12 Rules for a Cloud Data Management System (CDMS)

0 Modern Superset of an RDBMS

A CDMS is a modern superset of an RDBMS designed to meet the needs of modern, 21st century applications and deployment infrastructures.

In addition to delivering the full range of RDBMS capabilities, including SQL, ACID transactions, and supporting all of the tools and APIs that come with them, a CDMS must:

- Support modern datacenter hardware and management frameworks,
- Meet peak workload demands,
- Handle structured and unstructured datasets, and
- Support non-SQL paradigms.

In particular, a CDMS must embrace modern dynamic and flexible cloud computing environments.

1 Elastic Scale-out for Extreme Performance

A CDMS must deliver capacity on demand by adding or deleting computational and storage resources in a running database. A CDMS must be able to elastically scale out to very high transaction volumes – in the millions of transactions per second (TPS) – and web-scale database sizes – in the petabytes of data – by the addition of real or virtual machines, networks and storage systems to a live database. A CDMS must also scale in gracefully when resources are no longer needed.

2 Single Logical Database

No matter how complicated the application a CDMS must present its users the view of a single, logical, consistent and always available database. A CDMS must shield users from having to employ explicit partitioning, sharding or caching techniques to achieve massive database scalability. The CDMS must obviate or encapsulate these complexities, so that a developer or administrator can focus on using the database no matter the scale or complexity.

3 Run Anywhere, Scale Anywhere

A CDMS must be able to run on any infrastructure from single machines to private clouds, public clouds and combinations of the above. It must be able to run in a heterogeneous environment incorporating different machines, virtual machines, operating systems, or network infrastructures. A CDMS should excel on enterprise and commodity hardware equally.

4 Nonstop Availability

A CDMS must be capable of running continuously – for months or years – without failing or being made unavailable for maintenance.

A CDMS cannot have a single point of failure. It must presume infrastructure failure and be self-aware to detect it, handle systems changes and recognize extreme events like network partitions. It should

remain available, or if impossible then fail in a graceful and consistent fashion. It should be able to decide how to react to network partitions, either by failing some portion of the system or understanding how to reconcile changes when the network is stabilized.

A CDMS must support live rolling upgrades of hardware, operating systems and CDMS versions, and must support dynamic changes to schemas and other database administration tasks without shutting down CDMS availability.

5 Dynamic Multi-tenancy

A CDMS must be dynamically multi-tenant. It must be able to manage large numbers of databases on a finite set of resources, and be able to reassign resources to databases as needed. A CDMS must be able to hibernate inactive databases and wake them on demand.

6 Active/Active Geo-distribution

A CDMS must be able to run concurrently in multiple datacenters to support geographically distributed workloads, always-on applications, and for disaster recovery. A CDMS must deliver active/active operations with transactionally consistent semantics, work across and between Wide Area Networks and understand how to localize activity or caches.

7 Embrace Cloud

A CDMS must integrate and run in a cloud environment, and be designed to support cloud-scale performance requirements while being resilient against the inherent concurrency and latency challenges. It must be able to provide transactions per second (TPS) and load rate guarantees and maintain those rates as latency spikes and concurrent load grows. It must support cloud management frameworks and integrate with modern cloud stacks.

8 Store Anywhere, Store Redundantly

A CDMS must be able to store the data anywhere: Locally, remotely, in a datacenter or on a public or private cloud. A CDMS should also be able to store the data in whatever storage system is appropriate: on a directly attached file system, a local Key/Value store or on a cloud-based storage service. It should be able to store all data redundantly in multiple locations, simultaneously and with transactional consistency, using a heterogeneous mix of storage locations and storage technologies.

9 Workload Mix

A CDMS must be flexible in the kinds of workloads it supports, and efficient in running different workloads concurrently. It should be able to support highly scalable web-facing applications with primary requirements that include high transaction throughput, web-scale concurrency and very low latency. It should be able to support enterprise applications that involve complex transactions and a more even mix of reads and updates. It should be able to support analytical applications, with a premium on un-cached reads and long-running queries. A CDMS should also support logging-style applications with a focus on sustained appending of data.

A CDMS must be able to perform backups without taking the system down, and run analytical queries without interfering with transaction processing.

10 Tunable Durability Guarantees

A CDMS must allow a user to define infrastructure reliability constraints that control the trade off between durability guarantees and database performance. A user must be allowed to define whether a transactional commit means that the data is safely written to storage in one place, written to storage in K places, written to

storage in M non-collocated places, stored in RAM in N places, or something else.

11 Distributed Security

A CDMS must have enterprise class security at system level and database level, including:

- Authentication and access control of machines before they are accepted into the trusted group,
- Authentication and access control of database processes before they are allowed to participate in a particular database,
- Encryption of all communications between machines, and
- Database-level security for users of the database.

12 Empower Developers & Administrators

Empowering Developers:

- A CDMS must support rapid application development and frictionless application evolution,
- It should be easy to use, without time-consuming requirements for provisioning of database servers, or inflexible schemas that slow down application development,
- It must be integrated with modern programming languages and APIs, database development tools and application development frameworks,
- It must support flexible schemas with user-defined types in order to provide a clean layer for arbitrary language integration that is agnostic to row or column orientation. A CDMS should enable users to easily update and redefine data as their applications change.

Empowering Administrators:

- A CDMS should provide a single secure point of administration for all its databases and resources. It should make it simple to automate logging, auditing, profiling, process management and resource allocation,
- It should enable policy-driven, zero-admin services that manage the system as a whole,
- A CDMS should also support the separation of database administrators and systems administrators as roles using this single point of management as these roles are more distinct in a cloud environment.

215 First Street, Suite 005
Cambridge, MA 02142
+1 (617) 500-0001
www.nuodb.com