SQUARED Methods

Harness the power of technology…

White Paper Series

Data Architecture Patterns & Guidelines

What this white paper is about?

Data is lifeblood of organizations today. It is critical to have accurate, timely and trusted data in order to ensure effective decision-making.

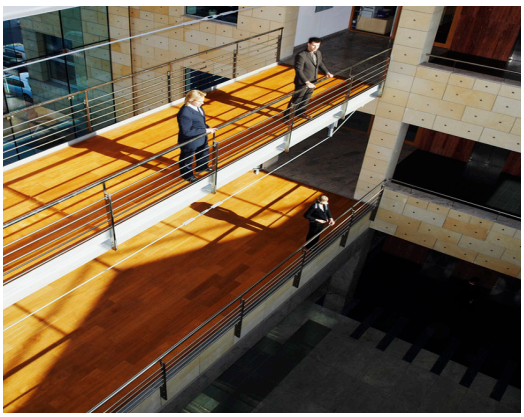
This paper discusses the best practice in creating a Data Architecture that supports all types of data access, from transactional to reporting/KPI and dashboards for organization's business users

Executive Summary

Increasingly, businesses of all sizes thrive on data. Success within the Information Technology hinges on how well organizations collect, manage and extract value from data. At every step, organizations need to leverage accurate, timely and trusted data for decision-making. This is very challenging task since many of the applications and systems deployed within organization are based on commercial IT products, custom developed components or legacy applications for which there is low understanding and knowledge available. This means there is limited visibility into the data assets to business users. In order to transform data from its origin in applications to a credible element in the decision-making process, it is imperative that organizations understand data and are able to present it in a consistent, shared manner and make it available on demand to support the key decision support systems

White Paper Series

Data Architecture Patterns & Guidelines



"We cannot trust the data, as we don't know where our master data is stored. Our inability to trust the data has significant impact on the strategic decision-making ability of senior management"

Need for Enterprise Data Architecture

In order to ensure that critical data assets are available to enterprise wide users in accurate, timely manner when needed, it is important to have a clear view of how to integrate these "islands" of data within organization. What is needed is a unified way of managing enterprise data across the entire organization. The requirement is for a clear architecture that addresses the data and information needs; including:

- Reporting (Operational, Trends, Historical)
- Analytics (Trends, Dimensional analysis)
- Decision support systems (Dashboards, KPIs)
- Aggregations (Combining data from multiple applications at different aggregation levels)

Typically, huge amount of time and effort is spent on addressing "data issues"; which includes identifying the sources of needed data, evaluating them, extracting and transforming data, dealing with data quality problems, and correcting software errors due to data-related problems. All these activities would be completed more effectively, rapidly and at lower cost with good data architecture practices & policies in place

To put in simple terms, our ability to "manage data across the enterprise" provides organizations with better ways of supporting decision support systems and better ways of leveraging critical data assets. This requires an investment in architecture, direction, and set of policies & governance practices to produce order from chaos.

Well-defined data architecture presents long-term benefits to organizations:

- Promotes a common understanding of enterprise data assets via best practices and governance policies and provides clear guidelines on which data store to use to address specific business needs
- Promotes clear ownership of data, sources of record and facilitates common understanding of data assets

White Paper Series

Data Architecture Patterns & Guidelines

- Ensures consistent meta-data is collected across the company and published for providing visibility into data assets
- Ensures that any edits, defaults or referential integrity needed to meet the data quality needs of downstream consumers are implemented in the source of record or as far upstream as possible given cost and resource limitations.
- Ensures that business data corrections are made to the source of record and flow through to downstream systems.
- Ensures availability of high quality data when needed for business specific needs
- Establishes clear policies for sharing data across sectors, functions and application boundaries to promote enterprise wide data sharing

Data Architecture is typically based on the foundation of five basic components – application specific Transactional Data Store (TDS), Operational Data Store (ODS), Enterprise Data Warehouse (EDW), and Data Marts (DM); bound together through the metadata in a central Metadata repository (MDR) and Master Data Management (MDM

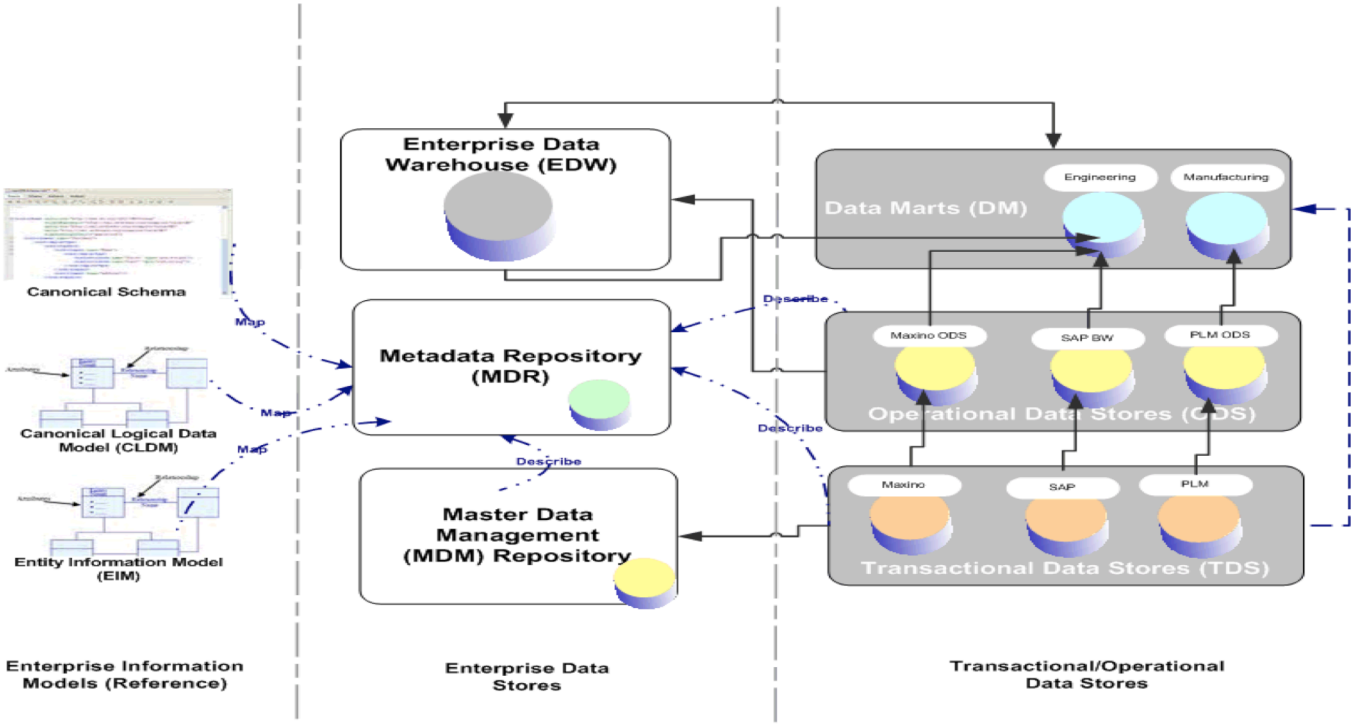


Figure 1: Typical Data architecture design patterns:

Data Architecture Patterns & Guidelines

Data Store Characteristics

The characteristics of the five primary types of data stores are summarized below.

Characteristic	Transactional Data Store (TDS)	Operational Data Store (ODS)	Enterprise Data Warehouse (EDW)	Data Mart (DM)	Master Data Store (MDM)
Purpose	Supports the creation and maintenance of business data. Geared toward specific business function. The Source of Record for most data.	Used by downstream applications to retrieve integrated corporate data originating from upstream sources of record.	Used by production applications and end-user community for known and unknown decision support, reporting, and analytic needs.	Used by production applications and end-user community for specific decision support, reporting, and analytic needs.	Used to store reference master data that can be leveraged by applications as trusted data source for specific master subject area
Data Content	Only data created by or used by the primary transactional applications that the TDS supports.	Real time or near real time Transactional Data that is needed by multiple applications for operational reporting	All business relevant data at a point in time.	Only data specific to subject or business area that mart is geared towards.	Only reference master data attributes for specific master subject area
Structure	Application or function oriented, specific, Highly normalized and detailed.	Typically normalized, Subject oriented, integrated, and detailed.	Subject oriented, integrated, detailed, and/or summarized. Partitioned and indexed to handle queries accessing large amounts of data.	Subject oriented, denormalized, aggregated and/or summarized.	Typically Subject oriented

White Paper Series

Data Architecture Patterns & Guidelines

Characteristic	Transactional Data Store (TDS)	Operational Data Store (ODS)	Enterprise Data Warehouse (EDW)	Data Mart (DM)	Master Data Store (MDM)
Typical Latency of Data	Current day, current transaction	Current day, possibly some delay for transactions (intra-day) based on requirements	Typically Previous day	Previous day or less, as defined by requirements	Data is reference only so very stable and minimal/non-frequent changes are expected
Includes History Y/N	N	N	Y	Y	History of reference data may be maintained as required
Data Creation and Update	Transactional, and/or on-line data entry.	Transactional feeds from TDS's. No direct on-line update.	Batch feeds from the ODS and TDS's. No direct on-line update.	Typically fed from EDW "base" tables typically in a batch manner. No direct on-line update. Can also draw data from ODS to facilitate analysis if needed	Typically based on transactional data attributes for Master Subject Area
Data Retrieval	Highly controlled and predictable - no ad-hoc queries.	Highly controlled and predictable – no or minimal ad-hoc queries.	Queries may be large and unpredictable. Ad-hoc abounds.	Queries tend to be repetitive or related. Some ad-hoc.	Queries for reference data tend to be repetitive.
Data Access Methods	Front end GUI, Web or batch programs. If application supports out-of-box reporting, it can be leveraged, but no direct ad-hoc access	Front end GUI, Web or batch programs through a data access layer which masks database structures. If ODS supports out-of-box reporting, it can be leveraged, limited or no direct ad-hoc access	Ad-hoc query tools, report writing tools, front end web or GUI.	OLAP tools, report writers. Some web or GUI. Some ad-hoc query tools.	Some web or GUI. Some ad-hoc query tools. API or web service access preferred

Performance	Tuned to specific purpose (i.e. on-line update/retrieval or batch processing).	Near real-time update from TDS and sub-second response to data retrieval requests.	Can take seconds to multiple minutes based upon query complexity.	Response time relative to amount of data being accessed.	Superior response time required due to enterprise wide reference

Architectural Considerations

Transactional Data Store (TDS)

- When deciding whether to use/extend an existing TDS or to create a new TDS, the following data aspects shall be considered:
  - Overlap in content of the data, state, timeliness and quality of the data
  - Performance goals of the transactional systems using the TDS (e.g. flexibility vs. performance or throughput), availability and recoverability requirements
  - System of record for the data
  - The TDS shall be modeled to meet the goals of the transactional systems it is supporting, which might be for flexibility or performance.
- Having redundant data across TDSs is not necessarily bad. What's important is that:
  - Any redundancy is documented and redundant data isn't updated or used as a source for other systems.
  - A production mechanism is put in place to ensure the concurrency of the redundant data.
  - The data copy is monitored to ensure correctness.
- User access to the TDS shall be restricted to production support personnel to minimize performance impacts to the associated applications.

Operational Data Store (ODS)

- Data is only stored in the ODS where there is a current or anticipated future operational reporting need based on the data across transactional applications and the ODS is populated from sources of record, generally from TDSs
- The exact mix of latency of data in an ODS can be determined based on the relevant TDSs and the specific business requirements for operational reporting
- Data can be either pushed to the ODS or pulled by the ODS depending upon a number of factors including the architecture of the source system, performance requirements of the source system, latency requirements for the ODS, and the volume of data being moved
- The ODS data Services should contain little or no business logic to encourage re-use

- It is not required to have separate database per ODS Data Service, and multiple databases can serve a single ODS Data Service
- The ODS is typically normalized, because it will integrate data from many different transactional systems and is used by many different transactional systems with possibly different goals.
- Production applications must access the ODS through the ODS Data Services and User access to the ODS should be restricted to production support personnel to minimize performance impacts to the applications using the ODS Service.

Enterprise Data Warehouse (EDW)

- The EDW is intended to support ad-hoc users and decision support applications such as reporting, modeling, OLAP analysis, etc.
- The EDW is designed to accommodate the large volumes of data and types of queries typical on a multi-terabyte warehouse.
- Because EDW is intended to be accessed directly by users, the data shall be categorized and protected to avoid unauthorized access

Data Marts (DM)

- Data marts are logically part of the warehouse environment. Data marts shall be deployed on the same database platform as the EDW
- Data marts are typically built from the EDW base tables to keep the load process as straight forward as possible, minimizing load times and the risk of introducing errors.
- It is acceptable in certain situations for Data Marts to be built from a combination of EDW base tables and ODS and/or TDS tables to combine the data. This is a case if not all data from ODS was populated in EDW but the requirements need access to some data in ODS. These shall be considered exceptions and put on the EDW roadmap for integration
- Data Marts may store pre-calculated analytical results in cases where the recreate time exceeds the required retrieval time.
- As a best practice, Data Marts shall be designed using Dimensional (Star) schemas or Snowflake schemas depending on the data & analytical needs of users

Metadata Repository (MDR)

- The MDR is a special database that's based upon a commercial product with extended use to add custom metadata. A web-based front-end is part of the tool set to share available metadata within the enterprise and to meet our meta-data requirements.
- Researching for visibility into enterprise data would typically use the MDR through the front-end application
- Transactional systems or traditional decision support applications shall not use MDR as a source of data

Data Architecture Patterns & Guidelines

Master Data Management (MDM) Repository

From Architectural standpoint, there are multiple options for implementing MDM solution. Some of the commonly used approaches are:

Repository architecture

In Repository style hub architecture, complete collection of master data for a subject area is stored in a single central data store. The data model for this central data store includes all attributes required by all applications that use the master data. The applications that consume, create, or maintain master data are all modified to use the master data in the hub, instead of the master data previously maintained in the application database.

Registry architecture

In Registry style hub architecture, master data for a subject area is stored in applications and transactional data sources where it is created, and a central registry maintains a list of keys that can be used to find the relevant attributes for the master subject area. So in many ways, this is exactly opposite to Repository style MDM Hub Architecture. While this approach allows master data to remain native to applications thereby eliminating need for major changes to the existing systems & application interfaces, there are some challenges with this approach

Hybrid architecture

The hybrid model incorporates features of both the repository and registry approaches. It recognizes that, in most cases, it is not practical to modify all applications to use a single master data repository, but it may not be feasible to make every MDM hub query a distributed due to complexity involved. So hybrid approach leaves the master-data records in the application databases and maintains keys in the MDM hub (registry model), at the same time; replicates some of the most important attributes for master entities in a central MDM hub, so that significant number of queries can be satisfied directly from the hub database, thereby reducing the complexity of common queries; while leaving less-commonly queried attributes the application database.

White Paper Series

Data Architecture Patterns & Guidelines