

Data Architecture

[Author Name]

Université de Toulon, LIS UMR CNRS 7020

2026-10-03

Table des matières

1 Purpose of this Document	1
2 Data Strategy Overview	1
3 Data Model Overview	1
4 Conceptual Data Model	2
5 Logical Data Model	2
6 Physical Data Model	3
7 Data Flow Architecture	3
8 Data Flow Diagrams	3
9 Data Flow Sequence	4
10 Data Storage Strategy	4
11 Data Governance	4
12 Data Security	5
13 Storage Solutions	5
14 Persistence Strategies	6
14.1 Data Access Patterns	6
14.2 Caching Strategy	6
14.3 Backup Strategy	6
15 Schema Management	6
16 Data Integration	6
17 Template Validation Checklist	7
18 Related Views	7

1 Purpose of this Document

This document defines the data architecture for [System Name], including data models, storage strategies, flows, and governance aspects.

2 Data Strategy Overview

Data Philosophy: [Approach to data management]

Primary Goals:

- [Goal 1, e.g., "Support business analytics"]
 - [Goal 2, e.g., "Ensure data integrity"]
 - [Goal 3, e.g., "Enable real-time processing"]

Key Principles:

- [Principle 1, e.g., "Data is an enterprise asset"]
- [Principle 2, e.g., "Single source of truth"]
- [Principle 3, e.g., "Privacy by design"]

3 Data Model Overview

Domain Models:

- Entity: [Primary entity name]
Purpose: [Business function]
Attributes: [Key fields]
Relations: [Connections to other entities]
Volume: [Expected data size]
Growth: [Projected growth rate]
- Entity: [Secondary entity name]
Similar structure as above

Storage Types:

Operational:

```

- Type: [Database type, e.g.,
"PostgreSQL"]
  Purpose: [Primary use cases]
  Scale: [Size/operations per
second]
  Key Tables: [Critical data
elements]
  Indices: [Important indices]

Analytical:
- Type: [Database type, e.g.,
"Snowflake"]
  Purpose: [Primary use cases]
  Scale: [Size/operations]
  Key Marts: [Important data marts]
  Access Patterns: [Query patterns]

```

```

LINE_ITEM {
  string id PK
  string order_id FK
  string product_id FK
  int quantity
  decimal unit_price
  decimal total_price
}

PRODUCT {
  string id PK
  string name
  string description
  decimal price
  int inventory_count
}

```

4 Conceptual Data Model

```

erDiagram
    CUSTOMER ||--o{ ORDER : places
    ORDER ||--|{ LINE_ITEM : contains
    PRODUCT ||--o{ LINE_ITEM :
"ordered in"
    CUSTOMER }|..|{ PAYMENT_METHOD :
has
    ORDER ||--|| INVOICE : generates
    INVOICE ||--o{ PAYMENT : receives

    CUSTOMER {
        string id PK
        string name
        string email
        date created_at
    }

    ORDER {
        string id PK
        string customer_id FK
        date created_at
        string status
        decimal total_amount
    }

```

5 Logical Data Model

```

@startuml
!define table(x) class x <<
(T,#FFAAAA) >>
!define primary_key(x) <b>x</b>
!define foreign_key(x) <i>x</i>

table(CUSTOMER) {
  primary_key(id): UUID
  name: VARCHAR(100)
  email: VARCHAR(100)
  created_at: TIMESTAMP
  status: VARCHAR(20)
}

table(ORDER) {
  primary_key(id): UUID
  foreign_key(customer_id): UUID
  created_at: TIMESTAMP
  status: VARCHAR(20)
  total_amount: DECIMAL(10,2)
}

table(LINE_ITEM) {

```

```

    primary_key(id): UUID
    foreign_key(order_id): UUID
    foreign_key(product_id): UUID
    quantity: INTEGER
    unit_price: DECIMAL(10,2)
    total_price: DECIMAL(10,2)
}

table(PRODUCT) {
    primary_key(id): UUID
    name: VARCHAR(100)
    description: TEXT
    price: DECIMAL(10,2)
    category: VARCHAR(50)
    inventory_count: INTEGER
}

CUSTOMER "1" -- "0..*" ORDER : places
ORDER "1" -- "1..*" LINE_ITEM :
contains
PRODUCT "1" -- "0..*" LINE_ITEM :
"ordered in"
@enduml

```

6 Physical Data Model

```

-- Example schema definition for a
key table

CREATE TABLE customers (
    id UUID PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL
    UNIQUE,
    created_at TIMESTAMP NOT NULL
    DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) NOT NULL
    DEFAULT 'active',
    last_login TIMESTAMP,
    preferences JSONB
);

CREATE INDEX idx_customers_email ON

```

```

customers(email);
CREATE INDEX idx_customers_status ON
customers(status);

```

7 Data Flow Architecture

```

@startuml
    database "Operational DB" as odb
    database "Analytics DB" as adb
    queue "Message Queue" as mq
    component "ETL Pipeline" as etl
    component "Data API" as api
    actor "Users" as users
    component "Applications" as apps
    component "BI Tools" as bi

    users --> apps
    apps --> api
    api --> odb : CRUD operations
    odb --> mq : Change events
    mq --> etl : Batch processing
    etl --> adb : Data loading
    bi --> adb : Analytics queries
    api --> adb : Reporting data
@enduml

```

8 Data Flow Diagrams

```

graph LR
    A[Source Systems] --> B[Data Ingestion]
    B --> C[Data Processing]
    C --> D[Data Storage]
    D --> E[Data Access]
    E --> F[Data Consumers]

    subgraph "Data Pipeline"
        B
        C
        D
    end
end

```

9 Data Flow Sequence

```
@startuml
participant "User Interface" as UI
participant "API Layer" as API
participant "Cache" as Cache
participant "Database" as DB
participant "Event Bus" as Events
participant "Analytics" as Analytics

UI -> API: Request Data
API -> Cache: Check Cache
alt data in cache
    Cache --> API: Return Cached Data
else data not in cache
    API -> DB: Query Data
    DB --> API: Return Data
    API -> Cache: Update Cache
end
API --> UI: Return Response
API -> Events: Publish Event
Events -> Analytics: Process Event
@enduml
```

10 Data Storage Strategy

```
Primary Data Stores:
- Type: [Storage type, e.g., "Relational Database"]
  Technology: [Specific technology, e.g., "PostgreSQL 13"]
  Purpose: [Primary use cases]
  Data Types: [Key data categories stored]
  Scaling: [Horizontal/Vertical approach]
  Backups: [Strategy]

- Type: [Storage type, e.g., "Document Database"]
  Technology: [Specific technology, e.g., "MongoDB 5"]
  Purpose: [Primary use cases]
```

```
Data Types: [Key data categories stored]
```

```
Scaling: [Approach]
```

```
Backups: [Strategy]
```

```
Caching Strategy:
```

```
- Level: [Application/Database/API/ etc.]
```

```
Technology: [e.g., "Redis 6"]
```

```
Purpose: [Use cases]
```

```
TTL: [Time to live policies]
```

```
Invalidation: [Strategy]
```

```
Data Lifecycle:
```

```
- Hot Data: [Storage and retention]
```

```
- Warm Data: [Storage and retention]
```

```
- Cold Data: [Storage and retention]
```

```
- Archival: [Strategy and technology]
```

11 Data Governance

```
Data Classification:
```

```
- PII Data:
```

```
Examples: [List of PII elements]
```

```
Protection: [Controls applied]
```

```
Access: [Who can access]
```

```
Retention: [How long kept]
```

```
- Business Critical:
```

```
Examples: [List of examples]
```

```
Protection: [Controls applied]
```

```
Access: [Who can access]
```

```
Retention: [How long kept]
```

```
- General Data:
```

```
Protection: [Controls applied]
```

```
Retention: [How long kept]
```

```
Data Lifecycle:
```

```
Collection:
```

```
Methods: [How data is collected]
```

```
Consent: [How consent is managed]
```

```

Validation: [Validation approaches]

Processing:
  Controls: [Processing controls]
  Transformations: [Key transformations]
  Quality: [Quality measures]

Storage:
  Encryption: [At-rest encryption approach]
  Segregation: [Data isolation approach]
  Backup: [Backup strategy]

Deletion:
  Process: [How data is deleted]
  Verification: [How deletion is verified]
  Records: [Deletion tracking]

Quality Controls:
  Validation:
    Input: [Input validation approaches]
    Business Rules: [Rule validation]
    Integrity: [Referential integrity]

  Monitoring:
    Metrics: [Data quality metrics]
    Alerts: [When alerts are triggered]
    Remediation: [How issues are fixed]

  Governance:
    Ownership: [Data ownership model]
    Stewardship: [Stewardship approach]
    Policies: [Key governance policies]

```

12 Data Security

```

Encryption:
  Transit: [Encryption methods for data in transit]
  Rest: [Encryption methods for data at rest]
  Keys: [Key management approach]

Access Control:
  Authentication: [Methods]
  Authorization: [Models - RBAC, ABAC, etc.]
  Rows/Columns: [Data filtering approach]
  Masking: [Data masking strategy]

Audit:
  Changes: [How data changes are tracked]
  Access: [How data access is logged]
  Reports: [Key audit reports]
  Retention: [How long audit data is kept]

```

13 Storage Solutions

Data Type	Storage Solution	Justification	Usage Patterns
Transactional	[Solution]	[Rationale]	[Access patterns]
Document	[Solution]	[Rationale]	[Access patterns]
Analytical	[Solution]	[Rationale]	[Access patterns]
Binary/Blob	[Solution]	[Rationale]	[Access patterns]

Data Type	Storage Solution	Justification	Usage Patterns
Time Series	[Solution]	[Rationale]	[Access patterns]
Cache	[Solution]	[Rationale]	[Access patterns]

14 Persistence Strategies

14.1 Data Access Patterns

- **Read Patterns**
 - [List common read patterns]
 - [List performance optimizations]
- **Write Patterns**
 - [List common write patterns]
 - [List durability guarantees]

14.2 Caching Strategy

- **Cache Levels**
 - [List each cache level]
 - [Purpose and invalidation approach]
- **Cache Consistency**
 - [Approach to maintaining consistency]
 - [Trade-offs made]

14.3 Backup Strategy

- **Backup Types**
 - [Full/Incremental/Differential approach]
 - [Schedule and retention]
- **Recovery Testing**
 - [Testing approach]
 - [Success criteria]

15 Schema Management

Version Control:

- Tool: [Schema migration tool, e.g., "Flyway"]
- Process: [Migration approach]
- Rollback: [How to handle failures]

Testing: [How migrations are tested]

Migration Strategy:

Planning:

- [Assessment steps]
- [Impact analysis]
- [Stakeholder approval]

Testing:

- [Test environments]
- [Data integrity checks]
- [Performance testing]

Deployment:

- [Deployment process]
- [Downtime requirements]
- [Rollback procedures]

Verification:

- [Post-deployment checks]
- [Monitoring approach]
- [Success metrics]

16 Data Integration

ETL/ELT Processes:

Tools: [Technologies used]

Schedules: [Frequency of processing]

Monitoring: [How processes are monitored]

APIs:

Data APIs: [API-based data access]

Formats: [Data exchange formats]

Standards: [API standards]

Event-driven:

Event Types: [Key data events]

Consumers: [Event consumers]

Guarantees: [Delivery guarantees]

17 Template Validation Checklist

- ☐ Data strategy defined
- ☐ Conceptual data model documented
- ☐ Logical data model documented
- ☐ Data flows mapped
- ☐ Storage strategies specified
- ☐ Data governance defined
- ☐ Security measures specified
- ☐ Schema management approach described
- ☐ Integration patterns identified
- ☐ Team review completed

18 Related Views

- System Context - System boundaries
- Logical Architecture - Overall system structure
- Integration Architecture - Data integration interfaces