Loren Heilig, Steffen Karch, Oliver Böttcher, Christiane Hofmann, Roland Pfennig

SAP NetWeaver™ Master Data Management

Galileo Press
Bonn • Boston
Contents at a Glance

1 Introduction ............................................................ 15

2 Master Data and Application Systems in Companies .............................................................. 19

3 From Silo to Services—MDM as a Central Component of a Service-Oriented Architecture ..... 45

4 Overview of MDM .................................................. 73

5 MDM as a Basis for Identity Management ............ 97

6 MDM in the Publishing Industry ......................... 133

7 MDM in an Enterprise Characterized by Mergers and Acquisitions ................................. 161

8 MDM: Technical Details ........................................ 183

9 Summary and Outlook ............................................ 307

A List of Acronyms .................................................. 317

B Literature ............................................................. 321

C The Authors ............................................................. 323
Contents

Acknowledgements ................................................................. 13

1 Introduction ........................................................................ 15

2 Master Data and Application Systems in Companies ........ 19
  2.1 What Is Master Data? ...................................................... 19
  2.2 Master Data in Enterprise IT .......................................... 22
  2.3 Examples of Problems with Master Data ....................... 28
    2.3.1 Example 1: Consumer Products ......................... 28
    2.3.2 Example 2: International Automobile Manufacturer .......... 30
    2.3.3 Example 3: Logistics Service Provider ............... 31
    2.3.4 Example 4: Flow Accounting .......................... 32
    2.3.5 More Real-Life Examples ................................ 34
  2.4 Solutions for Master Data Management ....................... 35
    2.4.1 Manual Solutions ........................................... 35
    2.4.2 The ALE Mechanism in SAP R/3 ..................... 35
    2.4.3 Modern Master Data Management ................ 37
  2.5 Today’s Requirements for Master Data Management .......... 39
  2.6 Conclusion ................................................................. 42

3 From Silo to Services—MDM as a Central Component of a Service-Oriented Architecture ...... 45
  3.1 Initial Situation at Many Companies ......................... 45
  3.2 Basic Principles of Enterprise SOA .......................... 48
    3.2.1 The Core/Context Model ................................ 52
    3.2.2 Benefits of Enterprise SOA .......................... 54
    3.2.3 Defining Features of an Enterprise SOA ........... 58
    3.2.4 SOA or Enterprise SOA ................................ 60
  3.3 A Platform for Enterprise SOA: SAP NetWeaver ........... 64
    3.3.1 Components of SAP NetWeaver .................... 64
    3.3.2 IT Practices and Scenarios .......................... 66
  3.4 MDM as the Backbone of Enterprise SOA ................. 68
    3.4.1 MDM Scenarios .......................................... 69
    3.4.2 Eliminating Functional Redundancies and Central Master Data Storage .......... 70
4 Overview of MDM .................................................... 73
   4.1 MDM Architecture .............................................. 74
   4.1.1 MDM as a Central Component of SAP NetWeaver .......... 74
   4.2 Overview of MDM Components ................................ 75
   4.2.1 SAP MDM Server ............................................. 76
   4.2.2 SAP MDM Console .......................................... 77
   4.2.3 SAP MDM Data Manager .................................... 78
   4.2.4 SAP MDM Import Manager/Server ....................... 80
   4.2.5 SAP MDM Syndicator .................................... 81
   4.2.6 MDM Java API ............................................. 82
   4.2.7 Workflows .................................................. 82
   4.2.8 SAP MDM Image Manager .................................. 82
   4.2.9 SAP MDM Publisher ...................................... 83
   4.3 Available MDM Scenarios ....................................... 83
   4.3.1 Rich Product Content Management (RPCM) ............... 84
   4.3.2 Global Data Synchronization (GDS) ..................... 87
   4.3.3 Customer Data Integration (CDI) ....................... 90
   4.3.4 ISV Partner Scenarios .................................... 93
   4.4 SAP NetWeaver MDM as a Toolbox .......................... 93
5 MDM as a Basis for Identity Management .................. 97
   5.1 Company and IT Landscape ................................... 99
   5.2 Scenario Description .......................................... 102
   5.2.1 User View Controls Authorizations ..................... 103
   5.2.2 Ideal World of Authorizations ........................... 105
   5.2.3 Requirements of Identity Management .................. 106
   5.3 Chosen Solution and Its Architecture ...................... 109
   5.3.1 Data Extraction ........................................... 110
   5.3.2 Data Import ................................................ 111
   5.3.3 Data Retention/MDM Repository .......................... 111
   5.3.4 Data Maintenance Processes .............................. 111
   5.3.5 Analyses .................................................... 112
   5.3.6 Unfulfilled Requirements ................................. 112
   5.4 Implementation ............................................... 112
   5.4.1 Creating the Business Blueprint ....................... 113
5.4.2 Creating the Data Model ..................................... 113
5.4.3 Extending the Data Model with Technical Parameters ..................................................... 119
5.4.4 Identifying and Implementing the Maintenance Processes ................................................... 119
5.4.5 Identifying and Connecting the Data Sources ................................................................. 126
5.4.6 Creating the Data Distribution ....................... 128
5.4.7 Creating the Reporting System ....................... 130
5.5 Summary and Outlook ................................................ 131

6 MDM in the Publishing Industry ........................................... 133
6.1 Description of the Problem ........................................ 134
6.1.1 The Publishing Industry .................................. 134
6.1.2 Significance of the Customer in the Publishing Industry ................................................... 135
6.2 The Enterprise and the IT Landscape ........................................ 135
6.2.1 ERP System (R/3) ........................................... 136
6.2.2 Customer Relationship Management (CRM) System ................................................ 137
6.2.3 SAP Business Information Warehouse (SAP BW) .................................................. 137
6.2.4 SAP Exchange Infrastructure (XI) .................... 137
6.2.5 SAP Enterprise Portal (EP) .............................. 138
6.2.6 Master Data Management (MDM) System .... 138
6.3 Scenario Description ................................................... 139
6.3.1 Central Questions .......................................... 139
6.3.2 Process Flow in Master Data Harmonization and Analysis ............................................. 140
6.4 Chosen Solution and Its Architecture ........................................ 141
6.4.1 Process Flow and Data Flow ......................... 142
6.4.2 Components in Detail .................................... 145
6.5 Implementation .......................................................... 147
6.5.1 Data Modeling of the Repository ................... 148
6.5.2 Defining Repository Properties ...................... 153
6.5.3 Formulating the R/3 Extraction Routine .......... 154
6.5.4 Import to MDM ............................................. 154
6.5.5 Export from MDM and Distribution via XI ..... 155
6.5.6 BW Analysis ................................................... 156
6.5.7 Representation in the SAP Enterprise Portal ... 158
6.6 Lessons Learned and a Look Ahead ..................... 159
## 7 MDM in an Enterprise Characterized by Mergers and Acquisitions

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Description of the Problem</td>
<td>162</td>
</tr>
<tr>
<td>7.2</td>
<td>The Enterprise and the IT Landscape</td>
<td>162</td>
</tr>
<tr>
<td>7.3</td>
<td>Scenario Description</td>
<td>167</td>
</tr>
<tr>
<td>7.4</td>
<td>Chosen Solution and Its Architecture</td>
<td>172</td>
</tr>
<tr>
<td>7.5</td>
<td>Procedure</td>
<td>175</td>
</tr>
<tr>
<td>7.5.1</td>
<td>Designing the Data Model</td>
<td>175</td>
</tr>
<tr>
<td>7.5.2</td>
<td>Implementing the MDM Repository</td>
<td>177</td>
</tr>
<tr>
<td>7.5.3</td>
<td>Creating the Catalog (Import)</td>
<td>178</td>
</tr>
<tr>
<td>7.5.4</td>
<td>Setting Up the OCI Connection</td>
<td>179</td>
</tr>
<tr>
<td>7.5.5</td>
<td>Implementing the Web Services</td>
<td>179</td>
</tr>
<tr>
<td>7.6</td>
<td>Lessons Learned and Future Prospects</td>
<td>180</td>
</tr>
</tbody>
</table>

## 8 MDM: Technical Details

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Architecture in Detail</td>
<td>183</td>
</tr>
<tr>
<td>8.1.1</td>
<td>MDM Server</td>
<td>184</td>
</tr>
<tr>
<td>8.1.2</td>
<td>System Requirements</td>
<td>184</td>
</tr>
<tr>
<td>8.1.3</td>
<td>Master/Slave Principle</td>
<td>185</td>
</tr>
<tr>
<td>8.1.4</td>
<td>Management Console CLIX</td>
<td>187</td>
</tr>
<tr>
<td>8.1.5</td>
<td>Backup Strategy</td>
<td>188</td>
</tr>
<tr>
<td>8.2</td>
<td>MDM Data Model</td>
<td>192</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Table Types</td>
<td>192</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Field Types and Options</td>
<td>204</td>
</tr>
<tr>
<td>8.3</td>
<td>MDM Console</td>
<td>208</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Structure of the User Interface</td>
<td>209</td>
</tr>
<tr>
<td>8.3.2</td>
<td>Administration</td>
<td>210</td>
</tr>
<tr>
<td>8.3.3</td>
<td>Special Features</td>
<td>213</td>
</tr>
<tr>
<td>8.4</td>
<td>MDM Data Manager</td>
<td>216</td>
</tr>
<tr>
<td>8.4.1</td>
<td>General Structure</td>
<td>216</td>
</tr>
<tr>
<td>8.4.2</td>
<td>The Functions of the MDM Data Manager</td>
<td>218</td>
</tr>
<tr>
<td>8.5</td>
<td>MDM Import Manager/Server</td>
<td>226</td>
</tr>
<tr>
<td>8.5.1</td>
<td>Architecture of the Import Manager</td>
<td>227</td>
</tr>
<tr>
<td>8.5.2</td>
<td>User Interface of the Import Manager</td>
<td>228</td>
</tr>
<tr>
<td>8.5.3</td>
<td>Port Concept</td>
<td>229</td>
</tr>
<tr>
<td>8.5.4</td>
<td>From Decentralized Data to the Import Concept</td>
<td>231</td>
</tr>
<tr>
<td>8.5.5</td>
<td>Import Options</td>
<td>233</td>
</tr>
<tr>
<td>8.5.6</td>
<td>Importing with the Import Manager and Import Server</td>
<td>243</td>
</tr>
</tbody>
</table>
8.6 MDM Syndicator/Server ............................................. 246
8.6.1 Structure of the Export Interface .................... 246
8.6.2 Export Formats .............................................. 248
8.6.3 Example: Exporting a Repository .................... 249
8.7 MDM Publisher .......................................................... 256
8.7.1 Structure of the User Interface ....................... 257
8.7.2 The Print-and-Publish Process ...................... 258
8.8 Integration of SAP Components with MDM
   Business Content .................................................. 263
   8.8.1 Business Content for MDM ............................ 263
   8.8.2 ERP Integration .......................................... 268
   8.8.3 BI Integration ............................................ 275
8.9 Integration into the Portal .................................. 280
   8.9.1 Connection between MDM and Portal .......... 281
   8.9.2 iView Types for MDM Systems .................. 283
   8.9.3 Page Layout ............................................. 285
   8.9.4 Integrating MDM Workflows into the Portal .. 286
   8.9.5 Permissions and Roles .............................. 286
8.10 Workflows .......................................................... 288
   8.10.1 Workflow in the SAP MDM Data Manager .... 288
   8.10.2 CAF Guided Procedures ........................... 290
   8.10.3 MDM in Guided Procedures ..................... 294
8.11 MDM Programming Interfaces .................................... 295
   8.11.1 Explanation of the Various APIs ............. 295
   8.11.2 Connection to the Web AS ............................. 295
   8.11.3 Java API ................................................. 297
   8.11.4 COM API .................................................. 303
   8.11.5 ABAP API ................................................ 305
8.12 Conclusion ................................................................. 306

9 Summary and Outlook ............................................. 307

   9.1 Why Master Data Management? ......................... 307
   9.2 MDM as the Foundation of Enterprise SOA .......... 309
   9.3 Enterprise SOA and MDM in Practice ................ 311
      9.3.1 MDM as a Basis for Standalone
             Applications .......................................... 311
      9.3.2 MDM as Harmonization and Consolidation
             Tool ...................................................... 311
      9.3.3 MDM as Central Master Data Management .... 312
   9.4 A Look Ahead .................................................. 313
Acknowledgements

“The team is the star.” This motto not only held true at the World Cup 2006 in Germany, but it is also a sentiment that we, the authors, became reacquainted with when adopting the concept of an SAP NetWeaver Master Data Management (MDM) compendium in the very writing of this book. It was Steffen Karch and Loren Heilig who initially brainstormed about how this could be done. Still, it took an entire team of distinguished authors, aided by countless individual contributors (working both in the background and in the foreground), to turn these ideas into reality—an achievement of which we are all very proud. Some key players on this winning team deserve a special mention; however, we don’t want to forget anyone, and so our heartfelt thanks go to all of those who contributed to this book.

No book can come into being without a publisher, and once again we received all the support we could wish for from SAP PRESS. First, we must thank Eva Tripp, who only took charge of the project shortly before its conclusion, but who nonetheless provided some very helpful comments and positive feedback. Next, our thanks must also go to Florian Zimnia, who provided invaluable support up to that point, as well as during the work on our SAP NetWeaver book.

At SAP, we wish to thank first and foremost the MDM Product Management team, and in particular Christian Behre, Andreas Seifried, Tim Goetz, Michael Theis, and Michael Reil, who provided us with reliable and up-to-date information, and without whom this book could not have attained its present level of quality.

At IBSolution, special thanks are due to the MDM team, who implemented the scenarios in a live environment, and whose considerable expert input into Chapter 8 ensured just the right level of technical detail. The efforts of Robert Herbert and Andreas Markert are particularly noteworthy.

We would also like to thank Oliver Donner, Andreas Hardt, and Stefan Wagner for their help with Chapter 7. Finally, special thanks
Acknowledgements

go to Gabriela Karch, who, for the third time, has created some excellent graphics for an SAP PRESS publication. Once again, her work has greatly enhanced the quality of the book.

Ludwigsburg, Germany, March 2007

Oliver Böttcher, Loren Heilig, Christiane Hofmann, Steffen Karch, and Roland Pfennig
To be able to use SAP NetWeaver MDM properly, you must understand the technical details and connections between the applications. This chapter, which can be used as an MDM compendium, will provide you with this very knowledge.

8 MDM: Technical Details

This chapter will provide you with a more detailed description of all the functions available in SAP NetWeaver Master Data Management (MDM) 5.5, including the content provided by SAP for enterprise resource planning (ERP) integration. The first two sections describe the MDM architecture and the data modeling entities available. Next, the MDM core components (i.e., MDM Console and the MDM Data Manager) are re-introduced (see Chapter 4 where they were first described). Then, the MDM Import Manager and MDM Syndicator for Export (Sections 8.5 and 8.6) are covered. Section 8.7 is devoted to special output using MDM Publisher. Section 8.8 covers the overall SAP integration provided in the context of Business Content. Sections 8.9 and 8.10 then look at integration into SAP Enterprise Portal and the various workflow options. In the last section, we describe the extensive programming interface, which provides all the previously covered functions for direct call from programs.

8.1 Architecture in Detail

The individual technical components of the MDM Server are discussed in more detail below, including a few basic technical settings for security and backup. There is also an explanation of how MDM can be set up for very high performance using the master/slave principle.
8.1.1 MDM Server

The MDM Server is based on a database in which the repositories are stored. When a repository is loaded in the MDM Console, its entire contents are available in the main memory of the MDM Server. This architecture allows read access to MDM to be faster than the normal SQL database management system by a factor of 100 with comparable amounts of data. However, the MDM Server is not an application on the Web Application Server (Web AS), but rather a standalone application with its own installation. Nevertheless, the MDM Server and the Web AS can use the same database.

The MDM Server can be installed on the following database and operating systems:

- Windows 2000 and 2003 Server with Oracle, MS SQL Server, or DB2
- HP-UX, AIX, Linux and Solaris with Oracle or DB2

8.1.2 System Requirements

A decisive factor in the high-performance operation of software is the sufficient storage capacity of the main memory. This is of particular significance when running an MDM system, since the MDM Server always keeps the data in a repository in main memory. From a performance standpoint, this delivers significant advantages, especially since read operations can run many times faster due to quicker access to the electronic memory. Operations and manipulations can then run in real time.

For the memory requirements of a system, it generally holds true that when you have more than an 80% load on the memory, data starts to be swapped out to the hard drive, resulting in a heavy load on both the CPU and the main memory. This means that the performance win due to memory-oriented data storage is lost. Another factor that influences performance is the number of repositories that are currently loaded, in which additional metadata must be loaded and managed.

---

1 More details on the versions available can be found in the current Product Availability matrix at service.sap.com/pam • NetWeaver • MDM 5.5.
To get an impression of the size of data storage within the MDM Server, we recommend that you start with benchmarks. For about one million records with rich content (for example, image or data attachments), main memory requirements of about 2.5 GB are indicated. This assumes that the records are stored directly in the repository, along with their attachments. If this is limited to just records without attachments, this memory size should be sufficient for about six million entries.

Criteria for the performance of the database server include the speed of the hard drive and the connection speed to the MDM Server. To avoid overlapping performance bottlenecks regarding main memory size and hard drive capacity, the MDM Server and the database server should be operated on separate systems.

To ensure both high performance and good data reliability, we recommend that you operate the database server with a RAID-5 system. This accelerates access, particularly read access, due to an even block distribution of the data over all the drives. At the same time, it uses parallelism to reduce the probability that two write operations take place at the same time on the same drive.

### 8.1.3 Master/Slave Principle

To further improve read performance of the MDM Server, you can use the master/slave principle. Here, multiple synchronized copies of the masters, the so-called slaves, are created. These slaves can be used as equivalent replacements of the master for read access. All write accesses, on the other hand, continue to take place on the master. If the master repository’s server has sufficient hardware resources, the slaves can be created on the same server. Otherwise, distributions onto other servers are possible and recommended.

In a master/slave architecture, repositories can be managed using either scale-in or scale-out clustering. Scale-in clustering is the management of multiple master or slave repositories on a single server. Figure 8.1 clarifies the architecture described with the distribution of slaves onto different servers, which is called scale-out clustering.
The master manages the database assigned to it and to which it has exclusive read and write privileges. The existence of the slaves connected (e.g., two production master repositories with their slaves) is not known to the master. The slave stores the assignment to the master, but the reverse is not the case. The query for data synchronization must therefore be started by the slave. This can be managed using the MDM Console or the CLIX tool (see Section 8.1.4), which can run the query under scheduled control.

The two slaves can each have their own database, or they can share access to a common database; however, at least one database independent of the master is required. The databases supporting the slaves can be different from that of the master, that is, different versions or even databases from different manufacturers can be used.

The advantages of this kind of scale-out architecture lie primarily in the better stability of the MDM Server due to the distribution of load over multiple servers. Moreover, the master and slave(s) can run on different operating systems. On the other hand, due to the separate
standalone servers, this architecture requires more resources, and possibly even more administrative overhead, as well as higher costs for procurement and maintenance.

In contrast to scale-out, the scale-in approach assumes a single physical server, which manages both the master repository and also the associated slaves. To make this possible, however, each repository must be assigned its own port.

The behavior of scale-in is analogous to scale-out clustering. Differences lie primarily in the advantages and disadvantages. The scale-in approach requires fewer resources relative to the number of MDM Servers, but cannot take as much load and requires extensive port configuration. Moreover, this approach poses a greater risk in the event of a server outage, since the associated repositories can no longer be accessed.

An existing slave repository can be “normalized” as needed. Normalization removes the master/slave relationship, so that the slave becomes a new, independent master repository. Correspondingly, this new master also has its own database, to which it has write permissions. The connection to the previous master is completely deleted, so this process is not reversible.

### 8.1.4 Management Console CLIX

The MDM Console Command Line Interface to MDM (CLIX) can be used to manage MDM and its repositories. Unlike the console, CLIX is independent of the operating system, and can therefore run just as well under Linux and Solaris. CLIX provides special commands, which can be executed through the CLIX command line. These commands can also be scheduled using scripts. The available commands enable you to perform most administrative tasks remotely.

Among the administrative tasks supported by CLIX are, for example, the starting and stopping of the MDM Server, the loading and unloading of repositories, and the mounting\(^2\) of individual repositories, for example. Activity reports providing information about the current status of the repository can also be extracted and written to files for storage or further processing. Design functionality like the

\(^2\) *Mounting* means making repositories available in the MDM Console.
changing or addition of repositories, tables, and fields are not supported by CLIX.

CLIX provides a special command syntax, which can be broken down into four groups. There are individual commands for the control of the MDM Server, for management, and for the copying of repositories, along with commands related to the underlying database management system.

The general command syntax has the following structure:

CLIX Command_Name [Arguments] [Control flags]

A command to control the MDM Console might look like this:

CLIX mdsStatus MDMHostSpec [-T seconds] [-D]

This command provides general information about the repositories currently loaded in the MDM Server. Additional examples, summaries of commands, and an overview of the optional flags are listed in the SAP MDM Console Reference Guide in the SAP Service Marketplace.\(^3\)

CLIX may not provide the full functionality of the console, but CLIX also offers certain functionality, which goes beyond the console’s scope. Functions can be used which the console doesn’t provide, particularly regarding MDM server commands, for instance, to parameters in the MDM server configuration file mds.ini.

### 8.1.5 Backup Strategy

In order to be able to fall back on the last data saved in case of data loss, the backup of data and systems is a part of the reliability aspects of the MDM Server. The idea is to keep the delta between the current data and the backups as small as possible.

In addition to the backup options of the underlying database system, the MDM Server provides its own functionality to extend the range of possibilities for creating a mature backup strategy. These options can be controlled directly from the MDM console.

---

\(^3\) You can find more information at [http://service.sap.com/instguides • NetWeaver • Release 04 • Installation • SAP MDM](http://service.sap.com/instguides • NetWeaver • Release 04 • Installation • SAP MDM).
There are basically three different options for backup of an MDM Server:

- Archival of a repository
- Creation of a slave to copy the master
- Duplication of a repository

An archived repository can be reloaded onto any arbitrary MDM Server of the same release level and the data will be restored, regardless of the underlying database system. This is even possible if the database system has been upgraded to a newer release or replaced by a system from a different manufacturer.

Archival is controlled from the MDM Console and can be started easily, even while a repository is running. All the data connected with the repository is also archived, that is, the actual data inventory, the repository structure, and also the import and syndication maps. Archived repositories are not stored in the database, but are placed directly onto the MDM Server as a file. There are other functions available, like the segmentation of the file size. The files to be archived can be distributed across different media if necessary. The size of the archived files will depend on the structure and data inventory of the repository.

The master/slave architecture is another backup option of the MDM Server. The slave repository has an archived data inventory until the next synchronization, which can be actively accessed as needed. If the system environment is distributed, the data inventory is distributed onto multiple servers, increasing the outage resistance of the data. But even if the master/slave principle is used within a single server, a clear separation of the repositories can be established (since changes to the master have no effect on the slave). For performance reasons, a slave can initially be synchronized with its associated master, which runs on a different server. Then, the archival can proceed, working from the updated slave. Delta synchronization, in particular, uses fewer resources than an archival, which helps performance.

Backup using duplication of repositories works in similarly to a master/slave configuration, with the main difference being that the data inventory is not actively available. The “inactive” copy of a repository should be stored on a different server for better protection against hardware outages.
Using the command line interface to MDM, CLIX, the possibilities described can also be coded into simple scripts. These scripts can be scheduled and performed automatically, for example, by executing a Windows task.

The MDM Server has other options for increasing data safety that go beyond direct backup of the database. The archival and duplication of repositories, in particular, offer advantages over a pure database backup, since they can be created independently from the underlying database system. In the event of a complete database crash, the archived repositories can also be loaded into a different database.

An additional backup of the database is still recommended, since the MDM backup (alone) is a cost- and time-effective extension to your overall data reliability and backup strategy. By using both variants, even in the event of a system crash, the database and the MDM data can be restored quickly.

Best practice is a nightly backup of the repositories (with offline storage) and a weekly backup of the database.

**Security**

The MDM Console, as the central administration and design element of the repositories of an MDM server, must be protected from unwanted access and manipulation. The MDM Console itself has no intrinsic security mechanism for access control. Thus anyone with access to the MDM Console also can connect to all associated MDM servers and their repositories. There, they can perform any manipulations, even a complete deletion of repositories.

The MDM Server can actually be password-protected, but a simple access permission to its installation directory makes this security mechanism superfluous. If you have access to the MDM Server’s configuration file `xcs.ini`, removing the attribute value in `XCS Scone` simply deletes the password. The password protection of the server has nothing to do with connections to the MDM Console. Thus access to the console means that after the MDM Server is mounted, the password can simply be changed with a graphical interface. This makes configured connections, for example, from the SAP Enterprise Portal to the MDM system, impossible. Only the startup of the MDM Server is password-protected.
Access to the MDM Console and to the MDM Server installation directory must therefore be protected by monitoring and securing access privileges on the operating-system level. User management must be used to assign access permissions at the directory level only to authorized users (generally administrators of the MDM Server), primarily to protect the central configuration file `xcs.ini` from unauthorized access. Furthermore, you must ensure that access to the MDM Console is also limited at the operating-system level.

It is therefore astounding that the CLIX tool is not capable of accessing a password-protected MDM Server. The possibility of providing a login with the corresponding password is simply not included.

From Service Pack 04 (SP04) on, MDM 5.5 provides an LDAP interface (Lightweight Directory Access Protocol),4 which enables you to store user information in a central directory. For example, user information that can be queried by MDM is stored in the LDAP-capable directory. This delegates the maintenance of that information to the LDAP service. The connection to MDM is secured using Secure Sockets Layer (SSL) or Kerberos, which ensures a secure, uniform authentication on a non-secure network.

To use the LDAP service, two basic settings must be made. First, you must activate the LDAP service in the `xcs.ini` file, and save the associated connection settings. Security configuration can also be done here. In the directory service, MDM needs only one attribute field, which contains the specified role names from MDM User Management separated with semicolons.

The interplay of MDM and the LDAP directory can be described as follows. When the user logs in to the MDM client (Data Manager), this connects to the MDM Server and passes the entries to it. Secured by SSL, the MDM Server connects via LDAP to the directory service and searches for the login name (the “distinguishedName”). This login name is found and sent back to the MDM Server, which then connects to the LDAP service again and passes the login information (including the password) provided by the user. Now, the permissions (MDM roles) are returned and compared to the rights in the repository of the role(s) requested for access.

4 The LDAP interface can, for instance, be operated in connection with Microsoft Active Directory, Novell eDirectory, or OpenLDAP.
8.2 MDM Data Model

At the start of every master data project is the data model. Which master data attributes must be managed centrally and how they should be managed, and which properties are available for tables and their fields are the basis for the modeling of the business environment. This section describes the MDM data model and explains how it can be adapted to implement the technical design as closely as possible.

8.2.1 Table Types

An MDM repository has multiple types of tables, which will be discussed below. Figure 8.2 shows an overview of the possible table types.

The possibilities discussed in this chapter for data modeling in MDM are semi-based on the information provided by SAP in the SAP MDM Console Reference Guide.
Main Table and Subtables

The standard tables include the main table and its subtables, that is, the tables related to it. These standard tables contain the actual master data. References from the main table to individual subtables through lookup fields, that is, reference fields to other tables that connect the main table record and the subtable record, can be used to model even complex data models in MDM.

For main tables and subtables, the following four table types exist in MDM:

- Flat tables
- Hierarchy tables
- Taxonomy tables
- Qualified tables

Flat tables are the most common table, which you should be familiar with if you’ve worked with database management systems. Flat tables contain rows and columns. A main table is always of type flat.

Hierarchy tables can display hierarchies in master data, such as product hierarchies, which are displayed in a tree menu. One example is an org chart, with branching between areas and departments. Taxonomies, on the other hand, are used to categorize or classify master data into coherent groups according to defined attributes, like product groups. For instance, a category “bottles” might automatically contain subcategories “1 liter bottles,” “1.5 liter bottles,” and so on, which can be used to form a uniform expression of the “volume” attribute.

Qualified tables are used to show different variants in the relationships between main tables and subtables in a simple, highly efficient way. This makes sense because individual values in the qualified table—depending on the relationship between the main table entry and the subtable entry—may be different for every main table entry. An example might be customer or supplier conditions, which have different pricing schemes for different quantities. To avoid having to make an entry in a relation table for every one of these variants, these fields can be defined as qualifiers. Therefore, the contents of the qualifier field is not related to the record in the subtable of which it is a part, but to the relationship of this record to the main table.
Relations between tables

Lookup fields need not occur only in the main table. Subtables can also reference other subtables. Just as in the main table, multiple lookup fields may be included. The only exceptions are Taxonomy and Qualified lookup tables, which can only be referenced from the main table.

These nested structure capabilities are particularly interesting when using search options in the Data Manager. Every lookup field can be used to further restrict an existing search using filter values in the lookup fields (see also Section 8.4).

Object Tables

It can be practical to attach file objects to a record, for instance, a PDF with a description or an image file for a product master record. In MDM, there are special object tables that store objects separately by file type. The advantage of these external object tables is that an object can be referenced in multiple places.

To date, the following object types are supported:

- Image files
- Text blocks
- HTML text
- PDF files

Example

- Customer Mayer pays a price of 5 USD per item for quantities between 1 and 100 items. From 101 to 200 items, he pays only 3.50 USD.
- Customer Smith pays 5.50 USD for 1 to 100 items and 4 USD from 101 to 200.

The product table stores all the product information, including these quantity scales. But where should the pricing information be stored?

In a relational database, another table would have to be built in which a custom price is stored for every customer and every quantity. In MDM, thanks to qualified tables, that's not necessary. The qualifier (here, the price) is not just related to the record of the table in which it is stored, but also to the main table record to which it is connected.

This not only “saves” a table, it keeps the product table manageable, because there continues to be only one record for every product, in which multiple prices can be stored for each customer.
Sound files
Video files
Binary files

This is interesting for applications such as product descriptions. In the product master record itself, no PDF is stored, just a lookup to the PDF object table in which the product description is stored. Thus every file is loaded into the repository just once, and can be managed centrally.

All objects except text blocks and HTML text, moreover, can reside on the fileserver and are referred to using path specifications. They do not have to be stored in the MDM repository. The administration of the files, however, takes place via MDM. Some of these files (e.g., images) can even be edited directly through MDM.

Object tables are predefined by SAP and are delivered in their required comprehensive structure. The fields are fixed and cannot be changed. For an overview of the structures in the various object tables, see the SAP MDM Console Reference Guide, located in the SAP Service Marketplace.

Special Tables

Special tables are used to represent additional repository and structure information. They are created automatically when a repository is created. The special tables include:

- Masks
- Families
- Image variants
- Relationships
- Workflows
- Data groups
- Validation groups

Masks are used to form subsets of a repository. This subset acts as a separate repository (for example, a certain product group within a product repository), with all its tables and structures, but it is not a
duplicate of the records. Thus there continues to be a single unique, central master record, which ensures its integrity.

Unlike a saved query, a mask is stored statically in the repository like a kind of snapshot of the set of records that are in the mask, so new records cannot be included dynamically in the mask based on their defined attributes.

The topic of masks is especially significant when examining permissions, which they support. Permissions concepts will be described in greater detail in the next section.

**Families**

*Families* can be used to group together records, which have the same value or the same attribute in certain fields. The families table contains one record for each family. So attributes valid for all family members can be assigned directly to the family and need not be maintained individually for each family member.

This means that instead of having to store group-specific information for every single record, the main table can be more manageable. This group-specific information (for example, for product categories) is stored in the families table and therefore applies to every family member.

The families table must also contain the lookup field in the main table, which is used to define family membership. This stored lookup field must be of type *taxonomy* or *hierarchy*. This allows the family records to be generated automatically.

**Image variants**

The images object table, which we already mentioned, can be used to attach images to a master record. Depending on the use of the master record, these images may have to comply with different criteria. For example, for a catalog generated from the MDM system, it is important to know whether it will be printed or published on the web. Depending on the medium, a different resolution of the product pictures will be needed. The *image variants* table is used for such purposes. Variants of image objects can be stored in a series of pre-defined fields. The processing options are, for instance, the image size, the resolution, the file type, trimming, the compression used, the color space, watermarks, and much more, some of which is specific to the file type.
Relationships between master records can be represented using the *relationships* table. These relationships show the relations between master records and how the objects they represent are related to one another. These relationships can be of type parent/child or sibling. Relationships can also exist between records in multiple tables, as long as they are of the types main, flat, hierarchy, or taxonomy.

The meaning of the relationship can differ according to the record used. A parent/child relationship between records in the main table, for example, for products, can indicate that the parent products are of a better quality and provide suggestions for up-selling. A parent/child relationship between a record from the main table and one from a subtable, on the other hand, might be used for parts of the products or accessories that are packaged along with the product.

Contrary to a hierarchy or taxonomy, relationships are completely free-form. They are not dependent on a certain value in a certain field, or on whether a relationship exists between two records; instead, they emerge solely from arbitrary business logic.

Relationships between multiple tables can be only single-level. Relationships within the same table, on the other hand, can be multi-level, that is, we can branch from child to grandchild to great-grandchild and so on.

Sibling relationships, on the other hand, are best suited for relationships between records of the same level (in the main table), for example, as ideas for the potential of cross-selling, or as a product alternative with similar performance characteristics. Alternative contacts (i.e., persons who are responsible for a certain topic) might also be given using sibling relationships in an employee management system.

The *workflow* table stores information on the workflows existing for this repository, for example, to which table the workflow refers. A reference to a workflow modeled in Visio is also stored, which defines how a certain group of records should be handled. For more information about workflow, see Section 8.10.

The *data groups* table is created automatically and is transparent to the user. This is where data groups are managed in which objects are organized in the MDM system.
Validation groups

Like the data groups table, the validation groups table is also transparent to the user. Validations are formulas, which can be executed on fields or attributes. These validations can be summarized in groups called validation groups. See Section 8.4 for more information.

System Tables

The system tables are used for data protection and administration of the repository. This applies to the following access-relevant tables:

- Roles
- Users
- Logins

System tables also include the following descriptive tables:

- Change tracking
- Client systems
- Ports
- URLs
- XML schemas
- Reports
- Logs

The system tables are located under the Admin node in the tree menu, thus are visually separate from the other tables as well.

Roles

Roles in MDM are a very powerful and flexible tool for the definition of permissions at the field level and the functional level. At the functional level, there are the permissions to insert, edit, delete, protect, or remove the protection from a record, or to group multiple records together. Check in/check out functions for the locking and unlocking of a record, and rollback and join permissions can also be assigned.

There are other table-specific functions, which can also be created only for this table type with selective permissions. For example, for an image object table, you can determine whether a role owner may rotate or clip an image. In taxonomy tables, there are a series of functions for editing attributes, which can be released or blocked individually.
But not only functions for data manipulation in the repository are covered here. Import and export permissions and the modification of the associated maps can also be specified (more on maps in Sections 8.5 and 8.6).

At the field level, you can determine for every table or—if detailed differentiation is needed in a table—for every field, whether a role should have read or write permissions.

To limit the records that can be displayed for a role, you can use masks, which act as virtual repositories. They show only a subset of the records in the limited table and provide them to only that role owner. Alternatively, you can also limit the records to be displayed by using a discrete value in a lookup table, or you can use a combination of these two aforementioned techniques. Thus the role owner sees only those records that correspond to a specific lookup value.

It therefore follows that the permissions of a role result from the combination of three factors:

- Permission to execute the function selected
- Permission to change the selected table
- Permission to change the selected record

The users table is used to store information about the people who should have access to the repository, along with their roles (a user can also have multiple roles). A user with multiple roles has permissions that are appropriate to the combination of those roles.

For every repository, there is an Admin role created with all permissions and an Admin user that can be changed at any time. To date, there is no integration with the role configuration of other SAP systems; all other users must be added manually.

Passwords for the users (not in clear text) and their email addresses are also entered here. The email addresses are required for workflow notifications in order to inform all participants of required workflow steps.

In the logins table, there is an overview of which users are currently accessing the repository and which client applications they are using to access it. The times of the system logins and the last activity performed are also shown.
Thus not only can access to a repository be monitored, but the administrator also has an overview of which users must be informed before the repository is unloaded for maintenance.

Change tracking The change tracking table is used to store changes to the data inventory of the repository. It can be individually specified for each field, whether the new value, the old value, or both should be stored in the change tracking table. Alternatively, change tracking can be deactivated for this value. For each entry in the change tracking table, the data and time, along with the user making the change, are recorded.

Client systems Client systems are the central tool in the MDM system, allowing the harmonization of master data from multiple systems throughout the enterprise along with individually specific configuration. For every system containing master data, a representative client system is created in MDM. This client system stores system-specific information describing the role of the system in the consolidation and harmonization of master data.

This information must also include whether a system is only a supplier of data (inbound), whether it is only a receiver (outbound), or both. If master records from MDM must be newly created in a client system during export because they didn't exist in the system, the client system table can be used to store the numeric range in which the MDM system may generate new keys.

These keys can also be in a qualified range. This means that MDM administers multiple numeric ranges for a client system, whose use depends on the properties of an attribute. For instance, if customer numbers in an application are dependent on the region, then the region of the customer to be created is checked during export from the MDM system, and a customer number from the appropriate numeric range is generated.

Client systems are essential for key mapping. Here, for each globally valid MDM key, the primary keys for the same record are stored in the various client systems.
If a record is created in a system where it hasn’t yet existed, a new entry is automatically created in the key mapping. This client system is also created there with the newly generated key. But even for different formats and data structures in these client systems, this procedure is essential. There can only be error-free transmission if the exported data is adapted from the data model of the MDM system to that of the receiving client system.

**Ports** are used to manage the connection between MDM and the client systems. They specify for each client system how imports and exports should be handled. For import, this means that it is stored in the appropriate port which file in what path should be imported with what import map and which XML schema for what client system. The same applies to export.

Ports are thus the prerequisite for being able to automate both imports and exports of master data into and out of the MDM system; otherwise, every import or export would have to be started manually.
A port can either be created for inbound or outbound. For a client system, which is used for both inbound and outbound, that is, where data is both imported and exported, two ports must be created.

**XML schemas**

XML schemas are intended for use in the Import Manager or Syndicator, that is, for import or export from the MDM system. This table stores the name of the schema and its storage location on the files server.

Schemas are required when the import or export should take place using ports. For every port, there must be an associated XML schema specified. For more information, see Sections 8.5 and 8.6.

The dependencies of client systems, ports, and XML schemas are shown in Figure 8.4.

---

**Figure 8.4** Relationships Between Client Systems, Ports, and XML Schemas

**URLs**

URLs can either be relative to certain records (or several), or to the attributes of taxonomy. The URL can also consist partially of placeholders. These placeholders are automatically replaced during the populating of the repository with specific attributes of each record, thereby forming a complete URL.
For various operations on the MDM Console, for example, update, copying, or archival of a repository, automatic reports are generated, which are stored in this table. These reports document all individual steps taken by MDM during the operation, and can be checked. A report is always relative to a special repository.

The logs table is used to store all the log files for the current server. For that reason, the logs table is the table that is relative not just to a certain repository, but to the entire server. These log files document all operations that are specific to the server, such as the loading of a repository.

The difference between reports and logs is shown in Figure 8.5.

**Example**

A URL can be created for product master data, which points to a Web-based product catalog. The ID of the product is given as a placeholder so that the URL can be individually adapted for every product and point to the direct address for the product.

Therefore, as a URL the following is stored: http://www.companyhomepage.net/productcatalog.php?product_id=[placeholder for ID]. This placeholder is then automatically replaced with the product ID for every record.

For various operations on the MDM Console, for example, update, copying, or archival of a repository, automatic reports are generated, which are stored in this table. These reports document all individual steps taken by MDM during the operation, and can be checked. A report is always relative to a special repository.

The logs table is used to store all the log files for the current server. For that reason, the logs table is the table that is relative not just to a certain repository, but to the entire server. These log files document all operations that are specific to the server, such as the loading of a repository.

The difference between reports and logs is shown in Figure 8.5.

**Figure 8.5 Reports and Logs**
8.2.2 Field Types and Options

Field types specify which value ranges (for example, purely numeric, integer, alphanumeric values, etc.) may be used by a field. The variety of field types and their properties can already model a large amount of logic, which sets an MDM repository apart from a conventional database. These field properties impose a series of limitations on creation and processing, which can already be semi-covered by master data specifics.

Data Types for Fields

MDM supports the field types listed in Table 8.1.

<table>
<thead>
<tr>
<th>Field Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Text field with less than 4,000 characters</td>
</tr>
<tr>
<td>Text Large</td>
<td>Text field with more than 4,000 characters</td>
</tr>
<tr>
<td>Text Normalized</td>
<td>Text field from which all non-alphanumeric characters should be removed. Can be used, for instance, for better searching.</td>
</tr>
<tr>
<td>Name</td>
<td>Text field with a structure appropriate to names (e.g., first name, middle name, last name)</td>
</tr>
<tr>
<td>Integer</td>
<td>Whole number, 4 bytes in length</td>
</tr>
<tr>
<td>Real</td>
<td>Floating point number, 4 bytes in length</td>
</tr>
<tr>
<td>Real8</td>
<td>Floating point number, 8 bytes in length</td>
</tr>
<tr>
<td>Boolean</td>
<td>Yes/no field</td>
</tr>
<tr>
<td>Log</td>
<td>Field of type Text Large with predefined structure for multiple blocks with timestamps</td>
</tr>
<tr>
<td>AutoID</td>
<td>Field of type Integer, which automatically counts up by one</td>
</tr>
<tr>
<td>Currency</td>
<td>Field of type Real8, which includes a freely definable currency symbol</td>
</tr>
<tr>
<td>GM Time</td>
<td>Field of type Time Stamp, which is relative to a special time zone</td>
</tr>
<tr>
<td>Measurement</td>
<td>Field of type Real to which a measurement unit is attached. MDM currently supports more than 750 units of measurement and allows simple conversion from one unit to another. For user-defined units of measurement beyond the standard ones, the application MDM Unit of Measure Manager (UOM) can be used.</td>
</tr>
</tbody>
</table>

Table 8.1 Field Data Types
Display Fields

A *display* field is the field in a table, which is used to represent the entire record. In a lookup table, for example, in the selection list for the lookup field, the values of all display fields are shown, from which you can select the record you want.

**Example**

In an address management system, the country could be a lookup field to avoid duplicates, ensure proper spelling, and provide error-free filtering. The lookup table Country would contain the abbreviations of the countries, along with their full names. When a new address is entered, a dropdown list appears in the Country field. Whether the abbreviation or the full name is displayed depends on which of those two fields was defined as a display field. The value of the display field also determines the names of nodes in hierarchies or taxonomies.

For each table, there can be multiple display fields defined, if one field alone is not unique or unclear. However, at least one field must be defined. However, there are a few exceptions: Special tables and object tables (except for masks) have no display fields. For hierarchies or taxonomies, the automatically generated field *Name* (of type *Text*) is a display field, which cannot be changed. This is because sib-

<table>
<thead>
<tr>
<th>Field Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literal Date</td>
<td>Date field</td>
</tr>
<tr>
<td>Literal Time</td>
<td>Time field</td>
</tr>
<tr>
<td>Create Stamp</td>
<td>Field of type <em>Time Stamp</em>, which is automatically populated with the current date and current time upon creation of a record</td>
</tr>
<tr>
<td>Time Stamp</td>
<td>Date and timestamp field, which is automatically updated on each change to a record</td>
</tr>
<tr>
<td>User Stamp</td>
<td>Field in which the code of the user who changed a record is stored</td>
</tr>
<tr>
<td>Mask</td>
<td>Field in which a subset of the master records is stored. This field is not visible; it is used only for searches.</td>
</tr>
<tr>
<td>Lookup</td>
<td>For table types Flat, Hierarchy, Taxonomy, Qualified, Image, Text Block, Text HTML, PDF, Sound, Video, and Binary Object</td>
</tr>
</tbody>
</table>

Table 8.1 Field Data Types (cont.)
ling names must be unique. For qualified tables, at least one of the display fields must be a non-qualifier.

**Unique Fields**

If a field is a unique field, then the value entered must be unique in this table. This can also be a combination of multiple values. That means that the corresponding value or the corresponding combination of values can occur only once in the table.

**Example**

A customer number can be assigned only once and should therefore be defined as a unique field. For the banking information, on the other hand, the combination of account number and bank number must be unique, and not just the individual fields themselves. In this case, the combination of the two fields would be defined as unique.

If not otherwise defined, a unique field does not need to be filled, that is, it can be left empty. The value “empty” (or NULL) is therefore not subject to the uniqueness constraint, since it may occur in multiple records.

**Multivalue**

A particular feature of the MDM data model—in contrast to the relational database model—is the possibility of defining fields to be multivalue-capable. That means that this field can be assigned multiple separate values. This applies particularly to lookup fields, in which multiple values from the lookup table can be stored. In relational databases, such an m:n relation must be represented with a third relation table.

Besides the various lookup fields, the field type Measurements can also be defined as multivalue.

**Additional Field Properties**

In addition to the field properties already addressed, there are a few other modeling possibilities, the most important of which are displayed in Table 8.2.
<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>If this field attribute is selected, this field must always be filled in for every record.</td>
</tr>
<tr>
<td>Writeable</td>
<td>If this attribute is selected, the field can no longer be changed after it is first assigned, and it is set to read-only.</td>
</tr>
<tr>
<td>Multilingual</td>
<td>Fields can be specified as multilingual and then store different values for each of the repository languages released. For every repository, the languages selected can always be changed. An example of a practical multilingual field is the title, which might take on both Herr and Mr. in a record that supports both German and English. The display is associated with the language selected when the repository is called (when calling the repository from clients, a login language is specified.)</td>
</tr>
<tr>
<td>Keyword</td>
<td>This attribute determines whether the field should be included in keyword searches (more on keywords in Section 8.3.3).</td>
</tr>
<tr>
<td>Selected Fields</td>
<td>In this attribute, it is determined whether the field is tracked by creating timestamps or user stamps. Any change to this field then results in an update of the stamps.</td>
</tr>
<tr>
<td>Symbol</td>
<td>If this field is a currency specification, this is where it is determined which currency symbol will be used.</td>
</tr>
<tr>
<td>Dimension</td>
<td>If this field is a dimension, it must be determined which dimension it is. Examples of dimensions are size, weight, surface, volume, etc.</td>
</tr>
<tr>
<td>Default Unit</td>
<td>Which unit is used for a value depends on the dimension. For instance, if weight is specified as the dimension, then grams, kilograms, pounds, tons, etc. are available here, but not the units of other dimensions like liters or meters. One of the possible units can be selected here.</td>
</tr>
<tr>
<td>Decimal Places</td>
<td>Here is where it is specified how many decimal places the values in this field have.</td>
</tr>
<tr>
<td>Lookup Table</td>
<td>If the field is of type Lookup, then here is where the lookup table involved is specified.</td>
</tr>
<tr>
<td>Default</td>
<td>For some values, preconfigured values can be set. Examples are boolean values, which are prepopulated with true or false when created in order to avoid NULL values. Similarly, date or time fields can automatically be set to the current date or current time. These are not timestamps, however, since they can be overwritten at any time.</td>
</tr>
<tr>
<td>Width</td>
<td>This attribute determines the maximum number of characters that the value in the field may contain.</td>
</tr>
</tbody>
</table>

Table 8.2 Field Properties
8.3 MDM Console

The console is the administration and data modeling tool in MDM. Here is where the structure of the repository and all its properties are determined and managed.

During processing in the console, a repository must be blocked for all other client applications, so that structure and content are not processed at the same time. The process of blocking is called unload and takes place in the console. Afterwards, the unloaded repository is no longer callable from other client applications.

All MDM clients retrieve repository information dynamically. So when the structure has changed in a repository and it is then loaded, the new structure is automatically available to the clients as well. Portal iViews, however, are static and must be adapted after structural changes.

If users are logged into a repository upon unloading, the console user is warned before unloading, however, this warning can be ignored and unloading will then proceed.

To unload (and thus edit) a repository, the server on which the repository is stored must first be mounted. For every MDM system,
multiple servers may be used. (For more information on the starting and mounting of MDM servers, see Section 8.3.3).

Which servers were started when the console started can be saved into a configuration file. This file can be automatically executed when the console starts, so that these servers no longer need to be started manually.

From the console, there is no thin client for connection to the portal (see also Section 8.9). The rich client must be installed locally for the desired user group, which is not a problem, since use of the console as an administrative tool should only be made available to certain qualified users. In SP04, no login to the system is necessary to use this tool—although MDM Servers can be password-protected—which makes selective installation even more important.

### 8.3.1 Structure of the User Interface

The user interface of the console is divided into three main parts:

- **Console Hierarchy Pane**
  The *Console Hierarchy* pane is used to navigate between the different levels in the form of a tree menu. The levels in the Console Hierarchy pane are the server level, the repository level, and the table level.

- **Records Pane**
  The *Records* pane is an overview of the objects on the selected level. If a server is selected, the Records pane shows an overview of all repositories on that server and the information relevant to these repositories (ports used, languages supported, etc.).

- **Detail Pane**
  The *Detail* pane gives a detailed view of the selected object (see Section 8.6 for more information). It provides an overview of the repository selected and allows you to change the attributes displayed if you want (like the port or language).

Figure 8.6 shows the structure of the MDM Console.
If a repository is selected in the Console Hierarchy pane, the Record pane shows all the associated tables and the Detail pane shows the selected table. On the lowest level, individual tables in a repository can be selected in the Console Hierarchy pane. In this case, the Record pane shows all the fields in the table and the Detail pane shows the selected field.

8.3.2 Administration

Besides the creating, editing, and deleting of a repository, MDM provides other options for the management of the master data inventory. One of the most important is the Verify Repository function, which checks the repository for inconsistencies, referential integrity, etc.

You can verify only the repository (Verify—Check), or correct all errors found immediately (Verify—Repair). For the latter, however,
the repository must be unloaded. This type of error found by using the Verify—Repair check is mostly at the level of the underlying database and is therefore not detectable for the administrator from the console.

As a result of using the Verify Repository function, a report is generated describing the number and severity of the errors found. Here, a Fatal Error is the highest class of error, which makes a repository unusable. Non-Fatal Errors, on the other hand, can lead to performance problems. The third class of error is Warnings, which don’t influence the executability of the repository.

**Master and Slave Repositories**

Another important function is the creation and synchronization of slave repositories in MDM. Slave repositories provide read access only to their data inventory, and are updated only via synchronization (manual or automatic) with the master repository.

**Example**

An application case for master/slave repositories is product master data. Throughout the enterprise, the goal is always to work with the same product data, so different subsidiaries or locations received distributed slaves of the master product repository, which are then all updated at once on a scheduled date. Similarly, the slave repositories on which both the printed and web catalogs are based are also synchronized. So, while the master is updated and edited over a longer period of time, read accesses to the desired master data are performed through the slaves, which represent a snapshot of the master on the date of the last synchronization, and do not allow write access. During synchronization, then, only the changes since the last synchronization need to be transmitted from the master, and not the entire repository, which keeps the downtime—needed by the slaves during the update—to a minimum.

Slaves and master can be located on different DMBS servers; for synchronization, the slave repository must simply be loaded and running.

Slaves and master can also be converted into “normal” repositories at a later date. However, this destroys the synchronization capability. If the master has been transformed into a normal repository, from that point on there is no way to update the associated slaves. However, it is still possible to have read access to them.
Archive and Unarchive Repository

Other administrative activities in the console are Archive and Unarchive Repository. These are used for backing up the repository structure and data. Unarchive Repository can also be used to load repositories, which were originally created in a different MDM system. Here, too, you can exchange or use SAP content. This is delivered in the form of a repository archive and can be loaded at the push of a button like any other archive. SAP currently delivers repository structures for Material, Employees, Customers, Suppliers, Product, and Business Partner.

Unlock Repository

As already mentioned, an MDM Server can call and process repositories on different DBMS servers. A repository may also be called simultaneously by multiple MDM Servers.

To avoid editing the same repository at the same time from two servers, resulting in inconsistencies in the structure, you can lock a repository. If an administrator wants to remove this lock, the Unlock Repository function is available.

Update Repository

If a new version of MDM has been loaded, repositories created with an older version may no longer work without additional effort. In order to continue using these repositories, there is an Update Repository function. The repository is adapted to the database schema of the new version and can then be used again.

Duplicate Repository

Besides the backup functions for archival, copies of a repository can also be created before taking actions like Update Repository with the Duplicate Repository function, and then those copies can be edited and tested to avoid risking harming the original.

Initial tests can be performed or new users trained using the copy. The Duplicate Repository function is also useful for “moving” to different DBMS servers, since a different server than the original repository can be given during copying.
During the copying process, there is no write access to the original repository so as to ensure the data integrity of the copy.

**Compact Repository**

Since all data is loaded into the main memory when a repository is loaded, larger deletion processes can lead to memory fragmentation and inefficient runtime behavior. To correct this, you can use the **Compact Repository** function, which reduces the size of the main memory used and increases its speed again.

### 8.3.3 Special Features

The MDM Console provides a series of special features, which make it possible to generate complex master data models. The repositories provided by SAP and their associated content can also be used as a starting point.

This content includes the import and syndication maps as well as roles, and is usable out of the box. The predefined repositories are: Customer, Employee, Supplier, Material, Business Partner, and Product. They are based on the R/3 data model and equipped with a wide variety of table and field structures. Extraction routines from R/3 are also available, as is content for the mapping to the SAP Exchange Infrastructure (XI). For the portal, too, there is predefined MDM content supplied, which is covered in more detail in Section 8.8.

But the customization of these or the creation of entirely new repositories is greatly simplified by the intuitive usability and the many modeling options.

In Section 8.2, the field properties Keyword and Cache were already addressed briefly. In the following sections, they will now be explained in greater detail.

**Keywording**

In *Keywording*, it is determined for each field whether or not it should participate in keyword searches. The search functions in the Data Manager can also be used to select individual records and groups using filter values for specific fields, but the keyword search...
filters all the fields at once, which support this function. You can find out more about search options in Section 8.4.

It follows that not all fields are suitable for keywording. For instance, keys like customer numbers are not well suited as keywords. The keyword search makes more sense—and has better performance—when it is relative to a term, which occurs in multiple records and when the term searched for can be found in more than one field.

Caching

_Caching_ is particularly interesting for qualified lookups, which contain record-specific attributes for a certain combination of main table and subtable.

In Section 8.2.1, a conditional table was used as an example. There were several different prices stored for a product for each individual customer. To avoid a long table in which the assignments are managed, the prices were defined as qualifiers.

Qualifiers thus refer not just to the table they are in, but also to the associated main table record. They describe this individual relationship between two actual records.

Only if the qualifiers are cached can they be filtered using keyword search. To make this context of the relation between these two records available for the keyword search, caching must be activated for the qualifier, that is, for the price.

Calculated Fields

Another interesting feature in MDM is the option of _calculated fields_, fields that display a value, which is the result of an expression. These expressions can easily be created using a dialog box in which dropdown lists of all relevant fields in the repository are displayed, along with the operators or functions that can be used. In comparison with a pocket calculator, the numerals 0 through 9 would be the fields, and the available operators or functions would be the calculation symbols.

Only numbers can be calculated by or be part of an expression in this feature. Texts can also be processed, that is, by removing empty spaces or calculating text lengths. _If/else_ instructions and other sim-
multiple functions are also provided. However, a value of type Integer, Real, Currency, Text, or Boolean must result from the expression.

Command Line Interface to MDM (CLIX)
The console has one significant failing: It is only available under Windows. As an alternative for Linux and Solaris computers, there is a Command Line Interface to MDM (CLIX). Almost all functions can be called through this interface.\(^7\)

Besides accessing MDM via Linux and Solaris systems, CLIX is also used for the automated execution of administrative activities in the maintenance area, through batch files. One such example is the automated archival of repositories. But MDM server operations like Mount and Start can also be automated.

Mount and Start
To be able to access an MDM Server from the console, it must be available. For this availability, there are two commands: Mount and Start. The difference between Mount and Start is as follows. For every use of a repository, the server on which it is located must first be started. This is also true for access from client applications or iViews. To obtain access from the console to this particular server, the Mount Server operation must be performed. Whether or not the server is running (that is, whether or not it has been started) depends on that very action (if it is not started, it can be mounted, but there is still no access to the repositories).

Even if Unmount Server is used to “leave” the server, it continues to run and can be addressed by other applications. It is another matter with the functions Mount Repository and Load Repository. A repository that is only added to the Console Hierarchy pane with Mount Repository is always unloaded.

Load Repository
For most operations in the console, the repository must be unloaded, since they affect the basic data structure. Parallel read access could

\(^7\) For more information on the capabilities of CLIX, you can consult the SAP MDM Console Reference Guide.
lead to missing information and write access to inconsistencies. Thus no other application besides the console may access a repository that is unloaded. **Load Repository** removes this block and makes the repository available again for all client applications. However, only a few console functions can be performed on this repository.

### 8.4 MDM Data Manager

The **MDM Data Manager** is the tool for the management of the central master data, from creation to maintenance to deletion. This is where functions to create, edit, search, and delete records are provided. Besides the basic functions just listed, MDM also provides a range of other functionality, like matching and merging, which simplify the maintenance of large amounts of master data. Here is where it becomes apparent that MDM is not simply an interface for access to databases, but a business model of master data. In the following sections, we’ll introduce the capabilities of the MDM Data Manager, followed by the functions, which will be covered in more detail.

#### 8.4.1 General Structure

The data model described in Section 8.2 already gave you a preview of the capabilities of the MDM, which we’ll elucidate here from the point of view of the end user.

Flexible and extensive modeling of data structures makes it possible to model almost any data easily, so there is really no limit to the amount of business use you can leverage from MDM. Whether in the personnel department, production, or customer services, the administrator of master data can work with the Data Manager. If only a subset of the functionality of the Data Manager is needed, then a simpler work environment can be provided through the portal. Thus, for example, a clerk in Human Resources can use the Data Manager or the portal to create a new employee, or a new item can be created in product design. Furthermore, by using self services, any employee can change his or her own address and banking information during the hiring process, triggering a workflow which can be verified and approved by the personnel department. To make simple functionality possible for the user, the working interface can also be displayed by the portal in a web browser.
In the structure of the interface, SAP took the market standard Windows interface as a basis. The only thing missing is a help facility integrated into the user interface, so that the SAP MDM Reference Guide or the SAP Service Marketplace must be used.

As usual for Windows-based applications, the Data Manager has a menu bar, a toolbar, and a status bar. Figure 8.7 shows this layout. The status bar includes information about the selected record, the current view, which is also called the view mode, and possible background activities, like a search for a record.

The large area in the middle of the window is called the main window and is broken down into three components.

In the area on the left, the user has the option of orientation based on categories and hierarchies and on values from lookup tables. A tree structure makes it possible for the user to navigate through the data structure based on different perspectives, for example, according to a hierarchy, and thus to be able to find the right record quickly. Moreover, the user also has a free-form search available which makes a
user-defined search possible. This area will be called the tree view in the following.

**Record list area**
The upper area on the right of the main window shows the records selected in the tree view. This area is called the record list area. In record mode, the view for editing or searching for records, all records are shown here and the current record is selected.

**Record editing area**
The remaining part of the main window contains the detailed view. This area is called the record editing area. Here, the fields and attributes of the record selected in the record list area can be viewed and changed. In record mode, this view is broken down into **Record Detail**, **Language Detail**, **Family Detail**, **Validations**, **Workflows**, and **Search Selections**.

### 8.4.2 The Functions of the MDM Data Manager

This section presents an overview of the functional scope of the Data Manager. Since a detailed presentation is beyond the scope of this book, we will limit ourselves to the central functions: the five view modes, the basic functions of searching, creating, deleting, editing, and help functions, and lastly, the additional functions, like merging.

**Views**
The flexible features of MDM are provided by a simple creation of hierarchies, an assignment of attributes to certain categories, and family definition. Using the different views, special information is allocated for this purpose, which is supported by the management of data hierarchies, for example.

**Record mode**
When it starts, the Data Manager is in record mode, which is the primary working view. From here, the content of the data can be managed. The other views are used to edit things like hierarchies or families.

Generally, one starts by selecting a record. The search for a record in record mode can be started from the tree view located in the left area of the main window. The search parameters are shown here. Each search parameter can be opened, showing filter settings. For instance, for a hierarchy, a hierarchical element can be selected to set a filter on the records to be shown. Particularly when searching in
hierarchies, the user is supported by a tree view, which greatly simplifies the setting of the filter. Once the filter is defined, only those records, which are entered under the selected hierarchical element, are shown in the record list area. An example for a filter on a lookup table is the title, which generally consists of the elements Mrs., Mr., or Company. After opening the search parameter for the title, the element Mrs., for example, can be selected. Afterwards, all those records with the value Mrs. stored as the title are displayed. All other search methods will be explained in the Basic Functions section.

If the record view is activated, the records are displayed in the record list area. Only those records that correspond to the filter settings are listed. All properties are shown for each record. These are the fields, which are valid for all records and not only for a specific group.

In order to change the properties of the active master record, the record editing area is used. Here, all information is shown broken down into the six tabs Record Detail, Language Detail, Family Detail, Validations, Workflows, and Search Selections.

Another viewing option is hierarchy mode. In this mode, the user of the Data Manager can view and change all hierarchies stored in the repository. In this view, table selection moves a new control element into the user’s focus. The hierarchy table to be edited can be selected here. Table selection in this view is not only possible for tables that were created as hierarchy tables. The tree view displays the tree structure of the hierarchy. Starting with the root element, all hierarchical levels and hierarchical elements are arranged below it, in a way similar to an org chart or an account plan. The capabilities of hierarchy tables are described in detail in Section 8.2.

In hierarchy mode, the record list area shows the entire contents of the selected hierarchy table at all times. The hierarchical element selected in the tree view is displayed.

If a record is selected in the record list area, then it is displayed with all its details in the record editing area. Here, changes can be made to the details and identifiers can be created in the languages needed.

The taxonomy determines how records belonging to different categories are defined. Through a taxonomy, an attribute can be attached to a record. These additional attributes are defined for records, which belong to a certain category. An example of this could be personal
data, which differs according to the role in the organization, and thus includes additional information. For a clerk who is also a fire protection monitor, for instance, the date of the last fire protection training might also be stored. The taxonomy mode can only be selected if there is a table of type Taxonomy in the repository. The differences between taxonomies and hierarchies and the scope of taxonomy functionality are examined in more detail in Section 8.2.

In the tree view of the taxonomy mode, as for the hierarchy mode, the tables to be edited are selected with the control element at the left end of the toolbar. Then the selected categories are displayed. In the record list area, a list of all defined attributes is displayed. Attributes are only "active" if they are associated with at least one category. This is indicated with a figure eight on its side. If the selected category is assigned one or more attributes, then these elements are represented with a bold font.

For the selected attribute, all details are displayed in the record editing area and can be changed there. The attributes can be created in different languages.

**Family mode**

In family mode, all existing records that are already assigned to a category can be subdivided into families. The taxonomy table in which the family definitions are to be stored is selected in the toolbar. Then the family can be selected in the tree view.

In the tree view, there is a distinction drawn between two elements. First, the categories of the taxonomy table are shown in turquoise. Over these categories, partitioning can be used to generate families. Families are displayed in a light violet in the tree view.

By defining families, records can be grouped together according to certain criteria. The prerequisite for this is that the elements of a family must also belong to a single category. In Section 8.2, there is a more detailed explanation of the options when working with families.

**Example**

In some countries, it is customary to congratulate employees who are celebrating their 50th, 60th, or 70th birthdays. Actually, this poses a particular burden for personnel departments, since the data for all employees must be viewed. With the Data Manager, this can be simplified. For example, all employees who will reach the age of 50 in the next calendar year can be grouped together in one family.
All existing families are displayed in the record list area. To edit a family or to view its details, the family must be selected in the record list area in the usual way. Then the details will be displayed in the record editing area. Here, besides the name and the description, the assignable values are also shown.

The matching mode provides functionality for searching for duplicated records. In SP04, a separate view was created for the Matching function that groups together all the information and functions needed.

Matching has been greatly improved in comparison with the previous release. In contrast to the old functionality, more configuration options and a more sensitive search method are available.

In the tree view and the record list area, approximately the same view is used as for record mode. The tree view is extended with a filter match class in addition to the usual search options. The record list area is also extended with this information for each record. In the record editing area, there are more changes. Here, there is an overview of the records for which matching terminated with more than one match point. Match points indicate how close the match is between two records. Besides this display of matching results, the functions for preparation of matching are also provided. These include the creation of transformations, rules, and strategies.

Since after matching, two or more identical records can be completely or partially merged, the functionality of Merging is also provided.

**Basic Functions**

**Searching**

The search for records can take place in two ways—using filters (as described above), or with the free-form search.

The free-form search is displayed in the lower part of the tree view. Here, all the record’s fields except for attributes from taxonomies are listed in rows. A search method can be determined for each field by selecting an operator and entering a value. You will note that the methods differ for each data field; for example, for a text field, you can choose from the operators contains, starts with, ends with, equals,
does not contain, like, and NULL. NULL is a value designation with its origins in the database world. This value stands for an empty data field. For fields of different types, other suitable operators are provided.

In addition to searching with operators, a search with expressions can also be used within the free-form search (see Figure 8.8). A Boolean expression can be created for this purpose, which returns a value of either true or false. There is a special dialog provided for the development of the expression, which includes functions for the creation of a Boolean expression. This is especially helpful for large quantities of data and with complex relations between the individual fields.

Figure 8.8 Expression Editor

Edit
Before you can start editing data, you must go through a check in and check out process. This process is necessary to maintain transaction safety, which ensures that data always remains consistent.

Two possibilities for this process, also called the check in/check out process, need to be distinguished. First, there is the option of locking a record. You can do this by either using the Records menu item or the context menu. To lock a record, you must select it and then select the Check Out Exclusive function. Now the record can be edited, while it remains locked for all other users. If other employees should only be warned, but not prevented from working on this record, then you can use the Check Out Nonexclusive function. If another user logs in and wants to edit the selected record, this record is first
marked as locked. However, the second user has the option of joining the checkout by using the **Join Check Out** function.

You use the **Check In** function to release the data for editing again. Only then are changes written to the database. If multiple users have worked on a record, then each of these users can end the shared check out process with the **Check In** function. The record of the user who started the check in process is then stored.

If the data changed during a checkout should not be stored, you should use the **Rollback** function. This ensures that the checkout has ended and the changes have been discarded.

Check in/out is provided in the Data Manager, but its use is not absolutely required except in the case of imports. Here, all records are automatically checked out. However, it is still advisable to use this function, because it ensures the consistency of the data and sustains a higher level of data quality.

Once the record is checked out, you can begin the actual work, namely, the editing. MDM provides a series of functions in the **Record Details** section of the record-editing area, which are user-friendly and easy to understand. These functions support the user when changing the contents of records. To move a record within the hierarchies, you only have to change the field that reflects the hierarchy table to the new hierarchical element. There is a similar behavior with assignment to families. Batch editing of the data is also supported.

On the **Language Details** tab, you can enter names for all fields and attributes in language- and country-specific extensions.

On the **Validations** tab, you can specify validity conditions for some or all records. If the validation should apply only to selected records, then the **Automatic Execution** field is assigned the value **None**. Otherwise, the values **Warning** or **Error** are specified to define that the user should receive a warning or an error message. If the values **Warning** or **Error** are specified, the validation applies to all new records. **Assignments** have a similar functionality. They differ in that after the validation of a record there is a standardized reaction. For instance, the contents of records can automatically be changed.
Editing multiple records simultaneously

When managing large amounts of data, you must always take into account the possibility of having to change a larger subset of the data in the same way. In short, the Data Manager is capable of editing multiple records at the same time. Multiple records can either be selected manually, or with a filter function. For example, during a reorganization, half of the employees of the Travel Management organizational unit may need to be moved into the new Global Travel Management unit. Using a filter, all employees in the old unit can be displayed and selected. The record details display shows the list of fields and attributes in different colors, symbolizing the effect of the change:

- **Red**
  Data is different for the selected records.

- **Blue**
  Data is the same, but some records have no information in this field.

- **Pink**
  Data is different and some records have no information in this field.

- **Black**
  Data is identical to the records for this field.

To change the data in the example, the Organizational Unit field would be selected and the value changed to Global Travel Management.

Creating records

Creating records is very similar to editing records; only the check out process is different: You don’t use Check Out, but rather Check Out New Record. From then on, the same options are available for creating as for editing.

Deleting records

To delete records, the records affected must first be selected, in order to delete them using the context menu or the Records menu item.
Additional Functions

Matching

As the amount of data increases, the probability that duplicate records are in the repository also increases. To help resolve this problem, the Data Manager has a functionality called Matching. In the context of matching, the data inventory is searched for duplicated entries, or duplicates.

Before the actual matching run can be executed, it is necessary to make some preparations. One of the first considerations should be how to locate duplicate records. Which data fields are necessary to search records for duplication? Once those fields are identified, so-called transformation rules can be used to change the contents of these data fields. This can prove beneficial if certain parts of a data field are interpreted and entered differently. For instance, an “ö” in German might be rewritten as “oe” to find matches between names with different spellings. The creation of synonyms is also possible, so that nicknames can be added to names. “Tom” can be treated as “Thomas” or “Tommy.” In the context of transformations, virtual data fields are generated. These don’t exist in a database or in the repository, but are defined only by the transformation rules and can be used in the next step, the specification of rules.

Rules are used to calculate the so-called match points, which specify the degree of matching between two records. To create a rule, first, all the fields that are to be used are selected. Then it is defined whether a match should apply to the complete value of the fields or attributes, or whether the match is performed on a character-by-character basis. When matching on a field or attribute basis, a data field of the first record is compared with the same data field of a second record. If the two values are similar, for example, “Tom” and “Tommy,” then the two records do not match on the basis of the selected field. A match is only assigned for identical values. If you select matching on a character basis, then a partial match will be detected. Then threshold values are specified in the Success, Failure, and Undefined fields. The three threshold values define when a field or attribute of two records is the same, similar, or different.

To execute a matching, a matching strategy must be defined. For this purpose, the rules to be used are selected and the limits are defined for the classes Low and High. Now a matching run can be performed.
Match records During a matching run (Match Records), there are three options: (1) the selected record can be compared with all other records; (2) the selected record can be compared with only all the selected records; or (3) the selected record can be compared with the result from a previous matching run. If the matching option is selected, only the matching strategy needs to be specified, and then the results of the matching run are already displayed in the record editing area. Here, the different rules are shown as columns and a total point score is provided. This allows the user to identify identical records quickly and then edit them.

Merging Merging can be used for additional editing.

Include and Set All records identified as duplicates during the matching run are now provided for selection for further editing. To begin merging, first, the records that are to be merged must be selected. Secondly, you can select the data fields and attributes that are to be moved into the new merged record.

Merge Once all the settings have been made, the Merge function is called to merge the records. After a confirmation message, the new record is created and all the old records are deleted. Any existing key mapping settings are retained and now point to the new record.

8.5 MDM Import Manager/Server

Both the MDM Import Manager and the MDM Import Server are used to load new data into a central MDM repository. The primary task of the Import Manager is the manual import of data and the creation of import maps. Import maps can be seen as a type of plan for the import of data. They store all the necessary actions and rules to import the data from a client system into a central repository. The Import Server is used only to import data. A third tool, subordinate to the Import Manager, is the Import Manager Batch, which performs imports of data based on a batch file.

8 The possibilities discussed in this chapter for the functionality of the MDM Data Manager are partly based on the information provided by SAP in the MDM Client Reference Guide.
Index

A

A2i 84
Administration
  Guided Procedure 291
Adobe Interactive Forms 58
Analytics 58, 66
Application independence 41
Application Link Enabling (ALE) 36
Aplistructure 52
Archive 212
Automated data exchange 153

B

Backup 188
Best-of-breed 67
Best-of-breed architecture 24
BI content 278
BI integration 275
BPM → Business Process Management 65
BPP 52
Business content 73, 311, 313
Business context 61
Business Intelligence 66
Business Intelligence tool 22
Business partner 266
Business Process Management (BPM) 65, 72
Business process platform 48, 52, 68, 72

C

Caching 214
Calculated fields 214
Callable objects 290, 294
Central master data management 69
Change tracking table 200
Chief Process Innovation Officer 64
Client system 200, 250, 269
Client/server architecture 23
CLIX (Command Line Interface to MDM) 187, 215
Cluster analysis 130
Collaboration 65
Communications channels 272
Compliance 69, 97
Composite Application Framework 290
Composite applications 48, 59, 65, 66
Composite roles 103
Composition platform 68
Configuring BI 277
Console 208
Consolidation 167
CORBA 48
Core/context model 52
CRM analysis 139
cross-selling analysis 157
revenue analysis 158
up-selling analysis 157
Customer 264
Customer Data Integration (CDI) 90
Customer data management (CDM) 40

D

Data cleansing functions 41
Data extraction and distribution 28
Data groups 197
Data import 28
Data integrity 26
Data Manager 313
Data mining 130
Data model 175, 192
Data modeling 311
Data processing
  file-integrated 22
  program-oriented 22
Data protection 26
Data security 26
Data synchronization 18, 26
Data transfer process (DTP) 279
Data types
  change data 21
  event data 21
  master data 21
  reference data 21
Index

status data 21
stock data 21
transaction data 21
Data warehouse system 20, 22
Database
distributed 25
federated 27
DBMS 27
Delta process 253
Deployment 70
Deployment options 55
Design Time
Guided Procedure 290
Desktop publishing program (DTP) 256
Display field 205
Duet 58, 59

E

Ecosystem 52
Employee 267
Employee Self Services 101
End-to-end process 50
Enterprise application integration (EAI) 40, 46
Enterprise information integration (EII) 40
Enterprise Portal integration 28
Enterprise Portal → SAP NetWeaver Portal 65
Enterprise service-oriented architecture 45
Enterprise services 48
Enterprise services architecture (ESA) → Enterprise SOA 45
Enterprise SOA 16, 45, 54, 166, 309
ERP integration 268
Exchange Infrastructure (XI) 36
Export functions 42
Extraction mechanisms 42
Extraction, transforming, loading (ETL) 40

F

Families 196

G

GDS Console 88
GDS Host 89
Global Data Synchronization (GDS) 87
Global template 167
Guided Procedures 63, 108, 288

H

Harmonization and consolidation 311
Hierarchy tables 193
Historicizing data 22

I

Identity management 18, 28, 106
Image variants 196
Import and export interfaces 311
InfoSpoke 276
Integration architecture 46, 63
Integration of BW 3.0/3.5 279
Integration of BW 7.0 279
Interfaces 273
Intermediate document (IDoc) 36
ISV 93
IT costs 308
IT practices 66
IT scenarios 66
Itemfield 71
iViews 67

J

Java API 297

K

Key mapping 200, 250
Keywording 213

L

LDAP 109, 191
Lifecycle management 58
Logfiles 203
Logins table 199
Loose coupling 47

M
Main table 193
Market segments 165
Masks 195
Master and slave repositories 211
Master data
  maintenance 30
  partially redundant 27
  redundant 27
  transformed 29
  unique 26
Master data consolidation 69
Master data harmonization 69
Master data management
  central 39, 312
  manual 35
  R/3-based 39
  requirements 16
Master data management (MDM) 19, 40
Master data management systems (MDM) 25
Master data services
  standardization 312
Master/slave 185
Matching set 260
Material 267
MDM Console 77, 177
MDM Data Manager 78, 288
MDM Image Manager 82
MDM Import Manager 80
MDM iViews 280, 282
  Attribute Search 284
  Free-Form Search iView 284
  Hierarchical Search iView 284
  ItemDetails iView 283
  Pick List Search 284
  Qualifier Search iView 284
  ResultSet iView 283
  Search State iView 284
  Text Search iView 284
MDM Java API 82
MDM port 256
MDM Publication Manager 246
MDM Publisher 83, 256
MDM → SAP NetWeaver Master Data Management (MDM) 65
MDM repository 177, 268
MDM Server 76, 184
MDM Syndicator 81, 246
MDM workflow 82
Mendocino → Duet 58, 59
Mergers and acquisitions 18, 55, 161
Migration 58
Modularity 42
Multi-tier architecture 49
Multivalue 206
mySAP ERP 2007 312

N
.NET web service 129

O
Object tables 194
OCI 173
Organizational structure decentralized 163

P
Partitioning 26
PDA → Personal digital assistant 66
PDF 71
Peer-to-peer 46
Personal digital assistant (PDA) 66
Platform 56, 63
Plugin 262
Port 269
Portal 280
  page content 285
  page layout 285
  portal content 280
  roles and users 286
  roles and users in the portal 282
Ports 201
Print & Publish 123
Process management 64
Process orientation 41
Product catalog 311
Product information management (PIM) 19, 40
## Index

<table>
<thead>
<tr>
<th>Production location</th>
<th>164</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publishing industry</td>
<td>134</td>
</tr>
</tbody>
</table>

### Q

<table>
<thead>
<tr>
<th>Qualified tables</th>
<th>193</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualifier</td>
<td>193, 251</td>
</tr>
</tbody>
</table>

### R

<table>
<thead>
<tr>
<th>Radio frequency identification (RFID)</th>
<th>58, 66</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-time capability</td>
<td>41</td>
</tr>
<tr>
<td>Realtime Data Acquisition</td>
<td>279</td>
</tr>
<tr>
<td>Relationships</td>
<td>197, 260</td>
</tr>
<tr>
<td>Remote key</td>
<td>250</td>
</tr>
<tr>
<td>Replication procedure</td>
<td>27</td>
</tr>
<tr>
<td>Reporting with SAP BW</td>
<td>28</td>
</tr>
<tr>
<td>Reports</td>
<td>203</td>
</tr>
<tr>
<td>Repository</td>
<td>77</td>
</tr>
<tr>
<td>Request-response</td>
<td>49</td>
</tr>
<tr>
<td>RFID → Radio Frequency identification</td>
<td>58, 66</td>
</tr>
<tr>
<td>Rich Product Content Management (RPCM)</td>
<td>84</td>
</tr>
<tr>
<td>Role-based/workflow capability</td>
<td>41</td>
</tr>
<tr>
<td>Roles</td>
<td>198</td>
</tr>
<tr>
<td>Runtime</td>
<td></td>
</tr>
<tr>
<td>Guided Procedure</td>
<td>290</td>
</tr>
</tbody>
</table>

### S

<table>
<thead>
<tr>
<th>Sales structure</th>
<th>169</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP Auto-ID infrastructure (AII)</td>
<td>66</td>
</tr>
<tr>
<td>SAP content</td>
<td>286</td>
</tr>
<tr>
<td>SAP Enterprise Portal → SAP NetWeaver Portal</td>
<td>65</td>
</tr>
<tr>
<td>SAP Exchange Infrastructure (XI)</td>
<td>65, 108, 173</td>
</tr>
<tr>
<td>SAP NetWeaver</td>
<td>310</td>
</tr>
<tr>
<td>SAP NetWeaver Application Server</td>
<td>65</td>
</tr>
<tr>
<td>SAP NetWeaver Developer Studio</td>
<td>65</td>
</tr>
<tr>
<td>SAP NetWeaver Master Data Management (MDM)</td>
<td>65</td>
</tr>
<tr>
<td>SAP NetWeaver MDM</td>
<td>17</td>
</tr>
<tr>
<td>architecture</td>
<td>18</td>
</tr>
<tr>
<td>components</td>
<td>18</td>
</tr>
<tr>
<td>SAP NetWeaver Mobile</td>
<td>66</td>
</tr>
<tr>
<td>SAP NetWeaver Portal</td>
<td>65</td>
</tr>
<tr>
<td>SAP Solution Manager</td>
<td>66</td>
</tr>
<tr>
<td>SAP Workflow</td>
<td>82</td>
</tr>
<tr>
<td>Sarbanes-Oxley Act (SOX)</td>
<td>55, 97</td>
</tr>
<tr>
<td>Security</td>
<td>190</td>
</tr>
<tr>
<td>Service creation</td>
<td>312</td>
</tr>
<tr>
<td>Service repository</td>
<td>64</td>
</tr>
<tr>
<td>Service-oriented architecture (SOA)</td>
<td>19, 25, 103, 166, 309</td>
</tr>
<tr>
<td>Shared services</td>
<td>49</td>
</tr>
<tr>
<td>Shop-floor systems</td>
<td>60</td>
</tr>
<tr>
<td>Siemens DirX</td>
<td>109</td>
</tr>
<tr>
<td>Single client</td>
<td>167</td>
</tr>
<tr>
<td>Single roles</td>
<td>103</td>
</tr>
<tr>
<td>Single sign-on</td>
<td>103</td>
</tr>
<tr>
<td>SOA</td>
<td>48, 54</td>
</tr>
<tr>
<td>SOBA</td>
<td>48</td>
</tr>
<tr>
<td>SOX → Sarbanes-Oxley Act</td>
<td>55, 97</td>
</tr>
<tr>
<td>SP04</td>
<td>312</td>
</tr>
<tr>
<td>Structure import (non-metadata)</td>
<td>42</td>
</tr>
<tr>
<td>Styles</td>
<td>261</td>
</tr>
<tr>
<td>Sub tables</td>
<td>193</td>
</tr>
<tr>
<td>Supplier</td>
<td>265</td>
</tr>
<tr>
<td>System Landscape Directory (SLD)</td>
<td>270</td>
</tr>
<tr>
<td>System requirements</td>
<td>184</td>
</tr>
</tbody>
</table>

### T

<table>
<thead>
<tr>
<th>Taxonomy tables</th>
<th>193</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCO</td>
<td>54</td>
</tr>
<tr>
<td>Thin client</td>
<td>280</td>
</tr>
<tr>
<td>Total cost of ownership</td>
<td>54</td>
</tr>
<tr>
<td>Transactional system</td>
<td>19</td>
</tr>
<tr>
<td>Transora</td>
<td>87</td>
</tr>
<tr>
<td>Two-phase commit procedure</td>
<td>27</td>
</tr>
</tbody>
</table>

### U

| UCCnet                               | 87  |
| UI patterns                          | 65  |
| Unarchive                            | 212 |
| Unique field                         | 206 |
| User interface                       | 47  |
| Users table                          | 199 |
Index

V
Validation groups  198
Verify Repository  210
Version management  42
Visual Composer  63

W
Web Application Server  →  SAP NetWeaver Application Server  65
Web Dynpro  292
Web service communication  180
Web service source system  279
Web services  49, 179, 313
Workflow  97
Workflow table  197

X
xApp  309
xApp Integrated Exploration & Production  60
xApp Manufacturing Integration and Intelligence  60
xApps  48, 57
XI  →  SAP Exchange Infrastructure (XI)  65
xIEP  60
xMII  60
XML  49, 60, 71, 128, 248
XML schemas  202
XSD  249