Best Practices for Moving to the Cloud using Data Models in the DaaS Life Cycle

Nuccio Piscopo

www.ERwin.com
# Table of Contents

Abstract........................................................................................................................................................................3

Introduction..................................................................................................................................................................3

Migrating Data to the Cloud: DaaS Direction.............................................................................................................3

Maas: Introducing and Applying Cloud Best Practices..............................................................................................7

Conclusion .................................................................................................................................................................10

Glossary ....................................................................................................................................................................10

Acknowledgments ......................................................................................................................................................11

References .................................................................................................................................................................11

About the Author .....................................................................................................................................................11
Abstract
Organizations are moving to the Cloud. As a result, administrative costs and the burdens of on-premise collaboration management are significantly reduced. Along with the benefits that Cloud implementations bring, however, come concerns about the following: data security, coherence, outage, availability and geo-location. Using a data model in migrating data from on-premise applications to the Cloud throughout DaaS should be the common way to minimize database development, implementation and maintenance resources, allowing companies to focus on their core business.

Introduction
In this paper, we propose guidelines that affect all aspects of Cloud DaaS services: security, performance, connections, and the security life cycle to govern what is really important in migrating to on-demand hosted applications, and managing this data in the Cloud. We introduce DaaS Best Practices based upon the DaaS Life Cycle for organizations planning to move to the Cloud. As data is moved to a Cloud-based architecture, it is important to keep visibility into core data structures and systems as data applications are created, shared and changed. This is what a company has to do to prepare for loading data onto a Cloud Service and to manage the business aspects in preparing to implement an on-demand Cloud provider offering. Finally, we apply the concept of MaaS through DaaS Best Practices by means of a proper use case as well.

Migrating Data to the Cloud: DaaS Direction
In the Relational Cloud model, DaaS requirements must satisfy the following, in order:

1. Efficient multi-tenancy;
2. Dynamic scalability;
3. Database privacy.

Within the DaaS infrastructure, data models are the service providers (MaaS or Model as a Service). Customers can use these models to satisfy the following DaaS properties:

- **Implementing and sharing database structure models.** This is the main way to better understand the challenges introduced into the database. Models should be implemented and deployed using a model-driven paradigm;
- **Verifying database model properties according to private and public Cloud requirements.** The data model contains properties mapping the Cloud service; partitions, accesses, rights and location, for example, have to be always documented into the model. With this approach, model differences are based on regular discovery, classifying the service changes and updating the archive;
- **Designing and testing new query types.** Specific query classes need to support heterogeneous database environments. Typical examples are aggregation queries and text pattern matching queries;
- **Designing of the data storage model.** The model should enable query processing directly against databases to strengthen privacy levels and secure changes from database providers;
- **Modeling databases to calculate “a priori” physical resources allocation.** How many resources does the service need? Do database partitions influence resource allocation and/or replication? Modeling the database means designing the service; calculating a priori these magnitudes drives both deployment and database growth;
- **Modeling databases to predict usage “early” and to optimize database handling.** Performance and availability are two of the main goals promised by the Cloud. Usage is not directly dependent upon the infrastructure and, as a consequence, could be a constraint. Calculating the usage rate means understanding the data application life cycle and then optimizing the database service properties;
- **Designing multi-database structures to control and document pricing schemes (databases elasticity and scalability).** Partitioning is the key to controlling deployment. Models contain deployment properties and map the host database service. Therefore, the model designs on-premise database elasticity and scalability. Still, we will see later that multi-database design is a way to control data persistence.
Elasticity, multi-tenancy, scalability together with physical and logical architectures represent the right guidelines to plan a strategy of data migration to the service Cloud although data and applications must be secured by introducing the following additional properties:

- data location;
- data persistence;
- data discovery and navigation;
- Cloud to Cloud comingling;
- backup;
- querying & aggregation;
- data inference.

All of the above properties are data model defined and consequently prior to defining and applying DaaS directions we have to clarify some important aspects regarding their relation to data security. The latter has three main, essential risk mitigation features to be underlined:

1. **confidentiality:** nobody can access, modify, share or use data without authorization;
2. **availability:** allows full authorized access response time through service continuity and data integrity;
3. **on-demand data secure deleting/shredding:** when the Cloud agreement ends or a breach occurs, the Customer can request that all or part of the data be destroyed. Finally, when the service is closed both the Customer and Provider have to be sure that data is completely and effectively erased and unrecoverable.

However, when loading data onto the Cloud Service, the organization is normally focused on the following key features:

- near-zero configuration and administration;
- high-performance;
- high availability;
- fast updates at low cost;
- multi-database partitioning;
- migration.

The above properties should be integrated through the DaaS best practices, as outlined in Figure 1 below.
The DaaS life cycle introduced above has a basic principle: it is data model designed. Data models play an integral role in defining, managing and protecting data in Cloud Computing scenarios (both Private and Public). Modeling is the key guideline in the DaaS architecture. In fact, models are “a priori” on-premise databases because they collect behaviors, documents and information concerning structures, access rights, security and database scaling, partitioning and evolution. CA ERwin® Data Modeler (ERwin) covers these aspects and governs the DaaS life cycle/workflow that can be modified, if necessary, according to Customer/Provider organizations’ constraints.

The following are definitions for the lifecycle states outlined in Figure 1.

**Create DB Model** - The data model contains information and properties about the deployed databases and partitioning architecture:
- Model partitioning contains location constraints specifying where data is stored in case of data retrieving. The database architecture and partitioning are defined at the model level;
- Commingling should be forbidden according to the partitioning specifications defined by the data model architecture;
- Backup is model-driven through the architecture and partitioning specifications defined by the model;
- Querying and aggregation are defined in the model. Any breach by an external query or aggregation not defined in the model is a data violation and therefore must be forbidden;
- Inference is defined in the model. Hence, although best practice prohibits breaches, they can be authorized according to query specifications defined and protected by model properties;
- When creating data models, it is important to define properties for confidentiality, availability, authenticity, authorization, authentication and integrity;
- Models contain configuration and administration properties; partitions define DBMS deployment, hardening, machines, scripts and stored procedures. Database forward/reverse engineering is continually updated as the DBMS changes;

**Generate/Update DB** – Automatic database generation and synchronization help in revising the model and its properties:
- Configuration and administration is simple and fast when it is model-driven. Generation has to meet the DBMS properties for different databases.

**Populate, Use and Test DB** – The database has to be run and tested ahead of deployment. This phase triggers continuous improvement:
- Before deploying the DB, a map of data has to be defined. Remote tests have to be regularly performed to discover deployed data. This helps in assuring data and regulatory authority compliance and legal controls;
- Queries and aggregation remote tests have to be regularly performed to discover deployed data. This also helps in assuring data and regulatory authority compliance and legal controls;
- Application monitoring and optimization classify object’s logic and level controls defined when creating the model. Here it is important that models are DBMS solution designed so object level controls are the appropriate ones;
- Application logic has to be constantly monitored. Continuous improvement is always active and, if needed, it applies to the Deploy & Share state.

**Deploy & Share** – This state is concerned with a target deployment at the Provider site. The model contains all of the necessary information to secure an assisted deployment:
- Deploying and sharing is guided from model properties and architecture definition. Discovery and geo location must be always available when asked for;
- Because a map of data is defined and updated, data discovery and geo-location is always available when asked for. This helps to check whether the infrastructure provider has persistence and to find out where outages happen;
- Deploying and sharing is guided from model properties and architecture definition.

**Archive & Change** – This phase is the heart of the life cycle. Evolutions, updates and possible deletion cannot be set without regular versions and snapshots comparison management:
- Updates have to be registered and archived at the customer site. Deletions and restores have to be guided by customer DB compare and approval;
- Backups have to be registered and archived at the customer site. Versioning on trunk and branches has to be supervised by the customer;
- Outage is covered by versions and changes archived based on partitioning. Content discovery assists in identifying and
auditing data to restore the service to previous versions. Rules for defined downtime determine timing and costs.

**Secure Delete** – Data has to be destroyed if necessary. The model contains all properties to conduct this activity that could be partial or final:

- According to information and properties defined in the *Create DB Model* state, procedures for removal must be available for effectively destroying data everywhere it is. Location constraints defined in the model identify persistent storage for retrieving and deletion;

In the following table we list DaaS requirements vs. DaaS life cycle properties and define DaaS directions supported in ERwin functionality for both Private and Public Cloud environments.

<table>
<thead>
<tr>
<th>Requirements</th>
<th>DaaS Life Cycle State</th>
<th>ERwin Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data location</td>
<td>Create DB Model</td>
<td>ERwin models contain partitioning properties and can include data location constraints. ERwin supports data labeling and classification. ERwin manages user tagging of data (a common Web 2.0 practice, through the use of user-defined properties (UDPs). ERwin can support compliant storage for sensitive data records.</td>
</tr>
<tr>
<td></td>
<td>Archive &amp; Change</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deploy &amp; Share</td>
<td></td>
</tr>
<tr>
<td>Data persistence</td>
<td>Create DB Model</td>
<td>For any partition, sub-model, or version of models, ERwin can label and trace data location. ERwin can define a map specifying where data is stored. Providers persistence can be registered. Data discovery can update partition properties to identify where data is located.</td>
</tr>
<tr>
<td></td>
<td>Archive &amp; Change</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deploy &amp; Share</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Secure Delete</td>
<td></td>
</tr>
<tr>
<td>Data discovery and navigation</td>
<td>Archive &amp; Change</td>
<td>Based on partitioning properties defined in ERwin, data and geo location can be verified or discovered.. This helps assure providers, data owners, and legal and regulatory authorities that data requested aligns with the locations defined.</td>
</tr>
<tr>
<td></td>
<td>Populate, Use and Test DB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deploy &amp; Share</td>
<td></td>
</tr>
<tr>
<td>Cloud to Cloud commingling</td>
<td>Create DB Model</td>
<td>ERwin contains information about possible commingling that can be authorized or not depending on the compensating controls the Provider supplies. ERwin can define and control special partitions for mixed data (classified/sensitive data) in commingling scenarios.</td>
</tr>
<tr>
<td></td>
<td>Deploy &amp; Share</td>
<td></td>
</tr>
<tr>
<td>Data backup</td>
<td>Create DB Model</td>
<td>ERwin guides data backup. Backup can be applied partially or globally and ERwin can direct data retrieval by partition, through model compare and versioning.</td>
</tr>
<tr>
<td></td>
<td>Archive &amp; Change</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deploy &amp; Share</td>
<td></td>
</tr>
<tr>
<td>Querying &amp; aggregation</td>
<td>Create DB Model</td>
<td>ERwin supplies functions to design and test query types. Specific query classes need to support heterogeneous database environments. Typical examples are aggregation queries and text pattern matching queries. ERwin supports query and aggregation testing to optimize database performance through multi-database structures.</td>
</tr>
<tr>
<td></td>
<td>Populate, Use and Test DB</td>
<td></td>
</tr>
<tr>
<td>Data inference</td>
<td>Create DB Model</td>
<td>ERwin supports inference and special data aggregation. All inferences and aggregations are defined, updated and tested into the model.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Create DB Model</td>
<td>ERwin guides rights assignment, access controls, rights management, and application data security starting from data model. ERwin sets logical and physical controls.</td>
</tr>
<tr>
<td></td>
<td>Populate, Use and Test DB</td>
<td></td>
</tr>
<tr>
<td>High availability</td>
<td>Deploy &amp; Share</td>
<td>ERwin model and partitioning configuration together with model changes and versions permits mastering of a recovery scheme and restoration when needed. ERwin is key for data inventory and discovery.</td>
</tr>
<tr>
<td></td>
<td>Archive &amp; Change</td>
<td></td>
</tr>
<tr>
<td>On-demand data secure deleting/shredding</td>
<td>Populate, Use and Test DB</td>
<td>ERwin creates a map for discovering data and ensuring that data requested can be retrieved quickly. This helps customers to effectively locate data in the Cloud and then, if necessary (due to closing contract, malware, low performance, etc.) Erasing/removing date wherever it is.</td>
</tr>
<tr>
<td>Near-zero configuration and administration</td>
<td>Create DB Model</td>
<td>ERwin models cover and contain all database properties including scripts, stored procedures, queries, partitions, changes and all configuration and administration properties. This means administrative actions decrease to leave more time for data design and update.</td>
</tr>
<tr>
<td></td>
<td>Generate/Update DB</td>
<td></td>
</tr>
<tr>
<td>High-performance</td>
<td>Populate, Use and Test DB</td>
<td>ERwin guides continuous improvement by following the data life cycle described above. Performance is managed through constant tests, deploying updates and governing model changes. ERwin checks persistence and can anticipate possible outage.</td>
</tr>
<tr>
<td></td>
<td>Deploy &amp; Share</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Archive &amp; Change</td>
<td></td>
</tr>
</tbody>
</table>
ERwin can be used by both Customers and Cloud Service Providers in both Private Cloud and Public Cloud environments. As outlined in Figure 2, the Relational Cloud deployment architecture is linked with the life cycle defined above. In this figure, we see that the Relational Cloud architecture aligns with the DaaS life cycle.

![Figure 2 – DaaS Life Cycle states and Relational Cloud coherence and continuity](image)

**MaaS: Introducing and Applying Cloud Best Practices**

*Best Practices throughout the Life Cycle*

Since a data model creates a map for database deployment, we can design life cycle state actions through Modeling ITIL® by UML (MIBU) to better explain the application of DaaS best practices. We introduce two examples concerning the 2 states *Create DB Model* and *Archive & Change*.
Best Practices for Moving to the Cloud using Data Models in the DaaS Life Cycle

Figure 3 – MIBU high level design to Create DB Model use case

The Create DB Model state provides guidance for designing and developing a Cloud database service. Modeling is also the first review before trying to change and improve the DaaS service capabilities. Archive & Change is concerned with developing and improving capabilities for new and modified database deployments. Archive & Change is strategic for database deletions and restores and for availability as well. Models allow service updates which are always guided by customer database compare and approval.

Figure 3 represents a MIBU high level use case design to Create DB Model state actions. As we can see, creating a model is the first and strategic step forward in DaaS. Here, we can understand how MaaS can help; existing canonical models inherit all properties which can be reused or modified for new services. In the DaaS Life Cycle, knowledge is continuously updated through service validation and testing in Populate, Use and Test DB state. Thus:

- a closed loop feedback runs to link Create DB Model operations;
- models achieve incremental improvements in service design and update;
- measuring and improving service increases model efficiency and effectiveness.

In Figure 3 MaaS is applied: the actor (Customer and/or Provider) looks for model knowledge in Service Operation capabilities and applies existing modules to create and integrate new models/services.

In the Archive & Change state persistence, restore, availability, deletion and change properties are encoded. Actions into the state aim to preserve models that have been deployed and, if necessary, help synchronize the Deploy & Share state. Models are provided to be compared before partial or total service deletion to the Secure and Delete state. The role of Archive & Change is essential; availability and restore depend upon the Archive & Change response and as a result, data loss prevention is managed by the parameters defined in the model.
This method of organizing the DaaS life cycle allows companies to build and control shared Cloud-ready data centers offering economies of scale and maintain a trusted environment and scaling security. This is the most practical way to check whether the Cloud infrastructure has persistence. Model mapping infrastructure controls persistent storage in which data can remain linked to specific computing instances. As we have already noted, models can include infrastructure and computer instances properties to check, when needed, where data is deployed and stored. The Archive & Change state highlights this advantage of model-driven DaaS; databases, and therefore, business services can be dynamically managed to SLAs and moved independently of the DBMS, machine or infrastructure. MaaS here plays a crucial role: the model containing infrastructure properties includes information to classify the on-premise database Cloud service. Mature and experienced Providers can therefore supply libraries containing models ready to be assembled and deployed following the best practices suggested by the above DaaS life cycle.

**DaaS Contract practice and main constraints**

The Cloud offers a great opportunity to manage highly available and scalable databases by decreasing cost, time and risks. We have introduced how the DaaS life cycle helps in applying best practices when migrating to the Cloud or administrating day-by-day Cloud activities.

Taking into consideration the risks associated with Cloud contracts, we introduce a set of best practices in this section that assist organizations in defining the best possible DaaS agreement. Best practices help define regulation controls that determine when and how applications can be deployed in the Cloud. This means that Cloud computing platforms are made up of different components from a variety of vendors but also of a variety of legal jurisdictions (countries, politics, risk management and compliance).

Applying the DaaS Life Cycle model can help manage data storage by using location constraints to check where your data is deployed and how it is implemented. Such constraints need to be clearly defined in the contract; persistence and dependencies have to be those classified (and regularly updated) in the model in order to standardize the platform technologies that underpin the service provided.
The main obligations that must be stipulated in the DaaS, SaaS contract are the following:

1. **Integrity defined at the model level has to be maintained through the service.** The monitoring executed by an ERwin model, for example, has to match what is defined into the initial data structure and classified in the same way;

2. **Country location has to be defined in the model partition and regularly monitored and compared.** Any mismatch is an infringement of the agreement and must be reconciled with the terms outlined in the SLA;

3. **Include and specify international regulations that the both Provider and the Vendor are responsible for during the service life cycle.** In detail, highlight directives containing data breach rules. Provider and Vendors are protected although any violation is a service penalty and the data owner must notify both Provider and Vendor in case of a breach;

4. **Specify location properties and not only in terms of country.** The site locating machines, racks and so on has to be the appropriate one (weight per square meter, fire safety, anti-flood, employee privileges and security service personnel);

5. **Identify trust boundaries throughout the IT architecture.** Data models and partitions are the right way to define trust boundaries and stewardship to prevent unauthorized access and sharing;

6. **Include the method to encrypt data transmitted across the network.** If different encryption is used by the provider/vendor, specify what and when it is to be used. The contract has to include how encryption is run on multi-tenant storage. List the rules concerning keys adoption;

7. **Once data has to be deleted, specify that data retention and deletion are the responsibility of the Provider.** Looking at data model mapping, data has to be destroyed in all locations defined and monitored. The Provider has to specify if data, for any reason, has been copied in different media and then shredded. The contract must include a provision for the customer to request an audit in order to certify that data has been deleted.;

8. **Models are key to ensuring that logical data segregation and control are effective after backup and recovery, test and compare are completed.** Include in the contract that a data model should be used to define the data architecture through the life cycle.

The best practices defined above are helpful guidelines in defining SaaS, DaaS contracts. Still, when designing a Cloud agreement, special measures should be taken into consideration to secure data and service in transit from/to the Provider:

- Enforce, if you can, and ensure security compliance through ISO 27001-2005 directions. Schedule vulnerability assessments and regular real-time visibility into data applications;
- Apply SSL, IPSec constraints to secure data movement into the data center. Perimeter protection is essential to prevent denial-of-service threats;
- Consider and include VLAN, VPN rules to secure data movement from/to the data center.

**Conclusion**

Companies are taking steps towards full-blown Cloud implementations by adopting DaaS and solutions that can enable Cloud services. The goal is keep costs down, reduce physical administration, grow availability and flexibility, and manage scalable Cloud databases. At the same time, however, organizations must pay special attention to risk assessment and mitigation. Organizations need best practices to be prepared for migrating to the Cloud. They have to adopt precautions to extend existing applications and leverage device density, while maintaining trusted environments by scaling security for pooled resources. The proposed solution looks at data models for DaaS using ERwin and the introduction of the DaaS Life Cycle. ERwin supports the key requirements of data services, incremental database deployment and progressive data structure provisioning, using a model-driven approach. DaaS also provides ERwin an opportunity to offer a unique utility-style model life cycle to accelerate Cloud database optimization and performance in migrating to the Cloud.
Glossary

**CapEx**: Capital Expenditure  
**DaaS**: Database as a Service  
**MaaS**: Model as a Service  
**MIBU**: Modeling ITIL by UML  
**OpEx**: Operational Expenditure  
**SaaS**: Software as a Service  
**SLA**: Service Level Agreement

Acknowledgements

I sincerely thank Donna Burbank for her precious feedback and encouragement on publishing the third paper of the series.

References

[1] N. Piscopo - ERwin® in the Cloud: How Data Modeling Supports Database as a Service (DaaS) Implementations  
http://erwin.com/whitepapers/detail/erwin_in_the_cloud_how_data_modeling_supports_database_as_a_service_daaS_im/

http://erwin.com/whitepapers/detail/ca_erwin_data_modelers_role_in_the_relational_cloud/


About the Author

Nuccio Piscopo has more than 20 years in the IT industry as a Data and Application change management technical executive, with a University Degree in Physics, post laurea from the International School of Physics “Enrico Fermi”, as well as a post laurea Master in Information Engineering (Computer Science). At present Nuccio is an independent consultant delivering SaaS, DaaS designing, startup, education and contracting consultancy, cloud audit risk assessment customer experience in migrating to the Cloud. In previous roles, Nuccio served at IR Semiconductors, Kraft, Bachmann/Cayenne, Platinum Technology and CA Technologies. Nuccio Piscopo can be contacted through http://it.linkedin.com/in/nucciopiscopo.