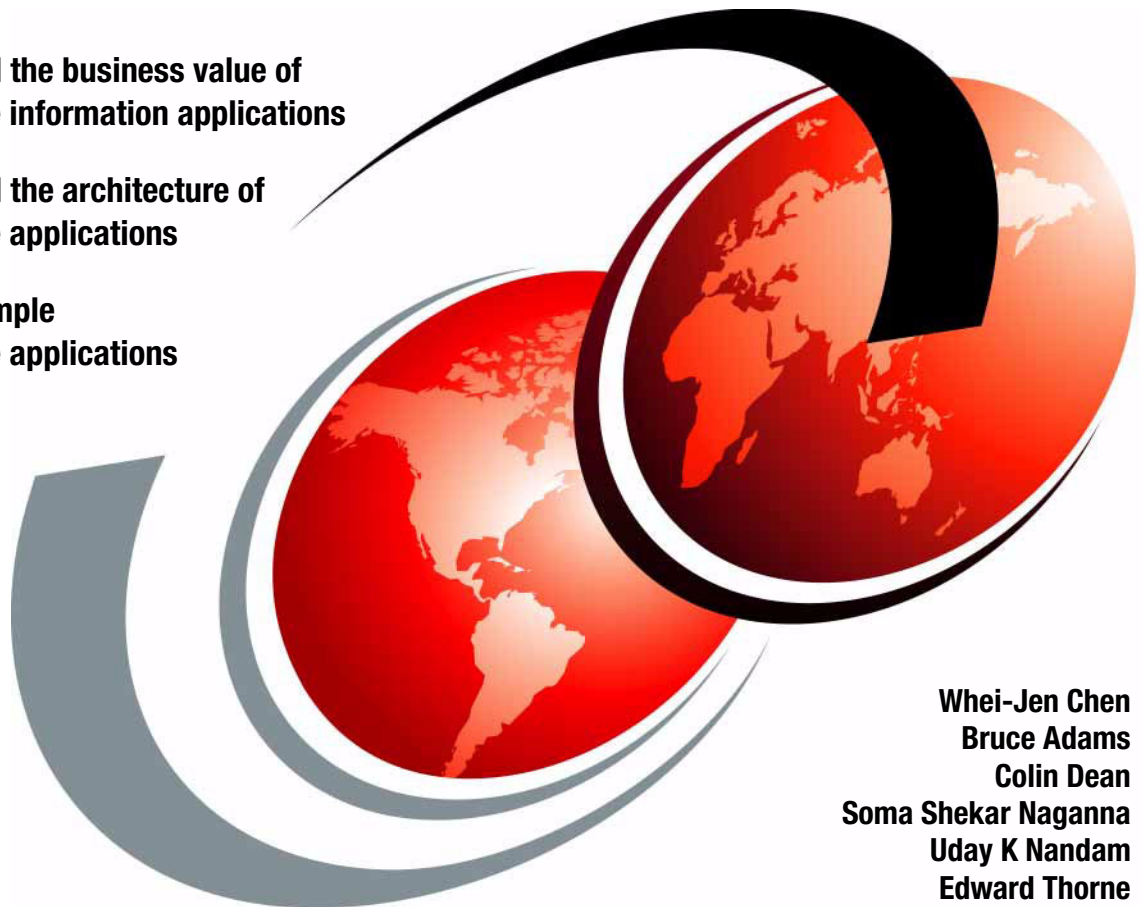


Building 360-Degree Information Applications

Understand the business value of 360-degree information applications

Understand the architecture of 360-degree applications

Explore sample 360-degree applications



Whei-Jen Chen
Bruce Adams
Colin Dean
Soma Shekar Naganna
Uday K Nandam
Edward Thorne



International Technical Support Organization

Building 360-Degree Information Applications

September 2014

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

Second Edition (September 2014)

This edition applies to IBM Watson Explorer Version 9.0.0.3, IBM InfoSphere Master Data Management Version 11.3.

© Copyright International Business Machines Corporation 2014. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
Authors	xi
Acknowledgement	xiii
Now you can become a published author, too!	xiv
Comments welcome	xiv
Stay connected to IBM Redbooks	xv
Summary of changes	xvii
September 2014, Second Edition	xvii
Chapter 1. Introduction to 360-degree information applications	1
1.1 Defining a 360-degree information application	2
1.2 Business value delivered through 360-degree information applications	3
1.3 Benefits of 360-degree information applications	4
1.4 Introduction to IBM Watson Explorer	5
1.5 Introduction to master data management	10
1.5.1 What master data management is	10
1.5.2 Integration between InfoSphere MDM and Watson Explorer	11
Chapter 2. IBM Watson Explorer overview	13
2.1 Introduction	14
2.2 Watson Explorer architecture	15
2.2.1 Connectors and crawling	17
2.2.2 Converting and augmenting data for indexing	17
2.2.3 Indexing data	19
2.2.4 Capitalizing on existing search applications through query routing	20
2.2.5 Querying repositories and displaying search results	21
2.3 Security in Watson Explorer Engine applications	21
2.4 Overview of the Watson Explorer Engine administration tool	22
2.5 Watson Explorer Engine requirements	24
2.5.1 Watson Explorer Engine server requirements	24
2.5.2 Watson Explorer Engine client requirements	25
2.6 Application Builder architecture	26
2.6.1 Modeling data and data relationships using entities	27
2.6.2 Displaying information using pages and widgets	30
2.6.3 Overview of the Application Builder administration tool	31

2.6.4	Scaling and availability for enterprise applications	32
2.7	Application Builder system requirements	35
2.7.1	Application Builder server requirements	35
2.7.2	Application Builder client requirements	36
Chapter 3.	IBM InfoSphere Master Data Management overview	37
3.1	InfoSphere Master Data Management (MDM)	38
3.2	IBM InfoSphere MDM Suite of products	38
3.2.1	IBM InfoSphere Master Data Management	38
3.2.2	InfoSphere MDM Collaboration Server	41
3.2.3	InfoSphere MDM Reference Data Management Hub (RDM)	41
3.3	InfoSphere MDM general product features	41
3.3.1	Matching and linking	41
3.3.2	Search	41
3.3.3	Security and audit	42
3.3.4	Extensibility	42
3.3.5	Standardization	43
3.3.6	Data load	43
3.3.7	Software development kit (SDK)	44
3.3.8	Stewardship and governance	44
3.3.9	Business process management	44
3.3.10	MDM Workbench	44
3.4	Architecture overview	45
3.4.1	InfoSphere MDM Standard Edition	46
3.4.2	InfoSphere MDM Physical/ Advanced Edition architecture	47
3.4.3	InfoSphere MDM Collaborative Edition architecture	48
3.4.4	InfoSphere MDM Reference Data Management architecture	49
3.5	Use cases	49
3.5.1	Government	50
3.5.2	Financial services	53
3.5.3	Healthcare	57
3.5.4	Retail	61
Chapter 4.	Planning a 360-degree information application	65
4.1	Identifying use cases and users	66
4.2	Identifying data sources	69
4.3	Modeling entity relationships	71
4.3.1	Matching entities with non-identical fields	72
4.4	Planning for deployment and implementation	73
4.4.1	Example hardware architecture	75
4.4.2	Scalability	77
Chapter 5.	Lab scenario overview	79
5.1	Key business challenge	80

5.2 Identifying use cases and users	81
5.3 Identifying the data sources	82
5.4 Modeling entity relationships	82
Chapter 6. Installation and initial configuration	83
6.1 Watson Explorer installation and configuration	84
6.2 InfoSphere MDM installation	85
6.3 InfoSphere MDM Connector installation and configuration	85
6.3.1 InfoSphere MDM Standard Edition connector.	85
Chapter 7. Building a golden record	97
7.1 Overview	98
7.2 MDM implementation styles	98
7.2.1 Virtual	98
7.2.2 Physical.	99
7.2.3 Hybrid	99
7.3 Sample process for building a virtual record	99
7.3.1 Three stages to construct and manage a golden record.	100
7.3.2 Workflow for arriving at a golden view	101
Chapter 8. Preparing data for IBM Watson Explorer Application Builder	109
8.1 Creating the search collections	110
8.2 Configuring the search collections and crawling the data	112
8.2.1 Configuring collection seeds	112
8.2.2 Testing the crawling and conversion processes	115
8.2.3 Adding a converter	118
8.2.4 Final search collection configuration	121
8.2.5 Crawling the data	123
8.3 Optional: Scheduling a refresh	127
Chapter 9. Creating an application with Application Builder	131
9.1 Defining entities.	132
9.1.1 Logging in	132
9.1.2 Connecting the engine	132
9.1.3 Creating an entity	134
9.1.4 Viewing entities	136
9.2 Enhancing an entity page	137
9.2.1 Enhancing an entity information widget	138
9.2.2 Adding an order-detail entity	140
9.2.3 Associating products with order details.	143
9.2.4 Formatting order details	145
9.2.5 Sorting orders by ship date	147
9.2.6 Creating a chart widget for top purchases	150
9.3 Searching entities	154

9.3.1 Making an entity searchable	154
9.3.2 Enhancing your search display	155
9.3.3 Adding search refinements	158
9.4 Conclusion	166
Chapter 10. IBM InfoSphere MDM Probabilistic Matching Engine for InfoSphere BigInsights	167
10.1 Introduction	168
10.2 Use cases	168
10.2.1 Probabilistic search	168
10.2.2 360-degree view	168
10.3 PME for BigInsights architecture	169
10.4 Installation	170
10.4.1 Prerequisites	170
10.4.2 Installing PME for BigInsights	171
10.5 Lab scenario	171
10.5.1 Business challenge	172
10.5.2 Data sources	172
10.5.3 MDM algorithm	172
10.6 Configuration	172
10.6.1 Site	173
10.6.2 Algorithm	174
10.6.3 Table	174
10.6.4 Dashboard	178
10.7 Derivation, comparison, and linking	179
10.7.1 Uploading the data	179
10.7.2 Importing the data	180
10.7.3 Verifying the import	180
10.7.4 Verifying derivation	181
10.7.5 Verifying comparison	183
10.7.6 Verifying linking	185
10.8 Probabilistic search	187
10.8.1 Dashboard	188
10.8.2 PME Search	190
10.9 360-degree view	191
10.9.1 PME Extract	191
10.9.2 Composite filters	195
Appendix A. Additional material	209
Locating the Web material	209
Using the Web material	210
System requirements for downloading the Web material	210
Downloading and extracting the Web material	210

Related publications 211

Online resources 211

Help from IBM 211

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

BigInsights™

DB2®

IBM Watson™


IBM®

InfoSphere®

Initiate Master Data Service®

Rational®

Redbooks®

Redbooks (logo) ®

Vivisimo®

WebSphere®

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

Preface

Today's businesses, applications, social media, and online transactions generate more data than ever before. This data can be explored and analyzed to provide tremendous business value. IBM® Watson™ Explorer and IBM InfoSphere® Master Data Management (InfoSphere MDM) enable organizations to simultaneously explore and derive insights from enterprise data that was traditionally stored in “silos” in enterprise applications, different data repositories, and in different data formats.

Applications developed using Watson Explorer and InfoSphere MDM identify data relationships across these silos, unlocking the business value that is inherent in a unified, 360-degree view of the information related to business entities, such as application users, customers, products, and so on.

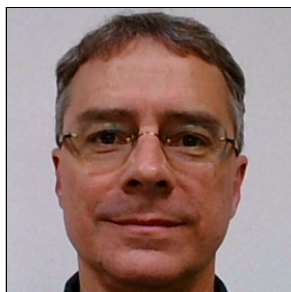
This IBM Redbooks® publication provides information about IBM Watson Explorer, IBM InfoSphere Master Data Management, and IBM InfoSphere MDM Probabilistic Matching Engine for InfoSphere BigInsights™ (PME for BigInsights). It gives you an overview, describes the architecture, and presents use cases that you can use to accomplish the following tasks:

- ▶ Understand the core capabilities of Watson Explorer, InfoSphere MDM, and PME for BigInsights.
- ▶ Realize the full potential of Watson Explorer applications.
- ▶ Describe the integration and value of the combination of Watson Explorer and InfoSphere MDM.
- ▶ Build a 360-degree information application.
- ▶ Learn by example by following a hands-on lab scenario.

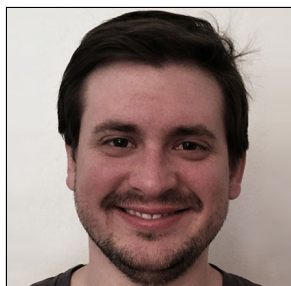
Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Whei-Jen Chen is a Project Leader at the ITSO, San Jose Center. She has extensive experience in application development, database design and modeling, and IBM DB2® system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development, and an IBM Certified IT Specialist.



Bruce Adams is an IBM Advisory Software Engineer working on Watson Explorer in Pittsburgh, Pennsylvania. Before Watson Explorer, Bruce worked on building a platform for large distributed systems at a major bank, distributed transaction processing software products for IBM, specialized database technology for a data analytics service provider, and process control systems for a steel maker. Bruce graduated from Carnegie Mellon University with a B.S. in applied math and computer science. Bruce is an advocate for local technology user groups. He hosts monthly meetings for local Clojure, Python, and Ruby programming groups, and is an organizer for the Steel City Ruby Conference.



Colin Dean is a Software Engineer on the Watson Explorer Connectivity team. Previously, he was a consultant on the Watson Explorer Lab Services team for five years, coming into IBM through the Vivísimo acquisition. He is an advocate for test-driven development, DevOps, security, and design thinking. Colin is a member of Pittsburgh Code and Supply and an organizer of Steel City Ruby Conference, DevOps Days Pittsburgh, and the Emerging Technology Series seminars, as well as Pittsburgh LAN Coalition and TheGXL tournament events.



Soma Shekar Naganna has over 14 years of experience in architecture and development of enterprise applications. Soma joined IBM India Software Labs in 2004 as a staff software engineer. He currently works as senior product architect on Master Data Management Server. His areas of expertise are Master Data Management, Master Data Management for Product Information Management, Data Quality, and Enterprise application integration. He holds a Bachelors degree in Electronics and Communication, and a Masters degree in Computer Science.



Uday K Nandam is a Technical Product Manager for IBM Watson Explorer. He joined the Product Management team in August 2013. Uday has contributed throughout the product, but he spends most of his time specifically on managing Application Builder for Watson Explorer. Before, IBM, Uday was an intern at Microsoft Inc. and Adobe Corporation as a Software Engineer. Uday graduated from the University of Texas at Dallas with a Bachelors degree in Software Engineering.



Edward Thorne is an Advisory Software Engineer working on InfoSphere Probabilistic Matching Engine for BigInsights in Austin, Texas. Before PME for BigInsights, Ed worked on traditional Master Data Management applications, including InfoSphere Master Data Management and IBM Initiate Master Data Service®. Prior to IBM, Ed worked on enterprise software for retail and content management. Ed graduated from Texas State University with a B.S. in Computer Science.

Acknowledgement

Thanks to the following people for their contributions to this project:

Luke Palamara, Senior Product Manager, Watson Explorer, IBM Watson Group, USA

Seth Bachman, Watson Product Manager, Watson Explorer, IBM Software Group, USA

Craig Muchinsky, Senior Technical Staff Member, InfoSphere MDM, IBM Software Group, USA

Thanks to the authors of the previous editions of this book.

- Authors of the first edition, Building 360-Degree Information Applications, published in 31 January 2014, were:

Bruce Adams
Whei-Jen Chen
Arun Manoharan
Luke Palamara
Jennifer Reed
Bill von Hagen

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-8133-01
for Building 360-Degree Information Applications
as created or updated on September 30, 2014.

September 2014, Second Edition


This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

- ▶ InfoSphere Master Data Management overview, installation, configuration, and building golden record
- ▶ IBM InfoSphere MDM Probabilistic Matching Engine for InfoSphere BigInsights installation and usage examples

Changed information

- ▶ Refreshing Watson Explorer (was Data Explorer) with Version 9.0 information



Introduction to 360-degree information applications

Researchers indicate that today's digital universe contains 2.7 zettabytes (2.7×10^{21} bytes) of data, and expect this to grow to 35 zettabytes by 2020. Social media, digital pictures and videos, customer transactions, system location and log data, and cell phone GPS signals are just a few of the sources of the massive volumes of complex data that is being generated at an extremely rapid pace. This is the phenomenon that is known as *big data*.

In its simplest terms, big data can be defined using four V's: *volume*, *velocity*, *variety*, and *veracity* of data. One major challenge that enterprises face today is that they have large volumes of constantly changing structured and unstructured data. This creates a problem where people are unable to find the information they need to get their job done.

The rate at which data is being produced is continually growing, which makes it difficult to even determine who within a company has a particular piece of information. Creating and applying the correct applications that pull data from various data repositories within the enterprise has become a critical mission.

This chapter describes IBM applications that overcome the big data challenge by creating a trusted and holistic 360-degree view of relevant data and the value that those applications deliver by highlighting enterprise use cases.

1.1 Defining a 360-degree information application

A 360-degree information application is a solution that harnesses all of the relevant pieces of information about any business entity, such as customers, products, parts, events, or business partners, by analyzing large sets of structured and unstructured data from multiple repositories.

For example, organizations store and manage information about their customers in multiple data repositories, such as customer relationship management (CRM) systems, customer service applications, data warehouses, and marketing portals. Typically, one must log in to each of these disparate applications, or talk to many people, just to even begin to really understand information about a customer.

An application that delivers a 360-degree view of a customer provides all of the relevant information to allow for a quick understanding of a customer so that action can be taken, such as answering a support call, making an up-sell or even retaining the customer. This unified view enables users to quickly verify a customer, identify products recently purchased, warranty information, billing information, recommendations, support tickets, and recent interactions and conversations, all within a single view and without having to move data from one system to another.

Figure 1-1 on page 3 shows a conceptual overview of a 360-degree information application, highlighting many of the capabilities and associated business value that such applications provide.

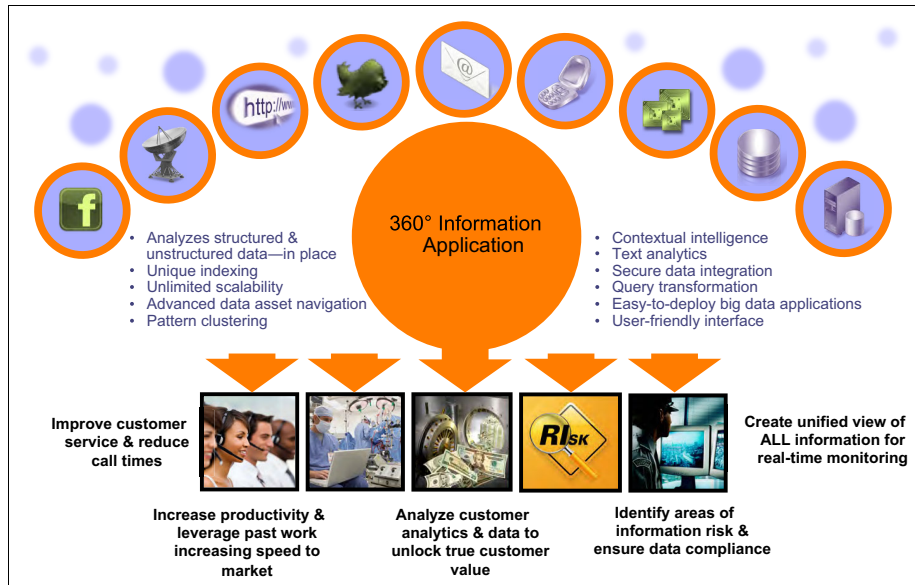


Figure 1-1 Conceptual view of a 360-degree information application

1.2 Business value delivered through 360-degree information applications

360-degree information applications built using IBM Watson Explorer deliver specific business value for many different types of organizations. The following list contains a few common examples:

- ▶ Improving customer service and reducing call times within a customer care center
- ▶ Increasing productivity and capitalizing on past work for corporate marketing teams, increasing speed to market
- ▶ Analyzing customer information and data to unlock true customer value for market segmentation
- ▶ Identifying areas of information risk, and ensuring data compliance within corporate risk management offices

The following list includes common use cases for 360-degree information applications:

- ▶ Customer service optimization
- ▶ Marketing insights portal
- ▶ Partner self-service applications
- ▶ Sales enablement and optimization
- ▶ Research and development optimization
- ▶ Product overview and support optimization

The next few sections take a deeper look at these use cases, and at how 360-degree information applications address them.

1.3 Benefits of 360-degree information applications

Key benefits of Watson Explorer 360-degree view information applications include the following capabilities:

- ▶ Explore and visualize large sets of structured and unstructured data in a meaningful way.
- ▶ Securely index large sets of rapidly changing data sets without needing to move that data.
- ▶ Extract entities and insights from unstructured data to quickly understand and cluster recurring themes
- ▶ Use multiple widgets and widget types to quickly create applications that provide a complete view of a customer.
- ▶ Identify entity relationships within enterprise data sets, capitalizing on IBM InfoSphere Master Data Management (MDM), CRM, and other structured data resources.
- ▶ Visualize real-time analytics of streaming data.
- ▶ Integrate with big data analytics and business intelligence applications.
- ▶ Integrate with Watson Product Portfolio and cognitive applications.

Watson Explorer and MDM are two leading products within IBM that enable organizations to start their big data initiatives with a 360-degree information application. These two products are not the only starting points for such applications, but are the correct ones if the business drivers for the application are similar to those listed in 1.2, “Business value delivered through 360-degree information applications” on page 3.

Other key products used in 360-degree information applications are IBM Watson Engagement Advisor, which helps customers get personalized answers to their questions, IBM InfoSphere BigInsights, which is the IBM based data analytics product based on the Hadoop open source project, and InfoSphere Streams, which analyzes data, such as social media or sensor data, in real-time.

Figure 1-2 provides a high-level view of the IBM big data platform and the hierarchical relationships between different products and capabilities.

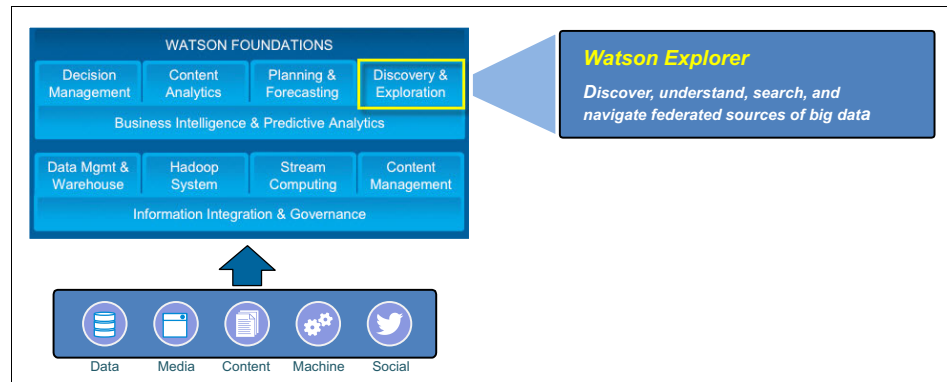


Figure 1-2 Overview of IBM big data platform

1.4 Introduction to IBM Watson Explorer

Watson Explorer provides a platform to navigate a broad range of enterprise content and big data to maximize return on information and gain insight. By delivering information to the correct people at the correct time, Watson Explorer enables organizations to gain the full value of their data through improved operations, real-time decisions, better understanding of customers, increased innovation, fast time to value, and insight into useful action.

The platform also provides the ability to rapidly deploy 360-degree information applications that combine structured, unstructured, and semi-structured data. This combination provides unique contextual views and insights that are not obvious when viewing data from a single source or repository.

Watson Explorer indexes large sets of structured, unstructured, and semi-structured data from disparate data sources. It also provides the ability to build big data exploration applications and 360-degree information applications. Watson Explorer enables customers to create a virtual view of relevant information about different entities, such as customers, products, events, and business partners.

These views are constructed from large sets of data stored in various internal and external data repositories, including CRM, enterprise resource planning (ERP), content management system (CMS), databases, and knowledge management (KM) applications with existing security models and without having to move data. Watson Explorer enables users to tap the value of big data by delivering the correct information at the point where employees need it the most in order to make informed and effective decisions.

With the correct big data solution, companies can quickly render an entirely new class of high-value business insight that can suggest useful actions to decision-makers and front-line employees across the organization. Equipped with this insight, companies can make substantially better decisions no matter what the business climate.

Organizations with differentiated big data strategies will be in a better position than their competitors to take full advantage of market expansion and emerging market opportunities, all enabled and supported by Watson Explorer.

Consider a customer service representative responding to a product question, a researcher trying to use past knowledge to push forward a new idea, a marketing lead trying to plan their next big campaign, or a CFO trying to identify where their organization is at risk for making secure content accessible. Watson Explorer can address all of these challenges by enabling users to explore and discover information in a secure way.

As the world of big data evolves, organizations are focused on how they can ensure that users of their lines of business are discovering insights when they need it the most. Figure 1-3 provides a hierarchical overview of the capabilities provided by different portions of Watson Explorer and its underlying framework.

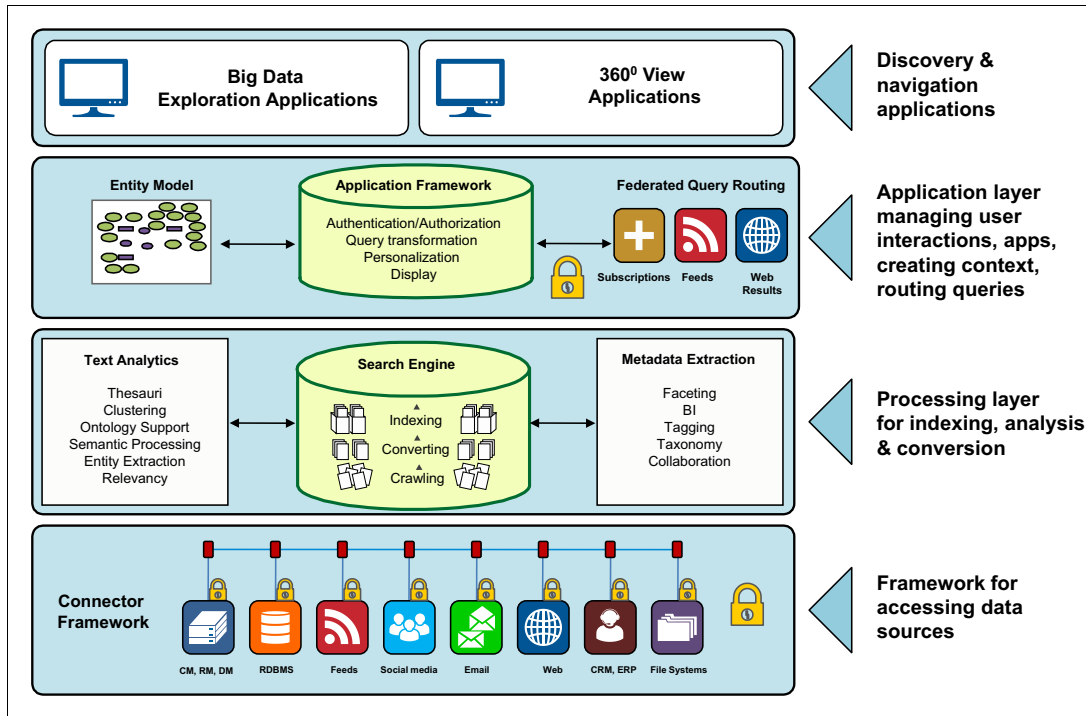


Figure 1-3 Watson Explorer architecture

Watson Explorer consists of two modules:

- **Engine**

Watson Explorer Engine is the result of continuous innovation over the past 10 years. Its ability to securely crawl and index large sets of data has won many customer references and kudos from leading industry analysts.

- **Application Builder**

Application Builder is an evolution of the Watson Explorer Engine module that caters to the needs of the growing big data market. Application Builder enables organizations to quickly and easily create applications that organize information around an enterprises big data, enabling analysis that can solve various business problems in the areas of customer experience, sales optimization, and research and development.

Application Builder uses data from the Engine module, enabling implementors to create domain-specific models, organize data related to entities, and create and modify widgets to meet business needs.

In addition to these stand-alone applications, Watson Explorer includes a powerful, enterprise-ready API known as the *BigIndex API*. This API provides high-level interfaces for basic Engine and Application Builder capabilities, such as creating search collections, searching indexes, and returning organized sets of search results.

The BigIndex API also provides simple API functions that automate traditional system administration tasks, such as distributing search collections across multiple hosts, and balancing search collection access and data storage requirements. See “Enterprise requirements and the BigIndex API” on page 34 for more detailed information about the BigIndex API and its capabilities.

The following list includes some of the key Watson Explorer capabilities:

- ▶ Indexed exploration, navigation, and discovery across a broad range of applications, data sources, and file types. These capabilities enable searching content that is stored in enterprise applications such as CMSes, CRM systems, supply chain management packages, email systems, relational database management systems, web pages, and networked file systems.
- ▶ Federated access to non-indexed systems, such as premium information services, supplier data, business partner portals and catalogs, and existing applications using Query Routing. Search results from these non-indexed sources can be merged with search results from indexed data sources, or they can be examined separately.
- ▶ Rapid deployment of 360-degree information applications that combine structured, unstructured, and partially structured data to provide unique contextual views and insights that are not obvious when viewing data from a single source or repository.
- ▶ Deep integration with IBM Text Analytics to extract insights that enable a quick understanding of huge volumes of unstructured content.
- ▶ Highly relevant navigation results for precise discovery against data sets of any size or structure. The relevance model accommodates diverse document sizes and formats while delivering consistent results.
- ▶ Sophisticated security model, including cross-domain and field-level security, to ensure that users only see content they would be able to view if logged in to the source application.
- ▶ Unique, schema-less index structure that is expressive and versatile, supporting advanced features, such as server push, rapid refresh or continuous updating, real-time searching, field-level updates, and security.
- ▶ Rich analytics, including clustering, categorization, entity and metadata extraction, faceted navigation, conceptual search, and document de-duplication.

- ▶ Rich, flexible set of connector plug-ins that support secure connections to many enterprise systems, fusing data together in a single view without moving the data.
- ▶ Dynamic page creation capabilities that enable applications to dynamically create individual pages for customers and products. Pages are generated dynamically by linking to the relevant data source. When new data appears in the data source system (CRM, MDM, and so on) new pages are automatically generated in 360-degree information applications that reference those data sources.
- ▶ Semantic logic through MDM integration that enables Application Builder to match different versions of the same data set across different data repositories. This feature is particularly helpful in locating unstructured content. (See 1.5, “Introduction to master data management” on page 10 for more information.)
- ▶ Two-way information exchange, enabling application users to both examine information from various repositories and augment that information, adding comments, ratings, and tags to indexed data. These collaborative capabilities simplify sharing high-value data between multiple users through shared spaces, activity posts, useful links, and so on.
- ▶ Real-time activity feeds enable users to see content changes that relate to critical, role-centric items of interest, such as customers and products for which they are responsible.
- ▶ Rich search capabilities enabling context-sensitive exploration, search, and navigation that provides additional insights and refined historical information.
- ▶ Highly elastic fully distributed architecture offering fault-tolerance, vertical and horizontal scalability, master-master replication, and shared-nothing deployment.

See Chapter 2, “IBM Watson Explorer overview” on page 13 for detailed information about how the design and architecture of Watson Explorer Engine and Watson Explorer Application Builder simplifies creating applications that deliver these capabilities.

One of the most complex aspects of dealing with huge amounts of information in discrete data repositories is identifying relationships between that data. Watson Explorer can capitalize on other IBM products that focus on identifying data relationships, such as MDM.

1.5 Introduction to master data management

Master data is high-value, core information, such as data on customers, patients, products, materials, accounts, and other entities, that drives critical enterprise processes. It is central to every business transaction, application, report, and decision. Master data is fundamental to enabling an organization to meet its strategic objectives, such as increased revenue, reduced cost, improved agility, increased compliance, and reduced risk.

Managing master data can be difficult, because this data is typically distributed across numerous source systems, each of which creates and maintains data in its own way. To be successful, organizations must integrate master data from disparate systems, creating single, complete, and accurate views of entities, and eliminating the missing, duplicate, and inconsistent information that can lead to poor decisions and inefficient processes.

1.5.1 What master data management is

Today's organizations are process-driven, and many of the most critical business processes depend on high-quality data (such as customer data, product data, and location data), and the relationships which are inherent in that data. These critical data assets, known as *master data*, are central to business operations.

Unfortunately, the data that these processes rely on is often of poor quality. Inaccurate, incomplete, missing, or duplicated data means that the business processes that depend on that data can be inefficient, ineffective, and costly, increasing business risk. This business data is also typically dispersed across multiple, non-integrated information repositories the data volumes of which continue to grow exponentially, compounding the problem over time.

Business users that are concerned about the effectiveness of key business processes need to be concerned about the accuracy of the data that feeds those processes.

Implementing a discipline around master data helps organizations determine what information needs to be in a widely recognized single source, a "golden record." It also helps users identify accurate data that reflects real-world changes in business conditions, and therefore determine the relevance and meaning that is needed for specific business processes and users.

1.5.2 Integration between InfoSphere MDM and Watson Explorer

InfoSphere MDM enables organizations to create single, trusted data views that increase the efficiency of processes and facilitate better decision making. This centralized data view enables those organizations to improve business results, lower costs, reduce risks, and achieve the agility to meet current and future business needs.

Watson Explorer enables organizations to gain more value from their MDM investment by integrating trusted master data with other enterprise data. The unique capabilities of Watson Explorer simplify connections to the MDM repository, enabling 360-degree information applications to capitalize on trusted master data as the foundation for consolidating and indexing additional sources of relevant data.

Figure 1-4 shows how the MDM integration of data from multiple sources enhances the 360-degree information applications that one can create using Watson Explorer.

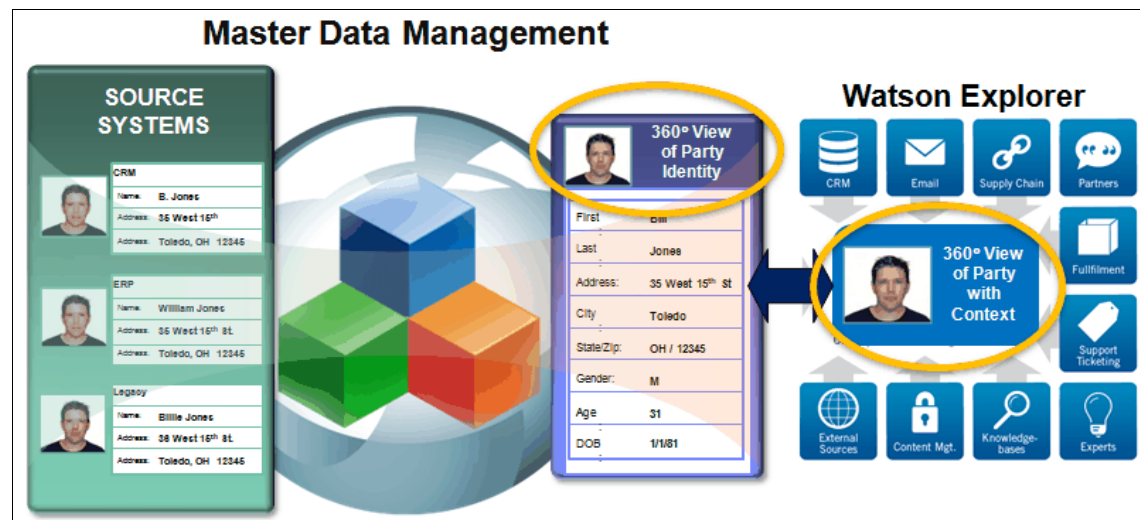


Figure 1-4 Integration between Watson Explorer and InfoSphere Master Data Management

Watson Explorer includes connectors to index MDM data. The MDM collection can then be used to seamlessly integrate InfoSphere MDM with IBM Watson Explorer. By capitalizing on this integration, organizations can create customized, customer-centric interfaces that automatically display relevant information.



IBM Watson Explorer overview

This chapter provides a high-level architectural overview of the IBM Watson Explorer modules that are used to create 360-degree information applications. The first few sections of this chapter provide information about the logical components and capabilities of Watson Explorer Engine, the interaction between them, and how Watson Explorer Engine enforces security when exploring enterprise information.

Subsequent sections of this chapter provide information about how Watson Explorer Application Builder builds on top of Watson Explorer Engine to develop applications that provide a 360-degree view of the enterprise, and explain how Application Builder provides those applications with the high-availability and high-reliability features that are required by today's enterprise applications.

This chapter also summarizes the server requirements for installing Watson Explorer Engine and Watson Explorer Application Builder, and the client requirements for using the Watson Explorer Engine and Application Builder applications.

2.1 Introduction

Watson Explorer Engine was originally developed as a framework for creating enterprise search applications that can simultaneously and securely search the contents of multiple, heterogeneous data repositories. Although Watson Explorer Engine is still used to develop pure search applications, its role in enterprise environments expands as organizations capitalize on its rich index capabilities to enable the broader exploration of organization's big data assets.

Watson Explorer Engine is integrated with other products in the IBM big data portfolio such as InfoSphere BigInsights, and provides flexible and extensible APIs that simplify its integration into other enterprise and big data products.

In the big data context, the latest generation of Watson Explorer Engine is most commonly used to develop data exploration and information navigation applications that enable users to explore vast quantities of enterprise data, regardless of where it is stored, providing fast time to value. Its navigation and exploration capability makes it easy to use Watson Explorer Engine as a stepping stone for and core component of enterprise big data projects.

Watson Explorer Engine's powerful search and exploration capabilities enable users to navigate all of the content that is available on an enterprise network, in enterprise applications, and in existing search applications that are of specific interest to a business or organization.

Watson Explorer Engine provides a flexible framework that JVM-based¹ plug-ins use to enable organizations to retrieve content from any network-accessible data repository, ranging from file systems to specific applications. It also includes sophisticated mechanisms for standardizing, indexing, and subsequently exploring that data.

Designed for enterprise environments where document security is critical, Watson Explorer Engine mirrors repository security models and faithfully applies user, group, or access control lists, and other security mechanisms to the content that it has indexed. Only users who have the ability to see specific content in a data repository are aware of, and can search, the equivalent content in Watson Explorer Engine's index. This extends to individual metadata fields, as well. This unique capability is known as *content-level security*.

¹ Java Virtual Machine. Connector plug-ins are written primarily in Java and Scala languages.

Watson Explorer Application Builder enables application developers to build browser-based applications that use Watson Explorer Engine to provide unified search and information navigation capabilities across all of an organization's content. Application Builder adds the ability to show relevant, personalized data based on its relationships to other relevant data.

For example, in organizations such as call centers or sales desks, where individuals are associated with certain products, certain customers, or share responsibilities across multiple domains, Application Builder allows you to create applications that personalize search or other information navigation results based on an individual's role in that organization.

These capabilities provide the next generation of relevance by not only delivering results that are relevant to a specific question, but also delivering results that are relevant to each user's organizational context. Application Builder can also use that context to better identify relevant content.

The next few sections go into more detail about the architecture and components of Watson Explorer Engine and Application Builder, providing information about their underlying components and the characteristics of each, and highlighting how these two products work together and complement each other.

2.2 Watson Explorer architecture

Watson Explorer Engine provides a flexible, modular architecture that enables it to interact with the widest possible number of networked data repositories and with other network-accessible data sources such as search engines. This architecture simplifies creating single information navigation applications that can simultaneously explore content stored in a variety of formats and locations.

This includes formats that are unique to specific applications and their associated data repositories. It also enables combination of different data formats that are found in networked repositories where heterogeneous content is to be expected, such as file systems or content management systems.

Exploring data repositories through Watson Explorer Engine uses the same capability that is used to search them, which is an index of the contents of all of the documents in that repository. Watson Explorer Engine creates an index of the documents from a single data repository in the following ways:

- Retrieving data, metadata, and security information from that repository, which is known as *crawling* that repository. When crawling a repository is impossible or is inefficient, Watson Explorer also supports server-side applications that push data from a data repository to a Watson Explorer installation.

- ▶ Converting retrieved data into a common format that is suitable for subsequent processing, and augmenting the retrieved data with relevant meta-information, such as the URL with which it was associated, its original file type, permission information, and a unique identifier for each retrieved item.
- ▶ Creating an index of that data. This sort of index was historically referred to as a *search collection*, but is now referred to as an *index* because of its association with big data exploration and information navigation, rather than being limited to facilitating enterprise search.

Figure 2-1 shows the functional components of Watson Explorer Engine, the interaction between them and external data repositories (identified as DATA in the figure), and their use in a typical information navigation or search application.

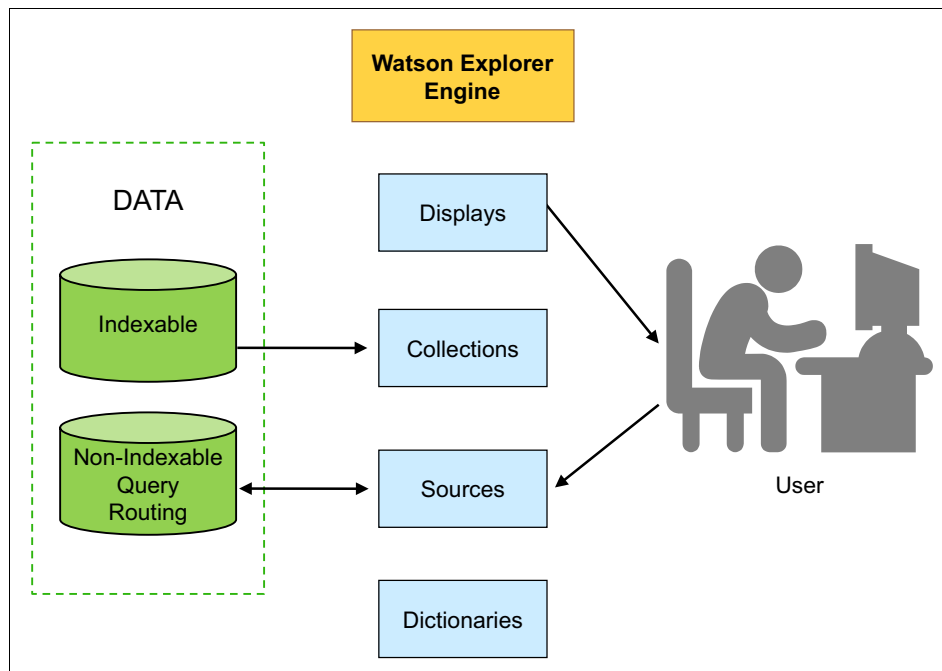


Figure 2-1 Watson Explorer Engine architecture diagram

The next few sections provide more detailed information about the components of this architecture and the process of creating an index of the contents of a remote data repository using Watson Explorer Engine.

2.2.1 Connectors and crawling

Watson Explorer Engine can access and index any data that can be accessed using a Universal Resource Locator (URL). Watson Explorer Engine provides *connectors* that enable it to access and retrieve the data in remote repositories. Some of these connectors are built into Watson Explorer Engine, but a majority are separate JVM-based plug-ins that use the APIs associated with a specific remote repository to facilitate retrieving the URL associated with each data item in that repository.

For example, the connectors for various databases use the Java Database Connectivity (JDBC) API to retrieve the URLs associated with the information in that database.

As URLs for each data item in a data repository are retrieved, they are either enqueued to a Watson Explorer Engine component known as the crawler, or are re-processed to facilitate subsequent enqueueing. For example, URLs in a structured, hierarchical data repository can either identify specific data items in that repository, or can indicate an additional level of hierarchy, just as files and directories contain data and represent hierarchy within a file system.

As URLs that identify data are enqueued to the crawler, the crawler retrieves the data associated with each enqueued URL, and passes on that data and associated metadata for subsequent processing by other Watson Explorer Engine components. Because these types of connectors retrieve both URLs and data from remote repositories, they are often referred to as *pull connectors*.

An alternative to crawling remote data repositories is to install a server-side component that can access those repositories, and that directly enqueues data from those repositories to Watson Explorer for indexing. These are known as *push connectors*, and are typically used where data repositories do not offer a suitable API, or where crawling a repository is impossible or is inefficient.

2.2.2 Converting and augmenting data for indexing

After the data identified by a URL is retrieved by the crawler from a remote repository, it is handed off to a series of applications, known as *converters*, for normalization and, if necessary, augmentation. The data that is retrieved by the crawler is typically in a format that is specific to the repository from which it was retrieved (or to its file type if retrieved from a heterogeneous resource, such as a file system or content management system).

Indexing data in different formats would require the indexer to understand all of those formats, which would increase its complexity, dilute its focus on indexing, and would also require changes to the indexer each time Watson Explorer Engine needed to index a previously-unseen type of document.

To be quickly and accurately indexed, and to enable the indexer to focus on indexing rather than data transformation, retrieved data is therefore normalized by converting it to a common format that is used internally by Watson Explorer Engine. This normalization is done by processing the data through a sequence of converters, which is generically referred to as a *conversion pipeline*.

Each converter in a conversion pipeline either converts the data from one format to another, or augments its input data by extracting and adding additional information from the input data before passing the data to the next converter in the pipeline. Each repository that is being crawled and indexed has its own conversion pipeline, which typically consists of a combination of converters:

- ▶ Standard converters that are delivered with Watson Explorer Engine
- ▶ Custom converters that are written by a Watson Explorer Engine application developer to extract or augment specific data in specific ways

Each converter has an associated input and output type. The only converters in a conversion pipeline that are triggered during the normalization and augmentation process for a given bit of crawled data are those that match the characteristics of the initial input data and the output of previous converters.

A conversion pipeline also follows a predefined conversion sequence, enabling augmentation of certain input data and subsequent format conversion by other converters that support the same input data type.

The eventual output format of any conversion pipeline is a common data format that is used internally by Watson Explorer Engine, and which is known as VXML (Vivísimo XML, for historical reasons). VXML has a well-documented, schema-less data syntax:

- ▶ Multiple attributes for each retrieved data item preserve information about the URL that was associated with the retrieved data, and preserve metadata about that data in the context of its original repository, such as data ownership, data protections, and so on.
- ▶ The ability to split the data associated with a single URL into multiple content items that can provide information about the structure and composition of the original data, or that can subsequently be processed or indexed differently.
- ▶ Converters can add unsearchable metadata used for only display purposes, such as content-level security ACLs or grouping identifiers.

When crawled data passes through a conversion pipeline and is converted into appropriately augmented VXML, that VXML is passed to the indexer for indexing.

2.2.3 Indexing data

Watson Explorer applications use an index to facilitate data exploration and information navigation, because it would be highly inefficient to explore each remote repository each time one looked for specific information. The benefits of indexing data include the following capabilities:

- ▶ Network traffic is minimized.
- ▶ Searches exploring multiple repositories are run simultaneously and in parallel, instead of searching each repository individually and sequentially.
- ▶ Repositories that have similar, desirable metadata can be searched while repositories that lack the metadata are not searched.

Although the tasks of creating an index for different types of data repositories vary widely, a Watson Explorer index is explored and used in the same way by any Watson Explorer applications.

Watson Explorer Engine information navigation and data exploration applications facilitate today's big data requirements by enabling the examination of multiple indexes simultaneously. Being able to use multiple indexes can provide several advantages:

- ▶ Enables optimizing the process of connecting to, and retrieving content from, specific data repositories, and standardizing, augmenting, and indexing that content to produce a given index, based on the type of input data that is associated with a given repository. This can provide higher performance when creating and subsequently updating a specific index.
- ▶ Reduces the size of any given index. This helps minimize both immediate storage requirements and infrastructure storage requirements for tasks such as backups.
- ▶ Reduces the time and resources that are required to access data within an index. Using smaller files requires less memory to load and query those files, and lookups within a given index are generally faster for smaller files.

Being able to use multiple indexes simultaneously also enables Watson Explorer Engine application developers to distribute the effort required to create and maintain these indexes across multiple systems. This includes the crawling and conversion phases that lead up to index creation.

Indexes that are being created on different systems, by different Watson Explorer Engine instances, provide the most up-to-date possible picture of an organization's enterprise data:

- ▶ They can focus on different subsets of enterprise data.
- ▶ They can be created in parallel.
- ▶ They can be updated in parallel.

To support the reliability, availability, and serviceability (RAS) requirements of enterprise applications, Watson Explorer Engine indexes can also be configured so that they are replicated across multiple Watson Explorer Engine installations. This is generally referred to as *distributed indexing*.

Updates to these replicas, based on changing content in the data repository that they are associated with, can be initiated by any replica, and are automatically distributed to all other replicas. Index synchronization is done transactionally to ensure successful and consistent updates across all index replicas.

2.2.4 Capitalizing on existing search applications through query routing

Watson Explorer applications that are designed to help explore and navigate big data typically rely on indexes of the various networked application and mixed-format data repositories found in an enterprise content. When creating traditional enterprise search applications or Watson Explorer Application Builder applications that deliver role or context-based information, it is often useful to be able to deliver data that is a combination of indexed content and search results that are provided by other applications.

The previous sections explained how network-available data content is indexed for exploration. This section highlights how search results from other applications can also be delivered in the context of Watson Explorer Engine and Application Builder applications.

Sometimes also referred to as *federated search*, the ability to combine search results from existing search applications with the data exploration and navigation capabilities of Watson Explorer is more precisely referred to as *query routing*.

This term means that exploratory queries can be routed to existing data search applications, and to content that has been indexed by Watson Explorer. This is often useful to enable 360-degree information applications to access all of the information resources that are available within an organization, regardless of a desirable repository's proclivity to be indexed.

Watson Explorer Engine applications use a component known as a *source* to access both indexed content and existing search applications. When exploring data that has been indexed by Watson Explorer Engine, a source enables fine-grained control over how matching data is returned and processed.

When routing an exploratory query to an existing search application, a source identifies how to submit that query to that application. The source also identifies how to prioritize and paginate the information that is returned from that search application, so that it can be successfully combined with the results of exploring indexed content.

2.2.5 Querying repositories and displaying search results

Data exploration and information navigation applications that are created using Watson Explorer Engine use a source to determine how to format and submit an exploratory request to both indexed content and to existing search applications. The data that is returned in response to such requests is formatted and shown in a web browser by a Watson Explorer Engine component known as a *display*.

These displays are created in the Watson Explorer Engine administration tool, which is described in 2.4, “Overview of the Watson Explorer Engine administration tool” on page 22.

The 360-degree information applications created using Watson Explorer Application Builder use the Application Builder administration tool to configure how the results of exploratory queries are displayed. The Application Builder administration tool enables developers to create widgets that display all of, or subsets of, the query results. The widgets also provide various techniques for navigating through those results, and of exploring sets of related results.

2.3 Security in Watson Explorer Engine applications

Watson Explorer Engine enables application developers to record access control list (ACL) information when crawling many types of resource. This information is stored with each item that is indexed. This information can then be used to enable data exploration and information navigation applications to respect those access control settings on indexed data, ensuring that users see only results that correspond to documents and other types of data that they are authorized to see.

Both Watson Explorer Engine and Application Builder applications that want to respect access controls require that each user of an application be authenticated as a valid user of those applications. This authentication is typically done through a networked security mechanism. The most common mechanism is a Lightweight Directory Access Protocol (LDAP) server, such as Microsoft Active Directory.

When a user is authenticated, their authenticated identity can be used to authorize or deny access to indexed content based on the ACL information that is associated with each indexed document or other type of indexed data.

A Watson Explorer Engine application has settings that identify a source of authentication information. After a user is authenticated through an HTTP authentication prompt, Windows Integrated Authentication, a local Engine account, or a custom login system, the application uses an installation-wide procedure to preserve that authentication information in a user profile. It then acquires, caches, and submits ACL data when the user submits a data exploration or information navigation query.

360-degree information applications that are created using Application Builder are supported by an application server that is included with each Application Builder installation, and that does the actual authentication required to access an application. Application Builder applications use an LDAP server, such as Active Directory, identified in the application's configuration information, to identify the users of an application and the authorization groups to which they belong.

2.4 Overview of the Watson Explorer Engine administration tool

The Watson Explorer Engine administration tool is a graphical, browser-based tool that is used to configure, manage, and debug the data exploration and information navigation applications created with Watson Explorer Engine. This tool provides a web browser-based graphical user interface (GUI) that enables application developers to configure the Watson Explorer Engine software without having to modify manually the XML configuration information that Watson Explorer Engine uses internally, except in highly advanced deployment scenarios.

The GUI also provides a convenient interface for viewing the online documentation that is provided for Watson Explorer Engine and the administration tool.

Figure 2-2 shows a sample window of the Watson Explorer administration tool.

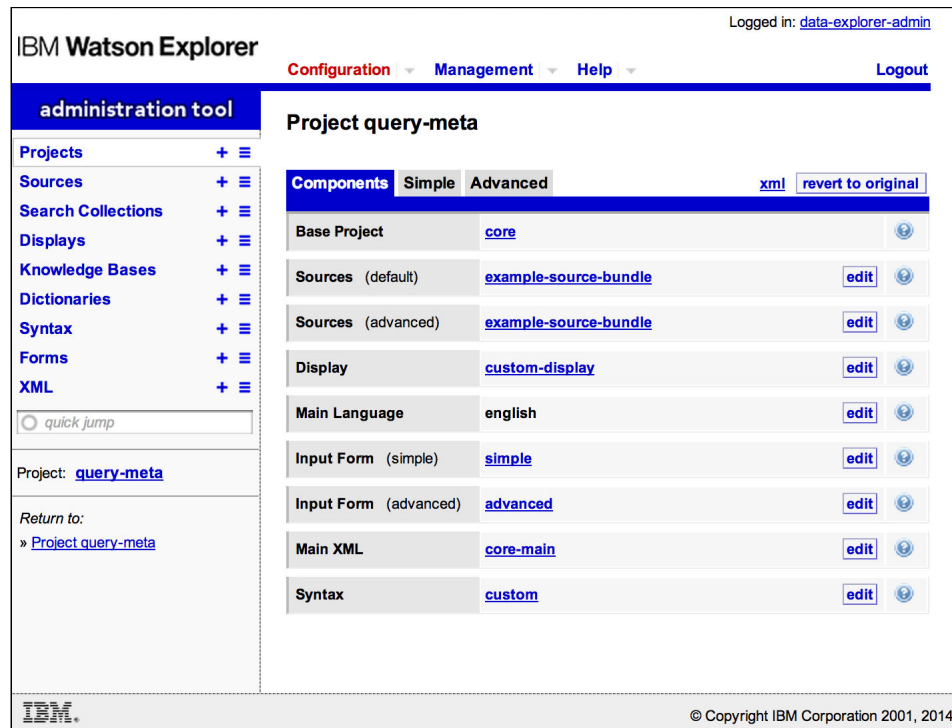


Figure 2-2 Watson Explorer Engine administration tool

The Watson Explorer Engine administration tool provides menus across the top of its interface:

- **Configuration**
Contains menu commands that enable an application developer to create projects, indexes (referred to as *search collections* in the administration tool), and the sources, displays, and other components that are used to create and configure data exploration and information navigation applications.
- **Management**
Contains menu commands that enable an application developer to examine the characteristics of the associated Watson Explorer Engine installation. It also has commands that the developer can use to manage the users who can access that installation, modify the components that it contains, schedule, start, and stop the services that are associated with any applications that are supported by that installation, and so on.

- ▶ Help

Contains menu commands that enable an application developer to search for online help on a specific topic, access the online help that is associated with the page that is currently shown in this tool, and so on.

When the Configuration or Management menus, or a command on those menus, is selected, the left navigation menu in the Watson Explorer Engine administration tool displays the contents of the associated menu. For example, the left navigation menu in Figure 2-2 on page 23 shows the commands that are available on the configuration menu because that menu is currently selected.

Mandatory fields, values, and options that must be selected or supplied a value are always indicated in two ways:

- ▶ The field name is bold.
- ▶ The field name is marked by a small red asterisk.

Each required field is identified in this way to make it easier for a user to differentiate mandatory fields from optional ones, which one might want to use to customize or further refine the data exploration or information navigation application, or for which one might only need to provide values in certain cases.

Tooltip help for all of the fields, options, and variables that can be set in the Watson Explorer Engine administration tool is available through a tooltip icon, a question mark inside a blue circle, that is displayed to the right of each object. Clicking this icon displays a short help message in a pop-up dialog. Many options within the admin tool have default values or provide examples when there is no sane default.

2.5 Watson Explorer Engine requirements

The next two sections identify the requirements for installing and running Watson Explorer Engine on a server, and the requirements for client systems that must access the Watson Explorer Engine administration tool, Watson Explorer Engine data exploration or information navigation applications, or both.

2.5.1 Watson Explorer Engine server requirements

To install and use Watson Explorer Engine on a given system at the time that this book was written, that system must satisfy the following requirements:

- ▶ Supported 64-bit hardware
 - A 64-bit (AMD64 or Intel 64) x86 system with a 2 GHz or better processor. A minimum of 4 GB of memory is suggested.

- ▶ Supported 64-bit operating system

A 64-bit Red Hat Enterprise Linux (RHEL) 5.3 Server or Advanced Platform distribution or later, RHEL 6.0 Server or later, equivalent 64-bit RHEL-based distributions, SUSE Linux Enterprise Server (SLES) 11.2 or greater, a 64-bit version of Microsoft Windows Server 2008 Datacenter and Enterprise editions, or Microsoft Windows Server 2012 Datacenter edition.

- ▶ Web server

Watson Explorer Engine includes an embedded Apache web server meant for development and installations with a small user base. We suggest to use web servers commonly used in enterprise environments with system web servers, such as an Apache HTTPd package provided by the operating system's software repositories, or Microsoft Internet Information Services (IIS) on Windows. When integrating Watson Explorer Engine with a system web server, that web server must be installed and configured before installing Watson Explorer Engine.

When integrating with a system web server, although Watson Explorer Engine has been shown to run on a wide variety of web server software, IBM suggests using one of the following servers:

- ▶ Apache 2.x
- ▶ Microsoft IIS 6, IIS 7, or IIS 7.5

It is important to review the Server Installation Requirements provided in the Watson Explorer documentation, in case these requirements have changed in subsequent releases of the software.

Note: As described in 2.7.1, "Application Builder server requirements" on page 35, Watson Explorer Application Builder runs in the context of an application server, and includes a bundled version of IBM WebSphere® Liberty Profile. Application Builder therefore does not require integration with a system web server.

2.5.2 Watson Explorer Engine client requirements

To access the Watson Explorer Engine administration tool, a data exploration or information navigation application that was developed using Watson Explorer Engine, or both, from any computer system, that system must provide a supported web browser. Watson Explorer Engine fully supports Microsoft Internet Explorer 6 or later and Firefox 3.6 or later.

No issues have been reported when using Watson Explorer Engine with other modern standards-based browsers, such as Safari 3 or later, Opera 8 or later, and Google Chrome. Using the Watson Explorer Engine administration tool in a browser requires that browser support for JavaScript is enabled.

The Watson Explorer Engine administration tool is not designed to be used through a reverse proxy, and is not supported in that environment. However, the search interface can be used through a reverse proxy or load balancer, as long as the user is always directed to the same server during a session.

2.6 Application Builder architecture

Watson Explorer Application Builder is a framework for developing 360-degree information applications that extend the capabilities of Watson Explorer Engine by enabling users to quickly locate and use information that is specific to their role, interests, and capabilities within an organization. Application Builder adds a data relationship definition layer that supports logical data items, known as *entities*, which are associated with specific data sources or user data.

Entity definitions allow administrators to specify relationships between data in different repositories by identifying related or equivalent data fields in each repository. These are conceptually analogous to the foreign key fields used in traditional applications, such as databases.

The Application Builder framework is written in Ruby using the Ruby on Rails web framework. The application runs atop JRuby² to support direct interaction with Java APIs. This combination results in an environment well-suited for well-organized, widely-supportable, and extensible applications.

Applications created with Application Builder provide a graphical, web-based interface for finding and exploring relevant data, delivering different types of information in discrete portions of an application's window (known as *widgets*). Multiple widgets can be displayed and updated simultaneously, delivering relevant textual and graphical information.

² JRuby is an implementation of the Ruby language written in Java that runs on the JVM.

Figure 2-3 provides a high-level view of how Application Builder uses Watson Explorer Engine, centralized enterprise services (such as LDAP), and its own relationship mapping and definition capabilities to provide authenticated users with up-to-date, role-relevant information that respects existing authorization and data permissions.

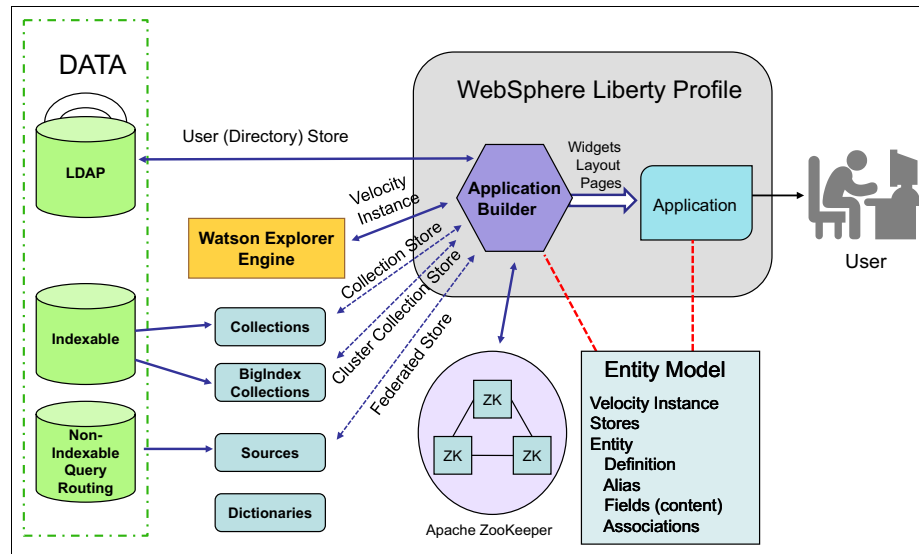


Figure 2-3 Watson Explorer Application Builder architecture

The next few sections provide more detailed information about how data is located, modeled, and displayed in Application Builder. These capabilities provide the foundations of the 360-degree, role-centric, and user-centric applications that can be created with Application Builder.

Subsequent sections of this chapter dive deeper into capabilities that are provided by other software:

- ▶ Apache ZooKeeper, which Application Builder uses to support 360-degree information applications in the enterprise context by synchronizing configuration
- ▶ Watson Explorer BigIndex API, which programmers can use to integrate Watson Explorer capabilities into new or existing applications

2.6.1 Modeling data and data relationships using entities

A fundamental problem in providing a unified, relevant view of information that is stored in multiple data repositories is that different repositories use different storage formats, data fields, record names, and security mechanisms.

Security mechanisms typically use repository-specific user names, group names, or their equivalents. Data repository security mechanisms typically provide both *authentication*, the mechanism that uniquely identifies a specific user, and *authorization*, which identifies the access rights that specific users have to the data in that repository.

Watson Explorer Engine minimizes issues that are traditionally found when simultaneously exploring heterogeneous data, by applying repository-specific authorization information when querying each repository. Associating each repository with its own authorization information is fairly straightforward, but limits data exploration across multiple repositories to parallel exploration of stand-alone repositories.

Improving the usability and value of enterprise data that is stored in different repositories requires mechanisms for both identifying relationships between authorized users and groups in different repositories, and identifying relationships between fields or other subsets of the data that is stored in the repositories.

Application Builder unifies the silos that different data repositories normally represent by adding an application configuration and data modeling layer known as an *entity model*. An Application Builder entity model is specific to each Application Builder application, and is a set of structured declarations that provide the following information for that application:

- Identify the different repositories that the application requires, or that can be accessed by an application.

Applications that are created using Application Builder often use standard external repositories, such as an LDAP repository that identifies the users of that application. Each data repository that an application uses is similarly defined in a collection store entity that provides a name for the index that is associated with that repository, defines how to locate and access that index, and so on.

In more advanced Application Builder applications that use the BigIndex API, each collection store entity definition is replaced by a cluster collection store definition that identifies the servers on which the index for that repository is stored, and how that index is stored. See 2.6.4, “Scaling and availability for enterprise applications” on page 32 for more detailed information about the BigIndex API and collection storage.

- Define the structure of the data in each repository.

Watson Explorer Engine uses equivalent information when indexing a data repository. Maintaining this information in a structured form in Application Builder provides a central integration point for all of the data that can be accessed in an Application Builder application.

- ▶ Define relationships between fields in different repositories, known as *associations*.

Associations can be made between fields whose content is an exact match, or can use products such as InfoSphere MDM to define and resolve inexact matches that are known as *fuzzy matches*.

An entity model therefore consists of some number of standard entities that are used by all Application Builder applications, and a larger set of entities that are specific to each application. Entity definitions identify the structure of the data that is used by each Application Builder application, and are typically used to identify 360-degree information application content:

- ▶ User data from an enterprise authorization source, such as Active Directory LDAP
- ▶ Product data from an index of a repository of product offering, catalogs, and so on
- ▶ Customer data from an index of sources, such as a CRM system and call center logs
- ▶ Real-time social media data such as comments in customer forums, Twitter channels, and so on

Each set of values that populate a built-in or application-specific entity definition is known as an *entity instance*.

Though stored in replicated network data sources by using ZooKeeper, an application's entity model can be exported as an XML file for backup or snapshot purposes. Example 2-1 shows a sample section of an XML entity model.

Example 2-1 Sample section of an entity model in XML

```
<velocity-instance password="*****" url="." username="joe-user">
  <serves name="books"/>
</velocity-instance>

<collection-store collection-name="example-metadata" name="books"/>

<entity-definition default-searchable="true" identifier="@hash"
  name="Book Title" store-name="books">
  <alias chain="" name="natural_associations" recommendable="false"/>
  <field external-name="author" name="author"/>
  <field external-name="hero" name="hero"/>
  <field external-name="publisher" name="publisher"/>
  <field external-name="title" name="title"/>
</entity-definition>
```

To make it easy to verify that an entity definition includes all of the field definitions and associations that are desirable to use in an application, the Application Builder administration tool includes an Entity Explorer component that is described in 2.6.3, “Overview of the Application Builder administration tool” on page 31.

For more information about where Application Builder applications store their entity model, see 2.6.4, “Scaling and availability for enterprise applications” on page 32.

2.6.2 Displaying information using pages and widgets

Application Builder applications use pages to display information associated with a specific type of entity (referred to as an *index page* for that entity type), or about a specific instance of an entity (referred to as a *show page*). Pages consist of one to three columns, each of which can contain zero or more widgets.

Each application that is created using Application Builder has a *home page*, which is the initial, user-centric page that is displayed after logging in to that application. The top of this page consists of a banner that includes a customizable graphic logo and provides convenient access to the search capabilities that have been defined for that application.

The results of any search submitted using the search box at the top of the page is displayed on a type of page known as a *search page*, which displays matching results in the center column and provides widgets in the left and right columns that make it easy to further refine the search results that are displayed.

As examples of widget content, Application Builder widgets can display relevant search results returned by exploring one or more data repositories, or can automatically display updates to mission-critical data sources. Application Builder includes several predefined widgets that one need only to populate the entities and associations that are relevant to the application.

Widgets are defined using a markup language known as YAML Ain’t Markup Language (simply called YAML). Each widget has a title bar that provides access to online help about that widget, and a standard widget display control that enables the user to expand or collapse a widget and the information that it contains to save window space. Widget-specific online help can be defined when creating widgets in the Application Builder administration tool.

Highly customized widgets can also be created using a general-purpose Embedded Ruby (ERB) widget that can be created in the Application Builder administration tool. ERB is essentially HTML with Ruby code embedded, and is evaluated by the Rails framework at run time to insert dynamic content. The ERB widget supports the use of custom Ruby code to produce content that is not available using the standard widget set.

2.6.3 Overview of the Application Builder administration tool

The Application Builder administration tool simplifies creating applications by providing a graphical front end for defining the resources required by an Application Builder application, and for specifying how the information in those resources is related, used, and displayed. This tool follows a progressive disclosure model, walking the application developer through different sections that help them build and validate applications:

- Back ends

Back ends identify the Watson Explorer Engine instances that support the Application Builder installation being configured, known as the *front end*. This section defines the Engine instances used.

- Entities

Entities model the structure of the data that is used in a 360-degree information application and identify the relationships between fields in different data repositories. This section controls the definition of entities.

- Pages and widgets

Pages and *widgets* meaningfully display entities and their associated data. This section controls the pages on which those entities are displayed and how entity information is displayed. The definition for each widget identifies the information that those widgets deliver, and sets widget-specific parameters.

Examples of per-widget parameters include the number of matching values that are displayed at one time within a widget, a string that displays if no entity instances that satisfy the content requirements for a widget are found, and so on. Page information enables selection and customization of the theme to be used throughout an application, how widgets are displayed on various pages, and what those widgets contain.

► Settings

Settings define search-related and display-related application options, such as the ways in which multiple entities can be searched (known as *search contexts*), whether capabilities such as query expansion are available in those searches, and the criteria for sorting search results and displaying selected search results in highlighted portions of a page, known as *spotlights*.

These different sections can be selected by clicking the header entry that contains their name. After back ends, entities, and pages and widgets have been defined, other options within these sections become available to enable customization and expansion of their content and how they are configured and displayed.

2.6.4 Scaling and availability for enterprise applications

The web-based nature of Application Builder applications liberates those applications from many of the limitations of traditional enterprise applications, such as local installation and system requirements, or only being available from a single server. The ease of access to web-based applications through any standards-compliant browser makes it even more important that the data resources that are required by web-based applications are always available.

Increased availability is usually facilitated by eliminating single points of failure that can prevent successful application execution and operation. Eliminating single points of failure is typically done by replicating application-specific configuration information and data on multiple systems on a network:

- Replicating data across multiple hosts provides opportunities for both reliability and performance improvements by distributing application load across multiple systems.
- Replicating configuration data across multiple systems eliminates configuration data as a single point of failure when balancing front-end application load across multiple systems.

Although data replication and front-end and back-end load balancing are common ways of ensuring that applications are always available, and can scale to satisfy increasing usage and numbers of users, enterprise applications must also support continually increasing amounts of data and associated storage.

Although Watson Explorer indexes are usually much smaller than the data repository for which they provide an index, using a flexible model for allocating and distributing the back-end storage that is associated with an Application Builder application eliminates many common storage problems. Using a flexible model also simplifies managing that data if it must be redistributed, expanded, or moved to new or additional storage devices or systems.

The next two sections provide an overview of how Application Builder addresses replicating configuration data and indexes to satisfy the requirements of today's enterprise applications, while building in the flexibility and scalability that is required to support the increasing demands of tomorrow.

Replicating configuration data using ZooKeeper

Application Builder applications store configuration data, such as their entity model, in a networked data repository known as ZooKeeper³.

ZooKeeper is an open source data repository that was designed to provide a highly-reliable, synchronized repository for data, such as configuration data that is required by large-scale distributed systems. Application-specific configuration data is stored in distinct namespaces, enabling single ZooKeeper installations to simultaneously support the configuration requirements of multiple applications.

ZooKeeper satisfies its RAS requirements by coordinating its content across multiple systems that are running a ZooKeeper server. These ZooKeeper instances must be associated with each other and support the same namespace for configuration data. A single ZooKeeper server is often used when developing Application Builder applications, but three or more ZooKeeper nodes are commonly used in actual deployments.

An odd number of ZooKeeper servers enables a set of related servers to use majority voting to select one of those nodes as an authoritative master node. Updates to any client server are synchronized with the master node, which then updates all other clients appropriately. Production environments often use at least five ZooKeeper nodes to account for system or network maintenance making one or more nodes unavailable without sacrificing availability or reliability.

Other software in the IBM big data portfolio uses ZooKeeper, such as BigInsights. Although both BigInsights and Watson Explorer use ZooKeeper, Watson Explorer solutions might not be able to use the same ZooKeeper ensemble as BigInsights. Therefore, maintain separate ensembles unless otherwise directed. Always check the system requirements and product guidance for version compatibility.

³ <http://zookeeper.apache.org>

Enterprise requirements and the BigIndex API

Data repository indexes and the mechanisms used to access them have historically been created manually, using the Watson Explorer Engine administration tool. Similarly, the distributed indexing capabilities of Watson Explorer Engine have usually been configured manually within that tool.

Being able to replicate and distribute indexes across multiple hosts is a fundamental requirement for the RAS expectations of enterprise applications, such as those created using Application Builder, but requiring manual intervention to do so is not scalable.

To enable the programmatic creation, distribution, and management of indexes and associated storage, Watson Explorer provides an enterprise-caliber API known as the BigIndex API.

This API hides the complexity that can be associated with creating indexes, querying those indexes or other content sources, and returning the results of those queries. The BigIndex API also hides the complexity of replicating Watson Explorer indexes across multiple servers, providing opportunities for automating load and storage balancing across participating servers, automating fault-tolerance, and porting indexes from one server to another.

BigIndex API functions that require these capabilities can simply use and configure them, rather than requiring thousands of hours to develop and test those capabilities for each application. Using the BigIndex API reduces code complexity and makes working with complex distributed systems as easy as working with a local client API.

To support enterprise application requirements, such as high availability and load balancing, indexes that are created using the BigIndex API are typically partitioned into several segments, known as *shards*, which are optionally replicated across the servers associated with a given index.

Segmenting indexes into shards also provides opportunities for reducing the footprint and memory requirements of an index on any given server by tuning the size of these shards and distributing them across larger numbers of servers.

The servers that are associated with an index created using the BigIndex API are identified in a cluster collection store entity in that application's entity model. This entity definition also includes attributes that specify the number of shards that are associated with that index, and the manner in which the index is used.

The BigIndex API was developed in conjunction with Application Builder, but makes it easy for other applications to incorporate the data exploration, navigation, and search capabilities that IBM Watson Explorer technologies provide. For more detailed information about the BigIndex API, see the documentation and example applications that are delivered as part of the Watson Explorer product.

2.7 Application Builder system requirements

The next two sections identify the requirements for installing and running Application Builder on a server. They also identify the requirements for client systems that need to access the Application Builder administration tool, 360-degree information applications that have been created using Application Builder, or both.

2.7.1 Application Builder server requirements

To install and use Watson Explorer Application Builder on a given system, that system must satisfy the following requirements:

- ▶ Supported 64-bit hardware. A 64-bit (AMD64 or Intel 64) x86 system with a 2 GHz or better processor. A minimum of 4 GB of memory is suggested.
- ▶ Supported 64-bit operating system. A 64-bit RHEL 5.3 Server or Advanced Platform distribution or later, RHEL 6.0 Server or later, equivalent 64-bit RHEL-based distributions, SLES 11.2 or greater, a 64-bit version of Microsoft Windows Server 2008 Datacenter and Enterprise editions, or Microsoft Windows Server 2012 Datacenter edition.

Watson Explorer Application Builder runs in the context of an application server, and includes a bundled version of IBM WebSphere Liberty Profile. Application Builder therefore does not require integration with a system web server, or that a system web server be installed and running on a server before installing Application Builder.

Applications that are created using Application Builder require access to one or more Watson Explorer Engine installations, and must also be able to contact a ZooKeeper server. Both Engine and ZooKeeper can be installed at the same time as Application Builder.

2.7.2 Application Builder client requirements

To access the Watson Explorer Application Builder administration tool, access a 360-degree information application that was developed using Application Builder, or both, from any computer system, that system must provide a supported web browser. Watson Explorer Application Builder fully supports Microsoft Internet Explorer 8 or later, and Firefox 17 or later.

No issues have been reported when using those product modules with other modern standards-based browsers, such as Safari 3 or later, Opera 8 or later, and Google Chrome. We suggest to always use the latest stable release of the a browser.



IBM InfoSphere Master Data Management overview

This chapter describes IBM InfoSphere Master Data Management (InfoSphere MDM), which is a portfolio of product offerings from IBM that offers pre-built domains.

The following topics are discussed:

- ▶ InfoSphere Master Data Management (MDM)
- ▶ IBM InfoSphere MDM Suite of products
- ▶ Architecture overview
- ▶ Use cases

3.1 InfoSphere Master Data Management (MDM)

IBM InfoSphere Master Data Management (InfoSphere MDM) is a portfolio of product offerings from IBM that caters to two different MDM domain styles: single domain and multi domain. InfoSphere MDM offers pre-built domains, such as customers, products, accounts, and custom domains.

With InfoSphere MDM, organizations create single, trusted views of data that increase the efficiency of processes and facilitate better decision making. By focusing on a unified “golden record” describing a domain-specific item such as a customer or product, the organizations are able to improve business results, lower costs, reduce risks and achieve the agility to meet current and future business needs.

3.2 IBM InfoSphere MDM Suite of products

This section briefly describes different product offering under InfoSphere MDM portfolio.

3.2.1 IBM InfoSphere Master Data Management

IBM InfoSphere Master Data Management is an industry-leading master data management (MDM) solution that supports multi-style and multi-domain master data management. InfoSphere MDM helps in creating a “golden view” of records for business analytics and other downstream operations where master data is key for success.

InfoSphere MDM supports probabilistic matching engine (PME) for BigInsights, which is real-time matching of master records at big data volumes in Hadoop (InfoSphere BigInsights).

InfoSphere MDM includes task key performance indicators (KPIs) to help monitor master data quality, evaluate data stewardship activities, and plan data stewardship resources

InfoSphere MDM supports four styles of master data management:

- ▶ Virtual
- ▶ Hybrid
- ▶ Physical
- ▶ Collaborative

This section covers the first three styles, while the next section covers the last.

Virtual

Virtual is a *registry* style of MDM where the system of record is the respective source system. Virtual MDM provides the linkage and duplicates across and with in the system. The matching and linking is achieved through statistical approach using probabilistic algorithms. Figure 3-1 illustrates the virtual MDM model.

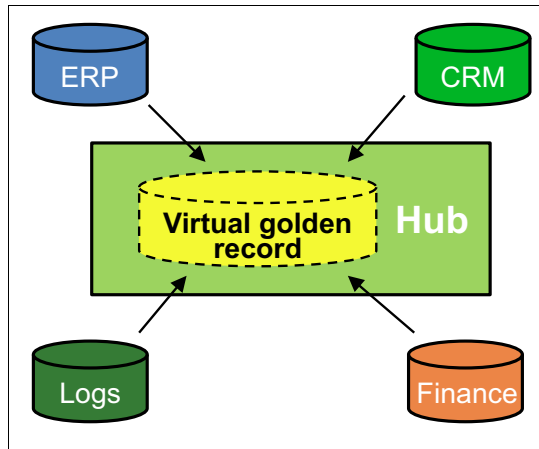


Figure 3-1 Virtual model

Physical

Physical is a *transactional* style of MDM where the golden record is persisted in MDM. MDM becomes the system of record. Physical MDM supports party (person or organization), product, and account domains by having predefined data model and services.

Figure 3-2 illustrates the physical model of MDM.

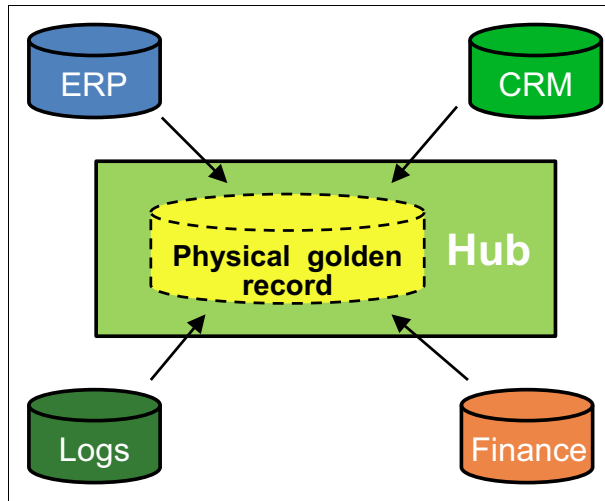


Figure 3-2 Physical model

Hybrid

Hybrid allows the seamless movement of master data from virtual to physical. *Only the golden view of interest is persisted from virtual to physical.* Figure 3-3 illustrates the hybrid model.

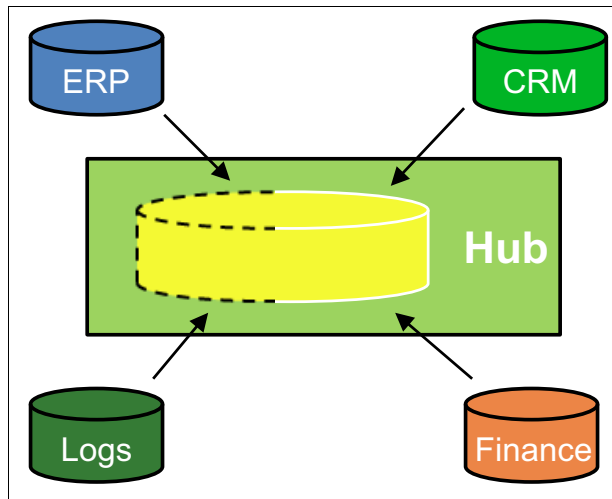


Figure 3-3 Hybrid model

3.2.2 InfoSphere MDM Collaboration Server

Collaborative MDM allows creation of a golden copy of product records. The InfoSphere MDM Collaboration Server offers the following capabilities:

- ▶ Flexible data model and workflows for creating and authoring products respectively
- ▶ Rich user interface, such as single- and multi-edit screens for data authoring
- ▶ Different types of search capabilities, such as attribute level to free form
- ▶ Rich set of domain specific APIs for all operations
- ▶ Integration with other products, such as WebSphere Commerce

3.2.3 InfoSphere MDM Reference Data Management Hub (RDM)

InfoSphere MDM Reference Data management (RDM) is built on InfoSphere Master Data Management. RDM enables definition and management of reference data at the enterprise level.

3.3 InfoSphere MDM general product features

This section outlines major features of InfoSphere MDM.

3.3.1 Matching and linking

InfoSphere MDM employs common matching technology across the entire portfolio. This technology uses statistical techniques to address data quality issues for matching and linking the data; by using probabilistic and deterministic approaches.

3.3.2 Search

InfoSphere MDM provides various types of search capability for data and entity searching.

Flex search

This is an exploratory search where the search happens on the data that is indexed. It supports various syntaxes such as ranges, wildcarding, and so on.

Probabilistic search

This search employs a statistical method to rank the records, which are returned based on the input threshold.

Field search

This is a field-based search that enables exact and wildcarded queries.

3.3.3 Security and audit

InfoSphere MDM supports both authentication and authorization. InfoSphere MDM uses Lightweight Directory Access Protocol (LDAP) for authentication. Authorization is at the transaction level, the service level, and the attribute level. InfoSphere MDM supports any standard LDAP and the integration is through WebSphere Application Server.

3.3.4 Extensibility

Various levels of extensions are possible within InfoSphere MDM.

Pre/Post

Hooks are provided at the framework where the customization can be added before and after the service call, sometimes referred to as callback handlers.

Behavioral

Behavioral extensions customize the service behavior. This can be hooked in either pre-transaction or post-transaction.

Data model

Different types of data model extensions are possible in InfoSphere MDM:-

Additions

Additions are the way to introduce new InfoSphere MDM entities. The new entities consist of creating altogether new servers or cluster of servers. The new entity can be an entirely new master data domain itself. This involves creation of new tables.

Extensions

Extensions are the way to add more attributes to an existing entity; this involves adding a few attributes to the existing table. Two types of extensions are possible:

- ▶ Inline, adding attributes to the same table
- ▶ Child table, adding a child table to the base entity

Dynamic attributes

This type of extension leverages XML schema and XML type in the database. Sometimes, a dynamic attribute is also referred to as a Soft attribute or Specification (Spec).

Adaptive Services Interface (ASI)

The Adaptive Services Interface (ASI) enables service customization for various organization needs. It enables a developer to map external XML services and data objects to InfoSphere MDM transactions and data objects using the Graphical Data Mapping (GDM) editor. The Adaptive Services Interface offers complex object-transaction mapping and data transformation.

3.3.5 Standardization

Standardization is one of the key things for achieving the good data quality standards and InfoSphere MDM supports standardization of data at various levels. Normally, data standardization process includes standardizing all alphabetic characters, removal of punctuation, checks for anonymous values, and ordering of the data.

InfoSphere MDM Advanced Edition also supports pluggable framework for adding a custom standardization.

3.3.6 Data load

InfoSphere MDM uses various data loading technique for loading the data into the portfolio.

Batch load

In batch load, InfoSphere MDM uses a customer-built utility such as bulk load and batch process to ingest the data into the MDM.

ETL based load

In the ETL based load, InfoSphere MDM uses the information server's data stage to load the data into the MDM Advanced Edition.

3.3.7 Software development kit (SDK)

The software development kit (SDK) includes a development kit for Java and SOAP web services to create custom applications and integrations.

Application programming Interface (API)

InfoSphere MDM comes with a rich set of APIs in the form of SDK, which offers a gateway to interact with the MDM system.

SOAP web services

InfoSphere MDM also offers standard SOAP web services to interact with the master data management server. An implementor can choose a different interaction gateway based on the deployment topology and choice of framework.

3.3.8 Stewardship and governance

Stewardship and governance is a user interface for manually merging, un-merging, authoring, and editing a golden record.

3.3.9 Business process management

Business process management provides capabilities for enabling human and machine intervention for multi-step and multi-role workflow.

3.3.10 MDM Workbench

MDM Workbench is tooling for InfoSphere MDM. It enables product customization. The tooling helps in algorithm development, extensions, additions, and query extensions. This workbench is a plug-in to IBM Rational® Application Developer (RAD) and Rational Software Architect (RSA).

3.4 Architecture overview

The general architecture diagrams (Figure 3-4 on page 46, Figure 3-5 on page 47, Figure 3-6 on page 48, and Figure 3-7 on page 49) consist of sections that are divided into major areas as follows:

- ▶ MDM consumers

The consumers include end-point applications such as user interfaces, business process management, toolkits, and widgets, which help in building user interfaces. Some are specific InfoSphere MDM-shipped, ready-to-use applications. Others are third-party applications.

- ▶ Core interfaces

The core interface consists of gateways to interact with the MDM applications such as SKD, REST APIs, web services, and message brokers.

- ▶ Core

Core, as the name says, is the core of MDM applications. Sometimes, it is called the engine or core framework. The core generally highlights the engine operations and domain models.

- ▶ IBM integrations

This area depicts all the integrations around IBM products that have some synergy with MDM.

- ▶ Tooling, toolkits, and other components

This area highlights the tooling offered by the MDM Workbench, which enables the rapid development of additions, extensions, and algorithm development. Workbench also helps in deploying the modified packages into the MDM engine (core). A toolkit is generally a set of widgets that help in user interface development.

- ▶ Other components

Other components include a variety of important aspects that are needed in an MDM environment, such as event management framework, utilities, auditing, caching, and so on.

3.4.1 InfoSphere MDM Standard Edition

Figure 3-4 illustrates the logical component architecture overview of the InfoSphere MDM Standard Edition (InfoSphere MDM SE). The MDM SE Core highlights key interactions that are specific to InfoSphere MDM Standard Edition.

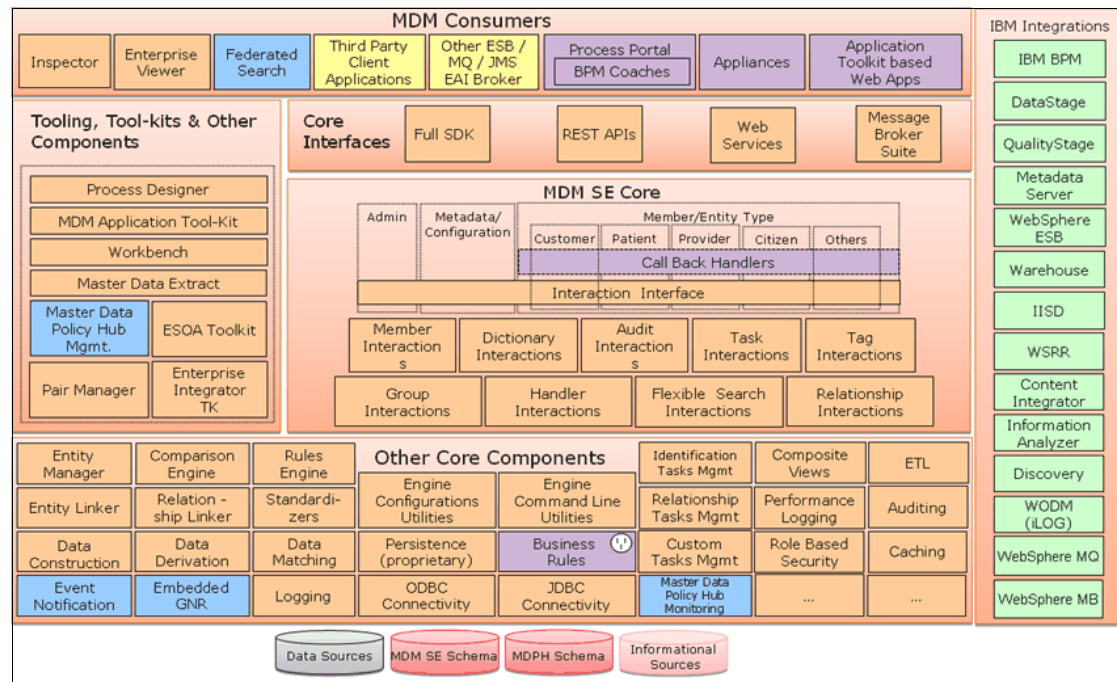


Figure 3-4 Standard edition architecture overview

3.4.2 InfoSphere MDM Physical/ Advanced Edition architecture

Figure 3-5 illustrates the logical component architecture overview of the InfoSphere MDM Advanced Edition. The MDM Core highlights domain specific services of InfoSphere MDM Advanced Edition.

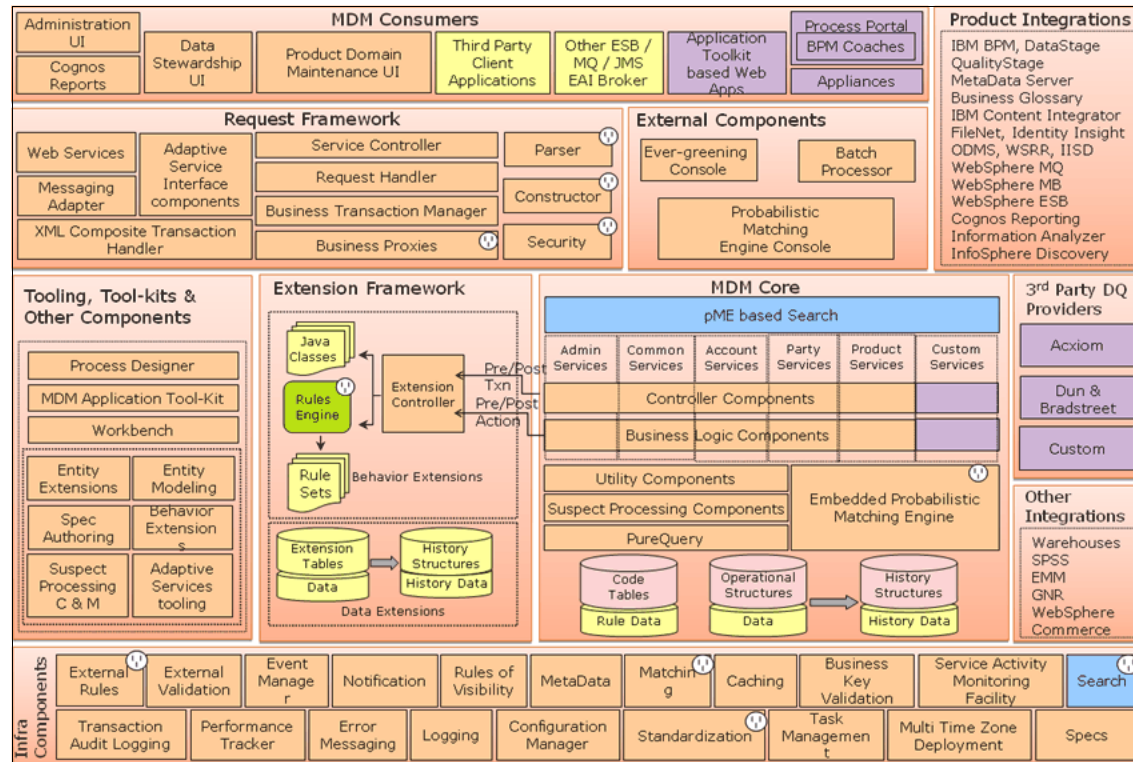


Figure 3-5 Advanced edition architecture overview

3.4.3 InfoSphere MDM Collaborative Edition architecture

Figure 3-6 illustrates the logical component architecture overview of the InfoSphere MDM Advanced Edition. The MDM CE Core highlights specific and key components of InfoSphere MDM Collaborative Edition.

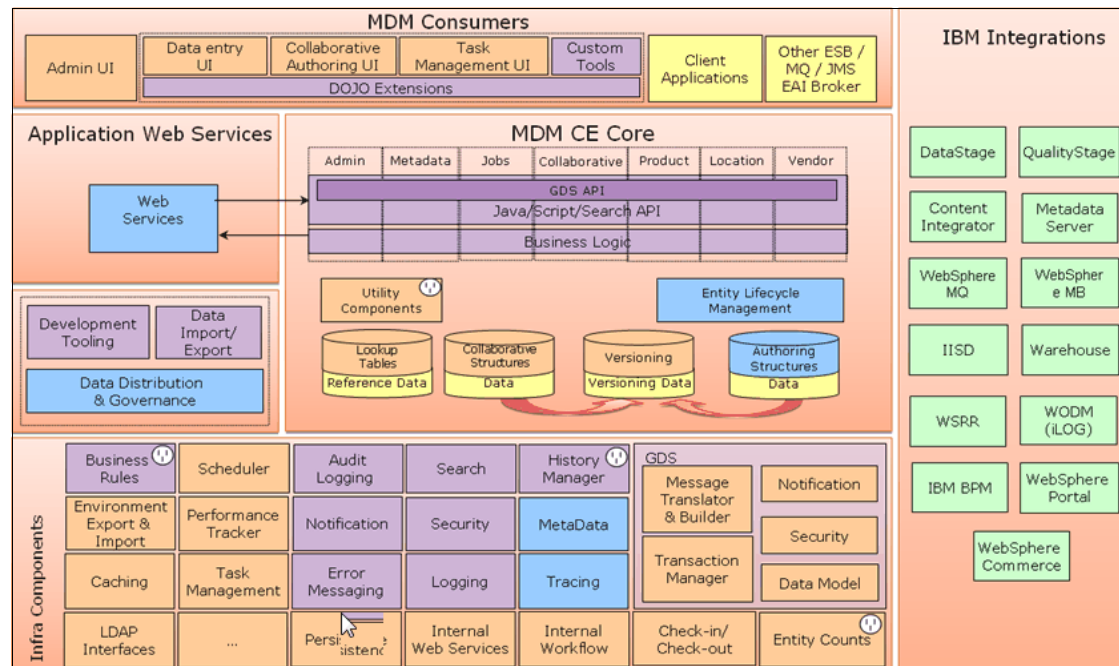


Figure 3-6 Collaborative edition architecture overview

3.4.4 InfoSphere MDM Reference Data Management architecture

Figure 3-7 illustrates the logical component architecture overview of the InfoSphere MDM Reference Data Management Edition.

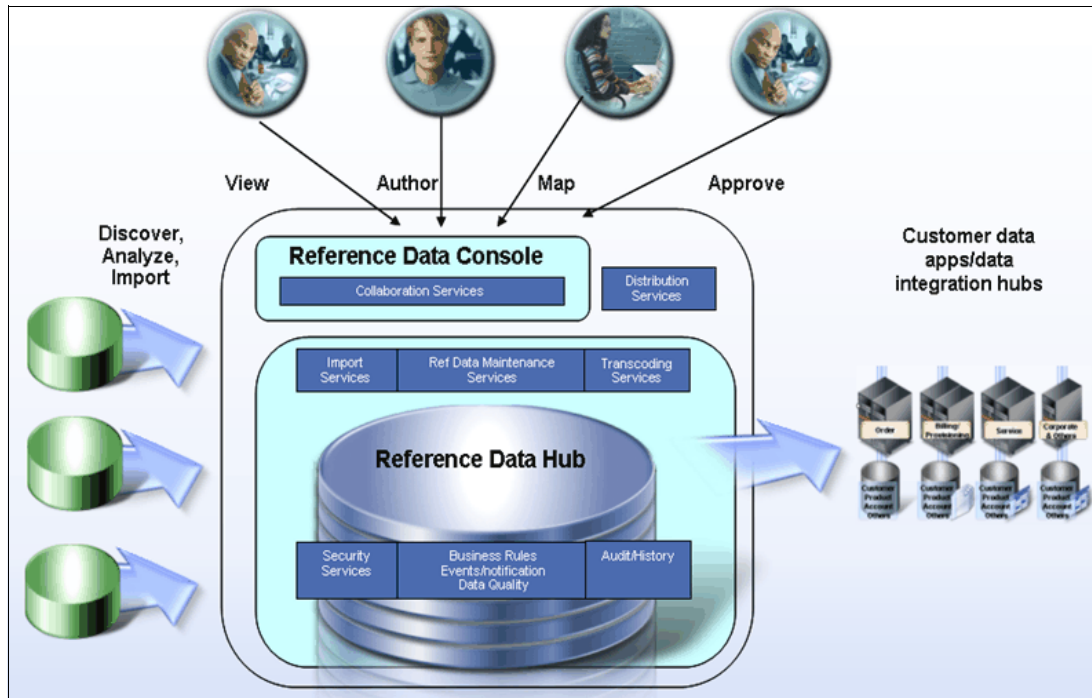


Figure 3-7 Reference data management edition architecture overview

3.5 Use cases

This section describes the additional value that IBM InfoSphere Master Data Management customers gain by capitalizing on Watson Explorer. It is broken into sections, and covers a particular industry in each section.

3.5.1 Government

The government industry is divided into two functional areas:

Citizen services	Citizen services often deal with state, local, and provincial governments, and the services that they provide.
Public safety	Public safety consists of law enforcement, military, and intelligence customers who are tasked to ensure security and stability, while also being aware of threats and preventing crime.

This section describes the use cases in citizen services and public safety.

Citizen services

Government agencies face demands from taxpayers to provide access to information in an intelligent, citizen-centric manner. For instance, someone involved in a home-remodeling project does not want to provide the same information first to the local planning commission and then to the permit department. Those departments should be able to access each other's information seamlessly.

In other words, citizens expect the same level of integrated service from government agencies as they receive from top-rated service companies.

By integrating Watson Explorer with MDM, governments are empowered with a complete, citizen-centric, e-Government application solution (Figure 3-8 on page 51). InfoSphere MDM provides government services the agility and flexibility to receive and share information rapidly. With the addition of Watson Explorer, they provide access to a current, accurate, and complete view of citizen and business information, and are able to respond to constituent needs in real time.

Both governments and citizens save time and money, and increase taxpayer satisfaction.

Citizens gain the following advantages:

- ▶ Access government information and services easily.
- ▶ Provide instantaneous feedback on issues and topics.
- ▶ Engage in dialog with officials.

Seamless interoperability exists across agencies and departments, enabling officials to accomplish the following improvements:

- ▶ Share and view the same information about individuals.
- ▶ Reduce administrative costs.
- ▶ Increase efficiencies and service levels.

Citizen Transaction History from MDM

Citizen info from MDM

Unstructured Internal information related to customer

Healthcare Information

Location	Date/Time	Clinic
DAYT29 TEST LAB	13 May 2012 @ 1100	C&P CHRISTIE
DAYT29 TEST LAB	17 June 2012 @ 1300	DIABETIC-BURKE
Pre-appointment activity:	17 June 2012 @ 0800	LAB
DAYT29 TEST LAB	26 July 2012 @ 1600	GERIATRICS
DAYT29 TEST LAB	2 Aug 2012 @ 1300	C&P CHRISTIE

Location

Reston, VA 20190 USA

Military Service History

Service	Start	End	Rank	Character
Army	06/11/1968	03/26/1976	COL	Honorable
Army	02/22/1980	02/21/1984	COL	Honorable
Army	01/11/1987	08/24/1993	COL	Unknown
Army	25/2004		COL	Unknown

Identifiers

EDIPI: [REDACTED]

Customer Type: Veteran

Status: Authenticated

Personal Information

First Name: George

Middle Initial: R

Last Name: H

Suffix: Senior

Date of Birth: 01 Mar

Gender: Male

Marital Status: Married

Occupation: Retired

Primary Contact Information

Address: 45 [REDACTED]

Wellness Reminders

Reminder	Date Due
Prostate Screening	DUE NOW
Colon Cancer Screening	DUE NOW
Influenza Vaccine	30 Oct 2012

Claims

Claim	Status
Doctor Wellness Visit	Applied
VA Hospital Visit	Protesting
VGI Policy	Active
Compensation	Active
Education	

Comments

Theodore G. Comment: "Col. H is retiring to Florida and wants to update his new address and information. He also wants info on VA treatment facilities in the new area. So as not to confuse appointments he is leaving address as is for now." CXO - Minutes ago

Figure 3-8 Example of citizen-centric application capitalizing on InfoSphere MDM and Watson Explorer

Law enforcement

In the past, as much as 95 percent of criminal activity was local, but times have changed. Criminals are now more mobile, using technology to reach beyond the usual geographical boundaries. To fight crime effectively, law enforcement officials must also use technology in new ways. Information sharing across jurisdictions can help law enforcement agencies get ahead of criminals and find them regardless of where they operate.

However, effective information sharing requires that the data coming into or provided by an agency is accurate, and is kept secure throughout the sharing process. It also assumes that the information one agency shares with another can still be controlled by the providing agency. Without these conditions, information cannot be shared effectively across agencies, and data silos will persist.

Information about a single suspect often exists in multiple systems within an organization. To ensure that officials have all of the relevant, updated information that they need while pursuing a case, law enforcement agencies must create a single, consolidated view of each entity associated with an investigation. MDM provides a consolidated view of a suspect from multiple sources, in real time.

MDM analyzes vast amounts of information from a variety of sources, including criminal reports, arrest records, prison reports, and court filings. MDM models the information by creating a multi-entity view that can include any of the following entities:

- ▶ Persons
- ▶ Property (weapons, automobiles, mobile phones, and so on)
- ▶ Locations
- ▶ Events (such as robberies, murders, assaults, and moving violations)

It is then able to find relationships within and among this information by identifying patterns or commonalities.

By integrating Watson Explorer with MDM, law enforcement can further empower their officers and investigators by incorporating additional unstructured data and social data into the information already provided by MDM. This creates an investigation-centric law enforcement application.

MDM helps organizations accomplish the following tasks:

- ▶ Identify suspects.
- ▶ Track suspects.
- ▶ Improve cross-jurisdictional cooperation.
- ▶ Control how much data is shared, and with whom.

It enables sensitive investigations to be kept private, but still enables investigators to discover leads in other jurisdictions.

By bringing in new untapped sources, Watson Explorer can trigger new leads through social media feeds, such as Twitter and Facebook. With this lead information, investigators can reach out earlier and close more cases, faster, and with fewer resources.

3.5.2 Financial services

The financial services industry often refers to both banking and insurance institutions. This section describes the master data management needs of banking and insurance institutions.

Banking

Banks focus on building the customer relationship, yet few are able to deliver customer insight upon which action can be taken to all channels, and therefore transform operational processes. Customer and product information is still fragmented across multiple systems.

Without an authoritative master record of their customers and products, banks might struggle to provide consistent service across all channels to sell the most appropriate products to customers, or to reduce operating expenses.

InfoSphere MDM enables banks to execute on this strategic vision by helping to make high-value core information available across the enterprise to support critical business processes.

Customers want (and have been given) more ways to interact with their banks. Individuals no longer interact solely with a single local branch. They also call and do business on the web and mobile applications.

Banks must ensure that local branches, call-center customer representatives, and online systems can access all customer information, collected through all previous interactions. This provides a seamless experience for customers, reduce attrition, and drive new sales.

MDM enables banks to implement cross-channel integration, and match the channel to the individual. For example, a client with few accounts and lower balances might prefer to use the self-service channel to avoid additional fees, while another customer with a complex financial situation might be willing to pay a higher price for service if they can speak to a service representative in person.

MDM also helps banks to recognize customers across multiple systems, and to provide customers with a complete view of their assets across channels. The result is a consistent experience, whether the customer is banking at the branch, online, or by mobile phone. At the same time, banks can use MDM to help reduce costs, streamline customer information processing, simplify client on-boarding and new account setup, and eliminate duplicate mailings.

MDM makes it easy to manage a customer's privacy across channels, and generate accurate and timely reports for regulatory compliance. Just centrally managing the hundreds of reference tables associated with regulations might help a bank save millions of dollars, significantly improve data quality, and reduce by weeks the time required to complete all of the necessary reports.

Because it helps companies consolidate information, MDM can also streamline business processes. One financial institution wanted an aggregate customer view to enable new marketing strategies and increase the number of products owned per customer.

This company deployed MDM to integrate data from eight business systems, to develop a more accurate view of its customers. The project almost immediately increased the marketing effectiveness of its rewards programs and its account penetration.

Integrating MDM with Watson Explorer can further enable this institution to capitalize on internal and external documents and social data. This further enhances its marketing effectiveness by making additional information about the customer available to client representatives, as shown in Figure 3-9 on page 55.

Customer search

Janet R

Logged in as Frank | Help

Purchase History

Date	Amount	Fund
2011-04	23k	U.S. Treasury
2011-01	25k	Short Duration
2010-10	18k	Dividend Value
2010-07	10k	Batterymarch S&P 500 Index Fund
2010-04	11k	ClearBridge Large Cap Growth Fund

Owned Products

- Bond Fund
- U.S. Treasury Reserves
- Short Duration Municipal Income Fund
- Dividend Value Fund
- New York Municipal Money Market Fund
- Oregon Municipals Fund
- Special Equity Fund
- New Jersey Municipals Fund

Personal Information

Janet R
 Senior Regional Sales Manager
 Department: Sales
 Office: Pittsburgh, PA
 jr@vanguard.com
 412.422.2499 x555

Stop Tracking

Associated Accounts

Investments

LinkedIn History

Investments
 Title: Senior Regional Sales Manager
 Years worked: Nov 2006 - Present

Regional Sales Manager
 Title: Regional Sales Manager
 Years worked: May 2000 - Nov 2006

Regional Sales Manager
 Title: Regional Sales Manager
 Years worked: Nov 1993 - May 2000
 Oppenheimer & Co./CIBC

Contact Activity Feed

Showing: All Activity | By Source | By Author | Filter Feed

What's new with this contact?

Irene T updated the document SAP - 1 days ago

Todd W updated the database Product Funds #322245 - 3 hours ago

Frank G commented to Janet R: "Scheduled a meeting with Janet for next week. I'll let everyone know how it goes!" CXO - 3 days ago

Chelle K commented to Janet R: "Investments has a new rep that we'll be working with. I've just added her to Salesforce. Frank, you should set up a meeting with her as soon as possible." - 3 days ago

Recent Conversations

Email: Fund Management Overview
 Janet, please find an overview of [redacted] and the funds we offer. This is only a high level...
 Exchange - 3 hours ago

Notes: Janet is new to [redacted] but has been a wholesaler for 10 years. New to [redacted] Funds.
 Salesforce - 2 days ago

Title: Introduction
 Welcome Janet to [redacted]. I wanted to introduce myself...
 Salesforce - 2 days ago

Email: Team, I just spoke to [redacted] and Jeffrey has left. Janet R is replacing him and I will...
 Exchange - 3 days ago

Figure 3-9 Example of a client representative window when looking up or engaging with a customer

Insurance

To attract and retain customers, insurance companies must put customers first. Building strong relationships and understanding the needs of individual customers (and households) is essential for reducing churn and increasing the number of insurance products carried by each customer.

Insurance companies also must keep up with the changing communication and buying preferences of their customers. Customers want (and have been given) more ways to interact with insurance companies. Insurance companies must ensure that local agents, call-center agents, and online systems can access all customer information, collected through all previous interactions, to provide a seamless experience for customers, reduce attrition, and drive new sales.

Unfortunately, at many insurance companies, customer information is scattered across multiple, disparate systems. Information pertaining to different types of policies and collected through different communication channels is disconnected. The scattering of information will only increase as insurance companies add new lines of business:

- ▶ New types of insurance policies
- ▶ Internet banking services
- ▶ Mobile self-service applications

Without bringing together that customer information into a single, 360-degree view of the customer, insurance company representatives miss important opportunities to strengthen customer relationships, and to sell more products during customer interactions.

For example, if a customer contacts a call-center agent to discuss property insurance, that agent might have little to no information about the customer's previous in-person interaction with a local agent. The customer will have to spend time repeating information already provided. The agent might also be unaware that the caller's spouse was recently investigating life insurance online, and might be interested in additional follow-up information.

InfoSphere MDM brings together customer information from across policy lines, business channels, and service locations, creating a single, centralized record for each customer to help improve business processes and application efficiency. That single record can be any of the following types:

- ▶ A virtual record, in which changes to customer information continue to be made through individual policy-based systems
- ▶ A physical record, in which the record is maintained in a centralized hub
- ▶ A hybrid record, in which only part of the record is centralized

In this way, InfoSphere MDM gives insurance companies the flexibility to manage customer data in the way that makes the most sense for their business.

For example, a call-center property insurance agent could access and act on information collected through a life insurance query submitted online by the caller's spouse. That agent could also offer auto insurance quotes for a child nearing driving age.

With a complete view of the household, the agent can provide more personalized service in real time, while offering additional services and multiple-policy discounts. Ultimately, a complete view of each household can help agents increase the average number of products that households buy, which could have a tremendous effect on overall revenues.

Now, take the previous example a step further by using Watson Explorer with MDM in a customer-centric application. By adding Watson Explorer, the insurance agent could see information held about the household and policy information. In addition, the agent could integrate that information into their view with past emails, documents, and social media interaction between the caller and the life insurance policy.

From this, the agent might see that the caller is planning a big trip for an anniversary and be able to strike up a conversation about travel and offer discounted travel insurance.

3.5.3 Healthcare

Creating a single view of each patient can help healthcare organizations realize the following improvements:

- ▶ Facilitate collaboration among healthcare providers.
- ▶ Reduce medical errors.
- ▶ Provide a consistent patient experience across multiple touch points.
- ▶ Enhance the efficiency of patient processes.

Integrating provider information from multiple disparate systems helps ensure efficient routing of patient information among caregivers, to foster collaboration. To do this, organizations must create an up-to-date registry of individual providers and provider facilities that draws on a wide range of internal and external data sources.

InfoSphere MDM helps organizations improve the quality, accuracy, and completeness of critical records through effective management of master patient and provider data. Healthcare organizations can select MDM to rapidly master patient or provider data using a registry (or *virtual*) approach that does not require changes in source systems.

Although MDM can provide virtual capabilities, it also provides a means to extend those virtual capabilities to create or migrate to a physical, centralized repository. This enables organizations to strategically support health information initiatives that require centralized data management.

Effective management of patient and provider information can deliver immediate benefits and a rapid return on investment (ROI). With a single, accurate view of each patient, organizations can establish a patient-centric approach that boosts the quality of care, streamlines processes, and improves patient satisfaction.

For example, they can achieve the following improvements:

- ▶ Make patient information readily available to healthcare providers at the point of care to accelerate treatment.
- ▶ Improve provider collaboration.
- ▶ Eliminate duplication of efforts.
- ▶ Avoid safety issues that can result from incomplete or incorrect patient information.

At the same time, organizations can reduce administrative costs and complexity while decreasing patient frustration with administrative processes.

Building an accurate, up-to-date provider directory results in similar contributions to improving the quality and efficiency of care. Organizations can swiftly route lab results, imaging studies, and continuity of care documents (CCDs) to the correct provider, at the correct location, in a format that works for both electronic and paper-based practices.

In addition, organizations can enhance the efficiency of claims processing and billing with the assurance that each provider's accurate contact, credentials, and licensing information is readily available.

With a foundation of high-quality, accurate data, healthcare organizations can implement solutions that increase the meaningful use of this valuable data to the benefit of patients, providers, and the organizations. In addition, data warehouse and analytics capabilities applied to patient and provider information can help organizations identify factors contributing to successful patient outcomes, so that they can improve treatment effectiveness.

Producing accurate performance metrics, meanwhile, supports transformations into accountable care organizations, which are reimbursed according to measured quality and efficiency of care.

Building on this foundation, Watson Explorer can capitalize on the data in MDM to provide a portal for patients and doctors, and then use it to integrate healthcare data not stored or mastered in the patient hub. Figure 3-10 shows an example of how Watson Explorer can pull in information collected in MDM (patient detail, visit history, and upcoming appointments), and also integrate additional information for blog posting and internal message feeds.

The screenshot displays a patient profile for George. The interface includes a top navigation bar with a search box and a 'Logged in as Eric' status. The main content area is divided into several sections:

- Healthcare Information:** A table listing visits to DAYT29 TEST LAB and C&P CHRISTIE, including dates and times. An annotation 'Citizen Transaction History from MDM' points to this section.
- Location:** A map showing the patient's location in Reston, VA.
- Identifiers:** Fields for EDIPI, Customer Type (Veteran), and Status (Authenticated).
- Military Service History:** A table listing service in the Army from 1968 to 2004.
- Personal Information:** Fields for First Name (George), Middle Initial (R), Last Name (H), Suffix (Senior), Date of Birth (01 Mar), Gender (Male), Marital Status (Married), and Occupation (Retired). An annotation 'Citizen info from MDM' points to this section.
- Wellness Reminders:** A list of reminders for Prostate Screening, Colon Cancer Screening, and Influenza Vaccine.
- Claims:** A list of claims for Doctor Wellness Visit, VA Hospital Visit, and VGLT Policy.
- Feed:** A section for updates and comments. A comment from Theodore G. is visible, mentioning retirement and a move to Florida. An annotation 'Unstructured Internal information related to customer' points to this comment.

Figure 3-10 Example of healthcare patient window capitalizing on both MDM and Watson Explorer information

Figure 3-11 is another example of integration between MDM and Watson Explorer that shows how an administrator-centric application can be driven by the same data. Rather than being centered around a particular patient, this application is centered around the effectiveness of the facility and its services, enabling administrators to uncover problems within the delivery chain.

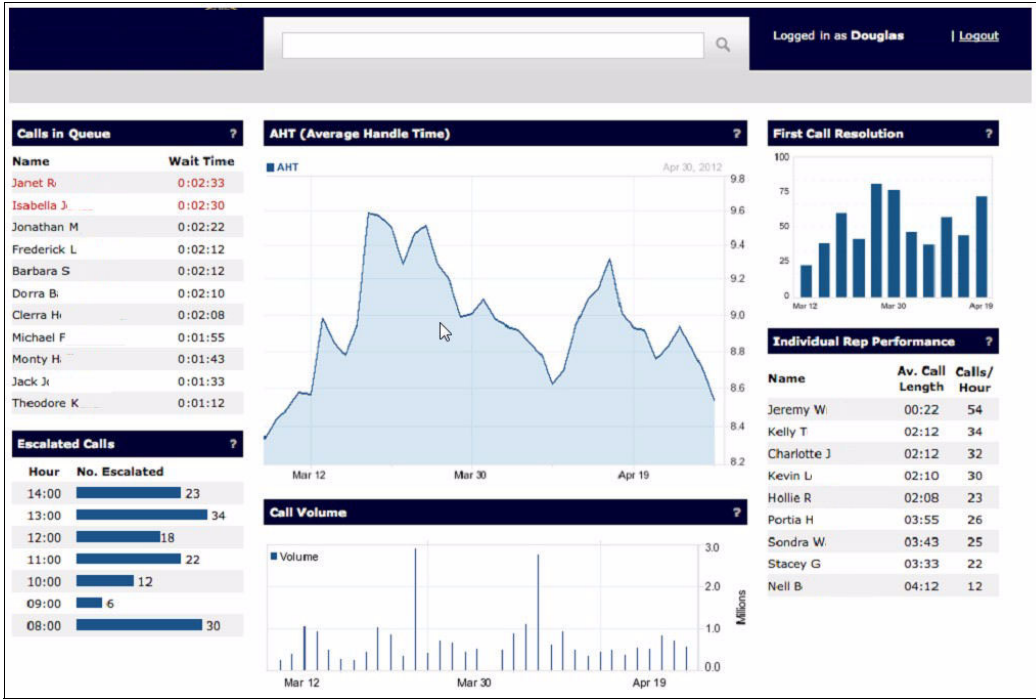


Figure 3-11 Example of an administrator-centric healthcare application

3.5.4 Retail

Master data (including customer, account, and product data, and information about other master domains) is the lifeblood of retail organizations. Without a way to manage master data across myriad sources, organizations are left with outdated, incomplete, inaccurate, and unsynchronized information.

An organization's critical business processes depend on master data. Without complete, accurate, and trusted master data, those processes are compromised and become inefficient.

InfoSphere MDM is designed to help organizations address the root causes of master data complexity, including the ways in which data is created, used, updated, managed, and analyzed. With MDM, organizations can centralize and manage critical data, creating a single, trusted source of information and a single view of each customer.

MDM enables organizations to manage all of the master domains that they choose to harmonize for use in business operations. It also provides a full range of MDM functions, including adding, deleting, updating, validating, and securing access to the master data.

Customer domain

By integrating MDM with Watson Explorer into a customer-centric application, bringing together customer information from a broad array of communication channels (including social media sources), retailers can build a single, 360-degree view of each customer. Retailers can use that single customer view to create better-targeted, personalized marketing campaigns and promotions that are tightly tied to customer preferences and behaviors.

For example, an online bookseller might create a personalized promotion using the categories of books that a customer has recently bought or shown interest in, as shared by the customer on a social networking site.

This detailed, trusted view enhances the effectiveness of cross-sell and up-sell efforts as well. With a complete record of a customer's purchases and searches, retailers can generate accurate recommendations for additional products while a purchase is in progress, potentially increasing market-basket totals.

Creating a single customer record, and capitalizing on Watson Explorer to integrate social media and other internal and external data, helps retailers reduce expenses and better target marketing campaigns. By eliminating the duplicate records that add significant costs to mailings and other marketing efforts, and by creating personalized campaigns, retailers can enhance the efficiency of their sales efforts and improve success rates.

Figure 3-12 shows an example of retail implementation of Watson Explorer.

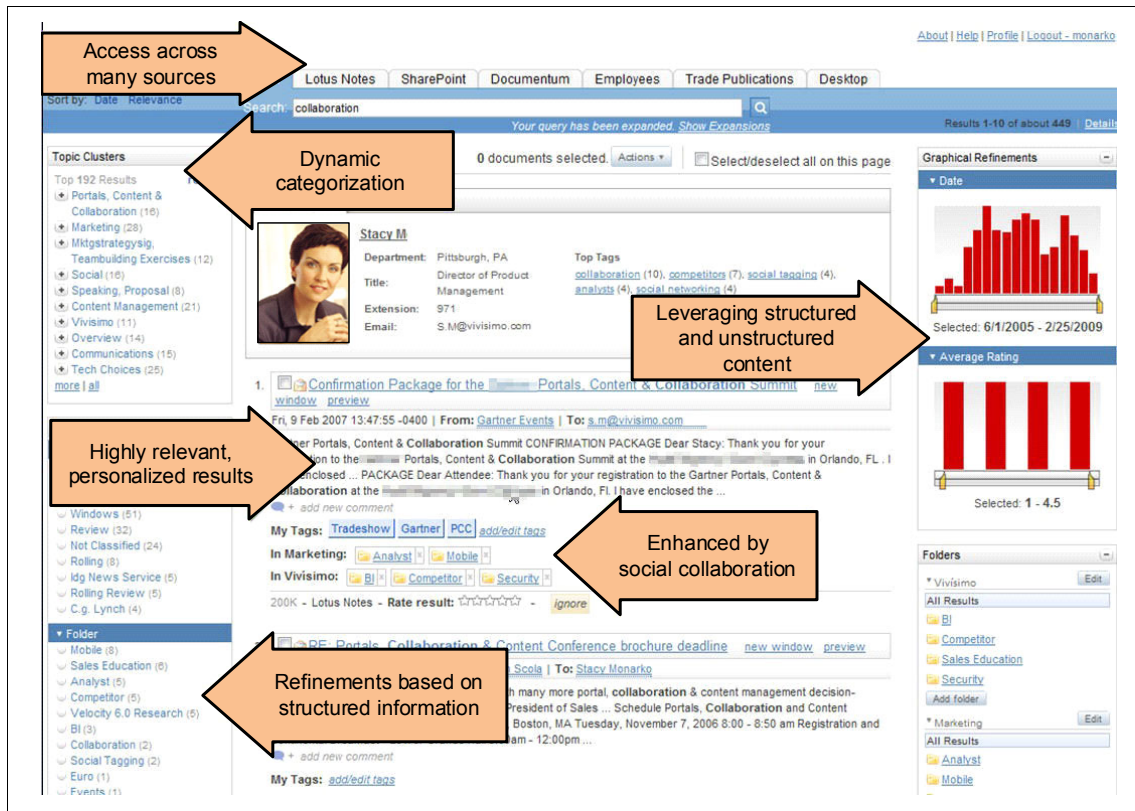


Figure 3-12 Retail example of capitalizing on Watson Explorer with InfoSphere MDM

Product domain

A single, centralized source for product information is critical for providing customers with the correct information, where, when, and how they want it. Customers buying a television, for example, should see the same product information whether they are accessing it in any of the following ways:

- ▶ Using a manufacturer's website
- ▶ Visiting a physical store
- ▶ Comparing prices on a smartphone
- ▶ Shopping on a retail e-commerce site

Delivering reliable information across channels helps produce a positive retail experience, which inspires confidence and loyalty, and ultimately leads to purchases.

Centralizing product information also helps streamline internal processes. A retailer might need to accomplish the following tasks:

- ▶ Collect product information from a foreign manufacturer.
- ▶ Translate that information.
- ▶ Deliver it to sales and marketing teams.

That process can be time-consuming, and might lead to errors as information is repeatedly entered and re-entered into systems. Using MDM, the retailer can create a single, trusted source of product information, and make it available to everyone inside and outside the organization, formatted appropriately for a variety of needs. With a single source of information, the retailer saves time and money, and reduces the likelihood of errors.

By creating a single view of the product, and by capitalizing on Watson Explorer to integrate social media, transactional data, and MDM data together into a single product-centric view, retailers are able to take advantage of the power that the unified product data provides.

The example shown in Figure 3-13 demonstrates how Watson Explorer accomplishes the following actions:

- ▶ Extracts product information from MDM
- ▶ Integrates it with information about transaction history and returns from SAP
- ▶ Provides that integrated information to the users
- ▶ Creates emails regarding the product
- ▶ Creates graphs showing the history of the product sales

This helps the product owner to uncover product issues that could lead to a negative effect on sales.

Product data integration is shown in Figure 3-13.

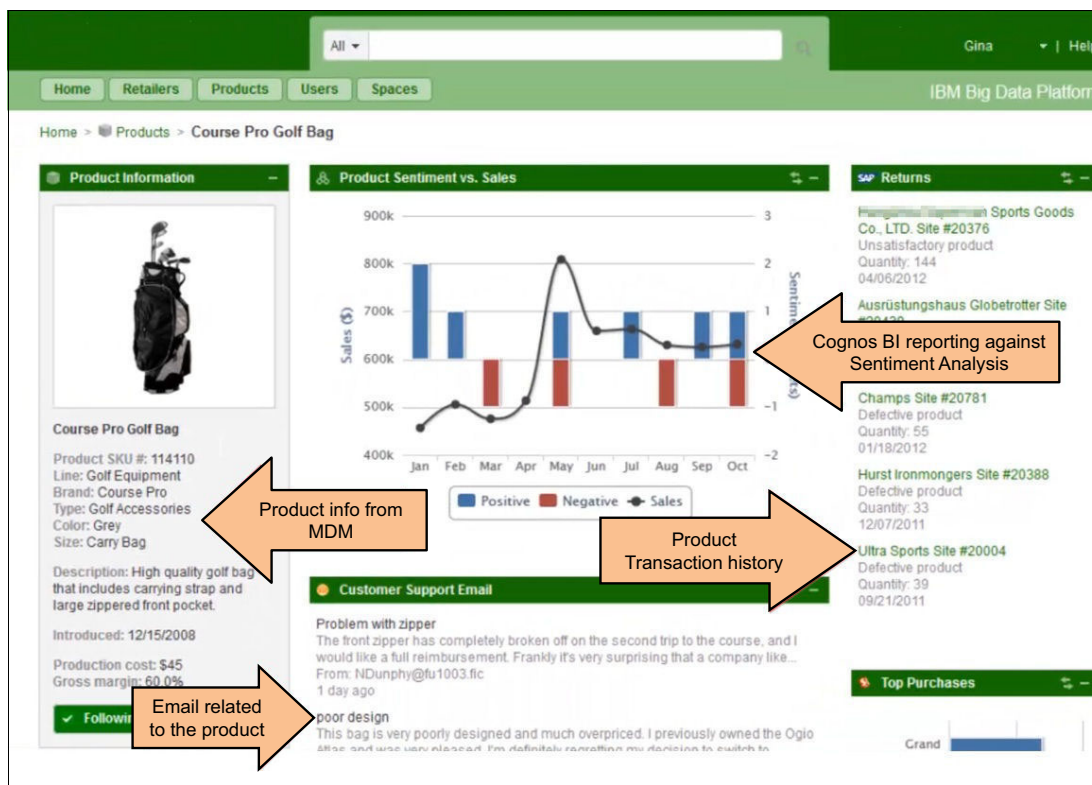



Figure 3-13 Example of combining InfoSphere Watson Explorer and InfoSphere MDM product data



Planning a 360-degree information application

Though 360-degree applications differ based on their users and the use cases that they are designed to address, the initial planning and design phases of their development are usually quite similar.

This chapter describes the steps that an organization should take when planning a 360-degree information application:

- ▶ Identifying use cases and users
- ▶ Identifying data sources
- ▶ Modeling entity relationships
- ▶ Planning for deployment and implementation

4.1 Identifying use cases and users

Understanding the unique capabilities of a 360-degree information application is critical to understanding the use cases that an organization can address with such an application. 360-degree applications provide users with a complete view of the data throughout an organization that is relevant to their role in that organization, regardless of the data repository or format in which that data is stored.

When planning a 360-degree application, the first question to ask is therefore “Who are your users and why do they need a 360-degree view?” The answer to this question typically drives all of the other requirements.

For example, suppose that you are the manager of the call center for a telecommunication (telecom) company. Your call center representatives need to find product information quickly to reduce the average handling time of calls, and to reduce repeat calls.

What type of information are they typically looking for? They will need specific information about the caller, and will also need information about common troubleshooting activities, different types of phones, the different wireless plans that are available, guidance about buying accessories for their phones, and so on.

All of this information is probably available, though it might be strewn across multiple repositories, and you might even have a distinct repository for each type of data. This is a key point. A 360-degree view does not create data for you, or make you change the existing ways in which that data is collected. Your data already exists in your repositories, and is already collected by your current user workflow.

The challenge is typically that a call center representative might need to click through multiple screens, or even use multiple applications, to find the answer he or she needs. This is because the associated data is scattered across different repositories.

This traditional model, where different applications or application screens are the only way to access different types of data, makes it hard for an agent to quickly locate and go to the information that he or she needs when servicing a customer on the phone.

What if you could have a unified view of this data and your representatives did not need to go from repository to repository searching for it? What if they could instantly see a single view of all of the information that is relevant to the customer that they are speaking with?

Customer information (from one repository) would appear beside troubleshooting activities (from a different repository) that the representatives might need to suggest, based on information about the handset that the customer owns (from yet another repository).

If call center representatives are incentivized to up-sell callers, you might also want to display other content, all of which might be stored in other repositories:

- ▶ Suggestions for wireless plan upgrades
- ▶ Information about new service plans that are being promoted
- ▶ Recommendations for complimentary accessory purchases

Figure 4-1 shows a sample application, created using IBM Watson Explorer Application Builder, that consolidates and centralizes relevant information for use when interacting with specific customers.

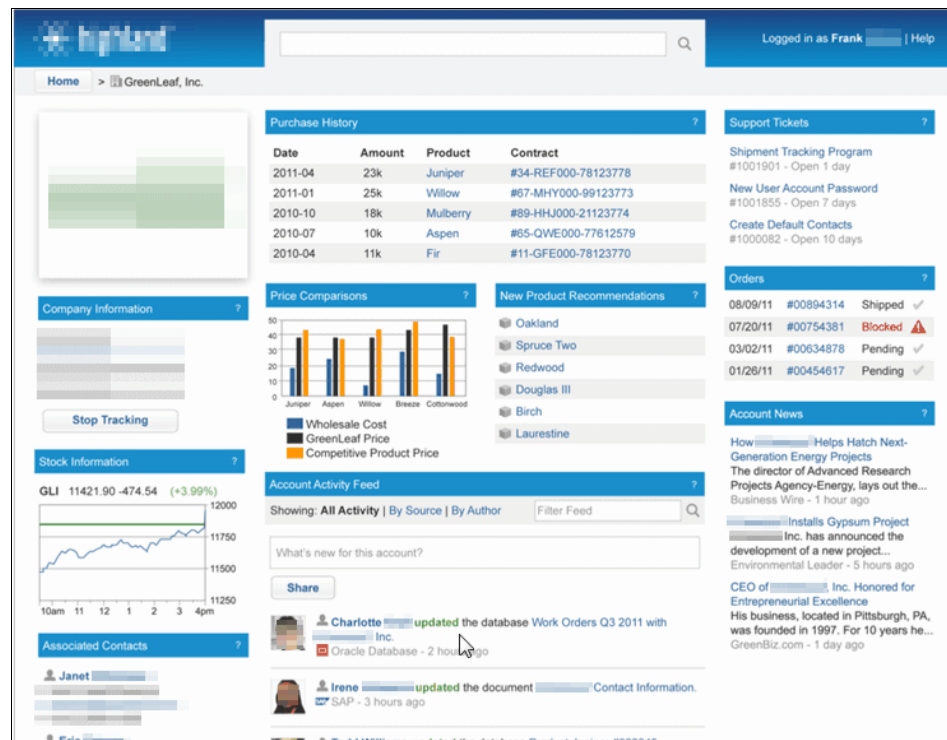


Figure 4-1 Customer-relevant information in a 360-degree application

As you begin thinking about the information that your users are looking for, you should begin to focus in on who your users are, and what their typical workflows are. Watching your users' workflows provides additional insights that complement your users' assessments of their requirements.

If you take what you find by watching user workflows and use those observations to augment what they have told you, you will be well on your way to being able to envision and deliver the 360-degree view that your users need.

As an example, suppose that your sales team is complaining that marketing is taking too long to turn around new market reports. Who are the targeted users of the application: your sales managers or your marketing team? Observing the workflow for creating these reports can identify opportunities for improving their production, actually benefiting both groups.

Suppose that you find that marketing is slow in turning around reports, because they are creating net new knowledge in response to each request, and that they are not capitalizing on existing material because they cannot find it! Your marketing team might not understand how long their research should take if they are used to the way things are.

Worse yet, you learn that they need information from other teams, and that there is no self-service, so they must wait for a colleague to email them material. What if you could make a 360-degree, self-service marketing workspace? By doing so, marketing could turn around reports much faster and positively affect sales. This is a problem you might not have been able to solve by asking. You likely had to witness their workflow to understand the root cause of the issue.

By expanding the context of a 360-degree information application, you can use the same infrastructure to provide a self-service workspace for sales, with a slightly different view than marketing. Your sales team is now happy, because they do not need to constantly email your marketing team to request the reports that they are looking for. Your users turned out to be both teams, each with their own use case and context.

When planning 360-degree applications, you should also think more broadly than just considering internal users. What about your external business partners and suppliers?

For example, IBM worked with a large industrial manufacturer whose major challenge was that their business partners did not have a 360-degree view of the products that they were re-selling. Business partners needed to call into different groups and visit different websites to find the information that they needed to support their own sales cycles.

This type of incremental complexity wastes time, increases business partner frustrations, and hurts sales. A well planned 360-degree information application can eliminate this sort of problem by consolidating relevant information, reducing business partner management costs while improving their satisfaction.

The key to understanding the users and use cases that you need to address is by seeing that if you follow the users, you will find the use case. Don't merely ask what they need, but also observe how they work. The goal of a 360-degree application is for a user to visit the application many times throughout the day. By taking the time to observe and ask questions, you will be well on your way to making that vision a reality.

The types of questions you should be answering are ones that historically require multiple systems and significant intuition to connect the information dots scattered across different silos.

4.2 Identifying data sources

Big data spans four dimensions: volume, velocity, variety, and veracity. One of the most powerful capabilities of a 360-degree Watson Explorer application is its ability to deal with the variety of your data. By variety, we mean both the data type and where the data is stored.

After you understand your users and your use cases, you can focus on the data sources that you need to include in your 360-degree application. The user requirements that you collect will identify the data sets that you must integrate into that 360-degree view.

List the data that you gathered from asking and observing your users. For example, in the telecom call center example, you found that your users would like to be able to quickly answer the following questions about a customer who has contacted them:

1. What products has this customer purchased?
2. Where does this customer live?
3. What issues has this customer had in the past?
4. How is this customer using our products?
5. What is available in inventory?

The answer to each question can come from data in a different repository, as shown in Figure 4-2 on page 70. You can often map each question to the data repositories where the answer is located:

1. SAP Document Management System (DMS)
2. Sales force CRM
3. Custom call record application based on a DB2 database
4. Social media
5. Inventory management system

Figure 4-2 illustrates the concept of data fusion.

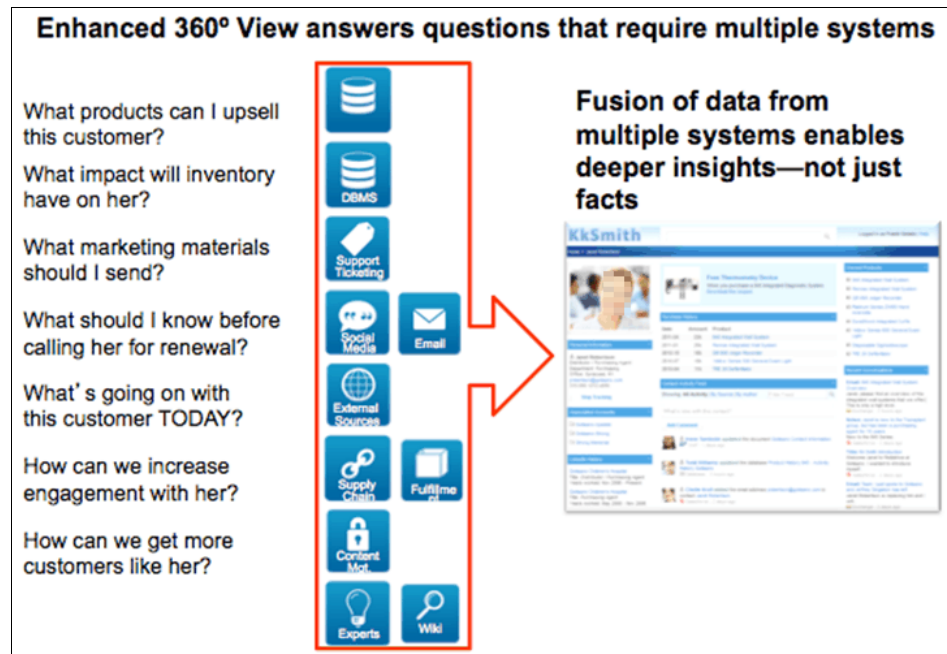


Figure 4-2 Driving data fusion by examining common application requests

The data sources that are required by a 360-degree application can be both structured and unstructured content. For example, where is the associated knowledge content stored?

The answer to this question can be more complex than simply saying that all of the relevant data is stored in SharePoint. You might have to consider which SharePoint farms or sites are most relevant in the information that your users need. This requires considering what types of security and access control your application needs to support and enforce.

A 360-degree information application built using Watson Explorer can flexibly control what data is available to the user. This security can be based on a custom permission scheme or based on the permissions in the system where that data originates. Enterprise-grade applications require that only the information that a user has permission to view can be seen within the higher-level, 360-degree application.

The following list includes questions that you should answer for each data source that you have identified to be included in your 360-degree information application:

- ▶ Who uses the information stored in this repository?
- ▶ What is the product name and version where the data is stored?
- ▶ Is security needed for this repository? If so, how are the ACLs attached to content in this repository? Are there any custom rules that need to be implemented that are not part of the repository's default security?
- ▶ How many documents (searchable, navigable entries) are in the repository?
- ▶ What is the estimated total size of the content that you would like to be part of your application?
- ▶ How often does your application's view of that data need to be refreshed? Every day, every hour, or every minute?
- ▶ How much data (by size or number of documents) is added, changed, and deleted per day? Per week?
- ▶ What languages are used in the records of this repository?
- ▶ What fields should be shown to users as part of a result in your application?

When using remote data sources, you must consider multiple factors when thinking about deploying a 360-degree application. What provides the most value to start with, and what can you augment over time?

For data sources that support security, supporting that security model is almost always mandatory, but certain data sources might not actually contain data that you need to prevent certain users from seeing. Similarly, you might be able to incrementally improve data refresh rates while an application is in use.

After a 360-degree application is in production, you might discover new data sources that provide additional information that you can use in that application. You must consider the users and their usage requirements for each emerging source of relevant data.

4.3 Modeling entity relationships

After you identify the data sources that people use, you can begin to extract the relationships that exist between the data in those sources. As described in 2.6, “Application Builder architecture” on page 26, the entity model for your application should capture the important business entities and relationships that your users are interested in for display, searching, and exploration in Watson Explorer.

Based on the types of observations that were described in the previous sections, starting to define entities and the mapping between them should not be difficult. The key to this process is identifying what is driving the context of each of your pages. If you are creating a product knowledge hub, the model should be focused on products and information that is related to those products, such as pricing, support details, and so on.

If the focus of your application is on giving each customer a personalized experience, then your model should be focused on enabling the customer to look at past purchases, configure preferences, explore support tickets, and so on. A good entity model presumes a deep understanding of the types of information that users would like to see, search for, and explore.

360-degree application should not introduce a new data silo, but should instead better capitalize on your existing capabilities and information. For example, if your requirements gathering has determined that you need a 360-degree view of customers, you can extract a customer list from a CRM system, and identify the account owner in sales so that you can start mapping sales leaders to customers.

You can also extract what products each customer owns, and use sales and marketing information to identify related products that they might be interested in. Modeling entity relationships is key to maximizing the benefits of your 360-degree application, but is relatively straightforward if you have spent enough time thinking about your use cases and your data.

The Application Builder administration tool enables you to create and fine-tune entity definitions, and to define the relationships that exist between fields in different entities. When complete, your entity definitions give your users a 360-degree view of your data.

Chapter 9, “Creating an application with Application Builder” on page 131 provides examples of defining entities and entity pages in Watson Explorer. You can also consult the Watson Explorer Application Builder documentation for detailed information about modeling entities.

4.3.1 Matching entities with non-identical fields

A 360-degree information application associates entities across repositories by finding exact matches on metadata. For example, Application Builder links a user in your CRM system to a user that exists in your sales database by looking for all accounts in the repository that have metadata on them containing that user's name. But what do you do when your CRM contains Joseph User and your sales database contains Joe User?

The solution for slight differences in metadata across repositories is to use external systems such as MDM, as explained in 1.5, “Introduction to master data management” on page 10.

4.4 Planning for deployment and implementation

You have your use case, you have identified your users, you know what data you want to use, and you have modeled your entity relationships. You are well on your way to a successful 360-degree information application deployment! The most successful deployments use a phased approach to implementing their application.

Figure 4-3 shows the different phases of deploying a sample client application, and provides an example of how to think about phasing in your deployment for maximum return on investment (ROI).

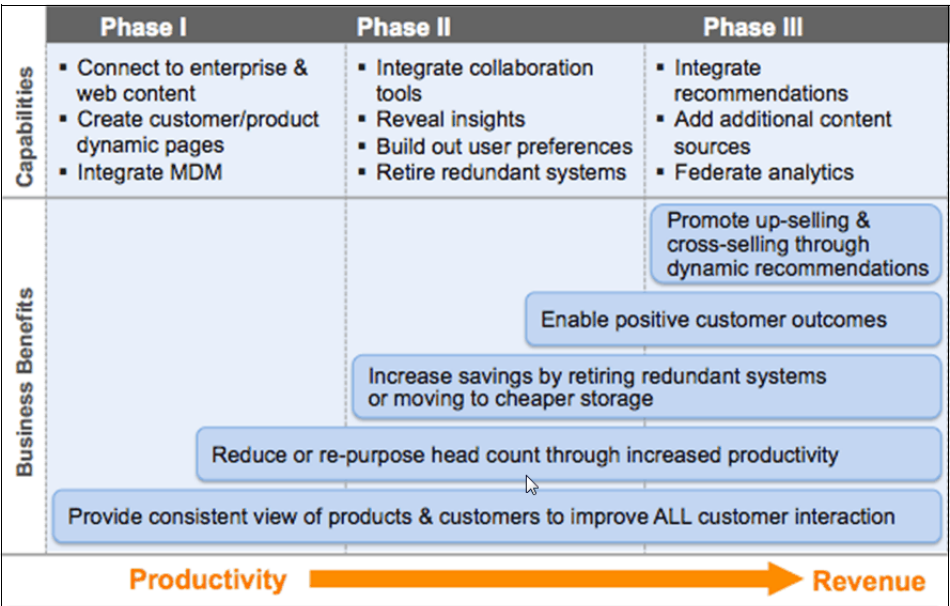


Figure 4-3 Phases in deploying and refining a sample application

Figure 4-3 maps the business benefits that you can expect across three phases of deploying a 360-degree Watson Explorer application. As more capabilities are brought online, additional business benefits continue to increase the ROI for developing your application.

In the first phase of deployment, we suggest connecting to the most critical data sources that contain customer and product information needed to improve customer interaction. This information is usually the key business information that you initially identified, as described in 4.1, “Identifying use cases and users” on page 66. If MDM is already deployed, you can capitalize on the data similarities and relationships that it identifies.

In this phase, you deploy customer and product pages similar to the ones that you see as part of the sample application that is described in Chapter 9, “Creating an application with Application Builder” on page 131. During pre-deployment analysis, these pages are often prototyped through *wireframes* that enable you to specify and experiment with general data layout without requiring initial implementation.

As a result of efficiencies that are identified during Phase I, you might even be able to reduce or reallocate head count for the next phase of your deployment.

In Phase II, we suggest integrating collaboration tools, both the built-in tools provided by the Watson Explorer Application Builder and any relevant tools that your organization might already have in place. During this phase, you can build out user preferences and enable front-line workers to begin customizing their application environment. Customization requests can also provide insights for future expansion of the application that you are deploying.

As a result of user confidence built by Phase I and Phase II capabilities at this point in your deployment, you might be able to consider retiring redundant systems, or at least moving them to the background to enable further savings. At this point, you also begin to see increased positive client outcomes.

In Phase III, we often see a successful integration of the Watson Explorer Application Builder recommendation engine, and of additional data sources and analytics that plug in from BigInsights, Streams, and other analytic systems. Before taking the leap from Phase II to Phase III, make sure that you understand how people are using the data that your application delivers. It is difficult to integrate good recommendations if you do not understand the data that you have to begin with.

During and after Phase III, you can expect your users to see additional up-selling and cross-selling opportunities as a result of the recommendation engine. Such add-on benefits only increase the level of confidence that front-line employees have in the system.

Although this section described a phased approach to application deployment, deploying a 360-degree information application does not have to take years. In fact, we have had companies that have been able to complete Phase I deployments in two months and then go on to Phase II and Phase III in similar time frames.

4.4.1 Example hardware architecture

This section explores a few example hardware architectures for a 360-degree information application. InfoSphere Watson Explorer deployments can be easily designed for reliability, availability, and recovery, all of which are must-haves for a successful enterprise software deployment.

Watson Explorer Application Builder provides both the tools and the framework for combining and presenting a 360-degree view of your data stores. Watson Explorer Engine or the BigIndex API are used to crawl those data stores and store the search collection data.

Watson Explorer Engine or the BigIndex API then delivers information from those collections as requested by the Application Builder. The Watson Explorer Engine can also route queries to external data sources to deliver dynamic content that is not indexed in a Watson Explorer Engine search collection.

Apache ZooKeeper contains the configuration information that is common across all Application Builder instances in a deployment. Every Application Builder instance connects to ZooKeeper and automatically obtains current configuration information when an application page is requested by the user.

Now, consider an example deployment. Assume that your application has a total of 500 users, and that you expect 100 simultaneous users at peak usage times. Also assume that you have five unique data stores:

- ▶ Your CRM
- ▶ A content management system
- ▶ Three custom applications that use a database back end

The total amount of data in these repositories is 1 Terabyte.

Finally, you'd also like to retrieve and integrate content from an external data source, such as Twitter, or an external data provider that your company subscribes to, such as a market research firm.

Figure 4-4 shows a hardware architecture that could be suggested for a deployment with these requirements.

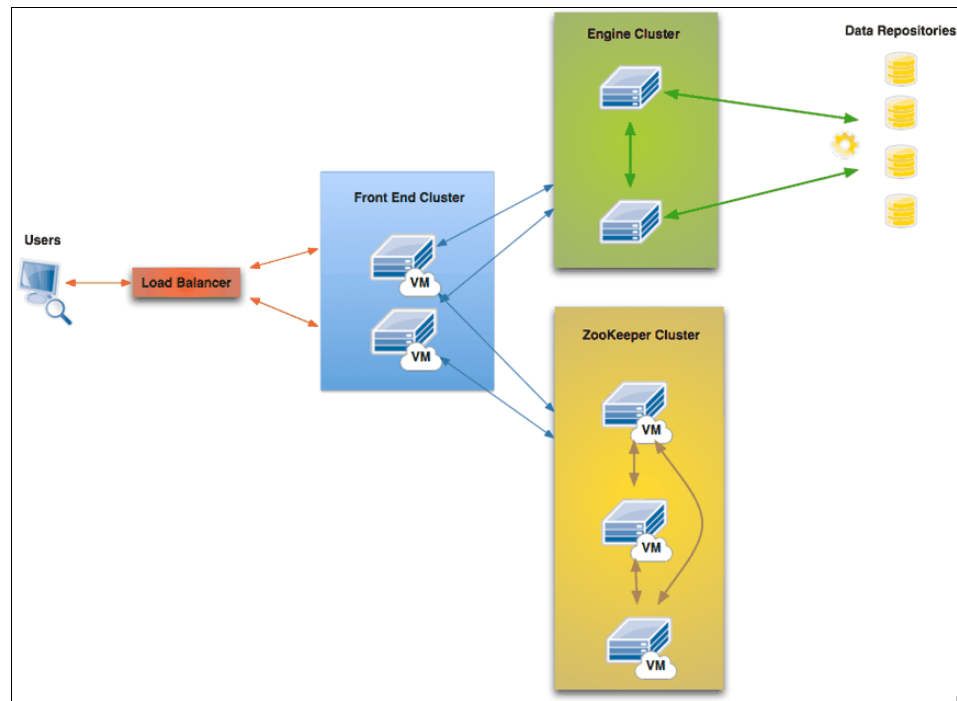


Figure 4-4 Example 360-degree information application hardware architecture

In this scenario, users interacting with the application are directed by a load balancer to the front-end cluster. The front-end cluster contains two virtual machines (VMs) running Application Builder. The load balancer determines which of the front ends each user's ID is directed to. Each front-end VM should be at least a quad-core 2 GHz+ system, with at least 8 GB of RAM and 30 GB of local disk space available.

The ZooKeeper Cluster contains three VMs, each of which have a single core processor, at least 4 GB of RAM, and 72 GB of local disk space available. The ZooKeeper machines provide application configuration information for the front-end cluster servers, and keep the Application Builder configuration in sync across the VMs in the cluster.

The front-end cluster servers query the Watson Explorer Engine cluster, which contains two physical servers. Each of these servers should have dual quad-core 2 GHz+ processors with 32 GB of RAM and 1 TB of available disk storage.

The front-end servers query an Engine server that is dedicated to responding to queries, while the other Engine server crawls and indexes the data. Using Watson Explorer's distributed indexing mechanism, the Engine instances keep their indexes in sync.

InfoSphere Watson Explorer deployments can be designed to provide 100% failover. The architecture described in this section provides for failover in each of the clusters that make up the 360-degree information application. In the front-end cluster, each of the VMs can act as the stand-alone front end if the other server malfunctions. The same is true of the ZooKeeper VMs.

Remember that although this section shows how to deploy InfoSphere Watson Explorer in clusters, it is also possible to combine these clusters to single machines. This consolidation is useful for development environments where hardware expenses must be minimized.

4.4.2 Scalability

InfoSphere Watson Explorer is highly scalable. This scalability gives architects confidence, knowing that they will not encounter limitations as the use of their applications and the amount of data ingested by those applications grows. Each of the three clusters (front-end, ZooKeeper, and Engine) can be scaled horizontally, and offer redundancy and scalability.

As an example, if you realize that your usage requires more front ends as a result of increasing use, you can merely add more VMs, and modify the load balancer to distribute to them. ZooKeeper also scales horizontally, so scaling the ZooKeeper layer means simply adding more instances.

Watson Explorer Engine clusters also scale horizontally. Engine performs two primary functions:

- ▶ Indexing (crawling)
- ▶ Responding to queries

Figure 4-5 shows an Engine cluster with a single tier. The single tier is used to both index data and respond to queries. A single tier cluster can be scaled horizontally by adding additional servers to the tier. In that case, the front-end cluster can be configured to use both servers, and to stop querying one of the servers in the cluster if a hardware failure prevents one of the servers from responding.

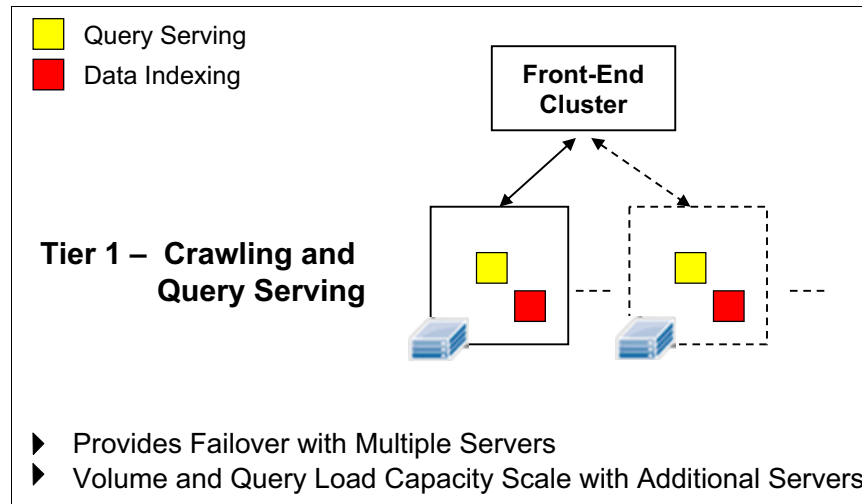


Figure 4-5 Single tier Watson Explorer Engine deployment

Using Watson Explorer Engine's Distributed Indexing feature, all indexes should be replicated to at least one of the other servers in the cluster. If a single server fails, the indexes already exist on at least one of the other machines. Application Builder detects if an Engine instance is not available, and will reroute queries appropriately, providing automatic query failover.



Lab scenario overview

This lab exercise demonstrates how to use IBM InfoSphere Master Data Management (InfoSphere MDM) and IBM Watson Explorer to create an application that provides a 360-degree view of the customers for a fictional company, the Sample Outdoors Company. This type of application would be used by customers in service, marketing, and sales to view all relevant information about their customers.

For the purposes of this lab exercise, the Sample Outdoors Company is a sporting goods manufacturer that sells directly to retailers, and also sells to online consumers using an e-commerce platform. The problem faced by the Sample Outdoors Company is that its operations use many different systems and web applications to manage different types of data. They also use different systems to support their operations in different countries. These are common problems faced by most complex businesses.

This chapter gives you an overview of the lab scenario of this book:

- ▶ It describes the business challenges that are addressed by the sample application for this lab scenario.
- ▶ It identifies common InfoSphere MDM and Watson Explorer use cases that apply to this application.
- ▶ It provides application-specific examples of the application planning concepts that were introduced in Chapter 4, “Planning a 360-degree information application” on page 65.

5.1 Key business challenge

A key challenge for employees working in customer service, marketing, and sales organizations is to find a single location that provides all of the relevant information about their customers and products.

Traditionally, employees have to spend many hours looking for different pieces of information and hunting for answers on various systems, because different portions of that information are stored in different, un-connected data stores. Such data stores are often referred to as *silos* because of their stand-alone nature.

Consolidating and providing all of the information about customers and products in one place is the solution to this challenge. One way to implement this solution is to use a single, monolithic application to store all corporate data, but that is expensive and inflexible.

This is because different applications are designed for different tasks, and use different data formats and data storage locations. In addition, some types of data (such as product data sheets and documentation) are typically stored in files that exist outside of a specific application.

InfoSphere Master Data Management brings powerful tools to assure a consistent match of key entities across disparate systems, key entities such as customers and products.

Watson Explorer makes it easy to create applications that provide a unified view of enterprise-specific data, incorporating data from internal and external sources without having to move data from one source to the other, and identifying relationships between the data that is stored in those different sources.

The searchable representation of enterprise data in Watson Explorer provides additional capabilities, such as the ability to augment data with metadata, such as comments or ratings. This enables users to identify high-value search results or add insights that can make it easier for other users to locate and use the associated search results.

This metadata can enhance the value of the original data regardless of the format or location in which that data is stored, and without affecting its use by the original application with which that data is associated.

5.2 Identifying use cases and users

As described in Chapter 4, “Planning a 360-degree information application” on page 65, identifying the users and use cases for an application is the most important step in creating successful 360-degree applications.

The following common use cases are relevant to the Sample Outdoors Company:

- ▶ Use case 1: Customer service representatives want to see real-time updates of customer-related information, such as products purchased, past interactions, status of recent orders, and available promotions that are relevant to that customer based on their purchase history.
- ▶ Use case 2: Sales representatives want to see the status of high profile projects, high priority support items opened by the customers, billing information, product recommendations, and so on.

This lab exercise focuses on use case 2, in which a sales representative needs to find the following information about a particular strategic customer:

- ▶ Most recent product purchases
- ▶ Support tickets and status
- ▶ Customer contact information

In addition, each sales representative also wants to be able to see a list of their strategic customers and recent activities.

5.3 Identifying the data sources

The next important step is to identify the data sources within the enterprise that hold all of the information about their customers. The data sources can be content management systems, database applications, customer relationship management (CRM) and enterprise resource planning (ERP) systems, portals, and custom applications.

Figure 5-1 shows the data sources in an enterprise that hold pieces of the information about its customers.

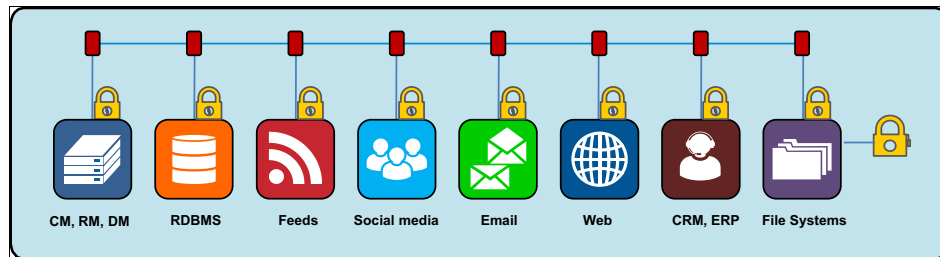


Figure 5-1 Potential data sources for a 360-degree application

In the example scenario used in this lab, DB2 databases are the data source that holds information about customers and the products that they have purchased. Watson Explorer includes standard connectors to connect to various data repositories, including DB2.

5.4 Modeling entity relationships

Identifying the relationships between users of the application, their key use cases, and the data that is available in the data repositories used within an organization is the fundamental step in developing a 360-degree information application. These items are known as *entities*, which are then connected to form the building blocks of a 360-degree application.

In the lab scenario used in this book, an entity is a person, customer, department, retailer, product, order, purchase, contact, or competitor.



Installation and initial configuration

This chapter provides a brief information about the installation and configuration of the following products:

- ▶ Watson Explorer installation and configuration
- ▶ InfoSphere MDM installation
- ▶ InfoSphere MDM Connector installation and configuration

6.1 Watson Explorer installation and configuration

Watson Explorer uses IBM Installation Manager to install Watson Explorer and related software modules, and can be installed either graphically or from the command-line. Watson Explorer can be installed by an administrator or a non-privileged user, either of which satisfies the requirements of the lab scenario that is presented in this book.

The lab scenario is introduced in Chapter 5, “Lab scenario overview” on page 79, and is explored in detail in Chapter 9, “Creating an application with Application Builder” on page 131.

Rather than duplicate installation information that is thoroughly described in the official product documentation, this book does not provide installation instructions.

For complete information about different options for installing Watson Explorer for production development and deployment, including integrating it with standard system startup, shutdown, and management procedures, see the *Watson Explorer Installation and Integration Guide*, which is available in the Watson Explorer Knowledge Center:

<http://ibm.com/support/knowledgecenter/SS8NLW>

To work through the lab scenario that is described in this document, you must perform the following tasks, in order:

1. Install Watson Explorer Engine, Watson Explorer Application Builder, and ZooKeeper, as described in the PDF installation documentation or in the public Knowledge Center. If you are installing Watson Explorer on a system that does not have Internet connectivity, you should also install a local copy of the Watson Explorer Knowledge Center as part of the installation process.
2. Integrate Watson Explorer Engine with a system web server, or use its embedded web server, and make sure that the selected web server is running.
3. Start ZooKeeper.
4. Start the WebSphere Application Server that supports Application Builder.

Note: For more detailed information, see the public Knowledge Center, or view or download the *Watson Explorer Installation and Integration Guide* using the links earlier in this section.

6.2 InfoSphere MDM installation

The detailed installation and configuration of the InfoSphere Master Data Management (InfoSphere MDM) is documented in the IBM Information Center at the following website:

<https://ibm.biz/mdm11-install-guide>

Note: In this book, we focus on the InfoSphere MDM Standard and Advanced Editions.

6.3 InfoSphere MDM Connector installation and configuration

Watson Explorer provides a framework for building connector plug-ins that run on the Java Virtual Machine. Connectors enable accessing and retrieving data. The InfoSphere MDM connector enables indexing data from MDM into Watson Explorer.

Note: The assumption here is that InfoSphere MDM is installed and the default party model from the domain template is deployed on the InfoSphere MDM instance.

6.3.1 InfoSphere MDM Standard Edition connector

InfoSphere MDM Standard Edition (InfoSphere MDM SE) connector is a virtual MDM connector available for download at the following website:

<https://ibm.biz/ide9-fixcentral>

Select **connector-mdm-se-3.1.1** to download the `connector-mdm-se-3.1.1-distrib.zip` file containing the InfoSphere MDM SE connector.

InfoSphere MDM SE connector installation

Complete the following steps to install InfoSphere MDM SE:

1. Unzip the distribution file under the Engine directory (for example, C:\IBM\IDE9.X\Engine). Unzipping the distribution file directly into the directory where the files need to be, not just a temporary directory.
2. Start the engine using the respective command, for example:
C:\IBM\IDE9.X\Engine\bin\velocity-startup.exe
3. Login to the Watson explorer as an administrator (Figure 6-1):

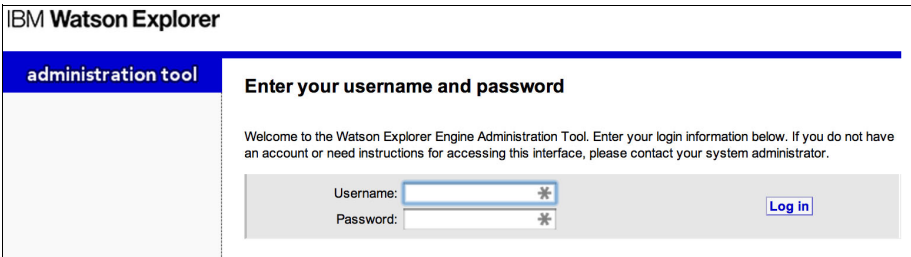


Figure 6-1 Login to Watson Explore

4. After the login, click **Installation** (Figure 6-2):

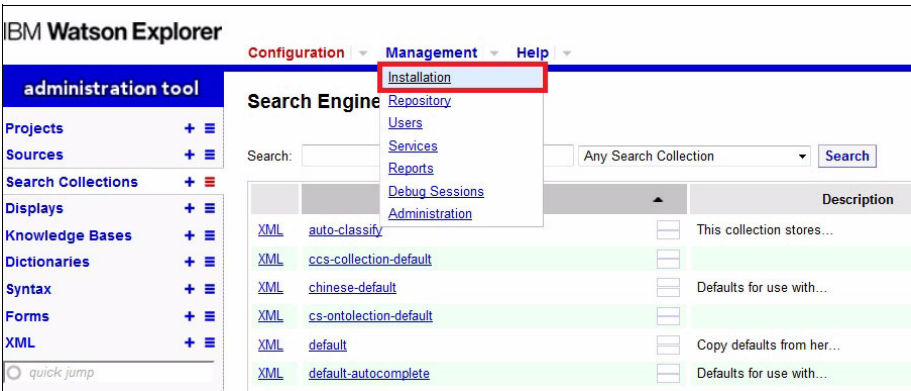


Figure 6-2 Installing MDM SE Connector

5. Click **unpack** (Figure 6-3) to complete the InfoSphere MDM SE connector installation.

Repository		unpack
Internal Nodes	1,074 nodes	
Overridden Internal Nodes	0 nodes	
Custom Nodes	59 nodes	
Custom Projects	0 projects	
Custom Collections	3 collections	

Figure 6-3 Unpacking MDM SE Connector

Configuring InfoSphere MDM SE connector

The connector has multiple seeds for configuring:

- Entity
- Security
- Relationship

InfoSphere MDM SE Entity Search Collection

InfoSphere MDM SE Entity search collection is a collection of entities from InfoSphere MDM where the entity is a golden view of a particular type of entity such as Person, Organization, and Products.

Use the following steps to create and configure a search collection:

1. Go to the Watson Explorer Admin Tool's Configuration section (Figure 6-4). Click the plus sign “+” next to Search Collections.

Input name and click **Add** to create a search collection.

The screenshot shows the IBM Watson Explorer Admin Tool interface. On the left is a navigation pane with a tree view containing 'Projects', 'Sources', 'Search Collections', 'Displays', 'Knowledge Bases', 'Dictionaries', 'Syntax', 'Forms', and 'XML'. The 'Search Collections' item is highlighted with a red box. The main area is titled 'New Search Collection' and contains a form to 'Create a new collection'. The form has a 'Name' field with the value 'MDM-Virtual (Name Token)', a 'Copy defaults from' dropdown set to 'default', and a 'Description' text area. An 'Add' button is in the top right of the form. At the bottom left, there is a 'Project' dropdown set to 'Query-Info' and a 'Return to:' section with links to 'Collection: MDM-Virtual', 'Project: Query-Info', and 'Search Engine: Query-Service'. The top of the page has a header with 'IBM Watson Explorer', navigation links 'Configuration', 'Management', and 'Help', and a 'Logout' button on the right.

Figure 6-4 Creating a search collection

2. After creating the search collection, in the Configuration tab under the crawling section, click **Add a new seed** (Figure 6-5).

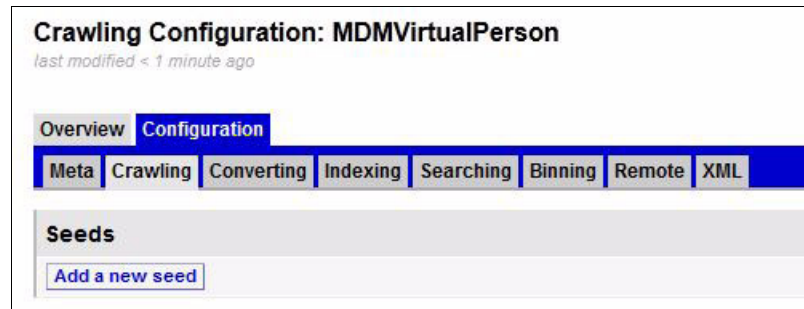


Figure 6-5 New seed for crawling

3. From the list, select **IBM MDM Standard Edition Connector** (Figure 6-6).

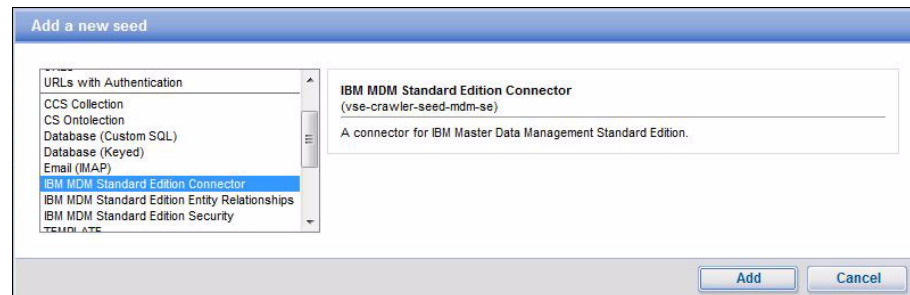


Figure 6-6 MDM Standard Edition connector

4. Configure the seed with the InfoSphere MDM Standard Edition details as shown in Figure 6-7 on page 89. Note that the JDBC details shown here are DB2 specific:
 - Host: Host name or IP address of InfoSphere MDM SE server is hosted
 - Username: A valid MDM user
 - Password: A valid password
 - SOAP port: port number where Webservices are hosted
 - SOAP Endpoint Protocol: http or https
 - SOAP Endpoint: /ibminitiatews/services/IdentityHub?wsdl
 - Database type: DB2 or Oracle, and the database configuration changes are based on the option selected.
 - JDBC Connection String: jdbc:db2://localhost:50000/MDMV113

- JDBC Username: A valid database user name of InfoSphere MDM SE
- JDBC Password: A valid password
- JDBC Class: `com.ibm.db2.jcc.DB2Driver`
- Member Type Name: member type
- Entity Type Name: Type of entity for which the collection is created
- Composite View Name: Optional parameter. If there is a specific view needed, provide the view name here.

Seeds

[Add a new seed](#)

Seed Component: **IBM MDM Standard Edition Connector** [edit](#) [remove](#) [view resolved](#)

Host	soshekar2	?
Username	mdmadmin	?
Password	*****	?
SOAP Endpoint Protocol	http	?
SOAP Port	9081	?
SOAP Endpoint	/bin/latew/services/identityhub7wsdl	?
Database Type	db2	?
JDBC Connection String	jdbc:db2://localhost:50000MDM/V113	?
JDBC Username	db2admin	?
JDBC Password	*****	?
Member Type Name	PERSON	?
Entity Type Name	mdmper	?

Advanced (3)

Entity ID Query Size	1,000	?
Maximum java heap size	1g	?

Advanced - Logging (3)

Enable connector bootstrap logging	<input checked="" type="checkbox"/>	?
INSECURE: Enable HTTP transport logging	<input checked="" type="checkbox"/>	?

Figure 6-7 MDM SE Seed configuration

5. Advanced options:

The advanced options allow more granular level of configuration and these include the following attributes:

- Entity ID query size: In simple words, this is a query pagination size.
- Segment code filter: Filter that allows selections of segments form InfoSphere MDM SE server; specify **“all”** if the segments are unknown.
- Maximum java heap size: Maximum Java heap size of the crawler.

The entire collection attributes are optional and, if not specified, a default value will be taken. Figure 6-8 shows details of the advanced option.

Advanced (3)

Entity ID Query Size	1000	Default: 50,000
Segment Code Filter		Default: MEMHEAD, MEMADDR, MEMATTR, MEMDATE, MEMIDENT, MEMNAME, MEMPHONE
Maximum java heap size	1g	Default: 384m

Figure 6-8 Details of the advanced options

6. Continuous update:

Continuous update option brings in delta changes from InfoSphere MDM SE to Watson explorer, changes such as new entity additions or entity modification. These changes are brought into the Watson explore based on the time stamp. At the end of every crawl, the time stamp is saved in Watson Explorer and only the delta changes are bought in for every new crawl. Figure 6-9 shows the crawl options.



Figure 6-9 Continuous update

InfoSphere MDM SE Entity collection converter

InfoSphere MDM SE Entity collection converter converts the InfoSphere MDM SE format of data into a Watson Explorer format so that the data can be indexed.

To add an InfoSphere MDM converter, in the settings for the collection that you created, under the converting subsection, click **Add new converter** to add IBM Standard Edition Entities converter (Figure 6-10).

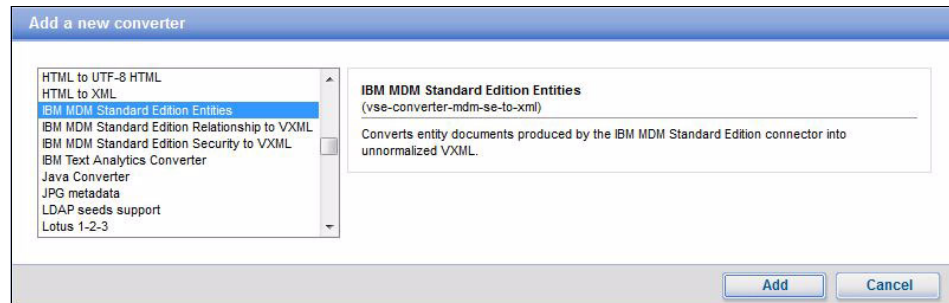


Figure 6-10 InfoSphere MDM SE converter

Figure 6-11 and Figure 6-12 show the configuration of the converters. The default options are sufficient for nearly all applications.

The screenshot shows the 'Converter Component: IBM MDM Standard Edition Entities' dialog box. It has tabs for 'OK', 'Apply', and 'Cancel'. The configuration is as follows:

- Type-In: application/mdm-se
- Type-Out: application/vxml-unnormalized
- Fallback: (unset)
- Output forking: (unset)
- Name: (empty field)
- Conditional: Test: uri, With: wildcard set

At the bottom, there is a link to 'Advanced (3)'.

Figure 6-11 Converter configuration 1

The screenshot shows the 'Advanced (3)' tab of the 'IBM MDM Standard Edition Entities' converter configuration. It contains the following settings:

- Name Mapping: (XPath) Default: `viv:choose(@name = 'Title' or @name = 'Description' or @name = 'Author', viv:str-to-lower(@name), @name, viv:str-to-lower(@name))`
- Action: (XPath) Default: `viv:choose($name = 'title' or $name = 'description'/'cluster', @action, @action, 'none')`
- Weight: (XPath) Default: `viv:choose($name = 'title', 3, 1)`
- Type: (XPath) Default: `viv:choose($name = 'title' or $name = 'snippet', 'html', 'text')`
- Output Action: (XPath) Default: `viv:choose($name = 'description' or $name = 'title' or $name = 'author', 'bold')`
- Change Values: (XPath) Default: `dyn:evaluate(viv:choose($name = 'author', 'viv:replace("[\\d]{1}-#","7")'))`
- Prepend Source Code: (unset) Default: false
- Prepend Attribute Code: (unset) Default: true

Description: Converts entity documents produced by the IBM MDM Standard Edition connector into unnormalized VXML.

Figure 6-12 Converter configuration 2

Testing InfoSphere MDM SE connector

After completing the configuration, use the following steps to test the connector:

1. Click the Overview section on the InfoSphere MDM Entity collection and click **start** as shown in Figure 6-13. Avoid using “Test it” with the MDM connector because “Test it” might time out before the initial parts of the MDM connector complete.



Figure 6-13 Starting the indexing process

2. The indexer starts indexing the InfoSphere MDM Entity data as shown in Figure 6-14.

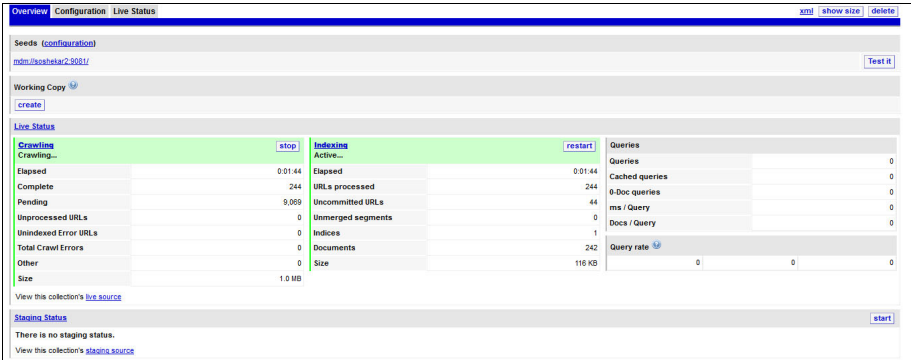


Figure 6-14 Crawling and indexing

3. After the indexing is complete, use **Test with project query-meta** on the left hand side of the page to test the connector. The example in Figure 6-15 uses “WANDA”, a first name of the person entity, as the search criteria.

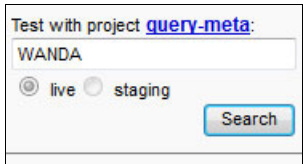


Figure 6-15 Test criteria for InfoSphere MDM entity search collection

Figure 6-16 shows the results of the search criteria. Note that the appearance of these results can be customized.

1. entity?id=1 new window preview	
Mdmsp Memhead Caud Recno:	2, 2, 2
Mdmsp Memhead Ent Recno:	1, 1, 1
Mdmsp Memhead Ent Recnos:	1, 1, 1
Mdmsp Memhead Link Type:	X, X, X
Mdmsp Memhead Match Code:	Unknown, Unknown, Unknown
Mdmsp Memhead Match Score:	0, 0, 0
Mdmsp Memhead Maud Recno:	2, 2, 2
Mdmsp Memhead Mem Idnum:	33, 32, 1
Mdmsp Memhead Mem Recno:	33, 32, 1
Mdmsp Memhead Mem Seqno:	5, 5, 5
Mdmsp Memhead Mem Stat:	A, A, A
Mdmsp Memhead Mem Verno:	0, 0, 0
Mdmsp Memhead Row Ind:	I, I, I
Mdmsp Memhead Src Code:	MDMSP, MDMSP, MDMSP
Mdmsp Memhead Src Recno:	1, 1, 1
Mdmsp Perlegalname Givenname1:	WANDA
Mdmsp Perlegalname Lastname:	WILLIAMS
mdm://soshekar2:9081/entity?id=1 - 10K - cache - MDMV2	
2. entity?id=75 new window preview	
Mdmsp Memhead Caud Recno:	2
Mdmsp Memhead Ent Recno:	75
Mdmsp Memhead Ent Recnos:	75
Mdmsp Memhead Link Type:	X
Mdmsp Memhead Match Code:	Unknown
Mdmsp Memhead Match Score:	0
Mdmsp Memhead Maud Recno:	2
Mdmsp Memhead Mem Idnum:	78
Mdmsp Memhead Mem Recno:	78
Mdmsp Memhead Mem Seqno:	6
Mdmsp Memhead Mem Stat:	A
Mdmsp Memhead Mem Verno:	0
Mdmsp Memhead Row Ind:	I
Mdmsp Memhead Src Code:	MDMSP
Mdmsp Memhead Src Recno:	1
Mdmsp Perlegalname Givenname1:	WANDA
Mdmsp Perlegalname Lastname:	MAYWEATHER
mdm://soshekar2:9081/entity?id=75 - 4K - cache - MDMV2	

Figure 6-16 InfoSphere MDM entity search results

Security of InfoSphere MDM search collection

The user (attribute) security for each InfoSphere MDM Entity can be replicated in the Watson Explorer. The groups and the authorization that are present in the InfoSphere MDM must be replicated using an InfoSphere MDM Security collection.

Use the following steps to replicate the attribute security in Watson Explorer:

1. Creating a new search collection (Figure 6-17): Click **Add** to create a security search collection.

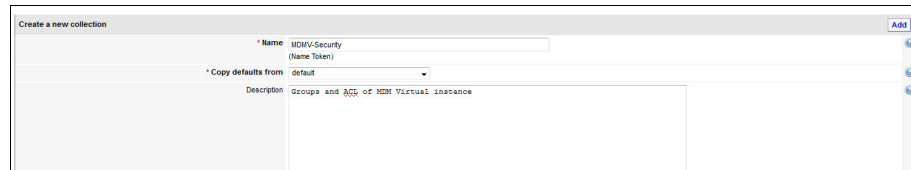


Figure 6-17 Security search collection

2. In the configuration tab under the crawling subsection, click **Add a new seed**. Select **IBM MDM Standard Edition Security**, and click **Add** as shown in Figure 6-18.

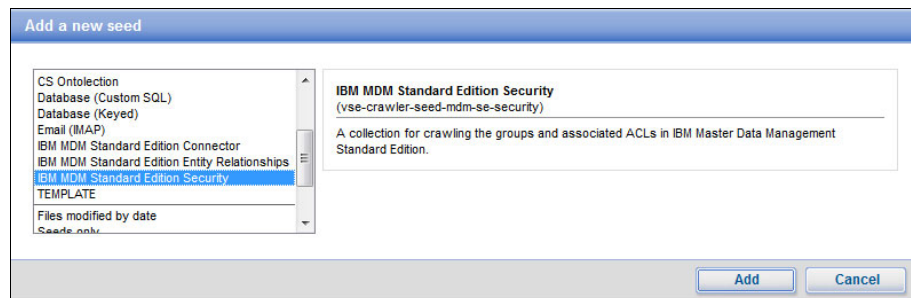
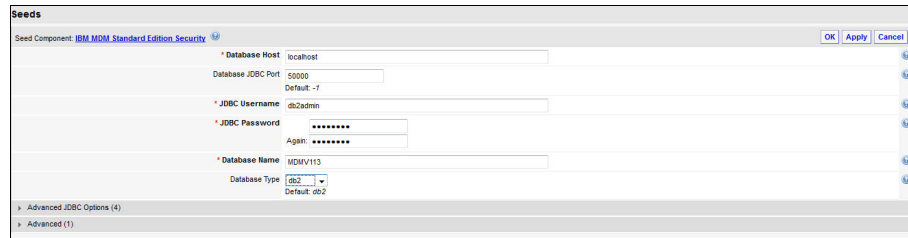


Figure 6-18 MDM standard edition security seed

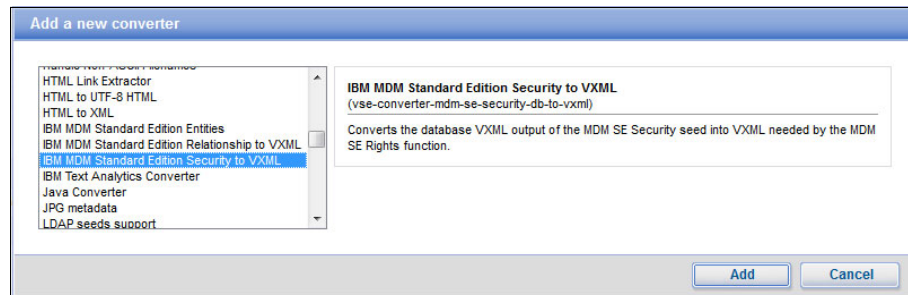
3. Add the details as shown in Figure 6-19. These details include database information where the InfoSphere MDM database is installed and configured.



The 'Seeds' configuration window shows the 'Seed Component: IBM MDM Standard Edition Security'. It includes fields for 'Database Host' (localhost), 'Database JDBC Port' (50000), 'JDBC Username' (db2admin), 'JDBC Password' (masked with asterisks), 'Database Name' (MDMV113), and 'Database Type' (db2). There are 'OK', 'Apply', and 'Cancel' buttons at the top right. A 'Advanced' section is partially visible at the bottom.

Figure 6-19 Configuration entry panel

4. Add a new converter under the configuration section and converting subsection as shown in Figure 6-20. Click **Add** to add the new convertor with the defaults.



The 'Add a new converter' dialog shows a list of converters on the left. 'IBM MDM Standard Edition Security to VXML' is selected. The right pane shows details for this converter: 'IBM MDM Standard Edition Security to VXML (vse-converter-mdm-se-security-db-to-vxml)' and a description: 'Converts the database VXML output of the MDM SE Security seed into VXML needed by the MDM SE Rights function.' 'Add' and 'Cancel' buttons are at the bottom right.

Figure 6-20 Security convertor configuration

5. Switch to the Overview section and click **start** in the Live Status section. The crawler and the indexer start to index the groups and Access Control List (ACL) from InfoSphere MDM to Watson Explorer.

Associating InfoSphere MDM entity search collection with the InfoSphere MDM security

Use the following steps to associate InfoSphere MDM entities search collection with the InfoSphere MDM security:

1. From the Overview section of MDM entity search collection. Click **Live Source** then select the **Form** tab.
2. Click **Add Form Component** to associate MDM groups and access control list for the search collection. Choose the **IBM MDM Standard Edition Group Rights** component. See Figure 6-21.

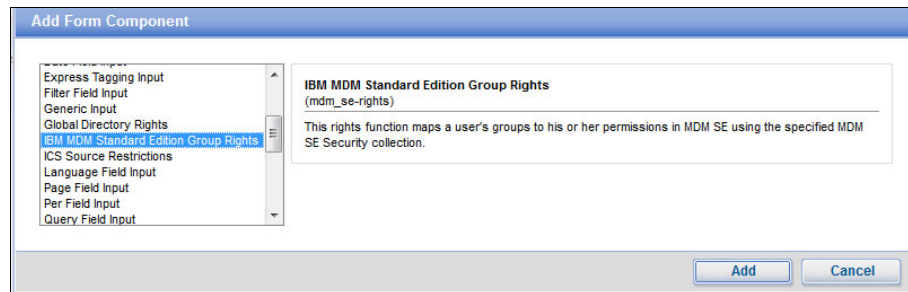


Figure 6-21 InfoSphere MMD Standard Edition groups and security

3. Add the required details as shown in Figure 6-22.



Figure 6-22 Security search collection association

InfoSphere MDM Entity relationship search collection

You can create entity relationships using MDM entity relationship search collection. The process of creating an entity relationship is same as creating the MDM entity search collection. InfoSphere MDM search collections created will be used in the Watson Explorer Application Builder to bring in a 360-degree view of the structured and unstructured information.

Note: For more details, see the *IBM InfoSphere Data Explorer Engine MDM SE Connector User Guide* that is shipped along with the connector as a PDF file, `mdm_se-connector-guide.pdf`.



Building a golden record

This chapter describes the principles of building a golden record using InfoSphere Master Data Management (InfoSphere MDM).

The following topics are discussed:

- ▶ Overview
- ▶ MDM implementation styles
- ▶ Sample process for building a virtual record

7.1 Overview

Constructing a golden record, also called a master record, is a common problem across various industries such as government, financial services, healthcare, and retail. Master data management (MDM) is the discipline that helps in arriving at the golden record, which sometime is also called a single version of truth. The need for a golden record arises from many different factors such as these:

- ▶ Growth in revenue
- ▶ Profitability
- ▶ Compliancy and risk requirements
- ▶ Cost optimization and efficiency

7.2 MDM implementation styles

In any MDM solution deployment, the key to success is arriving at the MDM implementation style. This section introduces various MDM implementation styles.

7.2.1 Virtual

The virtual style is also commonly known as *registry style*. Generally, any MDM journey starts here and, later on, evolves into the other styles. However, there is no hard rule to start from the virtual style. The MDM implementation can be started anywhere based on the enterprise and its organizational requirements.

In a virtual implementation, the source system continues to be the owner of the source record and only the critical information is passed into the MDM hub, where the matching and linking of records are done. These produce a virtual golden record or view. The virtual golden record is used for downstream processing such as data warehousing, analysis, and so on.

IBM InfoSphere Master Data Management Standard Edition (InfoSphere MDM SE) is the industry leader in virtual MDM, offering matching and linking capability which is probabilistic or deterministic in nature. The algorithm for creating a master record is assembled for specific requirements using the functions provided by InfoSphere MDM SE. InfoSphere MDM SE also supports reconciliation of matched records using the data steward capability.

7.2.2 Physical

The physical implementation is also called *operational style*. This is the most advanced MDM implementation style, where the MDM system is the owner of the master data. Data from the source system is moved to an MDM system. The data in the MDM system is not only read but also updated by the consuming system.

IBM InfoSphere Master Data Management Advanced Edition is a secure and high performing system with a pre-packaged data model for Party, Product, and Account domains. It includes probabilistic matching, and linking capability, along with the rich set of platform capabilities such as security, auditing, and history. InfoSphere MDM AE comes with a workbench for creating services and features that enhance and augment the built-in features of the product.

7.2.3 Hybrid

In a hybrid implementation, the master data ownership is shared across the MDM system and the other source systems. Two different implementation variations are possible in a hybrid style:

- ▶ All new records or entities will be persisted in MDM system, thereby producing the golden record. The ownership of data is shared, thus it is hybrid.
- ▶ For the master data from all new lines of business, the source data system will have ownership of the master data in the MDM system. Ownership of master data is similarly shared, thus it is hybrid.

IBM InfoSphere Master Data Management Advanced Edition (InfoSphere MDM AE) supports hybrid topology deployment, where master data ownership is a shared responsibility between InfoSphere MDM and other source systems that are part of the golden record building ecosystem.

7.3 Sample process for building a virtual record

Building a golden record depends on the adopted style. The example described in this section showcases the process of building a virtual MDM using InfoSphere MDM Standard Edition, because the virtual style is the starting point for most of the MDM implementation.

Figure 7-1 on page 100 shows a typical banking system with different lines of business such as core banking, credit cards, loans, and investments, each having its own copy of customer data.

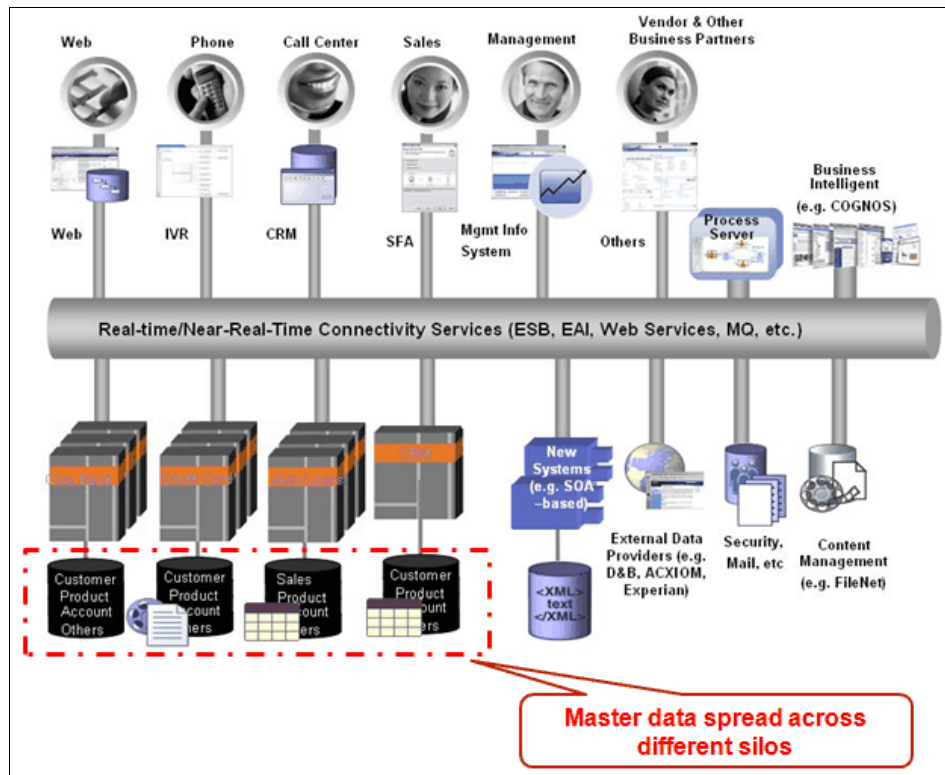


Figure 7-1 Typical banking system

Although the system is still uses the loosely coupled technology, the data resides in multiple source systems within the enterprise. Think about a merger case, where two different technologies and platforms come into play.

7.3.1 Three stages to construct and manage a golden record

The following three stages are used to construct and manage a golden record:

- Acquire:

In the *acquire* stage, the critical data is acquired through a standard ETL tool. This critical data is the core data. In a party domain, it is person or organization data.

- Act:

In the *act* phase, the data is massaged, standardized, and matched for creating and linking the entities that form the golden records.

► Manage:

The *manage* phase deals with other aspects of information governance of the data, until the data is retried.

7.3.2 Workflow for arriving at a golden view

Figure 7-2 shows the steps that are involved in building a golden view using InfoSphere MDM Standard Edition.

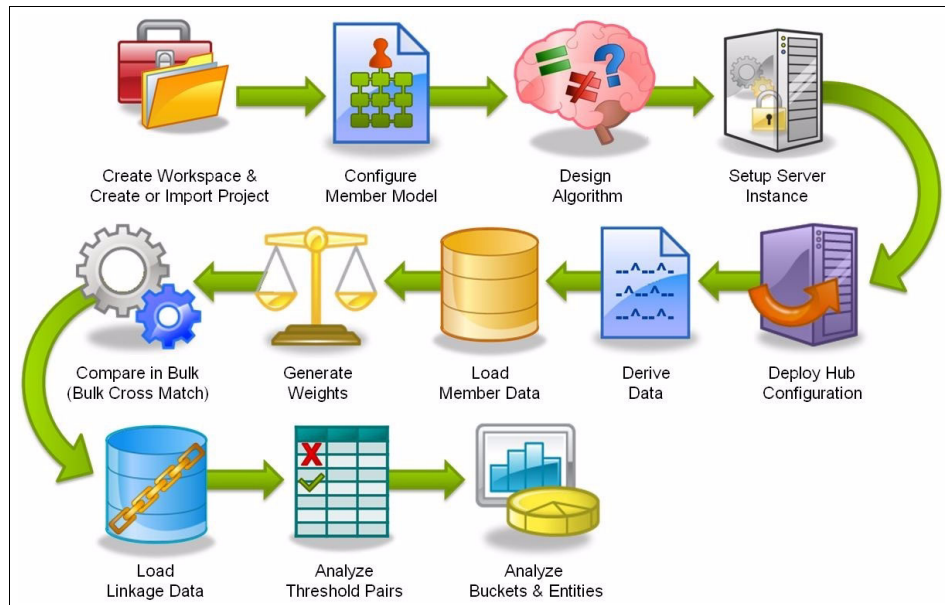


Figure 7-2 InfoSphere MDM Standard Edition core workflow

The workflow for arriving at a golden view with InfoSphere MDM SE includes the following major steps:

- Step 1: Creating a workspace
- Step 2: Configuring the model
- Step 3: Entity creation
- Step 4: Designing an algorithm
- Step 5: Setting up a server instance
- Step 6: Deploying the configuration
- Step 7: Importing the data
- Step 8: Weight generation
- Step 9: Bulk comparison
- Step 10: Threshold analysis

Step 1: Creating a workspace

The workspace is technically an Eclipse project that facilitates creation and design for algorithms.

Step 2: Configuring the model

In this step, you create the basic member model that consists of creating a member, attribute types, attributes, and source (source is name of actual external source where the data re-sides). Alternatively, you can import a predefined configurations into the workspace using the domain templates that provide predefined models for party, patient, provider, and so on.

Step 3: Entity creation

In InfoSphere MDM SE, an *entity* is a distinct type of master data such as person or organization that shares a common linking identifier called an Entity Identifier (EID) across all the matched master data entity. Figure 7-3 shows the relationship between the entity, members, and the algorithm associated with member model.

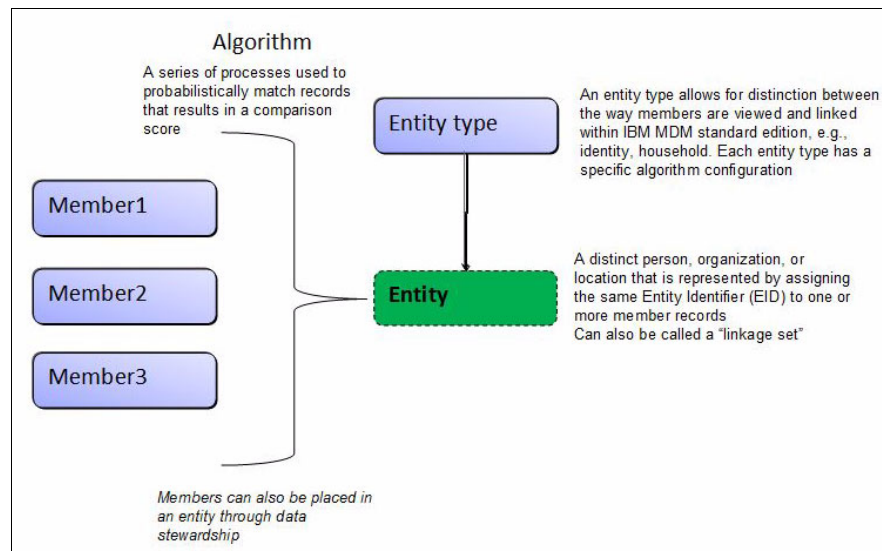


Figure 7-3 Relationship between entity, members, and the binding algorithm

Step 4: Designing an algorithm

Algorithm design involves deriving the match and link algorithm by using the predefined functions that are available in the workbench.

The following matching process is used:

- ▶ Optimize data for statistical comparisons:
 - Operate against raw data, no cleansing required.
 - Normalize and compact data, create derived data layer. Source data remains intact.
 - Apply phonetic equivalences, tokenization, nicknames, and so on in the matching process.
- ▶ Find all the potential matches:
 - Cast a wide net: partial matches, reversals, anonymous values, and so on.
 - The final candidate list is a combination (OR) of multiple bucket roles.
 - Each bucket role is created from functions of parts of a single attribute, combinations of multiple attributes, or both.
- ▶ Score accurately using probabilistic statistics:
 - Compare attributes one-by-one and produce a weighted score (likelihood ratio) for each pair of records.
 - Frequency weights specific to your business.

Figure 7-4 on page 104 shows the InfoSphere MDM SE matching approach with the sample data in the buckets and bucket hashes. Buckets are nothing but the grouping the data based on the attributes in the model; something like name and phone number or name and zip code. Bucket hashes are unique IDs created for the particular bucketing criteria. The end result is master data entities that are formed and ready for threshold analysis where the upper threshold and lower threshold are fixed.

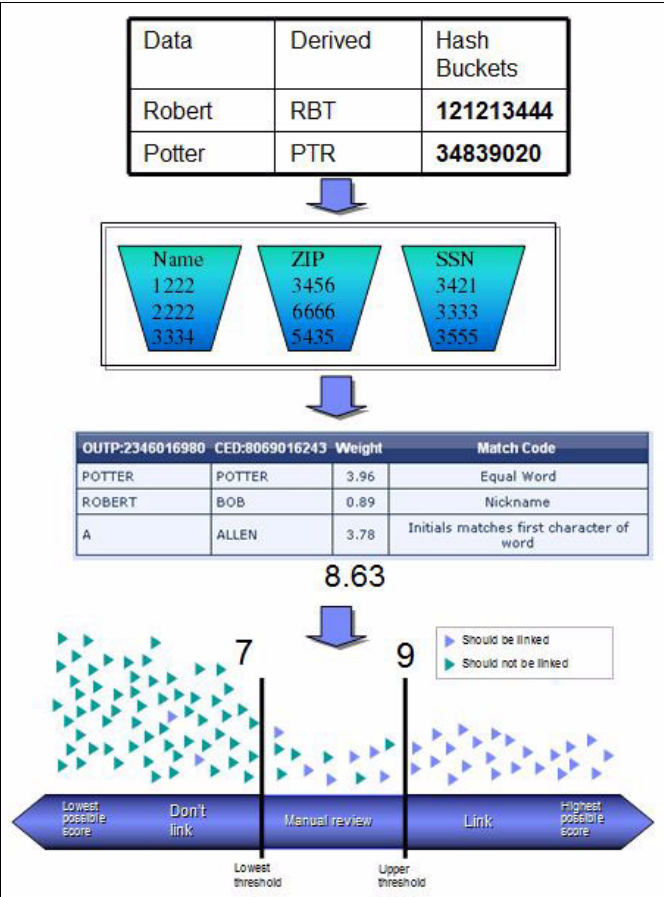


Figure 7-4 InfoSphere MDM matching process

Step 5: Setting up a server instance

This step involves connecting the workbench to an InfoSphere MDM instance. The connection requires the MDM instance server details such as host and the port details.

Step 6: Deploying the configuration

Deploy the configuration to the InfoSphere MDM instance as shown in Figure 7-5. This deploys the model that is developed in the workbench. The deployment includes core configuration, algorithms, string data, and weight data.

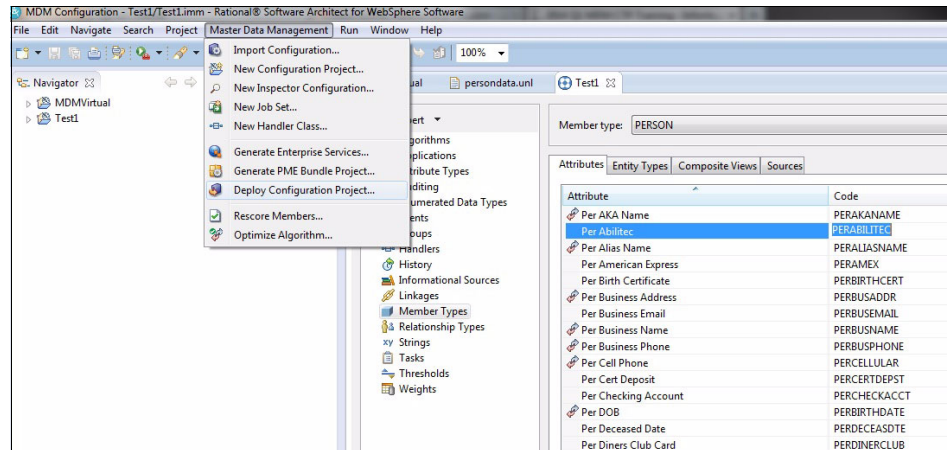


Figure 7-5 Deploy configuration

Figure 7-6 shows the steps that are completed at this point.

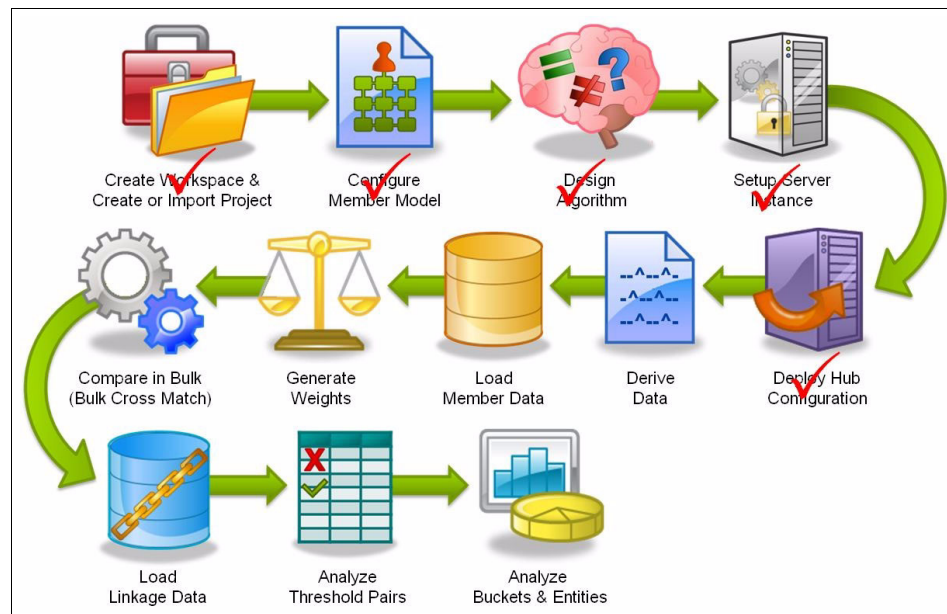


Figure 7-6 Intermediate workflow configuration state

Step 7: Importing the data

After the data is brought into a standard format using ETL tool, use InfoSphere MDM SE utility, mpxdata, to massage, create, and import the data.

The mpxdata utility uses raw data to build member unload files (.unl), generate comparison strings, assign bucket hashes, and create binary files.

Figure 7-7 shows the input file, configuration, and the associated job.

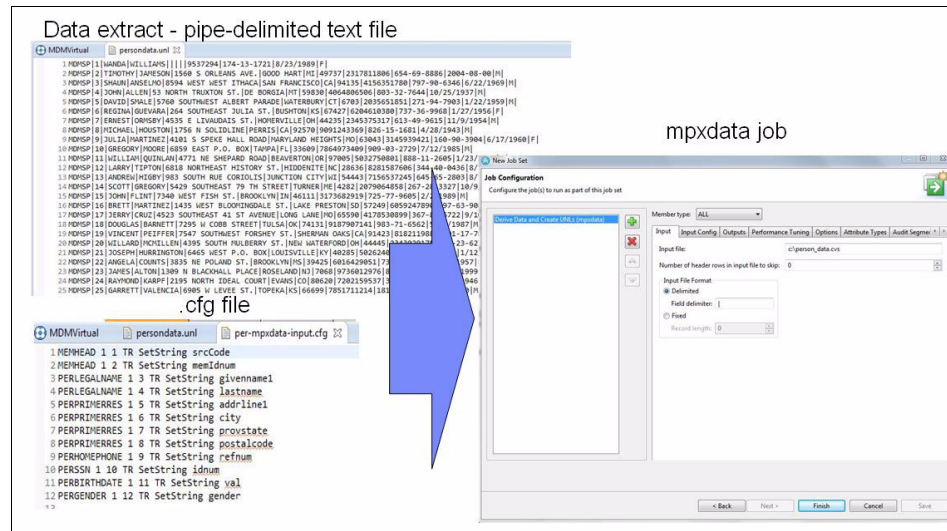


Figure 7-7 mpxdata job

Step 8: Weight generation

Weight generation is a process within the Master Data Engine that enables the generation and assignment of weight values to attributes. Weight assignments enable determination of a match or non-match between members. After the weight generation process, the configuration must be deployed.

Step 9: Bulk comparison

This process matches and links the entities in bulk by using the bulk cross match (BXM) utility. The BXM process is made up of two primary jobs: compare Members in bulk (mpxcomp) and link entities (mpxlink). After running the compare and link, the data must be loaded into the database.

Step 10: Threshold analysis

The objective of this step is to arrive at a threshold showing that two matched members are same; in other words, that the two matching records are same. The following tools are used in this step:

- Pair manager: A GUI tool that helps in arriving at the threshold.
- Threshold calculator: A tool in the workbench that helps in arriving at the threshold.

Figure 7-8 shows the status of the InfoSphere MDM SE core configuration workflow after step 11.

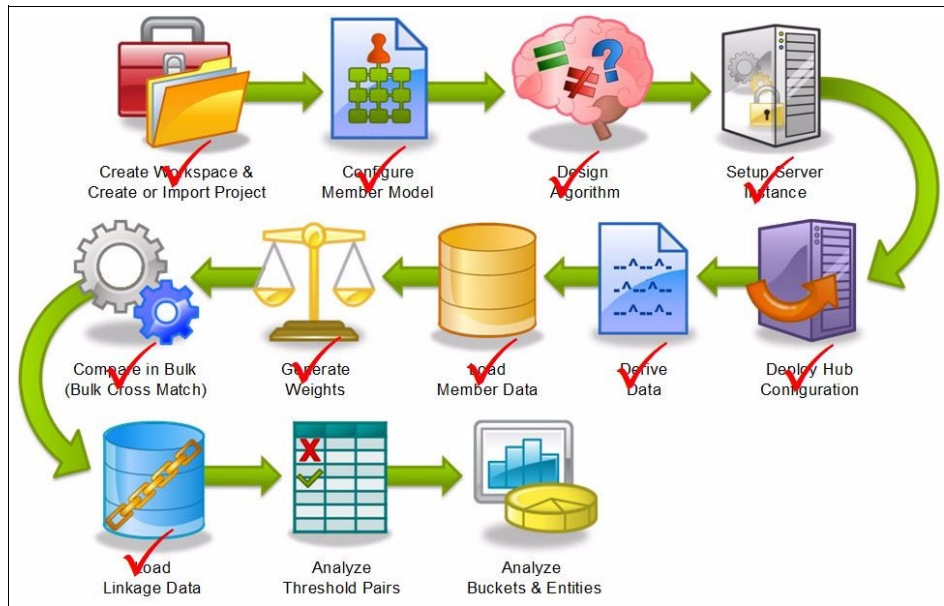


Figure 7-8 InfoSphere MDM Standard Edition workflow status

At this point, a golden view of the record or entity is created; the rest of the process is for fine tuning and report generation for analysis.

Figure 7-9 shows the final view of the enterprise application integration with InfoSphere MDM. Using SDK or Webservices of InfoSphere MDM, queries can be made to the enterprise application for viewing of the golden record.

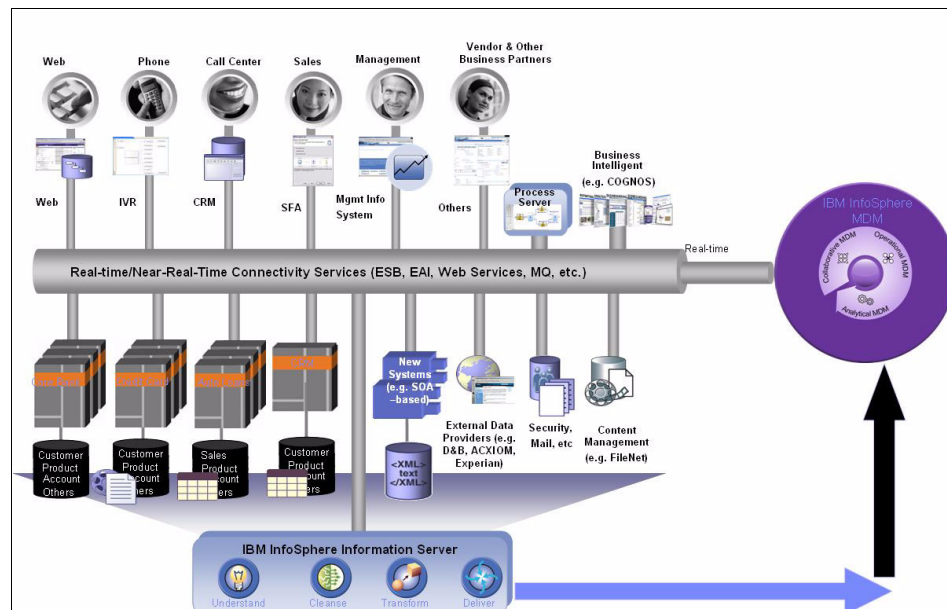



Figure 7-9 Enterprise application integrated with InfoSphere MDM



Preparing data for IBM Watson Explorer Application Builder

In this chapter, you prepare the IBM Watson Explorer Engine component for the example 360-degree information application based on the Sample Outdoor Company data. You create search collections that serve the Application Builder component configured in Chapter 9, “Creating an application with Application Builder” on page 131. After the search collections are created, you crawl the data and prepare the search collections for Application Builder integration.

The following topics are discussed:

- ▶ Creating the search collections
- ▶ Configuring the search collections and crawling the data
- ▶ Optional: Scheduling a refresh

8.1 Creating the search collections

In this section, you will create two search collections for the Sample Outdoor Company application, one for products and one containing orders.

To create the search collections, follow these steps:

1. Log in to the Watson Explorer Engine administration tool. If you used the default settings when installing Engine, you can log in to the administration tool at the following Universal Resource Locator (URL):

- Linux: `http://<serverhostname>/vivisimo/admin`
- Windows: `http://<serverhostname>/vivisimo/admin.exe`

The default user name to log in to the administration tool is `data-explorer-admin`. The default password is `TH1nk1710`.

After you are logged in to the tool, you will see a default project named `query-meta`. Along the left side of the window, you will see a menu enabling you to navigate Engine by the type of object that you want to work with. At the top of the window, you see a menu with three drop-downs:

- Configuration
- Management
- Help

2. Figure 8-1 shows the panel for the default project in the Watson Explorer administration tool. Click the Plus (+) symbol next to **Search Collections** to create a new search collection. The New Search Collection panel displays, as shown in Figure 8-2 on page 111.

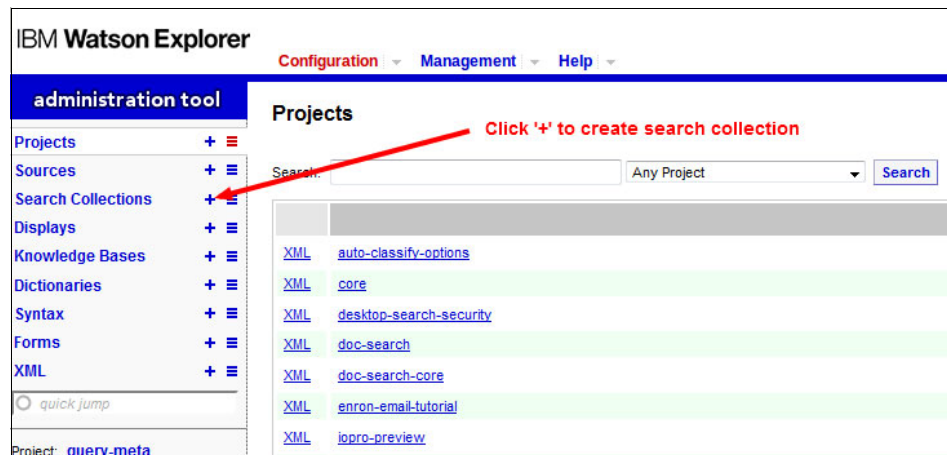


Figure 8-1 Working with the default Watson Explorer Engine project

3. In the New Search Collection panel, enter the name of the search collection that you want to create, and leave the **Copy defaults from** field set to default.

The **Copy defaults from** field enables you to create a new collection using the same settings that have already been configured in a different search collection. This can be useful if you are setting up multiple collections that require identical or very similar configurations.

The two collections that you must create for this lab are the appbuilder-products and appbuilder-order-details collections. You can only create one collection at a time, but you can quickly repeat the collection creation process to create the second collection by following the same instructions that you used when creating the first.

Configuration Management Help Logout

New Search Collection

Create a new collection

* Name: appbuilder-order-details (Name Token)

* Copy defaults from: default

Description:

Add

Figure 8-2 New Search Collection

4. To save the new search collection, click **Add**, as highlighted in Figure 8-2.

Tip: If you create a search collection and do not remember the exact name that you specified, you can list all of the search collections that exist on the Watson Explorer Engine installation that you are using. To do so, click the list icon (☰) next to **Search Collections**, as shown in Figure 8-3.

IBM Watson Explorer

Configuration Management Help

administration tool

Search Engine Collections

Search: Any Search Collection Search

	Name	Description
XML	appbuilder-order-details	
XML	appbuilder-products	

Figure 8-3 Listing search collections

8.2 Configuring the search collections and crawling the data

After you have created your collections, the next step is to configure each search collection so that it is associated with a specific data repository. You can then use Watson Explorer Engine to crawl and index that data.

8.2.1 Configuring collection seeds

The sample data used in this lab is originally from the Great Outdoors sample database (GSDB), which can be download from the IBM Integrated Data Management Information Center:

<https://ibm.biz/gsdb-sample-data>

For demonstration purposes, the data used in this lab is extracted from the GSDB, using the SQL queries shown in Example 8-1.

Example 8-1 Sample SQL queries to extract CSV data

```
-- for the "product" collection
select d.PRODUCT_NAME as title, d.*,
       p.PRODUCTION_COST, p.GROSS_MARGIN, p.PRODUCT_IMAGE
  from GOSALES.PRODUCT p
 join GOSALES.VIEW_PRODUCT_DETAILS_EN d
    on p.PRODUCT_NUMBER = d.PRODUCT_NUMBER))

-- for the "order-details" collection
select *
  from GOSALES.ORDER_DETAILS od
 join GOSALES.ORDER_HEADER oh
    on od.ORDER_NUMBER = oh.ORDER_NUMBER
 join GOSALES.ORDER_METHOD om
    on oh.ORDER_METHOD_CODE = om.ORDER_METHOD_CODE
 left outer join (select ORDER_DETAIL_CODE, count(*) as RETURN_COUNT
                  from GOSALES.RETURNED_ITEM
                  group by ORDER_DETAIL_CODE) ri
    on od.ORDER_DETAIL_CODE = ri.ORDER_DETAIL_CODE))
```

Redirect the output of these queries into files with the names that were provided earlier in this section.

For your convenience, you can download the pre-extracted data used in the examples in this book. See Appendix A, “Additional material” on page 209 for information about retrieving the sample data used in this chapter. After downloading that data, extract it to create the order-details and products directories.

Now that the CSV files are in a directory on the system where Watson Explorer Engine is running, you will add seeds to the search collections that tell Watson Explorer Engine’s crawler where to begin crawling the data.

The setup steps are same for both collections. The following instructions demonstrate creating the seed for the appbuilder-products collection:

1. Browse to the appbuilder-products collection overview window. Click the **Configuration** tab at the top row of tabs and then click the **Crawling** tab, as shown in Figure 8-4.
2. Click **Add a new seed**. The Add a new seed window displays.

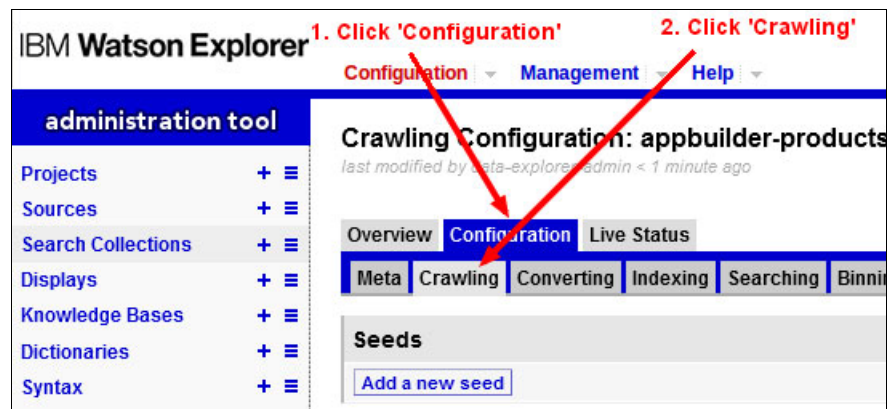


Figure 8-4 Adding a seed for a search collection

3. In the Add a new seed window, select the **Files** seed and click **Add**, as shown in Figure 8-5. The window closes, and an entry for the Files seed displays in the Seeds portion of the **Crawling** tab.

Note: A search collection can have multiple seeds, which can even connect to different repositories. In most cases, a new search collection is created for every new repository that is being crawled, but that is not a hard rule. In fact, if your application requires that a single search result is made up of data from multiple repositories, it would be advantageous to use a single collection to index the multiple repositories.

This would make it easy to merge the data from each of those repositories into what Watson Explorer refers to as *virtual documents*, which are composed of data from different sources that is combined into a single result in the index.

Figure 8-5 shows how to add a new seed.

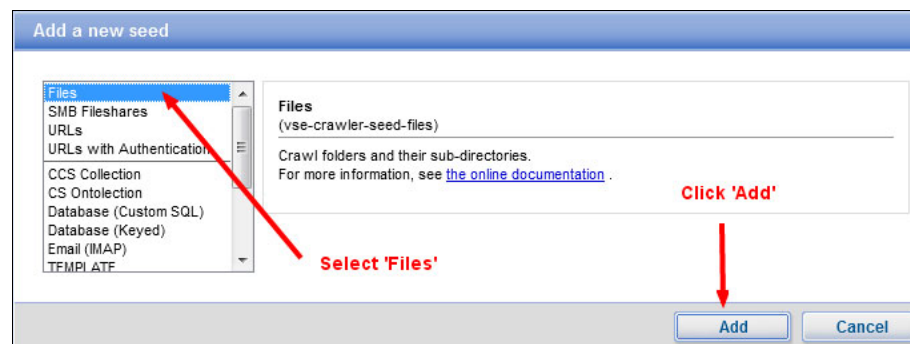


Figure 8-5 Adding the Files seed

4. In the Files portion of the Seeds section of the **Crawling** tab, enter the directory of the data files, as shown in Figure 8-6. In this example, running on a Linux system, that path is /home/labuser/products. If you are using a Windows computer, for the example, the name of the directory would be C:\labuser\products. Click **OK** to save the modified seed.

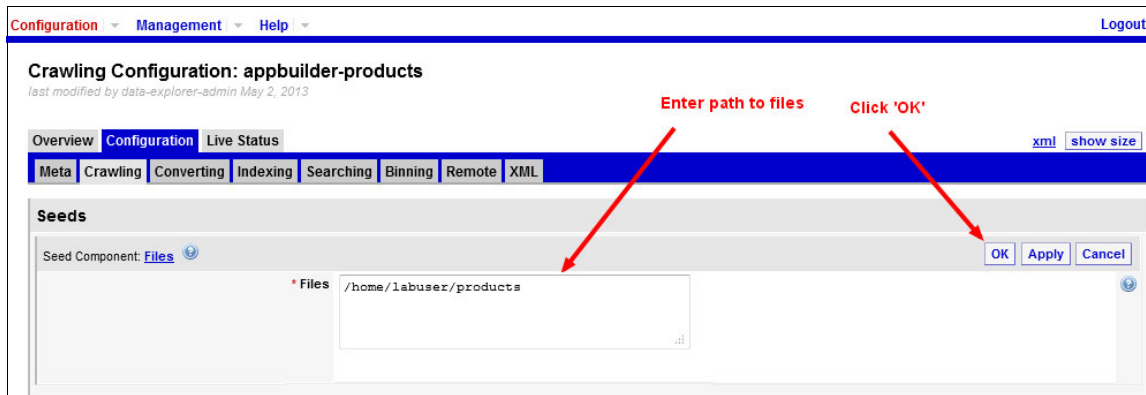


Figure 8-6 Seeds file location

8.2.2 Testing the crawling and conversion processes

Before proceeding, you should test that the Watson Explorer Engine can access, crawl, and process your sample data correctly. To verify this, perform the following steps:

1. Test that the files are in the location and you are able to crawl them. As shown in Figure 8-7, go to the **Overview** tab on the collection and click **Test It** to the right of the seed URL.

Note: You might have to start the query service before you proceed, if that service is not already running on your system. If it is not, a yellow warning banner displays when you go to the **Overview** tab. Click the **Query service** link in this banner to go to the page for that service, and click **Start** to start that service. Then you can return to the **Overview** page and test your ability to crawl the data files for this sample collection.

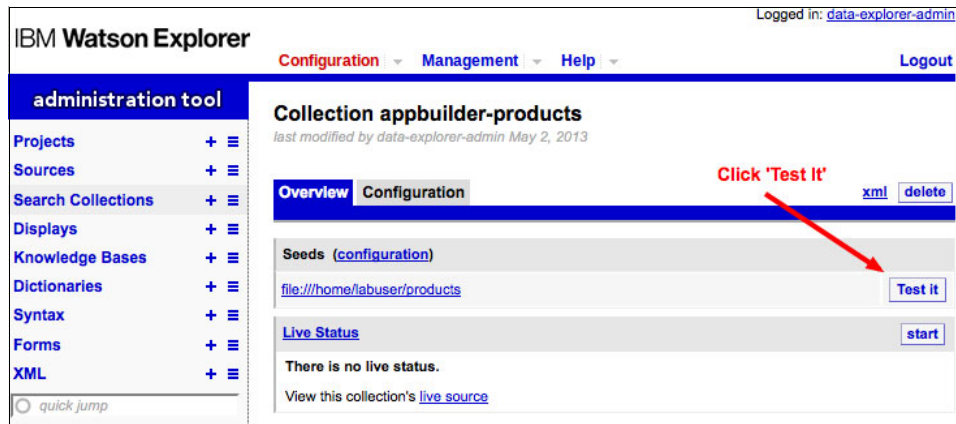


Figure 8-7 Test that the specified file can be crawled successfully

2. If everything worked, the **Converting** tab displays, showing sections that correspond to a single stage of the conversion process for the file in the directory that your seed identified. The specified directory contains a single file, products-0.csv. Click **Test It** next to that file, and Watson Explorer Engine will proceed to process that file as though it were actually crawling that directory. The result is a window showing the different converters that operate on the file, as shown in Figure 8-8 on page 117.

For more information about converters and how they work, see the section about creating and configuring Watson Explorer Engine applications in the documentation that is available in the Watson Explorer Knowledge Center:

<http://ibm.com/support/knowledgecenter/SS8NLW>

Figure 8-8 shows the converters.

IBM Watson Explorer

Configuration Management Help

Logged in: data-explorer-admin Logout

administration tool

Projects Sources Search Collections Displays Knowledge Bases Dictionaries Syntax Forms XML

quick jump

Return to:

- Collection appbuilder-products
- Collection appbuilder-order-details
- Project query-meta
- Collection appbuilder-products(wrking-copy)
- Tutorial: Combining Content into Virtual Documents
- Search the Documentation
- Search Engine Query Service
- Scheduler Service
- Collection testing
- Collection ics-default

Converting Configuration: appbuilder-products

last modified by data-explorer-admin < 1 minute ago

Overview Configurationxml show size delete

MetaCrawlingConvertingIndexingSearchingBinningRemoteXML

Trace the Conversion Process XML Mode (switch to [URL Mode](#))Test it

* XML

<corawl-url url="file:///home/labuser/products/products-0.csv" change-id="1mRoNTfL9vQCSn5dXcbXXw==" />

Content type

Force a recrawl

Enable debugging

Force allow

Enqueue all URLs

Conversion trace

file:///home/labuser/products/products-0.csv

ACL: +mongodb +db2inst1 +zk +irc +mail +sys +speech-dispatcher +messagebus +proxy +ba +syslog +libvirt-qemu +lisp +libuid +colord +libvirt-dnsmasq +kernoops +hplip +lightdm +dnsmasq +avahi +usbmux +www-data +sync +pulse +lp +news +vde2-net +sshd +whoopsie +nobody +wlp +saned +man +root +backup +games +uucp +gnats +avahi-autoipd +db2sdfe1 +bin +rtkit +daemon

76.827	76.827	unknown	unknown	Does not match the condition	Binary file extensions (filter): When url matches wc-set "#.aif "#.arc "#.aiff "#.asf "#.au "#.avi "#.bin "#.bmp "#.cab "#.class "#.dbx "#.dll "#.exe "#.fpt "#.gif "#.img "#.iso "#.jar "#.kpg "#.lib "#.max "#.mdb "#.mp4 "#.mpa "#.mpeg "#.mpg "#.mpeg "#.mov "#.moov "#.msi "#.ns2 "#.ns3 "#.ns4 "#.ocx "#.ogg "#.p65 "#.pfc "#.png "#.psd "#.qt "#.qxd "#.ra "#.ram "#.rpm "#.rm "#.sea "#.so "#.smi "#.smil "#.swp "#.sys "#.tif "#.tiff "#.tmb "#.vsd "#.wav "#.wma "#.wmv "#.yimg "#.yps "#.~\$".doc "#.~/~\$".doc "#.~/Thumbs.db	edit
76.827	76.827	unknown	text/csv		Guess content (content identifier): guess-content --type-override text/html text/html-to-utf8 '/opt/IBM/IDE/Engine/tmp/viv_en_R9kSvy' > '/opt/IBM/IDE/Engine/tmp/viv_8hFcb'	edit
76.827	123.209	text/csv	text/html		CSV to HTML: ('/opt/IBM/IDE/Engine/bin/csv2html '/opt/IBM/IDE/Engine/tmp/viv_en_R9kSvy') > '/opt/IBM/IDE/Engine/tmp/viv_vcv_AAFg0'	edit
123.209	123.209	text/html	text/html		Attribute: filetypes = html (separator [])	
123.209	184.832	text/html	application/xml		HTML to XML (html-xsl parser)	edit

Figure 8-8 Testing the conversion process for a single sample file

8.2.3 Adding a converter

If you crawl CSV files, as you are doing in for these sample collections, you must add a special CSV to VXML converter to the sequence of converters that Watson Explorer engine runs before indexing, known as the *conversion pipeline*. The data that Watson Explorer Engine crawls must be converted into an internal format known as IBM Vivisimo® XML (VXML) before it can be indexed.

For more information about VXML and the crawling and conversion process, see the section about creating and configuring Watson Explorer Engine applications in the documentation that is available in the Watson Explorer Knowledge Center:

<http://ibm.com/support/knowledgecenter/SS8NLW>

Complete the following steps to add a converter:

1. Go to the **Configuration** → **Converting** tab for the search collection, and scroll down until you see the list of converters that are currently used, and click **Add a new converter**, as shown in Figure 8-9.

IBM Watson Explorer

Configuration Management Help

Logged in: data-explorer-admin

Logout

administration tool

Projects Sources Search Collections Displays Knowledge Bases Dictionaries Syntax Forms XML

Return to:

- Collection appbuilder-products
- Collection appbuilder-order-detail
- Project query-meta
- Collection appbuilder-products(working-copy)
- Tutorial: Combining Content into Virtual Documents
- Search the Documentation
- Search Engine Query Service
- Scheduler Service
- Collection testing
- Collection ics-default

quick jump

1. Click 'Configuration' Tab

2. Click 'Converting' Tab

Converting Configuration: appbuilder-products

last modified by data-explorer-admin < 1 minute ago

Overview Configuration

Meta Crawling Converting Indexing Searching Binning Remote XML

Trace the Conversion Process URL Mode (switch to XML Mode)

* URL

Content type

Force a recrawl

Enable debugging

Force allow

Enqueue all URLs

Test it

Click 'Add a new converter'

Add a new converter

Expand all component references to explore the details of the converters.

#	Type In	Type Out	What	edit	delete
0	vivisimo/fallback	application/vxml-unnormalized	Converter Component: Unknown documents	edit	delete
1	unknown	dead	Converter Component: Binary file extensions (filter)	edit	delete
2	unknown	unknown	Type-out from: guess-content --type-override text/html text/html-to-utf8 %source_file	edit	delete

Figure 8-9 Adding a new conditional setting to a search collection

2. In the Add a new converter window, scroll down and select the CSV to VXML converter, and click **Add**, as shown in Figure 8-10.

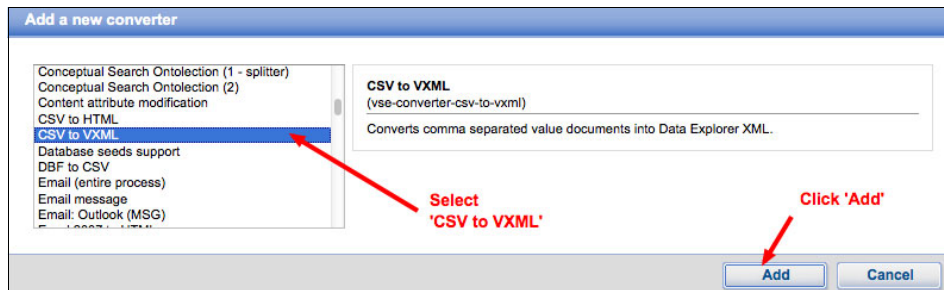


Figure 8-10 Adding a new converter

3. A new converter displays at the top of the list of current converters. Verify that the **Type-In** field specifies text/csv, and that the **Type-Out** field specifies application/vxml-unnormalized. Click **OK** to collapse this section and add the CSV to VXML converter to the conversion pipeline for this collection.
4. To verify that all is working properly, go back to the **Overview** tab of the search collection and repeat the **Test It** process that was described in 8.2.2, "Testing the crawling and conversion processes" on page 115.

You should see that new converter is being properly invoked, as shown in Figure 8-11.

Converting Configuration: appbuilder-products

last modified by data-explorer-admin < 1 minute ago

OverviewConfiguration

xmlshow sizedelate

MetaCrawlingConvertingIndexingSearchingBinningRemoteXML

Trace the Conversion Process XML Mode (switch to URL Mode)

Test it

* XML

<crawl-url url="file:///home/labuser/products/products-0.csv" change-id="lmRoNTfL9vQCSn5dXcbXXw==" />

Content type

Force a recrawl

Enable debugging

Force allow

Enqueue all URLs

Conversion trace

file:///home/labuser/products/products-0.csv

ACL: +mongodb +db2inst1 +zk +irc +mail +sys +speech-dispatcher +messagebus +proxy +ba +syslog +libvirt-qemu +list +libuuid +colord +libvirt-dnsmasq +kernoops +hplip +lightdm +dnsmasq +avahi +usbmux +www-data +sync +pulse +lp +news +vde2-net +sshd +whoopsie +nobody +wlp +saned +man +root +backup +games +uucp +gnats +avahi-autoipd +db2sdfe1 +bin +rtkit +daemon

76.827	76.827	Does not match the condition	Binary file extensions (filter): When url matches wc-set *#.aif *#.arc *#.aiff *#.asf *#.au *#.avi *#.bin *#.bmp *#.cab *#.class *#.dbx *#.dll *#.exe *#.fpt *#.gif *#.img *#.iso *#.jar *#.kpg *#.lib *#.max *#.mdb *#.mp4 *#.mpa *#.mpg *#.mpeg *#.mov *#.moov *#.msi *#.ns2 *#.ns3 *#.ns4 *#.ocx *#.ogg *#.p65 *#.pfc *#.png *#.psd *#.qt *#.qxd *#.ra *#.ram *#.rpm *#.rm *#.sea *#.so *#.smi *#.smil *#.swp *#.sys *#.tif *#.tiff *#.tmb *#.vsd *#.wav *#.wma *#.wmv *#.ymg *#.yps *#-~\$*.doc *#/~\$*.doc *#/Thumbs.db	edit
76.827	76.827		Guess content (content identifier): guess-content --type-override text/html text/html-to-utf8 '/opt/IBM/IDE/Engine/tmp/viv_en_aE6GN1' > '/opt/IBM/IDE/Engine/tmp/viv_Uzje8u'	edit
76.827	671.892		CSV to VXML: ('/opt/IBM/IDE/Engine/bin/csv2vxml' '/opt/IBM/IDE/Engine/tmp/viv_en_aE6GN1') > '/opt/IBM/IDE/Engine/tmp/viv_vcv_57F7uY'	edit
671.892	687.367		Normalization (xsl parser)	edit

Figure 8-11 Verifying that the CSV to VXML converter is being executed

8.2.4 Final search collection configuration

Two final configuration steps are required so that each of your search collections is ready for the 360-degree information application:

- ▶ Configure the search collection to fast-index the content fields that you reference later in your Application Builder entity model.
- ▶ Configure the search collection to generate dictionaries, which are used by the query autocomplete feature of Application Builder.

By configuring search collections to fast-index specific data from search results, you notify the collection to store these fields in memory for fast retrieval and manipulation. This is necessary for fields that you use in an entity definition when you configure Application Builder.

To configure the collection to fast-index content and generate dictionaries, perform the following steps:

1. Go to the **Configuration** → **Indexing** tab for the search collection.
2. Click **Edit** to enable you to modify the setting used when indexing data.
3. Configure the indexing options in the General section and the dictionary-related options in the Term expansion support areas, as shown in Figure 8-12 on page 122:
 - a. Select **Modify**.
 - b. Enter the appropriate information.
 - c. Expand **Term expansion support**.
 - d. Set **Generate dictionaries** to true.
 - e. Click **OK**.

Indexing Configuration: appbuilder-products
last modified by data-explorer-admin 9 minutes ago

Overview **Configuration** Live Status [xml](#) [show size](#)

Meta **Crawling** **Converting** **Indexing** Searching Binning Remote XML

Global Settings OK Apply Cancel

▼ General (4)

Link analysis steps repetitions
 Default: 100 repetitions

1. Click 'Modified' Fast Index ☒ Modified ☐ Default

2. Enter information

5. Click 'OK' OK Apply Cancel

Indexed Fast Index
 Default: false

Maximum content bytes
 Default: 100,000,000 bytes

► Indices (9)

► Duplicate filtering (4)

4. Set 'Generate dictionaries' to true

▼ Term expansion support (4)

3. Expand 'Term expansion support' Generate dictionaries
 Default: false

Dictionary directory
 Default: expansions

Dictionary stemmers
 Default: case

Dictionary delanguage
 Default: true

Figure 8-12 Configuring indexing and term expansion (dictionary) options for a search collection

4. Click **OK** to save your changes to the indexing and term expansions options.
 If you have already crawled your data, you might receive a yellow warning message in the administration tool indicating that you must restart the indexing service. If you receive this message, follow the instructions provided on the window and click **Restart**.

8.2.5 Crawling the data

After configuring your search collection, complete the following steps to crawl the data:

1. Go to the **Overview** tab on the search collection and click **Start**, as shown in Figure 8-13.

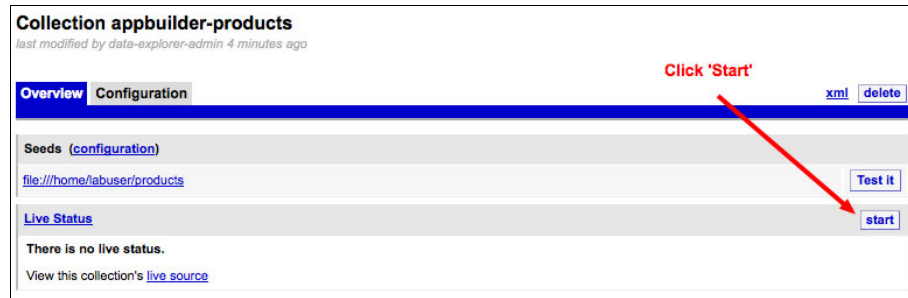


Figure 8-13 Starting a Watson Explorer Engine crawl

While the data is being crawled, the area under the Live Status link turns green. When the crawl is finished, it should look similar to Figure 8-14.

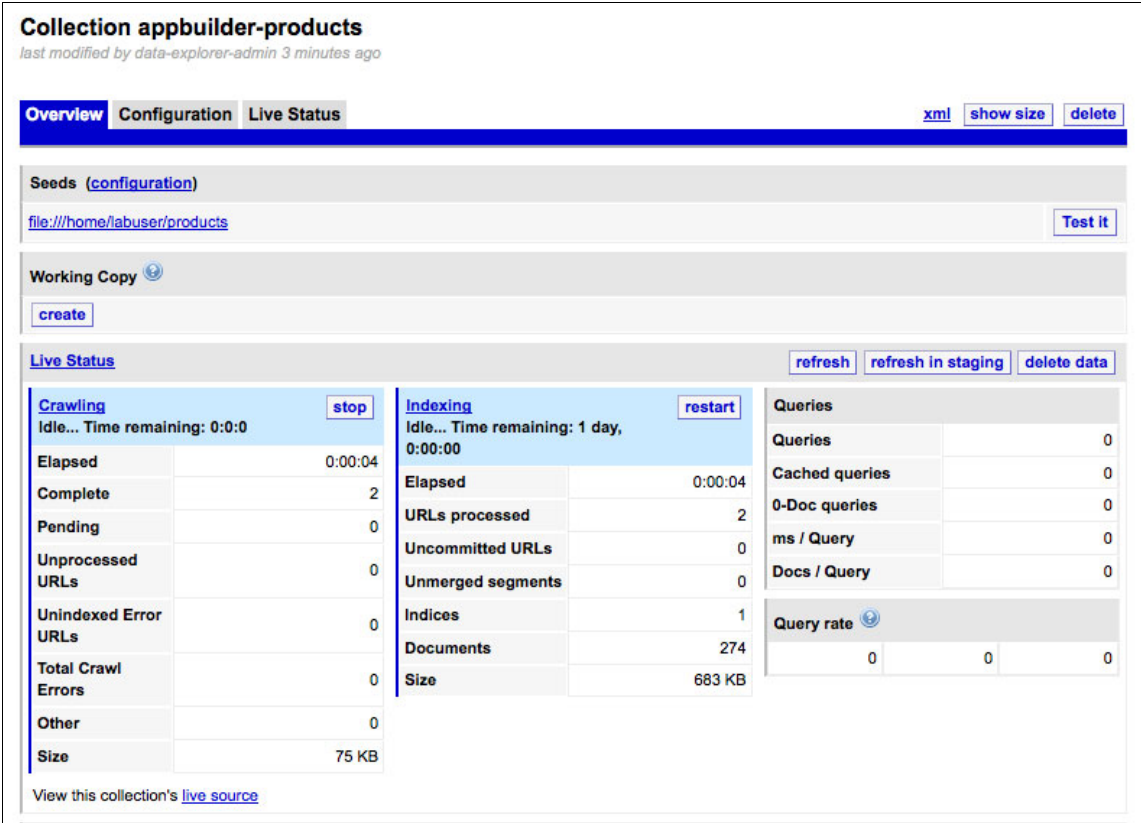


Figure 8-14 Viewing summary and status information for a completed crawl

- To verify that the data is indexed and usable, query the search collection within Watson Explorer Engine and make sure that data is returned. To do so, click **Search** on the left side while still on the Overview page of the search collection, as shown in Figure 8-15.

IBM Watson Explorer

Configuration Management Help

administration tool

Projects Sources Search Collections Displays Knowledge Bases Dictionaries Syntax Forms XML

quick jump

Test with project **query-meta**:

live staging Search

Return to:

- Collection appbuilder-products
- Scheduler Service
- Collection appbuilder-order-details
- Project query-meta
- Collection appbuilder-products(working-copy)
- Tutorial: Combining Content into Virtual Documents
- Search the Documentation
- Search Engine Query Service
- Collection testing
- Collection ics-default

Collection appbuilder-products

last modified by data-explorer-admin 6 minutes ago

Overview Configuration Live Status

Seeds (configuration)

file:///home/labuser/products

Working Copy

create Click 'Search'

Live Status

Crawling resume

Complete May 29, 2013 14:42:24.

Elapsed	0:00:05
Complete	2
Pending	0
Unprocessed URLs	0
Unindexed Error URLs	0
Total Crawl Errors	0
Other	0
Size	75 KB

View this collection's [live source](#)

Staging Status

There is no staging status.

View this collection's [staging source](#)

Figure 8-15 Testing a collection with a sample query

When a search completes without error, a tab opens to a search result page. You should see search results corresponding to the products in the sample collection, as shown in Figure 8-16 on page 126.

turn debugging on

IBM Watson Explorer

Search

Results 1-10 of about 273 | Details

Topic Clusters

Top 120 Results

remix

+ Camping Equipment (44)

+ Unspecified, Personal Accessories (34)

+ Mountaineering Equipment (19)

+ Outdoor Protection (14)

+ Watches, Strap (9)

+ Golf Equipment (12)

+ Edge, Stainless steel (6)

+ Comes With A Life-Time Warranty (2)

+ Other Topics (1)

1. [Venue](#) [new window](#) [preview](#)

Product_description:

The Venue has a fun and fashionable oversized watch face, with a bright colored synthetic leather strap.

Base_product_number:

125

Product_line:

Personal Accessories

Production_cost:

0.00

Product_image:

P65PA3WT10.jpg

Product_type:

Watches

Color:

Red

Gross_margin:

0.0

Product_line_code:

993

Product_name:

Venue

Introduction_date:

2005-04-01 00:00:00.0

Product_type_code:

960

Product_size:

One-size

Brand:

Relax

Product_brand_code:

756

Product_size_code:

853

Product_color_code:

922

Product_number:

125150

file:///home/labuser/products/products-0.csv - 75K - [cache](#) - appbuilder-products

2. [TrailChef Water Bag](#) [new window](#) [preview](#)

Product_description:

Lightweight, collapsible bag to carry liquids easily. Wide mouth for easy filling. Holds 10 liters.

Base_product_number:

1

Product_line:

Camping Equipment

Production_cost:

4.00

Product_image:

P01CE1CG1.jpg

Product_type:

Cooking Gear

Color:

Clear

Gross_margin:

0.33

Product_line_code:

991

Product_name:

TrailChef Water Bag

Introduction_date:

1995-02-15 00:00:00.0

Product_type_code:

951

Product_size:

10 liters

Brand:

TrailChef

Product_brand_code:

701

Product_size_code:

808

Product_color_code:

908

Product_number:

1110

file:///home/labuser/products/products-0.csv - 75K - [cache](#) - appbuilder-products

3. [TrailChef Canteen](#) [new window](#) [preview](#)

Product_description:

Aluminum canteen. Rugged fleece-lined cover with belt clips, removable shoulder sling and small pocket for water purification tablets bottle. Holds 2 liters

Base_product_number:

2

Product_line:

Camping Equipment

Production_cost:

9.22

Product_image:

P02CE1CG1.jpg

Product_type:

Cooking Gear

Figure 8-16 Search results from indexed collection data

Now that you have configured the appbuilder-products collection, you can configure the appbuilder-order-details collection using the same steps.

126 Building 360-Degree Information Applications

8.3 Optional: Scheduling a refresh

In Watson Explorer installations that index live data, the data repositories upon which your search collections are based are likely to be changing. One way to keep the search collection index up to date is by creating a scheduled task in the Watson Explorer Engine administration tool. This task will automatically start a “refresh” of the search collection at a time interval of your choosing.

This section walks you through setting up this type of task in the Watson Explorer Engine administration tool. Follow these steps to create a scheduled task:

1. In the administration tool, select **Services** from the **Management** drop-down list (Figure 8-17).

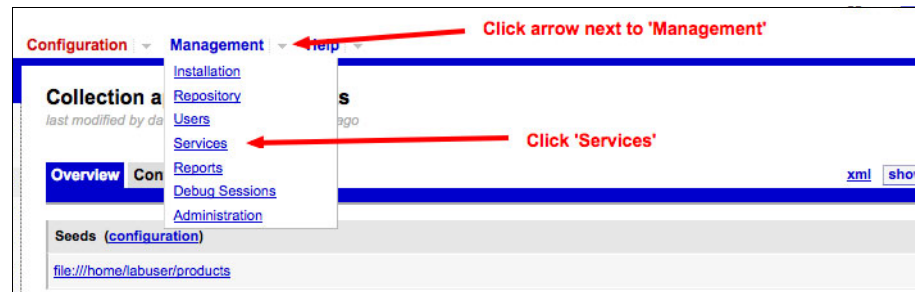


Figure 8-17 Accessing Watson Explorer Engine services

2. Click **Add Scheduled Task**, as shown in Figure 8-18. The Add Scheduled Task window displays.

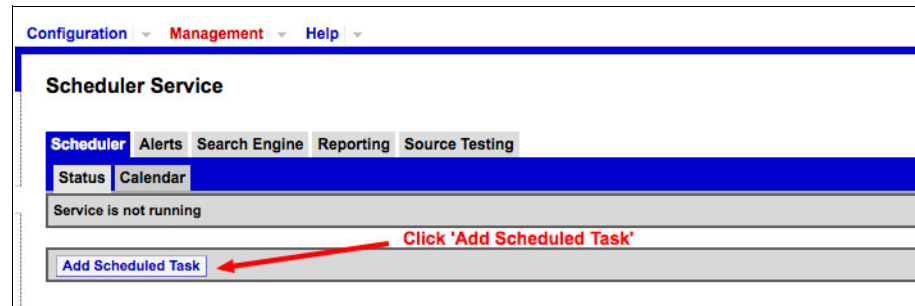


Figure 8-18 Adding a new scheduled task

3. In the **Add Scheduled Task** window, select **Start a Search Collection crawl** and then click **Add**, as shown in Figure 8-19.

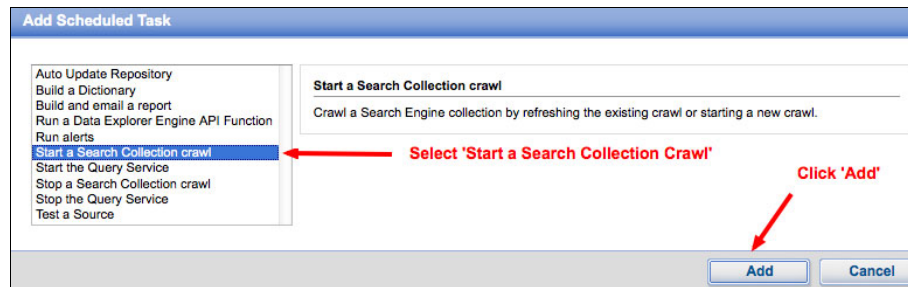


Figure 8-19 Specifying the type of scheduled task to add

4. In the **Collection** field, enter the name of the collection that you want the scheduled task to operate on. Choose **Refresh the crawl** for the Crawl type option (Figure 8-20).

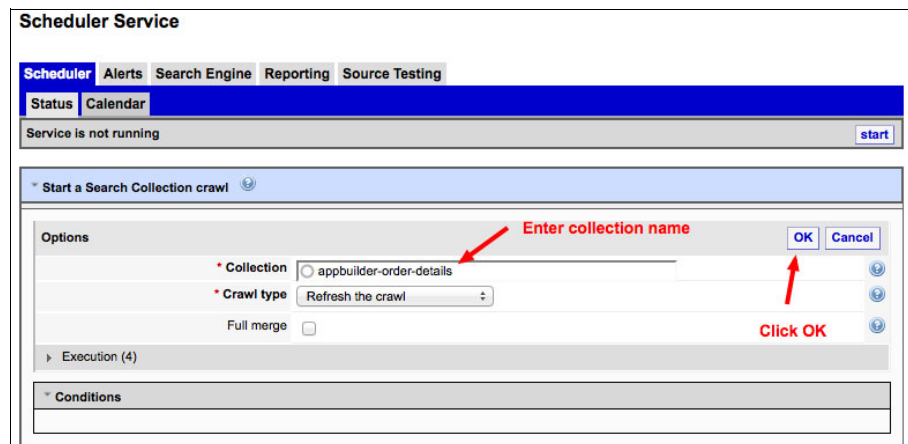


Figure 8-20 Identifying the collection and crawl type for a crawler task

- Expand the task and the **Conditions** portion of the window and click **Add Scheduled Time**, as shown in Figure 8-21. The Add Scheduled Time window displays.

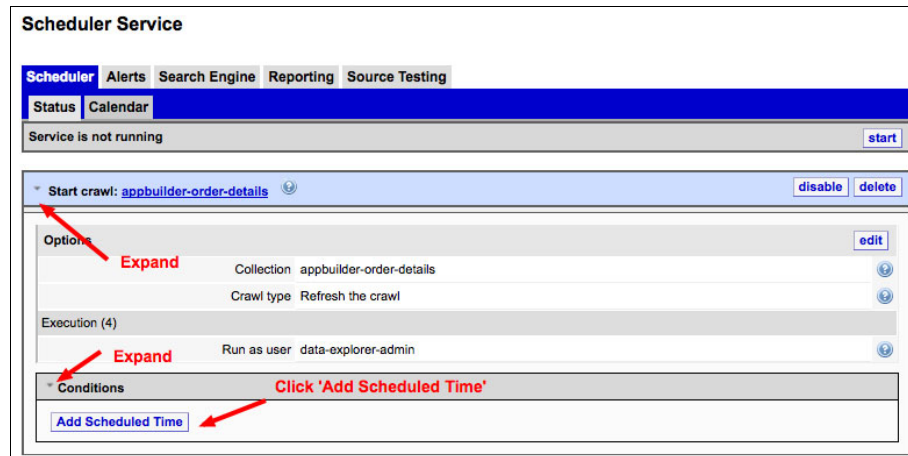


Figure 8-21 Scheduling the time for a recurring task

- In the Add Scheduled Time window, shown in Figure 8-22, choose the interval that you want the scheduled task to run on. For this example, you choose **Daily**. Click **Add**.

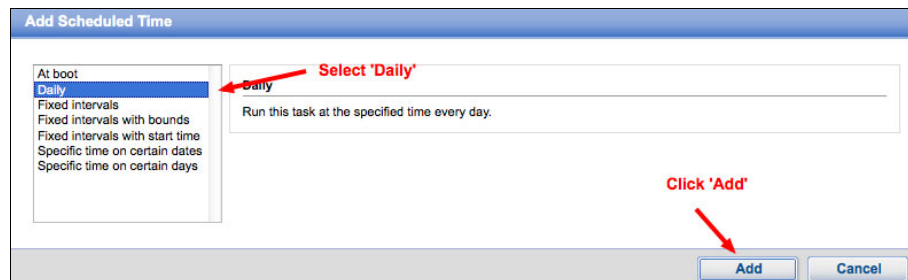


Figure 8-22 Specifying execution frequency for a scheduled task

7. Enter the time of day at which you would like the scheduled task to run, and click **OK**, as shown in Figure 8-23.

The screenshot shows the 'Scheduler Service' configuration window. At the top, there are tabs for 'Scheduler', 'Alerts', 'Search Engine', 'Reporting', and 'Source Testing'. Below these are sub-tabs for 'Status' and 'Calendar'. A status bar indicates 'Service is not running' with a 'start' button. A blue bar contains the text 'Start crawl: [gppbuilder-order-details](#)'. The main area is divided into sections: 'Options' (Collection: appbuilder-order-details, Crawl type: Refresh the crawl), 'Execution (4)' (Run as user: data-explorer-admin), and 'Conditions'. The 'Conditions' section is expanded, showing a 'Daily' schedule. A time picker is set to 12:00:00 AM. A red arrow points to the time picker with the text 'Enter time'. Another red arrow points to the 'OK' button with the text 'Click \'OK\''. The 'OK' and 'Cancel' buttons are at the bottom right of the 'Conditions' section.

Figure 8-23 Specifying the time of day for scheduled task execution

8. The scheduler service must be running to start the scheduled task at its next defined interval. To start the scheduler service, click **start** in the upper right of the window on the Scheduler Service page, as shown in Figure 8-23.



Creating an application with Application Builder

In this chapter, you create an example 360-degree information application using IBM Watson Explorer Application Builder.

You use the work done in Chapter 8, “Preparing data for IBM Watson Explorer Application Builder” on page 109, for access to Sample Outdoor Company data. You use information describing products and order details. Application Builder uses entities to structure that data. You then configure widgets for displaying data details, and configure web pages to assemble related widgets into a meaningful presentation for a user.

The following topics are discussed:

- ▶ Defining entities
- ▶ Enhancing an entity page
- ▶ Searching entities
- ▶ Conclusion

9.1 Defining entities

Application Builder's management console enables you to set up data sources and relationships. You also use the management console to set up how information is displayed to users. You create pages and populate pages with widgets. Each widget is focused on providing a specific view of information.

9.1.1 Logging in

To log in, follow these steps:

1. In a browser, go to `http://localhost:8080/AppBuilder`.

Because of the time required for application server initialization, loading this first page might take several seconds.

2. To go to the administration console, select **data-explorer-admin** → **Manage Watson Explorer application** as shown in Figure 9-1.

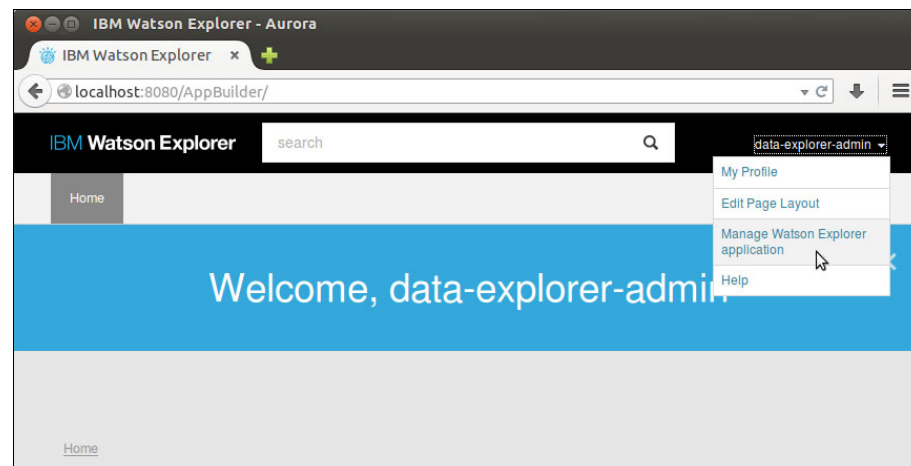


Figure 9-1 Go to the management console

9.1.2 Connecting the engine

The Welcome page of Application Builder management console, shown in Figure 9-2 on page 133, gives an overview of the tasks that you must perform:

1. Click **Get Started** to begin.

First, you need to add a data source into the Application Builder configuration. In this lab, this data source is the Watson Explorer Engine and associated collections that were configured in the previous chapter.

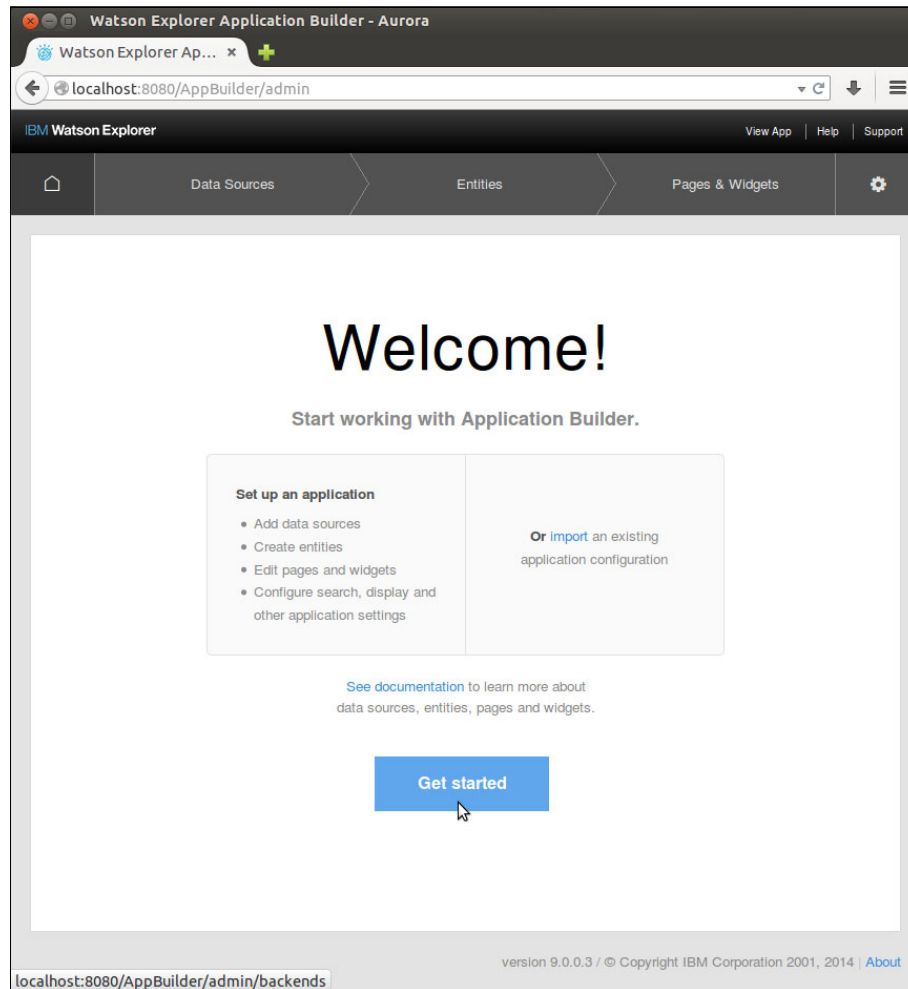
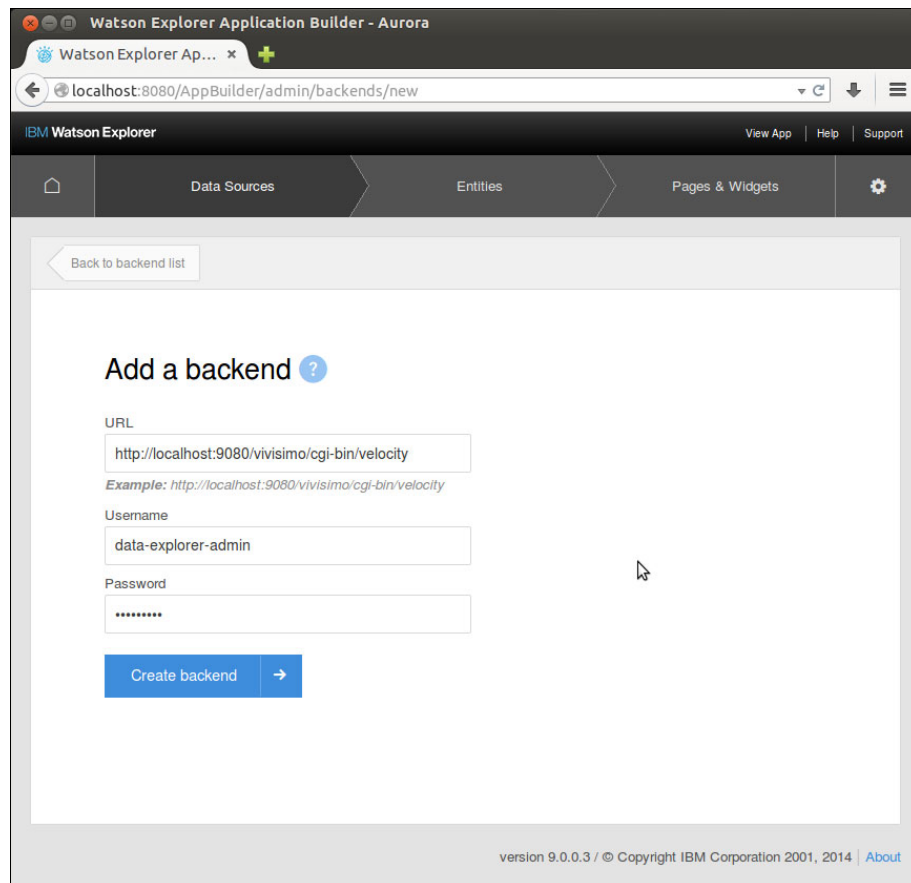


Figure 9-2 Application Builder management console Welcome page

2. On the **Data Sources** page, select **Add a backend**.
3. Complete the three fields, **URL**, **Username**, and **Password**, as shown in Figure 9-3 on page 134. A slight difference exists in the URL used on Linux compared to Windows:
 - For Linux, use `http://<localhost>:9080/vivisimo/cgi-bin/velocity`
 - For Windows, use `http://<localhost>:9080/vivisimo/cgi-bin/velocity.exe`

4. For the purposes of this exercise, the **Username** is data-explorer-admin and the **Password** is TH1nk1710. After the three fields are completed, click **Create Backend**, as shown in Figure 9-3.



The screenshot shows the 'Watson Explorer Application Builder - Aurora' web interface. The browser address bar displays 'localhost:8080/AppBuilder/admin/backends/new'. The application header includes 'IBM Watson Explorer' and navigation links for 'View App', 'Help', and 'Support'. A main navigation bar contains 'Data Sources', 'Entities', and 'Pages & Widgets'. The central content area is titled 'Add a backend' with a help icon. It features three input fields: 'URL' (containing 'http://localhost:9080/vivisimo/cgi-bin/velocity'), 'Username' (containing 'data-explorer-admin'), and 'Password' (masked with dots). An example URL is provided below the first field. A blue 'Create backend' button with a right arrow is at the bottom. The footer shows 'version 9.0.0.3 / © Copyright IBM Corporation 2001, 2014 | About'.

Figure 9-3 Adding a new backend

9.1.3 Creating an entity

Entity definitions are the primary building blocks of Application Builder's data model. From the work in Chapter 8, "Preparing data for IBM Watson Explorer Application Builder" on page 109, you have data available in the Watson Explorer Engine installation.

Now you create an entity in Application Builder that enables us to use that data:

1. Go to **Entities** and click **Create new entity**.
2. Specify product as the entity name in the **Name entity** field.
3. In the **Pick a data store** field, make sure that **Collection** is selected, and choose the collection named **appbuilder-products**.
4. Click **Create Entity**, as shown in Figure 9-4.

The screenshot shows the 'Create an entity' form in the Watson Explorer Application Builder. The browser address bar indicates the URL is localhost:8080/AppBuilder/admin/entities/new. The navigation bar includes 'Data Sources', 'Entities', and 'Pages & Widgets'. The form has a 'Back to entities list' button at the top left. The main section is titled 'Create an entity' and contains two parts: 'Name entity' and 'Connect to data'. In the 'Name entity' section, the text 'product' is entered in a text box. Below it is a note: 'Pick an informative, singular noun that describes the type of data this entity represents. This value will be displayed in the application and cannot be changed.' The 'Connect to data' section has a 'Data store' label and a 'Choose a type:' section with four radio buttons: 'Collection' (selected), 'Federated', 'Cluster Collection', and 'Directory'. Below this is a 'Collection:' dropdown menu with 'appbuilder-products' selected. At the bottom of the form is a blue 'Create entity' button with a right-pointing arrow. A mouse cursor is hovering over the button.

Figure 9-4 Creating the product entity

9.1.4 Viewing entities

After an entity is set up, the application you are building begins to take shape. To look at what you have so far, follow these steps:

1. Click **View App** at the top of the page, toward the right. Notice the new **Products** tab under the **IBM Watson Explorer** heading.
2. Click **Products** to see a list of available products, as shown in Figure 9-5.

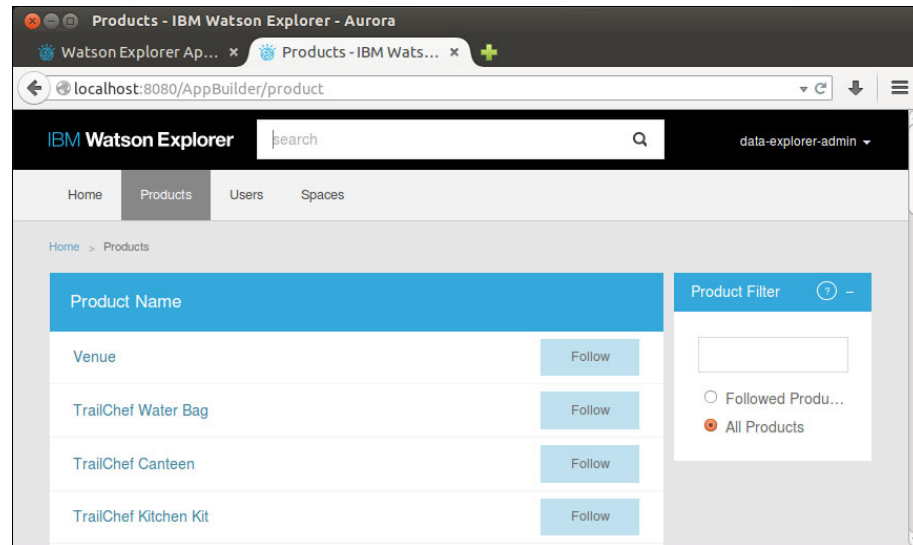


Figure 9-5 Product list

3. Click one of the products in the list, such as **TrailChef Water Bag**, to see the page that is associated with that product, as shown in Figure 9-6.

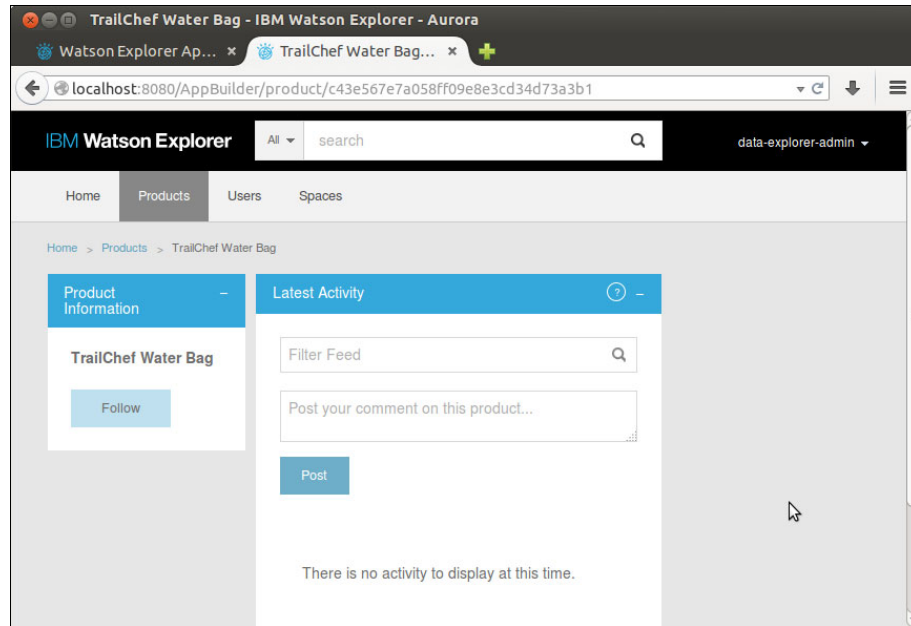


Figure 9-6 Viewing a simple product page

9.2 Enhancing an entity page

An entity page, such as the product page that you just created, is used to see details about a specific entity. You enhance the simple product page by adding more fields into the **Product Information** widget. You further enhance the product page by adding a subordinate entity for product orders, and displaying a few views of order history data to the product page.

9.2.1 Enhancing an entity information widget

To enhance the product page, follow these steps:

1. In the Application Builder management console, go to the **Pages & Widgets** section and click the triangle to the right of **Product** → **Detail page**.
2. Click **product_information** in the Available Widgets section, as shown in Figure 9-7.

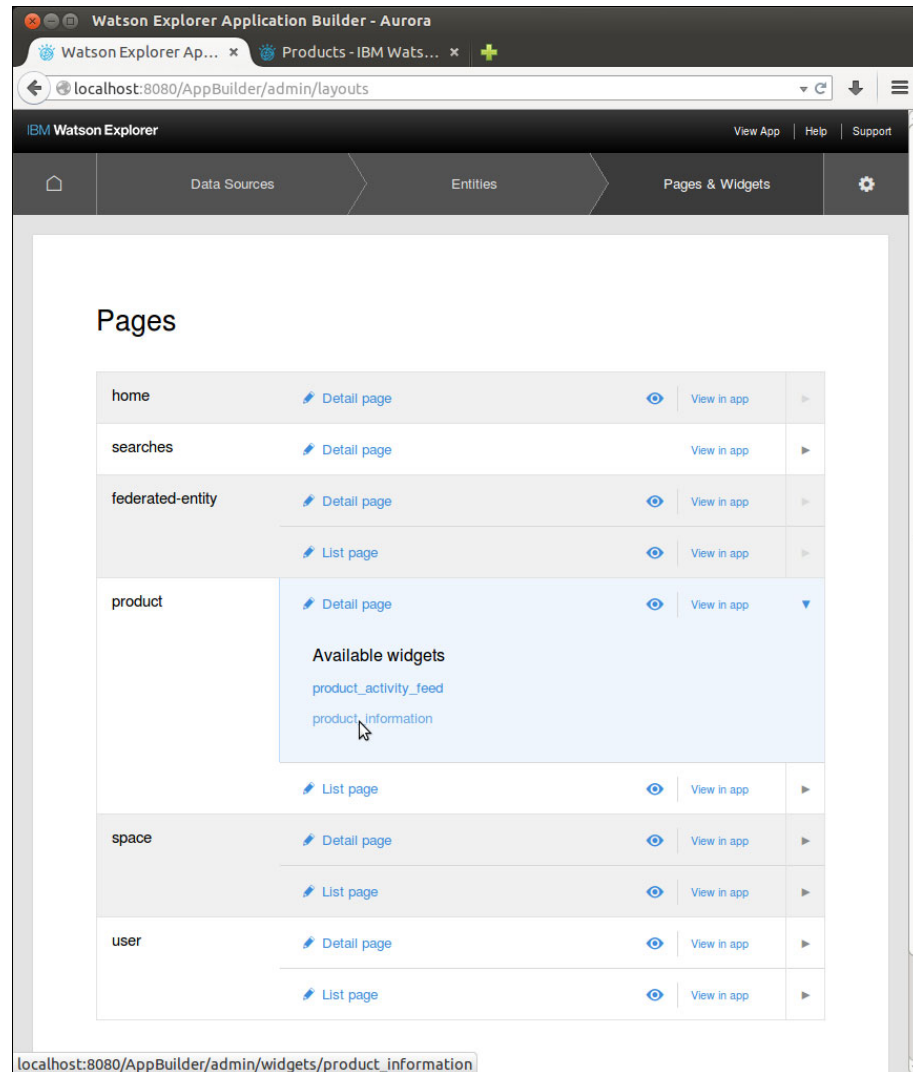


Figure 9-7 Listing pages that are focused on the product entity

3. On the Edit Entity Information widget / product_information page, shown in Figure 9-8, click in the box labeled **Choose existing fields to display** and choose several fields:
- product_name
 - product_line
 - brand
 - product_type
 - color
 - product_size
 - production_cost
 - gross_margin

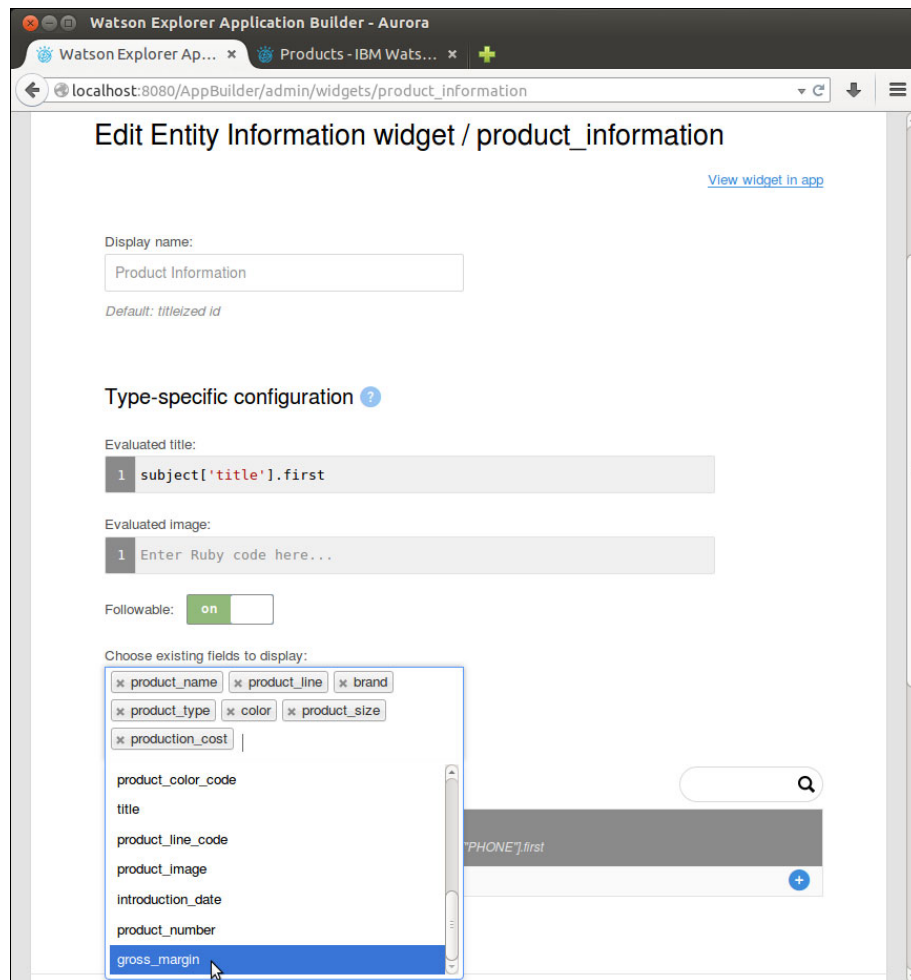


Figure 9-8 Adding other product information fields to display

4. Click **Save Changes** at the bottom of the page to ensure that your changes are permanent.
5. After saving your changes, click **View Widget in App** in the upper right corner of the page to see the enhanced product information widget (Figure 9-9).

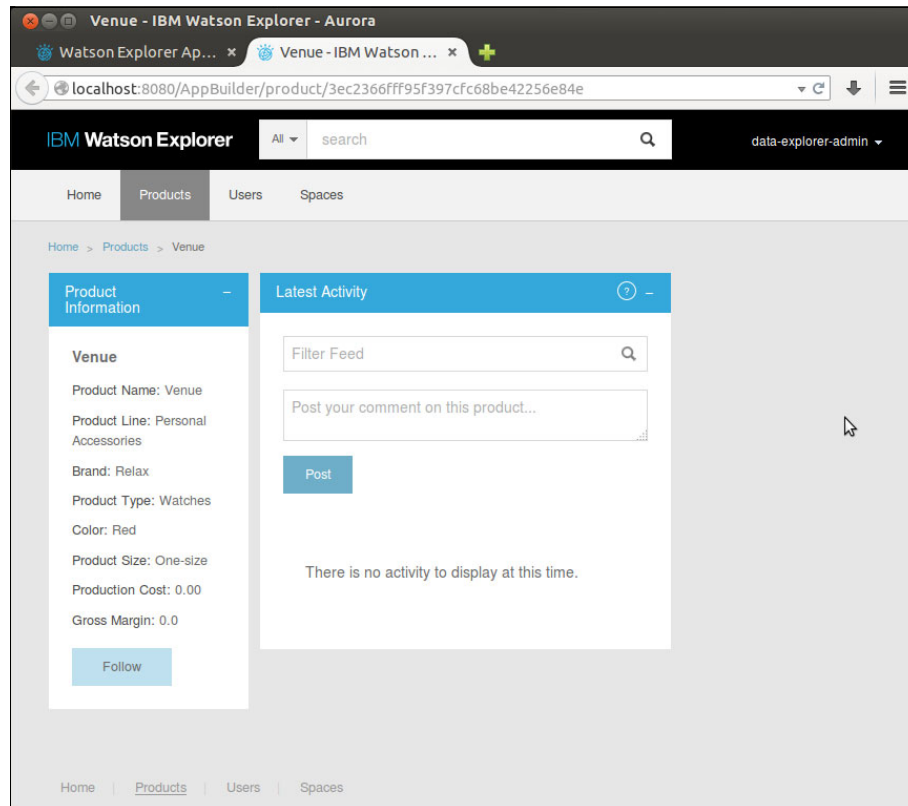


Figure 9-9 Viewing additional information in the enhanced product information widget

9.2.2 Adding an order-detail entity

You identified the search collection that provides product information in 9.1, “Defining entities” on page 132, and expanded the information that you display from it in 9.2, “Enhancing an entity page” on page 137. The setup made *Products* a key entity, with an index page that lists products and a focused page that provides detailed information for each product.

In this section, you create a subordinate entity that provides information about order details for each product. You create this as a subordinate entity because the application only uses orders to augment information in relation to other entities (in this case, products).

To add a subordinate entry, follow these steps:

1. In the management console, go to the **Entities** section and click **Create a new entity**. This redisplay the **Create an entity** page that you saw in Figure 9-4 on page 135.
2. This time, complete the **Name entity** field with `order-detail`, pick the **Collection** named `appbuilder-order-details`, and click **Create Entity**.

As shown in Figure 9-10, creating the `order-detail` entity caused a list page, a detail page, and widgets to be generated.

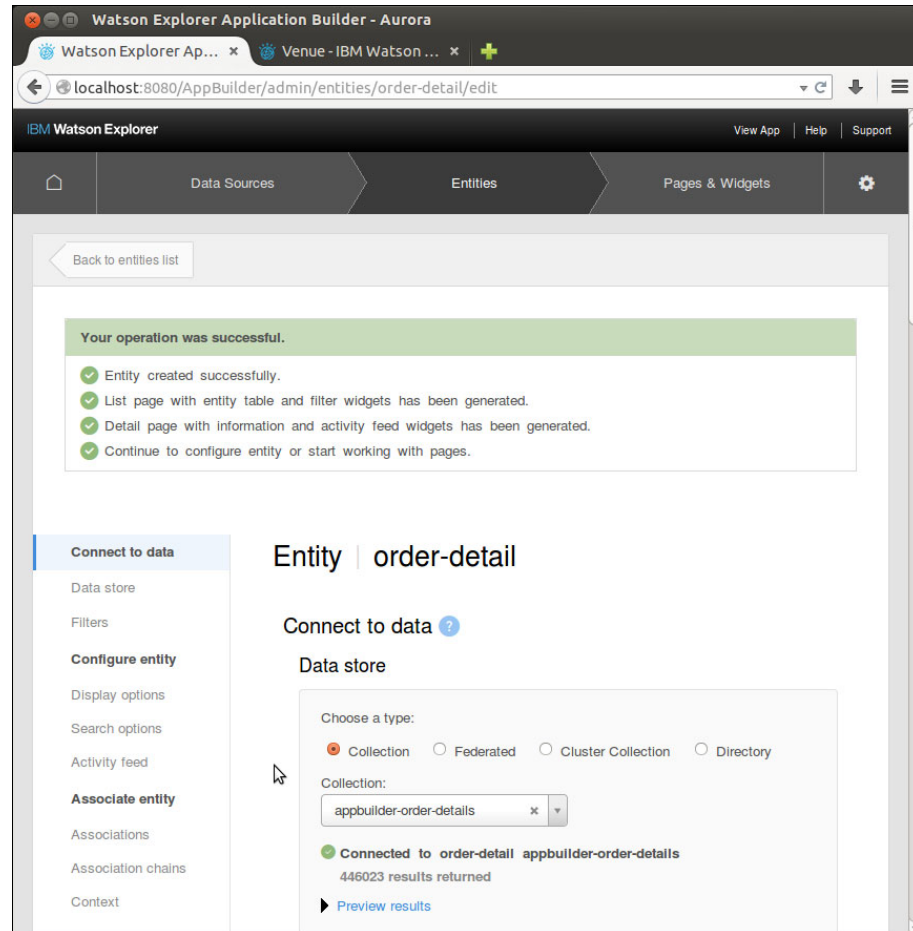


Figure 9-10 Creating pages and widgets as part of entity creation

As mentioned earlier, you only want order details to enhance the product detail page, rather than displaying them as stand-alone pages. You therefore hide the order detail pages to avoid distracting the users of the application that you are building.

3. To hide these pages, go to the **Pages & Widgets** section in the management console and click the visibility icons for **order-detail Detail page** and **List page**, as shown in Figure 9-11.

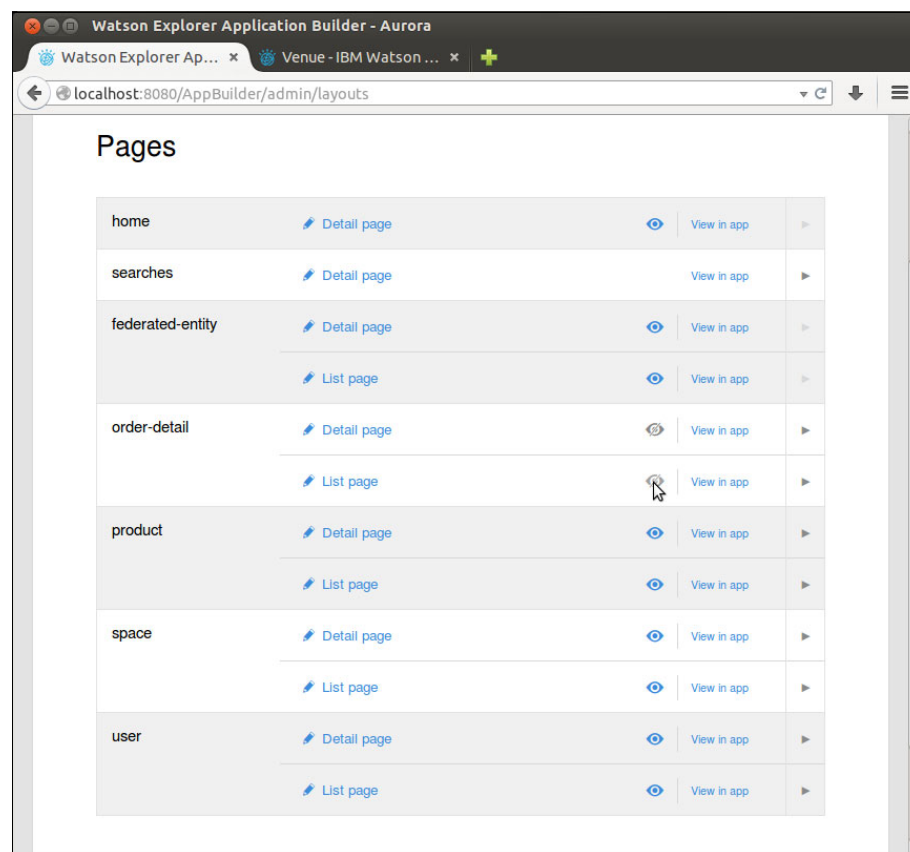


Figure 9-11 Hiding order-detail and list pages

9.2.3 Associating products with order details

To use order detail data on a product detail page, you need to set up an association from the product entity to the order-detail entity.

To add this association, follow these steps:

1. Go to the **Entities** section of the management console and select the existing product entity.
2. Scroll to the bottom of the page to find **Associations**.
3. Specify the following details, as shown in Figure 9-12:
 - a. Enter orders as the **Association name**.
 - b. Choose product_number for the **From field**.
 - c. Choose order-detail for the **To entity**.
 - d. Choose product_number for the **To field**.
4. Click **Save Entity** to make your changes permanent.

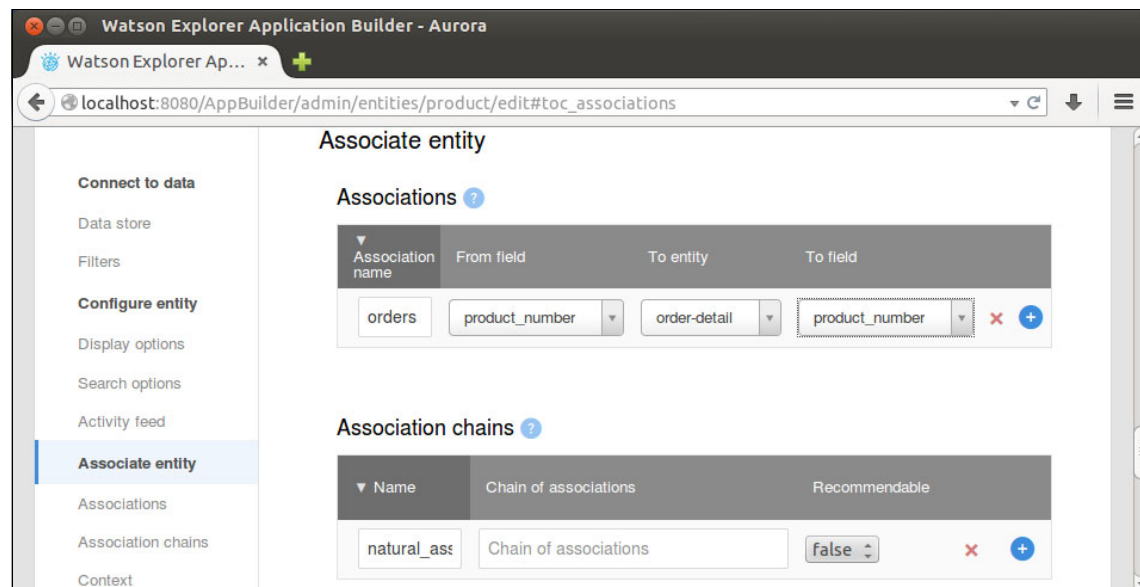


Figure 9-12 Adding an association

Adding this association created a new product widget named `orders_list`. You can now place that widget on the product detail page to see what it shows.

5. Go to **Pages & Widgets** and click **Detail page** for **product**.
6. Click and drag the `orders_list` widget from the available widgets box to the box on the lower right, as shown in Figure 9-13. Click **Save Changes**.

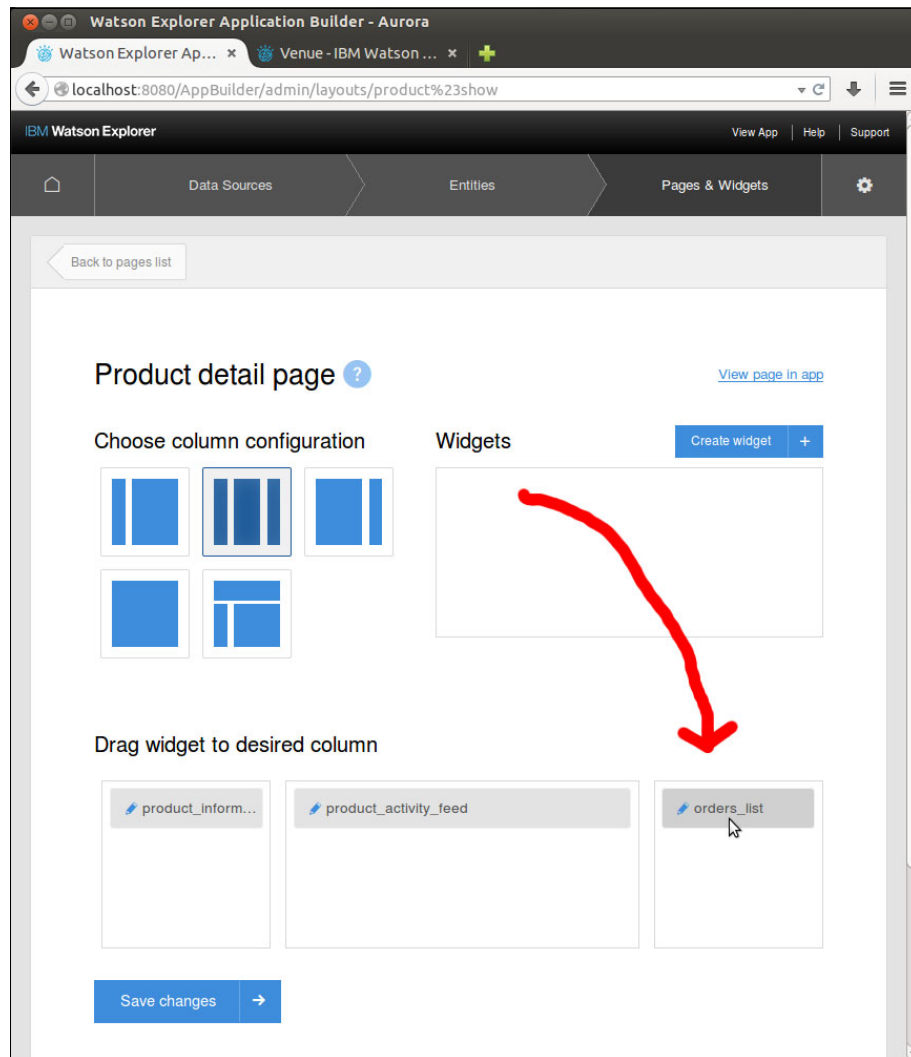


Figure 9-13 Dragging a widget to a specific column of a page

7. After saving those changes, click **View Page in App** to see the primitive `orders_list` widget that you just created.

9.2.4 Formatting order details

You can improve the list of purchases shown in the `orders_list` widget by specifying what fields you want to see for the `order-detail` entity type. The default field shown when displaying any entity is the **title** field. You enhance the display for `order-detail` by customizing the display view.

The custom setup displays five fields:

- ▶ `retailer_name`
- ▶ `order_number`
- ▶ `quantity`
- ▶ `order_method_en`
- ▶ `ship_date`

Both formatting and displaying additional fields is done as part of the entity configuration for `order-detail`:

1. Go to the **Entities** section of the management console and select the existing **order-detail** entity.
2. Scroll to find **Display options**, and then enter the text shown in Example 9-1 into the **Custom entity deploy** text box, as shown in Figure 9-14.
3. As always, remember to scroll to the bottom of the page and click **Save Entity** to ensure that your changes are permanent.

Example 9-1 View configuration for the order-detail display

```
<%= entity_label(entity, :class => "item") do %>
  <div class="metadata">
    <div><%= entity["retailer_name"].first %></div>
    <div>Order #: <%= entity["order_number"].first %></div>
    <div>Quantity: <%= entity["quantity"].first %></div>
    <div>Method: <%= entity["order_method_en"].first %></div>
    <div>Shipped: <%= find_and_format_time(entity, 'ship_date') %></div>
    <br/>
  </div>
<% end %>
```

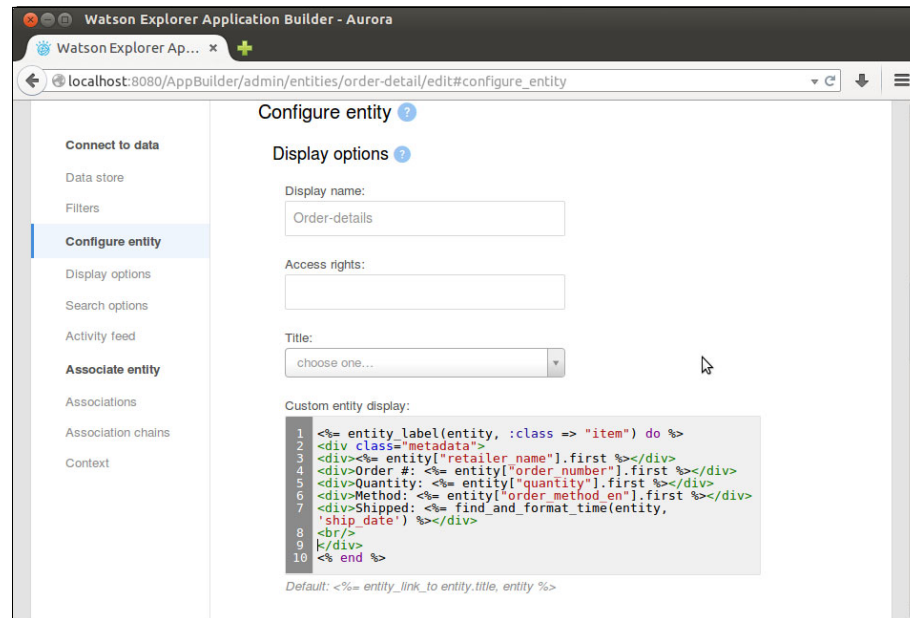


Figure 9-14 Customize order-detail display

After making these changes, each item in the order list provides meaningful information. However, a few shortcomings remain. For example, the list is only an arbitrary collection of ten orders. It would be better to display the most recent orders.

Next, adjust the `orders_list` widget to do exactly that.

9.2.5 Sorting orders by ship date

Sorting is done in Watson Explorer Engine. To use a field for sorting or categorization, make sure that each field that you use for sorting or categorizing is fast-indexed in the search collection in Watson Explorer Engine. The procedure for setting fast-indexed fields is described in 8.2.4, “Final search collection configuration” on page 121.

To begin to setup sorting, follow these steps:

1. For sorting and some later sections of this exercise, you must set several fields as fast-indexed on the `appbuilder-order-details` search collection, as shown in Example 9-2.

Example 9-2 Fast index fields in appbuilder-order-details

```
last-modified|date
product_number|set
ship_date|date
purchases_quantity|int
quantity|int
retailer_name|set
```

After making those changes to the search collection in the Watson Explorer Engine administration tool, you enable the application to use the sorting capability.

2. Go to the **Pages & Widgets** section of the Application Builder management console and find the `orders_list` widget under **product** by clicking the arrow to the right of the **Detail page** label.
3. Click **orders_list** and replace the text in **Entities to query** with the text shown in Example 9-3.

Example 9-3 Orders list Entities to query

```
subject.orders.sorted_by(field('ship_date').in_descending_order)
```

After making that change to display the most recent orders, also change the widget's **Display Name** to Recent Orders.

4. The edit widget page with both of these changes is shown in Figure 9-15. After making the changes, click **Save Changes** at the bottom of the page to make your changes permanent.

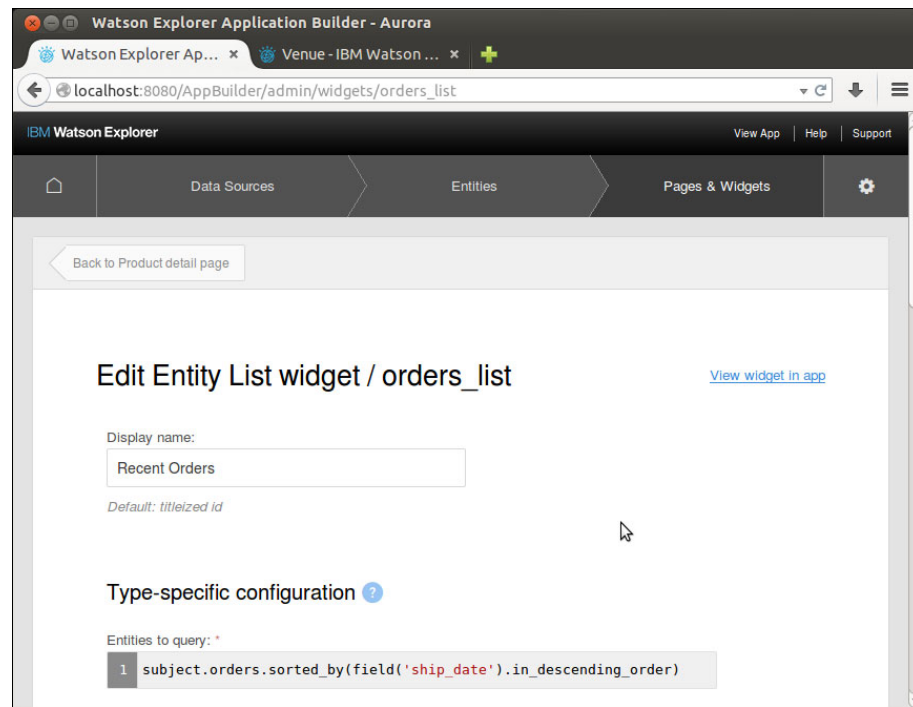


Figure 9-15 Enhancing the orders list widget to display recent orders

5. As a status check, look at what you have now. Click **View Widget in App** to see the new Recent Orders widget, as shown in Figure 9-16.

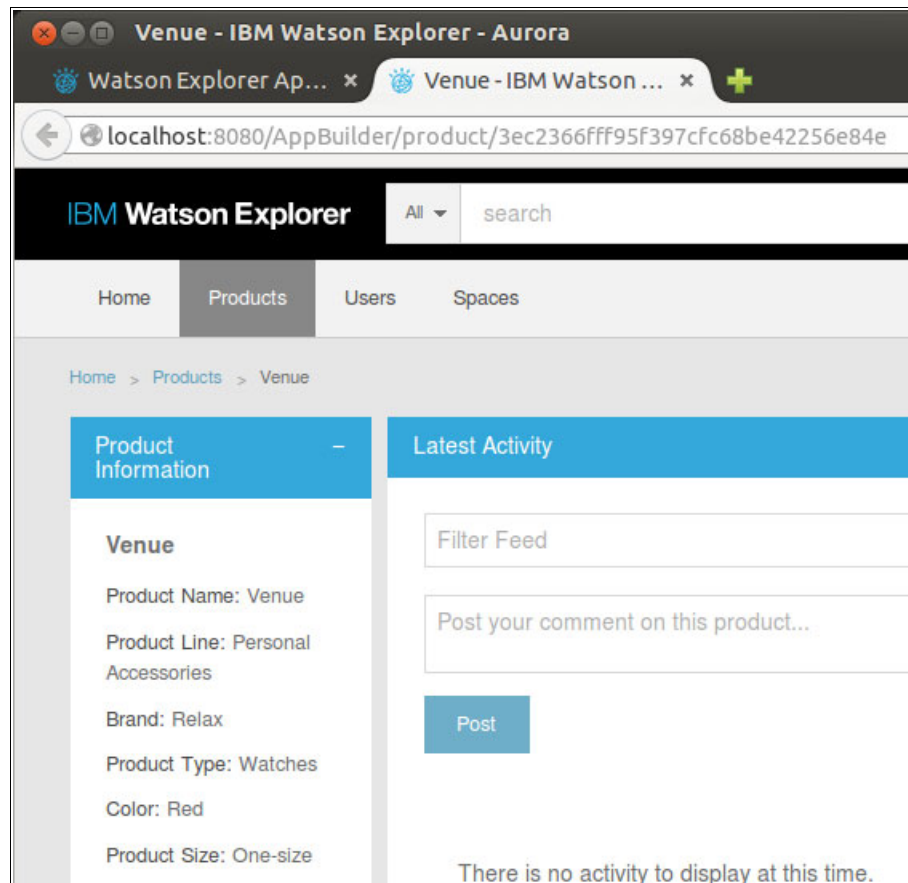


Figure 9-16 Recent Orders widget as seen in the application

9.2.6 Creating a chart widget for top purchases

Application Builder supports many building blocks for views. Next, add a widget to create a chart of the largest purchases for a product:

1. Go to **Pages & Widgets** and click the **Detail Page** for product.
2. On the Product Detail Page, click **Create Widget** and select **Entity Chart** under **New widget**, as shown in Figure 9-17.

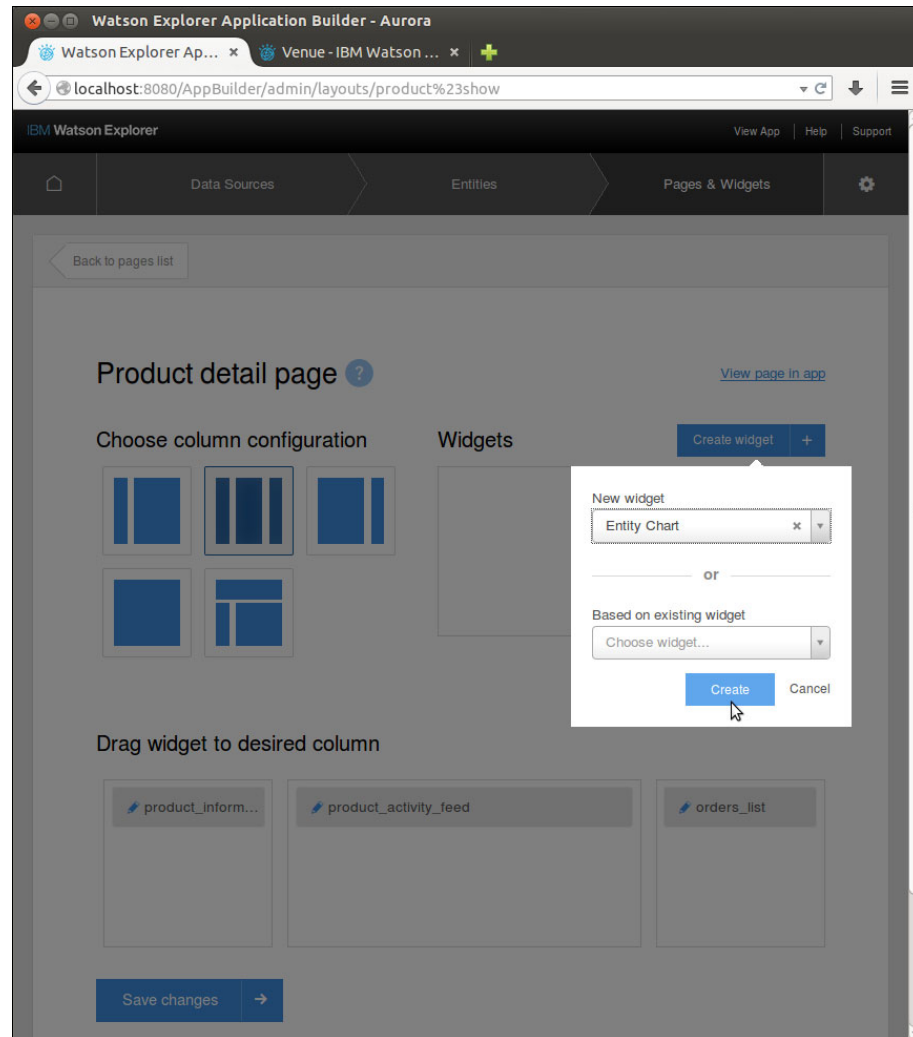


Figure 9-17 Creating an entity chart widget on the product detail page

3. Give the new widget the **Id** `top_purchases`.
4. Specify the details of the chart in the **Chart Configuration** section.
5. Next, make a bar chart, fairly tall, with the largest values at the top:
 - a. Label the data with the string `Quantity`.
 - b. Limit the chart to showing the top five values
 - c. Identify the data source.

Chart configuration is expressed in JSON, which provides extremely precise chart definition, as shown in Example 9-4.

Example 9-4 JSON for Entity Chart Configuration

```
{
  "chart": {
    "type": "bar",
    "height": 370,
    "sort_by_value": true
  },
  "series": [
    {
      "name": "Quantity",
      "facet_id": "purchases_quantity",
      "faceting":
"@subject.orders.faceted_by(xpath('$retailer_name').with_facet_id(facet_id)
.without_pruning.then(sum(xpath('$quantity')))).faceting",
      "point_limit": 5
    }
  ]
}
```

6. Enter the JSON in Example 9-4 on page 151 in the **Chart Configuration** area on the **New Entity Chart widget** page, as shown in Figure 9-18.

Watson Explorer Application Builder - Aurora

Watson Explorer Ap... x Venue - IBM Watson ... x

localhost:8080/AppBuilder/admin/widgets/new?entity_type=product&page_type=show&based_on=en

New Entity Chart widget

Id: *

top_purchases

A widget ID can contain alphanumeric characters, underscores, and hyphens, and must contain at least one letter or number.

Note: You cannot change this value later.

Display name:

Default: titleized id

Type-specific configuration ?

Chart configuration: *

```
1 {
2   "chart": {
3     "type": "bar",
4     "height": 370,
5     "sort_by_value": true
6   },
7   "series": [
8     {
9       "name": "Quantity",
10      "facet_id": "purchases_quantity",
11      "faceting":
12        "@subject.orders.faceted_by(xpath('$retailer.name').with_facet_id(facet_id).without_pruning.then(sum(xpath('$quantity')))).faceting",
13      "point_limit": 5
14    }
15  ]
16 }
```

Figure 9-18 Configure new entity chart widget for top purchases

7. After saving the new widget, click **Back to Product Detail Page** and drag the new widget into the center column. This drag operation is much like what you did when positioning the orders_list widget, as shown in Figure 9-13 on page 144.
8. Click **Save Changes** and then click **View Page in App**. A chart of the top purchases now displays on the product detail page in Figure 9-19.

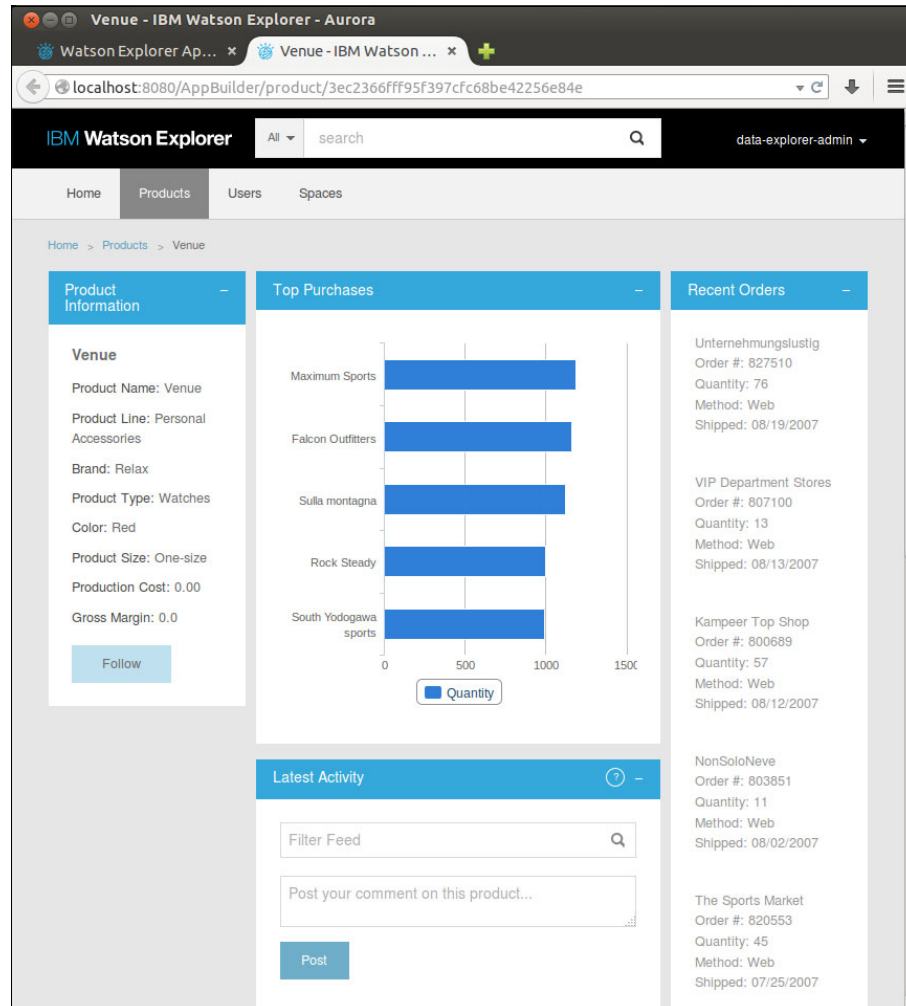


Figure 9-19 Product page including top purchases

9.3 Searching entities

The entity page that you created gives a detailed view of a single product. The product list page provides a simple mechanism to find products by scrolling through the entire product list. Next, make it easier to find products by making them searchable.

9.3.1 Making an entity searchable

To make an entity searchable, follow these steps:

1. In the Application Builder management console, go to the **Entities** section and click **product**.
2. On the product page, click **Search Options** to get to the Searchable option and turn **Searchable** to **on**, as shown in Figure 9-20. Remember to scroll to the bottom of the page and click **Save Entity** to save your changes before proceeding.

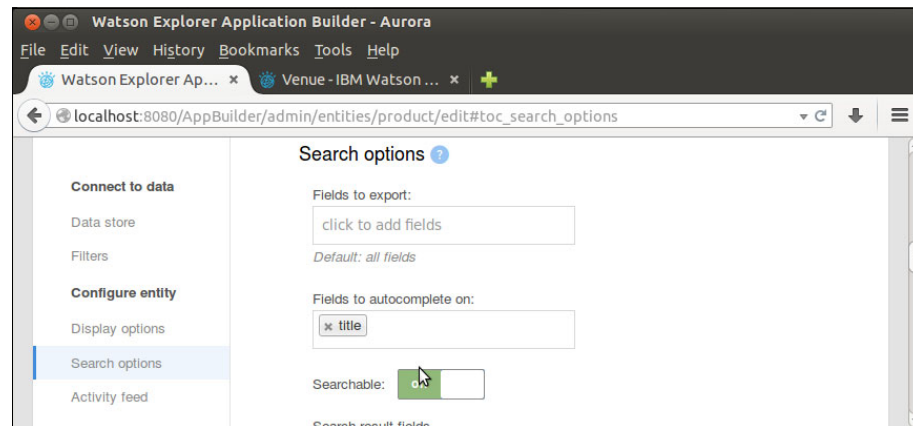


Figure 9-20 Making an entity searchable

Now you can search for products in the application. So far, you only see product names in the search results, as shown in Figure 9-21 on page 155, which shows the results of a simple search for the word `golf`.

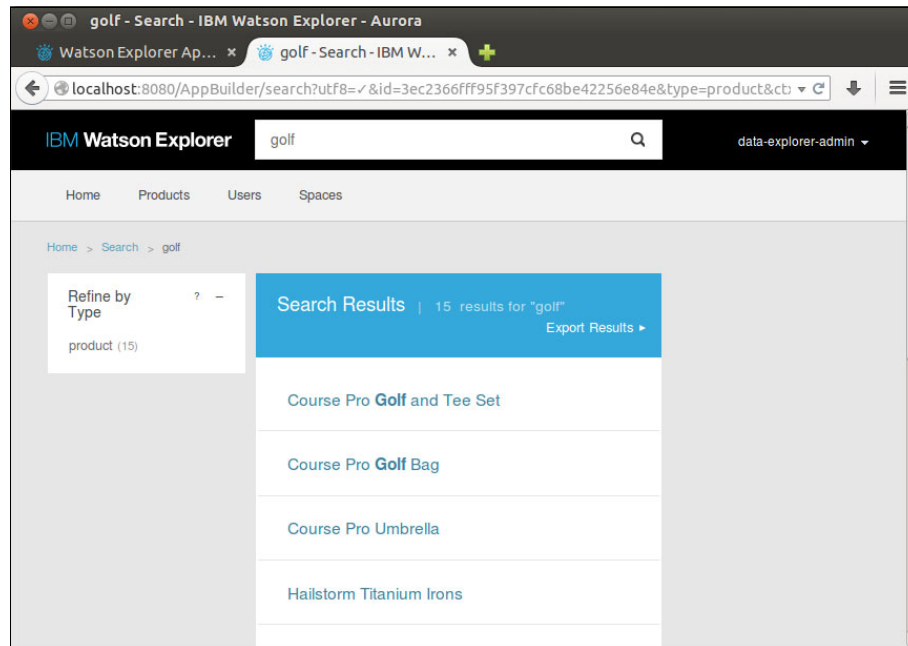


Figure 9-21 Simple search results for the term “golf”

9.3.2 Enhancing your search display

You can enhance how search results are displayed by choosing additional fields to be shown in your search results:

1. Return to the Application Builder management console, go to the Entities section, and click **product**.
2. Scroll down to the table under **Search result fields** that contains **Field Name** and several other columns.
3. Click the **+** at the right side to add each field.
4. Add several fields that are likely to contain interested searchable text:
 - a. brand
 - b. product_description
 - c. product_line
 - d. product_number
 - e. product_size
 - f. product_type

The selected set of these is shown in Figure 9-22.

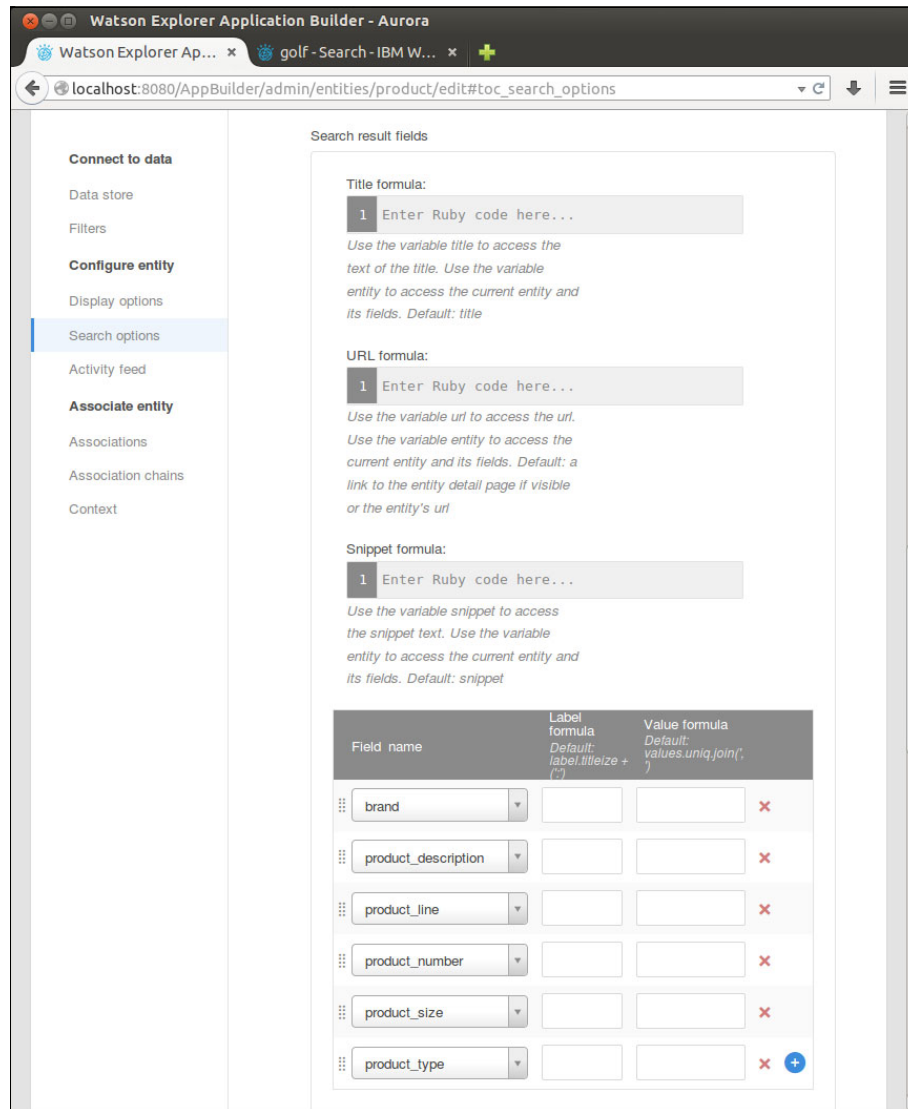


Figure 9-22 Add fields for Search Display

5. After adding fields to the search display, the search results are much more informative, as shown in Figure 9-23.

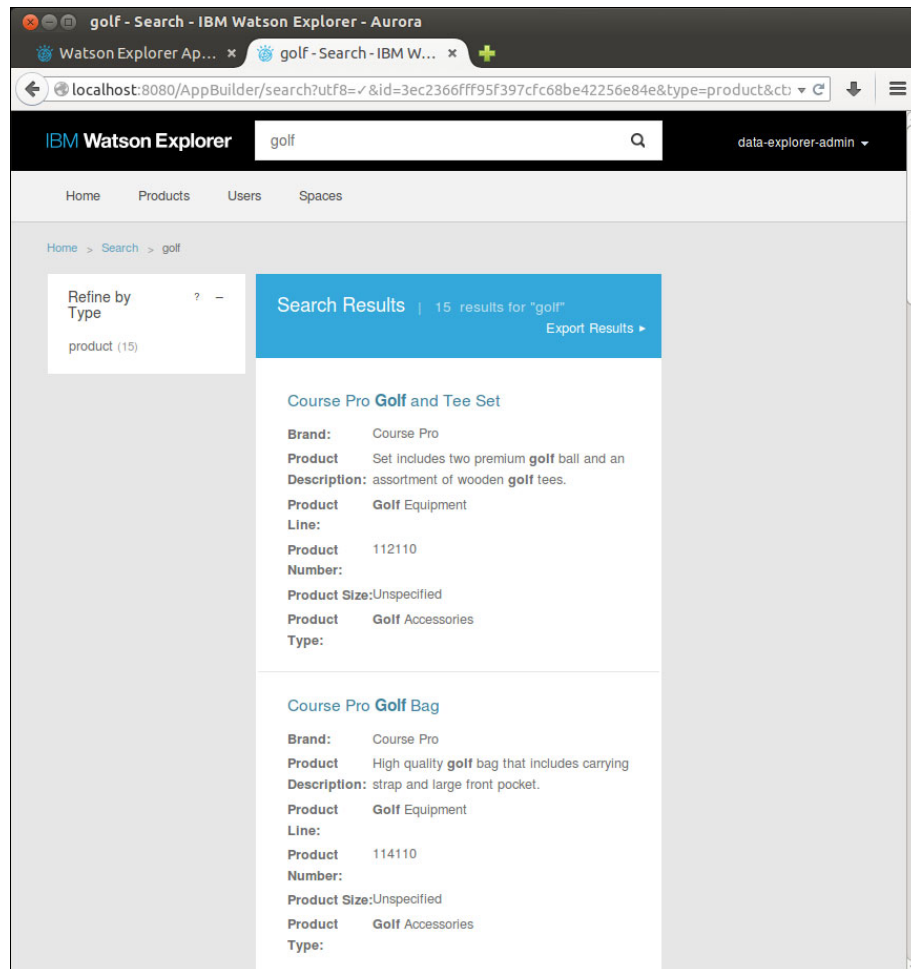


Figure 9-23 Search results with additional fields

9.3.3 Adding search refinements

Search refinements give users more information about their search results, and also provide controls for refining the results that are being shown. One of the most common type of search refinements to add is a refinement based on a specific data field in your results. In this case, you add a refinement on the `product_line` field:

1. First, you make sure that `product_line` is fast-indexed in Watson Explorer Engine. The procedure for setting fast-index fields is detailed in 8.2.4, “Final search collection configuration” on page 121.

The fast-index settings for the `appbuilder-products` search collection must be as shown in Example 9-5.

Example 9-5 Fast index fields in appbuilder-products

```
last-modified|date
product_number|set
product_line|set
introduction_date|date
```

Now you create and configure a widget to enable users to refine search results using the `product_line` field.

2. In the Application Builder Management Console, go to **Pages & Widgets** and click **Detail Page** for **searches**.

3. Click **Create Widget** and choose **Refinement**, as shown in Figure 9-24.

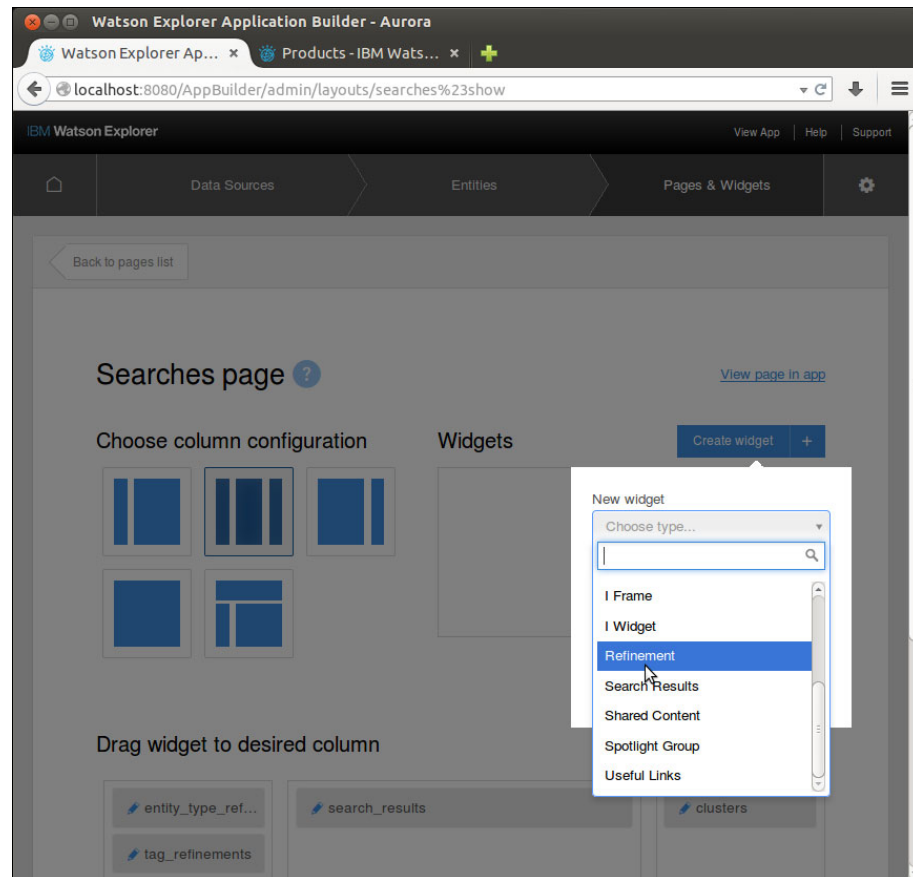
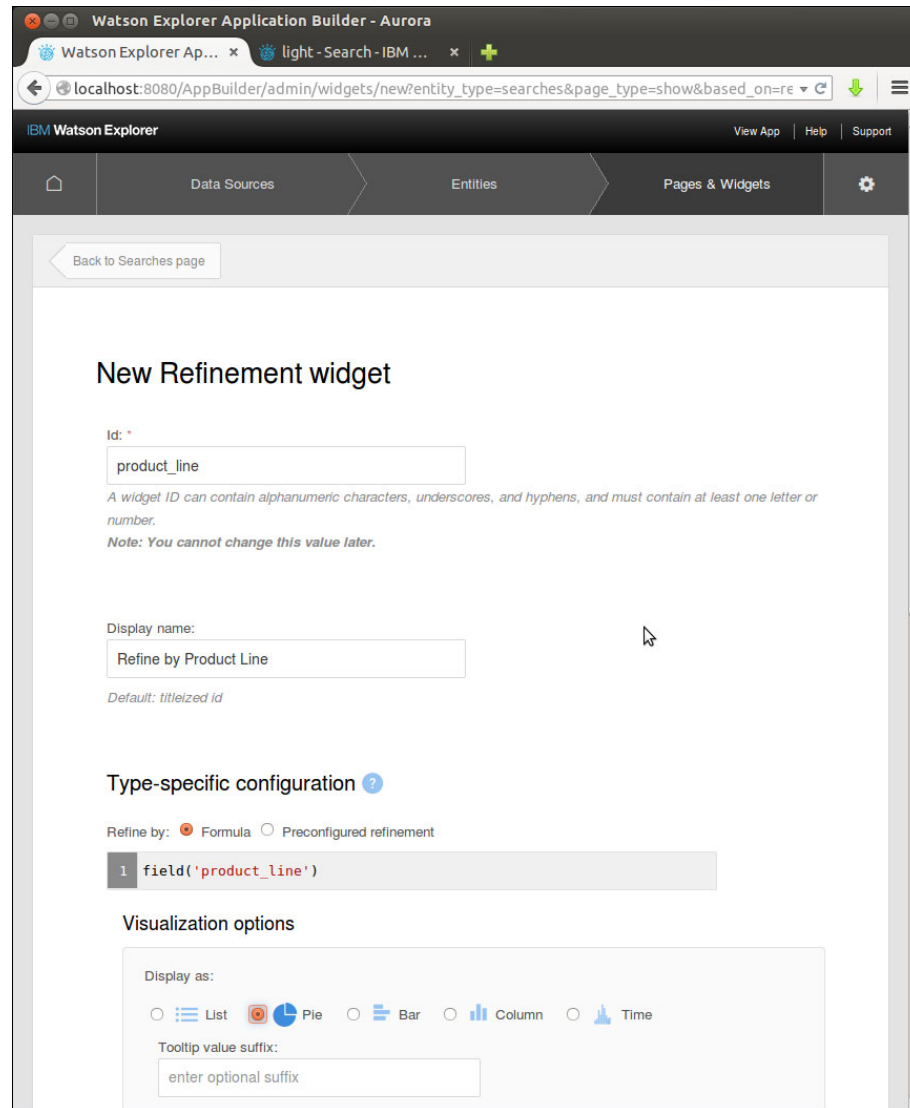


Figure 9-24 Create Widget → refinement

4. On the New Refinement Widget page, complete the following fields, as shown in Figure 9-25:
 - a. Enter `product_line` as the **Id**.
 - b. Enter Refine by Product Line as the **Display Name**.
 - c. Enter `field('product_line')` as the **Refine by: Formula**.
 - d. Select **Pie** under **Display as**.



Watson Explorer Application Builder - Aurora

Watson Explorer Ap... x light - Search - IBM ... x +

localhost:8080/AppBuilder/admin/widgets/new?entity_type=searches&page_type=show&based_on=re

IBM Watson Explorer View App | Help | Support

Data Sources Entities Pages & Widgets

Back to Searches page

New Refinement widget

Id: *

product_line

A widget ID can contain alphanumeric characters, underscores, and hyphens, and must contain at least one letter or number.

Note: You cannot change this value later.

Display name:

Refine by Product Line

Default: titleized id

Type-specific configuration ?

Refine by: ☒ Formula ☐ Preconfigured refinement

1 field('product_line')

Visualization options

Display as:

☐ List ☒ Pie ☐ Bar ☐ Column ☐ Time

Tooltip value suffix:

enter optional suffix

Figure 9-25 New Refinement Widget for product line

5. After saving the product_line refinement widget, you must add the widget to the search page. Click the **Back to Searches Page** button near the top of the page. In the layouts for **Searches page** and drag the product_line widget to the top of the right column. This is similar to the widget placement shown in Figure 9-13 on page 144.

The search results page now includes a Refine by Product Line widget, as shown in Figure 9-26.

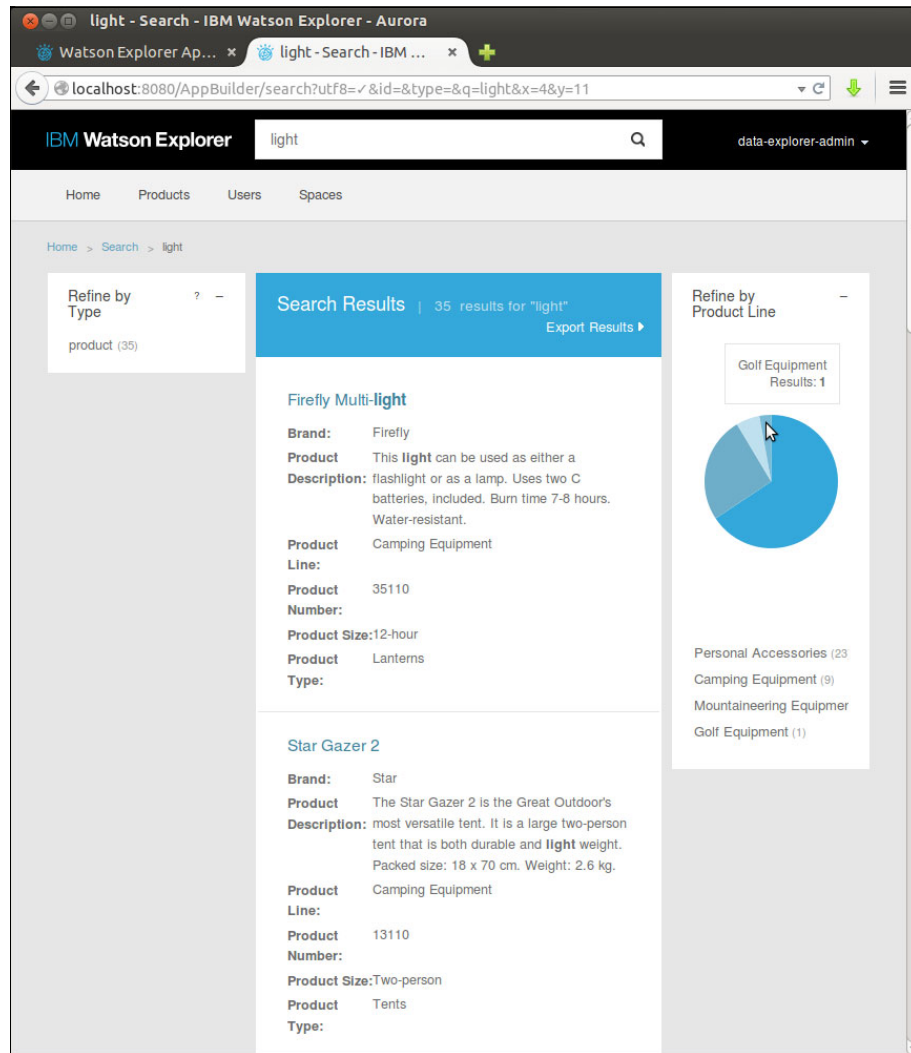


Figure 9-26 Search results with the Refine by Product Line widget

6. Clicking a refinement narrows the search results, as shown in Figure 9-27.

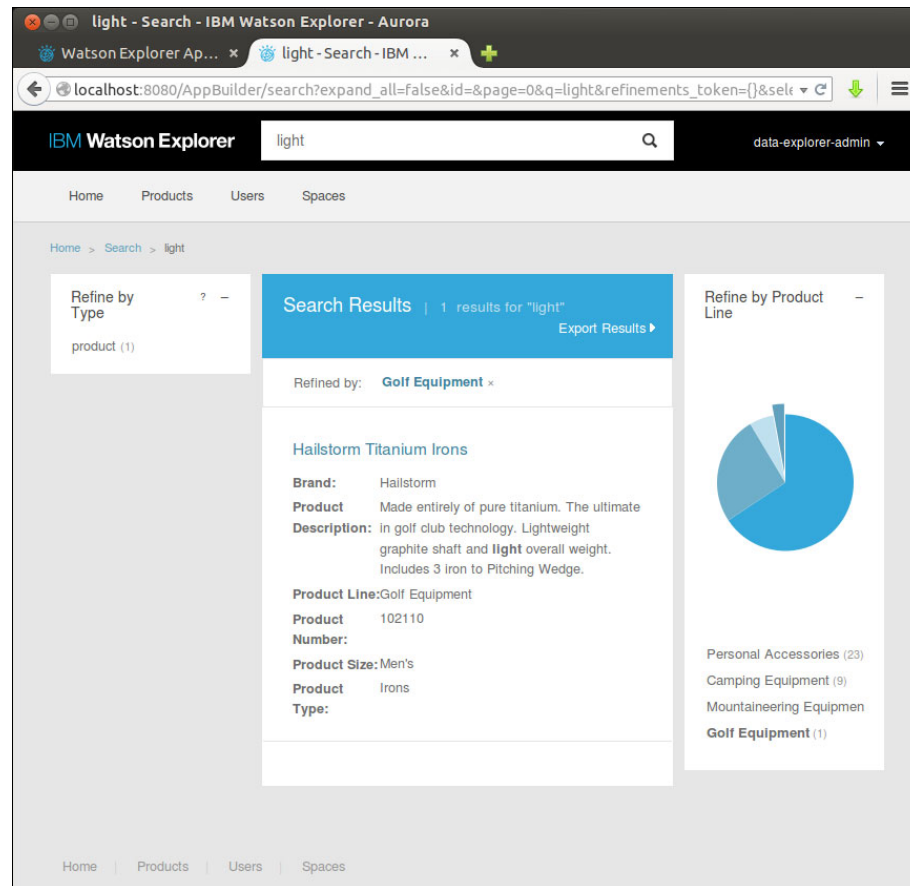


Figure 9-27 Refined search results

Adding another search refinement lets you see how refinements interact with each other.

1. Return to the **Detail Page** for **searches** and create another **Refinement** widget, as shown in Figure 9-24 on page 159.
2. On the New Refinement Widget page, complete the following fields, as shown in Figure 9-28 on page 163:
 - a. Enter `introduction_date` as the **Id**.
 - b. Enter Refine by Introduction Year as the **Display Name**.
 - c. Enter `field('introduction_date').with_intervals_of(1.year.to_i)` as the **Refine by: Formula**.

- d. Select **Time** under **Display as**.
- e. Enter {value:%Y} as the **X axis label**.

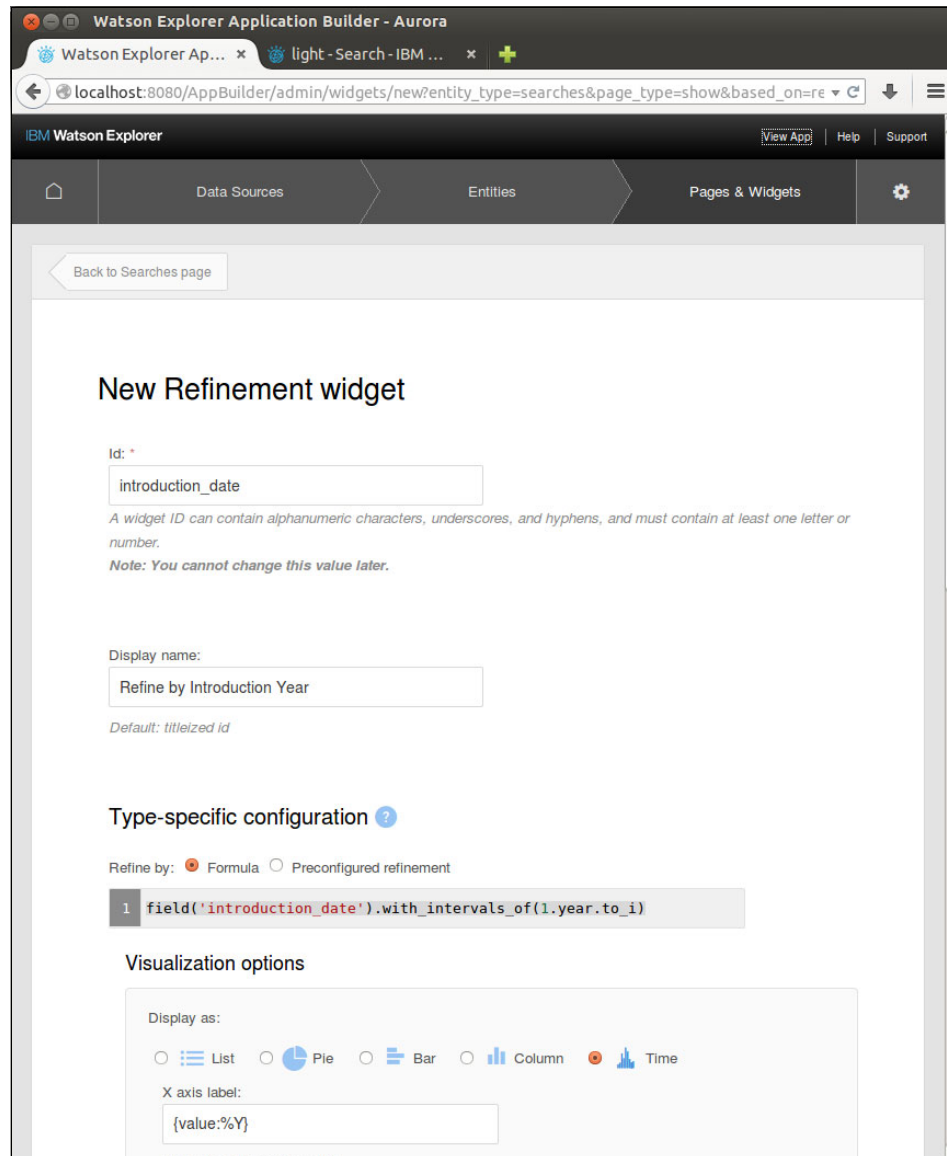


Figure 9-28 New Refinement Widget for introduction date

3. After saving the introduction_date refinement widget, you must, as before, add the widget to the search page.

The search results page now includes a **Refine by Introduction Year** widget, as shown in Figure 9-29.

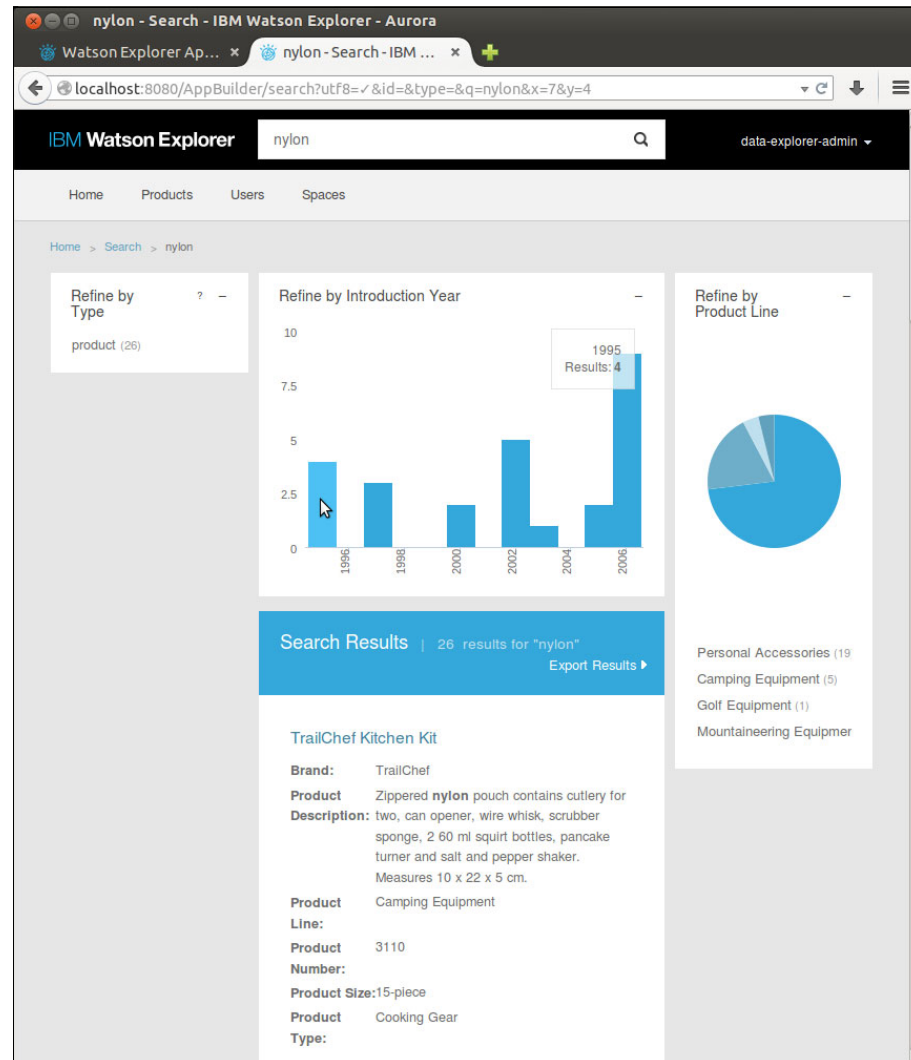


Figure 9-29 Search results with Refine by Introduction Year widget

Clicking the refinement for 1995 both narrows the search results and adjusts the selectable pie chart in the **Refine by Product Line** widget to match the narrower result set, as shown in Figure 9-30.

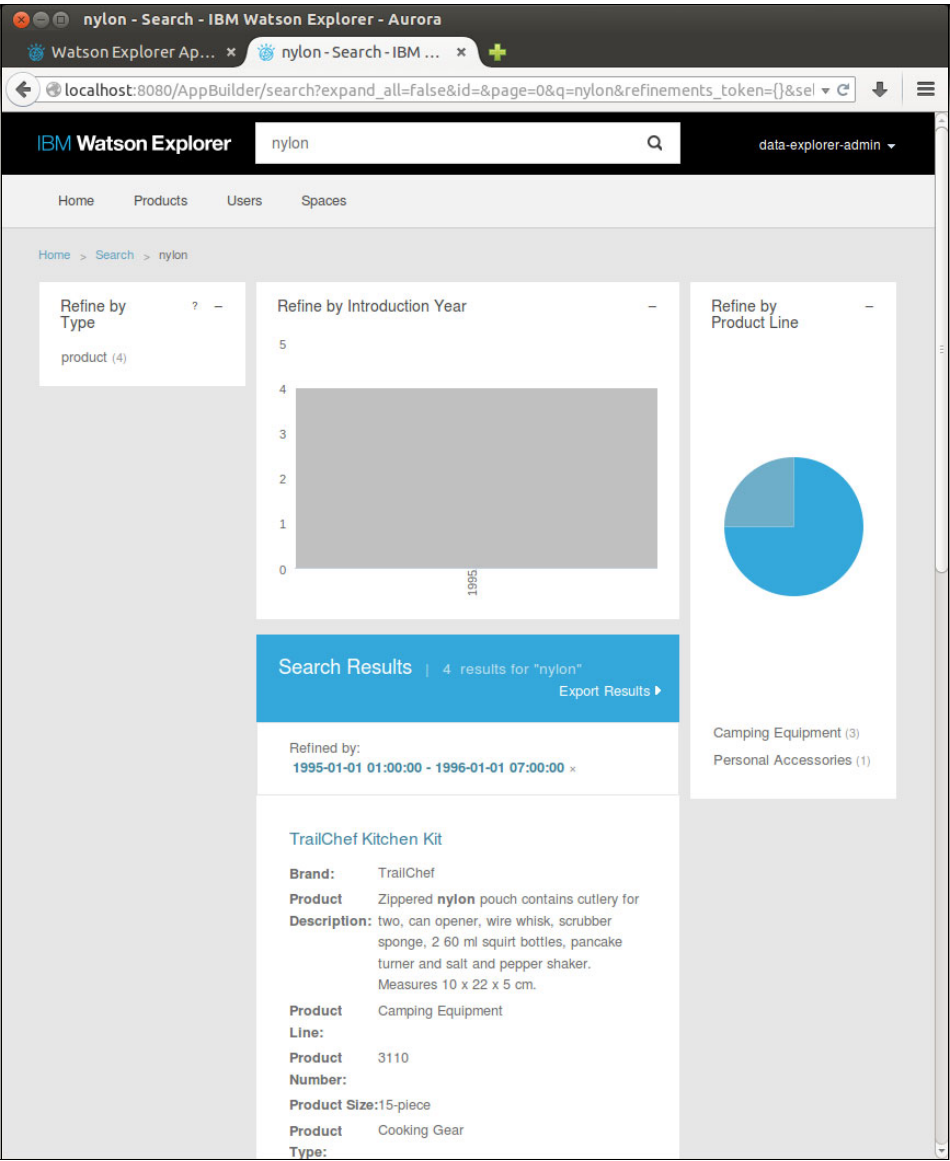


Figure 9-30 Search results refined by Introduction Year

9.4 Conclusion

This chapter showed you how to build a 360-degree information application, describing the basics of defining entities, sources, associations, widgets, and pages.

The next steps in most 360-degree applications would be to continue to add more context around the key entity. A common enhancement would be Twitter widgets that look up mentions of your products, your product categories, or competing products.

Similarly, you could add more key entities, such as retailers, sales channels, or marketing campaigns, displaying core information about each instance of those entities while also relating those entities to other entities in the 360-degree application.



IBM InfoSphere MDM Probabilistic Matching Engine for InfoSphere BigInsights

This chapter provides information about IBM InfoSphere MDM Probabilistic Matching Engine for InfoSphere BigInsights. It describes how the product is typically used. We provide instructions about how to install the product and provide an example of its usage with some sample data and configuration.

This chapter also provides an opportunity to develop a custom extension for customizing the 360-degree view of entity data. These topics are discussed:

- ▶ Introduction
- ▶ Use cases
- ▶ PME for BigInsights architecture
- ▶ Installation
- ▶ Lab scenario
- ▶ Configuration
- ▶ Derivation, comparison, and linking
- ▶ Probabilistic search
- ▶ 360-degree view

10.1 Introduction

IBM InfoSphere MDM Probabilistic Matching Engine for InfoSphere BigInsights, also referred to as PME for BigInsights or big data matching, is a software solution built on top of the InfoSphere BigInsights platform. PME for BigInsights is used to process large volumes of information using the parallel computing power of Apache Hadoop and Apache HBase.

10.2 Use cases

The typical use cases for PME for BigInsights fall into two classes. Big Data users typically want to search large volumes of data quickly using a probabilistic algorithm or they want to obtain a 360-degree view of their data.

10.2.1 Probabilistic search

Probabilistic searches of large volumes of data is a common use case of most PME for BigInsights customers. They typically have data sets in the hundreds of millions of records on up through billions or tens of billions of records. Quickly searching data sets of that size requires efficient indexing and algorithm configuration and sufficient cluster hardware.

Performing probabilistic searches using PME for BigInsights can be accomplished using the Dashboard, a light weight sample web application, or the PME Search application, a MapReduce application that performs searches using criteria stored in an HBase table as input.

10.2.2 360-degree view

PME for BigInsights provides a 360-degree view of mastered data by combining data from one or more records that make up an entity into a composite view. The default implementation provides what is called the Entity Most Current Attribute (EMCA) view by selecting the most current value for an attribute based on the time stamp of the column in the database. This 360-degree view is similar to Watson Explorer's 360-degree view in that it combines data from one or more sources to get the bigger picture of a single entity, rather than Watson Explorer's larger view of multiple entities.

Similar to the search application, composite views can be seen by accessing the data programmatically or by using the PME Extract MapReduce application to export the mastered entity data to a new HBase table.

10.3 PME for BigInsights architecture

PME for BigInsights provides a complete solution for mastering big data on the IBM InfoSphere BigInsights platform. It is built on the same probabilistic matching engine used in the Standard and Advanced editions of IBM InfoSphere Master Data Management. Though PME for BigInsights is based on the same underlying technology, it has no dependency on the operational server deployments of those editions. However, there is one part of the InfoSphere MDM tool set used by PME for BigInsights. The configuration of the matching algorithm is accomplished with the InfoSphere MDM Workbench.

Figure 10-1 shows the functional components of PME for BigInsights and how they relate to the components in InfoSphere BigInsights. The primary components on the left are normally located on the master node of the cluster. The secondary components on the right are normally located on the slave nodes of the cluster.

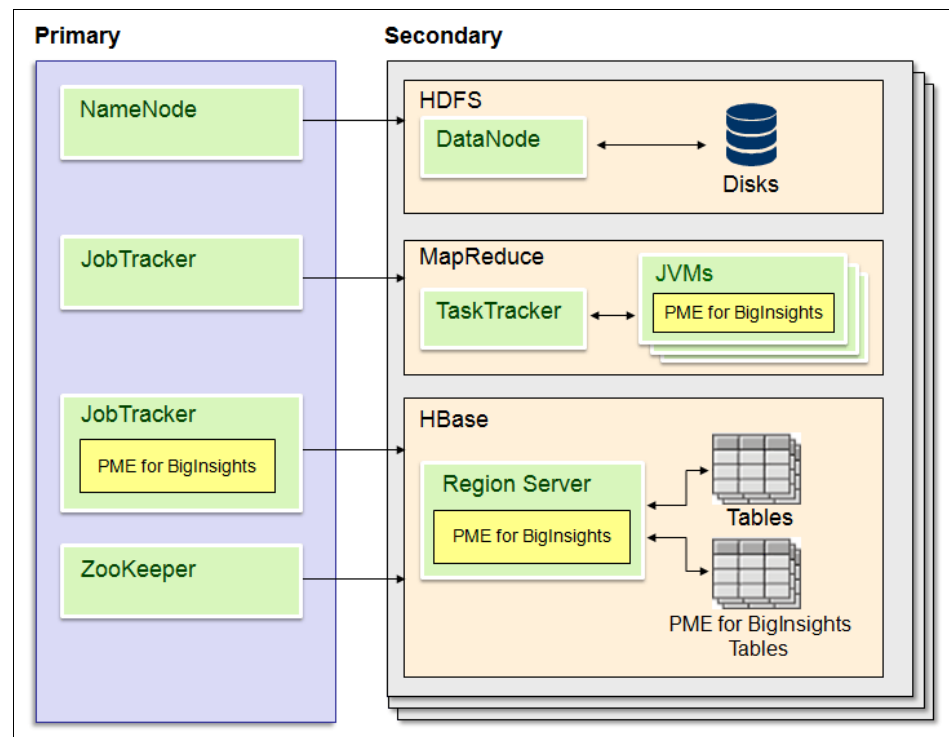


Figure 10-1 PME for BigInsights architecture diagram

PME for BigInsights exists as a series of HBase coprocessors and MapReduce applications. This architecture allows flexibility in how and when the data is processed. Users have the option of allowing data to be processed automatically when the data is added to their HBase table or manually in batches by using the MapReduce applications.

The HBase coprocessors facilitate automatic processing of the data when it is added and modified within an enabled HBase table. The coprocessors also serve as an entry point for the PME for BigInsights API.

The MapReduce applications were designed to process big data in batches. PME for BigInsights supplies several applications for different purposes. Each application handles a specific task necessary to master the data stored in HBase.

Mastering data with PME for BigInsights requires three distinct operations to be performed:

1. The input data has to be derived. The derivation process applies the matching algorithm's standardization and bucketing logic to the input data and stores the results in the PME for BigInsights tables for use in matching.
2. The derived data has to be compared. This comparison process applies the comparison functions from the matching algorithm with the function weights to determine how each record in a comparison bucket scores against the other records in the bucket.
3. The compared records are linked into entities. This linking process groups records together that score above the configured autolinking threshold of the matching algorithm.

These three operations occur automatically on an enabled HBase table unless explicitly disabled by configuration.

10.4 Installation

PME for BigInsights uses IBM Installation Manager to install its features onto the master node of the InfoSphere BigInsights cluster. Several prerequisites must be met before installing big data matching.

10.4.1 Prerequisites

Because PME for BigInsights runs on top of InfoSphere BigInsights, the system prerequisites are the same as they are for BigInsights. At the time that this book was written, PME for BigInsights required InfoSphere BigInsights version 2.1.2.

Detailed information about system requirements and installation instructions can be found in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSPT3X_2.1.2/com.ibm.swg.im.infosphere.biginsights.welcome.doc/doc/welcome.html

The lab scenario in 10.5, “Lab scenario” on page 171 can be run on a single node pseudo-cluster or larger clustered installation. For simplicity, the InfoSphere BigInsights Quick Start Edition can be used to perform the lab scenario. You can obtain a no-charge copy at the following website:

<http://www-01.ibm.com/software/data/infosphere/biginsights/quick-start/>

The Quick Start web site provides both a native installation and virtualization images. The virtualization image is the quickest way to get up and running and was used during the authoring of this book. It contains both IBM InfoSphere BigInsights and an Eclipse development environment that are useful for the lab scenario.

10.4.2 Installing PME for BigInsights

For complete information about installing PME for BigInsights, see the installation topic *Probabilistic Matching Engine for InfoSphere BigInsights*, which is available in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.swg.im.mdmhs.pmebi_install.doc/topics/installing_pme_bi.html

To work through the lab scenario that is described in this chapter, you must perform the following tasks, in order:

1. Run the IBM Installation Manager and installation scripts.
2. Install the Dashboard. We configure the Dashboard user interface as a part of the lab scenario in 10.6.4, “Dashboard” on page 178. Make sure to start the HBase REST server and the WASLP server instance hosting the Dashboard.

For more detailed information about installing PME for BigInsights and the Dashboard, see the public knowledge center link above.

10.5 Lab scenario

The lab scenario covers most aspects of PME for BigInsights that users would typically encounter in applying the software solution to solve a large master data management problem.

10.5.1 Business challenge

A typical master data management (MDM) challenge is identifying and merging duplicates in a single data source or reconciling multiple data sources to identify records that represent the same entity in a given domain. This can occur in multiple domains: people, patients, customers, organizations, and so on. In fact, it is common enough that InfoSphere MDM includes a template called Party that is intended to represent many of these entities.

10.5.2 Data sources

The data used in this lab scenario is typical data that might be collected for a person. It is available as a part of the product installation and can be found in the installation path specified in IBM Installation Manager during installation. The file that contains the data and other configuration files is the `com.ibm.mdm.bi.pme.sample.zip` archive.

It should be noted that the sample data used here is randomly generated from a pool of input values and does not represent any real person. Any such representation is entirely coincidental and unintentional.

10.5.3 MDM algorithm

To keep the lab scenario simple, we use the Party algorithm template that is shipped with InfoSphere MDM Workbench. The algorithm is for use with PME for BigInsights. This algorithm is a part of sample package mentioned in 10.5.2, “Data sources” on page 172.

In a real deployment, users would use the InfoSphere MDM Workbench to create an algorithm specific to their needs. For more information about developing a matching algorithm using InfoSphere MDM Workbench, see the *Developing the algorithm* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.svg.im.mdmhs.pmebi.doc/topics/develop_algorithm.html

10.6 Configuration

You can start the initial configuration for the lab scenario after all the installation tasks in 10.4, “Installation” on page 170 are completed. The next few sections go over the configuration process in detail.

10.6.1 Site

Site configuration typically only needs to be accomplished once and is captured in the `pme-site.xml` file. This configuration file conforms to the standard Hadoop configuration file format and contains configuration properties that apply to all the instances of PME for BigInsights running in the InfoSphere BigInsights cluster. There are several properties that can be configured, some of the more common properties are shown in Table 10-1.

Table 10-1 Configuration properties in `pme-site.xml`

Property name	Default value
<code>pme.queue.log.dir</code>	<code>/hadoop/pme</code>
<code>pme.link.per.worker.memory.mb</code>	2000
<code>pme.link.check.interval.seconds</code>	60
<code>pme.link.quiet.period.seconds</code>	300

The first property, `pme.queue.log.dir`, identifies the directory to where the PME for BigInsights queue services will log their output. This directory must exist on all the slaves and must be writable to the biadmin group. You can create the directory using the command shown in Example 10-1.

Example 10-1 Queue Directory creation command

```
$BIGINSIGHTS_HOME/IHC/sbin/slaves.sh mkdir -m 775 /hadoop/pme
```

Note: If the `pme.queue.log` directory does not exist or is not writable when a table is enabled for big data matching, an error message will appear in the HBase region server logs indicating the invalid configuration. PME for BigInsights features will be disabled until the directory is created and HBase is restarted.

The second property, `pme.link.per.worker.memory.mb`, is used to tune the memory available to the workers during autolinking. For the lab scenario, this default value is acceptable.

The last two properties `pme.link.check.interval.seconds` and `pme.link.quiet.period.seconds` respectively control how frequently the automatic linking process checks to see if linking is necessary and how long it should wait after the last change before initiating a linking job.

Unless you have a need to change the defaults, we advise to use the default setting and not to create a **pme-site.xml** configuration file. For additional information about the pme-site.xml configuration file, see the *Importing the big data matching configuration* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.swg.im.mdmhs.pmebi.doc/topics/importing_pme.html

10.6.2 Algorithm

We discussed the MDM algorithm in 10.5.3, “MDM algorithm” on page 172. For the lab scenario, the algorithm file is provided as a part of the sample distributed with the product. If you have not already extracted the files from com.ibm.mdm.bi.pme.sample.zip, extract them now using the command shown in Example 10-2.

Example 10-2 Extracting the sample files

```
unzip com.ibm.mdm.bi.pme.sample.zip -d sample
```

Note: If you installed PME for BigInsights using the root user, you might have to specify a different directory to extract the files to.

The algorithm configuration is stored in the mdmpme-config-person.zip archive and must be distributed to all the nodes in the cluster. You accomplish this by copying the archive to the Hadoop configuration staging directory using the command shown in Example 10-3.

Example 10-3 Deploying the algorithm

```
cp mdmpme-config-person.zip $BIGINSIGHTS_HOME/hdm/hadoop-conf-staging
```

Synchronizing the configuration to the cluster from the staging directory will occur in Example 10-4 on page 175.

10.6.3 Table

The table configuration used is also packaged within the sample. The file pme-person.xml contains the table configuration settings that we use for the first part of the lab scenario.

Copy pme-person.xml to the Hadoop configuration staging directory and synchronize the configuration with the following two commands shown in Example 10-4 on page 175.

Example 10-4 Deploying the table configuration

```
cp pme-person.xml $BIGINSIGHTS_HOME/hdm/hadoop-conf-staging
syncconf.sh hadoop
```

The values specified in the configuration file identify the algorithm configuration used and map the HBase columns to algorithm attributes and fields. Some of the properties are shown in Table 10-2.

Table 10-2 Sample pme-person.xml properties

Property Name	Value
pme.person.configzip	mdmpme-config-person.zip
pme.person.recordtype	PERSON
pme.person.algorithms	mdmper
pme.person.srcdefault	MDMSP
pme.person.pf:fname#PERLEGALNAME:givenname1	java.lang.String

As you can see, all of the properties begin with pme and contain the name of the table being configured: person. The remaining part of the property identifies what is being configured. For instance we set the default source for our data to MDMSP because that is the only source that exists in our default configuration. The last property maps the HBase column pf:fname to the matching algorithm's PERLEGALNAME attribute's givenname1 field.

For additional information about the table configuration file, see the *Creating configuration files to map the big data matching parameters* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.svg.im.mdmhs.pmebi.doc/topics/creating_configuration_files.html

Now that you deployed the table configuration and synchronized the configuration across the cluster, you must to restart all the services to make sure the configuration archive and files are loaded. You can restart the InfoSphere BigInsights cluster with the two commands shown in Example 10-5.

Example 10-5 Restarting the InfoSphere BigInsights cluster

```
$BIGINSIGHTS_HOME/bin/stop-all.sh
$BIGINSIGHTS_HOME/bin/start-all.sh
```

After the cluster is restarted, you are ready to create the table in HBase. Before creating the table, you must know how many regions to create. We suggest that you create as many regions as you have map capacity in MapReduce. To find the capacity of your cluster, open the Hadoop Map/Reduce Administration console. You can access it by clicking the **Access secure cluster servers** link on the InfoSphere BigInsights console. When the list of servers appears, click the **jobtracker** link. The map capacity is listed in the Cluster Summary table.

Figure 10-2 shows the map task capacity for our lab environment running in the InfoSphere BigInsights Quick Start edition. The map task capacity of four corresponds to the number of cores available to the virtual machine that hosts the installation.

Cluster Summary (Heap Size is 15.63 MB/2.02 GB)										
Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Te
0	0	0	1	0	0	0	0	4	4	8.0

Figure 10-2 Map task capacity

To create the table, you must use the HBase shell. Start the HBase shell with the command shown in Example 10-6.

Example 10-6 Starting the HBase shell

```
$BIGINSIGHTS_HOME/hbase/bin/hbase shell
```

When the shell prompt appears, you are ready to create the table. Create the table with the command in Example 10-7.

Example 10-7 Table creation command

```
create 'person', {NAME => 'pf', COMPRESSION => 'SNAPPY'}, {NUMREGIONS  
=> 4, SPLITALGO => 'HexStringSplit', DURABILITY => 'ASYNC_WAL'}
```

The statement in Example 10-7 creates a table named “person” with a column family “pf” to store data using the Snappy compression algorithm. We suggest using Snappy compression on textual data for its ability to quickly compress and decompress data but sacrificing some amount of compression. The number of regions matches the map task capacity of four in our lab system.

The split algorithm used is `HexStringSplit` because the row keys in our sample data are globally unique identifiers (GUIDs). We suggest using GUIDs because they improve the distribution of data throughout the cluster. Finally, the durability setting ensures changes are written to HBase's write ahead log asynchronously to improve performance.

The `list` command shows you the tables in HBase. Example 10-8 shows the output after creating the table.

Example 10-8 List command output

```
TABLE
BIMonitoring
BIMonitoringSummary
BIMonitoringSummary180
BIMonitoringSummary900
LogMetadata
LogRecords
person
7 row(s) in 0.0800 seconds
```

Now that the table is created, you must enable it for PME for BigInsights. This is accomplished with a series of commands as shown in Example 10-9. The first command disables the table in HBase. Before enabling the table for big data matching, the table must be disabled in HBase. The second command modifies the table structure to add an additional column family for record metadata. It also triggers the creation of several index tables used by PME for BigInsights. The third command re-enables the table in HBase. The fourth command is used to request that HBase balance the regions across the available region servers.

Example 10-9 Enable table commands

```
disable 'person'
pme_enable 'person'
enable 'person'
balancer
```

After running these four commands, the output of the list command should resemble Example 10-10. In the output, you can see the index tables created to support PME for BigInsights. At this point, you are ready to start adding data to your table.

Example 10-10 list command output

```
TABLE
BIMonitoring
```

```
BIMonitoringSummary
BIMonitoringSummary180
BIMonitoringSummary900
LogMetadata
LogRecords
person
person.pmebktidx
person.pmeentidx
person.pmememidx
10 row(s) in 0.0440 seconds
```

For additional information about table configuration, see the *Configuring tables* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.svg.im.mdmhs.pmebi.doc/topics/configuring_tables.html

10.6.4 Dashboard

Similar to the other configuration files, the Dashboard configuration is also packaged in the sample.

Note: The `ui_config.xml` contains configuration for all tables used with PME for BigInsights. If necessary, back up any existing file or manually combine the information from the sample with the existing file.

Copy the file `ui_config.xml` from the sample to the big data matching directory with the command in Example 10-11.

Example 10-11 Deploying the Dashboard configuration

```
cp ui_config.xml $BIGINSIGHTS_HOME/pme
```

Deploying the Dashboard configuration from the sample completes the configuration tasks for the lab scenario. You should be able to access the application using the URL `http://<host>:9081/pme/` and replacing `<host>` with the host name housing your InfoSphere BigInsights console.

For additional information about the Dashboard configuration, see the *Installing and configuring the Dashboard* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.svg.im.mdmhs.pmebi_install.doc/topics/installing_ui.html

10.7 Derivation, comparison, and linking

With everything configured, you are now ready to begin the process of loading the data, deriving the metadata, performing comparisons, and linking similar records together into common entities. As we stated before, this is configured to happen automatically by default when data is added to the HBase table.

10.7.1 Uploading the data

The first step in getting the data is uploading it to the cluster's file system by using the InfoSphere BigInsights console. Click the **Files** tab and create the **/users/biadmin/person** path. After the path is created, upload the **person-10k.tsv** data file from the sample to that path. When you are done, it should resemble Figure 10-3.

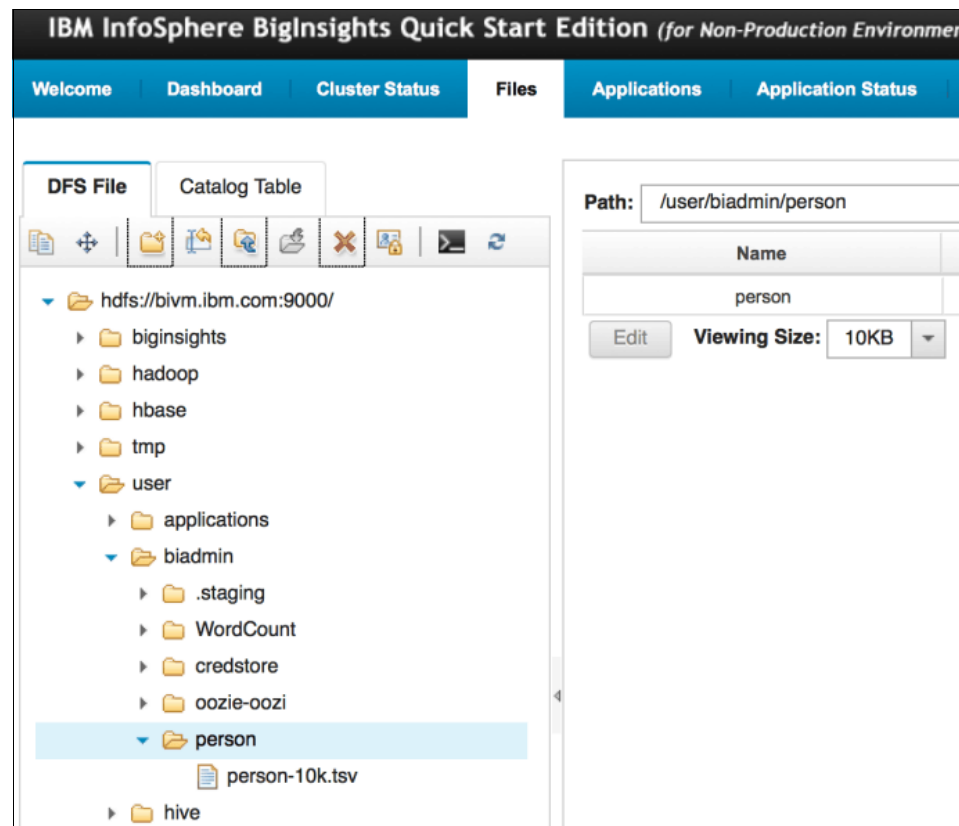


Figure 10-3 InfoSphere BigInsights Console after file upload

If you click the file name in the file hierarchy, you can see some of the contents of the file in the right side of the console. The file contains tab separated values as this is what **ImportTsv**, an HBase utility, works with by default.

10.7.2 Importing the data

With the data uploaded to the distributed file system, you can now import data into HBase. There are several mechanisms that can be used to load data into HBase. The easiest to use for our lab scenario is **ImportTsv**. This command line tool submits a MapReduce job to load the data from the distributed file system into HBase.

The command listed in Example 10-12 runs the HBase ImportTsv utility. Several command line arguments are used to control the MapReduce behavior and to define the column mapping from the source file, `person-10k.tsv`, to the columns in HBase that store the data. The last two arguments identify the table to import the data into and the distributed file system path where the file or files to be imported are located.

Example 10-12 ImportTsv command

```
$BIGINSIGHTS_HOME/hbase/bin/hbase
org.apache.hadoop.hbase.mapreduce.ImportTsv
-Dmapred.map.tasks.speculative.execution=false
-Dmapred.reduce.tasks.speculative.execution=false
-Dimporttsv.columns=pf:srccode,HBASE_ROW_KEY,pf:memidnum,pf:fname,pf:lname,pf:hstreet,pf:hcity,pf:hstate,pf:hzip,pf:harea,pf:hphone,pf:bstreet,pf:bcity,pf:bstate,pf:bzip,pf:barea,pf:bphone,pf:ssn,pf:sin,pf:bdate,pf:sex,pf:update_dt person /user/biadmin/person
```

ImportTsv outputs information to the console about its progress. When the command is complete, it outputs some job counters. In particular, you should verify that the input and output record counts are correct. The sample data file contains ten thousand records. **ImportTsv** also reports any bad lines it encountered.

10.7.3 Verifying the import

As soon as the data was uploaded, the background processes that take care of automatic derivation, comparison, and linking took over. You can verify that the processes are operating through different means.

To verify that derivation and comparison have taken place, you can use the HBase shell used during the table configuration in 10.6.3, “Table” on page 174.

Use the command in Example 10-6 on page 176 to start the HBase shell. After you have the shell prompt, you can use the **count** command to get a count of the records in the table. See Example 10-13 for the command and output. As you can see, all ten thousand records were imported into HBase.

Example 10-13 count command output

```
hbase(main):001:0> count 'person'
Current count: 1000, row: 1a054637-e8a6-3bcd-b425-97eb4a5ce6fe
Current count: 2000, row: 33e5c410-7867-37ef-b108-d01ba0a54386
Current count: 3000, row: 4ebd752e-1ed6-3dd8-91cd-274ecd7042e0
Current count: 4000, row: 676727a2-f547-3b47-ad77-bc2fb6c9391d
Current count: 5000, row: 80cd9900-ce58-33db-8d7d-595fec5cb741
Current count: 6000, row: 9992bbbf-e9c2-3679-8cd5-e46d5345e15c
Current count: 7000, row: b2a8bc5b-0985-3d1b-b6d6-e3572f0b66c4
Current count: 8000, row: cc73d914-74a0-3764-9ea3-ba10f289a0d1
Current count: 9000, row: e5fe475b-8cf5-3ee2-be09-4a8cedc5c39c
Current count: 10000, row: fffb0091-56b7-352e-aa59-a9174e0fecbb
10000 row(s) in 0.8560 seconds
```

10.7.4 Verifying derivation

To verify that derivation has taken place, you must take a look at the metadata for a record with a **get** command. See Example 10-14 for the command and output. The **get** command retrieves only the column family used by PME for BigInsights. The presence of output here confirms that derivation has taken place. Left the third argument off would have retrieved all the columns for the specified row. If you did that by accident, that is okay, as long as you see data in the “pme” column family, you know derivation has taken place.

Example 10-14 get command output for verifying derivation

```
hbase(main):002:0> get 'person', '594ad504-e9a1-3e34-b1c7-3e00a1a6f090', 'pme'
COLUMN      CELL
pme:b        timestamp=463738504531218434, value=\x00\x0D\x13\xCBE\xFC\xA7\x10\xCB\x01\x1BX[\xC1\xA5d\x05\x8F!\x7FMX(T\x8E:#?EX\xB9\xB9'\x05%\xFF\xC9W;5\x0C\x84:0Y\xB66\xDF\xEB\xF4<d*\xBDm\x98\xA6\x95=\xB3\xB2\xBDwe\xB3*IS\xC0\x15\x15B0<K\xA9\xD9\x1B\xA2W\x1A\x10_\xAF\x8E\x5C\x9F[\x81\xEEj\x0Bp\xF6m\xB9"\xCEx\x0A\x962\x9B\x1A\xE3,
pme:c        timestamp=463738504531218434, value=\x00\x01\x01\x00\xB5SMITH:MICHAEL ---[full comparison value omitted]---
pme:h        timestamp=463740079324598272, value=\x00\x00\x01\x06o\x87\xB6\x7FA\x10\x00\x01\x00\x01\x00\x06o\x87\xB6\x7F\x01\x10\x00\x01\x1A
3 row(s) in 0.0490 seconds
```

You might notice that the data in Example 10-14 on page 181 is not easily interpreted. In particular, the data in the “pme:b” and “pme:h” columns are not useful in their binary form. The “pme:c” value contains the standardized comparison value for the record and is somewhat readable.

Note: The comparison value text contains all the data used by the matching algorithm to compare records. Depending on your algorithm configuration and the data used for matching, this could contain sensitive information. If that is the case, you should treat this string just like the sensitive data it came from.

The algorithm we use for the lab scenario works with identifying information such as names, addresses, birth dates, and government identifiers. Because the comparison value contains the data used by the algorithm, some of which is considered sensitive identifying information, we exclude it from the output here.

Another useful way to verify derivation has occurred and to see what all that binary data means is to use the `pme_dump` command. See Example 10-15 for the command and output.

Example 10-15 bme_dump command output for verifying derivation

```
hbase(main):003:0> pme_dump 'person', '594ad504-e9a1-3e34-b1c7-3e00a1a6f090'
ROW          DATA
594ad504-e9a1-3e34-b1c7-3e00a1a6f090 column=[type=recmeta], value=[srcRecno=1, memRecno=4637385
04531218432, entRecnos={1:463738504527024128}, selfScores=
{1:282}, bktHashes={1426310658429864705, 19704257242362034
07, 2413732966322966074, 2539824962235606789, 273812847495
7220996, 4192949892221496308, 4351650133022647957, 4446093
783966528298, 5283777984619237180, 5452127536619657744, 68
94886082989228526, 7641325396469293774, 864989117841322065
2}, cmpVal=SMITH:MICHAEL (full cmpVal omitted) ]
1 row(s) in 0.1240 seconds
```

The output in Example 10-15 is considerably more readable than the output from Example 10-14 on page 181. The output here contains the record metadata which consists of several pieces of information. The source record number identifies the source in our algorithm configuration that the record came from. In the lab scenario, all the records come from the same source.

The member and entity record numbers are assigned by the system. Do not worry if yours do not match. The self scores show how well this record scores against itself for all the configured algorithms. The number before the colon, “1” in this case, is the entity type number and, like the source record number, comes from the algorithm configuration. Because our lab scenario only has one algorithm, or entity type, configured, you see the single entity record number and score.

The bucket hashes shown identify all the buckets that this record belongs to. When comparing records, we only compare records that belong to the same bucket. We use the bucket information to verify that comparisons have taken place in 10.7.5, “Verifying comparison” on page 183.

Finally, the last piece of metadata is the comparison value. As mentioned in the note on page 182 it can contain sensitive information. We exclude it from the output for this particular reason.

10.7.5 Verifying comparison

To verify that comparison has taken place, you look for a bucket that has multiple records assigned to it. Those buckets have a comparison cache that is created when comparison takes place.

In Example 10-15 on page 182, one of the buckets listed was 1426310658429864705. Example 10-16 shows the command to retrieve this bucket from the bucket index table and the output. The output is not particularly helpful. It seems HBase was unable to find the bucket contents.

Example 10-16 Looking for a bucket

```
hbase(main):004:0> get 'person.pmebktidx', 1426310658429864705
COLUMN          CELL
0 row(s) in 0.0140 seconds
```

The **pme_dump** command provides a different result. Example 10-17 contains the output from **pme_dump** for the 1426310658429864705 bucket. The output shows that this bucket contains multiple records. A bucket that only contains a single record does not have anything to compare and is not useful to verify that comparisons are taking place.

Several records were deleted from the output for space and because they are not relevant for verifying that comparison has taken place. The final row in the output with the type “bktcomp” is the comparison information created when record comparisons take place. The matchScores value shown contains sixteen comparisons. Each comparison contains two member record numbers separated by a tilde and their comparison score after the colon. The sixteen match scores represent the comparison values that exceeded the algorithm’s autolinking threshold from all the combinations of comparisons that can be made with the nine records in the bucket.

Example 10-17 pme_dump command output for verifying comparison

```
hbase(main):005:0> pme_dump 'person.pmebktidx', 1426310658429864705
ROW          DATA
```

```

\xD3\xA6\xFD,\x13\xC column=[type=bktidx, recRow=0101fc9d-245a-3d82-ad80-a1d3fd
BE\xFC\xA7\x10\xCB\x 57b607], value=[srcRecno=1, memRecno=463738515516100609, e
01 [bktHash=14263106 ntRecnos={1:463738515520294913}, selfScores={1:292}, bktHa
58429864705] shes={}, cmpVal=SMITH:MIKE (full cmpVal omitted) ]
\xD3\xA6\xFD,\x13\xC column=[type=bktidx, recRow=26e41a05-33ed-3152-a678-e4a024
BE\xFC\xA7\x10\xCB\x f122b7], value=[srcRecno=1, memRecno=463738509979619331, e
01 [bktHash=14263106 ntRecnos={1:463738509979619337}, selfScores={1:285}, bktHa
58429864705] shes={}, cmpVal=SMITH:MIKE (full cmpVal omitted) ]
\xD3\xA6\xFD,\x13\xC column=[type=bktidx, recRow=594ad504-e9a1-3e34-b1c7-3e00a1
BE\xFC\xA7\x10\xCB\x a6f090], value=[srcRecno=1, memRecno=463738504531218432, e
01 [bktHash=14263106 ntRecnos={1:463738504531218433}, selfScores={1:282}, bktHa
58429864705] shes={}, cmpVal=SMITH:MICHAEL (full cmpVal omitted) ]
... [6 records omitted for brevity] ...
\xD3\xA6\xFD,\x13\xC column=[type=bktcomp, entTypeno=1], value=[checksum=363309
BE\xFC\xA7\x10\xCB\x 1628, matchScores={463738501523902464~463738515516100609:2
01 [bktHash=14263106 85, 463738501523902464~463738509979619331:285, 46373850152
58429864705] 3902464~463738501553262594:292, 463738501553262594~4637385
15516100609:285, 463738501553262594~463738509979619331:285
, 463738504522829826~463738506280243200:244, 4637385045228
29826~463738506276048896:275, 463738504522829826~463738504
531218432:264, 463738504522829826~463738504527024131:282,
463738504527024131~463738506280243200:244, 463738504527024
131~463738506276048896:275, 463738504527024131~46373850453
1218432:264, 463738504531218432~463738506280243200:225, 46
3738504531218432~463738506276048896:257, 46373850627604889
6~463738506280243200:237, 463738509979619331~4637385155161
00609:285}]
10 row(s) in 0.1500 seconds

```

The output from **pme_dump** shows that the row keys were hashed for better distribution. This is why we could not find the bucket using the HBase **get** command in Example 10-16 on page 183. Using the raw row key as displayed in Example 10-18 with the HBase **get** command produces results similar to those in Example 10-17.

Example 10-18 get command output for verifying comparison

```

hbase(main):006:0> get 'person.pmebktidx', "\xD3\xA6\xFD,\x13\xCBE\xFC\xA7\x10\x
CB\x01"
COLUMN          CELL
b:0101fc9d-245a-3d82 timestamp=463738515520294915, value=\x00\x00\x00\x01\x06o\
-ad80-a1d3fd57b607 x87\xB9\x0E\x01\x10\x01\x01\x00\x01\x00\x06o\x87\xB9\x0EA\
x10\x01\x01$\x00\x00\x00\x01\x01\x00\xB6SMITH:MIKE
---[full comparison value omitted]---
b:26e41a05-33ed-3152 timestamp=463738509983813632, value=\x00\x00\x00\x01\x06o\
-a678-e4a024f122b7 x87\xB7\xC4\x01\x10\x03\x01\x00\x01\x00\x06o\x87\xB7\xC4\x
01\x10\x09\x01\x1D\x00\x00\x00\x01\x01\x00\xAFSMITH:MIKE
---[full comparison value omitted]---
b:594ad504-e9a1-3e34 timestamp=463738504531218434, value=\x00\x00\x00\x01\x06o\
-b1c7-3e00a1a6f090 x87\xB6\x7FA\x10\x00\x01\x00\x01\x00\x06o\x87\xB6\x7FA\x10

```



```

\01\01\1A\00\00\00\01\01\00\00\B5SMITH:MICHAEL
---[full comparison value omitted]---
... [6 records omitted for brevity] ...
c:\00\01 timestamp=463738522990350338, value=\00\x8C\xD8\x8C\x94,\
x07\x06o\x87\xB5\xCC\x01\x10\x00\x03\x06o\x87\xB9\x0E\x01\
x10\x01\x01\x1D\x06o\x87\xB7\xC4\x01\x10\x03\x01\x1D\x06o\
x87\xB5\xCD\xC1\x10\x02\x01$\x06o\x87\xB5\xCD\xC1\x10\x02\
x02\x06o\x87\xB9\x0E\x01\x10\x01\x01\x1D\x06o\x87\xB7\xC4\
x01\x10\x03\x01\x1D\x06o\x87\xB6~\xC1\x10\x02\x04\x06o\x87\
\xB6\xE7\x81\x10\x00\x00\xF4\x06o\x87\xB6\xE7A\x10\x00\x01\
\x13\x06o\x87\xB6\x7FA\x10\x00\x01\x08\x06o\x87\xB6\x7F\x0\
1\x10\x03\x01\x1A\x06o\x87\xB6\x7F\x01\x10\x03\x03\x06o\x8\
7\xB6\xE7\x81\x10\x00\x00\xF4\x06o\x87\xB6\xE7A\x10\x00\x0\
1\x13\x06o\x87\xB6\x7FA\x10\x00\x01\x08\x06o\x87\xB6\x7FA\
x10\x00\x02\x06o\x87\xB6\xE7\x81\x10\x00\x00\xE1\x06o\x87\
xB6\xE7A\x10\x00\x01\x01\x06o\x87\xB6\xE7A\x10\x00\x01\x06\
o\x87\xB6\xE7\x81\x10\x00\x00\xED\x06o\x87\xB7\xC4\x01\x10\
\x03\x01\x06o\x87\xB9\x0E\x01\x10\x01\x01\x1D
10 row(s) in 0.0790 seconds

```

As you can see in Example 10-18 on page 184, the data is not usable when read from HBase directly. Any time you want to look at the metadata created by PME for BigInsights in the HBase console, use the **pme_dump** command.

For additional information about **pme_dump** and the other HBase shell commands available for PME for BigInsights, see the *Additional HBase Shell commands* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.swg.im.mdmhs.pmebi.doc/topics/hbase_console_commands.html

10.7.6 Verifying linking

To verify linking, you can use the **pme_dump** command with the entity index table and a row key for the table. As you might expect, the entity index table row keys are the entity record numbers. The record in Example 10-15 on page 182 belongs to entity 463738504527024128. Your identifier will be different so be sure to use your identifier. See Example 10-19 for the command and output.

Example 10-19 pme_dump command output for verifying linking

```

hbase(main):007:0> pme_dump 'person.pmeentidx', 463738504527024128
ROW      DATA
7\xE44G\x06o\x87\xB6 column=[type=entidx, recRow=594ad504-e9a1-3e34-b1c7-3e00a1a
\x7F\x01\x10\x00 [en 6f090], value=
tRecno=4637385045270
24128]
7\xE44G\x06o\x87\xB6 column=[type=entidx, recRow=6f892880-c444-3ba2-9867-ec128ce

```

```

\x7F\x01\x10\x00 [en 6c348], value=
tRecno=4637385045270
24128]
7\xE44G\x06o\x87\xB6 column=[type=entidx, recRow=733953e6-bf13-35b6-a111-9116334
\x7F\x01\x10\x00 [en e2ad9], value=
tRecno=4637385045270
24128]
7\xE44G\x06o\x87\xB6 column=[type=entidx, recRow=97d0d6d6-1e45-35cf-8a43-3a65159
\x7F\x01\x10\x00 [en 6c7ef], value=
tRecno=4637385045270
24128]
7\xE44G\x06o\x87\xB6 column=[type=entidx, recRow=b05355b3-6ebc-34f5-8d95-3526971
\x7F\x01\x10\x00 [en 69b42], value=
tRecno=4637385045270
24128]
5 row(s) in 0.0300 seconds

```

The row keys are again hashed. The output here identifies all the records that are a part of the identified entity. We know from running **pme_dump** in Example 10-17 on page 183 that there were nine records in the bucket plus the comparison data that made up the 10 rows in the output. It is not unreasonable to expect those records to all be in the same entity as long as their comparison scores are above the autolinking threshold specified by the algorithm configuration. In the lab scenario, the bucket actually contains two entities. One entity consists of four records; the other entity contains the remaining five records (Example 10-19).

Another quick way to verify that entity linking has taken place is to count the entities in the system. You can do this in the HBase shell using the **count** command on the entity index table. See Example 10-20 for the command and output.

Example 10-20 count command output

```

hbase(main):008:0> count 'person.pmeentidx'
Current count: 1000, row: )\xAF\x95\xAA\x06o\x87\xB7\xD8\xC1\x10\x05
Current count: 2000, row: T\x13\x12\x90\x06o\x87\xB0\x14A\x10\x0B
Current count: 3000, row: \x80\xA8\xCE\xE2\x06o\x87\xB9\x10\x81\x10\x03
Current count: 4000, row: \xACL\xA6i\x06o\x87\xBA\xD0\xC1\x10\x06
Current count: 5000, row: \xD7\x93\x17^\x06o\x87\xBB\xBB\x81\x10\x07
5906 row(s) in 0.4760 seconds

```

There are 5906 records in the entity index. We know that we started with ten thousand records which all start out as their own entity. The fact that there are less than ten thousand records in the entity index suggests that entity linking has taken place.

When automatic processing is enabled, as it is for the lab scenario, data derivation happens immediately when a row is inserted or updated in the table. The comparison is queued for processing after derivation and happens quickly as well. Linking, however, is operating on a polling interval and quiet period. That means it can take several minutes from when a change is made before linking takes place.

Note: Autolinking behavior can be influenced by the site configuration properties as described in 10.6.1, “Site” on page 173.

You can view the Hadoop Map/Reduce Administration console described in 10.6.3, “Table” on page 174 to see if a linking job is in progress or to verify that a linking job has completed. The linking jobs are named to identify the table being linked as shown in Figure 10-4.



Completed Jobs								
Jobid	Started	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Re C
job_201405061343_0001	Tue May 06 13:53:32 EDT 2014	NORMAL	biadmin	importtsv_person	100.00% 	1	1	10
job_201405061343_0002	Tue May 06 13:59:37 EDT 2014	NORMAL	hbase	PME Link person	100.00% 	3	3	10

Figure 10-4 Completed jobs

10.8 Probabilistic search

Searching the data stored in HBase can be accomplished in one of several ways. You can search interactively using the Dashboard and ad-hoc criteria. You can also perform large batch searches using the PME Search application. If you want to perform searches programatically, you can use PME for BigInsights APIs to perform searches as well.

For more information about the PME for BigInsights API, see the *Using the Java and REST APIs for big data matching* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.svg.im.mdmhs.pmebi.doc/topics/pme_bi_api.html

10.8.1 Dashboard

The Big Data Matching Dashboard is a sample application that is shipped with PME for BigInsights. It is a purpose-built sample that allows you to do ad-hoc searches against the data stored in HBase using the probabilistic matching algorithm. You can access the Dashboard by pointing your browser to **http://<hostname>:<port>/pme** where the host name and port correspond to the WebSphere Liberty Profile you created in 10.4.2, “Installing PME for BigInsights” on page 171. Figure 10-5 shows the search windows.

Big Data Matching Dashboard

IBM

Input table:

person

Algorithm:

mdmper

Search Type:

Entity

Member

Minimum Score:

Search

First Name

Last Name

SSN

SIN

Gender

Birth Date

Street

City

State

ZipCode

Business Street

Business City

Business State

Business Zip Code

Home Phone

Business Phone

Entity Id	Score	First Name	Last Name	SSN	SIN	Gender	Birth Date	Street	City
No items to display									

Figure 10-5 Big Data Matching Dashboard

The search window is divided into two parts. The search criteria on the left and the results on the right. To perform a search, complete the following steps:

1. Select the table you want to search and the algorithm to use.

In the lab scenario, there is only one of each so we accept the defaults.

2. Select the type of search you want to perform.

You can search for entities or member records that match the criteria. The minimum score allows you to exclude records that score below the provided value when compared to the criteria.

We perform an entity search first. Using our earlier example data, enter the criteria in Table 10-3 for the search.

Table 10-3 Search Criteria

Field	Criteria
First Name	Mike
Last Name	Smith
Birth Date	2013-05-27
Minimum Score	50

3. After entering the criteria, click **Search**. Figure 10-6 shows the results of our entity search.

Big Data Matching Dashboard

IBM

Input table: person

Algorithm: mdmper

Search Type: ☒ Entity ☐ Member

Minimum Score: 50

Search

First Name: Mike

Last Name: Smith

SSN:

SIN:

Gender:

Birth Date: 2013-05-27

Street:

City:

State:

ZipCode:

Business Street:

Business City:

Business State:

Business Zip Code:

Home Phone:

Business Phone:

Entity Id	Score	First Name	Last Name	SSN	SIN	Gender	Birth Date	Street	City	State	ZipCode
463738504527024128	82	MICHAEL	SMITH		-3410	M	2013-05-27				
463738504527024128	82	MICHAEL	SMITH		-3410	M	2013-05-27				
463738504527024128	82	MICHAEL	SMITH		-3140	M	2013-05-27				
463738504527024128	82	MICHAEL	SMITH		-3410	M	2013-05-27				
463738504527024128		MICHAEL	SMITH		-3410	M	2031-05-27				

Figure 10-6 Entity search results

On the left side of the results, you see the entity identifier assigned by PME for BigInsights. Your identifier values will likely be different but the remainder of the output should be similar.

When performing an entity search, all members of an entity that match the criteria are returned. As you can see in the fifth row, no score is present. Closer examination of that row shows the birth date value does not match our criteria. The last two digits of the year are transposed. You might also have noticed that we did not search for the exact name from the earlier records. In this case, we used a common nick name. Because of these differences and the criteria, the fifth record listed in the results has no comparison score.

If we leave the search criteria and switch the search type to member, we get a similar result. Figure 10-7 shows the results for the member search.

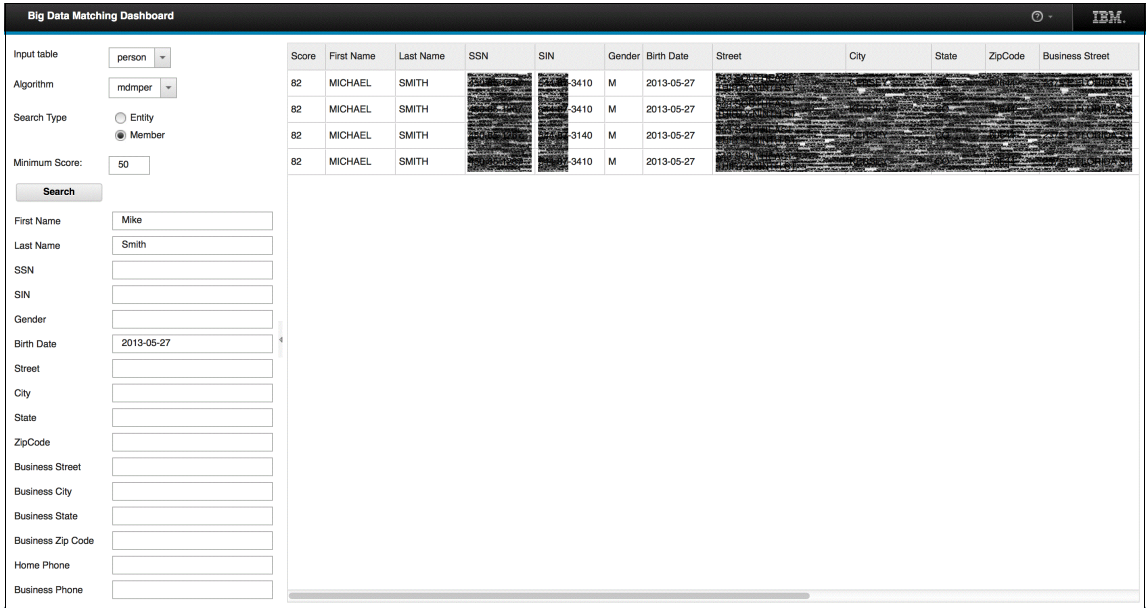


Figure 10-7 Member search results

The first thing you might notice is the entity identifier is no longer included in the search result. Because we did a member record search, the entity identifier is not included. Also notice that the record that did not score against the criteria is also not included in the search result. When performing a member search, only those records that score against the criteria are included.

10.8.2 PME Search

The Dashboard is useful if you have a small amount of data and want to perform ad-hoc searches of the data. When you have a large set of criteria that you want to search with as a batch, the PME Search application can help you do that.

The PME Search application uses a separate table from the record table as its input and can use the same table or another table to collect its output. The criteria in the input table should have the same qualifiers as the record table. This is how the search application knows which algorithm fields the criteria are mapped to.

The application works by dividing the search criteria in the input table and running the searches in parallel across the cluster against the record table. The

application then writes the results of the search into the output table using the input record's row key as the row key for the output record.

For more information about the PME Search application and detailed steps about how to run it, see the *Running the PME Search application* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.svg.im.mdmhs.pmebi.doc/topics/running_pme_search.html

10.9 360-degree view

In 10.2.2, “360-degree view” on page 168, we introduced the concept of a composite view. By default, PME for BigInsights assembles a composite view by selecting the most current value for a given column from the records that make up the entity. This is by definition the Entity Most Current Attribute (EMCA) view.

Composite views are available when retrieving entities through the programming APIs and when using the PME Extract application.

10.9.1 PME Extract

The PME Extract application is used to extract the entity data from PME for BigInsights. It allows you to extract the entity linkage data that shows which records belong to each entity. It also allows you to extract a composite view of the entity.

Installation

Because users' needs vary, PME for BigInsights MapReduce applications are installed as needed. To use the PME Extract application, you must upload the application to InfoSphere BigInsights and deploy it. You can find the instructions to upload and deploy the application in the *Configuring and deploying the applications* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.svg.im.mdmhs.pmebi.doc/topics/config_deploy_apps.html

Preparation

Similar to the PME Search application, PME Extract writes its output to an HBase table. Before you can run the application, you must create the table. The application expects the table to have the same column families as the source table. We create the output table with a statement similar to the one we used to create the record table in Example 10-7 on page 176. If necessary, start the HBase Shell using the command in Example 10-6 on page 176. Create the output table in the HBase shell using the command in Example 10-21.

Example 10-21 Output table create command

```
create 'person.extract', {NAME => 'pf', COMPRESSION => 'SNAPPY'}, {NAME
=> 'pme', COMPRESSION => 'SNAPPY'}, {NUMREGIONS => 4, SPLITALGO =>
'HexStringSplit', DURABILITY => 'ASYNC_WAL'}
```

The table creation command in Example 10-21 includes a second column family “pme” that was created automatically when we enabled the record table for use with PME for BigInsights. If you want to include the linkage information in the extract output, the “pme” column family must exist. The rest of the table creation settings are the same as they were when we created the record table. You can use the **list** command to verify that the extract table was created.

Extraction

To run the PME Extract application, open the InfoSphere BigInsights console. Click **Run** on the left side of the Applications tab and select the **PME Extract** application. The application and its parameters are displayed on the right side. Enter the parameters listed in Table 10-4.

Table 10-4 PME Extract Parameters

Parameter	Value
Record table name	person
Output table name	person.extract
Algorithm name	mdmper
View type	composite
View versions	1
View arguments	
Use Bulk Load	unchecked
Working directory	/tmp

You must enter the first three parameters listed in Table 10-4 on page 192. The rest of the values are the defaults. These parameters scan the entities in the “person” table and write the selected output to the “person.extract” table using the “mdmper” algorithm.

In our example, we obtain a composite output that includes data attributes. If you want just the entity-record linkages, select “linkonly” for your view type. By default you only get the most current version of a column or attribute. We ignore view arguments for now. The bulk load option is useful with large volumes of data.

Finally, the application requires a place in the distributed file system to write temporary files. By default that is the /tmp directory.

To run the application, click the **Run**. You can assign an execution name or leave it blank. The execution name makes it easy to identify an execution in the Application History view below the application parameters.

To verify and examine the output of the application, return to the HBase console. If necessary, start the HBase Shell using the command in Example 10-6 on page 176.

Example 10-22 shows the results of running the **count** command on the output table. The 5906 row counts match the number of entitles in the linkage verification steps in Example 10-20 on page 186.

Example 10-22 count command output from person.extract table

```
hbase(main):001:0> count 'person.extract'
Current count: 1000, row: 29af95aa-e24a-32f9-a77b-5852132507fd
Current count: 2000, row: 54131290-f052-33e1-a965-106cece81023
Current count: 3000, row: 80a8cee2-758a-3141-a445-02a73098bc9b
Current count: 4000, row: ac4ca669-a127-3515-af81-c64e548dc012
Current count: 5000, row: d7931760-f26a-309a-8785-c4a914fd8630
5906 row(s) in 0.7970 seconds
```

Let us look for our example person from earlier, Michael Smith. Example 10-23 shows the **scan** command to scan the output table for his entity and the results.

Example 10-23 Output table scan results

```
hbase(main):002:0> scan 'person.extract', {FILTER =>
"SingleColumnValueFilter('pf','fname',=,'binary:MICHAEL') AND
SingleColumnValueFilter('pf','lname',=,'binary:SMITH')", COLUMNS => ['pf:bdate',
'pf:fname', 'pf:lname', 'pf:memidnum', 'pf:sin', 'pf:update_dt', 'pme']}
ROW                                COLUMN+CELL
37e43447-4d18-3eef-8 column=pf:bdate, timestamp=1399398810537, value=2031-05-27
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:fname, timestamp=1399398810537, value=MICHAEL
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:lname, timestamp=1399398810537, value=SMITH
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:memidnum, timestamp=1399398810537, value=7402
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:sin, timestamp=1399398810537, value=###-##-3410
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:update_dt, timestamp=1399398810537, value=2012-0
f7b-9c185648d382 5-17T08:12:46.022-0500
37e43447-4d18-3eef-8 column=pme:, timestamp=1401758204339, value=\x06o\x87\xB6\
f7b-9c185648d382 x7F\x01\x10\x00
```

```

37e43447-4d18-3eef-8 column=pme:594ad504-e9a1-3e34-b1c7-3e00a1a6f090, timestamp
f7b-9c185648d382      =1401758204339, value=
37e43447-4d18-3eef-8 column=pme:6f892880-c444-3ba2-9867-ec128ce6c348, timestamp
f7b-9c185648d382      =1401758204339, value=
37e43447-4d18-3eef-8 column=pme:733953e6-bf13-35b6-a111-9116334e2ad9, timestamp
f7b-9c185648d382      =1401758204339, value=
37e43447-4d18-3eef-8 column=pme:97d0d6d6-1e45-35cf-8a43-3a651596c7ef, timestamp
f7b-9c185648d382      =1401758204339, value=
37e43447-4d18-3eef-8 column=pme:b05355b3-6ebc-34f5-8d95-352697169b42, timestamp
f7b-9c185648d382      =1401758204339, value=
1 row(s) in 0.3860 seconds

```

The **scan** command listed in Example 10-23 scans the “person.extract” table looking for rows that match the filter criteria specified. In this case, we look for rows that have MICHAEL for the first name column and SMITH for the last name column. To reduce the output, we limit results to the data columns that are significant for our analysis.

The “pf” columns come directly from the record table. By default, the column value with the most recent time stamp is returned. The **sin** column value is masked in Example 10-23. However, it does appear unmasked in the output.

The “pme” columns contain the linkage information. The first value in the “pme” column family has an empty qualifier and is the entity identifier encoded as bytes. The HBase shell encodes the non-printable bytes using an escaped hex notation. The eight bytes, \x06o\x87\xB6\x7F\x01\x10\x00, in big-endian order, are the number 463738504527024128. This is the same **entrecno** we saw earlier in Example 10-15 on page 182. The remaining “pme” family columns identify the rows that belong to the entity. Each non-empty qualifier identifies a row participating in the entity.

The output in Example 10-23 on page 193 comes from the third record in the set of Michael Smith records of the input data set. Why the third record? PME for BigInsights assigns member and entity record numbers to the records when the record data is initially derived. With no timestamp column identified in the data, ImportTsv assigns the same timestamp to all of the records imported in the batch.

As you can see in Example 10-23 on page 193, all the columns in the “pf” family have the same timestamp as do all of the records in the entity. When this happens, the tie breaker is the record with the highest member record number (**memrecno**). Because **memrecnos** are assigned semi-sequentially by the system, the record created last should have the highest **memrecno**. The third record was processed last during the import because of the way the HBase client groups records together destined for the same region before sending them to the server.

As you can see in the data in Example 10-23 on page 193, the birth date returned in the entity is 2031-05-27. If you recall from Figure 10-6 on page 189, the last row in the search results contained a birth date that had the last two digits of the year transposed from our search criteria of 2013-05-27. Is this the best record or set of columns to represent our entity? The data contains an update date (update_dt) column that indicates when it was last updated in the source system. Could we use this update date value to help us decide which values to use when building the 360-degree view for our entity?

10.9.2 Composite filters

The extracted data in Example 10-23 on page 193 shows Michael's SIN in the output. There might be times when you do not want identifiers like that to be shown completely and times when you do. Wouldn't it be nice to be able to mask the output as necessary?

Using a custom composite filter allows users to modify the output behavior of the composite views of entities. The programming API is flexible and is based on the HBase `org.apache.hadoop.hbase.filter.Filter` class. The PME for BigInsights API includes a base class that users must use when creating custom composite filters. The `com.ibm.mdm.bi.pme.hbase.api.CompositeFilter` class serves as the starting point for all customizations for composite views.

The usage of `CompositeFilter` is slightly different than `Filter`. The `Filter` class is used to filter rows during operations in HBase. The context of `Filter` invocation is per row in HBase. The context of `CompositeFilter` invocation for composite views is per entity. One or more HBase rows that are part of the same entity are passed to a `CompositeFilter`. To make each row unique, a composite row key is used. This key can be retrieved from each `Cell` using the `getCompositeRowKey(Cell)` method on `CompositeFilter`. This method decodes the `Cell`'s row key byte array into an object with methods to access metadata about the record.

Just like the HBase `Filter` class, the `CompositeFilter` subclasses can expect the following call sequence:

- ▶ `reset()`: reset the filter state before filtering a new entity.
- ▶ `filterAllRemaining()`: true means filtering is completed; false means keep going.
- ▶ `filterRowKey(byte[],int,int)`: true means drop this entity; false means include.
- ▶ `filterKeyValue(Cell)`: decides whether to include or exclude this `Cell`.
- ▶ `transform(Cell)`: if the `Cell` is included, let the filter transform the `Cell`.

- ▶ `filterRowCells(List)`: allows direct modification of the final list of `Cells`.
- ▶ `filterRow()`: last chance to drop the entire entity based on the sequence of filter calls.

The base classes have default implementations that cause no filtering to occur. This means each filter can pick which methods they need to support their filtering needs.

For more information about the PME for BigInsights API and how to access the Javadoc, see the link to the IBM Knowledge Center in 10.8, “Probabilistic search” on page 187.

Creating a composite filter

You can create a custom composite filter to solve some of the problems that we pointed out earlier such as selecting the correct data and dynamically modifying the filter behavior with view arguments. If you use the IBM InfoSphere BigInsights Quick Start edition, you can use the bundled Eclipse installation to create the custom composite filter. The directions that follow are based on using Eclipse or similar integrated development environment (IDE).

Creating the project

The first thing you need is a project to contain our custom composite filter. Create a new Java Project named `CustomCompositeFilter`. Make sure the execution environment selected is “JavaSE-1.6” to correspond to the JRE used by InfoSphere BigInsights.

Gathering dependencies

Before creating a composite filter, you must gather up a few dependencies required to create the class. Obtain a copy of the PME for BigInsights API and certain HBase client libraries.

Table 10-5 lists the dependencies required to construct a custom composite filter. The PME for BigInsights API files are installed into the path selected during installation with the IBM Installation Manager. This is typically `~/IBM/MDMPME4BI` if you ran Installation Manager as a non-administrator or `/opt/IBM/MDMPME4BI` if you ran Installation Manager as an administrator. The PME for BigInsights API Javadoc are also located in the same path in the file `com.ibm.mdm.bi.pme.hbase.api.javadoc.zip`.

Table 10-5 Composite Filter Dependencies

Path	Library	Description
MDMPME4BI Install	com.ibm.mbm.bi.pme.hbase.api.jar	PME for BigInsights API
\$BIGINSIGHTS_HOME/hbase/lib	hbase-client-0.96.0.jar	HBase client API
\$BIGINSIGHTS_HOME/hbase/lib	hbase-common-0.96.0.jar	Hbase common API
\$BIGINSIGHTS_HOME/hbase/lib	protobuf-java-2.5.0.jar	Google Protocol Buffer API

Create a new `lib` folder at the root of the project. Add the libraries listed in Table 10-5 to the newly created `lib` folder. Add all four libraries to the project build path. Your project should resemble Figure 10-8 after copying the libraries to the project and adding them to the project build path.

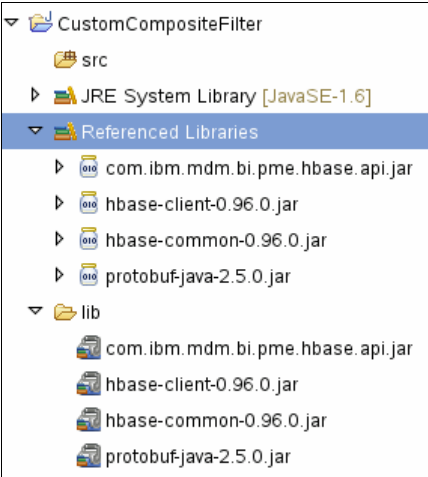


Figure 10-8 CustomCompositeFilter project with libraries

Creating the filter class

In the IDE, create a new Java class with the properties listed in Table 10-6.

Table 10-6 Filter class properties

Property	Value
Package	com.example.pme.hbase.filter
Name	RecordDateValueFilter
Superclass	com.ibm.mdm.bi.pme.hbase.api.CompositeFilter

Your newly created class should look similar to Example 10-24.

Example 10-24 RecordDateValueFilter class listing

```
package com.example.pme.hbase.filter;

import com.ibm.mdm.bi.pme.hbase.api.CompositeFilter;

public class RecordDateValueFilter extends CompositeFilter {

}
```

This class skeleton is the starting point for the custom composite filter. Add some documentation to the class to make it clear what this filter does. Example 10-25 contains the class Javadoc that explains what the custom composite filter does.

Example 10-25 RecordDateValueFilter class Javadoc

```
/**
 * This filter returns a composite entity view from a set of input rows that are
 * the source data for the entity records. It does this by identifying the most
 * current rows based on update dates from the source system included in the
 * data. In the event not all fields are present in the most current row, the
 * filter will try to use data from the next most current row to fill in any
 * missing information.
 */
```

To accomplish the filtering as described, you need a few pieces of information about the table. The three constants listed in Example 10-26 represent the facts about the table we used and the data contained in it.

Example 10-26 Initial constants for RecordDateValueFilter class

```
/* This format should match the date string format read from HBase. */
private static final DateFormat DATE_FORMAT =
    new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSZ");

/* The family of the data columns. */
private static final byte[] FAMILY = Bytes.toBytes("pf");
/* The qualifier of the date column. */
private static final byte[] DATE_QUALIFIER = Bytes.toBytes("update_dt");
```

Here you declare the `DateFormat` object that you will use later to parse the string in the date column. Declare the constants for the column family and qualifier as byte arrays. The HBase API operates primarily with byte arrays. The `Bytes` utility in the `org.apache.hadoop.hbase.util` package contains many helper methods to convert between byte arrays and standard Java primitives.

Filtering data

Filtering cells based on data within a cell in the row presents a unique challenge. Based on the call sequence on page 195, you have two opportunities to examine cells. The first opportunity happens when each cell in the entity is passed in turn to the `filterKeyValue(Cell)` method. The second opportunity is in the `filterRowCells(List)` when all of the cells are available for examination and modification.

To accomplish the goal of selecting cells from the row with the most recent update date, you must record when each record was updated during the `filterKeyValue(Cell)` method invocations and then use that data in the `filterRowCells(List)` method to remove unwanted cells from the output. Record the update dates as state within the filter. Example 10-27 contains a new variable `lastUpdates` and an implementation of `reset()` to reset the state of the filter between entities.

Example 10-27 lastUpdates variable and initial reset() method implementation

```
/* Sorted storage for the update date and corresponding memrecno. */
private TreeMap<Date, Long> lastUpdates = new TreeMap<Date, Long>();

/*
 * This method is called at the start of the filtering process. Its main
 * purpose is to clear any state stored during the prior usage of the
 * filter.
 */
@Override
public void reset() throws IOException
{
    // clear any prior entity's data from the filter
    lastUpdates.clear();
}
```

In this example, we use a `TreeMap` to store the `memrecno` ordered by the last update date. This allows us to have an ordered list of `memrecnos` when we need to decide which cells are included in the entity output.

To fulfil the requirements stated in the filter Javadoc in Example 10-25 on page 198, there are two things to do in the `filterKeyValue(Cell)` calls:

- Remove any empty cells from the output.

The first task is easily accomplished in this method because each cell is handled in order. The order ensures all the cells from the same column are presented together with the most current timestamp first. In the event of a tie in the timestamp the highest `memrecno` is presented first.

- Record when a member record was last updated.

The second task requires a little more effort because you must identify the cell as the update date cell and then parse the date value of the cell. After parsing, store the update date and memrecno in the `lastUpdates` state variable.

Example 10-28 presents the implementation of the `filterKeyValue(Cell)` method.

Example 10-28 filterKeyValue(Cell) implementation

```
/*
 * This method is called with each cell as ordered by the family, qualifier,
 * timestamp, type, and memrecno.
 */
@Override
public ReturnCode filterKeyValue(Cell cell) throws IOException
{
    // if this cell contains no value
    if (cell.getValueLength() == 0) {
        // discard it
        return ReturnCode.SKIP;
    }

    // if this cell contains our last update data
    if (CellUtil.matchingFamily(cell, FAMILY) &&
        CellUtil.matchingQualifier(cell, DATE_QUALIFIER)) {
        // get the memrecno from the composite row key on the cell
        long memrecno =
            CompositeFilter.getCompositeRowKey(cell).getMemRecno();
        // get the string date representation from cell bytes
        String dateString = Bytes.toString(cell.getValueArray(),
            cell.getValueOffset(), cell.getValueLength());
        try {
            // parse the date
            Date date = DATE_FORMAT.parse(dateString);
            // store it in our sorted map
            lastUpdates.put(date, memrecno);
        } catch (ParseException pe) {
            // unable to parse the date string
            throw new IOException(new StringBuilder()
                .append("The date value '").append(dateString)
                .append("' could not be parsed.").toString());
        }
    }

    // include all non-empty cells
    return ReturnCode.INCLUDE;
}
```

One last step remains in filtering unwanted cells from the entity output. You can use the update date information recorded in `filterKeyValue(Cell)` to modify the list of cells passed to `filterRowCells(List)`. Example 10-29 shows the beginning of the `filterRowCells(List)` implementation. The first thing to establish a prioritized list of memrecnos to sort the entity cells with.

Example 10-29 filterRowCells(List) implementation part 1 - prioritized memrecnos

```
/*
 * This method is invoked after all cells have been individually filtered
 * and transformed. Anything remaining in the list after this method will
 * be returned to the framework.
 */
@Override
public void filterRowCells(List<Cell> cells) throws IOException
{
    // get the prioritized memRecnos based on update dates
    final List<Long> prioritizedMemRecnos = new ArrayList<Long>();
    // reverse natural ordering to place most recent updated memrecnos first
    for (Long memRecno : lastUpdates.descendingMap().values()) {
        prioritizedMemRecnos.add(memRecno);
    }
}
```

Now that you have a prioritized list of memrecnos, you can sort the list of cells using a custom comparator. The code to perform the sort using the custom comparator is shown in Example 10-30.

Example 10-30 filterRowCells(List) implementation part 2 - sorting the cells

```
// sort the cells based on memRecno priority
Collections.sort(cells, new Comparator<Cell>() {
    @Override
    public int compare(Cell left, Cell right)
    {
        long leftMemRecno, rightMemRecno;
        try {
            // get the memrecnos for the two cells
            leftMemRecno =
                CompositeFilter.getCompositeRowKey(left).getMemRecno();
            rightMemRecno =
                CompositeFilter.getCompositeRowKey(right).getMemRecno();
        } catch (IOException ioe) {
            // this should not happen since the row keys are valid
            throw new RuntimeException(ioe);
        }

        int leftIdx, rightIdx;
        // get the index of the memrecno from the priority list
        leftIdx = prioritizedMemRecnos.indexOf(leftMemRecno);
```

```

        rightIdx = prioritizedMemRecnos.indexOf(rightMemRecno);
        // if neither memrecno is found
        if (leftIdx == -1 && rightIdx == -1) {
            // equivalent
            return 0;
        } else if (leftIdx == -1 && rightIdx > -1) {
            // left not found, right exists
            return 1; // left greater than right
        } else if (leftIdx > -1 && rightIdx == -1) {
            return -1; // left less than right
        }

        // return the difference in indexes as their relativity
        return leftIdx - rightIdx;
    }
});

```

Now, all the unwanted cells are removed from the list of cells passed to the method. To ensure that a column is only included once, keep track of the cells included in the output. Because the cells are now sorted by the prioritized memrecnos and still grouped by column, you are guaranteed to include the most up to date column value based on the update date of the row the column originated from. Example 10-31 contains the remainder of the `filterRowCells(List)` implementation.

Example 10-31 filterRowCells(List) implementation part 3 - removing cells

```

// create a unique set of fields we've seen
Set<String> fieldsSeen = new HashSet<String>();
Iterator<Cell> cellIterator = cells.iterator();
// iterate over all the sorted cells
while (cellIterator.hasNext()) {
    Cell cell = cellIterator.next();
    // construct a unique field string
    StringBuilder sb = new StringBuilder();
    sb.append(Bytes.toString(cell.getFamilyArray(),
        cell.getFamilyOffset(), cell.getFamilyLength()));
    sb.append(':');
    sb.append(Bytes.toString(cell.getQualifierArray(),
        cell.getQualifierOffset(), cell.getQualifierLength()));
    String field = sb.toString();
    // if the field has already been included in the output
    if (fieldsSeen.contains(field)) {
        // remove it from the data set
        cellIterator.remove();
    } else {
        // leave it in the data set and add the field to those seen
    }
}

```

```

        fieldsSeen.add(field);
    }
}
}
}

```

Transforming data

So far we discussed filtering data from the results. If you must modify data during filtering, there is support for that as well. The `CompositeFilter` class includes a `viewArgs` property that is passed to the filter to affect the filter behavior at run time. You can use this class to dynamically decide when to show or mask a piece of sensitive data such as these.

To make it easy for users of our filter to know what arguments the filter supports, we declare our argument name as a constant in our filter class. Example 10-32 contains the constant definition.

Example 10-32 MASK_IDENTIFIERS argument constant

```

/* View argument to mask the identifier columns */
public static final String MASK_IDENTIFIERS = "maskIdentifiers";

```

We create a new instance variable in our filter to hold the argument value that might or might not have been supplied in the `viewArgs` property of the filter. Example 10-33 shows the `maskIdentifiers` variable.

Example 10-33 maskIdentifiers instance variable

```

/* View argument to mask identifiers */
private boolean maskIdentifiers;

```

The `viewArgs` property of the filter is passed as a `String`. The structure of the `String` is up to the implementer. For our example, we use a `key=value` approach and separate key-value pairs with semi-colons. Example 10-34 shows the modifications to the `reset()` method to parse the `viewArgs` property of the filter and set the value of `maskIdentifiers`.

Example 10-34 reset() method changes to parse viewArgs and set maskIdentifiers

```

// parse the view args
Properties args = new Properties();
// we use semicolons to chain properties on a single line
// ex: key1=value1;key2=value2;
args.load(new StringReader(getViewArgs().replace(';', '\n')));
// set our args
maskIdentifiers =
    Boolean.parseBoolean(args.getProperty(MASK_IDENTIFIERS, "true"));

```

Note that the default value of `maskIdentifiers` is true if the value is not present in the `viewArgs` property.

Now that we have a means to specify that we want to mask identifiers, we need to implement the masking. First we write a simple method to mask a source string leaving the last few characters unmasked. The implementation of `maskString` is shown in Example 10-35.

Example 10-35 `maskString(String, int)` implementation

```
/**
 * A generic method to mask a string leaving a fixed number of the last
 * characters visible. Only alphanumeric values are masked. Any other
 * characters are returned as they were in the source. Values shorter than
 * the number of visible characters are not masked.
 * @param source the value to mask
 * @param numChars the number of unmasked characters
 * @return the masked value
 */
protected String maskString(String source, int numChars)
{
    // identify where to split the string to get at most the last numChars
    final int maskSplit = Math.max(0, source.length()-numChars);
    String clear = source.substring(maskSplit);
    String masked = source.substring(0, maskSplit);
    // return our new masked value
    return new StringBuilder().append(masked.replaceAll("[a-zA-Z0-9]", "**"))
        .append(clear).toString();
}
```

We can now use the `maskString` method to mask cell values. We create two new constants to identify the columns that we want to mask as shown in Example 10-36.

Example 10-36 Qualifier constants for masked identifiers

```
/* The qualifier of the SSN column. */
private static final byte[] SSN_QUALIFIER = Bytes.toBytes("ssn");
/* The qualifier of the SIN column. */
private static final byte[] SIN_QUALIFIER = Bytes.toBytes("sin");
```

Using these constants, we test each `Cell` object as it is passed to the `transformCell(Cell)` method. If we are masking identifiers and the cell is either of the two identifiers, we use the `maskString` method to mask all but the last four characters of the identifier. Example 10-37 shows the implementation of the `transformCell(Cell)` method.

Example 10-37 *transformCell(Cell)* implementation

```
/*
 * This method is called after filterKeyValue(Cell) to allow for changes to
 * the returned cell.
 */
@Override
public Cell transformCell(Cell cell) throws IOException
{
    // if we're not masking
    if (!maskIdentifiers) {
        // return quickly
        return cell;
    }

    // if this is an identifier
    if (CellUtil.matchingFamily(cell, FAMILY) &&
        (CellUtil.matchingQualifier(cell, SSN_QUALIFIER) ||
         CellUtil.matchingQualifier(cell, SIN_QUALIFIER))) {
        // get the string from the cell bytes
        String value = Bytes.toString(cell.getValueArray(),
            cell.getValueOffset(), cell.getValueLength());
        // mask all but the last four characters
        value = maskString(value, 4);
        // construct a new cell to return
        cell = CellUtil.createCell(CellUtil.cloneRow(cell),
            CellUtil.cloneFamily(cell), CellUtil.cloneQualifier(cell),
            cell.getTimestamp(), cell.getTypeByte(), Bytes.toBytes(value));
    }

    return cell;
}
```

Example 10-37 on page 205 completes the source code for our example CompositeFilter class RecordDateValueFilter. The complete source code for the file and a skeleton project is available for download from the IBM Redbooks website. For the download instructions, see Appendix A, “Additional material” on page 209.

Deploying a composite filter

With the code completed, the next task is to deploy the filter. Package the filter in a Java Archive (JAR) file for deployment by using the following steps:

1. In Eclipse, with the project selected, then select **File** → **Export...** and select **JAR file** from the Java folder in the Export dialog. Click **Next>** to advance the dialog to JAR Export.

2. In JAR Export:

- Ensure the “src” folder is selected and any remaining resources are not selected.
- The “Export generated class files and resources” and “Compress the contents of the JAR file” options should be selected by default. If they are not, select them now.
- The remaining options can be left unselected.
- We use an export destination of `com.example.pme.hbase.filter.jar` to be consistent with the package name of our filter. This file name is used for the remainder of the examples in this chapter.

3. Click **Finish** to export our class and create the JAR file.

Note: If you did not specify a path when exporting, check the workspace root directory for the JAR file.

Now that you have created our JAR file that contains the filter, you must make it available to the PME Extract MapReduce application and HBase. The easiest way to do this is to copy the file to HDFS from the master node and then copy it from there to the HBase region servers using the commands in Example 10-38.

Example 10-38 Filter deployment commands

```
hadoop fs -put -f com.example.pme.hbase.filter.jar
/biginsights/oozie/sharedLibraries/pme/com.example.pme.hbase.filter.jar

$BIGINSIGHTS_HOME/IHC/sbin/slaves.sh hadoop fs -get
/biginsights/oozie/sharedLibraries/pme/com.example.pme.hbase.filter.jar
$BIGINSIGHTS_HOME/hbase/lib/com.example.pme.hbase.filter.jar
```

The first command puts the filter JAR in the HDFS directory for PME for BigInsights shared libraries. These libraries are shared amongst all the PME for BigInsights MapReduce applications. The second command causes all the slaves to copy the file from HDFS to the HBase `lib` directory.

With the library deployed, the only remaining step is to configure the filter to be used with your table. In 10.6.3, “Table” on page 174, `pme-person.xml` file is copied from the sample to the Hadoop configuration staging directory. Edit that copy with `vi` or any text editor. The following command is for `vi`:

```
vi $BIGINSIGHTS_HOME/hdm/hadoop-conf-staging/pme-person.xml
```

Add the configuration property listed in Example 10-39 to the file.

Example 10-39 viewfilterclass property for pme-person.xml

```
<property>
  <name>pme.person.mdmper.viewfilterclass</name>
  <value>com.example.pme.hbase.filter.RecordDateValueFilter</value>
</property>
```

The property name contains the table name, person, and the algorithm the view filter applies to, mdmper. This allows each table to have a view filter defined for each algorithm supported by the table.

After saving the change, synchronize the table configuration to the cluster and restart HBase using the commands in Example 10-40.

Example 10-40 Configuration synchronization commands

```
synconf.sh hadoop
stop.sh hbase
start.sh hbase
```

Note: Restarting HBase is necessary for the region servers to load the new table configuration and add the filter JAR to the HBase class path.

In addition to the `viewfilterclass` property, there is also an optional `viewcolumns` property that can be specified to reduce the data supplied to the filter. For more information about this optional property, see the *Sample XML for PME Extract* topic in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.3.0/com.ibm.svg.im.mdmhs.pmebi.doc/topics/pme_extract_sample.html

Testing the custom filter

With the configuration deployed, you can now run the PME Extract application again. Follow the instructions in “Extraction” on page 192 to run the application again using the same parameters as the first execution.

If you run the **scan** command from Example 10-23 on page 193 again, you should see results similar to those in Example 10-41.

Example 10-41 Output table scan results

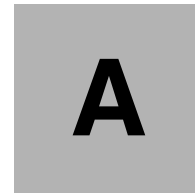
```

hbase(main):002:0> scan 'person.extract', {FILTER =>
"SingleColumnValueFilter('pf','fname',=,'binary:MICHAEL') AND
SingleColumnValueFilter('pf','lname',=,'binary:SMITH')", COLUMNS => ['pf:bdate',
'pf:fname', 'pf:lname', 'pf:memidnum', 'pf:sin', 'pf:update_dt', 'pme']}
ROW                                COLUMN+CELL
37e43447-4d18-3eef-8 column=pf:bdate, timestamp=1399398810537, value=2013-05-27
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:fname, timestamp=1399398810537, value=MICHAEL
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:lname, timestamp=1399398810537, value=SMITH
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:memidnum, timestamp=1399398810537, value=7401
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:sin, timestamp=1399398810537, value=***-**-3410
f7b-9c185648d382
37e43447-4d18-3eef-8 column=pf:update_dt, timestamp=1399398810537, value=2013-0
f7b-9c185648d382      3-28T05:03:33.014-0500
37e43447-4d18-3eef-8 column=pme:, timestamp=1401850106587, value=\x060\x87\xB6\
f7b-9c185648d382      x7F\x01\x10\x00
37e43447-4d18-3eef-8 column=pme:594ad504-e9a1-3e34-b1c7-3e00a1a6f090, timestamp
f7b-9c185648d382      =1401850106587, value=
37e43447-4d18-3eef-8 column=pme:6f892880-c444-3ba2-9867-ec128ce6c348, timestamp
f7b-9c185648d382      =1401850106587, value=
37e43447-4d18-3eef-8 column=pme:733953e6-bf13-35b6-a111-9116334e2ad9, timestamp
f7b-9c185648d382      =1401850106587, value=
37e43447-4d18-3eef-8 column=pme:97d0d6d6-1e45-35cf-8a43-3a651596c7ef, timestamp
f7b-9c185648d382      =1401850106587, value=
37e43447-4d18-3eef-8 column=pme:b05355b3-6ebc-34f5-8d95-352697169b42, timestamp
f7b-9c185648d382      =1401850106587, value=
1 row(s) in 0.6640 seconds

```

If you compare these results with those in Example 10-23 on page 193, you will notice a few differences. The `pf:bdate` value is now 2013-05-27. It was previously 2031-05-27. Also, the `pf:sin` value is now masked by the filter to prevent the entire number from showing.

If you run the PME Extract application again with the `View Arguments` parameter value of `maskIdentifiers=false` and perform the **scan** in Example 10-41 again, you will see that Social Security Number is no longer masked.



Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG24-8133-01>

Alternatively, you can go to the IBM Redbooks website at the following website:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG24-8133-01.

Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
lab-data.zip	Compressed sample data
CustomCompositeFilter.zip	Source code and the project skeleton for Eclipse/RAD for Chapter 10

System requirements for downloading the Web material

The Web material requires the following system configuration:

Hard disk space:	250 MB minimum
Operating System:	Windows or Linux
Processor:	64-bit x86_64
Memory:	4 GB minimum

Downloading and extracting the Web material

Create a subdirectory (folder) on your workstation, and extract the contents of the Web material .zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

Online resources

These websites are also relevant as further information sources:

- ▶ IBM Watson Explorer Knowledge Center:
<http://ibm.com/support/knowledgecenter/SS8NLW>
- ▶ IBM Big Data:
<http://www.ibm.com/big-data/us/en/>
- ▶ IBM big data platform:
<http://www.ibm.com/software/data/bigdata/>
- ▶ IBM Master Data Management:
<http://www.ibm.com/software/data/master-data-management/>

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services



Building 360-Degree Information Applications

(0.2" spine)
0.17" <-> 0.473"
90 <-> 249 pages



Building 360-Degree Information Applications



Understand the business value of 360-degree information applications

Understand the architecture of 360-degree applications

Explore sample 360-degree applications

Today's businesses, applications, social media, and online transactions generate more data than ever before. This data can be explored and analyzed to provide tremendous business value. IBM Watson Explorer and IBM InfoSphere Master Data Management (InfoSphere MDM) enable organizations to simultaneously explore and derive insights from enterprise data that was traditionally stored in "silos" in enterprise applications, different data repositories, and in different data formats.

This IBM Redbooks publication provides information about Watson Explorer 9.0, InfoSphere MDM, and IBM InfoSphere MDM Probabilistic Matching Engine for InfoSphere BigInsights (PME for BigInsights). It gives you an overview, describes the architecture, and presents use cases that you can use to accomplish the following tasks:

- ▶ Understand the core capabilities of Watson Explorer, InfoSphere MDM, and PME for BigInsights.
- ▶ Realize the full potential of Watson Explorer applications.
- ▶ Describe the integration and value of the combination of Watson Explorer and InfoSphere MDM.
- ▶ Build a 360-degree information application.
- ▶ Learn by example by following hands-on lab scenarios.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks