UDM Framework Technical Reference

Detailed Technical Specifications and Code Reference

Introduction

This document provides comprehensive technical reference for the UDM Framework, including complete table schemas, stored procedures, functions, views, and code examples. Use this as a quick reference guide during implementation and maintenance.

Core Framework Tables

Reference.SourceSystem

Purpose: Master configuration table for all data source mappings

Full Schema:

```
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 CLUSTERED
```

```
    ([SourceSystemId] ASC)
    WITH (
        PAD_INDEX = OFF,
        STATISTICS_NORECOMPUTE = OFF,
        IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON,
        FILLFACTOR = 80,
        OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF
    ) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Foreign key for hierarchical relationships
ALTER TABLE [Reference].[SourceSystem]
ADD CONSTRAINT [FK_SourceSystem_ParentSourceSystem]
FOREIGN KEY ([ParentSourceSystemId])
REFERENCES [Reference].[SourceSystem]([SourceSystemId])
GO
```

Column Descriptions:

Column	Type	Description
SourceSystemId	bigint	Auto-increment primary key
SourceDescription	nvarchar(100)	Friendly name for source (e.g., "Salesforce Accounts")
SourceDatabase	nvarchar(100)	Source database name (e.g., "DS_Stage_CRM")
SourceSchema	nvarchar(50)	Schema in source database (usually "dbo")
SourceTable	nvarchar(100)	Table name in source
SourceIdentifierColumn	nvarchar(100)	Primary key column in source table
TargetSchema	nvarchar(50)	DWH schema (Sales, Product, Contact, etc.)

Column	Type	Description
TargetTable	nvarchar(100)	DWH table name
TargetIdentifierColumn	nvarchar(100)	PK column in DWH table (usually "[Table]Id")
ValidFromDate	date	When this mapping becomes active
ValidToDate	date	When this mapping expires (9999-12-31 for active)
ParentSourceSystemId	bigint	FK to parent SourceSystem (for EAV sources)
SourceKeyColumn	nvarchar(100)	For EAV: column containing property names
SourceValueColumn	nvarchar(100)	For EAV: column containing property values

Indexes:

```
CREATE NONCLUSTERED INDEX IX_SourceSystem_TargetTable
ON [Reference].[SourceSystem] ([TargetSchema], [TargetTable])
INCLUDE ([SourceSystemId], [SourceDescription])

CREATE NONCLUSTERED INDEX IX_SourceSystem_ValidDates
ON [Reference].[SourceSystem] ([ValidFromDate], [ValidToDate])
INCLUDE ([SourceSystemId])
```

Reference.SourceColumnMapping

Purpose: Column-level mappings from source to target

Full Schema:

```
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 CLUSTERED
    ([SourceColumnMappingId] ASC)
    WITH (
        PAD_INDEX = OFF,
        STATISTICS_NORECOMPUTE = OFF,
        IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON,
        OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF
    ) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [Reference].[SourceColumnMapping]
ADD CONSTRAINT [FK_SourceColumnMapping_SourceSystem]
FOREIGN KEY ([SourceSystemId])
REFERENCES [Reference].[SourceSystem]([SourceSystemId])
GO
```

Indexes:

```
CREATE NONCLUSTERED INDEX IX_SourceColumnMapping_SourceSystemId
ON [Reference].[SourceColumnMapping] ([SourceSystemId])
INCLUDE ([SourceColumnName], [TargetAlias])
```

Reference.PropertyReference

Purpose: Controls which EAV properties are active

Full Schema:

```
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 CLUSTERED
    ([PropertyReferenceId] ASC)
    WITH (
        PAD_INDEX = OFF,
        STATISTICS_NORECOMPUTE = OFF,
        IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON,
        OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF
    ) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [Reference].[PropertyReference]
ADD CONSTRAINT [FK_PropertyReference_SourceSystem]
FOREIGN KEY ([SourceSystemId])
REFERENCES [Reference].[SourceSystem]([SourceSystemId])
GO
```

Fallout.Fallout

Purpose: Tracks data quality issues

Full Schema:

```
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_Fallout2] PRIMARY KEY CLUSTERED
    ([FalloutId] ASC)
    WITH (
        PAD_INDEX = OFF,
        STATISTICS_NORECOMPUTE = OFF,
```

```
        IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON,
        FILLFACTOR = 80,
        OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF
    ) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [Fallout].[Fallout]
ADD CONSTRAINT [FK_Fallout_ErrorCode]
FOREIGN KEY ([ErrorCodeId])
REFERENCES [Fallout].[ErrorCode]([ErrorCodeId])
GO

ALTER TABLE [Fallout].[Fallout]
ADD CONSTRAINT [FK_Fallout_SourceSystem]
FOREIGN KEY ([SourceSystemId])
REFERENCES [Reference].[SourceSystem]([SourceSystemId])
GO
```

Indexes:

```
CREATE NONCLUSTERED INDEX IX_Fallout_SourceSystemId
ON [Fallout].[Fallout] ([SourceSystemId])
INCLUDE ([ErrorCodeId], [InsertDate], [SolveDate])

CREATE NONCLUSTERED INDEX IX_Fallout_SolveDate
ON [Fallout].[Fallout] ([SolveDate])
WHERE [SolveDate] IS NULL

CREATE NONCLUSTERED INDEX IX_Fallout_InsertDate
ON [Fallout].[Fallout] ([InsertDate] DESC)
INCLUDE ([ErrorCodeId], [SourceSystemId])
```

Fallout.ErrorCode

Purpose: Error taxonomy and descriptions

Full Schema:

```
CREATE TABLE [Fallout].[ErrorCode](
    [ErrorCodeId] [int] NOT NULL,
    [ErrorDescription] [nvarchar](200) NULL,
    [ErrorCategory] [nvarchar](50) NULL,
    [ResolutionGuidance] [nvarchar](max) NULL,
    CONSTRAINT [PK_ErrorCode] PRIMARY KEY CLUSTERED
    ([ErrorCodeId] ASC)
    WITH (
        PAD_INDEX = OFF,
        STATISTICS_NORECOMPUTE = OFF,
        IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON,
        OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF
    ) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Standard Error Codes:

```
-- Referential Integrity (11-19)
INSERT INTO [Fallout].[ErrorCode] VALUES
(11, 'Missing Foreign Key: Lead/Opportunity', 'Referential Integrity', 'C
(12, 'Missing Foreign Key: Channel/Marketing', 'Referential Integrity', '
(13, 'Missing Foreign Key: Status/Lookup', 'Referential Integrity', 'Ensu
(14, 'Missing Foreign Key: Product', 'Referential Integrity', 'Verify Pro
(15, 'Missing Foreign Key: Customer/Contact', 'Referential Integrity', 'C

-- Data Format (20-29)
INSERT INTO [Fallout].[ErrorCode] VALUES
(20, 'Invalid Data Type', 'Data Format', 'Review data type conversion err
(21, 'Invalid Date Format', 'Data Format', 'Check date values for correct
(22, 'Invalid Numeric Value', 'Data Format', 'Verify numeric fields conta

-- Business Logic (30-39)
INSERT INTO [Fallout].[ErrorCode] VALUES
(30, 'Business Rule Violation', 'Business Logic', 'Review business rule v
(31, 'Negative Amount (where positive expected)', 'Business Logic', 'Chec
(32, 'Future Date (where past expected)', 'Business Logic', 'Verify date
```

```
-- Data Quality (40-49)
INSERT INTO [Fallout].[ErrorCode] VALUES
(40, 'Duplicate Record', 'Data Quality', 'Identify and merge duplicate re
(41, 'Missing Required Field', 'Data Quality', 'Ensure required fields ar
GO
```

Core Stored Procedures

BuildHistoryTracking

Purpose: Creates history table for change tracking on source tables

Full Code (DS_Stage):

```
CREATE PROCEDURE [dbo].[BuildHistoryTracking]
    @TableFull NVARCHAR(100)
AS
/*
Peter van Rij 22-01-2020:

This Stored procedure creates copies of existing tables in the schema His
to track updates and deletes in the source data itself. This will be used
in manipulation of tables in DWH containing validity periods.

Input needs to include schema

Example:
EXEC [dbo].[BuildHistoryTracking] '[dbo].[tbl_OrderFeedback]'
*/
BEGIN
    SET NOCOUNT ON

    DECLARE @TableFullInput NVARCHAR(500) = REPLACE(REPLACE(@TableFull, '
    DECLARE @Table_Name NVARCHAR(100) = RIGHT(@TableFullInput, CHARINDEX

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

```
END
GO
```

Usage:

```
-- Enable history tracking for a source table
EXEC [dbo].[BuildHistoryTracking] '[dbo].[Products]'

-- Verify history table created
SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'History' AND TABLE_NAME = 'Products'
```

CreateHistoryTable

Purpose: Generates DDL for history table based on source table structure

Full Code (DS_Stage):

```
CREATE PROCEDURE [dbo].[CreateHistoryTable]
    @table_name SYSNAME
AS
/*
Creates a history table in the History schema that mirrors the source tab
structure, with additional ChangeType and ChangeTime columns.

Example: @table_name = 'dbo.Products'
*/
BEGIN
    SET NOCOUNT ON

    DECLARE
        @object_name SYSNAME,
        @object_id INT

    SELECT
        @object_name = '[History].[' + o.name +']',
        @object_id = o.[object_id]
    FROM sys.objects o WITH (NOWAIT)
    JOIN sys.schemas s WITH (NOWAIT) ON o.[schema_id] = s.[schema_id]
```

```
    WHERE s.name + '.' + o.name = @table_name
        AND o.[type] = 'U'
        AND o.is_ms_shipped = 0

    DECLARE @SQL NVARCHAR(MAX) = ''

    -- Build CREATE TABLE statement
    SELECT @SQL = 'CREATE TABLE ' + @object_name + CHAR(13) +
        '(  ChangeType nvarchar(100) NULL,ChangeTime DateTime NULL, ' + C
        STUFF((
            SELECT CHAR(9) + ', [' + c.name + '] ' +
                CASE WHEN c.is_computed = 1
                    THEN 'AS ' + cc.[definition]
                    ELSE UPPER(tp.name) +
                        CASE WHEN tp.name IN ('varchar', 'char', 'varbina
                               THEN '(' + CASE WHEN c.max_length = -1 THE
                             WHEN tp.name IN ('nvarchar', 'nchar')
                               THEN '(' + CASE WHEN c.max_length = -1 THE
                             WHEN tp.name IN ('datetime2', 'time2', 'date
                               THEN '(' + CAST(c.scale AS VARCHAR(5)) + '
                             WHEN tp.name IN ('decimal', 'numeric')
                               THEN '(' + CAST(c.[precision] AS VARCHAR(5
                            ELSE ''
                        END +
                        CASE WHEN c.collation_name IS NOT NULL THEN ' COL
                        CASE WHEN c.is_nullable = 1 THEN ' NULL' ELSE ' N
                        CASE WHEN dc.[definition] IS NOT NULL THEN ' DEFA
                END + CHAR(13)
            FROM sys.columns c WITH (NOWAIT)
            JOIN sys.types tp WITH (NOWAIT) ON c.user_type_id = tp.user_t
            LEFT JOIN sys.computed_columns cc WITH (NOWAIT) ON c.[object_
            LEFT JOIN sys.default_constraints dc WITH (NOWAIT) ON c.defau
            WHERE c.[object_id] = @object_id
            ORDER BY c.column_id
            FOR XML PATH(''), TYPE
        ).value('.', 'NVARCHAR(MAX)'), 1, 2, CHAR(9) + ' ')+')'

    -- Execute the CREATE TABLE statement
    EXEC(@SQL)
END
GO
```

Usage:

```
-- Manually create history table (usually called by BuildHistoryTracking)
EXEC [dbo].[CreateHistoryTable] 'dbo.Customers'
```

ReadSourceTable_NewRecords

Purpose: Extracts new/changed records from Stage to Flow tables

Pseudo-Code (simplified for clarity):

```
CREATE PROCEDURE [dbo].[ReadSourceTable_NewRecords]
    @SourceSystemId BIGINT
AS
BEGIN
    SET NOCOUNT ON

    -- Get source configuration
    DECLARE @SourceDatabase NVARCHAR(100)
    DECLARE @SourceSchema NVARCHAR(50)
    DECLARE @SourceTable NVARCHAR(100)
    DECLARE @SourceIdCol NVARCHAR(100)
    DECLARE @TargetSchema NVARCHAR(50)
    DECLARE @TargetTable NVARCHAR(100)

    SELECT
        @SourceDatabase = SourceDatabase,
        @SourceSchema = SourceSchema,
        @SourceTable = SourceTable,
        @SourceIdCol = SourceIdentifierColumn,
        @TargetSchema = TargetSchema,
        @TargetTable = TargetTable
    FROM Reference.SourceSystem
    WHERE SourceSystemId = @SourceSystemId

    -- Build column mapping list
    DECLARE @ColumnMappings NVARCHAR(MAX)
    SELECT @ColumnMappings = STRING_AGG(
        CONCAT('[', SourceColumnName, '] AS [', TargetAlias, ']'), ', '
    )
```

```
    FROM Reference.SourceColumnMapping
    WHERE SourceSystemId = @SourceSystemId
        AND SourceColumnName IS NOT NULL

    -- Build Flow table name
    DECLARE @FlowTable NVARCHAR(200) = CONCAT(
        'Flow.', @TargetSchema, '_', @TargetTable, '_', @SourceSystemId
    )

    -- Create Flow table if doesn't exist (simplified)
    IF OBJECT_ID(@FlowTable) IS NULL
    BEGIN
        DECLARE @CreateSQL NVARCHAR(MAX) =
            'CREATE TABLE ' + @FlowTable + ' (' +
            -- Dynamic columns based on SourceColumnMapping
            ')'
        EXEC(@CreateSQL)
    END

    -- Clear existing Flow data
    EXEC('TRUNCATE TABLE ' + @FlowTable)

    -- Insert new records only (incremental)
    DECLARE @InsertSQL NVARCHAR(MAX) = '
    INSERT INTO ' + @FlowTable + '
    SELECT ' + @ColumnMappings + ',
           ' + CAST(@SourceSystemId AS VARCHAR) + ' AS SourceSystemId
    FROM ' + @SourceDatabase + '.' + @SourceSchema + '.' + @SourceTable +
    WHERE NOT EXISTS (
        SELECT 1
        FROM DS_DWH.' + @TargetSchema + '.' + @TargetTable + ' t
        WHERE t.SourceSystemId = ' + CAST(@SourceSystemId AS VARCHAR) + '
          AND t.SourceSystemIdentifier = CAST(s.' + @SourceIdCol + ' AS N
    )'

    EXEC(@InsertSQL)

    -- Return row count
    DECLARE @RowCount INT
    EXEC('SELECT @RowCount = COUNT(*) FROM ' + @FlowTable)

    PRINT 'Extracted ' + CAST(@RowCount AS VARCHAR) + ' new records to '
```

```
END
GO
```

Usage:

```
-- Extract new records for SourceSystemId 42
EXEC [dbo].[ReadSourceTable_NewRecords] @SourceSystemId = 42

-- Verify Flow table populated
SELECT COUNT(*) FROM Flow.Product_Product_42
```

Key Views

Reference.VW_ColumnMappingAssistant

Purpose: Shows available source columns for unmapped target columns

Full Code (DS_Stage):

```
CREATE VIEW [Reference].[VW_ColumnMappingAssistant] AS
/*
Shows available columns in source table for unmapped target columns.
Ignores columns that are already used.
SourceSystem table needs to be filled for this to work.
*/
WITH UnmappedSourceSystems AS (
    SELECT DISTINCT SourceSystemId
    FROM [DS_DWH].[Reference].[VW_UnmappedTargetColumns]
)
SELECT
    SS.SourceSystemId,
    SS.SourceDescription,
    c.name AS SourceColumn,
    t.name AS SourceTable,
    s.name AS SourceSchema
FROM sys.columns c
JOIN sys.tables t ON c.object_id = t.object_id
JOIN sys.schemas s ON s.schema_id = t.schema_id
```

```
JOIN [DS_DWH].[Reference].SourceSystem SS
    ON SS.SourceTable = t.name
    AND SS.SourceSchema = s.name
    AND SS.SourceDatabase = DB_NAME()
JOIN UnmappedSourceSystems USS ON USS.SourceSystemId = SS.SourceSystemId
LEFT JOIN [DS_DWH].[Reference].SourceColumnMapping SCM
    ON SCM.SourceSystemId = SS.SourceSystemId
    AND SCM.SourceColumnName = c.name
WHERE SCM.SourceColumnName IS NULL
GO
```

Usage:

```
-- Find unmapped columns for SourceSystemId 42
SELECT * FROM [Reference].[VW_ColumnMappingAssistant]
WHERE SourceSystemId = 42
ORDER BY SourceColumn
```

Fallout.VW_TotalFallout

Purpose: Summary of open fallout issues

Full Code (DS_DWH):

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

```
GROUP BY
    ss.SourceSystemId,
    ss.SourceDescription,
    ec.ErrorCodeId,
    ec.ErrorDescription,
    ec.ErrorCategory
GO
```

Usage:

```
-- View all open fallout
SELECT * FROM [Fallout].[VW_TotalFallout]
ORDER BY FalloutCount DESC

-- Fallout for specific source
SELECT * FROM [Fallout].[VW_TotalFallout]
WHERE SourceSystemId = 42
```

User-Defined Functions

FN_ReplaceSpecialChars

Purpose: Removes special characters from strings

Full Code:

```
CREATE FUNCTION [dbo].[FN_ReplaceSpecialChars]
    @string VARCHAR(100)
RETURNS VARCHAR(100)
AS
BEGIN
    DECLARE @i INT = 1
    DECLARE @OriginalString VARCHAR(100) = @string COLLATE SQL_Latin1_Ger
    DECLARE @ModifiedString VARCHAR(100) = ''

    WHILE @i <= LEN(@OriginalString)
    BEGIN
        IF SUBSTRING(@OriginalString, @i, 1) LIKE '[a-Z]'
```

```
        BEGIN
            SET @ModifiedString = @ModifiedString + SUBSTRING(@OriginalSt
        END
        SET @i = @i + 1
    END

    RETURN @ModifiedString
END
GO
```

Mobile.fn_HashNVar_NullAware (Creative Clicks)

Purpose: Null-safe hashing for complex group keys

Full Code:

```
CREATE FUNCTION [Mobile].[fn_HashNVar_NullAware]
    @Input NVARCHAR(MAX)
RETURNS NVARCHAR(MAX)
WITH SCHEMABINDING
AS
BEGIN
    RETURN ISNULL(@Input, '<NULL>')
END
GO
```

Usage in fn_ARPU_GroupKey:

```
CREATE FUNCTION [Mobile].[fn_ARPU_GroupKey] (
    @CarrierId NVARCHAR(150),
    @NetworkId NVARCHAR(150),
    -- ... multiple parameters
)
RETURNS VARBINARY(32)
WITH SCHEMABINDING
AS
BEGIN
    RETURN HASHBYTES('SHA2_256',
        Mobile.fn_HashNVar_NullAware(@CarrierId) +
```

```
        Mobile.fn_HashNVar_NullAware(@NetworkId) +
        -- ... concatenate all dimensions
    )
END
GO
```

Standard Table Patterns

Dimension Table Pattern

Template:

```
CREATE TABLE [Schema].[DimensionName](
    [DimensionNameId] [bigint] IDENTITY(1,1) NOT NULL,
    [DimensionNameNK] [nvarchar](100) NOT NULL,          -- Natural Key
    -- Business columns
    [Column1] [datatype] NULL,
    [Column2] [datatype] NULL,
    -- Timeslice columns
    [ValidFromDate] [date] NOT NULL,
    [ValidToDate] [date] NOT NULL,
    -- Source traceability
    [SourceSystemId] [bigint] NOT NULL,
    [SourceSystemIdentifier] [nvarchar](100) NULL,
    CONSTRAINT [PK_DimensionName] PRIMARY KEY CLUSTERED ([DimensionNameId
)
GO

-- Standard indexes
CREATE NONCLUSTERED INDEX IX_DimensionName_NK
    ON [Schema].[DimensionName]([DimensionNameNK])

CREATE NONCLUSTERED INDEX IX_DimensionName_ValidToDate
    ON [Schema].[DimensionName]([ValidToDate])
    WHERE [ValidToDate] = '9999-12-31'

CREATE NONCLUSTERED INDEX IX_DimensionName_SourceSystemId
    ON [Schema].[DimensionName]([SourceSystemId])
```

Fact Table Pattern

Template:

```
CREATE TABLE [Schema].[FactName](
    [FactNameId] [bigint] IDENTITY(1,1) NOT NULL,
    [FactNameNK] [nvarchar](100) NOT NULL,
    [FactDate] [datetime] NOT NULL,
    -- Dimension foreign keys
    [Dimension1Id] [bigint] NULL,
    [Dimension2Id] [bigint] NULL,
    -- Measures
    [Measure1] [decimal](18,2) NULL,
    [Measure2] [int] NULL,
    -- Source traceability
    [SourceSystemId] [bigint] NOT NULL,
    [SourceSystemIdentifier] [nvarchar](100) NULL,
    CONSTRAINT [PK_FactName] PRIMARY KEY CLUSTERED ([FactNameId])
)
GO

-- Standard indexes
CREATE NONCLUSTERED INDEX IX_FactName_FactDate
    ON [Schema].[FactName]([FactDate])

CREATE NONCLUSTERED INDEX IX_FactName_Dimension1Id
    ON [Schema].[FactName]([Dimension1Id])

-- Partitioning (for large tables)
-- Create partition function and scheme, then:
-- CREATE CLUSTERED INDEX IX_FactName_FactDate
--     ON [Schema].[FactName]([FactDate])
--     ON PS_FactName([FactDate])
```

Property Table Pattern (EAV)

Template:

```
CREATE TABLE [Schema].[EntityProperties](
    [EntityPropertyId] [bigint] IDENTITY(1,1) NOT NULL,
    [EntityId] [bigint] NOT NULL,
    [PropertyName] [nvarchar](100) NOT NULL,
    [PropertyValue] [nvarchar](500) NULL,
    [ValidFromDate] [date] NULL,
    [ValidToDate] [date] NULL,
    CONSTRAINT [PK_EntityProperties] PRIMARY KEY CLUSTERED ([EntityPrope
    CONSTRAINT [FK_EntityProperties_Entity]
        FOREIGN KEY ([EntityId])
        REFERENCES [Schema].[Entity]([EntityId])
)
GO

CREATE NONCLUSTERED INDEX IX_EntityProperties_EntityId
    ON [Schema].[EntityProperties]([EntityId])

CREATE NONCLUSTERED INDEX IX_EntityProperties_PropertyName
    ON [Schema].[EntityProperties]([PropertyName])
```

Naming Conventions

Tables

- **Schemas**: PascalCase (Sales, Product, Contact)
- **Tables**: PascalCase singular (Order, Customer, Product)
- **Primary Keys**: [TableName]Id (OrderId, CustomerId)
- **Natural Keys**: [TableName]NK (OrderNK, CustomerNK)

Columns

- **Columns**: PascalCase (OrderDate, CustomerName)
- **Foreign Keys**: [ReferencedTable]Id (CustomerId, ProductId)
- **Boolean flags**: Is[Condition] (IsActive, IsDeleted)
- **Timestamps**: [Action]Date or [Action]Timestamp (CreatedDate, ModifiedTimestamp)

Constraints

- **Primary Key**: PK_[TableName]
- **Foreign Key**: FK_[TableName]_[ReferencedTable]
- **Unique**: UQ_[TableName]_[ColumnName]
- **Check**: CK_[TableName]_[ColumnName]
- **Default**: DF_[TableName]_[ColumnName]

Indexes

- **Non-clustered**: IX_[TableName]_[ColumnName(s)]
- **Unique**: UX_[TableName]_[ColumnName(s)]

Schemas

- **Business Domains**: Sales, Finance, Product, Contact, Marketing
- **Framework**: Reference, Fallout, Flow, Bridge, Logging
- **Stage**: History, Reference
- **Reporting**: Dim, Fact, Model

Configuration Settings

Database Settings

DS_DWH (recommended):

```
ALTER DATABASE [DS_DWH] SET RECOVERY SIMPLE
ALTER DATABASE [DS_DWH] SET AUTO_UPDATE_STATISTICS ON
ALTER DATABASE [DS_DWH] SET PAGE_VERIFY CHECKSUM
ALTER DATABASE [DS_DWH] SET COMPATIBILITY_LEVEL = 150
```

DS_Stage (recommended):

```
ALTER DATABASE [DS_Stage] SET RECOVERY FULL  -- For point-in-time recover
ALTER DATABASE [DS_Stage] SET AUTO_UPDATE_STATISTICS ON
ALTER DATABASE [DS_Stage] SET PAGE_VERIFY CHECKSUM
```

Conclusion

This technical reference provides the foundational code and patterns for implementing the UDM Framework. Use it as a quick reference during development, and adapt the patterns to your specific business requirements while maintaining the core principles of the framework.

Next: 11_FAQ_Troubleshooting.md - Common questions and solutions