ISSN 2277-3061

Data Access Layer: A Programming Paradigm on Cloud

Pratiyush Guleria
Systems Analyst, NIELIT Chandigarh, B.O Shimla, INDIA
pratiyushguleria@gmail.com

Abstract

Database is important for any application and critical part of private and public cloud platforms. For compatibility with cloud computing we can follow architectures like three tier architecture in .Net Technologies such that database layer should be separate from user and business logic layers. There are some other issues like following ACID properties in databases, providing dynamic scalability by using Shared-disk Architecture and efficient multi-tenancy, elastic scalability, and database privacy.

Keywords: Multi-tenancy, Elastic Scalability, Business, Shared-disk.

Journal: International Journal of Computers & Technology
Vol 11, No.3
editor@cirworld.com
www.cirworld.com, member.cirworld.com

1. INTRODUCTION

There are two database architectures—shared-disk and shared-nothing—for their compatibility with cloud computing. The shared-disk database architecture is ideally suited to cloud computing. The shared-disk architecture requires fewer and lower-cost servers, it provides high-availability, it reduces maintenance costs by eliminating partitioning, and it delivers dynamic scalability. The database architecture called shared-disk, which eliminates the need to partition data, is ideal for cloud databases. Shared-disk databases allow clusters of low-cost servers to use a single collection of data, typically served up by a Storage Area Network (SAN) or Network Attached Storage (NAS). All of the data is available to all of the servers; there is no partitioning of the data. The shared-disk DBMS architecture has other important advantages—in addition to elastic scalability—that make it very appealing for deployment in the cloud [1]. As part of the Cloud computing the service model Database-as-a-Service (DBaaS) has been recognized, where application can access highly available, scaled, and elastic data store services on demand with the possibility of paying only for the resources are actually consumed.

2. STORAGE ARCHITECTURES IN DATABASE SYSTEMS

Shared-nothing and shared-disk are two widely-used storage architectures in database systems. Shared-nothing Storage architecture involves data partitioning which splits the data into independent sets. These data sets are physically located on different database servers. Each server processes and maintains its piece of the database exclusively which makes shared-nothing databases easily scalable. Due to inherent scalability, applications designed to work on shared-nothing storage architecture are suitable for Cloud. But data partitioning used in this architecture does not work well with cloud. It is very difficult to virtualize a shared-nothing database as it becomes very complex and difficult to maintain due to data partitioning. It needs a piece of middleware to route database requests to the appropriate server. As more servers are added, data has to be repartitioned. Data partitioning should be done very carefully, otherwise data shipping (passing of the information from one machine to the other machine for processing) and joining will become difficult. More data shipping means more latency and network bandwidth bottlenecks. These issues reduce database performance badly [8].Shared-disk is a master-master configuration, so each node provides fail-over for the other nodes. This reduces the number of servers required by half when using a shared-disk database. Higher CPU Utilization can be achieved using Shared-disk.Using shared-disk database we can purchase lower-cost commodity servers instead of paying a large premium for high-end computers. This also extends the lifespan of existing servers, since they needn't deliver cutting-edge performance [1].

3. DATABASE AS A SERVICE

Early DBaaS efforts include Amazon RDS (Relational Database Service) and Microsoft SQL Azure, which are promising in terms of establishing the market need for such a service, but which do not address three important challenges: efficient multi-tenancy, elastic scalability and database privacy [2]. A good DBaaS must support database and workloads of different sizes. The challenges arise when a database workload exceeds the capacity of a single machine. A DBaaS must therefore support scale-out, where the responsibility for query processing (and the corresponding data) is partitioned amongst multiple nodes to achieve higher throughput. Database Replication and Failure Detection, Failover are required for fault-tolerant databases [5].Database As a Service (DBaaS) provides agility to customers due to the simplicity of the data access and the fact that they don't need extensive knowledge of the underlying data. Its implementation is easy because the changes are minimal [9].

4. PROGRAMMING MODEL FOR APPLICATION DESIGN

Windows Azure is Microsoft's platform for cloud computing. Moving data layer to the Cloud introduces an issue how an application can access data from the Cloud data store services with full functionality of accessing like traditional database service. Microsoft offers little functionality regarding elasticity. Microsoft also offers SQL Azure for databases in the cloud. To ensure this possibility, the application needs to implement a Data Access Layer (DAL) separately in order to enable access to Cloud data, where DAL is responsible for encapsulating the data access functionalities and interacts with business logic within the application system. This reduces the application complexity and brings the solutions for managing entire enterprise data [6]. Application can be designed in such a manner that database layer works separately and business layer interacts with database layer using object and getters, setter's property. Data Access Layer (DAL) is a common layer in all layered architecture of Information Systems. One of the most important steps in architectural design of information systems is designing the DAL [12]. The first function of the Data Access Layer is supported by the Data Layer Classes which implement at least the select, update, insert and delete functions as the minimal set needed for the operation of a typical database. The Data Access Layer lies between the Object Model of the Application and the Data Model of the database thus functioning like an adapter between the two. The second sub-layer of the Data Access Layer consists of the Data Type Classes which are in memory recipients of the entity classes instances in the format of the Data Model corresponding to the persistent storage [13].Most existing tools operate at a database level and by separating databases it provides high data isolation as well as it is easy to move the application from on-premises to a hosted environment. Example of Business Logic and Data Access Layer are mentioned below. In this example, three properties are used i.e. EID,SALARY,NAME in a business class "emp" and is implemented in Data Access Layer using "insert" function.

4.1 Business Layer

```
namespace Three_Tier_Arch.Business
{
public class emp
{
   private int id, salary;
    private string name;
    public int EID
    {
        get {return id ;}
        set {id=value;}
    }
  public int SALARY
    {
        get {return salary ;}
        set {salary = value ;}
 }
 public string NAME
    {
        get {return name ;}
        set {name = value ;}
    }
    public void insert ()
 {
     Three_Tier_Arch.Data.emp.insert (this);
}
}
}
```

4.2 Data Access Layer

```
namespace Three_Tier_Arch.Data
{
public class emp
{
    private static readonly string _connectionString = string. Empty;
    public static void insert (Three_Tier_Arch.Bussiness.emp obj)
   {
    SqlConnection con = new SqlConnection (_connectionString);
    SqlDataAdapter adp=new SqlDataAdapter ("insert into
emp(empid,empname,empsalary)values(@empid,@empname,@empsalary)",con);
    DataSet ds = new DataSet ();
    adp.SelectCommand.Parameters.AddWithValue ("@empid", obj.EID);
    adp.SelectCommand.Parameters.AddWithValue ("@empname", obj.NAME);
    adp.SelectCommand.Parameters.AddWithValue ("@empsalary", obj.SALARY);
    adp.Fill(ds,"emp");
    }
    static emp()
        {
         _connectionString =WebConfigurationManager.ConnectionStrings["emp"].ConnectionString;
        if(string.IsNullOrEmpty(_connectionString))
            throw new Exception("No connection string configured in Web.Config file");
       }
}
```

5. DATABASE PARTITIONING AND DESIGN APPROACHES

Relational Cloud uses database partitioning for two purposes: (1) to scale a single database to multiple nodes, it becomes useful when the load exceeds the capacity of a single machine, and (2) to enable more granular placement and load balance on the back-end machines compared to placing entire databases [2]. Partitioning, also known as sharding, is used by cloud data management systems to achieve scalability. There is a variety of partitioning schemes used by different systems on different levels. Some systems partition data on the file level while others horizontally partition the key space or table [3].Systems may be strictly row-based, or allow for column storage. In row-based storage all of a record's fields are stored contiguously on disk. With column storage, different columns or groups of columns can be stored separately (possibly on different servers). Row-based storage supports efficient access to an entire record (including low latency reads and insertion/update in a serving-oriented system), and is ideal if we typically access a few records in their entirety. Column-based storage is more efficient for accessing a subset of the columns, particularly when multiple records are accessed [4]. There are some issues and design approaches followed by recent systems in building a replicated database over a wide area network. When data is replicated over large geographical distances, then it becomes hard to maintain ACID properties [7]. There are some systems which don't follow all the ACID properties like Amazon SimpleDB, Yahoo's PNUTS; Google BigTable has weak Isolation, Consistency and Atomicity on Shared-nothing Architecture. But to ensure integrity of the data, Data Access Layers need to maintain the ACID properties so as to provide fault-tolerant services to users [14].

6. CONCLUSION

Data Access Layer provides functionality related to security as well as helps in performing intensive computational tasks directly in the storage and it helps in saving bandwidth and computing resources on the client application. It communicates with user and business layers. The Data Access Layer becomes more helpful when scalability increases on cloud. It provides a way to design an application with a clean separation of code into their functional areas within an application and it also helps in simplifying the architecture which is required on cloud. On DAL, we incorporate database CRUD (Create, Read, Update and Delete) operations which is also helpful on Cloud based platform i.e. Azure. Apart from simplifying the Architecture, properties of DAL can be reused on other applications.

REFERENCES

[1] Mike Hogan, Cloud Computing & Databases, Scaling-up, Scaling-out & Scaling-in, November 14, 2009

[2] Carlo Curino, et.al," Relational Cloud: A Database-as-a-Service for the Cloud".

[3] Siba Mohammad,et.al,"Cloud Data Management: A Short Overview and Comparison of Current Approaches"

[4] Brian F. Cooper, et.al,"Benchmarking Cloud Serving Systems with YCSB", SoCC'10, June 10-11, 2010, Indianapolis, Indiana, USA.Copyright 2010 ACM 978-1-4503-0036-0/10/06.

[5] http://acid.mht.

[6] http://elib.uni-stuttgart.de/opus/volltexte/2012/7803/pdf/MSTR_3305.pdf

[7] Daniel J. Abadi," Data Management in the Cloud: Limitations and Opportunities", Copyright 2009 IEEE

[8] Indu Arora,Dr. Anu Gupta," Cloud Databases: A Paradigm Shift in Databases", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 3, July 2012

[9] http://en.wikipedia.org/wiki/Data_as_a_service.

[10] Varsha Jawale, et.al,"Secure Cloud Based Document Management System", International Journal of Engineering Research & Technology (IJERT), Vol.2 Issue 3, March-2013.

[11] Ms.Sonali S.Kale, Prof.Mr.Ravindra H.Borhade,"Implementation of SaaS Multitenancy in Cloud Computing", International Journal of Electronics Communication and Computer Engineering", Vol 4, Issue 1, ISSN (Online):2249-071X, ISSN (Print):2278-4209.

[12] GholamAli Nejad HajAli Irani, Zahra Jafari," Performance analysis on data access patterns in layered Information Systems, an Architectural Perspective", International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1, May 2012,ISSN (Online): 1694-0814

[13] Robert Dollinger,Daniel V.Goulet,David Gibbs "Automating the Development of Data Access Layer", Information Systems Education Journal,Vol 4,Number 32,July 13,2006,ISSN:1545-679X.

[14] Pratiyush Guleria,"ACID Support and Fault-Tolerant Database Systems on Cloud: A Review", International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE), Vol 1, Issue 8, October 2011, ISSN: 2277-9043.

[15] http://www.visioncloud.eu,Deliverable D30.1,Data Access Layer: Design and Open Spec, Release 1.0

[16] https://secure.techrepublic.com

BIBLIOGRAPHY OF AUTHOR

Pratiyush Guleria

Presently working as Systems Analyst in NIELIT (National Institute of Electronics and IT), Chandigarh, Branch Office Shimla, H.P, INDIA.

Educational Qualification: Mtech (Gold Medalist) in Computer Science from Himachal Pradesh University, Shimla.

MBA From Indira Gandhi National Open University(IGNOU)

Btech in Information Technology from I.E.E.T Baddi, Distt Solan, Himachal Pradesh University.

Experience: More Than 5 Years of Experience in Teaching and Software Field.

Areas of Interest: Cloud Computing,Web Technologies,Data Mining,Knowledge Management.