Database Security, Backup, and Recovery Procedures

For an IT help-desk scenario

Marcelo Fernandes
Software Engineer
Wellington, New Zealand
marceelofernandes@gmail.com

I. Introduction

With the presence of regular security threats, database backup strategies are essential for most modern technology systems. These strategies can be composed of multiple preventive actions and controlled procedures that are programmed to happen in the event of a database error, corruption, or attack. Different levels of effort can be made to maintain proper security of a database. Some protections such as role-based privileges can be considered as a cheap solution, whereas sophisticated pipelines that backup and restore data can be quite expensive to build and maintain depending on the criticality and volume under which the data is being operated.

II. IT help-desk scenario

In order to demonstrate the development of backup and recovery routines we will consider an IT help-desk database scenario implemented using Oracle. In this scenario, operators will take calls from callers who are seeking to solve a problem from a given piece of equipment. If the operator cannot solve the problem, they will assign the problem to a specialist based on a problem type. All the data captured during the process is further used for reporting. The database schema for this scenario is found in Fig. 1.

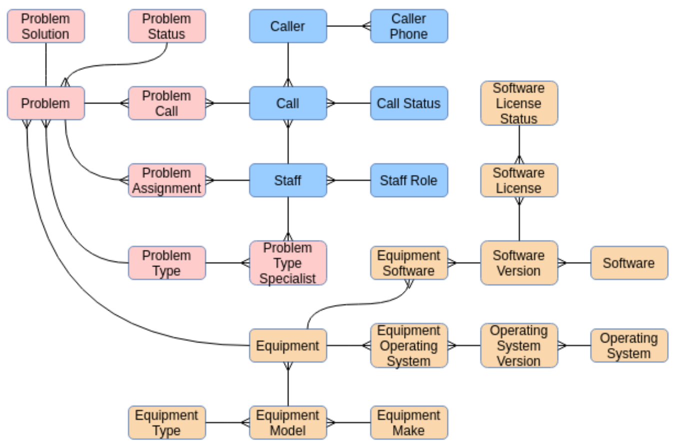

Fig. 1. IT help-desk schema.

One major consideration for this scenario is that it is not acceptable to lose any data if a disk failure damages any of the database files. For that reason, as recommended by Oracle [1], the database will run on ARCHIVELOG mode. This mode will create redo log archives during run time, which can be used together with a full backup to completely restore the database without data loss. Additionally, this mode allows the DBA to perform hot backups, i.e, to backup the database while the database is still online, avoiding any down-time.

In order to enhance the security against failures, an up-to-date standby database can run side by side with the original one by continuously applying the redo logs. In this business case, having a standby database ready to use means that if a database failure happens during business hours, the application database connection can be switched to the standby database and the operators can resume their work while the DBA inspects what went wrong with the principal database. The standby database should function in a completely separated machine so that problems related to the original database machine don't cascade onto the standby one.

Considering that a downtime would block operators' work, any database downtime caused during business hours should be avoided. Having a low tolerance for downtime means that more regular backups need to happen. Minimising the time window between backups will result in less redo log archives, and the database will therefore be quicker to restore. In this scenario, a full backup could be generated daily as the volume of data won't be large enough to be problematic.

Exceptional operations that make structural changes on the database should also have their own backup routine. A backup should be triggered before and after these operations are run. On ARCHIVELOG, Oracle suggests [1] that the DBA creates a control file backup using either the RMAN or ALTER DATABASE command, along with the BACKUP CONTROLFILE parameter. These kinds of operations can be potentially dangerous and should be triggered only during non-business hours (if possible).

The backup strategy is then a composition of database mirroring (standby database), incremental backups (via redo logs), and hot backups (via ARCHIVELOG mode). Along with these strategies, it is important to take into account eventual disasters (earthquakes, tsunamis, etc.) that could end up in equipment loss. For that reason, generating more than one backup copy and spreading these across servers in different locations should be considered.

This scenario is built on the premises of a multi-user environment. Different staff members will be manipulating the data and internal threats may occur. Users can deliberately use their access privilege to jeopardise, steal, destroy, or perform other malicious actions on the data. In order to avoid those actions, the DBA should grant only the necessary permissions to each user. This can be managed via user groups, i.e, operators should belong to one group that has a set of permissions for the tables that they manipulate during their work flow, but no more than that [2].

Simple access to the database should also be avoided. Passwords used to connect to the database should not be simplistic. Additionally, the connection to the server where the database is hosted should only be available through a VPN with a 2-factor authentication framework to ensure that outside threats are minimised [3].

On the application layer, SQL injection (SQLi) threats are one of the most dangerous when it comes to database-driven applications [4]. Attackers can exploit queries to bypass authentication, obtain private data, and to run any arbitrary code. In order to be protected against these threats, the application needs to put emphasis on escaping all user supplied input, and on using prepared statements with parameterised queries so that attackers cannot change the intent of a query. If the application is to be hosted online as a web-application, it should be built around a modern framework where security threats and risks of SQLi are already mitigated. In this help-desk scenario, there isn't a need for users outside of the organisation to use the application. This means that the application could be hosted in a local network, or through an organisation-owned VPN.

III. Conclusion

More database threats are developed and perfected each day [5]. With these, comes the challenge of continuously adapting database security measures. The strategy presented here for the proposed scenario is, in reality, the basic set of security measures needed for such an application. A few topics to expand for this scenario in a future essay should take into account: Stored Procedures, use of Views, controlling the operating system permissions, anti-virus programs, data encryption (moving away from plain-text), among other security measures.

References

[1] Docs.oracle.com. n.d. *Backup And Recovery Strategies*. [online] Available at: https://docs.oracle.com/cd/B10501_01/server.920/a96519/strategy.htm [Accessed 27 October 2020].

[2] Ibm.com. 2019. Database Security: An Essential Guide. [online] Available at: https://www.ibm.com/cloud/learn/database-security [Accessed 27 October 2020].

[3] Sumitra, B., 2014. A Survey of Cloud Authentication Attacks and Solution Approaches. International Journal of Innovative Research in Computer and Communication Engineering, 2(10).

[4] Zhang, L., Zhang, D., Wang, C., Zhao, J. and Zhang, Z., 2019. ART4SQLi: The ART of SQL Injection Vulnerability Discovery. IEEE Transactions on Reliability, 68(4), pp.1470-1489.

[5] Cherry, D., 2011. Securing SQL Server. Burlington, MA: Syngress.