary DB: Start (MRP);
Step 9: In Secondary DB: rm (RECOVERY_TRIGGERED.lck);
Step 10: end;
```

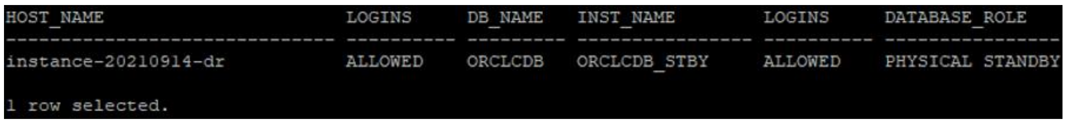
4. RESULTS AND DISCUSSION

As shown in Figures 4 and 5. The primary and standby (DR) databases have been configured and The DR setup is in read-only mode. The role of the database confirms the nature of its behavior towards end-user transactions. DML operations are allowed in the primary database, while read-only operations are allowed in the physical standby. As a result, the reports can be generated on a standby database.



```
HOST_NAME                      LOGINS     DB_NAME   INST_NAME        LOGINS     DATABASE_ROLE
------------------------------ ---------- --------- ---------------- ---------- ----------------
instance-20210912-0828         ALLOWED    ORCLCDB   ORCLCDB          ALLOWED    PRIMARY

1 row selected.
```

Figure 4. The primary database configuration



```
HOST_NAME                      LOGINS     DB_NAME   INST_NAME        LOGINS     DATABASE_ROLE
------------------------------ ---------- --------- ---------------- ---------- ----------------
instance-20210914-dr           ALLOWED    ORCLCDB   ORCLCDB_STBY     ALLOWED    PHYSICAL STANDBY

1 row selected.
```

Figure 5. The standby database configuration

The official data sets from Oracle's websites have been used in this simulation to get accurate results. In loop mode, the transaction concurrency count has been increased by using shell scripts. The process of generating load on the primary database is not a new one, and many fine procedures can be found across the web. Using any of the methods should be sufficient to pick it up, but it is recommended to execute them with multiple sessions to generate enough load of data definition language/data manipulation language (DDL/DML) transactions commit. In this paper, the load generation topic is omitted since already existed across the web [40]. However, the reader can choose any load generation method of his choice, but the only thing that ought to be remembered is that it should generate enough transaction concurrent load on primary DB.

As stated in earlier research, the various information systems share and transform data through the metadata system when multiple source environments are considered as sources [41]. The proposed model stores and retrieves metadata information in a uniform format since block-to-block copies are used in physical DR databases. A uniform database architecture is applicable to all domains, including data mining, image processing, data mining, and the internet of things (IoT). In the field of image processing and facial recognition, the data must be identical in both the primary and secondary databases. When there is a very slight mismatch between the data of an individual image, the ability of image processing methods to recognize faces is greatly diminished [42]. In some image processing techniques near the edges, the guided image filter is effective as a smoothing operator [43]. It is possible to store processed image content after smoothing in primary databases only, not in DR databases since those are read-only. In data mining, novel patterns can be extracted from datasets through knowledge discovery [44]. These data mining activities can also be performed on both primary and DR databases. Artificial intelligence is heavily reliant on IoT devices. By studying current barriers to the IoT, it could be possible to predict future applications that can be deployed globally [45]. Several time-consuming and resource-consuming testing activities must be performed on the data to identify such barriers. Such stress testing activities can be carried out with both the primary and DR databases, or it can also be feasible to offload the work to the DR database. A methodology for designing secure databases was presented in earlier research [46], [47]. Today, very advanced and advanced encryption methods are available. Once it is implemented at the primary level and automatically, it will also be applied at the DR level.

Figure 6 illustrates that the Lag was observed at the standby side as the load was gradually increased on the primary. As a result, the standby DB begins to fall behind the primary database and runs out of sync. This lag is caused by media recovery process (MRP) slowness to apply the changes at the same rate as they are generated in primary or by insufficient network bandwidth speed between primary and standby. There is a need to either increase the network bandwidth or enhance the processing speed at the standby databases to resolve this issue. However, both traditional solutions will have a negative impact on cost management. A key feature of the proposed methodology is that it tackles these issues very effectively without impacting the infra budget.

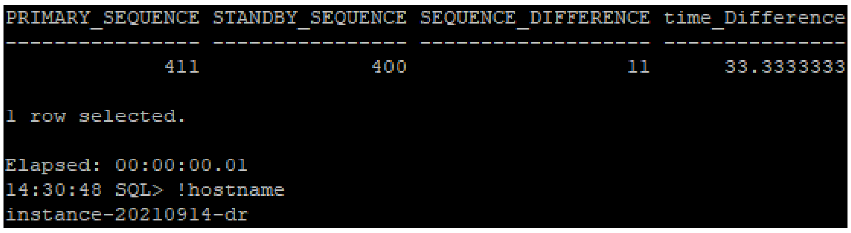

```
PRIMARY_SEQUENCE STANDBY_SEQUENCE SEQUENCE_DIFFERENCE time_Difference
---------------- ---------------- ------------------- ---------------
             411              400                  11      33.3333333

1 row selected.

Elapsed: 00:00:00.01
14:30:48 SQL> !hostname
instance-20210914-dr
```

Figure 6. The incident of lag

According to the findings and results, the total data size of committed transactions at primary (T_c) is a significant cause of lag (L_g) creation at standby on the time factor can be defined as (3), (4), and (5).

$$if\ T_c > \mu(D_s),\ L_g \propto t \tag{3}$$

$$if\ T_c < \mu(D_s),\ L_g \propto t^{-1} \tag{4}$$

$$if\ T_c \approx \mu(D_s), L_g \geq 0 \tag{5}$$

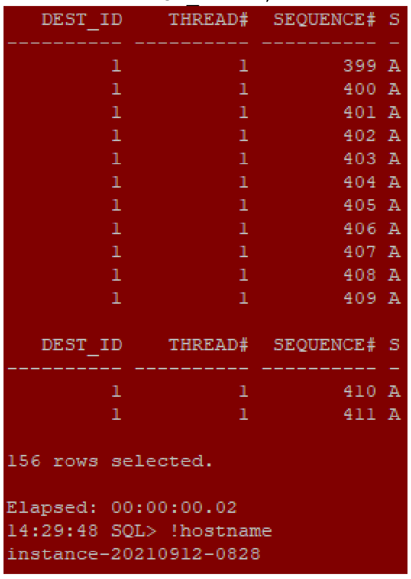
The two conditions such as whenever the sequence difference greater than 4 and the timestamp difference greater than 30 minutes are met, the proposed methodology of standby database will overrule the synchronization process and switch to adopting asynchronous process. The standby database starts pulling archive logs, incremental backups, and newly added data files from the primary database and proceeds to apply them to the standby database. Once the standby database has been successfully recovered, the recovery process automatically switches the replication method from asynchronous to synchronous mode. As a result, the standby DB will be renewed to handle the real-time analytics reports as before.

In Figure 6, these two conditions have been met. Therefore, the standby database began resolving replicating issues automatically. In the end, the DR database will be synchronized with the primary database without the need for any manual intervention or DBA involvement. As shown in Figure 7, the status is described in detail.

As depicted in Figure 7, the 'SEQUENCE#' column lists the sequence number and their status with the 'S' column. The 'S' column has the value as 'A', which means the archive log is archived and is ready to be shipped to the standby site to get applied in it. The most recently archived log sequence is '411'. The 'DEST_ID' shows the primary side pointer id for the archive log destination. The column 'THREAD#' represents the instance-specific process for archive log generation.

Figure 8 depicts the significant delay or lag between the primary and standby databases. As per sequence message, the standby was attempting to apply sequence ‘401’. However, the primary has already archived sequence ‘411’ most recently. As a result of the very high number of concurrent transactions at primary, the gap keeps on increasing. Here, the proposed methodology takes over the control with the intelligence of the decision-making capabilities and resolves the gap issues automatically. The proposed methodology will be proven if the database acknowledges the status of recovery accurately. This accuracy can be achieved if the column “APPLIED” value becomes “YES” for sequence 411.

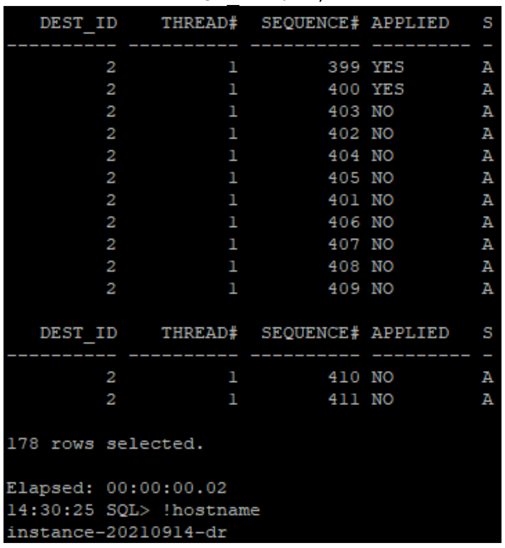
```
SQL> select DEST_ID, THREAD#, SEQUENCE#, STATUS, COMPLETION_TIME from V$ARCHIVED_LOG where
DEST_ID =1;
```



```
   DEST_ID    THREAD#  SEQUENCE# S
---------- ---------- ---------- -
         1          1        399 A
         1          1        400 A
         1          1        401 A
         1          1        402 A
         1          1        403 A
         1          1        404 A
         1          1        405 A
         1          1        406 A
         1          1        407 A
         1          1        408 A
         1          1        409 A

   DEST_ID    THREAD#  SEQUENCE# S
---------- ---------- ---------- -
         1          1        410 A
         1          1        411 A

156 rows selected.

Elapsed: 00:00:00.02
14:29:48 SQL> !hostname
instance-20210912-0828
```

Figure 7. The presentation of the current state of primary DB

```
SQL> select DEST_ID, THREAD#, SEQUENCE#, STATUS, COMPLETION_TIME from V$ARCHIVED_LOG where
DEST_ID !=1;
```

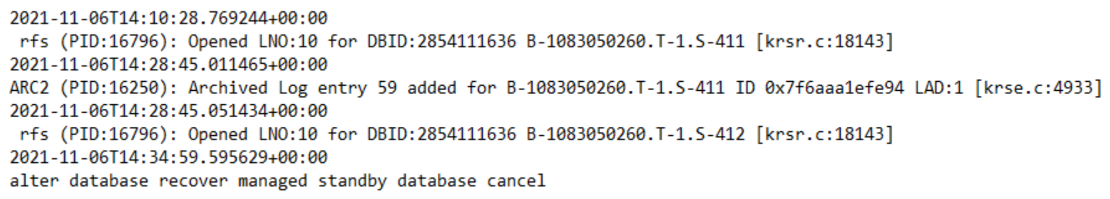


```
   DEST_ID    THREAD#  SEQUENCE# APPLIED   S
---------- ---------- ---------- --------- -
         2          1        399 YES       A
         2          1        400 YES       A
         2          1        403 NO        A
         2          1        402 NO        A
         2          1        404 NO        A
         2          1        405 NO        A
         2          1        401 NO        A
         2          1        406 NO        A
         2          1        407 NO        A
         2          1        408 NO        A
         2          1        409 NO        A

   DEST_ID    THREAD#  SEQUENCE# APPLIED   S
---------- ---------- ---------- --------- -
         2          1        410 NO        A
         2          1        411 NO        A

178 rows selected.

Elapsed: 00:00:00.02
14:30:25 SQL> !hostname
instance-20210914-dr
```

Figure 8. The presentation of the current state of standby DB

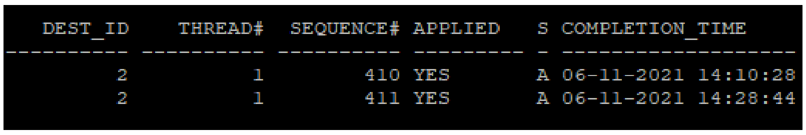
Figure 9 displays the proof of the proposed methodology working according to the code. The standby alert log shows the timestamp of the very recently received sequence ‘411’ and the open state of the remote file server (RFS) for sequence ‘412’. As shown in Figure 6, the sequence gap was stood at ‘11’ and the standby fell in ‘33’ minutes behind the primary. Hence, the proposed methodology of the standby database had been provoked automatically and started working to trigger the first command of the code ‘alter database recover managed standby database cancel;’ from a proposed logic.

According to Figure 10, the proposed intelligence for standby was successfully executed and achieved the desired outcomes. The results of the validation indicate that the standby with primary databases synchronized successfully and the lag between primary and standby databases was resolved. A value of YES is displayed in the column ‘APPLIED’ indicating that all transactions from the archive logs of 410 and 411 have been successfully updated to the standby database. In the standby database, a column named ‘COMPLETION_TIME’ represents the time stamp details associated with the completion of a sequence. Since each sequence will be applied to the database in the numbering order of sequences, the timestamps will always be in ascending order. 'ARCHIVE' means that the sequence number is applied to the standby database and then archived successfully. The archived sequence can be deleted from the file system as well since it is no longer required.



```
2021-11-06T14:10:28.769244+00:00
 rfs (PID:16796): Opened LNO:10 for DBID:2854111636 B-1083050260.T-1.S-411 [krsr.c:18143]
2021-11-06T14:28:45.011465+00:00
ARC2 (PID:16250): Archived Log entry 59 added for B-1083050260.T-1.S-411 ID 0x7f6aaa1efe94 LAD:1 [krse.c:4933]
2021-11-06T14:28:45.051434+00:00
 rfs (PID:16796): Opened LNO:10 for DBID:2854111636 B-1083050260.T-1.S-412 [krsr.c:18143]
2021-11-06T14:34:59.595629+00:00
alter database recover managed standby database cancel
```

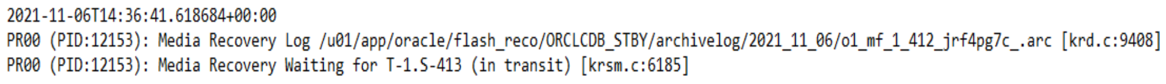
Figure 9. The event occurrence of methodology in alert log of standby DB



```
   DEST_ID    THREAD#  SEQUENCE# APPLIED   S COMPLETION_TIME
---------- ---------- ---------- --------- - -------------------
         2          1        410 YES       A 06-11-2021 14:10:28
         2          1        411 YES       A 06-11-2021 14:28:44
```

Figure 10. The event of successful recovery

The proof of operations is illustrated in Figure 11. When the server timestamp ‘2021-11-06T14:36:41.618684+00:00’ was reached, the synchronization was completed, and the subsequent sequence ‘412’ at standby DB was applied. The MRP process began to wait for the arrival of subsequence sequence ‘413’ at standby DB. Before the proposed intelligence starts working, there was ‘33’ minutes of lag between the primary and standby databases. After successfully executing the proposed methodology, recovery was achieved in ‘101’ seconds.



```
2021-11-06T14:36:41.618684+00:00
PR00 (PID:12153): Media Recovery Log /u01/app/oracle/flash_reco/ORCLCDB_STBY/archivelog/2021_11_06/o1_mf_1_412_jrf4pg7c_.arc [krd.c:9408]
PR00 (PID:12153): Media Recovery Waiting for T-1.S-413 (in transit) [krsm.c:6185]
```

Figure 11. The event of exact timestamp recording of successful recovery

The total time taken for recovery and synchronization:

$$\begin{aligned} &= (Logic\ Start\ time) - (Logic\ End\ time) \\ &= (2021-11-06T14{:}34{:}59.595629+00{:}00) - (2021-11-06T14{:}36{:}41.618684 \\ &\qquad\qquad + 00{:}00) \\ &= 1.69\ minutes\ (or)\ 101.62\ seconds. \end{aligned}$$

5. CONCLUSION

In the event of a catastrophic failure or outage, disaster recovery sites may prove to be extremely valuable for bringing the business back online as soon as possible. Standby databases presented many challenges from the perspective of keeping them in sync with highly transactional primary databases. In previous research, the Author has proposed one of the possible and helpful solutions to get standby databases in sync within a short period of time. However, it was always a choice of the user and a manual process that triggered it. The proposed methodology in this paper provided the intelligence that allowed the standby databases to take appropriate actions by themselves automatically based on current conditions. The proposed methodology has been able to recover the standby database and achieve synchronization more quickly than a manual process. This paper concentrates solely on Oracle databases to limit the scope of study and narrow down the research area. The proposed methodology is not yet implemented in other technologies of databases. This future research area is open to all researchers.

REFERENCES

[1] A. Sukma, Suharjito, and Diana, “Replication system of oracle database standard edition by utilizing traditional archived log,” *Procedia Computer Science*, vol. 135, pp. 140–146, 2018, doi: 10.1016/j.procs.2018.08.159.

[2] J. Lindström, V. Raatikka, and J. Ruuth, “IBM solidDB: in-memory database optimized for extreme speed and availability,” *IEEE Data Engineering Bulletin*, vol. 36, no. 2, pp. 14–20, 2013.

[3] A. M. Salman, P. Sailaja, G. Venkataswamy, and N. P. Supriya, “Database replication: a survey of open source and commercial tools,” *International Journal of Computer Applications*, vol. 13, no. 6, pp. 1–8, 2011.

[4] R. Chopade and V. K. Pachghare, “Ten years of critical review on database forensics research,” *Digital Investigation*, vol. 29, pp. 180–197, Jun. 2019, doi: 10.1016/j.diin.2019.04.001.

[5] Oracle, “Oracle database 21c,” *Oracle*, 2020. https://docs.oracle.com/en/database/oracle/oracle-database/21/whats-new.html (accessed Mar. 21, 2022).

[6] T. Pohanka and V. Pechanec, “Evaluation of replication mechanisms on selected database systems,” *ISPRS International Journal of Geo-Information*, vol. 9, no. 4, Apr. 2020, doi: 10.3390/ijgi9040249.

[7] K. K. Ezechiel, S. Kant, and R. Agarwal, “Analysis of database replication protocols,” in *2nd International Conference on Recent Multidisciplinary Research*, 2018, pp. 75–83.

[8] H. Nakanishi *et al.*, “Design for the distributed data locator service for multi-site data repositories,” *Fusion Engineering and Design*, vol. 165, Apr. 2021, doi: 10.1016/j.fusengdes.2020.112197.

[9] A. V Kalayda, “Promising directions for the development of modern databases,” *Journal of Physics: Conference Series*, vol. 2131, no. 2, Dec. 2021, doi: 10.1088/1742-6596/2131/2/022087.

[10] V. Shaik and K. Natarajan, “Asynchronous method of oracle: a cost-effective and reliable model for cloud migration using incremental backups,” in *Lecture Notes in Networks and Systems*, vol. 290, 2021, pp. 417–427.

[11] L. Wong, N. S. Arora, L. Gao, T. Hoang, and J. Wu, “Oracle streams: a high performance implementation for near real time asynchronous replication,” in *2009 IEEE 25th International Conference on Data Engineering*, Mar. 2009, pp. 1363–1374, doi: 10.1109/ICDE.2009.121.

[12] T. Shareef, K. Sharif, and B. Rashid, “A survey of comparison different cloud database performance: SQL and NoSQL,” *Passer Journal of Basic and Applied Sciences*, vol. 4, no. 1, pp. 45–57, Feb. 2022, doi: 10.24271/psr.2022.301247.1104.

[13] M. G. Kahn *et al.*, “Migrating a research data warehouse to a public cloud: challenges and opportunities,” *Journal of the American Medical Informatics Association*, vol. 29, no. 4, pp. 592–600, Mar. 2022, doi: 10.1093/jamia/ocab278.

[14] Y. Fu and C. Soman, “Real-time data infrastructure at uber,” in *Proceedings of the 2021 International Conference on Management of Data*, Jun. 2021, pp. 2503–2516, doi: 10.1145/3448016.3457552.

[15] J.-H. Hwang, M. Balazinska, A. Rasin, U. ˇ C. Etintemel, M. Stonebraker, and S. Zdonik, “A comparison of stream-oriented high availability algorithms.” Tech. Rep. CS-03--17.

[16] Oracle, “Active data guard FAQ,” *Oracle*. https://www.oracle.com/database/technologies/faq-active-data-guard.html (accessed Mar. 21, 2022).

[17] P. Carbone, M. Fragkoulis, V. Kalavri, and A. Katsifodimos, “Beyond analytics: the evolution of stream processing systems,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, Jun. 2020, pp. 2651–2658, doi: 10.1145/3318464.3383131.

[18] Oracle, “Concepts and administration,” *Oracle*. http://ora-srv.wlv.ac.uk/oracle19c_doc/sbydb/managing-oracle-data-guard-physical-standby-databases.html (accessed Mar. 21, 2022).

[19] B. P. Reviewed *et al.*, “Cloud data security enhancement approach for redundant server failure,” *Al-Dar Research Journal for Sustainability*, vol. 3, no. 2, pp. 1–38, 2019.

[20] N. Z. Azeemi, O. Al-Basheer, and G. Al-Utaibi, “Zero down time-smart data guard for collaborative enterprise dataware systems,” *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 16, pp. 3282–3293, 2020.

[21] CMO, “Disaster recovery best practices,” Ministry of Electronics and Information Technology (MeitY), 2017. Accessed: Mar. 21, 2022. [Online]. Available: https://www.meity.gov.in/writereaddata/files/DR%20Best%20Practices.pdf

[22] E. Roth and A. Haeberlen, “Do not overpay for fault tolerance!,” in *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, May 2021, pp. 374–386, doi: 10.1109/RTAS52030.2021.00037.

[23] M. A. Hossain, M. I. Hasan, M. R. Islam, and N. Ahmed, “A novel recovery process in timelagged server using point in time recovery (PITR),” in *2021 24th International Conference on Computer and Information Technology (ICCIT)*, Dec. 2021, pp. 1–5, doi: 10.1109/ICCIT54785.2021.9689808.

[24] P. Potineni *et al.*, “Oracle data guard concepts and administration, 19c,” *Oracle Help Center*. Accessed: Mar. 21, 2022. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/19/sbydb/data-guard-concepts-and-administration.pdf

[25] Oracle, “Oracle 21c new feature-prepare database for data guard,” *Oracle*. https://oracledbwr.com/oracle-21c-new-feature-prepare-database-for-data-guard/ (accessed Mar. 21, 2022).

[26] D. S. B. Shabnam Bhura, “E-conference on artificial intelligence and machine learning (eC-AI & ML) 2020: Research opportunities and applications in the field of engineering & science,” in *Tcetmumbai.in*, D. S. V. Bijith Marakarkandy, Ed. 2020, pp. 64–69.

[27] Oracle, “Oracle data guard broker properties,” *Oracle*. 2017, https://docs.oracle.com/database/121/DGBKR/dbpropref.htm (accessed Mar. 21, 2022).

[28] mdin, “How to change RMAN config for standby DB,” *Thinking Out Loud*. https://mdinh.wordpress.com/2021/10/12/how-to-change-rman-config-for-standby-db/ (accessed Mar. 21, 2022).

[29] C. K. Wee and R. Nayak, "Adaptive load forecasting using reinforcement learning with database technology," *Journal of Information and Telecommunication*, vol. 3, no. 3, pp. 381–399, Jul. 2019, doi: 10.1080/24751839.2019.1596470.
[30] "Rolling forward standby database when archives missing in primary in 12c," *DBACLASS*. https://dbaclass.com/article/rolling-forward-standby-database-when-archives-missing-in-primary-in-12c (accessed Mar. 21, 2022).
[31] "Oracle database 18c: Roll forward physical standby using RMAN incremental backup in single command," *The geek diary*. https://www.thegeekdiary.com/oracle-database-18c-roll-forward-physical-standby-using-rman-incremental-backup-in-single-command/ (accessed Mar. 21, 2022).
[32] Oracle, "Oracle data guard command-line interface reference," *Oracle Help Center*. https://docs.oracle.com/database/121/DGBKR/dgmgrl.htm#DGBKR595 (accessed Mar. 21, 2022).
[33] D. Burleson, "Data guard fal_client and fal_server," *Burleson Consulting*. 2015. https://www.dba-oracle.com/t_data_guard_fal_server_fal_client.htm (accessed Mar. 21, 2022).
[34] M. Macías and J. Guitart, *Distributed applications and interoperable systems*, vol. 12718. Cham: Springer International Publishing, 2021.
[35] F. Wang and N. Helian, "Scanning once a large distributed database to mine global association rules by growing a prefix tree for each local transaction," *Management Information Systems*, vol. 7, pp. 3–18, 2003.
[36] Oracle, "Use database links with autonomous database," *Oracle Help Center*. https://docs.oracle.com/en/cloud/paas/autonomous-database/adbsa/database-links.html (accessed Mar. 21, 2022).
[37] R. Nawaz and T. R. Soomro, "Role of Oracle active data guard in high availability database operations," *International Journal of Applied Information Systems*, vol. 5, no. 5, pp. 18–21, Apr. 2013, doi: 10.5120/ijais13-450914.
[38] E. Chen, "What GLOBAL_NAMES do to DB Links," *Ed Chen Logic*. https://logic.edchen.org/what-global_names-do-to-db-links/ (accessed Mar. 21, 2022).
[39] A. A. Nour, "Security criteria and indicators of RDBMSS: a comparative study," *IOSR Journal of Computer Engineering*, vol. 19, no. 02, pp. 56–67, Apr. 2017, doi: 10.9790/0661-1902035667.
[40] "Example scripts for Oracle - Testing and scalability," *Use the Index, Luke!* https://use-the-index-luke.com/sql/example-schema/oracle/performance-testing-scalability (accessed Mar. 21, 2022).
[41] F. F. Hasan and M. S. A. Bakar, "An approach for metadata extraction and transformation for various data sources using R programming language," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 26, no. 3, pp. 1520–1529, Jun. 2022, doi: 10.11591/ijeecs.v26.i3.pp1520-1529.
[42] Y. S and N. Purnachand, "Convolutional neural network-based face recognition using non-subsampled shearlet transform and histogram of local feature descriptors," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 4, pp. 1079–1090, Dec. 2021, doi: 10.11591/ijai.v10.i4.pp1079-1090.
[43] Y. S. and P. N., "An effective face recognition method using guided image filter and convolutional neural network," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 23, no. 3, pp. 1699–1707, Sep. 2021, doi: 10.11591/ijeecs.v23.i3.pp1699-1707.
[44] M. A. Al-Hagery, "Extracting hidden patterns from dates' product data using a machine learning technique," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 3, pp. 205–214, Dec. 2019, doi: 10.11591/ijai.v8.i3.pp205-214.
[45] F. G. Abdulkadhim, Z. Yi, C. Tang, M. Khalid, and S. A. Waheeb, "A survey on the applications of IoT: an investigation into existing environments, present challenges and future opportunities," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 3, Jun. 2020, doi: 10.12928/telkomnika.v18i3.15604.
[46] W. C. Alisawi, A. A. AlMuhsen Hussain, and W. A. Alawsi, "Estimate model of system management for database security," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 14, no. 3, pp. 1391–1394, Jun. 2019, doi: 10.11591/ijeecs.v14.i3.pp1391-1394.
[47] M. A. Naji, D. A. M. Hammood, and N. A. R. Al Debagh, "Methods for database ciphering," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 17, no. 3, pp. 1499–1505, Mar. 2020, doi: 10.11591/ijeecs.v17.i3.pp1499-1505.

BIOGRAPHIES OF AUTHORS

Vaheedbasha Shaik earned a Master of Technology (M.Tech) in Database Systems from SRM University in India. Currently pursuing a Ph.D. in Computer Science Engineering from CHRIST (Deemed to be University), Bangalore, India. He is currently a Senior Consultant in Deloitte USI for the database administration role. He is a working professional on cloud computing, distributed database systems, database administration, database development, database real application cluster administration, database disaster recovery administration, and cloud infrastructure design and are among his research interests. He can be contacted at email: vaheedbasha.shaik@res.christuniversity.in

Natarajan Kalyanasundaram earned a Ph.D. in Computer Science & Engineering. He has about Twenty-Four years of experience in Teaching Computer Science & Engineering Subjects (University college level) in India and abroad. He has about Four years of experience in Industries level Software Training. He is currently working as an Associate Professor, Department of Computer Science and Engineering, CHRIST (Deemed to be University), Bangalore, India. His interested research areas are networking, cyber security, RDBMS databases infrastructure configurations, cloud computing, on-prem cloud integrations with the cloud. He can be reachable at email: natarajan.k@christuniversity.in.