UDM Framework Implementation Guide

Step-by-Step Implementation Instructions

Introduction

This guide provides detailed, actionable steps for implementing the UDM Framework across three common scenarios: greenfield implementations, legacy system migrations, and hybrid integrations. It includes prerequisites, timelines, and best practices from real-world deployments.

Prerequisites and Readiness Checklist

Technical Prerequisites

Infrastructure:

- [ ] SQL Server 2017+ (or Azure SQL Database)
- [ ] Azure subscription (for ADF, if cloud-based)
- [ ] Network connectivity between source systems and DWH
- [ ] Sufficient storage (estimate 3-5x source data size)
- [ ] Compute resources (CPU/RAM based on data volumes)

Access and Permissions:

- [ ] DBA access to create databases and users
- [ ] Read access to all source systems
- [ ] Azure portal access (if using ADF)
- [ ] Source system documentation and schemas

Skills and Resources:

- [ ] SQL Developer (T-SQL, stored procedures, views)
- [ ] ETL Developer (Azure Data Factory or SSIS experience)

- [ ] Data Architect (data modeling, requirements gathering)
- [ ] BI Developer (Power BI, Tableau, or similar)
- [ ] Project Manager (coordination, timeline management)

Organizational Readiness

Stakeholder Alignment:

- [ ] Executive sponsorship secured
- [ ] Business requirements documented
- [ ] Success criteria defined
- [ ] Budget approved
- [ ] Timeline agreed upon

Data Governance:

- [ ] Data ownership identified
- [ ] Data quality standards defined
- [ ] Security and compliance requirements documented
- [ ] Data retention policies established

Scenario 1: Greenfield Implementation

Overview

Definition: Building the UDM Framework from scratch for a new data warehouse.

Timeline: 6-12 weeks (depending on number of sources)

Implementation Timeline:

```
gantt
    title Greenfield Implementation Timeline (Medium: 10-14 weeks)
    dateFormat YYYY-MM-DD
    section Phase 1: Foundation
    Environment Setup           :done, phase1, 2024-01-01, 2w
    Framework Tables            :done, phase1b, 2024-01-01, 2w
    section Phase 2: First Source
    Pilot Data Source          :active, phase2, 2024-01-15, 2w
```

```
    section Phase 3: Core Sources
    Core Dimensions            :phase3a, 2024-01-29, 2w
    Primary Facts              :phase3b, 2024-02-12, 2w
    Marketing Dimensions       :phase3c, 2024-02-26, 1w
    Finance Tables             :phase3d, 2024-03-05, 1w
    section Phase 4: Reporting
    Reporting Layer            :phase4, 2024-03-12, 2w
    section Phase 5: Production
    Production Rollout        :crit, phase5, 2024-03-26, 2w
```

Best For:

- New organizations or divisions
- Replacing legacy DWH completely
- Cloud-native implementations

Phase 1: Environment Setup (Week 1-2)

Step 1.1: Create Database Instances

SQL Server (On-Premises or Azure SQL):

```
-- Create Stage database
CREATE DATABASE [DS_Stage]
ON PRIMARY
( NAME = N'DS_Stage', FILENAME = N'F:\data\DS_Stage.mdf',
  SIZE = 1GB, MAXSIZE = UNLIMITED, FILEGROWTH = 256MB )
LOG ON
( NAME = N'DS_Stage_log', FILENAME = N'G:\log\DS_Stage_log.ldf',
  SIZE = 512MB, MAXSIZE = 10GB, FILEGROWTH = 128MB )
GO

-- Create DWH database
CREATE DATABASE [DS_DWH]
ON PRIMARY
( NAME = N'DS_DWH', FILENAME = N'F:\data\DS_DWH.mdf',
  SIZE = 5GB, MAXSIZE = UNLIMITED, FILEGROWTH = 512MB )
LOG ON
( NAME = N'DS_DWH_log', FILENAME = N'G:\log\DS_DWH_log.ldf',
  SIZE = 1GB, MAXSIZE = 50GB, FILEGROWTH = 256MB )
GO
```

```
-- Create Reporting database
CREATE DATABASE [DS_Reporting]
ON PRIMARY
( NAME = N'DS_Reporting', FILENAME = N'F:\data\DS_Reporting.mdf',
  SIZE = 2GB, MAXSIZE = UNLIMITED, FILEGROWTH = 256MB )
LOG ON
( NAME = N'DS_Reporting_log', FILENAME = N'G:\log\DS_Reporting_log.ldf',
  SIZE = 512MB, MAXSIZE = 10GB, FILEGROWTH = 128MB )
GO
```

Configure Database Settings:

```
-- DS_DWH configuration
USE [DS_DWH]
GO
ALTER DATABASE [DS_DWH] SET RECOVERY SIMPLE
ALTER DATABASE [DS_DWH] SET AUTO_UPDATE_STATISTICS ON
ALTER DATABASE [DS_DWH] SET PAGE_VERIFY CHECKSUM
GO
```

Step 1.2: Create Schema Structure

DS_Stage Schemas:

```
USE [DS_Stage]
GO

CREATE SCHEMA [History]
GO
CREATE SCHEMA [Reference]
GO
-- Add additional source-specific schemas as needed
```

DS_DWH Schemas:

```
USE [DS_DWH]
GO
```

```
-- Business domains
CREATE SCHEMA [Sales]
GO
CREATE SCHEMA [Finance]
GO
CREATE SCHEMA [Product]
GO
CREATE SCHEMA [Contact]
GO
CREATE SCHEMA [Marketing]
GO

-- Framework domains
CREATE SCHEMA [Reference]
GO
CREATE SCHEMA [Fallout]
GO
CREATE SCHEMA [Flow]
GO
CREATE SCHEMA [Bridge]
GO
CREATE SCHEMA [Logging]
GO
```

DS_Reporting Schemas:

```
USE [DS_Reporting]
GO

CREATE SCHEMA [Dim]
GO
CREATE SCHEMA [Fact]
GO
CREATE SCHEMA [Model]
GO
-- Add industry-specific schemas as needed
```

Step 1.3: Deploy Framework Tables

Reference Domain Tables:

```
USE [DS_DWH]
GO

-- SourceSystem table (master configuration)
CREATE TABLE [Reference].[SourceSystem](
    [SourceSystemId] [bigint] IDENTITY(1,1) NOT NULL,
    [SourceDescription] [nvarchar](100) NULL,
    [SourceDatabase] [nvarchar](100) NULL,
    [SourceSchema] [nvarchar](50) NULL,
    [SourceTable] [nvarchar](100) NULL,
    [SourceIdentifierColumn] [nvarchar](100) NULL,
    [TargetSchema] [nvarchar](50) NULL,
    [TargetTable] [nvarchar](100) NULL,
    [TargetIdentifierColumn] [nvarchar](100) NULL,
    [ValidFromDate] [date] NULL,
    [ValidToDate] [date] NULL,
    [ParentSourceSystemId] [bigint] NULL,
    [SourceKeyColumn] [nvarchar](100) NULL,
    [SourceValueColumn] [nvarchar](100) NULL,
    CONSTRAINT [Pk_SourceSystem] PRIMARY KEY ([SourceSystemId])
)
GO

-- SourceColumnMapping table
CREATE TABLE [Reference].[SourceColumnMapping](
    [SourceColumnMappingId] [bigint] IDENTITY(1,1) NOT NULL,
    [SourceSystemId] [bigint] NOT NULL,
    [SourceColumnName] [nvarchar](100) NULL,
    [TargetAlias] [nvarchar](100) NOT NULL,
    CONSTRAINT [PK_SourceColumnMapping] PRIMARY KEY ([SourceColumnMapping
    CONSTRAINT [FK_SourceColumnMapping_SourceSystem]
        FOREIGN KEY ([SourceSystemId])
        REFERENCES [Reference].[SourceSystem]([SourceSystemId])
)
GO

-- PropertyReference table
CREATE TABLE [Reference].[PropertyReference](
    [PropertyReferenceId] [bigint] IDENTITY(1,1) NOT NULL,
    [PropertyTypeId] [int] NOT NULL,
    [SourceSystemId] [bigint] NOT NULL,
```

```
    [SourceColumn] [nvarchar](100) NOT NULL,
    [Ignore] [bit] NOT NULL DEFAULT 1,
    CONSTRAINT [PK_PropertyReference] PRIMARY KEY ([PropertyReferenceId])
    CONSTRAINT [FK_PropertyReference_SourceSystem]
        FOREIGN KEY ([SourceSystemId])
        REFERENCES [Reference].[SourceSystem]([SourceSystemId])
)
GO
```

Fallout Domain Tables:

```
-- ErrorCode table
CREATE TABLE [Fallout].[ErrorCode](
    [ErrorCodeId] [int] NOT NULL,
    [ErrorDescription] [nvarchar](200) NULL,
    [ErrorCategory] [nvarchar](50) NULL,
    CONSTRAINT [PK_ErrorCode] PRIMARY KEY ([ErrorCodeId])
)
GO

-- Populate standard error codes
INSERT INTO [Fallout].[ErrorCode] VALUES
(11, 'Missing Foreign Key: Lead', 'Referential Integrity'),
(12, 'Missing Foreign Key: Channel', 'Referential Integrity'),
(13, 'Missing Foreign Key: OrderStatus', 'Referential Integrity'),
(14, 'Missing Foreign Key: Product', 'Referential Integrity'),
(15, 'Missing Foreign Key: Customer', 'Referential Integrity'),
(20, 'Invalid Data Type', 'Data Format'),
(21, 'Invalid Date Format', 'Data Format'),
(30, 'Business Rule Violation', 'Business Logic'),
(40, 'Duplicate Record Detected', 'Data Quality')
GO

-- Fallout table
CREATE TABLE [Fallout].[Fallout](
    [FalloutId] [int] IDENTITY(1,1) NOT NULL,
    [ErrorCodeId] [int] NOT NULL,
    [SourceSystemId] [bigint] NOT NULL,
    [SourceSystemIdentifier] [nvarchar](100) NOT NULL,
    [InsertDate] [datetime] NOT NULL,
    [SolveDate] [datetime] NULL,
```

```
    CONSTRAINT [PK_Fallout] PRIMARY KEY ([FalloutId]),
    CONSTRAINT [FK_Fallout_ErrorCode]
        FOREIGN KEY ([ErrorCodeId])
        REFERENCES [Fallout].[ErrorCode]([ErrorCodeId]),
    CONSTRAINT [FK_Fallout_SourceSystem]
        FOREIGN KEY ([SourceSystemId])
        REFERENCES [Reference].[SourceSystem]([SourceSystemId])
)
GO
```

Step 1.4: Deploy Framework Stored Procedures

History Tracking Procedures (in DS_Stage):

```
USE [DS_Stage]
GO

-- CreateHistoryTable procedure (full code in DS_Stage.sql)
CREATE PROCEDURE [dbo].[CreateHistoryTable]
    @table_name SYSNAME
AS
BEGIN
    -- Creates History.TableName for change tracking
    -- (See DS_Stage.sql for complete implementation)
END
GO

-- BuildHistoryTracking procedure
CREATE PROCEDURE [dbo].[BuildHistoryTracking]
    @TableFull NVARCHAR(100)
AS
BEGIN
    DECLARE @Table_Name NVARCHAR(100) =
        RIGHT(@TableFull, CHARINDEX('.', REVERSE(@TableFull), 1) - 1)

    IF OBJECT_ID('[History].' + @Table_Name) IS NULL
    BEGIN
        EXEC [dbo].[CreateHistoryTable] @TableFull
    END
```

```
END
GO
```

Initialization Procedures (in DS_DWH):

```
USE [DS_DWH]
GO

-- InitSourceColumnMapping procedure
CREATE PROCEDURE [Reference].[InitSourceColumnMapping]
AS
BEGIN
    -- Generates SourceColumnMapping rows for new SourceSystems
    -- (Implementation omitted for brevity - populates from metadata)
END
GO

-- InitPropertyReference procedure
CREATE PROCEDURE [Reference].[InitPropertyReference]
AS
BEGIN
    -- Discovers properties from EAV sources
    -- (Implementation omitted for brevity)
END
GO

-- ReadSourceTable_NewRecords procedure
CREATE PROCEDURE [dbo].[ReadSourceTable_NewRecords]
    @SourceSystemId BIGINT
AS
BEGIN
    -- Extracts new records to Flow tables
    -- (See ETL Patterns document for logic)
END
GO
```

Phase 2: First Data Source Integration (Week 3-4)

Step 2.1: Choose Pilot Data Source

Criteria for First Source:

- Simple structure (5-20 columns)
- Small-medium size (< 1M rows)
- High business value
- Available documentation
- Cooperative source system owner

Example: Customer dimension from CRM

Step 2.2: Load Raw Data to Stage

Create Stage Table:

```
USE [DS_Stage]
GO

CREATE TABLE [dbo].[CRM_Customers](
    [CustomerId] [int] NOT NULL PRIMARY KEY,
    [CustomerName] [nvarchar](200) NULL,
    [Email] [nvarchar](200) NULL,
    [Phone] [nvarchar](50) NULL,
    [CreatedDate] [datetime] NULL,
    [ModifiedDate] [datetime] NULL,
    [IsActive] [bit] NULL
)
GO

-- Set up history tracking
EXEC [dbo].[BuildHistoryTracking] '[dbo].[CRM_Customers]'
GO
```

Load Data (via SSIS, ADF, or simple script):

```
-- Example: Linked server load
INSERT INTO DS_Stage.dbo.CRM_Customers
SELECT * FROM [CRM_Server].[CRM_DB].[dbo].[Customers]
```

Step 2.3: Create DWH Target Table

```
USE [DS_DWH]
GO

CREATE TABLE [Contact].[Customer](
    [CustomerId] [bigint] IDENTITY(1,1) NOT NULL,
    [CustomerNK] [nvarchar](100) NOT NULL,
    [CustomerName] [nvarchar](200) NULL,
    [Email] [nvarchar](200) NULL,
    [Phone] [nvarchar](50) NULL,
    [IsActive] [bit] NULL,
    [ValidFromDate] [date] NOT NULL,
    [ValidToDate] [date] NOT NULL,
    [SourceSystemId] [bigint] NOT NULL,
    [SourceSystemIdentifier] [nvarchar](100) NULL,
    CONSTRAINT [PK_Customer] PRIMARY KEY ([CustomerId])
)
GO

CREATE NONCLUSTERED INDEX IX_Customer_NK
    ON [Contact].[Customer]([CustomerNK])
GO

CREATE NONCLUSTERED INDEX IX_Customer_ValidToDate
    ON [Contact].[Customer]([ValidToDate])
GO
```

Step 2.4: Configure Metadata

Add SourceSystem Entry:

```
INSERT INTO [Reference].[SourceSystem] (
    SourceDescription,
    SourceDatabase,
    SourceSchema,
    SourceTable,
    SourceIdentifierColumn,
    TargetSchema,
    TargetTable,
    TargetIdentifierColumn,
    ValidFromDate,
```

```
    ValidToDate
) VALUES (
    'CRM Customers',
    'DS_Stage',
    'dbo',
    'CRM_Customers',
    'CustomerId',
    'Contact',
    'Customer',
    'CustomerId',
    '2024-01-01',
    '9999-12-31'
)
-- Note the SourceSystemId assigned (e.g., 1)
```

Initialize Column Mappings:

```
-- Run initialization procedure
EXEC [Reference].[InitSourceColumnMapping]

-- Manually map columns
UPDATE [Reference].[SourceColumnMapping]
SET SourceColumnName = 'CustomerId'
WHERE SourceSystemId = 1 AND TargetAlias = 'SourceSystemIdentifier'

UPDATE [Reference].[SourceColumnMapping]
SET SourceColumnName = 'CustomerName'
WHERE SourceSystemId = 1 AND TargetAlias = 'CustomerName'

UPDATE [Reference].[SourceColumnMapping]
SET SourceColumnName = 'Email'
WHERE SourceSystemId = 1 AND TargetAlias = 'Email'

-- Continue for all columns...
```

Step 2.5: Create ADF Pipeline (or SSIS Package)

Azure Data Factory Setup:

1. **Create Linked Services**:

- Source: DS_Stage database connection
- Target: DS_DWH database connection

2. **Create Datasets**:
   - Flow table dataset (parameterized)
   - Target table dataset (Contact.Customer)
   - Fallout table dataset
3. **Create Dataflow**: `LoadCustomers`
   - Source: Flow table
   - Transformations: Column mappings, validations
   - Sinks: Contact.Customer, Fallout.Fallout
4. **Create Pipeline**: `Load_Contact_Customer`
   - Activity 1: Execute SP `ReadSourceTable_NewRecords` (@SourceSystemId = 1)
   - Activity 2: Execute Dataflow `LoadCustomers`
   - Activity 3: Log completion

Step 2.6: Test and Validate

Initial Load Test:

```
-- Populate Flow table
EXEC [dbo].[ReadSourceTable_NewRecords] @SourceSystemId = 1

-- Check Flow table
SELECT * FROM [Flow].[Contact_Customer_1]

-- Run ADF pipeline (manually trigger)

-- Validate results
SELECT COUNT(*) FROM [Contact].[Customer] WHERE SourceSystemId = 1
SELECT COUNT(*) FROM [Fallout].[Fallout] WHERE SourceSystemId = 1

-- Verify data quality
SELECT TOP 100 * FROM [Contact].[Customer] WHERE SourceSystemId = 1
```

Incremental Load Test:

```
-- Add new record in Stage
INSERT INTO DS_Stage.dbo.CRM_Customers VALUES (999, 'Test Customer', 'tes

-- Run pipeline again
EXEC [dbo].[ReadSourceTable_NewRecords] @SourceSystemId = 1
-- (Then trigger ADF pipeline)

-- Verify new record loaded
SELECT * FROM [Contact].[Customer] WHERE SourceSystemIdentifier = '999'
```

Phase 3: Expand to Core Sources (Week 5-8)

Priority Order:

1. Core dimensions (Customers, Products)
2. Primary facts (Orders, Transactions)
3. Marketing dimensions (Campaigns, Channels)
4. Properties (EAV attributes)

Repeat Steps 2.2-2.6 for each additional source, following these guidelines:

Week 5: Add Products and Orders
Week 6: Add Marketing Dimensions
Week 7: Add Finance Tables
Week 8: Add Properties and Bridge Tables

Phase 4: Reporting Layer (Week 9-10)

Step 4.1: Create Dimension Views

```
USE [DS_Reporting]
GO

-- Current-state customer dimension
CREATE VIEW [Dim].[Customer] AS
SELECT
    c.CustomerId,
    c.CustomerNK,
    c.CustomerName,
```

```
    c.Email,
    c.Phone,
    c.IsActive
FROM DS_DWH.Contact.Customer c
WHERE c.ValidToDate = '9999-12-31'  -- Current records only
GO
```

Step 4.2: Create Fact Views

```
-- Orders with denormalized dimensions
CREATE VIEW [Fact].[Orders] AS
SELECT
    o.OrderId,
    o.OrderDate,
    c.CustomerName,
    p.ProductName,
    o.Amount,
    ch.ChannelName
FROM DS_DWH.Sales.Order o
LEFT JOIN DS_DWH.Contact.Customer c ON o.CustomerId = c.CustomerId
LEFT JOIN DS_DWH.Product.Product p ON o.ProductId = p.ProductId
LEFT JOIN DS_DWH.Marketing.Channel ch ON o.ChannelId = ch.ChannelId
GO
```

Step 4.3: Connect BI Tools

Power BI Setup:

1. Connect to DS_Reporting database
2. Import or DirectQuery mode
3. Build semantic model
4. Create initial dashboards

Phase 5: Production Rollout (Week 11-12)

Step 5.1: Schedule Pipelines

Daily ETL Schedule:

```
00:00 - Stage Layer Sync (from source systems)
02:00 - DWH Load - Dimensions
03:00 - DWH Load - Facts
04:00 - DWH Load - Properties
05:00 - Reporting Layer Refresh
06:00 - BI Cache Refresh
```

Step 5.2: Set Up Monitoring

Fallout Monitoring Query:

```
-- Daily fallout check
CREATE VIEW [Fallout].[VW_DailyFalloutReport] AS
SELECT
    CAST(f.InsertDate AS DATE) as FalloutDate,
    ss.SourceDescription,
    ec.ErrorDescription,
    COUNT(*) as RecordCount
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 CAST(f.InsertDate AS DATE), ss.SourceDescription, ec.ErrorDescri
```

Pipeline Monitoring (ADF):

- Configure email alerts for failures
- Set up Azure Monitor dashboards
- Define SLAs for pipeline completion

Step 5.3: Documentation and Training

Deliver:

- ☐ Data dictionary
- ☐ ETL pipeline documentation
- ☐ Fallout resolution procedures
- ☐ BI tool user guides
- ☐ Support contacts and escalation paths

Training Sessions:

- [ ] Technical team (ETL maintenance)
- [ ] Data stewards (fallout resolution)
- [ ] Business users (BI tools)

Scenario 2: Legacy System Migration

Overview

Definition: Migrating from an existing data warehouse to the UDM Framework.

Timeline: 12-20 weeks (depending on complexity)

Best For:

- Replacing aging on-premises DWH
- Modernizing ETL processes
- Moving to cloud infrastructure

Phase 1: Assessment (Week 1-2)

Step 1.1: Document Current State

Inventory:

- [ ] List all source systems and tables
- [ ] Document current ETL processes
- [ ] Catalog existing reports and dashboards
- [ ] Identify data quality issues
- [ ] Map current users and access patterns

Analysis:

- [ ] Data volumes and growth rates
- [ ] ETL execution times
- [ ] Report usage statistics
- [ ] Pain points and limitations

Step 1.2: Define Migration Strategy

Parallel Run Approach (Recommended):

```
flowchart TD
    Start[Legacy DWH<br/>In Production] --> Phase1[Phase 1: Parallel Run<

    Phase1 --> Legacy1[Legacy DWH<br/>Continues Operations]
    Phase1 --> BuildUDM[Build UDM Framework<br/>In Parallel]
    Phase1 --> Compare[Compare Outputs<br/>Regular Validation]

    Compare --> Phase2{Validation<br/>Successful?}

    Phase2 -->|Yes| Phase2Cutover[Phase 2: Gradual Cutover]
    Phase2 -->|No| FixIssues[Fix Issues<br/>Continue Comparison]
    FixIssues --> Phase2

    Phase2Cutover --> Migrate1[Migrate Report 1<br/>Validate]
    Migrate1 --> Migrate2[Migrate Report 2<br/>Validate]
    Migrate2 --> MigrateN[Migrate Report N<br/>Validate]

    MigrateN --> Phase3{All Reports<br/>Migrated?}

    Phase3 -->|No| Migrate2
    Phase3 -->|Yes| Phase3Decom[Phase 3: Decommission]

    Phase3Decom --> Archive[Archive Legacy<br/>Audit Period]
    Archive --> Retire[Retire Infrastructure]

    style Start fill:#e1f5ff
    style Phase1 fill:#fff4e1
    style Phase2Cutover fill:#e8f5e9
    style Phase3Decom fill:#f3e5f5
    style Retire fill:#c8e6c9
```

Phase 2: Parallel Build (Week 3-12)

Follow Greenfield Steps (Scenario 1), but with these additions:

Step 2.1: Mapping Legacy to UDM

For each legacy table, determine UDM equivalent:

Legacy Table	Legacy Schema	UDM Table	UDM Schema	Notes
tbl_Customers	dbo	Customer	Contact	Rename, add timeslices
tbl_Products	dbo	Product	Product	Add properties table
tbl_Orders	dbo	Order	Sales	Normalize FKs
OrderDetails	dbo	OrderLine	Sales	New table (line items)

Step 2.2: Data Reconciliation

Create Comparison Views:

```
-- Compare customer counts
SELECT 'Legacy' as Source, COUNT(*) as CustomerCount
FROM Legacy_DWH.dbo.tbl_Customers
UNION ALL
SELECT 'UDM' as Source, COUNT(DISTINCT CustomerNK) as CustomerCount
FROM DS_DWH.Contact.Customer

-- Compare order totals by month
SELECT
    'Legacy' as Source,
    YEAR(OrderDate) as OrderYear,
    MONTH(OrderDate) as OrderMonth,
    COUNT(*) as OrderCount,
    SUM(Amount) as TotalAmount
FROM Legacy_DWH.dbo.tbl_Orders
GROUP BY YEAR(OrderDate), MONTH(OrderDate)
UNION ALL
SELECT
    'UDM' as Source,
    YEAR(OrderDate),
    MONTH(OrderDate),
```

```
    COUNT(*),
    SUM(Amount)
FROM DS_DWH.Sales.Order
GROUP BY YEAR(OrderDate), MONTH(OrderDate)
ORDER BY OrderYear, OrderMonth, Source
```

Phase 3: Report Migration (Week 13-16)

Step 3.1: Prioritize Reports

Migration Order:

1. Low-complexity, high-usage reports
2. Executive dashboards
3. Operational reports
4. Ad-hoc analysis queries

Step 3.2: Migrate Each Report

For each report:

1. Analyze current SQL/queries
2. Map to UDM views
3. Rebuild in BI tool
4. Compare outputs side-by-side
5. User acceptance testing
6. Cutover

Phase 4: Decommission Legacy (Week 17-20)

Week 17-18: Final validation and cutover
Week 19: Monitoring and stabilization
Week 20: Archive legacy, retire infrastructure

Scenario 3: Hybrid Integration

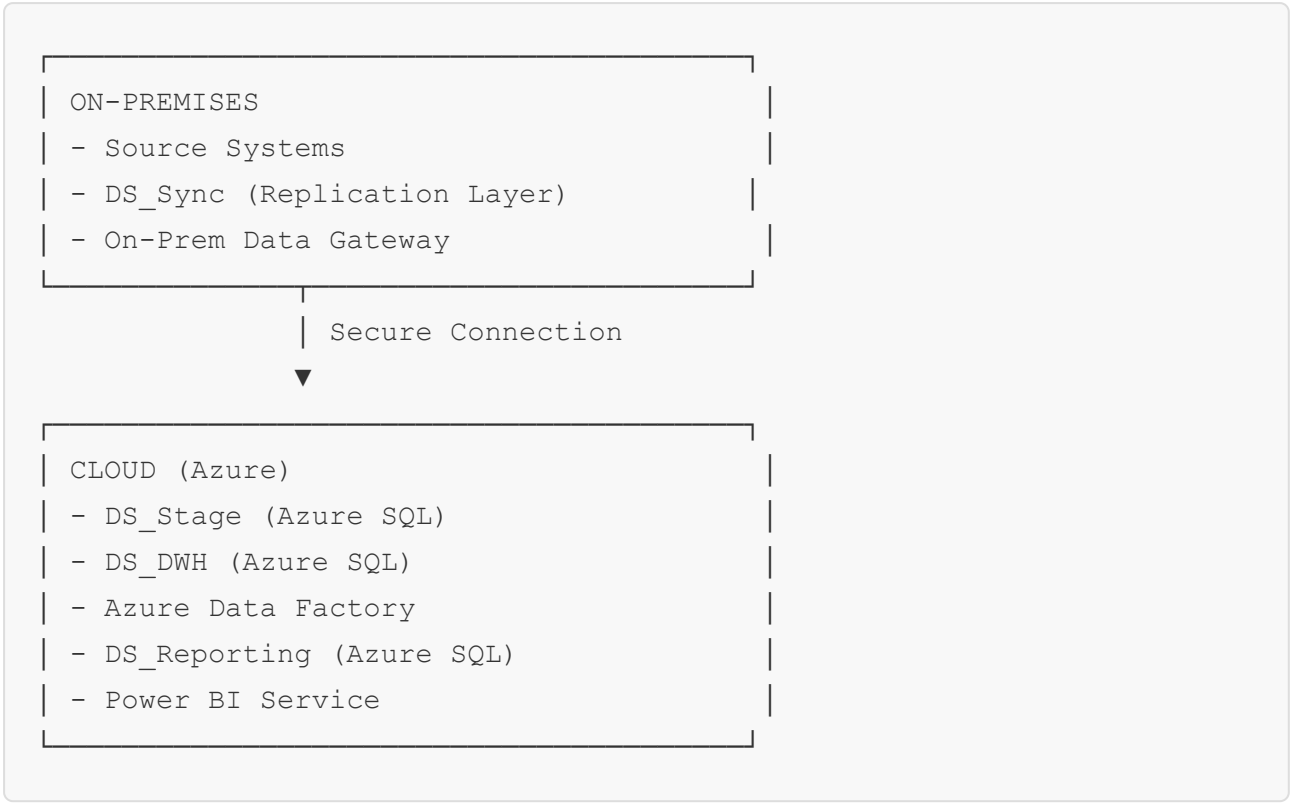
Overview

Definition: Integrating UDM with existing systems in a hybrid environment.

Best For:

- Gradual cloud adoption
- Keeping some systems on-premises
- Multi-platform architectures

Architecture Pattern



Implementation Steps

Week 1-2: Set up Azure infrastructure and connectivity
Week 3-4: Implement DS_Sync replication layer
Week 5-8: Build cloud DWH (follow Greenfield steps)
Week 9-10: Migrate BI tools to cloud
Week 11-12: Optimize and stabilize

Estimated Timelines by Complexity

Small Implementation (1-5 Sources)

- **Greenfield**: 6-8 weeks
- **Migration**: 12-14 weeks
- **Hybrid**: 10-12 weeks

Medium Implementation (6-15 Sources)

- **Greenfield**: 10-14 weeks
- **Migration**: 16-20 weeks
- **Hybrid**: 14-18 weeks

Large Implementation (16+ Sources)

- **Greenfield**: 16-24 weeks
- **Migration**: 24-32 weeks
- **Hybrid**: 20-28 weeks

Common Implementation Challenges

Challenge 1: Source System Access

Issue: Difficulty getting read access or understanding source schemas

Solution:

- Engage source system owners early
- Use data profiling tools
- Start with well-documented sources
- Build relationships with IT teams

Challenge 2: Data Quality Issues

Issue: Discovering poor quality data during integration

Solution:

- Embrace fallout system (don't block on quality)
- Document issues systematically
- Work with data owners to improve at source
- Set realistic expectations on cleanup timeline

Challenge 3: Scope Creep

Issue: Stakeholders requesting additional sources/features mid-project

Solution:

- Maintain strict scope for Phase 1
- Create backlog for future phases
- Use MVP approach (minimum viable product)
- Demonstrate value early to secure continued funding

Challenge 4: Performance Issues

Issue: Slow ETL or query performance

Solution:

- Implement indexing strategy early
- Use incremental loading religiously
- Partition large fact tables
- Optimize ADF dataflows (broadcast joins, partitioning)
- Consider Azure SQL elastic pools for burst capacity

Success Criteria and KPIs

Technical KPIs

ETL Performance:

- ETL completion within scheduled window (e.g., by 6am daily)
- < 1% data load failures
- Fallout resolution within 48 hours

Data Quality:

- < 5% fallout rate per source
- 100% source system traceability
- Complete audit trail

System Performance:

- Report load times < 10 seconds (95th percentile)
- Database size within budget
- Backup/recovery RTO < 4 hours

Business KPIs

User Adoption:

- 80%+ of target users active monthly
- Decrease in legacy system usage
- Positive user satisfaction scores

Time to Insight:

- Faster report generation
- Reduced time for ad-hoc analysis
- Self-service analytics adoption

ROI:

- Reduced ETL maintenance hours
- Faster integration of new sources
- Improved decision-making (measured by business outcomes)

Post-Implementation Best Practices

Ongoing Maintenance

Daily:

- Monitor ETL pipeline execution
- Review fallout reports
- Check data quality metrics

Weekly:

- Review new fallout trends
- Performance optimization
- User feedback review

Monthly:

- Capacity planning
- Security patching
- Documentation updates

Quarterly:

- Add new data sources
- Implement requested features
- Architecture review and optimization

Continuous Improvement

Metrics to Track:

- New source integration time (should decrease over time)
- Fallout trends (should improve as sources mature)
- User satisfaction scores
- Query performance metrics

Optimization Opportunities:

- Consolidate similar dataflows
- Refactor complex transformations
- Implement caching strategies
- Tune database performance

Conclusion

Successful UDM Framework implementation requires:

1. **Proper Planning**: Clear requirements, realistic timelines, stakeholder alignment
2. **Phased Approach**: Start small, prove value, expand incrementally
3. **Quality Focus**: Embrace fallout system, track and improve data quality
4. **User-Centric**: Involve business users early, deliver value quickly
5. **Documentation**: Maintain clear documentation throughout
6. **Flexibility**: Adapt framework to business needs while maintaining core principles

By following this guide and adapting it to your specific context, organizations can successfully implement the UDM Framework and realize its benefits: faster integration, better data quality, and scalable analytics foundation.

Next: 06_Adding_Data_Sources.md - Operational guide for adding new data sources