A NEW PERSPECTIVE ON DATA ANALYSIS

HTTPS://WWW.VISIBLESYSTEMSCORP.COM

VISIBLE SYSTEMS CORPORATION 24 SCHOOL STREET 2ND FL, BOSTON, MA 02108

Contents

Inside the Mind of a Data Analyst - A New Perspective by Visible Systems Corporation

Visible Systems is thrilled to introduce you to this eBook on "Inside the Mind of a Data Analyst – A New Perspective" where we will explore innovative approaches to data analysis and offer fresh insights into the evolving role of the Data Analyst in modern organizations.

What We'll Explore

The traditional approach to data management is no longer sustainable in our complex, multi-source environments. Today, we'll dive deep into the challenges currently facing data teams and propose a robust, entity-centric solution.

1. The Challenge of Scaling Intelligence

We begin by covering the critical, everyday problems faced by analysts, specifically the challenges of **scaling intelligence** across the business and the inherent **pitfalls of traditional end-to-end pipelines**. You'll see how the reactive "pipeline-per-request" mindset leads to inefficiency, fragmentation, and zero reusability.

2. A New Perspective: Domain and Entity Modeling

We will introduce a **new domain entity perspective**—a fundamental shift in how we think about, structure, and deliver data. This approach focuses on creating stable, reusable **core entities** that serve as the universal building blocks for every business question.

3. Strategy and Implementation

Finally, we will demonstrate the significant **benefits of this approach**, including greater flexibility, a faster response to new requests, and the establishment of a single source of truth. We will conclude with **practical steps for implementation**, showing you how to start building an iterative, scalable domain model today.

Our goal is to move you away from building one-off data *sets* and toward architecting reusable data *solutions*.

Harnessing the Power of the Data Analyst

Data Analysts hold **untapped strategic power** within organizations. By leveraging their unique combination of technical skills and business acumen, companies can unlock new opportunities, anticipate market shifts, and drive strategic decision-making. This presentation is about delving into how we can fully harness this potential and empower analysts to become central agents of change.

Overcoming Inertia: Empowerment and Expertise

For Data Analysts to effectively drive innovation, two internal strategies are crucial:

1. Building Analyst Confidence and Strategy

Overcoming professional inertia is paramount. Analysts must be empowered to move beyond reactive reporting. This requires leadership support for strategies like **continuous learning** and the embrace of new tools and methodologies. Empowered analysts are confident analysts, ready to challenge outdated processes and drive organizational change.

2. Mastering Technical Depth

Building deep **technical expertise is essential** to bridge the gap between business questions and data infrastructure. This means mastering core skills like **SQL and to some extent generative AI**, completing complex, real-world projects, and staying updated with the latest tools and cloud environments. This rigorous approach enhances problem-solving capabilities and enables analysts to influence architectural decisions.

Focus: Modular, Reusable Data Solutions

Our core focus today is on shifting from building fragile, one-off reports to creating **modular, reusable data insights**.

By adopting a strategic, entity-based approach to data modeling, analysis teams can create flexible, scalable solutions that meet diverse and evolving business needs **efficiently and effectively**. This is the key to finally solving the persistent challenge of **scaling intelligence**—a challenge that traditional, end-to-end data pipelines are ill-equipped to handle. We will explore those challenges next.

The Pitfalls of Traditional Pipelines (excepts below taken from a presentation by Doug McFarlane, Staff Analytics Engineer, SpotHero during Starburst Trino Day).

The **traditional, pipeline-centric approach** to data analysis—where a new pipeline is custom-built for every business question—is fundamentally flawed and unsustainable. This methodology inevitably leads to a state of **data fragmentation and inefficiency**.

Let's look at how we think about our data pipelines, how we think about our requests, how we think about data, with the idea being that we can maybe move away from thinking about building these pipelines and start thinking about things from a more domain, entity perspective. Let's reconsider how we approach our data pipelines and incoming requests. The goal is to shift from a "pipeline-per-request" mindset, which is common and reactive, to a more sustainable and scalable domain-based, entity-centric perspective.

The Scenario: A Typical Week of Data Requests

In a typical week, multiple business units submit data requests. These situations often involve evolving requirements and expectations during the process. The starting point is clarified, data sources are identified, and necessary additions are considered. For the finance request, the process begins with orders data, which is joined with Salesforce revenue rules and catalog data. Tables are merged, filtered, transformed, and metrics, calculations, and classifications are added before final delivery to stakeholders. This workflow generally follows a consistent sequence from initial data source to completion.

Marketing requires conversion metrics by acquisition channel by week's end. Product management requests feature adoption statistics over time by Thursday, which involves sales territory data and win rate for reporting. Finance needs recognized revenue figures broken down by product line.

So, what do we do? How do we manage this?

We're looking for conversion rate and acquisition channel pay. So, what is that conversion rate? You know, that would be trying to understand how many people purchase our products. We know that data lives in S3. We also need to understand who these customers are. We know that customer data can be found in our Redshift backend database.

Okay, we need to get some orders data in there that's also in Redshift. Finance needs to recognize revenue by product line. But before we can do that, we need to report sales by win rate. We will need to pull opportunities from Salesforce, and the number of those opportunities that have changed, giving us our win rate - the percentage of customers that have purchased. Recall, our revenue will be coming from Redshift. And one more. Let's not forget about the product people. They're interested in feature adoption analysis segmented by customer types. This will need to be merged with adoption by customer that we got from Redshift for which we will need to write some rules around.

Let's take a step back and look at what we have committed to. It looks like we're going to be building four new pipelines comprised of four new data sets. Okay, so we've got our Databricks, marketing attribution summary. We got our finance revenue rolled up. We also have sales territory metrics and some product adoption analysis. So, we got four new pipelines, a handful of data platforms and different data analyst teams spread across a couple of different projects.

Let's Summarize

Marketing (by Friday): Requires analysis of conversion rates by acquisition channel.

Product (by Thursday): Seeks monitoring of feature adoption trends over time.

Sales (ASAP): Requests a report detailing win rates by sales territory.

Finance (by Friday): Needs recognized revenue data by product line, dependent on the Sales report.

There are four distinct requests, each accompanied by strict deadlines.

The Reactive Workflow: A Source-by-Source Plan

Given the impending deadlines, the team organizes the work, addressing each request as an independent project:

For Marketing (Conversion):

- Access purchase data from S3.
- Integrate with customer data stored in the Redshift backend.
- Combine with orders data, also from Redshift.

For Sales (Win Rate):

- Retrieve opportunity data from Salesforce.
- Calculate win rates by identifying converted opportunities (customers who have purchased).

For Finance (Revenue):

- Await completion of the Sales report.
- Extract revenue data from Redshift and link it to the new win-rate figures.

For Product (Adoption):

- Obtain usage data from Redshift.
- Merge with customer segment data from Redshift.
- Develop new, custom business rules tailored for this analysis.

Outcome: Four Pipelines, No Reusability

Let's take a step back and look at what we've just committed to. By treating each request as a unique pipeline, we are now on the hook to build. Upon review, the chosen approach results in the creation of four dedicated pipelines:

- "Databricks Marketing Attribution Summary"
- "Sales Territory Metrics" table
- "Finance Revenue Roll-up"
- "Product Adoption Analysis" dataset

This fragmented strategy generates isolated datasets across multiple platforms, often built by separate analysts. While immediate requests are fulfilled, no reusable assets have been established for future needs. The result is a fractured ecosystem. We've created four new, single-use pipelines and four new datasets, spread across a handful of data platforms and likely built by different analysts. We have successfully answered the four questions, but we have created no reusable assets for the future.

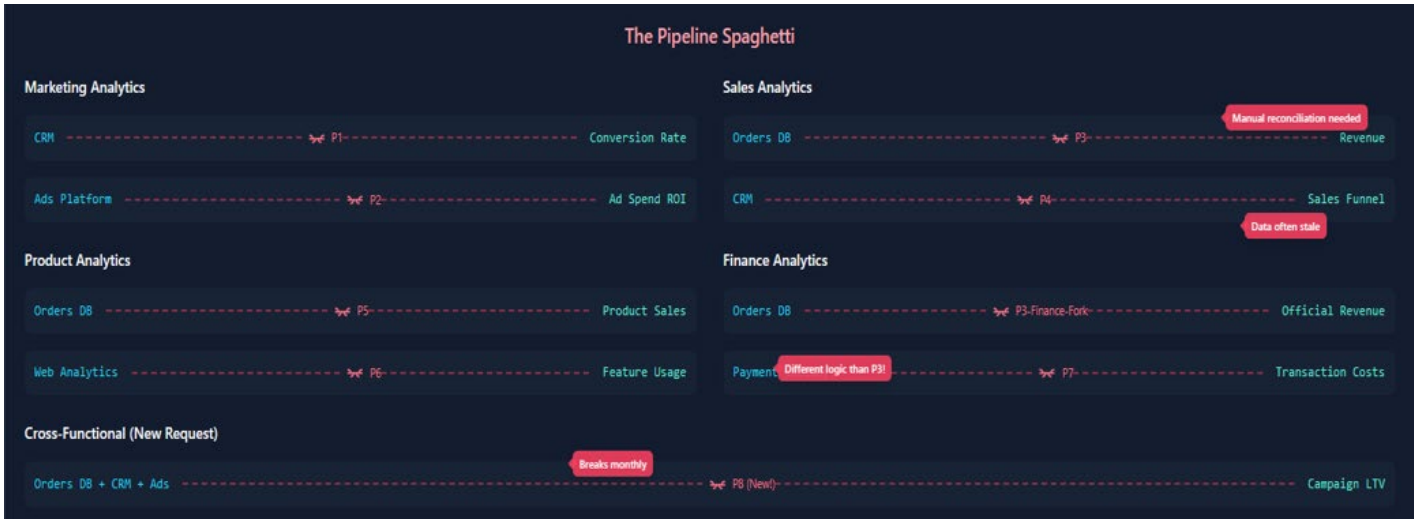
The Cost of Fragile Ecosystems

When analysts operate in a reactive "get-it-done" mode, they create a tangled web of specialized pipelines that are difficult to manage and nearly impossible to reuse. This custom-built ecosystem is incredibly fragile:

- **Data Silos and Inconsistency:** Each pipeline calculates metrics slightly differently and pulls data independently, creating silos and eroding confidence in the final numbers.
- **High Maintenance and Technical Debt:** Every pipeline is a maintenance burden. When a source system changes, multiple custom-built pipelines must be updated, drastically increasing technical debt and maintenance costs.
- **Slow Turnaround:** Simple questions can reveal the true cost of this technical debt. The time to deliver **reliable insights** often stretches into weeks, as analysts must spend precious time rebuilding and patching existing logic rather than focusing on analysis.

It's just what we've got to do. It's just how things work. Let's think about this though, what are we building? We know what we're doing, but what is it really? Let's go back to that marketing example again. What was it that they wanted? What was conversion rate, customer acquisition? Let's just think about this. We generally like to start from the source whenever we're going to build that full pipeline. We're going to pull in everything we need, get that end-to-end solution ready, and ship that table, data set, report, dashboard, or whatever it is.

Let's start with our sources. Let's start with S3. We need to join in user segmentation. Okay, where is that? Again, that's in Redshift. Good, we're out of S3 and we're moving towards Redshift. We're in that space already. Now let's go ahead and materialize something. We're ready to roll feeling pretty good about where we're at. We're going to look forward to that dashboard, that final goal. Let's see, was there anything else? We needed acquisition channel. So that comes from our UTM parameters in the data segment we've already pulled, materialized and left this segment. I guess we could go back and redo that, but we're kind of running towards the deadline. Okay, we're just going to shift everything over, move it all to the left. We need to move forward. We'll just add that afterwards. It's not the most effective or efficient, but it'll get the job done. Okay, I think we're good, but they now also want to understand how the customer is using the promotions, and which promotions are most effective. So, I guess we somehow need to squeeze that in. They need it right away. This is going to be a problem so we will just list it for now.



Deconstructing the "Get It Done" Mindset

Frequently, data teams encountering numerous requests adopt a reactive approach, stating: "This is simply what must be done; it is the standard process." However, it is essential to critically evaluate the outcomes of operating solely in this manner.

Case Study: The Marketing Pipeline

Consider a marketing request for "conversion rate by acquisition channel." The conventional response involves constructing a comprehensive pipeline tailored to this specific inquiry. This generally entails sourcing raw event data from S3, integrating user segmentation information from Redshift within the data warehouse, and materializing an intermediate table with the intention of delivering a refined dashboard. We build our intermediate table. At this point, we feel good. We're on track to deliver the final dashboard.

While this procedure appears effective initially, inherent flaws in the "pipeline-per-request" strategy soon emerge.

First Inefficiency: The Patch

As the team progresses, unforeseen requirements may arise, such as retrieving the acquisition channel from previously processed UTM parameters. Under time constraints, the optimal solution would be to revise and rebuild the materialized view. Instead, a temporary fix is often applied by rejoining segment data later in the pipeline. Although expedient, this method sacrifices efficiency and maintainability for immediate results.

Second Inefficiency: Scope Creep

We've patched the pipeline and are finally ready to ship, but a new, related request comes in: "This is great! Can you also show how customers are using promotions, and which ones are most effective?"

This should be a simple addition—it's related data. But in our model, this is a major problem. Our entire pipeline was custom-built for one narrow purpose. We can't just "squeeze in" the promotions data. It would require another set of Joins and a significant rebuild.

Upon completion of the initial pipeline, stakeholders might submit additional requests, such as analyzing customer interactions with promotions and measuring their effectiveness. Despite the related nature of this data, integrating it into a custom-built pipeline poses significant challenges, necessitating extensive modifications and additional Joins. Consequently, teams may defer such requests, assign them to new projects, and ultimately deliver single-use pipelines that do not support evolving business needs.

This approach highlights inefficiencies and limitations intrinsic to reactive, request-driven data workflows, underscoring the need for adaptable solutions that can accommodate future requirements.

The Allure of the Linear "Source-to-Completion" Pipeline

We are all familiar with the scenario: requirements and expectations shift after we have already started building. This uncertainty forces us into a very linear thought process: Where do we begin with this data? What do we need to add?

Let's use a typical finance request as an example

We start by pulling the base Orders data.

We go to Salesforce to pull in the necessary Revenue Rules.

We join Catalog Information to get the data organized.

We then join these disparate tables together.

Finally, we filter, transform, and add all the required metrics, calculations, and classifications.

We ship the final table. The request is closed, and the customer is happy.

The "Source-to-Completion" Habit

Notice the pattern. We consistently move from source to completion. This end-to-end approach feels logical because it seems to mirror the natural flow of data. As a result, we think about our deliveries in this linear way: "Where do we begin? What needs to change? How do we give it to the customer?"

The mental model for almost every request simplifies to a repeatable, tactical pattern:

Pull all the data together.

Filter the results.

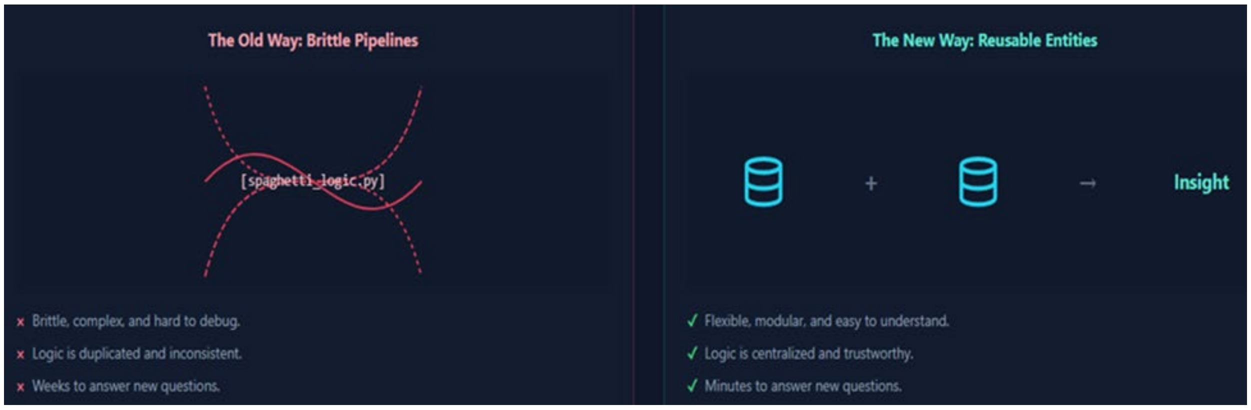
Transform it. Ship it.

The Reinforcing Loop

This linear thinking isn't just a habit; it's reinforced by the very analytics tools we work with, which are often designed to support this same "pull, transform, deliver" workflow.

Addressing these critical issues is key to achieving consistent, reliable data. The solution is to move away from the linear, source-to-completion mindset.

From Data Sets to Building Blocks: A Domain-Oriented Approach to Analytics


The Old Way: Brittle Pipelines
[spaghetti_logic.py]
× Brittle, complex, and hard to debug.
× Logic is duplicated and inconsistent.
× Weeks to answer new questions.
The New Way: Reusable Entities
+
→
Insight
✓ Flexible, modular, and easy to understand.
✓ Logic is centralized and trustworthy.
✓ Minutes to answer new questions.

When handling data split across systems—like "product promotion sales" attributes in both a backend database and Salesforce—we can either join data early or model sources separately. Modeling each system end-to-end and keeping them isolated preserves context and details. By joining them only at the final stage, we gain control over attribute selection and enhance cross-system data management.

The **new domain entity perspective** shifts the focus from building disposable, end-to-end *pipelines* to creating reusable, durable data **entities**. These entities serve as stable, high-quality building blocks that can be quickly combined to answer any business question without rebuilding the underlying data infrastructure.

Comparing a Traditional Data Warehousing vs. Entity-Based

Traditional Approach:
- Rigid, predefined schemas
- ETL (Extract, Transform, Load) focused
- Separate tables for each department
- Static, pre-aggregated reports
- High maintenance overhead
- Difficult to adapt to changing business needs
- Multiple versions of similar metrics
- Siloed data interpretation

Entity-Based Approach:
- Flexible, domain-oriented modeling
- Focus on business entities
- Centralized, reusable data components
- Dynamic, runtime aggregation
- Lower maintenance complexity
- Easily adaptable to new requirements
- Consistent metric definition

Key Differences:
- Traditional: Build specific pipelines
- Entity-Based: Build flexible building blocks
- Traditional: Source-centric
- Entity-Based: Domain-centric
- Traditional: Batch processing
- Entity-Based: Runtime processing

Analytics Teams Build the Company Domain Model

Analytics and data engineering teams are responsible for creating a unified company domain model – a challenging but essential task. While application teams develop components across various domains, analytics must integrate them to answer cross-domain questions, such as linking products, promos, and integrations from different systems. Even application teams face difficulties maintaining a consistent domain model

.

A Practical Example: Modeling with Core Entities

Consider a situation involving four business requests from Marketing, Sales, Finance, and Product. Although their requirements differ, it is possible to define four main entities common to all: Customers, Orders, Sessions, and Products By constructing models based on these core entities, relevant dimensions (attributes for filtering) and measures (facts for calculation) can be established. Modeling in this manner results in an organized system, where most facts are associated with a specific entity.

Entity	Dimensions	Measures
Customers	Customer Type	Count of Active Customers, Count of Churned Customers
Orders	Order Type, Order Date, Payment Method	Order Count, Revenue
Sessions	UTM Channel, Session Length	Conversion Rate, Session Count
Products	Product Type, Price	Product Net Revenue (Price - COGS)

The Entity-Centric ELT Process: Building the Core

The foundation of the entity-centric approach is a rigorous **Extract, Load, Transform (ELT)** process designed to create a central, normalized, and highly reusable data model. Unlike the traditional ETL process that focuses on immediate transformation for a single report, this ELT approach focuses on **data quality and semantic consistency** first.

Transformation: From Raw Data to Core Entity

The ELT pipeline takes data from disparate sources (the Bronze Layer), cleans and standardizes it, and transforms it into durable **core entities** within the Silver Layer.

1. **Extract & Load (Bronze):** Raw, untouched data is ingested and loaded directly into the data lake or warehouse, maintaining its original structure and context.
2. **Transform (Silver):** This is where the magic happens. We execute a robust set of transformations:
    - **Cleaning:** Handling nulls, removing duplicates, and ensuring data type consistency.
    - **Normalization:** Structuring data around the core entities (Users, Orders, Products, etc.).
    - **Combination:** Where attributes for a single entity exist in multiple source systems (e.g., customer details in both CRM and the backend), they are reconciled and combined into a single, comprehensive entity representation.

The result is a clean, structured, and validated entity model that provides a **reliable foundation for all subsequent analytics and business intelligence**.

The Payoff: Answering Any Question with "Building Blocks"

This entity-based model makes answering complex business questions a simple matter of combining the correct "building blocks." We keep all entities modeled at their independent grain (their lowest level of detail) and join them as needed, leveraging strong compute power to aggregate at runtime. Here are example aggregations.

Q: What is our conversion rate by acquisition channel?

A: Combine the Orders entity (revenue) and the Sessions entity (attribution channel).

Q: What is our recognized revenue by product line?

A: Combine the Orders entity (revenue) and the Products entity (product category).

Q: Are power users adopting new features more?

A: Combine Orders, Sessions, and Users to get feature usage, user segments, and conversion variance.

How these entities help resolve data inconsistencies - let's use the "Orders" entity as an

Marketing Department Request:
- Conversion rate by acquisition channel
- Uses Orders entity joined with Sessions entity
- Calculates: Order count / Session count
- Filters by UTM parameters

Finance Department Request:
- Recognized revenue by product line
- Uses Orders entity joined with Products entity
- Calculates: Sum of order prices
- Groups by product category

Sales Department Request:
- Win rate by sales territory
- Uses Orders entity joined with Geographic entity
- Calculates: Orders with successful conversion / Total opportunities
- Groups by sales territory

Product Department Request:
- Feature adoption by user segment
- Uses Orders entity joined with Users and Sessions entities
- Tracks feature usage rates
- Segments by user type

In each case, the Orders entity remains consistent but is flexibly joined with other entities to answer different business questions. This approach allows:
- Consistent metric definitions
- Reusable data components
- Runtime aggregation
- Reduced data redundancy

The Medallion Architecture as a Framework

This approach focuses on constructing reusable, entity-based components rather than creating narrow, end-to-end datasets. The Medallion Architecture illustrates this concept:

Bronze Layer: This contains raw data, sourced directly with minimal transformation.

Silver Layer: Serving as the domain-oriented layer, system-specific entities are developed here. Examples include dimension tabled in Salesforce and other applications which are cleaned, normalized, and prepared for further use.

Gold Layer: This is the analytics-ready layer, where individual components from the Silver layer are integrated to produce unified, conformed entity representations suitable for downstream consumption.

Entity-based modeling supports the development of scalable and reusable domain models. This provides a structured method for creating metrics or dimensions and lays the groundwork for establishing a reliable data source within the organization.

Flexible Queries and Runtime Aggregations

By creating these clean core entities at their lowest grain (Silver Layer), we shift the complexity away from repeated pipeline builds and toward **flexible querying and runtime aggregation** in the final Gold Layer.

- **Decoupling:** The metrics and dimensions are calculated *at the point of consumption* rather than being hardcoded into the pipeline.
- **Performance:** We leverage modern, strong compute engines (like Starburst or Databricks) to perform complex joins and aggregations on the fly.
- **Agility:** Because the core entities are stable, a new business request (e.g., "new conversion rate calculation") simply requires a new query joining existing, trusted entities, rather than building a new, fragile end-to-end data pipeline.

This disciplined process ensures that data is always **clean, structured, and ready to answer any question**, making the data analyst's job one of *insight creation* rather than *data engineering*.

Core Entities: The Building Blocks of Intelligence

Handling Entity Definition Conflicts

1. Conflict Identification Methods
- Compare metric calculations across departments
- Track divergent interpretations
- Identify source of definitional differences

2. Resolution Approach
- Convene cross-functional working group
- Document each department's perspective
- Analyze root causes of definitional variations
- Prioritize business value and accuracy

3. Conflict Resolution Strategies
- Create canonical definition
- Establish primary source of truth
- Define calculation precedence
- Develop clear tie-breaking rules

4. Technical Mitigation
- Build flexible entity models
- Allow multiple calculation methods
- Create metadata tracking variations
- Implement transparent calculation logic

5. Governance Mechanisms
- Establish data stewardship committee
- Create formal review process
- Require documentation for metric changes
- Implement version control for entity definitions

6. Communication Protocols
- Transparent reporting of definition changes
- Notify stakeholders of metric modifications
- Provide clear migration paths
- Maintain historical calculation methods

At the heart of the domain-entity perspective is the concept of **Core Entities**. These are the fundamental nouns of your business, such as **Customers, Products, Promotions, and Orders**. Instead of building custom tables for every report, the focus shifts to creating comprehensive, clean, and normalized models for these base entities first.

When these core entities are defined correctly, they automatically form a **Single Source of Truth (SSOT)**. This SSOT is stable, consistent, and reliable, enabling consistent insights across all departments of the organization.

The Payoff: Why Entity Modeling Works

Adopting an entity-based model delivers significant and measurable benefits that directly address the pain points of the traditional pipeline approach:

- **Single Source of Truth:** All metrics derived from an entity, such as Revenue from the Orders entity, are calculated the same way, eliminating data confusion and reducing argument over "whose numbers are right."
- **Increased Agility:** New business questions (e.g., "What is the conversion rate of new customer segment X?") can be answered in minutes or hours by simply combining existing, trusted entity models, rather than weeks spent building a new pipeline.
- **Reduced Engineering Overhead:** By building the entity once and reusing it, you drastically cut down on maintenance, patching, and technical debt. Updating a source system now only requires updating one entity model, not dozens of specialized pipelines.
- **Reusable Components:** Every entity model becomes a durable asset that the entire analytics team can leverage. This moves the team out of the reactive cycle and into a proactive, strategic role.
- **Data Consistency:** The rigorous ELT process (as covered in the Silver Layer of the Medallion Architecture) ensures data is always clean, structured, and validated before it's used for analysis.

Putting the Model into Practice: Excelling at Ad Hoc Analysis

The power of the new model extends far beyond simply generating predefined dashboards. Its real strength lies in empowering users to conduct sophisticated **ad hoc analysis**.

Because the core entities are stable and semantically consistent, users can:

1. **Construct Queries Visually:** Business users, even those without deep SQL expertise, can use self-served tools to visually drag-and-drop entity relationships (e.g., join Customers to Orders to Products).
2. **Gain Insights Quickly:** By leveraging the inherent structure and the power of runtime aggregation, users can get answers to complex, unique questions quickly and efficiently, democratizing data access throughout the organization. This capability transforms data analysis from a bottleneck into a competitive advantage.

Conclusion: The Future is Modular

By shifting our fundamental approach to data modeling, moving from building fragile, end-to-end pipelines to architecting durable, reusable **Core Entities**—we build a consistent, flexible data foundation that is ready for any challenge.

Key Principles for a Modern Analytics Approach

There is a lot to consider, but the journey can begin today. Simply thinking about your data in terms of entities is a huge first step and will immediately reveal commonalities in your day-to-day work.

As you move forward, keep these core principles in mind:

1. Build Once, Use Repeatedly

The primary goal is reusability. Focus on how you can build your core entities one time and then reuse them across multiple reports, analyses, and business requests.

2. Build for the Solution, Not Just the Ask

Our stakeholders often speak a different language, and their "ask" isn't always the "solution" we need to build. Sometimes, we must build a flexible solution for ourselves that allows us to deliver on their specific ask, rather than building a one-off product for that ask alone.

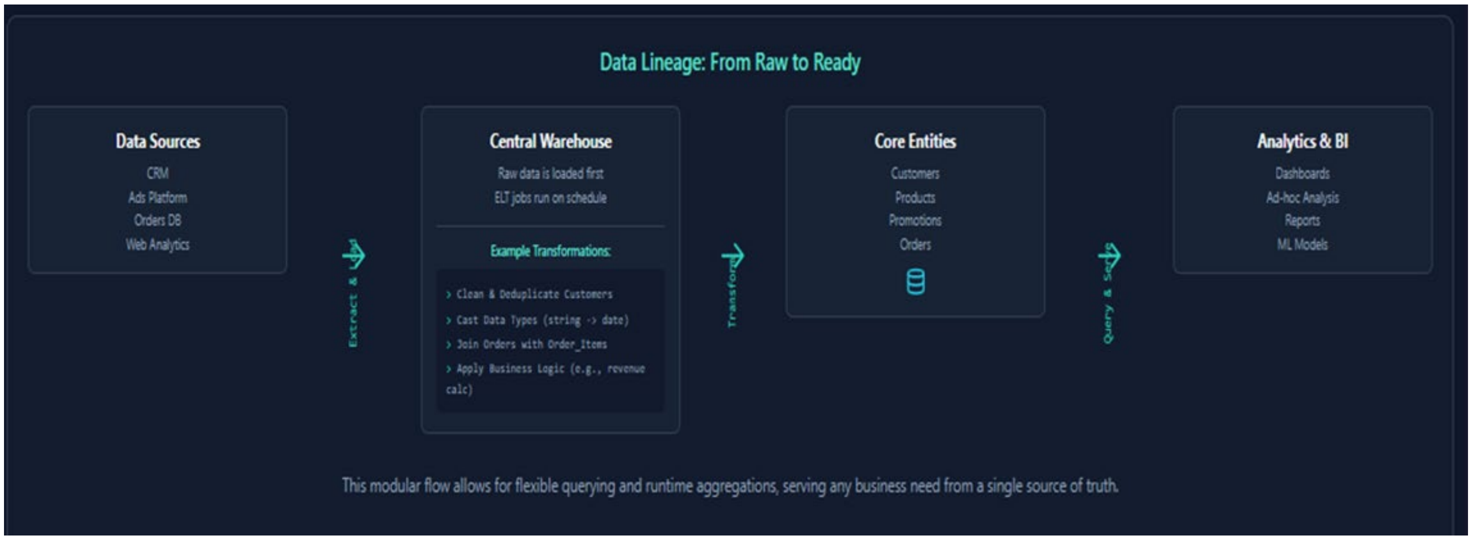
3. Let Architecture Follow Your Thinking

This isn't a talk about the Medallion architecture, or any other buzzword. This is about how we approach and think about complex data problems. Once you have a clear framework and mindset centered on domains and entities, the right architecture will naturally fall into place to support it.

4. This is an Iterative, Not All-or-Nothing, Process

You do not need to rebuild your entire data warehouse at once. This approach can be built iteratively. Start by modeling a few key entities and use them as the source for new requests. You can scale bit by bit, proving value at each step.

Whether you move to a fully domain-driven development method or simply use these concepts as a way to think through your requests, there is immense value in this approach.



This modular flow allows for flexible querying and runtime aggregations, serving any business need from a single source of truth.

The Payoff: Faster, Smarter Decisions

How to Implement a Governance Approach for Entity-Based Data

1. Metric Definition Standards
- Create centralized metric dictionary
- Define calculation methods
- Establish clear ownership for each metric
- Standardize naming conventions

2. Entity Ownership
- Assign domain experts to each core entity
- Define clear responsibilities
- Create cross-functional review process
- Establish change management protocols

3. Data Quality Controls
- Implement automated data validation
- Set up consistency checks
- Monitor metric variations
- Create alerting mechanisms for unexpected changes

4. Documentation Requirements
- Maintain comprehensive entity documentation
- Track data lineage
- Document source systems
- Create clear transformation rules

5. Versioning Strategy
- Implement semantic versioning for entities
- Track historical changes
- Maintain backward compatibility
- Create deprecation processes for outdated models

6. Access and Security
- Role-based access controls
- Implement data masking
- Audit entity access
- Ensure compliance with data privacy regulations

This change in methodology is transformative, empowering faster and smarter data-driven decisions across the organization:

- **Revisiting the CEO's Question:** Remember the initial request for conversion rate by acquisition channel? In the traditional model, this took weeks of complex integration and custom joins. With the new entity-based approach, it becomes a **simple query** combining the Orders and Sessions entities, reducing time to insight from weeks to **minutes**. This agility empowers analysts to deliver timely, accurate answers every time.

- **A Unified Foundation:** Every department—Marketing, Sales, Finance, and Product—is now pulling metrics from the same trusted source, establishing a single, undisputed source of truth.

What's Next? Building Trust and Maturity

The shift to an entity model is the crucial first step. To further leverage this unified data asset for strategic advantage, future enhancements should focus on maximizing usability and reliability:

- **Proactive Anomaly Monitoring:** Implement automated monitoring over your core entities to flag unexpected changes in data quality or business trends immediately.

- **Implementing a Semantic Layer:** Enhance the business-IT alignment by defining all key metrics and relationships in a single semantic layer (like the one provided by Visible Systems). This allows business users to query data using familiar language (e.g., "Monthly Revenue") instead of complex SQL.

- **Building Trust with a Data Catalog:** Documenting and classifying all core entities and their dimensions in a data catalog ensures easy discoverability and usage, fostering data literacy across the organization.

The end-to-end pipeline approach creates multiple tables with similar metrics calculated differently, leading to confusion about which number is correct. The entity-based modeling approach builds a semantic layer with standardized entities that can be joined and aggregated as needed.

By building these core entities with clear definitions and attributes, the team aims to:

- Reduce metric divergence
- Create a single source of truth
- Allow flexible querying and runtime aggregation
- Minimize duplicate or conflicting data representations

FOR MORE INFORMATION EMAIL US AT

contact@VisibleSystemscorp.com

OR VISIT OUR WEBSITE AT

https://www.visiblesystemscorp.com

Thank you for exploring the path toward transforming data analysis from a reactive bottleneck into a proactive, strategic advantage.

Let's take the next steps together in transforming your data analysis.

THE NEXT SECTION LOOKS AT A RETAIL BUSINESS AND EXPLAINS HOW TO IMPLEMENT A DOMAIN, ENTITY-CENTRIC APPROACH TO DATA ANALYTICS.

A New Perspective

Analytics Begin with an Understanding of the Business

Achieving a unified semantic layer requires recording key business definitions and data-centric business rules directly within the semantic model, alongside technical database schemas, procedures, and other data. This alignment guarantees that implementations are directly connected to the organization's strategic needs. The statement, **"Analytics begin with an understanding of the business,"** is not just a best practice—it is the prerequisite for building a scalable and trustworthy data architecture. Without this foundational understanding, data teams are reduced to reactive mechanics, building pipelines that are technically sound but strategically worthless.

1. Defining the Domain (The "Nouns")

Effective analytics start by identifying the fundamental building blocks of the organization, often called **Core Entities** or **Business Domains**.

Instead of asking, *"Where is the data for this report?"* we must ask, *"What are the key nouns of our business?"* These nouns—such as **Customers, Products, Orders, and Promotions**—form the shared language of the company. When you build your data model around these stable concepts, you create a semantic foundation that is universally understood.

2. Ensuring Strategic Alignment

In a reactive environment, technical solutions often solve a narrow data problem but miss the strategic goal. By understanding the business first, analysts ensure:

- **Metric Consistency:** You define "Revenue" or "Active User" once, aligning the technical calculation with the finance or product team's business definition. This is the **Single Source of Truth (SSOT)** that eliminates arguments over conflicting numbers.
- **Prioritization:** You focus resources on modeling the entities that drive the most critical business functions (e.g., modeling the *Orders* entity first, as it touches Sales, Finance, and Product).
- **Context over Raw Data:** You understand *why* certain data points are captured, allowing you to clean, combine, and classify raw data into meaningful business attributes (e.g., transforming a raw timestamp into an "Order Date" dimension).

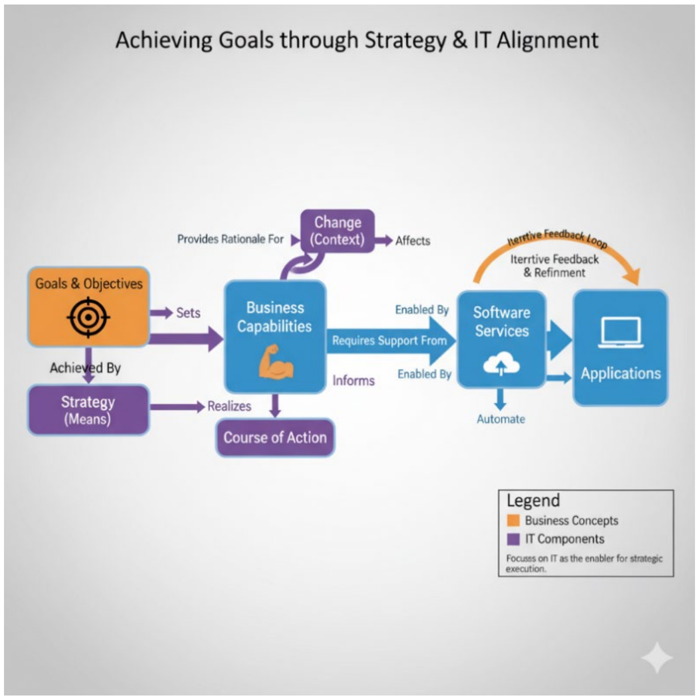
3. Shifting from Reactive to Proactive

When the business domain is the starting point, the analytics team gains agility. Instead of responding to every request by building a new, fragile **end-to-end pipeline**, the analyst simply combines existing, trusted **Core Entities** to answer the question.

THE SHIFT TRANSFORMS the analyst's role from a **data engineer** focused on moving and transforming data, into a **strategic consultant** focused purely on **insight creation**. It allows the team to build for the long-term solution, rather than just the immediate, temporary ask.

This foundational approach enables the modular, reusable data insights we discussed, making your data ready for any future business question.

Let's break it down from the business perspective:



- **Starting Point:** Value Stream & Strategy
  - Your business likely has a **Value Stream** (the sequence of activities your company performs to deliver value to customers). This value stream targets certain **Goals**.
  - Your overall **Strategy** incorporates these goals and achieves them.
- **Setting Your Sights:** Goals & Objectives
  - Your **Goals** are the broad achievements you aim for (e.g., increase market share, improve customer satisfaction).
  - These goals are broken down into more specific, measurable **Objectives**. An objective might be "Reduce customer service response time by 20%."
- **Making It Happen: Strategy (Course of Action)**
  - To achieve your objectives, you define a **Strategy** – the specific steps or initiatives your team will undertake.
  - Implementing these courses of action often involves **Change** within the organization. This change provides the rationale for developing new objectives or modifying existing ones, and it also affects your capabilities.
- **The Bridge to IT:** Capability
  - This is where IT becomes vital. Your objectives and courses of action highlight the **Capabilities** your business *needs* to possess or enhance. For example, to reduce customer service response time, you need the *capability* for efficient ticket management and real-time communication.
  - **Capability** is the central link. It's derived from your software services and automates your applications.
- **The IT Angle: Software Service & Application**
  - These capabilities are often **encapsulated** by **Software Services**. Think of a "customer relationship management service" or a "data analytics service."

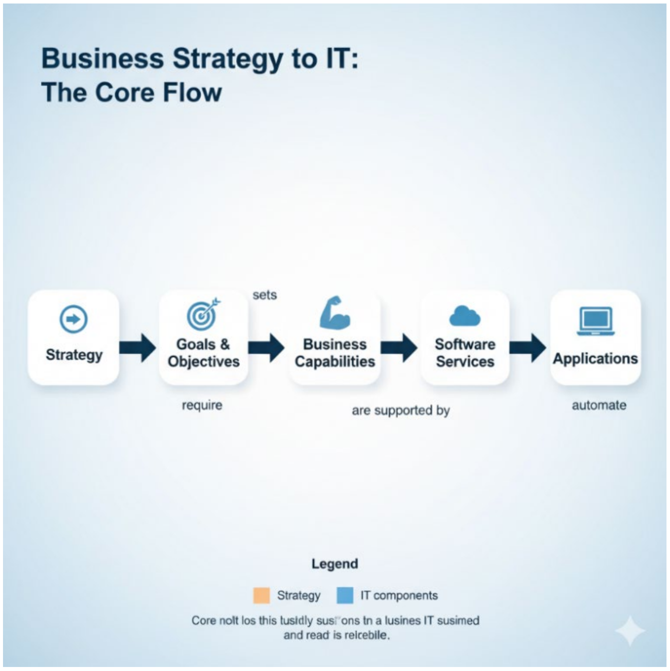
- These software services, in turn, **automate** specific **Applications** (like your CRM software, your website, or your internal reporting tools). These applications **contain** actual functionalities.

In essence, as a business manager, the diagram below tells you:

1. Your strategy is dictated by your goals and objectives.
2. To achieve these objectives, you need specific business capabilities.
3. These capabilities are directly supported and often enabled by your IT landscape – the software services and applications you use.
4. Any strategic change or new course of action you envision will likely have a direct impact on the capabilities you need and, consequently, on your IT architecture.

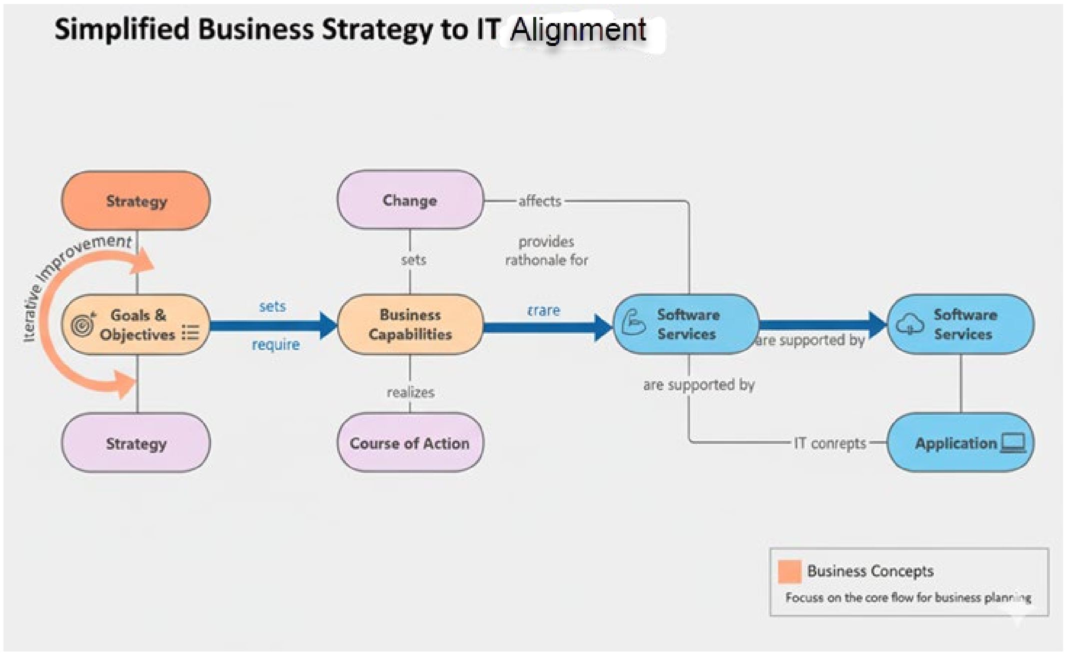
Therefore, when you're planning new initiatives, always consider the capabilities required and how your existing or future IT systems will support them. IT isn't just a cost center; it's an enabler of your business strategy and capabilities.

Here's a simplified visual representation of the core flow for a business manager:



BUILDING A DATA MODEL that links or cross-references the data upon which the strategy depends is essential for you to know whether you are on track to achieve your goals and objectvies. In doing so, you connect the high-level capabilities and software services to the actual information entities that need to be managed.

In our Retail Business example, I'll provide a **conceptual/logical data model** focusing on the main entities and their relationships. This isn't a physical database schema with all columns and data types, but rather a representation of the key business data elements.

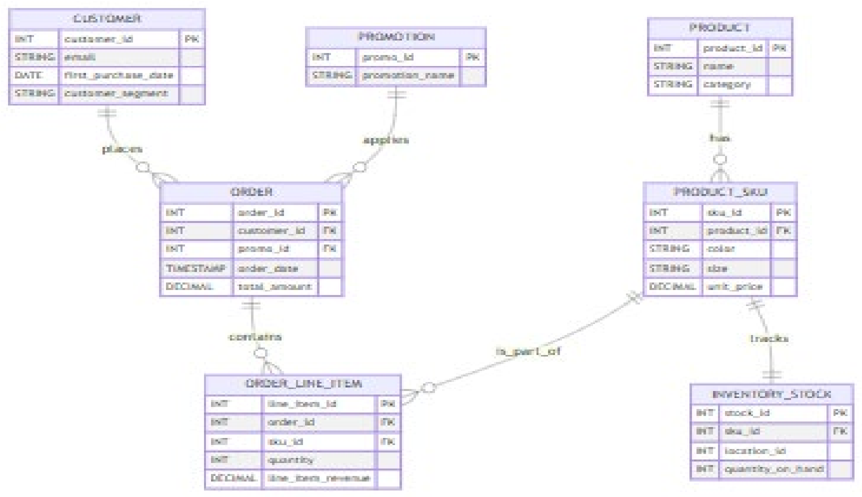


Conceptual Data Model for Fashion Retailer **(Retail Business Example)**

This example aligns perfectly with the principle of building reusable **Core Entities**. For a Direct-to-Consumer (DTC) Fashion Retailer, the data model must center around the customer journey and product lifecycle. This data model outlines the major entities and how they relate, supporting the retailer's "Direct-to-Consumer Digital First" strategy. This diagram confirms the key relationships that enable flexible, entity-based analytics. A visual Entity Relationship Diagram (ERD) is the best way to summarize the complexity of the domain model.

- Customers place Orders (1:M) for LTV and segmentation.
- Orders contain many Order Line Items (1:M), representing the transaction fact.
- Product SKUs (the specific variant) link to Line Items and Inventory (1:1/1:M) for granular sales and stock tracking.
- Promotions can apply to many Orders (1:M) for campaign analysis.

This data model provides a structured view of the information the "Fashion Retailer" needs to manage. It connects customer profiles with their orders, the products they buy, how those products are managed in inventory, and how interactions are used for targeted marketing. This foundation is crucial for supporting their e-commerce operations, personalizing the customer experience, and achieving their digital-first growth objectives.



The model uses the entity-based approach to ensure that a question like "What was the revenue of Large Blue T-Shirts sold to VIP Customers via the Mobile App last quarter?" can be answered by combining the CUSTOMER, PRODUCT_SKU, and ORDER entities through their defined relationships (PK-FK joins).

These entities capture the facts of the sale.

Entity	Attribute	Role	Data Type	Description
Order	order_id	PK	INT	Unique transaction identifier.
Order	customer_id	FK (CUSTOMER)	INT	Link to the purchasing customer.
Order	promo_id	FK (PROMOTION)	INT	Link to the promotion used (if any).
Order	order_date	DIM	TIMESTAMP	Date and time the order was placed.
Order	channel	DIM	STRING	Sales channel (e.g., Web, Mobile App).
Order	total_amount	MEASURE	DECIMAL	Gross total charged to the customer.
Order Line Item	line_item_id	PK	INT	Unique identifier for this line item.
Order Line Item	order_id	FK (ORDER)	INT	Link to the parent order.
Order Line Item	sku_id	FK (PRODUCT_SKU)	INT	Link to the specific item variant sold.
Order Line Item	quantity	MEASURE	INT	Number of units sold.
Order Line Item	line_item_revenue	MEASURE	DECIMAL	Revenue generated by this line item.
INVENTORY_STOCK	stock_id	PK	INT	Unique inventory record identifier.
INVENTORY_STOCK	sku_id	FK (PRODUCT_SKU)	INT	Link to the tracked item variant.
INVENTORY_STOCK	location_id	DIM	INT	ID of the warehouse or fulfillment center.
INVENTORY_STOCK	quantity_on_hand	MEASURE	INT	Real-time stock count.
INVENTORY_STOCK	last_restock_date	DIM	DATE	When inventory was last updated.

Core Entities:

1. **Customer:** An individual who shops at the retailer.
    - *Attributes:* Customer ID, First Name, Last Name, Email, Phone, Address, Account Creation Date, Loyalty Program Status, Marketing Opt-in Flag.
2. **Product:** The fashion items the retailer sells.
    - *Attributes:* Product ID, SKU, Name, Description, Brand, Category (e.g., Dresses, Jeans, Shoes), Gender (e.g., Women's, Men's), Price, Color, Material, Size.
3. **Inventory (Stock Item):** A specific quantity of a Product in a particular variant and location.
    - *Attributes:* Inventory ID, Product ID (FK), Size, Color (if not part of Product ID), Location ID (FK), Quantity On Hand, Quantity Reserved, Reorder Level.
4. **Location (Store/Warehouse):** Physical places where products are stored or sold.
    - *Attributes:* Location ID, Name, Address, Type (e.g., Retail Store, E-commerce Warehouse, Return Center), Store Hours (if retail).
5. **Order:** A customer's request to purchase one or more products.
    - *Attributes:* Order ID, Customer ID (FK), Order Date, Total Amount, Shipping Address, Billing Address, Payment Method, Status (e.g., Pending, Shipped, Delivered, Canceled), Discount Applied.
6. **Order Line Item:** Details of each product within an Order.
    - *Attributes:* Order Line ID, Order ID (FK), Product ID (FK), Quantity, Price at Sale, Size, Color.
7. **Shipment:** The physical delivery of an Order (or part of an Order).
    - *Attributes:* Shipment ID, Order ID (FK), Carrier ID (FK), Tracking Number, Shipment Date, Estimated Delivery Date, Actual Delivery Date, Shipping Cost, Status.
8. **Carrier:** The shipping company handling deliveries.
    - *Attributes:* Carrier ID, Name, Contact Info, Service Level.
9. **Return:** A customer returning purchased items.
    - *Attributes:* Return ID, Order ID (FK), Return Date, Reason, Status (e.g., Initiated, Received, Refunded), Refund Amount.
10. **Customer Interaction / Event:** Any touchpoint or action by the customer (for personalization/marketing).
    - *Attributes:* Interaction ID, Customer ID (FK), Timestamp, Type (e.g., Website Visit, Product View, Cart Add, Email Open, Purchase), Details (e.g., URL, Product ID, Email Subject).
11. **Marketing Campaign:** A planned marketing effort.
    - *Attributes:* Campaign ID, Name, Start Date, End Date, Budget, Target Audience, Type (e.g., Email, Social Media, Ad).
12. **Customer Segment:** Groups of customers with similar characteristics (for targeted marketing).
    - *Attributes:* Segment ID, Name, Description, Criteria.

Key Relationships **(Conceptual)**

- **Customer** places **Orders**. (1-to-Many)
- **Customer** has multiple **Customer Interactions**. (1-to-Many)
- **Customer** can belong to one or more **Customer Segments**. (Many-to-Many)
- **Order** contains multiple **Order Line Items**. (1-to-Many)
- **Order** can result in one or more **Shipments**. (1-to-Many)
- **Order** can have one or more **Returns**. (1-to-Many)
- **Order Line Item** refers to a **Product**. (Many-to-1)
- **Shipment** is handled by a **Carrier**. (Many-to-1)
- **Inventory** tracks **Products** at a **Location**. (Many-to-1 for both)
- **Marketing Campaign** targets **Customer Segments**. (Many-to-Many)
- **Customer Interaction** might be part of a **Marketing Campaign**. (Many-to-1, or Many-to-Many if a single interaction contributes to multiple campaigns).

Customers can place multiple Orders, each Order can result in several Shipments, and each Shipment is handled by a Carrier.

- Customer
- - Customer ID, Name, Email, ...
- Order
- - Order ID, Customer ID, Order Date, ...
- Shipment
- - Shipment ID, Order ID, Carrier ID, ...
- Carrier
- - Carrier ID, Name, ...

Orders may have Returns associated with them. Customer Interactions are also tracked. Each Order consists of multiple Order Line Items, each referring to a specific Product.

- Return
- - Return ID, Order ID, ...
- Interaction
- - Interaction ID, Customer ID, ...
- Order Line Item
- - Order Line ID, Order ID, Product ID, ...

Products are managed in Inventory, which tracks the quantity of each Product at various Locations.

- Product
- - Product ID, SKU, Name, Size/Color, ...
- Inventory
- - Inventory ID, Product ID, Location ID, Quantity

- Location
- - Location ID, Name, Address, Type

Marketing Campaigns target Customer Segments, supporting personalized marketing efforts and data-driven strategies.

- Marketing Campaign
- - Campaign ID, Name, ...
- Customer Segment
- - Segment ID, Name, ...

Entity	Attributes
Customer	Customer ID, Name, Email, ...
Order	Order ID, Customer ID, Order Date, ...
Shipment	Shipment ID, Order ID, Carrier ID, ...
Carrier	Carrier ID, Name, ...
Return	Return ID, Order ID, ...
Interaction	Interaction ID, Customer ID, ...
Line Item	Order Line ID, Order ID, Product ID, ...
Product	Product ID, SKU, Name, Size/Color, ...
Inventory	Inventory ID, Product ID, Location ID, Quantity, ...
Location	Location ID, Name, Address, Type
Marketing Campaign	Campaign ID, Name, ...
Customer Segment	Segment ID, Name, ...

Beyond Core (Principal) Entities: Type, Secondary, Structure, Intersecting, Role.

What is a "Type" Entity?

In the domain-entity perspective, a "type" entity is a structural model that provides **context, classification, and definition** to a core entity. It is usually a static or slowly changing set of values (a dimension) that you use to slice and filter your data (the facts).

The goal of a "type" entity is to make the core entity immediately useful for analysis.

Example: Transforming an Order into an Order Type

Entity Type	Description	Purpose in Analysis
Core Entity (Fact)	Orders (Contains transactional data like order_id, revenue, timestamp).	**What happened?** (Used for measures/metrics)
Type Entity (Dimension)	Order Type (Contains context like purchase_source, payment_method, fulfillment_status).	**How, where, or when did it happen?** (Used for filtering and grouping)

When an analyst wants to answer: "What is the revenue breakdown by **payment method**?" the Payment Method is provided by the **Order Type** entity, which is then joined to the core **Orders** entity.

Common "Type" Entities in a Domain Model

In the context of the core entities mentioned in the files (Users, Orders, Products, Sessions), the corresponding "type" entities could include:

- **User Type:** Models the dimension of a user, such as User Segment, Account Status, or Subscription Tier.
- **Product Type:** Models classifications like Product Line, Category, or Color.
- **Session Type:** Models how a session originated, such as UTM Channel (organic, paid, social) or Device Type (mobile, desktop).

This structured approach ensures that all **classifications and attributes** are consistently applied, which is key to achieving the **Metric Consistency** and **Single Source of Truth** as mentioned in the Analytics Begin with Business Understanding section.

What is a "Secondary" Entity?

A Secondary Entity sits between two Core Entities to resolve a relationship where one Core Entity can be linked to multiple instances of another Core Entity, and vice versa. It acts as an **intersection** that captures the necessary context and metrics *at the point of interaction*.

The Role of the Secondary Entity

1. **Resolves Many-to-Many Relationships:** A single **Customer** can have many **Promotions**, and a single **Promotion** is applied to many **Customers**. A Secondary Entity is needed to record *which* customer used *which* promotion and *when*.
2. **Captures Granular Facts:** It stores the metrics (measures) and context (dimensions) specific to that unique relationship.

Practical Example: Promotions

In your model, you have two Core Entities: **Customers** and **Promotions**.

Entity Type	Example Content	Relationship
Core Entity A	**Customers** (e.g., User ID, Segment)	Many Customers...
Secondary Entity	**Customer_Promotion_Use**	...have many uses...
Core Entity B	**Promotions** (e.g., Promo ID, Type, Discount)	...of many Promotions.

The **Customer_Promotion_Use** Secondary Entity would contain key facts about the interaction:

- **Foreign Keys:** customer_id and promotion_id
- **Measures:** discount_amount_applied, order_count_with_promo
- **Dimensions:** date_used, channel_applied

By creating this Secondary Entity, you adhere to the principle of keeping your Core Entities clean and stable, as highlighted in **Analytics Begin with an Understanding of the Business** section. You can now join Customer_Promotion_Use to both the Customers entity and the Promotions entity to get a complete, non-redundant view of your marketing effectiveness.

In the context of the domain-oriented modeling approach we've been discussing, a "Structured Entity" is best understood as the fully refined, conformed, and complete **Core Entity** that resides in the **Gold Layer** of the architecture.

It represents the analytics team's finished product, ready for consumption by BI tools and analysts.

What is a "Structured" Entity?

A **Structured Entity** is the culmination of the cleansing and transformation process. It is characterized by three key traits:

1. Conformed and Unified

As noted in **A Domain-Oriented Approach to Analytics Modeling**, technical implementations often struggle to maintain a singular domain model across multiple platforms (e.g., Salesforce data vs. backend data). The Structured Entity solves this:

- It is the **conformed representation** of a Core Entity (like Customer) where attributes from multiple source systems have been reconciled, combined, and standardized.
- It establishes the **Single Source of Truth** for all metrics and dimensions related to that entity, eliminating data silos and consistency issues (as detailed in **Core Entities and Their Payoff**).

2. Analytics-Ready Grain

A Structured Entity is modeled at its natural, lowest level of detail (its grain). This ensures maximum flexibility for downstream analysis:

- It has undergone the rigorous cleaning and normalization from the Silver Layer (per **The Entity-Centric ELT Process).**
- It is *not* pre-aggregated for a specific report. Instead, it is ready to be joined and aggregated **at runtime**, enabling analysts to answer complex ad hoc questions quickly (Gold Layer philosophy).

3. Combining Facts and Dimensions

A Structured Entity typically represents the complete answer to "What is this thing and what happened?"

- It contains the **Measures (Facts)** that roll up to it (e.g., Revenue for an Order).
- It includes the necessary **Type Entity** attributes (Dimensions) for filtering and context (e.g., Payment Method, Acquisition Channel).

What is an "Intersecting" Entity?

An **"Intersecting Entity"** is effectively another name for a **Secondary Entity** (or sometimes called a *Junction Entity* or *Association Entity*). Its purpose is fundamental to maintaining a clean and accurate data model, especially in the **Silver Layer** (as defined in A Domain-Oriented Approach to Analytics Modeling).

Here is a breakdown of what an Intersecting Entity is and why it's necessary:

What is an Intersecting Entity?

An Intersecting Entity is a specific model designed to handle **many-to-many relationships** between two or more **Core Entities**.

When you have two Core Entities, for example, a **Customer** and a **Product**—that can be linked to each other multiple times (one customer buys many products, and one product is bought by many customers), you cannot cleanly model that relationship in either of the core tables.

The Intersecting Entity is created to sit exactly at that **intersection** of the relationship, capturing the facts and context *only* related to that specific connection.

Core Functions and Attributes

1. **Resolves Complexity:** It prevents data redundancy and distortion by separating the facts of the relationship from the attributes of the core objects.
2. **Captures Metrics at the Join:** It is the ideal place to store metrics and context that are only relevant when the two entities meet.

Example: Order_Line_Item

Instead of trying to store Product details on the Order entity, or Order details on the Product entity, you create an Intersecting Entity:

Entity Type	Example Entity	Data Captured
Core Entity A	Orders	Order Date, Customer ID, Total Revenue
Intersecting Entity	**Order_Line_Item**	**Order ID, Product ID**, Quantity Purchased, Unit Price
Core Entity B	Products	Product Category, Cost of Goods Sold (COGS), Color

By creating the **Order_Line_Item** Intersecting Entity, you can now accurately calculate metrics like Product Net Revenue (mentioned in A Domain-Oriented Approach to Analytics Modeling, by joining the Orders (for the customer context) and the Products (for COGS) **through** the Intersecting Entity.

This approach ensures that your Core Entities remain **stable, consistent, and reusable**, serving as robust building blocks for any metric the business needs.

What is a "Role" Entity?

The concept of a **"Role" Entity** is a modeling technique designed to keep your **Core Entities** clean and stable, especially when you are interacting with the business in many different ways.

It is a specialized type of entity that separates the universal attributes of a **Core Entity** from the attributes that are unique to its specific **contextual function** in a process.

Defining the Role Entity

The primary purpose of a Role Entity is to resolve situations where a single **Core Entity** (like a Product or Customer) can take on **multiple, distinct functions** across various business processes.

If you were to store all the attributes required for every possible function on the Core Entity, that entity would become bloated, complex, and prone to inconsistent updates. The Role Entity provides a dedicated structure for these role-specific facts.

Core Entity vs. Role Entity

Let's use the **User** entity, a fundamental noun in your model (as defined in A Domain-Oriented Approach to Analytics Modeling):

Entity Type	Example Attributes	Purpose
Core Entity: User	User ID, Signup Date, Primary Email, Total Lifetime Value	Stores attributes that are **universal** to the person or account, regardless of how they are currently interacting with the system.
Role Entity: Affiliate	Affiliate ID, Commission Rate, Payout Method, Last Referral Date	Stores attributes relevant **only** to the user's function as a marketing affiliate.
Role Entity: Vendor	Vendor ID, Warehouse Address, Inventory Count, SLA Agreement Tier	Stores attributes relevant **only** to the user's function as a supplier or vendor.

Why This Approach is Critical

By isolating the "role," the domain model adheres to the principles of a **Single Source of Truth** and **reusability** (from Core Entities and Their Payoff):

1. **Cleaner Core Entity:** The central User entity remains focused on core identity facts. Any team needing basic user info can pull it without wading through hundreds of role-specific fields.
2. **Increased Agility:** If Finance needs to change how vendor commissions are calculated, they only need to update the logic in the **Vendor** Role Entity. The core User entity and the separate **Affiliate** Role Entity remain untouched, preventing fragile pipeline dependencies (as discussed in The Pitfalls of Traditional Pipelines).
3. **Contextual Analysis:** The analyst can easily combine the core facts (the User) with the role facts (the Affiliate) and the transactional facts (the Orders entity) to produce highly contextual, accurate insights (e.g., "What is the lifetime value of users who also serve as vendors?").

Using this knowledge let's revisit our Retail Business.

A New Business Venture – Retail Business Contract Award

You run a garments manufacturing company and just recently were awarded a contract to supply a new line of clothing to the retail business in our subsequent case study.

To meet the demand and schedule, you employ a clothes designer and a team of cutters and sewers. The contract awarded to you requires that you supply ladies' and men's wear to various boutique stores run by the fashion retail business.

You subcontracted with an offshore firm to sew children's wear for export. The subcontractor is responsible for sourcing all materials and designs for this new line.

A Domain-Oriented, Entity-Cenric Approach to Analytics

The statement indicates different **types** of workers: sewers, cutters, designers – these are secondary entities of worker (as principal entity).

There are also different **types** of garments: ladies' wear, men's wear, children's wear – all secondary entities of Garment.

Workers make garments: Garment Worker is shown as an **intersecting** entity.

Customers are implied: they are Boutiques and Department Stores. A valid solution would be to show these as **secondary** entities under Customer (as the principal entity) and in a **role** entity.

Customers are interested in Products: this is shown by the **intersecting** entity Client Product.

Product is a principal entity: Material and Garment are **secondary** entities, controlled by *Product Type*. *Garment* is thus called a "*typed secondary*" – as it has *secondary* entities under it.

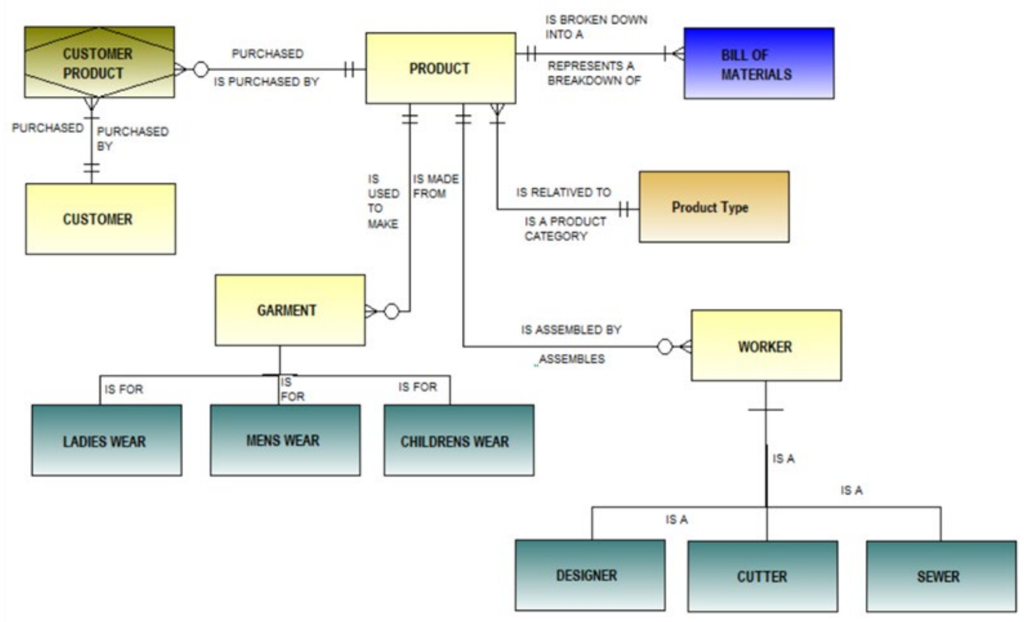
Product Bill of Material is a **structure** entity. It enables any garment to be specified so that it can be manufactured from any material.

New Business Venture Data Model

The following is one suggested solution. Other solutions may be valid depending on the specific business needs.

LEGEND

🟧	Type Entity
🟨	Principal Entity
🟩	Secondary Entity
🟦	Structure Entity
🟫	Intersecting Entity
⬜	Role Entity



The Payoff of Materialized Views using an Entity-based Model

This entity-based model makes answering complex business questions a simple matter of combining the correct "building blocks." We keep all entities modeled at their independent grain (their lowest level of detail) and join them as needed, leveraging strong compute power to aggregate at runtime.

Here are example aggregations.

These aggregations are accomplished through Materialized Views (MVs) which are pre-computed, optimized tables that bring together data from multiple Core and Intersecting Entities, ready for fast querying by business users.

Based on the provided Entity-Relationship (ER) diagram for a garment manufacturing and customer purchase system, here are three examples of materialized views that would be created to serve common business questions.

MVs are a crucial component of the modern, entity-centric data architecture, specifically serving as the finished, optimized **Structured Entities** in the **Gold Layer**. They provide significant benefits by acting as pre-computed, durable assets that move complexity and processing time away from the moment of analysis.

Here are the primary benefits that MVs provide within this entity-based framework:

---

1. Drastically Improved Query Performance

The single biggest advantage of a Materialized View is speed.

- **Pre-Aggregation:** MVs consolidate data from multiple **Core Entities** and **Intersecting Entities** (which often require complex joins) into a single, denormalized table. The heavy lifting of joins, filters, and calculations is done once, on a schedule (e.g., nightly or hourly), rather than every time an analyst runs a report.
- **Reduced Time to Insight:** By serving pre-calculated data, MVs allow analysts to get complex answers in **seconds** instead of minutes or hours, which is critical for **real-time insights** and timely decision-making.

2. Establishing a Single Source of Truth (SSOT)

MVs enforce data consistency by centralizing the definition of key metrics.

- **Metric Consistency:** When metrics like "Monthly Recognized Revenue" are built into a materialized view, they are defined and calculated the same way for every user and every dashboard. This directly addresses the issue of data silos and inconsistency described in The Pitfalls of Traditional Pipelines.

- **Trust and Reliability:** If everyone is pulling numbers from the same, reliable MV, it eliminates arguments over "whose numbers are right," fostering trust in the data (a goal highlighted in the Conclusion: The Future is Modular section on "Building Trust").

3. Optimizing Resource Usage

Materialized Views are a resource-efficient way to serve data-a- scale.

- **Reduced Compute Cost:** Instead of having dozens of analysts running complex, costly joins on raw data simultaneously, the compute power is used once to build the MV. Downstream queries then hit the MV, consuming far fewer resources.
- **Decoupling:** MVs allow you to decouple the query layer from the core transformation layer. This ensures that changes to the underlying Bronze or Silver data models do not immediately break downstream reports, providing stability and reducing engineering overhead (per Core Entities and Their Payoff).

4. Supporting Data Democratization

MVs are perfectly structured for business users to consume data easily.

- **Simplified Access:** Because the MV is pre-joined and optimized, business users can leverage self-served tools (like the visual query builder mentioned in Rewritten Benefits Summary without needing to understand complex SQL or how the original entities relate to one another.
- **Analytics-Ready Grain:** MVs present data in a structure that directly matches common business questions, moving the focus away from **data engineering** and toward **insight creation** (as discussed in The Entity-Centric ELT Process.

THE PAYOFF: FASTER, SMARTER DECISIONS
THIS CHANGE IN METHODOLOGY IS TRANSFORMATIVE, EMPOWERING FASTER AND SMARTER DATA-DRIVEN DECISIONS ACROSS THE ORGANIZATION:

Examples of Materialized Views (Gold Layer Entities)

1. MV: Daily_Customer_Product_Sales

This view answers common retail and finance questions related to sales performance by aggregating customer and product data, eliminating the need to join transactional tables every time a report is run.

- **Business Question:** *What were the total daily sales volume and revenue, segmented by **Product Category** and **Customer Type**?*

- **Source Entities Joined:**
    - **Core Entity:** CUSTOMER (for customer attributes/types)
    - **Intersecting Entity (Fact):** CUSTOMER_PRODUCT (the transaction/sale)
    - **Type Entity (Dimension):** PRODUCT_TYPE (for categorization)
- **Key Fields (Dimensions & Measures):**
    - Sale_Date
    - Product_Category (from PRODUCT_TYPE)
    - Customer_Segment (from CUSTOMER)
    - Total_Units_Sold (Aggregated measure)
    - Gross_Revenue (Aggregated measure)

2. MV: Product_Manufacturing_Cost

This view serves the operations and finance teams by pre-calculating the cost structure of each item, making Profit & Loss (P&L) analysis instantaneous.

- **Business Question:** *What is the full Cost of Goods Sold (COGS) for each **PRODUCT** based on the labor and materials involved?*
- **Source Entities Joined:**
    - **Core Entity:** PRODUCT (the final item)
    - **Intersecting Entity:** BILL_OF_MATERIALS (details of raw inputs)
    - **Intersecting Entity:** WORKER (details of assembly labor)
    - **Core Entity:** GARMENT (the specific item being made)
- **Key Fields (Dimensions & Measures):**
    - Product_ID
    - Garment_Type (e.g., Ladies Wear, Mens Wear)
    - Total_Material_Cost (Aggregated from BILL_OF_MATERIALS)
    - Total_Labor_Cost (Aggregated from WORKER time)
    - COGS_Total (Calculated measure: Material + Labor)

3. MV: Labor_Efficiency_Metrics

This view serves HR and operations by tracking the productivity and skill level of the workforce involved in production.

- **Business Question:** *What is the monthly productivity of each **WORKER** based on their **ROLE** (Designer, Cutter, Sewer) and the **GARMENT** type they worked on?*
- **Source Entities Joined:**

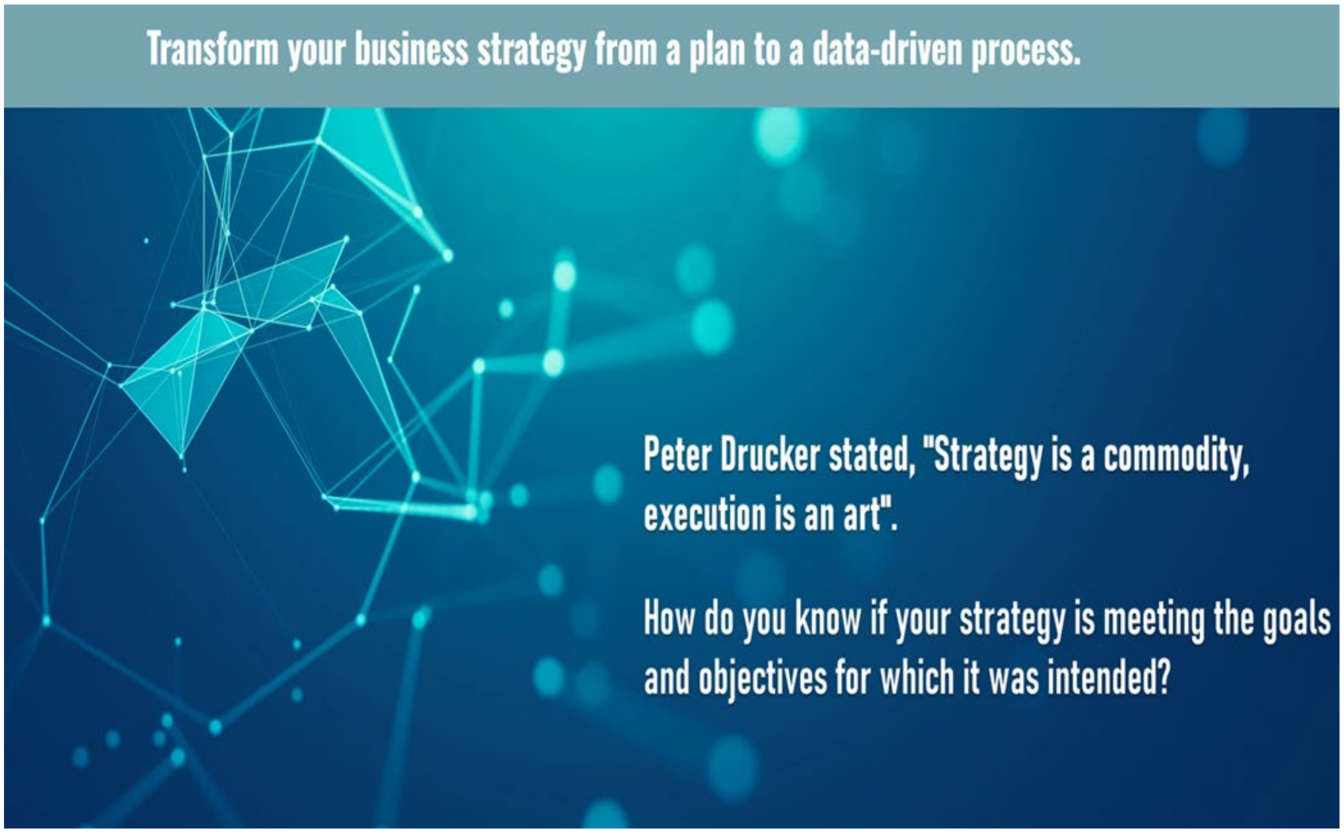
    - **Core Entity:** WORKER (for worker attributes/roles)
    - **Intersecting Entity:** PRODUCT (the assembly link)
    - **Type Entity (Dimension):** GARMENT (the item category)
- **Key Fields (Dimensions & Measures):**
    - Worker_ID
    - Worker_Role (from the Worker sub-types: Designer, Cutter, Sewer)
    - Garment_Category
    - Total_Products_Assembled (Aggregated measure)
    - Average_Assembly_Time (Aggregated measure)

These **Materialized Views** exemplify how the entity-based approach shifts the analytical focus: instead of running complex, multi-join queries against raw data, analysts now query these stable, pre-built **Entities** for instant, trustworthy insights.

Think of it as a handshake between technology and business.

HERE IS THE LINK TO A BRIEF DEMO

HTTPS://YOUTU.BE/KYXAEPNLNHM



Transform your business strategy from a plan to a data-driven process.

Peter Drucker stated, "Strategy is a commodity, execution is an art".

How do you know if your strategy is meeting the goals and objectives for which it was intended?

Visible®

Visualize. Align. Transform.™

Visualize patterns, align strategy and transform change into meaningful business outcomes.

Visible Systems Corporation
24 School Street 2nd floor
Boston, MA 02108
(+1) 617-902-0767