International Journal of Engineering & Scientific Research
Vol.14 Issue 02, February 2026
ISSN: 2347-6532 Impact Factor: 8.076
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Designing Data Products on a Lakehouse: A Data Mesh Approach

Amol Bhatnagar, Director Sogeti Capgemini USA.

Abstract
The convergence of lakehouse architectures and data mesh principles represents a fundamental shift in how organizations approach data management and analytics at scale. This paper explores the design and implementation of data products within a lakehouse infrastructure, utilizing data mesh as the operating model. We examine how domain-oriented ownership, coupled with the technical capabilities of modern lakehouse platforms, enables organizations to build scalable, discoverable, and self-service data products. Through detailed examples from the insurance domain, we demonstrate practical implementations of data contracts, service level agreements, and governance frameworks that support treating data as a product while maintaining the flexibility and cost-effectiveness of lakehouse storage.

1. Understanding Data Products: Definition and Examples

1.1 Defining the Data Product

A data product is a self-contained, purpose-built data asset that is designed, developed, and maintained with the same rigor and product thinking applied to customer-facing software products. Unlike traditional data artifacts such as reports or dashboards, data products encapsulate not just the data itself, but also the metadata, quality guarantees, access mechanisms, documentation, and governance policies necessary to make that data valuable and usable by its intended consumers.

The fundamental characteristics that distinguish a data product include product thinking applied to data assets, where each data product has a clear purpose, defined users, and measurable value metrics. Data products exhibit self-service accessibility, meaning consumers can discover, understand, and access the data without requiring intervention from the data product team. They maintain well-defined interfaces through published schemas, APIs, or query endpoints that abstract away implementation details. Quality and reliability guarantees are essential, with explicit service level objectives for freshness, accuracy, completeness, and availability. Clear ownership and accountability ensure a dedicated team maintains and evolves the product based on consumer needs. Finally, comprehensive documentation and metadata make the data product understandable and trustworthy to potential consumers.

1.2 The Data Product Spectrum

Data products exist along a spectrum of complexity and purpose. At the foundational level, source-aligned data products provide cleaned, standardized versions of operational system data. For example, a policy administration system in an insurance company might produce a "Policy Master" data product that cleanifies and enriches raw policy data with standardized codes, calculated fields, and audit trails. Aggregate data products combine data from multiple sources to provide integrated views. An insurance carrier might create a "Customer 360" data product that merges data from policy systems, claims systems, billing platforms, and customer service interactions to provide a complete view of each policyholder's relationship with the company.

Consumer-aligned data products are purpose-built for specific use cases or user communities. A "Claims Analytics" data product might be designed specifically for actuaries and include pre-calculated loss ratios, development triangles, and reserve adequacy metrics that would be cumbersome for analysts to compute repeatedly from raw claims data.

At the most sophisticated level, machine learning feature stores represent data products designed explicitly to support model training and inference, providing versioned, point-in-time correct features with lineage tracking and monitoring capabilities.

1.3 Concrete Examples from Practice

Consider an insurance company's "Auto Claims Loss Experience" data product. This product serves actuarial teams, underwriters, and product managers who need to understand loss patterns to inform pricing and risk selection decisions. The data product includes claim-level detail with calculated severity measures, aggregated summaries by various dimensions such as geography, vehicle type, and

driver characteristics, historical trending data showing how loss patterns evolve over time, and benchmark comparisons against industry standards and peer companies.

The product team publishes this data through multiple interfaces: a SQL-queryable table in the lakehouse for analysts comfortable with data manipulation, a curated set of dimensional models optimized for business intelligence tools, a REST API providing programmatic access for applications and automated processes, and pre-built dashboards and reports for common analytical patterns.

Quality guarantees are explicit and monitored. The data is refreshed daily with claims closed in the previous 24 hours, achieving 99.5 percent completeness for required fields, maintaining consistency with regulatory financial reporting within defined tolerances, and providing data lineage documentation tracing every metric back to source systems.

Another example is a "Policyholder Behavior Propensity" data product that serves marketing and customer retention teams. This product combines historical policy data, customer interaction history, external demographic and psychographic data, and predictive scores for various behaviors such as likelihood to renew, propensity to purchase additional coverage, and risk of lapse.

The product is updated weekly with new scores, includes confidence intervals and model performance metrics, maintains stable identifiers allowing tracking of individual customers over time, and documents model versions and feature importance to ensure transparency and trust.

1.4 The Value Proposition

Data products create value by reducing the friction in data consumption. Instead of each analytical team independently sourcing, cleaning, and transforming the same underlying data, they can leverage a well-maintained data product that encapsulates that work. This acceleration of insight generation is complemented by improved data quality and consistency, as centralized product ownership ensures systematic quality management rather than ad-hoc, inconsistent approaches across multiple consuming teams.

Data products enable better governance and compliance. When data flows through well-defined product interfaces, organizations gain visibility into who is using what data for which purposes, making it easier to enforce privacy requirements, audit access, and manage sensitive information appropriately.

Perhaps most importantly, data products align data capabilities with business value. By treating data as products with clear purposes and user communities, organizations ensure that data investments directly support business objectives rather than accumulating as undifferentiated technical assets of uncertain value.

2. Data Mesh Architecture and Insurance Domain Implementation

2.1 The Data Mesh Paradigm

Data mesh represents a sociotechnical approach to data architecture that applies product thinking and domain-driven design principles to analytical data management. Introduced by Zhamak Dehghani, data mesh addresses the scalability and organizational challenges that arise when centralized data platforms attempt to serve the diverse needs of large, complex enterprises.

The architecture rests on four foundational principles. Domain-oriented decentralized data ownership recognizes that the teams closest to data generation and consumption understand that data best. Rather than centralizing all data management in a single platform team, responsibility is distributed to domain teams who own their data products as first-class concerns alongside their operational applications.

Data as a product thinking requires domain teams to treat their data outputs with the same care and rigor they apply to customer-facing products, including user experience considerations, quality assurance, versioning, and lifecycle management.

Self-service data infrastructure provides domain teams with the platforms, tools, and capabilities they need to build, deploy, and operate their data products independently without creating bottlenecks through centralized teams. This infrastructure includes common frameworks for data storage, processing, cataloging, access control, and monitoring.

Federated computational governance balances autonomy with coherence by establishing global standards and policies that all data products must satisfy while allowing domain teams flexibility in how they implement those standards within their products.

2.2 Mapping Data Mesh to Insurance Domain Structure

Insurance companies possess natural domain boundaries that align well with data mesh principles. A typical property and casualty insurer might organize around several core domains, each becoming a center of data product ownership and development.

The underwriting and risk selection domain owns data products related to risk assessment, pricing, and policy issuance. This includes products such as risk scoring models, underwriting guidelines and eligibility rules, pricing algorithms and rate tables, and application and quote data with quality enrichments.

The policy administration domain manages the lifecycle of insurance policies and owns products including policy master data with standardized attributes, endorsement and renewal history, coverage and limits detail, and billing and payment schedules.

The claims management domain handles loss events and produces data products such as claims detail with calculated severity and development, medical and repair cost benchmarks, fraud detection signals and investigations, and litigation and subrogation tracking.

The customer and distribution domain focuses on policyholder relationships and distribution channels, creating products like customer master data with household relationships, agent and broker performance metrics, marketing campaign response and attribution, and customer interaction history across all touchpoints.

Supporting domains such as finance and actuarial own data products for financial reporting, regulatory compliance, reserve calculations, and loss development projections.

2.3 Insurance Data Mesh Implementation Example

Consider how a mid-sized regional auto and homeowners insurer might implement data mesh principles. The company has struggled with a centralized data warehouse that has become a bottleneck, with the small data engineering team unable to keep pace with requests from underwriting, actuarial, claims, and marketing departments. Data quality issues persist because the central team lacks deep domain expertise, and business users have begun creating shadow IT solutions by extracting data to spreadsheets and personal databases.

The organization embarks on a data mesh transformation by first identifying domain boundaries aligned with organizational structure and business capabilities. They establish the underwriting domain, claims domain, policy administration domain, customer and marketing domain, and finance and actuarial domain.

For each domain, they designate data product owners who have both business context and technical accountability. In the claims domain, for instance, the claims analytics manager becomes the data product owner, supported by data engineers embedded within the claims organization rather than isolated in a central IT group.

The platform team shifts from trying to build all data products centrally to providing self-service infrastructure. They deploy a lakehouse platform based on Delta Lake and Databricks that provides medallion architecture patterns with bronze, silver, and gold layers, data cataloging through Unity Catalog for discovery and governance, automated data quality frameworks using Great Expectations, orchestration capabilities through Apache Airflow, and access control and row-level security capabilities.

The claims domain begins by defining their foundational data product: the "Claims Master" product. This product takes raw claims data from the claims management system and produces a cleaned, enriched dataset that includes standardized claim types and cause-of-loss categories mapped to industry taxonomies, calculated fields such as claim duration, time to first payment, and current reserve adequacy, linkages to policy data to provide coverage and deductible context at time of loss, and geocoded loss locations with enriched property characteristics for property claims.

The product publishes a data contract specifying the schema with required and optional fields and their data types, freshness guarantees with daily updates by 6 AM Eastern time, quality thresholds requiring 99 percent completeness for core fields and zero tolerance for invalid policy references, and breaking change policies with 90-day notice for schema changes that would break downstream consumers.

The actuarial domain consumes this Claims Master product to build their own "Loss Development" data product, which the central team previously would have attempted to create. The actuarial team applies their deep expertise in loss reserving methodologies to create a product specifically designed

for reserve analysis and pricing studies, incorporating age-to-age development factors, case versus paid loss patterns, and large loss emergence tracking.

The marketing domain builds a "Customer Lifetime Value" data product by combining the Policy Master product from policy administration, Claims Master product from claims, and their own customer interaction data. This cross-domain composition demonstrates how the mesh architecture enables building sophisticated analytical products by assembling well-defined components.

2.4 Organizational and Cultural Transformation

Implementing data mesh requires more than architectural changes. The insurance company must evolve team structures, creating embedded data engineering roles within domain teams rather than maintaining strict separation between business and technology groups. Product management disciplines are applied to data, with product managers or product owners for significant data products who maintain backlogs, prioritize enhancements based on consumer needs, and measure product adoption and value.

Incentives and metrics shift from measuring the central data team's productivity to measuring business outcomes enabled by data products, such as time from question to insight, data-driven decision velocity, and reduction in manual data preparation effort across the organization.

Communication and collaboration patterns evolve as domain teams share best practices, reusable components, and lessons learned through communities of practice. Regular data product showcases allow teams to demonstrate their products and inspire cross-pollination of ideas.

The transformation takes time, often 18 to 24 months to reach meaningful maturity, but the insurance company begins seeing benefits within the first few months as domain teams gain autonomy and accelerate delivery of analytical capabilities that directly serve their business priorities.

3. Applying Data Mesh Principles on Lakehouse Platforms

3.1 The Lakehouse Foundation

Lakehouse architectures merge the flexibility and cost-effectiveness of data lakes with the management capabilities and performance characteristics traditionally associated with data warehouses. By storing data in open formats on object storage while providing ACID transactions, schema enforcement, and query optimization, lakehouses eliminate the need to maintain separate systems for different analytical workloads.

For data mesh implementations, lakehouses provide several critical capabilities. Open storage formats such as Delta Lake, Apache Iceberg, and Apache Hudi enable data products to be accessed by diverse consumption tools without lock-in to proprietary systems. This openness is essential for the self-service principle, as different domains may prefer different analytical tools.

Unified governance through catalogs like Unity Catalog, AWS Glue Catalog, or Apache Atlas provides centralized discovery while maintaining decentralized ownership. Domains can register their data products in the catalog with rich metadata, making them discoverable across the organization while retaining full control over the products themselves.

Separation of storage and compute allows domains to provision the processing resources they need independently without interfering with other domains' workloads. This isolation is crucial for the autonomy that data mesh requires.

Schema evolution and versioning capabilities inherent in lakehouse table formats support the product lifecycle management necessary when data products evolve over time while maintaining backward compatibility for existing consumers.

3.2 Implementing Domain Boundaries in the Lakehouse

Physical organization of the lakehouse should reflect domain ownership while enabling cross-domain data discovery and consumption. A well-designed lakehouse for data mesh employs several organizational strategies.

Storage namespacing uses cloud storage bucket hierarchies and lakehouse database schemas to create clear domain boundaries. An insurance lakehouse might organize storage as s3://company-lakehouse/domains/claims/, s3://company-lakehouse/domains/underwriting/, s3://company-lakehouse/domains/policy_admin/, with each domain having full control over their namespace while following common conventions.

Within each domain space, the medallion architecture pattern applies, with bronze, silver, and gold layers representing increasing levels of refinement and business-oriented structure. The claims domain stores raw claims system extracts in claims.bronze.raw_claims, applies cleansing and standardization

to produce claims.silver.clean_claims, and creates business-oriented products in claims.gold.claims_master and claims.gold.loss_development.

Access control policies are defined at the domain level, with domain teams managing permissions for their data products. The platform provides the mechanisms through row-level security, column-level security, and dynamic views, while domains configure specific policies. The claims domain might grant the actuarial domain read access to claims.gold.claims_master while restricting access to personally identifiable information through column masking.

3.3 Self-Service Infrastructure Capabilities

The platform team provides shared infrastructure that domain teams use to build their data products independently. This infrastructure includes ingestion frameworks offering connectors to common source systems, change data capture capabilities for real-time or near-real-time data movement, and orchestration templates for scheduled data loads.

Data quality frameworks provide reusable validation rules, automated quality testing integrated into data pipelines, and quality score calculation and tracking. Domains configure quality rules specific to their products while leveraging common testing infrastructure.

Transformation and modeling tools include SQL and DataFrame APIs for data manipulation, support for declarative modeling tools such as dbt for maintaining transformation logic as code, and lineage tracking automatically capturing upstream dependencies and downstream usage.

Publishing and consumption interfaces provide SQL query access through compute engines like Databricks SQL or Amazon Athena, REST APIs generated automatically from data product schemas, and streaming access through technologies like Apache Kafka for real-time consumption needs.

Observability and monitoring capabilities track data freshness with automated staleness detection, monitor quality metrics and alerting on threshold violations, track usage patterns to understand who is consuming which data products, and measure performance characteristics such as query latency and throughput.

3.4 Federated Governance Implementation

Computational governance balances global standards with domain autonomy. The governance approach operates at multiple levels.

Global policies are established centrally and enforced through platform mechanisms. These might include data classification standards defining sensitivity levels such as public, internal, confidential, and restricted, retention policies specifying how long different classes of data must be maintained, privacy requirements such as GDPR or CCPA compliance controls, and security baselines including encryption requirements and access logging.

Domain-level policies are defined by domain data product owners within the constraints of global standards. The claims domain might define that all claims data is classified as confidential, require multi-factor authentication for accessing detailed claim notes, and implement specific retention schedules based on statute of limitations considerations for different claim types.

Product-level specifications are documented for each data product in the catalog. The claims_master data product specifies exactly who the data steward is, what the intended use cases are, what the access request process entails, what the SLAs are, and what the deprecation policy is.

The platform enforces compliance through automated policy enforcement. When a domain team publishes a data product, the platform validates that appropriate classification tags are applied, required metadata is complete, access controls align with classification levels, and quality thresholds are defined.

3.5 Cross-Domain Collaboration Patterns

While domains own their data products independently, many valuable analytical use cases require combining data across domains. The lakehouse enables several collaboration patterns.

Data product composition allows consumer domains to join and combine data products from multiple source domains. The marketing domain creates customer_lifetime_value by reading from policy_admin.gold.policy_master, claims.gold.claims_master, and their own customer_interactions product. Each source product maintains its independent lifecycle while the consuming product manages the integration logic.

Shared dimension management addresses common entities like customer, geography, and time. Rather than each domain creating duplicate dimension tables, a dedicated master data domain might own authoritative customer_master and geography_dimension products that other domains reference. This

shared ownership model requires careful governance to ensure the shared dimensions meet all consuming domains' needs.

Federated query capabilities in modern lakehouse platforms allow joining across domain-owned datasets without physically copying data. The underwriting domain can query claims.gold.claims_master directly when evaluating an applicant's prior loss history without the claims domain needing to push data into an underwriting-specific dataset.

Event-driven data products support real-time use cases by publishing domain events to shared event streaming platforms. When a new claim is filed, the claims domain publishes a claim_filed event that the customer service domain consumes to update customer interaction history and the finance domain consumes to update financial projections.

3.6 Platform Evolution and Domain Enablement

The platform team's role transforms from building data products to enabling domain teams to build their own products. This requires investment in developer experience through comprehensive documentation, example patterns and reference implementations, reusable code libraries and templates, and self-service provisioning portals for domain teams to request infrastructure resources.

The platform team provides consultation and support, including office hours and direct assistance, architecture reviews for complex data products, performance optimization guidance, and training programs on lakehouse capabilities and best practices.

Continuous improvement happens through feedback loops where the platform team regularly surveys domain teams about pain points and missing capabilities, monitors platform usage patterns to identify opportunities for optimization, and shares innovations where one domain develops a useful pattern that could benefit others.

This symbiotic relationship between the platform team and domain teams creates a virtuous cycle. As the platform becomes more capable and easier to use, domain teams can build more sophisticated data products more quickly. As domain teams push the boundaries of what's possible, they surface requirements that drive platform evolution.

4. Domain-Oriented Data Models for the Data Mesh

4.1 The Shift from Enterprise Models to Domain Models

Traditional enterprise data warehouse approaches pursued comprehensive enterprise data models that attempted to create a single unified representation of all organizational data. While intellectually appealing, these efforts often failed because they couldn't keep pace with business change, became bottlenecks as every change required central approval, and resulted in overly generic models that served no use case particularly well.

Data mesh embraces domain-oriented modeling, recognizing that different domains have fundamentally different perspectives on shared concepts. A customer in the underwriting domain is an applicant to be risk-assessed. In the policy administration domain, that same customer is a policyholder with coverage elections and payment obligations. In the claims domain, they become a claimant with loss experiences. In the marketing domain, they represent a set of preferences, behaviors, and lifetime value projections.

Rather than forcing these perspectives into a single model, domain-oriented modeling allows each domain to model data according to their specific needs and mental models while establishing clear interfaces and translation points where domains interact.

4.2 Bounded Contexts and Ubiquitous Language

Drawing from domain-driven design, data mesh applies the concept of bounded contexts to data modeling. A bounded context defines the boundary within which a particular domain model applies and a ubiquitous language is consistently used.

In our insurance example, the claims domain's bounded context includes concepts such as claim, which represents a request for payment under policy coverage, claimant, the party making the claim, loss_event, the incident that triggered the claim, reserve, the estimated amount that will ultimately be paid, and severity, the actual or projected cost of the claim.

These terms have precise meanings within the claims context that may differ from how other domains think about similar concepts. The finance domain might view a claim primarily as a liability on the balance sheet, while the underwriting domain sees it as a loss experience data point for risk assessment.

Within its bounded context, the claims domain models data using terminology and structures that make sense to claims professionals. The claims_master data product might include fields such as claim_status with domain-specific values like open_pending_investigation, open_in_litigation, closed_with_payment, subrogation_recovery, and date_of_loss, loss_notification_date, first_payment_date representing the claim lifecycle.

4.3 Domain Model Architecture Patterns for Data Products

Effective domain data models in a lakehouse follow architectural patterns that balance expressiveness with usability.

The source-aligned layer maintains close fidelity to operational systems, primarily focused on capturing data as it exists in transactional systems with minimal transformation. The claims domain's bronze layer contains raw tables mirroring the claims management system's structure, preserving all system-specific codes and identifiers.

The domain-cleansed layer applies domain knowledge to standardize, enrich, and correct data within the domain's bounded context. The claims domain's silver layer maps system-specific claim type codes to standardized taxonomies, calculates derived fields meaningful to claims professionals such as time_from_loss_to_notification and current_reserve_adequacy, geocodes loss locations and enriches them with weather data, property characteristics, and other contextual information, and resolves duplicates and corrects known data quality issues.

The domain product layer organizes data for consumption by specific user communities and use cases. The claims domain publishes multiple gold layer products, including claims_master as a comprehensive claim-level dataset for general analytical use, loss_development as claim triangles and development patterns for actuarial analysis, large_loss_tracking for enterprise risk management and catastrophe planning, and fraud_investigation_queue with prioritized claims exhibiting fraud indicators.

4.4 Modeling for Cross-Domain Consistency in the Data Mesh

While each domain models data according to its needs, cross-domain interoperability requires coordination around shared concepts. Several strategies enable this without sacrificing domain autonomy.

Conforming dimensions represent shared business entities modeled consistently across domains. All domains agree on a common customer identifier, basic customer attributes like name and address, and standard geographic hierarchies. Each domain can extend these conforming dimensions with domain-specific attributes, but the common core enables joining datasets across domains.

Anti-corruption layers translate between domain models at the boundaries. When the underwriting domain consumes the claims_master product, it might apply an anti-corruption layer that maps claims terminology into underwriting concepts. What the claims domain calls severity becomes loss_amount in underwriting's model. Claim_type values are mapped to underwriting_relevant_loss_categories.

Canonical events provide a neutral representation of business occurrences that cross domain boundaries. When a policy is issued, a policy_issued event is published with a schema that both the policy administration domain and the downstream consuming domains agree upon, even if their internal models differ.

Reference data management addresses code sets and classifications used across domains. The platform provides shared reference datasets for state_codes, country_codes, and industry_standard_loss_causes that all domains use rather than creating incompatible domain-specific versions.

4.5 Example of Domain Modeling in Practice: Insurance Claims

Consider the detailed modeling approach for the claims domain's primary data product. The logical model organizes information around core entities and their relationships.

The claim entity serves as the root aggregate with attributes including claim_id as the unique identifier, policy_id linking to the policy under which the claim is made, date_of_loss, loss_notification_date, claim_status, and claim_type with standardized values such as auto_collision, auto_comprehensive, home_fire, home_water_damage.

Related entities include claimant with party_id, role in values such as insured, third_party, passenger, injury_details, and contact_information. Coverage with coverage_type such as bodily_injury, property_damage, medical_payments, limits_and_deductibles, and coverage_applicability_determination. Payment with payment_id, payment_date, payment_amount,

payee, and payment_type as indemnity, expense, deductible_recovery. Reserve with reserve_type such as case_reserve, ibnr_reserve, reserve_amount, reserve_date, and reserve_basis_explanation.
The physical implementation in the lakehouse uses Delta Lake tables with the model translated into an efficient storage structure. Partitioning strategies optimize query performance by partitioning claims_master by claim_close_year and claim_type, allowing queries filtered by these dimensions to scan minimal data.
Clustering and Z-ordering within partitions further optimize access patterns. The table is Z-ordered by policy_id to accelerate queries joining claims to policies.
Schema evolution is managed through Delta Lake's schema evolution capabilities, allowing backward-compatible changes like adding new columns without breaking existing consumers. Breaking changes are handled through versioning, with claims_master_v2 introduced alongside claims_master_v1 during a transition period.
Metadata and documentation are comprehensive, with every field including business definitions understood by claims professionals, data lineage showing source systems and transformations, sample values and acceptable ranges, and quality rules and validation logic.

5. Advantages of Data Products in a Mesh: Discoverability and Self-Service Analytics

5.1 The Discoverability Challenge

In traditional centralized data platforms, discoverability is a persistent challenge. Analysts don't know what data exists, where to find it, whether it's trustworthy, or who to ask for access. This leads to repeated requests to data teams, duplicated effort as analysts create their own extracts and transformations, underutilization of existing capabilities, and slow time-to-insight as analysts spend more time searching for data than analyzing it.
Data mesh with well-designed data products transforms discoverability by making data products first-class citizens with rich metadata, clear ownership, and self-service access.

5.2 Data Product Catalogs and Metadata Management

The foundation of discoverability is a comprehensive data catalog that serves as a marketplace for data products. Modern lakehouse platforms provide catalog capabilities that domain teams use to publish their products.
Catalog entries for each data product include business metadata with product name, description, and purpose in plain language understandable to non-technical users. A business glossary defines domain-specific terminology used in the product. Use case examples demonstrate how the product has been applied to solve business problems. Owner information provides contact details for the product team.
Technical metadata specifies schema details with field names, types, and descriptions, storage location and access methods, refresh frequency and SLA commitments, data lineage showing source systems and transformation logic, and quality metrics and validation rules.
Operational metadata tracks usage statistics showing query volumes and active users, performance characteristics such as typical query response times, incident history documenting outages and issues, and version history showing product evolution over time.
Consumer reviews and ratings allow users to share experiences and provide feedback, creating transparency about product quality and usability similar to consumer product reviews.

5.3 Search and Discovery Capabilities

Effective catalog implementations provide multiple discovery paths to accommodate different user needs and search patterns.
Keyword search allows users to search across product names, descriptions, field names, and documentation. An underwriting analyst searching for "prior claims" would discover the claims_master product, loss_development product, and customer_loss_history product, each with descriptions explaining their different purposes.
Faceted browsing enables filtering by domain such as claims, underwriting, policy administration, data classification such as public, internal, confidential, update frequency such as real-time, daily, monthly, and consumer rating with minimum quality thresholds.
Lineage-based discovery allows users to explore upstream and downstream relationships. An analyst using the loss_development product can navigate upstream to discover it derives from claims_master, which in turn sources from the claims management system. They can navigate downstream to see what other products or reports depend on loss_development.

Tag-based organization enables flexible categorization beyond rigid hierarchies. Products might be tagged with subject areas such as claims, reserving, catastrophe, use cases like fraud detection, pricing, reporting, data domains such as customer, policy, claim, payment, and compliance requirements like PII, GDPR, SOX.

5.4 Insurance Domain Example: Underwriter Discovery Journey

Consider a commercial underwriter joining the insurance company who needs to analyze loss experience for manufacturing risks in the Midwest. Their discovery journey illustrates the value of well-cataloged data products.

The underwriter begins by searching the data catalog for "manufacturing losses". The search returns several relevant data products. The claims_master product appears with a description explaining it provides comprehensive claim detail for all lines of business with daily refresh. The industry_loss_benchmarks product offers external benchmark data for various industry classifications. The commercial_underwriting_analytics product provides pre-built analytical views specifically for commercial lines underwriters.

Exploring the commercial_underwriting_analytics product, the underwriter finds it includes loss ratios by industry and geography, pricing adequacy indicators, competitive market position analysis, and trending of key risk indicators. The documentation explains this product is specifically designed for underwriting use cases and includes sample queries for common analyses.

Checking the lineage, the underwriter sees this product combines claims_master for internal loss data, policy_master for exposure and premium information, and industry_benchmarks for external comparisons. Reviews from other underwriters rate it highly for ease of use and relevance to underwriting decisions.

The product's access instructions provide a one-click request process. The underwriter submits a request explaining their role and use case. Within minutes, an automated approval workflow grants access based on their job role and data classification policies, and the underwriter begins querying the data through their preferred business intelligence tool.

What might have taken days or weeks in a traditional environment—finding the right data, requesting access, waiting for data team assistance, learning the data structure—happens in under an hour through self-service discovery and access.

5.5 Self-Service Analytics Capabilities

Discoverability alone is insufficient; users must also be able to independently access and analyze data products without constant intervention from data teams.

Direct query access through SQL engines like Databricks SQL or Athena allows analysts to query data products using familiar SQL syntax. The lakehouse's query optimization ensures good performance without requiring analysts to understand physical storage details.

Business intelligence tool integration enables analysts to connect Excel, Tableau, Power BI, or other BI tools directly to data products. Pre-built semantic models and dimensional structures make it easy to create visualizations and reports without deep technical knowledge.

Programmatic access through REST APIs allows applications and data scientists to consume data products programmatically. The actuarial team builds a pricing model that calls the claims_master API to retrieve relevant loss history for each underwriting submission automatically.

Notebook environments such as Jupyter or Databricks notebooks provide exploratory analysis capabilities for more sophisticated users. Data scientists access data products through DataFrames APIs, applying machine learning algorithms and statistical analyses.

Export capabilities allow users to extract datasets for use in specialized tools. An actuary might export triangulated loss development data to actuarial modeling software that doesn't support direct database connectivity.

5.6 Governance Enabling Self-Service

Self-service doesn't mean ungoverned access. The lakehouse platform enforces policies that enable broad access while protecting sensitive data and maintaining compliance.

Role-based access control automatically grants permissions based on job roles. All underwriters get read access to underwriting-relevant products. Actuaries access reserving and pricing products. Senior leadership accesses executive dashboards. Access is provisioned automatically based on HR system integration rather than requiring manual approval for each request.

Attribute-based access control provides finer-grained policies. A claims adjuster sees only claims for their assigned geographic territory. A product manager sees data only for their product lines. Row-level security filters data based on user attributes without requiring separate datasets for each user population.

Column-level security and masking protect sensitive fields. Most users see claims_master with personally identifiable information masked, showing only that a claimant exists without revealing their name or contact details. Investigators with proper authorization see unmasked data. This allows broad access to analytical data while protecting privacy.

Dynamic data masking applies different policies based on context. A data scientist building fraud detection models sees synthetic data for development and testing but accesses real data only in controlled production environments with full audit logging.

Usage monitoring and audit logs track all access to data products, supporting compliance requirements and detecting anomalous usage patterns. When a user suddenly exports large volumes of data outside normal patterns, security teams receive alerts to investigate potential data exfiltration.

5.7 Insurance Domain Example: Accelerated Claims Analytics

An insurance company's claims analytics team exemplifies the transformation enabled by data mesh with strong discoverability and self-service.

Previously, the team spent 60 percent of their time on data acquisition and preparation. Each analytical project began with requests to IT for data extracts, weeks of waiting while the request was prioritized, rounds of clarification about exactly what data was needed, manual reconciliation when extracted data didn't match expectations, and custom transformation logic to prepare data for analysis.

After implementing data mesh on the lakehouse, the team's workflow transforms dramatically. They begin new projects by searching the data catalog for relevant products, immediately finding the claims_master and loss_development products with comprehensive documentation. They explore sample queries and documentation to understand what's available. They request access through self-service workflows, receiving approval within minutes based on their role.

They directly query the data products using SQL or their BI tools without waiting for IT. They discover additional products through lineage exploration, such as the weather_events product that helps explain catastrophe losses. They combine multiple products to create new analytical views without needing to understand ETL logic.

The team now spends 80 percent of their time on actual analysis rather than data preparation. Time from question to initial insights drops from weeks to hours. The number of analytical projects completed increases by a factor of three. Crucially, the team can respond to urgent business questions without waiting for data team availability.

When a severe hailstorm affects multiple states, executive leadership asks for loss estimates within hours. The analytics team immediately queries claims_master filtered by date and geocoded loss locations, joins to policy_master to identify exposed policies not yet reported, and references historical_catastrophe_patterns to project ultimate losses. They deliver preliminary estimates within two hours rather than the days required under the old model.

5.8 Enabling Innovation Through Accessibility

When data becomes discoverable and accessible through self-service, innovation accelerates. Teams can explore ideas without significant upfront investment or coordination overhead.

A claims process improvement team hypothesizes that certain claim characteristics predict long settlement times. They search the catalog and find claims_master includes all the data elements they need. They quickly prototype an analysis, identify key predictors, and propose process changes. The entire cycle from hypothesis to recommendation takes two weeks rather than three months.

A product development team wants to design a new insurance product targeting a specific customer segment. They discover the customer_analytics and loss_experience products provide exactly the data needed to assess market opportunity and price the coverage appropriately. They build a business case with confidence in data-driven projections.

A data scientist exploring machine learning approaches to fraud detection discovers pre-built feature sets in the fraud_indicators product, accelerating model development. They also find the model_training_data product that provides properly labeled examples of known fraud cases, eliminating months of work to curate training data.

This democratization of data access doesn't mean chaos. The governance frameworks ensure access is appropriate, usage is audited, and sensitive data remains protected. But within those guardrails, teams have unprecedented freedom to explore, innovate, and create value from data.

Conclusion

The combination of lakehouse infrastructure and data mesh operating principles represents a powerful paradigm for analytical data management at scale. By organizing data products around domains, treating data with product thinking, providing self-service infrastructure, and implementing federated governance, organizations can overcome the scalability limitations of centralized data platforms while avoiding the chaos of completely decentralized approaches.

The lakehouse provides the technical foundation with open storage formats, ACID transactions, unified governance, and separation of storage and compute that enable independent domains to build and maintain their data products. Data mesh provides the organizational model that aligns data capabilities with business domain expertise and decision-making.

Through concrete implementation patterns including domain-oriented data models bounded by context but interconnected where necessary, comprehensive data contracts and SLAs that formalize product commitments, clear ownership and accountability for data product quality and evolution, reusable assets that eliminate redundant effort and ensure consistency, and rich metadata enabling discoverability and self-service access, organizations create analytical capabilities that scale with business complexity rather than becoming bottlenecks.

The insurance industry examples throughout this paper demonstrate practical applications of these principles. From claims analytics to underwriting decision support, from actuarial reserving to fraud detection, data mesh on lakehouse enables insurance companies to leverage data more effectively, respond faster to market changes, and make better risk-based decisions.

As organizations continue evolving their data architectures, the lakehouse-plus-mesh approach offers a roadmap for building analytical capabilities that are scalable, governed, and aligned with business value creation. The journey requires both technical implementation and organizational transformation, but the benefits in agility, quality, and business impact make it a compelling direction for data-driven enterprises.

As organizations continue evolving their data architectures, the lakehouse-plus-mesh approach offers a roadmap for building analytical capabilities that are scalable, governed, and aligned with business value creation. The journey requires both technical implementation and organizational transformation, but the benefits in agility, quality, and business impact make it a compelling direction for data-driven enterprises.

References

Armbrust, M., Ghodsi, A., Xin, R., & Zaharia, M. (2021). Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. *Proceedings of CIDR 2021*. Available at: http://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

Databricks. (2023). *The Databricks Lakehouse Platform*. Databricks Documentation. Retrieved from https://docs.databricks.com/lakehouse/

Dehghani, Z. (2019). How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh. *Martin Fowler's Blog*. Retrieved from https://martinfowler.com/articles/data-monolith-to-mesh.html

Dehghani, Z. (2020). Data Mesh Principles and Logical Architecture. *Martin Fowler's Blog*. Retrieved from https://martinfowler.com/articles/data-mesh-principles.html

Dehghani, Z. (2022). *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media.

Delta Lake. (2024). *Delta Lake Documentation: ACID Transactions and Schema Evolution*. Linux Foundation. Retrieved from https://docs.delta.io/

Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional.

Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., & Stoica, I. (2011). Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. *Proceedings of NSDI 2011*.

Hewitt, E. (2020). *Technology Strategy Patterns: Architecture as Strategy*. O'Reilly Media.

Inmon, W. H. (2005). *Building the Data Warehouse* (4th ed.). Wiley.

Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). Wiley.
Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.
Machado, D., Gomes, A., & Melo, P. (2022). Data Mesh: Concepts and Principles of a Paradigm Shift in Data Architecture. *Applied Sciences*, 12(21), 11326. https://doi.org/10.3390/app122111326
Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems* (2nd ed.). O'Reilly Media.
Reis, J., & Housley, M. (2022). *Fundamentals of Data Engineering: Plan and Build Robust Data Systems*. O'Reilly Media.
Sadalage, P. J., & Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional.
Sawadsky, N., & Khazaei, H. (2022). Data Mesh: A Systematic Gray Literature Review. *arXiv preprint* arXiv:2304.01062.
Tstatsoulis, C., & Mehta, A. (2023). Implementing Data Mesh in Financial Services: Challenges and Solutions. *Journal of Financial Data Science*, 5(2), 45-62.
Unity Catalog. (2024). *Unity Catalog: Unified Governance for Data and AI*. Databricks Documentation. Retrieved from https://docs.databricks.com/data-governance/unity-catalog/
Vernon, V. (2013). *Implementing Domain-Driven Design*. Addison-Wesley Professional.
Wiegers, K., & Beatty, J. (2013). *Software Requirements* (3rd ed.). Microsoft Press.
Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Stoica, I. (2016). Apache Spark: A Unified Engine for Big Data Processing. *Communications of the ACM*, 59(11), 56-65.