Microsoft EDW Architecture, Guidance and Deployment Best Practices

# Chapter One: Overview

**By Microsoft Corporation**

**Acknowledgements:**

**Contributing writers from Solid Quality Mentors**: Larry Barnes

**Technical reviewers from Microsoft**: Benjamin Wright-Jones

**Contributing editors from Solid Quality Mentors**: Kathy Blomstrom

**Published:**

**Applies to:** SQL Server 2008 R2

[Chapter One: Overview 1](#_Toc274371183)

[Introduction 3](#_Toc274371184)

[Why a Data Warehouse? 3](#_Toc274371185)

[Data Warehouse Access Patterns 5](#_Toc274371186)

[Data Warehouse Components 6](#_Toc274371187)

[Data Warehouse Life Cycle and Team Model 7](#_Toc274371188)

[Data Warehouse Life Cycle 7](#_Toc274371189)

[Data Warehouse Team Model 8](#_Toc274371190)

[Data Stewardship 10](#_Toc274371191)

[Data Warehouse Projects 10](#_Toc274371192)

[Managing a Data Warehouse 11](#_Toc274371193)

[Chapter Roadmap 12](#_Toc274371194)

[Chapter 2 – Data Architecture 12](#_Toc274371195)

[Chapter 3 – Data Integration 12](#_Toc274371196)

[Chapter 4 – Operations Management and Security 12](#_Toc274371197)

[Chapter 5 – Querying, Performance Monitoring, and Tuning 13](#_Toc274371198)

[Getting Started 13](#_Toc274371199)

## Introduction

An Enterprise Data Warehouse (EDW) has long been considered a strategic asset for organizations. An EDW has multiple business subject areas which reside in multiple databases.

The success of a data warehouse requires more than the underlying hardware platform and software products. It also requires executive-level sponsorship and governance programs, a solid team structure, strong project management, and good communications.

This document does not focus on those subjects. Instead, the focus is on the best practices and repeatable patterns for developing and maintaining large data warehouses on the Microsoft SQL Server platform (SQL Server), as the focus areas diagram in Figure 1-1 shows.



**Figure 1-1**: Microsoft EDW Architecture, Guidance and Deployment Best Practices focus areas

Note that migrating data warehouses to SQL Server from other platforms is an important topic, but it is also out of scope for this document.

The audience for this document is the project team tasked with designing, building, deploying, maintaining, and enhancing components for a data warehouse built on top of the SQL Server platform and contains the following chapters:

* Introduction – Providing an overview of the toolkit, key data warehouse phases and roles, and an introduction of the remaining chapters
* Data Architecture – Covering design of the database architecture and data models
* Data Integration – Addressing data movement to and from the data warehouse
* Database Administration – Explaining how to manage and maintain the data warehouse
* Querying, Performance Monitoring, and Tuning – Covering optimization and monitoring of query and load performance

Although this document is primarily written for the data warehouse team’s data architects, data developers, and database administrators (DBAs), other team members involved in the development and maintenance of a data warehouse may also find this document useful.

## Why a Data Warehouse?

What’s the benefit of implementing a data warehouse? Although the answer is probably already clear if you’re reading this paper, it’s worth revisiting the challenges that data warehouses address.

Relational databases have traditionally been the data store for Line of Business (LOB) applications. These applications require a “write-optimized” database that supports thousands of concurrent users. Figure 1-2 shows the characteristics of a LOB application. 

**Figure1-2**: Line of Business application characteristics

Table 1 lists some characteristics of an LOB workload, which is also often referred to as On Line Transaction Processing or OLTP.

|  |  |
| --- | --- |
| **Characteristic** | **Line of Business (OLTP)** |
| Concurrent Users | Many (100s - 10,000+) |
| Database Access | Stored Procedures (Stored Procs.), SQL |
| Transaction Scope  | Short: One record – several records |
| Database Operations | Singleton Selects, Inserts, Updates, Deletes |
| Data Volatility | Volatile, Current Data |

**Table 1-1:** LOB workload characteristics

Everything works well until the business starts running large reports against the database, as shown in Figure 1-3.



**Figure 1-3**: Reporting against a LOB database

In such scenarios, you’ll start seeing LOB database reporting issues, including:

* **Blocking and locking** - Reports of any size will request many rows from many tables joined together in one Select statement. This results in the report blocking LOB application activity.
* **Poor performance** – Report performance is often sub-optimal even if there’s no blocking simply because of the amount of joins required to produce the results.
* **Lack of history** – Many reports, such as trend analysis, require historical data to report trends over. LOB systems overwrite historical data with current data, making it impossible to report on historical data.
* **Scope** – Organizations frequently need to report on enterprise level information. This requires information from multiple subject areas, which can span multiple LOB systems.

These LOB reporting limitations then become requirements for a data warehouse:

* **Scope** – The ability to report on the “big picture”
* **History** – The ability to report on historical information
* **Read optimized** – Data models that are tuned in support of large queries

These requirements in turn feed into the following challenges for properly designing and implementing the data warehouse:

* **Scope** – The scope of a data warehouse often crosses organizational boundaries and requires strong communication and governance at the enterprise level.
* **Scale** – The large volumes of data involved require an appropriate database configuration, a scalable software platform, and correctly configured, scalable hardware.
* **Read performance** – Read-optimizing large volumes of data requires a solid data model and, often, physical partitioning schemes to meet the performance needs of the business.
* **Load performance** – Loading large amounts of data into a read-optimized data model within decreasing windows of time requires efficient load procedures.

Data warehouses have traditionally addressed the above challenges by serializing different classes of activities, as we’ll see in the next section.

### Data Warehouse Access Patterns

As Data warehouses mature over time, the emphasis shifts from making data available to consumers to the production of data. Many times this translates into the Data warehouse having a **Production data area** and a **Consumption data area**. Data production processes typically run at night as shown in Figure 1-4. The start time traditionally has depended upon when Source systems have completed their daily processing and post-processing.



**Figure 1-4**: Data warehouse access patterns

Note that the Consumption area is unavailable to consumers during a certain period of time each day; usually this is in the early morning hours when usage is at its lowest. This is due to the fact that load and maintenance activities are write intensive and perform best when they have exclusive access to the database. Also note that the Production area also has backup and maintenance activities, but these can be scheduled throughout the day and are less dependent upon consumer activity.

However, there is increasing pressure to compress the period of time that data warehouse consumption area is closed to consumers, including:

* Requirements to keep the data warehouses available for longer periods. Picture a global Fortune 1000 company with business consumers in all time zones. In these cases, the argument can be made that the Data warehouse should always be open for business.

Combine these demands with increased volumes of data plus the desire for more current data, and you can see the challenges data warehouse teams face as they try to meet all business requirements while maintaining good performance and producing high quality data. Data quality is foundational to business trusting the data within the Data warehouse and is a core requirement for the Data production processes.

## Data Warehouse Components

Let’s look more closely at the data warehouse, beginning with an overview of its components. The Corporate Information Factory (CIF), a well-known architecture in the industry, defines a data warehouse as:

*A subject-oriented, integrated, time variant and non-volatile collection of data used to support the strategic decisions making process for the enterprise.*

For more information on CIF visit Bill Inmon’s web site: <http://www.inmoncif.com>

A data warehouse is fed by one or more sources and, in turn, is accessed by multiple consumers, as Figure 1-5 shows.



**Figure 1-5**: Data warehouse components

A data warehouse consists of the following components:

* The **Production area** is where source data is cleansed, reconciled and consolidated. This is the where a majority of a Data warehouse team’s time and resources are spent.
* The **Consumption area** is where data is transformed for consumption. Reporting rules are applied and the data is normalized and in some cases aggregated. Consumers then use SQL to access the data.
* The **Platform** is the hardware and software products, including network and SAN, which the data warehouse is implemented on.
* **Data integration** processes area responsible for the data movement and transformation as it makes it way from sources through the Production data area to the Consumption data area.

For more information on these components, read Chapter 2 – Data architecture.

A data warehouse has a long life span and is rarely implemented all at once. This is due to a variety of factors including mergers and acquisition, changing business landscape, the request for new subject areas and new questions asked by the business. This translates into ongoing activities in support of a data warehouse as well as ongoing projects to implement new features and functionality. Let’s look at an overview of the common activities in a data warehouse life cycle and the team responsible for each activity.

## Data Warehouse Life Cycle and Team Model

Key factors for any successful data warehouse include:

* The team responsible for developing and maintaining the data warehouse
* The methodology used to develop and deploy the data warehouse
* The processes employed to ensure the ongoing alignment of the data warehouse with its business users

To achieve success in each of these areas, you need to take the organization’s culture and people into account. Thus, it’s difficult to recommend a prescriptive set of best practices without detailed knowledge of an organization.

Because of this, detailed team models, project methodologies and processes, and governance activities are out of scope for this document. However, we will review them briefly in the following section, and then focus the remainder of this document on patterns and best practices for data warehouses implemented on SQL Server.

### Data Warehouse Life Cycle

Figure 1-6 shows a high-level abstraction of a data warehouse’s life cycle, not representing any specific methodology, and highlights the activities that this toolkit focuses on.



**Figure 1-6**: Data warehouse life cycle

The life cycle components include:

* **Governance** – This oversight activity ensures that the organization is aligned around a data warehouse.
* **Define** – This phase, involving developing solid business requirements and translating them into a design, is critical to every data warehouse project’s success.
* **Develop-Test** – This important phase includes development and testing of data models, integration processes, and consumer SQL access for subject areas within the data warehouse.
* **Deploy** – Deployment activities promote data warehouse project deliverables across different physical areas including development, test, QA, and production.
* **Maintain** – Maintenance activities are the ongoing tasks that support a successful data warehouse.

Governance and the Define project phase are beyond the scope of this document. Although we will discuss deployment topics in the remaining chapters, it will be a smaller discussion than that for the Develop-Test and Maintain stages within the life cycle.

### Data Warehouse Team Model

Figure1-7 shows the key roles within the different phases of the data warehouse life cycle.



**Figure 1-7**: Data warehouse roles

Data warehouse roles fall into these general categories:

* **Business** – This group, essential in data warehouse governance and the Define phase, represents the key business roles within the data warehouse team: business sponsors, business analysts, and data stewards.
* **Develop and test** – Developers, quality insurance staff, and data stewards are key team members while the solution is being developed and tested.
* **Technical oversight** – The data architect has an oversight role in all technical phases of the data warehouse.
* **Maintain** – DBAs, data stewards, and IT operations staff are responsible for the ongoing data warehouse maintenance activities.

In addition, release engineering, DBAs, and IT operations are involved in solutions deployment across environments. As noted earlier, although deployment topics will be addressed in the toolkit, they aren’t a primary focus for this.

Table 1-2 maps the level of involvement for different roles in the data warehouse life cycle.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Role | Define | Dev-Test | Deploy | Maintain |
| Project Manager |   |   |   |   |
| Business Analyst |   |   |   |   |
| Data Steward |   |   |   |   |
| Data Architect |   |   |   |   |
| Developer |   |   |   |   |
| QA |   |   |   |   |
| Release Engineering |  |   |   |   |
| DBA |   |   |   |   |
| IT Operations |   |   |   |   |

**Table 1-2**: Data warehouse roles and responsibilities

### Data Stewardship

The data steward role is key team member. Data stewards are responsible for data quality. This role is different from the traditional Quality Assurance role, which focuses on product quality. The ideal data steward has significant tenure within an organization and understands both the business and the source system or systems supporting the business.

The data steward is involved in all phases of a data warehouse:

* **Governance** – The data steward contributes to the business and technical metadata deliverables from ongoing data governance activities and is responsible for ensuring business trust in data warehouse results.
* **Define** – At this stage, the data steward provides subject matter expertise for the business system and underlying database(s) that are sources for the data warehouse.
* **Develop and Test** – The data steward provides subject matter expertise when identifying and diagnosing data quality issues during the development and test phases.
* **Maintain** – During ongoing maintenance, the data steward is responsible for identifying data exceptions and correcting them at their source.

Note that data stewards are not a primary audience for this toolkit; the focus, instead, is on the tools and frameworks that are developed in support of data stewardship.

As noted earlier, our goal is to provide best practices and guidance for:

* Data architects providing oversight and architecture design for the data warehouse
* Database and integration developers implementing the data warehouse
* DBAs responsible for managing and maintaining the Data warehouse.

### Data Warehouse Projects

Most data warehouse projects are responsible for delivering the code and data models that implement an area within the data warehouse. Other projects are foundational and deliver the frameworks used by other projects. Figure 1-8 shows the primary deliverables for each phase of the project and the role mostly responsible for the delivery.



**Figure 1-8**: Roles and primary deliverables

Remember that the Define phase is out of scope for this document. The key project roles and deliverables for the other data warehouse phases are:

* **Data architect** – Responsible for the platform and database architectures
* **Database developer** – Responsible for data models, supporting data objects, SQL queries, and reports
* **Integration developer** – Responsible for integration code and frameworks

### Managing a Data Warehouse

Most of a data warehouse’s costs are related to ongoing management and maintenance. Figure 1-9 shows the key roles and responsibilities for these efforts.



**Figure 1-9**: Ongoing maintenance roles and responsibilities

Ongoing roles responsible for continuing data warehouse management and maintenance are:

* DBAs who manage and monitor the data warehouse
* Data stewards who monitor data quality and repair data exceptions
* IT operations staff who manage and monitor platform components supporting the data warehouse

## Document Roadmap

Building on this overview of the data warehouse life cycle and key roles, we are ready to dig into the patterns and best practices for the development and tests phases of a data warehouse project and the ongoing maintenance of a data warehouse on Microsoft’s SQL Server database platform. Let’s look at what you will find in the remaining four chapters of this toolkit.

### Chapter 2 – Data Architecture

Targeted at the data architects responsible for the database and platform architecture of a SQL Server data warehouse and the database developers responsible for the data models, Chapter 2 covers the following major topics:

* Introduction and overview of roles, responsibilities, and concepts
* Database architecture
* Master Data and Master Data Management (MDM)
* Platform architecture
* SQL Server database considerations
* Data modeling
* Conclusion and Resources

### Chapter 3 – Data Integration

The primary audience for Chapter 3 includes the developers responsible for all data integration code and frameworks as well as the data architects who provide oversight of frameworks, patterns, and best practices. It addresses these essential topics:

* Introduction
* Data Integration Concepts and Patterns
* Data Integration overview
* ETL frameworks
* Data Quality
* Data integration best practices
* SQL Server Integration Services best practices
* Conclusion and Resources

### Chapter 4 – Operations Management and Security

Directed at the DBAs responsible for ongoing data warehouse maintenance, monitoring, and troubleshooting, Chapter 4 covers best practices and guidance in the following areas:

* Introduction and Context
* DBA Data warehouse essentials
* Working with a Data warehouse database server
* Managing databases
* Database server security
* Managing database change
* Business continuity
* Maintenance
* Monitoring tools
* Conclusion and Resources

### Chapter 5 – Querying, Performance Monitoring, and Tuning

The last chapter in the toolkit is targeted at database developers responsible for writing efficient queries and optimizing existing queries against the data warehouse. It’s organized into the following sections:

* Introduction and query optimization overview
* Querying
* Monitoring
* Performance tuning
* Best practices summary
* Conclusion and Resources

Note that product specific best practices and guidance within Chapters 2 – 5 are for the symmetric multi-processing (SMP) versions of SQL Server 2008 R2, i.e. specific guidance for [SQL Server 2008 R2 Parallel Data Warehouse](http://www.microsoft.com/sqlserver/2008/en/us/parallel-data-warehouse.aspx) (PDW) is out of scope for the initial release of this document.

### Getting Started

Each of these chapters contains not only concepts, but also patterns and best practices that Data warehouse practitioners can use to successfully implement SQL Server Data warehouses within their organizations. These patterns and best practices have been successfully utilized by the content authors in client SQL Server Data warehouse engagements over the last ten to fifteen years. In addition, there are links to web content which expands on the material within each chapter.

The reader can choose to read all four chapters or to go directly to the chapter which is closely aligned to the reader’s Roles and responsibilities within the Data warehouse effort.