UNIVERSITÄT LEIPZIG

On Metadata Interoperability in Data Warehouses

Hong Hai Do, Erhard Rahm

Report Nr. 01(2000)

ISSN 1430-3701

On Metadata Interoperability in Data Warehouses

Hong Hai Do, Erhard Rahm

Department of Computer Science, University of Leipzig
Augustusplatz 10/11, 04109 Leipzig, Germany

March 2000

Abstract. In current data warehouse environments there is either no or only insufficient support for a consistent and comprehensive metadata management. Typically, a multitude of largely autonomous and heterogeneously organized repositories coexist. We categorize the major metadata types and their interdependencies within a three-dimensional classification approach. We then investigate how interoperability and integration of metadata can be achieved based on a federated metadata architecture and standardization efforts such as OIM and CWM. In particular, we examine synchronization alternatives to keep replicated metadata consistent. We also give an overview of currently available commercial repositories and discuss interoperability issues to couple data warehouses with information portals.

1 Introduction

The successful operation and use of data warehouses heavily depends on the effective management of a multitude of metadata. This metadata is needed to describe all relevant aspects of the warehouse data obtained from heterogeneous sources, maintained in a central warehouse and possibly multiple data marts, and made accessible in various ways, e.g. for querying, OLAP, navigation or data mining [MA97, De97]. A comprehensive metadata management is also needed to guarantee a high quality of the warehouse data and provide sufficient flexibility to extend the scope of the warehouse to new data sources and new information requirements.

Data warehouses integrate massive amounts of heterogeneous and possibly inconsistent source data. This *data integration* should be based on the integration of the corresponding metadata to explicitly describe how the warehouse schema is derived from metadata of the operational sources and how to properly perform data extraction, transformation and consolidation. *Metadata integration* also has to deal with heterogeneity as source systems substantially differ in the degree and form they provide describing metadata. The main problem is to deal with semantic heterogeneity for which no automatic solutions are possible [HST99]. This is because available metadata of the data sources (database schemata, file structures) primarily describe structure; semantic aspects are typically restricted to some integrity constraints, hopefully meaningful names and describing comments. Often the meaning of data is not represented at all but hidden in application programs or only known to some people. Prerequisite to metadata integration is *metadata interoperability*, i.e. the ability to exchange metadata or provide access to metadata among the various components of a data warehouse configuration.

A typical data warehouse environment is shown in Figure 1. It consists of file systems and DBMSs managing the operational sources, the data warehouse and data marts. Furthermore, a variety of tools from different vendors is usually involved for data modeling, ETL tasks (extraction, transformation, loading), and data access (OLAP, querying, etc.). All of these components create and maintain metadata, e.g. within database catalogs, dictionaries or tool-specific repositories. Typically metadata is maintained independently in a largely isolated way and in specific representation formats. Metadata required by more than one component, i.e. shared metadata, needs to flow between the components as indicated in Figure 1 typically resulting in metadata replication. The use of different metadata representation models complicates metadata interoperability between components, in particular for propagation of

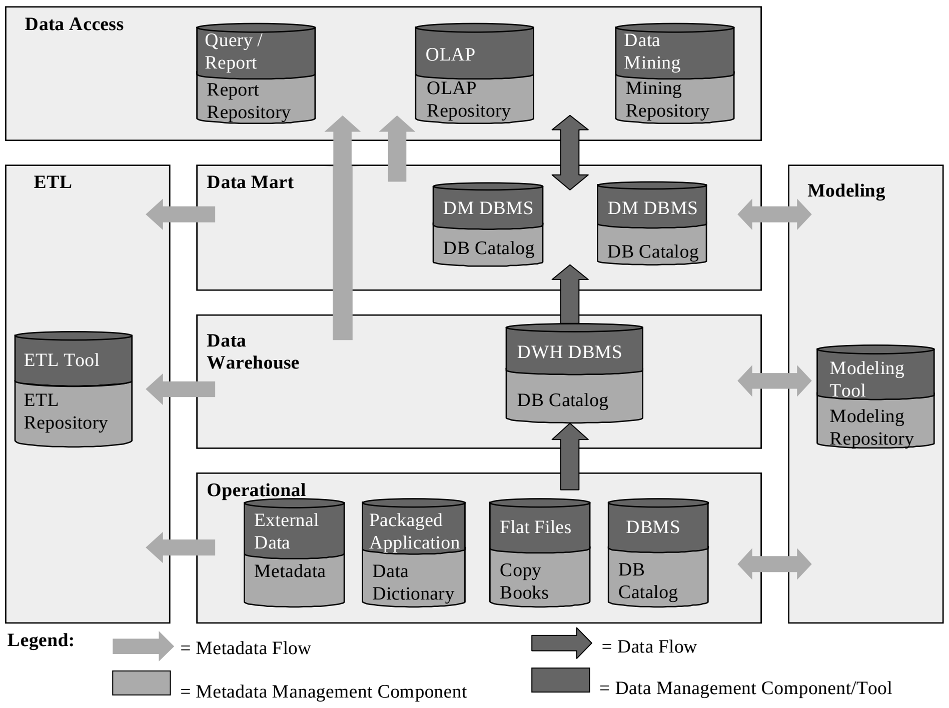

Figure 1: Distribution of data and metadata in a data warehouse

updates to keep replicated metadata consistent. It is almost impossible for end-users or data access tools to make good use of such decentralized and heterogeneous metadata. In practice, these limitations are often found [Ma99] thereby severely restricting the usefulness and scope of a data warehouse or even causing failures of entire data warehouse projects.

Consistently supporting *shared metadata* is thus crucial for data warehouses.[1] Repository support is needed permitting tools and other data warehouse components to access, create, and extend shared metadata. Another key requirement is a powerful information model for uniformly representing all types of relevant metadata. This *metadata model* should be extensible to represent new types of metadata. Particularly important is support for metadata providing a *business-oriented view of the warehouse data* abstracting from technical details such as table and column names. This requirement is not sufficiently supported in current systems. It requires a solution to the semantic heterogeneity problem and a mapping of data elements to a consistent set of business terms.

Data warehouses are increasingly accessed via so-called *information portals* [Wh99, Fi99, Ha99], providing a uniform Web-based interface for different access forms such as OLAP, querying, report generation, navigation or data mining. Furthermore, portals offer integrated access to a variety of structured and unstructured data maintained outside data warehouses, in particular news channels and digital libraries containing text documents or multimedia contents. A personalized view of all data is provided to end-users considering their interest profiles and access rights. Obviously, information portals require a powerful repository to provide this functionality. This also introduces new interoperability requirements for the cooperation between the portal repository and warehouse-specific repositories.

In this paper, we discuss the interoperability issues for metadata management in data warehouses in more detail. This discussion is based on a three-dimensional classification of the major types of metadata which is introduced in Section 2. Section 3 gives a brief overview of current metadata models required for integrating different types of metadata, especially the competing standardization efforts OIM and CWM. In Section 4, we discuss architectural

[1] An example of shared metadata is the warehouse schema that may be created by a modeling (CASE) tool, implemented in the warehouse DBMS, used by an ETL tool for the definition of data transformations and required by data access tools for the definition of queries and reports.

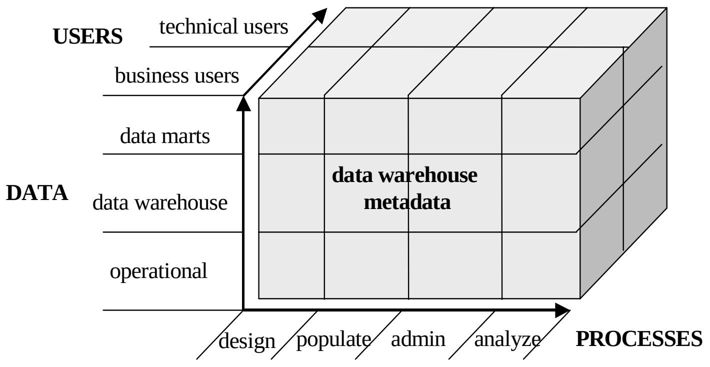

Figure 2: Three-dimensional classification of data warehouse

alternatives for management of shared metadata. In particular, we differentiate between centralized, decentralized and federated approaches and distinguish various cooperation forms (API-based, exchange files, use of wrappers). Approaches to maintaining consistency of replicated metadata are discussed in Section 5. Section 6 contains a brief comparison of commercial repository products. Finally, we discuss the use of information portals with respect to metadata interoperability.

2 Classification of Data Warehouse Metadata

Addressing the metadata management problem requires a thorough understanding of what types of metadata need to be dealt with. There have been several attempts to classify data warehouse metadata [De97, SVV99, Wi98] but still there are no generally agreed-upon criteria. This is also because metadata can be viewed from different perspectives depending on what a particular study is focussing on. Furthermore, it is not possible to categorize warehouse metadata into disjoint subclasses because of numerous dependencies between different metadata types which, in fact, are one cause for the importance of supporting shared metadata.

2.1 Overview

We propose a simple classification scheme for data warehouse metadata differentiating along the dimensions data, processes and users. Figure 2 shows the resulting three-dimensional classification cube. The ***data*** dimension classifies metadata according to the data being described. Based on the warehouse architecture, we distinguish between metadata associated with operational systems, the data warehouse, and data marts.

While the data dimension represents a static, component-based view, the ***processes*** dimension deals with dynamic aspects and covers metadata associated with the four main warehouse processes of designing, populating, administering, and analyzing. The *design process* is typically supported by a modeling tool and defines conceptual models, views etc. of the warehouse and data marts. It requires an integration of metadata obtained from the data sources. The *population process* is typically supported by ETL tools and has to feed the data warehouse and data marts with data required for analysis. It has to implement data integration by executing rules and scripts for transforming, cleansing and mapping operational data into the data warehouse or data marts. The *administration process* comprises operation, maintenance, and tuning of the entire data warehouse environment except the operational systems. Finally, the *analysis processes* are supported by specific end-user data access tools for decision support, e.g. for navigating/browsing, querying/reporting, OLAP, and data mining.

Our third dimension concerns ***users*** where we roughly distinguish the two groups of technical and business users. *Technical users* include database administrators, designers, and programmers. They are primarily concerned with the task of developing and maintaining the system. *Business users* include managers, analysts, etc. who use the technical infrastructure

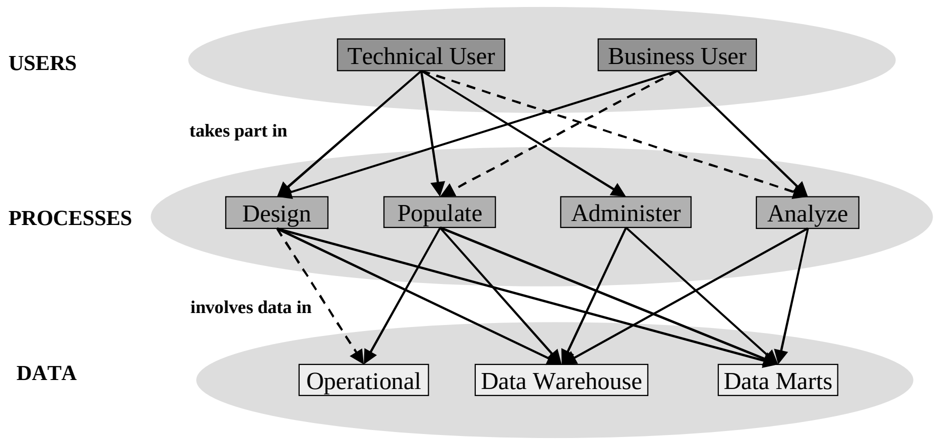

Figure 3: Associations between *users*, *processes* and *data*

developed by the technical users to perform data analysis. To address the different demands for metadata of these user groups, we differentiate between *technical* and *business metadata*. This classification criterion is quite common in the commercial world, e.g. [In98, IBM98b, Ma99].

As already mentioned, there are tight relationships between the various subcases with respect to their metadata requirements. For instance, technical users are concerned with several processes and several warehouse components. On the other hand, not all possible combinations in our three-dimensional cube are relevant, e.g., the combination [Operational Systems, Analyze, Technical User]. In Figure 3 we have indicated the main associations between the three dimensions some of which are less pronounced (dashed lines). In the following, we discuss this for technical and business metadata by looking at the processes and data they are associated with.

2.2 Technical Metadata

Technical metadata is relevant to all warehouse processes and all data components (operational systems, data warehouse, data marts).

- *Data schemata* (e.g. database schemata or file structures) represent the most important part of technical metadata. They are generated and implemented during the designing or the populating process, and maintained during the administering process. They are also involved in the analyzing process to access and retrieve data from the data warehouse or data marts. Data schemata of operational systems are heterogeneously represented, e.g. file- or record-oriented, relational, object-oriented, hierarchical or network schemata. By contrast, the data warehouse and data marts are homogeneous in nature and usually described either by relational or multidimensional database schemata.
- Technical descriptions about the source and target systems involved in the populating process (server names, network addresses, database names, file names, etc.)
- *Data dependencies* between the various layers of a warehouse environment (operational data - warehouse data; warehouse - data marts; warehouse/data marts - data analysis objects such as queries, reports, data cubes) need to be maintained to systematically manage and propagate changes. Some of these are specified for the population process, e.g. rules for data cleansing, transforming, and mapping. The analyzing process generates technical metadata about how the analysis-specific objects are created (e.g. SQL statements, aggregations, filters), and in which form they are stored (e.g. database tables, files).
- Metadata for administration purposes includes *system statistics* for performance tuning (usage pattern, CPU and I/O requirements, ...), e.g. to define index structures or pre-com-

pute aggregations. *Data refresh* is based on technical metadata for scheduling population jobs, status information on the progress and success of such jobs, etc. *Purging* and *archiving* old warehouse data is based on metadata such as the time and frequency of purging actions, strategy for selecting „old“ data, etc.

2.3 Business Metadata

Business metadata helps business users to understand the meaning of the data and supports business-oriented data analysis. It can be generated by design, population, or analyze processes.

- *Information models* (or conceptual data models) represent a technology-independent specification of the data required by the enterprise to run its business. Typically, they are based on the entity-relationship approach or UML (Unified Modeling Language). Information models can exist for operational systems (operational data models), the data warehouse or data marts. In addition, a comprehensive *enterprise data model* is desirable defining the main business information objects, relationships among them and integrity constraints (business rules).
- Enterprise modeling is to be based on a consistent definition of *business terms*. Business terms describe business-relevant entities in natural language and should be familiar to end-users. The descriptions are maintained in a lexicon. Construction of the enterprise data model and business terms is discussed in [LJ99].
- To allow business-oriented data access, *mappings between business terms and data mart / data warehouse data elements* (e.g. tables, attributes) are needed. Business users should be able to formulate ad-hoc queries using familiar business terms instead of a complex query language such as SQL. The mappings may be defined during the design of the enterprise data model, e.g. after the definition of business terms is completed.
- Business users require understandable *descriptions of predefined queries, reports and data cubes* in the system. Related business metadata includes responsible person (data stewardship), owner (data ownership), creation time, frequency of updates, corresponding topics and business areas, etc.
- Business users want *information about the state and the quality of the data* in the data warehouse and data marts. Based on technical metadata generated during the populating and the administering processes (ETL rules, scheduling of ETL jobs and purging activities, ...), business metadata entails information about data lineage (where the data comes from), data accuracy (how the data has been transformed), or data timeliness (when was the data last refreshed or purged).
- *Metadata for personalization* such as user profiles, user roles, subject areas of business data, and associations between users, user roles, subject areas and data elements in the data warehouse / data marts.

Obviously, there are associations and overlaps between technical and business metadata too. In contrast to technical metadata, business metadata is not yet widely supported in data warehouses. It entails a substantial definition and modeling effort, in particular to define a consistent set of business terms and to develop a comprehensive enterprise data model.

3 Metadata Models

Managing the different types of metadata requires repositories with a powerful metadata model. Several vendors offer general repositories supporting data warehousing environments as well as other application domains such as software development (see Section 6). In addition, most data warehousing tools use built-in repositories for their specific functionality thus typically covering a particular subset of the metadata. Several research efforts have also proposed metadata models suitable for data warehousing [BH98, JJQ+99, MSR99].

3.1 Requirement for a Uniform Representation of Warehouse Metadata

As discussed in the introduction, metadata from multiple sources needs to be integrated and thus mapped between different repositories. If every participating repository uses a different metadata model the mapping process becomes very complex and hard to maintain. Furthermore, it may not be possible to propagate all required metadata due to differences in the expressive power of the metadata models. These problems are analogous to the well-known schema translation and integration problems of federated database systems [SL90]. Schema translation is performed to transform different database schemata into a common data model (e.g., object-oriented model). Schema integration then combines the homogeneously represented but independent database schemata into a combined global schema. The main problem in this step is to resolve semantic schema conflicts. Apparently, the proposed research approaches have had little impact in the commercial world so far.

The metadata integration problem of data warehouses is partially more complex since not only schemata but a multitude of additional metadata needs to be dealt with. Uniformly representing this metadata (and thus solving the translation problem) can substantially be simplified by a powerful metadata model standard if it is supported by many repositories. Currently, there are two competing industry efforts for such a standard metadata model for data warehouses, namely OIM and CWM. The OIM (Open Information Model) effort is led by the Metadata Coalition[2] (MDC) and supported by Microsoft and other companies [MDC99]. CWM (Common Warehouse Metamodel) has been introduced more recently; it is an approach of the Object Management Group[3] (OMG) and supported by IBM, Oracle and others [OMG99]. We briefly discuss and compare these approaches in the following.

3.2 Technical and Business Metadata in OIM and CWM

The aim of OIM is to generally support tool interoperability across technologies and companies via a shared information model. OIM contains metadata from a broad range of subject areas, i.e. software development, object-oriented analysis, business engineering, data warehousing, and knowledge management. CWM, on the other hand, addresses metadata interchange specifically in the data warehouse environment, i.e. between warehouse tools. Both metadata models make use of OMG's standard modeling language UML (Unified Modeling Language) for the specification, visualization and documentation of their metadata types. The object-orientation supports extensibility of the models by defining additional (specialized) submodels. CWM also conforms to the Meta Object Facility (MOF), an OMG metadata interface standard used for defining and representing metadata as CORBA objects.

Technical Metadata		**OIM Submodel**	**CWM Submodel**
Data Schemata	Relational	*Relational Database Schema*	*Relational Data Resource*
	Record-oriented	*Record-oriented Database Schema*	*Record-oriented Data Resource*
	Multidimensional	*OLAP Schema*	*Multidimensional Data Resource*
			OLAP
	XML	*XML Schema*	*XML Data Resource*
Data Transformation		*Data Transformations*	*Transformation*
Warehouse Administration		▨	*Warehouse Deployment*
			Warehouse Process
			Warehouse Operation

Table 1: OIM and CWM submodels for technical metadata

[2] www.mdcinfo.com

[3] www.omg.org

As summarized in Table 1, OIM and CWM have similar submodels for representing *technical metadata* such as data schemata, (i.e. relational, record-oriented, multidimensional, XML), and data transformations. CWM includes three additional submodels describing other technical aspects of the data warehouse environment:

- *Warehouse Deployment*: records how the software systems in a data warehouse are used and where they are installed (which computer, geographical location).
- *Warehouse Process*: documents process flow used to execute the transformations. A warehouse process associates a transformation to a series of events (scheduled, internal or external) used to trigger the execution of the transformation.
- *Warehouse Operation*: covers runtime metadata, e.g., about time and state of recent executions of transformations, and a historical record of metadata changes

Business Metadata		**OIM Submodel**	**CWM Submodel**
General Aspects	Contact Information	*Generic Elements*	*CWM Foundation*
	Textual Descriptions		
Data Analysis	Reporting	*Report Definitions*	*Information Reporting*
			Information Visualization
	Data Mining	▨	*Data Mining*
Knowledge Management	Conceptual Modelling	*Semantic Definitions*	▨
	Business Term Modelling	*Knowledge Descriptions*	*Business Nomenclature*

Table 2: OIM and CWM submodels for business metadata

Regarding *business metadata*, both OIM and CWM provide basic metadata types to describe general aspects of business information such as contact information (data stewardship), textual decriptions, etc., which can be attached to other model elements. Furthermore, both models contain some similar submodels devoted to business metadata from several areas of data warehousing, such as data analysis and knowledge management (see Table 2).

- *OIM Report Definitions* / *CWM Information Reporting*: both provide metadata types to represent formatted reports (report formatting definitions, relationships between report fields and corresponding function / query expressions, ...).
- *OIM Knowledge Descriptions* / *CWM Business Nomenclature*: both provide metadata types to describe and categorize information. A semantic network of taxonomies, concepts, terms, glossaries, etc. can be built allowing the sharing and the collaboration of knowledge in an organization.

Besides such overlap, both OIM and CWM contain some unique submodels:

- *OIM Semantic Definitions*: accommodates conceptual models of user information which are basically built on three semantic concepts: entities, relationships, and dictionary entries. Schema-to-semantic mappings linking physical data with business concepts and relationships defined by means of linguistic phrases allow end-users to interact with a database without learning a data retrieval and manipulation language such as SQL.
- *CWM Information Visualization:* provides the foundation for the *CWM Information Reporting* submodel. It defines generic metadata types describing the mechanism for visualizing, i.e. rendering, an arbitrary model element in two dimensions (e.g. displaying a query result set in different formats, such as printed reports, Web page, charts, etc.).
- *CWM Data Mining*: contains metadata types and associations related to the data mining process, such as a generic data mining model, mining settings driving the construction of the mining model, input attributes and their usage, mining result set, etc.

Currently, both OIM and CWM do not address the management of users, access rights, user profiles and the association between users and information objects. Therefore, personalized views on warehouse data are not yet supported.

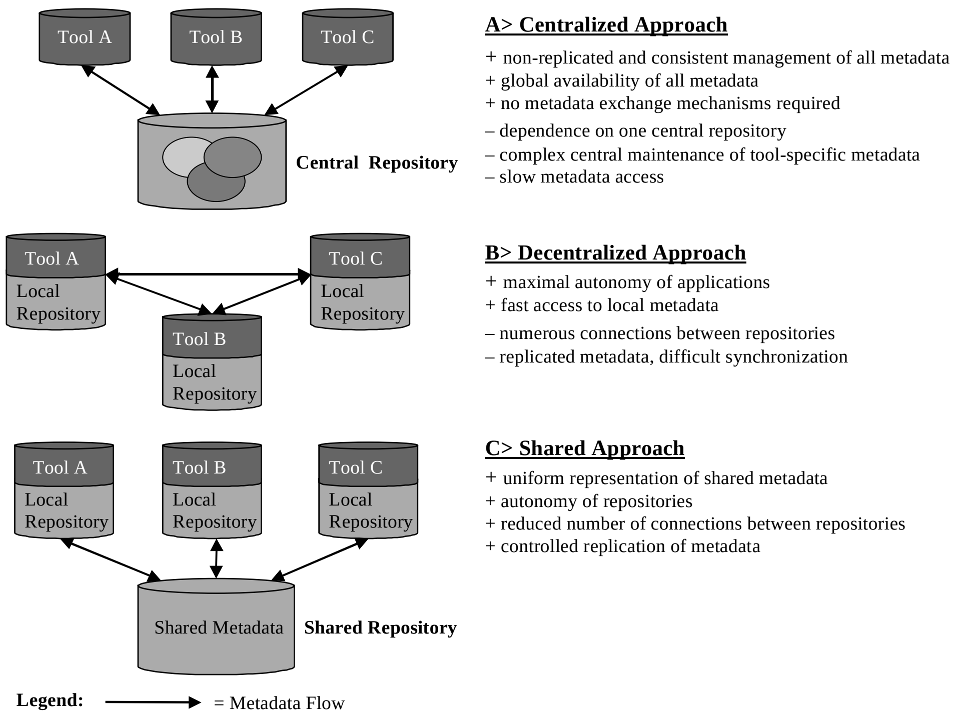
Tool A
Tool B
Tool C
Central Repository
A> Centralized Approach
+ non-replicated and consistent management of all metadata
+ global availability of all metadata
+ no metadata exchange mechanisms required
– dependence on one central repository
– complex central maintenance of tool-specific metadata
– slow metadata access
Tool A
Local Repository
Tool C
Local Repository
Tool B
Local Repository
B> Decentralized Approach
+ maximal autonomy of applications
+ fast access to local metadata
– numerous connections between repositories
– replicated metadata, difficult synchronization
Tool A
Local Repository
Tool B
Local Repository
Tool C
Local Repository
Shared Metadata
Shared Repository
C> Shared Approach
+ uniform representation of shared metadata
+ autonomy of repositories
+ reduced number of connections between repositories
+ controlled replication of metadata
Legend:
= Metadata Flow

Figure 4: Architectural approaches for metadata management

The possibility to uniformly represent metadata from multiple sources simplifies but does not solve the metadata integration problem. In particular, semantic heterogeneity has to be dealt with when designing the warehouse schemata and defining the required transformations.

4 Architectural Alternatives

We first discuss general architectures for shared metadata that may be represented according to one of the standard metadata models. Thereafter, we compare major alternatives for metadata interoperability, in particular exchange files and metadata APIs as well as the use of wrappers for metadata mapping.

4.1 General Architectures for Metadata Management

Regarding the way metadata is managed and shared between applications, we generalize three approaches for metadata management in a heterogeneous environment: centralized, decentralized, and shared or federated (Figure 4). In addition, a mixed approach is discussed.

Centralized Approach

In this case, all tools and the warehouse and data mart DBMSs directly access a central repository. They do not store and maintain metadata locally. The central repository is used to manage shared as well as tool/DBMS-specific metadata. The biggest advantage of this approach is that a non-replicated and consistent management of all metadata can be achieved. Every component has access to all existing and current metadata. However, this approach would require that all tools and DBMSs depend on the central repository even for specific metadata not relevant for other components. Such a loss of autonomy is typically unacceptable, in particular if components from multiple vendors need to cooperate. In addition, there is a performance problem because all metadata accesses would require interaction with the central repository.

Decentralized Approach

This approach represents the other extreme that is typical for current data warehouse environments (Section 1). All tools and DBMSs possess their own (local) metadata repository and communicate with each other to exchange metadata. This supports maximal autonomy and

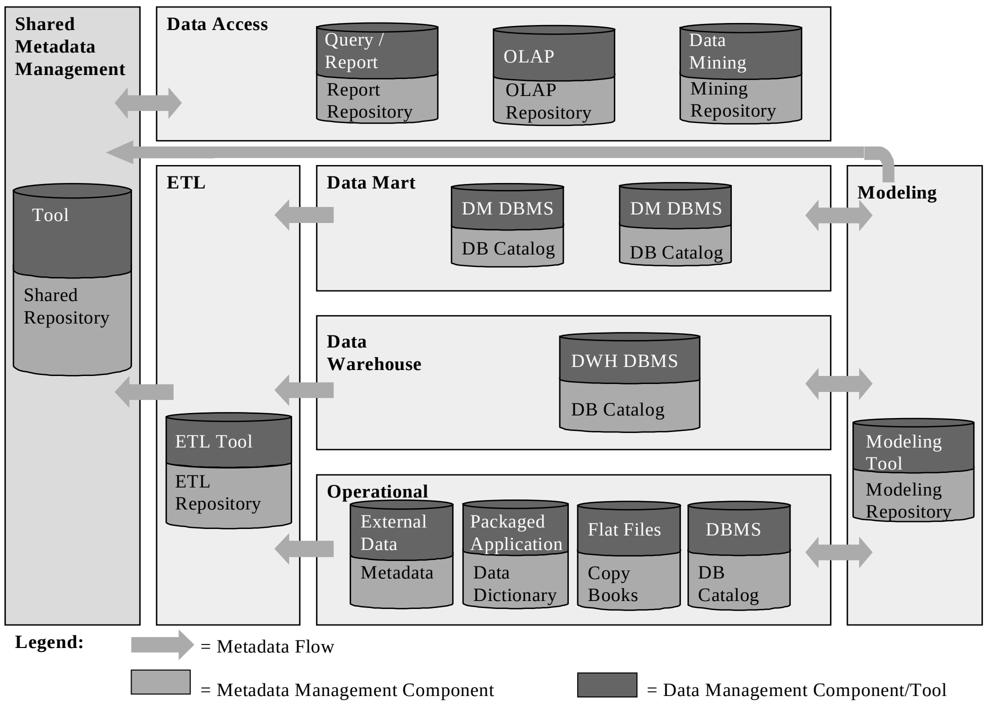

Figure 5: Mixed metadata architecture with a shared repository

performance for tool/DBMS-specific metadata. On the other hand, numerous bidirectional connections are needed to exchange shared metadata. Each connection may involve a complex mapping if metadata is differently represented in the local repositories. Furthermore, metadata is replicated in several components and difficult to keep current and consistent.

Shared Approach

This approach tries to combine the advantages of two previous approaches. Each tool/DBMS possesses its own repository for its local metadata thus supporting autonomy and fast access for this metadata. In addition, each component supports a metadata exchange interface to a common repository managing all shared metadata. While the metadata may be heterogeneously represented within the local repositories, a uniform representation is supported by the shared repository, e.g. based on a standard metadata model such as OIM or CWM. In contrast to the decentralized approach, the number of tool-to-tool connections and the mapping overhead can be significantly reduced and metadata replication can be tracked and controlled centrally.

The shared approach represents a *federated* metadata architecture preserving the autonomy and supporting heterogeneity of the participating repositories. Each repository decides which part of its metadata is to be exported, e.g., by defining a corresponding export schema. The shared repository unifies and combines these export schemata within a common metadata model thus improving its usability for other tools and end-users.

Mixed Approach

The aforementioned approaches can be combined within a mixed or hybrid metadata architecture. For instance, while the central approach seems to be of little relevance it can be utilized at the data access layer, e.g. when multiple client invocations of an access tool share the tool's local repository. On the other hand, the shared/federated approach seems to be the best choice but may not be achievable in its pure form in multi-vendor environments, e.g. if the operational systems or the data warehouse DBMS do not directly support a shared repository. Figure 5 shows such a mixed approach where a shared repository for managing globally relevant metadata is introduced, but only connected to some components. Compared to the original situation of Figure 1, we achieve a controlled flow of metadata from the ETL tool and the modeling tool to the data access tools, so that dependencies between metadata in those tools

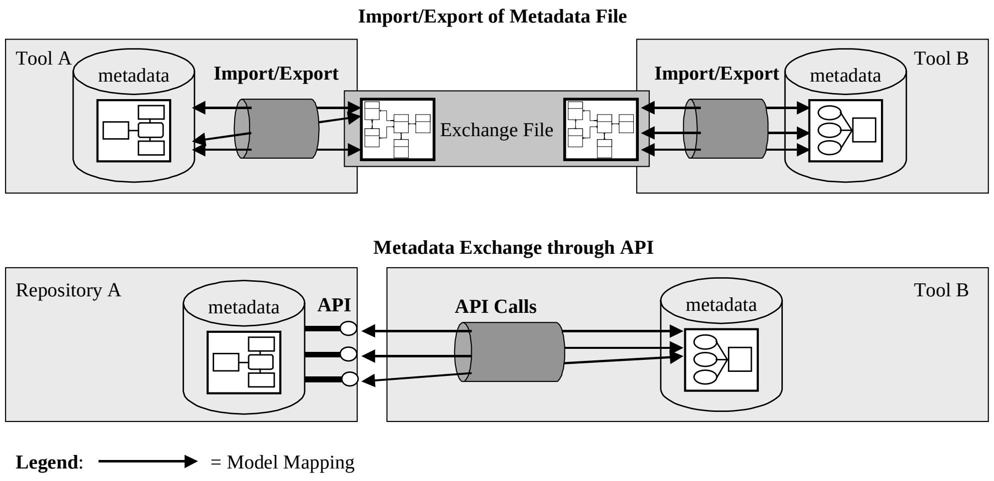

Figure 6: Metadata exchange methods as model mapping

can be recorded. The backflow of metadata from data access tools to the shared repository enables users to inform themselves about objects created during data analysis, e.g. queries, reports, or data cubes. Data marts, data warehouse and operational systems are not directly connected to the shared repository. However, for them, the modeling and ETL repositories represent shared repositories so that the architecture of Figure 5 can be seen as a multi-level repository federation.

4.2 Interoperability Mechanisms

Most of the discussed metadata architectures require the ability to communicate and exchange metadata between repositories. The exchange of metadata corresponds to a mapping process between two metadata models. Metadata exchange can more or less use the same methods than for data exchange between different data management systems. In particular, file exchange and API approaches can be used (Figure 6). In addition, a wrapper-based middleware approach can be employed to perform mappings between different metadata models. In order to provide more flexibility, some repositories implement several exchange methods at the same time.

File Exchange

The simplest method for metadata interoperability is to write metadata to a file of a specified (standardized) format that can be imported by other tools or repositories. Such exchange files are easily portable across multiple platforms. Furthermore, they can be produced and consumed even by tools and applications without a local repository. The file exchange method implies an *asynchronous* access to external metadata, actually to copies of it, which leads to two disadvantages. First, metadata is only current at the time of import. Second, the mechanism for translating between a tool's proprietary metadata format and the file format is hardcoded in the tool and needs to be adapted when the exchange format evolves.

Early specifications for metadata exchange formats include MDIS (Metadata Interchange Specification) [MDC97] and CDIF (Case Data Interchange Format) [EIA97]. MDIS was introduced by the Metadata Coalition in 1996. It allows the representation of different kinds of database schemata (relational, object-oriented, record-oriented, multidimensional) as well as inter-database relationships (textual descriptions of transformations and mappings between databases). MDIS has not been enhanced or extended since 1997. CDIF was specifically developed for transfering data models between CASE tools and is based on an integrated and extensible metadata model. CDIF is supported by CASE tools such as CA/ Platinum ERwin and Oracle Designer.

Currently, most promising appears the use of XML (Extensible Markup Language) [Ch99], a standard language to represent data on the Web. XML supports so-called Document Type

Definitions (DTD) to specify the structure of documents and other files. It is thus well-suited as the basis for standardizing the formats for exchanging data or metadata. With regard to data warehousing, XML has gained substantial importance as it is supported by the standard metadata models OIM and CWM (see Section 3) and commercial repositories (e.g., Microsoft Repository, Viasoft Rochade). CWM supports XMI (XML Metadata Interchange) for defining mappings of its metadata into XML; OIM provides an XML-based exchange format based on the former Microsoft specification XIF (XML Interchange Format). Currently, XML is not yet widely supported by modeling, ETL and data access tools but this is expected to change soon.

Repository API

A repository usually comes with a proprietary application programming interface (API) to its contents: metadata can be retrieved, inserted and manipulated. Repositories document their API and typically provide an application development environment. The API allows other repositories and tools to *synchronously* access and exchange metadata. This enables access to the current metadata of a repository. The API can also provide the means to extend and customize the metadata model of the repository. The disadvantage of this method is however the time and effort to be spent on the development of such applications.

Proprietary APIs are typically supported only by some other vendors. For instance, the Metadata Exchange (MX and its successor MX2) architecture of Informatica provides a set of COM-based APIs to the repository underlying its ETL tool PowerCenter/PowerMart [Inf98]. Tools of several vendors (e.g. Cognos, Business Objects) use these APIs to retrieve and push metadata from/into the repository. According to Informatica, MX2 is compatible with OIM. An MX2 software development environment is also included for developing custom applications.

Relational and object-oriented data APIs such as ODBC and OLE/DB also provide access to metadata and can thus be used for metadata exchange. For instance, the multidimensional API standard OLE/DB for OLAP includes commands to query and manipulate metadata describing the schemata of OLAP data sources.

Metadata Wrapper

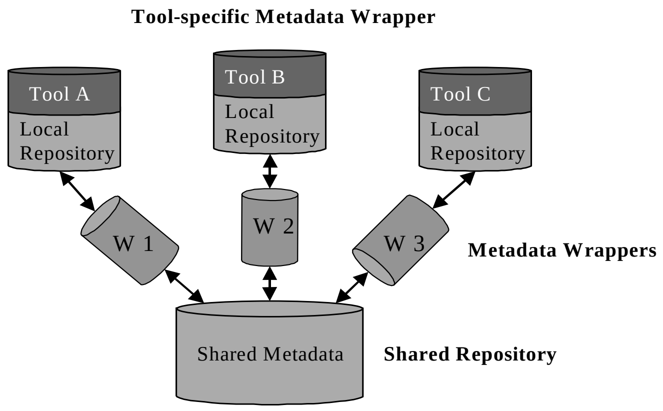

Figure 7: Metadata exchange through tool-specific *metadata*

In federated data architectures, wrappers are used to hide API differences of heterogeneous data sources [RS97]. A wrapper performs the mapping between the API of a data source and a common API that may be used by other middleware, such as a mediator providing uniform access to the underlying sources. Such an approach can also be used in the data warehouse context to provide uniform metadata access to heterogeneous repositories (Figure 7). In particular, it facilitates the integration of heterogeneous metadata within a shared repository. The metadata exchange between the source repositories and the wrapper and between wrapper and shared repository may be performed either API-based (synchronously) or asynchronously with exchange files. Typically, bidirectional wrappers are needed to both write metadata into and read metadata from the shared repository. The wrapper approach is followed by several commercial repositories (see Section 6).

5 Metadata Synchronization

While metadata changes less frequently than data, metadata updates are more difficult to deal with. This is because metadata updates not only affect the data that is described (e.g, deletion of an attribute) but also other metadata objects due to metadata interrelationships (e.g., referential integrity constraints). These problems already occur in a local (database) environment; specific research aspects have been addressed in the area of schema evolution [RR97, ALP91, KC88, DGN99]. In our context, we have to additionally synchronize repositories sharing metadata with each other. In particular, updates of replicated metadata need to be detected and propagated automatically in order to keep this metadata consistent. Furthermore, updated metadata needs to be applied (integrated) within a repository, e.g. to keep interrelationships with metadata from other repositories consistent. For example, changes in the data warehouse schema may cause errors in the execution of some ETL jobs, queries or reports when they are not adapted appropriately.

In the following we first discuss general approaches to these synchronization problems. We then turn to publish/subscribe approaches that may be used in combination with a shared repository.

5.1 Replication Control and Update Propagation

Numerous methods for replication control have been proposed in the area of distributed databases [ÖV99]. Most of them such as ROWA (read one, write any) or voting schemes strive for one copy serializability where all replicas are mutually consistent and each read is guaranteed to get access to the most recent transaction-consistent copy of a replicated object. Furthermore, each copy can typically both be read and updated. Unfortunately, such strict approaches are too restrictive and expensive to be widely applicable. In particular, they introduce substantial delays for synchronously updating multiple database copies (e.g., during commit processing) and require global concurrency control for strictly serializing updates. Furthermore, availability problems arise when some of the copies to be updated are not accessible due to system failures etc.

Such approaches are thus not appropriate for controlling replication of shared metadata. This is also because repositories are heterogeneous and largely autonomous and not ready to participate in a global concurrency control scheme for serializing updates. We therefore focus on „lazy“ synchronization approaches with weaker consistency goals [PSM98] which are also found in commercial DBMSs. Such strategies can be characterized by a publish/subscribe approach. A tool or repository changing (a copy of) shared metadata is called a „publisher“, a recipient of (a copy of) shared metadata is called a „subscriber“. Of course, a component may have both roles, publisher and subscriber, at the same time.

The following alternatives for metadata replication control can be differentiated:

- *„Update anywhere“ vs. „primary copy“*
  In the general case any copy of a replicated metadata object may be updated (n publishers per object), however without prior serialization. This requires an approach for merging concurrently performed updates and for detecting and resolving conflicting updates (e.g., duplicate identifiers). In commercial DBMSs (e.g., Microsoft SQL Server) conflict resolution for „update anywhere“ replication may be specified by application-specific routines or manually [Ham99]. This approach is quite flexible but the system loses control over maintaining consistency.
  These problems are avoided with only one publisher per object which is called the object’s owner or master. It maintains the primary copy of a replicated object that is

always up-to-date. The subscribers hold secondary or slave copies that are read-only and may not yet reflect all changes, depending on the update propagation strategy. We consider such an approach as sufficient for our purposes and thus assume that each shared metadata object has a primary repository where all updates are to be performed. This is a natural approach since most metadata objects originate from some repository - which will be the primary repository - from where they migrate to other repositories (subscribers).

- *Timing of update propagation: synchronous, ASAP, or deferred*
  Update propagation between a publisher and subscribers can be synchronous or asynchronous. Synchronous propagation occurs within the publishing transaction and requires a coordinated commit protocol between publisher and subscribers. This is an expensive approach and requires the repositories to participate in a standardized commit protocol, e.g., based on the XA interface [GR93]. An asynchronous propagation of updates occurs after commit of the publishing transaction and results in a delayed refresh of subscribed copies. We differentiate two variants: ASAP (as soon as possible) and deferred. With ASAP, copies are only slightly behind the primary copy but require every update to be propagated almost immediately. The deferred approach allows a less frequent (e.g., periodic) refresh with reduced communication overhead but resulting in less current subscriptions, similarly to snapshot replication [AL80].
- *Method for update propagation: notification (push) vs. probing (pull)*
  Two methods are differentiated depending on whether the publisher or subscribers are more active for update propagation. With notification, each publisher registers the set of metadata it shares with any subscriber. Updates are notified or pushed by the publisher to all affected subscribers either synchronously, ASAP or in a deferred way. The strategy requires that all participating repositories support a common interface for metadata registration and update notification (API-based interoperability).
  With probing (pull), on the other hand, each subscriber is responsible for keeping its imported metadata consistent. Periodically, it probes for new and changed metadata in other repositories and imports (pulls) the relevant updates. This implies a deferred propagation of updates. The probing approach can be implemented within the subscriber itself (API-based) or by an external scheduler invoking scan programs to check the relevant metadata sources (publishers). The latter approach is also applicable to a file-based metadata exchange. Probing permits obtaining metadata (updates) from repositories offering little or no support for replication control.

After a repository receives updated metadata the changes have to be applied and integrated with the existing metadata. In the simplest case, older versions of metadata objects are simply replaced. However, changes may also result in metadata conflicts that cannot be resolved automatically, e.g. due to dependencies to other metadata. Hence, a strategy for conflict consolidation needs to be provided. If the subscribing repository supports *versioning*, old metadata can remain and the updates are stored as a new version. The details of such a versioning depend on the supported granularity of metadata versioning (e.g., attribute, object, class, schema) and type of update.

A problem in this context lies in the fact that a given repository (subscriber) may obtain metadata updates from multiple sources (publishers) with different propagation methods and different timing approaches. As a result, even the most recent metadata objects can refer to different points in time making it difficult to group them together so as to obtain transaction-consistent versions of the metadata. The problem may be controlled by not independently propagating every metadata change but by only periodically performing such updates. For instance, modifications of source schemata may be propagated in batches, e.g. weekly or together with changes of the warehouse schema.

We have differentiated several alternatives for automatically propagating updates between repositories to refresh replicated metadata. Based on our discussion we favor a lazy publish/

subscribe replication control based on a primary copy approach for updates and a deferred update propagation either based on notification or probing. We now discuss how such an approach can be combined with the use of a shared repository.

5.2 Replication Control with a Shared Repository

The shared repository architecture can ease the task of metadata synchronization. It provides an almost up-to-date and possibly versioned set of all shared metadata and allows a centralized replication control for this shared metadata. A particular publish-and-subscribe approach (also called „hub-and-spoke“ [Ar99]) can be applied where the shared repository has both the role of a subscriber and publisher. Other publishers and subscribers access the shared repository to transfer/receive updates of shared metadata. The shared repository can register all publishers and subscribers together with the set of metadata they export or import.

For update propagation each of the methods discussed can be chosen. The main choice we left open was the distinction between a notification (pull) and probing (push) approach. These alternatives exist for two propagating steps: update propagation from a publishing repository to the shared repository (Step 1) and update propagation from the shared repository to subscribers (Step 2). This results in four possible combinations where each of the steps is either implemented by the shared repository or by the external publishers/subscribers (Table 3).

Combination	Implementation of Step 1	Implementation of Step 2
1: Push / Pull	Publishers	Subscribers
2: Push / Push	Publishers	Shared Repository
3: Pull / Pull	Shared Repository	Subscribers
4: Pull / Push	Shared Repository	Shared Repository

Table 3: Possible implementations of publish-and-subscribe

- *Combination 1 (Push/Pull):* The publishers push (propagate) their changes to the shared repository. The subscribers, on their own, detect the updates made in the shared repository and pull (import) them into their repository. This approach actually does not make active use of the shared repository for replication control: it only plays the role of a data management component.
- *Combination 2 (Push/Push):* The publishers push their changes to the shared repository which further pushs those changes to the subscribers. The shared repository registers the set of subscribed metadata.
- *Combination 3 (Pull/Pull):* The shared repository detects and pulls changes made by the publishers. The subscribers detect and pull updates made in the shared repository. The shared repository registers the set of each provider’s published metadata.
- *Combination 4 (Pull/Push):* The shared repository detects and pulls changes made in the publishers, and pushs them further to the subscribers. The sets of both published and subscribed metadata are maintained and tracked by the shared repository.

Depending on the combination, functionality for update propagation is to be implemented either in both publisher/subscriber repositories and the shared repository (2, 3), only in publisher/subscriber repositories (1), or only in the shared repository (4). Combination 1 is obviously the most inflexible approach because each tool and repository to be used must provide replication support. By contrast, combination 4 seems particularly promising for automating update propagation since it can be implemented entirely in the shared repository. However, both combinations 3 and 4 require the implementation of a comprehensive *change detection* which can be complex due to the heterogeneity of the repositories and tools.

Combination 2 is already realized in commercial repositories, e.g. Ardent MetaStage, although the propagation process is largely performed manually. Here, a tool can either directly publish its metadata into the central MetaStage repository or into a queue for approval by an authorized administrator. MetaStage tracks metadata subscriptions and notifies changes to the users of affected tools so that they can request an update from MetaStage.

6 Commercial Repositories for Shared Metadata Management

In this section we discuss several commercial repository solutions for data warehouse metadata management: Ardent MetaStage, IBM DataGuide, Microsoft Repository, Sybase Warehouse Control Center (WCC), and Viasoft Rochade. Based on the available information, we summarize in Table 4 major features of these products with respect to the metadata model, metadata interoperability, and other functions. Of course, only a snapshot can be provided (as of late 1999) as the products are constantly evolving.

Several repositories (DataGuide, MS Repository, Rochade) provide mechanisms to extend and customize their metadata model. Mostly, a proprietary metadata model is in use. MS Repository is based on the OIM standard metadata model; Viasoft and IBM have announced support for OIM and CWM, respectively. Typically, the repositories use a relational database for storing the metadata, except for Rochade which is based on a proprietary repository engine. With documented repository tables, metadata access and exchange can be performed using SQL.

All repositories implement import and export mechanisms for exchanging shared metadata. DataGuide and WCC use MDIS to import technical metadata from ETL tools such as Ardent DataStage and ETI Extract, and tool-specific wrappers to pass this metadata to data access tools such as Cognos Impromptu and Business Objects. DataGuide's wrappers are bidirectional allowing the import of metadata from several data access tools to describe queries, reports, and data cubes. Microsoft provides wrappers to import metadata from its own products, OLAP Services and English Query, into MS Repository. Rochade comes with two kinds of wrappers called buses and scanners. Buses support bidirectional metadata exchange and are available for various CASE tools (e.g. CoolGen, ERwin, ERStudio) and relational DBMSs. Scanners perform metadata extraction from programming languages (e.g. COBOL, JCL). Ardent's wrappers are called MetaBrokers and support bidirectional metadata exchange with MetaStage. Currently, several data access tools (e.g., Cognos Impromptu, Business Objects), modeling tools (e.g., ERwin, ERStudio), and Ardent's own ETL tool DataStage can be integrated with MetaStage by means of MetaBrokers. Despite such methods for metadata exchange, interoperability of a repository is generally limited to comparatively few tools because of the overhead of developing and maintaining metadata wrappers.

For metadata synchronization, MetaStage and WCC support the publish-and-subscribe paradigm. MetaStage is able to track subscriptions and notifies tool users when the subscribed metadata has been changed in the repository. WCC provides a facility to validate schema metadata against physical databases. MS Repository supports versioning of objects and their relationships, whereas Rochade provides versioning at the attribute level, the finest level of granularity.

Regarding end-user support, all repositories include proprietary or Web-based user interfaces for browsing, navigating and searching metadata. WCC supports „semantic views“ based on alias names for physical database tables and columns. In DataGuide, comments, contact and support information can be defined and attached to information objects. The searching facility is often based on pattern search in textual metadata. Except for DataGuide, the repositories lack a lexicon where business users can look up definitions of business terms, synonyms or other related terms.

Impact analysis informing end-users of dependencies among data is usually implemented through a query facility. MetaStage and Rochade provide query templates to speed up the construction of common queries such as for data lineage and impact analysis. Generated que-

Aspects			Ardent MetaStage	IBM DataGuide	Microsoft Repository	Sybase WCC	Viasoft Rochade
Metadata Model			proprietary	proprietary, extensible	OIM, extensible	proprietary	proprietary, extensible
Underlying DBMS			UniVerse	DB2	SQL Server	Adaptive IQ, SQL Server	proprietary
Exchange Mechanisms (in 2 directions)	File Format	Import	▨	IBM Tag Language, MDIS (ETI Extract)	XML/XIF	MDIS (DataStage), WAM (Warehouse Architect), ERX (ERWin)	XML/XIF
		Export	▨			▨	
	API	Import	MetaBroker Toolkit	proprietary (C)	proprietary (COM)	▨	proprietary (C++)
		Export				▨	
	Wrapper	Import	Cognos Impromptu, Business Objects, ERwin, ERStudio	Cognos Impromptu, Business Objects, CASE Tools	OLAP Services, English Query	▨	CASE tools, DBMS, Programming Languages
		Export		Cognos Impromptu, Business Objects	▨	Cognos Impromptu, Business Objects, English Wizard	CASE tools, DBMS
Metadata Synchronization			publish-subscribe	▨	▨	publish-subscribe	▨
	Versioning, Configuration		▨	▨	(COM) object level	▨	attribute level
User Interface	Browsing, Navigating, Search		Web	proprietary GUI + Web	proprietary GUI	proprietary GUI + Web	proprietary GUI + Web
	Impact Analysis		query builder	▨	view of ETL metadata	view of ETL metadata	query builder
Further Information			[Ar99] www.ardentsoftware.com	[IBM98a, IBM98b] www-4.ibm.com/software/data/vw	[BBC+99, BB99] msdn.microsoft.com/repository	[Ov98, Sy99] www.sybase.com/bid	[Vias97] www.viasoft.com

Table 4: Commercial repository approaches

ries can be stored for reuse or shared with other users. In MS Repository and WCC, a browsable view based on technical metadata defined by the ETL process is provided allowing to explore the relationships between source and target objects of an ETL job.

Most repositories do not yet support the integration of unstructured data and do not allow end-users to drill down from metadata to physical data. Although the users can see in the repositories which data exists in the data warehouse, typically they have to perform data access operations outside of the repository. DataGuide allows at least direct invocation of external programs for data access. Its metadata schema supports both structured and unstructured data. In particular, predefined object types are provided to describe database schemata as well as various types of documents, multimedia and internet files.

Little attention has been paid on user management and personalization of user views on metadata. Repository accesses are typically controlled by access rights defined on the underlying repository database. Users are usually not allowed to tailor and customize their user interfaces to obtain only the information they are actually interested in. A mechanism for automatic delivery and distribution of information to be shared among end-users is not achieved yet. Current implementations of publish-and-subscribe, as in MetaStage and WCC, provide metadata delivery to a limited number of warehouse tools and still largely depend on user interaction.

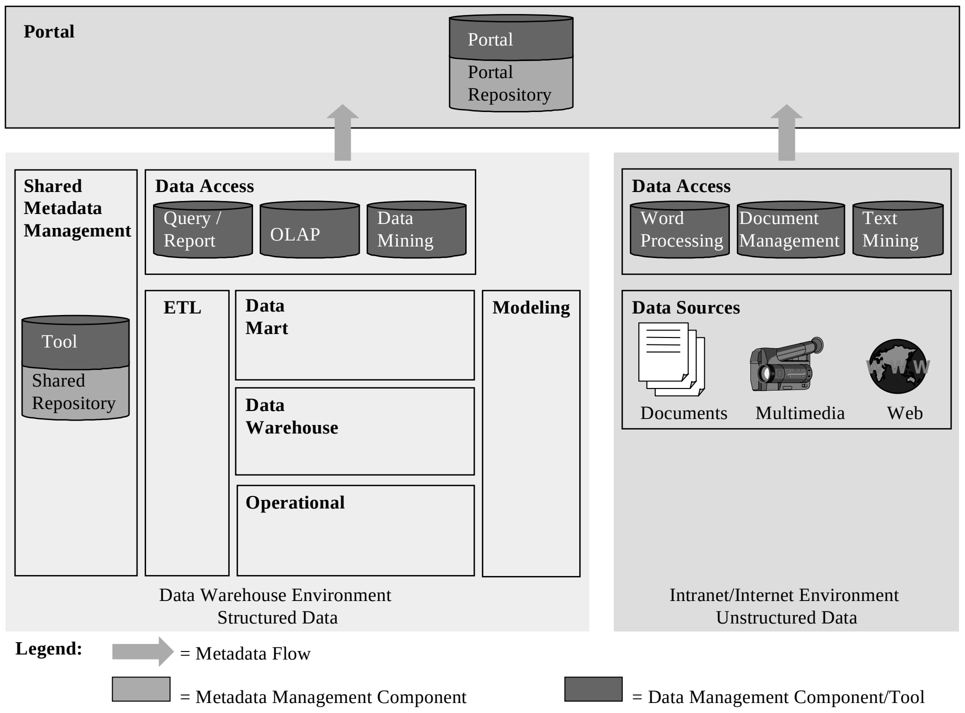

Figure 8: The information portal and relevant metadata flows

7 Information Portals

As pointed out above, current data warehouses and repositories do not sufficiently support business users. The situation can be improved by the proposed shared repository integrating technical and business metadata and providing the base for various data access tools and end-user interaction. However, business users also demand access to a variety of other information not maintained in the company's data warehouse. In particular, unstructured or semistructured data such as documents, news, Web pages, or multimedia objects need to be accessed in a simple and uniform way. The recently introduced concept of *enterprise information portals* (EIP) [Wh99, Fi99, Ha99] aims at meeting these requirements.

7.1 Extending the Data Warehouse by an Information Portal

As shown in Figure 8 such an information portal can be viewed as a common user interface to both structured business data, maintained in the data warehouse, and unstructured data possibly coming from external (Web) data sources. People also talk of combining „business intelligence“ with „knowledge management“. The focus of business intelligence is on applying algorithms, such as, querying, reporting, OLAP, and data mining, for processing structured data to identify trends and patterns that can be used to improve business strategies. Knowledge management, on the other hand, focuses more on the processing of unstructured data and the organization of such data into meaningful information that business users can search, modify, analyze and share. Characteristic techniques of knowledge management include indexing, categorization, content retrieval and distribution.

A key service provided by information portals is an individually customizable user interface for almost universal data access. Personalized view and delivery of data is based on the management of associations between users, user roles, subject areas of business data and single data elements within a portal repository. Users are assigned specific roles with corresponding rights for data access and to perform specific operations (e.g. query/report definition, execution, etc.). Furthermore, business users can specify sophisticated interest profiles by subscribing to single information objects or entire subject areas of interest and information sources (databases, files, Web pages, news channels, document stores, etc.) and specifying the frequency for their updates.

Delivery of subscribed data and thus distribution of business data is managed by a publish-and-subscribe engine. The portal maintains physical copies of the published objects at a central place (a DBMS or the Web server where the portal resides) or only descriptions of the objects with a link pointing to their location. Heterogeneous data requires different methods to access and visualize. A simple solution usually implemented by portal tools is to allow seamless invocation of external tools.

7.2 Coupling Portal and Warehouse Repositories

Available information portals, e.g. Viador E-Portal [Viad99], VIT SeeChain [VIT99], Hummingbird EIP [Hu99], Information Advantage Eureka [In99] use a repository typically built on a proprietary metadata model. Table 4 compares the typical content of a portal repository with a shared warehouse repository indicating that there are some overlaps but also different sets of metadata. The warehouse repository focuses on describing technical and (to some degree) business metadata describing structured data, while portal repositories cover both structured and unstructured data. Based on technical metadata gathered during the ETL process, the shared repository can inform business users about existing data dependencies and diverse aspects of data quality in the data warehouse. The main weakness of warehouse repositories is their lack of end-user support in data access and delivery. On the other hand, the information portal has more advanced support for user access rights and interest profiles. Using Web technologies, it facilitates the integration, categorization and access of unstructured data outside of the data warehouse. Some portals, e.g. Viador E-Portal, also has integrated facilities for performing querying, reporting and OLAP. However, information portals so far lack the capability to capture data dependencies and to express quality aspects of warehouse data.

Due to these differences in metadata scope and functions, the two repositories should be coupled with each other. The first question to clarify is how business metadata should be separated between the warehouse repository and the portal repository. A strict approach would be to keep all business metadata in the portal repository, for both the warehouse and other data sources. This would leave primarily technical metadata for the warehouse repository which could already be covered by the ETL repository. The other possibility would be to maintain warehouse-specific business metadata in a shared warehouse repository, thereby supporting business-oriented data analysis without portal and thus increased autonomy for the warehouse environment. In both cases, the mappings between the warehouse and portal metadata need to be maintained, e.g. replicated in both repositories. Metadata interoperability is needed to feed the information portal, e.g. with business metadata describing the data in the data warehouse for user navigation. The warehouse repository should be aware of new dependencies of warehouse data with the objects created in the portal. For interoperability and replication control the technical approaches discussed above can be applied, for instance an XML-based metadata exchange.

8 Summary

Currently, the metadata architecture of data warehouse environments is characterized by the co-existence of a multitude of largely autonomous and heterogeneously organized repositories. The use of different metadata models makes it difficult to exchange and integrate metadata in order to optimally perform data integration and make good use of the warehouse data, especially for business users.

We have provided a three-dimensional classification of metadata for data warehousing differentiating between various types of technical and business metadata. Current tools and repositories primarily manage technical metadata describing aspects such as the structure of data.

Aspects		Shared Warehouse Repository	Portal Repository
Technical Metadata	Data Sources	database schemata	database schemata, file formats, file location, URL
	Data Dependencies	operational <-> data warehouse, data marts <-> queries, reports, cubes	data warehouse, data marts <-> queries, reports, cubes
	Data Updating	scheduling of ETL jobs	scheduling of search, of report and cube update
Business Metadata	Conceptual Model, Business Terms	database conceptual models	subject areas of business data
	Data Quality	correctness (derived from ETL rules), currency (derived from scheduling of ETL jobs), completeness (derived from logging of ETL jobs)	currency (derived from scheduling)
	Business Description of Information Objects	queries, reports, cubes	queries, reports, cubes, documents, URL
	Personalization	user authorization	users, user groups, subject areas, information objects, associations among them
Functionality	Browsing, Navigating	yes	yes
	Content Search	no (search in metadata only)	yes (search in metadata and data)
	Metadata Synchronization	yes (metadata publish-subscribe)	no
	Impact Analysis	yes	no
	Data Access, Visualization	no	yes
	Personalized Data View and Delivery	no	yes (data publish-subscribe)

Table 5: Typical functionality of warehouse and portal repositories

However, end-users primarily need business metadata describing the meaning and usage context of data. We face two problems concerning the integration of technical and business metadata. First, while technical metadata is omnipresent, not every tool provides an open interface to its technical metadata. Second, unlike technical metadata, business metadata has first to be defined and associated with the corresponding technical metadata.

To better support metadata interoperability we have proposed a federated architecture utilizing a shared repository in addition to the tool- and DBMS-specific repositories. The shared repository integrates metadata from modeling and ETL tools. Furthermore, it can pass and import metadata to/from data access tools. In order to cope with a large spectrum of metadata, a generic repository with a comprehensive, extensible metadata model is required. OIM and CWM represent two remarkable approaches towards a standard metadata model for data warehouses. However, they still do not cover all kinds of relevant metadata and are not yet widely supported in commercial tools. Currently, vendors are forced to build a multitude of proprietary metadata wrappers to map metadata between different repositories.

As metadata replication is unavoidable in data warehouse environments, an automatic propagation of metadata updates is needed for replication control. We have discussed the major alternatives and propose the use of a lazy replication control with a deferred update propagation. Our federated architecture based on a shared repository facilitates such a replication control by naturally supporting a publish/subscribe approach. Particularly attractive seems an approach where the repository pulls metadata updates from publishers and pushs these updates to all relevant subscribers.

We have finally discussed the interoperability of data warehouses and information portals in order to provide business users integrated Web-based access to both structured and unstructured data. Such an approach requires interoperability of warehouse repositories and the portal repository and is subject to further study.

Acknowledgement: We would like to thank Phil Bernstein and Sergej Melnik for helpful comments.

References

AL80 Adiba, M., Lindsay, B.: *Database Snapshots*. Proc. 6th Intl. Conf. Very Large Data Bases (VLDB) 1980, 86-91.

ALP91 Andany, J.; Leonard, M.; Palisser, C.: *Management of Schema Evolution in Databases*. Proc. 17th Intl. Conf. Very Large Data Bases (VLDB) 1991, 161-170.

Ar99 Ardent Software: *MetaBroker Technology Overview and MetaStage Product Preview.* White Papers, January 1999.

BB99 Bernstein, P.A.; Bergstraesser, T.: *Meta-Data Support for Data Transformations Using Microsoft Repository.* IEEE Data Engineering Technical Bulletin, 22(1), 1999, 9-14.

BBC+99 Bernstein, P.A.; Bergstraesser, T.; Carlson, J.; Pal, S.; Sanders, P.; Shutt, D.: *Microsoft Repository Version 2 and the Open Information Model*. Information Systems, 24(2), 1999, 71-98.

BH98 Becker, J.; Holten, R.: *Conceptual Specification of Management Information Systems*. Wirtschaftsinformatik, 40(6), 1998, 483-492. In German.

Ch99 Chawathe, S.S.: *Describing and Manipulating XML Data*. IEEE Data Engineering Technical Bulletin, 22(3), 1999, 3-9.

De97 Devlin, B.: *Data Warehouse - From Architecture to Implementation*. Addison-Wesley Longman, 1997.

DGN99 Deruelle, L.; Goncalves, G.; Nicolas, J.C.: *Local and Federated Database Schema Evolution - An Impact Propagation Model*. Proc. 10th Intl. Conf. Database and Expert Systems Applications (DEXA) 1999, 902-911.

EIA97 Electronic Industries Association: *CDIF CASE Data Interchange Format - Overview.* http://www.eigroup.org/cdif/electronic-extracts/OV-extract.pdf.

Fi99 Firestone, J.M.: *Defining the Enterprise Information Portal*. White Paper, July 1999. http://www.dkms.com/EIPDEF.html.

GR93 Gray, J.; Reuter, A.: *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

Ha99 Hall, C.: *Enterprise Information Portals: Hot Air or Hot Technology.* Business Intelligence Advisor, 3(11), November 1999. http://www.cutter.com/bia/bia9911a.html.

Ham99 Hammond, B.: *Merge Replication in Microsoft's SQL Server 7.0*. Proc.Intl. Conf. Management of Data (SIGMOD) 1999, p. 527.

HST99 Härder, T.; Sauter, G.; Thomas, J.: *The Intrinsic Problems of Heterogeneity and an Approach to their Solution*. VLDB Journal, 8(1), 1999, 25-43.

Hu99 Hummingbird: *The Hummingbird EIP.* White Paper, December 1999. http://www.hummingbird.com/whites/index.html.

IBM98a IBM: *Managing Data Guides 5.2.* 1998. ftp://ftp.software.ibm.com/ps/products/visualwarehouse/info/vw52/DGMANAGE.PDF.

IBM98b IBM: *Metadata Management for Business Intelligence Solutions*. White Paper, 1998. http://www-4.ibm.com/software/data/pubs/papers.

In98 Inmon, W.H.: *Enterprise Metadata*. DM Review, November 1998. http://www.dmreview.com/master.cfm?NavID=55&EdID=298.

In99 Information Advantage: *MyEureka – Business Intelligence Portal*. White Paper, February 1999.

Inf98 Informatica: *Informatica Announces General Availability of First Object-Based API for Metadata Exchange*. Press Release, March 1999. http://www.metadataexchange.com.

JJQ+99 Jarke, M.; Jeusfeld, M.A.; Quix, C.; Vassiliadis, P.: *Architecture and Quality in Data Warehouses: an Extended Repository Approach*. Information Systems, 24(3), 1999, 229-253.

KC88 Kim, W.; Chou, H.T.: *Versions of Schema for Object-Oriented Databases*. Proc. 14th Intl. Conf. Very Large Data Bases (VLDB) 1988, 148-159.

LJ99	Lehmann, P.; Jaszewski, J.: *Business Terms as a Critical Success Factor for Data Warehousing*. Proc. Workshop Design and Management of Data Warehouses (DMDW) 1999.
MA97	Murray, D.; Anahory, S.: *Data Warehousing in the Real World - A Practical Guide for Building Decision Support Systems*. Addison Wesley Longman, 1997.
Ma99	Marco, D.: *Metadata Moves Mainstream*. Microsoft White Paper, January 1999. http://www.microsoft.com/SQL/bizsol/metadata.htm.
MDC97	Metadata Coalition: *Metadata Interchange Specification (MDIS) – Version 1.1*. August 1997. http://www.mdcinfo.com/MDIS/MDIS11.html.
MDC99	Metadata Coalition: *Open Information Model – Version 1.1*. Proposal, August 1999. http://www.mdcinfo.com/OIM/MDCOIM11.html.
MSR99	Müller, R.; Stöhr, T.; Rahm, E.: *An Integrative and Uniform Model for Metadata Management in Data Warehousing Environments*. Proc. Workshop Design and Management of Data Warehouses (DMDW) 1999. http://dol.uni-leipzig.de/pub/1999-22.
OMG99	Object Management Group: *Common Warehouse Metamodel (CWM) Specification*. Revised Submission, February 2000. http://www.omg.org/techprocess/meetings/schedule/CWMI_RFP.html.
Ov98	Ovum Ltd.: *Ovum Evaluates – Data Warehouse Tools and Strategies*. 1998.
ÖV99	Özsu, M.T., Valduriez, P.: *Principles of Distributed Database Systems*. Prentice Hall, 2nd ed., 1999
PSM98	Pacitti, E., Simon, E., Melo, R.: *Improving Data Freshness in Lazy Master Schemes*. Proc. 17th Intl. Conf. Distributed Computing Systems (ICDCS), 1998, 164-171.
RR97	Ra, Y.G.; Rudensteiner, E.: *A Transparent Schema Evolution System Based on Object-Oriented View Technology*. IEEE Transactions on Knowledge and Data Engineering, 9(4), 1997, 600-623.
RS97	Roth, M.T.; Schwarz, P.: *Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources*. Proceedings of 23rd Intl. Conf. Very Large Data Bases (VLDB) 1997, 266-275.
SL90	Sheth, A.P.; Larson, J.A.: *Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases*. Computing Surveys 22(3), 1990, 183-236.
SVV99	Staudt, M.; Vaduva, A.; Vetterli, T.: *The Role of Metadata for Data Warehousing*. Techn.Report 99.06., Dep. of Information Technology, Univ. of Zurich, September 1999.
Sy99	Sybase: *Warehouse Studio – A Foundation for Data Warehouse Delivery*. White Paper, 1999. http://www.sybase.com/bid/ws_whitepaper.pdf.
Viad99	Viador: *Viador E-Portal Suite*. Corporate Brochure, 1999. http://www.viador.com/pdfs/CB02_5.pdf.
Vias97	Viasoft: *The Rochade Repository Environment – The Foundation for the Information Asset Warehouse*. White Paper, 1997. http://www.viasoft.com/download_src/white_papers/IAW.pdf.
VIT99	VIT: *VIT SeeChain Portal – An Information Portal for the Enterprise*. White Paper, August 1999. http://www.vit.com/white_papers/dbaipaper.html.
Wh99	White, C.: *Using Information Portals in the Enterprise*. DM Review, April 1999. http://www.dmreview.com/master.cfm?NavID=198&EdID=61.
Wi98	Wieken, J.-H.: *Metadata for Data Marts and Data Warehouses*. The Data Warehouse Concept, Gabler, Wiesbaden, 1998, 275-315. In German.