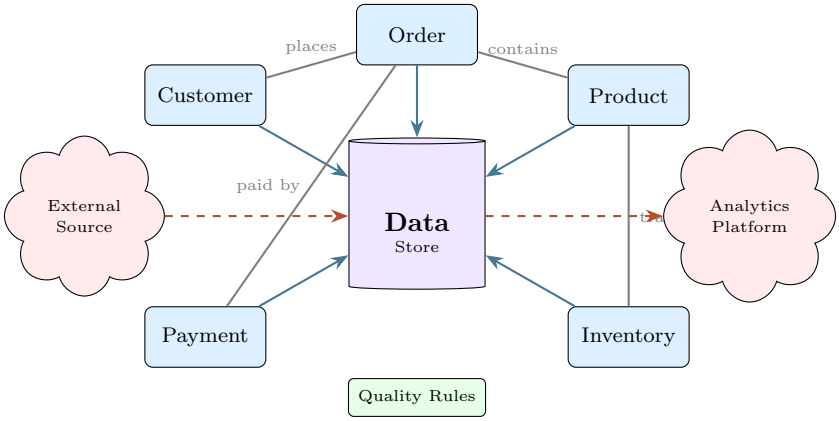
Information Viewpoint

Architecture Viewpoint Specification

Data Architecture & Information Management



Version:	2.0
Status:	Release
Classification:	ISO/IEC/IEEE 42010 Compliant
Last Updated:	December 12, 2025

Based on the Views and Beyond approach to software architecture documentation

Contents

1 Viewpoint Name

Viewpoint Identification

Name:	Information Viewpoint
Synonyms:	Data Viewpoint, Data Architecture View, Information Architecture View, Data Model View, Conceptual Data View, Logical Data View, Data Flow View
Identifier:	VP-INFO-001
Version:	2.0

1.1 Viewpoint Classification

The Information Viewpoint addresses data-centric concerns within software architecture. While not explicitly defined as a single style in the original Views and Beyond approach, it represents a critical cross-cutting concern that intersects with module, component-and-connector, and allocation viewpoints. This viewpoint provides a unified perspective on how data is structured, stored, flows through, and is managed by the system.

Table 1: Viewpoint Classification Taxonomy

Attribute	Value
Style Family	Cross-Cutting / Data-Centric
Primary Focus	Data Structure, Flow, Storage, and Governance
Abstraction Level	Conceptual, Logical, and Physical
Temporal Perspective	Static Structure and Dynamic Flow
Related Styles	Data Model, Data Flow, Repository
IEEE 42010 Category	Information Viewpoint
TOGAF Equivalent	Data Architecture

1.2 Viewpoint Scope

The Information Viewpoint encompasses multiple perspectives on data within a system:

- **Conceptual Data Model:** Business-oriented view of data entities and relationships independent of implementation, expressed in terms familiar to business stakeholders.
- **Logical Data Model:** Detailed structure of data entities, attributes, relationships, and constraints without physical implementation details.
- **Physical Data Model:** Implementation-specific data structures including tables, columns, indexes, partitions, and storage configurations.

- **Data Flow Model:** Movement of data through system components, transformations applied, and integration patterns used.
- **Data Lifecycle Model:** States data passes through from creation to archival or deletion, including retention policies.
- **Data Quality Model:** Rules, constraints, and processes ensuring data accuracy, completeness, and consistency.
- **Data Security Model:** Classification, access control, encryption, and privacy protection mechanisms.

2 Overview

The Information Viewpoint provides a comprehensive framework for documenting the data architecture of a software system. It addresses how data is structured, stored, accessed, transformed, and governed throughout its lifecycle. In modern data-driven systems, this viewpoint is essential for ensuring data quality, consistency, security, and compliance.

2.1 Purpose and Scope

The primary purpose of this viewpoint is to establish a clear understanding of the system's data assets, their structures and relationships, where data resides, how it flows between components, and how it is protected and governed. This understanding is critical for system design, integration, compliance, and evolution.

Viewpoint Definition

The Information Viewpoint defines the structure, semantics, storage, flow, and governance of data within a system. It encompasses data entities and their relationships, data storage mechanisms, data movement patterns, data quality rules, and data protection measures. This viewpoint bridges business information needs with technical data implementation.

2.2 Key Characteristics

The Information Viewpoint exhibits several distinctive characteristics:

Multi-Level Abstraction: Data architecture is documented at conceptual (business), logical (design), and physical (implementation) levels, enabling communication with different stakeholder groups and supporting design refinement.

Static and Dynamic Perspectives: The viewpoint captures both static data structures (entities, relationships, schemas) and dynamic data behavior (flows, transformations, lifecycle states).

Cross-System Scope: Data often spans multiple systems, databases, and services. This viewpoint addresses data integration, synchronization, and consistency across system boundaries.

Governance Integration: Data governance concerns including quality, security, privacy, and compliance are integral to this viewpoint, not afterthoughts.

Business Alignment: Data models explicitly connect to business concepts, ensuring that technical data structures support business information requirements.

2.3 Relationship to Other Viewpoints

The Information Viewpoint connects to other architectural viewpoints in significant ways:

Table 2: Relationships to Other Viewpoints

Viewpoint	Relationship
Development	Data model entities are implemented as domain classes/modules. Repository patterns connect data to code.
Component-and-Connector	Data stores are components; data flows are connectors. Runtime data access patterns are defined.
Deployment	Physical data stores are deployed to infrastructure. Replication and distribution strategies are realized.
Security	Data classification drives security controls. Access policies map to data elements. Encryption requirements apply.
Operational	Backup, recovery, and archival procedures apply to data stores. Monitoring covers data health.
Business/Functional	Business entities map to data entities. Business rules become data constraints.

2.4 Data Architecture Layers

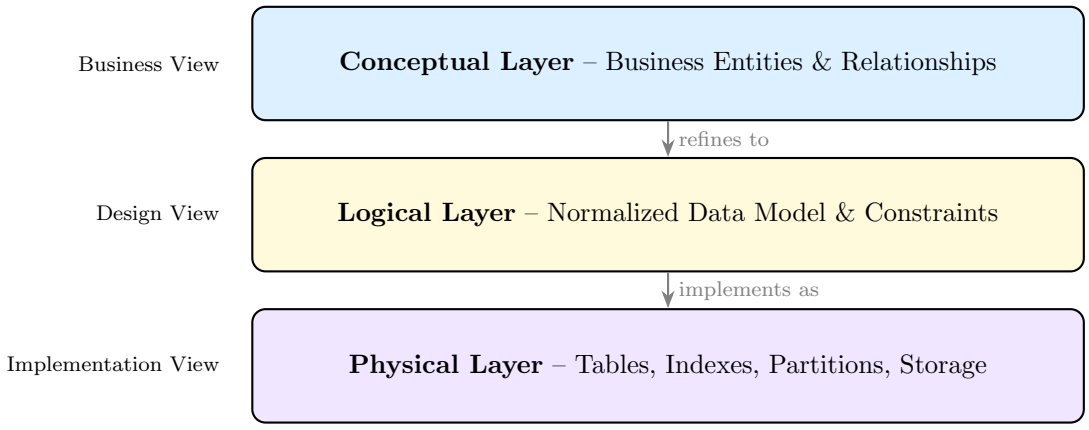


Figure 1: Data Architecture Abstraction Layers

3 Concerns

This section enumerates the architectural concerns that the Information Viewpoint is designed to address. These concerns represent the fundamental questions stakeholders have about the system's data architecture.

3.1 Primary Concerns

C1: Data Structure and Modeling

- What are the key data entities in the system?
- What attributes define each entity?
- What relationships exist between entities?
- What constraints ensure data integrity?
- How does the data model align with business concepts?

C2: Data Storage and Persistence

- Where is data physically stored?
- What database technologies are used?
- How is data partitioned and distributed?
- What indexing strategies optimize access?
- How is storage capacity planned and managed?

C3: Data Flow and Integration

- How does data move through the system?
- What transformations are applied to data?
- How is data synchronized across systems?
- What integration patterns are used (ETL, CDC, API)?
- How is data consistency maintained across boundaries?

C4: Data Quality and Integrity

- What rules ensure data accuracy?
- How is data completeness validated?
- What processes detect and correct data errors?
- How is data freshness maintained?
- What metrics measure data quality?

C5: Data Security and Privacy

- How is sensitive data classified?
- What access controls protect data?
- How is data encrypted at rest and in transit?
- What data masking or anonymization is applied?
- How are privacy regulations (GDPR, CCPA) addressed?

C6: Data Lifecycle Management

- What states does data pass through?
- What retention policies apply?
- How is data archived?
- When and how is data deleted?
- How are lifecycle transitions triggered?

C7: Data Governance and Ownership

- Who owns each data domain?
- What stewardship responsibilities exist?
- How are data standards enforced?
- What approval processes govern data changes?
- How is data lineage tracked?

C8: Data Performance and Scalability

- What are data volume projections?
- How does the system handle data growth?
- What query performance requirements exist?
- How is data access optimized?
- What caching strategies are employed?

C9: Data Availability and Recovery

- What data redundancy exists?
- How is data backed up?
- What recovery point objectives (RPO) apply?
- What recovery time objectives (RTO) apply?
- How is data disaster recovery handled?

C10: Data Compliance and Audit

- What regulatory requirements affect data?
- How is data access audited?
- What compliance certifications are required?
- How is data sovereignty addressed?
- What audit trails are maintained?

3.2 Concern-Quality Attribute Mapping

Table 3: Concern to Quality Attribute Mapping

Concern	Integrity	Security	Performance	Availability	Scalability	Compliance	Maintaintic.	Interop.
Data Structure	●	○	○	–	○	○	●	○
Storage	○	○	●	●	●	○	○	–
Flow/Integration	●	○	○	○	○	○	○	●
Quality	●	–	–	–	–	●	○	○
Security/Privacy	○	●	○	–	–	●	○	–
Lifecycle	○	○	○	○	●	●	●	–
Governance	●	○	–	–	–	●	●	○
Performance	–	–	●	○	●	–	○	–
Availability	○	–	○	●	○	○	○	–
Compliance	○	●	–	○	–	●	○	○

● = Primary impact, ○ = Secondary impact, – = Minimal impact

4 Anti-Concerns

Understanding what the Information Viewpoint is *not* appropriate for helps stakeholders avoid misapplying this viewpoint.

4.1 Out of Scope Topics

AC1: Application Logic and Behavior

- Business process workflows and orchestration
- Algorithm implementations and computation logic
- User interface behavior and interactions
- Service orchestration patterns
- Event-driven behavior specifications

AC2: Infrastructure Details

- Server hardware specifications
- Network topology and routing
- Container orchestration configuration
- Operating system configuration
- Cloud infrastructure provisioning

AC3: Code Organization

- Module decomposition and dependencies

- Class hierarchies and inheritance
- Package structure and namespaces
- Build system configuration
- Source code repository organization

AC4: User Experience

- Screen designs and layouts
- Navigation flows
- Accessibility requirements
- Localization and internationalization UI
- User feedback and error messaging

AC5: Project Management

- Development schedules and milestones
- Resource allocation and budgeting
- Risk management processes
- Team organization and roles
- Vendor management

Common Misapplications

Avoid using the Information Viewpoint for:

- Documenting runtime component interactions (use C&C Viewpoint)
- Specifying deployment topology (use Deployment Viewpoint)
- Defining code module structure (use Development Viewpoint)
- Detailing business processes (use Process/Functional Viewpoint)
- Specifying API contracts (use Interface Specifications)

5 Typical Stakeholders

The Information Viewpoint serves multiple stakeholder communities, each with distinct information needs about the system's data architecture.

5.1 Primary Stakeholders

Table 4: Primary Stakeholder Analysis

Stakeholder	Role Description	Primary Interests
Data Architects	Design enterprise data structures	Data modeling, naming standards, integration patterns, master data management
Database Administrators	Manage database systems	Physical schemas, indexes, performance tuning, backup/recovery, storage management
Data Engineers	Build data pipelines	ETL/ELT processes, data transformations, data quality, pipeline orchestration
Data Stewards	Govern data domains	Data quality rules, business definitions, data ownership, compliance
Privacy Officers	Ensure data privacy	Personal data inventory, consent management, privacy impact, regulatory compliance
Security Architects	Protect data assets	Data classification, access control, encryption, threat modeling

5.2 Secondary Stakeholders

Table 5: Secondary Stakeholder Analysis

Stakeholder	Role Description	Primary Interests
Software Developers	Implement applications	Domain models, data access APIs, ORM mappings, data validation
Business Analysts	Define requirements	Business entities, data requirements, reporting needs, data dictionaries
Data Scientists	Analyze data	Data availability, data quality, feature engineering, data access
Compliance Officers	Ensure regulatory compliance	Audit trails, retention policies, data sovereignty, regulatory mapping
Operations Teams	Maintain production systems	Data monitoring, incident response, capacity management, disaster recovery
Business Stakeholders	Use data for decisions	Data definitions, data availability, reporting capabilities, data accuracy

5.3 Stakeholder Concern Matrix

Table 6: Stakeholder-Concern Responsibility Matrix

	Structure	Storage	Flow	Quality	Security	Lifecycle	Governance	Perform.	Avail.	Compliance
Data Architect	R	A	R	A	C	A	R	C	C	C
DBA	C	R	C	C	A	C	I	R	R	I
Data Engineer	C	C	R	R	C	C	I	A	C	I
Data Steward	A	I	I	R	I	R	R	I	I	A
Privacy Officer	I	I	C	C	A	A	A	I	I	R
Security Arch.	C	C	C	I	R	C	C	I	C	A

R = Responsible, A = Accountable, C = Consulted, I = Informed

6 Model Types

The Information Viewpoint employs several complementary model types to capture different aspects of the data architecture. Each model type serves specific documentation purposes.

6.1 Model Type Catalog

MT1: Conceptual Data Model

- *Purpose:* Show business entities and relationships at high level
- *Primary Elements:* Business entities, relationships, business rules
- *Key Relationships:* Associates, contains, specializes
- *Typical Notation:* Simplified ER diagram, concept maps

MT2: Logical Data Model

- *Purpose:* Define normalized data structures with full detail
- *Primary Elements:* Entities, attributes, keys, relationships, constraints
- *Key Relationships:* One-to-one, one-to-many, many-to-many
- *Typical Notation:* ER diagrams (Chen, Crow's foot, IDEF1X)

MT3: Physical Data Model

- *Purpose:* Document implementation-specific database structures
- *Primary Elements:* Tables, columns, indexes, partitions, constraints
- *Key Relationships:* Foreign keys, triggers, procedures
- *Typical Notation:* Database diagrams, DDL scripts

MT4: Data Flow Diagram

- *Purpose:* Show movement and transformation of data

- *Primary Elements:* Processes, data stores, external entities, flows
- *Key Relationships:* Data flow, transformation, aggregation
- *Typical Notation:* DFD (Yourdon, Gane-Sarson), data pipelines

MT5: Data Lineage Diagram

- *Purpose:* Track data origin, transformations, and destinations
- *Primary Elements:* Sources, transformations, targets, dependencies
- *Key Relationships:* Derived-from, transformed-by, feeds-into
- *Typical Notation:* Lineage graphs, dependency trees

MT6: Data Classification Schema

- *Purpose:* Document data sensitivity and handling requirements
- *Primary Elements:* Classification levels, data categories, handling rules
- *Key Relationships:* Classified-as, requires, governed-by
- *Typical Notation:* Classification matrices, data catalogs

MT7: Data Lifecycle Model

- *Purpose:* Define states and transitions in data lifecycle
- *Primary Elements:* States, transitions, triggers, retention periods
- *Key Relationships:* Transitions-to, triggered-by, expires-after
- *Typical Notation:* State diagrams, lifecycle charts

MT8: Data Quality Rules Model

- *Purpose:* Specify data quality constraints and validation rules
- *Primary Elements:* Rules, dimensions, thresholds, remediation
- *Key Relationships:* Validates, measures, triggers
- *Typical Notation:* Rule tables, quality dashboards

MT9: Master Data Model

- *Purpose:* Define authoritative sources for shared data
- *Primary Elements:* Master entities, golden records, sources
- *Key Relationships:* Mastered-by, synchronized-from, authoritative-for
- *Typical Notation:* MDM architecture diagrams

6.2 Model Type Relationships

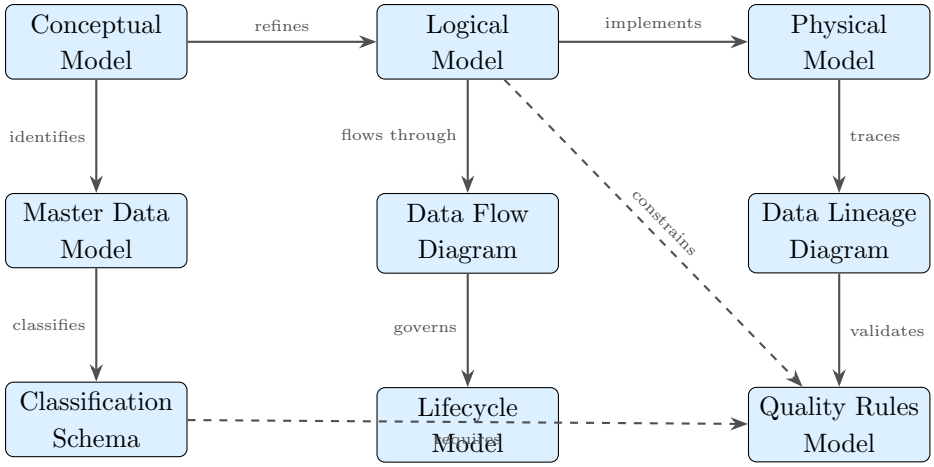


Figure 2: Model Type Dependency Relationships

7 Model Languages

For each model type, specific languages, notations, and modeling techniques are prescribed. This section documents the vocabulary and grammar for constructing information views.

7.1 Entity-Relationship Notation

Entity-Relationship diagrams are the primary notation for data modeling. Several notation variants exist:

7.1.1 Chen Notation

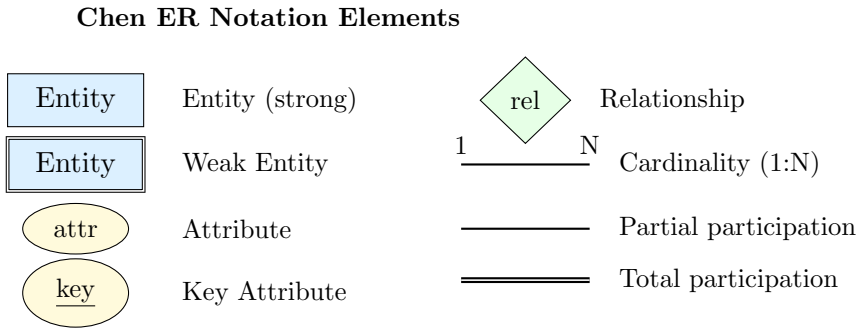


Figure 3: Chen ER Notation Legend

7.1.2 Crow's Foot Notation

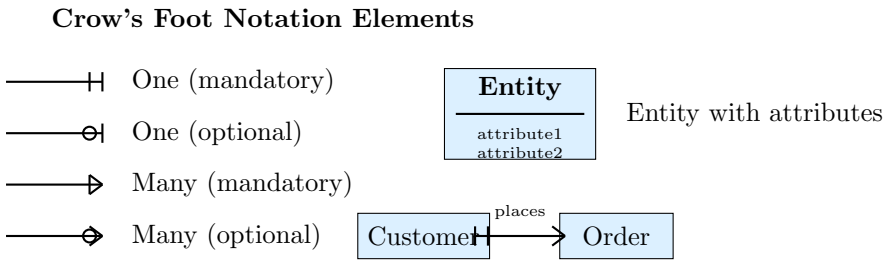


Figure 4: Crow's Foot Notation Legend

7.2 Data Flow Diagram Notation

Data Flow Diagrams (DFDs) use standardized symbols to show data movement:

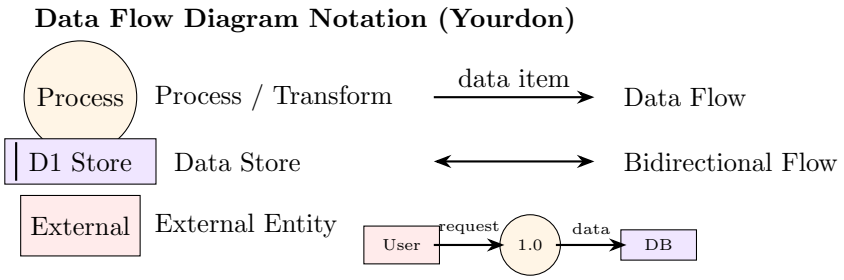


Figure 5: Data Flow Diagram Notation Legend

7.3 Data Definition Languages

Physical data models are often expressed using database-specific DDL:

```
-- Customer entity
CREATE TABLE customer (
    customer_id     UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email           VARCHAR(255) NOT NULL UNIQUE,
    first_name      VARCHAR(100) NOT NULL,
    last_name       VARCHAR(100) NOT NULL,
    phone           VARCHAR(20),
    created_at      TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at      TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    status          VARCHAR(20) NOT NULL DEFAULT 'active'
        CHECK (status IN ('active', 'inactive', 'suspended')),

    -- Audit fields
    created_by      VARCHAR(100),
    updated_by      VARCHAR(100)
);

```

```
-- Order entity with foreign key
CREATE TABLE orders (
    order_id        UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    customer_id     UUID NOT NULL REFERENCES customer(customer_id),
    order_date      TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    total_amount    DECIMAL(12,2) NOT NULL CHECK (total_amount >= 0),
    status          VARCHAR(20) NOT NULL DEFAULT 'pending'
        CHECK (status IN ('pending', 'confirmed', 'shipped', 'delivered'
           , 'cancelled')),

    -- Indexing for common queries
    INDEX idx_customer_orders (customer_id),
    INDEX idx_order_date (order_date)
);
```

Listing 1: SQL DDL Example

7.4 Data Catalog Specifications

Data catalogs document metadata about data assets:

Table 7: Example Data Catalog Entry Format

Attribute	Value
Asset Name	customer
Type	Table
Database	commerce_db
Schema	public
Owner	Customer Domain Team
Steward	Jane Smith
Classification	Confidential - PII
Description	Master customer records containing profile and contact information
Retention	7 years after last activity
Update Frequency	Real-time
Row Count	2.5M
Quality Score	94%
Tags	customer, profile, PII, master-data

7.5 Tabular Specifications

7.5.1 Entity Attribute Table

Table 8: Example Entity Attribute Specification

Attribute	Data Type	Nullable	Key	Default	Description
customer_id	UUID	No	PK	auto-gen	Unique identifier
email	VARCHAR(255)	No	UK	–	Email address
first_name	VARCHAR(100)	No	–	–	First name
last_name	VARCHAR(100)	No	–	–	Last name
phone	VARCHAR(20)	Yes	–	NULL	Phone number
status	VARCHAR(20)	No	–	'active'	Account status
created_at	TIMESTAMP	No	–	NOW()	Creation time

7.5.2 Data Quality Rules Table

Table 9: Example Data Quality Rules Specification

Rule ID	Entity	Attribute	Rule	Severity	Action
DQ-001	Customer	email	Valid email format	Critical	Reject
DQ-002	Customer	phone	Valid phone format	Warning	Flag
DQ-003	Order	total	Must be ¿= 0	Critical	Reject
DQ-004	Order	customer_id	Must exist in Customer	Critical	Reject
DQ-005	Product	price	Within expected range	Warning	Review

8 Viewpoint Metamodels

This section defines the conceptual metamodel underlying the Information Viewpoint, establishing the vocabulary of element types, their properties, and valid relationships.

8.1 Core Metamodel

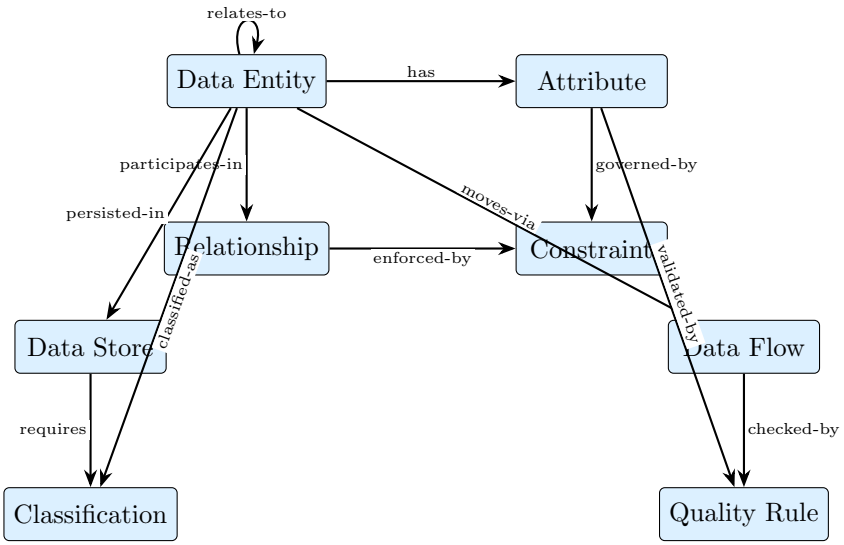


Figure 6: Information Viewpoint Core Metamodel

8.2 Entity Definitions

Entity: Data Entity

Definition: A distinguishable object or concept about which data is captured and stored. Represents a business object, event, or concept of interest.

Attributes:

- `entityId`: Unique identifier for the entity
- `name`: Business name for the entity
- `description`: Detailed business description
- `type`: Entity type (master, transactional, reference, derived)
- `domain`: Business domain ownership
- `version`: Schema version
- `status`: Lifecycle status (draft, active, deprecated)
- `aliases`: Alternative names or synonyms
- `businessRules`: Associated business rules

Constraints:

- Each entity must have a unique identifier attribute
- Entity names must be unique within a domain
- Entities must be assigned to exactly one owning domain
- Master entities must have defined golden record rules

Entity: Attribute

Definition: A property or characteristic of a data entity that captures a specific piece of information.

Attributes:

- `attributeId`: Unique identifier
- `name`: Attribute name (following naming standards)
- `description`: Business meaning
- `dataType`: Logical data type
- `length`: Maximum length (if applicable)
- `precision`: Numeric precision (if applicable)
- `nullable`: Whether null values are allowed
- `defaultValue`: Default value if not provided
- `keyType`: Key classification (PK, FK, AK, none)
- `sensitivity`: Data sensitivity classification
- `piiFlag`: Whether attribute contains PII

Constraints:

- Attribute names must be unique within an entity
- Data types must be from approved type list
- PII attributes must have appropriate classification
- Key attributes cannot be nullable

Entity: Relationship

Definition: An association between two or more data entities that captures a business rule or structural connection.

Attributes:

- `relationshipId`: Unique identifier
- `name`: Relationship name (verb phrase)
- `description`: Business meaning of relationship
- `sourceEntity`: Origin entity
- `targetEntity`: Destination entity
- `sourceCardinality`: Cardinality at source (1, 0..1, *, 1..*)
- `targetCardinality`: Cardinality at target
- `type`: Relationship type (association, aggregation, composition, generalization)
- `mandatory`: Whether relationship is required
- `bidirectional`: Whether navigable both directions

Constraints:

- Many-to-many relationships require junction entities
- Composition relationships imply cascade delete
- Circular relationships must be explicitly justified

Entity: Data Store

Definition: A persistent storage mechanism that holds data entities, such as a database, file system, or data lake.

Attributes:

- `storeId`: Unique identifier
- `name`: Store name
- `type`: Storage technology (RDBMS, NoSQL, Object, File)
- `technology`: Specific product (PostgreSQL, MongoDB, S3)
- `environment`: Deployment environment
- `connectionInfo`: Connection parameters
- `capacity`: Storage capacity
- `retentionPolicy`: Data retention configuration
- `backupPolicy`: Backup configuration
- `encryptionConfig`: Encryption settings

Constraints:

- Production stores must have backup policies
- Stores with PII must have encryption enabled
- Connection credentials must be stored securely

Entity: Data Flow

Definition: A movement of data between system components, processes, or data stores, including any transformations applied.

Attributes:

- `flowId`: Unique identifier
- `name`: Flow name
- `description`: Purpose of the data flow
- `source`: Origin component or store
- `target`: Destination component or store
- `dataElements`: Data items in the flow
- `frequency`: Flow frequency (real-time, batch, on-demand)
- `volume`: Expected data volume
- `transformations`: Transformations applied
- `protocol`: Transport protocol
- `errorHandling`: Error handling strategy

Constraints:

- Flows must specify error handling behavior
- Cross-boundary flows must be encrypted
- High-volume flows must have capacity planning

Entity: Classification

Definition: A categorization of data based on sensitivity, regulatory requirements, or business importance.

Attributes:

- `classificationId`: Unique identifier
- `level`: Classification level (Public, Internal, Confidential, Restricted)
- `category`: Data category (PII, PHI, PCI, Financial, IP)
- `handlingRequirements`: Required handling procedures
- `encryptionRequired`: Whether encryption is mandated
- `accessRestrictions`: Access control requirements
- `retentionRequirements`: Retention period requirements
- `regulatoryBasis`: Applicable regulations

Constraints:

- Higher classifications inherit lower classification requirements
- PII classification requires GDPR/CCPA compliance measures
- Restricted data requires audit logging

Entity: Quality Rule

Definition: A specification of a data quality requirement that data must satisfy, including validation logic and remediation actions.

Attributes:

- `ruleId`: Unique identifier
- `name`: Rule name
- `description`: Business rationale
- `dimension`: Quality dimension (accuracy, completeness, consistency, timeliness, validity, uniqueness)
- `expression`: Rule logic/expression
- `severity`: Violation severity (critical, major, minor, warning)
- `threshold`: Acceptable threshold
- `action`: Action on violation (reject, flag, correct, notify)
- `owner`: Rule owner

Constraints:

- Critical rules must have defined remediation procedures
- Rules must be testable and measurable
- Rule ownership must be assigned

Entity: Constraint

Definition: A rule or restriction that governs the values, relationships, or behavior of data elements.

Attributes:

- `constraintId`: Unique identifier
- `name`: Constraint name
- `type`: Constraint type (primary key, foreign key, unique, check, not null, default)
- `expression`: Constraint definition
- `scope`: Affected elements
- `enforcement`: Enforcement timing (immediate, deferred)
- `violationAction`: Action on violation

Constraints:

- Primary key constraints must be non-null and unique
- Foreign key constraints must reference existing keys
- Check constraints must be deterministic

8.3 Relationship Definitions

Table 10: Metamodel Relationship Definitions

Relationship	Source	Target	Description
has	Data Entity	Attribute	Entity contains this attribute
participates-in	Data Entity	Relationship	Entity is involved in this relationship
governed-by	Attribute	Constraint	Attribute is restricted by constraint
persisted-in	Data Entity	Data Store	Entity data is stored here
moves-via	Data Entity	Data Flow	Entity data transfers through this flow
classified-as	Data Entity	Classification	Entity has this sensitivity level
validated-by	Attribute	Quality Rule	Attribute is checked by this rule
derives-from	Data Entity	Data Entity	Entity is computed from source entity
masters	Data Entity	Data Entity	Entity is authoritative source
transforms	Data Flow	Data Entity	Flow modifies entity data

9 Conforming Notations

Several existing notations and modeling languages conform to the Information Viewpoint metamodel.

9.1 Entity-Relationship Diagrams

ER diagrams in various notations (Chen, Crow's Foot, IDEF1X, Barker) are the primary notation for data modeling.

Conformance Level: Full conformance for structural data modeling.

Tool Support: ERwin, PowerDesigner, Lucidchart, draw.io, MySQL Workbench, pgModeler.

9.2 UML Class Diagrams

UML class diagrams can represent data models, particularly when modeling object-relational mappings.

Conformance Level: Partial conformance; better for object models than relational.

Tool Support: Enterprise Architect, Visual Paradigm, StarUML, PlantUML.

9.3 Data Flow Diagrams

DFDs (Yourdon/DeMarco, Gane-Sarson) show data movement through systems.

Conformance Level: Full conformance for flow modeling.

Tool Support: Lucidchart, draw.io, Visio, SmartDraw.

9.4 BPMN Data Objects

BPMN includes data object notation that can document data in process context.

Conformance Level: Partial conformance; process-centric focus.

Tool Support: Camunda, Bizagi, Signavio, ARIS.

9.5 Data Catalog Tools

Modern data catalog platforms provide comprehensive data documentation:

Table 11: Data Catalog and Governance Tools

Tool	Type	Key Capabilities
Collibra	Data Governance	Catalog, lineage, quality, stewardship
Alation	Data Catalog	Discovery, collaboration, governance
Apache Atlas	Open Source	Metadata, lineage, classification (Hadoop)
AWS Glue Catalog	Cloud Native	Schema discovery, ETL integration
Azure Purview	Cloud Native	Unified governance across Azure
dbt	Transform Tool	Documentation, lineage, testing
Great Expectations	Quality	Data validation, documentation
DataHub	Open Source	Metadata search, lineage, discovery

Table 12: Notation Comparison Matrix

Feature	Chen ER	Crow's Foot	UML	DFD	BPMN	Catalog	Custom
Entities	●	●	●	○	○	●	●
Attributes	●	●	●	–	–	●	●
Relationships	●	●	●	–	○	●	●
Cardinality	●	●	●	–	–	○	●
Data Flow	–	–	○	●	●	○	●
Classification	–	–	○	–	–	●	●
Lineage	–	–	–	○	○	●	●
Quality Rules	–	–	○	–	–	●	●
Standardized	●	●	●	●	●	–	–

● = Strong support, ○ = Limited support, – = Not applicable

10 Model Correspondence Rules

Model correspondence rules define how elements in information models relate to elements in other architectural views.

10.1 Development View Correspondence

Correspondence Rule CR-01: Entity to Class Mapping

Rule: Each data entity in the logical model should correspond to one or more domain classes in the development view.

Formal Expression:

$$\forall e \in DataEntities : \exists C \subseteq Classes : implements(C, e)$$

Rationale: Ensures data structures are properly implemented in code.

Verification: ORM mapping review, code inspection.

Correspondence Rule CR-02: Attribute to Property Mapping

Rule: Each attribute in a data entity must map to a property in the corresponding domain class with compatible type.

Formal Expression:

$$\forall a \in entity.attributes : \exists p \in class.properties : maps(a, p) \land compatible(a.type, p.type)$$

Rationale: Ensures complete and type-safe data mapping.

Verification: ORM configuration audit.

10.2 Component-and-Connector View Correspondence

Correspondence Rule CR-03: Data Store to Component Mapping

Rule: Each data store in the information model must correspond to a component in the C&C view.

Formal Expression:

$$\forall ds \in DataStores : \exists c \in Components : realizes(c, ds)$$

Rationale: Ensures all storage is architecturally represented.

Verification: C&C diagram review.

Correspondence Rule CR-04: Data Flow to Connector Mapping

Rule: Data flows crossing component boundaries must have corresponding connectors in the C&C view.

Formal Expression:

$$\forall df \in DataFlows : crossesBoundary(df) \Rightarrow \exists conn \in Connectors : carries(conn, df)$$

Rationale: Ensures data movement is architecturally visible.

Verification: Flow-connector mapping matrix.

10.3 Deployment View Correspondence

Correspondence Rule CR-05: Data Store to Infrastructure Mapping

Rule: Each data store must be allocated to deployment infrastructure in the deployment view.

Formal Expression:

$$\forall ds \in DataStores : \exists n \in Nodes : hosts(n, ds)$$

Rationale: Ensures all data has defined physical location.

Verification: Deployment diagram audit.

10.4 Security View Correspondence

Correspondence Rule CR-06: Classification to Security Control Mapping

Rule: Each data classification level must have corresponding security controls implemented.

Formal Expression:

$$\forall c \in Classifications : \exists S \subseteq SecurityControls : protects(S, c)$$

Rationale: Ensures security requirements are implemented.

Verification: Security control assessment.

10.5 Correspondence Verification Matrix

Table 13: Cross-View Correspondence Verification

Rule ID	Source View	Target Elements	Verification Method
CR-01	Data Entities	Domain Classes	ORM mapping review
CR-02	Attributes	Class Properties	Type compatibility check
CR-03	Data Stores	C&C Components	Diagram alignment
CR-04	Data Flows	Connectors	Flow-connector mapping
CR-05	Data Stores	Infrastructure Nodes	Deployment audit
CR-06	Classifications	Security Controls	Control assessment

11 Operations on Views

This section defines the methods and procedures for creating, interpreting, analyzing, and implementing information views.

11.1 Creation Methods

11.1.1 View Development Process

Step 1: Establish Context and Scope

1. Identify business domains and subject areas
2. Gather business requirements and use cases
3. Review existing data assets and systems
4. Identify key stakeholders and data owners
5. Define scope boundaries for the data model

Step 2: Develop Conceptual Data Model

1. Identify business entities from requirements
2. Define entity relationships using business terms
3. Validate with business stakeholders
4. Document business rules affecting data
5. Establish entity ownership by domain

Step 3: Develop Logical Data Model

1. Refine entities with full attribute definitions
2. Normalize to appropriate normal form (typically 3NF)
3. Define primary and foreign keys
4. Specify data types and constraints
5. Document cardinality and optionality
6. Resolve many-to-many relationships

Step 4: Define Data Flows

1. Identify data sources (internal and external)
2. Map data movement between systems
3. Document transformations applied
4. Specify integration patterns (ETL, API, CDC)
5. Define error handling and recovery

Step 5: Establish Data Governance

1. Classify data by sensitivity
2. Define data quality rules
3. Establish retention policies
4. Document compliance requirements
5. Assign data stewardship responsibilities

Step 6: Develop Physical Data Model

1. Select storage technologies
2. Denormalize for performance where justified
3. Define indexes and partitioning strategies
4. Specify physical constraints
5. Plan storage capacity
6. Generate DDL scripts

Step 7: Validate and Review

1. Verify alignment with requirements
2. Check correspondence with other views
3. Review with data architects and DBAs
4. Validate security and compliance
5. Document rationale for key decisions

11.1.2 Documentation Templates

Entity Specification Template:

```
Entity Specification
====================
Entity ID:        [ENT-XXX]
Name:             [Entity Name]
Domain:           [Business Domain]
Type:             [Master | Transactional | Reference | Derived]
Status:           [Draft | Active | Deprecated]

Business Description:
  [Detailed business meaning and purpose]

Attributes:
  +---------------+----------+------+-----+---------+--------------------+

  | Name          | Type     | Null | Key | Default | Description
    |
  +---------------+----------+------+-----+---------+--------------------+

  | [attr_name]   | [type]   | Y/N  | PK  | [value] | [description]
    |
  +---------------+----------+------+-----+---------+--------------------+


Relationships:
  - [Relationship Name]: [Target Entity] ([Cardinality])
  - ...

Constraints:
  - [Constraint Name]: [Definition]
  - ...

Data Classification:
  - Sensitivity:     [Public | Internal | Confidential | Restricted]
  - Contains PII:    [Yes/No]
  - Regulations:     [GDPR, CCPA, HIPAA, PCI-DSS, etc.]

Quality Rules:
  - [Rule ID]: [Rule Description]
  - ...

Lifecycle:
  - Creation:        [How data is created]
  - Retention:       [Retention period]
  - Archival:        [Archival policy]
  - Deletion:        [Deletion policy]

```

```
Owner:                [Data Owner]
Steward:              [Data Steward]
Last Updated:         [Date]
```

Listing 2: Entity Specification Template

11.1.3 Common Patterns and Idioms

Pattern: Slowly Changing Dimension (SCD)

Context: Need to track historical changes to reference data over time.
Solution: Implement versioning strategy for dimensional data.
Types:

- **Type 1:** Overwrite old value (no history)
- **Type 2:** Add new row with version/date range (full history)
- **Type 3:** Add new column for previous value (limited history)
- **Type 4:** Separate history table (full history, separate)
- **Type 6:** Hybrid combining Types 1, 2, and 3

Use When: Audit requirements, historical reporting, trend analysis.

Pattern: Event Sourcing

Context: Need complete audit trail and ability to reconstruct state.
Solution: Store all state changes as a sequence of events rather than current state only.
Characteristics:

- Events are immutable and append-only
- Current state derived by replaying events
- Complete audit history preserved
- Enables temporal queries and debugging

Use When: Financial systems, compliance requirements, complex business logic.

Pattern: Data Vault

Context: Enterprise data warehouse requiring flexibility and audit capability.
Solution: Model data using Hubs (business keys), Links (relationships), and Satellites (descriptive attributes).
Components:

- **Hub:** Unique business keys with load metadata
- **Link:** Relationships between hubs
- **Satellite:** Descriptive attributes with temporal tracking

Use When: Data warehouse, multiple source integration, changing requirements.

Pattern: Polyglot Persistence

Context: Different data types have different storage requirements.

Solution: Use different database technologies for different data needs.

Typical Combinations:

- RDBMS for transactional data with ACID requirements
- Document store for flexible schemas and nested data
- Graph database for highly connected data
- Time-series database for metrics and events
- Search engine for full-text search
- Cache for hot data and session state

Use When: Diverse data requirements, performance optimization needs.

Table 14: Data Modeling Patterns Summary

Pattern	Description	Use When
Surrogate Key	System-generated ID separate from business key	Integration, performance, privacy
Audit Columns	Standard columns for tracking changes	Compliance, debugging
Soft Delete	Flag records as deleted rather than removing	Recovery, audit requirements
Temporal Tables	Built-in versioning with valid time	Historical queries, audit
Sharding	Horizontal partitioning across servers	High volume, scalability
CQRS	Separate read and write models	Complex domains, scalability
Materialized Views	Pre-computed query results	Performance, complex aggregations

11.2 Interpretive Methods

11.2.1 Reading Data Models

When interpreting an information view, stakeholders should follow this systematic approach:

1. **Understand the Scope:** Identify what subject area or domain is modeled and at what level of abstraction (conceptual, logical, physical).
2. **Identify Key Entities:** Locate the central business entities and understand their purpose and ownership.
3. **Trace Relationships:** Follow relationships between entities, noting cardinality and optionality

to understand business rules.

4. **Examine Attributes:** Review attribute definitions, data types, and constraints to understand what data is captured.
5. **Check Classifications:** Identify sensitive data and understand applicable security and compliance requirements.
6. **Follow Data Flows:** If flow diagrams are present, trace data movement to understand integration and transformation.
7. **Review Quality Rules:** Understand what validation and quality constraints apply to the data.

11.2.2 Stakeholder-Specific Interpretation Guides

For Business Analysts: Focus on the conceptual model and business entity definitions. Verify that business concepts are correctly represented and relationships reflect actual business rules.

For Developers: Examine the logical model for domain class design. Understand attribute types for ORM mapping. Note constraints that must be enforced in code.

For DBAs: Review the physical model for implementation details. Assess indexing strategies, partitioning schemes, and storage requirements.

For Data Scientists: Identify available data elements and their meaning. Understand data quality characteristics and any transformations applied.

11.3 Analysis Methods

11.3.1 Data Model Quality Analysis

Normalization Analysis

Purpose: Assess data model adherence to normal forms.

Process:

1. Check 1NF: Atomic values, no repeating groups
2. Check 2NF: No partial dependencies on composite keys
3. Check 3NF: No transitive dependencies
4. Check BCNF: Every determinant is a candidate key
5. Document justified denormalizations

Outputs:

- Normal form assessment per entity
- Identified anomalies and redundancies
- Denormalization justifications

11.3.2 Data Quality Assessment

Data Quality Dimension Analysis

Purpose: Measure data quality across standard dimensions.

Dimensions:

- **Accuracy:** Data correctly represents real-world values
- **Completeness:** Required data is present
- **Consistency:** Data agrees across systems/time
- **Timeliness:** Data is current and available when needed
- **Validity:** Data conforms to defined formats/rules
- **Uniqueness:** No unintended duplicate records

Outputs:

- Quality scores by dimension and entity
- Identified quality issues
- Remediation recommendations

11.3.3 Data Lineage Analysis

Lineage Impact Analysis

Purpose: Understand data dependencies and change impact.

Inputs:

- Data lineage graph
- Element to be changed
- Type of change

Process:

1. Identify source element in lineage
2. Trace all downstream dependencies
3. Classify impact (schema, logic, reports)
4. Identify affected systems and teams
5. Estimate remediation scope

Outputs:

- Downstream impact list
- Affected systems and processes
- Change coordination requirements

11.4 Implementation Methods

11.4.1 Physical Implementation

Translating data models to physical implementations follows established practices:

Table 15: Implementation Mapping by Technology

Technology	Entity Maps To	Relationship Maps To	Constraint Maps To
RDBMS	Table	Foreign Key	CHECK, UNIQUE, NOT NULL
Document DB	Collection	Embedded/Reference	Validation Schema
Graph DB	Node	Edge	Property Constraints
Key-Value	Key Prefix	Key Structure	Application Logic
Time-Series	Measurement	Tags/Fields	Retention Policies

11.4.2 Data Governance Implementation

Table 16: Governance Control Implementation

Control Area	Mechanisms	Tools
Access Control	RBAC, ABAC, Row-level security	Database permissions, IAM policies
Encryption	TDE, Column-level, TLS	Database encryption, KMS
Masking	Dynamic, Static	Data masking tools, views
Audit	Audit logs, CDC	Database audit, SIEM
Quality	Validation, Profiling	Great Expectations, dbt tests
Lineage	Metadata capture	Atlas, Collibra, DataHub
Retention	TTL, Archival jobs	Lifecycle policies, scheduled jobs

12 Examples

This section provides concrete examples of information views for common scenarios.

12.1 Example 1: E-Commerce Conceptual Data Model

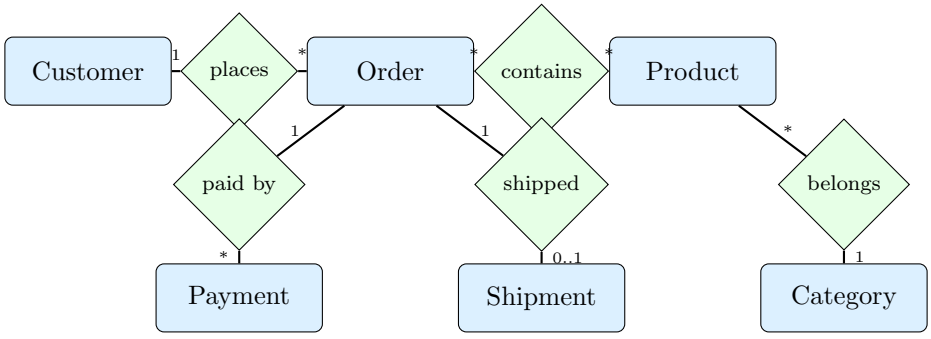


Figure 7: E-Commerce Conceptual Data Model

Description: This conceptual model shows the core business entities for an e-commerce system. Customers place Orders, which contain Products. Orders are paid by Payments and shipped via Shipments. Products belong to Categories. This high-level view communicates business concepts without implementation details.

12.2 Example 2: Logical Data Model (Crow's Foot)

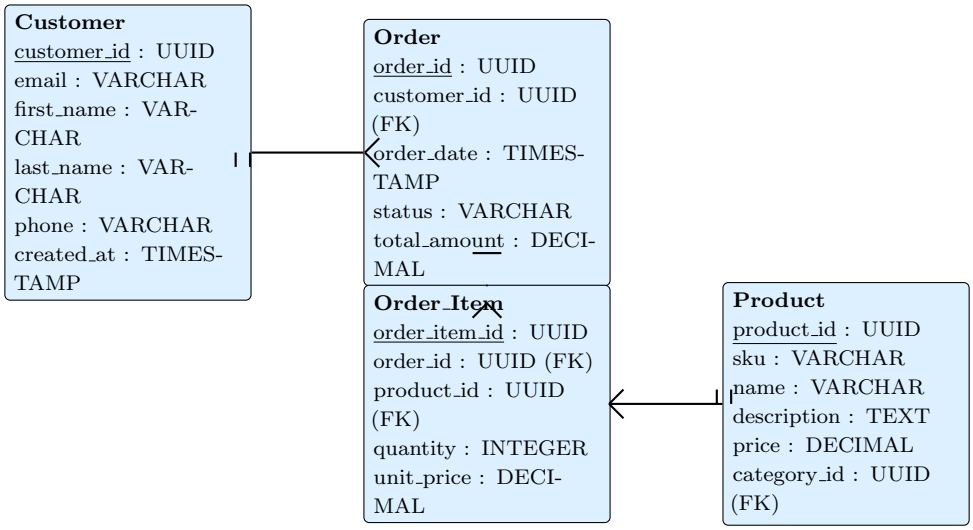


Figure 8: E-Commerce Logical Data Model (Partial)

Description: This logical model shows detailed entity definitions with attributes, data types, and keys. The Crow's Foot notation indicates cardinality: Customer has many Orders, Order has many Order_Items, and Order_Item references one Product. This model is implementation-independent but fully specified.

12.3 Example 3: Data Flow Diagram

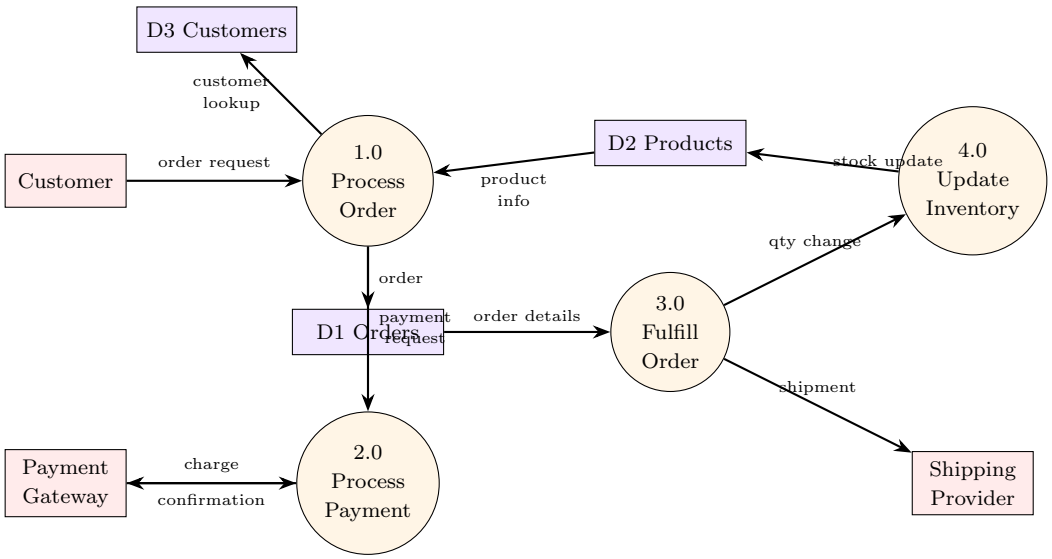


Figure 9: Order Processing Data Flow Diagram

Description: This DFD shows data movement through an order processing system. External entities (Customer, Payment Gateway, Shipping Provider) interact with processes. Data stores (Orders, Products, Customers) persist data. Arrows show data flows with labels indicating the data items transferred.

12.4 Example 4: Data Classification Schema

Table 17: Data Classification Matrix

Level	Examples	Handling Requirements	Access
Public	Product catalog, Public content	No restrictions	Anyone
Internal	Employee directory, Internal docs	Internal network only, no external sharing	Employees
Confidential	Customer PII, Financial data	Encryption required, audit logging, need-to-know	Authorized roles
Restricted	Payment cards, Health records, Auth credentials	Strong encryption, MFA, strict audit, minimal retention	Named individuals

Description: This classification schema defines four sensitivity levels with corresponding handling requirements and access restrictions. Each data element must be classified, and the classification

determines applicable security controls.

13 Notes

13.1 Privacy and Compliance Considerations

Privacy by Design Principles

When developing information architecture, incorporate privacy from the start:

1. **Data Minimization:** Collect only necessary data
2. **Purpose Limitation:** Use data only for stated purposes
3. **Storage Limitation:** Retain only as long as needed
4. **Consent Management:** Track and honor consent preferences
5. **Right to Access:** Enable data subject access requests
6. **Right to Erasure:** Support data deletion requests
7. **Data Portability:** Enable data export in standard formats

13.2 Versioning Considerations

Data models evolve over time and require careful version management:

- Maintain schema version history with change documentation
- Use database migration tools (Flyway, Liquibase, Alembic)
- Plan backward compatibility for schema changes
- Document breaking vs. non-breaking changes
- Coordinate schema changes across dependent systems
- Implement schema registry for event-driven systems

13.3 Tooling Recommendations

Table 18: Recommended Tools by Use Case

Use Case	Recommended Tools
Data Modeling	ERwin, PowerDesigner, Lucidchart, dbdiagram.io, SqlDBM
Data Catalog	Collibra, Alation, DataHub, Apache Atlas, AWS Glue
Data Quality	Great Expectations, dbt tests, Soda, Monte Carlo
Data Lineage	Collibra, Atlan, OpenLineage, dbt
Schema Migration	Flyway, Liquibase, Alembic, Atlas
Data Masking	Delphix, Informatica, AWS DMS

13.4 Common Pitfalls

Common Mistakes to Avoid

1. **Skipping Conceptual Model:** Jumping to physical design without business alignment
2. **Over-Normalization:** Creating performance problems with excessive joins
3. **Under-Normalization:** Creating data anomalies with insufficient normalization
4. **Ignoring Data Quality:** Assuming data will be clean without validation
5. **Missing Audit Trail:** Not tracking who changed what and when
6. **Hardcoded Values:** Embedding reference data in application code
7. **Unclear Ownership:** No defined responsibility for data domains
8. **Security Afterthought:** Adding data protection after design is complete

14 Sources

14.1 Primary References

1. Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., & Stafford, J. (2010). *Documenting Software Architectures: Views and Beyond* (2nd ed.). Addison-Wesley Professional.
2. ISO/IEC/IEEE 42010:2011. *Systems and software engineering — Architecture description.* International Organization for Standardization.
3. Hoberman, S. (2009). *Data Modeling Made Simple* (2nd ed.). Technics Publications.
4. Kimball, R. & Ross, M. (2013). *The Data Warehouse Toolkit* (3rd ed.). Wiley.
5. DAMA International. (2017). *DAMA-DMBOK: Data Management Body of Knowledge* (2nd ed.). Technics Publications.

14.2 Supplementary References

6. Fowler, M. (2002). *Patterns of Enterprise Application Architecture.* Addison-Wesley Professional.
7. Evans, E. (2003). *Domain-Driven Design.* Addison-Wesley Professional.
8. Kleppmann, M. (2017). *Designing Data-Intensive Applications.* O'Reilly Media.
9. Linstedt, D. & Olschimke, M. (2015). *Building a Scalable Data Warehouse with Data Vault 2.0.* Morgan Kaufmann.
10. Reis, J. & Housley, M. (2022). *Fundamentals of Data Engineering.* O'Reilly Media.

14.3 Online Resources

- DAMA International: https://www.dama.org/

- Data Modeling Zone: https://www.datamodelzone.com/
- dbdiagram.io: https://dbdiagram.io/
- Great Expectations: https://greatexpectations.io/
- dbt Documentation: https://docs.getdbt.com/

A Information View Checklist

Use this checklist to verify completeness of information architecture documentation:

Item	Complete?
Data Modeling	
Conceptual model with business stakeholder approval	☐
Logical model with full attribute specifications	☐
Physical model with technology-specific details	☐
Entity relationships with cardinality defined	☐
Constraints documented and justified	☐
Data Flow and Integration	
Data flows identified and documented	☐
Integration patterns specified	☐
Transformations documented	☐
Error handling defined	☐
Data Governance	
Data classification applied to all entities	☐
Data quality rules defined	☐
Retention policies established	☐
Data ownership assigned	☐
Stewardship responsibilities defined	☐
Security and Privacy	
Sensitive data identified and classified	☐
Access control requirements specified	☐
Encryption requirements documented	☐
Privacy requirements addressed (GDPR, CCPA)	☐
Audit requirements defined	☐
Validation	
Correspondence with other views verified	☐
Business stakeholder review completed	☐
Technical review completed	☐
Compliance review completed	☐

B Glossary

Attribute A property of an entity that captures a specific piece of information.

Cardinality The number of instances of one entity that can be associated with instances of another entity.

CDC Change Data Capture; a technique for identifying and capturing changes to data.

Conceptual Model A high-level data model representing business entities and relationships.

Data Catalog A metadata repository that provides information about data assets.

Data Classification Categorization of data based on sensitivity and handling requirements.

Data Flow Movement of data between components, systems, or processes.

Data Lineage The tracking of data's origin, movement, and transformation over time.

Data Quality The degree to which data meets requirements for accuracy, completeness, and consistency.

Data Steward An individual responsible for data quality and governance within a domain.

Entity A distinguishable object or concept about which data is captured.

ETL Extract, Transform, Load; a data integration pattern.

Foreign Key An attribute that references a primary key in another entity.

Logical Model A detailed data model independent of physical implementation.

Master Data Core business entities shared across the organization.

Normalization The process of organizing data to reduce redundancy and improve integrity.

Physical Model An implementation-specific data model for a particular technology.

PII Personally Identifiable Information; data that can identify an individual.

Primary Key An attribute or set of attributes that uniquely identifies an entity instance.

Relationship An association between two or more entities.

Retention Policy Rules governing how long data is kept before archival or deletion.

C Entity Specification Template

```
================================================================================

ENTITY SPECIFICATION
================================================================================
```

```
1. IDENTIFICATION
-----------------
Entity ID:          [ENT-XXX]
Name:               [Entity Name]
Domain:             [Business Domain]
Version:            [X.Y]
Status:             [Draft | Review | Approved | Deprecated]
Owner:              [Data Owner]
Steward:            [Data Steward]
Last Updated:       [Date]

2. BUSINESS DESCRIPTION
-----------------------
Definition:
  [Clear business definition of what this entity represents]

Business Rules:
  - [Business rule 1]
  - [Business rule 2]

Usage Context:
  [How and where this entity is used in business processes]

3. ATTRIBUTES
-------------
+---------------+----------+------+-----+---------+-------+-----------------+

| Name          | Type     | Null | Key | Default | Class | Description
         |
+---------------+----------+------+-----+---------+-------+-----------------+

| [name]        | [type]   | Y/N  | PK  | [val]   | [C/I] | [description
   ]     |
| [name]        | [type]   | Y/N  | FK  | [val]   | [C/I] | [description
   ]     |
| [name]        | [type]   | Y/N  |     | [val]   | [C/I] | [description
   ]     |
+---------------+----------+------+-----+---------+-------+-----------------+


Key: PK=Primary Key, FK=Foreign Key, UK=Unique Key, AK=Alternate Key
Class: C=Confidential, I=Internal, P=Public, R=Restricted

4. RELATIONSHIPS
----------------
```

```
+------------------+---------------+-------------+-------------------------+
| Relationship     | Related Entity | Cardinality | Description             |
+------------------+---------------+-------------+-------------------------+
| [verb phrase]    | [Entity]       | [1:N]       | [description]           |
+------------------+---------------+-------------+-------------------------+

5. CONSTRAINTS
--------------
+------------------+----------+-----------------------------------------+
| Constraint Name  | Type     | Definition                              |
+------------------+----------+-----------------------------------------+
| [name]           | [type]   | [constraint definition]                 |
+------------------+----------+-----------------------------------------+

Types: PK, FK, UK, CHECK, NOT NULL, DEFAULT

6. DATA CLASSIFICATION
----------------------
Sensitivity Level:  [Public | Internal | Confidential | Restricted]
Contains PII:       [Yes | No]
PII Elements:       [List of PII attributes if applicable]
Regulations:        [GDPR | CCPA | HIPAA | PCI-DSS | SOX | etc.]
Encryption:         [At Rest | In Transit | Both | None]

7. DATA QUALITY RULES
---------------------
+----------+-------------+---------------------+----------+----------------+
| Rule ID  | Attribute   | Rule                | Severity | Action         |
+----------+-------------+---------------------+----------+----------------+
| [DQ-XXX] | [attribute] | [rule description]  | [level]  | [action]       |
+----------+-------------+---------------------+----------+----------------+

```

```
Severity: Critical, Major, Minor, Warning
Action: Reject, Flag, Correct, Notify

8. LIFECYCLE
------------
Creation:           [How data is created]
Retention Period:   [Duration]
Archival Policy:    [Policy description]
Deletion Policy:    [Policy description]
Legal Hold:         [Applicable | Not Applicable]

9. PHYSICAL IMPLEMENTATION
--------------------------
Database:           [Database name]
Schema:             [Schema name]
Table:              [Table name]
Partitioning:       [Strategy if applicable]
Indexes:            [Key indexes]

10. LINEAGE
-----------
Source Systems:     [List of source systems]
Target Systems:     [List of consuming systems]
Transformations:    [Key transformations applied]

11. CHANGE HISTORY
------------------
+------------+---------+---------------+-----------------------------------+
| Date       | Version | Author        | Change Description                |
+------------+---------+---------------+-----------------------------------+
| [date]     | [X.Y]   | [name]        | [description]                     |
+------------+---------+---------------+-----------------------------------+


================================================================================
```

Listing 3: Complete Entity Specification Template