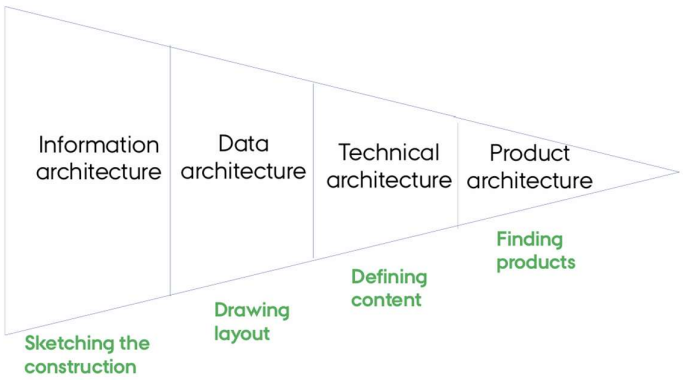
Data Architecture

THE BI ARCHITECTUAL TIERS



What Is Data Architecture? (Simplified Explanation)

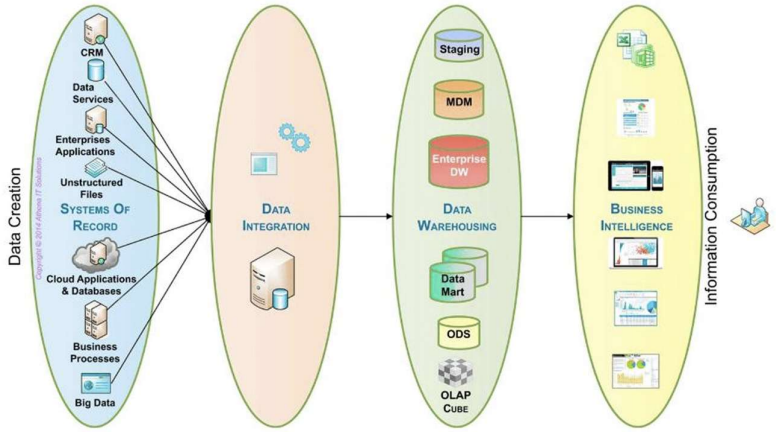
- Data architecture is like the master blueprint for how an organization handles its data.
  - It answers questions such as:
  - Where does the data come from? (collection)
  - Where is it kept? (storage)
  - How is it cleaned or transformed? (processing)
  - Who can use it, and how? (sharing & governance)
  - Who is responsible for what? (roles & accountability)
- Why It Matters
  - A strong data architecture:
    - Ensures data supports the business strategy
    - Data isn't just collected randomly. The architecture makes sure data helps the organization achieve goals like improving efficiency, customer experience, or decision-making.
    - Follows FAIR principles

      These are standards that guide how data should be managed:

FAIR Principle	Meaning
Findable	Easy to locate with proper metadata
Accessible	People/systems can retrieve it when needed
Interoperable	Works across tools and platforms
Reusable	Clean, well-documented, usable for future tasks

- In short:
  - Data architecture = the structured plan that ensures data flows smoothly, safely, and meaningfully across the organization.

Data architecture workflow



The diagram shows the end-to-end journey of data inside an organization, from where it is created to how it is consumed for insights.
It has four major stages:

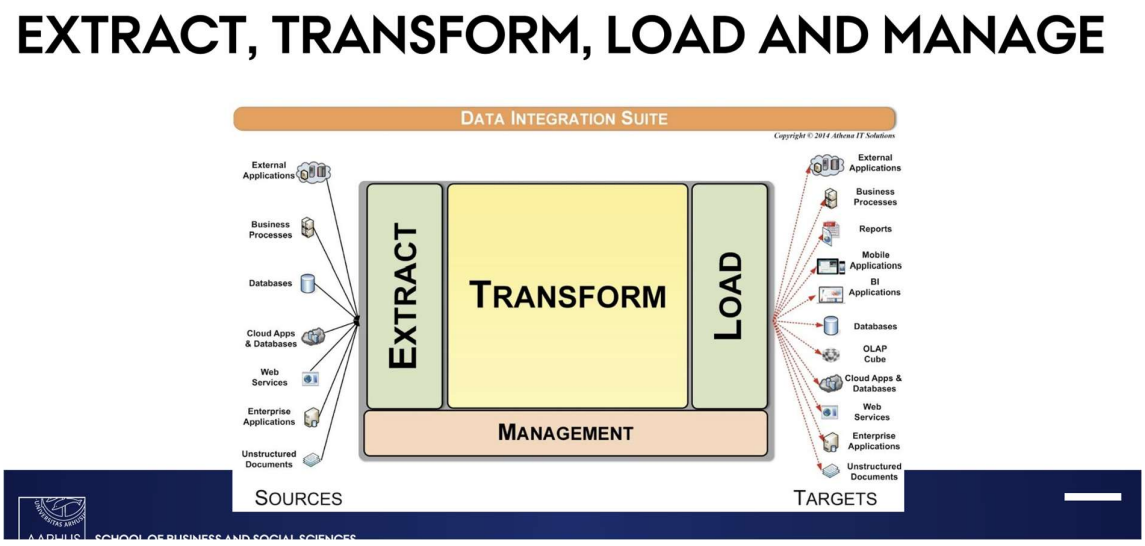
- Data Creation (Systems of Record)
    - This is where raw data is originally generated.
    - Examples shown in the diagram:
        - CRM systems (customer data)
        - Data services
        - Enterprise applications (ERP, HR systems, etc.)
        - Unstructured files (documents, emails, logs)
        - Cloud applications & databases
        - Business processes
        - Big data sources (IoT, social media, sensors)
    - *These are the sources that feed all downstream analytics.*
- Data Integration
    - This stage takes data from all systems of record and combines, cleans, and prepares it.
    - Typical processes here:
        - ETL/ELT (Extract, Transform, Load)
        - Data pipelines
        - Data quality checks
        - Metadata management
    - Goal: move raw data into a structured, consistent format.
- Data Warehousing
    - This is where cleaned data is stored and organized for analysis.
    - Parts shown:
        - Staging Temporary space for raw incoming data
        - MDM (Master Data Management), Ensures consistency of key data (customers, products)
        - Enterprise Data Warehouse (EDW), Centralized, integrated storage
        - Data Marts Smaller, department-focused subsets (e.g., sales, finance)
        - ODS (Operational Data Store), Holds near-real-time operational data
        - OLAP Cube, Multidimensional structure used for fast analysis
    - Goal: provide reliable, consolidated data for analytics.
- Business Intelligence (BI) — Information Consumption
    - This is the final stage where end-users consume insights.

- Examples shown:
  - Dashboards
  - Reports
  - Visualizations
  - Self-service analytics tools
  - Mobile BI
- Goal: help business teams make decisions based on accurate data.

Overall Workflow Flow

Data Creation → Data Integration → Data Warehousing → Business Intelligence

This is the full pipeline, showing how data transforms from raw input to actionable business insights.

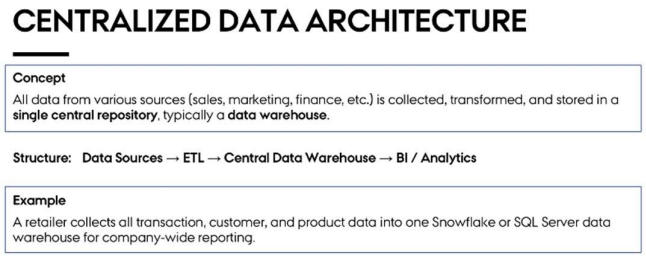


Modern Data Architecture Models

Cloud-era reinterpretations — 2015–present

Model	Core Idea
Centralized Warehouse	Unified analytics repository (e.g., Snowflake, BigQuery).
Decentralized	Each department owns its own analytics store.
Hub-and-Spoke / Lakehouse	Combines raw data storage (lake) + structured models (warehouse).
Data Lake	Raw, schema-on-read storage for diverse data.
Data Mesh	Domain-oriented, decentralized, governed sharing of data products.
Data Fabric	Automated, metadata-driven integration layer across sources.

Centralized Data Architecture – Explained

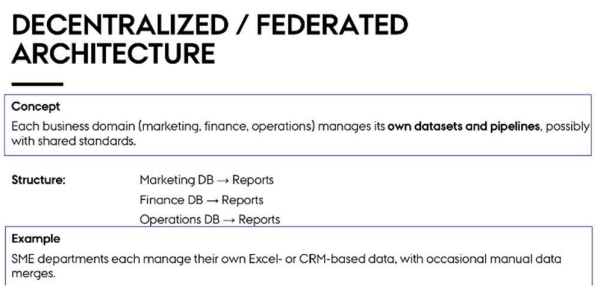


- Concept
  In a centralized data architecture, all data from different departments or systems is collected, cleaned, transformed, and stored in one central place—usually a Data Warehouse.
    - This means:
        - Sales, marketing, finance, operations, HR…
          → all send their data into one system
        - The data is integrated using ETL processes
        - Everyone in the organization uses the same trusted data source for reporting and analytics
    - It is the classic model used for BI and enterprise reporting.
- Structure (Simple Flow)
  Data Sources → ETL / ELT → Central Data Warehouse → BI / Analytics Tools
    - Data Sources:
        - CRMs, ERPs, transaction systems, files, APIs, cloud apps, etc.
    - ETL / ELT:
        - Data is extracted, cleaned, standardized, and loaded into the warehouse.
    - Central Data Warehouse:
        - A single repository where all enterprise data lives.
          Examples: Snowflake, Azure SQL, Redshift, BigQuery, SQL Server DW
    - BI & Analytics:
        - Dashboards, reports, ad-hoc analytics, KPIs.
- Example
    - A retail company collects:
        - sales transactions
        - customer data
        - product details
        - store performance metrics
    - It loads them all into one centralized data warehouse (e.g., Snowflake or SQL Server) so the entire business uses one unified data source for reporting.
- Strengths
    - Single source of truth
    - High data quality & consistency
    - Easier governance and security

- Works well for structured, enterprise data
- Limitations
  - Can become a bottleneck as data grows
  - Slow to scale with new data sources
  - IT teams become gatekeepers
  - Less flexible for real-time analytics or big data use cases

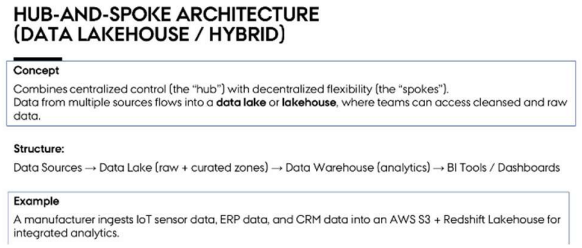
Decentralized / Federated Data Architecture

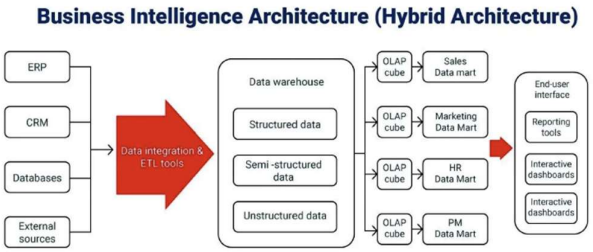
- Concept
  In a decentralized (or federated) architecture, each business domain manages its own data, storage, pipelines, and reporting tools.
  Instead of one central data warehouse, you have multiple independent data systems, one per department or domain.



  - This means:
  - Marketing controls its own data
  - Finance controls its own data
  - Operations controls its own data
  - They may follow shared data standards, but there is no single central repository
- Structure (Simple Flow)
  Marketing DB → Marketing Reports
  Finance DB → Finance Reports
  Operations DB → Operations Reports
  - Each unit builds and manages its own:
    - databases
    - transformation logic
    - KPIs
    - dashboards/reporting tools
- Example
  - In a small or medium enterprise (SME):
    - Each department stores data independently (e.g., Excel, CRM, local databases)
    - They produce their own reports
    - When a company-wide report is needed, teams manually merge the data
  - Example tools:
    - Excel files, department-specific CRMs, Google Sheets, QuickBooks, local SQL databases.
- Strengths
  - High flexibility for departments
  - Faster local decision-making
  - Less dependence on a central IT team
  - Good for small organizations or ones with independent units
- Limitations
  - No single source of truth
  - Inconsistent metrics or definitions across departments
  - Lots of data silos
  - Manual effort needed for cross-department analysis
  - Hard to scale as the organization grows

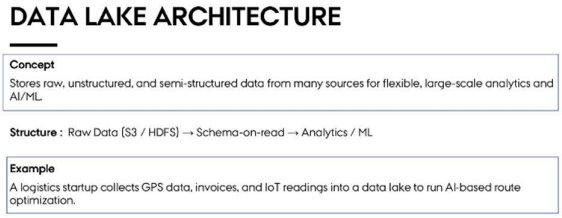
Hub-and-Spoke Data Architecture (Data Lakehouse / Hybrid)





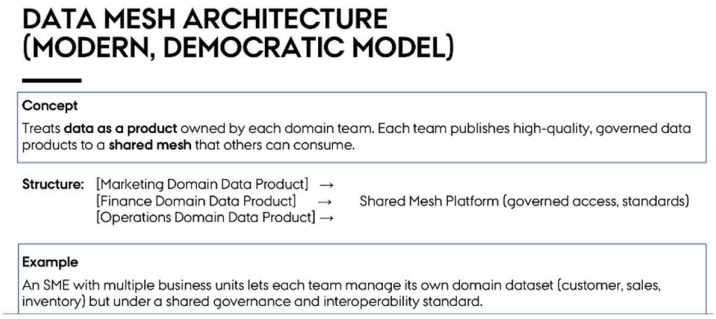
- Concept
  In a hub-and-spoke architecture, there is a centralized hub that stores all data, combined with decentralized spokes that allow teams to access and analyze the data independently.
    - The hub centralizes raw and curated data from multiple sources.
    - Spokes allow teams to work with data without duplicating storage or pipelines.
    - Teams can access both cleansed (curated) data and raw data for analytics, AI/ML, or reporting.
- Structure (Simple Flow)
  Data Sources → Data Lake (Raw + Curated Zones) → Data Warehouse (Analytics) → BI Tools / Dashboards
    - Each team or business unit accesses the hub via "spokes":
        - IoT / Sensor Data → Hub → Analytics Dashboards
        - ERP Data → Hub → Finance / Operations Dashboards
        - CRM Data → Hub → Marketing / Sales Dashboards
- Each unit uses the hub for:
    - Accessing centralized datasets
    - Running analytics and reports
    - Building dashboards or KPIs
- Example
    - A manufacturer ingests:
        - IoT sensor data (production)
        - ERP data (inventory, procurement)
        - CRM data (customers)
    - Data is ingested into AWS S3 + Redshift Lakehouse:
        - Stored in raw and curated zones in the lake
        - Processed into Redshift for analytics
        - Visualized in BI dashboards for integrated insights
    - Strengths
        - Single source of truth with centralized data
        - Teams can work independently without duplicating storage
        - Easier to scale for analytics across multiple domains
        - Supports advanced analytics and machine learning
    - Limitations
        - Requires robust central governance and data management
        - Hub can become a bottleneck if not properly designed
        - Higher initial setup cost compared to fully decentralized systems

Data Lake Architecture



- Concept
  A data lake stores raw, unstructured, and semi-structured data from multiple sources in a centralized repository.
  - Enables large-scale analytics and AI/ML workloads.
  - Data is ingested as-is; schema is applied only when the data is read (schema-on-read).
  - Ideal for storing diverse data formats such as JSON, CSV, logs, images, or sensor data.
- Structure (Simple Flow)
  Raw Data (S3 / HDFS / Cloud Storage) → Schema-on-Read → Analytics / ML / BI
  - Raw Data Zone: Stores unprocessed data from all sources.
  - Processing Layer: Data can be transformed, cleaned, or enriched as needed for analytics.
  - Consumption Layer: Data is accessed by analytics tools, dashboards, or AI/ML models.
- Example
  - A logistics startup collects:
    - GPS tracking data from delivery vehicles
    - Invoice and transaction data
    - IoT sensor readings from trucks
  - All data is stored in AWS S3 data lake:
    - Data is ingested in raw form
    - Schema-on-read is applied when running queries or ML models
    - AI models use the data for route optimization and predictive maintenance
  - Tools Used: AWS S3, HDFS, Databricks, Spark, Python ML libraries, BI tools
- Strengths
  - Can store any type of data without upfront schema design
  - Scales easily for big data workloads
  - Supports advanced analytics, AI, and machine learning
  - Flexible for exploratory and ad-hoc analytics
- Limitations
  - Raw data can become a “data swamp” without governance
  - Schema-on-read can slow down queries if not optimized
  - Requires strong metadata management and cataloging
  - Security and access control can be complex

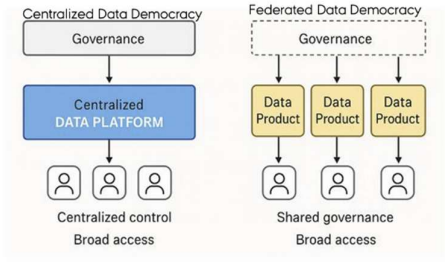
Data Mesh Architecture (Modern, Democratic Model)



- Concept
  Data Mesh treats data as a product, owned and managed by individual domain teams.
  - Each team is responsible for high-quality, governed data products.
  - Data products are published to a shared mesh, making them discoverable and consumable by other teams.
  - Encourages domain ownership, interoperability, and self-serve access while maintaining governance standards.
- Structure (Simple Flow)

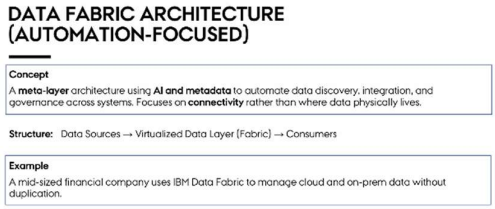
[Marketing Domain Data Product] →
[Finance Domain Data Product] → Shared Mesh Platform (governed access, standards)
[Operations Domain Data Product] →

    - Domain Teams (Spokes): Own and maintain their datasets, apply transformations, and ensure quality.
    - Shared Mesh (Hub): Provides a platform for discoverability, interoperability, governance, and secure access.
    - Consumers: Teams across the organization can query or integrate data products without controlling them directly.
- Example
    - An SME with multiple business units:
        - Marketing team manages customer engagement data
        - Finance team manages invoicing and revenue data
        - Operations team manages inventory and logistics data
    - All teams publish data products to a shared platform with:
        - Standardized metadata
        - Access control policies
        - Interoperability standards
    - Other teams can consume these products for analytics, reporting, or machine learning without duplicating data.
    - Tools/Platforms: Databricks, Snowflake, AWS Lake Formation, Apache Kafka, dbt, Tableau/Power BI
- Strengths
    - Decentralized ownership promotes accountability and data quality
    - Data products are discoverable and reusable across domains
    - Supports scalability for large organizations
    - Encourages a self-serve analytics culture
- Limitations
    - Requires cultural change and domain accountability
    - Strong governance and interoperability standards are essential
    - Initial setup is more complex than centralized architectures
    - Can be challenging for small organizations without domain expertise
- Data Mesh – "Mini Data Kingdoms"? Not Really.
  Data Mesh enables data democracy, not isolated silos. Think of it like a federation, where ownership is decentralized, but access is shared.



    - Key Principles:
        - Decentralized Ownership, Not Access: Teams own their data products, but everyone can access and consume them via the shared mesh.
        - Data as a Product: Each dataset is treated like a product with quality, documentation, and clear ownership.
        - Central Catalog: A single catalog lists all data products and their owners for discoverability.
        - Local Empowerment: Teams can innovate and manage their own datasets independently.
        - Accountability, Not Anarchy: Governance and standards ensure reliability while promoting self-serve access.
    - Summary:
      Data Mesh is not about creating "mini data kingdoms." It's about giving teams responsibility for their data while enabling organization-wide access and interoperability—a balance of autonomy and governance.

Data Fabric Architecture (Automation-Focused)



- Concept
  Data Fabric is a meta-layer architecture that leverages AI and metadata to automate:
  - Data discovery
  - Data integration
  - Data governance across heterogeneous systems
  - Focuses on connectivity and accessibility rather than where data physically resides.
  - Provides a virtualized, unified view of data from multiple sources for consumption by analytics, AI, or BI tools.
- Structure (Simple Flow)
  Data Sources → Virtualized Data Layer (Fabric) → Consumers (Analytics / BI / AI)
  - Data Sources: Cloud, on-premises, structured, semi-structured, or unstructured data
  - Virtualized Layer (Fabric): Creates an integrated, unified view of data without moving or duplicating it
  - Consumers: BI dashboards, analytics platforms, AI/ML models, or applications access the data seamlessly
- Example
  - A mid-sized financial company uses IBM Data Fabric to:
    - Integrate data from cloud services and on-prem systems
    - Avoid duplicating data in a central repository
    - Provide business users and analysts with a single, virtual view of the organization's data
  - Tools/Platforms: IBM Data Fabric, Talend, Informatica, Denodo, Microsoft Purview
- Strengths
  - Eliminates data silos by connecting all systems
  - Reduces data duplication and storage costs
  - Automates discovery, integration, and governance
  - Enables real-time access to data across environments
- Limitations
  - Complexity in initial setup and configuration
  - Requires strong metadata management
  - Dependent on AI/automation accuracy for data integration
  - May not fully replace traditional warehouses for heavy analytics

OVERVIEW

Model	Centralization	Governance Strength	Flexibility	Scalability	Data Democracy Fit
Centralized	High	Strong	Low	Medium	⚠️ Limited
Decentralized	Low	Weak	High	Low	⚠️ Risky
Hub-and-Spoke (Lakehouse)	Medium	Medium-Strong	Medium	High	✅ Balanced
Data Lake	Low	Weak	High	Very High	⚠️ Needs governance
Data Mesh	Distributed	Strong (federated)	High	High	✅ Excellent
Data Fabric	Virtualized	Strong (automated)	Very High	Very High	✅ Emerging best practice

How to choose a Model

- Centralized Data Warehouse
  - Scenario Fit:
    - Company wants a single source of truth for finance and sales reporting.
    - Example: Using BigQuery or Snowflake to consolidate ERP and CRM data.
  - Pros: Standardized metrics, reliable reports.
  - Cons: Expensive, may slow down department-specific experimentation.
- Decentralized / Departmental Analytics
  - Scenario Fit:
    - Marketing tracks campaigns independently in Google Sheets or CRM dashboards.
    - Finance keeps its own QuickBooks reports.
  - Pros: Quick setup, low cost, flexible.
  - Cons: Data silos, inconsistent KPIs, manual company-wide reporting.
- Hub-and-Spoke / Lakehouse
  - Scenario Fit:
    - SME collects raw sales, website, and inventory data in S3, transforms into Redshift for analytics.
    - Teams access curated datasets for dashboards.
  - Pros: Combines flexibility + governance, supports AI/ML later.
  - Cons: Requires more initial setup and IT expertise.
- Data Lake
  - Scenario Fit:
    - SME with IoT devices (e.g., small logistics or manufacturing startup) collects GPS, sensor, and invoice data.
    - Data is stored raw in S3 or HDFS for AI-based route optimization.
  - Pros: Handles unstructured data, future-proof for ML.
  - Cons: Needs governance to avoid data swamp.
- Data Mesh
  - Scenario Fit:
    - SME with independent business units managing their own datasets but wanting to share insights via a governed platform.
    - Example: Marketing, Finance, and Operations teams publish curated datasets to a central catalog for cross-team consumption.
  - Pros: Encourages accountability, scalable for growth.
  - Cons: Overkill for very small companies; needs cultural buy-in.
- Data Fabric
  - Scenario Fit:
    - SME using hybrid cloud + on-premises systems (e.g., ERP on-prem + SaaS CRM + cloud storage).
    - Data Fabric provides a virtualized layer for unified access without duplication.
  - Pros: Reduces integration complexity, real-time access.
  - Cons: Expensive and complex; better suited for mid-sized to large SMEs.

SME Scenario	Recommended Model	Why
Limited resources, single data analyst	**Hub-and-Spoke (Simplified)**	Low cost, manageable, scalable.
Tech-savvy startup with multiple data owners	**Mini Data Mesh**	Encourages ownership and flexibility.
Early-stage company with Excel-based reporting	**Centralized**	Simplicity over complexity.
Data-heavy SME (IoT, digital marketing)	**Hybrid Lakehouse**	Balances flexibility and control.

Classic Models

Often from Kimball, Inmon, and early data warehousing theory — 1990s–2010s

Model	Description
Enterprise Data Warehouse (EDW)	A single, centralized, integrated warehouse for the entire enterprise.
Independent Data Marts	Department-specific warehouses, not integrated.
Operational Data Store (ODS)	A layer that integrates operational data in near real time, often for reporting.
Classic Data Architecture	Usually a mix of ODS + EDW + marts; batch ETL processes.
Hub-and-Spoke / Corporate Information Factory (CIF)	Central EDW (hub) with dependent data marts (spokes).
Enterprise Data Bus	Integration via shared data standards and messaging bus (SOA or EAI style).

Enterprise Data Warehouse (EDW)

- Concept
  - A single, centralized, integrated repository for the entire organization's data.
  - Stores structured data from multiple operational systems (ERP, CRM, finance, HR, etc.).
  - Supports enterprise-wide reporting, analytics, and decision-making.
- Structure (Simple Flow)
  Multiple Source Systems (ERP, CRM, Finance, Operations)
  ↓
  Enterprise Data Warehouse (Central Hub)
  ↓
  BI Tools / Dashboards / Analytics
  - Integration: Combines data from all departments into a consistent format.
  - Access: Business units and analysts access the warehouse for reporting, KPIs, and analytics.
- Example
  - A multinational company consolidates:
    - Sales data from CRM
    - Financial transactions from ERP
    - Employee data from HR system
  - All data is stored in a central EDW (e.g., Snowflake, Teradata, Oracle).
  - Management uses dashboards to monitor performance across the enterprise.
- Strengths
  - Single source of truth for the entire enterprise
  - Consistent metrics and definitions across departments
  - Supports strategic decision-making and enterprise analytics
- Limitations
  - High initial setup and maintenance cost
  - Less flexibility for department-specific experimentation
  - Slower to adapt to new data types or real-time requirements

Independent Data Marts

- Concept
  - Department-specific warehouses that are not integrated with the rest of the organization's data.
  - Each department manages its own data, transformations, and reporting independently.
  - Typically used when centralized enterprise data warehouses are not available or are too costly to implement.
- Structure (Simple Flow)
  Sales DB → Sales Reports
  Finance DB → Finance Reports

Marketing DB → Marketing Reports

- Each department builds and manages:
  - Its own databases
  - Transformation logic
  - KPIs
  - Dashboards and reporting tools
- Example
  - A small company:
    - Marketing tracks campaign data in Google Sheets
    - Finance keeps records in QuickBooks
    - Operations maintains inventory data in a local SQL database
  - When a company-wide report is needed, teams manually merge data from each department.
- Strengths
  - Quick to implement with minimal IT support
  - High flexibility for departments
  - Faster local decision-making
- Limitations
  - No single source of truth
  - Inconsistent metrics across departments
  - Data silos make cross-department analysis difficult
  - Hard to scale as the organization grows

Operational Data Store (ODS)

- Concept
  - A centralized layer that integrates operational data from multiple source systems in near real-time.
  - Typically used for operational reporting, dashboards, and short-term analytics.
  - Acts as an intermediate layer between source systems and enterprise data warehouses.
- Structure (Simple Flow)
  Source Systems (ERP, CRM, POS, IoT) → Operational Data Store (ODS) → Reporting / Dashboards
  - Key Feature:
    - Near real-time integration ensures up-to-date information for operational decision-making.
    - Often includes light transformations to clean or standardize data.
  - Access: Analysts and business users access the ODS for day-to-day operational insights, not deep historical analysis.
- Example
  - A retail company collects:
    - Point-of-sale transactions
    - Inventory updates
    - Customer orders
  - Data is ingested into an ODS in near real-time, allowing managers to monitor:
    - Stock levels
    - Sales performance
    - Order fulfillment
  - The ODS feeds the EDW for long-term historical analysis.
- Strengths
  - Provides current, near real-time operational data
  - Enables fast reporting and decision-making
  - Simplifies data integration for downstream analytics

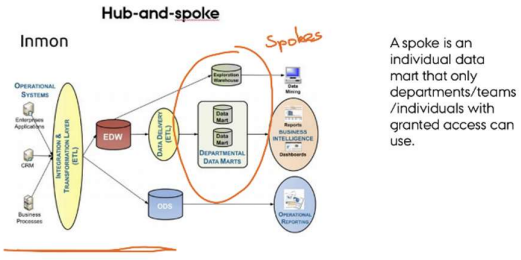
- Limitations
    - Not designed for complex historical analysis
    - Limited support for advanced analytics or AI/ML
    - Requires maintenance to ensure data consistency and low latency

Classic Data Architecture

- Concept
    - Traditional enterprise data architecture, typically from the 1990s–2010s.
    - Usually a combination of:
        - Operational Data Store (ODS): Near real-time operational reporting
        - Enterprise Data Warehouse (EDW): Centralized, integrated repository for analytics
        - Data Marts: Department-specific subsets of data for specialized reporting
    - Relies heavily on batch ETL processes to move and transform data.
- Structure (Simple Flow)
  Source Systems → ODS (Near Real-Time) → ETL → EDW (Central Warehouse) → Data Marts → BI / Dashboards
    - Batch ETL: Data is extracted, transformed, and loaded at scheduled intervals (daily, weekly).
    - ODS: Handles operational reporting needs.
    - EDW: Provides a single source of truth for enterprise analytics.
    - Data Marts: Serve department-specific analytics and dashboards.
- Example
    - A retail chain:
        - POS transactions feed the ODS for daily sales reporting.
        - ETL pipelines load historical sales and inventory data into the EDW.
        - Marketing, Finance, and Operations access data marts for departmental reports and KPIs.
- Strengths
    - Provides a structured, governed approach to data management
    - Supports both operational and analytical reporting
    - Clear separation of responsibilities (ODS vs. EDW vs. marts)
- Limitations
    - Batch ETL introduces latency; near real-time insights are limited
    - Complex to maintain multiple layers (ODS, EDW, marts)
    - Less flexible for ad-hoc or AI/ML workloads
    - Can be costly for small or rapidly growing organizations

Hub-and-Spoke Model (Classic)

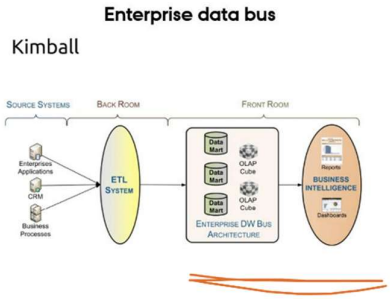
- Concept:
    - A central data warehouse (hub) stores integrated data from multiple sources.
    - Spokes are department-specific data marts that provide controlled access to relevant subsets of data.
    - Only users with granted permissions can access each spoke.
- Structure (Simple Flow):
  Central Data Warehouse (Hub)
  ↓
  [Sales Data Mart] → Sales Team
  [Finance Data Mart] → Finance Team



[Marketing Data Mart] → Marketing Team

- Key Features:
  - Departments get tailored data for reporting without affecting the central warehouse.
  - Promotes data governance and controlled access.
  - Ensures a single source of truth while enabling team-specific analytics.
- Strengths:
  - Centralized management with controlled departmental access
  - Reduces duplication of data
  - Structured approach for reporting and BI
- Limitations:
  - Can be rigid; not ideal for ad-hoc or exploratory analytics
  - Building and maintaining multiple data marts adds complexity
  - Slower to adapt to new data sources or analytics needs
- Summary:
  Classic hub-and-spoke models were foundational for modern data architecture. They emphasized centralized governance with departmental flexibility, but are less suited for today's big data, AI, and real-time analytics requirements, which is why newer models like Lakehouse, Data Mesh, and Data Fabric have emerged.

Enterprise Data Bus (EDB)

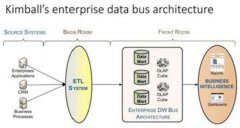


- Concept
  - Integration architecture using a shared messaging bus to connect multiple systems.
  - Relies on common data standards to enable interoperability.
  - Often implemented in Service-Oriented Architecture (SOA) or Enterprise Application Integration (EAI) style.
  - Focuses on real-time or near real-time data exchange rather than centralized storage.
- Structure (Simple Flow)
  Source Systems → Enterprise Data Bus (Messaging / Event Bus) → Target Systems / Consumers
  - Source Systems: ERP, CRM, IoT, legacy applications
  - Enterprise Data Bus: Standardizes messages and handles routing, transformation, and delivery
  - Consumers: Applications, BI tools, downstream analytics systems
- Example
  - A financial services firm integrates:
    - Core banking system
    - CRM
    - Risk management platform
  - All systems communicate via a shared message bus using standardized XML/JSON messages.
  - Enables real-time transaction monitoring, alerts, and analytics without centralizing all data.
- Strengths
  - Real-time or near real-time integration
  - Loosely coupled systems reduce dependencies
  - Promotes interoperability across heterogeneous systems
  - Supports event-driven architectures and agile integration
- Limitations
  - Complexity in designing and managing the bus
  - Requires strict adherence to data standards
  - Governance and monitoring are critical to avoid message failures
  - Not optimized for analytical workloads; often needs complementary EDW or data lake

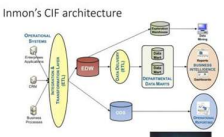
Choice Criteria: Kimball vs. Inmon

Aspect	Kimball Approach (Dimensional / Bottom-Up)	Inmon Approach (Corporate / Top-Down)
Core Idea	Build **data marts** for specific business areas first, then integrate into a data warehouse	Build a **centralized enterprise data warehouse (EDW)** first, then create data marts from it
Data Flow	Bottom-up: Marts → Integrated Warehouse	Top-down: EDW → Marts
Design Focus	Dimensional modeling: **star/snowflake schemas**, user-friendly for BI	Normalized design (3NF), enterprise-wide integration and consistency
Implementation Speed	Faster: business units get early results from individual marts	Slower: central warehouse takes time to build but ensures standardization
Flexibility	Highly flexible for individual departments; easy to modify marts	Less flexible; changes in central EDW can be complex
Data Governance	Light governance per mart; risk of inconsistent metrics	Strong governance and consistent definitions across enterprise
Best For	Organizations needing **quick, department-level insights**	Large enterprises needing **a single source of truth**
ETL Complexity	Moderate: ETL per mart	Higher: ETL from operational systems to EDW and then to marts
Cost / Resources	Lower upfront cost; can grow incrementally	Higher upfront cost; requires significant planning and resources

Kimball's enterprise data bus architecture



Inmon's CIF architecture



Characteristics	Favours Kimball	Favours Inmon
Business decision support requirements	Tactical	Strategic
Data integration requirements	Individual business requirements	Enterprise-wide integration
The structure of data	KPI, business performance measures, scorecards…	Data that meet multiple and varied information needs and non-metric data
Persistence of data in source systems	Source systems are quite stable	Source systems have high rate of change
Skill sets	Small team of generalists	Bigger team of specialists
Time constraint	Urgent needs for the first data warehouse	Longer time is allowed to meet business' needs.
Cost to build	Low start-up cost	High start-up costs

Decision Guidelines

1. Need fast insights / incremental rollout? → Kimball
2. Require a single source of truth with strong governance? → Inmon
3. Small or medium enterprises (SMEs): Kimball's bottom-up approach is usually more practical.
4. Large, complex enterprises: Inmon's top-down approach ensures long-term consistency.
5. Hybrid: Many organizations combine both—start with Kimball-style marts, then gradually unify under an Inmon-style EDW.

Modern architecture Models

(Cloud-era reinterpretations — 2015–present)

Model	Core Idea	Rough Mapping to Classic Model
Centralized Warehouse	Unified analytics repository (e.g., Snowflake, BigQuery).	→ **EDW** Centralized enterprise warehouse for structured analytics
Decentralized	Each department owns its own analytics store.	→ **Independent Data Marts** Department-specific warehouses, siloed but flexible
Hub-and-Spoke / Lakehouse	Combines raw data storage (lake) + structured models (warehouse).	→ **CIF / Classic Data Architecture** modernized Integrates raw + curated data for analytics and BI
Data Lake	Raw, schema-on-read storage for diverse data.	→ Expanded version of **ODS + staging area** Handles large-scale, exploratory, or AI/ML workloads
Data Mesh	Domain-oriented, decentralized, governed sharing of data products.	→ Conceptual successor to **Enterprise Data Bus** Decentralizes ownership but ensures governed access
Data Fabric	Automated, metadata-driven integration layer across sources.	→ Evolution of **Enterprise Data Bus** with AI & automation Virtualized, real-time, and self-serve integration

Takeaway 1: Modern architectures evolve classic models

- Classic models like EDW, ODS, and data marts were great for structured, batch-oriented reporting.
- Modern needs—big data, unstructured data, AI/ML, and real-time insights—require new architectures.
- So, modern architectures like Data Lake, Lakehouse, Data Mesh, and Data Fabric are built on top of the classic concepts but handle more types of data and more flexible use cases.
- Example:
    - An EDW handles structured sales and finance data.
    - A Lakehouse can handle that same structured data plus IoT sensor data, JSON logs, and images, all in one place.

Takeaway 2: Many organizations combine models

- Modern data architectures are not one-size-fits-all. Companies often mix and match depending on need:
    - Lakehouse + Data Mesh: Gives central storage of raw + curated data (Lakehouse) but lets individual teams own and share their datasets (Data Mesh).
    - Data Fabric: Adds automation and metadata-driven integration, connecting multiple systems in real time.
- Key Idea: You don't need to pick just one model; the combination depends on your business priorities.

Takeaway 3: Mapping shows evolution from classic to modern

- Classic architecture: EDW, ODS, data marts, Enterprise Data Bus → structured, batch, centralized.
- Modern architecture:
    - Lakehouse → combines raw (like ODS) + curated warehouse (like EDW)
    - Data Mesh → decentralizes data ownership like an evolved Enterprise Data Bus
    - Data Fabric → adds automation, AI, and self-serve access
- Think of it like this:
    - Classic = structured, centralized, batch
    - Modern = flexible, domain-oriented, real-time, big data ready

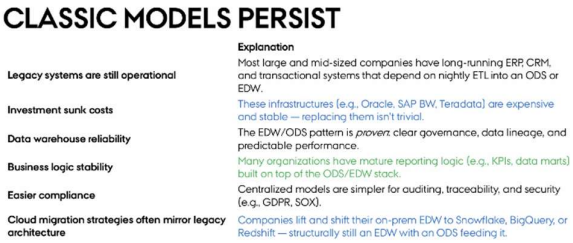
SIMPLIFIED ALIGNMENT TABLE

Classic Model	Modern Equivalent
Enterprise Data Warehouse	Centralized Warehouse
Independent Data Marts	Decentralized Architecture
Operational Data Store	Data Lake (staging/raw zone)
Classic / CIF (Hub-and-Spoke)	Hub-and-Spoke / Lakehouse
Enterprise Data Bus	Data Mesh / Data Fabric

Classic Data Models Persist

Explanation:

Even with modern data architectures emerging, classic models like EDW, ODS, and data marts remain widely used. There are several reasons for this persistence:



- Legacy Systems Still Operational
    - Most large and mid-sized companies run long-standing ERP, CRM, and transactional systems.
    - These systems depend on nightly ETL into an ODS or EDW.
- Investment and Reliability
    - Companies have sunk costs in hardware, software, and personnel training.
    - Systems like Oracle, SAP BW, Teradata are expensive but extremely stable.
    - The EDW/ODS pattern is proven, providing:
        - Clear governance
        - Data lineage
        - Predictable performance
- Mature Business Logic
    - Reporting logic, KPIs, and data marts are already built on top of the ODS/EDW stack.
    - Changing architecture would require rebuilding these systems, which is costly and risky.
- Compliance and Auditability
    - Centralized models are simpler to audit, secure, and comply with regulations like GDPR or SOX.
- Cloud Migration Often Mirrors Legacy Architecture
    - Companies often lift-and-shift their on-prem EDW to cloud solutions like Snowflake, BigQuery, or Redshift.
    - Structurally, it's still an EDW with an ODS feeding it, just on modern infrastructure.
- Summary:
    - Classic data models persist because they are stable, reliable, compliant, and already integrated into business processes.
    - Even as modern architectures like Lakehouse, Data Mesh, and Data Fabric emerge, many organizations continue to run and modernize classic EDW/ODS systems, often in combination with new models.

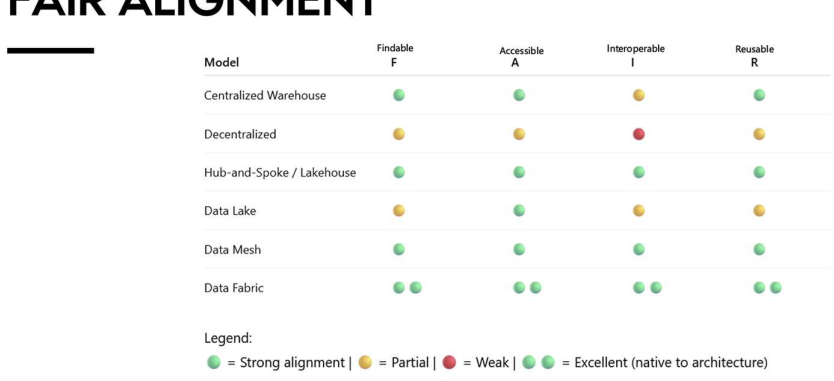
Architecture Enabling FAIR Principles

FAIR Principle	Architectural Feature That Enables It	Examples
Findable	Metadata catalogs, data dictionaries, and global identifiers	Data mesh registries, Data Fabric metadata layers, Snowflake Data Catalog
Accessible	APIs, query layers, governed access control, authentication	Power BI, Looker, Data Fabric virtual connectors
Interoperable	Common data models, semantic layers, standard vocabularies	Data Mesh "data contracts," Lakehouse schema-on-read
Reusable	Provenance tracking, documentation, versioning, clear licenses	Lakehouse "curated zones," Fabric lineage tools

Explanation of FAIR in Architecture

- Findable:
  - Data should be easy to locate within an organization.
  - Metadata catalogs and unique identifiers make datasets discoverable across teams.
- Accessible:
  - Data must be retrievable through standardized interfaces (APIs, query layers) with proper security.
  - Tools like BI platforms and virtual connectors allow safe and governed access.
- Interoperable:
  - Data should work across systems and domains.
  - Common models, semantic layers, and contracts ensure different teams can use the same datasets consistently.
- Reusable:
  - Data should be well-documented, versioned, and licensed to allow future use.
  - Curated zones, lineage tools, and provenance tracking ensure datasets are reliable for multiple applications, including analytics, ML, and reporting.

Data Architecture Model	Findable	Accessible	Interoperable	Reusable	Overall FAIR Alignment
1. Centralized Data Warehouse	✅ **High** — Central repository + common metadata make data easily searchable.	✅ **High** — Clear, controlled access points (SQL, BI).	⚠️ **Medium** — Integration usually via fixed schemas; rigid standards.	✅ **High** — Curated, well-documented data supports reuse.	**Strong on F, A, R** but limited flexibility (I).
2. Decentralized / Federated	⚠️ **Low-Medium** — Data scattered, minimal metadata.	⚠️ **Medium** — Each domain manages access independently.	❌ **Low** — Little standardization across systems.	⚠️ **Medium** — Data reused locally, but poor cross-domain reuse.	**Weak FAIR** — autonomy without interoperability.
3. Hub-and-Spoke (Hybrid / Lakehouse)	✅ **High** — Metadata and catalog central in hub.	✅ **High** — Unified access layer across structured/unstructured data.	✅ **High** — Shared standards + flexible schema-on-read.	✅ **High** — Central curation zone + domain flexibility.	**Balanced & FAIR-compliant** — good mix of control and agility.
4. Data Lake	⚠️ **Medium** — Data may be findable if cataloged; often not.	✅ **High** — Accessible to analysts; scalable cloud access.	⚠️ **Medium** — Variety of formats; needs governance to integrate.	⚠️ **Medium** — Raw data reusable with effort.	**Potentially FAIR**, but becomes a *data swamp* without metadata discipline.
5. Data Mesh	✅ **High** — Domain data products registered in central catalog.	✅ **High** — Governed APIs; access by design.	✅ **High** — Shared vocabularies, interoperable "data contracts."	✅ **High** — Data products documented, versioned, and stewarded.	**Excellent FAIR fit** — decentralization with strong governance.
6. Data Fabric	✅ **Very High** — Automated metadata discovery and indexing.	✅ **Very High** — Virtualized layer ensures seamless, governed access.	✅ **Very High** — AI-driven metadata unifies formats dynamically.	✅ **Very High** — Provenance, lineage, and automation enhance reuse.	**Leading-edge FAIR** — fully aligned with all principles.



Low Cost Architecture

- When we say "low-cost data architecture" for SMEs and data democracy, we mean creating a smaller, cheaper, but still effective version of the data systems that big companies use.
    - Scaled-down and affordable:
      Instead of huge, expensive enterprise systems (like Oracle, SAP BW, or Teradata), SMEs use cloud storage and compute that scale with their needs. This keeps costs low.
    - FAIR-aligned:
      Even though it's smaller, the architecture still follows FAIR principles:
        - Findable: Data can be easily discovered
        - Accessible: Teams can access it safely
        - Interoperable: Data can work across tools and systems
        - Reusable: Data is well-documented and versioned
    - Cloud, open-source, and self-service tools:
        - Cloud storage/warehouses (e.g., AWS S3, BigQuery) replace expensive hardware
        - Open-source ETL, analytics, and metadata tools (e.g., dbt, Metabase, Amundsen) replace costly enterprise software
        - Self-service access empowers teams to explore and use data without depending entirely on IT
- In short: It's about giving SMEs enterprise-like data capabilities—integrated, governed, and FAIR—without the high cost or complexity of full-scale enterprise systems.

Key Design Principles

Design	Low-Cost Implementation
Cloud-first, pay-as-you-go	Use SaaS or PaaS (Google Cloud, Microsoft Azure, AWS Free Tier, or even Zoho/HubSpot) to avoid hardware costs.
Open-source or freemium tools	Adopt free versions of ETL, visualization, and metadata tools (e.g., Airbyte, Metabase, Superset).
Automation over manpower	Use low-code integration tools (Zapier, Make, Power Automate) to connect systems instead of coding custom APIs.
Self-service analytics	Use Power BI, Google Looker Studio, or Tableau Public to let business users explore data.
Centralized but simple storage	Store all cleaned data in a single warehouse or data lake — often Google BigQuery (free tier) or PostgreSQL on a cloud VM.
Data governance "light"	Define roles (data owner, data steward) and use naming conventions and metadata sheets (Google Sheets or Notion).
Scalability through modularity	Start with one business domain (sales, customers), then expand gradually.

Example

Retail SME – "UrbanWear Boutique"

Business Context

A small clothing retailer with:

- Online store (Shopify)
- POS in two physical stores
- Marketing data from Instagram and Mailchimp
- Monthly sales reports in Excel

URBANWEAR BOUTIQUE - ARCHITECTURE

Layer	Tool	Purpose	Cost
Sources	Shopify, Square, Instagram Ads	Sales & marketing data	Existing
Ingestion	Airbyte or Meltano	Free ETL syncs	$0
Storage	Supabase / Cloud SQL (PostgreSQL)	Cloud warehouse	~$25/month
Modeling	dbt Core (open source)	Transformations	$0
Visualization	Looker Studio / Power BI Free	Dashboards	$0
Governance	Google Sheets + Great Expectations	Metadata + validation	$0
Orchestration	GitHub Actions / CRON jobs	Scheduling	$0

- Key Idea

- This architecture is low-cost because it mostly uses free or inexpensive cloud and open-source tools.
- It's FAIR-aligned: the data is Findable, Accessible, Interoperable, and Reusable.
- Even as a small business, UrbanWearBoutique can see sales and marketing performance in dashboards without expensive enterprise systems.

NordicRetail ApS

Company Background

- **Size:** ~60 employees
- **Industry:** Retail & e-commerce
- **Goal:** Combine sales, marketing, and customer data for reporting and better decision-making — *on a small budget.*

- Step-by-Step Explanation
  - Source Layer:
    - Pull data from Shopify (online sales), HubSpot (CRM), Google Analytics (website activity), and Excel (manual reports).
  - Ingestion Layer:
    - Use tools like Airbyte or Meltano to automatically pull data from all sources into a central storage system.
  - Storage Layer:
    - Store the raw and cleaned data in PostgreSQL on Google Cloud SQL (free tier) for structured querying and analysis.
  - Transformation Layer:
    - Use dbt Core to clean the data, join tables, and build analytics-ready models.
  - Visualization Layer:
    - Business users can explore data through dashboards in Power BI or Looker Studio.
  - Governance Layer:
    - Keep track of data definitions and run tests using Google Sheets and Great Expectations to ensure quality and FAIR compliance.
  - Automation Layer:
    - Schedule ETL and transformation jobs with GitHub Actions or CRON so the system runs automatically without manual intervention.
- Key Points
  - Most tools are free or low-cost, making it suitable for small businesses.
  - The architecture is modular and scalable: NordicRetail can start with sales data and expand to marketing or customer data later.
  - Supports FAIR principles: data is findable, accessible, interoperable, and reusable.

Common Mistakes SMEs Should Avoid

- Over-engineering early
  - Mistake: Adopting complex, enterprise-grade tools before the business is ready.
  - Consequence: Wastes money, creates unnecessary complexity, and slows adoption.
  - Fix: Start with low-cost, cloud-based, or open-source tools that match current needs.
- Ignoring documentation
  - Mistake: Not maintaining data dictionaries or metadata, even in small setups.
  - Consequence: Teams struggle to understand datasets; mistakes increase.
  - Fix: Maintain simple documentation using Google Sheets, Notion, or lightweight metadata tools.
- Poor access management
  - Mistake: Sharing credentials informally instead of defining roles.
  - Consequence: Security risks, accidental data loss, or breaches.
  - Fix: Define user roles (e.g., data owner, analyst) and use proper access controls.
- No data quality checks
  - Mistake: Skipping simple validations on incoming data.
  - Consequence: Mistrust in dashboards and reports; bad decisions based on inaccurate data.
  - Fix: Implement lightweight checks with tools like Great Expectations or simple validation scripts.
- Lack of scalability mindset
  - Mistake: Building a monolithic solution without modularity.
  - Consequence: Hard to add new data sources or domains as the business grows.
  - Fix: Start modular, e.g., begin with one domain (sales) and expand gradually.

The Layers of Data Architecture

From Transaction to Transformation

1. Transactional Systems (Source Layer)
    - These are the systems that generate data continuously in day-to-day operations.
    - Examples:
        - ERP (Enterprise Resource Planning)
        - CRM (Customer Relationship Management)
        - POS (Point of Sale) systems
    - Purpose: Capture raw business events such as sales, orders, or customer interactions.
2. Operational Data Store (ODS)
    - Integrates data from multiple transactional systems in near real-time.
    - Acts as an operational mirror of the business.
    - Purpose: Provide current, consolidated data for operational reporting and short-term decisions.
    - Features:
        - Light transformations or cleaning
        - Near real-time updates
3. Analytical / BI / Enterprise Data Warehouse (EDW)
    - Periodically extracts data from the ODS for historical analysis, trends, and strategic reporting.
    - Purpose: Support long-term decision-making, trend analysis, and KPI monitoring.
    - Features:
        - Structured, integrated data
        - Supports dashboards, reporting, and business intelligence tools

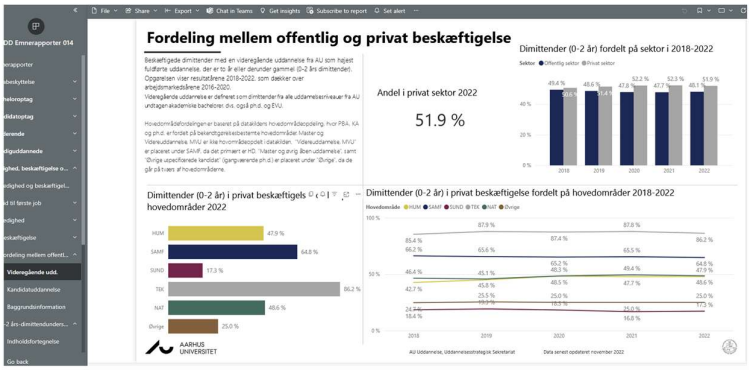
- Key Idea:
    - Each layer has a distinct role:
        - Transactional systems: capture raw events
        - ODS: near real-time operational view
        - EDW/BI: historical analysis and strategic insights

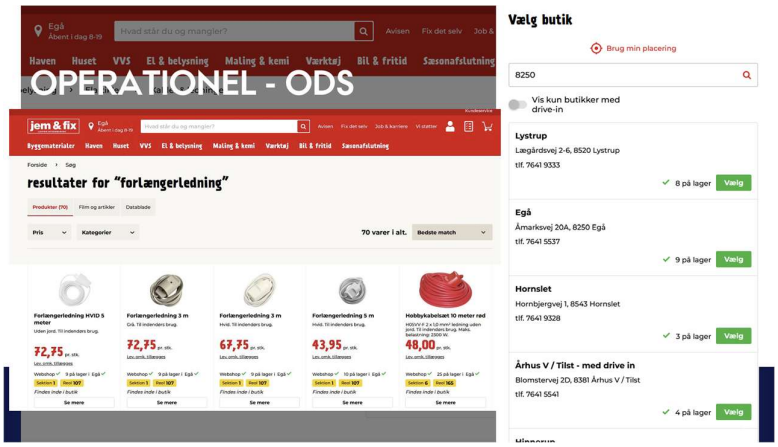
🧱 Layer-by-Layer Explanation

Layer	Main Role	Data Latency	Users	Data Lifespan	Examples
Transactional Systems (OLTP)	Capture day-to-day business operations (orders, invoices, sensor data).	Real-time	Operational staff	Very short (minutes/ hours)	ERP, CRM, POS
Operational BI / ODS	Consolidate and harmonize operational data for *live reporting* and process control.	Near real-time (minutes)	Middle management, operations control	Short (days/ weeks)	Live sales dashboard, production monitoring
Analytical BI (EDW / Data Warehouse)	Store, integrate, and analyze historical data from ODS and other sources.	Batch (daily, weekly)	Management, analysts, data scientists	Long-term (months/years)	Trend analysis, forecasting, planning

Operational Data Store (ODS)

- Concept:
  - The ODS is a bridge between operational systems and analytical systems.
  - While often misunderstood, it is indispensable in enterprise data architecture.
- Purpose / Role:
  - Integrate data from multiple operational systems
    - ERP, CRM, IoT devices, POS systems, etc.
    - Provides a consolidated, near real-time view of operational data.
  - Provide operational intelligence
    - Enables actionable insights within hours instead of days.
    - Supports daily operations and short-term decisions, unlike EDWs which focus on historical analysis.
- Bridge to analytical systems
    - Feeds Enterprise Data Warehouses (EDW), BI tools, and ML pipelines.
    - Ensures that analytical systems have clean, integrated, and current data to work with.
- Key Features:
  - Near real-time data integration
  - Light transformations (cleaning, validation, standardization)
  - Supports operational reporting and dashboards
- Key Idea:
  - The ODS is the “live mirror” of the business: it lets organizations see what’s happening now, while the EDW focuses on historical trends.





ODS in Modern Architectures

- Hub-and-Spoke / Lakehouse
    - The ODS serves as the “staging zone” or “bronze layer” in a lakehouse architecture.
    - Layers in a Lakehouse:
        - Bronze: Raw, current operational data (like a traditional ODS)
        - Silver: Cleaned, historical, integrated data
        - Gold: Business-ready, semantic data for reporting and analytics
    - Role: Acts as the entry point for operational data, which is then refined into analytics-ready datasets.
- Data Mesh
    - In a domain-oriented architecture, each domain team can have its own ODS-like layer.
    - Purpose: Consolidate operational inputs from that domain before publishing a governed data product.
    - Role: Provides local operational intelligence and ensures high-quality data products for sharing.
- Data Fabric
    - The ODS becomes virtualized rather than physically stored.
    - Data is queried live from operational systems via APIs or federated query engines.
    - Role: Provides real-time access to operational data without duplication, enabling analytics across systems.

Why Many SMEs and Enterprises Still Need an ODS

- The Operational Data Store (ODS) remains relevant because it provides near real-time integration of operational data, bridging transactional systems and analytical layers. It’s ideal for use cases where current, actionable data is needed quickly, without waiting for batch processing in an EDW.

Business Case	Why ODS is Ideal
Retail – daily sales dashboards	Needs near real-time integration from POS and e-commerce
Manufacturing – production monitoring	Combines MES, ERP, and sensor data quickly
Banking – risk and compliance checks	Requires current but not historical snapshots
Logistics – fleet tracking	Integrates multiple live data streams (GPS, orders, fuel)
Healthcare – patient record summaries	Combines EMR and lab results rapidly without duplication

- Key Takeaways
    - The ODS is not about historical analysis—it’s about current, operational intelligence.
    - It reduces latency between transactional systems and actionable insights.
    - It’s useful across industries where timely operational decisions matter.
    - Even in modern architectures (Lakehouse, Data Mesh, Data Fabric), the ODS concept remains relevant, either as a staging layer, a domain-specific layer, or a virtualized access layer.

Classic + Modern Hybrid Architecture

- Modern data architectures don’t replace classic layers like ODS or EDW—instead, they absorb and extend them, creating a hybrid system that combines reliability with flexibility and scalability.

Modern architecture doesn’t replace ODS or EDW — it **absorbs** them.

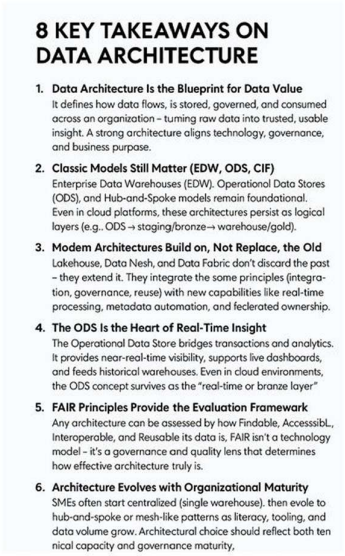
Classic Layer	Modern Equivalent	Purpose Today
ODS	Staging / Real-time integration layer	Operational integration
EDW	Curated analytics warehouse	Central governance
Data Marts	BI semantic models	Self-service
Data Lake	Raw + semi-structured data	Flexibility
Data Mesh / Fabric	Federated or automated integration	Scalability

- Key Takeaways
    - Classic layers remain essential: ODS, EDW, and data marts still provide reliability, governance, and operational insight.
    - Modern layers add flexibility and scale:
        - Data Lakes handle unstructured or semi-structured data for AI/ML and exploration.
        - Data Mesh / Fabric enable federated ownership, automation, and seamless integration.
    - Hybrid approach: Combines the stability of classic architectures with the agility of modern architectures, allowing SMEs and enterprises to modernize without discarding proven systems.

Data Democracy

Data democracy

Industry (SME context)	Key Challenges	Potential Benefits of Data Democracy
Manufacturing (SMEs)	- Limited data infrastructure and IT staff- Legacy machinery without IoT connectivity- Data silos across production, quality, and supply chain- Cost of modern analytics tools	- Predictive maintenance using affordable sensors and cloud tools- Improved production planning and waste reduction- Empowered shop-floor employees to identify bottlenecks- Faster feedback loops with suppliers
Agriculture / AgriTech	- Inconsistent data capture (manual logs, small-scale IoT)- Low data literacy among farm operators- Poor connectivity in rural areas	- Precision farming decisions (fertilizer, irrigation)- Shared dashboards between farmers and cooperatives- Cost optimization through data-driven crop management
Education / Training Providers	- Fragmented student information systems- Lack of analytics expertise- Privacy concerns under GDPR/FERPA- Limited budgets for cloud data platforms	- Better student engagement analytics- Data-driven curriculum improvement- Transparent reporting for funders and accreditation bodies
Local Retail / E-commerce Startups	- Small-scale data spread across Shopify, POS, social media- Dependency on third-party platforms- Lack of master data management- Inconsistent customer data	- Personalized marketing campaigns- Optimized stock management- Improved customer retention with loyalty insights- Faster decisions using low-cost BI dashboards
Professional Services (Consulting, Accounting, Legal)	- Client confidentiality and privacy constraints- Data stored in isolated tools (Excel, CRMs)- Limited automation of data processing	- Centralized client insights dashboard- Enhanced forecasting and workload planning- Improved collaboration and transparency with clients
Nonprofit and NGOs	- Donor, volunteer, and beneficiary data scattered across systems- Inconsistent recordkeeping- Low investment in analytics- Data ethics and consent issues	- Evidence-based impact reporting- Improved resource allocation- Greater transparency to stakeholders- Stronger case-making for funding
Logistics and Small Transport Operators	- Manual scheduling and routing data- Data not digitized (paper manifests)- Limited tracking capabilities	- Route optimization via open-source mapping tools- Real-time vehicle and fuel monitoring- Reduced fuel and maintenance costs
Creative and Media SMEs	- Highly unstructured data (images, videos, social metrics)- Limited metadata management- Platform dependency (YouTube, Instagram)	- Data-driven audience engagement- Better content targeting- Performance insights for monetization
Construction / Trades	- Fragmented data across projects and contractors- Low automation in site data capture- Versioning and document management issues	- Shared dashboards for project visibility- Cost and material optimization- Improved safety tracking through data visibility



8 KEY TAKEAWAYS ON DATA ARCHITECTURE

1. **Data Architecture Is the Blueprint for Data Value**
   It defines how data flows, is stored, governed, and consumed across an organization – turning raw data into trusted, usable insight. A strong architecture aligns technology, governance, and business purpase.
2. **Classic Models Still Matter (EDW, ODS, CIF)**
   Enterprise Data Warehouses (EDW). Operationol Data Stores (ODS), and Hub-and-Spoke models remain foundational. Even in cloud platforms, these architectures persist as logical layers (e.g., ODS → staging/bronze→ warehouse/gold).
3. **Modem Architectures Build on, Not Replace, the Old**
   Lakehouse, Data Nesh, and Data Fabric don't discard the past – they extend it. They integrate the some principles (integra- tion, governance, reuse) with new capabilities like real-time processing, metadata automation, and feclerated ownership.
4. **The ODS Is the Heart of Real-Time Insight**
   The Operational Data Store bridges transactions and analytics. It provides near-real-time visibility, supports live dashboards, and feeds historical warehouses. Even in cloud environments, the ODS concept survives as the "real-time or branze layer"
5. **FAIR Principles Provide the Evaluation Framewark**
   Any architecture can be assessed by how Findable, Accesssibl., Interoperable, and Reusable its data is, FAIR isn't a technology model – it's a governance and quality lens that determines how effective architecture truly is.
6. **Architecture Evolves with Organizational Maturity**
   SMEs often start centralized (single warehouse). then evole to hub-and-spoke or mesh-like patterns as literacy, tooling, and data volume grow. Architectural choice should reflect both ten nical capacity and governance maturity,

Cases

Case 1 - GreenPulse Energy Solutions

Descriptives
Sector: Renewable energy services (solar installation & maintenance)
Employees: ~80
Headquarters: Copenhagen, Denmark

Business Background
GreenPulse installs and services solar panels for private homes and small businesses.
They operate across several Danish regions and are expanding into Sweden and Germany.
The company wants to become more data-driven — to:

- Improve forecasting of energy output
- Optimize technician scheduling
- Track customer satisfaction and warranty claims
- Manage spare parts inventory

Management has limited budget but wants everyone to have access to data insights, not just IT.

Current Data Environment
GreenPulse has several data sources:

1. CRM System (HubSpot): stores customer info, contracts, and service history.
2. Operations App (Fieldy): mobile app used by technicians to log visits and maintenance.
3. Accounting System (Visma): financial data, invoices, supplier records.
4. IoT Platform: solar panels send real-time data (production, faults) to a cloud dashboard.

How Data Is Managed

- Data from each source is automatically exported nightly to cloud storage (Google Cloud Storage).
- A small data integration script (Python on Google Cloud Function) transforms the files and loads them into BigQuery tables.
- Data from all systems are stored in one shared cloud database, divided into "raw," "cleaned," and "reporting" zones.
- Power BI dashboards connect to BigQuery and are accessible to all departments.
- The company's *data champion* maintains a Google Sheet as a metadata catalog (with dataset names, owners, and last update dates).
- Quality checks (duplicates, nulls, date ranges) are automated with Great Expectations (open-source).
- Management encourages teams to request new dashboards or add fields, but changes are centralized through one data engineer.

Case 2 — Nordic Precision Tools (SME Manufacturing)

Descriptives
Industry: Metal component manufacturing for automotive suppliers
Employees: ~120
Location: Gothenburg, Sweden

Business Context
Nordic Precision Tools (NPT) produces customized metal parts and precision gears.
It runs several small production plants, each with its own scheduling and inventory system.
Management wants to use data to:

- Improve machine utilization
- Monitor energy costs
- Reduce scrap rates
- Forecast raw material needs

They've recently begun their digital transformation journey.

Data Environment

- Each plant maintains its own Excel files for production logs.
- Sales and orders are recorded in Visma ERP, which exports monthly CSV files.
- Quality control data is stored in separate Access databases at each site.
- There is no shared data warehouse.
- Data analysts at headquarters request files by email each month and merge them manually in Excel.
- There is no formal data catalog — datasets are identified by filenames.
- Reporting dashboards are built in Power BI Desktop by one analyst and shared as PDFs by email.
- Each department uses its own version of key metrics (e.g., "scrap rate" defined differently in each plant).

Case 3 — EcoAid International (Nonprofit / NGO)

Descriptives

Industry: Environmental nonprofit managing reforestation projects
Employees: ~60
Offices: Netherlands, Kenya, and Indonesia

Business Context

EcoAid collects and manages environmental and social impact data to support reforestation projects.
They collaborate with:

- Local field partners
- Donors and sponsors
- Volunteers
- Government environmental agencies

They want to make data more transparent and sharable for global stakeholders.

Data Environment

- Each country office has a local database or Google Sheet where field data is collected.
- A central data team in Amsterdam has built an internal platform using Airtable and AWS services.
- Each country publishes its own dataset ("data product") monthly (e.g., *Tree Survival Rates*, *Volunteer Hours*, *Carbon Offsets*).
- These datasets are validated locally, then uploaded to the shared platform through a governed API.
- The central platform automatically generates metadata (source, date, steward, license).
- Donors can access project dashboards via a public Power BI portal with role-based permissions.
- All datasets follow shared definitions (ISO country codes, project IDs, environmental taxonomies).
- EcoAid encourages each team to own and improve its data quality, supported by central governance guidelines.

Give an architectural sketch of case 1 data flow

Based on the description, which modern data architecture model best describes Case 1, 2, 3setup?

Case 1 — GreenPulse Energy Solutions

Key Observations:

- Centralized cloud storage and BigQuery warehouse.
- Data from multiple sources (CRM, IoT, ERP) integrated nightly.
- Zones for raw, cleaned, and reporting data (bronze/silver/gold concept).
- Power BI dashboards are accessible to all departments, but changes are centralized through a data engineer.
- Metadata catalog exists, quality checks automated.

Analysis:

- Combines centralized control with some accessibility to teams.
- Classic hub-and-spoke / lakehouse model: raw + curated data in one central store, BI dashboards for consumption.

Best Fit: Hub-and-Spoke / Lakehouse

Case 2 — Nordic Precision Tools (SME Manufacturing)

Key Observations:

- Each plant and department manages its own data independently.
- No shared warehouse; data is merged manually in Excel.
- Key metrics are inconsistent across departments.
- Reporting is local and manual.

Analysis:

- No central repository.
- Each department is fully responsible for its own data, including metrics and reporting.

Best Fit: Decentralized / Federated Data Architecture

Case 3 — EcoAid International (Nonprofit / NGO)

Key Observations:

- Each country office owns and publishes its own datasets ("data products").
- Central platform aggregates datasets with governance and metadata.
- Local teams are accountable for data quality, but all datasets follow shared standards.
- Data is accessible to external stakeholders via dashboards.

Analysis:

- Domain-oriented, governed sharing of data.
- Local ownership with central standards and metadata aligns with data as a product concept.

Best Fit: Data Mesh

General

Step 1: Identify Key Features of the Organization

Look at:

- Size (SME vs. large enterprise)
- Data sources (CRM, ERP, IoT, spreadsheets, etc.)
- Data complexity (structured, unstructured, real-time, batch)
- Team setup (central IT vs. domain teams)
- Reporting / BI usage (self-service or centralized)
- Governance (metadata, quality checks, standards)

Step 2: Examine How Data is Managed

- Is there a central repository (warehouse or lake)? → Suggests centralized or hub-and-spoke/lakehouse.
- Do departments/domains manage their own datasets independently? → Suggests decentralized / federated architecture.
- Are datasets treated as products with metadata, ownership, and governed sharing? → Suggests data mesh.
- Is integration automated and virtualized, with AI or metadata layers? → Suggests data fabric.

Step 3: Look at the Data Flow

- From source → staging → analytics / dashboards: Traditional ETL, single warehouse → centralized / lakehouse.
- Manual merging and local reporting per team: → Decentralized / federated.
- Domain-owned, governed datasets shared via APIs: → Data mesh.
- Virtualized live access across multiple sources: → Data fabric.

Step 4: Match Key Principles / Characteristics

Architecture Type	Indicators
Centralized / Warehouse	Single analytics repository, central IT, unified KPIs, BI dashboards
Decentralized / Federated	Independent departmental datasets, local metrics, manual integration
Hub-and-Spoke / Lakehouse	Raw + curated data zones, shared warehouse, accessible dashboards
Data Lake	Raw, semi-structured/unstructured data, schema-on-read
Data Mesh	Domain-owned datasets (“data products”), governed sharing, metadata catalog, FAIR principles
Data Fabric	Automated, metadata-driven integration, virtualized live access

Step 5: Confirm with Contextual Clues

- Budget limitations → likely low-cost / SME-friendly architecture
- Growth / scaling needs → may indicate mesh or fabric
- Data maturity → early stage = decentralized/federated; more mature = hub-and-spoke/lakehouse

Step 6: Make a Reasoned Choice

- Map the observed features and data flow to the characteristics table.
- Pick the model that fits most of the key features, rather than trying to match perfectly.
- Explain reasoning with observations from Step 1–5.

Make a FAIR assessment on the cases architecture

Case 1 — GreenPulse Energy Solutions

Architecture: Hub-and-Spoke / Lakehouse

FAIR Principle	Assessment	Details / Examples	Strengths	Limitations
Findable	High	Metadata catalog maintained in Google Sheets lists datasets, owners, and last updates; BigQuery central repository	Easy for team members to locate datasets; searchable by name and type	Simple spreadsheet catalog may become hard to scale as data volume grows
Accessible	High	Power BI dashboards connect to BigQuery; access is controlled centrally	All departments have self-service access to dashboards; queries are standardized	Real-time access is limited; only nightly ETL updates
Interoperable	High	Shared schemas in BigQuery; cleaned "reporting" zone for consistent metrics	Cross-team use possible; different departments can use same tables reliably	Some flexibility limited by central schema; adding new sources requires central engineer intervention
Reusable	High	Curated "reporting" zone and automated validation via Great Expectations	Teams can reuse datasets for new dashboards, forecasting models, and reporting	May not support external or public data sharing easily
Overall FAIR Alignment	Balanced & FAIR-compliant	Central control + curated zones + self-service dashboards balance accessibility and governance	Strong cross-team usability; scalable for growth	Slight dependence on centralized engineer for changes

Summary:
GreenPulse achieves **high FAIR compliance** with controlled governance and a modern lakehouse design, balancing accessibility with reliability.

Case 2 — Nordic Precision Tools (NPT)

Architecture: Decentralized / Federated

FAIR Principle	Assessment	Details / Examples	Strengths	Limitations
Findable	Low–Medium	Data scattered in Excel files and Access DBs; no central catalog	Local teams can find their own data	Difficult to locate cross-plant datasets; filenames are inconsistent
Accessible	Medium	Files shared manually via email; dashboards distributed as PDFs	Teams can access local or department-level reports	No real-time access; manual process prone to delays and errors
Interoperable	Low	Metrics differ across plants (e.g., scrap rate definitions); no common vocabularies	Teams understand local formats	Cross-domain integration requires heavy manual reconciliation
Reusable	Medium	Local data can be reused in Excel for internal reports	Some ability to create insights locally	Hard to reuse for company-wide dashboards or historical analysis
Overall FAIR Alignment	Weak FAIR	Autonomy without standardization hinders interoperability and reusability	Simple and low-cost for small teams	Poor scalability; high risk of inconsistent KPIs and duplicate work

Summary:
NPT demonstrates **weak FAIR compliance**, as autonomy and decentralized processes limit cross-plant integration and reuse. This is typical for early-stage SMEs that have not yet invested in a centralized or governed data architecture.

Case 3 — EcoAid International

Architecture: Data Mesh

FAIR Principle	Assessment	Details / Examples	Strengths	Limitations
Findable	High	Each data product is registered in a central catalog; metadata includes source, steward, license, and update date	Easy for internal teams and external donors to locate datasets	Reliant on teams maintaining accurate metadata
Accessible	High	Governed APIs and role-based dashboards; public dashboards for donors	Access is controlled yet broad; stakeholders can query relevant datasets directly	External access depends on proper API implementation
Interoperable	High	Shared vocabularies (ISO country codes, project IDs); consistent schemas for environmental metrics	Enables integration across countries, projects, and external stakeholders	Requires all teams to follow the standard strictly
Reusable	High	Versioned, documented, stewarded datasets; clear licenses allow reuse	Datasets can be used for reporting, research, or donor updates	Reuse depends on central platform availability and governance adherence
Overall FAIR Alignment	Excellent FAIR fit	Strong decentralization + governance ensures full FAIR compliance	Supports scaling across domains, geographies, and external stakeholders	Complexity may be high for smaller SMEs without technical expertise

Summary:
EcoAid International exemplifies **excellent FAIR alignment**. The data mesh allows **domain ownership with central governance**, making data findable, accessible, interoperable, and reusable across teams and countries.

General

Generalized Detailed FAIR Assessment Table – All Options

Architecture Type	Findable (F)	Accessible (A)	Interoperable (I)	Reusable (R)	Overall FAIR Alignment	Key Indicators / Questions	Strengths / Advantages	Risks / Notes
Centralized Warehouse / EDW	High: central repo + metadata catalog	High: centralized access via SQL/BI	Medium: fixed schemas, moderate standards	High: curated & documented	Strong F, A, R; medium I	Single repository? Metadata catalog? Standard dashboards?	Reliable, audited, predictable BI	Limited flexibility, slower integration of new sources
	Medium: partial metadata, multiple repositories	Medium: some dashboards, some restricted access	Low: inconsistent standards between teams	Medium: partially curated datasets	Medium FAIR	Partial catalog or inconsistent access	Can support some FAIR principles	Hard to locate or reuse data across teams
	Low: no central catalog, scattered datasets	Low: manual or file-based access	Low: no integration standards	Low: ad hoc or raw data only	Weak FAIR	Manual reporting, minimal governance	Low cost for small setups	Difficult to scale, inconsistent KPIs
Decentralized / Federated	High: each domain catalogs locally	High: local access control	Medium: partial standards across domains	Medium: datasets reusable locally	Balanced FAIR (rare)	Does each department maintain metadata? Access controlled?	Autonomy with partial FAIR	Hard to combine across domains
	Medium: some datasets documented, inconsistent naming	Medium: some dashboards or shared files	Low: inconsistent vocabularies	Medium: reuse within domains only	Weak FAIR	Partial compliance; some standardization	Flexible for teams	Poor cross-department interoperability
	Low: scattered spreadsheets, no metadata	Low: manual sharing	Low: incompatible metrics	Low: little reuse	Weak FAIR	No catalog, minimal governance	Easy to implement	High risk of silos, inconsistent KPIs
Hub-and-Spoke / Lakehouse	High: central hub catalog	High: unified dashboards & queries	High: shared standards & schema-on-read	High: curated + domain flexibility	Balanced & FAIR	Raw → cleaned → reporting zones? Metadata central?	Combines control + agility	Requires central engineering, maintenance of metadata
	Medium: catalog partially maintained	Medium: some self-service dashboards	Medium: partial schema-on-read	Medium: partial curation	Medium FAIR	Hub partially used, spokes scattered	Flexible, semi-governed	Risk of inconsistent usage across spokes
	Low: hub metadata missing, scattered spokes	Low: limited dashboards	Low: no standardization	Low: raw only	Weak FAIR	Mostly decentralized, minimal governance	Quick to deploy	Poor cross-team use and reuse
Data Lake	High: well-cataloged	High: cloud/API access	Medium: needs semantic integration	Medium: curated zones available	FAIR possible	Cataloged raw/semi-structured data? Governance applied?	Scalable, supports diverse data types	Without governance → data swamp
	Medium: partial cataloging	Medium: some APIs	Medium: mixed formats	Medium: partial curation	Medium FAIR	Partial metadata, inconsistent access	Flexible, low-cost	Hard to reuse and integrate for analytics
	Low: no catalog	Low: file-based access	Low: heterogeneous formats	Low: raw, undocumented	Weak FAIR	Essentially a "data swamp"	Cheap storage	Risky for analytics and decision-making
Data Mesh	High: all domain products registered	High: governed APIs & role-based dashboards	High: shared vocabularies, "data contracts"	High: versioned, stewarded	Excellent FAIR	Are data products registered? Metadata and governance enforced?	Decentralized + fully FAIR	Needs strict adherence to governance, more complex setup
	Medium: only partial cataloging	Medium: limited API access	Medium: inconsistent standards	Medium: partial reuse	Medium FAIR	Only some domains follow standards	Partial FAIR compliance	Cross-domain interoperability weaker
	Low: no catalog, no governance	Low: manual sharing only	Low: inconsistent metrics	Low: datasets not stewarded	Weak FAIR	Rare in practice	Minimal FAIR support	High risk of inconsistency, poor reuse
Data Fabric	Very High: automated discovery & catalog	Very High: virtualized queries, governed access	Very High: AI-driven integration, semantic layers	Very High: automated lineage & provenance	Leading-edge FAIR	Metadata auto-discovered? Virtualized access? Lineage tracked?	Can integrate heterogeneous sources dynamically	Complexity & cost; depends on full implementation
	High: manual cataloging + partial virtualization	High: API access	High: partially unified formats	High: partially documented	Balanced FAIR	Partial automation	Supports FAIR but requires governance	Less "leading-edge" without full automation
	Medium / Low: partial virtual layers, minimal metadata	Medium / Low: limited access	Medium / Low: only some sources unified	Medium / Low: minimal lineage	Medium / Weak FAIR	Low adoption of fabric features	May improve with investment	Still better than ungoverned decentralization
Operational Data Store (ODS)	Medium: mirrors operational systems	High: near real-time dashboard access	Medium: integrates multiple operational sources	Medium: short-term reuse	Medium FAIR	Is operational data integrated in near real-time? Accessible via dashboards?	Real-time actionable insights	Limited historical reuse, short-term snapshots
Independent Data Marts	Low–Medium	Medium	Low	Medium	Weak FAIR	Are marts isolated? Metrics standardized?	Quick local reporting, autonomy	No single source of truth, manual integration for cross-department reports
Enterprise Data Bus / SOA / EAI	Medium	High	High	Medium	Medium FAIR	Shared services or APIs? Standards enforced?	Cross-system integration, interoperability	Governance complexity, not self-service friendly
Hybrid / Classic + Modern	Varies by component: EDW, lakehouse, mesh, fabric	Varies: F/A/I/R depend on implementation	Depends on combined layers	Depends	Balanced to High FAIR	Are classic EDW + modern components integrated? Metadata and governance unified?	Leverages legacy + modern tools	Complexity; orchestration needed to avoid confusion

How to Use This Table in Exams

1. Identify the architecture type from the case.
2. For each FAIR principle, select High / Medium / Low based on indicators.
3. Justify with case-specific evidence (data sources, flow, ownership, access).
4. Assess overall FAIR alignment: Weak / Medium / Balanced / Excellent / Leading-edge.
5. Optionally, note improvements (metadata, governance, APIs, standards).