

# UDM Framework Core Domains

## Universal Schema Domains and Data Models

### Introduction

The UDM Framework organizes data into logical domains that represent universal business concepts. These domains are industry-agnostic, meaning they adapt to any business model while maintaining consistent patterns and relationships. This document details each core domain, its purpose, key entities, and cross-domain relationships.

## Domain Architecture Overview

### The Seven Core Domains

```
graph TB
    subgraph Sales["1. SALES DOMAIN"]
        Orders[Orders]
        Leads[Leads]
        Invoices[Invoices]
        Transactions[Transactions]
    end

    subgraph Finance["2. FINANCE DOMAIN"]
        Payments[Payments]
        Revenue[Revenue Recognition]
        Invoicing[Invoicing]
    end

    subgraph Product["3. PRODUCT DOMAIN"]
        Products[Products]
        ProductGroups[Product Groups]
        Offers[Offers]
    end
```

```
subgraph Contact["4. CONTACT DOMAIN"]
    Customers[Customers]
    Users[Users]
    Households[Households]
    Organizations[Organizations]
end

subgraph Marketing["5. MARKETING DOMAIN"]
    Campaigns[Campaigns]
    Channels[Channels]
    Publishers[Publishers]
    AdSpend[Ad Spending]
end

subgraph Reference["6. REFERENCE DOMAIN"]
    SourceSystem[SourceSystem]
    ColumnMapping[ColumnMapping]
    PropertyRef[PropertyReference]
    Lookups[Lookup Tables]
end

subgraph Fallout["7. FALLOUT DOMAIN"]
    FalloutTable[Fallout Table]
    ErrorCodes[Error Codes]
    Monitoring[Monitoring Views]
end

Sales --> Finance
Sales --> Product
Sales --> Contact
Sales --> Marketing
Finance --> Contact
Product --> Sales
Contact --> Sales
Marketing --> Sales

Reference -.->|Configures| Sales
Reference -.->|Configures| Finance
Reference -.->|Configures| Product
Reference -.->|Configures| Contact
Reference -.->|Configures| Marketing
```

```
Fallout -.->|Tracks Quality| Sales
Fallout -.->|Tracks Quality| Finance
Fallout -.->|Tracks Quality| Product
```

```
style Sales fill:#e3f2fd
style Finance fill:#e8f5e9
style Product fill:#fff3e0
style Contact fill:#f3e5f5
style Marketing fill:#fce4ec
style Reference fill:#fff9c4
style Fallout fill:#ffebee
```

# 1. Sales Domain

## Purpose

The Sales domain captures all revenue-generating transactions and customer interactions that lead to sales. It's the heart of the business data warehouse, tracking orders, leads, invoices, and the entire sales lifecycle.

## Key Entities

### 1.1 Orders

**Primary Table:** `Sales.Order`

**Purpose:** Core transactional records of customer purchases/orders

**Key Columns:**

```
CREATE TABLE [Sales].[Order] (
    [OrderId] [bigint] IDENTITY(1,1) NOT NULL,           -- Surrogate key
    [OrderNK] [nvarchar](100) NOT NULL,                 -- Natural key
    [OrderDate] [datetime] NULL,                        -- When order placed
    [CustomerId] [bigint] NULL,                         -- FK to Contact
    [ProductId] [bigint] NULL,                          -- FK to Product
    [CampaignId] [bigint] NULL,                        -- FK to Marketing
    [OrderStatusId] [int] NULL,                        -- FK to OrderStatus
```

```

[Amount] [decimal](18,2) NULL,           -- Order value
[ValidFromDate] [date] NULL,             -- Timeslice start
[ValidToDate] [date] NULL,              -- Timeslice end
[SourceSystemId] [bigint] NOT NULL,      -- Source traceability
[SourceSystemIdentifier] [nvarchar](100) NULL, -- Source PK
CONSTRAINT [PK_Order] PRIMARY KEY ([OrderId])
)

```

## Companion Tables:

- `Sales.OrderProperties` : EAV for flexible order attributes
- `Sales.OrderStatusHistory` : Track status changes over time

## Industry Examples:

- **E-Commerce**: Product purchases
- **Price Comparison**: Comparison service orders
- **Subscriptions**: Subscription activations
- **B2B**: Service contracts

## 1.2 Leads

**Primary Table:** `Sales.Lead`

**Purpose:** Potential customers or sales opportunities before conversion

### Key Columns:

```

CREATE TABLE [Sales].[Lead] (
    [LeadId] [bigint] IDENTITY(1,1) NOT NULL,
    [LeadNK] [nvarchar](100) NOT NULL,
    [LeadDate] [datetime] NULL,
    [ContactId] [bigint] NULL,           -- FK to Contact
    [ProductId] [bigint] NULL,         -- Product of interest
    [LeadSourceId] [int] NULL,         -- Where lead came from
    [LeadStatusId] [int] NULL,        -- Current status
    [ConvertedToOrderId] [bigint] NULL, -- Conversion to order
    [ValidFromDate] [date] NULL,
    [ValidToDate] [date] NULL,
    [SourceSystemId] [bigint] NOT NULL,

```

```
CONSTRAINT [PK_Lead] PRIMARY KEY ([LeadId])  
)
```

### Use Cases:

- Marketing funnel analysis
- Conversion rate tracking
- Lead scoring and qualification
- Sales pipeline management

## 1.3 Invoices

**Primary Table:** `Sales.Invoice`

**Purpose:** Billing documents and payment requests

### Key Relationships:

- Links to Orders (one order → multiple invoices)
- Links to Customers
- Links to Finance domain for payment tracking

### Attributes:

- Invoice number, date, due date
- Line items and amounts
- Tax calculations
- Payment terms
- Invoice status (sent, paid, overdue)

## 1.4 Order Status

**Primary Table:** `Sales.OrderStatus`

**Purpose:** Dimension table for order lifecycle states

### Examples:

- Pending
- Confirmed
- Processing

- Shipped
- Delivered
- Completed
- Cancelled
- Refunded

**Pattern:** Type 2 SCD with timeslices for status definitions that change over time

## Cross-Domain Relationships

### **Sales → Contact:**

- Every order/lead links to a customer/user
- Enables customer lifetime value analysis

### **Sales → Product:**

- Orders reference products purchased
- Product performance analysis

### **Sales → Marketing:**

- Campaign attribution
- Channel effectiveness

### **Sales → Finance:**

- Revenue recognition
- Payment reconciliation

---

## 2. Finance Domain

---

### **Purpose**

The Finance domain manages financial data including payments, revenue tracking, invoicing, and accounting integration. It provides the financial view of business operations.

### **Key Entities**

#### **2.1 Payments**

**Primary Table:** `Finance.Payment`

**Purpose:** Track customer payments and payment methods

**Key Columns:**

```
CREATE TABLE [Finance].[Payment] (
    [PaymentId] [bigint] IDENTITY(1,1) NOT NULL,
    [PaymentNK] [nvarchar](100) NOT NULL,
    [PaymentDate] [datetime] NULL,
    [InvoiceId] [bigint] NULL, -- Link to invoice
    [CustomerId] [bigint] NULL, -- Link to customer
    [Amount] [decimal](18,2) NULL,
    [PaymentMethodId] [int] NULL, -- Card, bank, etc.
    [PaymentStatusId] [int] NULL, -- Success, failed
    [TransactionReference] [nvarchar](200) NULL, -- External reference
    [ValidFromDate] [date] NULL,
    [ValidToDate] [date] NULL,
    [SourceSystemId] [bigint] NOT NULL,
    CONSTRAINT [PK_Payment] PRIMARY KEY ([PaymentId])
)
```

**Payment Methods:**

- Credit/Debit Card
- Bank Transfer
- Digital Wallets (PayPal, Stripe, etc.)
- Cash
- Check

## 2.2 Revenue Recognition

**Purpose:** Track revenue according to accounting rules

**Attributes:**

- Recognition date
- Recognition method (upfront, amortized, usage-based)
- Recognized amount vs. invoiced amount
- Deferred revenue tracking

## 2.3 Invoicing

### Integration with Sales:

- Invoice header (from Sales or Finance)
- Invoice line items
- Tax calculations
- Payment application

## 2.4 Financial Dimensions

**Cost Centers:** Organizational units for financial tracking

**GL Accounts:** General ledger account mapping

**Business Units:** Multi-entity accounting

## Cross-Domain Relationships

### Finance → Sales:

- Payments link to orders and invoices
- Revenue recognition based on sales

### Finance → Contact:

- Customer payment history
- Credit management

---

## 3. Product Domain

---

### Purpose

The Product domain maintains the catalog of products, services, or offerings that the business sells. It includes hierarchies, attributes, pricing, and product lifecycle management.

### Key Entities

#### 3.1 Products

**Primary Table:** `Product.Product`

**Purpose:** Core product catalog

**Key Columns:**

```
CREATE TABLE [Product].[Product] (
    [ProductId] [bigint] IDENTITY(1,1) NOT NULL,
    [ProductNK] [nvarchar](100) NOT NULL,
    [ProductName] [nvarchar](200) NULL,
    [ProductDescription] [nvarchar](max) NULL,
    [ProductGroupId] [bigint] NULL, -- FK to hierarchy
    [SKU] [nvarchar](50) NULL,
    [Price] [decimal](18,2) NULL,
    [IsActive] [bit] NULL,
    [ValidFromDate] [date] NOT NULL,
    [ValidToDate] [date] NOT NULL,
    [SourceSystemId] [bigint] NOT NULL,
    CONSTRAINT [PK_Product] PRIMARY KEY ([ProductId])
)
```

**Companion Tables:**

- `Product.ProductProperties` : EAV for flexible product attributes
- `Product.ProductImage` : Product photos/media
- `Product.ProductPricing` : Price history and variations

**Industry Adaptations:**

- **E-Commerce:** Physical goods, variants (size, color)
- **Price Comparison:** Energy plans, insurance policies
- **SaaS:** Subscription tiers, features
- **Mobile:** Carrier plans, data packages

### 3.2 Product Groups / Categories

**Primary Table:** `Product.ProductGroup`

**Purpose:** Hierarchical product organization

**Structure:**

```
CREATE TABLE [Product].[ProductGroup] (
    [ProductGroupId] [bigint] IDENTITY(1,1) NOT NULL,
    [ProductGroupNK] [nvarchar](100) NOT NULL,
    [ProductGroupName] [nvarchar](200) NULL,
    [ParentProductGroupId] [bigint] NULL,           -- Self-referencing
    [Level] [int] NULL,                             -- Hierarchy level
    [ValidFromDate] [date] NULL,
    [ValidToDate] [date] NULL,
    [SourceSystemId] [bigint] NOT NULL,
    CONSTRAINT [PK_ProductGroup] PRIMARY KEY ([ProductGroupId])
)
```

### Hierarchy Example:

```
Level 1: Electronics
  Level 2: Computers
    Level 3: Laptops
    Level 3: Desktops
  Level 2: Mobile Devices
    Level 3: Smartphones
    Level 3: Tablets
```

## 3.3 Offers

**Primary Table:** `Product.Offer`

**Purpose:** Special pricing, promotions, or bundled products

#### Attributes:

- Offer name and description
- Discount type (percentage, fixed amount)
- Valid date range
- Product applicability
- Terms and conditions

## 3.4 Suppliers / Vendors

**Primary Table:** `Product.Supplier`

**Purpose:** Track product sources for procurement

**Relationships:**

- Products link to suppliers
- Pricing variations by supplier
- Inventory management

## Cross-Domain Relationships

**Product → Sales:**

- Orders reference products
- Product performance metrics

**Product → Marketing:**

- Campaigns promote specific products
- Product recommendations

**Product → Contact:**

- Product preferences per customer
  - Personalization
- 

## 4. Contact Domain

---

### Purpose

The Contact domain manages all person and organization entities that interact with the business: customers, users, households, companies, and their relationships.

### Key Entities

#### 4.1 Customers

**Primary Table:** `Contact.Customer`

**Purpose:** Business relationship with an individual or organization

## Key Columns:

```
CREATE TABLE [Contact].[Customer] (
    [CustomerId] [bigint] IDENTITY(1,1) NOT NULL,
    [CustomerNK] [nvarchar](100) NOT NULL,
    [CustomerName] [nvarchar](200) NULL,
    [CustomerType] [nvarchar](50) NULL,                -- B2C, B2B
    [Email] [nvarchar](200) NULL,
    [Phone] [nvarchar](50) NULL,
    [SegmentId] [int] NULL,                            -- Customer segment
    [IsActive] [bit] NULL,
    [ValidFromDate] [date] NOT NULL,
    [ValidToDate] [date] NOT NULL,
    [SourceSystemId] [bigint] NOT NULL,
    CONSTRAINT [PK_Customer] PRIMARY KEY ([CustomerId])
)
```

## Companion Tables:

- `Contact.CustomerProperties` : EAV for custom attributes
- `Contact.CustomerAddress` : Multiple addresses per customer
- `Contact.CustomerSegment` : Segmentation rules

## 4.2 Users

**Primary Table:** `Contact.User`

**Purpose:** Individual people who interact with the business (may or may not be customers)

### Distinction:

- **User:** Individual person with login, identity
- **Customer:** Business relationship, may represent household or organization
- **Relationship:** One customer can have multiple users; one user can belong to multiple customers

### Attributes:

- User credentials (hashed, or reference to auth system)
- Personal information
- Preferences

- Activity history

### 4.3 Households

**Primary Table:** `Contact.Household`

**Purpose:** Group related individuals (family units)

**Use Cases:**

- B2C businesses where purchasing decisions are household-level
- Utility services (energy, telecom)
- Insurance products
- Subscription services

**Relationships:**

- Multiple contacts belong to one household
- Household-level offers and marketing

### 4.4 Organizations

**Primary Table:** `Contact.Organization`

**Purpose:** Companies or institutions (B2B scenarios)

**Attributes:**

- Company name
- Industry
- Size (employees, revenue)
- Parent company (hierarchy)
- Account manager

### 4.5 Contact Relationships

**Primary Table:** `Contact.ContactRelationship`

**Purpose:** Map relationships between contacts

**Relationship Types:**

- Family member

- Business partner
- Referral
- Decision maker / influencer

## Cross-Domain Relationships

### Contact → Sales:

- Customers place orders
- Lead-to-customer conversion

### Contact → Finance:

- Payment history
- Credit limits
- Billing information

### Contact → Marketing:

- Campaign targeting
- Segmentation
- Communication preferences

---

## 5. Marketing Domain

---

### Purpose

The Marketing domain tracks all marketing activities: campaigns, channels, publishers, ad spending, and attribution. It enables marketing performance analysis and ROI calculation.

### Key Entities

#### 5.1 Campaigns

**Primary Table:** `Marketing.Campaign`

**Purpose:** Marketing initiatives with specific goals and budgets

**Key Columns:**

```

CREATE TABLE [Marketing].[Campaign] (
    [CampaignId] [bigint] IDENTITY(1,1) NOT NULL,
    [CampaignNK] [nvarchar](100) NOT NULL,
    [CampaignName] [nvarchar](200) NULL,
    [CampaignType] [nvarchar](50) NULL,          -- Email, Social,
    [StartDate] [date] NULL,
    [EndDate] [date] NULL,
    [Budget] [decimal](18,2) NULL,
    [ChannelId] [bigint] NULL,                  -- FK to Channel
    [ValidFromDate] [date] NOT NULL,
    [ValidToDate] [date] NOT NULL,
    [SourceSystemId] [bigint] NOT NULL,
    CONSTRAINT [PK_Campaign] PRIMARY KEY ([CampaignId])
)

```

### Campaign Types:

- Email marketing
- Social media (Facebook, Instagram, LinkedIn)
- Search engine marketing (Google Ads, Bing Ads)
- Display advertising
- Content marketing
- Affiliate marketing
- Influencer marketing

## 5.2 Channels

**Primary Table:** Marketing.Channel

**Purpose:** High-level marketing channels (aggregated view)

### Examples:

- Organic Search (SEO)
- Paid Search (SEM)
- Direct
- Referral
- Social Media
- Email
- Affiliate

- Display

**Usage:**

- Attribution modeling
- Channel mix optimization
- Budget allocation

### 5.3 Publishers

**Primary Table:** `Marketing.Publisher`

**Purpose:** Third-party sites/platforms where ads appear

**Examples:**

- Google Ads
- Facebook
- Affiliate networks
- Display ad networks
- Influencer channels

**Attributes:**

- Publisher name
- Publisher type
- Commission structure
- Performance metrics

### 5.4 Ad Spending

**Primary Table:** `Marketing.AdSpend`

**Purpose:** Track marketing costs at granular level

**Key Columns:**

```
CREATE TABLE [Marketing].[AdSpend] (  
    [AdSpendId] [bigint] IDENTITY(1,1) NOT NULL,  
    [SpendDate] [date] NOT NULL,  
    [CampaignId] [bigint] NULL,  
    [ChannelId] [bigint] NULL,
```

```
[PublisherId] [bigint] NULL,  
[Amount] [decimal](18,2) NULL,  
[Clicks] [int] NULL,  
[Impressions] [bigint] NULL,  
[SourceSystemId] [bigint] NOT NULL,  
CONSTRAINT [PK_AdSpend] PRIMARY KEY ([AdSpendId])  
)
```

### Metrics:

- Cost per click (CPC)
- Cost per thousand impressions (CPM)
- Click-through rate (CTR)

## 5.5 Attribution

**Purpose:** Link orders/revenue back to marketing touchpoints

### Patterns:

- First-touch attribution
- Last-touch attribution
- Multi-touch attribution
- Time-decay models

### Implementation:

- Orders include CampaignId, ChannelId, PublisherId
- Attribution views calculate ROI by marketing dimension

## Cross-Domain Relationships

### Marketing → Sales:

- Campaign attribution on orders
- Conversion tracking

### Marketing → Contact:

- Campaign targeting
- Customer acquisition source

## Marketing → Finance:

- Marketing spend vs. revenue
  - ROI calculation
- 

## 6. Reference Domain

---

### Purpose

The Reference domain contains framework metadata, configuration tables, and shared lookup values. It's the "glue" that holds the framework together and enables metadata-driven ETL.

### Key Entities

#### 6.1 SourceSystem

**Primary Table:** `Reference.SourceSystem`

**Purpose:** Master configuration for all data source mappings

*See Framework Philosophy document for detailed structure*

#### Usage:

- Defines source-to-target mappings
- Controls ETL behavior
- Enables dynamic pipeline generation

#### 6.2 SourceColumnMapping

**Primary Table:** `Reference.SourceColumnMapping`

**Purpose:** Column-level mapping between source and target

#### Usage:

- Maps source column names to target aliases
- Enables Flow table generation
- Handles naming inconsistencies across sources

## 6.3 PropertyReference

**Primary Table:** `Reference.PropertyReference`

**Purpose:** Control which EAV properties are active

**Workflow:**

1. System discovers available properties from sources
2. Data stewards review and enable relevant properties
3. ETL automatically loads enabled properties

## 6.4 Lookup Tables

**Common Lookups:**

- Countries, states, cities
- Currencies
- Time dimensions (date, calendar)
- Business units
- Status codes

**Pattern:**

- Simple dimension tables
- Often manually populated or from external standards (ISO codes)
- Shared across domains

## Framework Tables

**Metadata Tables:**

- `Reference.SourceSystem`
- `Reference.SourceColumnMapping`
- `Reference.PropertyReference`

**Configuration Tables:**

- `Reference.ErrorCode` (fallout error definitions)
  - `Reference.DataType` (type mappings)
-

## 7. Fallout Domain

---

### Purpose

The Fallout domain tracks data quality issues, ETL errors, and provides a resolution workflow. It's unique to the UDM Framework and critical for data governance.

### Key Entities

#### 7.1 Fallout

**Primary Table:** `Fallout.Fallout`

**Purpose:** Log data quality issues for resolution

*See Framework Philosophy document for detailed structure*

#### Key Columns:

- `FalloutId`: Unique identifier
- `ErrorCodeId`: Type of error
- `SourceSystemId`: Which source had the issue
- `SourceSystemIdentifier`: Specific record with issue
- `InsertDate`: When error occurred
- `SolveDate`: When resolved (NULL if open)

#### 7.2 ErrorCode

**Primary Table:** `Fallout.ErrorCode`

**Purpose:** Define types of data quality issues

#### Standard Error Codes:

- 11, 12, 13, etc.: Missing foreign keys (one per dimension)
- 20-29: Data type/format errors
- 30-39: Business rule violations
- 40-49: Duplicate detection
- 50-59: Data completeness issues

## 7.3 Fallout Views

**VW\_TotalFallout:** Summary of open issues

```
CREATE VIEW [Fallout].[VW_TotalFallout] AS
SELECT
    ss.SourceDescription,
    ec.ErrorDescription,
    COUNT(*) as FalloutCount,
    MAX(f.InsertDate) as LatestFallout,
    MIN(f.InsertDate) as OldestFallout
FROM Fallout.Fallout f
JOIN Reference.SourceSystem ss ON f.SourceSystemId = ss.SourceSystemId
JOIN Fallout.ErrorCode ec ON f.ErrorCodeId = ec.ErrorCodeId
WHERE f.SolveDate IS NULL
GROUP BY ss.SourceDescription, ec.ErrorDescription
```

### Usage:

- Daily data quality dashboards
- SLA monitoring
- Root cause analysis

## Fallout Resolution Workflow

1. **Detection:** ETL identifies issue, logs to Fallout
2. **Alert:** Monitoring system notifies data steward
3. **Analysis:** Review fallout records, identify root cause
4. **Fix:** Correct at source or adjust business rules
5. **Reprocess:** Re-run ETL for corrected records
6. **Close:** SolveDate updated automatically when record processes successfully

---

## Additional Domains (Industry-Specific Extensions)

While the seven core domains are universal, implementations often add industry-specific domains:

### Interaction Domain

**Purpose:** Customer interactions beyond purchases

**Entities:**

- Customer service calls
- Chat sessions
- Support tickets
- Website visits

**Industries:** Call centers, customer support, SaaS

## Inventory Domain

**Purpose:** Stock levels, warehousing, fulfillment

**Entities:**

- Inventory levels
- Warehouses
- Stock movements
- Reorder points

**Industries:** Retail, e-commerce, manufacturing

## ResultLog Domain

**Purpose:** Customer journey tracking (specific to Pricewise/CC)

**Entities:**

- User sessions
- Search queries
- Comparison actions
- Clicks and conversions

**Industries:** Price comparison, affiliate marketing

---

# Cross-Domain Entity Relationship Diagram

---

## Core Relationships

erDiagram

```
Contact ||--o{ Order : "places"
Product ||--o{ Order : "purchased_in"
Marketing ||--o{ Order : "attributed_to"
Order ||--o{ Invoice : "generates"
Invoice ||--o{ Payment : "paid_by"
```

```
Contact {
    bigint CustomerId PK
    nvarchar CustomerNK UK
    nvarchar CustomerName
    nvarchar Email
    date ValidFromDate
    date ValidToDate
    bigint SourceSystemId FK
}
```

```
Product {
    bigint ProductId PK
    nvarchar ProductNK UK
    nvarchar ProductName
    bigint ProductGroupId FK
    date ValidFromDate
    date ValidToDate
    bigint SourceSystemId FK
}
```

```
Marketing {
    bigint CampaignId PK
    nvarchar CampaignNK UK
    nvarchar CampaignName
    bigint ChannelId FK
    date StartDate
    date EndDate
    bigint SourceSystemId FK
}
```

```
Order {
    bigint OrderId PK
    nvarchar OrderNK UK
    datetime OrderDate
```

```

        bigint CustomerId FK
        bigint ProductId FK
        bigint CampaignId FK
        decimal Amount
        bigint SourceSystemId FK
    }

Invoice {
    bigint InvoiceId PK
    nvarchar InvoiceNK UK
    bigint OrderId FK
    bigint CustomerId FK
    decimal Amount
    datetime InvoiceDate
    bigint SourceSystemId FK
}

Payment {
    bigint PaymentId PK
    nvarchar PaymentNK UK
    bigint InvoiceId FK
    bigint CustomerId FK
    decimal Amount
    datetime PaymentDate
    bigint SourceSystemId FK
}

Reference {
    bigint SourceSystemId PK
    nvarchar SourceDescription
    nvarchar SourceDatabase
    nvarchar SourceTable
    nvarchar TargetSchema
    nvarchar TargetTable
}

Fallout {
    int FalloutId PK
    int ErrorCodeId FK
    bigint SourceSystemId FK
    nvarchar SourceSystemIdentifier
    datetime InsertDate

```

```

        datetime SolveDate
    }

    Reference ||--o{ Contact : "configures"
    Reference ||--o{ Product : "configures"
    Reference ||--o{ Order : "configures"
    Reference ||--o{ Invoice : "configures"
    Reference ||--o{ Marketing : "configures"

    Fallout ||--o{ Order : "tracks_quality"
    Fallout ||--o{ Contact : "tracks_quality"
    Fallout ||--o{ Product : "tracks_quality"

```

## Key Relationship Patterns

### Hub-and-Spoke:

- Sales.Order is the central hub
- Spokes to Contact, Product, Marketing, Finance

### Hierarchies:

- Product.ProductGroup (self-referencing)
- Contact.Organization (parent company)
- Marketing.Channel (sub-channels)

### Many-to-Many (via Bridge tables):

- Contact-to-Contact relationships
- Product bundles
- Campaign-to-Product associations

### Temporal Relationships:

- All entities have ValidFromDate/ValidToDate
- Point-in-time joins require date range matching

---

## Domain Design Principles

### 1. Normalized in DWH, Denormalized in Reporting

**DWH Layer:** Proper normalization

- Reduces redundancy
- Ensures data integrity
- Simplifies updates

**Reporting Layer:** Denormalized views

- Optimizes query performance
- Simplifies user queries
- Pre-joins dimensions

## 2. Consistent Patterns Across Domains

**Every Entity Has:**

- Surrogate key (Id)
- Natural key (NK)
- Valid date range (ValidFromDate, ValidToDate)
- Source traceability (SourceSystemId, SourceSystemIdentifier)

**Every Domain Has:**

- Core entity tables
- Property tables (EAV) where applicable
- Dimension tables for lookups
- Supporting tables for relationships

## 3. Flexible Extension Points

**Property Tables:** Add attributes without schema changes

**Custom Schemas:** Add industry-specific domains alongside core

**Views:** Create custom reporting structures without altering DWH

---

## Conclusion

---

The UDM Framework's domain architecture provides:

1. **Universal Applicability:** Core domains work across industries
2. **Clear Organization:** Logical grouping of related entities

3. **Consistent Patterns:** Same design principles throughout
4. **Flexibility:** EAV and custom domains for specific needs
5. **Traceability:** Source system tracking on every record
6. **Quality:** Built-in fallout tracking across all domains
7. **History:** Timeslices preserve complete timeline

These seven core domains—Sales, Finance, Product, Contact, Marketing, Reference, and Fallout—form the foundation of any UDM implementation, with industry-specific domains added as needed.

---

Next: [04 ETL Patterns.md](#) - Detailed ETL methodology and patterns