



C o m m u n i t y E x p e r i e n c e D i s t i l l e d

Pentaho Analytics for MongoDB

Combine Pentaho Analytics and MongoDB to create powerful
analysis and reporting solutions

Bo Borland

[PACKT] open source*
PUBLISHING community experience distilled

Pentaho Analytics for MongoDB

Combine Pentaho Analytics and MongoDB to create powerful analysis and reporting solutions

Bo Borland



BIRMINGHAM - MUMBAI

Pentaho Analytics for MongoDB

Copyright © 2014 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: February 2014

Production Reference: 1180214

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78216-835-5

www.packtpub.com

Cover Image by Andrey Bayda (<http://shutterstock.com>)

Credits

Author

Bo Borland

Reviewers

Rio Bastian

Pooya Esfandiar

Gretchen Moran

Khaled Tannir

Acquisition Editors

James Jones

Meeta Rajani

Content Development Editor

Govindan K

Technical Editors

Dennis John

Gaurav Thingalaya

Project Coordinator

Aboli Ambardekar

Proofreaders

Simran Bhogal

Maria Gould

Ameesha Green

Indexers

Priya Subramani

Hemangini Bari

Monica Ajmera Mehta

Rekha Nair

Graphics

Abhinash Sahu

Yuvraj Mannari

Production Coordinators

Conidon Miranda

Kyle Albuquerque

Cover Work

Kyle Albuquerque

About the Author

Bo Borland is the vice president of field technical sales at Pentaho, a leading Big Data analytics software provider. He has a passion for building teams and helping companies improve performance with data analytics. Prior to joining Pentaho, Bo worked as a management consultant at Deloitte and as a solution architect at Cognos, an IBM company. He founded a successful analytics consulting company, Management Signals, which merged with Pentaho in 2012. His 14 years' experience in professional analytics includes roles in management, sales, consulting, and sales engineering.

Pentaho Corporation is a leading Big Data analytics company headquartered in Orlando, FL, with offices in San Francisco, London, and Portugal. Pentaho tightly couples data integration with full business analytics capabilities into a single, integrated software platform. The company's subscription-based software offers SMEs and global enterprises the ability to reduce the time taken to design, develop, and deploy Big Data analytics solutions.

I wish to thank the wonderful team at Pentaho for their support and encouragement of this project, especially Rebecca Shomair and Monie TenBroeck for proactively reaching out to offer a helping hand, and Gretchen Moran for her valuable assistance as a technical reviewer for this book.

I also want to thank my wife, Alison, and daughters, Lin and Greta, for graciously accepting the fact that many nights and weekends were spent writing this book.

About the Reviewers

Rio Bastian is a happy software developer already working on several IT projects. His interests include business intelligence, data integration, tuning SQL, and tuning Java code. He has also been a Pentaho Business Intelligence trainer for a server company in Indonesia and Malaysia. He is currently focused on the development of an airline customer loyalty program in PT. Aero Systems Indonesia, an IT consultant specializing in airline industries. In his spare time, he tries to share his experience in developing software through his personal blog altanovela.wordpress.com. You can reach him on Skype via `rio.bastian` or via e-mail at altanovela@gmail.com.

Pooya Esfandiar is a software engineer and data analyst. He received an MS in Computer Science from the University of British Columbia in 2010 while working at the data management and mining labs. His research interests include simulation, data mining, and machine learning in social networks. He has worked with a few companies and organizations in Iran and Canada to solve business problems in information systems, general software engineering, and data analysis. He once designed an in-house analytics solution using Pentaho and MongoDB and is now working as a software engineer at Amazon.com, Inc.

Gretchen Moran is working as an independent Pentaho consultant on a variety of business analytics and Big Data projects. She has 15 years' experience in the business intelligence industry, developing software and providing services for a number of companies, including Hyperion Solutions and Pentaho Corporation.

Gretchen continues to contribute to Pentaho Corporation's latest software initiatives while managing the daily adventures of her two children, Isabella and Jack, with her husband Doug.

Khaled Tannir has been working with computers since 1980. He began programming with the legendary Sinclair Zx81 and afterwards with all Commodore home computer products (VIC-20, Commodore 64, Commodore 128D, and Amiga 500).

He has a Bachelor's degree in Electronics, a Master's degree in System Information Architectures in which he graduated with a professional thesis, and he completed his education with a Master of Research degree.

He is a Microsoft Certified Solution Developer (MCSD) and has more than 20 years' technical experience leading the development and implementation of software solutions and giving technical presentations. He works as an independent IT consultant and has worked as an infrastructure engineer, senior developer, and enterprise/solution architect for many companies in France and Canada.

With a significant experience in Microsoft .NET/Servers and Oracle Java technologies, he has extensive skills in online/offline application design, system conversions, and multilanguage applications on both the Internet and desktops.

He spends his time researching on new technologies, learning about them, and looking for new adventures in France, North America, and the Middle East. He owns an IT and electronics laboratory with many servers, monitors, open electronics boards (such as Arduino, Netduino, Raspberry Pi, and .NET Gadgeteer), and some smartphone devices based on Windows Phone, Android, and iOS operating systems.

In 2012, he contributed to the EGC 2012 (International Complex Data Mining forum at Bordeaux University, France) and presented his work on *how to optimize data distribution in a cloud computing environment* in a workshop session. This work aims to define an approach to optimize the use of data mining algorithms such as k-means and Apriori in a cloud computing environment.

He is the author of *RavenDB 2.x Beginner's Guide*, Packt Publishing and *Optimizing Hadoop MapReduce*, Packt Publishing.

He aims to get a PhD degree in Cloud Computing and Big Data and wants to learn more and more about these technologies.

He enjoys taking landscape and night photos, travelling, playing video games, creating funny electronics gadgets with Arduino / .NET Gadgeteer and of course, spending time with his wife and family. You can reach him at contact@khaledtannir.net.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
Chapter 1: Getting Started with Pentaho and MongoDB	5
MongoDB technology overview	6
Pentaho technology overview	9
Installing MongoDB	10
Installing MongoDB as a Windows service	12
Restoring the sample clickstream MongoDB database	13
Installing Pentaho	14
Summary	15
Chapter 2: MongoDB Database Fundamentals	17
MongoDB database objects	17
Sample clickstream database objects	19
MongoDB data modeling	21
Normalized models	21
Denormalized models	22
MongoDB query methods	24
Query exercise 1	24
Read operations	25
Query exercise 2	26
Query operators	26
Querying arrays	27
Summary	29
Chapter 3: Using Pentaho Instaview	31
Accessing and connecting Instaview to MongoDB	31
Parsing and profiling a MongoDB collection	33
Adding a MongoDB query expression	35
Creating and saving an analysis view and Instaview	38
Summary	42

Chapter 4: Modifying and Enhancing Instaview Transformations	43
Opening an existing Instaview	44
Data integration	45
Adding a new data source	45
CSV file input	47
Stream lookup	48
Creating a new analysis view from blended data	50
Summary	52
Chapter 5: Modifying and Enhancing Instaview Metadata	53
Model design with dimensions and measures	53
Open an existing Instaview	54
Modifying measures and dimensions	55
Session duration measure	56
Session count measure	57
Event count measure	57
Referring URL dimension	57
Other dimension changes	58
Creating a new analysis view	60
Summary	61
Chapter 6: Pentaho Report Designer Fundamentals	63
Pentaho Report Designer features	63
Data sources	64
Report elements	64
Aggregations and calculations	65
Formatting and output	65
Navigating through Pentaho Report Designer	65
Report workspace	66
The Structure tab	68
The Data tab	68
The Style and Attributes tabs	68
The palette	69
The main menu and toolbar	71
The tab toolbar	71
Interface reference	72
Creating a MongoDB connection and query	73
Adding a MongoDB data source	74
Adding and formatting report elements	76
Adding a message field to your report	76
Adding number-fields to your report	78

Adding calculated values to your report	79
Summary	81
Chapter 7: Pentaho Report Designer Prompting and Charting	83
Adding additional MongoDB queries	83
Adding a bar chart query	84
Adding a pie chart query	85
Visualizing your data with charts	86
JFreeChart chart types	87
Subreports	87
Chart data collectors and properties	88
Creating a bar chart	89
Modifying bar chart properties	91
Creating a pie chart	92
Creating a report prompt	95
Creating a new parameter	95
Adding parameters to existing report queries	97
Creating subreport import parameters	98
Summary	100
Chapter 8: Deploying Pentaho Analytics to the Web	101
Publishing a Report Designer report to the Web	102
Publishing the clickstream report	102
An introduction to the Pentaho User Console	104
Running and scheduling the clickstream report	106
Enabling your Instaview output for the Web	108
Copying and modifying the Instaview transformation	109
Using the Data Source Wizard to model your data	112
Creating a JDBC connection and default metadata model	113
Customizing the metadata model	114
Creating Analyzer Views and Dashboard Designer dashboards	118
Creating a map view in Analyzer	118
Creating a heat grid in Analyzer	120
Creating a dashboard using Dashboard Designer	121
Summary	123
Index	125

Preface

MongoDB and Pentaho go together like yin and yang. They are emerging as a powerful combination for scalable data storage, processing, and analytics. Leading companies are pairing these complementary technologies together in development labs and production to deliver innovative analytics. These innovations are creating worldwide demand for developers with skills in both Pentaho and MongoDB.

You want to make an impact by creating innovative data storage capabilities or eye-catching data visualizations. Wouldn't it be great if you could quickly ramp up on both technologies to develop a turn-key solution for your organization? However, as with any new and emerging technology combination, the availability of organized knowledge on the combined topic is scarce.

Pentaho Analytics for MongoDB will show you how to develop an analytic solution that you can demonstrate to your colleagues. It is a practical guide to get you started with both Pentaho and MongoDB, beginning with basic MongoDB data modeling and querying and then advancing to data integration, analysis, and reporting with Pentaho. Each chapter guides you through using different components of the Pentaho platform to create analytic models and reports using a sample MongoDB database.

What this book covers

Chapter 1, Getting Started with Pentaho and MongoDB, introduces you to the powerful combination of MongoDB and Pentaho and provides step-by-step guidance on how to install and configure both technologies and restore the sample MongoDB data provided with this book.

Chapter 2, MongoDB Database Fundamentals, expands on the topic of data modeling and explains MongoDB database concepts essential to querying MongoDB data with Pentaho.

Chapter 3, Using Pentaho Instaview, shows you how to visualize data by connecting Pentaho to MongoDB. You use Instaview with the sample MongoDB database to analyze and visualize the website clickstream data.

Chapter 4, Modifying and Enhancing Instaview Transformations, introduces Pentaho Data Integration (PDI) – the ETL tool used by Instaview to extract, load, and transform data from various data sources.

Chapter 5, Modifying and Enhancing Instaview Metadata, explores metadata by explaining dimensional modeling concepts and how to model metadata to better reflect business requirements.

Chapter 6, Pentaho Report Designer Fundamentals, teaches you the basics of Pentaho Report Designer (PRD) to build pixel-perfect reports sourced directly from MongoDB databases.

Chapter 7, Pentaho Report Designer Prompting and Charting, expands on the previous chapter by teaching you additional advanced PRD features. You can enhance your report with new queries, charts, and a prompt designed to make the report more interactive.

Chapter 8, Deploying Pentaho Analytics to the Web, is all about web-enabling your MongoDB data using Pentaho methods and web interfaces for connecting to, modeling, and analyzing our sample clickstream data in a web browser.

What you need for this book

We need the following software for this book:

- Pentaho Business Analytics v5.0.2 (64-bit for Windows)
- MongoDB v2.2.3 (64-bit for Windows)

This book provides two data sources for use throughout the book, a MongoDB database of sample web clickstream data, and an associated comma-separated (CSV) file containing geographic data. Both files are available as a free download from: <http://www.packtpub.com/support>.

Who this book is for

This book is intended for business analysts, data architects, and developers new to either Pentaho or MongoDB, who want to be able to deliver a complete solution for storing, processing, and visualizing data. It's assumed that you already have experience in defining the data requirements needed to support business processes and exposure to database modeling, SQL query, and reporting techniques.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. The following are some examples of these styles and an explanation of their meaning.

Code words in text are shown as follows: `"$.event_data[0].event."`


When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in **bold**:


```
{ $match : {referring_url : "${ReferringURLParam}" } },
{ $unwind : "$event_data" },
{ $group : { _id : "$browser", event_count : { $sum : 1 } } },
{$sort:{event_count: -1}}
```

Any command-line input or output is written as follows:

```
cd \
move C:\mongodb-win32-* C:\mongodb
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Select and drag the **CSV file input** step onto the canvas."

 Warnings or important notes appear in a box like this.

 Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title through the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books — maybe a mistake in the text or the code — we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website, or added to any list of existing errata, under the Errata section of that title.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Getting Started with Pentaho and MongoDB

Many people have chosen MongoDB – a leading NoSQL database – as their data storage solution over a relational database approach, and now they need to analyze and report on their data residing in MongoDB. Pentaho, the leading commercial open source analytics platform, decided to meet this need by forming an early partnership with MongoDB, enabling analytics and reporting on the document-oriented data structures inside MongoDB.

This chapter will introduce you to the powerful combination of MongoDB and Pentaho and will provide step-by-step guidance on how to install and configure both technologies as well as restore the sample MongoDB data provided with this book. We start with an overview of the technologies and then set up a local installation of both technologies along with sample data on a single Windows machine.

The following are the topics that we will cover in this chapter:

- Overview of Pentaho and MongoDB
- Installing MongoDB
- Restoring the sample MongoDB database
- Installing Pentaho Enterprise Edition

We will focus on the Pentaho Enterprise Edition for this book, because it includes the Big Data utility, Instaview, which we use in later chapters. By the end of this chapter, you will have Pentaho and MongoDB running on a Windows computer with the sample MongoDB database provided with this book and restored to your MongoDB server.

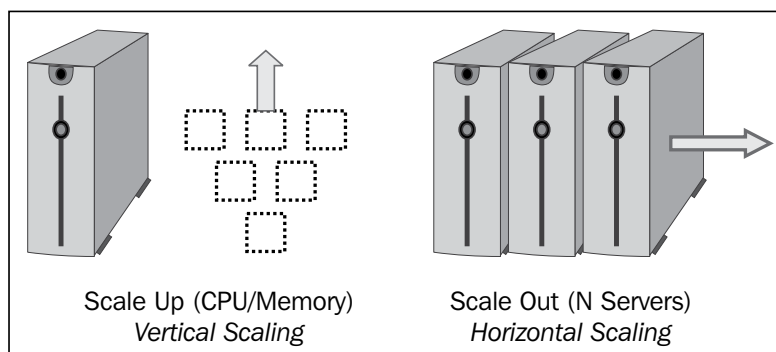
MongoDB technology overview

Modern businesses capture huge volumes and varieties of data using several different data storage methods. There is no one-size-fits-all data storage method, because each technology has evolved to tackle the data challenges or opportunities of that specific time in history. We continue to see new and innovative data storage solutions as data volume, variety, and velocity grows, and as people figure out new ways to use data. The following is a small sampling of the variety of data sources you might encounter in a single organization:

- **Simple tabular data files:** CSV, text files, and MS Excel
- **Commercial relational databases:** Oracle, SQL Server, and DB2
- **Open source relational databases:** MySQL and PostgreSQL
- **Modern, web-oriented data sources:** XML, JSON, web services, and APIs
- **Hadoop distributions:** Apache, Hortonworks, Cloudera, MapR, and Intel
- **Analytical databases:** Vertica, Greenplum, and InfoBright
- **Machine generated data sources:** Application logs, web server logs, configuration files, sensor data, message queues, and filesystem audit logs
- **NoSQL databases:** MongoDB, Redis, Cassandra, HBase, and CouchDB

Organizations invest heavily in these storage technologies and the skills needed to capture, store, and process data. MongoDB has emerged as a leader in the **NoSQL** category of databases. Because you are reading this book, you are probably well aware of the differences between MongoDB and relational databases; however, it is important to review and remind ourselves where MongoDB came from and why it is popular alternative to relational databases.

MongoDB is a **document-oriented database** designed to conquer some of the modern data storage challenges that developers and IT departments experience when using traditional relational databases. These modern data storage challenges started with the rise of the internet and high traffic websites such as Amazon and Google. These companies' websites attracted millions of users and subsequently massive volumes of website log, clickstream, and event data. Traditional relational database methods for handling growing data mostly involved **scaling up** (that is, vertical scaling) by adding more CPU and RAM to a single, often proprietary database server. This method of scaling was expensive and had limits on how far you could scale a single server. As a result, Google and Amazon decided to solve these data challenges by developing their own distributed data stores that could easily **scale out** (that is, horizontal scaling) across hundreds or thousands of commodity servers, as shown in the following figure. Horizontal scaling made it easier to scale dynamically by adding more machines to the cluster without any downtime or limits to compute capacity.



These early pioneers of distributed databases inspired a NoSQL data storage movement that included MongoDB. The term NoSQL is a popular way to describe MongoDB, because MongoDB does not use SQL, but NoSQL is not just about the query language. It has more to do with the way data is stored than just the query language. For many, the name NoSQL is inadequate, because it simply describes a query language and not the true essence of MongoDB, which is a horizontally scalable, distributed document database. Surprisingly, the name NoSQL originated simply as a way to describe these emerging distributed data stores using a short and unique Twitter hashtag, #nosql, for the purpose of advertising a meet up on the topic. As the story continued, the hashtag name stuck and is widely in use today!

Why would a database solution not leverage SQL, the most popular database query language used by developers and organizations all over the world? One key reason is that the SQL query language is not designed to efficiently query the nested constructs involved in hierarchical JSON documents, which form the foundation of the MongoDB document data model. JSON documents are language independent text files that represent data and are built on two primary data structures: nested collections of name/value pairs and ordered lists such as arrays. The following example shows the JSON representation of a dataset that describes the movie *Forrest Gump*. The JSON representation contains a parent object for movie information, a nested object representing the production company, and an array of cast member objects, shown as follows:

```
{
  "movie": "Forrest Gump",
  "rating": "PG-13",
  "duration_min": 142,
  "production_company": {
    "name": "Paramount Pictures",
    "streetAddress": "5555 Melrose Ave",
    "city": "Los Angeles",
    "state": "CA",

```



```
        "postalCode": 90038
      },
      "cast": [
        {
          "character": "Forrest Gump",
          "person": "Tom Hanks"
        },
        {
          "character": "Jenny Curran",
          "person": "Robin Wright"
        }
      ]
    }
  ]
}
```

Each object is a comma-separated collection of key-value pairs enclosed in curly braces. MongoDB has its own query language built from the ground as a powerful way to retrieve, process, and update JSON documents. Document-oriented data storage is an alternative to SQL-based relational databases, and it offers some unique advantages that we will discuss in more detail in the next chapter. You can also learn more about JSON at json.org or Wikipedia.org/wiki/JSON.

MongoDB's use of JSON pairs each key with a complex data structure known as a document, and these documents can contain many different key-value pairs, key-array pairs, or even nested documents. MongoDB document-oriented data models enable the following benefits over relational databases:

- It provides speedy and easy horizontal scaling by auto-sharding and grouping related data together in document collections instead of separate database tables that require joins to pull the data back together. The multitable joins of **relational database management systems (RDBMS)** reduce performance and make horizontal scaling more difficult.
- It provides faster and easier application development by providing a data model that maps to native programming language objects. JSON's universal data structures make this possible and are supported by virtually any modern programming language. This makes MongoDB popular among developers because it permits a one-to-one mapping between object-oriented software objects and database entities. It also makes data interchange between software and databases easier.
- It provides a dynamic schema that makes it easier than enforced RDBMS schemas to manage and evolve your data model. MongoDB allows for insertion of data without a predefined database schema. This gives software developers more flexibility to define and manipulate the database schema instead of relying on a separate database administrator to maintain schema changes.

Pentaho technology overview

Pentaho software consists of a suite of analytics products called **Pentaho Business Analytics**, providing a complete analytics software platform. This end-to-end solution includes data integration, metadata, reporting, OLAP analysis, ad-hoc query, dashboards, and data mining capabilities. The platform is available in two offerings: a **community edition (CE)** and an **enterprise edition (EE)**. We will focus on the enterprise edition for this book because Instaview is included only in this edition of Pentaho.

Throughout the book, you will see that we group and refer to the platform in two major categories, **Pentaho Data Integration (PDI)** and **Pentaho Business Analytics (BA)**. Even though we refer to PDI and BA as separate server categories, Pentaho tightly couples the PDI server with the BA server into a single platform offering. This unique approach helps companies solve their data integration challenges with multiple, diverse data sources including Big Data sources and instantly gain insight into business analytics for a broad set of users.

PDI gives users a graphical user interface to a parallel processing ETL engine to solve data integration challenges. The user interface reduces data integration complexity by eliminating the need to code data extractions, data transformations, and data loads. Some additional PDI benefits include:

- Broad connectivity to any type of data source including native support for Big Data sources such as Hadoop, NoSQL, and analytic databases
- Integrated self-service Big Data analytics with Instaview – a utility that simplifies Big Data connectivity and bundles the Pentaho OLAP interface into PDI
- An open, pluggable Java architecture that makes it easy to develop plugins to extend the platform
- A parallel processing engine that can be dynamically scaled across multiple servers in a cluster

BA provides web-based interfaces to create business models and interactive reports as well as analysis views and dashboards. The focus is on ease-of-use while providing a complete set of reporting and analysis capabilities that include the following web-based components:

- **Interactive Reporting:** Relational ad hoc queries and basic tabular, parameterized reporting
- **Analyzer:** OLAP interface for analysis and visualization
- **Dashboard Designer:** Easy-to-use, interactive dashboard creation

It also includes the following client-based components:

- **Metadata Editor:** Developer interface for metadata modeling
- **Schema Workbench:** Developer interface to model OLAP cubes
- **Report Designer:** Advanced report development to build any type of parameterized report

As mentioned earlier, Pentaho is an early mover into the Big Data space as the first major BI vendor to extend its analytics platform with Big Data capabilities in May 2010. The partnership with MongoDB is one of the first few Big Data partnerships for Pentaho and since then, Pentaho continues to deliver MongoDB innovations. The Pentaho-MongoDB solution covers the entire Big Data life cycle from data extraction and preparation to data discovery, which we will explore throughout this book. Now that we have reviewed both technologies, it is time to install them on your computer.

Installing MongoDB

Installing MongoDB on Windows or Linux is quick, easy, and well-documented in the MongoDB manual at <http://docs.mongodb.org/manual/>. We will install it on one Windows computer and restore the provided sample database for development purposes only. The database does not enable any authentication by default, so before you load any other personal or confidential data into MongoDB or consider it for your production environment, you will want to read about recommended security practices at <http://docs.mongodb.org/manual/core/security>.



MongoDB has not supported Windows XP since the release of version 2.2, so you will need a more recent version of Windows available for this install. Also, while we will not be using large sample MongoDB datasets for this book, it is recommended that you download the 64-bit version if your machine is 64-bit compliant, because the 32-bit version is limited to 2 GB of data.

The first thing we need to do is download the latest production release of the software. Included in the download are the MongoDB server, MongoDB shell, backup and restoration tools, import and export tools, and GridFS to manage large files that exceed the document size limit of 16 MB. MongoDB is self-contained without other system dependencies, so you can install and run MongoDB from any folder you want. Perform the following steps to set up and run the MongoDB server via the command prompt and set up MongoDB as a Windows service:

1. Download the Windows 64-bit production release from www.mongodb.org/downloads.
2. Extract the downloaded archive to `C:\`.



Your folder should look like the following with [version] being the version you downloaded:
`C:\mongodb-win32-x86_64-[version]`.

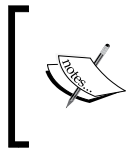
3. Open a new command prompt window using administrator rights by right-clicking on the command prompt program and selecting **Run as Administrator**.
4. Within the command prompt, issue the following commands:
5. Create the required data folder using the command prompt for MongoDB to store its files:

```
cd \
move C:\mongodb-win32-* C:\mongodb
```

```
md data
md data\db
```

6. Start the MongoDB database process from the command prompt:

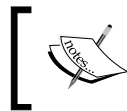
```
C:\mongodb\bin\mongod.exe
```



The `mongod.exe` console window output will display **waiting for connections** if `mongod.exe` is running correctly. MongoDB also provides an HTTP interface for administrators, which can be accessed in a browser at `http://localhost:28017`.

7. Establish a connection to MongoDB using the `mongo.exe` shell, which will connect to the default test database on localhost using the default port of 27017:

```
C:\mongodb\bin\mongo.exe
```



The `mongo.exe` console window output will display **MongoDB shell version:[current version] connected to: test** if `mongo.exe` is running correctly.

8. Insert a document in the test collection of the test database, and then retrieve it using the `find` command:

```
db.test.save( {abc: 1 } )  
db.test.find ( )
```



The `mongo.exe` console window will display the document you just created because it is the only document that exists in the collection, and you did not specify any matching criteria with your `find` command:

```
{ "_id" : ObjectId("51daa85f987a76f4380728ea"), "abc" : 1 }
```

The `_id` field is the unique identifier for a collection that serves as the primary key field. If you don't specify the `_id` field values, MongoDB uses system-generated `ObjectIds` as the default value. Your `ObjectId` will differ from the one shown previously.

Installing MongoDB as a Windows service

Set up MongoDB as a Windows service to start automatically on reboots:

1. Open a new command prompt window using administrator rights by right-clicking on the command prompt program and selecting **Run as Administrator**.
2. Create a folder for the database logfiles and a configuration file for logging by entering the following commands:

```
cd C:\mongodb\log  
echo logpath = C:\mongodb\log\mongo.log > C:\mongodb\mongodb.cfg
```

3. Install the MongoDB service:

```
C:\mongodb\bin\mongod.exe --config C:\mongodb\mongodb.cfg --install
```

4. Start the MongoDB service:

```
net start MongoDB
```



Use the following command to stop the MongoDB service:

```
net stop MongoDB
```

Also, use the following command to remove the MongoDB service:

```
C:\mongodb\bin\mongod.exe --remove
```

If you prefer to use a GUI interface to interact with your MongoDB instance, MongoVUE is one of many third-party options. MongoVUE is a MongoDB desktop application for Windows OS that makes it easier to manage and query your MongoDB databases. You can download a free limited version at <http://www.mongovue.com/>.

Restoring the sample clickstream MongoDB database

The sample clickstream database used throughout this book contains web session information along with user clickstream events that occur within that session. The session is the header record that captures information about a web user's IP address, browser, referring URL, session date, and session length. The following is a sample JSON text output of a single session record containing the seven fields available in the collection:

```
{
  "_id" : ObjectId("512d200e223e7f294d13d44c"),
  "id_session" : "71E1FF1A25B84045864FCEC447F8D012",
  "ip_address" : "47.54.245.196",
  "browser" : "Safari",
  "date_session" : ISODate("2013-01-01T17:36:32Z"),
  "duration_session" : 7.43,
  "referring_url" : "www.123.com"
}
```

One or more clickstream events can occur within a single session. These events include visited site, signup newsletter, watched video, completed lead form, and added item to cart. The event records link to the session records by the unique identifier for the session called `[id_session]`. The following is an example of the JSON text output of a single event record with the `id_session` field highlighted in bold text:

```
{
  "_id" : ObjectId("512d1ff2223e7f294d13c0c4"),
  "id_session" : "310FA578571D4D90847E33E8B894703D",
  "event" : "Visited Site"
}
```

Now, let's restore the sample clickstream MongoDB database provided with this book. You will need to download the database file and perform the following steps to restore the database from command line using the `mongorestore` program:

1. Download both the Pentaho database ZIP and `zip_codes.csv` files from <http://www.packtpub.com/support>.
2. Extract the downloaded ZIP file to `C:\mongodb\bin\dump`.



Your database directory should look like the following:
`C:\mongodb\bin\dump\pentaho`

3. Open a new Windows command prompt and issue the following commands:

```
C:\mongodb\bin\mongorestore /mongodb/bin/dump/pentaho
```

Installing Pentaho

Installing Pentaho on Windows, Linux, or Mac OS is quick and easy with the graphical installer option. This option, by default, installs the entire BA platform, PostgreSQLSolution database, Tomcat application server, and Sun JRE needed to provide a completely functioning analytics server. Typically, you would install the server components on a server and the client components on your workstation, but for development and training purposes, we are going to install all of the server and client components on a single Windows computer. The following list shows all of the platform components that will be installed on your computer:

- **Client tools:** Aggregation Designer, Design Studio, Metadata Editor, Pentaho Data Integration, Report Designer, and Schema Workbench
- **Web tools:** Analyzer, Dashboard Designer, Interactive Reporting, Enterprise Console, and Mobile
- **BA server:** Enterprise Console and User Console
- **DI server:** Data Integration Server
- **Others:** Tomcat App Server, PostgreSQL Solution DB, and Sun JRE

Perform the following steps to download and install Pentaho on your Windows computer:

1. Check to make sure your computer meets the minimum hardware requirements to run Pentaho server components at http://infocenter.pentaho.com/help/topic/supported_components/reference_supported_components.html.
2. If your computer is 64-bit compliant, download Pentaho Business Analytics 64-bit Windows version to your computer from <http://www.pentaho.com/download>.



You will be asked to register your contact information for a 30-day evaluation of the Pentaho Enterprise Edition. Your download will begin after registration. The downloaded filename should look like the following with [version] being the version you downloaded:
pentaho-business-analytics-[version]-x64.exe

3. Double-click on the downloaded executable file, `pentaho-business-analytics-[version]-x64.exe`, to run the installer program, which will launch a Pentaho splash screen.
4. When prompted to choose a setup type, select **Default**, which will install the entire suite of server and client components to a default directory of `C:\Program Files\Pentaho`.
5. You will be prompted to enter a single master password for the PostgreSQL solution repository and enterprise console, and publish it to the web feature. Enter your chosen password and click on **Next**.



When the installation wizard completes, the BA and DI servers should be running on the following default ports:

- 5432: PostgreSQL server
- 8080: BA server Tomcat web server startup port
- 9080: DI server port

If these ports are not available during the installation, Pentaho will automatically increment the port number by one until an available port is found.

6. Upon wizard completion, select the option of starting the **Pentaho User Console**, which will launch the PUC login screen.
7. Click on the **Login as an Evaluator** link, which will use the default administrator login with `admin` as the username and `password` as the password, and then click on the **Go** button to access Pentaho.

Summary

Congratulations! You should have a working installation of Pentaho, which installs the BA and DI servers as Windows systems services that will start automatically the next time you boot your computer. Please refer to the official **Pentaho Installation Guide** at <http://infocenter.pentaho.com> for detailed installation information. In the next chapter, we will explore the MongoDB data model and establish a connection from Pentaho to the sample MongoDB database.

2

MongoDB Database Fundamentals

The previous chapter introduced the document-oriented storage of MongoDB, and we learned that MongoDB JSON-style documents are built on two primary data structures, a nested collection of documents and arrays. This chapter elaborates on this topic and explains the MongoDB database concepts essential for learning how to query MongoDB data with Pentaho. We will start by discussing MongoDB database objects and design methods and then learn how to query the restored sample clickstream database using the mongo shell.

The following are the topics that we will cover in this chapter:

- MongoDB database objects
- Sample MongoDB clickstream database
- MongoDB data modeling
- MongoDB query methods

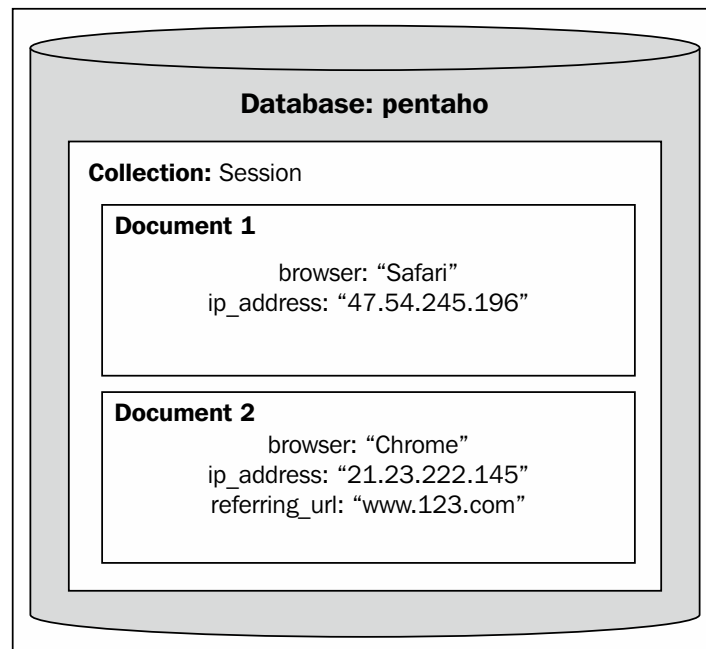
By the end of this chapter, you will have a foundational knowledge of MongoDB design and the primary query methods needed to start retrieving data from the sample database.

MongoDB database objects

We will begin our journey of understanding MongoDB database objects by comparing each object to its equivalent RDBMS object when applicable. There are some similarities between MongoDB and RDBMS objects, although the object names are different. MongoDB object names include databases, collections, documents, fields, and indexes.

The highest level container in a MongoDB server instance is a database. A **database** consists of one or more collections. A **collection** holds a grouping of documents just as a RDBMS table holds a grouping of records. A **document** is made up of one or more fields, which as you probably guessed, are similar to table fields (that is, columns). Relational databases define fields at the table level while document-oriented databases define fields at the document level. Additionally, as with a RDBMS, MongoDB allows you to create indexes on fields for better sorting and lookup performance.

The following figure shows a basic object relationship in a MongoDB database with one collection containing two documents and a total of three fields. The mandatory ObjectId fields, the unique keys for a collection, are left out intentionally to keep the illustration simple.



Typically, all documents in a collection will be similar and serve a related purpose. However, you will notice in the previous figure that **Document 2** has a third field, [referring_url], which does not exist in the **Document 1**. This highlights a major difference between MongoDB document collections and RDBMS tables that documents within the same collection can have different fields. In other words, MongoDB collections do not enforce a schema.

People often refer to this as having a **dynamic schema** or **schema-less** database. A dynamic schema gives developers the ability to define and manipulate the database on the fly by inserting new fields, such as the `[referring_url]` field, which do not already exist in other documents of the same collection. This level of schema flexibility allows developers to model documents to closely resemble software application objects. In reality, you will find most documents within a single collection sharing a common structure, thus making downstream reporting much easier.

Sample clickstream database objects

The sample `pentaho` database consists of a fictional data model and data used to capture all of the web events that occur when users interact with a web application. These online user interactions, often referred to as clickstream data, give marketers insight into website user behavior. The sample database contains three collections for capturing clickstream events by session.

The definition of a session can vary across companies and web applications; however, it will always have a defined start point and end point. We don't need to worry about how a session is defined. Instead, just know that a unique web session is identified in the database by the unique identifier field `id_session`, and all events that occur within a single session will all have the same `id_session` value.

The following table provides a list of objects that form the fundamental building blocks of a MongoDB database and the associated object names in the `pentaho` database.

MongoDB objects	Sample MongoDB object names
Database	<code>pentaho</code>
Collection	<code>sessions</code> , <code>events</code> , and <code>sessions_events</code>

MongoDB objects	Sample MongoDB object names
Document	<p>The sessions document fields (key-value pairs) are as follows:</p> <ul style="list-style-type: none"> • <code>_id: 512d200e223e7f294d13607</code> • <code>ip_address: 235.149.145.57</code> • <code>id_session: DF636FB593684905B335982BEA3D967B</code> • <code>browser: Chrome</code> • <code>date_session: 2013-02-20T15:47:49Z</code> • <code>duration_session: 18.24</code> • <code>referring_url: www.xyz.com</code> <p>The events document fields (key-value pairs) are as follows:</p> <ul style="list-style-type: none"> • <code>_id: 512d1ff2223e7f294d13c0c6</code> • <code>id_session: DF636FB593684905B335982BEA3D967B</code> • <code>event: Signup Free Offer</code> <p>The sessions_events document fields (key-value pairs) are as follows:</p> <ul style="list-style-type: none"> • <code>_id: 512d200e223e7f294d13607</code> • <code>ip_address: 235.149.145.57</code> • <code>id_session: DF636FB593684905B335982BEA3D967B</code> • <code>browser: Chrome</code> • <code>date_session: 2013-02-20T15:47:49Z</code> • <code>duration_session: 18.24</code> • <code>referring_url: www.xyz.com</code> • <code>event_data: [array]</code> <ul style="list-style-type: none"> • <code>event: Visited Site</code> • <code>event: Signup Free Offer</code>

MongoDB data modeling

MongoDB data modeling is an advanced topic that would consume multiple chapters in this book if covered in depth. We will instead focus on a few key points regarding data modeling so that you will understand the implications for your report queries. These points touch on the strengths and weaknesses of embedded data models.

The most common data modeling decision in MongoDB is whether to normalize or denormalize related collections of data. Denormalized MongoDB models embed related data together in the same database documents while normalized models represent the same records with references between documents. This modeling decision will impact the method used to extract and query data using Pentaho, because the normalized method requires joining documents outside of MongoDB.



With normalized models, Pentaho Data Integration is used to resolve the reference joins.

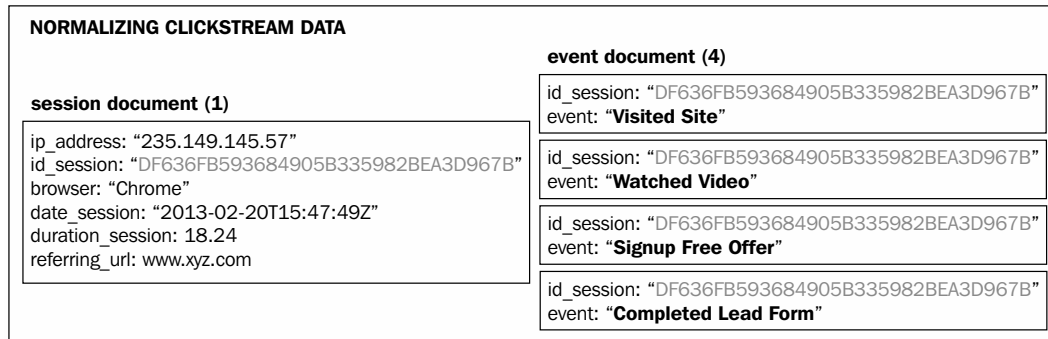


The table in the previous section shows three collections in the `pentaho` database: `sessions`, `events`, and `sessions_events`. The first two collections, `sessions` and `events`, illustrate the concept of normalizing the clickstream data by separating it into two related collections with a reference key field in common. In addition to the two normalized collections, the `sessions_events` collection is included to illustrate the concept of denormalizing the clickstream data by combining the data into a single collection.

Normalized models

Because multiple clickstream events can occur within a single web session, we know that sessions have a one-to-many relationship with events. For example, during a single 20-minute web session, a user could invoke four events by visiting the website, watching a video, signing up for a free offer, and completing a sales lead form. These four events would always appear within the context of a single session and would share the same `id_session` reference.

The data resulting from the normalized model would include one new session document in the `sessions` collection and four new event documents in the `events` collection, as shown in the following figure:



Each event document is linked to the parent session document by the shared `id_session` reference field, whose values are highlighted in red.

This normalized model would be an efficient data model if we expect the number of events per session to be very large for a couple of reasons. The first reason is that the current version of MongoDB limits the maximum document size to 16 megabytes, so you will want to avoid data models that create extremely large documents. The second reason is that query performance can be negatively impacted by large data arrays that contain thousands of event values. This is not a concern for the clickstream dataset, because the number of events per session is small.

Denormalized models

The one-to-many relationship between sessions and events also gives us the option of embedding multiple events inside of a single session document. Embedding is accomplished by declaring a field to hold either an array of values or embedded documents known as **subdocuments**. The `sessions_events` collection is an example of embedding, because it embeds the event data into an array within a session document. The data resulting from our denormalized model includes four event values in the `event_data` array within the `sessions_events` collection as shown in the following figure:

DE-NORMALIZING CLICKSTREAM DATA**sessions_events document (1) - with event data array**

```
ip_address: "235.149.145.57"
id_session: "DF636FB593684905B335982BEA3D967B"
browser: "Chrome"
date_session: "2013-02-20T15:47:49Z"
duration_session: 18.24
referring_url: www.xyz.com
event_data {4}
---event: "Visited Site"
---event: "Watched Video"
---event: "Signup Free Offer"
---event: "Completed Lead Form"
```

As you can see, we have the choice to keep the session and event data in separate collections, or alternatively, store both datasets inside a single collection. One important rule to keep in mind when you consider the two approaches is that the MongoDB query language does not support joins between collections. This rule makes embedded documents or data arrays better for querying, because the embedded relationship allows us to avoid expensive client-side joins between collections. In addition, the MongoDB query language supports a large number of powerful query operators for accessing documents by the contents of an array. A list of query operators can be found on the MongoDB documentation site at <http://docs.mongodb.org/manual/reference/operator/>.

To summarize, the following are a few key points to consider when deciding on a normalized or denormalized data model in MongoDB:

- The MongoDB query language does not support joins between collections
- Currently, the maximum document size is 16 megabytes
- Very large data arrays can negatively impact query performance

In our sample database, the number of clickstream events per session is expected to be small – within a modest range of only one to 20 per session. The denormalized model works well in this scenario, because it eliminates joins by keeping events and sessions in a single collection. However, both data modeling scenarios are provided in the *pentaho* MongoDB database to highlight the importance of having an analytics platform, such as Pentaho, to handle both normalized and denormalized data models.

MongoDB query methods

The MongoDB query language is built from the ground up as a powerful way to retrieve, process, and update documents. In this section, we are going to use the **mongo shell** interface to learn fundamental MongoDB query structures and practice querying documents from the `pentaho` database. We will build on this query knowledge in the next chapter by replacing the mongo shell with Pentaho for retrieving data from MongoDB.



You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

Query exercise 1

Let's connect to the mongo shell and execute some often used help commands:

1. Establish a connection to MongoDB using the `mongo.exe` shell:
`C:\mongodb\bin\mongo.exe`
2. Show the available databases:
`show dbs`
3. Show a list of general help methods:
`help`
4. Show help for database methods:
`db.help()`
5. Switch to the sample database, `pentaho`, and show the available collections:
`use pentaho`
`show collections`



The console window output will display three collections in the `pentaho` database: `events`, `sessions`, and `session_events`.

6. Count the number of documents in the `sessions` collection:
`db.runCommand({ count: 'sessions' })`



The console window output will display, { "n":1000, "ok":1 }, indicating a count of 1000 documents.

7. Show help for `sessions` collection methods:

```
db.sessions.help()
```



The console window output will show help for the `session` collection including the following `find()` methods which are often used to retrieve data:

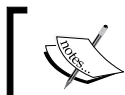
- `db.sessions.find(...).count()`
- `db.sessions.find(...).limit(n)`
- `db.sessions.find(...).sort(...)`

Read operations

The MongoDB query language supports the four CRUD database operations. CRUD is an acronym for the following database operations: create, read, update, and delete. This book focuses only on read operations, which are used to retrieve documents from MongoDB for reporting and analysis. Read operations include all the operations needed to query MongoDB, and the `find()` method is the primary method used to retrieve MongoDB documents. The `find()` method is similar to the `select` statement in the SQL query language. MongoDB queries accept objects and arguments, using the following syntax for the `find()` method:

```
db.collection.find( <query>, <projection> )
```

You must first specify a database that you will be working with. Once you have selected a database, you can define a collection within your query using the `db.collection` object followed by an optional **query argument** and **projection argument**.



The MongoDB query language supports the querying of only one collection per query. We will learn in later chapters how to use Pentaho Data Integration to get around this limitation.

The `<query>` argument specifies the criteria for which documents to retrieve within the collection. If you don't specify a query argument, all of the documents in the collection will be returned. The `<projection>` argument specifies what fields to return in your query results; likewise, if you don't specify a projection, all fields in the collection will be returned.

Query exercise 2

In this exercise, we give MongoDB queries a try using the pentaho database connection established in the previous exercise by performing the following steps:

1. Query the `sessions` collection and retrieve all of the documents and fields in the collection:

```
db.sessions.find ()
```



Using the `find()` method without a query argument will return all of the documents in the collection. The console window output for this query will display all of the available fields in the first 20 documents.

2. Query the `sessions` collection to retrieve only the web sessions with a Chrome browser by adding equality criteria to the query argument:

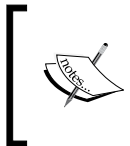
```
db.sessions.find ( {browser: 'Chrome'} )
```



The console window output will display all of the available fields in the first 20 documents that match the criteria of having a Chrome browser.

3. Restrict the fields returned by the query to only the `id_session` and `browser` fields by adding a projection argument:

```
db.sessions.find ( {browser: 'Chrome'}, {id_session: 1, browser: 1} )
```



After the projection field name, you specify either 1 to include a field or a 0 to exclude a field. The console window outputs the `id_session`, `browser`, and the system default, `_id`, fields in the first 20 documents that have a Chrome browser.

Query operators

The `find()` method is a container for a more powerful way to create queries using the `$query` operator. The `$query` operator can be used to define document queries by comparison, logic, data type, array size, and others. For example, if we wanted to query the `sessions` collection for sessions that have a Chrome browser and session duration in minutes greater than or equal eighteen minutes, we would add the `$gte` comparison operator as follows:

```
db.sessions.find ( { browser: 'Chrome', duration_session: { $gte: 18 } } )
```

By default, the two fields used to specify the filter criteria are joined with the `and` conjunction. If we want to query for sessions that have a Chrome browser or session duration greater than or equal eighteen minutes, we need to add the `$or` and `$gte` query operators. The `$or` operator evaluates two expressions and selects the documents that satisfy at least one of the expressions as follows.

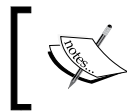
```
db.sessions.find ( $or: [ { browser: 'Chrome'}, {duration_session: {$gte: 18}}] } )
```

The examples of other comparison operators include `$lt` for less than, `$lte` for less than or equal, `$gt` for greater than, and `$ne` for not equal.

Querying arrays

Data arrays contain a set of values that are often used to filter MongoDB queries. When specifying equality criteria for multiple values in an array, MongoDB will look for an exact match of the array values and the order of those values. First, let's query for a single event in the array. You will recall that the `sessions_events` collection holds the `event_data` array containing clickstream events. To find all of the sessions that contain the Added Item To Cart event, we issue the following query against the `event_data` array:

```
db.sessions_events.find ( {"event_data.event": 'Added Item To Cart'})
```



The console window will display the first 20 documents that contain this event along with any other events that occurred during the session.

The session with `id_session = E8995C988FD3441AA3077DE435AFC3EC` contains the following event sequence: Visited Site, Watched Video, Signup Newsletter, Added Item To Cart, and Completed Lead Form. In the next query, we will look for two events, Visited Site and Watched Video, and see how MongoDB handles equality criteria for multiple events. To find sessions where users visited the website and then watched a video in that exact order, we simply add both events to the query as follows:

```
db.sessions_events.find ( {"event_data.event": [ 'Visited Site', 'Watched Video' ] } )
```



The console window will display only five documents that match the exact criteria of someone who visited the website and then watched a video. It does not include the session we identified with these two events, because the default filter on multiple events in an array is to look for an exact match.

So, filtering on a single value uses a `contains` equality match, while filtering on multiple values restricts the query to only exact matches. This is too restrictive for scenarios when you would like find documents that contain these two events along with other possible events that occurred in the same session. The MongoDB query language includes the `$all` query operator to expand the equality criteria and retrieve sessions that contain both of the specified events, shown as follows:

```
db.sessions_events.find ( {"event_data_event": {$all: [ 'Visited Site',  
'Watched Video' ] } } )
```



The console window will display many more documents that match the criteria of someone visiting the website and watching a video in addition to performing any other sequence of events in a web session.

Query exercise 3

In this exercise, you will query MongoDB to find web sessions where a user watched a video, signed up for a free offer, and added an item to their shopping cart. For this exercise, assume we don't care about the order of the events or occurrence of other events in the same session. The steps for this are as follows:

1. Establish a connection to MongoDB using the `mongo.exe` shell:
`C:\mongodb\bin\mongo.exe`
2. Switch to the `pentaho` database:
`use pentaho`
3. Query the `sessions_events` collection, and retrieve all of the documents and fields in the collection:

```
db.sessions_events.find ()
```



The console window output will display all of the available fields in the first 20 documents.

4. Query the `sessions_events` collection to retrieve web sessions that contain the following three events: Added Item To Cart, Signup Free Offer, and Watched Video:

```
db.sessions_events.find ( {"event_data.event": {$all: ['Added Item  
To Cart', 'Signup Free Offer', 'Watched Video'] } } )
```



The console window output will display all of the available fields in the first 20 documents that match your criteria.

- Count the number of web sessions that contain the following three events: Added Item To Cart, Signup Free Offer, and Watched Video:

```
db.runCommand ( {count: 'sessions_events',  
query: {"event_data.event": {$all: ['Added Item To Cart', 'Signup  
Free Offer', 'Watched Video']} } })
```



The console window output will display a count of 37 documents that match your criteria.

Summary

Congratulations! Now, you know how to query the clickstream database using the mongo shell! You can add any combination of events to your array query and gain a powerful insight into the sequences of events that occur with online user behavior. The next chapter expands on this query knowledge and introduces you to the Pentaho interface to query MongoDB.

3

Using Pentaho Instaview

In the previous chapter, we reviewed the MongoDB clickstream database and how we query it using the mongo shell. Now, we begin to gain insight into this data by connecting Pentaho to MongoDB. This chapter focuses on **Instaview**, a component of Pentaho Data Integration, designed to instantly parse, model, and analyze data from Big Data technologies such as MongoDB, Hadoop, and Cassandra. We will use Instaview with the sample MongoDB database to analyze and visualize the website clickstream data.

The following topics are covered in this chapter:

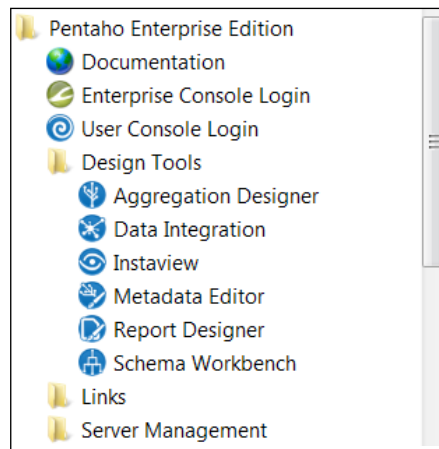
- Accessing and connecting Instaview to MongoDB
- Parsing and profiling a MongoDB collection
- Adding a MongoDB query expression
- Creating and saving an analysis view and Instaview

By the end of this chapter, you will have Instaview connected to the clickstream data for visualizing web session duration by browser and referring URL.

Accessing and connecting Instaview to MongoDB

In *Chapter 1, Getting Started with Pentaho and MongoDB*, you installed both the Pentaho server and client components on a single Windows computer. Instaview is one of those client components; it is exposed as a perspective through PDI. There are a couple of ways to access Instaview, either directly from your Windows Start menu or from the Instaview perspective inside PDI.


You will notice both Instaview and Data Integration in the following Start menu screenshot:



Both links launch PDI; however, the **Instaview** link launches you directly into the Instaview perspective of PDI. Alternatively, you can launch Instaview or PDI from their respective Windows batch files, *Instaview.bat* and *Spoon.bat*, located at {Pentaho Install Directory}/design-tools/data-integration.

The Instaview perspective allows PDI developers to quickly and easily connect to a single MongoDB collection and parse the document hierarchy into a set of fields for analysis. You can then analyze the data with Instaview without ever leaving PDI. After you define the connection, detect the fields, and refine the query, Instaview will launch into an automated, three-step process consisting of the following tasks:

1. Auto-build a PDI transformation to extract data from MongoDB, and load a sample dataset into an in-memory database.

 PDI transformations are usually manually created in PDI to define the extraction, transformation, and loading of data. Instaview automates the creation of these transformation files, thus reducing the time to insight and the complexity for developers.

2. Auto-build a cube model consisting of dimensions and measures.
3. Execute the PDI transformation to extract and load the data into an in-memory database for analysis.

After completing the preceding steps, Instaview will launch Analyzer to analyze the data.

Parsing and profiling a MongoDB collection

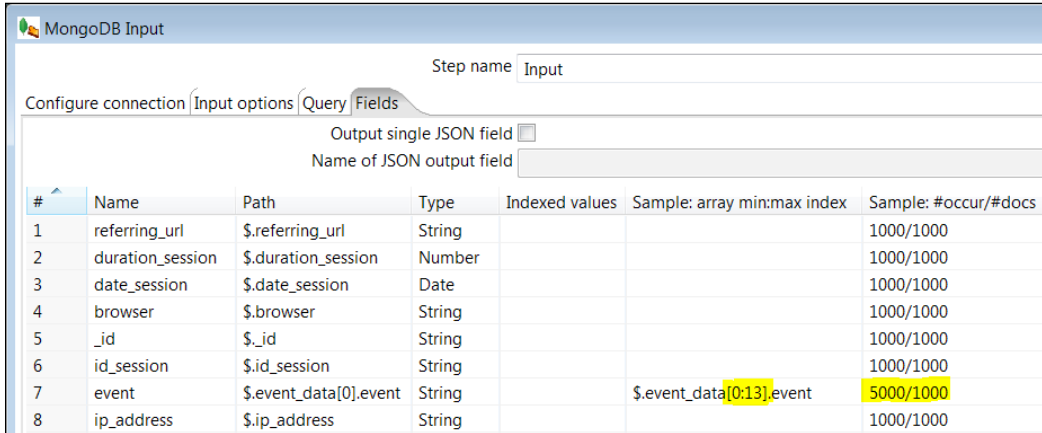
In this exercise, we'll launch Instaview from the Start menu and connect to the MongoDB database to build an **Analyzer Report**, showing total session duration by browser and referring URL.

1. Click **Instaview** from the Windows Start menu.
2. Instaview will take a few seconds to launch, and then you will be presented with two buttons. Click on the **Create New** button.
3. You have the option to choose which Big Data technology to analyze. Select **MongoDB** and click on **OK**.
4. The **MongoDB Input** step will appear and expose four important tabs for configuration to retrieve records from a single collection. These four tabs are described in the following bullets:
 - **Configure connection:** Configure a server connection with the following default information:
Host name: localhost and Port: 27017
 - **Input options:** Configure a database and collection by clicking on the **Get** buttons and selecting the following values:
Database: pentaho and Collection: sessions_events
 - **Query:** This tab is used to add a query expression to refine a read request. Leave this tab blank for now; we will revisit it later in this section.
 - **Fields:** Parse the collection document hierarchy into fields. Pentaho's **Schema on Read** functionality parses fields, paths, and data types and profiles the collection. The profile results are displayed as document statistics. You can then review and adjust the field configurations as needed.



Uncheck **Output single JSON field** and then click on **Get Fields**. Enter 10000 for the **Number of documents to sample**, and then click on **OK**.

- Review the paths and document statistics provided for each field, as shown in the following screenshot:



#	Name	Path	Type	Indexed values	Sample: array min:max index	Sample: #occur/#docs
1	referring_url	\$.referring_url	String			1000/1000
2	duration_session	\$.duration_session	Number			1000/1000
3	date_session	\$.date_session	Date			1000/1000
4	browser	\$.browser	String			1000/1000
5	_id	\$_id	String			1000/1000
6	id_session	\$.id_session	String			1000/1000
7	event	\$.event_data[0].event	String		\$.event_data[0:13].event	5000/1000
8	ip_address	\$.ip_address	String			1000/1000



The **Path** column indicates the JSON path of the field. The event path in line 7, `$.event_data[0].event`, contains brackets, indicating that this field is an array. The bracketed part of the array holds a zero—the first key value of the array.

The **Sample: array min:max** column for the event field highlighted in line 7 shows a minimum of zero array values to a maximum of thirteen array values within at least one document.

The **Sample: #occur/#docs** column for the event field is also highlighted in line 7 shows that Pentaho detected at total of 5000 array values within 1000 documents.

- The **MongoDB Input** step is now configured to retrieve records into the field structure defined in the **Fields** tab. Click on the **Preview** button at the bottom, enter a preview size of 3000 records, and then click **OK**. You will see only 1000 rows appear in the preview output, as highlighted in the following screenshot:

#	referring_url	duration_session	date_session	browser	_id	id_session
1	www.123.com	7.43	2013/01/01 00:00:00.000	Safari	5182b343ef8ed3fbbf910ae6	71E1FF1A25B84
2	www.xyz.com	11.42	2013/01/01 00:00:00.000	Chrome	5182b343ef8ed3fbbf910ae7	FA264412AB8D
3	www.abc.com	5.2	2013/01/01 00:00:00.000	IE	5182b343ef8ed3fbbf910ae8	ABF0062C6A48
4	www.xyz.com	5.59	2013/01/01 00:00:00.000	Opera	5182b343ef8ed3fbbf910ae9	C354C6923BE54



We expect 1000 to rows because the statistical data on the **Fields** tab indicates that the `sessions_events` collection contains a total of 1000 documents. It also indicates that the `events_data` array holds 5000 total events. SQL queries will need 5000 rows, one row for each event to interpret the data, yet our preview query returns only the first array value for each of the 1000 documents. This is a good example of why hierarchical document structures are problematic for SQL-based reporting tools. In the next section, we will examine how to add a query expression to the MongoDB Input step to fix this problem.

7. Close the preview output box and click on the **Query** tab.

Adding a MongoDB query expression

Pentaho allows you to enter a MongoDB query expression to restrict and process the incoming stream of data. MongoDB provides a very useful query operator, `$unwind`, to help downstream reporting interpret MongoDB document hierarchies. The `$unwind` operator works by denormalizing (that is, duplicating) the document information for each array value.

Pentaho can leverage the `$unwind` operator to duplicate the session document fields for each available event in the array. So, a document with five events becomes five documents. Each document is identical except for the value of the event, which is one of the values in the original `event_data` array. The steps for this are as follows:

1. Enter the query, `{ $unwind : "$event_data" }`, in the **Query expression (JSON)** textbox.
2. Check **Query is aggregation pipeline** and then return to the **Fields** tab.



The **aggregation pipeline** is a MongoDB framework for data aggregation. We check this option because `$unwind` is part of that framework.

- Click on the **Get fields** button, enter 10000 for **Number of documents to sample**, and then click on **OK**.



You will notice a couple of changes to the event field after clicking on **OK**. As shown in line 7 of the following screenshot, the **Path** field does not contain brackets and the **Sample: array min:max** value disappears. Most importantly, the **Sample:#occur/#docs** values changed to **5000/5000**, indicating that the query aggregation pipeline using the \$unwind operator will deliver 5000 documents as rows in the query output.

MongoDB Input						
Step name MongoDB Input						
Configure connection Input options Query Fields						
Output single JSON field <input type="checkbox"/>						
Name of JSON output field json						
#	Name	Path	Type	Indexed values	Sample: array min:max index	Sample: #occur/#docs
1	referring_url	\$.referring_url	String			5000/5000
2	duration_session	\$.duration_session	Number			5000/5000
3	date_session	\$.date_session	Date			5000/5000
4	browser	\$.browser	String			5000/5000
5	_id	\$_id	String			5000/5000
6	id_session	\$.id_session	String			5000/5000
7	event	\$.event_data.event	String			5000/5000
8	ip_address	\$.ip_address	String			5000/5000

- Click on the **Preview** button, enter 10000 for **Number of rows to preview**, and then click on **OK**.



A total of 5000 rows will be displayed. You will see several duplicate id_session values, one record for each event from that session's array of events. For example, if id_session = 71E1FF1A25B840458-64FCEC447F8D012, there are a total of nine records for the nine events in that session.

- Close the **Examine preview data** window, and you are now ready for Instaview to query MongoDB and prepare a cube model for analysis.
- Click on the **OK** button on the **MongoDB Input** step to execute the three-step Instaview process to load 5000 rows into the in-memory cache.




The three boxes represent these steps, and each shows a checkmark and **Done** if the process executes error-free as highlighted in the following screenshot:

The screenshot shows the 'Configure' window of the Instaview application. At the top, there is a 'MongoDB' section with a checked checkbox and an 'Edit...' button. Below this is a checkbox for 'Auto run Analysis when ready (Sample Size)' and a 'Run' button. The main area contains three panels: 'Data Integration' with a 'Done' status and an 'Edit' button; 'Model' with a 'Done' status and an 'Edit' button; and 'Data Cache' with a 'Done' status, a '5000 rows in cache' message, and a 'Clear' button. At the bottom, there is a 'Data last refreshed - 2/16/14 2:44:31 PM' timestamp and a 'Refresh' button.

After the three-step process successfully completes, Instaview automatically launches the Pentaho Analyzer interface to analyze the data using the generated cube model.

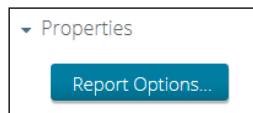
7. Review the cube measures and dimensions displayed in Analyzer.

 Instaview makes a determination of measures and dimensions based on the source field data types. Integers and numbers are added as both measures and dimensions, while string and date fields are added as dimensions.

Creating and saving an analysis view and Instaview

Pentaho bundles its popular and widely-used OLAP interface, **Analyzer**, into Instaview. Analyzer gives users a simple, drag-and-drop interface to create advanced crosstabs and visualizations of data. It is used for data exploration and creating analysis views for reporting and use in dashboards.

- To build an Analyzer Report, you drag-and-drop the available fields into the **Layout** drop zones for **Rows**, **Columns**, and **Measures**.
 - Drag **Browser** to **Rows** and Analyzer returns a single browser column with a distinct list of available browsers as rows.
 - Drag **Referringurl** to **Columns**. Analyzer returns three referring URLs as columns.
 - Drag **Durationsession** to **Measures**. Analyzer returns the sum of **Durationsession** in minutes into the crosstab cells.
- Click on the **Report Options** button.



- Check the following three items: **Show Grand Totals for Rows**, **Show Grand Totals for Columns**, and **Show drillthrough links on Measure cells**.
- Click on **OK** and review your new report. Your report should resemble the following screenshot:

No Filters				
Browser	Referringurl			Grand Total
	www.123.com	www.abc.com	www.xyz.com	
	Durationsession	Durationsession	Durationsession	Durationsession
Chrome	2473	5648	10402	18523
Firefox	2212	7415	11447	21074
IE	1838	3934	5086	10859
Opera	893	2428	2826	6147
Safari	1066	1623	2656	5344
Grand Total	8482	21048	32417	61947

- Click on the drillthrough link on **893** – the cell representing session duration for sessions using Opera and referred by the site `www.123.com`.



The drillthrough to the detail window will display 72 session records over four pages. The sum of `Durationsession` will equal the **893** cell's value.

- Hover over the `Durationsession` column header and a dropdown icon will appear. Click on the icon and select **Sort Descending**. Notice the longest session duration for this dataset is `19.84` minutes.

Referringurl: `www.123.com` - Browser: Opera - [Export to CSV](#)

Browser	Datesession	Durationsession	Event
Opera	2013-03-19 00:00:00.000000	19.84	Sort Ascending
Opera	2013-03-19 00:00:00.000000	19.84	Sort Descending
Opera	2013-03-19 00:00:00.000000	19.84	Columns
Opera	2013-03-19 00:00:00.000000	19.84	
Opera	2013-03-19 00:00:00.000000	19.84	

Watched Video

- Click on the **Export to CSV** link in the top left corner of the drillthrough window. This will export 72 session records to the `.csv` format.
- Validate the total session duration by opening the CSV file in a spreadsheet application and adding `sum (C2:C73)` to an empty cell. You will get the same **893** value that you drilled into from the Analyzer crosstab.

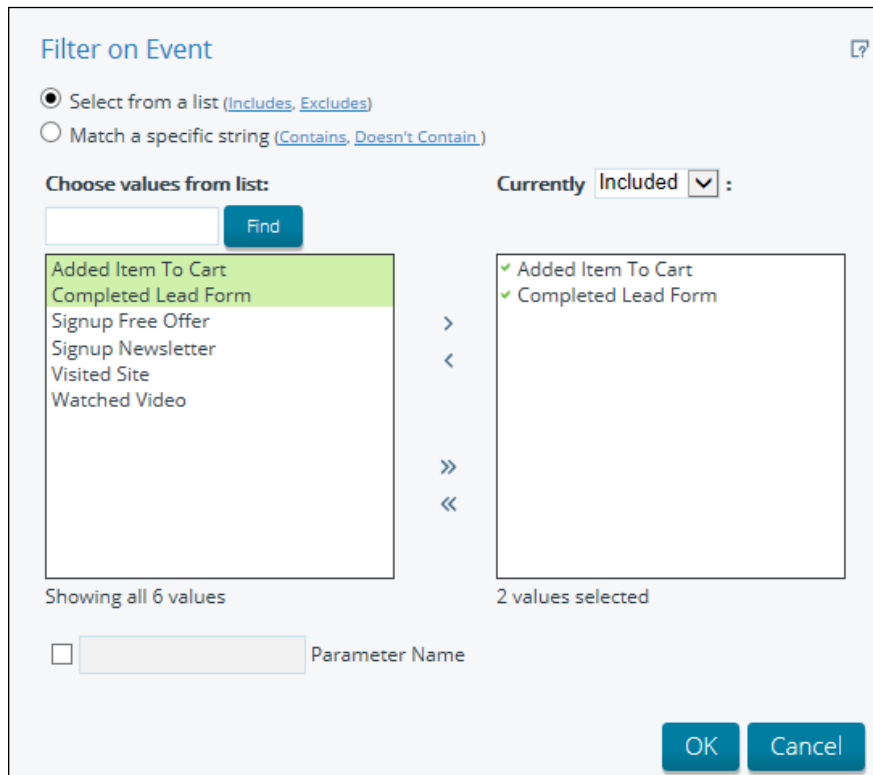


You may have noticed the session duration values repeating for each event in a session. This is a problem that will cause our total session duration to be artificially inflated when summarized for every record in the session. For example, the `Idsession = 95C4D9E016E74A89B0ABBBB40-8C30642` represents a session that lasted 12.31 minutes. However, because there are five events in that session, the duration is repeated five times for incorrect total session duration of 61.55 minutes (5×12.31).

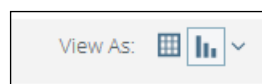
Don't worry about this issue for now! We will fix the session duration values in the next chapter by editing the cube model. For this chapter, just remain focused on using Instaview for quick prototyping and instant analysis.

- Close your spreadsheet application without saving, and then return to your Analyzer report.

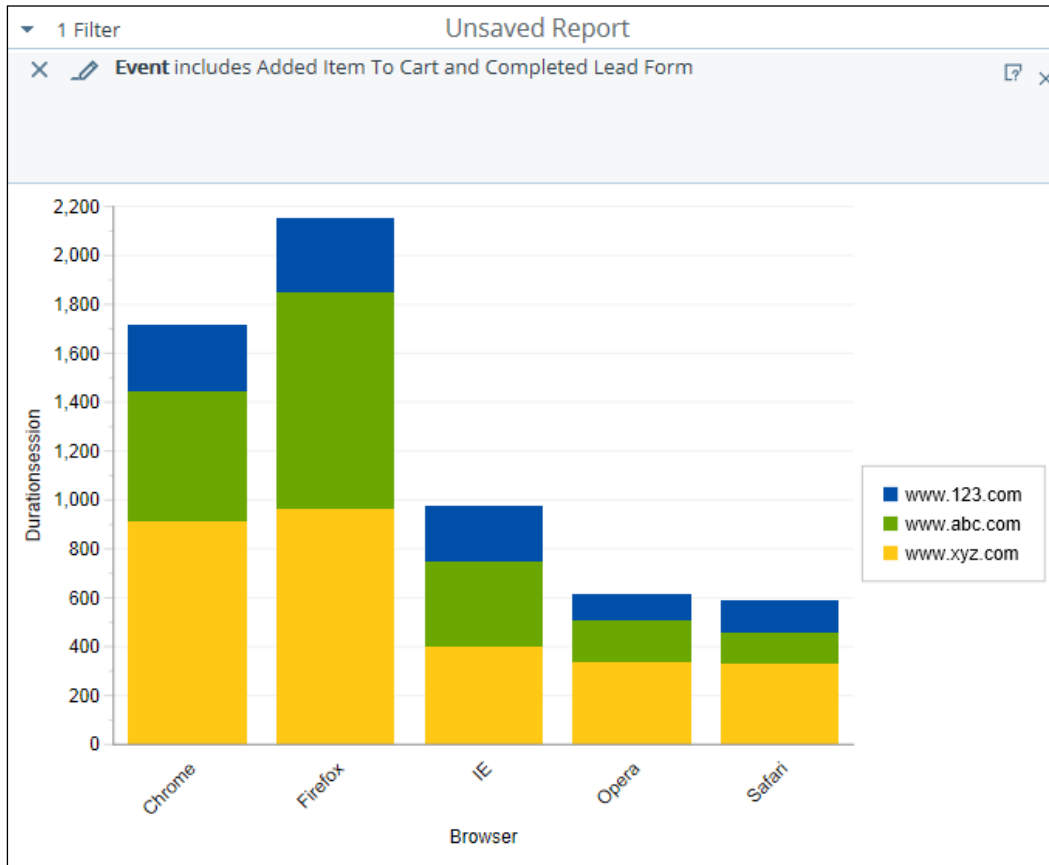
10. There are multiple ways in Analyzer to filter data. Suppose, you want to filter the report for sessions that have at least one Added Item to Cart or Completed Lead Form event. Hover over the Event dimension in the **Available fields** section, right click on it, and select **Filter**. The filter utility will display a distinct list of available events to filter your report.
11. Select Added Item To Cart and Completed Lead Form by keeping the **Shift** key and then click on the right arrow to add it to the **Currently Included** area, as shown in the following screenshot:



12. Select **OK** and return to your report. To visualize the crosstab data in a chart, click on the dropdown arrow to the right of the chart icon and select **Stacked Column**.



13. The resulting stacked column chart in the following figure shows that Firefox is the most used browser for the current subset of data. Click on **Save View** to save this view as Top Browsers by URL.



14. Click on **Save** to save your Instaview as Clickstream. When you save an Instaview, you are saving the connection definition, PDI transformation, metadata model, and any saved Analyzer views together. After saving your Clickstream Instaview, you can close it and reopen again as needed to continue development or perform a data refresh.

Summary

You are now well on your way to gaining insight from the MongoDB clickstream database using Instaview. In this chapter, we established a connection to MongoDB and parsed the `events_sessions` collection into a set of fields. This involved using the `$unwind` query operator to duplicate session information for each event array value. This result of this query operation was similar to the query results you would expect from a one-to-many join between two relational tables. We then used Analyzer to analyze the resulting data and create a stacked column chart. In the next chapter, we will continue working with Instaview to learn how to edit the metadata for additional metrics and analysis views.

4

Modifying and Enhancing Instaview Transformations

Pentaho Data Integration (PDI) is a graphical ETL tool used by Instaview to extract data from various data sources into the Instaview data cache for analysis. PDI transformations are created by Instaview to define this flow of data from source to target. Instaview gives users the ability to modify the underlying PDI transformation that is automatically generated during the initial data load process. This chapter shows users how to edit the default PDI transformation and add a new data source to create an enriched dataset for analysis.

The topics that we will cover in this chapter are as follows:

- Open an existing Instaview
- Data integration
- Add a new data source
- Create a new analysis view from the blended data

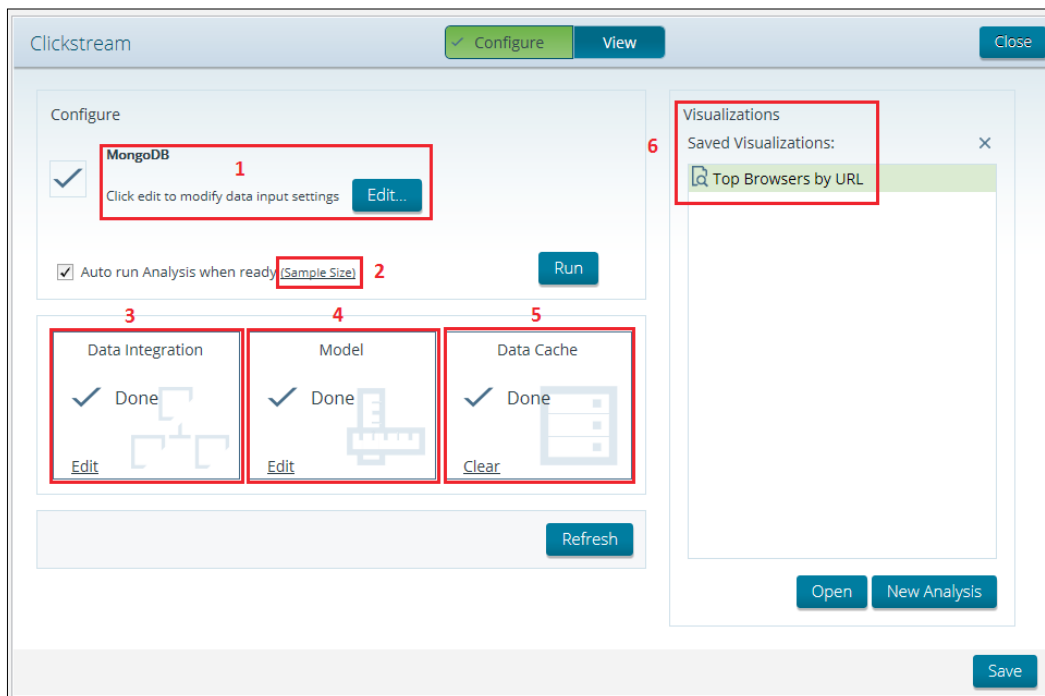
By the end of this chapter, you will have enhanced an Instaview by adding a new data source that contains geographic information. The new data source will give users the ability to analyze the sample clickstream data by geography.

Opening an existing Instaview

At this point, you should have created, saved, and closed the Clickstream Instaview. We need to reopen Clickstream to make modifications to the initial design. To open a saved Instaview, launch Instaview, and on the Welcome screen, you will see an **Open Existing** button. Click on **Open Existing** and choose the **Clickstream** Instaview file you created in the previous chapter. You can have only one Instaview file open at any one time. The Instaview file contains the following six objects:

1. A new data source connection to MongoDB
2. A defined sample size
3. A data integration transformation to extract data from MongoDB and load it into the in-memory data cache
4. A metadata model to define dimensions and measures
5. An in-memory database table containing the data cache
6. Analyzer views for visualizing data in a table or chart

These six objects are bundled together into a single Instaview file. The corresponding object locations are highlighted red and numbered in the following screenshot of Instaview in the **Configure** mode:



Data integration

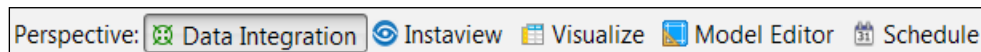
PDI is a powerful ETL tool that is used for a broad set of data integration use cases such as data warehouse development, data migration, data cleansing, and more. We focus on the graphical designer, **Spoon**, and cover only a small portion of PDI functionality in this book. Spoon is an easy-to-use, drag-and-drop environment with over 100 out-of-the-box steps that are used to create transformations. When you are ready to learn PDI in-depth, there are excellent books for learning PDI that cover each and every feature.

The Instaview build process creates a PDI **transformation** file. Transformations describe the ETL data flows such as reading source data, transforming the data, and loading the data to a target output. A transformation can be edited by clicking on **Edit** in the **Data Integration** section of Instaview, shown inside the third red box in the previous screenshot. After editing a transformation, you have to run the Instaview data load process again for the updates to be applied to the data. The next section describes how to add a second data source to your Instaview.

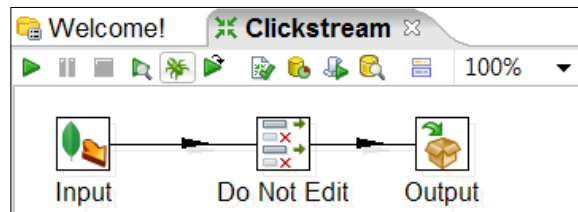
Adding a new data source

The `zip_codes.csv` flat file provided with this book contains a mapping of IP addresses to geographies down to the ZIP code level. If you don't already have this file, it can be downloaded at <http://www.packtpub.com/support>. We will use PDI to link `ip_address` from the ZIP code file to `ip_address` in the clickstream data. By blending geographic data such as country, state, city, and ZIP code with clickstream data, we will be able to analyze clickstream metrics by geography. In addition, once the data is blended with geographic data, you are able to visualize it on Google or Open Street Maps within Instaview.

The data integration **Edit** hyperlink takes you to the **Data Integration** perspective of PDI. This perspective gives you the full set of available steps for creating powerful transformations. The screenshot for this is as follows:



Notice that the default **Clickstream** transformation is opened and contains three steps, **Input**, **Do Not Edit**, and **Output**, which are connected by arrows called **hops**. Hops define the direction of the streaming data from one step to the next, shown as follows:

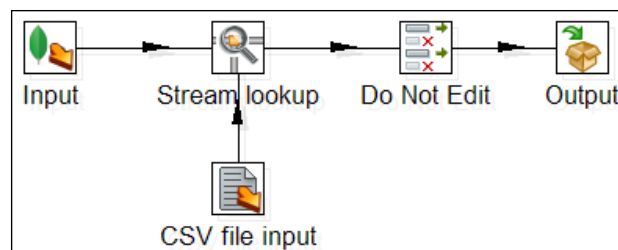


This simple transformation retrieves data from MongoDB and the specification we defined in the Instaview data source connection and then loads the data into an in-memory database for analysis.

The **Design** tab to the left organizes all of the available steps into 24 folders. These folders can be expanded to see a list of steps for that category. At first, the sheer number of available steps can be intimidating, but after some experience building transformations, you will start to know which steps are best for a particular transform requirement. A full list of transformation steps and descriptions are available via the **Transformation Steps Documentation** link on the **Welcome** tab, which takes you to the following Pentaho wiki page: <http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+Steps>.

The search input box at the top of the folder list is a handy search feature for finding steps. By typing **lookup** into the search box, you will see a list of steps that have this search term in the step name or description. A step's description appears by hovering over the step name with your mouse pointer. Hover your mouse over the **Stream lookup** step, and read the step description.

We will be using this **Stream lookup** step along with the **CSV file input** step to lookup data incoming from the `zip_codes.csv` flat file. The resulting transformation from the next exercise will match the one shown in the following screenshot:

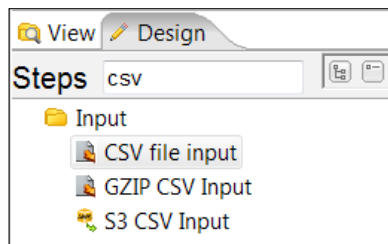


Completed Transformation


CSV file input

In this first part of the exercise, we add the **CSV file input** step and configure the step properties:


1. Make sure you are in the **Data Integration** perspective of PDI. If you are still in Instaview's **Configure** mode, click on the **Edit** hyperlink in the **Data Integration** section to switch to PDI.
2. Under the **Design** tab, type `csv` into the steps search box and find the **CSV file input** step from the list of steps.



3. Select and drag the **CSV file input** step onto the canvas.
4. Double-click on the **CSV file input** step and the Edit properties' dialog box appears.

 This area is used to specify the location of the `zip_codes.csv` file and configure properties for the step and incoming fields.

5. Click on the **Browse** button inside the properties' dialog box and browse to the `zip_codes.csv` file saved on your local disc. Select the file and click on **Open**.
6. Click on the **Get Fields** button at the bottom of the properties' dialog box to scan the incoming records and detect all available fields.
7. Enter `2000` for the sample size entry box and click on **OK**.

 Pentaho will scan all of the file records to determine field name, data type, format, and length. The **Scan results** dialog box appears after the scan is complete to display a summary of scan results. Upon reviewing the scan results, you will notice the `ip_address` field is interpreted as a numerical data type. We need to change this field's data type to `string` to match the `ip_address` field data type in MongoDB.

8. Click on **Close** to return to the properties' dialog box.

- Under the **Type** column, select **String** for the `ip_address` row.

#	Name	Type	Format	Length
1	ip_address	String	#,###,###,##	15
2	zip_code	Integer	#	15

- Lastly, uncheck **Lazy conversion?** and then click on **OK**.



Lazy conversion delays or avoids data conversion of raw binary format data from flat files to improve transformation performance when conversion is not necessary. We uncheck **Lazy Conversion** in our transformation to allow it to be converted and joined to the MongoDB string data encoded in UTF-8.

Stream lookup

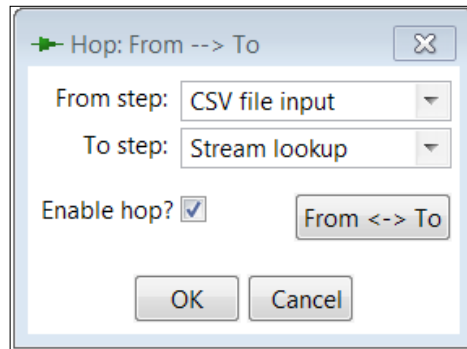
The next series of instructions tells to you add the **Stream Lookup** step to the canvas and connect it to the **Input**, **Do Not Edit**, and **CSV file input** steps using hops. You can refer back to the completed transformation in **Completed Transformation** figure to confirm your hops are set up correctly.

- Select and drag the **Stream Lookup** step onto the canvas.
- While dragging the step onto the canvas, hold your left mouse button down and move the **Stream Lookup** step directly over the hop arrow between the **Input** and **Do Not Edit** steps. This will split the existing hop into two separate hops.
- Release your mouse while the hop is highlighted to connect the **Stream lookup** step to newly split input and output hops.




Two other methods for creating a hop are to select two steps using the Lasso tool on the canvas, right-click on one of the selected steps, and choose **New Hop**. Alternatively, you can keep the *Shift* key pressed and click a step and move the mouse towards the downstream step. This will paint a hop arrow with the mouse pointer until you click on a downstream step. Upon clicking on the downstream step, select **Main output of step** to finalize the hop connection.

4. Select the **Stream lookup** using the Lasso tool and **CSV file input** steps to highlight them both.
5. Right-click on one of the highlighted steps and select **New Hop** from the menu.
6. Make sure the **From step** option is set to `CSV file input` and the **To Step** option is set to `Stream lookup` as shown in the following screenshot, and then click **OK**:



7. Double-click on the **Stream Lookup** step and the Edit properties' dialog box appears.


 Stream lookup properties are used to define the key field used to join the two datasets and the lookup fields that will be retrieved from joined records in the `zip_codes.csv` file.

8. In the **Lookup step** drop-down box, select **CSV file input**.
9. In **The key(s) to look up value(s):** section, click on the **Field** column and select `ip_address` from the list of available fields. Then, click on the **LookupField** column and choose `ip_address` from that list of fields.
10. Click on the **Get lookup fields** button at the bottom right of the properties' dialog box. This will populate the **Specify the fields to retrieve** section with all fields from the `zip_codes.csv` file.

- Right-click on the `ip_address` (row 1) in the **Specify the fields to retrieve** section and select **Delete selected lines** from the menu. The resulting Edit properties' dialog box should resemble the following screenshot.

Stream Value Lookup

Step name: Stream lookup



Lookup step: CSV file input

The key(s) to look up the value(s):

#	Field	LookupField
1	ip_address	ip_address

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	zip_code			Integer
2	city			String
3	state			String
4	country			String

- Click on **OK** and then click on the Save icon  on the PDI toolbar to save your changes.
- Click on the **Instaview** perspective icon  to return to Instaview **Configure** mode.
- Click on the **Run** button to execute your updated data integration transformation and blend the two sources of data.


Once the Instaview build process completes successfully, it will launch you back in the analysis interface to create a new analysis on the blended data.

Creating a new analysis view from blended data

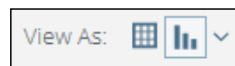
Now that Instaview has successfully blended our clickstream and geographic data, let's visualize the data on a Google map:

- Make sure you are connected to the Internet so the Google mapping feature will work.

- If you are in the Instaview **Configure** mode instead of the analysis interface, click on the **New Analysis** button.

 The measures and dimensions are updated in the **Available fields** section to include the Country, State, City, and Zipcode fields from the `zip_codes.csv` file.

- Drag `Durationsession` to the **Measures** drop zone in the **Layout** section.
- Drag `Country` to the **Rows** drop zone in the **Layout** section.
- Drag `State` to the **Rows** drop zone in the **Layout** section.
- In the **View As** section, click on the dropdown next to the chart icon and then select **Geo Map**:



- Expand the small plus icon on the right-hand side of the map and select the **Google Physical** base layer so that your visualization matches the following screenshot:



- Click on the **Save View** button and name the visualization `Chapter04-GeoMap`. Click on **OK**.

Summary

We now have an updated Clickstream Instaview with blended data and a couple of saved visualizations. However, the default metadata model, which was automatically generated by Instaview, does not give us an accurate view of the data. It needs significant model enhancements to create a better dimensional model for analysis. The next chapter explains dimensional modeling concepts and provides exercises for building a model that will give us an accurate and improved insight of our blended dataset.

5

Modifying and Enhancing Instaview Metadata

With Instaview, you can easily access, transform, and visualize data without the deep technical experience needed to stage data and design analytics solutions. The Instaview build process automatically creates a data transformation, metadata layer, and data cache, as seen in the previous chapter. This chapter shows readers the **Model Editor** perspective to modify and enhance the generated metadata model saved inside Instaview. These enhancements will make the data models more accurate, useful, and intuitive for your analytics users.

The following are the topics that we will cover in this chapter:

- Model design with dimensions and measures
- Modifying measures and dimensions
- Creating a new analysis view

By the end of this chapter, you will have an updated metadata model to better reflect the business requirements.

Model design with dimensions and measures

Instaview metadata is made up of measures and dimensions. A **measure** is a standard numerical unit used to express the size, amount, or degree of something. As numeric data elements, measures are aggregated at query time by each dimension defined in the metadata. For example, the total volume of shares traded in the stock market in the history represents a single measure with a large, aggregated value in units. That summarized measure value is not very useful for investors unless the dataset also contains dimensionality.

The total volume of shares traded for **Google stock (GOOG)** on October 18, 2013, is a more meaningful measure value. **Dimensions** such as stock symbol and time add context to the measure of volume of shares. Dimensional data becomes powerful for business users when modeled correctly with multiple dimensions and measures to form a **multidimensional model** for analysis.

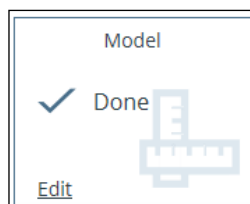
In our sample clickstream data, event dimensions provide context to the event measures. When dimensional context is applied to the event measures, we can answer questions such as: who performed the event as indicated by `ip_address`, what type of event is indicated by the event, and when did the event occur as indicated by `date_session`. The dimensional structure is defined by the available fields from the connection query. We can even combine multiple fields into levels to form dimension **hierarchies**. Hierarchies allow users to drill down each level in the hierarchy to the lowest level defined in the dimension. For example, a typical geography dimension would contain four levels in the following order from highest to lowest: country, state, city, and postal code. This hierarchy allows users to summarize revenue, for example, by country and then drill down to the hierarchy to postal code. By default, every dimension contains at least one hierarchy and one level; however, a single dimension can contain multiple hierarchies with one or more levels in each hierarchy.

A measure value in a densely populated dataset will exist for most combinations of dimension members. For example, the dimension member combination of `ip_address = "20.49.248.146"`, `event = "Watched Video"` and `date_session = "2013-03-21"` will result in an `event_count` measure value of 1. Pentaho Instaview combines measures and dimensions into a multidimensional model, so users can simultaneously analyze multiple measures and dimensions to detect patterns and anomalies and gain valuable insight into the data.

Open an existing Instaview

At this point, you should have created, saved, and closed the Clickstream Instaview modifications from the previous chapter. We need to reopen Clickstream to make modifications to the original model design. The initial model is autogenerated with dimensions and measures defined by each field's data type. By default, non-numeric data type fields are added as dimensions, while numeric data types are added as measures and dimensions. Rarely does the initial model meet the needs of the business, so our first task is to enhance the default model. Our goal is to make the model intuitive and easy to use.

In Instaview's **Configure** mode, navigating to **Model | Edit** launches you into the **Model Editor** perspective of PDI, as shown in the following screenshot:



While in edit mode, you will see a list of available fields that can be used to create your measures and dimensions along with a default multidimensional model. The following table shows the icons, names, and attributes of the available model objects:

Model object icon	Model object name	Model object attributes
	Measure	Display Name and Selected Aggregation
	Dimension	Dimension Name and Time Dimension Option
	Hierarchy	Hierarchy Name
	Level	Name, Contains only unique member Option, Source Column, and Ordinal Column

In the next section, we will make several changes to improve the model for better analysis.

Modifying measures and dimensions

In this section, we make several changes to improve the model for better analysis. It is important to check the aggregation setting for every measure in a model.

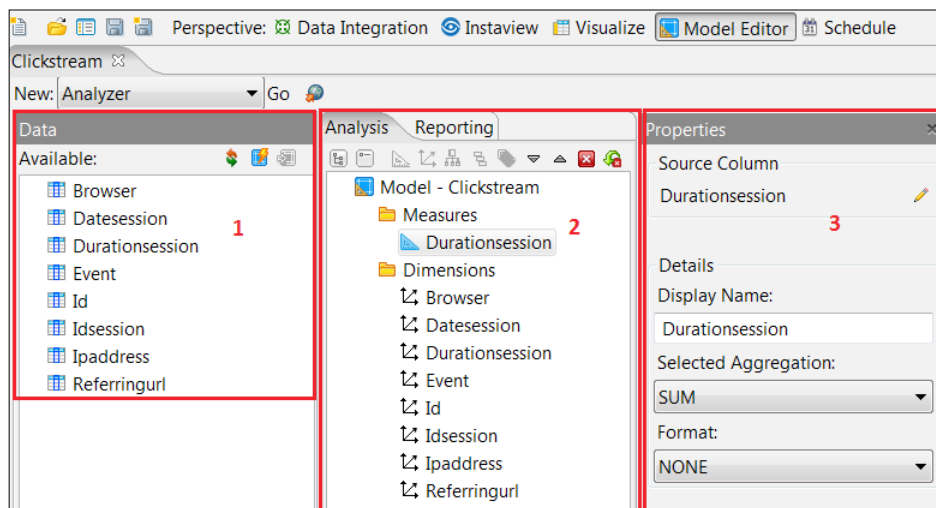
Aggregation defines the numerical function to apply to the measure when aggregating its values across the model dimensions.

1. So, the first set of model changes in this section are for measures. We change the Display Name and Selected Aggregation for Durationsession and create two new measures: Session Count, to count sessions, and Event Count, to count events. By the end of this section, your model will have three measures with three different aggregation settings: average, count distinct, and count.

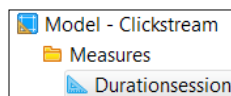
Session duration measure

In the following exercise, you will rename the `Durationsession` to a more business-friendly name and change its aggregation from sum to average, because it is incorrect to summarize averages. The steps for this are as follows:

1. While in Instaview's **Configure** mode, click on the **Edit** hyperlink in the **Model** section to start making changes. You will see the default Clickstream model containing one measure and eight dimensions.
2. The three sections available to edit a model are as follows:
 1. The **Data** tab (1) provides a list of available fields defined by your data source connection. To create a dimensional model, you drag the available fields from the **Data** section into the **Measures** and **Dimension** folders located in the **Analysis** tab (section 2).
 2. The **Analysis** tab (2) shows the measures and dimensions for analysis.
 3. The **Properties** tab (3) displays the available properties for each object you select.



3. Click on the `Durationsession` measure to highlight it, and the **Properties** tab (3) will appear to the right with dropdown menus for changing display name, selected aggregation, and format:

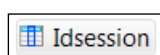


4. On the **Properties** tab (3), change the measure name by typing `Session Duration` in the **Display Name** textbox.
5. Change **Selected Aggregation** to **AVERAGE**.

Session count measure

Now, we create a new session count measure that is calculated by counting the number of distinct `Idsession` values:

1. Drag `Idsession` from the **Data** tab (1) to the **Measures** folder on the **Analysis** tab (2).



2. Click on **Idsession** in the **Measures** folder to highlight it.
3. On the **Properties** tab (3), change the measure name by typing `Session Count` in the **Display Name** textbox.
4. Change **Selected Aggregation** to **COUNT_DISTINCT**.

Event count measure

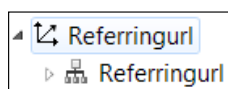
Now, we create a new event count measure that is calculated by counting the `Event` values:

1. Drag **Event** from the **Data** tab (1) to the **Measures** folder on the **Analysis** tab (2).
2. Click on **Event** in the **Measures** folder to highlight it.
3. On the **Properties** tab (3), change the measure name by typing `Event Count` in the **Display Name** textbox.
4. Change **Selected Aggregation** to **COUNT**.

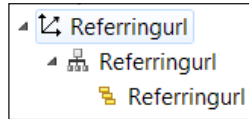
Referring URL dimension

The next set of model changes are for dimensions. We start by creating a three-level hierarchy made up of the `Referringurl`, `Ipaddress`, and `Idsession` fields. This hierarchy gives users the option to drill down from referring URL to IP address and session ID. The steps for this are as follows:

1. Expand the `Referringurl` dimension to display the `Referringurl` hierarchy.



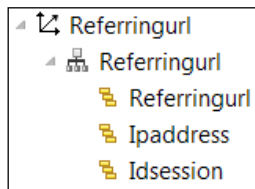
2. Expand the `Referringurl` hierarchy to display the `Referringurl` level.



3. To add a new level to the hierarchy, drag **IpAddress** from the **Data** tab (1) to the area directly below the `Referringurl` dimension level. A black level line will appear when you hover just under the level, as shown in the following screenshot:



4. Drag **Idsession** from the **Data** tab (1) to the area directly below the `IpAddress` dimension level. The result is a three-level hierarchy that looks as follows:

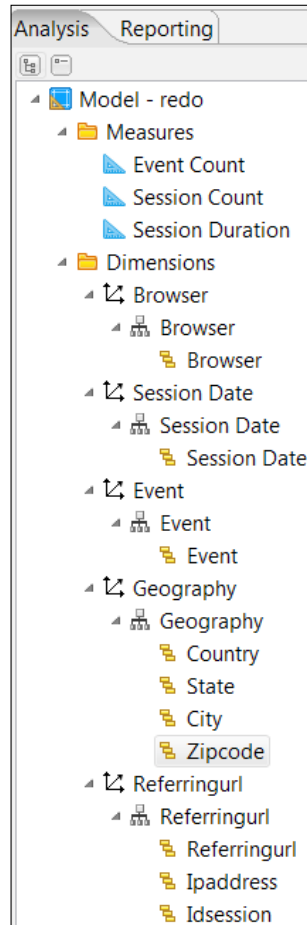


Other dimension changes

The final set of dimension changes is cosmetic. We rename a dimension and remove unnecessary or redundant dimensional information.

1. Rename the `Datesession` dimension, hierarchy, and level by highlighting each object and subsequently changing the **Display Name** to `Session Date`.
2. Delete the `Durationsession` dimension by clicking on the dimension to highlight it and then clicking on the red **X** button to delete it.
3. Delete the `Id`, `Idsession`, and `IpAddress` dimensions using the same method as in the previous step.
4. Click on the **Save** icon from the toolbar to save your model changes.

5. Compare your measures and dimensions to the model in the following screenshot:



6. When you are done comparing the models, click on the **Instaview** perspective in the top-right hand side of the window to return to Instaview's **Configure** mode:

Perspective: Data Integration Instaview

7. Click on the **Run** button in the top-left area of **Configure** mode to refresh the model with the new fields, new geography data, and updated metadata.

Creating a new analysis view

You should be in the **View** mode after refreshing your Instaview. With a newly updated model, it is a good idea to test model changes by creating a new analysis. The analysis created in the following exercise will validate that our measures are correctly aggregating across the `Referringurl` dimension:

1. Click on the **New Analysis** button.



The measures and dimensions are updated in the **Available fields** section.

2. Drag all three measures, `Event Count`, `Session Count`, and `Session Duration`, to the **Measures** drop zone in the **Layout** section.
3. Drag `Referringurl` to the **Rows** drop zone in the **Layout** section.
4. Click on the **Report Options** button, checkmark **Show Grand Totals for Columns**, and then click on **OK**.
5. Right click on the **Session Duration** column header and choose **Subtotals** from the right-click menu options.
6. Uncheck the default **Sum** aggregation and check **Average**.
7. Click on **OK**.

Your table values should match those in the following figure:

Referringurl	Event Count	Session Count	Session Duration
www.123.com	718	146	12
www.abc.com	1,704	335	12
www.xyz.com	2,578	519	13
Grand Sum	5,000	1,000	-
Grand Average	-	-	12

8. Click on the **Save View** button and name the visualization as `Chapter05-Measures_Table`. Click on **OK**.
9. Click on the **Close** button in the top-right corner of your screen, and save your Clickstream Instaview changes.

Summary

Congratulations! You have just completed the Clickstream Instaview by updating the metadata model with additional geography fields, new measures, and a hierarchy with multiple levels. As you can see, Instaview allows you to easily access, transform, and visualize data for agile development and quick prototyping. Now, we are ready to move towards more production-oriented Pentaho tools designed to deliver MongoDB data at scale and into production. The next chapter will show you the power of **Report Designer**, which is used to build prompted reports against MongoDB data.

6

Pentaho Report Designer Fundamentals

Pentaho Report Designer (PRD) gives users the power to build real-time reports sourced directly from MongoDB databases. This chapter introduces users to the Report Designer GUI and provides a quick tutorial for building a report from the sample website clickstream data stored in MongoDB.

The following are the topics that we will cover in this chapter:

- Pentaho Report Designer features
- Navigating through Pentaho Report Designer
- Creating a MongoDB connection and query
- Adding and formatting report elements

By the end of this chapter, you will be familiar with the Pentaho Report Designer GUI and will have developed your first real-time production report on a MongoDB database.

Pentaho Report Designer features

PRD is a powerful client-based reporting tool for building highly formatted, parameterized reports. With PRD, you can connect to virtually any data source and display data on one or more pages, with each page containing multiple tables and charts. Once a report is developed and tested, it can be published to the Pentaho BA server and accessed via a web browser.

Data sources

PRD provides connectivity to a variety of data sources including the following:

- MongoDB
- Relational databases
- Pentaho analysis cubes
- PDI transformations (access to inline ETL, Big Data, and web services)
- Pentaho metadata
- XML files via XPath queries
- Manually-defined data tables

Report elements

Report elements are the basic building blocks for a PRD report. They can be added to a report by dragging-and-dropping them from the palette of elements. Each report element has a comprehensive list of attributes and styles for configuration. Standard report elements include the following:

- Labels, messages, numbers, and data fields
- Static and dynamic images
- A variety of barcode standards
- Charts and sparklines
- Data- and query-driven parameters
- Cascading prompts
- Subreports

PRD **subreports** are a powerful and commonly used feature allowing you to create a single master report with multiple subreports across different data sources. Each subreport can have its own query, which receives query parameters from the master report query. We will build subreports in later exercises to enable multiple queries in a single report.

Aggregations and calculations

PRD supports the most common aggregation types including sum, average, count, count-distinct, min, and max. Additionally, all element properties in Report Designer can have formulas. There is a built-in formula editor for graphically designing expressions that can be used for conditional formatting or data-driven report elements. More in-depth calculation documentation is provided by Pentaho at http://infocenter.pentaho.com/help/topic/report_designer_user_guide/topic_adding_calculations.html.

For complex calculations, calculations such as OpenFormula for MS Excel and scriptable expressions are supported. More information on OpenFormula can be found at <http://en.wikipedia.org/wiki/OpenFormula> and <https://www.oasis-open.org/committees/download.php/16826/openformula-spec-20060221.html>.

Formatting and output

PRD gives you the flexibility to control the look and feel of every reporting element, including conditional formatting using style and attribute expressions. The report definition can even be dynamically changed at runtime. Lastly, reports can be output to a variety of different formats including the following:

- PDF
- Excel and CSV
- HTML
- Text and RTF
- XML

Navigating through Pentaho Report Designer

PRD is launched from your Windows **Start** menu under the Pentaho application folder. Within that folder, you will find the **Report Designer** item within the Pentaho Business Analytics subdirectory. Alternatively, you can launch PRD from the Windows batch file located under `\pentaho\design-tools\report-designer\report-designer.bat`.

Upon launching PRD, the **Welcome** screen will appear to help new users get started with running sample reports, creating a new report, or using the Report Wizard to assist in developing a simple report:



Report workspace

Upon clicking on **New Report**, a blank report will be created as a new tab in the report workspace. PRD is a banded report writer because the workspace is divided into horizontal layout bands that extend across the page. Each band contains different information for constructing a logical report layout. PRD will display a default set of layout bands, as shown in the following screenshot:

100%		0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0	
Page Header	-													
Report Header	-													
Details	-													
Report Footer	-													
Page Footer	-													

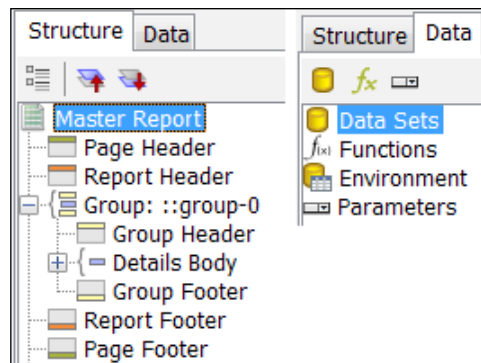
The currently selected **Page Header** band is highlighted in gray by clicking in the white band area to the right of the band title. Bands are resized by clicking-and-dragging the horizontal band border lines up or down. The following table describes the different report bands:

Band	Description
Page Header	The Page Header band contains elements at the absolute top of each page. The content in this band will be included at the top of each page in a multipage report.
Report Header	The Report Header band contains elements only on the first page of a report, just below the Page Header band.
Group Header	The Group Header band is enabled only when you want to display headers for each grouping in your data query.
Details	The Details band contains the bulk of your report data. It also contains the Details Header and Details Footer bands, hidden by default, and used to help format table column headers and footers.
No Data	If your query does not return any data, the No Data band contents that you add will appear in your report.
Group Footer	The Group Footer band is enabled only when you want to display footers for each grouping in your data query.
Report Footer	The Report Footer band contains elements only on the last page of a report, just above the Page Footer band.
Page Footer	The Page Footer band contains elements at the absolute bottom of each page. It will be included at the bottom of each page in a multipage report.
Watermark	The contents of the Watermark band become a watermark to your report.

The No Data and Watermark bands are hidden by default. You can go to each element in the **Structure** pane, and in the **Attributes** pane, select **hide-on-canvas** and **false** to make it show on the canvas.

The Structure tab

Every band and element in the workspace also appears as a hierarchy in the **Structure** tab in the upper-right corner of PRD. The workspace and the **Structure** tab mirror each other, and can be used to add or delete elements from the report. Notice in the following screenshot, how the default structure elements match the order of the workspace bands, starting with **Page Header** and ending with **Page Footer**:



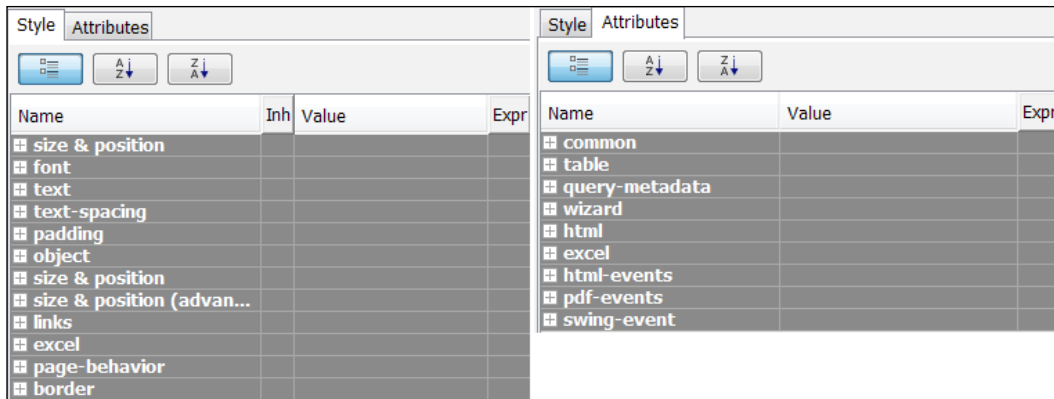
You can right-click on layout bands in the **Structure** tab to add report elements to that band.

The Data tab

The **Data** tab is also shown in the previous screenshot. This tab is where you begin development – by defining a new data source connection and query. You can define multiple queries for a report, and they will all reside here. However, you are allowed only one active query per master or subreport. The **Data** tab is also the location for adding functions, environment variables, and parameters. Once you define a query, you can drag-and-drop query data elements from the **Data** tab to one of the layout bands.

The Style and Attributes tabs
















The top-level **Master Report** element is selected on the **Structure** tab in the previous screenshot. When you select any element in the **Structure** tab, a detailed list of element properties organized by category will appear on the **Style** and **Attribute** tabs below it. The following screenshot shows a collapsed view of all property categories for both tabs:



The palette

The palette contains all of the reporting elements used to build a report. You can click on any element icon from the palette and drag it onto a layout band. The following table describes the different report elements:

Element icon	Element name	Element description
	Label	A static text string that cannot be changed dynamically.
	Text Field	A dynamic text field that can be changed dynamically through a query or function.
	Number Field	A dynamic numerical field that displays numerical data from a query.
	Date Field	A dynamic date field that displays date information from a query.
	Message Field	A combination static/dynamic report element typically used to combine static text along with dynamic elements.
	Resource Label	A static text string that maps to a resource bundle, allowing you to localize labels.
	Resource Field	A dynamic text string that maps to a resource bundle, allowing you to localize any database field.
	Resource Message	A dynamic text string that combines multiple elements and maps to a resource bundle, allowing you to localize the combined content.
	Image Field	A dynamic image field that displays images stored in a database.

Element icon	Element name	Element description
	Image	A static image report element that displays images embedded in the report.
	Ellipse	A round or oval vector graphic.
	Rectangle	A rectangle vector graphic.
	Horizontal Line	A horizontal line vector graphic.
	Vertical Line	A vertical line vector graphic.
	Survey Scale	A sliding scale chart element.
	Chart	A graphical chart element that displays query data in a chart.
	Simple Barcodes	A barcode chart element.
	Bar Sparkline	A bar sparkline chart element.
	Line Sparkline	A line sparkline chart element.
	Pie Sparkline	A pie sparkline chart element.
	Band	A layout report element used to group other elements together.
	Sub report	A separate report page that is nested inside the master report. Subreports can reference separate queries that can be linked to the master report.
	Table of Content	A subreport element that generates a table of contents based on the groups in your report.
	Index	A subreport element that generates an index based on fields in your report. The index will display instances and page numbers in which the field name appears.

The main menu and toolbar

The main toolbar at the top of the **Report Designer** window is used for common file operations such as cut, copy, and paste. You can hover the mouse pointer over an icon to reveal the purpose of each button. The main menu houses several features for designing your report. The **View** menu exposes a few important alignment features that you should be aware of before we begin developing a report. By default, **Grids** should be enabled. Grids provide an X and Y axis with evenly-spaced hash marks to help you align your report elements. You can set the distance between the hash marks by navigating to **View | Grids | Settings**. In addition, **Guides** are vertical or horizontal lines that help you align elements. **Guides** are created by clicking on the page rulers at the top-left corner of the report page. They can be removed by right-clicking on the guide and selecting **Delete**.

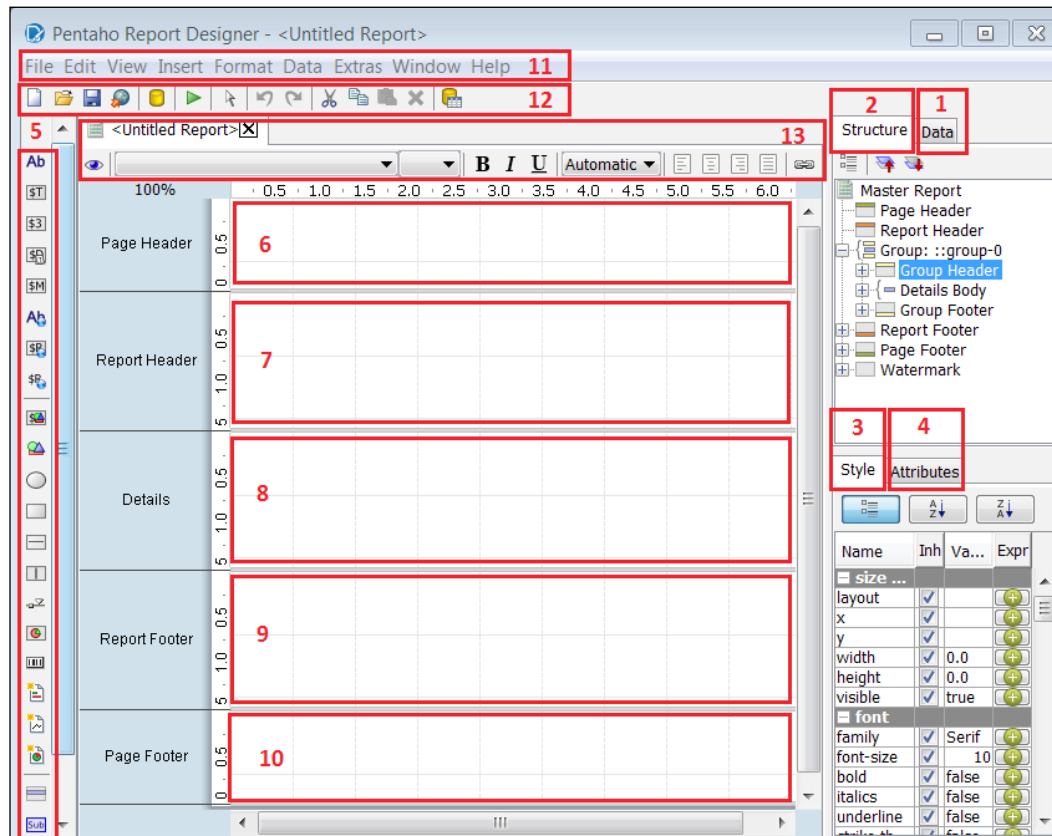
Additional alignment features from the **View** menu worth mentioning are **Element Alignment Hints**, **Snap to Elements**, and **Show Overlapping Elements**. **Element Alignment Hints** can be helpful as you begin your report, but the hint lines become cumbersome once you have several objects on the page. For our first report, be sure to disable **Element Alignment Hints** and enable **Snap to Elements** and **Show Overlapping Elements**.

The tab toolbar

Each open report will have its own tab in Report Designer. You can have multiple report tabs open with the currently selected report highlighted in blue. Beneath the **Report** tab is a tab toolbar for modifying fonts and previewing your report. The eye icon to the far left of the tab toolbar will often be used to execute the report and preview how it looks when published. Click on the eye icon to preview your report and it will change to a pencil (edit) icon. Click on the pencil icon to return to the edit mode.

Interface reference

The following screenshot divides the Report Designer interface into 13 sections for reference:



The following table lists each section, numbered as per the preceding screenshot:

Section#	Section name
1	The Data tab
2	The Structure tab
3	The Style tab
4	The Attributes tab
5	Palette
6	The Page Header band
7	The Report Header band

Section#	Section name
8	The Details band
9	The Report Footer band
10	The Page Footer band
11	The main menu
12	The main toolbar
13	The tab toolbar

Creating a MongoDB connection and query

The first step in PRD report development is to create a database connection and execute one or more queries. This section explains how to create a MongoDB connection and query. When creating a MongoDB connection, there are several MongoDB-specific configuration options. We do not use some of the more advanced configuration options for exercises in this book, but it is important to be aware of them. MongoDB-specific configuration features, which are not used in the exercises, include the following:

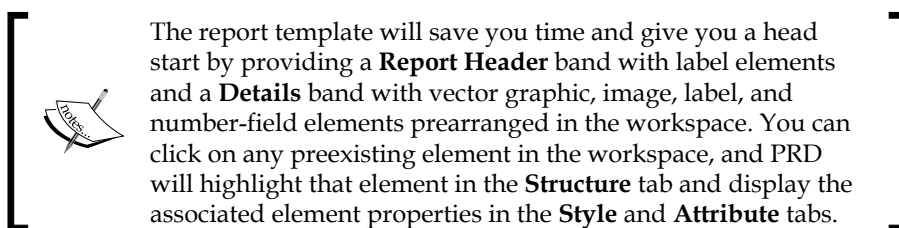
- **Use all replica set members:** Replica sets provide redundancy and increase data availability. If there is a replica set with more than one host, Pentaho discovers all hosts listed in the host field.
- **Read preference:** This tells Pentaho which node to read first — **primary**, **primaryPreferred**, **secondary**, **secondaryPreferred**, or **nearest**.
- **Tag set:** Tag sets let you customize write concern and read preferences for a replica set.
- **Fields expression:** This controls the query fields to return. This is called **projection** in MongoDB terms. This is enabled only when the query is not an aggregation pipeline query. If empty, all fields are returned.

For more information on replication in MongoDB, visit <http://docs.mongodb.org/manual/replication/>.

Adding a MongoDB data source

A PRD report template, `report_template_chapter06.prpt`, has been provided to give you a head start developing your first report. You will need to open this report template in PRD to begin. Perform the following steps to add a MongoDB datasource:

1. If you are on the **Welcome** screen, close it to return to the main window.
2. Click on the open icon on the menu toolbar or navigate to **File | Open** from the main menu and browse to the `report_template_chapter06.prpt` file provided with this book.



3. Click on the **Data** tab, right-click on **Data Sets** and select **MongoDB** from the list.
4. While in the **MongoDB Data Source** dialog box, click on the green circle with a plus symbol to add a new query.
5. Rename `Query 1` to `metrics_summary_query`.
6. We now need to complete the four data source tabs starting with the **Configure connection** tab. Type `localhost` for **Host name** and `27017` for **Port**.
7. On the **Input Options** tab, click on **Get DBs** and select **pentaho** from the list of databases.
8. Click on **Get collections** and select `sessions_events` from the list of collections.
9. On the **Query** tab, type or paste the following query into the **Query expression** input box. This query is located in the code files provided with this book so that you can copy and paste the code:

```
{ $unwind : "$event_data" },
{ $group : {
  _id: 1,
  Visits : { $sum: { $cond: [ { $eq: [ "$event_data.event",
"Visited Site" ] }, 1, 0 ] } },
```

```

    Offers : {$sum: { $cond: [ { $eq: [ "$event_data.event",
"Signup Free Offer" ] },1,0] }},
    Leads : {$sum: { $cond: [ { $eq: [ "$event_data.event",
"Completed Lead Form" ] },1,0] }},
    Purchases : {$sum: { $cond: [ { $eq: [ "$event_data.event",
"Added Item To Cart" ] },1,0] }},
    Total_Events: {$sum: 1}
  } } }

```

10. Be sure to check the **Query is aggregation pipeline** option.



This query is an aggregation pipeline because it uses aggregation framework operators such as \$unwind, \$group, and \$sum. The \$unwind operator will unwind the event data array so that we can then accurately count and summarize each instance of an event. The result of this query will be five summary metrics for the report. For more information on the aggregation framework, visit <http://docs.mongodb.org/manual/reference/operator/aggregation/>.

11. On the **Fields** tab, click on **Get fields**. Pentaho will scan the `sessions_events` collection to detect the collection schema and parse the available document fields.
12. Click on the **Preview** button to run the query and view the results. Your query results should match the numbers in the following screenshot:

Preview					
Visits	_id	Purchases	Total_Events	Leads	Offers
2379		1	156	5000	321
					505

13. Click on **OK** to return to the workspace.



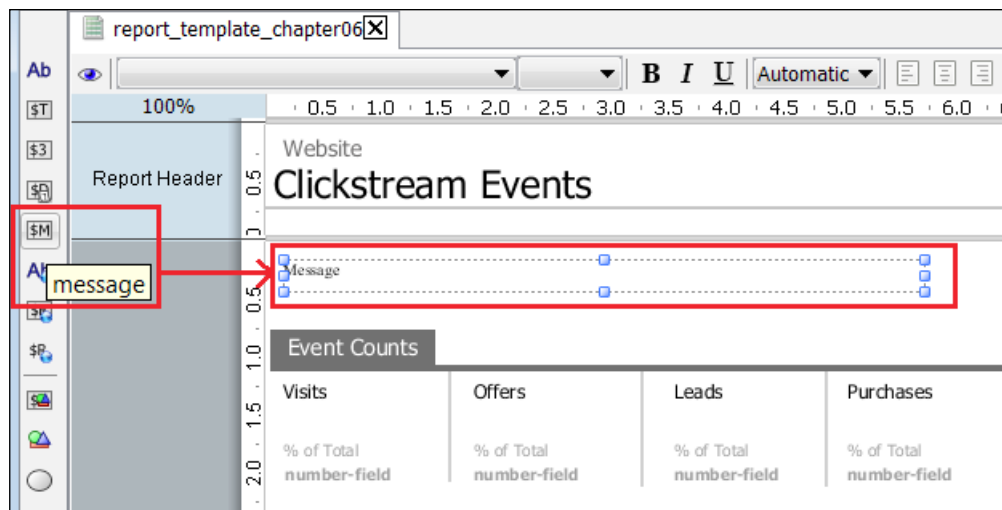
On the **Data** tab, notice **Data Sets** has a new MongoDB entry, and beneath that entry is a `metrics_summary_query` folder. You can expand this folder by double-clicking on it to see the six available fields from your query.

Adding and formatting report elements

The next step in the Report Designer report development is to add data elements from the `metrics_summary_query` folder to the report and modify the element formats. This section explains how to easily add message and number-fields to the report.

Adding a message field to your report

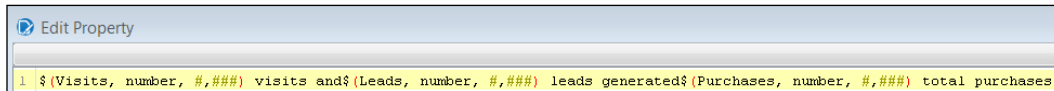
1. Click on the message field icon from the palette on the left-hand side and drag it directly above the rectangular **Event Counts** label on the **Details** band, as shown in the following screenshot:



2. Resize the **Message** field to the width of the page by selecting and dragging the right-hand side border of the **Message** field to the right.
3. Double-click on the **Message** field and an edit button will appear on the right-hand side of the **Message** field. Click on this edit button to make edits.
4. Type or paste the following code in the **Edit Property** input box and then click on **OK**. This query is located in the code files provided with this book, so that you can copy and paste the code.

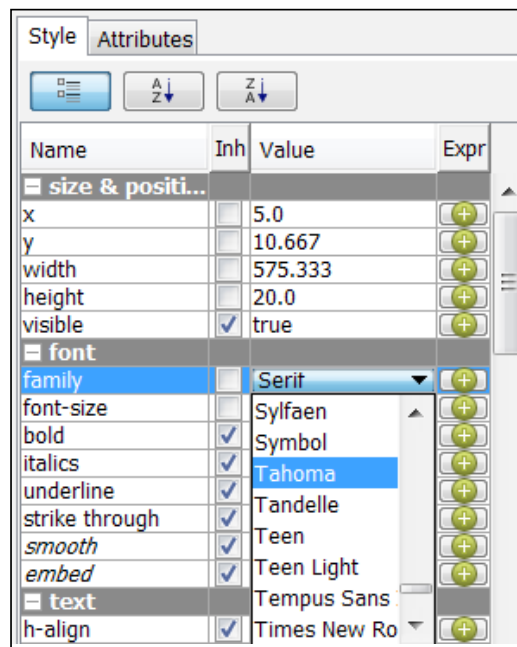
```
$(Visits, number, #,###) visits and$(Leads, number, #,###) leads  
generated$(Purchases, number, #,###) total purchases!
```

The following screenshot shows the mentioned query in the **Edit Property** window:



Documentation on the calculation syntax using LibFormula, as used in the previous calculation, is provided on the Pentaho wiki at <http://wiki.pentaho.com/display/Reporting/LibFormulaSyntax>.

5. Click on the **Structure** tab and locate the font-size property in the **Style** tab.
6. Change the font of the **Message** field to Tahoma and the font size to 20, as shown in the following screenshot:

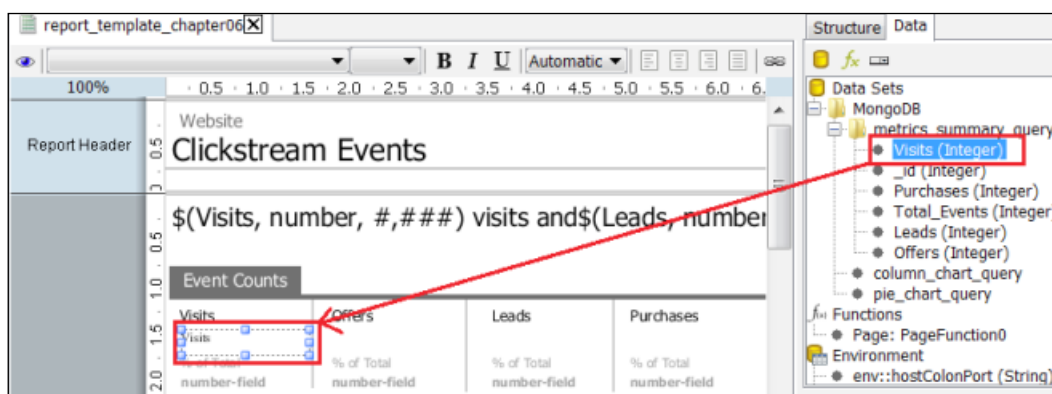


7. Click on the eye icon on the tab toolbar to preview your report and test the code. You should see the **Message** field display **2,379 visits and 321 leads generated 156 total purchases!**.
8. Click on the pencil icon (edit) to return to the edit mode.


Adding number-fields to your report

There are a couple of methods for adding number-fields to your report. The first method is to drag a number-field from the palette and then assign it to a field from the data query. The second method is to drag query fields from the **Data** tab directly onto the report. Because the query fields are of the integer data type, they will be added as number-fields in the workspace. We will use the second method because it requires fewer steps to complete. The steps are as follows:

1. On the **Data** tab, click on the `Visits` data field from under `metrics_summary_query` and drag it underneath the **Visits** label on the **Details** band, as shown in the following screenshot:



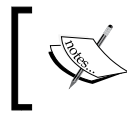
2. Click on the `Offers` query element from under `metrics_summary_query` and drag it underneath the **Offers** label on the **Details** band.
3. Click on the `Leads` query element from under `metrics_summary_query` and drag it underneath the **Leads** label on the **Details** band.
4. Click on the `Purchases` query element from under `metrics_summary_query` and drag it underneath the **Purchases** label on the **Details** band.

 To align the left-hand side edge of each number-field to the left edge of the label above it, you can drag the left border of either element, and the snap-to-grid feature will assist with alignment. Alternatively, you can click on both elements keeping the *Shift* key pressed and then right-click on one of the highlighted elements to navigate to **Alignment | Left**.

5. Click on all four new number-fields, keeping the *Shift* key pressed, to select them together for editing.

6. With all four number-fields highlighted, right-click on one of the highlighted fields and navigate to **Alignment** | **Top**.
7. With all four number-fields still highlighted, click on the **Structure** tab and locate the `font-size` property in the **Style** tab.
8. Change the font to `Tahoma` and `font-size` to `20`.
9. Click on the eye icon on the tab toolbar to preview your report. Your report should match the following screenshot:

Clickstream Events			
2,379 visits and 321 leads generated 156 total purchases!			
Event Counts			
Visits	Offers	Leads	Purchases
2,379	505	321	156
% of Total	% of Total	% of Total	% of Total



Notice how the **% of Total** fields are not yet complete in the preceding screenshot. You'll learn how to fix that in the next exercise.

Adding calculated values to your report

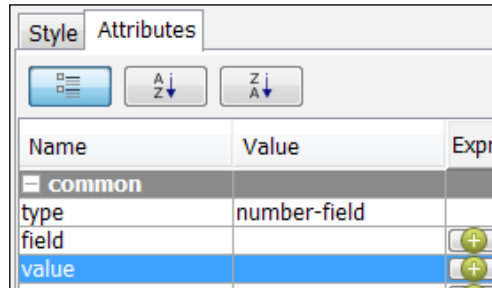
This last section of this chapter steps you through adding the percentage of total calculations to number-fields. These calculations use the `metrics_summary_query` values as numerators and denominators in calculating each metric's percentage of total `Visits`.

1. While in edit mode, notice the four existing number-fields just below the **% of Total** labels for each of the four metrics in the details workspace.
2. Highlight the first number-field under `Visits`:

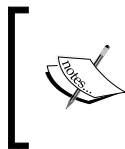
Event Counts	
Visits	
Visits	
% of Total	
number-field	

3. With **number-field** highlighted, click on the **Structure** tab and then the **Attributes** tab below it.

4. Locate the value property within the **common** category.
5. Click on the round, green **Expr** button for the value property:



6. This will open the **Edit Expression** dialog box. Type or paste the following code into the **Formula** input box:
`= [Visits] / [Visits]`
7. Click on **OK**.



Notice that the round, green **Expr** button has changed to a pencil icon, indicating that edits have occurred for this property. Now, let's make a similar calculation edit for the remaining three metrics.

8. Click on the **number-field** element under **Offers** to select it and then edit its value property by clicking on the **Expr** button.
9. Type or paste the following code into the **Edit Expression** formula input box and then click on **OK**:
`= [Offers] / [Visits]`
10. Click on the **number-field** element under **Leads** to select it and then edit its value property by clicking on the **Expr** button.
11. Type or paste the following code into the **Edit Expression** formula input box and then click on **OK**:
`= [Leads] / [Visits]`
12. Click on the **number-field** element under **Purchases** to select it and then edit its value property by clicking on the **Expr** button.
13. Type or paste the following code into the **Edit Expression** formula input box and then click **OK**:
`= [Purchases] / [Visits]`

14. Click on the eye icon on the tab toolbar to preview your report. Your report should match the following screenshot:

Clickstream Events			
2,379 visits and 321 leads generated 156 total purchases!			
Event Counts			
Visits	Offers	Leads	Purchases
2,379	505	321	156
% of Total	% of Total	% of Total	% of Total
100%	21.2%	13.5%	6.6%

15. From the main menu navigate to **File | Save As** and save the report on your local hard drive as `chapter06_clickstream_report`.

Summary

Congratulations! You have just completed the first of two chapters on building a Pentaho report that queries real-time data from a MongoDB database! This chapter has exposed you to the powerful integration between Pentaho Report Designer and MongoDB, and this is just the beginning. The next chapter will continue the development of this PRD report by adding new queries, charts, and prompts.

7

Pentaho Report Designer Prompting and Charting

The previous chapter introduced Pentaho Report Designer (PRD) by showing you how to build a simple, single-query report on the MongoDB clickstream data. Now we are going to learn some additional advanced features to enhance your report with new queries, charts, and a report prompt.

The following are the topics that we will cover in this chapter:

- Adding additional MongoDB queries
- Visualizing your data with charts
- Creating a report prompt

By the end of this chapter, you will have finished development for a professional-looking clickstream analytics report that produces queries directly from MongoDB data.

Adding additional MongoDB queries

PRD can execute multiple simultaneous queries against MongoDB. Each query can be used to populate different metrics, tables, or charts. This is an important feature when you need an operational dashboard or a briefing-book-style report with multiple views of data sourced from different data sources. Our report specifications, listed as follows, call for a dashboard-style report showing event count totals:

- Displayed as summary metrics at the top of the report
- Sorted by `event_type` and displayed in a column chart
- Sorted by `browser` and displayed in a bar chart

The PRD report you created in *Chapter 6, Pentaho Report Designer Fundamentals*, named `chapter06_clickstream_report.prpt`, has an existing MongoDB dataset connection and query. The existing query, `metrics_summary_query`, is used to populate the message and number-fields. In this section, we create two additional queries to populate the column and bar charts.

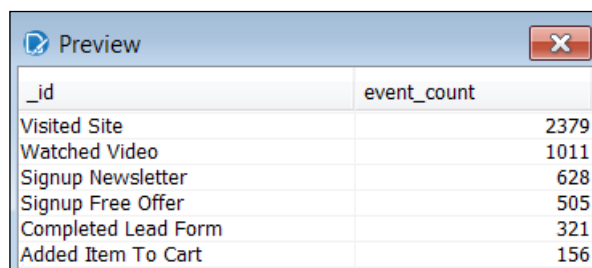
Adding a bar chart query

The following steps in this exercise will guide you through adding a query for the count of events by event type:

1. If you are on the **Welcome** screen, close it to return to the main window.
2. Click the open icon on the menu toolbar or navigate to **File | Open** from the main menu and browse for the `chapter06_clickstream_report.prpt` file that you created in *Chapter 6, Pentaho Report Designer Fundamentals*.
3. Click on the **Data** tab, right-click on the existing **MongoDB** dataset, and select **Edit Datasource** from the list.
4. While in the **MongoDB Data Source** dialog box, click on the green circle with a plus symbol to add a new query.
5. Rename `Query 1` to `bar_chart_query`.
6. We now need to complete the four data source tabs starting with the **Configure connection** tab. Type `localhost` for **Host name** and `27017` for **Port**.
7. On the **Input Options** tab, click on **Get DBs** and select **pentaho** from the list of databases.
8. Click on **Get collections** and select `sessions_events` from the list of collections.
9. On the **Query** tab, type or paste the following query into the **Query expression** input box. This query is located in the code files provided with this book so that you can copy and paste the code:

```
{ $unwind : "$event_data" },
{ $group : { _id : "$event_data.event", event_count : { $sum : 1 }
} },
{$sort:{event_count: -1}}
```
10. Be sure to check the **Query is aggregation pipeline** option.
11. On the **Fields** tab, click on **Get fields**. Pentaho will scan the `sessions_events` collection to detect the collection schema and parse the document fields defined by the query.

12. In the **Name** column, rename `_id` to `event_type`.
13. Click on the **Preview** button to run the query and view the results. Your query results should match the numbers shown in the following screenshot:



<code>_id</code>	<code>event_count</code>
Visited Site	2379
Watched Video	1011
Signup Newsletter	628
Signup Free Offer	505
Completed Lead Form	321
Added Item To Cart	156

14. Click on **OK** to return to the workspace.
15. Navigate to **File | Save As** and save an updated copy as `chapter07_clickstream_report`.



On the **Data** tab, notice that the **MongoDB** dataset now contains two queries.

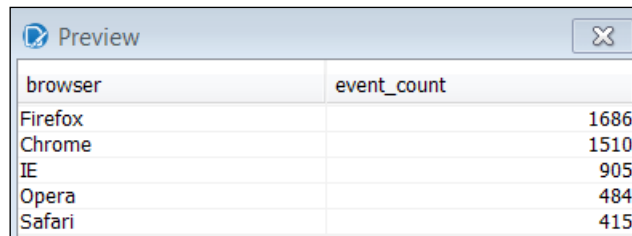
Adding a pie chart query

The following steps in this exercise will guide you through adding a query for the count of events by the web browser type:

1. Click on the **Data** tab, right-click on the existing **MongoDB** dataset, and select **Edit Datasource** from the list.
2. While in the **MongoDB Data Source** dialog box, click on the green circle with a plus symbol to add a new query.
3. Rename **Query 1** to `pie_chart_query`.
4. We now need to complete the four data source tabs starting with the **Configure connection** tab. Type `localhost` for **Host name** and `27017` for **Port**.
5. On the **Input Options** tab, click on **Get DBs** and select **pentaho** from the list of databases.
6. Click on **Get collections** and select `sessions_events` from the list of collections.


7. On the **Query** tab, type or paste the following query into the **Query expression** input box. This query is located in the code files provided with this book so that you can copy and paste the code:

```
{ $unwind : "$event_data" },
{ $group : { _id : "$browser", event_count : { $sum : 1 } } },
{$sort:{event_count: -1}}
```
8. Be sure to check the **Query is aggregation pipeline** option.
9. On the **Fields** tab, click on **Get fields**. Pentaho will scan the `sessions_events` collection to detect the collection schema and parse the document fields defined by the query.
10. In the **Name** column, rename `_id` to `browser`.
11. Click on the **Preview** button to run the query and view the results. Your query results should match the numbers in the following screenshot:



browser	event_count
Firefox	1686
Chrome	1510
IE	905
Opera	484
Safari	415

12. Click on **OK** to return to the workspace.
13. Click on the save icon to save your work.

 On the **Data** tab, notice the **MongoDB** dataset now contains three queries.

Visualizing your data with charts

Charts are a powerful feature for visualizing data, and PRD provides two types of charts: **JFreeChart** and **sparkline** charts. JFreeChart is a Java chart library with support for a wide range of chart types. There are 17 JFreeChart chart types available in PRD.

JFreeChart chart types

The following chart types are included out of the box in PRD.

- Bar
- Line
- Area
- Pie
- Multi-Pie
- Bar Line Combination
- Ring
- Bubble
- Scatter Plot
- XY Bar
- XY Line
- XY Area
- Extended XY Line Chart
- Waterfall
- Radar
- XY Area Line

Subreports

All charts are query-driven, and the bar and pie charts we develop will each require separate queries; however, the Master Report can only be associated with one query. **Subreports** are an important PRD feature because each subreport can be tied to a separate data source and query. A PRD report can contain an unlimited number of subreports, giving you the flexibility to design reports that contain multiple data points and visualizations. The clickstream Master Report is already associated with the `metrics_summary_query` query.



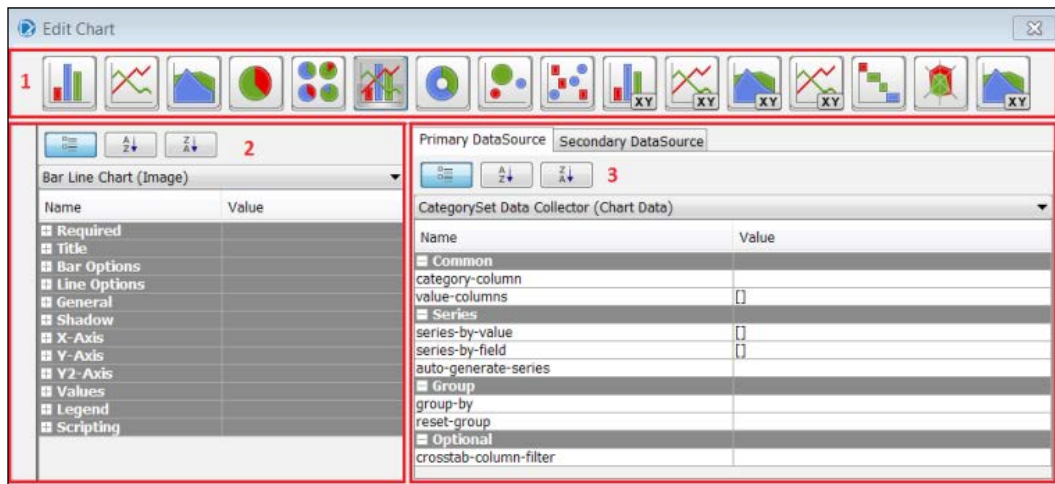
The Master Report query can be changed by highlighting **Master Report** in the **Structure** tab and selecting a query from the **Value** dropdown in the **Attributes** tab.

We are required to add two subreports, one for each chart, because the charts are driven by two separate queries.

When you add a subreport to your Master Report, PRD will prompt you to choose between an inline or banded subreport element and pick a query to associate with that subreport. **Inline subreports** can be placed beside other elements, while **banded subreports** occupy 100 percent of the page width.

Chart data collectors and properties

After adding a subreport to your Master Report, you can then double-click on the subreport element to open it as another tab. This second tab is where you add a chart. Once added, the **Edit Chart** dialog box presents several chart types and chart configuration options. Each chart type is associated with one or more data collectors that have different customizable properties. The following screenshot shows the main sections of the **Edit Chart** dialog box:



The following table lists the numbered sections of the preceding screenshot:

Section #	Section name
1	Chart Types
2	Chart Properties
3	Data Collectors

Data Collectors limit your query to the columns needed for the chart, and they specify which query fields to use for chart categories, XY axis values, and series values. The typical workflow in the **Edit Chart** dialog box is as follows:

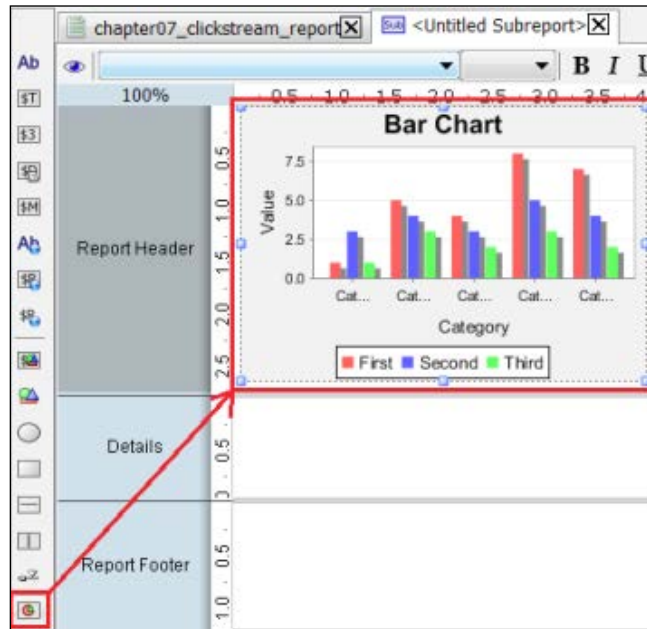
1. Choose a chart type.
2. Configure the data collector.
3. Preview the chart.
4. Iterate between preview and chart properties to customize the chart.
5. The following two sections step you through this workflow to add a bar chart and pie chart to your report.

Creating a bar chart

The following steps of this exercise will guide you through adding a bar chart that shows the count of events by the event type:

1. Open the `chapter07_clickstream_report` PRD file that you saved in the previous section.
2. Click on the subreport icon from the palette on the left-hand side and drag it directly below the `Visits` metric on the **Details** band.
3. Select **Inline** when prompted to choose a subreport type.
4. Select `bar_chart_query` when prompted by the **Select Data Source** dialog box.

5. After choosing a query, a new tab, **<Untitled Subreport>**, will open. Click on the chart icon from the palette on the left-hand side and drag it onto the subreport **Report Header** band, as shown in the following screenshot:



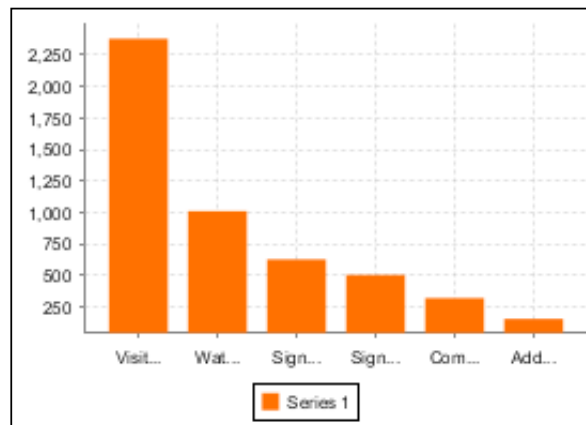
Subreports added to the **Report Header**, **Report Footer**, or **No Data** band will be executed once by the Master Report. Subreports added to the **Group Header** or **Group Footer** bands will be executed each time they are displayed. Subreports added to the **Details** band will be executed once for each row in the Master Report data source.

6. Double-click on the **Bar Chart** element you just added and the **Edit Chart** dialog box will appear. The **Primary DataSource** tab contains two **Data Collector** properties, category-column and value-columns, that you assign to query fields.
7. Select the **Value** field for category-column and choose the `event_type` field from the drop-down menu.
8. Select the **Value** field for value-columns and then click on the edit button to open the **Edit Array** dialog box.
9. Select `event_count` from the **Available Items** section and click on the add items arrow to add `event_count` to the **Selected Items** section.

- Click on **OK** and make sure your **Data Collector** properties match the next screenshot:

CategorySet Data Collector (Chart Data)	
Name	Value
Common	
category-column	event_type
value-columns	[event_count]

- Click on **OK** again to close the **Edit Chart** dialog and return to your report.
- Click the eye icon on the tab toolbar to preview your report with the new bar chart. You should see a chart image that resembles the following screenshot:

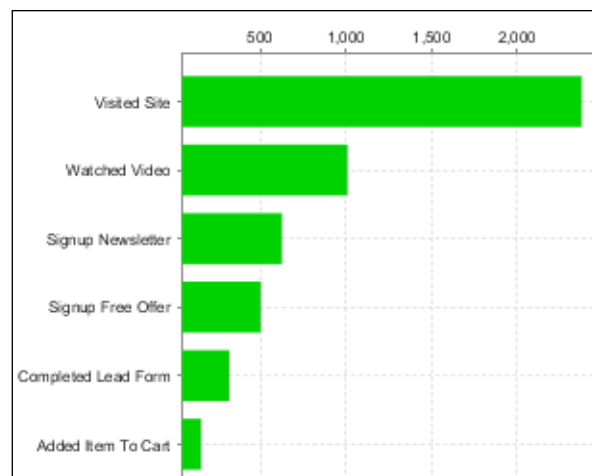


Modifying bar chart properties

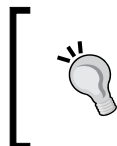
As you can see from the previous screenshot, we need to make a few cosmetic changes to the bar chart. The remaining steps have you set chart properties to change the bar color, remove the legend, and rotate the bars horizontally:

- Click on the pencil icon on the tab toolbar to return to the report edit mode.
- Double-click on the **Bar Chart** element to return to the **Edit Chart** dialog box. The **Bar Chart (Image)** section contains the following properties to make the required changes: `horizontal`, `series-color`, and `show-legend`.
- Select the **Value** field for the `horizontal` property and choose **True** from the drop-down menu.
- Select the **Value** field for the `series-color` property and then click on the edit button to open the **Edit Array** dialog box.

5. In the **Selected Items** section, select the green color to match the MongoDB logo. Click on the yellow **Move Up** arrow to move your selected color to the top of the list and then click on the **OK** button.
6. Scroll to the bottom of the properties list to the **Legend** category. Select the **Value** field for the show-legend property and choose **False** from the drop-down menu.
7. Click on the eye icon on the tab toolbar to preview your report with the new bar chart. You should see a chart image that resembles the following screenshot:



8. Click on the save icon to save your work.



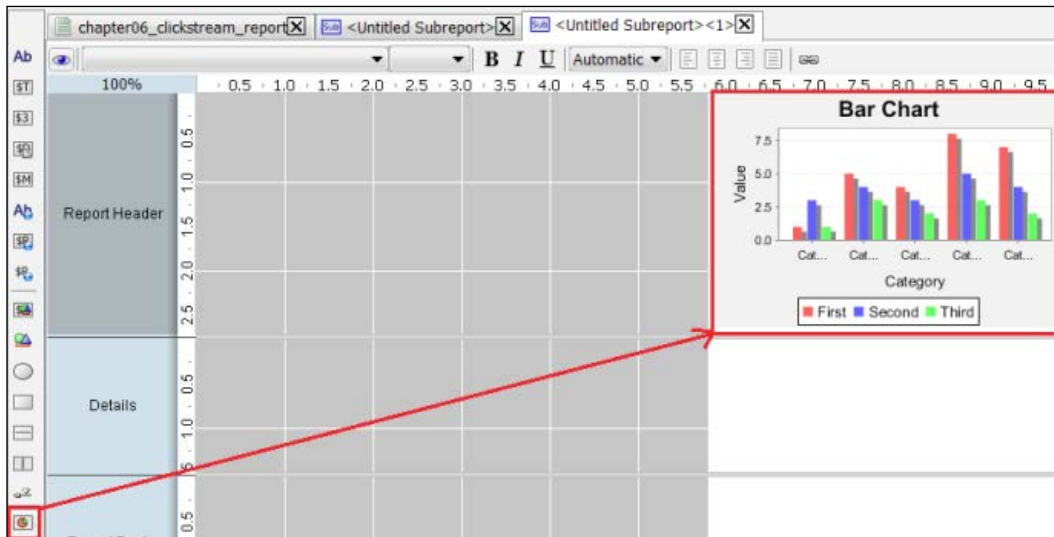
You can resize both the chart and subreport elements to achieve the appropriate chart location, alignment, and sizing. It takes a little practice to get used to how the subreport sizing impacts the chart and vice versa.

Creating a pie chart

The following steps in this exercise will guide you through adding a pie chart that shows the count of events by the web browser type:

1. Open the chapter07_clickstream_report PRD file that you saved in the previous section.
2. Click on the subreport icon from the palette on the left-hand side and drag it directly below the Purchases metric on the **Details** band.

3. Select **Inline** when prompted to choose a subreport type.
4. Select `pie_chart_query` when prompted by the **Select Data Source** dialog box.
5. After choosing a query, a third tab, **<Untitled Subreport><1>**, will open. Click on the chart icon from the palette on the left-hand side and drag it onto the new subreport's **Report Header** band, as shown in the following screenshot:

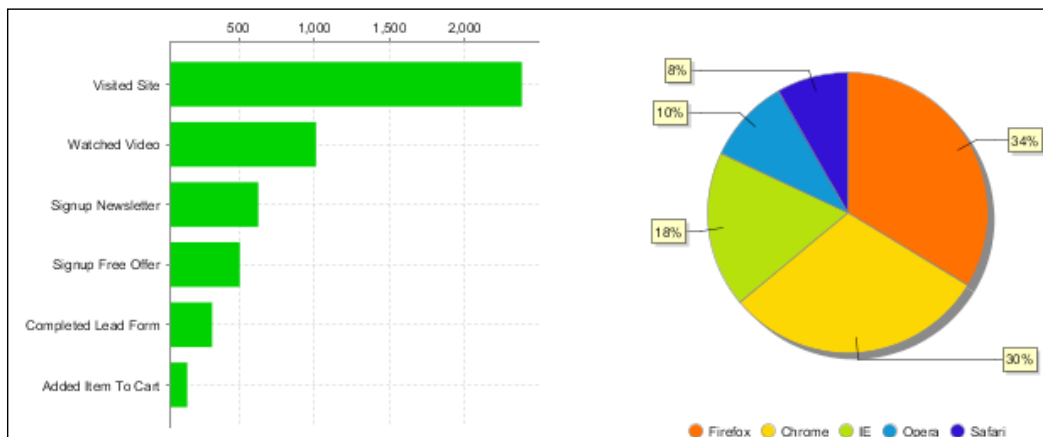


6. Double-click on the **Bar Chart** element you just added and the **Edit Chart** dialog box will appear.
7. Select **Pie Chart** from the chart type menu.
8. The **Primary DataSource** tab contains two **Pie DataSet Collector** properties, `value-column` and `series-by-field`, to assign to query fields.
9. Select the **Value** field for `value-column` and choose the `event_count` field from the drop-down menu.
10. Select the **Value** field for `series-by-field` and then click on the edit button to open the **Edit Array** dialog box.
11. Select `browser` from the **Available Items** section and click on the add items arrow to add it to the **Selected Items** section.

12. Click on **OK** and you will see a list of **Pie DataSet Collector** properties similar to the following screenshot:

Pie DataSet Collector (Chart Data)	
Name	Value
Common	
value-column	event_count
Series	
series-by-field	[browser]

13. The **Pie Chart (Image)** section contains the `legend-border` property removing the pie chart legend border. Scroll to the **Legend** properties category, select the **Value** field for the `legend-border` property, and then select **False** from the drop-down menu.
14. Click on **OK** again to close the **Edit Chart** dialog and return to your report.
15. Click on the eye icon on the tab toolbar to preview your report with the new bar chart. You should see two chart images similar the following screenshot:



16. Click on the save icon to save your work.

Creating a report prompt

Prompts and parameterization can be used to create more flexible and dynamic PRD reports. Report prompts allow a user to filter report data based on values selected or entered in the prompt display. Query parameters accept the prompt values selected by the user and use those values to filter the report data. PRD offers different prompt display types including: value lists, radio buttons, checkboxes, selection buttons, textbox, textarea, and date picker. Prompt values can be driven by static tables or dynamic queries.

Our clickstream data contains the `referring_url` field with values for three referring websites: `www.123.com`, `www.abc.com`, and `www.xyz.com`. We want to give users the ability to filter the clickstream report by the referring website. In this section, we enhance our existing report by adding a list prompt sourced by a MongoDB query on `referring_url`.

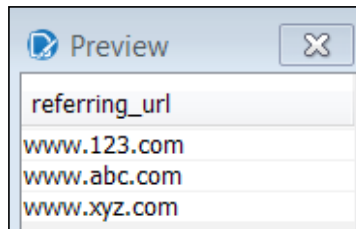
Creating a new parameter

The first step is to define a new parameter. Parameters are defined in the **Parameters** section of the **Data** tab. When you add a new parameter, the **Add Parameter** dialog box opens with options for defining a parameter, creating a prompt query, and configuring the prompt display. Perform the following steps to create a new parameter:


1. Make sure you are on the **Master Report** tab, and click on the **Data** tab and scroll to the **Parameters** section at the bottom.
2. Right-click on **Parameters** and select **Add Parameter**.
3. Once in the **Add Parameter** dialog box, click on the green circle with a plus symbol and select **MongoDB** from the list. This will launch the **MongoDB Data Source** dialog box.
4. Click on the green circle with a plus symbol to add a new query.
5. Rename **Query 1** to `referringurl_prompt`.
6. We now need to complete the four data source tabs starting with the **Configure connection** tab. Type `localhost` for **Host name** and `27017` for **Port**.
7. On the **Input Options** tab, click on **Get DBs** and select **pentaho** from the list of databases.
8. Click on **Get collections** and select `sessions_events` from the list of collections.

9. On the **Query** tab, type or paste the following query into the **Query expression** input box. This query is located in the code files provided with this book so that you can copy and paste the code:

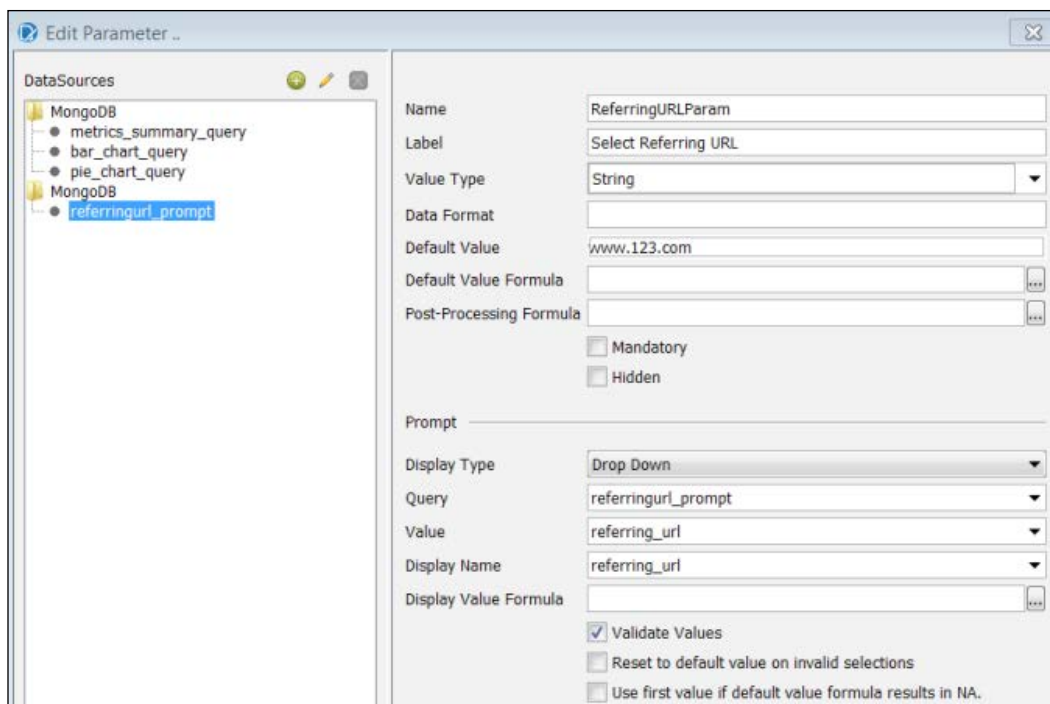
```
{ $group: { _id: "$referring_url" } },  
{ $sort: { _id: 1 } }
```
10. Be sure to check the **Query is aggregation pipeline** option.
11. On the **Fields** tab, click on **Get fields**. Pentaho will scan the `sessions_events` collection to detect the collection schema and parse the document fields defined by the query.
12. In the **Name** column, rename `_id` to `referring_url`.
13. Click on the **Preview** button to run the query and view the results. Your query results should match the numbers given in the following screenshot:



14. Close the **Preview** box and click on **OK** to return to the workspace.

 On the **Data** tab, notice that the **MongoDB** dataset now contains four queries.

15. Select the new `referringurl_prompt` query from the list.
16. Complete the parameter and prompt fields as follows:
 1. In the **Name** field, enter `ReferringURLParam`. This is the parameter name that will be used to filter the report queries.
 2. In the **Label** field, enter `Select a Referring URL:.`
 3. In the **Default Value** field, enter `www.123.com`.
 4. In the **Display Type** field, select **Drop Down** from the list.
17. Compare your **Edit Parameter** dialog box to the one shown in the following screenshot:



Adding parameters to existing report queries

Now that you have defined a parameter and prompt display, the next step is to modify the three existing report queries by adding the `ReferringURLParam` parameter to each query. Perform the following steps to add parameters to existing report queries:

1. Click on the **Data** tab, right-click on the existing **MongoDB** dataset, and select **Edit DataSource** from the list.
2. While in the **MongoDB Data Source** dialog box, highlight the first query, `metrics_summary_query`, and click on the **Query** tab to edit the query.
3. On the **Query** tab, add only the highlighted line of code shown in the following code to the top row of the existing query. Even though the syntax for PRD parameterization looks similar to the MongoDB syntax, the two are unrelated.

```
{ $match : {referring_url : "${ReferringURLParam}"},
  { $unwind : "$event_data" },
  { $group : { _id : "$browser", event_count : { $sum : 1 } } },
  {$sort:{event_count: -1}}
```

4. While still in the **MongoDB Data Source** dialog box, highlight the second query, `bar_chart_query`, and click on the **Query** tab to edit the query.
5. On the **Query** tab, add the same highlighted line of code shown as follows to the top row of the existing query:

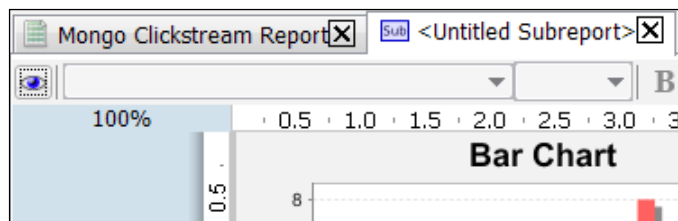
```
{ $match : {referring_url : "${ReferringURLParam}" } },  
{ $unwind : "$event_data" },  
{ $group : { _id : "$event_data.event", event_count : { $sum : 1 } } },  
{ $sort : { event_count : -1 } }
```
6. Highlight the third query, `pie_chart_query`, and click on the **Query** tab to edit the query.
7. On the **Query** tab, add the same highlighted line of code to the top row of the existing query:

```
{ $match : {referring_url : "${ReferringURLParam}" } },  
{ $unwind : "$event_data" },  
{ $group : { _id : "$browser", event_count : { $sum : 1 } } },  
{ $sort : { event_count : -1 } }
```
8. Click on **OK** to close the **MongoDB Data Source** dialog box and return to the report.

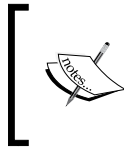
Creating subreport import parameters

The subreport queries contain parameters for the referring URL, and the prompt control on the Master Report will need to pass the referring URL into each of the subreport for the charts to function properly. We define import parameters to handle the passing of parameters from parent reports to subreports. The following steps have you define an import parameter in each subreport to handle the passing of the referring URL from master to subreport:

1. Make sure you are on the bar chart subreport tab by double-clicking on the bar chart image on the Master Report to open the associated subreport, as shown in the following screenshot:



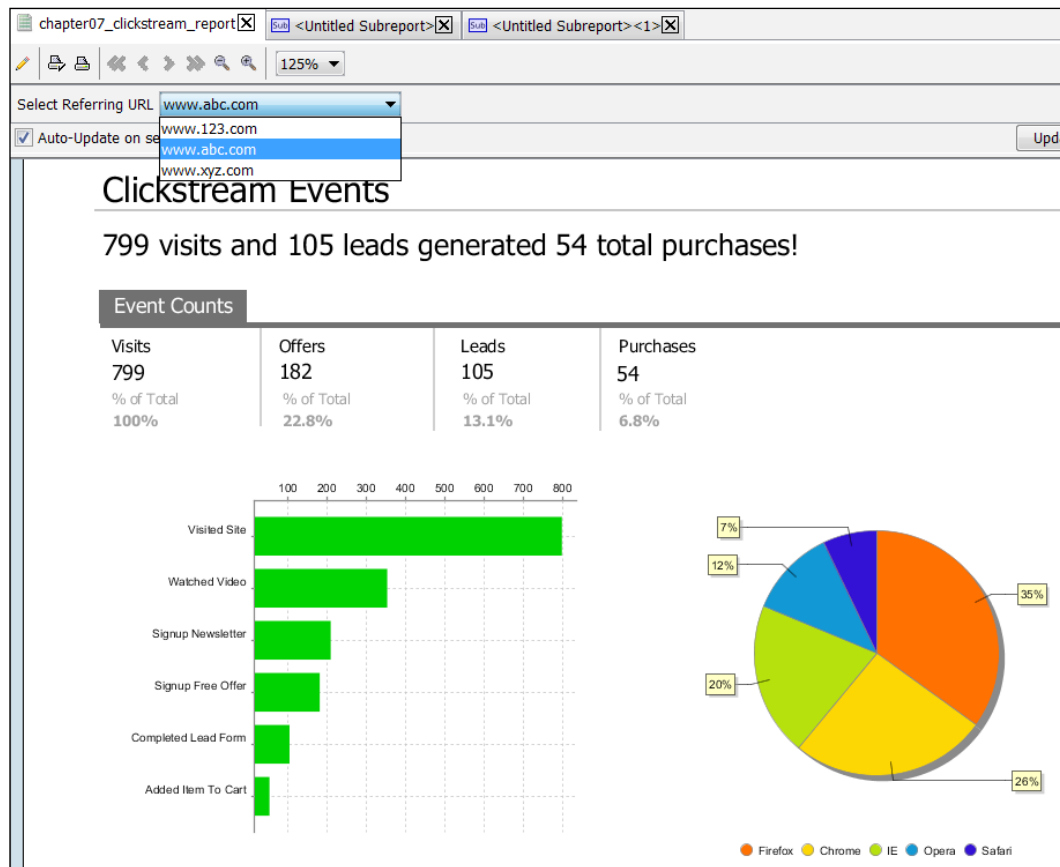
2. Click on the **Data** tab and scroll to the **Parameters** section at the bottom.
3. Right-click on **Parameters** and select **Edit Subreport Parameters**.
4. Once in the **Subreport Parameters** dialog box, you will see the **Import Parameters** and **Export Parameters** sections.
5. Click on the green circle with a plus symbol in the **Import Parameter** section, and a drop-down list will appear under each column.



The **Outer Name** column refers to the name this parameter is going to be exported as, while the **Inner Name** column refers to the name of the property in the Master Report.

6. Select `ReferringURLParam` from the dropdown for both the **Outer Name** and **Inner Name** columns.
7. Click on **OK** to return to the **Subreport** tab.
8. Select the pie chart subreport to create an identical import parameter.
9. Right-click on **Parameters** and select **Edit Subreport Parameters**.
10. Click on the green circle with a plus symbol in the **Import Parameter** section.
11. Select `ReferringURLParam` from the dropdown for both the **Outer Name** and **Inner Name** columns.
12. Click on **OK** to return to the **Subreport** tab so that you can test the report.
13. Click on the eye icon on the tab toolbar to preview your report.
14. You will see the new prompt at the top of your report with a label that reads, **Select Referring URL**. From the prompt drop-down list, select `www.abc.com`.

15. Your report should run for `www.abc.com` and match the numbers shown in the following screenshot:



Summary

The result of your hard work over the last two chapters is a PRD dashboard-style report that queries data from MongoDB. Your finished report contains four queries, four summary metrics, two charts, and a prompt; all of which were developed in PRD, a thick-client report authoring tool. Now your report is ready to be shared with and used by others who may not have PRD installed on their computers. Pentaho makes it easy to share your PRD report by publishing it to the Pentaho BA server so that users can run the report in a web browser. The next chapter is all about web-enablement and how to get the content you developed throughout this book into a web browser.

8

Deploying Pentaho Analytics to the Web

The previous chapters show users how to build reports and analysis views using Pentaho Instaview and Report Designer, both of which are thick-client components. This final chapter is all about web-enabling your MongoDB data using Pentaho methods and web interfaces for connecting to and modeling and analyzing our sample clickstream data in a web browser.

The following are the topics that we will cover in this chapter:

- Publishing a Report Designer report to the Web
- An introduction to the Pentaho User Console
- Enabling your Instaview output for the Web
- Using the Data Source Wizard to model your data
- Creating Analyzer Views and Dashboard Designer dashboards

This chapter uses the Pentaho web interfaces from the Pentaho BA server installation you completed in *Chapter 1, Getting Started with Pentaho and MongoDB*. By the end of this chapter, you will be familiar with the web-based capabilities of the Pentaho platform and will have a couple of web-enabled analysis views combined into a single dashboard.


Publishing a Report Designer report to the Web

Pentaho makes it easy to publish a PRD report to the **Business Analytics (BA)** server and give users web access to the report. You simply need to choose the location for the report and the default output format, and your published report definition gets stored on the BA server in the central database repository for access via a web browser. Once published, PRD reports can be run live or scheduled for consumption over the Web or via e-mail.


Publishing the clickstream report

The following steps in this exercise guide you through publishing your existing PRD report to the BA server, which allows users to run and schedule:

1. Click on the open icon on the menu toolbar or navigate to **File | Open** from the main menu and browse to the `chapter07_clickstream_report.prpt` file that you created in *Chapter 7, Pentaho Report Designer Prompting and Charting*.
2. From the main menu, navigate to **File | Publish...**, and the **Login** dialog box will appear.

[ As an alternative to the file menu publish option, there is also a handy Publish button on the main toolbar to the right-hand side of the save document icon.]

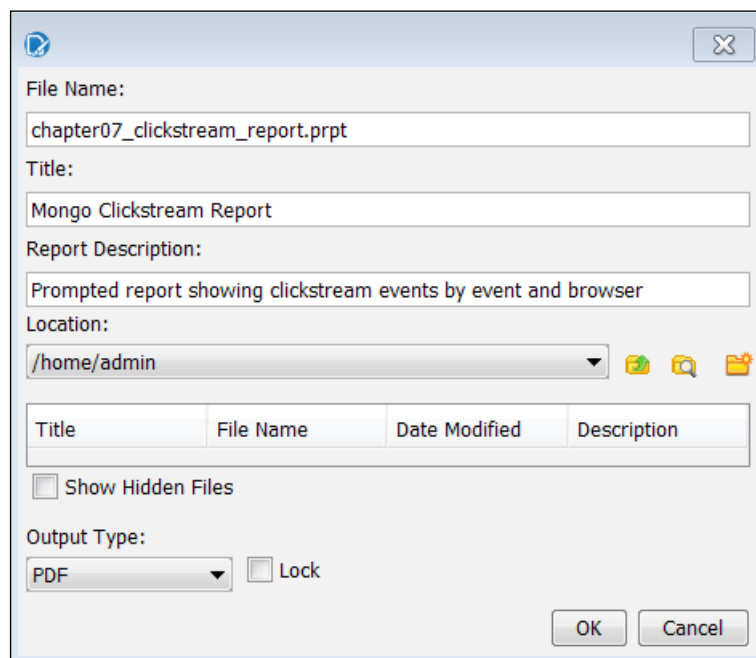
3. Confirm that the **URL** is `http://localhost:8080/pentaho` and the **Version** is Pentaho BI-Server 5.0.

[ This URL is the default installation path to launch the web-based Pentaho User Console. Port 8080 is the default BA Server Tomcat web server start-up port that was created in *Chapter 1, Getting Started with Pentaho and MongoDB*.]

4. For the login information, enter admin for **User:**, password for **Password:**, and check **Remember These Settings**.
5. Click on **OK** to log in.

6. The **Publish** dialog box will appear. Enter the following information on the screen:
 - **Title:** Mongo Clickstream Report
 - **Report Description:** Prompted report showing clickstream events by event and browser
 - **Location:** /home/admin
 - **Output Type:** PDF

The following screenshot shows the entered values:



7. Click on **OK** to publish the report.
8. Select **Yes** when prompted to launch the report. The report viewer will launch the report in the PDF format inside your default web browser. Adobe Acrobat Reader is required to view PDF files.

An introduction to the Pentaho User Console

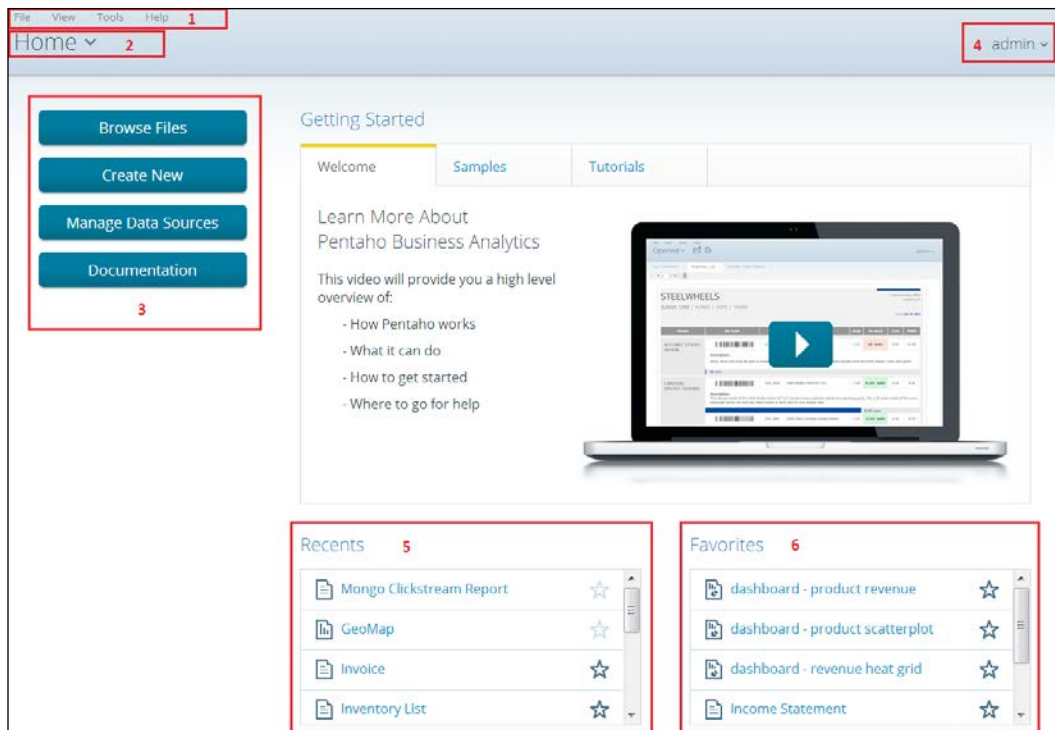
Many traditional web applications are menu-driven, with endless options to expand and navigate, making it easy to get lost in the menus and lack of perspective. Getting things done can be difficult and is intimidating for business users that crave a simple and intuitive web experience. Pentaho decided there was a better way, and built a new, modern web interface with Version 5.0. They took a successful navigation design, which started with Pentaho Mobile, and extended it into the new **Pentaho User Console (PUC)**.

PUC is a secure, web-based interface to the BA server. It is the single location for all users to consume reports and analyze data. PUC gives power users a web-based design environment to create new analysis, interactive reports, and dashboards. In addition to the design interfaces, PUC also provides administrators with server configuration, security, and schedule management features.

PUC exposes the following web-based tools and utilities:

- Analyzer
- Interactive Reporting
- Dashboard Designer
- Data Source Wizard
- Data Source Management
- Report Schedule Management
- BA Server Administration

The following screenshot highlights the six main sections of PUC:



The following table provides the section name for each of the numbered squares in the preceding screenshot:

Section #	Section name
1	Main menu
2	Perspectives
3	Home perspective buttons
4	Current login username
5	Current user's recent activity
6	Current user's favorite reports

When you log in to PUC, your initial, default page is the **Home** perspective. The **Home** perspective provides quick access to common actions, recently viewed content, and your favorite content. The **Getting Started** section at the top is a place for new users to browse sample reports, analysis views, dashboards, and to watch video tutorials.

In addition to the **Home** perspective, there are four other perspectives in PUC including: **Browse Files**, **Opened**, **Schedules**, and **Administration**. Perspectives focus the screen on the task at hand and remove the distraction of having too many objects on your page. For example, you can have multiple reports open and accessible in the **Open** perspective, while simultaneously browsing files in the **Browse** perspective.

Information consumers frequently browse for report files and perform report actions on those files, such as cut-copy-paste, report execution, and report scheduling. The **Browse Files** button, as shown in the following screenshot, launches the browse files perspective and focuses the screen on browsing for directories and a series of file actions. The **Create New** button is the entry point into the web-based design tools by giving you the choice to launch into Analyzer, Interactive Reporting, and Dashboard Designer.




Running and scheduling the clickstream report


The following steps will have you browse to and open your recently published clickstream report and create a personal report schedule:

1. Launch a web browser and enter the URL `http://localhost:8080/pentaho` to launch PUC.
2. Select **Login as an Evaluator** and click on the **GO** button.
3. After PUC loads, click on the **Browse Files** button and navigate to **Home | admin**.
4. Click on **Mongo Clickstream Report** to highlight it and notice the **File Actions** section that appears to the right.
5. From the list of **File Actions**, select **Open** to run the report. The report executes and renders a PDF report on a new tab in the **Opened** perspective. The report will run automatically when you select different prompt values.


6. After reviewing your report, close the report by clicking on the **X** icon on the report tab. This returns you to the **Home** perspective.
7. Click on **Browse Files** again, and Pentaho remembers your previous browsing session and returns you to the `admin` folder with the clickstream report automatically highlighted.
8. From the list of **File Actions**, select **Schedule**. The **Schedule** wizard appears to assist in creating a personal report schedule for the clickstream report.
9. Under **Schedule Name** type Clickstream Report.

 Generated Content Location is the directory where your generated report will reside. You will notice that your home directory, `/home/admin`, is selected by default.

10. Select the **Next** button twice to get to the **Parameters** step.
11. Select `www.abc.com` from the **Select Referring URL** drop-down list.
12. Select **PDF** from the **Output Type** drop-down list and then click on **OK**.

 Within the **PUC Administration** perspective, you can configure Pentaho to communicate with a mail server. If you have a mail server configured, the next step in the **Schedule** wizard offers the ability to set up an e-mail distribution list for this report schedule; however, because it is not configured, the wizard will end and prompt you to view your newly created schedule in the **Schedules** perspective.

13. Select **Yes** when prompted to view your list of schedules. PUC will launch you into the **Schedules** perspective to manage your personal report schedules.
14. In the **Schedule Name** column click on the **Clickstream Report** row to highlight it.

 A set of toolbar icons at the top-right corner of your screen allows you to execute, edit, or delete the highlighted report schedule:



Keep in mind that when you create a recurring schedule, the server will continue to run that schedule into the future. Each time the report schedule runs, the report's output is loaded to the BA server repository. Over time, the report output files can grow large and consume too much unnecessary space in the repository. It is a good idea to delete any schedules you create for practice. To delete the **Clickstream Report**, select it from the list of schedules and click on the blue **X** icon to delete it. Additionally, the **Administration** perspective allows administrators to schedule the deletion of report output files that are older than a specified number of days.

Enabling your Instaview output for the Web

Instaview is a client-based analysis tool for single-user, agile development and analysis. Pentaho Analyzer is the OLAP analysis component embedded inside of Instaview. Analyzer works and performs well on MongoDB data because of Instaview's in-memory data cache mechanism. However, this local cache is not accessible to the BA server, so to enable web-based OLAP analysis on MongoDB data, we need to load the appropriate MongoDB data into a JDBC-compliant relational or analytical database.

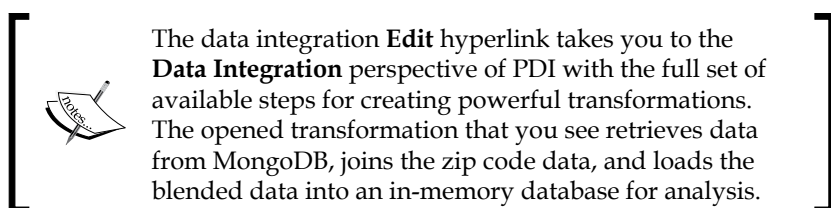
If you remember, in *Chapter 3, Using Pentaho Instaview*, and *Chapter 4, Modifying and Enhancing Instaview Transformations*, Instaview automatically generated a data integration transformation, and we enhanced it to blend geographic data with MongoDB clickstream data. We want to leverage this valuable data integration work to avoid creating a new transformation from scratch. This section shows how Instaview can enable an agile development and analysis environment for users to quickly create initial analytical models that serve as a "head start" for production deployments to the Web.

We learned in *Chapter 4, Modifying and Enhancing Instaview Transformations*, that the Instaview build process creates a PDI transformation file to read MongoDB data, transform the data, and load the data to an in-memory cache for analysis. Instaview also gives users the ability to modify the underlying PDI transformation that is automatically generated during the initial data load process. In this section, we create a copy of that transformation and modify the output step to load to a PostgreSQL database that ships and installs automatically during the Pentaho graphical installer process.

Copying and modifying the Instaview transformation

We need to reopen our Instaview file to get access to its underlying PDI transformation. To open a saved Instaview, launch Instaview, and on the **Welcome** screen you will see an **Open Existing** button. Click on **Open Existing** and choose the **Clickstream** Instaview you created.

1. Once in Instaview, a PDI transformation can be edited or copied by clicking on **Edit** in the **Data Integration** section of Instaview.



2. From the **PDI** main menu, select **File | Save as...** and save the transformation as `t_postgres_load` in the `Chapter 08` directory.
3. Delete the **Output** step on the canvas by right-clicking on it and selecting the **Delete** step from the menu. This deletes the **Output** step and the hop connected to it.
4. Under the **Design** tab, type `table` into the **Steps** search box and find the **Table output** step from the list of steps.
5. Select and drag the **Table output** step onto the canvas.
6. Select the **Do Not Edit** and **Table output** steps using the Lasso tool to highlight them both.
7. Right-click on one of the highlighted steps and select **New Hop** from the menu.
8. Make sure the **From step** is set to **Do Not Edit** and the **To step** is set to **Table output**, and then click on **OK**.
9. Double-click on the **Table output** step to edit the properties.
10. Create a new PostgreSQL database connection by clicking on the **New** button for **Connection**. This will open the **Database Connection** box that is used to configure your database connection.
11. In the **Database Connection** dialog box, fill in the information as shown:
 - **Connection Name:** `pentaho_chapter08`
 - **Connection Type:** `PostgreSQL`
 - **Host Name:** `localhost`

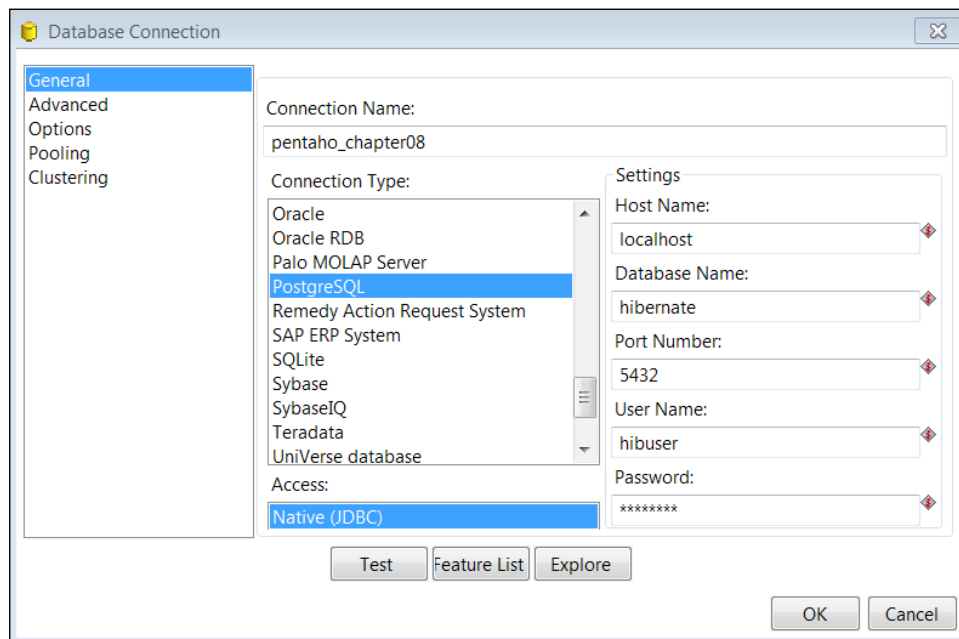
- **Database Name:** hibernate
- **Port Number:** 5432
- **User Name:** hibuser
- **Password:** password



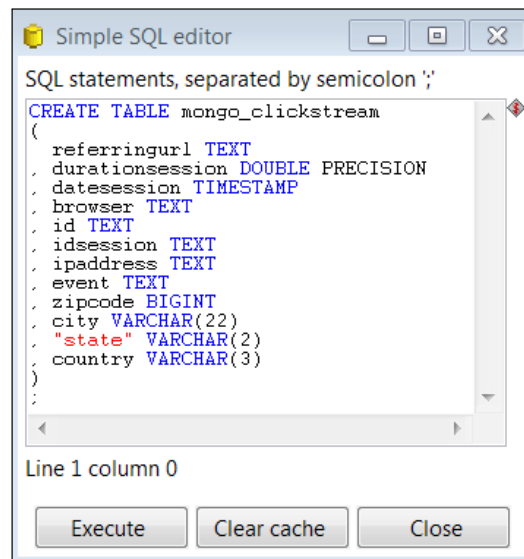
The default port number for PostgreSQL is 5432. However, if PostgreSQL was already installed on your machine prior to the installation of Pentaho, you will have two instances of PostgreSQL running, and the Pentaho version will run by default on the next available number, port 5433.

The **hibernate** database is one of the three Pentaho BA repository databases created in PostgreSQL by Pentaho. Hibernate holds data related to audit logging. We use this database for this training exercise to reduce setup time and complexity; however, it is not recommended to use any of the BA server repository databases for storing external reporting data. Additionally, you are welcome to modify the target database connection to load to other relational database platforms or databases.

12. Your **Database Connection** dialog box should match the following screenshot:

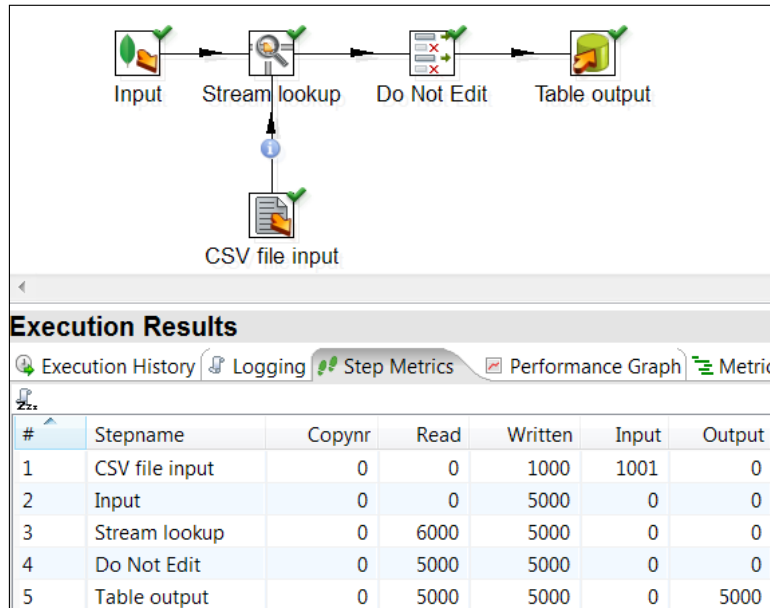


13. Click on the **Test** button and confirm that the test results show Connection to database [pentaho_chapter08] is OK.
14. Click on **OK** twice to return to the **Table output** properties dialog.
15. For **Target table** name enter mongo_clickstream.
16. Select **Truncate table**.
17. Click on the **SQL** button at the bottom of the dialog box to generate the mongo_clickstream table in PostgreSQL. The **Simple SQL editor** dialog will launch and show the generated SQL as follows:



18. Click on the **Execute** button, and the results appear with a confirmation of the SQL executed on the PostgreSQL database.
19. Click on **OK** to close the results page, and then click on **Close** to return to the **Table output** properties dialog box.
20. Click on **OK** to return to the main canvas.
21. Click on the save icon to save your work.
22. Run your transformation by navigating to **Action | Run** from the main menu and then clicking on the **Launch** button at the bottom of the **Execute a transformation** dialog box.

23. A successful transformation execution loads 5000 blended clickstream records to PostgreSQL, as shown in the following screenshot:



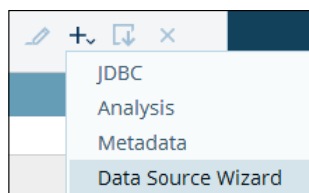
Using the Data Source Wizard to model your data

Pentaho provides a web-based Data Source Wizard in PUC to enable business users to quickly create data sources for analysis and interactive reporting. The Data Source Wizard guides you through the process of creating a connection to CSV files and JDBC-compliant data sources and creating cube and relational metadata models. The Data Source Wizard is accessed from the **Manage Data Sources** button on the **Home** perspective. Now that we have successfully loaded the blended clickstream dataset into a PostgreSQL database, we can use the wizard to access it from a browser.

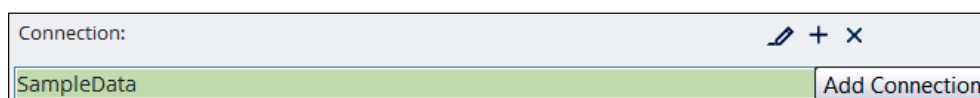
Creating a JDBC connection and default metadata model

In the next exercise, we create a JDBC connection to the hibernate database and model metadata using the Data Source Wizard:

1. Launch a web browser and enter the URL `http://localhost:8080/pentaho` to launch PUC.
2. Select **Login as an Evaluator** and **Administrator**, and then click on the **GO** button.
3. After PUC loads, click on the **Manage Data Sources** button.
4. Click on the plus icon to expand a list of data sources and select **Data Source Wizard**, as shown in the following screenshot:



5. **Select Source Type** is the first step in defining a data source. **Data Source Name** is the name that end users will see when creating new reports. Enter `pentaho_chapter08` for **Data Source Name** and select **Database Table(s)** for **Source Type**.
6. The data source name is always associated with a database connection. Click on the plus icon to define the connection:

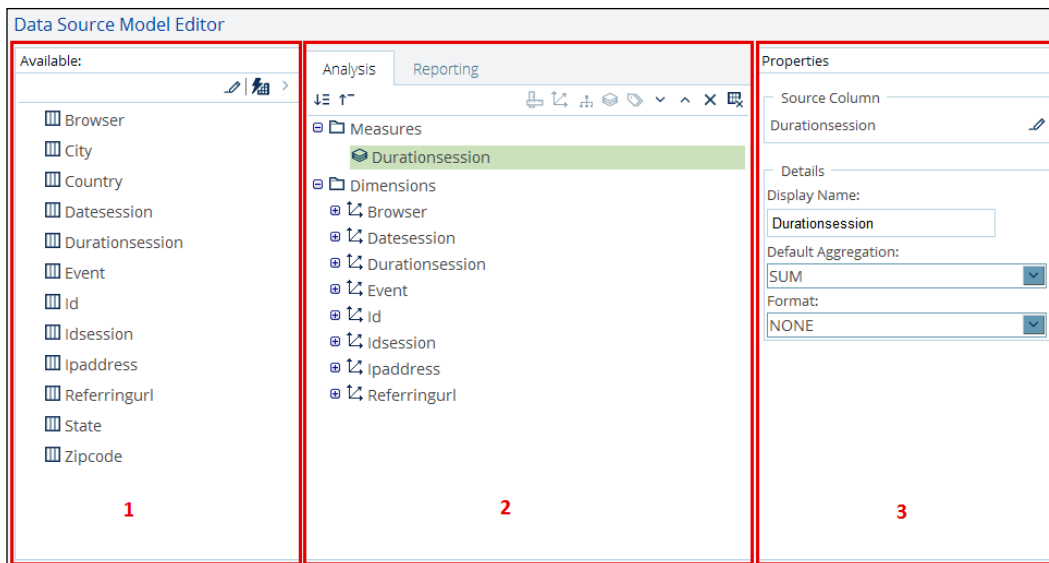


7. In the **Database Connection** dialog box, enter the following information:
 - **Connection Name:** `pentaho_chapter08_postgres`
 - **Host Name:** `localhost`
 - **Database Type:** `PostgreSQL`

- **Database Name:** hibernate
 - **Port Number:** 5432
 - **User Name:** hibuser
 - **Password:** password
8. Click on the **Test** button and confirm that the test results show `Connection to database [pentaho_chapter08_postgres]` is OK.
 9. Click on **OK** twice to return to the **Data Source Wizard**. The new `pentaho_chapter08_postgres` data source will be highlighted in the **Connection** list.
 10. In the **Create data source for** section at the bottom of the dialog box, select **Reporting and Analysis**.
 11. Click on the **Next** button to proceed to the next step.
 12. For the **Select Tables** step, click on the **Schema** dropdown and select **public**.
 13. Select `"public"."mongo_clickstream"` from the updated list of **Available Tables** and then click on the right arrow icon to add it to the **Selected Tables** list.
 14. Select `"public"."mongo_clickstream"` from the updated **Fact Table** list below.
 15. Click on the **Finish** button to create the data source and default metadata model.
 16. When prompted to customize the model, select **Customize model now** and then click on **OK**.

Customizing the metadata model

At this point, you should be at the **Data Source Model Editor** step of the **Data Source Wizard**. The model editor has three main sections for defining your model. You can drag-and-drop fields from the **Available fields** section on the left-hand side to the **Analysis** and **Reporting** modeling tabs section in the middle. You can also drag-and-drop existing model elements within the **Metadata Models** section to rearrange and design analysis and reporting models. The following screenshot highlights the three main sections of the **Data Source Model Editor**:



The following table provides the section name for each of the numbered squares in the previous screenshot:

Section #	Section name
1	Available data source fields
2	Analysis and reporting metadata models
3	Model element properties

The steps of the following exercise will guide you through the process of making changes to the default cube model:

1. On the **Reporting** tab, highlight the **Durationsession** measure, and then select a **Default Aggregation** of **AVERAGE** in the **Properties** section.
2. On the **Analysis** tab, highlight the **Durationsession** dimension and click on the **X** icon to delete it.
3. On the **Analysis** tab, highlight the **Id** dimension and click on the **X** icon to delete it.
4. On the **Analysis** tab, highlight the **Idsession** dimension and click on the **X** icon to delete it.

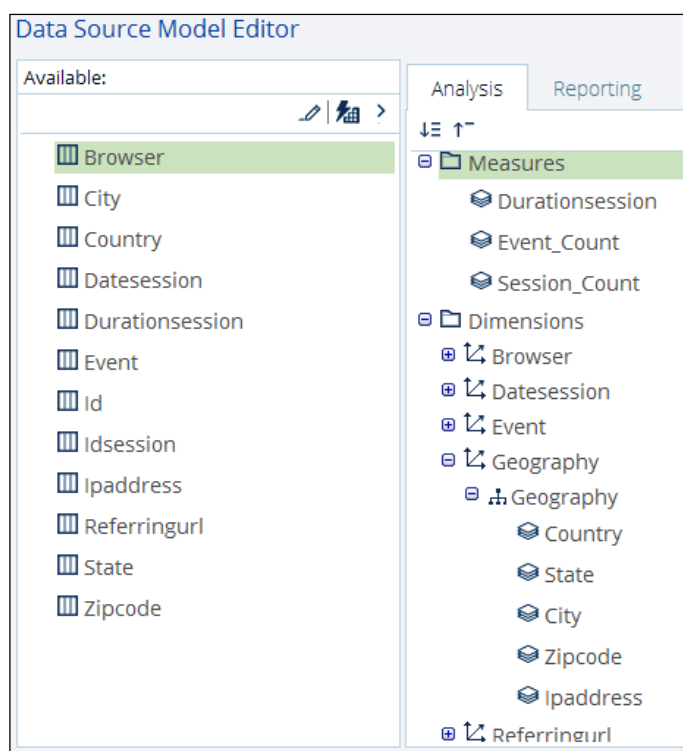
5. On the **Analysis** tab, highlight the **IpAddress** dimension, and in the **Properties** section's **Dimension Name** entry box, rename **IpAddress** to **Geography**.
6. Click on the **Geography** dimension's plus icon (+) to expand to the nested **IpAddress** hierarchy.
7. Select the **IpAddress** hierarchy and rename **IpAddress** to **Geography**.
8. Click on the **Geography** hierarchy plus icon (+) to expand to the nested **IpAddress** level.
9. Select and drag **Country** from the **Available** section to the precise area directly above the **IpAddress** dimension level, as shown in the following screenshot:



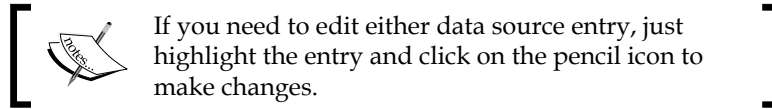
You can add fields as new hierarchies or new levels within an existing dimension. When dragging-and-dropping fields into a dimension as a new level, it is important to have the horizontal blue line appear directly over the level name and indented exactly, as shown in the previous screenshot. To have a field added as a new hierarchy instead of a level, the blue line should appear less indented.

10. With the **Country** level highlighted, change the value of **Geography Type** on the **Properties** section to **Country**.
11. Select and drag **State** from the **Available** section to the precise area directly above the **IpAddress** dimension level.
12. With the **State** level highlighted, change the value of **Geography Type** on the **Properties** section to **State**.
13. Select and drag **City** from the **Available** section to the precise area directly above the **IpAddress** dimension level.

14. With the `City` level highlighted, change the value of **Geography Type** on the **Properties** section to `City`.
15. Select and drag `Zipcode` from the **Available** section to the precise area directly above the **Ipaddress** dimension level.
16. With the `Zipcode` level highlighted, change the value of **Geography Type** on the **Properties** section to `Postal Code`.
17. To create a new measure to count events, select and drag **Event** from the **Available** section to the `Measures` folder on the **Analysis** tab.
18. With the **Event** measure highlighted, change the value of **Display Name** on the **Properties** section to `Event_Count`.
19. To create a new measure for counting sessions, select and drag **Idsession** from the **Available** section to the `Measures` folder on the **Analysis** tab.
20. With the `Idsession` measure highlighted, change the value of **Display Name** on the **Properties** section to `Session_Count` and **Default Aggregation** to `COUNT_DISTINCT`.
21. Compare your analysis model changes to the following screenshot:



22. Click on **OK** to complete your customized model and return to the **Data Sources** management utility.
23. Scroll through the data source list to locate the following two new entries:
 - **JDBC:** pentaho_chapter08_postgres
 - **Data Source Wizard:** pentaho_chapter08



24. Click on the **Close** button to return to the PUC **Home** perspective.

Creating Analyzer Views and Dashboard Designer dashboards

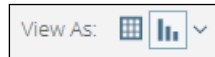
By now, you must be familiar with Pentaho Analyzer because we used it within Instaview to create visualizations against Instaview's in-memory database. The only difference here is that we can now use Analyzer within a web browser to access our blended clickstream data that was loaded to a PostgreSQL database. In this last section, we create two new Analyzer views and save them to the BA server. Once saved, the Analyzer views are assembled into a two-pane Dashboard Designer dashboard.

Creating a map view in Analyzer

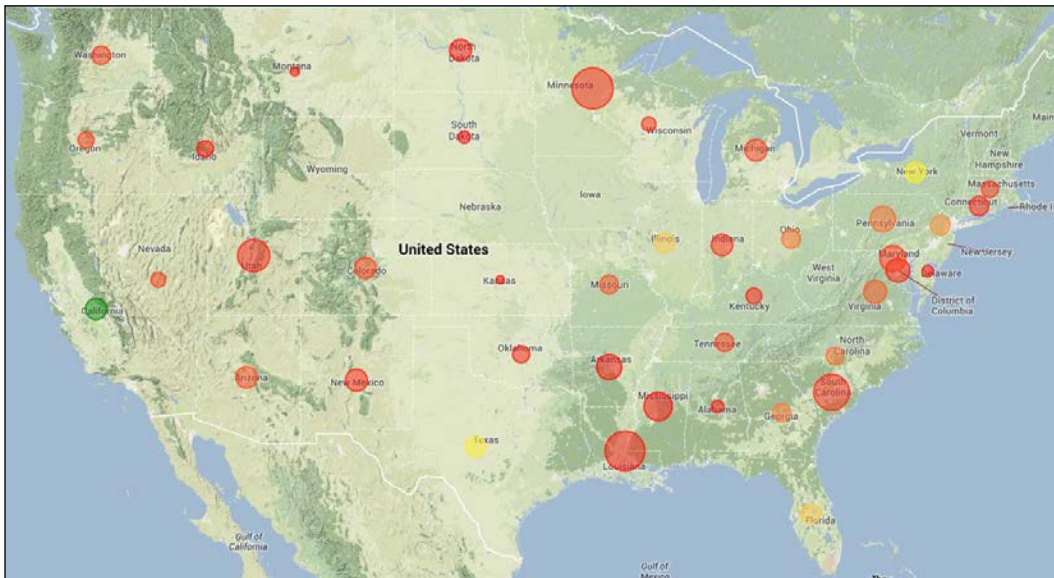
The following steps of the exercise will guide you through creating a Geo Map with Analyzer to view the number of web events and average session duration by country:

1. Make sure you are connected to the Internet so the Google mapping feature will work.
2. From the PUC **Home** perspective, click on the **Create New** button and then on the **Analysis Report** button.
3. When prompted to select a data source, scroll down the **Data Sources** list, select pentaho_chapter08: pentaho_chapter08, and then click on **OK**.
4. Drag Event_Count, to the **Measures** drop zone in the **Layout** section.
5. Drag Country to the **Rows** drop zone in the **Layout** section.

6. Drag **State** to the **Rows** drop zone in the **Layout** section.
7. In the **View As** section, click on the dropdown next to the chart icon and select **Geo Map**, as shown:



8. To filter Hawaii out of the map view, click on the red circle over **Hawaii** to highlight it and then select the **Exclude** button from the filter menu that appears at the top of the map.
9. Expand the small plus icon on the right edge of the map and select the **Google Physical** base layer.
10. Drag **Durationsession** to the **Size By** drop zone in the **Layout** section. Your resulting map should resemble the following screenshot:

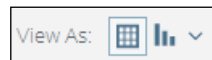


11. The resulting Google Map shown in the preceding screenshot reveals that users in California are driving the most site visits, while Minnesota and Louisiana users have the highest average session duration.
12. Click on the Save icon from the toolbar and save the view as Chapter08-GeoMap in the default /home/admin location.
13. Click on the **Save** button to return to the saved view.

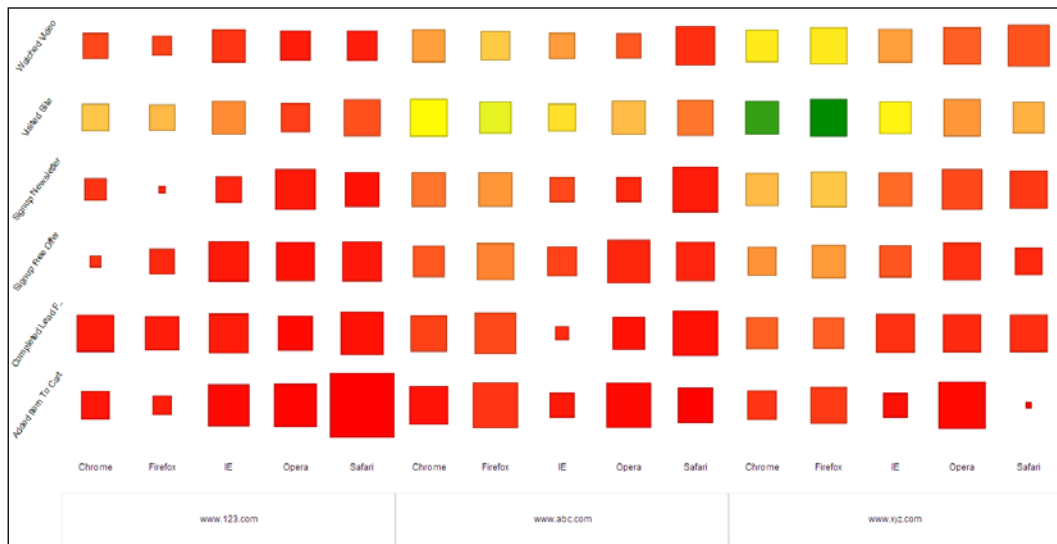
Creating a heat grid in Analyzer

The steps of this exercise will guide you through modifying the Geo Map to create a Heat Grid with Analyzer to view the number of web events and average session duration by country.

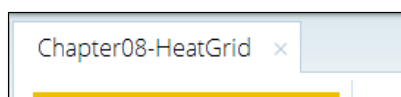
1. Make sure you still have the **Chapter08-GeoMap** view open from the previous exercise.
2. In the **View As** section, click on the table icon to return to a table view.



3. Right-click on the Country column header and select **Remove from Report**.
4. Right-click on the State column header and select **Remove from Report**.
5. In the **View As** section, click on the dropdown next to the chart icon and select **Heat Grid**.
6. Drag Referringurl to the **X-axis** drop zone in the **Layout** section.
7. Drag Browser to the **X-axis** drop zone in the **Layout** section and place it directly beneath Referringurl.
8. Drag Event to the **Y-axis** drop zone in the **Layout** section.
9. Compare your heat grid to the following screenshot:



10. The resulting heat grid shown in the previous screenshot reveals that `www.xyz.com` is driving the most site visits from Chrome and Firefox users, while `www.123.com` is resulting in the highest average session duration from Safari users.
11. Click on the **Save As** icon from the toolbar and save the view as `Chapter08-HeatGrid` in the default `/home/admin` location.
12. Click on the **Save** button to return to the saved view.
13. Close the view by clicking on the **X** icon on the **View** tab:




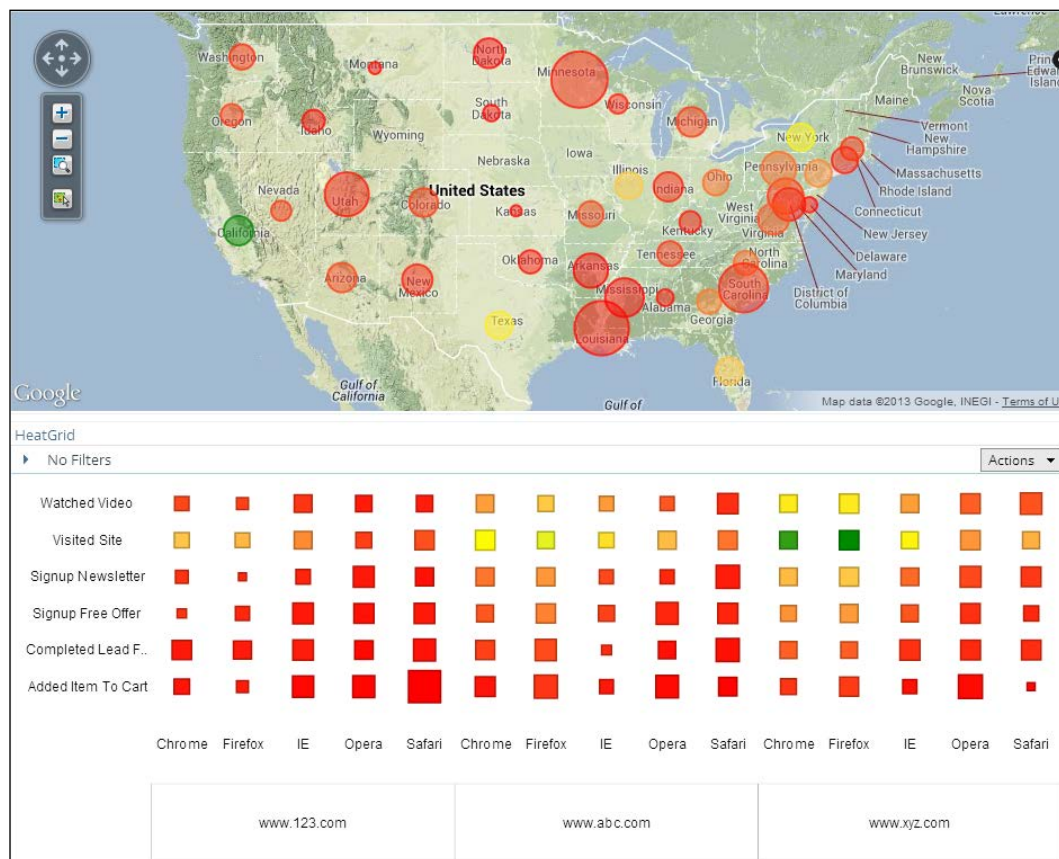
Creating a dashboard using Dashboard Designer

Dashboard Designer gives business users the power to easily assemble their own dashboards. The web-based GUI guides users through the process of choosing a layout template, style theme, and the content for display. Dashboard Designer is best suited for displaying content from Analyzer, Interactive Reporting, or Report Designer; however, it can also natively generate its own charts and data tables or link to external web pages.

In this final exercise, you create a simple dashboard using the map and heat grid from the previous two Analyzer exercises. Perform the following steps to create a dashboard using Dashboard Designer:

1. From the PUC **Home** perspective, click on the **Create New** button, and then click on the **Dashboard** button. This will launch Dashboard Designer.
2. On the **Templates** tab at the bottom of Dashboard Designer, select the **1 over 1** template consisting of two dashboard panes stacked vertically.
3. In the **Browse** section on the left-hand side of your screen, expand the `Home` folder and select the `admin` folder.
4. The two Analyzer views created in the previous exercises appear in the **Files** section below the **Browse** section. Select the `Chapter08-GeoMap` view and drag it into the top dashboard pane called **Untitled 1**.
5. In the **Title** section, rename **Untitled 1** to `GeoMap` and click on the **Apply** button.

6. Select the Chapter08-HeatGrid view and drag it into the bottom dashboard pane called **Untitled 2**.
7. In the **Title** section, rename **Untitled 2** to HeatGrid and click on the **Apply** button.
8. Click on the **Save** icon from the toolbar and save the view as Chapter08-Dashboard in the default /home/admin location.
9. In the top toolbar section, click on the pencil icon  to leave the edit mode and view your dashboard.
10. Compare your dashboard to the following screenshot:



Summary

Congratulations on finishing this book and learning how to get the most of your MongoDB data with Pentaho Business Analytics! This chapter provided an introduction to Pentaho's web-based reporting and analysis capabilities for MongoDB. You are now familiar with the interfaces for connecting to, modeling, and analyzing blended MongoDB clickstream data from a web browser.

Keep in mind that we did not cover Pentaho Metadata, Pentaho Interactive Reporting, and Pentaho Mobile, or discuss the several tools and add-ins available for Pentaho Business Analytics or MongoDB. There are also many more exciting and powerful features of MongoDB, Pentaho Analyzer, and Dashboard Designer that could not be included in this book. You are encouraged to use this book as a starting point and then continue to expand your knowledge of both Pentaho and MongoDB by exploring the product documentation online at infocenter.pentaho.com and docs.mongodb.org, and by attending classroom training for both MongoDB and Pentaho.

Index

Symbols

\$all query operator 28
\$or operator 27
\$query operator 26
\$unwind operator 35
_id field 12
<projection> argument 25
<query> argument 25
[referring_url] field 19

A

Add Parameter dialog box 95
aggregation 55
aggregation framework
 URL 75
aggregation pipeline 35
aggregation types 65
analysis view
 creating 38-41
 saving 38-41
Analyzer Views
 creating 118
Attributes tab 68, 87

B

Band element 70
bar chart
 creating 89, 90
 properties, modifying 91, 92
bar chart query
 adding 84, 85
Bar Sparkline element 70
BA server 14, 102

blended data

new analysis view, creating from 50, 51

Browse button 47

C

calculated values

adding, to report 79-81

calculation documentation, Pentaho

URL 65

Chart element 70

chart types, PRD

Area 87
Bar 87
Bar Line Combination 87
Bubble 87
Extended XY Line Chart 87
Line 87
Multi-Pie 87
Pie 87
Radar 87
Ring 87
Scatter Plot 87
Waterfall 87
XY Area 87
XY Area Line 87
XY Bar 87
XY Line 87

clickstream report

publishing 102, 103
running 106
scheduling 107

Client tools 14

collection 18

community edition (CE) 9
CSV file input 47, 48

D

Dashboard Designer dashboards
creating 118-122

data integration 45

data source

about 64
adding 45, 46

data source, adding

CSV file input 47, 48
Stream lookup 48-50

Data Source Wizard

JDBC connection, creating 113, 114
metadata model, customizing 114-118
using, for modeling data 112

Data tab 68

data visualization

charts, using 86

data visualization, with charts

bar chart, creating 89
bar chart properties, modifying 91
chart data collectors 88
JFreeChart chart types 87
pie chart, creating 92-94
subreports 87

Date Field element 69

Denormalized models 22, 23

Details band 67

dimensions

about 54
changes 58, 59
hierarchies 54
URL dimension, referring 57, 58

DI server 14

document 18

E

Edit Chart dialog box

sections 88
typical workflow 89

Edit Parameter dialog box 96

Ellipse element 70

enterprise edition (EE) 9

event count measure 57

event field 34

existing Instaview

opening 44

Expr button 80

F

find command 12

find() method 25, 26

G

Get Fields button 47

Get lookup fields button 49

Go button 15

Google stock (GOOG) 54

Group Footer band 67

Group Header band 67

H

heat grid

creating, with Analyzer 120, 121

hibernate database 110

hierarchies 54

hops 46

Horizontal Line element 70

I

id_session field 13

Image element 70

Image Field element 69

Index element 70

inline subreports 88

Instaview

about 108
accessing, to MongoDB 31, 32
connecting, to MongoDB 31, 32
creating 38-41
output, enabling for Web 108
saving 38-41
transformation, copying 109
transformation, modifying 109-111

Instaview metadata

about 53

- dimensions 54
- existing Instaview, opening 54, 55
- measures 53

interface reference 72

J

JDBC connection, Data Source Wizard
creating 113, 114

JFreeChart 86

JSON
URL 8

L

Label element 69

Line Sparkline element 70

M

main toolbar 71

map view
creating, with Analyzer 118, 119

Master Report tab 95

measures
about 53, 54
event count measure 57
session count measure 57
session duration measure 56

message field
adding, to report 76, 77

Message Field element 69

metadata model, Data Source Wizard
customizing 114

MongoDB
about 6-8
advantages 8
installing 10-12
installing, as Windows service 12
Instaview, accessing to 31, 32
Instaview, connecting to 31, 32
sample clickstream, restoring 13, 14
URL 73

MongoDB collection
parsing 33-35
profiling 33-35

MongoDB connection
creating 73

MongoDB database
URL 12

MongoDB database objects 17, 18

MongoDB data modeling
about 21
Denormalized models 22, 23
Normalized models 21, 22

MongoDB data source
adding 74, 75

MongoDB documentation
URL 23

MongoDB manual
URL 10

MongoDB queries
adding 83-86
bar chart query, adding 84, 85
pie chart query, adding 85, 86

MongoDB query expression
adding 35-37

MongoDB query methods
query exercise 1 24, 25
query operators 26, 27
read operations 25

MongoDB Data Source dialog box 98

N

New Analysis button 51

new analysis view
creating 60
creating, from blended data 50, 51

No Data band 67

Normalized models 21, 22

Number Field element 69

number-fields
adding, to report 78, 79

O

OK button 36

Open Existing button 44

OpenFormula
URL 65

Outer Name column 99

P

Page Footer band 67

Page Header band 67

palette 69

palette, element

Band element 70

Bar Sparkline element 70

Chart element 70

Date Field element 69

Ellipse element 70

Horizontal Line element 70

Image element 70

Image Field element 69

Index element 70

Label element 69

Line Sparkline element 70

Message Field element 69

Number Field element 69

Pie Sparkline element 70

Rectangle element 70

Resource Field element 69

Resource Label element 69

Resource Message element 69

Simple Barcodes element 70

Sub Report element 70

Survey Scale element 70

Table of Content element 70

Text Field element 69

Vertical Line element 70

Pentaho

about 9, 10

BA server 14

Client tools 14

DI server 14

downloading 14

installing 14, 15

Web tools 14

Pentaho Analyzer 108

Pentaho Business Analytics 9

Pentaho Data Integration (PDI)

about 9, 43

advantages 9

Pentaho Report Designer

about 83

chart types 87

JFreeChart 86

sparkline 86

Pentaho Report Designer, features

aggregation types 65

data sources 64

formatting 65

output 65

report elements 64

Pentaho Report Designer, navigating through

Attributes tab 68

Data tab 68

interface reference 72

main toolbar 71

palette 69

report workspace 66, 67

Structure tab 68

Style tab 68

tab toolbar 71

Pentaho User Console (PUC)

about 104

clickstream report, running 106

clickstream report, scheduling 107

home perspective 106

main menu 105

perspectives 105

web-based tools 104

Pentaho wiki page

URL 46

pie chart query

adding 85, 86

Pie Sparkline element 70

PRD report. *See* **Report Designer report**

Preview button 34, 36, 75, 85, 96

Primary DataSource tab 93

Q

query

creating 73

querying arrays

about 27, 28

query exercise 3 28, 29

query operators

about 26, 27

querying arrays 27, 28

Query tab 86, 97

R

read operations

about 25, 26

Query exercise 2 26

Rectangle element 70

ReferringURLParam parameter 97

relational database management system (RDBMS) 8

report

calculated values, adding to 79-81

message field, adding to 76, 77

number-fields, adding to 78, 79

Report Designer report

clickstream report, publishing 102, 103

publishing 102

report elements 64

Report Footer band 67

Report Header band 67

report prompt

creating 95

new parameter, creating 95, 96

parameters, adding to existing report

queries 97, 98

subreport import parameters,

creating 98, 99

report workspace 66, 67

Resource Field element 69

Resource Label element 69

Resource Message element 69

Run button 50

S

sample pentaho database 19

Save View button 51

session count measure 57

session duration measure 56, 57

Simple Barcodes element 70

sparkline chart 86

Stream lookup 48-50

Structure tab 68, 87

Style tab 68

subdocuments 22

Sub Report element 70

subreport import parameters

creating 98

subreports

about 87

inline subreports 88

Subreport tab 99

Survey Scale element 70

T

Table of Content element 70

tab toolbar 71

Text Field element 69

U

URL dimension

referring 57

V

value property 80

Vertical Line element 70

W

watermark band 67

Web tools 14



Thank you for buying **Pentaho Analytics for MongoDB**

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.packtpub.com.

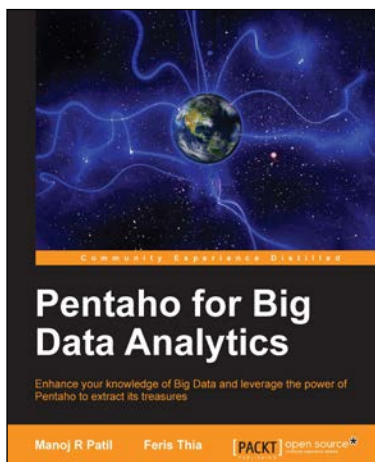
About Packt Open Source

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



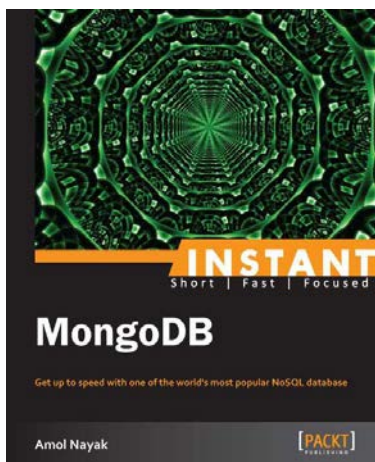
Pentaho for Big Data Analytics

ISBN: 978-1-78328-215-9

Paperback: 118 pages

Enhance your knowledge of Big Data and leverage the power of Pentaho to extract its treasures

1. A guide to using Pentaho Business Analytics for Big Data analysis.
2. Learn Pentaho's visualization and reporting tools with practical examples and tips.
3. Precise insights into churning Big Data into meaningful knowledge with Pentaho.



Instant MongoDB

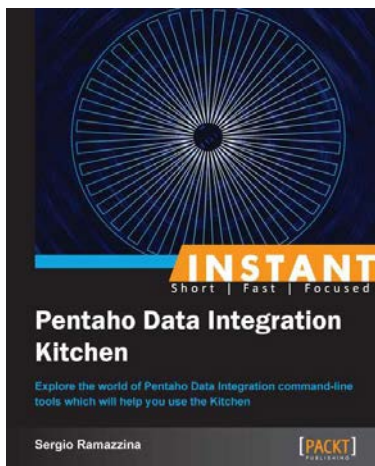
ISBN: 978-1-78216-970-3

Paperback: 72 pages

Get up to speed with one of the world's most popular NoSQL database

1. Learn something new in an instant! A short, fast, focused guide delivering immediate results.
2. Query in MongoDB from the mongo shell.
3. Learn about the aggregation framework and Map Reduce support in Mongo.
4. Tips and tricks for schema designing and how to develop high-performance applications using MongoDB.

Please check www.PacktPub.com for information on our titles



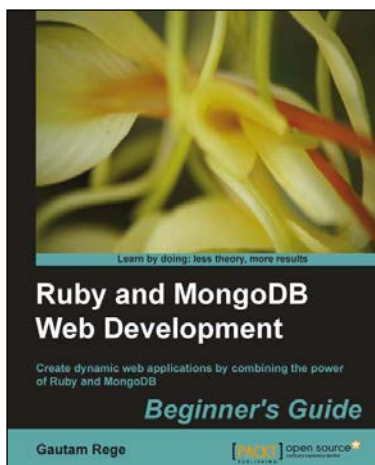
Instant Pentaho Data Integration Kitchen

ISBN: 978-1-84969-690-6

Paperback: 68 pages

Explore the world of Pentaho Data Integration command-line tools which will help you use the Kitchen

1. Learn something new in an instant! A short, fast, focused guide delivering immediate results.
2. Understand how to discover the repository structure using the command-line scripts.
3. Learn to configure the log properly and how to gather the information that helps you investigate any kind of problem.
4. Explore all the possible ways to start jobs and learn transformations without any difficulty.



Ruby and MongoDB Web Development Beginner's Guide

ISBN: 978-1-84951-502-3

Paperback: 332 pages

Create dynamic web applications by combining the power of Ruby and MongoDB

1. Step-by-step instructions and practical examples to creating web applications with Ruby and MongoDB.
2. Learn to design the object model in a NoSQL way.
3. Create objects in Ruby and map them to MongoDB.

Please check www.PacktPub.com for information on our titles