Adventure Works Data Warehouse ETL Requirements

Summary

This document describes the ETL requirements for the Adventure Works Data Warehouse project to support sales analysis. The ETL processes for this iteration of the project must perform one historical extraction and daily incremental extractions of product, customer, and sales data for loading into a dimensional model.

Change Log

This section is reserved for tracking changes to requirements.

Change Date	Changed By	Description

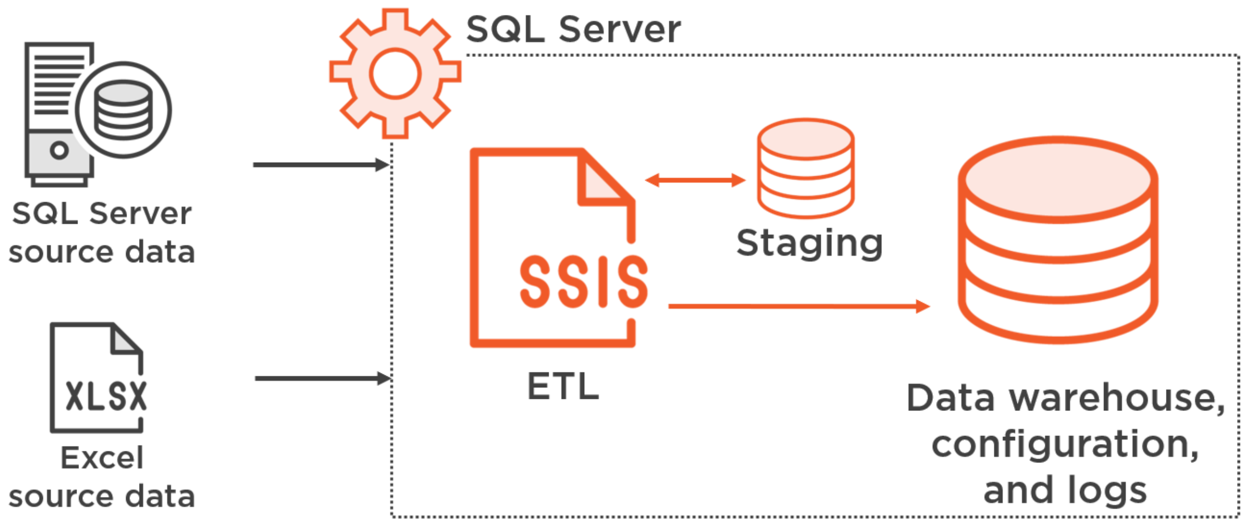
Standards

This section describes the standards to be implemented in the solution design and development and includes the following subsections:

- ETL architecture
- Auditing and logging mechanism
- System availability requirements
- Process standards
- Extraction framework
- Slowly changing dimension handling
- Business rule validation strategy
- Notification requirements

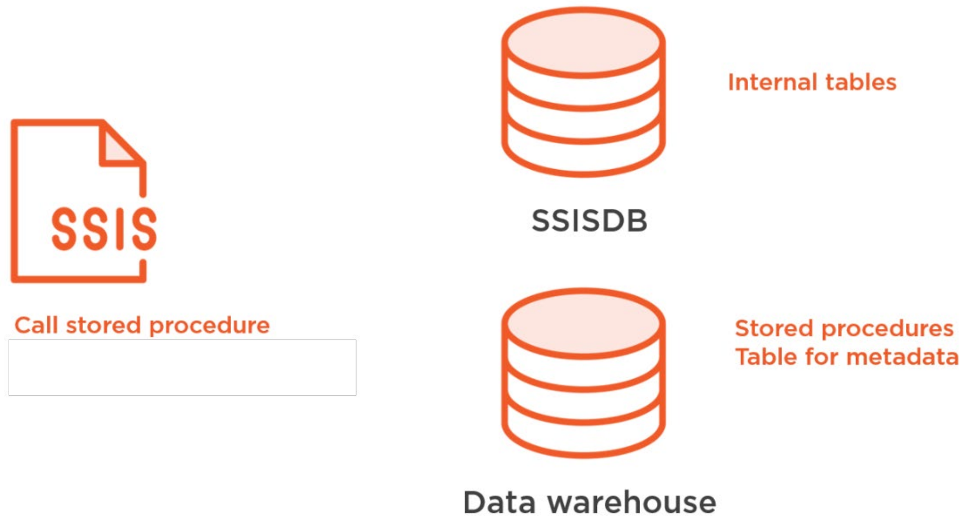
ETL architecture

There is one source of sales operations and forecasting data coming from a SQL Server database and another source of data coming from Excel spreadsheets. A second SQL Server will host SSIS to perform the ETL work. It will perform extract jobs to get the source data which it will then store In a staging database located on the same server. SSIS will also execute transformation jobs on the staged data and then load the data into a data warehouse located on the same server.



Auditing and logging mechanism

The auditing and logging mechanism (ALM) for This project will use a combination of components as shown in the diagram below. The SSISDB automatically captures information about package execution and requires no additional configuration. Additional metadata about package execution will be captured by executing stored procedures at the beginning and end of the control flow in each package. The first stored procedure will insert a new row into the metadata table in the data warehouse while the second stored procedure will update that row to reflect successful execution and to log data flow metrics. Variables must also be populated by strategically placed row count transformations in the data flow to track the number of inserts, updates, and errors.



Storage.

The log entries generated by the ALM are stored in the DimAudit in the data warehouse database. Package operations are logged automatically by SSIS and stored in tables accessible in internal tables and views. See Implementation Details below for more information.

Maintenance.

The DimAudit table will be backed up as part of the data warehouse database routine backup. On an annual basis, IT will review ALM usage to determine whether to remove historical rows prior to a specified period of time.

Implementation details.

The solution project must contain a connection manager to store the requisite connection information for the database. Although this iteration of the project will host the auditing and logging tables in the data warehouse, the location may change in the future. Therefore, a separate connection manager for the ALM is required.

SQL Server

Only one table is required to implement the ALM. This table will contain log entries related to package batches, package execution, and metrics about inserts, updates, and errors that occur during package execution.

The SSIS catalog database (SSISDB) contains tables and views that provide metadata about packages, versions, package tasks, and package execution such as parameter values, property overrides, and errors.

Tables

The DimAudit table exists in the data warehouse schema to associate a row inserted into a table with a specific package (PackageName, PackageGUID, and PackageVersionGUID). This table will track the hierarchy of package executions from the outermost parent package to leaf-level child package. When a parent package calls a child package, the parent's audit key will be passed to the child package and included in the new row added to the DimAudit table.

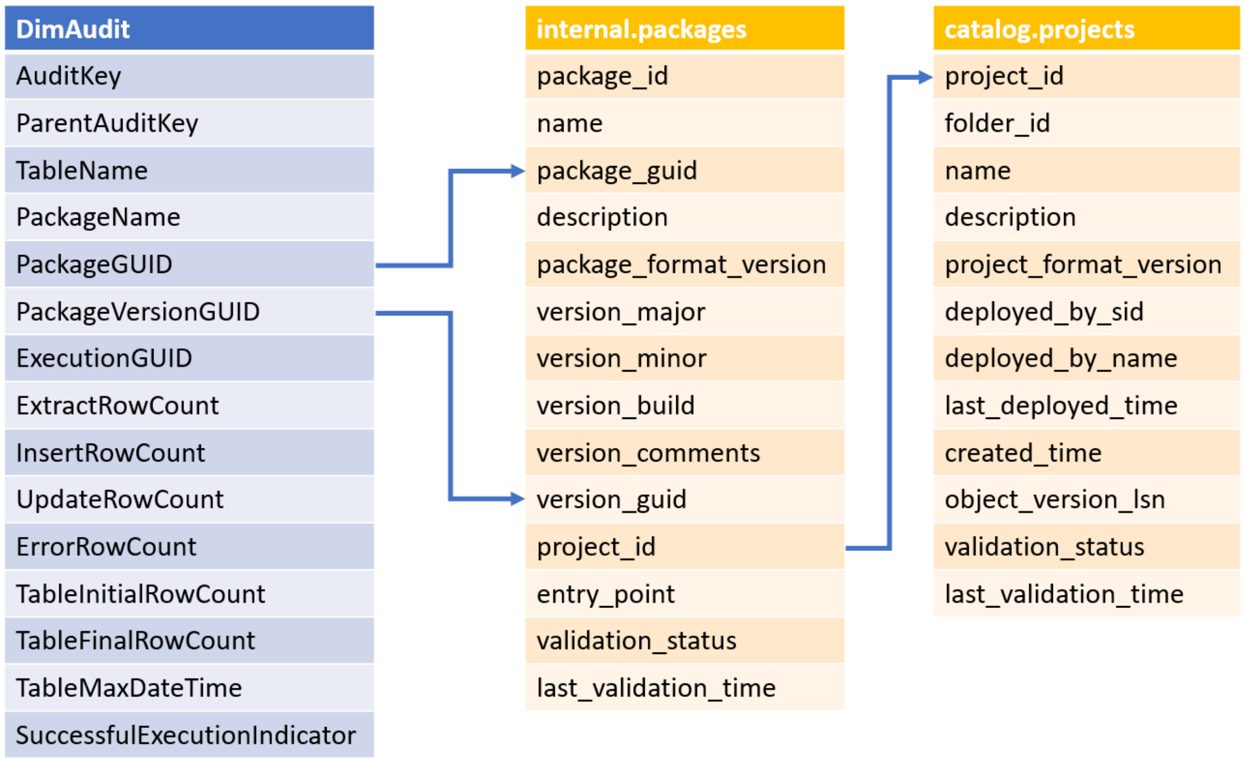
```sql
CREATE TABLE [DimAudit](
	[AuditKey] [int] IDENTITY(1,1) NOT NULL,
	[ParentAuditKey] [int] NOT NULL,
	[TableName] [varchar](50) NOT NULL,
	[PackageName] [varchar](50) NOT NULL,
	[PackageGUID] [uniqueidentifier] NULL,
	[PackageVersionGUID] [uniqueidentifier] NULL,
	[ExecutionGUID] [uniqueidentifier] NULL,
	[ExtractRowCount] [bigint] NULL,
	[InsertRowCount] [bigint] NULL,
	[UpdateRowCount] [bigint] NULL,
	[ErrorRowCount] [bigint] NULL,
	[TableInitialRowCount] [bigint] NULL,
	[TableFinalRowCount] [bigint] NULL,
	[TableMaxDateTime] [datetime] NULL,
	[SuccessfulExecutionIndicator] [char](1) NOT NULL,
 CONSTRAINT [PK_DimAudit] PRIMARY KEY CLUSTERED
(
       [AuditKey] ASC
)
)
GO

ALTER TABLE [DimAudit] ADD  DEFAULT ('Unknown') FOR [TableName]
GO

ALTER TABLE [DimAudit] ADD  DEFAULT ('Unknown') FOR [PackageName]
GO

ALTER TABLE [DimAudit] ADD  DEFAULT ('N') FOR [SuccessfulExecutionIndicator]
GO

ALTER TABLE [DimAudit]  WITH CHECK ADD  CONSTRAINT [FK_DimAudit_ParentAuditKey] FOREIGN
KEY([ParentAuditKey])
REFERENCES [DimAudit] ([AuditKey])
GO

ALTER TABLE [DimAudit] CHECK CONSTRAINT [FK_DimAudit_ParentAuditKey]
GO
```

Because the outermost parent package will not have a parent audit key, a row must be added to the table manually to provide a value for the ParentAuditKey column and avoid failing the foreign key constraint for that column.

```
SET IDENTITY_INSERT DimAudit ON
GO
INSERT INTO [dimAudit]
           ([AuditKey]
           ,[ParentAuditKey]
           ,[TableName]
           ,[PackageName]
           ,[PackageGUID]
           ,[PackageVersionGUID]
           ,[ExecutionGUID]
           ,[ExtractRowCount]
           ,[InsertRowCount]
           ,[UpdateRowCount]
           ,[ErrorRowCount]
           ,[TableInitialRowCount]
           ,[TableFinalRowCount]
           ,[TableMaxDateTime]
           ,[SuccessfulExecutionIndicator])
     VALUES
           (-1,
           -1
           ,'No Table'
           ,'No Package'
           ,null
           ,null
           ,null
           ,null
           ,null
           ,null
           ,null
```

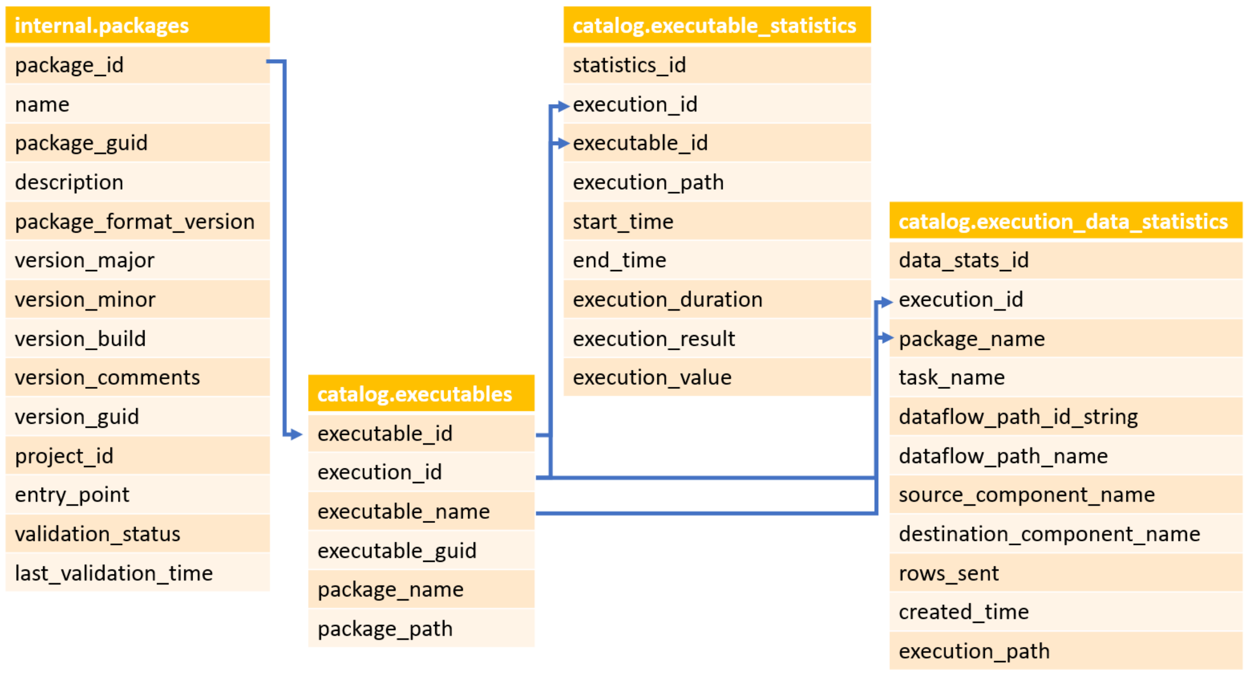
```
          ,null
          ,null
          ,null

          ,'Y')
GO

SET IDENTITY_INSERT DimAudit OFF
GO
```

The DimAudit table relates to two objects in the SSISDB database, internal.packages (to see all current and replaced packages) and catalog.projects, which contain additional metadata about the package and the project to which it belongs.



The internal.packages table in turn relates to several other views which collectively provide access to metadata about package execution:

- Catalog.executables
- Catalag.executable_statistics
- Catalog.execution_data_statistics



Stored Procedures.

Two stored procedures must exist in the data warehouse database to support the ALM:

- spAuditBegin – Inserts a row into DimAudit when a package begins and returns an audit key to be associated with each new row loaded into a table by the package.
- spAuditEnd – Updates the row inserted by spAuditBegin with metadata generated during package execution and updates the status of the package to reflect successful completion.

spAuditBegin

```
CREATE PROCEDURE [spAuditBegin]
  @TableName varchar(100),
  @PackageName varchar(100),
  @PackageID uniqueidentifier,
  @VersionGUID uniqueidentifier,
  @ExecutionGUID uniqueidentifier,
  @ParentAuditKey int

AS
BEGIN
INSERT INTO [dimAudit]
           (
            [ParentAuditKey]
           ,[TableName]
           ,[PackageName]
           ,[PackageGUID]
           ,[PackageVersionGUID]
```

```
                ,[ExecutionGUID]
                ,[ExtractRowCount]
                ,[InsertRowCount]
                ,[UpdateRowCount]
                ,[ErrorRowCount]
                ,[TableInitialRowCount]
                ,[TableFinalRowCount]
                ,[TableMaxDateTime]
                ,[SuccessfulExecutionIndicator])
          VALUES
                (@ParentAuditKey
                ,@TableName
                ,@PackageName
                ,@PackageID
                ,@VersionGUID
                ,@ExecutionGUID
                ,0
                ,0
                ,0
                ,0
                ,NULL
                ,NULL
                ,NULL
                ,'N')

     SELECT MAX(AuditKey) AS AuditKey FROM [DimAudit] WHERE TableName = @TableName AND
     ExecutionGUID = @ExecutionGUID
     END
```

spAuditEnd

```
     CREATE PROCEDURE [spAuditEnd]
       @TableInitialRowCount bigint,
       @ExtractRowCount bigint,
       @TableInsertRowCount bigint,
       @TableUpdateRowCount bigint,
       @TableFinalRowCount bigint,
       @ErrorRowCount bigint,
       @AuditKey int

     AS
     BEGIN
     UPDATE [dimAudit]
     SET
              TableInitialRowCount = @TableInitialRowCount
            , ExtractRowCount = @ExtractRowCount
            , InsertRowCount = @TableInsertRowCount
            , UpdateRowCount = @TableUpdateRowCount
            , TableFinalRowCount = @TableFinalRowCount
            , ErrorRowCount = @ErrorRowCount
            , SuccessfulExecutionIndicator='Y'
     WHERE AuditKey = @AuditKey
     END
```

SSIS

Connection Manager definition.

The project must contain a connection manager that connects to the database hosting the ALM's DimAudit table. This is a separate connection than the connection used for data warehouse operations to support the potential movement of the DimAudit table to a different location.

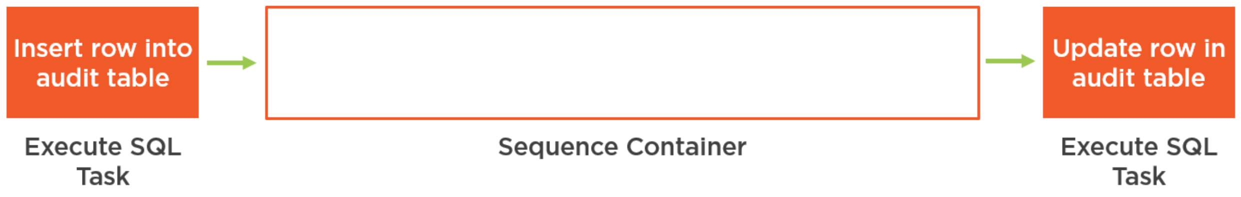
Package variables.

The ALM requires variables to store values to be used as input parameters for the spAuditBegin and spAuditEnd stored procedures:

- Table name: the name of the table used as an extraction source or a data load
- Row count errors: the number of rows that could not be inserted into a destination due to an error in the data flow
- Row count extracts: the number of rows entering the data flow pipeline from a source
- Row count inserts: the number of rows attempted to be inserted into a destination
- Row count updates: the number of rows attempted to be loaded into the SCD staging table for bulk update
- Row count initial: the number of rows in a dimension or fact table prior to a data load
- Row count final: the number of rows in a dimension or fact table after the data load

Package tasks.

To support ALM, a package must contain Execute SQL Tasks at the beginning and end of the package control flow.



The first Execute SQL Task should be configured with the following properties:

- Use the ALM connection manager
- Call the spAuditBegin stored procedure
- Input parameters: TableName user variable; system variables for package name, package ID, package version GUID, execution instance GUID; and the package parameter containing the parent audit key
- Output: A single row/column containing the audit key of the newly inserted row

The last Execute SQL Task should be configured with the following properties:

- Use the ALM connection manager
- Call the spAuditEnd stored procedure
- Input parameters: Row count variables for extracts, updates, inserts, errors, table initial count, table final count, and the audit key associated with the job

System availability requirements

This section describes requirements related to the systems and services hosting the data warehouse and the systems and services on which the data warehouse depends. It also states requirements related to user access during data warehouse operations and to restarts should ETL operations fail to complete successfully within the processing window.

High availability requirements.

None. The data warehouse is not a mission critical system.

ETL processing window.

Processing will be scheduled from 10 pm to 5 am local time at the Adventure Works headquarters, seven days per week. Query access to the data warehouse will be closed during this time.

System dependencies.

Source servers and services will be monitored by system administrators. If these servers or services experience downtime, the system administrators will notify the data warehouse administrators if the ETL processing window will be interrupted. If processing is delayed and must occur after 5 am, users will be notified that the data warehouse is unavailable and an estimated time of availability will be provided.

Restart requirements.

SSIS job restarts must be supported. The solution design must enable an operator to restart the entire ETL process from the beginning or from a specific job mid-stream to continue.

Process standards

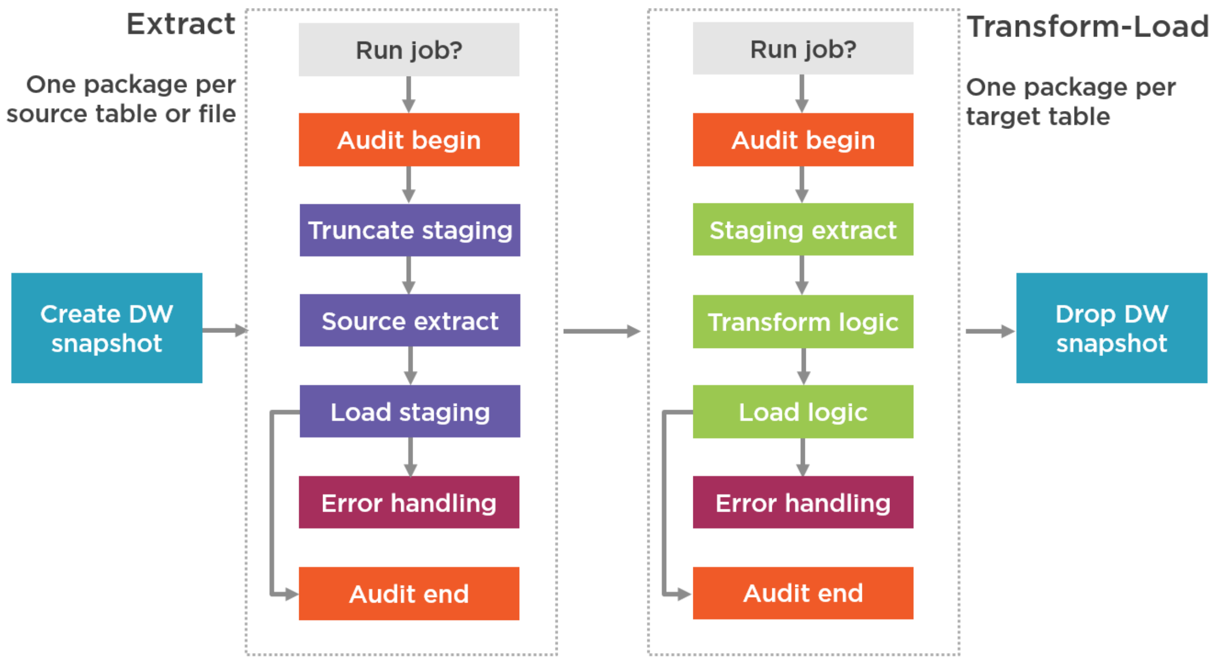
This section describes the general architecture of the solution and includes standard operations that apply before and after specific conditions.

General solution architecture.

The architecture of the solution is shown in the diagram below and conforms to the sequencing of jobs as follows:

- Create a snapshot of the data warehouse as a fall-back plan for processing errors in the transform-load jobs
- Execute extract jobs - one SSIS package per source table or file. Each package should include the following steps:
    - Check if the job should run
    - Run the spAuditBegin stored procedure.
    - Truncate the applicable staging table
    - Extract source data
    - Load extracted data into staging table
    - Redirect error rows to a flat file for manual processing
    - Run the spAuditEnd stored procedure

- Execute transform-load jobs - one SSIS package per target table. Each package should include the followings steps:
    - Check if the job should run
    - Run the spAuditBegin stored procedure
    - Extract data from the applicable staging table
    - Execute transformations
    - Load transformed data into target table
    - Redirect error rows to a flat file for manual processing
    - Run the spAuditEnd stored procedure
- Drop the snapshot at the end of processing.
- If a transform-load job fails, revert the database to the snapshot, and start over.

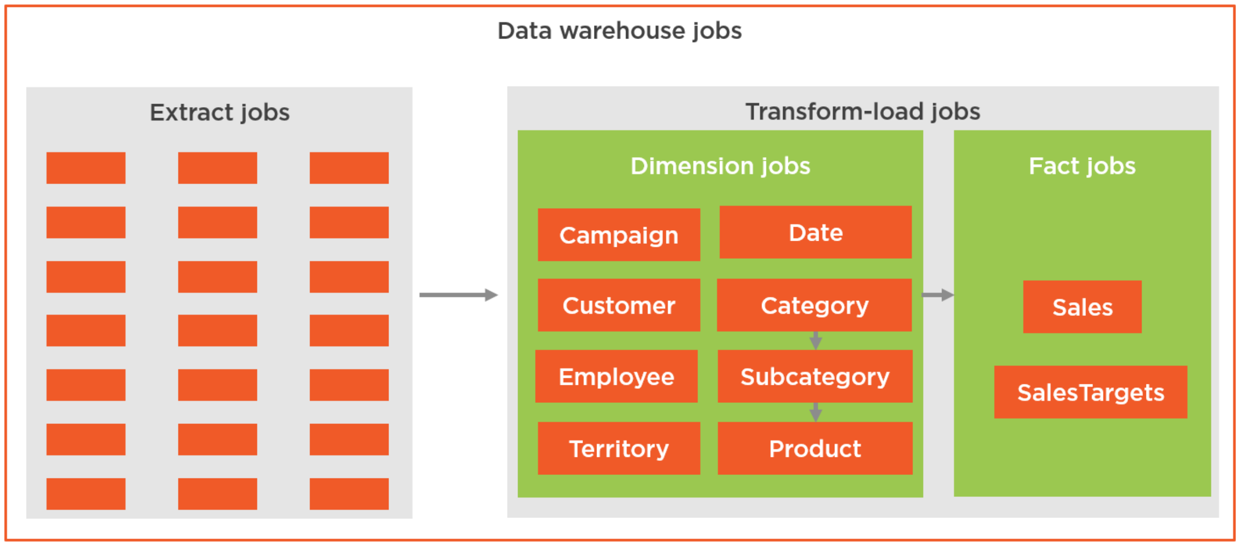


Workflow orchestration.

The solution is organized into parent jobs and child jobs as shown in the diagram below. These jobs are organized hierarchically as follows:

- Data warehouse jobs is the outermost parent job which calls Extract jobs and Transform-load jobs. All extract jobs must complete prior to beginning the transform-load jobs.
    - Extract jobs is a parent job that calls individual jobs to extract source data to staging. These jobs can run in parallel.
    - Transform-load jobs is a parent job that calls Dimension jobs and Fact jobs. All dimension jobs must complete prior to beginning the fact jobs.

- Dimension jobs is a parent job that calls individual jobs to perform transform and load operations on a dimension table. Most jobs run in parallel. An exception is the jobs related to DimProduct which is a snowflake dimensions. These jobs must run in sequence with Category first, Subcategory second, and Product third.
- Fact jobs is a parent job that calls individual jobs to perform transform and load operations on a fact table. All jobs run in parallel.



Error handling

Before loading data into a target table, the following error handling procedures should be implemented in a processing job:

- Column errors resulting from nulls, data type mismatches, or value range violations should be anticipated and the appropriate transformations should be applied to correct these errors.
- Structure errors resulting from referential integrity violations should result in the redirection of the row to an error file.

For this solution, any rows that cannot be fixed for loading into a target table should be redirected to a flat file for manual processing. An IT application specialist will be assigned the responsibility to monitor processing operations each morning and take the appropriate steps to fix and reprocess the redirected rows.

Extraction framework

To minimize the impact on data sources, data will be extracted from source tables by applying the following rules:

- A query containing no joins will select all columns from a single table.
- The selected data will be loaded into a staging table with no transformations applied.
- The data in the staging table will be available until the next operation to load the staging table is executing, at which time the staging table is truncated. This procedure allows the transform-load jobs to retrieve data from the staging table as many times as needed without impacting source systems.
- The query to extract source data for any dimension tables will not be filtered, due to the small size of these tables.
- The query to extract source data for the sales fact table will be filtered for incremental extracts so that only data that was added to the table since the last extraction will be loaded into staging. The DimAudit table will store the date/time of the last extraction for the fact table.
- The query to extract source data for the sales targets fact table will not be filtered, due to the small size of the source file.
- Because the source system does not support deletions, no consideration for identifying deleted rows is needed in the solution.

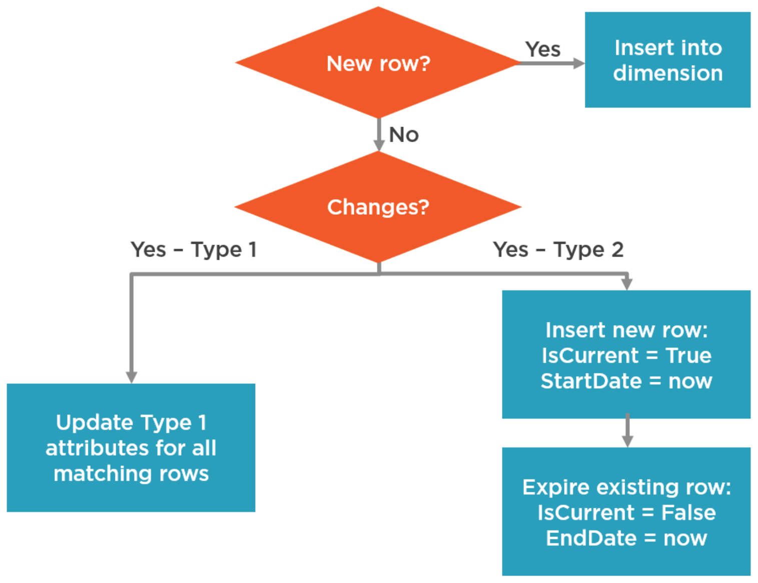
Slowly changing dimension handling

The source-to-target mapping document identifies Type 1 and Type 2 columns for slowly changing dimension (SCD) handling.

Dimension tables that support only Type 1 have no changes to their design for SCD. However, tables supporting Type 2 contain the following columns that must be populated during transform-load operations:

- IsCurrent
- StartDate
- EndDate

Prior to loading data into a dimension table, the following logic for SCD handling must be implemented:


New row?
Yes
Insert into dimension
No
Changes?
Yes – Type 1
Yes – Type 2
Insert new row: IsCurrent = True StartDate = now
Update Type 1 attributes for all matching rows
Expire existing row: IsCurrent = False EndDate = now

Business rule validation strategy

Not all transform-load jobs will have business rules to validate. However, for those jobs that do require business rule validation, the following steps are to be implemented:

- Any row that violates a business rule will be corrected to use default values and loaded into the target table

Notification requirements

No automated notifications are necessary at this time. IT application specialists will be given responsibility to monitor logs and error files on a daily basis. Conditions that require review include:

- Data failing business rule validation will be logged in a flat file
- Data that cannot be loaded into a target table will be logged in a flat file with an error code
- Operations that generate errors and/or fail will have entries in the SSISDB database that describe the error or failure – see catalog.operation_messages

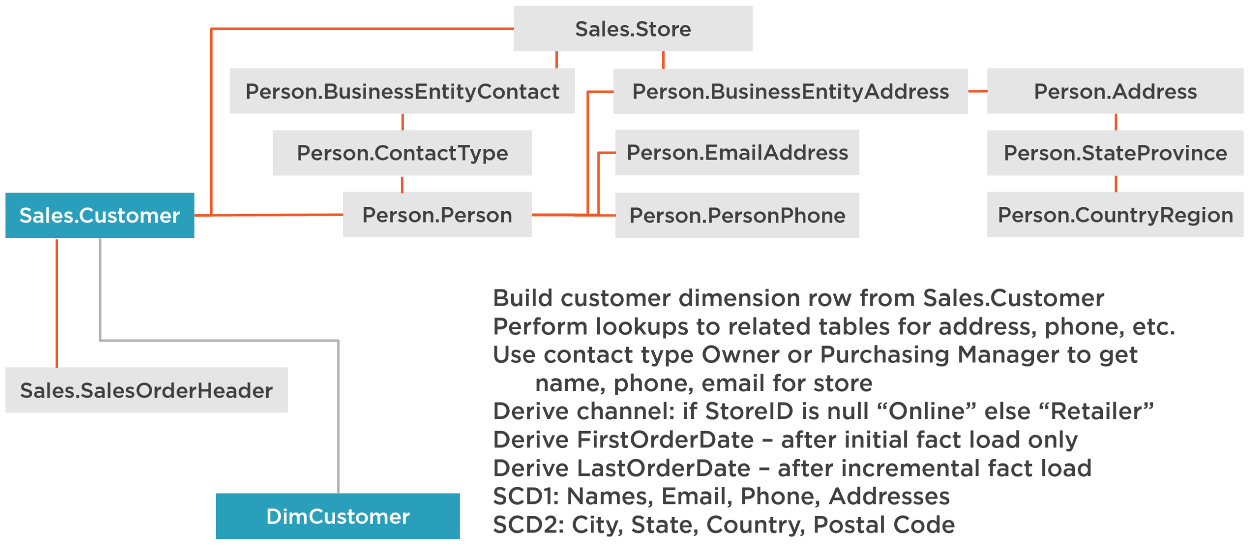
Job-level details

This section describes the details for each job that must be performed to load the target tables into the data warehouse.

Additional reference materials can be found in the following documents:

- Adventure Works Data Dictionary.xlsx (sources)
- AWDW Data Dictionary.xlsx (target)
- Source to Target Mapping.xlsx
- Data Profiling Report.pdf

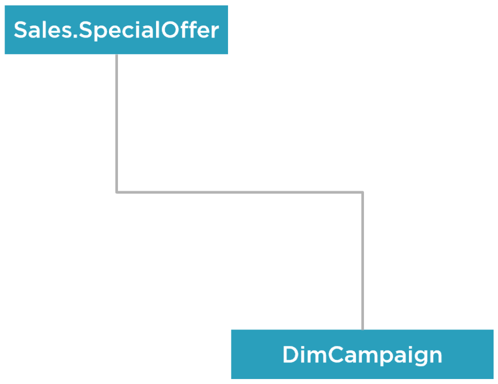
DimCustomer



Build customer dimension row from Sales.Customer
Perform lookups to related tables for address, phone, etc.
Use contact type Owner or Purchasing Manager to get
name, phone, email for store
Derive channel: if StoreID is null “Online” else “Retailer”
Derive FirstOrderDate – after initial fact load only
Derive LastOrderDate – after incremental fact load
SCD1: Names, Email, Phone, Addresses
SCD2: City, State, Country, Postal Code

Table	DimCustomer
Load frequency	Daily
Estimated data volume for initial load	20,000
Estimated data volume for incremental loads	30
Late arriving data?	N
Dependencies?	N
Deviations from standards?	N

DimCampaign



Build campaign dimension row from Sales.SpecialOffer

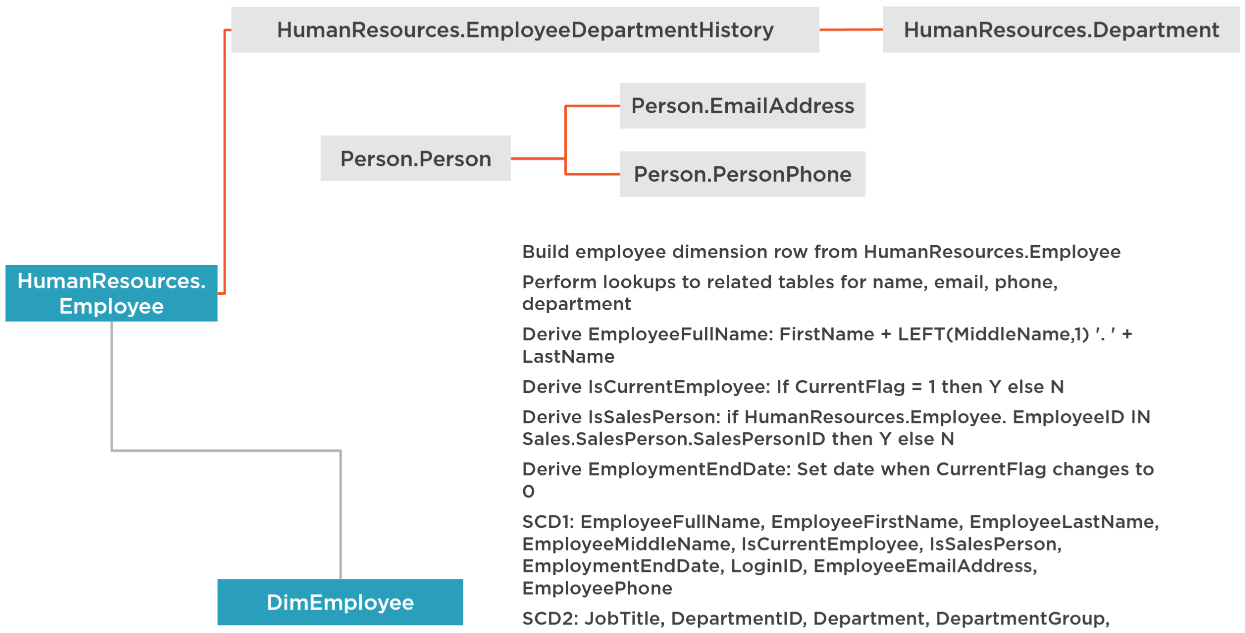
Truncate Description length to 50

Convert StartDate and EndDate from datetime to date

SCD1: CampaignName, CampaignType, CampaignCategory, DiscountPct, CampaignStartDate, CampaignEndDate, MinQty, MaxQty

Table	DimCampaign
Load frequency	Daily
Estimated data volume for initial load	20
Estimated data volume for incremental loads	5
Late arriving data?	N
Dependencies?	N
Deviations from standards?	N

DimEmployee



Build employee dimension row from HumanResources.Employee

Perform lookups to related tables for name, email, phone, department

Derive EmployeeFullName: FirstName + LEFT(MiddleName,1) '. ' + LastName

Derive IsCurrentEmployee: If CurrentFlag = 1 then Y else N

Derive IsSalesPerson: if HumanResources.Employee. EmployeeID IN Sales.SalesPerson.SalesPersonID then Y else N

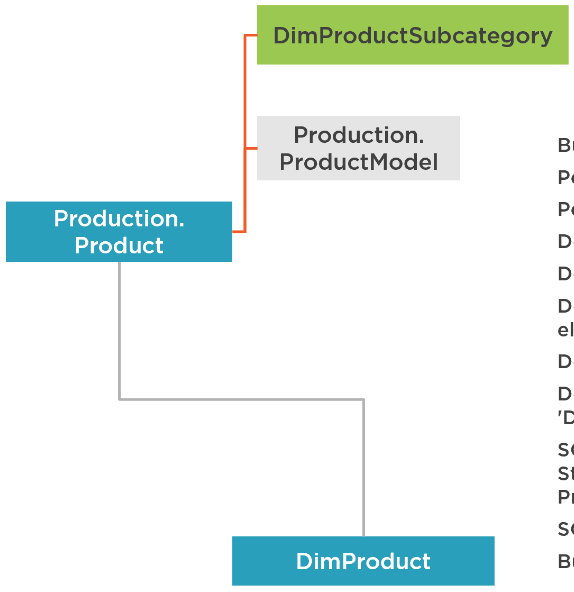
Derive EmploymentEndDate: Set date when CurrentFlag changes to 0

SCD1: EmployeeFullName, EmployeeFirstName, EmployeeLastName, EmployeeMiddleName, IsCurrentEmployee, IsSalesPerson, EmploymentEndDate, LoginID, EmployeeEmailAddress, EmployeePhone

SCD2: JobTitle, DepartmentID, Department, DepartmentGroup, HireDate

Table	DimEmployee
Load frequency	Daily
Estimated data volume for initial load	300
Estimated data volume for incremental loads	5
Late arriving data?	N
Dependencies?	N
Deviations from standards?	N

DimProduct



Build product dimension row from Production.Product

Perform lookup to Production.ProductModel to get model name

Perform lookup to DimProductSubcategory to get FK

Derive Class: Decode H/M/L/NULL as High/Medium/Low/No class

Derive Style: Decode M/W/U/NULL as Men/Women/Unisex/No style

Derive Size: Decode L/M/S/NULL to Large/Medium/Small/No Size; else keep Size

Derive SizeUnitOfMeasureCode: Decode NULL to N/A

Derive ProductCurrentStatus: If DiscontinuedDate <> NULL then 'Discontinued' ELSE 'Current'

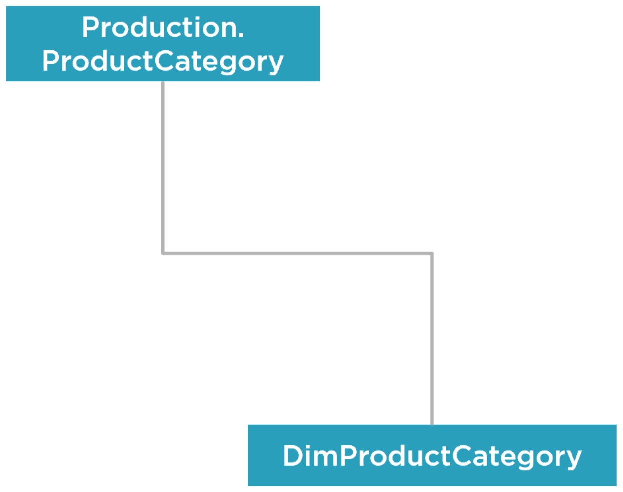
SCD1: ProductName, ProductModelID, ModelName, Color, Class, Style, Size, SizeUnitOfMeasureCode, SellStartDate, SellEndDate, ProductCurrentStatus

SCD2: ProductSubcategoryKey, StandardCost, ListPrice

Business Rule: ListPrice must be greater than StandardCost

Table	DimProduct
Load frequency	Daily
Estimated data volume for initial load	500
Estimated data volume for incremental loads	5
Late arriving data?	N
Dependencies?	DimProductSubcategory
Deviations from standards?	N

DimProductCategory

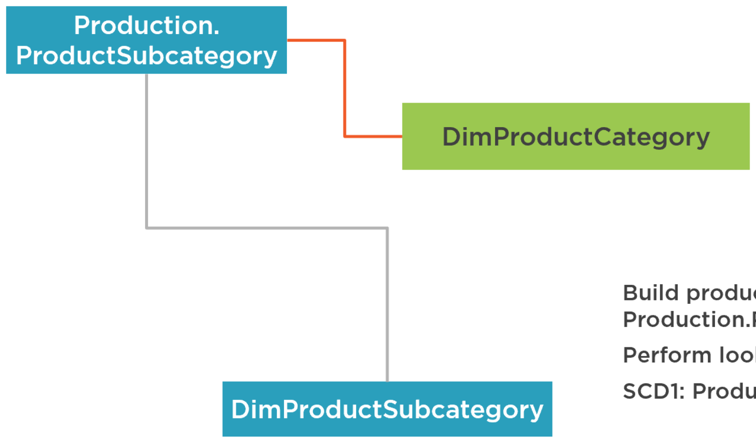


Build product category dimension row from Production.ProductCategory

SCD1: ProductCategoryName

Table	DimProductCategory
Load frequency	Daily
Estimated data volume for initial load	4
Estimated data volume for incremental loads	0
Late arriving data?	N
Dependencies?	N
Deviations from standards?	N

DimProductSubcategory



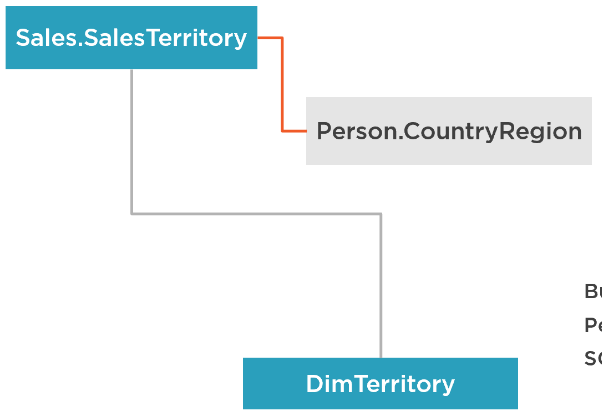
Build product subcategory dimension row from Production.ProductSubcategory

Perform lookup to DimProductCategory to get FK

SCD1: ProductSubcategoryName, ProductCategoryKey

Table	DimProductSubcategory
Load frequency	Daily
Estimated data volume for initial load	40
Estimated data volume for incremental loads	0
Late arriving data?	N
Dependencies?	DimProductCategory
Deviations from standards?	N

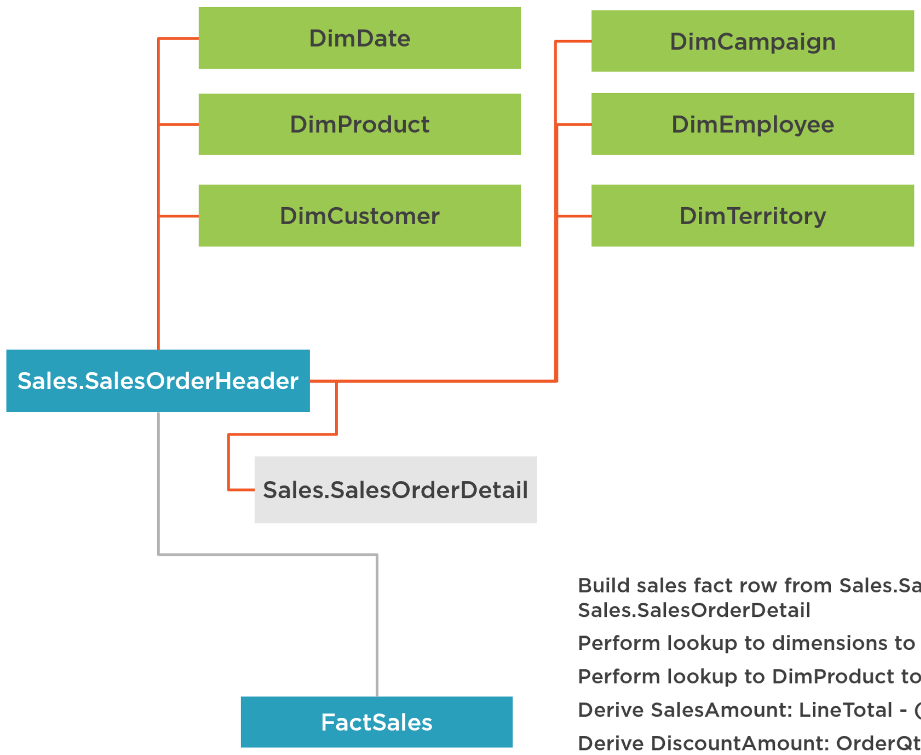
DimTerritory



Build territory dimension row from Sales.SalesTerritory
Perform lookup to Person.CountryRegion to get country name
SCD1: TerritoryName, CountryName, GroupName

Table	DimTerritory
Load frequency	Daily
Estimated data volume for initial load	10
Estimated data volume for incremental loads	0
Late arriving data?	N
Dependencies?	N
Deviations from standards?	N

FactSales



Build sales fact row from Sales.SalesOrderHeader and Sales.SalesOrderDetail

Perform lookup to dimensions to get FKs

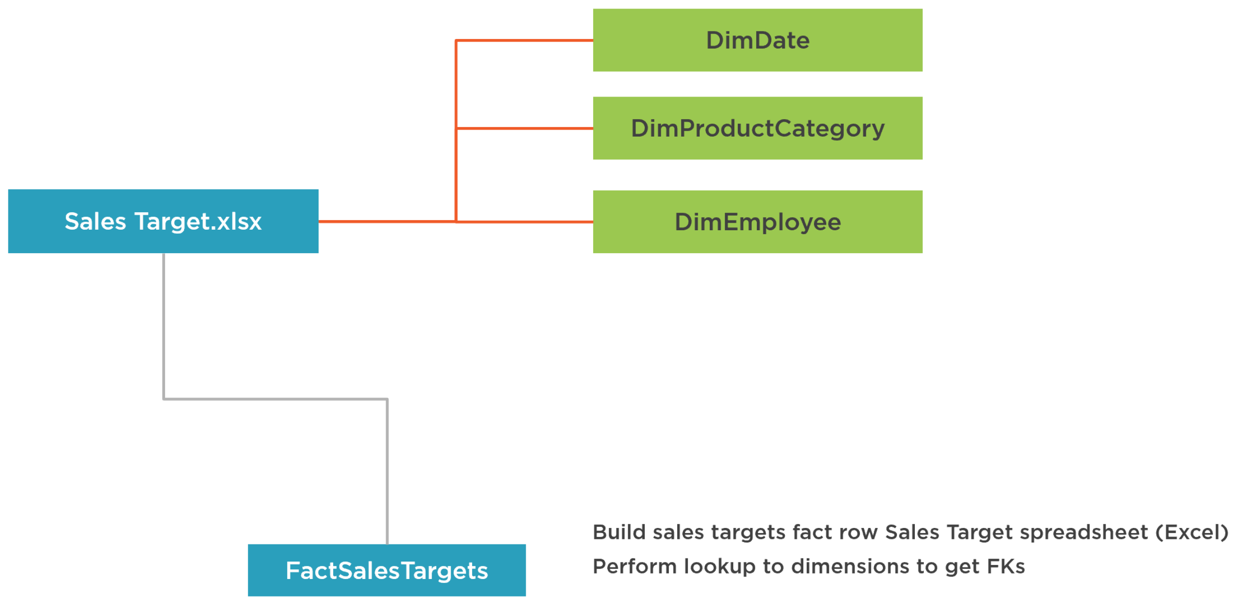
Perform lookup to DimProduct to get standard cost

Derive SalesAmount: LineTotal - (OrderQty * UnitPriceDiscount)

Derive DiscountAmount: OrderQty * UnitPriceDiscount

Table	FactSales
Load frequency	Daily
Estimated data volume for initial load	120,000
Estimated data volume for incremental loads	30
Late arriving data?	N
Dependencies?	All dimensions
Deviations from standards?	N

FactSalesTargets



Build sales targets fact row Sales Target spreadsheet (Excel)
Perform lookup to dimensions to get FKs

Table	FactSalesTargets
Load frequency	Quarterly
Estimated data volume for initial load	700
Estimated data volume for incremental loads	80
Late arriving data?	N
Dependencies?	Dimensions: Date, ProductCategory, Employee
Deviations from standards?	N