



Big Data Beyond the Hype

A Guide to Conversations for Today's Data Center

- Expand what you know about Big Data, putting you on track to go beyond the hype
 - Stay on top of all the changes, including the cloud, Hadoop-based analytics, streaming analytics, warehousing (including Big SQL), NoSQL, integration, governance, and more
 - See how IBM Watson and as-a-service models such as IBM Bluemix add essential dimensions to the Big Data story
 - Learn about the Big Data Zones Model that brings a new approach to managing data, faster to deploy...faster to insights...and with less risk
 - Gain confidence in your Big Data projects and learn about the importance of governance in a Big Data world
-

Big Data Beyond the Hype

About the Authors

Paul Zikopoulos is the Vice President of Technical Professionals for IBM's Information Management division and leads its World-Wide Competitive Database and Big Data teams. Paul is an award-winning writer and speaker with more than 20 years of experience in information management and is seen as a global expert in Big Data and Analytic technologies. Independent groups often recognize Paul as a thought leader with nominations to SAP's "Top 50 Big Data Twitter Influencers," Big Data Republic's "Most Influential," Analytica's "Top 100," and Analytics Week's "Thought Leader in Big Data and Analytics" lists. Technopedia listed him a "Big Data Expert to Follow," and he was consulted on the topic by the popular TV show *60 Minutes*. Paul has written more than 350 magazine articles and 18 books, some of which include *Hadoop for Dummies*, *Harness the Power of Big Data*, *Understanding Big Data*, *DB2 for Dummies*, and more. Paul has earned an undergraduate degree in Economics and an MBA. In his spare time, he enjoys all sorts of sporting activities, including, apparently, hot yoga. Ultimately, Paul is trying to figure out the world according to Chloë—his daughter. Follow him on Twitter @BigData_paulz.

Dirk deRoos is IBM's World-Wide Technical Sales Leader for IBM's Big Data technologies. Dirk has spent the past four years helping customers build Big Data solutions featuring InfoSphere BigInsights for Hadoop and Apache Hadoop, along with other components in IBM's Big Data and Analytics platform. Dirk has co-authored three books on this subject area: *Hadoop for Dummies*, *Harness the Power of Big Data*, and *Understanding Big Data*. Dirk earned two degrees from the University of New Brunswick in Canada: a bachelor of computer science and a bachelor of arts (honors English). You can reach him on Twitter @Dirk_deRoos.

Christopher Bienko covers the Information Management Cloud Solutions portfolio for IBM's World-Wide Technical Sales organization. Over the better part of two years, Christopher has navigated the cloud computing domain, enabling IBM customers and sellers to stay abreast of these rapidly evolving technologies. Prior to this, Christopher was polishing off his freshly minted bachelor of computer science degree from Dalhousie University. Christopher also holds a bachelor of science degree from Dalhousie University, with a

double major in biology and English. You can follow his musings on Twitter @ChrisBienko and see the world through his lens at <http://flickr.com/photos/chrisbienko/>.

Rick Buglio has been at IBM for more than eight years and is currently a product manager responsible for managing IBM's InfoSphere Optim solutions, which are an integral part of IBM's InfoSphere lifecycle management portfolio. He specializes in Optim's Data Privacy and Test Data Management services, which are used extensively to build right-sized, privatized, and trusted nonproduction environments. Prior to joining the Optim team at IBM, he was a product manager for IBM's Data Studio solution and was instrumental in bringing the product to market. Rick has more than 35 years of experience in the information technology and commercial software industry across a vast number of roles as an application programmer, business analyst, database administrator, and consultant, and has spent the last 18 years as a product manager specializing in the design, management, and delivery of successful and effective database management solutions for numerous industry-leading database management software companies.

Marc Andrews is an IBM Vice President who leads a worldwide team of industry consultants and solution architects who are helping organizations use increasing amounts of information and advanced analytics to develop industry-specific Big Data and Analytics solutions. Marc meets with business and technology leaders from multiple industries across the globe to share best practices and identify how they can take advantage of emerging Big Data and Analytics capabilities to drive new business value. Marc is a member of the IBM Industry Academy and has been involved in several multibillion-dollar acquisitions in the information management and analytics space. He has a bachelor of economics degree from the Wharton Business School at the University of Pennsylvania and holds three patents related to information integration. You can reach him at marc.andrews@us.ibm.com.

About the Technical Editor

Roman B. Melnyk is a senior member of the DB2 Information Development team. Roman edited *DB2 10.5 with BLU Acceleration: New Dynamic In-Memory Analytics for the Era of Big Data*; *Harness the Power of Big Data: The IBM Big Data*

Platform; Warp Speed, Time Travel, Big Data, and More: DB2 10 for Linux, UNIX, and Windows New Features; and Apache Derby: Off to the Races. Roman co-authored Hadoop for Dummies, DB2 Version 8: The Official Guide, DB2: The Complete Reference, DB2 Fundamentals Certification for Dummies, and DB2 for Dummies.

Big Data Beyond the Hype

A Guide to Conversations
for Today's Data Center

Excerpt

Paul Zikopoulos
Dirk deRoos
Christopher Bienko
Rick Buglio
Marc Andrews



New York Chicago San Francisco
Athens London Madrid Mexico City
Milan New Delhi Singapore Sydney Toronto

Copyright © 2015 by McGraw-Hill Education. All rights reserved. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

ISBN: 978-0-07-184466-6

MHID: 0-07-184466-X

The material in this eBook also appears in the print version of this title: ISBN: 978-0-07-184465-9,
MHID: 0-07-184465-1.

eBook conversion by codeMantra
Version 1.0

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence of a trademarked name, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

McGraw-Hill Education eBooks are available at special quantity discounts to use as premiums and sales promotions or for use in corporate training programs. To contact a representative, please visit the Contact Us page at www.mhprofessional.com.

The contents of this book represent those features that may or may not be available in the current release of any on premise or off premise products or services mentioned within this book despite what the book may say. IBM reserves the right to include or exclude any functionality mentioned in this book for the current or subsequent releases of any IBM cloud services or products mentioned in this book. Decisions to purchase any IBM software should not be made based on the features said to be available in this book. In addition, any performance claims made in this book aren't official communications by IBM; rather, they are the results observed by the authors in unaudited testing. The views expressed in this book are ultimately those of the authors and not necessarily those of IBM Corporation.

Information has been obtained by McGraw-Hill Education from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, McGraw-Hill Education, or others, McGraw-Hill Education does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from the use of such information.

TERMS OF USE

This is a copyrighted work and McGraw-Hill Education and its licensors reserve all rights in and to the work. Use of this work is subject to these terms. Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without McGraw-Hill Education's prior consent. You may use the work for your own noncommercial and personal use; any other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED "AS IS." MCGRAW-HILL EDUCATION AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DISCLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. McGraw-Hill Education and its licensors do not warrant or guarantee that the functions contained in the work will meet your requirements or that its operation will be uninterrupted or error free. Neither McGraw-Hill Education nor its licensors shall be liable to you or anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damages resulting therefrom. McGraw-Hill Education has no responsibility for the content of any information accessed through the work. Under no circumstances shall McGraw-Hill Education and/or its licensors be liable for any indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.

Book number 19: One day I'll come to my senses. The time and energy needed to write a book in your "spare time" are daunting, but if the end result is the telling of a great story (in this case, the topic of Big Data in general and the IBM Big Data and Analytics platform), I feel that it is well worth the sacrifice.

Speaking of sacrifice, it's not just the authors who pay a price to tell a story; it's also their loved ones, and I am lucky to have plenty of those people. My wife is unwavering in her support for my career and takes a backseat more often than not—she's the "behind-the-scenes" person who makes it all work: Thank you. And my sparkling daughter, Chloë, sheer inspiration behind a simple smile, who so empowers me with endless energy and excitement each and every day. Over the course of the summer, we started to talk about Hadoop. She hears so much about it and wanted to know what it is. We discussed massively parallel processing (MPP), and I gave her a challenge: Create for me an idea how MPP could help in a kid's world. After some thought, she brought a grass rake with marshmallows on each tine, put it on the BBQ, and asked "Is this MPP, BigDaDa?" in her witty voice. That's an example of the sparkling energy I'm talking about. I love you for this, Chloë—and for so much more.

I want to thank all the IBMers who I interact with daily in my quest for knowledge. I'm not born with it, I learn it—and from some of the most talented people in the world...IBMers.

Finally, to Debbie Smith, Kelly McCoy, Brad Strickert, Brant Hurren, Lindsay Hogg, Brittany Fletcher, and the rest of the Canada Power Yoga crew in Oshawa: In the half-year leading up to this book, I badly injured my back...twice. I don't have the words to describe the frustration and despair and how my life had taken a wrong turn. After meeting Debbie and her crew, I found that, indeed, my life had taken a turn...but in the opposite direction from what I had originally thought. Thanks to this studio for teaching me a new practice, for your caring and your acceptance, and for restoring me to a place of well-being (at least until Chloë is a teenager). Namaste.

—Paul Zikopoulos

To Sandra, Erik, and Anna: Yet again, I've taken on a book project, and yet again, you've given me the support that I've needed.

—Dirk deRoos

To my family and friends who patiently put up with the late nights, short weekends, and the long hours it takes to be a part of this incredible team: Thank you. I would also like to thank those who first domesticated Coffee arabica for making the aforementioned possible.

—Christopher Bienko

I would like to thank my immediate family for supporting my contribution to this book by putting up with my early and late hours and, at times, my stressed-out attitude. I especially want to thank my wife, Lyndy, my in-house English major, who helped edit and proofread my first draft and this dedication. I would also like to thank Paul Zikopoulos for providing me with the opportunity to be part of this book, even though there were times when I thought I was crazy for volunteering. Finally, I would like to thank my entire Optim extended family for “having my back” at work and for allowing me to “go stealth” as needed so that I could focus on making this project a reality.

—Rick Buglio

I would like to thank all of our clients for taking the time to tell us about their challenges and giving us the opportunity to demonstrate how we can help them. I would also like to thank the entire IBM Big Data Industry Team for their continued motivation and passion working with our clients to understand their business needs and helping them to find ways of delivering new value through information and analytics. By listening to our clients and sharing their experiences, we are able to continuously learn new ways to help transform industries and businesses with data. And thank you to my family, Amy, Ayla, and Ethan, for their patience and support even when I am constantly away from home to spend time with companies across the world in my personal pursuit to make an impact.

—Marc Andrews

CONTENTS AT A GLANCE

PART I

Opening Conversations About Big Data

1	Getting Hype out of the Way: Big Data and Beyond	3
2	To SQL or Not to SQL: That’s Not the Question, It’s the Era of Polyglot Persistence	31
3	Composing Cloud Applications: Why We Love the Bluemix and the IBM Cloud	59
4	The Data Zones Model: A New Approach to Managing Data	97

PART II

Watson Foundations

5	Starting Out with a Solid Base: A Tour of Watson Foundations	123
6	Landing Your Data in Style with Blue Suit Hadoop: InfoSphere BigInsights	131
7	“In the Moment” Analytics: InfoSphere Streams	179
8	700 Million Times Faster Than the Blink of an Eye: BLU Acceleration	209
9	An Expert Integrated System for Deep Analytics	249
10	Build More, Grow More, Sleep More: IBM Cloudant	269

PART III
Calming the Waters: Big Data Governance

11	Guiding Principles for Data Governance	303
12	Security Is NOT an Afterthought	309
13	Big Data Lifecycle Management	329
14	Matching at Scale: Big Match	343

CONTENTS

Foreword	xix
Acknowledgments	xxi
Introduction	xxiii

PART I

Opening Conversations About Big Data

1	Getting Hype out of the Way: Big Data and Beyond	3
	There's Gold in "Them There" Hills!	3
	Why Is Big Data Important?	5
	Brought to You by the Letter V:	
	How We Define Big Data	8
	Cognitive Computing	12
	Why Does the Big Data World	
	Need Cognitive Computing?	15
	A Big Data and Analytics Platform Manifesto	17
	1. Discover, Explore, and Navigate Big Data Sources	18
	2. Land, Manage, and Store	
	Huge Volumes of Any Data	20
	3. Structured and Controlled Data	21
	4. Manage and Analyze Unstructured Data	22
	5. Analyze Data in Real Time	24
	6. A Rich Library of Analytical Functions and Tools	24
	7. Integrate and Govern All Data Sources	26
	Cognitive Computing Systems	27
	Of Cloud and Manifestos... ..	27
	Wrapping It Up	28
2	To SQL or Not to SQL: That's Not the Question, It's the Era of Polyglot Persistence	31
	Core Value Systems: What Makes	
	a NoSQL Practitioner Tick	33
	What Is NoSQL?	36
	Is Hadoop a NoSQL Database?	38

- Different Strokes for Different Folks:
 - The NoSQL Classification System 39
 - Give Me a Key, I'll Give You a Value:
 - The Key/Value Store 39
 - The Grand-Daddy of Them All: The Document Store 40
 - Column Family, Columnar Store, or
 - BigTable Derivatives: What Do We Call You? 46
 - Don't Underestimate the Underdog: The Graph Store 47
- From ACID to CAP 54
 - CAP Theorem and a Meatloaf Song:
 - "Two Out of Three Ain't Bad" 55
- Let Me Get This Straight:
 - There Is SQL, NoSQL, and Now NewSQL? 57
- Wrapping It Up 58

- 3** Composing Cloud Applications: Why We Love the Bluemix and the IBM Cloud 59
 - At Your Service: Explaining Cloud Provisioning Models 61
 - Setting a Foundation for the Cloud:
 - Infrastructure as a Service 64
 - IaaS for Tomorrow...Available Today:
 - IBM SoftLayer Powers the IBM Cloud 67
 - Noisy Neighbors Can Be Bad Neighbors:
 - The Multitenant Cloud 69
 - Building the Developer's Sandbox with Platform as a Service 71
 - If You Have Only a Couple of Minutes:
 - PaaS and IBM Bluemix in a Nutshell 72
 - Digging Deeper into PaaS 74
 - Being Social on the Cloud: How Bluemix Integrates Platforms and Architectures 76
 - Understanding the Hybrid Cloud:
 - Playing Frankenstein Without the Horror 77
 - Tried and Tested: How Deployable Patterns Simplify PaaS 79
 - Composing the Fabric of Cloud Services: IBM Bluemix 82
 - Parting Words on Platform as a Service 85
 - Consuming Functionality Without the Stress:
 - Software as a Service 85
 - The Cloud Bazaar: SaaS and the API Economy 87
 - Demolishing the Barrier to Entry for Cloud-Ready Analytics: IBM's dashDB 89

	Build More, Grow More, Know More:	
	dashDB’s Cloud SaaS	92
	Refinery as a Service	93
	Wrapping It Up	94
4	The Data Zones Model:	
	A New Approach to Managing Data	97
	Challenges with the Traditional Approach	100
	Agility	100
	Cost	101
	Depth of Insight	102
	Next-Generation Information Management Architectures	103
	Prepare for Touchdown: The Landing Zone	104
	Into the Unknown: The Exploration Zone	105
	Into the Deep: The Deep Analytic Zone	108
	Curtain Call: The New Staging Zone	109
	You Have Questions? We Have Answers!	
	The Queryable Archive Zone	112
	In Big Data We Trust: The Trusted Data Zone	115
	A Zone for Business Reporting	115
	From Forecast to Nowcast:	
	The Real-Time Processing and Analytics Zone	116
	Ladies and Gentlemen, Presenting...	
	“The Data Zones Model”	118

PART II
Watson Foundations

5	Starting Out with a Solid Base:	
	A Tour of Watson Foundations	123
	Overview of Watson Foundations	124
	A Continuum of Analytics Capabilities:	
	Foundations for Watson	126
6	Landing Your Data in Style with Blue Suit Hadoop:	
	InfoSphere BigInsights	131
	Where Do Elephants Come From: What Is Hadoop?	133
	A Brief History of Hadoop	135
	Components of Hadoop and Related Projects	136
	Open Source...and Proud of It	137
	Making Analytics on Hadoop Easy	141

- The Real Deal for SQL on Hadoop: Big SQL 142
- Machine Learning for the Masses:
 - Big R and SystemML 147
- The Advanced Text Analytics Toolkit 149
- Data Discovery and Visualization: BigSheets 152
- Spatiotemporal Analytics 154
- Finding Needles in Haystacks of Needles:
 - Indexing and Search in BigInsights 155
- Cradle-to-Grave Application Development Support 155
 - The BigInsights Integrated Development Environment 156
 - The BigInsights Application Lifecycle 157
 - An App Store for Hadoop: Easy Deployment and Execution of Custom Applications 159
- Keeping the Sandbox Tidy:
 - Sharing and Managing Hadoop 160
 - The BigInsights Web Console 160
 - Monitoring the Aspects of Your Cluster 161
 - Securing the BigInsights for Hadoop Cluster 162
 - Adaptive MapReduce 163
 - A Flexible File System for Hadoop: GPFS-FPO 164
- Playing Nice: Integration with Other Data Center Systems 167
 - IBM InfoSphere System z Connector for Hadoop 168
 - IBM PureData System for Analytics 168
 - InfoSphere Streams for Data in Motion 169
 - InfoSphere Information Server for Data Integration 170
 - Matching at Scale with Big Match 170
 - Securing Hadoop with Guardium and Optim 171
 - Broad Integration Support 171
- Deployment Flexibility 172
 - BigInsights Editions: Free, Low-Cost, and Premium Offerings 172
 - A Low-Cost Way to Get Started:
 - Running BigInsights on the Cloud 174
 - Higher-Class Hardware:
 - Power and System z Support 176

	Get Started Quickly!	177
	Wrapping It Up	177
7	“In the Moment” Analytics: InfoSphere Streams	179
	Introducing Streaming Data Analysis	179
	How InfoSphere Streams Works	181
	A Simple Streams Application	182
	Recommended Uses for Streams	184
	How Is Streams Different from CEP Systems?	184
	Stream Processing Modes: Preserve Currency or Preserve Each Record	185
	High Availability	185
	Dynamically Distributed Processing	187
	InfoSphere Streams Platform Components	188
	The Streams Console	188
	An Integrated Development Environment for Streams: Streams Studio	191
	The Streams Processing Language	195
	Source and Sink Adapters	198
	Analytical Operators	199
	Streams Toolkits	202
	Solution Accelerators	205
	Use Cases	205
	Get Started Quickly!	207
	Wrapping It Up	207
8	700 Million Times Faster Than the Blink of an Eye:	
	BLU Acceleration	209
	What Is BLU Acceleration?	214
	What Does a Next Generation Database Service for Analytics Look Like?	214
	Seamlessly Integrated	217
	Hardware Optimized	217
	Convince Me to Take BLU Acceleration for a Test Drive	218
	Pedal to the Floor: How Fast Is BLU Acceleration?	219
	From Minimized to Minuscule: BLU Acceleration Compression Ratios	220
	Where Will I Use BLU Acceleration?	222

- How BLU Acceleration Came to Be:
 - Seven Big Ideas 225
 - Big Idea #1: KISS It! 225
 - Big Idea #2: Actionable Compression
and Computer-Friendly Encoding 226
 - Big Idea #3: Multiplying the Power of the CPU 229
 - Big Idea #4: Parallel Vector Processing 231
 - Big Idea #5: Get Organized...by Column 232
 - Big Idea #6: Dynamic In-Memory Processing 234
 - Big Idea #7: Data Skipping 236
 - How Seven Big Ideas Optimize the Hardware Stack 237
 - The Sum of All Big Ideas: BLU Acceleration in Action ... 238
 - DB2 with BLU Acceleration Shadow Tables:
 - When OLTP + OLAP = 1 DB 241
 - What Lurks in These Shadows Isn't Anything
to Be Scared of: Operational Reporting 241
 - Wrapping It Up 246
- 9** An Expert Integrated System for Deep Analytics 249
 - Before We Begin: Bursting into the Cloud 252
 - Starting on the Whiteboard: Netezza's Design Principles ... 253
 - Appliance Simplicity: Minimize the Human Effort 254
 - Process Analytics Closer to the Data Store 254
 - Balanced + MPP = Linear Scalability 255
 - Modular Design: Support Flexible
Configurations and Extreme Scalability 255
 - What's in the Box? The Netezza Appliance
 - Architecture Overview 256
 - A Look Inside a Netezza Box 257
 - How a Query Runs in Netezza 261
 - How Netezza Is a Platform for Analytics 264
 - Wrapping It Up 266
- 10** Build More, Grow More, Sleep More: IBM Cloudant 269
 - Cloudant: "White Glove" Database as a Service 271
 - Where Did Cloudant Roll in From? 276
 - Cloudant or Hadoop? 278
 - Being Flexible: Schemas with JSON 279
 - Cloudant Clustering: Scaling for the Cloud 279

Avoiding Mongo-Size Outages: Sleep Soundly with Cloudant Replication	281
Cloudant Sync Brings Data to a Mobile World	283
Make Data, Not War: Cloudant Versioning and Conflict Resolution	284
Unlocking GIS Data with Cloudant Geospatial	287
Cloudant Local	289
Here on In: For Techies...	290
For Techies: Leveraging the Cloudant Primary Index	290
Exploring Data with Cloudant’s Secondary Index “Views”	292
Performing Ad Hoc Queries with the Cloudant Search Index	293
Parameters That Govern a Logical Cloudant Database	294
Remember! Cloudant Is DBaaS	298
Wrapping It Up	299

PART III

Calming the Waters: Big Data Governance

11	Guiding Principles for Data Governance	303
	The IBM Data Governance Council Maturity Model	304
	Wrapping It Up	308
12	Security Is NOT an Afterthought	309
	Security Big Data: How It’s Different	310
	Securing Big Data in Hadoop	313
	Culture, Definition, Charter, Foundation, and Data Governance	314
	What Is Sensitive Data?	315
	The Masquerade Gala: Masking Sensitive Data	316
	Don’t Break the DAM: Monitoring and Controlling Access to Data	323
	Protecting Data at Rest	325
	Wrapping It Up	326
13	Big Data Lifecycle Management	329
	A Foundation for Data Governance: The Information Governance Catalog	330
	Data on Demand: Data Click	333

- Data Integration 334
- Data Quality 337
- Veracity as a Service: IBM DataWorks 338
- Managing Your Test Data: Optim Test Data Management ... 341
- A Retirement Home for Your Data: Optim Data Archive 341
- Wrapping It Up 342
- 14** Matching at Scale: Big Match 343
 - What Is Matching Anyway? 344
 - A Teaser: Where Are You Going to Use Big Match? 345
 - Matching on Hadoop 346
 - Matching Approaches 347
 - Big Match Architecture 348
 - Big Match Algorithm Configuration Files 349
 - Big Match Applications 350
 - HBase Tables 350
 - Probabilistic Matching Engine 350
 - How It Works 351
 - Extract 352
 - Search 352
 - Applications for Big Match 353
 - Enabling the Landing Zone 353
 - Enhanced 360-Degree View of Your Customers 353
 - More Reliable Data Exploration 354
 - Large-Scale Searches for Matching Records 355
 - Wrapping It Up 356

FOREWORD

Through thousands of client engagements, we've learned organizations that outperform are using data for competitive advantage. What sets these leaders apart? It's their ability to get three things right. First, they drive business outcomes by applying more sophisticated analytics across more disparate data sources in more parts of their organization—they infuse analytics everywhere. Second, they capture the time value of data by developing “speed of insight” and “speed of action” as core differentiators—they don't wait. Third, they look to change the game in their industry or profession—they shift the status quo. These game changers could be in how they apply Big Data and Analytics to attract, grow, and retain customers; manage risk or transform financial processes; optimize operations; or indeed create new business models. At their very core, they directly or indirectly seek to capitalize on the value of information.

If you selected this book, you likely play a key role in the transformation of your organization through Big Data and Analytics. You are tired of the hype and are ready to have a conversation about seizing the value of data. This book provides a practical introduction to the next generation of data architectures; introduces the role of the cloud and NoSQL technologies; and discusses the practicalities of security, privacy, and governance. Whether you are new to this topic or an expert looking for the latest information, this book provides a solid foundation on which to grow.

Our writers are accomplished scientists, developers, architects, and mathematicians. They are passionate about helping our clients turn hype into reality. They understand the complexities of rapidly shifting technology and the practicalities of evolving and expanding a data architecture already in place. They have worked side-by-side with clients to help them transform their organization while keeping them on a winning path.

I'd like to acknowledge Paul, Dirk, Chris, Rick, and Marc for sharing their deep knowledge of this complex topic in a style that makes it accessible to all of us ready to seize our moment on this Big Data journey.



Beth Smith

General Manager, IBM Information Management

This page intentionally left blank

ACKNOWLEDGMENTS

Collectively, we want to thank the following people, without whom this book would not have been possible: Anjul Bhambhri, Rob Thomas, Roger Rea, Steven Sit, Rob Utzschneider, Joe DiPietro, Nagui Halim, Shivakumar Vaithyanathan, Shankar Venkataraman, Dwaine Snow, Andrew Buckler, Glen Sheffield, Klaus Roder, Ritika Gunnar, Tim Vincent, Jennifer McGinn, Anand Ranganathan, Jennifer Chen, and Robert Uleman. Thanks also to all the other people in our business who make personal sacrifices day in and day out to bring you the IBM Big Data and Analytics platform. IBM is an amazing place to work and is unparalleled when you get to work beside this kind of brain power every day.

Roman Melnyk, our technical editor, has been working with us for a long time—sometimes as a coauthor, sometimes as an editor, but always as an integral part of the project. We also want to thank Xiamei (May) Li who bailed us out on Chapter 14 and brought some common sense to our Big Match chapter. Bob Harbus helped us a lot with Chapter 8—the shadow tables technology—and we wanted to thank him here too.

We want to thank (although at times we cursed) Susan Visser, Frances Wells, Melissa Reilly, and Linda Currie for getting the book in place; an idea is an idea, but it takes people like this to help get that idea up and running. Our editing team—Janet Walden, Kim Wimpsett, and Lisa McCoy—all played a key role behind the scenes, and we want to extend our thanks for that. It's also hard not to give a special sentence of thanks to Hardik Popli at Cenveo Publisher Services—the guy's personal effort to perfection is beyond apparent. Finally, to our McGraw-Hill guru, Paul Carlstroem—there is a reason why we specifically want to work with you—you did more magic for this book than any other before it...thank you.

This page intentionally left blank

INTRODUCTION

The poet A.R. Ammons once wrote, “A word too much repeated falls out of being.” Well, kudos to the term *Big Data*, because it’s hanging in there, and it’s hard to imagine a term with more hype than *Big Data*. Indeed, perhaps it is repeated too much. *Big Data Beyond the Hype: A Guide to Conversations for Today’s Data Center* is a collection of discussions that take an overused term and break it down into a confluence of technologies, some that have been around for a while, some that are relatively new, and others that are just coming down the pipe or are not even a market reality yet. The book is organized into three parts.

Part I, “Opening Conversations About Big Data,” gives you a framework so that you can engage in Big Data conversations in social forums, at keynotes, in architectural reviews, during marketing mix planning, at the office watercooler, or even with your spouse (nothing like a Big Data discussion to inject romance into an evening). Although we talk a bit about what IBM does in this space, the aim of this part is to give you a grounding in cloud service delivery models, NoSQL, Big Data, cognitive computing, what a modern data information architecture looks like, and more. This chapter is going to give you the constructs and foundations that you will need to engage conversation that indeed can hype Big Data, but allow you to extend those conversations beyond.

In **Chapter 1**, we briefly tackle, define, and illustrate the term *Big Data*. Although its use is ubiquitous, we think that many people have used it irresponsibly. For example, some people think Big Data just means Hadoop—and although Hadoop is indeed a critical repository and execution engine in the Big Data world, Hadoop is not solely Big Data. In fact, without analytics, Big Data is, well, just a bunch of data. Others think Big Data just means more data, and although that *could be* a characteristic, you certainly can engage in Big Data without lots of data. Big Data certainly doesn’t replace the RDBMS either, and admittedly we do find it ironic that the biggest trend in the NoSQL world is SQL.

We also included in this chapter a discussion of *cognitive computing*—the next epoch of data analytics. *IBM Watson* represents a whole new class of industry-specific solutions called Cognitive Systems. It builds upon—but is

not meant to replace—the current paradigm of programmatic systems, which will be with us for the foreseeable future. It is often the case that keeping pace with the demands of an increasingly complex business environment requires a paradigm shift in what we should expect from IT. The world needs an approach that recognizes today’s realities as opportunities rather than challenges, and it needs computers that help with probabilistic outcomes. Traditional IT relies on search to find the location of a key phrase. Emerging IT gathers information and combines it for true discovery. Traditional IT can handle only small sets of focused data, whereas today’s IT must live with Big Data. And traditional IT is really just starting to ubiquitously work with machine language, whereas what we as users really need is to be able to interact with machines the way we communicate—by using natural language. All of these considerations, plus the cancer fighting, *Jeopardy!* winning, wealth managing, and retail-assisting gourmet chef known as IBM Watson itself, are discussed in this chapter.

After providing a solid foundation for how to define and understand Big Data, along with an introduction to cognitive computing, we finish this chapter by presenting to you our Big Data and Analytics platform manifesto. This vendor-neutral discussion lays out the architectural foundation for an information management platform that delivers dividends today and into the future. IBM has built such a platform, and we cover that in Part II of this book. If you’re taking a journey of discovery into Big Data, no matter what vendor (or vendors) you ultimately partner with, you will need what we outline in this part of Chapter 1.

Chapter 2 introduces you to something we call *polyglot persistence*. It’s an introduction to the NoSQL world and how that world is meant to complement the relational world. It’s about having access to a vast array of capabilities that are “right-fit, right-purpose” for you to deliver the kinds of solutions you need. There are well over 150 NoSQL databases, so we break them down into types and discuss the “styles” of NoSQL databases. We also discuss things like CAP and ACID, which should give you a general understanding of what differentiates the NoSQL and SQL worlds. We assume you have a pretty good knowledge of the relational world, so we won’t be focusing on relational databases here (although it’s definitely part of a polyglot architecture). Entire books have been written about NoSQL, so consider this chapter

a primer; that said, if you thought NoSQL was something you put on your resume if you don't know SQL, then reading this chapter will give you a solid foundation for understanding some of the most powerful forces in today's IT landscape.

In the cloud, you don't build applications; you *compose* them. "Composing Cloud Applications: Why We Love the Bluemix and the IBM Cloud" is the title for our dive into the innovative cloud computing marketplace of *composable services* (**Chapter 3**). Three key phrases are introduced here: *as a service*, *as a service*, and *as a service* (that's not a typo, our editors are too good to have missed that one). In this chapter, we introduce you to different cloud "as a service" models, which you can define by business value or use case; as your understanding of these new service models deepens, you will find that this distinction appears less rigid than you might first expect. During this journey, we examine IBM SoftLayer's flexible, bare-metal infrastructure as a service (IaaS), IBM Bluemix's developer-friendly and enterprise-ready platform as a service (PaaS), and software as a service (SaaS). In our SaaS discussion, we talk about the IBM Cloud marketplace, where you can get started in a free-*m*ium way with loads of IBM and partner services to drive your business. We will also talk about some of the subcategories in the SaaS model, such as data warehouse as a service (DWaaS) and database as a service (DBaaS). The IBM dashDB fully managed analytics service is an example of DWaaS. By this point in the book, we would have briefly discussed the IBM NoSQL Cloudant database—an example of DBaaS. Finally, we will tie together how services can be procured in a PaaS or SaaS environment—depending on what you are trying to get done. For example, IBM provides a set of services to help build trust into data; they are critical to building a data refinery. The collection of these services is referred to as IBM DataWorks and can be used in production or to build applications (just like the dashDB service, among others). Finally, we talk about the IBM Cloud and how IBM, non IBM, and open source services are hosted to suit whatever needs may arise. After finishing this chapter, you will be seeing "as a service" everywhere you look.

Part I ends with a discussion about where any organization that is serious about its analytics is heading: toward a data zones model (**Chapter 4**). New technologies introduced by the open source community and enhancements developed by various technology vendors are driving a dramatic shift in

how organizations manage their information and generate insights. Organizations have been working hard to establish a single, trusted view of information across the enterprise but are realizing that traditional approaches lead to significant challenges related to agility, cost, and even the depth of insights that are provided. A next generation of architectures is emerging, one that is enabling organizations to make information available much faster; reduce their overall costs for managing data, fail fast, and archive on day zero; and drive a whole new level of value from their information.

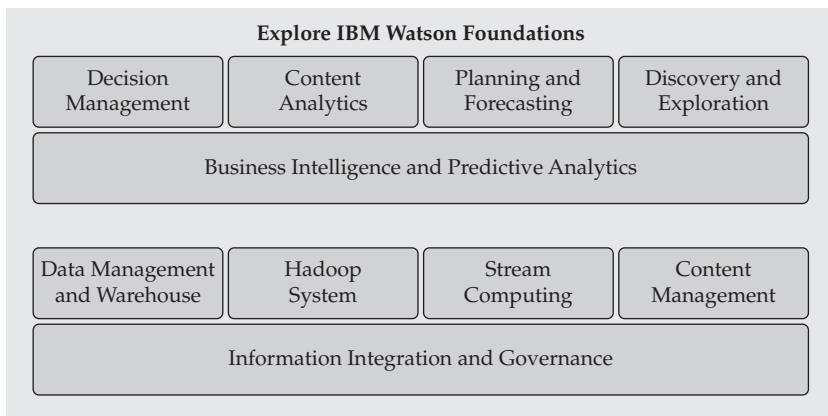
A modern architecture isn't about the death of the enterprise data warehouse (EDW); it's about a polyglot environment that delivers tangible economies of scale alongside powerful capabilities as a whole. This confluence results in deeper insights, and these insights are materialized more quickly. So, is the EDW dead? Not even close. However, there is a move away from the traditional idea of EDWs as the "center of the universe" to the concept of an environment in which data is put into different "zones" and more fit-for-purpose services are leveraged on the basis of specific data and analytic requirements.

Chapter 4 talks about the challenges most companies are trying to overcome and the new approaches that are rapidly evolving. Read about sandboxes, landing zones, data lakes (but beware of the data swamp), fail-fast methodologies, day zero archives, data refineries, and more. We also discuss the elephant in the room: Hadoop.

What is a data refinery? It's a facility for transforming raw data into relevant and actionable information. Data refinement services take the uncertainty out of the data foundation for analysis and operations. Refined data is timely, clean, and well understood. A data refinery is needed to address a critical problem facing today's journey-bound Big Data organizations: Data seems to be everywhere *except* where we need it, *when* we need it, and in the *reliable* form that we need. That refinery also has to be available as a set of services such that the information flows can be composed in the cloud with discrete pieces of refinery logic—and traditionally available on premise too. The data holds great potential, but it's not going to deliver on this potential unless it is made available quickly, easily, and cleanly to both people and systems. A data refinery and the zone architecture we cover in this chapter go hand in hand when it comes to a next-generation information management architecture.

Part II, “IBM Watson Foundations,” covers the IBM Big Data and Analytics platform that taps into all relevant data, regardless of source or type, to provide fresh insights in real time and the confidence to act on them. IBM Watson Foundations, as its name implies, is the place where data is prepared for the journey to cognitive computing, in other words, IBM Watson. Clients often ask us how to get started with an IBM Watson project. We tell them to start with the “ground truth”—your predetermined view of good, rational, and trusted insights—because to get to the start line, you need a solid foundation. IBM Watson Foundations enables you to infuse analytics into every decision, every business process, and every system of engagement; indeed, this part of the book gives you details on how IBM can help you get Big Data beyond the hype.

As part of the IBM Big Data and Analytics portfolio, Watson Foundations supports all types of analytics (including discovery, reporting, and analysis) and predictive and cognitive capabilities, and that’s what we cover in **Chapter 5**. For example, Watson Foundations offers an enterprise-class, nonforked Apache Hadoop distribution that’s optionally “Blue suited” for even more value; there’s also a rich portfolio of workload-optimized systems, analytics on streaming data, text and content analytics, and more. With governance, privacy, and security services, the platform is open, modular, trusted, and integrated so that you can start small and scale at your own pace. The following illustration shows an overview map of the IBM Watson Foundations capabilities and some of the things we cover in this part of the book.



Chapter 6 describes Hadoop and its related ecosystem of tools. Hadoop is quickly becoming an established component in today's data centers and is causing significant disruption (in a good way) to traditional thinking about data processing and storage. In this chapter, you'll see what the buzz is all about as we delve into Hadoop and some big changes to its underlying architecture. In addition to its capabilities, Hadoop is remarkable because it's a great example of the power of community-driven open source development. IBM is deeply committed to open source Hadoop and is actively involved in the community. You'll also learn about IBM's Hadoop distribution, called *IBM InfoSphere BigInsights for Hadoop* (BigInsights), which has two main focus areas: enabling analysts to use their existing skills in Hadoop (such as SQL, statistics, and text analytics, for example) and making Hadoop able to support multitenant environments (by providing rich resource and workload management tools and an optional file system designed for scalability, multitenancy, and flexibility). If you are looking for 100 percent pure open source Hadoop, BigInsights has it (we call it "naked Hadoop"). Like other vendors (such as Cloudera), IBM offers optional proprietary Hadoop extensions that we think deliver greater value, such as visualization, security, SQL programmability, and more. We like to refer to this as "Blue Suit Hadoop." So if you want to leverage open source Apache Hadoop without any fancy add-ons but with full support that includes all the back testing and installation complexities handled for you, BigInsights does that—and it also lets you dress it up with enterprise-hardening capabilities.

Chapter 7 describes how you can analyze data before it has landed on disk—it's the "analytics for data in motion" chapter (sometimes, it seems that analytics for data at rest gets all the attention). This is an area in which IBM has differentiated itself from any other vendor in the marketplace. In fact, few Big Data vendors even talk about data in motion. IBM is well traveled in this area, with a high-velocity Big Data engine called *InfoSphere Streams*. Think about it: From EDWs to RDBMSs to HDFS, the talk is always centered on harvesting analytics "at rest." Conversations about Big Data typically end here, forgetting about how organizations can really benefit by moving their at-rest analytics (where they *forecast*) to the frontier of the business (where they can *nowcast*). We call this *in-the-moment analytics*. The irony is that we all operate "in the moment" every time we write an email, yet we don't demand it of our analytics systems. For example, when you misspell a

word, does it get flagged with a disapprovingly red squiggly underline as you type? That's in the moment. How many of us configure our spelling analytics engine to engage only after the entire email has been written? Not many. In fact, it's more likely the case that you have autocorrect turned on, and if a spelling error is discovered as you type, it's automatically corrected. What's more, as you spot words the analytics dictionary doesn't recognize, you can opt to "learn" them (harvesting them at rest), and subsequent occurrences of the same misspelled words are corrected or highlighted (in motion).

Did you know that it takes between 300 and 400 milliseconds to blink the human eye? When you hear that analytics is occurring "in the blink of an eye," it certainly sounds fast, doesn't it? What you might not know is that BLU Acceleration analytics, in its sweet spot, is about *700 million times faster* than the blink of an eye (yes, you read that right), and this is the focus of **Chapter 8**. You also might not know that BLU Acceleration is a technology and not a product—so you are going to find it in various on-premise products and off-premise services (we talk about that in this chapter too). BLU Acceleration is a confluence of various big ideas (some new and some that have been around for a while) that together represent what we believe to be the most powerful in-memory analytics system in today's competitive marketplace. In this chapter, we introduce you to those ideas. BLU Acceleration technology first debuted in an on-premise fashion in the DB2 10.5 release. What we find even more exciting is its availability in both PaaS and SaaS provisioning models as a newly hosted or managed (depending if you leverage the service from the IBM Cloud Marketplace or Bluemix) analytics service called dashDB!

One of the things that makes BLU Acceleration so special is that it is optimized to work *where the data resides*, be it on a traditional spinning disk, a solid-state disk (SSD), in system memory (RAM), or even in the CPU cache (L1, L2, and so on). Let's face it, in a Big Data world, not all of the data is going to fit into memory. Although some nameplate vendors have solutions that require this, the BLU Acceleration technology does not. In fact, BLU Acceleration treats system memory as the "new disk." It really doesn't want to use that area to persist data unless it needs to because RAM is too darn slow! (Now consider those vendors who talk about their "great" in-memory system RAM databases.) This is yet another differentiator for the BLU Acceleration technology: It's built from the ground up to leverage 700 million times faster storage mechanisms than the blink of an eye.

If you are using the BLU Acceleration technology on premise through DB2, you simply must take notice of the fact that it has also been extended to bring reporting directly on top of OLTP systems with *shadow table* support in the DB2 Cancun release (10.5.0.4). Clients testing this new capability have seen incredible performance improvements in their latency-driven extract, transform, and load (ETL) protocols, not to mention a jaw-dropping performance boost to their reports and analytics, without sacrificing the performance of their OLTP systems.

The IBM PureData System for Analytics (PDA), powered by Netezza technology, is a simple data appliance for serious analytics. It simplifies and optimizes the performance of data services for analytic applications, enabling complex algorithms to run in minutes, not hours. We don't mind saying that the Netezza technology that powers this system pretty much started the appliance evolution that every major database vendor now embraces. And although there is a lot of imitation out there, we don't believe that other offerings are in the PDA class when it comes to speed, simplicity, and ease of operation in a single form factor; deeply integrated analytics; and the flattening of the time-to-value curve. **Chapter 9** describes the *IBM PureData System for Analytics*.

In this chapter, you learn about patented data filtering by *field programmable gate arrays* (FPGAs). You will also learn about PDA's rich built-in analytical infrastructure and extensive library of statistical and mathematical functions—this capability is referred to as IBM Netezza Analytics (INZA). INZA is an embedded, purpose-built, advanced analytics platform—delivered *free* with every system. It includes more than 200 scalable in-database analytic functions that execute analytics in parallel while removing the complexity of parallel programming from developers, users, and DBAs. In short, it lets you predict with more accuracy, deliver predictions faster, and respond rapidly to changes. In fact, we don't know of any competing vendor who offers this type of technology in their warehousing appliances. The second best part (it is hard to beat free) is that some of these functions are part of dashDB, and more and more of them will make their way over to this managed analytics service. In fact, so will the Netezza SQL dialect, making dashDB the analytics service to burst Netezza workloads into the cloud.

Chapter 10 is the “build more, grow more, sleep more” chapter that covers *IBM Cloudant*. This chapter continues the exploration of JSON and NoSQL

databases (covered in Chapter 2) with additional detail and a touch of technical depth about the IBM Cloudant DBaaS solution—another kind of SaaS offering. Cloudant’s earliest incarnation was as a data management layer for one of the largest data-generating projects on Earth: the Large Hadron Collider (LHC), operated by the European Organization for Nuclear Research (CERN). Among the many subject areas that particle physicists investigate with the LHC are the origin of the universe and how to replicate the conditions around the Big Bang, to name but a few. This data store technology matured into Cloudant, a fully managed NoSQL data-layer solution that eliminates complexity and risk for developers of fast-growing web and mobile applications. We examine how the flexibility of a JSON document data model ensures that you can perform powerful indexing and query work (including built-in Apache Lucene and geospatial searches) against nearly any type of structured or unstructured data, on an architecture that is a managed and scaled database as a service for off-premise, on-premise, and occasionally connected mobile environments.

Part III, “Calming the Waters: Big Data Governance,” covers one of the most overlooked parts of any Big Data initiative. In fact, if you’re moving on a Big Data project and you aren’t well versed with what we talk about in this part of the book, you’re going to get beyond the hype alright, but where you end up isn’t going to be what you had in mind when you started. Trust us. In fact, it’s so overlooked that although it’s part of Watson Foundations from a product perspective, we decided to give this topic its own part. We did this out of a concern for what we are seeing in the marketplace as companies run to the pot of gold that supposedly lies at the end of the Big Data rainbow. Although, indeed, there is the potential for gold to be found (assuming you infuse your Big Data with analytics), when you run for the gold with a pair of scissors in your hand, it can end up being painful and with potentially devastating consequences. In this part, we cover the principles of information governance, why security must *not* be an afterthought, how to manage the Big Data lifecycle, and more.

Chapter 11 describes what really sets IBM apart in the Big Data marketplace: a deep focus on data governance. There is universal recognition in the relational database space that the principles of data governance (such as data quality, access control, lifecycle management, and data lineage) are critical success factors. IBM has made significant investments to ensure that the key

principles of data governance can be applied to Big Data technologies. This is not the chapter to miss—we can't think of a more important chapter to read before beginning any Big Data conversation. Think of it this way: If you were to plan a trip somewhere, would you get there faster, safer, and more efficiently by using a GPS device or intuition? Governance is about using a GPS device for turn-by-turn directions to effective analytics.

Chapter 12 is the “security is *not* an afterthought” chapter in which we discuss the importance of security in a Hadoop environment and the ever-changing world of Big Data. We share some of our personal field experiences on this topic, and we review the core security requirements for your Hadoop environment. If data is sensitive and needs to be protected in an RDBMS, why do some folks think it doesn't have to be protected in HDFS (Hadoop's file system)? As more and more organizations adopt Hadoop as a viable platform to augment their current data housing methods, they need to become more knowledgeable about the different data security and protection methods that are available, weighing the pros and cons of each.

We finish this chapter with a somewhat detailed introduction to the IBM services that are available for this domain, including their abilities to govern, protect, and secure data in Hadoop environments as it moves through the data lifecycle—on or off premise. We cover the *InfoSphere Guardium* and *InfoSphere Optim* family of data lifecycle management services. There is so much talk about “data lakes” these days, but without an understanding of the material that we cover in this chapter, it's more likely to be a data swamp!

Chapter 13 extends the conversation from the safeguarding of information to making the data trusted—and there's a big difference. As lines of business, data scientists, and others get access to more and more data, it's essential that this data be surfaced to them as artifacts for interrogation, not by its location in the polyglot environment (for example, sentiment data in Hadoop and mobile interaction data in Cloudant). This enables you to see a manifest of data artifacts being surfaced through a Big Data catalog and to access that data through your chosen interface (Excel spreadsheet, R, SQL, and so on). To trust the data, you need to understand its provenance and lineage, and that has everything to do with the metadata. Let's face it, metadata isn't sexy, but it's vitally important. Metadata is the “secret sauce” behind a successful Big Data project because it can answer questions like “Where did this data come

from?” and “Who owns this metric?” and “What did you do to present these aggregated measures?”

A business glossary that can be used across the polyglot environment is imperative. This information can be surfaced no matter the user or the tool. Conversations about the glossarization, documentation, and location of data make the data more trusted, and this allows you to broadcast new data assets across the enterprise. Trusting data isn't solely an on-premise phenomenon; in our social-mobile-cloud world, it's critical that these capabilities are provided as services, thereby creating a data refinery for trusted data. The set of products within the *InfoSphere Information Server* family, which is discussed in this chapter, includes the services that make up the IBM data refinery and they can be used traditionally through on-premise product installation or as individually composable discrete services via the IBM DataWorks catalog.

Master data management (matching data from different repositories) is the focus of **Chapter 14**. Matching is a critical tool for Big Data environments that facilitate regular reporting and analytics, exploration, and discovery. Most organizations have many databases, document stores, and log data repositories, not to mention access to data from external sources. Successful organizations in the Big Data era of analytics will effectively match the data between these data sets and build context around them at scale and this is where IBM's Big Match technology comes to play. In a Big Data world, traditional matching engines that solely rely on relational technology aren't going to cut it. IBM's Big Match, as far as we know, is the only enterprise-capable matching engine that's built on Hadoop, and this is the focus of the aptly named “Matching at Scale: Big Match” Chapter 14.

Ready...Set...Go!

We understand that when all is said and done, you will spend the better part of a couple of days of your precious time reading this book. But we're confident that by the time you are finished, you'll have a better understanding of requirements for the right Big Data and Analytics platform and a strong foundational knowledge of available IBM technologies to help you tackle the most promising Big Data opportunities. You will be able to get beyond the hype.

Our authoring team has more than 100 years of collective experience, including many thousands of consulting hours and customer interactions. We have experience in research, patents, sales, architecture, development, competitive analysis, management, and various industry verticals. We hope that we have been able to effectively share some of that experience with you to help you on your Big Data journey beyond the hype.

A handwritten signature in cursive script, appearing to read "Dick Toren".Three handwritten signatures in cursive script, reading "Chris Biondo", "Paul Byrd", and "Mark Jordan" from left to right.

P.S. It's a never-ending one!

Part I

Opening Conversations About Big Data

This page intentionally left blank

1

Getting Hype out of the Way: Big Data and Beyond

The term *Big Data* is a bit of a misnomer. Truth be told, we're not even big fans of the term—despite that it is so prominently displayed on the cover of this book—because it implies that other data is somehow small (it might be) or that this particular type of data is large in size (it can be, but doesn't have to be). For this reason, we thought we'd use this chapter to explain exactly what Big Data is, to explore the future of Big Data (cognitive computing), and to offer a manifesto of what constitutes a Big Data and Analytics platform.

There's Gold in "Them There" Hills!

We like to use a gold mining analogy to articulate the opportunity of Big Data. In the "olden days," miners could easily spot nuggets or veins of gold because they were highly visible to the naked eye. Let's consider that gold to be "high value-per-byte data." You can see its value, and therefore you invest resources to extract it. But there is more gold out there, perhaps in the hills nearby or miles away; it just isn't visible to the naked eye, and trying to find this hidden gold becomes too much of a gambling game. Sure, history has its gold rush fever stories, but nobody ever mobilized millions of people to dig everywhere and anywhere; that would be too expensive. The same is true for the data that resides in your enterprise data warehouse today. That data has been invested in and is trusted. Its value is obvious, so your business invested

4 Big Data Beyond the Hype

in cleansing it, transforming it, tracking it, cataloging it, glossarizing it, and so on. It was harvested...with care.

Today's miners work differently. Gold mining leverages new age capital equipment that can process millions of tons of dirt (low value-per-byte data) to find nearly invisible strands of gold. Ore grades of 30 parts per million are usually needed before gold is visible to the naked eye. In other words, there's a great deal of gold (high value-per-byte data) in all of this dirt (low value-per-byte data), and with the right equipment, you can economically process lots of dirt and keep the flakes of gold that you find. The flakes of gold are processed and combined to make gold bars, which are stored and logged in a safe place, governed, valued, and trusted. If this were data, we would call it *harvested* because it has been processed, is trusted, and is of known quality.

The gold industry is working on chemical washes whose purpose is to reveal even finer gold deposits in previously extracted dirt. The gold analogy for Big Data holds for this innovation as well. New analytic approaches in the future will enable you to extract more insight out of your forever archived data than you can with today's technology (we come back to this when we discuss cognitive computing later in this chapter).

This is Big Data in a nutshell: It is the ability to retain, process, and understand data like never before. It can mean more data than what you are using today, but it can also mean different kinds of data—a venture into the unstructured world where most of today's data resides.

If you've ever been to Singapore, you're surely aware of the kind of downpours that happen in that part of the world; what's more, you know that it is next to impossible to get a taxi during such a downpour. The reason seems obvious—they are all busy. But when you take the "visible gold" and mix it with the "nearly invisible gold," you get a completely different story. When Big Data tells the story about why you can't get a cab in Singapore when it is pouring rain, you find out that it is *not* because they are all busy. In fact, it is the opposite! Cab drivers pull over and stop driving. Why? Because the deductible on their insurance is prohibitive and not worth the risk of an accident (at fault or not). It was Big Data that found this correlation. By rigging Singapore cabs with GPS systems to do spatial and temporal analysis of their movements and combining that with freely available national weather service data, it was found that taxi movements mostly stopped in torrential downpours.

Why Is Big Data Important?

A number of years ago, IBM introduced its Smarter Planet campaign (“Instrumented, Interconnected, and Intelligent,” for those of you who didn’t get the T-shirt). This campaign anticipated the Big Data craze that hit the IT landscape just a few short years later.

From an instrumentation perspective, what *doesn't* have some amount of coding in it today? In a Big Data world, we can pretty much measure anything we want. Just look at your car; you can’t diagnose a problem these days without hooking it to a computer. The wearables market is set to explode too; even fashion house Ralph Lauren developed a shirt for the U.S. Open tennis championship that measures its wearer’s physiological markers such as heart rate, breathing, stress levels, and more. You can imagine how wearables can completely change the way society treats cardiac patients, post-traumatic stress disorders, train engineers, security forces, and more. As you can imagine, instrumentation has the ability to generate a heck of a lot of data. For example, one electric car on the market today generates 25GB of data during just one hour of charging.

One important Big Data capability is capturing data that is getting “dropped to the floor.” This type of data can yield incredible insights and results because it enriches the analytics initiatives that are going on in your organization today. *Data exhaust* is the term we like to use for this kind of data, which is generated in huge amounts (often terabytes per day) but typically isn’t tapped for business insight. Online storefronts fail to capture terabytes of generated clickstreams that can be used to perform web sessionization, optimize the “last mile” shopping experience, understand why online shopping baskets are getting abandoned, or simply understand the navigational experience. For example, the popular Zynga game FarmVille collects approximately 25TB of log data per day, and the company designs iterations of the game based on these interactions. We like to think of this data as digital body language because it shows you the path that a client took to reach a destination—be that the purchasing of a pair of socks or the selling of cheese in an online game. Stored log files are a corpus of data describing the operational state (and outages for that matter) of your most important networks—you could analyze this data for trends when nothing obvious has gone wrong to find the “needle in the stack of needles” that reveals potential downstream problems. There’s

an “if” here that tightly correlates with the promise of Big Data: “If you could collect and analyze all the data...” We like to refer to the capability of analyzing all of the data as *whole-population analytics*. It’s one of the value propositions of Big Data; imagine the kind of predictions and insights your analytic programs could make if they weren’t restricted to samples and subsets of the data.

In the last couple of years, the data that is available in a Big Data world has increased even more, and we refer to this phenomenon as the *Internet of Things (IoT)*. The IoT represents an evolution in which objects are capable of interacting with other objects. For example, hospitals can monitor and regulate pacemakers from afar, factories can automatically address production-line issues, and hotels can adjust temperature and lighting according to their guests’ preferences. This development’s prediction by IBM’s Smarter Planet agenda was encapsulated by the term *interconnected*.

This plethora of data sources and data types opens up new opportunities. For example, energy companies can do things that they could not do before. Data gathered from smart meters can provide a better understanding of customer segmentation and behavior and of how pricing influences usage—but only *if* companies have the ability to use such data. Time-of-use pricing encourages cost-savvy energy consumers to run their laundry facilities, air conditioners, and dishwashers at off-peak times. But the opportunities don’t end there. With the additional information that’s available from smart meters and smart grids, it’s possible to transform and dramatically improve the efficiency of electricity generation and scheduling. It’s also possible to determine which appliances are drawing too much electricity and to use that information to propose rebate-eligible, cost-effective, energy-efficient upgrades wrapped in a compelling business case to improve the conversion yield on an associated campaign.

Now consider the additional impact of social media. A social layer on top of an instrumented and interconnected world generates a massive amount of data too. This data is more complex because most of it is unstructured (images, Twitter feeds, Facebook posts, micro-blog commentaries, and so on). If you eat Frito-Lay SunChips, you might remember its move to the world’s first biodegradable, environmentally friendly chip bag; you might also remember how loud the packaging was. Customers created thousands of YouTube videos showing how noisy the environmentally friendly bag was. A “Sorry, but I can’t hear you over this SunChips bag” Facebook page had hundreds of

thousands of Likes, and bloggers let their feelings be known. In the end, Frito-Lay introduced a new, quieter SunChips bag, demonstrating the power and importance of social media. It isn't hard to miss the careers lost and made from a tweet or video that went viral.

For a number of years, Facebook was adding a new user every three seconds; today these users collectively generate double-digit terabytes of data every day. In fact, in a typical day, Facebook experiences more than 3.5 billion posts and about 155 million "Likes." The format of a Facebook post is indeed structured data. It's encoded in the JavaScript Object Notation (JSON) format—which we talk about in Chapter 2. However, it's the *unstructured* part that has the "golden nugget" of potential value; it holds monetizable intent, reputational decrees, and more. Although the structured data is easy to store and analyze, it is the unstructured components for intent, sentiment, and so on that are hard to analyze. They've got the potential to be very rewarding, *if...*

Twitter is another phenomenon. The world has taken to generating over 400 million daily expressions of short 140 character or less opinions (amounting to double-digit terabytes) and commentary (often unfiltered) about sporting events, sales, images, politics, and more. Twitter also provides enormous amounts of data that's structured in format, but it's the unstructured part within the structure that holds most of the untapped value. Perhaps more accurate, it's the combination of the structured (timestamp, location) and unstructured (the message) data where the ultimate value lies.

The social world is at an inflection point. It is moving from a text-centric mode of communication to a visually centric one. In fact, the fastest growing social sites (such as Vine, Snapchat, Pinterest, and Instagram, among others) are based on video or image communications. For example, one fashion house uses Pinterest to build preference profiles for women (Pinterest's membership is approximately 95 percent female). It does this by sending collages of outfits to clients and then extracting from the likes and dislikes preferences for color, cut, fabric, and so on; the company is essentially learning through error (it's a methodology we refer to as *fail fast* and we talk about it later in this book). Compare this to the traditional method whereby you might fill in a questionnaire about your favorite colors, cuts, and styles. Whenever you complete a survey, you are guarded and think about your responses. The approach that this fashion house is using represents unfiltered observation of raw human behavior—instant decisions in the form of "likes" and "dislikes."

Most of today's collected data is also temporally and spatially enriched. For example, we know where one of the stars of the television show *Myth Busters* lives, not because he told us, but because he tweeted a picture of his car with location-based services (LBS) enabled on his smart device and ended up sharing his home's geographic location with more than a million of his closest friends! Most people don't know what LBS is, but they have it turned on because they're using some mobile mapping application. These days, folks let you know when they have checked in at the gym or what restaurant they are in through their social apps.

Such data, with its built-in location awareness, represents another tremendous opportunity for finer granularities of personalization or profiled risk assessment, *if...* Today, a number of major credit card companies have programs that are based on this approach. For example, if you purchase a coffee using your credit card, the company will profile your location (through LBS) and your purchase history and then communicate an offer from a retailer near your current location that is tailored to you or vice versa.

Brought to You by the Letter V: How We Define Big Data

Somehow Big Data got tagged with the stem letter *V* to describe its attributes. We've seen some definitions use so many *Vs* that we are tempted to add "*V* for Vendetta" to rise up against this trend and revolt. With that said, we're not going to invent a new nomenclature to describe Big Data, but we will limit ourselves to just four *Vs*: *volume*, *variety*, *velocity*, and *veracity*.

No Question About It: Data Volumes Are on the Rise

Volume is the obvious Big Data attribute. At the start of this chapter, we gave you all kinds of statistics that were out of date the moment we sent the manuscript to the printers! We've already mentioned some of the causes for data explosion, which we categorize as *social-mobile-cloud* and the *Internet of Things*. So, we'll just leave you with one more statistic that will be out of date by the time you get this book: 5PB of data is generated *every day* from mobile devices alone. That's about 10,000 Blu-ray Discs. Don't forget—volume is relative. So if your analytical models are measured in gigabytes, then terabytes represent Big Data in your world from a volume perspective. Don't fall in the "I have more data than you have" trap. It's not always about how

much data you have; it's what you do with it that counts. Like we always say, Big Data without analytics is...well...just a bunch of data.

Variety Is the Spice of Life

The *variety* characteristic of Big Data is really about trying to capture all of the data that pertains to our decision-making process. Making sense out of unstructured data, such as opinion and intent musings on Facebook or analyzing images, isn't something that comes naturally for computers. However, this kind of data complements the data we use to drive decisions today. Most of the data out there is semistructured or unstructured. (To clarify, all data has some structure; when we refer to *unstructured data*, we are referring to the subcomponents that don't have structure, such as the free-form text in a comments field or the image in an autodedated and geocoded picture.)

Consider a customer call center. Imagine being able to detect the change in tone of a frustrated client who raises his voice to say, "This is the third outage I've had in one week!" A Big Data solution would identify not only the terms *third* and *outage* as negative events trending to a consumer churn event, but also the tonal change as another indicator that a customer churn incident is about to happen. A Big Data solution could do more than just deescalate such a scenario—it could prevent it.

All of this insight can be gleaned from unstructured data. Now combine this unstructured data with the customer's system of record data such as transaction history (the structured data with which we're familiar), and you've got a personalized model of this consumer: his value, how "brittle" he is becoming as your customer, and much more. (You could start this analysis by attempting to analyze recorded calls, not in real time, and evolve the solution to one that analyzes the spoken word in real time.)

Adelos has developed one of the most sophisticated sound classification systems in the world. This system is used for real-time perimeter security control; a thousand sensors are buried underground to collect and classify detected sounds so that appropriate action can be taken (dispatch personnel, dispatch aerial surveillance, and so on) depending on the classification. Consider the problem of securing the perimeter of a nuclear reactor that's surrounded by parkland. The Adelos system could almost instantaneously differentiate the whisper of the wind from a human voice or the sound of a human footstep from the sound of running wildlife. In fact, if a tree were to fall in one of its protected forests, Adelos can affirm that it makes a sound

and classify what the sound was (in this case a falling tree), even if no one is around to hear it. Image and sound classification is a great example of the variety characteristic of Big Data.

How Fast Can You Analyze? The Velocity of Your Data

One of our favorite but least understood characteristics of Big Data is *velocity*. We define velocity as the rate at which data arrives at the enterprise and the time that it takes the enterprise to process and understand that data. We often ask our clients, “After data arrives at your enterprise’s doorstep, how long does it take you to do something about it or to even know that it has arrived?” (We don’t like most of the answers we get to this question.)

Think about it for a moment. The opportunity-cost clock on your data starts ticking the moment the data hits the wire. As organizations, we’re taking far too long to spot trends or to pick up valuable insights. It doesn’t matter what industry you are in; being able to more swiftly understand and to respond to data signals puts you in a position of power. Whether you are trying to understand the health of a traffic system, a patient, or apply a stress test to a position in a booked loan portfolio, reacting faster gives you an advantage. Velocity is one of the most overlooked areas in the Big Data craze, and it’s an area in which we believe IBM is unequalled in the capabilities and sophistication that it provides.

You might be thinking that velocity can be handled by Complex Event Processing (CEP) systems, and although they might seem applicable on the surface, in the Big Data world, they fall short. Stream processing enables advanced analysis across diverse data types (not just relational data) with high messaging data rates and low latency. For example, one financial services sector (FSS) client analyzes and correlates more than 5 million market messages per second to execute algorithmic option trades with an average latency of 30 microseconds. Another client analyzes more than 500,000 Internet Protocol detail records (IPDRs) per second, which equals more than 6 billion IPDRs per day, on more than 4PB of data per year to understand trending and the current health of its networks. Consider an enterprise network security problem. In this domain, threats come in microseconds, so you need technology that can respond and keep pace. However, you also need something that can capture lots of data quickly and analyze it to identify emerging signatures and patterns on the network packets as they flow across the network infrastructure.

Finally, from a governance perspective, consider the added benefit of a Big Data analytics velocity engine. If you have a powerful analytics engine that can apply complex analytics to data as it flows across the wire and you can glean insight from that data without having to land it, you might not have to subject this data to retention policies, which can result in huge savings for your IT department.

Today's CEP solutions are targeted to approximately tens of thousands of messages per second at best, with seconds-to-minutes latency. Moreover, the analytics are mostly rules based and applicable only to traditional data types (in contrast to the Adelos example earlier). Don't get us wrong; CEP has its place (which is why IBM Streams ships with a CEP toolkit), but it has fundamentally different design points. CEP is a nonprogrammer-oriented solution for applying simple rules to discrete and "complex" events.

Not a lot of people are talking about Big Data velocity because there aren't a lot of vendors that can deal with it, let alone integrate at-rest technologies with velocity to deliver economies of scale for an enterprise's current investment. The ability to have seamless analytics for both at-rest and in-motion data moves you from the *forecast* model that's so tightly aligned with traditional warehousing and energizes the business with a *nowcast* model. The whole point is getting the insight that you learn at rest to the frontier of the business so that it can be analyzed and understood as it happens.

Data Here, Data There, Data, Data Everywhere: The Veracity of Data

Veracity is a term that's being used more and more to describe Big Data; it refers to the quality or trustworthiness of the data. Tools that help handle Big Data's veracity discard "noise" and transform the data into trustworthy insights.

A Big Data platform gives businesses the opportunity to analyze all of their data (whole population analytics) and to gain a better understanding of their business, their customers, the marketplace, and so on. This opportunity leads to the Big Data conundrum: Although the economics of deletion have caused a massive spike in the data that's available to an organization, the percentage of the data that an enterprise can understand is on the decline. A further complication is that the data that the enterprise is trying to understand is saturated with both useful signals and lots of noise (data that can't be trusted or that isn't useful to the business problem at hand).

Embedded within all of this noise are potential useful signals: The person who professes a profound disdain for her current smart phone manufacturer and starts a soliloquy about the need for a new one is expressing monetizable intent. Big Data is so vast that quality issues are a reality, and *veracity* is what we generally use to refer to this problem domain; no one believes everything they read on the Internet, do they? The fact that one in three business leaders don't trust the information that they use to make decisions is a strong indicator that veracity needs to be a recognized Big Data attribute.

Cognitive Computing

Technology has helped us go faster and go further—it has literally taken us to the moon. Technology has helped us to solve problems previous generations couldn't imagine, let alone dream about. But one question that we often ask our audiences is “Can technology think?” Cognitive systems is a category of technologies that uses natural language processing and machine learning (we talk about these disciplines in Chapter 6) to enable people and machines to interact more naturally and to extend and magnify human expertise and cognition. These systems will eventually learn, think, and interact with users to provide expert assistance to scientists, engineers, lawyers, and other professionals in a fraction of the time it now takes.

When we are asked the “Can technology think?” question, our answer is “Watson can!” Watson is the IBM name for its cognitive computing family that delivers a set of technologies unlike any that has come before it. Rather than forcing its users to think like a computer, Watson interacts with humans...on human terms. Watson can read and understand natural language, such as the tweets, texts, studies, and reports that make up the majority of the world's data—a simple Internet search can't do that. You likely got your first exposure to cognitive computing when IBM Watson defeated Brad Rutter and Ken Jennings in the *Jeopardy!* challenge. This quiz show, known for its complex, tricky questions and smart champions, was a perfect venue for the world to get a first look at the potential for cognitive computing.

To play, let alone win, the *Jeopardy!* challenge, Watson had to answer questions posed in every nuance of natural language, including puns, synonyms and homonyms, slang, and jargon. One thing you might not know is that during the match, Watson was not connected to the Internet. It knew only what it had amassed through years of interaction and learning from a large

set of unstructured data. Using machine learning, statistical analysis, and natural language processing to find and understand the clues in the questions, Watson compared possible answers by ranking its confidence in their accuracy. What's more, without machine learning to build confidence in its answers, Watson wasn't going to beat any *Jeopardy!* challenge winner.

Back when Watson was a trivia guru, it could sort through the equivalent of 200 million pages of data to uncover an answer in three seconds. Today, the Watson technology is available on the cloud as a service, 24 times faster, and has gone from the size of a master bedroom to three stacked pizza boxes! Read that sentence again, and now imagine what your Big Data world is going to look like in a couple of years. This exponential increase in capacity has changed what Watson can do today, the things that IBM has planned on the horizon for tomorrow, and the things that Watson will do in the far future that we don't even know about yet.

Watson represents a first step into cognitive systems, a new era of computing. While it makes use of programmatic computing (more on that in a bit), it adds three additional capabilities that make Watson truly unique: natural language processing, hypothesis generation and evaluation, and dynamic learning. Although none of these capabilities by itself is unique to Watson, the combination delivers the power to move beyond programmatic computing and to unlock the world of global unstructured data. Watson technology empowers you to move from a keyword-based search that provides a list of locations to an intuitive, conversational means of discovering a set of confidence-ranked responses.

When asked a question, Watson generates an hypothesis and comes up with both a response and a level of confidence. Watson, however, also shows you the steps that it took to get to that response, which is a form of reasoning. You don't program Watson; you work with it. Through these interactions, it learns much like humans do. Every experience makes it smarter and faster.

Watson can also teach. For example, new generations of doctors are helping Watson learn the language of medicine. In turn, Watson is helping to teach doctors by providing possible treatment options. Watson (and Watson-like technologies) are now assisting doctors at Memorial Sloan Kettering with diagnostics by providing a variety of possible causes for a set of symptoms. Watson can help doctors narrow down the options and pick the best treatments for their patients. The doctor still does most of the thinking; cognitive computing doesn't replace human intelligence—it augments it. Watson is

there to make sense of the data and to help to make the process faster and more accurate.

Watson is also learning the language of finance to help wealth management advisors plan for their clients' retirement. For example, the IBM Watson Engagement Advisor can react to a caller's questions and help front-line service personnel find the right answers faster. The IBM Watson Discovery Advisor helps researchers uncover patterns that are hiding in mountains of data and then helps them share these new insights with colleagues. With Watson Analytics, you can ask Watson a question, and Watson then shows you what the data means in a way that is easy for anyone to understand and share—no techie required. Watson Data Explorer helps you to search, interrogate, and find the data wherever it might be—and this capability is a pillar to any Big Data and Analytics platform (we talk about this in the “A Big Data and Analytics Platform Manifesto” section in this chapter). Finally, there is also IBM Watson Developer's Cloud that offers the software, technology, and tools that developers need to take advantage of Watson's cognitive power over the cloud anytime.

For city leaders, such new systems can help them to prepare for major storms by predicting electrical outages. They can also help them to plan evacuations and prepare emergency management equipment and personnel to respond in the areas that will need help the most. These are just a few examples that show how Watson and cognitive computing are game changers. Personalization applications that help you to shop for clothes, pick a bottle of wine, and even invent a recipe (check out Chef Watson: www.bonappetit.com/tag/chef-watson) are another area in which IBM cognitive computing capabilities are redefining what Big Data solutions *can* do and what they *should* do.

The one thing to remember about Watson long after you've read this book is that *Watson provides answers to your questions*. Watson understands the nuances of human language and returns relevant answers in appropriate context. It keeps getting smarter, learning from each interaction with users and each piece of data it ingests. In other words, you don't really program Watson; you learn with it.

You might be familiar with Apple's Siri technology. People often ask us, “What's the difference?” Siri is preprogrammed, and all possible questions

and her answers must be written into the application using structured data. Siri does not learn from her interactions either. Watson, on the other hand, reads through all kinds of data (structured, semistructured, and unstructured) to provide answers to questions in natural language. Watson learns from each interaction and gets smarter with time through its machine learning capabilities. We were pretty excited when IBM and Apple recently announced their partnership because we believe that Siri and Watson are going to go on a date and fireworks are going to happen! We also see the potential for IBM technologies such as the NoSQL Cloudant document store to get deeply embedded into iOS devices, thereby changing the game of this technology for millions of users.

Why Does the Big Data World Need Cognitive Computing?

You can more or less categorize the last century and a bit of computing into three eras.

- **The Tabulating System Era** Circa 1900 onward. Think tabulation machines and calculators.
- **The Programmable System Era** Circa 1950 onward. Programmable computers to run analyses in a deterministic manner.
- **Today's Cognitive Systems Era** Circa 2011 onward. Systems that learn and interact with users through natural language, as well as some other technologies under the IBM Watson brand.

The systems of yesterday and today delivered tremendous business value and societal benefits by automating tabulation and harnessing computational power to boost both enterprise and personal productivity. Cognitive systems will forever change the way people interact with computing systems. People will extend their expertise across any domain of knowledge and make complex decisions involving the extraordinary volumes and types of data that can be found on the domain in the fast-moving world of Big Data.

The need for cognitive computing comes from the Big Data world in which we live. Figure 1-1 shows why cognitive computing is *the* requirement for the next generation of understanding, insight, and productivity.

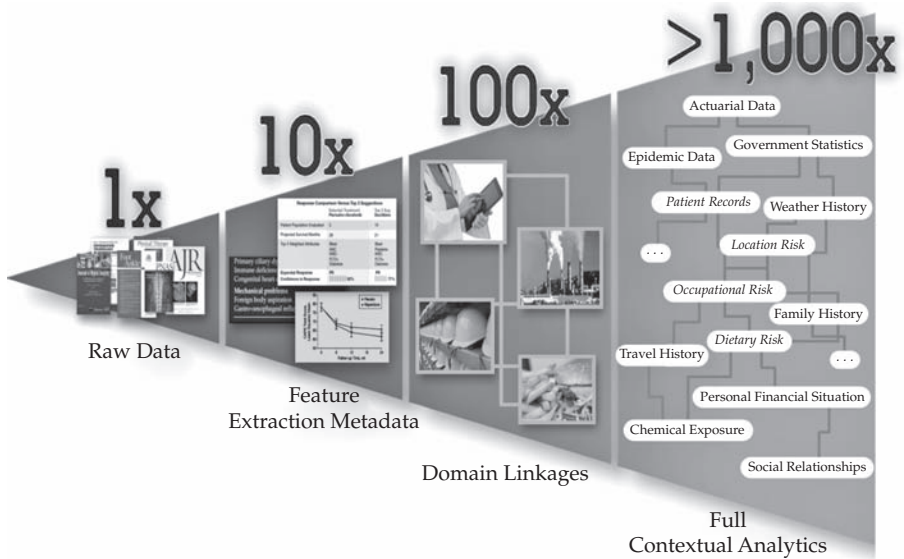


Figure 1-1 Big Data analytics and the context multiplier effect

Let’s assume you use a wearable device that tracks your steps during the day and sleep patterns at night. In this case, the raw data is the number of steps and hours of sleep. From this raw data, the device can calculate the number of calories burned in a day; this is an example of *feature extraction*. You can also use this device’s software to see at what points in the day you are active; for example, you were highly active in the latter part of the day and sedentary in the morning. During your sleeping hours, the device is able to leverage the metadata to map your sleeping habits. This device’s accompanying application also enables you to log what you eat in a day and how much you weigh; this is a great example of *domain linkages*. You are mixing information about diet with physical markers that describe your overall sleep and activity levels. There might also be a social ecosystem where others share information such as occupation, location, some family medical history (such as “My dad died of a heart attack”), travel information, hobbies, and so on. Such a community can share information and encourage friendly competition to promote health. Such an infrastructure can represent a corpus for full contextual analytics. It can factor in other variables, such as the weather, and uncover trends, such as low activity, and alert you to them. This is a

simple example, the point is that as interest in such a problem domain grows, so too does the data. Now think about the field of cancer research. Consider how many publications in this area are generated every day with all the clinical trials and the documentation behind both successes *and* failures. Think about all the relevant factors, such as a patient's family history, diet, lifestyle, job, and more. You can see that when you want to connect the dots—including dots that you can't even see at first—the data is going to get unmanageable and cognitive help is needed.

That help can come in the form of the new class of industry-specific solutions called cognitive systems, of which IBM Watson is a good example. The next generation of problem solvers is going to learn much faster than we ever did with cognitive computing technologies like Watson, and, in turn, Watson will learn much faster with our help. Cognitive computing will enable developers to solve new problems and business leaders to ask bigger questions. Together, people and technology will do things that generations before could not have imagined.

A Big Data and Analytics Platform Manifesto

To help an analytics-focused enterprise reach its potential, it is important to start with a checklist of “imperatives” for a Big Data and Analytics platform. We know that the trusted platforms of yesterday and, in some cases, today have limitations. We see these limitations daily as companies strive to transform themselves into data-driven, decision-making organizations. The limitations of traditional approaches have resulted in failed projects, expensive environments, non-scalable deployments, and system slowdowns.

You can't have a Big Data platform without relational database management systems (RDBMSs) or Hadoop. You need a framework to process and understand natural language. The data in your platform has to be governed and integrated; some data must have a sense of veracity to it, while others you cannot necessarily trust in the name of discovery. Moreover, there is no point in hosting a lot of data if you can't find it. In a Big Data world, it's no longer a “find the needle in a haystack” challenge; it's a “find the needle in a stack of needles” challenge.

The Big Data and Analytics platform that you build has to tackle *from the beginning* one of the biggest challenges that we have seen yet: how to realize

24/7 *decisioning* (rather than just 24/7 data collection). A big part of that capability, and surely the focus of the next inflection point, is cognitive computing.

When putting together the building blocks of a Big Data and Analytics platform, it is imperative that you start with the requirement that the platform *must* support all of your data. It must also be able to run all of the computations that are needed to drive your analytics, taking into consideration the expected service level agreements (SLAs) that will come with multitiered data. Figure 1-2 shows a vendor-neutral set of building blocks that we feel are “must have” parts of a true Big Data platform—it’s our Big Data and Analytics Manifesto.








Cognitive Computing	Discover, Explore, and Navigate Big Data Sources		Federated Discovery, Search, and Navigation	①
	Land, Manage, and Store Huge Volumes of Any Data		Hadoop File System Hadoop Processing 24/7 Data Collection	②
	Structure and Controlled Data		In-Memory Analytics, MPP Analytic Appliances for Extreme Performance	③
	Analyze Unstructured Data		Text Analytics Engine Content Analytics Video/Audio Analytics	④
	Analyze Data in Real Time		Stream Computing 24/7 Decisioning	⑤
	Rich Library of Analytical Functions and Tools		In-Database Analytics Libraries Big Data Visualization	⑥
	Integrate and Govern All Data Sources		Integration, Data Quality, Security, Lifecycle Management, MDM	⑦

Figure 1-2 A Big Data and Analytics platform manifesto: imperatives and underlying technologies

1. Discover, Explore, and Navigate Big Data Sources

We believe that when it comes to boosting your analytics IQ, we are all guilty of the same thing: not knowing the things that we could already know. Considering the amount of data that is continually pouring into organizations, data collection is not the issue. Tools such as Google Desktop or Apple Spotlight are designed to find “stuff” on your laptop that you don’t know or

forgot you have. It's the same in the analytics world. With all the hype about Big Data, it's easy to see why companies can't wait to analyze new data types and make available new volumes of data with dreams of epiphanies that are just "around the corner." But any Big Data project should start with the "what you could already know" part. From personalized marketing to finance to public safety, your Big Data project should start with what you have. In other words, know what you have before you make big plans to go out and get more. Provision search, navigation, and discovery services over a broad range of data sources and applications, both inside and outside of your enterprise, to help your business uncover the most relevant information and insights. A plan to get more mileage out of the *data that you already have* before starting other data gathering projects will make understanding the new data that much easier.

The process of data analysis begins with understanding data sources, figuring out what data is available within a particular source, and getting a sense of its quality and its relationship to other data elements. This process, known as *data discovery*, enables data scientists to create the right analytic models and computational strategies. Traditional approaches required data to be physically moved to a central location before it could be discovered. With Big Data, this approach is too expensive and impractical.

To facilitate data discovery and unlock resident value within Big Data, the platform must be able to discover data that is *in place*. It has to be able to support the indexing, searching, and navigation of different sources of Big Data. Quite simply, it has to be able to facilitate discovery in a diverse set of data sources, such as databases, flat files, or content management systems—pretty much any persistent data store (including Hadoop) that contains structured, semistructured, or unstructured data.

And don't forget, these cross-enterprise search, discovery, and navigation services must strictly adhere to and preserve the inherent security profiles of the underlying data systems. For example, suppose that a financial document is redacted for certain users. When an authorized user searches for granular details in this document, the user will see all the details. But if someone else with a coarser level of authorization can see only the summary when directly accessing the data through its natural interfaces, that's all that will be surfaced by the discovery service; it could even be the case that the search doesn't turn up any data at all!

In a Big Data world, it's not just about being able to search your data repositories (we cover these repositories in Chapter 4); it's also about creating a platform that promotes the rapid development and deployment of search-based applications that leverage the Big Data architecture that you've put in place—and that's the point, right? You aren't investing in Big Data because you don't want people to use the data. These aren't data science projects; they are strategic business imperatives.

When you invest in the discovery of data that you already have, your business benefits immensely. Benefits include enhanced productivity with greater access to needed information and improved collaboration and decision making through better information sharing.

It's simple. In the same manner that humans forget stuff they used to know as they get older, organizations forget things too. We call it organizational amnesia and it's way more of an epidemic than you think. Keeping in mind that innovation is facilitated by making information available to empowered employees, when you start having conversations about getting beyond the Big Data hype, don't forget to include the information you already have—the information that you may have just forgotten about.

2. Land, Manage, and Store Huge Volumes of Any Data

Few technologies have gotten more buzz over the past few years than Hadoop (the subject of Chapter 6). When coupled with Big Data, you have enough hype to write a whole library of trendy technical books. There's a reason for all the excitement around these technologies. Traditional data storage and analytics tools have not been cutting it when dealing with Big Data. From a *volume* perspective, many tools start to become impractical when thresholds in the dozens of terabytes are exceeded. Sometimes, "impractical" means that the technology simply won't scale any further or that a tipping point is reached in terms of how much time it takes to work with data sets. And sometimes, "impractical" means that although the technology can scale, the licensing, administrative, and hardware costs to deal with increased data volumes become unpalatable. From a *variety* perspective, traditional analytic tools work well with only structured data, which represents, at most, 20 percent of the data in the world today.

Considering the pressing need for technologies that can overcome the volume and variety challenges for data at rest, it's no wonder that business magazines and online tech forums alike are buzzing about Hadoop. And it's not all talk either. The IT departments in most Fortune 500 companies have done more than a little experimentation with Hadoop. The problem is that many of these initiatives have stagnated in the "science project" phase. The challenges are common: It is easy and exciting to start dumping data into these repositories; the hard part comes with what to do next. The meaningful analysis of data that is stored in Hadoop requires highly specialized programming skills, and for many algorithms, it can be challenging to put them into a parallelizable form so that they can run in Hadoop. And what about information governance concerns, such as security and data lifecycle management, where new technologies like Hadoop don't have a complete story?

Some traditional database vendors consider Hadoop to be no more than a data preparation opportunity for their data warehousing projects. We disagree with that view and discuss how we see Hadoop at play for analytics, data integration, operation excellence, discovery and more in Chapter 6.

It is not just Hadoop that's at play here. The JSON specification is ubiquitous for how mobile information is encoded today. Understanding relationships between entities (people, things, places, and so on) is something that requires the use of graphs. Some data needs to "bit bucket" to just store any data and retrieve it via a simple key. All of these fall under the NoSQL database genre, and we cover that in Chapter 2.

3. Structured and Controlled Data

Despite what some might want you to believe, we are not seeing the end of the RDBMS era. In fact, it is quite the opposite. A Big Data and Analytics platform needs to be *polyglot* (see Chapter 2), and that means you are going to end up using NoSQL, Hadoop, *and* RDBMS technologies. These technologies support various information management *zones*, which we introduce you to in Chapter 4. One of the zones requires that data be highly structured, controlled, and harvested.

If we had to pick an area that comes close to being as hot as Hadoop or Big Data, it would be *in-memory columnar databases*. In-memory computing offers the advantage of speed in data-intensive processing for analytics and

reporting. As the cost of computer memory drops, in-memory processing “obliterates the wait” for business answers, and RDBMS technologies are behind this success. What’s more, as the memory in CPU caches continues to expand, there is even more opportunity to get data into the CPU cache, which is inordinately faster than system RAM. These technologies are appealing because they typically require much less DBA involvement compared to traditional solutions. For example, performance-tuning objects such as indexes are not needed or created, which allows data scientists and line-of-business (LOB) owners to self-serve their analytics. Just load the data and go! Indeed, the future buyer of IT is the LOB, and collectively, they are driving the demand for software as a service (SaaS) analytics; IBM dashDB and Amazon Redshift are perfect examples of cloud-driven SaaS analytics services (we talk about this in Chapter 3).

Some of your data might be populated from the Hadoop repository, as discussed in the previous section, but some of it might come from another source that is trending in the structured and controlled data world: the analytical appliance (Chapter 9). Traditional architectures decoupled analytical environments from data environments. Analytical software would run on its own infrastructure and retrieve data from back-end data warehouses or other systems to perform complex analytics. The rationale was that data environments were optimized for faster data access but not necessarily for advanced mathematical computations. Hence, analytics were treated as a distinct workload that had to be managed in a separate infrastructure. This architecture was expensive to manage and operate, created data redundancy, and performed poorly with increasing data volumes. The analytic architecture prescribed in this manifesto is one that can run both data processing and complex analytics on the same platform. It needs to deliver petabyte-scale throughput by seamlessly executing analytic models inside the platform, against the entire data set, without replicating or sampling data. It must also enable data scientists to iterate through different models more quickly to facilitate discovery and experimentation with a “best fit” yield.

4. Manage and Analyze Unstructured Data

For a long time, data has been classified on the basis of its type—structured, semistructured, or unstructured. Existing infrastructures typically have barriers that prevent the seamless correlation and holistic analysis of these

data classifications, and independent systems have been used to store and manage these different data types. We've also seen the emergence of hybrid systems that can be disappointing because they don't natively manage all data types. For the most part—there are some exceptions—we find the “trying to fit a circle shape through a square hole” approach comes with a lot of compromise.

Of course, organizational processes don't distinguish among data types. When you want to analyze customer support effectiveness, structured information about a customer service representative's conversation (such as call duration, call outcome, customer satisfaction survey responses, and so on) is as important as unstructured information gleaned from that conversation (such as sentiment, customer feedback, and verbally expressed concerns). Effective analysis needs to factor in *all* components of an interaction and to analyze them within the same context, *regardless* of whether the underlying data is structured or not. A Big Data and Analytics platform must be able to manage, store, and retrieve both unstructured and structured data, but it must also provide tools for unstructured data exploration and analysis. (We touch on this in Chapter 6, where we cover Hadoop.)

This is a key point that we do not want you to miss because it is often overlooked. You likely do not have the appetite or the budget to hire hundreds of expensive Java developers to build extractors. Nor are you likely a LinkedIn- or Facebook-like company; you have a different core competency—your business. To process and analyze the unstructured data that is found in call logs, sentiment expressions, contracts, and so on, you need a rich ecosystem in which you can develop those extractors. That ecosystem must start with a tool set: visually rich tooling that enables you to compose and work with the extractions. You will want to compose textual understanding algorithms through a declarative language; in other words, you should not have to rely on bits and bytes Java programming, but rather on an app-based composition framework that is more consumable throughout the enterprise. If you want to query Hadoop data at rest, you are likely looking for an SQL-like interface to query unstructured data. This way, you can leverage the extensive skills in which you have invested. Finally, the ecosystem should have the ability to compile the higher-level declarations (the queries that you write) into code that can be executed in an efficient and well-performing manner, just like the SQL that is optimized under the covers by every RDBMS vendor.

Of course, you can apply the same requirements to video and audio, among other data types. For example, IBM hosts what we believe to be the world's largest image classification library. Pictures are retrieved on the basis of attributes that are "learned" through training sets. Imagine combing through thousands of photos looking for winter sports without having to tag them as winter sports. You can try it for yourself at <http://mp7.watson.ibm.com/imars/>.

5. Analyze Data in Real Time

Performing analytics on activity as it unfolds presents a huge untapped opportunity for the enterprise. For the most part, we still see analytic models and computations running on at-rest data that's stored in repositories (traditional databases, Hadoop, and so on). Although this works well for transpired events that occurred a few minutes, hours, or even days before, these repositories generally rely on disk drives to store and retrieve data. Even the best-performing disk drives and memory-optimized databases have unacceptable latencies when reacting to certain events in real time.

Enterprises that want to boost their analytics IQ need the ability to analyze data *as it is being generated* and then take appropriate action. The idea is to derive insight *before* the data is stored. We refer to this type of data as *streaming data* and the analysis of it as the *analytics of data in motion*. Depending on the time of day or other contexts, the volume of the data stream can vary dramatically. For example, consider a stream of data that is carrying stock trades in an exchange. Depending on trading events, that stream can quickly swell from 10 to 100 times its normal volume. This implies that a Big Data platform not only has to support analytics on data at rest, but also has to effectively manage increasing volumes of data streams and support analytics on data in motion. To get beyond the Big Data hype, you have to ensure you are not just having conversations about analytics on data at rest. You have to converse about how insights gleaned through analytics on at-rest data get applied to data in motion. We cover this topic in Chapter 7.

6. A Rich Library of Analytical Functions and Tools

One of the key goals of a Big Data and Analytics platform should be to reduce the *analytic cycle time*, the amount of time that it takes to discover and transform data, develop and score models, and analyze and publish results. When

your platform empowers you to run extremely fast analytics, you have the foundation on which to support multiple analytic iterations and speed up model development. Although this is the goal, there needs to be a focus on improving developer productivity. By making it easy to discover data, develop and deploy models, visualize results, and integrate with front-end applications, your organization can enable practitioners, such as analysts and data scientists, to be more effective. We refer to this concept as the *art of consumability*. Consumability is key to democratizing Big Data across the enterprise. You shouldn't just *want*; you should *always demand* that your Big Data and Analytics platform flatten the time-to-analysis curve with a rich set of accelerators, libraries of analytic functions, and a tool set that brings agility to the development and visualization process. Throughout this book, we illustrate how this part of the manifesto is rooted across the IBM Big Data and Analytics platform.

A new take on an old saying is appropriate here: "*A picture is worth a thousand terabytes.*" Because analytics is an emerging discipline, it is not uncommon to find data scientists who have their own preferred approach to creating and visualizing models. It's important that you spend time focusing on the visualization of your data because that will be a key consumability attribute and communication vehicle for your platform. It is *not the case* that the fancier the visualization, the better—so we are not asking you to use your Office Excel 3D mapping rotations here. That said, Big Data does offer some new kinds of visualizations that you will want to get acquainted with, such as Streamgraphs, Treemaps, Gantt charts, and more. But rest assured, your eyes will still be looking over the traditional bar charts and scatter plots too.

Data scientists often build their models by using packaged applications, emerging open source libraries, or "roll your own" approaches to build models with procedural languages. Creating a restrictive development environment curtails their productivity. A Big Data and Analytics platform needs to support interaction with the most commonly available analytic packages. There should also be deep integration that facilitates the pushing of computationally intensive activities, such as model scoring, from those packages into the platform on premise or in the cloud. At a minimum, your data-at-rest and data-in-motion environments should support SAS, SPSS, and R. The platform also needs to support a rich set of "parallelizable" algorithms (think: machine learning) that has been developed and tested to run on Big Data (we cover all

of this in Chapter 6), with specific capabilities for unstructured data analytics (such as text analytic routines) and a framework for developing additional algorithms. Your platform should provide the ability to visualize and publish results in an intuitive and easy-to-use manner.

7. Integrate and Govern All Data Sources

Over the last few years, the information management community has made enormous progress in developing sound data management principles. These include policies, tools, and technologies for data quality, security, governance, master data management, data integration, and information lifecycle management. These principles establish veracity (trust) in the data and are extremely critical to the success of any analytics program. For example, if a certain masking policy is used for credit card data in the RDBMS, why isn't the same policy used to mask the data when it resides in Hadoop's HDFS file system? If you want to know where a certain set of data comes from and what was done to it before you draw inferences from it, does it matter where the data resides? No! Unfortunately, from what we can tell, although many agree with our mindset here, many don't practice it.

We struggled when it came to the location of this part of the manifesto in Figure 1-2. Its current location suggests that the entire Big Data and Analytics platform should sit on the ability to integrate and govern all data sources—which is our intent. On the other hand, we see too many people treat this topic as an afterthought—and that leads to security exposure, wasted resources, untrusted data, and more. We actually think that you should scope your Big Data architecture with integration and governance in mind from the very start. In fact, you should probably do this for all the data assets in your organization. Think about it this way: Deciding *not* to govern some “toss-away” data is actually a governance decision. By handling all data that you are keeping for analytics as though it will be governed, you are on the right path to a modern information management architecture.

We get it. Governance can be akin to when your mother asked you to clean up your room; you didn't like it, but at the same time, you now know how good it was for you. We will tell you this: Organizations that do more than take a minimal effort to comply with an approach to governance end up creating regulatory dividends by repurposing the same data for other uses. A Big Data and Analytics platform should embrace these principles and treat

them as foundational and intrinsic to the platform itself, which is a guideline that represents the main theme of Part III.

Cognitive Computing Systems

As we discussed earlier in this chapter, cognitive computing systems learn from and interact naturally with people to extend what either humans or machines could do on their own. They help human experts to make better decisions by penetrating the complexity of Big Data. When you look at the capabilities that are outlined in our Big Data and Analytics platform manifesto, it becomes obvious how these capabilities can feed cognitive systems so that artificial intelligence meets business intelligence. That is where the next frontier of discovery is to be found, be it a cure for cancer or concierge shopping services. For IBM, the place where data is prepared for cognitive systems is Watson Foundations, the focus of Part II.

Of Cloud and Manifestos...

We didn't explicitly come out and say it (yet), but the cloud is going to play a big part of your Big Data and Analytics platform. Whether you are using it for bursty workloads that need the scalability relief that this deployment model can offer or for turnkey discovery, you should ensure that the capabilities that are described in this manifesto are available as composable services in the cloud if it makes sense to do so.

When you assemble your Big Data and Analytics platform, ensure that the vendors with which you partner showcase their ability to deliver the services that are outlined in this manifesto, in multiple modalities, to suit your business needs.

Cloud-based delivery models for IT infrastructure, frictionless development, and service-provisioned capabilities are so critical to a Big Data strategy that we dedicated an entire chapter to it in this book! For the most part, the IBM Information Management capabilities cloud service offerings (such as IBM DataWorks and dashDB) will drive the innovations that you will see in its on-premise appliance and software solution offerings (such as the BLU Acceleration technology in DB2). This is all part of a *cloud-first* strategy that is born "agile" and drives IBM's ability to more quickly deliver Big Data analytic capability to the marketplace in the manner in which it will be consumed.

Now reflect on some of the required Big Data and Analytics platform capabilities we outlined in our manifesto. It means you may want to leverage a cloud service to perform one of the many data refinery activities available in the IBM Cloud. The catalog of these services is collectively known as IBM DataWorks. IBM BLU Acceleration technology is available as a turnkey analytics powerhouse service called dashDB. (dashDB also includes other services, some of which are found in the IBM DataWorks catalog, along with Netezza capabilities and more.)

The cloud-first strategy isn't just for IBM's Information Management brand either. Watson Analytics is a set of services that provide a cloud-based predictive and cognitive analytics discovery platform that's purposely designed for the business user (remember the new buyer of IT we talked about earlier in this chapter).

Need a JSON document store to support a fast-paced and agile app dev environment? It's available in the IBM cloud as well; Hadoop cluster, yup; SQL data store, yup—you get the point. From standing up computing resources (infrastructure as service), to composing applications (platform as a service—think IBM Bluemix), to running them (the IBM Cloud marketplace SaaS environment), the richest capability of service we know of is available from IBM. These services are mostly available in a freemium model to get going with options for enterprise-scale use and sophistication. Finally, the IBM Cloud services catalog doesn't just contain IBM capabilities, it includes services delivered through a rich partner ecosystem and open source technologies too.

Wrapping It Up

In this chapter, we introduced you to the concept of Big Data. We also highlighted the future of analytics: cognitive computing. Finally, we shared with you our Big Data and Analytics platform manifesto.

We kept most of this chapter vendor agnostic. We wanted to give you a solid background against which you can evaluate the vendors that you choose to help you build your own Big Data and Analytics platform and to describe the things to look for when you design the architecture. In short, this chapter gave you the key points to guide you in any Big Data conversation. The remaining chapters in this part of the book cover NoSQL and the cloud.

By the time you finish this part, you will have a solid understanding of the main topics of discussion in today's data-intensive analytics landscape.

Of course, it goes without saying that we hope you will agree that IBM is a Big Data and Analytics partner that has journeyed further with respect to strategic vision and capabilities than any other vendor in today's marketplace. And although we would love you to implement every IBM solution that maps to this manifesto, remember that the entire portfolio is modular in design. You can take the governance pieces for database activity monitor and use those on MongoDB or another vendor's distribution of Hadoop. Whatever vendor, or vendors, you choose to partner with on your Big Data journey, be sure to follow this Big Data and Analytics manifesto.

This page intentionally left blank

2

To SQL or Not to SQL: That's Not the Question, It's the Era of Polyglot Persistence

The word *polyglot* is used to refer to an individual who knows multiple languages. What does this have to do with information technology? Well, this is the term that's been adopted to describe what IBM calls a *Next Generation Data Platform Architecture*, one that combines the use of multiple persistence (storage) techniques to build, store, and manage data applications (apps) in the most efficient way possible. We believe that apps will become polyglot in the sense that end users interact with a single interface, while the back-end app is a combination of persistence technologies and APIs working in harmony “under the covers.” If you look at the IBM Bluemix platform as a service (PaaS) model, which we talk about in the next chapter, it becomes apparent how next-generation apps will be composed of a set of polyglot services. What's more, consuming functionality through the use of an API service is the future of app development—the API economy we call it. So beyond just data, the apps of the future (and many of today) are polyglot.

Polyglot persistence is a handyman's toolbox approach to solving complex data-layer problems. The data platform (toolbox) of tomorrow—which was really needed yesterday—has to bring various technologies (tools) to support a polyglot environment. This enables organizations to break down

complicated data-layer problems into segments and to select the best technology for each problem segment. No technology completely replaces another; there are existing investments in database infrastructure, skills honed over decades, and existing best practices that cannot (and should not) be thrown away. Innovation around polyglot solutions and NoSQL technologies will be *complementary* to existing SQL systems. The question is and always has been “What is available to help the business, and what is the problem that we are trying to solve?” If you’re having a Big Data conversation that’s intended to get beyond the hype, you need to verse yourself with polyglot concepts—the point of this chapter.

The easiest way to look at polyglot persistence is first to understand the emergence of the two most ubiquitous data persistence architectures in use today: SQL and NoSQL. (We are setting aside for the moment another, more recent approach, which is aptly named NewSQL.) NoSQL is not just a single technology, a fact that makes any conversations on this topic a bit more complicated than most. There are many kinds of NoSQL technologies. Even the NoSQL classification system is full of complexity (for example, the data model for some columnar family stores is a key/value format, but there are key/value databases too). In fact, the last time we looked, there were more than 125 open source NoSQL database offerings! (For a summary of the current landscape, see 451 Group’s Matt Aslett’s data platforms landscape map at <http://tinyurl.com/pakbgbd>.)

The global NoSQL market has been forecasted to be \$3.4 billion by 2020, experiencing 21 percent compounded annual growth rates (CAGRs) between 2015 and 2020, according to the value-added reseller firm Solid IT. The bottom line is that NoSQL technologies continue to gain traction (some much more than others); but the majority have ended up as spins-offs or off-shoots or have just faded away. In this chapter we are going to talk about the styles of NoSQL databases and refer to some of the ones that have made their mark and didn’t fade away.

You likely interact with NoSQL databases (directly or indirectly) on a daily basis. Do you belong to any social sites? Are you an online gamer? Do you buy things online? Have you experienced a personalized ad on your mobile device or web browser? If you answered “yes” to any of these questions, you’ve been touched by NoSQL technology. In fact, if you’re a Microsoft Xbox One gamer, have leveraged the Rosetta Stone learning platform to

make yourself a polyglot, bought concert tickets through Ticketmaster, or put a document in Dropbox, you're actually interacting with one of IBM's NoSQL offerings, Cloudant, which handles more than 10 billion transactions per day and supports millions of databases (see Chapter 10). NoSQL comes from an evolution in the IT landscape that is characterized by massive amounts of data around *interactions*—not just transactions. This is the rapid-fire “fuzzy” data that powers today's *systems of engagement* (more about these systems later), smart devices, and web apps. To address this shift, new models of app development, data storage, and data access are needed. In fact, you will find that NoSQL has an even larger role to play as we move from people-driven systems of engagement to the engagement of “things”—the Internet of Things.

In this chapter, we introduce you to the world of polyglot persistence and why it's critical for enterprises to adopt this methodology. We dispel some doomsday myths and articulate how the IBM Big Data and Analytics platform's DNA has been engineered for this specific methodology. We focus on NoSQL in this chapter because that area is likely newer for most readers. We assume that relational database management systems (RDBMSs) are well understood in the organization where you build, test, and deploy apps, so won't talk about them here unless it helps the NoSQL discussion.

By the time you finish reading this chapter, you will have a firm understanding of what NoSQL is all about, including its applicability in a Big Data world (hint: it's not what you put on your résumé if you don't have any SQL skills). You'll also appreciate how a polyglot environment can deliver enormous benefits and economies of scale to your business and that single technology approaches are not always up to the task for the requirements at hand; indeed, the “one ring to rule them all” approach is best left to Tolkien books.

Core Value Systems: What Makes a NoSQL Practitioner Tick

One of the key reasons behind the emergence of NoSQL is its practitioners. Consider database administrators (DBAs), developers, and the term *performance*. Performance to a DBA typically means in relation to a service level agreement (SLA): “Is it running as fast as I promised that it would run?” To developers performance means: “How fast can I build my app?”

Think about it—today’s social-mobile-cloud apps evolve so rapidly, and the barrier to entry for new apps is so flat, thanks to the cloud, that developers need two things: an endless supply of super-caffeinated beverages and an app lifecycle that supports a near-continuous innovation and integration of code changes.

Developers deserve and expect better than “good enough” technologies, and their tools and infrastructure must continually improve. At first, this doesn’t sound like great news for folks on the operations side of the house. Developers have become so influential in our social-mobile-cloud world that the emergence of as-a-service cloud models and the marriage between developers and operations personnel (DevOps) directly reflect the need for businesses to be agile if they want to be innovative. The IBM Bluemix platform as a service (PaaS) development ecosystem is all about agility—it provides developers with hundreds of IBM, non-IBM, and open source services to compose apps in a hosted environment powered by Softlayer—and that includes a number of NoSQL services. (We talk about Bluemix and the cloud in Chapter 3.)

Developers are the most visible force driving NoSQL technologies because line-of-business (LOB) owners are asking for quick delivery, and developers are the ones building the apps. There’s an undeniable confluence between the changing nature of the data in a Big Data world and emerging technologies that seek to quench the thirst by developers for better ways to not only persist that data, but work with it in an agile manner too. Make no mistake about it: NoSQL is here to stay.

To better appreciate a developer’s demand for agility, let’s assume you’re a developer who is enhancing a mobile shopping cart app for your company. You’ve been tasked with personalizing this app by an executive team that believes this enhancement will help drive users to make more “no-thought, on-the-spot” purchases. You’ve been on this project for two months, coming off the heels of an earlier project that took three months to complete. Your life is characterized by these types of project-to-project assignments. As you design the personalization aspects of your app, you decide that it’s a great idea to add a Facebook ID field so that tailored offers can be shared within the user’s network.

Traditionally, if the data is stored in an RDBMS, the developer has to go to the DBA and request a database schema change. The DBA schedules the work required to fulfill the request. Perhaps the developer will see something in the test system within a day or two (“We’ll let you know when it’s

done” is the stuff that makes developers cringe). Now, think about this app in production...your change request to surface the new Facebook scoring module falls into the realm of DBA “limbo.” Ask colleagues who have been in similar situations how long this might take and the answer might shock you. In some companies, it’s months! (Additional data modeling could create even further delays.)

Developers want to be able to add, modify, or remove data attributes at will, without having to deal with middlemen or “pending approval” obstacles. They want to experiment, learn from failure, and be agile. If they are to fail, they want to fail fast (we talk about this methodology in Chapter 4). Agility is prized because mobile app users expect continuous functional enhancements, but the friction of traditional development approaches typically doesn’t allow for that. In most environments, developers have to keep “in sync” with DBA change cycles.

NoSQL databases appeal to these developers because they can evolve apps rapidly without DBA or data modeler intervention. There are other scenarios as well. Speed of transaction matters in a social-mobile-cloud world; depending on the app, many developers are willing to serve up “stale” results (by using data that might be slightly out of date and will in all likelihood eventually be right, it’s just not right now) in exchange for more responsive and faster apps—in some cases they are okay if the data is lost—we discuss these trade-offs later in this chapter. We think Big Data conversations that include polyglot open the door to a more satisfactory trade-off between consistency and availability.

This agility scenario sums up what we believe are the fundamental differences between the SQL and NoSQL worlds. There are others. For example, in the SQL world, a lot of the work to store data has to be done up front (this is often referred to as *schema-first* or *schema-on-write*), but getting data out is pretty simple (for example, Excel easily generates SQL queries). In the NoSQL world, however, it’s really easy to store data (just dump it in), and much of the work goes into programming ways to get the data out (*schema-later* or *schema-on-read*).

As previously mentioned, NoSQL was born out of a need for scalability. Most (not all) NoSQL technologies run on elastic scale-out clusters, but most RDBMSs are hard-pressed to scale horizontally for transactions. Microsoft SQL Server doesn’t have anything here that isn’t some kind of workaround; at the

time of writing, SAP HANA doesn't support scale-out for OLTP, and those with experience trying to get Oracle RAC to scale know the effort that is involved. From a transactional perspective, it's fair to give IBM's DB2 for System z and DB2 pureScale offerings due credit here. Their well-proven transactional scale-out and performance-optimizing services are made possible by IBM's Coupling Facility technology for bringing in and removing resources without having to alter your apps to achieve the near-linear performance gains that you would expect when scaling a cluster. It's also fair to note that almost every RDBMS vendor has at least a semi-proven solution for scaling out business intelligence (BI) apps—but these are very focused on structured data.

Table 2-1 lists a number of other differences between the NoSQL and SQL worlds. We aren't going to get into the details for each of them (or this would be solely a NoSQL book), but we touch on some throughout the remainder of this chapter. Take note that what we're listing in this table are general tendencies—there are exceptions.

The NoSQL World (Schema Later)	The SQL World (Schema First)
New questions? No schema change	New questions? Schema change
Schema change in minutes, if not seconds	Schema: permission process (we're not talking minutes here)
Tolerate chaos for new insights and agility (the cost of "getting it wrong" is low)	Control freaks (in many cases for good reasons)
Agility in the name of discovery (easy to change)	Single version of the truth
Developers code integrity (yikes!)	Database manages integrity (consistency from any app that access the database)
Eventual consistency (the data that you read might not be current)	Consistency (can guarantee that the data you are reading is 100 percent current)
All kinds of data	Mostly structured data
Consistency, availability, and partition tolerance (CAP)	Atomicity, consistency, isolated, durable (ACID)

Table 2-1 *Characteristics of the NoSQL and SQL Worlds*

What Is NoSQL?

In our collective century-plus experience with IT, we honestly can't recall a more confrontational classification for a set of technologies than NoSQL. More ironic is the popular trend of adopting SQL practices and terminology in NoSQL solutions. We're sure that there are practitioners who feel that the

name is apt because, as they see it, a war is coming—a sort of *Game of Thrones* (“The Database”) where the Wildings of the North (the NoSQL “barbarians”) are set to take over the regulated (and boring) world of RDBMSs. Although this sounds pretty exciting, it’s simply not going to happen. The term *NoSQL* was the happy (some would say not happy) result of a developer meet-up that was pressed to come up with a name for SQL-alternative technologies.

Popular NoSQL products—which we classify later in this chapter—include Cloudant, MongoDB, Redis, Riak, Couchbase, HBase, and Cassandra, among others. These products almost exclusively run on Linux, leverage locally attached storage (though sometimes network-attached storage), and scale out on commodity hardware.

SQL, which has been around for 40 years, is the biggest craze in the NoSQL Hadoop ecosystem space. SQL is so hot because it’s incredibly well suited to querying data structures and because it is a declarative language, which is so important because it enables you to get an answer without having to program *how* to get it.

This matters. It matters when it comes to broadening insights about your organization. A terrific example of this involves the IBM SQL on Hadoop engine (called Big SQL) available in InfoSphere BigInsights for Hadoop—a “just the query engine” stand-alone MPP Hadoop SQL processing engine that quite simply is the best in its class. The IBM approach is different than other vendors such as Oracle, Teradata, Pivotal, and Microsoft; they focus on a “use the RDBMS to either submit a remote query or run the entire query with the involvement of a database” approach. A true Hadoop query engine runs the MPP SQL query engine on Hadoop, uses Hive metadata, and operates on data stored in Hadoop’s file system using common Hadoop data types. (For details on SQL on Hadoop and Big SQL see Chapter 6).

We believe that Big SQL can deliver incredible performance benefits and is more suitable over anything we’ve seen in the marketplace, including Cloudera Impala. Why? First, IBM has been in the MPP SQL parallelization game for more than four decades; as you can imagine, there’s decades of experience in the Big SQL optimizer. Second, IBM’s Big SQL offers the richest SQL support that we know of today in this space—it supports the SQL used in today’s enterprises. That’s very different from supporting the SQL that’s easiest to implement, which is the tactic of many other vendors, like those that support query-only engines. If Hadoop SQL compatibility were a golf

game, we'd say that the IBM Big SQL offering is in the cup or a tap-in birdie, whereas others are in the current position of trying to chip on to save a bogey.

Don't lose sight of the motivation behind polyglot persistence: attaching the right approach and technology for the business problem at hand. Conversations around SQL and NoSQL should focus more on how the two technologies complement one another. In the same way that a person living in Canada is better served by speaking English *and* French, a data architecture that can easily combine NoSQL and SQL techniques offers greater value than the two solutions working orthogonally. In our opinion—and there are those who will debate this—the rise of NoSQL has really been driven by developers and the requirements we outlined earlier in this chapter. New types of apps (online gaming, ad serving, and so on), the social-mobile-cloud phenomenon, and the acceptance of something we call *eventual consistency* (where data might not necessarily be the most current—no problem if you are counting “Likes” on Facebook but a big deal if you're looking at your account balance) are all significant factors.

Is Hadoop a NoSQL Database?

Having referenced Hadoop in this chapter already, it is worth taking a moment to tackle a question that we get quite often: “Is Hadoop a NoSQL store?” This question can confuse a lot of people. We already touched on how Hadoop and IBM's Big SQL technologies are a great example of the ubiquitous demand for SQL support across almost all emerging data-related technologies, so it's a natural question to ask.

We classify Hadoop as an ecosystem of software packages that provides a computing framework. These include MapReduce, which leverages a K/V (key/value) processing framework (don't confuse that with a K/V database); a file system (HDFS); and many other software packages that support everything from importing and exporting data (Sqoop) to storing transactional data (HBase), orchestration (Avro and ZooKeeper), and more. When you hear that someone is running a Hadoop cluster, it's likely to mean MapReduce (or some other framework like Spark) running on HDFS, but others will be using HBase (which also runs on HDFS). Vendors in this space include IBM (with BigInsights for Hadoop), Cloudera, Hortonworks, MapR, and Pivotal. On the other hand, NoSQL refers to non-RDBMS SQL database solutions such as HBase, Cassandra, MongoDB, Riak, and CouchDB, among others.

At least for now, Hadoop is mostly used for analytic workloads. Over time, we expect that to change; indeed, parts of its ecosystem, such as HBase, are being used for transactional work and have emerging database properties for transactional control. Like NoSQL databases, Hadoop is implemented on commodity clusters that scale out and run on Linux (although Hortonworks did port its Hortonworks Data Platform (HDP) to Windows in partnership with Microsoft). The disk architecture behind this ecosystem is almost always locally attached disks.

Different Strokes for Different Folks: The NoSQL Classification System

As it turns out, NoSQL is a broad topic. To make it easier to talk about NoSQL, we will introduce you to the four “flavors” that the marketplace has settled on when it comes to classifying NoSQL databases. That being said, keep in mind that some of the classification lines are blurred because some flavors are being mixed for interoperation. For example, the NoSQL HBase database (shipped as part of the IBM InfoSphere BigInsights for Hadoop offering) has both key/value and columnar influences within its capabilities.

Give Me a Key, I’ll Give You a Value: The Key/Value Store

A *key/value* (K/V) database provides access to a value primarily through a key. As a simple analogy, think of the way you interact with files on your laptop as a key/value store (it isn’t...but as an analogy it works). Let’s assume you want to open a picture called `AnnaErikChloePicnic.jpg` located in the `c:\temp` directory. To work with this file, you would use the `c:\temp` key to locate it and then pull *all of the file* into your app for viewing. In a nutshell, that’s a K/V store. Values in a K/V store are generally said to be *opaque binary objects* because you can’t really see into them unless you pull all the values out, but you can pretty much store anything you want in them (we say “generally” because some K/V stores offer the ability to support partial key retrieval through manually defined metadata). Keeping in mind our analogy, for the most part, you can drop anything you want into a file system directory (text files, graphic files, audio) and locate the information by using the key,

which in this analogy's case is the directory pathname. In the same way you don't load half a photo, in a true K/V store, you don't fetch a portion of the values. (If you want to do that, you should likely be using a NoSQL document database, which we cover in the next section.) K/V stores are less suited for complex data structures as well; they typically can't be used as a substitute for JSON-encoded NoSQL document database stores, such as IBM Cloudant, which is more suited for persisting complex data structures.

K/V stores are popular for user sessions and shopping cart sessions because they can provide rapid scaling for simple data collections. For example, you might use a K/V store to create a front end for a retail web site's product catalog (this is information that doesn't change very often); if inconsistent changes do occur, they're not going to create major problems that grind service to a halt. NoSQL's eventually consistent data model applies conflict resolution strategies under the covers while the distributed database "agrees" (eventually) to a consistent view of your data. Data that is typically found in K/V stores isn't highly related, so practitioners rely on basic programmatic CRUD (create, read, update, delete) interfaces to work with them.

We estimate that K/V stores represent about 24 percent of the NoSQL market. Examples of K/V databases include MemcacheDB, Redis, Riak, Amazon DynamoDB, and the one we aren't supposed to say out loud (Volde-mort), among others. IBM provides HBase (packaged with the InfoSphere BigInsights for Hadoop offering, available on premise and off premise through the IBM Cloud), which includes a number of enterprise-hardening and accessibility features such as SQL and high-availability services. (HBase has columnar characteristics as well, so we return to that product in our discussion about columnar stores.) Finally, it's fair to note that IBM's Cloudant document store has been used with great success for many of the use cases mentioned earlier.

The Grand-Daddy of Them All: The Document Store

The *document store* is basically a K/V store, but instead of the value being a smattering of numbers, web session token IDs, or a string, it's a document, such as a customer record or even a book. It has structure that you can see and work with, and it is the most intuitive way for persisting objects that we can see and think about in the real world. Document stores are by far the most popular

NoSQL technology deployed today. Our research suggests that document stores represent more than 63 percent of today's NoSQL revenue landscape.

What you store in a NoSQL document database can be any kind of data at all, so you'll sometimes see this genre competing with the K/V store that we talked about in the previous section. In fact, we've seen ever-increasing adoption of document stores over K/V stores for apps that require a NoSQL data layer solution. For example, Cloudant (a JSON document data store) has rich-text searching facilities, geospatial capabilities, geoload balancing, and a *masterless* (not master-slave) architecture that is ideally suited for the data layer tasks that can hamper a K/V store. We've seen a number of clients implement Cloudant instead of Amazon Dynamo DB because of the markedly better performance and flexibility of Cloudant's data layer (flip to Chapter 10 for a detailed look at Cloudant). Other examples of document databases include Couchbase, MarkLogic, and MongoDB. As well, there are RDBMS solutions such as DB2 and Informix, which have introduced JSON services within their relational engines.

The key thing to remember about a document store is that it's a way to store and retrieve documents by way of a primary key, which serves as the document ID. The corresponding value is the document itself. Technologies in this space typically enable you to index the documents, and those documents can be nested (think of a family tree as an example). And as long as the incoming data meets some loose definition of a document, you are able to store that data in a document database. You can see some overlap with the pure K/V use cases right away; this type of NoSQL database is great for a product catalog, especially a frequently changing one! NoSQL document data stores are characteristically flexible: They are ideal for persisting objects where the schema is not well defined or is likely to change in the future; now reflect back on the kinds of apps we noted are being built for the social-mobile-cloud Big Data world. And moreover, you don't need a DBA to jump in and go through the mundane work of redefining a database schema—the schema for each document is self-describing, so that one document's schema can be altered without having to lock down the entire database to make the change. These are some of the reasons why the document store is the kingpin of the NoSQL world at this time. What kind of document gets stored in a document store? The document store genre of the NoSQL world is dominated by one word: *JSON*.

What Is JSON, and Why Is It the “New Cool”?

If you know what JSON is, you can skip this section. If you think that JSON is the name of the person in the corner office, please read on. We’re not going to cover all the details of JSON in this section, so if you want to learn more about it than what we cover here, check out www.JSON.org.

JSON stands for JavaScript Object Notation. It has become the de facto lightweight data interchange format and is predominantly used to exchange data between programs that were written in virtually any modern programming language. JSON has trumped XML as the industry’s darling markup language because of JSON’s readability (for both humans and machines), a self-describing schema, hierarchical data structures (you can have values nested within complex objects, such as having separate mobile and office phone numbers in a `phoneNumber` object), it’s less verbose, and more. JSON’s syntax will be familiar to anyone who has worked with a modern programming language, and it is designed to be minimalist (it’s shorter and more compact, which is great when you are sending data everywhere in a mobile world). For example, the XML representation of `<name>John</name>` can alternatively be expressed in JSON as `"name" : "John"`; you can appreciate how JSON not only presents data in a more readable format, but also significantly reduces the footprint of your documents.

We have mentioned that NoSQL databases typically have no predefined and rigid schemas for storing data; this leads to another benefit: modeling. It seems to us that using a single JSON document to define a product catalog is a lot easier than shredding and joining a number of tables in a third normal form (3NF) database schema. Because there is no predefined schema with JSON, it can represent an endless number of data structures. If a developer is able to persist multiple classes of data structures within the same database, then the process of designing a database to fit a client’s business model is simplified. For example, imagine a commercial retail store that wants to model its inventory; the fields and descriptors needed for products in the home cleaning supplies section will require very different data representations than a widescreen television from the electronics department. Using JSON, these two different product taxonomies can be stored side by side, thanks to the flexibility of JSON’s document-specific schema.

JSON is well suited to the NoSQL world because that world is not cluttered by the overuse of NULL values that you typically find in relational

database tables that store sparse data sets. (In a social-mobile-cloud world, there are hundreds or thousands of stored attributes, and not all of them have data values. Think of a detailed social profile that is barely filled out—this is an example of a sparse data set.)

The multitude of data types that you can represent with JSON are intuitive and comprehensive: strings, numbers, Booleans, nulls, arrays, and objects (nested combinations of these data types). JSON is composed with curly brackets (`{ }`) and can contain arrays (unlike XML), which are denoted with square brackets (`[]`). You use double quotation marks (`" "`) to denote a string. For example, `35` is a number on which you can perform arithmetic operations, whereas `"35"` is a string—both could be used for `age`, but the data type that you choose has implications for what you can do with the data.

In almost every use case that we've seen, JSON processing is much faster than XML processing; it's much easier to read and write too. As its name implies, it's closely related to the ubiquitous JavaScript programming language—ubiquitous in the mobile apps space. XML, on the other hand, is fairly difficult for JavaScript to parse; its processing involves a lot of infrastructure overhead.

Today we are in a world that IBM calls the “API economy,” which serves as the backbone to the as-a-service model that dominates today's cloud (we talk about this in Chapter 3). JSON is a natural fit for these services—in fact, if you programmatically interact with Facebook, Flickr, or Twitter services, then you're familiar with how these vendors leverage JSON-based APIs for sending data to and receiving data from end users. By the time the RDBMS world got sophisticated with XML, developers had moved to JSON because it is so naturally integrated with almost any programming language used today (not just JavaScript). JSON has a much simpler markup and appearance than XML, but it is by no means less sophisticated in terms of functionality. This app shift has led developers to request that their data be stored in JSON document store NoSQL databases.

Most new development tools support JSON but not XML. In fact, between 2009 and 2011, one in every five APIs said “good-bye” to XML, and 20 percent of all new APIs supported *only* JSON (based on the 3,200 web APIs that were listed at Programmable Web in May 2011).

“Less is better: The less we need to agree upon for interoperation, the more easily we interoperate,” said Tim O'Reilly (the personality behind those tech books with animal covers). The logical representation, readability, and flexibility of JSON are compelling reasons why JSON is the de facto data interchange

format for the social-mobile-cloud world. These are the drivers behind JSON's success—they directly map to the developer's value system we commented on earlier—and are precisely the reason why NoSQL document stores such as IBM Cloudant are dominating the NoSQL marketplace today.

Enough Already: What Does a JSON Document Actually Look Like?

The following is an abbreviated example of a JSON document. It shows the profile of a business that is returned from a spatially aware Google-based API search for highly rated health and wellness vendors. This search has also pulled in reviews for the business from popular social sites such as Yelp and Facebook—all stitched together through the API economy.

```
{
  "ownerName"      : "Debbie",
    "ownerlastName" : "Smith",
    "age"           : 35,
    "businessProfile" : "Hot Yoga",
    "businessName"  : "Power Yoga Canada",
    "address"       :
      {
        "streetAddress" : "22 Stevenson Road South",
        "city"          : "Oshawa",
        "province"      : "ONT",
        "postalCode"    : "L1J5L9"
      },
    "socialScores"  :
      [
        { "yelpScore"      : "*****" },
        { "facebookReview" : "5/5" }
      ]
}
```

As you can see, a JSON document is pretty simple. And if you want to add another set of attributes to this generated document, you can do so in seconds. For example, you might want to add an `Instructors` array with corresponding bios and certification information. A developer could do this with a few simple keystrokes, thereby avoiding the begging and bribing rituals around DBA-curated schema change requests of the past.

But I Still Use XML!

We didn't say that XML was dead; it has just assumed a different place in IT architectures. Standard data interchanges for finance and healthcare (among others) are still heavily vested in XML. You will find XML being used for many different things—from documentation to database manipulation and more. We are not dismissive of XML's value in these use cases, but rather we are suggesting why JSON has risen to the top of the food chain. It is simple: JSON is dead easy for programmers to work with. JSON's integration into modern-day programming languages means that you don't, for example, have to concern yourself with document object model (DOM) or simple API for XML (SAX) parsers.

XML deserves a document store too! IBM first debuted a document database almost a decade ago in the DB2 9 release with its pureXML technology. Unlike most RDBMS vendors that have the facilities to store XML documents, the “pure” part of the pureXML name was purposeful. It is a true XML document store with its own engine and processing framework for working with XML documents. This engine can stand alone as an XML database or seamlessly interoperate with DB2's relational side. pureXML is widely used today, supports XQuery and schema validations, and is freely available in all editions of DB2. You can also, of course, store XML in Hadoop or NoSQL databases.

As we said, you don't hear as much about XML these days, but it's still widely used. If Jonathan Goldsmith (the Dos Equis beer spokesperson, otherwise known as the “World's Most Interesting Man”) were to comment here, he'd say, “I don't always use XML, but when I do, I use DB2 pureXML. Keep interchanging data, my friends.”

IBM has a number of solutions in the JSON persistence area (some of which are part of the traditional RDBMS landscape technology, such as Informix and DB2). However, the most proven, scalable, and turnkey NoSQL document store that IBM provides is a fully managed service called Cloudant (see Chapter 10). You can leverage Cloudant services off premise through a hosted and fully managed database as a service model (DBaaS)—a derivative of software as a service (we cover details about as-a-service models and the cloud in Chapter 3). In addition, IBM recently announced a new on-premise Cloudant offering too!

Column Family, Columnar Store, or BigTable Derivatives: What Do We Call You?

Another type of NoSQL database genre is referred to as *columnar*. Although the storage mechanics indeed include column orientation, the term *columnar*, as used elsewhere, means quite a bit more. With that said, the market has labeled this NoSQL genre as *columnar*, so we'll stick with that.

Columnar databases can be found in both RDBMS and NoSQL worlds (for example, we talk about the dashDB analytics service—which is a columnar database—throughout this book). In the NoSQL world, this database genre accounts for about 10 percent of the revenue share of the NoSQL database landscape.

You'll find that columnar NoSQL databases are quite popular for apps that require high-volume, low-latency writes; indeed, there are petabyte-scale use cases associated with these databases. They are specifically designed to store millions of columns with billions of rows. In fact, HBase documentation recommends RDBMS technology unless you are into the hundreds of millions of rows and up territory. Various use cases around the Internet of Things (IoT)—sensor data, user activity logs, and typical telecommunications call detail record (CDR) apps—come to mind as usage domains for this type of technology.

Columnar databases are noted for their *column families*. As the name suggests, columns belong to a column family. Similar to a K/V store, data is retrieved though a key, which in this case is a column lookup; furthermore, columns that are frequently accessed together are likely to belong to the same column family. For example, if you want a client's first name, you would get that by accessing the `FirstName` column. However, it's reasonable to assume that you might also want to access that person's last name and billing address as a way of describing the client—these fields would therefore likely belong to the same column family.

If you went looking for the birthplace of the NoSQL columnar movement, you'd find yourself knocking at Google's doorstep, reading their research publication entitled *BigTable: A Distributed Storage System for Structured Data* (2006). Curiously enough, Google didn't call BigTable a database in this paper; it called it "a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers." Like other NoSQL technologies, BigTable is

schema-less—you can define columns within rows and on the fly. Perhaps puzzling to RDBMS practitioners is the fact that one row can have a different number of columns than another row that belongs to the same database.

HBase and Cassandra are probably the most popular names in the NoSQL columnar space—both descend from BigTable. There is a lot of fierce competition between them in the marketplace, with zealots on each side claiming victory. Both technologies have a lot in common. They both log but don't support “real” transactions (more about this topic later), they both have high availability and disaster recovery services that are implemented through replication, both are Apache open source projects, and more. We examined the strengths and weaknesses of both technologies, but we tend to favor HBase (and not just because IBM ships it as part of BigInsights for Hadoop). HBase is a more general-purpose tool, and it's also an established part of the Hadoop ecosystem. For example, when you install HBase through BigInsights for Hadoop, you don't need to iteratively download, install, configure, and test other open source components. Instead, simply launch the installer, and it performs the necessary work on single-node and cluster environments. After installation, the tool automatically performs a “health check” to validate the installation and report on the results. Monitoring of your environment is centralized and ongoing administration is simplified because it's integrated into the Hadoop management tooling that is provided by IBM. Ultimately, HBase just feels like a much richer and more integrated experience, especially for users of Hadoop. Cassandra can also run on Hadoop, but we are told it runs best in standalone mode, where it's really good at high-velocity write operations.

Don't Underestimate the Underdog: The Graph Store

Graph stores have been different from the start. For example, although most NoSQL solutions are all about scaling out, graph databases are all about interactions and relationships: who knows who, what is related to what, and so on. The key point is that in a graph database, the relationships among data points are mirrored in the modeling and structure of that data within the database itself. In essence, you can trace the relationships between data based on how it is persisted. Typically, graph stores are found servicing apps whose data is highly interconnected and in situations where tracing data point relationships

is key, such as a web site recommendation engine for purchasing that is based on social circles and influencers. Graph store databases are like those tiny fish in a big aquarium that look really cool—they aren't plentiful, but we are going to keep a watchful eye because we expect great things from them.

As you can imagine, in the IoT world where we live, graph database technologies are poised to become a big deal. We think that Social Network Analysis (SNA) is a front-runner for graph database apps—it maps and measures relationship flows between entities that belong to groups, networks, operational maps, and so on. If you're an online dater, there's no question that you've been "touched" by a graph database. The social phenomenon Snap AYI (Are You Interested) has socially graphed hundreds of millions of profiles for its matchmaking recommendations engine to statistically match people that are more likely to be compatible. (Imagine what Yenta could have done with this technology—think *Fiddler on the Roof*...wait for it...got it? Good. If not, Google it.)

Although online dating might not be your cup of tea, we are almost sure that our readers participate in professional social forums such as LinkedIn or Glassdoor; both use graph store technologies. Glassdoor's social graph has nearly a billion connections, and it openly talks about the accuracy of its recommendations engine (which is powered by graph database technology). Twitter? It released its own graph store, FlockDB, to the open source community.

Imagine the usefulness of graph databases for solving traffic gridlock problems by using location data from smart phones to orchestrate the speeds and paths that drivers should take to reach their destinations; as creepy as that sounds, it's still a graph database use case. The classic traveling salesperson problem, network optimization, and so on—these kinds of problems are aptly suited for graph databases. For a great example of how this type of analysis can have a true impact on a business, consider a company like UPS—the "What can brown do for you?" delivery folks. If every UPS driver drove one extra mile a day, it would mean an average of 30,000 extra miles for UPS—at a cost of \$10,000,000 per year! Graph database technology could identify the relationships among drivers, fuel, and costs sooner. Ten million dollars in annual savings is certainly a strong incentive for companies that are on the fence about what NoSQL technology can do for their business.

Finally, IBM Watson (briefly discussed in Chapter 1) uses graph stores to draw conclusions from the corpus of data that it uses to identify probable

answers with confidence. There is a lot of math going on behind the scenes in a graph database, but hey, that's what computers are for.

Graph processing is heating up and is the fastest-growing database in the NoSQL space. Some RDBMS vendors are incorporating graph technology as part of their database solution. For example, both IBM DB2 and Teradata have at one time or another announced such capabilities to the marketplace. That said, we believe graph store technologies belong squarely in the NoSQL realm and out of the RDBMS world, for reasons that are beyond the scope of this book.

There are a number of graph databases available to the NoSQL market today. For example, Apache Giraph is an iterative graph processing system that is built for high scalability and that runs on Hadoop. Other examples include Titan and Neo4J. IBM is working closely with graph stores in Watson, as mentioned previously, and in social apps that are deployed across the company. We wanted to formally state IBM's direction in the graph store space, but as it stands today our lawyers see fit to give us that loving mother stare of "We know better" on this topic (at least we thought it was loving), so we'll leave it at this: Expect potentially big "titan-sized" things around the time that this book is in your hands (or sometime thereafter).

How Graph Databases Work

Unlike other databases, which store data in columns, rows, or key/value pairs, graph stores identify data through nodes and edges, each of which has its own defining properties.

It's really easy to evolve a "schema" with these databases; just add a new edge or node (that's the agility part of NoSQL). The performance benefit of graph databases stems from the fact that other databases have to perform numerous joins to bring together the data connection (a hint at why we think graph capability doesn't belong in the RDBMS world), whereas a graph database traverses from object to object. Remember, graph stores don't just store data; they store the relationships between things of interest, be it people, products, destinations, and so on.

There are frameworks around which graph database metadata and languages are programmed; Resource Description Framework (RDF) is one of them. Another one is called the Web Ontology Language (OWL—a curious acronym considering the name); *ontology* is a fancy word for describing

conversations that precisely describe things and their relationships. OWL is actually built on top of RDF and was designed to process information on the web. Its inspiration was the W3C work on the semantic web. (The inclusion of semantic data in web pages so that they could become a “web” of data, be linked together, and thus be searchable by those paths was the graph database “aha!” moment.) The ability of computers to use such metadata to understand information about web resources is the idea behind the semantic web, and RDF emerged as a W3C standard for describing web resources.

Graph databases describe an entity (called a *node* in the graph world) with properties, along with that entity’s connections (referred to as *edges*), through an ontology of linked fields. These types of data entities, expressed in the form of subject-predicate-object, are called *triples*, and they are stored in specialized triple-store databases: graph stores.

Consider the relationship statement “IBM is a company.” In a triple, “IBM” is the subject, “is a” represents the predicate, and “company” represents the object. When you combine this triple with other triples that deal with the same subject, you have a graph in which the edges define the relationships among the items that you are describing.

Within IBM, graph databases are used to better connect groups. Consider the classic organization chart; you’ve seen them a hundred times. If you’ve ever heard the phrase “Love (or something like that) flows down from the top,” this is where it comes from. Ironically, this is *not* how organizations tend to operate in interconnected environments. They don’t take direction according to a rigid structure; this is often referred to as *matrix management*. For example, in many countries, company employees must report to someone in their local country, and as a result, you’ll often find a reporting structure in which an employee reports to an executive, even if they don’t work for that executive or in the executive’s organization. In reality, people are linked to many other persons in multiple ways. Matrix management makes things more interconnected and more complex. For this reason, we decided to use it as our sample problem domain for illustrating how graph databases work.

IBM is a company of more than 400,000 people, and as you can imagine, when it comes to enablement activities across the enterprise, getting the Information Management brand’s story to the rest of the company involves social complexities. A graph database can be exceptionally helpful in this

area. SmallBlue is an internal IBM app that enables you to find colleagues—both from your existing network of contacts and also from the wider organization—who are likely to be knowledgeable in subject areas that interest you. This app can suggest professional network paths to help you reach out to colleagues more accurately (from an interest perspective) and more quickly. Figure 2-1 uses our group’s real app to show you how a graph store works.

A graph always has an ordered pair (just like the K/V stores we talked about earlier): a vertex (node) and edge. A graph is defined by its vertices and edges. Nodes can be about anything that interests you, and that’s why we often refer to them as *entities*. As you can see in Figure 2-1, a graph can have many nodes and edges that are connected in multiple ways. This figure shows a general graph from which you can learn a lot of things. For example, who might be the best gatekeeper for a certain class of information in the network? In other words, who is the key group connector? That individual should be the one with the shortest path to the highest *degree* (number of linkages) to other persons.

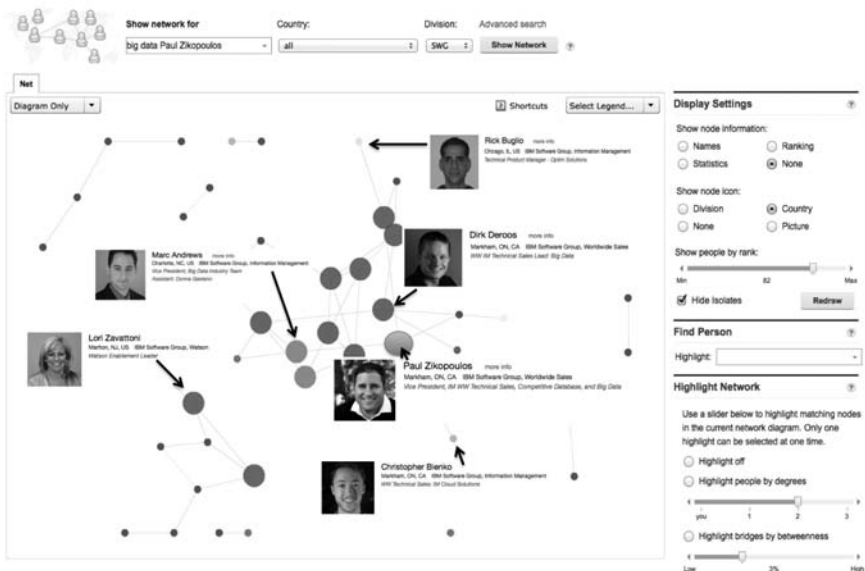


Figure 2-1 Using a graph database to discover and understand relationships in a large organization

A basic graph has details about every node and vertex (vertices are bidirectional). For example, the fact that Dirk knows Paul doesn't necessarily mean that Paul knows Dirk until a vertex showing that Paul knows Dirk is added. In a graph store, it's important to understand that these connections are weighted; not all connections are equal. Weightings can help you to understand the relationships and how they affect a network. Weightings can also use color or line thickness to articulate attributes. For example, in Figure 2-1, Dirk has a larger circle connecting him to Paul than Chris does. Although they both work for Paul, Dirk and Paul have worked together for many years, published books together, and previously worked side-by-side in development—all this strengthens their relationship. Marc Andrews is a vice president in a different segment and therefore has a different relationship with Paul; although that relationship doesn't have similar attributes to what Dirk and Paul have, Marc and Paul work closely together and have equal hierarchical standing at IBM. Rick Buglio is in a different relationship altogether; he doesn't report to Paul, neither is he part of the department, but something connects them (for example, this book), and so on.

We created this visualization by searching our graph store for Big Data attributes; we were specifically looking for the strongest spoke of the book's authors into a new group, or an individual with the greatest number of spokes, into Big Data domains that we aren't already a part of—all in an effort to spread our gospel of Big Data. Any time you try to figure out multiple entry points into a community—be it for internal enablement, charity fundraising, or to get a date from a pool of people—it's a graph problem. This domain of analysis involves concepts such as closeness centrality, "betweenness" centrality, and degree centrality. As it turns out, Lori Zavattoni works in IBM's Watson division; we identified her by using a social graph within IBM to find an enablement leader who seems to be a major influencer and shares spokes with other groups. We kept this graph simple, so keep in mind that a "true" graph can contain thousands (and more likely millions) of connections. We also left the connection graph at a coarse degree of separation (2, as shown on the right side of this figure); you can throttle this (and other) attributes up or down. We reached out to Lori to see what we can do around Big Data and Watson and the result was a section on the topic in Chapter 1.

There are graph subtypes such as a *tree* and a *linked list*. A tree graph enables any node to talk to any other node, but it cannot loop back (cycle) to

the starting node except by retracing its exact steps through the path it took to get there (this kind of breadcrumb trail is popular for web page navigation analysis). If a connection were made between Paul and Lori, the only way to Lori's connections is through Paul; therefore, Dirk must go through Paul to get to Lori. Of course, Dirk could just send Lori a quick email—but if this were a voter identification list, a political party would use Paul to get to Lori and gain access to her connections only through that first link with Paul. In fact, this *exact* technique was used by the reelection team for Barack Obama's second presidential campaign.

The best way to describe a linked list (which is more restrictive than a tree graph) is to think of a graph that resembles a connect-the-dots picture. Each node sequentially connects to another node, though in this case you might not end up with a pretty picture. Examining a company's supply chain or procurement channel might produce a linked list graph. For example, the only way to get a doctor's appointment with a specialist in Canada is to get a referral by your general practitioner; so if you've got a skin issue, your family doctor is part of the linked list to the dermatologist—there is only one kind of graph in a linked list.

As you've seen, graph databases are great for highly connected data because they are powerful, come with a flexible data model, and provide a mechanism to query connected data in a highly performing manner. They are, however, not perfect. Aside from the fact that they are still maturing, the schema that is produced by a graph database can get complex in a hurry, and transactions aren't always ACID-compliant (more on this in a bit). What's more, because they can grow rapidly, one of the traditional challenges with graphs has been (and still is) the processing time that is required to work through all of the data permutations that even relatively small graphs can produce (we think that this is a short-term challenge). That said, we feel that the greatest short-term weakness of graph databases is that they require a new way of thinking—a paradigm where everything that you look at is a graph. The open source movement is taking the lead and you can see quarterly gains in graph database adoption as a reflection of those efforts. We think that this is a great example of why polyglot persistence is key to your strategy—an architecture that leverages multiple capabilities (in a variety of different forms) can provide the most value as businesses make the jump to NoSQL.

From ACID to CAP

To better understand some of the characteristics of NoSQL databases, it's best to start with traditional RDBMS technology. *Atomicity*, *consistency*, *isolation*, and *durability* (commonly referred to as the ACID properties of a relational database) are cornerstones of the traditional SQL world. These properties help to guarantee that database transactions are processed reliably. There are entire books with chapters dedicated to each ACID property, but we limit this discussion to those details that help to clarify NoSQL characteristics. It's important to understand trade-offs between how relational databases work and how NoSQL databases work; these aren't "one is better than the other" kinds of discussions—it's more like a "one is better suited to solving a particular problem" discussion.

Atomicity means that a transaction should be committed or undone as a whole. In the event of a failure, all operations that were "in flight" should be undone, and any changes should be rolled back to the previous state. Have you ever wanted to buy something online and clicked Buy, only to find your Internet connection dropped and the purchase didn't go through? Atomicity means that this transaction never happened and that your credit card won't be charged for a purchase that didn't happen because the transaction could not be completed successfully without interruption.

Consistency means that a transaction that is applied to a system in a consistent state should leave the system in a consistent state, and data that is modified by the transaction should be validated according to any business rules that are defined in the database (we alluded to this in Table 2-1). Imagine that two business partners, Adam and Bob, share the same checking account. Consistency means that at any time and from any location or device that is connected to the bank's network, Adam and Bob should be able to see how much money is in their account, and both should see the same balance in that account. Triggers offer another example of consistency. Suppose that a defined database trigger fires whenever a client delete operation occurs. For this transaction to complete successfully, the client record must be inserted into a call-back table. The delete isn't considered successful if the trigger didn't fire.

Isolation means that when concurrent transactions run on a database, they operate independently of one another and can't interfere with each other. If both transactions want to access the same data value, what happens next

depends on the concurrency model that is in effect; one transaction waits until the other one completes, or it simply reads an older copy of the data, among other options (each option has potential consequences, such as a phantom read or nonrepeatable read, among others). Isolation is used to deliver performance and consistency in an RDBMS system.

The fourth ACID property of a relational database is *durability*, which means that completed transactions should remain permanent and be protected from system failures. All major RDBMS vendors offer some sort of mechanism that logs transactions on disk (ironically, all vendors in this space pretty much follow the IBM-invented ARIES logging protocol). For example, if you receive verification that you have successfully paid a bill, that bill is paid, and if the database has to re-create the transaction after a failure, it has all the information on disk to successfully replay the transaction.

In summary, ACID databases mostly prize consistency of the data over the availability of the data. (There are some exceptions that require more explanation. For example, both Oracle and DB2 support a “readers don’t block writers and writers don’t block readers” operational model. But this is outside the scope of this book.)

CAP Theorem and a Meatloaf Song: “Two Out of Three Ain’t Bad”

The acronym CAP stands for *consistency*, *availability*, and *partition tolerance*. *Consistency* here is similar to the ACID property. Some might think of it as single-copy consistency because they define it at the single partition level in a database cluster. We believe it to mean that all nodes in the same system see the same data at the same time. Think back to Adam and Bill having the same view of their checking account.

Availability is a guarantee that every request to the database receives a response, either success or failure. That’s a very different view of availability than what many think of in the RDBMS world. For example, if you can ping your RDBMS server, does that mean it’s available? Ping always returns a response—the packet was either successfully delivered or not. For others, availability means that a transaction can be processed.

Partition tolerance means that a clustered system continues to operate even in the presence of lost messages or failures that occur in parts of the system; in other words, node failures shouldn’t cause the whole database to collapse.

Being partition tolerant also means being able to scale into distributed systems; you can see how RDBMSs traditionally place less emphasis on partition tolerance, given how difficult many of these systems are to scale horizontally. Partition tolerance helps with the elastic scalability of NoSQL databases (those that scale out). Although the SQL world is weaker in the partition tolerance transactional area (with the exception of DB2 pureScale and DB2 for System z technologies that are based on IBM's Coupling Facility, which was designed for this purpose), the NoSQL world prioritizes partition tolerance.

The CAP theorem (first proposed by Eric Brewer in his "Principles of Distributed Computing" keynote address and later refined by many) states that you can guarantee only two out of the three CAP properties simultaneously—and that's not bad according to the hit song writer, Meatloaf. In other words, if your database has partitions (particularly if it's distributed), do you prioritize consistency or the availability of your data? (We should note that trade-offs between consistency and availability exist on a continuum and are not an "all-or-nothing" proposition.)

Clearly, reduced consistency and weakened transactional "ACIDity" don't sound like a good idea for some banking use cases that we can think of—and yet there are many apps where high availability trumps consistency if your goal is to be more scalable and robust. As you might expect, in a social-mobile-cloud world, systems of engagement and systems of people apps could very well make this kind of trade-off. For example, if you're checking how many Repins you have on Pinterest, does it matter if that number is accurate at a specific point in time? Or can you settle for being eventually consistent, if it means that your users get values (whether or not they are the most up to date) on demand?

NoSQL practitioners are, more often than not, willing to trade consistency for availability on a tunable spectrum. In fact, one NoSQL database with a "mongo" footprint is well known for data writes that can just disappear into thin air. Ultimately, the NoSQL solution that you select should be based on business requirements and the type of apps that you're going to support.

There is a subset of the CAP theorem that refers to a persistence layer being basic availability, soft state, and eventual consistency—BASE. We're not going to get into these finer-level details in this chapter because our goal was to give you a basic overview, which we feel we've done.

Let Me Get This Straight: There Is SQL, NoSQL, and Now NewSQL?

So far, we've mainly discussed NoSQL databases—the focus of this chapter. But NewSQL is certainly an emerging topic—the next hot topic if you will. Because we only wanted to focus on NoSQL in this chapter, we include only a short primer on NewSQL technology, but keep your eyes on this space. NewSQL is the NoSQL of the SQL age (insert cheeky emoticon here).

We know where the name NewSQL comes from: a research paper penned by 451 Group's Matt Aslett. He refers to a new class of operational databases that can't use NoSQL solutions because the apps that they support have stringent transactional and consistency requirements. Indeed, the flexibility and scalability of NoSQL databases comes with a trade-off between consistency and availability. In some spaces, this just isn't acceptable. In his paper, Matt Aslett states that NewSQL databases differ greatly with respect to their internals (architecture, implementation), but they share two distinguishing features: They all support the relational data model and use SQL as their primary interface.

NewSQL's intent is to deliver on all three aspects of the CAP theorem; as you can imagine, because there are a number of people who think a database can deliver only one or two—but not all three—of the CAP theorem properties, this area is the subject of hot debate. Some NewSQL databases use HDFS as the underlying file system, others support a proprietary file system, and others let you choose. Much like NoSQL servers, you're likely to find NewSQL implemented on Linux in a scale-out cluster with locally attached disks; you will also find network-attached storage. Off-premise cloud deployment is common for this technology because the design favors high-speed interconnections within and between clusters. The calling card of NewSQL is the marriage between agility and scalability while still emphasizing the importance of ACID-compliant transactions.

Although the name implies that these databases use a new kind of SQL, NewSQL databases use the SQL that we have known and loved for half a century. The focus for this class of database system is the automatic resharding and balancing of data across geographically dispersed machines. You can see the allure of NewSQL. One of the drivers for NoSQL databases is seamless scalability, which can be hampered in the SQL world by the inherent write latency that is a side effect of high-consistency data requirements.

Wrapping It Up

In this chapter, we covered SQL, NoSQL, and even touched on NewSQL database technologies as options for today's data centers. We also discussed the concept of polyglot persistence. Supporting a polyglot Big Data and Analytics platform means selecting a variety of data-layer technologies that address the widest array of use cases that your apps might encounter. We also talked about how in the API economy, these layers will be fully abstracted—we talk more about this in the next chapter. Elsewhere in this book, you'll read about NoSQL technologies such as the Cloudant document store, HBase, the Hadoop ecosystem, in-memory columnar database technology, and traditional RDBMS technologies across all form factors: from off-premise hosted or fully managed services, to "roll-your-own" on-premise solutions, and appliances. Together, these polyglot tools represent the next generation of data-layer architectures and solutions for tomorrow's Big Data and Analytics challenges.

3

Composing Cloud Applications: Why We Love the Bluemix and the IBM Cloud

A journey toward the adoption of the “cloud” to meet business targets is in many ways a transition from traditional systems of record to systems of engagement and systems of people. Consider the emergence of social media—it’s hard to pinpoint exactly when *selfies* and *tweets* became part of our vocabulary. Just like the Web evolved from being a rigid set of pages to a space that’s much more organic, interactive, and integrated, we are bearing witness to a transformation in computing from hardened silos to flexible “as a service” models. Why? Three words: social-mobile-cloud.

Social-mobile-cloud has dramatically accelerated social change in totally unanticipated ways; in fact, it has completely changed and altered the flow of information on the entire planet. Information used to flow from a few centralized sources out to the masses. Major media outlets like The BBC, CNN, NY Times, and Der Spiegel were dominant voices in society, and were able to control conversations about current events. Social-mobile-cloud has obliterated the dominance of mass media voices and changed the flow of information to a many-to-many model. In short, the spread of mobile technology and social networks, as well as unprecedented access to data is changing how individuals, groups, and organizations engage. We call this mode of engagement *The Why*.

Of course, all this new data has become the new basis for competitive advantage. We call this data *The What*. All that's left is *The How*. And this is where the cloud comes in. This is where you deploy infrastructure, software, and even services that deliver analytics and insights in an agile way. In fact, if you're a startup today, would you go to a venture investment firm with a business plan that includes the purchase of a bunch of hardware and an army of DBAs to get started? Who really does that anymore? You'd get shown the door, if you even make it that far. Instead, what you do is go to a cloud company like SoftLayer (the IBM Cloud), swipe your credit card to get the capacity or services, and get to work. It allows you to get started fast and cheap (as far as your credit card is concerned). You don't spend time budgeting, forecasting, planning, or going through approval processes: You focus on your business.

The transformational effects of The How (cloud), The Why (engagement), and The What (data) can be seen across all business verticals and their associated IT landscapes. Consider your run-of-the-mill application (app) developer and what life was like before the cloud era and "as a service" models became a reality. Developers spent as much time navigating roadblocks as they did writing code. The list of IT barriers was endless: contending with delays for weeks or months caused by ever-changing back-end persistence (database) requirements, siloed processes, proprietary platform architectures, resource requisitions, database schema change synchronization cycles, cost models that were heavily influenced by the number of staff that had to be kept in-house to manage the solution, processes longer than this sentence, and more. Looking back, it is a wonder that any code got written at all!

Development is a great example. Today we see a shift toward agile processes. We call this *continual engineering*. In this development operations (DevOps) model, development cycles get measured in days; environment stand up times are on the order of minutes (at most hours); the data persistence layer is likely to be a hosted (at least partially) and even a fully managed service; the platform architecture is loosely coupled and based on an API economy and open standards; and the cost model is variable and expensed, as opposed to fixed and capital cost depreciated over time.

As is true with so many aspects of IT and the world of Big Data, we anticipate this wave of change has yet to reach its apex. Indeed, there was too much hype around cloud technologies, and practitioners had to get beyond that just like Big Data practitioners are now facing the hype barrier. But today

there is no question about it, the cloud is delivering real value and agility. It has incredible momentum, it is here to stay, and the conversation is now well beyond the hype. The cloud was once used for one-off projects or test workloads, but now it's a development hub, a place for transactions and analytics, a services procurement platform where things get done. In fact, it's whatever you decide to make it. That's the beauty of the cloud: All you need is an idea.

At Your Service: Explaining Cloud Provisioning Models

We broadly categorize the cloud's provisioning models and concepts into the three domains that are shown in Figure 3-1, which serve as the focus for much of this chapter: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS).

The provisioning models in this figure all stand in contrast to traditional IT, where you'd have to manage the entire technology stack. We call the traditional approach an *on-premise* solution (in your place of work), while the "as a service" models are *off-premise* solutions (they reside "in the cloud").

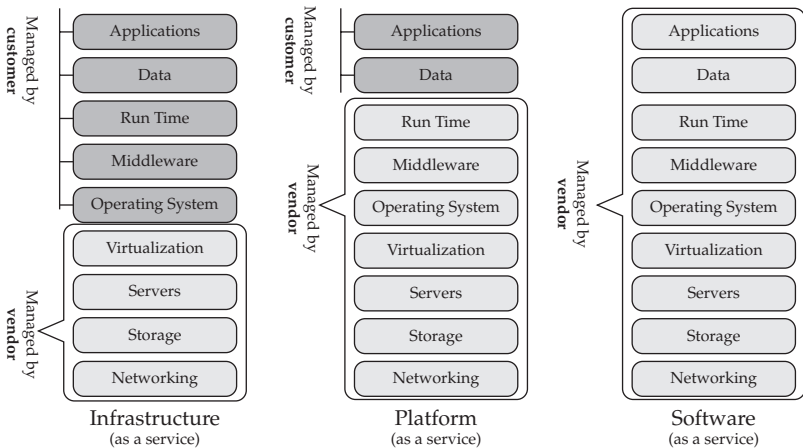


Figure 3-1 A high-level comparison of vendor and customer responsibilities at each as-a-service level

To help you better understand the practical differences between an on-premise solution and the "as a service" models listed in Figure 3-1, we've

come up with the following analogy (we'd like to thank IBMer Albert Barron for this story's inspiration).

Imagine you have a son who's just started his sophomore year at university, and is living in your home. He just mentioned how he found a tequila lime chicken recipe he got from the Chef Watson's bon appetit site (<http://bonappetit.com/tag/chef-watson>) and wants to use it to impress a date. You laugh and remind him that you taught him how to boil an egg over the summer. After some thought, your son concludes that making a gourmet dish is too much work (cooking, cleaning, preparation, and so on) and his core competencies are nowhere to be found in the kitchen; therefore, he decides to instead take his date to a restaurant. This is similar to the SaaS model, because the restaurant manages everything to do with the meal: the building and kitchen appliances, the electricity, the cooking skills, and the food itself. Your son only had to make sure he cleaned himself up.

A year passes and your son is now in his junior year, but is still living in your home (we can hear the painful groan of some of our readers who know this situation too well). He's got another date and wants to arrange for a more intimate setting: your house. He buys all the ingredients and digs out the recipe, but pleads for help with making the meal. Being a loving parent, you also want your son to impress his date (this may speed up his moving out), and you prepare the meal for him. This is similar to the PaaS model, because you (the vendor) managed the cooking "platform": You provided the house and kitchen appliances, the electricity, and the cooking skills. You even bought candles! However, your son brought the food and the recipe.

In his senior year, you're pleased that your son has moved into his own apartment and has been taking on more responsibility. Just before he moved, he tells you he's found "the one" and wants to really impress this person with his favorite tequila lime chicken recipe. But this time he wants to class it up, so he's hosting it and he's making it. He's had to go and find a place to live (floor space), and his rent includes appliances (a stove and fridge) and utilities. He takes a trip to IKEA and gets some tasteful place settings and a kitchen table and chairs (your 20-year-old couch wouldn't be suitable for such an important guest to sit and eat). Once he's settled into his apartment, he goes to the grocery store to get the ingredients, and then comes home and cooks the meal—he essentially owns the process from start to finish. Your son owns everything in the food preparation process except for the core infrastructure, which he's renting: This is IaaS. As an aside, since he has sole

access to all of the resources in his apartment, this IaaS model would be called *bare-metal* infrastructure in cloud-speak.

In this IaaS scenario, imagine if your son shared an apartment with roommates. Everything from the contents in the fridge, to the stove, to place settings, to the tables and chairs are potentially shared; we call this a *multitenant* infrastructure in cloud-speak. Those roommates? They can literally be what we call “noisy neighbors” in the cloud space. In so many ways, they can get in the way of an expected planned experience. For example, perhaps they are cooking their own food when he needs access to the stove, they planned their own important date on the very same evening, they might have used the dishes and left them dirty in the sink, or worse yet...drank all the tequila! In a multitenanted cloud environment, you share resources; what’s different in our analogy is the fact that you don’t know who you’re sharing the resources with. But the key concept here is that when you share resources, things may not operate in a manner in which you expected or planned.

A few years later, your son has a stable job (finally!) and has just bought his own house. He’s newly married, and for old-time’s sake, wants to make his spouse the tequila lime chicken recipe that started their romance. This time, he truly owns the entire process. He owns the infrastructure, he’s cooking the meal, and he bought the ingredients. In every sense, this is similar to an on-premise deployment.

Notice what’s common: In every one of these scenarios, at the end of the evening, your son eats tequila lime chicken. In the on-premise variation of the story, he does all of the work—in some scenarios he has to share the resource and in others he doesn’t. There are scenarios in which he has other people take varying amounts of responsibility for the meal. Today, even though your son is now independent and has the luxury of his own infrastructure, he still enjoys occasionally eating at restaurants; it’s definitely easier for him to go out for something like sushi or perhaps he is just too tired to cook after a long week. This is the beauty of the “as a service” models—you can consume services whenever the need arises to offload the management of infrastructure, the platform, or the software itself; in cloud-speak you’ll hear this referred to as *bursting*.

As your understanding of these service models deepens, you will come to realize that the distinctions among them can begin to blur. Ad hoc development strategies of the past have given way to predefined development

“stacks,” and going forward, we will see these stacks refined into development “patterns” that are available as a service. Today, the expert integrated IBM PureSystems family (PureFlex, PureApplication, and PureData systems) combines the flexibility of a general-purpose system, the elasticity of an on-premise cloud, and the simplicity of an appliance. These systems are integrated by design and are finely tuned by decades of experience to deliver a simplified IT experience. You can think of these as foolproof recipes. For example, a data warehouse with a BLU Acceleration pattern (we talk about this technology in Chapter 8) is available for a PureFlex system, which equates to tracing paper for a beautiful picture (in this case, a high-performance turn-key analytics service). Patterned solutions are not exclusive to on premise: Pattern design thinking is driving service provisioning for cloud-based environments. For example, IBM makes the BLU Acceleration technology available in PaaS and SaaS environments through a hosted or managed analytics warehousing service called dashDB.

From small businesses to enterprises, organizations that want to lead their segments will quickly need to embrace the transition from low-agility software development strategies to integrated, end-to-end DevOps. The shift toward cloud models ushers in a new paradigm of “consumable IT”; the question is, what is your organization going to do? Are you going to have a conversation about how to take part in this revolutionary reinvention of IT? Or will you risk the chance that competitors taking the leap first will find greater market share as they embrace the cloud?

Setting a Foundation for the Cloud: Infrastructure as a Service

The key business initiative driving IaaS adoption is the need to flatten IT expenditures by leveraging cloud-based data centers in place or in support of on-premise IT investments. Those who relish the terminology of finance might express this strategy as a transfer of capital expenditure (CAPEX) to operational expenditure (OPEX). From an accounting perspective, CAPEX cannot be fully written off in the period during which the expense was incurred. It has to be amortized. OPEX costs, on the other hand, are deductible in the accounting period during which they were incurred. This is an attractive proposition, and there is certainly a financial edge to be gained by moving to the cloud. But this

isn't the true reason for the cloud's momentum. Moving to the cloud gives you increased agility and the opportunity for capacity planning that's similar to the concept of using electricity: It's "metered" usage that you pay for as you use it. In other words, IaaS enables you to consider compute resources as if they are a utility like electricity. Similar to utilities, it's likely the case that your cloud service has tiered pricing. For example, is the provisioned compute capacity multi-tenant or bare metal? Are you paying for fast CPUs or extra memory? Never lose sight of what we think is the number-one reason for the cloud and IaaS: it's all about *how fast can you provision the computing capacity*.

When organizations can quickly provision an infrastructure layer (for example, the server component for IaaS shown in Figure 3-1), the required time to market is dramatically reduced. Think of it this way: If you could snap your fingers and have immediate access to a four-core, 32GB server that you could use for five hours, at less than the price of a combo meal at your favorite fast-food restaurant, how cool would that be? Even cooler, imagine you could then snap your fingers and have the server costs go away—like eating the combo meal and not having to deal with the indigestion or calories after you savor the flavor. Think about how agile and productive you would be. (We do want to note that if you are running a cloud service 24x7, 365 days a year, it may not yield the cost savings you think; but agility will always reign supreme compared to an on-premise solution.)

A more efficient development and delivery cycle leads to less friction and lowers the bar for entry when it comes to innovation and insights: Imagine the opportunities and innovations that your organization can realize when the traditional pain points and risk associated with costly on-premise IT infrastructures are reduced by a cloud-based, scalable, "as a service" solution.

We often talk about three significant use-case patterns emerging for adopters of IaaS data layers, and we think they should be part of your Big Data conversations too. The first involves a discussion on ways for organizations to reduce their large and ongoing IT expenses through the optimization of their existing data centers. When such companies move their infrastructure to a managed cloud, they begin to enjoy consolidation of services, virtualized deployments that are tailored to their storage and networking requirements, and automatic monitoring and migration.

The second pattern involves the ability to accelerate the time to market for new applications and services. Clients can leverage new environments and

topologies that are provisioned on the cloud with ease, often by simply selecting the sizing that is appropriate for their workloads and clicking “Go” to kick off the compute resource spin-up process. It might not be real magic, but a couple of swipes and gestures can be just as impressive.

The third pattern involves an environment that is running a client’s apps and services on top of an IaaS layer. These clients can gain immediate access to higher-level enterprise-grade services, including software for *composing* (developing) apps in the cloud or *consuming* cloud apps such as IBM’s dashDB analytics warehouse or IBM’s DataWorks refinery services, and so on. These are the PaaS and SaaS flavors of the cloud. Each service builds on the other and delivers more value in the stack. It’s kind of like a set of Russian dolls—each level of beauty gets transferred to the next and the attention to detail (the capabilities and services provided from a cloud perspective) increases. In other words, the value of IaaS gives way to even more value, flexibility, and automation in PaaS, which gives way to even more of these attributes in SaaS (depending on what it is you are trying to do).

What are some of the available services for organizations that leverage an IaaS methodology? Figure 3-2 shows IaaS at its most fundamental level. A vendor delivers complete management of the hardware resources that you request for provisioning over the cloud: virtual machine (VM) provisioning, construction and management of VM images, usage metering, management of multitenant user authentication and roles, deployment of patterned solutions, and management of cloud resources.

Earlier in this chapter we referred to cloud services as hosted or managed. With *managed* IaaS providers, you gain access to additional resources such as patching and maintenance, planning for capacity and load, managed responses to data-layer events (including bursts in activity, backup, and disaster recovery), and endpoint compliance assurances for security—all of which are handled for you! In a *hosted* environment, these responsibilities fall on you. It’s the difference between sleeping over at mom’s house (where you

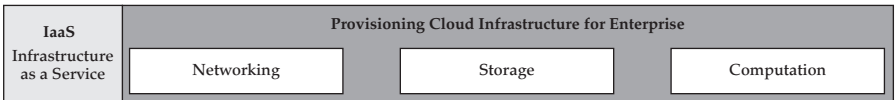


Figure 3-2 Infrastructure as a service (IaaS) provides the foundational level for any service or application that you want to deploy on the cloud.

are expected to still do a lot of work—you are hosted) or sleeping at a resort (managed—it's all done for you in a luxurious manner).

Finally, IaaS providers like IBM are increasingly supplying integration support with on-premise infrastructure to offer hybrid cloud deployments, advanced security (threat and vulnerability assessments), tailored provisioning (for storage and networking requirements), and full orchestration to ensure that services on the cloud work in concert.

IaaS for Tomorrow...Available Today: IBM SoftLayer Powers the IBM Cloud

Regardless of what analysts say, the cloud is not a commodity. And no matter what provisioning model you use, the physical hardware—along with hundreds of miles of networking cable, among other things—lives in some brick-and-mortar facility. One thing we want you to know when it comes to cloud discussions is that no two clouds are built the same way. We believe the IBM Cloud (powered by SoftLayer) offers its clientele the highest-performing cloud infrastructure available. It's a cloud platform that started as a cloud platform and was built from the ground up to be a managed cloud service. Because it's a managed service, you have the assurance that IBM SoftLayer manages and maintains this infrastructure layer for you.

As we mentioned earlier, the IBM Cloud (across the spectrum of IaaS, PaaS, and SaaS provisioning options) is powered by SoftLayer. It can provision private, public, or hybrid cloud services. To date, IBM has invested more than \$1 billion (US) to extend SoftLayer's data centers across the globe. By the time this book goes to print, there will be about 40 geographically dispersed server farms worldwide. Each data center is built, outfitted, and operated with the same consistent level of service, giving you consistent capabilities and availability anywhere in SoftLayer's footprint. From what we can tell, it has more global data centers than any other vendor in its peer group; in fact, in many parts of the world, where regulations for some entities mandate that data stay within its country of origin, SoftLayer is the *only* "big name" choice.

One of the things that sets SoftLayer's data centers apart is the consistently high performance clients experience when provisioning services at any of its locations worldwide. SoftLayer's data centers operate with first-class computing, storage, and networking gear. All of these pieces come

together to enable the industry's most advanced "network within a network." SoftLayer's fast, resilient private network backbone forms the core of its triple-network architecture. Specifically, its data centers are connected by three distinct (yet deeply integrated) public, private, and internal management networks to deliver lower total networking costs, better access, and higher data transfer rates. In our experience, we've seen the IBM Cloud push 2TB of data per second between locations around the world (they weren't close to each other) into a private network with less than 40ms of latency. Unlike other competitors who made their name selling books online, SoftLayer won't lock you into one vendor's technology. For example, the other cloud vendor we just hinted at mandates that the data has to be loaded from their provisioned storage in order to get it into their analytic service. Perhaps you want to load it directly from your own on-premise data repository. Why should you be forced to stage your data on a cloud storage tier (for a fee, of course), only to move it again?

SoftLayer is an all-in-one, automated platform—all the services that you require are delivered within a single management system that's entirely consumable through web services. All the SoftLayer data centers and networks share a single management touch point; one tool ties it all together and gives you control over everything—each bare-metal server, virtual server, storage device, you name it—from a single pane of glass.

Disaster and *downtime* are words that no one wants to hear in any IT setting. Indeed, talk of availability and disaster recovery can stir up feelings of anxiety for companies that are not prepared for the volatile web. SoftLayer's strategy for building a global infrastructure footprint that's managed as a service doesn't just prepare for outages and disasters to happen—it expects them! As a SoftLayer customer, your data and infrastructure are more robust because of this posture.

SoftLayer can provision services in a dedicated or shared multitenant environment (more on that in a bit). We think that SoftLayer's provisioning versatility is unique in the industry when you consider that analyst firm Gartner commented that "by 2017... half of all large enterprises would have a hybrid cloud deployment." Think about it. Long before IBM bought SoftLayer as the backbone for its cloud service, it had decades of experience in on-premise deployments. Next, IBM deployed a number of expert-designed systems that provide on-premise clouds. Today, IBM is leading the industry

when it comes to off-premise deployments on SoftLayer technology. What's more, IBM has embraced the hybrid cloud; for example, you can seamlessly burst work into the IBM Cloud and back to the ground again if required. The tally of vendors who can do this as seamlessly and efficiently as IBM is small, and we think this places IBM at the top of the short list for hybrid cloud architectures. It's important that the conversations you have about Big Data not be limited to the cloud, but rather include a discussion of a *hybrid* ecosystem. (In case it's a point of confusion, a hybrid cloud means a mixture of off-premise and on-premise investments, as we refer in our examples. A hybrid cloud's definition also includes a mix of clouds; for example, private and public. If a private cloud is set up for your company by your company, then we classify it as on premise, because the servers are owned by your company and reside on company property. Their capacity is provisioned across the company as a cloud model, and thus it's a private cloud, but it is still an on-premise investment.)

We see a lot of marketing around hybrid cloud from various vendors, but more often than not, the words ring hollow. These vendors require customers to completely change the way their apps work and need to be deployed in order to "fit" into their cloud model. *True* hybrid architectures, such as what the IBM SoftLayer infrastructure provides, are able to adapt and integrate with the existing on-premise infrastructure. If your IaaS solution requires that you change your apps to fit rigidly constrained models, or force fits you into a cost-metered staging area, we suggest reassessing the merits of that provider. Having a flexible cloud infrastructure that adapts to your business requirements (and not the other way around) is going to matter as you grow your Big Data platform.

Noisy Neighbors Can Be Bad Neighbors: The Multitenant Cloud

Before we cover the other "as a service" models, it's worth taking a moment to examine a fundamental aspect of SoftLayer's deployment options: bare-metal infrastructure for private single-tenant cloud environments.

Bare metal is not an '80s big hair-band satellite radio channel, but a term that describes completely dedicated infrastructure on the cloud. Many of the largest vendors that offer cloud services today do it solely over a shared,

public, multitenant, and often virtualized environment. One obvious concern in this environment is privacy—forced multitenancy (sharing cloud infrastructure) is a major pain point for companies that want to host sensitive data on the cloud.

But there are more issues at stake. Performance on a bare-metal server will always outperform the same provisioned infrastructure if it's virtualized—there is always some performance loss in a virtualized environment. For situations where fastest performance is a must, you will want a bare-metal server, and that's not an option with every cloud provider. Provisioning multitenant resources also means having to contend with the reality of shared resources; performance is never consistent and might not necessarily live up to the full potential that was promised on paper. We attribute this unpredictability to the “noisy neighbors” effect. In a shared cloud service, the noisy neighbor is exactly what you might imagine: Other users and applications sharing the same provisioned infrastructure on a multitenant environment can bog down performance and potentially ruin the experience for everyone else (think back to our example of a young man hosting a date at a shared residence). Perhaps those neighbors are generating a disproportionate amount of network traffic or are heavily taxing the infrastructure to run their apps. The end result of this effect is that all other users of the shared environment suffer degraded performance. In this kind of setting, you can't be assured of consistent performance, which means you can't pass the assurance of predictable behavior down to your customers. In the same manner that inconsistent query response times foreshadow the death of a warehouse, the same applies for services in the cloud.

In a provisioned multitenant cloud environment, you don't know who your neighbors are. Imagine for a moment that you are unknowingly sharing your cloud with some slick new gaming company that happened to hit the jackpot: Their app has “gone viral.” Who knew that an app for uploading pictures of your boss to use in a friendly game of “whack-a-mole” would catch on? Within days, millions join this virtual spin on a carnival classic. All this traffic creates a network-choking phenomenon. Why do you care? If this gaming studio and its customers happen to be sharing the same multitenant environment as you are, then the neighborhood just got a lot noisier, and your inventory control system is going to run a lot like the aforementioned game—up and down.

The IBM Cloud gives clients the option to provision a true bare-metal, dedicated IaaS that removes “noisy neighbors” and privacy concerns from the equation altogether. It can provision and manage an infrastructure layer that is exclusively dedicated to your company’s cloud services or hosted apps. Your services benefit from the guaranteed performance and privacy that come from a bare-metal deployment. IBM is uniquely positioned to seamlessly integrate bare-metal and virtual servers, all on demand, as part of a unified platform. SoftLayer also provides autoscaling, load balancing, different security models (for open and commercial technologies), and a variety of server provisioning options (dedicated and multitenant). Every client has a preference; some might have a business need that fits a hybrid model of multitenancy and dedicated bare metal working side by side; others might be fine with virtualization for all their servers. SoftLayer offers the IaaS flexibility to meet and exceed the requirements of any client—including those who don’t want to live beside noisy neighbors.

Building the Developer’s Sandbox with Platform as a Service

We get a little more technical in this section, so let’s ease into the discussion with a look at what defines platform as a service (PaaS) and how IBM’s Bluemix fits into that framework. Grab your shovel—it’s time to hit the PaaS sandbox.

Bluemix is IBM’s PaaS for composing, managing, and running applications that work in many settings—from web, mobile, Big Data, smart devices, Internet of Things, and more—all supported by an open standards environment based on Cloud Foundry. Bluemix enables you to provision complete development environments faster than it takes for you to reboot your laptop after a blue screen. Choose a language run time, or bring your own, and run with it! The catalog of IBM, third-party, and open source API-fronted services enables you to work without ever needing to step outside of the Bluemix environment. Enterprises and businesses today also want to integrate with their on-premise infrastructure; Bluemix can make that happen with support for environments. Do you want to connect to existing on-premise assets; link those to public and private clouds; and do so with a fully supported development, deployment, monitoring, and logging toolkit (DevOps)? Bluemix can do that too! What’s more, IBM hardens your entire solution with enterprise-grade

security, so there is no need to fret about the volatile world of the cloud. Lastly, the way you pay for using Bluemix is much like how you're billed for electricity—you pay for what you use. And if you are using one of the many freemium services that are available on Bluemix, unlike your electricity, you don't pay anything at all!

If You Have Only a Couple of Minutes: PaaS and IBM Bluemix in a Nutshell

When you go to any cloud provider (for example, IBM SoftLayer, Amazon, or Microsoft Azure), you can provision a hardware layer with a couple of mouse gestures, grab a coffee, and voilà: a server! What IaaS gives you is the shell: the infrastructure layer that supports the software or services you build and run. Storage, firewalls, load balancers, CPU, memory—all of these components are provisioned in IaaS and are at your disposal. But consider for a moment what's missing from this picture. If you recall from Figure 3-1, not even the operating system (OS) shows up until the PaaS pillar. So, when we say that IaaS gives you *all* the parts that you need to start installing software, we are talking about the absolute *minimum*—it's like buying a laptop with no OS.

Quite simply, IBM Bluemix enables clients to rapidly compose, deploy, and manage cloud applications. We like to tell clients that Bluemix allows you to go live in seconds because a developer can choose any language or run time, or even bring their own. In fact, you can use Bluemix to build, manage, and run all manner of apps for web, mobile, and Big Data solutions. It ties together development backbones such as node.js, Java, mobile back-end services, application monitoring, analytics service, database services, and more. Bluemix takes the work out of standing up a dev environment, or forcing developers to deal with virtual machine images or hardware to get stuff done (you don't have to prepare the tequila lime chicken, thinking back to our university student analogy). With a few swipes or keystrokes, you can provision instances of your applications with the necessary development services to support them. This streamlining translates into countless hours saved when compared to traditional app dev preparation such as installation, configuration, troubleshooting, and the seemingly endless amounts of time spent reacting to never-ending changes in requirements; instead, Bluemix lets you relentlessly innovate. Indeed, Bluemix lets developers go from zero to production in one command!

IBM envisions Bluemix as a ecosystem that answers the needs and challenges facing developers, while at the same time empowering and enabling businesses to leverage their resources in the most efficient way possible. This means Bluemix provides organizations with a cloud platform that requires very little in-house technical know-how, as well as cost savings. At the same time, Bluemix lets organizations dynamically react to users' demands for new features. The Bluemix platform and the cloud provide the elasticity of compute capacity that organizations require when their apps explode in popularity.

As a reflection of IBM's commitment to open standards, there are ongoing efforts to ensure Bluemix is a leader when it comes to the open standards that surround the cloud. As such, Bluemix is an implementation of IBM's Open Cloud Architecture, leveraging Cloud Foundry to enable developers to rapidly build, deploy, and manage their cloud applications, while tapping a growing ecosystem of available services and run-time frameworks.

The API economy is at the core of Bluemix. It includes a catalog of IBM, third-party, and open source services and libraries that allows developers to compose apps in minutes. Quite simply, Bluemix offers the instant services, run times, and managed infrastructure that you need to experiment, innovate, and build cloud-first apps. Bluemix exposes a rich library of IBM proprietary technologies. For example, the BLU Acceleration technology behind the dashDB analytics service, the Cloudant NoSQL document store, IBM DataWorks refinery services, and more. IBM partner-built technologies (like Pitney Bowes location services) as well as open source services (like MongoDB and MySQL, among others) are also part of its catalog. The Bluemix catalog is by no means static—Bluemix continues to add services and shape the platform to best serve its community of developers.

In practical terms, this means that cloud apps built on Bluemix will reduce the time needed for application and infrastructure provisioning, allow for flexible capacity, help to address any lack of internal tech resources, reduce TCO, and accelerate the exploration of new workloads: social, mobile, and Big Data.

Bluemix is deployed through a fully managed IBM Cloud infrastructure layer. Beyond integration in the IBM Cloud, Bluemix has integration hooks for on-premise resources as well. Flexibility is the design point of Bluemix and it lets you seamlessly pull in resources from public and private clouds or other on-premise resources.

Layered security is at the heart of Bluemix. IBM secures the platform and infrastructure and provides you with the tools you need to secure the apps that you compose. IBM's pedigree in this area matters, and this is baked into the apps you compose on this platform.

Bluemix has a dynamic pricing scheme that includes a freemium model! It has pay-as-you go and subscription models that offer wide choice and flexibility when it comes to the way you want to provision Bluemix services. What's more, and perhaps unlike past IBM experiences, you can sign up and start composing apps with Bluemix in minutes!

Digging Deeper into PaaS

What are the key business drivers behind the market demand for PaaS? Consider an application developer named Jane. PaaS provides and tailors the environment that Jane needs to compose and run her apps, thanks to the libraries and middleware services that are put in her development arsenal. Jane and other end users don't need to concern themselves with how those services are managed or organized "under the covers." That complexity is the responsibility of the PaaS vendor (IBM) to manage and coordinate. More than anything, this new development paradigm decreases the time to market by accelerating productivity and easing deployment of new apps over the cloud. Are you a mobile game developer? If so, you can provision a robust and rich back-end JSON data store for your app in seconds, attach to it a visualization engine, and perform some analytics. It's like baking a cake—simply grab the premeasured and taste-tested ingredients, and bake away!

The Bluemix PaaS model platform enables new business services to be built and delivered without significant operational and capital expenditures, thanks to a platform that runs on top of a managed SoftLayer infrastructure and integrates cleanly between services. The traditional components of a development stack—the operating system, the integrated development environment (IDE), the change management catalog, the bookkeeping and tooling that every developer needs—can be provisioned with ease by using the PaaS architecture of Bluemix. If you are a developer of apps that are born and launched over the cloud, PaaS is your playground. Continuous delivery and code iteration become a reality when your apps and services can be implemented on a platform that supports the full DevOps process from beginning to end.

PaaS offers such tremendous potential and value to users like Jane because there are more efficiencies to be gained than just provisioning some hardware. Take a moment to consider Figure 3-3. Having access to subsystems and linkages to databases, messaging, workflow, connectivity, web portals, and so on—this is the depth of customization that developers crave from their environment. It is also the kind of tailored experience that’s missing from IaaS-only vendors. The layered approach shown in Figure 3-3 demonstrates how PaaS is, in many respects, a radical departure from the way in which enterprises and businesses traditionally provision and link services over distributed systems. Traditionally, classes of subsystems would be deployed independently of the app that they are supporting. Similarly, their lifecycles would be managed independently of the primary app as well. If Jane’s app is developed out of step with her supporting network of services, she will need to spend time (time that would be better spent building innovative new features for her app) ensuring that each of these subsystems has the correct versioning, functionality mapping for dependencies, and so on. All of this translates into greater risk, steeper costs, higher complexity, and a longer development cycle for Jane.

With a PaaS delivery model like Bluemix, these obstacles are removed: App developers and users don’t need to manage installation, licensing, maintenance, or availability of underlying support services. The platform assumes responsibility for managing subservices (and their lifecycles) for you. When the *minutiae* of micromanaging subservices are no longer part of the equation, the potential for unobstructed, end-to-end DevOps becomes possible. With a complete DevOps framework, a developer can move from concept to full production in a matter of minutes. Jane can be more agile in her development, which leads to faster code iteration and a more polished product or service for her customers. We would describe this as the PaaS approach to *process-oriented design and development*.

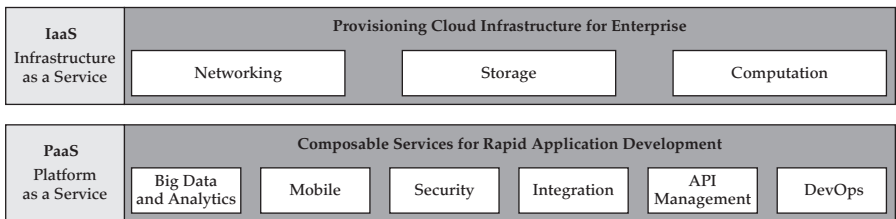


Figure 3-3 Platform as a service (PaaS) provides an integrated development environment for building applications that are powered by managed services.

Being Social on the Cloud: How Bluemix Integrates Platforms and Architectures

The cloud is a vast space, and the apps that you develop for it could very well need to reach out and “mingle” with other apps. Bluemix comes fully equipped with a catalog of integration and coordination services for external applications and other public clouds so that your apps can stay social. Later in this chapter, we discuss deployable PaaS and design patterns that make it easy to expose catalogs of apps and services. This style of automated deployment is ideally suited for PaaS and is paving the way for the *composable business*.

There are three potential PaaS configurations that you can run purely on the cloud. These three models have had the most success and widespread adoption among PaaS developers. A key differentiator for the IBM Cloud is that it can support all three of them (which is not the case with most other vendors).

In Figure 3-4 you can see the three main types of PaaS configurations. Common to all three of these PaaS configurations is that our developer Jane is at the top of the stack. Jane needs to compose new cloud applications with as little friction as possible, and so with each configuration (Types I–III), we have two clouds—Cloud A and Cloud B—running on top of an IaaS layer. Cloud A is running completely different services (mobile back-end libraries, Internet of Things device connectivity services) than Cloud B (database back ends, NoSQL document data stores). However, to power Jane’s app, she

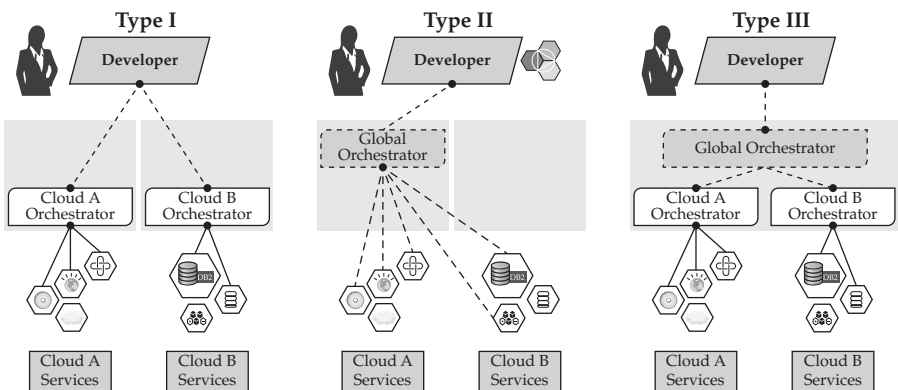


Figure 3-4 Three configurations of a platform as a service architecture. The cloud services can be governed by cloud-specific (Type I), global (Type II), or a hybrid of orchestrators (Type III).

needs to tap into both Cloud A and B. How she does that integration and how these required services are coordinated between those clouds (something we refer to as *orchestration*) is where we start to differentiate between the PaaS approaches. Whether Jane provisions services through a “single pane of glass” (a global orchestrator that has a predefined assortment of services to choose from) or defines linkages with these cloud services directly (even if they live on clouds that are separate from her PaaS environment) determines the type of PaaS architecture her solution requires.

Need a breather? Remember that this layer of complexity—how the pieces of PaaS and its services fit together—are not the concern of the developer. This is what’s great about PaaS after all! IBM handles the integration of services behind the scenes and determines how those services come together. In other words, the PaaS infrastructure in Bluemix determines how Jane’s instructions to a “single pane of glass” are translated across multiple cloud environments (internal or even external public clouds) to deliver the functionality that her apps need. We just added this tech section here in case you want a taste of the details that are going on behind the scenes, but never lose sight that with Bluemix, you don’t have to care about such details.

Understanding the Hybrid Cloud: Playing Frankenstein Without the Horror

What exactly do we mean by *hybrid cloud deployments*? It’s a term that is commonly heard in conversations about bridging on-premise and off-premise deployments. There is a huge elephant in the room any time we talk about moving services to the cloud, and that elephant is named “current investments.” Companies with an invested on-premise infrastructure are understandably reluctant to rip out those systems and migrate completely to the cloud, unless there is a sound business plan that surrounds the suggestion. We don’t think companies should rip and replace stuff. They need to come up with a strategic plan and, where it makes sense, leverage the cloud, and where it doesn’t, leverage what they have. Anyone who decrees “rip and replace everything” is pulling back into the cloud’s hype phase. Quite simply, as should be the case with any initiative, ensure that there is a sound business plan that surrounds the strategy.

A hybrid cloud environment that’s stitched together cross-premise can be either marvelous or monstrous. The key to leadership and innovation around

infrastructure hybridization is to start thinking in terms of “composite templates.” When workload requirements are well defined, you can leverage services in a way that plays to their strengths. For example, perhaps an off-premise cloud infrastructure would be well suited for experimental batch operations and an on-premise infrastructure would be well suited for performance-critical tasks. At the same time, perhaps governance regulations dictate a completely different premise location criterion. Also, never forget that without a network connection, the hybrid cloud model falls apart. A robust network is a must.

The key is to design hybrid access to services across infrastructures (on the cloud and on the ground). Composite templates are akin to the IBM PureSystems deployable patterns that we mentioned earlier. When hybrid services and access patterns are well defined and understood, companies such as IBM can wrap provisioning and architecture expertise into repeatable, deployable patterns.

The picture that we are painting is less the image of Frankenstein’s creature and much more a carefully crafted model made out of Lego bricks. Consider Figure 3-5: The concept of composable services is central to Bluemix

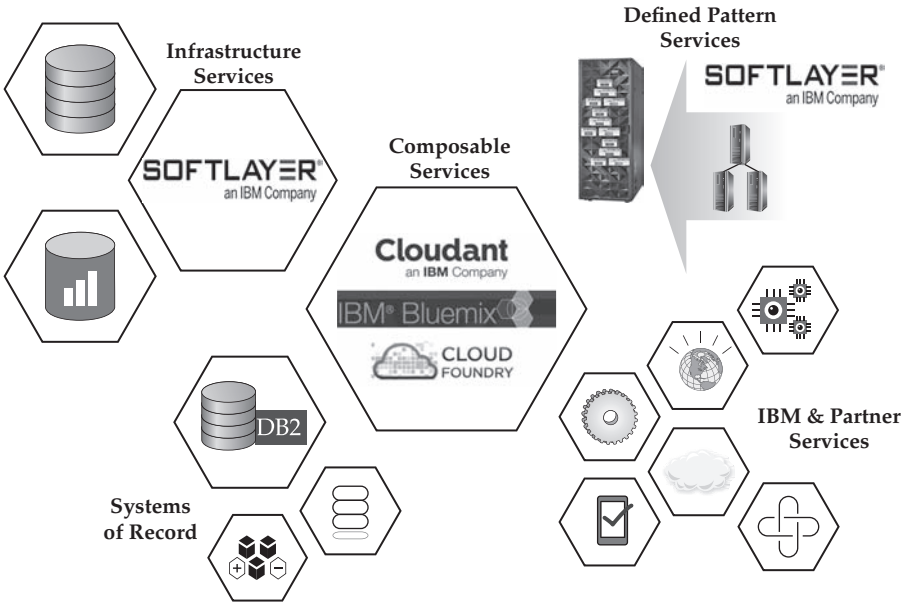


Figure 3-5 IBM Bluemix’s catalog of composable services is central to the integration of on-premise and off-premise infrastructure services.

and the IBM Cloud as a whole. What differentiates IBM's platform from competitors is the continuum of services that can be accessed across both on-premise *and* off-premise resources. Cloud-ready solutions such as SoftLayer make this possible, bursting data on demand to the cloud, as needed.

Tried and Tested: How Deployable Patterns Simplify PaaS

At its core, PaaS gives you control over the provisioning and automation of a mind-boggling number of patterned middleware services (development environments, run times, packaged services, business patterns, and so on) and the metering of application usage. For managed PaaS solutions, you can perform monitoring on a per-app basis (with respect to usage and performance), manage app and service licenses, conduct identity and security management, and manage push/pull access from mobile devices.

Truly advanced PaaS solutions such as IBM Bluemix provide data caching services, autoscaling in response to increasing (or decreasing) demand for services that are running on top of the platform, workload automation, and cloud bursting (for example, to a dedicated IaaS bare-metal environment like IBM SoftLayer) as needed. Bluemix's selection of reusable design patterns and automated service composition adds value, whereas its open ecosystem of services (including non-IBM proprietary and open source services) and integration patterns with traditional repositories add capability. A PaaS such as Bluemix enables organizations to offer their development teams a full-featured, end-to-end, continuous delivery environment where services can be provisioned and are composable on the cloud.

One question that you might be asking is "What is the relationship between PaaS and patterns?" Patterns enable you to encode business and development best practices into a deliverable that you can manage and deploy in a repeatable way. They are typically applied to existing problems that have been part of a marketplace or environment for a period of time. Only after a business's pain points have been thoroughly studied and understood can you have hardened best practices and access patterns to overcome those challenges—these are the "standard operating procedures" that are well suited for translation into deployable patterns. An apt analogy is vaccination against disease: There might be an experimental vaccine for a new disease that seems to improve a patient's health, but it is not until the efficacy

and safety of the treatment are well understood that the vaccine is mass produced for the general population.

To apply this analogy to business patterns, let's consider Figure 3-6. Although there might exist several ad hoc one-off strategies for solving a particular business use case, it isn't until the strategy demonstrates successful results over a period of time that IBM will consider it to be an expert deployable "patterned" solution for clients. IBM's entire PureSystems family is built around this concept of standardization and proven patterns to address business challenges of innovation, speed, and time to market. What's more, all of the expertise and effort that went into developing these on-premise patterned expert-baked solutions are now being leveraged for the Bluemix PaaS environment.

The challenge with deploying patterns to a cloud environment is that, from a design perspective, patterns are geared more toward existing technologies and are not necessarily optimized for iterative deployment of ever-evolving applications and architectures. For example, what is the pattern for taking a cookie-cutter implementation and days later stitching in a new service that just changed the scope of what your app is all about? We are getting at the fact that the cloud is a highly dynamic environment. The idea of establishing a hardened pattern around an environment where new code is deployed on the first day, a new database is added on the second day, a queuing system and then a monitoring system are tacked on after that, and who knows what else, becomes seemingly impossible.

So if apps that are born and developed on the cloud are built iteratively and gradually, how can design patterns ever get traction in such an environment? The answer is PaaS, because PaaS can be broken down and modeled in much the same way as *subpatterns*. There is a surprising amount of synergy between design patterns and the way that practitioners provision environments and services over the cloud. With this in mind, let's explore a layered approach to application development in the cloud.

At the top of Figure 3-6, developer Jane is actively making use of the Bluemix core services to compose and run cloud-native apps; this is what we refer to as the *Bluemix Fabric*. Meanwhile, Jane is taking advantage of a large catalog of external services (available through the API economy; more on this topic in a bit) such as database back ends, queuing services, and so on. The key idea here is that these back-end external services *can be provisioned using patterns*.

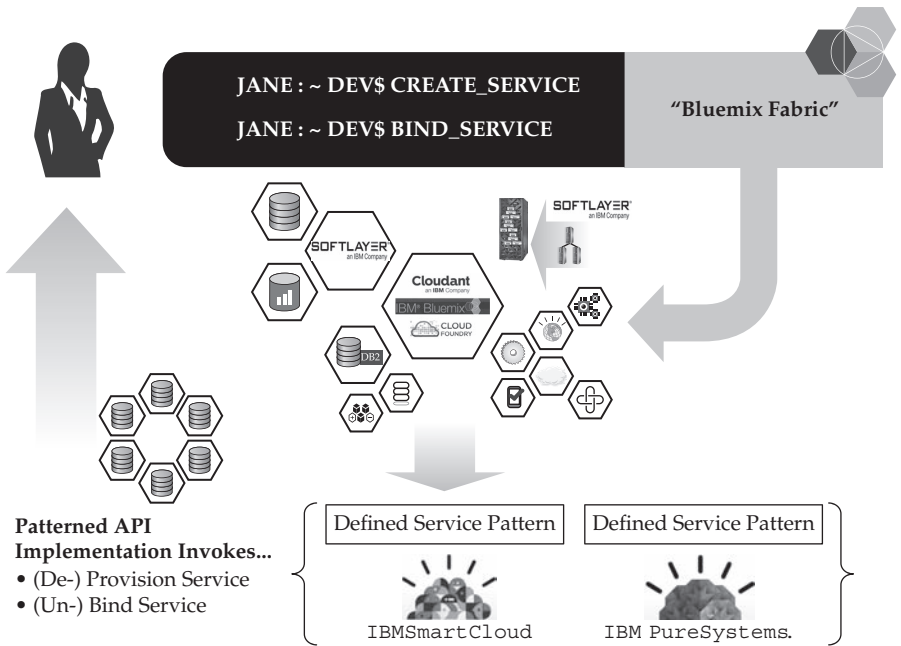


Figure 3-6 The PaaS “Bluemix Fabric” leverages design patterns to create a frictionless and productive environment for developers through services that are provisioned over the cloud.

For example, at some point in the coding cycle in Figure 3-6, Jane will need a service that can handle a certain requirement for her app—perhaps applying an XSL style sheet transformation that was requested by an end user. “Under the covers” Bluemix pushes down a design pattern that is used to instantiate the respective piece of software. After it is ready, the necessary hooks and API connectors to the service are returned to Jane so that she can tie them into the new app that is being developed on top of the PaaS. When design patterns are correctly leveraged as part of a PaaS environment, the complexities of provisioning and looping in external services are made transparent to the developer. In other words, Jane doesn’t need to worry about *how* it’s achieved, Jane simply moves forward with the confidence that the underlying PaaS environment handles that for her all abstracted through an API. Ultimately, this translates into the frictionless and more productive experience that developers like Jane thrive on.

Composing the Fabric of Cloud Services: IBM Bluemix

The culture around modern apps and tools for delivering content is constantly changing and evolving. Five years ago, there was no such thing as Kickstarter, the global crowd-funding platform. Four years ago, Instagram did not exist. There are more mobile devices today than there are people on the planet. We are in the “app now,” “want it now” era of technology. This shift isn’t a millennial or generational phenomenon, it’s a fundamental shift in the collective degree of impatience to get projects delivered faster and to get the things we want done faster. In many respects, this impatience is due to the general perception that the burdens of IT infrastructure are gone.

To be successful in this new culture, modern application developers and enterprises must adopt four key fundamental concepts:

- First is *deep* and *broad* integration. What people use apps for today encompasses more than one task. For example, today’s shopper expects a seamless experience to a storefront across multiple devices with different modalities (some will require gesture interactions, whereas others will use taps and clicks).
- The second concept is *mobile*. Applications need a consistent layer of functionality to guarantee that no matter how you interact with the storefront (on your phone, a tablet, or a laptop), the experience feels the same but is still tailored to the device. Here we are talking about the functionality that has to be consistent (we’ve all been frustrated by apps where the mobile version can’t do what the web version does), and for that to happen, the customer-facing endpoints need to be deeply integrated across all devices.
- Third, developers of these apps need to be *agile* and *iterative* in their approach to design. Delivery cycles are no longer set according to static points in time (a yearly release of your on-premise software, for example). Users expect continuous improvement and refinement of their services and apps, and code drops need to be frequent enough to keep up with this pace of change. After all, the cloud culture is so different: For example, clients can decide to keep their license on a monthly or even daily decision checkpoint. It’s easy to go elsewhere. Continuous delivery has moved from initiative to imperative.

- Finally, the *ecosystem* that supports these apps must appeal to customer and developer communities. Business services and users want to be able to use an app without needing to build it and own it. Likewise, developers demand an environment that provides them with the necessary tools and infrastructure to get their apps off the ground and often have next-to-zero tolerance for roadblocks (like budget approvals, DBA permissions, and so on).

These expectations and requirements precisely fit into the concept of PaaS and building cloud-native apps. Put another way, this is exactly the kind of environment that IBM Bluemix promotes and delivers.

As previously mentioned, Bluemix provides a PaaS over the IBM Cloud (powered by SoftLayer and Cloud Foundry) to deliver a venue for testing and deploying your apps. At its core, Bluemix is an open standards compliant, cloud-based platform for building, managing, and running apps of all types, especially for smart devices, which are driving the new data economy. You can think of Bluemix as a catalog of composable services and run times that is powered by IBM, open source, or third-party vendor-built technologies running on top of a managed hardware infrastructure. The goal is to make the process of uploading and deploying applications to the cloud as seamless as possible so that development teams can hit the ground running and start programming from day one. Bluemix capabilities include a Java framework for mobile back-end development, application monitoring in a self-service environment, and a host of other capabilities...all delivered through an as-a-service model. Bluemix offers a services-rich portfolio without the burden of mundane infrastructure tasks. Its target audience—developers, enterprise line-of-business users—can say goodbye to the drudgery of installing software, configuring hardware and software, fulfilling middle-ware requirements, and wrestling with database architectures. Bluemix has you covered.

As previously mentioned, Bluemix offers a catalog of services, drawing from the strengths of the IBM portfolio around enterprise-grade security, web, database management, Big Data analytics, cross-services and platform integration, DevOps, and more. These are known quantities—things that clients have come to expect when doing business with IBM. One of the things that the Bluemix team found is that developers often identify themselves with the technology that they choose to use. With this in mind, IBM also

wants to ensure that the Bluemix platform provides a choice of code base, language and API support, and an infrastructure that will attract developers from communities that IBM has not traditionally addressed in the past. For example, Bluemix supports Mongo-based apps. Quite simply, Bluemix eliminates entry barriers for these programmers and developers by offering a streamlined, cost-flexible, development and deployment platform.

Figure 3-7 offers a tiny glimpse at the Bluemix service catalog where developers can browse, explore, and provision services to power their apps. Being able to provide a full lifecycle of end-to-end DevOps is important, given the experimental, iterative, sandbox-like approach that we expect developers to bring to this platform.

The entire app lifecycle is cloud agile—from planning the app to monitoring and managing these services. All of these services are supported by the Bluemix platform right “out of the box.” This ultimately enables developers to integrate their apps with a variety of systems, such as on-premise systems of record (think transactional databases) or other public and private services that are already running on the cloud, all through Bluemix APIs and services.

If you want other developers on Bluemix to tap into the potential of these systems, you can work with IBM to expose your APIs to the Bluemix audience by becoming a service partner. Consider the fundamental Bluemix initiatives that we outlined: Bluemix is a collaborative ecosystem that supports

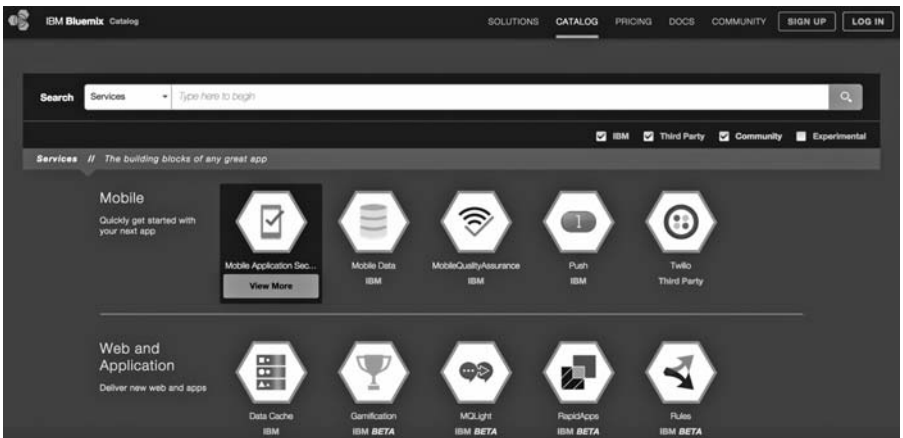


Figure 3-7 The Bluemix catalog of composable services comes from IBM, partner, and open source technologies that are enabled for the continuous, agile development of cloud apps.

both developers and customers, and enabling our partners to integrate across these verticals is a key part of that initiative.

Parting Words on Platform as a Service

We dedicated much of this chapter to PaaS, a key anchor of the as-a-service model. PaaS is about more than simply giving developers a sandbox to work in. Platforms such as IBM Bluemix enable transparent integration among multiple environments “under the covers.” What this means for our users is access to a myriad of services and tools, all through a single “pane of glass.” Bluemix rapidly brings products and services to market at a lower cost, thanks to the flexibility and on-demand nature of cloud-based PaaS. The fact that PaaS is a low up-front investment vehicle for application development encourages experimentation and innovative development, without concern for software configuration and hardware provisioning, or the need for on-hand staff to manage the infrastructure. Furthermore, Bluemix’s iterative nature encourages developers to continuously deliver new functionality to their applications. Bluemix is designed so that developers can iterate on their code and app functionality as rapidly and painlessly as possible. Bluemix extends a company’s existing investments in IT infrastructure. Many clients have heterogeneous, complex, on-premise environments that must be linked to new born-on-the-cloud applications.

With an on-demand and services-based infrastructure, Bluemix enables developers to go from raw to fully baked apps in as short a time as possible. Clients are able to securely connect to their existing on-premise infrastructure to power these apps, ensuring that apps made in the cloud can effectively leverage the resources that these companies have on the ground. With DevOps, teams both large and small are able to automate the development and delivery of applications: Bluemix makes the process of continuous iteration simple and straightforward for both developers and line-of-business users.

Consuming Functionality Without the Stress: Software as a Service

If PaaS is oriented toward the developer who wants to code and build the next generation of applications in a frictionless environment, then software as a service (SaaS) is geared much more toward the line-of-business (LOB)

user who needs to consume the capabilities of a service that is already built and ready to deploy. SaaS consumers span the breadth of multiple domains and interest groups: human resources, procurement officers, legal departments, city operations, marketing campaign support, demand-generation leads, political or business campaign analysis, agency collaboration, sales, customer care, technical support, and more. The variety of ways that business users can exploit SaaS is extraordinarily complex: It can be specific (to the extent that it requires tailored software to address the problem at hand), or it can require a broad and adaptable solution to address the needs of an entire industry.

Not every SaaS vendor can guarantee that their packaged software will satisfy every vertical and niche use case we’ve outlined here. Nor do all SaaS vendors offer a catalog of services that supports deep integration hooks and connectors for when you need to scale out your solution or roll in additional tools in support of your business end users. Figure 3-8 highlights the fact that SaaS can apply to anything, as long as it is provisioned and maintained as a service over the cloud.

What differentiates IBM’s solution is that it offers the complete stack—from IaaS to PaaS to SaaS—in support of every possible customer use case. We would argue that IBM’s portfolio stands alone in being able to offer both tailored and extensible SaaS deliverables to address the various dimensions of *consumable* and *composable* functionality that are required by enterprises today.

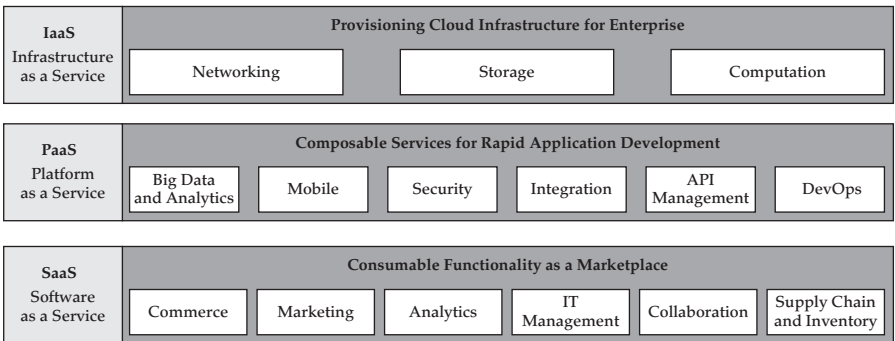


Figure 3-8 SaaS delivers software that runs on the cloud, enabling you to consume functionality without having to manage software installation, upgrades, or other maintenance.

And consider this: As developers build their apps on the Bluemix PaaS platform, they can seamlessly bring their solutions to a SaaS market and a sophisticated IBM Cloud Computing Marketplace (ibm.com/cloud-computing/us/en/marketplace.html), where new clients can consume them. Throughout this section, we explore SaaS and IBM's offerings in this space in more detail, but first, it is worth setting the stage by describing how cloud distributions are reshaping software.

The Cloud Bazaar: SaaS and the API Economy

In the marketplace today, there is a trend toward what many have called the *API economy*. The API economy is the ability to programmatically access anything through well-defined protocols and APIs. API economies enable developers and end users to easily and seamlessly integrate services with any new apps that they are building or consuming on the cloud. This is quite a departure from how things were done before. Many offerings that could historically be described as SaaS were driven by the user interface (UI). Today, companies such as Twitter and eBay leverage API integration into third-party services to drive revenue and traffic *far more* than they rely on a UI-driven strategy for attracting users.

At the most basic level, SaaS apps are software that is hosted and managed in the cloud. Salesforce.com is a well-known SaaS offering for customer relationship management (CRM). Instead of hosting your own CRM system on premise, the app resides on the cloud. Here's the stack: All software components are pre-installed (SaaS); the database and application server that support the app are already in place (PaaS); this all lives on top of a fully managed infrastructure layer (IaaS). As a client, you just interact with the app; there is no need to install or configure the software yourself. The key thing to remember is that although the app feels like a discrete piece of business logic, under the covers the SaaS model is more likely a collection of dozens (even hundreds or thousands) of APIs working in concert to provide the logic that powers the app.

Consider a service like Twitter. Its UI is minimal and straightforward: You have a timeline, 140 characters to compose your message, and a button to post your tweet. Under the covers, however, it calls a dizzying number of

services through APIs. The real value of Twitter lies in its ability to seamlessly integrate with other apps. Have you ever bought something online and found an integrated tweet button for sharing your shopping experience? That vendor is leveraging Twitter’s API to help you announce to the world that your new golf club is the solution to your slicing issues. When you tweet, underlying service authorizations can store your tweeted image in Flickr and leverage Google Maps to enrich the tweet with geospatial information visualizations. The API economy also lets you use that same tweet to tell all your Facebook friends how your golf game is set to improve. Such a neatly orchestrated set of transactions between distinct services is made possible and is done seamlessly by the API economy that drives today’s cloud SaaS marketplace, as shown in Figure 3-9.

Having a well thought-out and architected API is crucial for developing an ecosystem around the functionality that you want to deliver through an as-a-service property. Remember, the key is designing an API for your services and software that plays well with others. Your API audience is not only people in your company, but partners and competitors too. Consumers of the SaaS API economy require instant access to that SaaS property (these services need to be always available and accessible), instant access to the API’s documentation, and instant access to the API itself (so that developers can program and code against it). Furthermore, it is critical that the API architecture be extensible, with hooks and integration end points across multiple domains (Figure 3-9) to promote cooperation and connectivity with new services as they emerge.

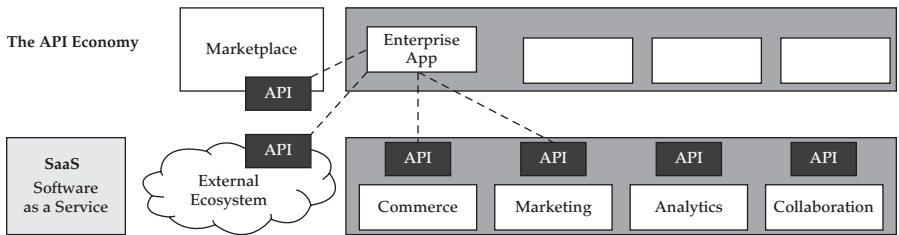


Figure 3-9 IBM’s “API economy” is made possible by a unified cloud architecture of well-defined integration end points between external ecosystems, cloud marketplaces, and IBM’s SaaS portfolio.

Demolishing the Barrier to Entry for Cloud-Ready Analytics: IBM's dashDB

In this section we want to talk about an analytics service from IBM called dashDB. As we went to press, the dashDB services were scheduled for release in both a PaaS model (for those composing apps that require analytics) and a SaaS model (for those that simply need a turnkey analytics service). What we describe in this section is the end goal; it might be the case that some of the things we talk about here aren't available when you first get this book, but they will come soon after. We are also going to talk about dashDB from a SaaS perspective.

Moving forward, you are going to see the IBM Information Management brand deliver capabilities on the cloud first in a much faster manner than with traditional models. This is in lockstep with the constant feature delivery and innovation users of the cloud expect from the services they consume there. For example, at its core, dashDB is the confluence of Netezza (Chapter 9) and BLU Acceleration (Chapter 8) technologies; it also includes some IBM DataWorks refinery services (Chapter 13), a Cognos visualization engine, and runs on SoftLayer.

What does continual integration look like? Netezza (also known as the IBM PureData System for Analytics) is famous for its deeply embedded IBM Netezza Analytics engine (INZA). Over time, more and more of the INZA capabilities will get deeply embedded into dashDB. The vision is that at some point you will be able to run all the same INZA algorithms that you can run in Netezza today directly on dashDB. The same can be said for the Netezza SQL API. IBM dashDB includes Oracle compatibility too, although this capability comes from another source (DB2 has a native execution layer for Oracle PL/SQL so that service is embedded in dashDB as well; you can see how the apps made of service components are the future). This continual feature delivery methodology empowers organizations with the ability to burst more and more analytics into the cloud from their Netezza appliances with less and less friction. At the same time, innovations that arrive in the cloud first will eventually find their way on premise. For example, the INZA capabilities being built into dashDB will be integrated into on-premise software instantiations of BLU Acceleration, like DB2.

So what is the difference when consuming dashDB as a PaaS or SaaS model? Developers can use PaaS to compose an app. They might want to

surface the application in a SaaS model and charge for it down the road; IBM gives you that option as a service partner. That said, clients can use dashDB in a PaaS model for their own analytics needs—it's just not managed in the same way as a SaaS model. However, for customers who simply want to consume software without building and assembling it themselves, a ready-to-go and managed SaaS solution might be just the ticket. Remember, SaaS isn't for developers; it's for users and consumers who need to achieve new business outcomes rapidly and cost effectively.

You may see dashDB referred to as a data warehouse as a service (DWaaS)—which we think is best conceptualized as an added layer of classification atop the software as a service (SaaS) model. Fundamentally, DWaaS is very much a SaaS offering—clients consume the functionality of the service (in the case of dashDB, that functionality entails best-of-breed performance and analytics) without concern over provisioning or managing the software's lifecycle themselves.

IBM's portfolio of SaaS offerings is built on the same foundations of resiliency, availability, and security as the IaaS and Bluemix PaaS layers we talked about in this chapter. What differentiates dashDB from the pack are the three pillars that set DWaaS apart from the traditional on-premise data warehousing that has come before it: simplicity, agility, and performance. Let's take a moment to tease apart what each of these pillars offers clients that provision this service. *Simplicity*: dashDB is easy to deploy, easy to use, requires fewer resources to operate, and is truly "load and go" for any type of data you can throw at it. *Agility*: "train of thought" analytics are made possible by dashDB's integration across the IBM Big Data portfolio, making your apps faster and delivering actionable business insight sooner. *Performance*: dashDB delivers BLU Acceleration's signature, industry-leading performance at a price the market demands—all within a cloud-simple form factor. In ten words or less, dashDB gives faster actionable insight for unlocking data's true potential.

IBM dashDB is all about the shifting client values around how functionality is consumed and the explosive growth that we are witnessing in the delivery of SaaS on the cloud. Customers today are focused on assembling the components and services that they need to achieve their business outcomes. In terms of the marketplace, analytics are becoming increasingly valuable; in fact, we'd go so far as to say that in the modern marketplace, analytics are indispensable—like we said before, from initiative to imperative. Analytics

are driving decisions about how skills are deployed across an organization, how resources and infrastructure are distributed in support of a business, and how clients access the mountains of data that they are actively creating or collecting.

One question being asked by many businesses is “How do we rapidly transition from data *collection* to data *discovery and insight* to drive decision making and growth strategies?” As companies move aggressively into this space, they quickly realize that the skills that are required to build the tools and perform data analysis are still pretty scarce; the combination of statistician and DBA guru in one individual is hard to come by! Cost becomes a major obstacle for companies that want to deploy or manage business analytics suites with an on-premise infrastructure and appliances. The future innovators in this space will be companies that enable their partners and customers to bypass the risk and costs that were previously associated with analytics. This is the drive behind SaaS delivery and in particular IBM’s dashDB.

The idea of an analytics warehouse that can be consumed as a service is a product of powerful solutions and infrastructure emerging at just the right time...over the cloud. Speaking from our own firsthand interactions with clients, we can confidently say that the vast majority of IT and enterprise customers want to move at least part of their resources into a cloud environment. The trend is increasingly toward applications that are “born” in the cloud, developed and optimized for consumption over distributed systems. A key driver behind this momentum is centered on CAPEX containment, which is a SaaS product that scales elastically and can be “right-sized” for your business, translating into smaller up-front costs and greater flexibility down the road.

SaaS solutions are distinctly “cloud first.” We believe that following best practices and architecting a software solution for the *cloud first* makes translating that asset into an on-premise solution much more straightforward and painless. In terms of the performance of on-premise and cloud deliverables, the laws of physics are ultimately going to win. With the cloud, there is the unavoidable fact that your data and instructions need to travel across a wire. In most cases, the latency in cloud deployments will be at least *slightly* greater (whether or not it is significant is another question). Being able to use in-memory analytic capabilities, perform data refinement, and ingest data for analysis in a real-time environment, all in the cloud, is propelling the industry toward new opportunities for actionable business insight. This is all being

made possible through cloud-based SaaS. Don't forget—the cloud isn't so much about raw throughput performance as it is about agility and scale: how fast you can build, deploy, and iterate applications; how fast can you scale; or how fast you can attach (and tear down) add-on services such as analytic data marts.

Build More, Grow More, Know More: dashDB's Cloud SaaS

IBM dashDB can be provisioned as an expansion to the Cloudbant NoSQL data layer, with purpose-built hooks for handling the ingestion and migration of data between Cloudbant's JSON document and the dashDB analytics service. This is the kind of stuff we were referring to earlier in this chapter when we talked about integration of service points. For example, a mobile gaming app that stores information in Cloudbant can back-end sync the JSON data into dashDB for light-speed analytics on collected information through a built-in facility.

The predecessor to dashDB on Bluemix is the Analytics Warehouse Service, which was solely based on BLU Acceleration technology. Over time, you will see users of this service migrate to dashDB, which provides a richer managed service and comes with the extended set of capabilities we outlined earlier.

IBM dashDB fits the pattern of whichever analytics use case your organization needs to fulfill. It can be procured as an integrated analytic warehouse that is embedded with Watson Analytics, coupled with Cloudbant's NoSQL data stores, or integrated with services available in the IBM DataWorks data refinery catalog. Alternatively, dashDB can serve as a stand-alone analytics engine for clients looking to leverage the elasticity and flexibility of a DWaaS (remember: software as a service with added granularity specific to data warehousing) cloud deliverable. Again, for application developers working with IBM Bluemix, dashDB can form a vital piece in your composable fabric of services running atop IBM's managed PaaS.

The takeaway message for dashDB is as follows: It's a high-performance analytics service, it's delivered in a managed "cloud-easy" way (there is nothing for you to do but click and analyze), it has best-in-class simplicity in terms of provisioning and usage, and it comes with a rich set of built-in security features for enterprise customers. IBM dashDB enables you to build more, grow more, and know more.

We say build more because dashDB gives you the flexibility to get started without making costly CAPEX investments, which reduces risk and mitigates cost. The provisioning of this SaaS takes less than a dozen or so minutes for 1TB of data, and migration and performance have been tuned to delight users with their first experience. To support bursty workloads from on-premise infrastructures to dashDB in the cloud, a lot of effort has been put into data transfer rates. (Mileage is going to vary—remember that your network connection speed matters.) When these pain points are eliminated, LOB owners are free to experiment and pursue new insights that are born out of the cloud—in short, they can build more.

The beauty of the SaaS model is that capability enhancements can be delivered to your business *without* performance degradation or downtime to your customers. Consumption of its analytics capabilities becomes your only concern—that, and growing your business with your newfound insights, and this is why we say dashDB lets you grow more. Think about it: You can put personnel on discovering new insights and leveraging investigative techniques you could never apply before, or you can spend your time upgrading your on-premise investments.

IBM dashDB's in-database analytics also includes R integration for predictive modeling (alongside the INZA functionality that will continue to work its way into the service offering as dashDB iterates over time). R's deeply woven integration with dashDB delivers in-database functions that span the analytic gamut: linear regression, decision trees, regression trees, K-means, and data preparation. Other statistical showcases, such as support for geospatial analytics and developing analytics extensions through C++ UDX, will eventually round out dashDB's analytics portfolio.

From a security perspective, dashDB is Guardium-ready (covered in Chapter 12). Industry-leading data refinery services will add free (and some for fee) masking, transformation, and enrichment, as well as performance-enhancing operations, which we briefly discuss in the next section.

Refinery as a Service

Both Bluemix and Watson Analytics are platforms that consume and produce data; as such, they need governance and integration tools that fit into the SaaS model we've been describing in this chapter. IBM's engineering teams are taking its rich data governance capabilities and exposing them as a

collection of rich composite and data refinery services known as DataWorks. Available at the time of writing are data load, masking, and data selection services that you can leverage in an integrated manner with your other services. We cover these capabilities in Part III of this book.

Wrapping It Up

Having well-performing and powerful analytics tools is critical to making data warehousing in the cloud a worthwhile endeavor. If you can get an answer in seconds rather than hours, you are more likely to think about the next step of analysis to probe your data even further. If you can set up a system in less than an hour instead of waiting months for budgetary approval and dealing with the numerous delays that get between loading dock and loading data, you can deliver business results sooner. This is why we say that dashDB and its BLU Acceleration technologies enable “train of thought” analytics: Having a high-performance analytics SaaS toolkit keeps business users hungry for more data-driven insight.

In this chapter, we implicitly touched on the four core elements that are central to IBM’s brand and its cloud-first strategy. The first is *simplicity*: offering customers the ability to deploy, operate, and easily manage their apps and infrastructure. “Load and go” technology makes it possible to get off the ground and soar into the cloud quickly and cost effectively. *Agility* is next: If your business can deploy and iterate new applications faster, you will be positioned to adapt to a rapidly changing marketplace with ease. IBM’s as-a-service portfolio (for infrastructure, development platforms, and software) makes that a reality. The third element (it’s a big one) is *integration* with Big Data. Analytics that are being run in platforms such as Hadoop do *not* live in isolation from the type of questions and work that is being done in data warehousing. These technologies work in concert, and workflows between the two need to exist on a continuum for businesses to unlock the deep and actionable insight that is within reach. The fourth element in IBM’s cloud vision is a *unified experience*. As we show in Figure 3-10, the integration hooks, interoperability, migration services, consistent and elastic infrastructure, and platform services layers that IBM’s cloud solution has in place (and continues to develop) offer a truly holistic and unified experience for developers and enterprise customers alike.

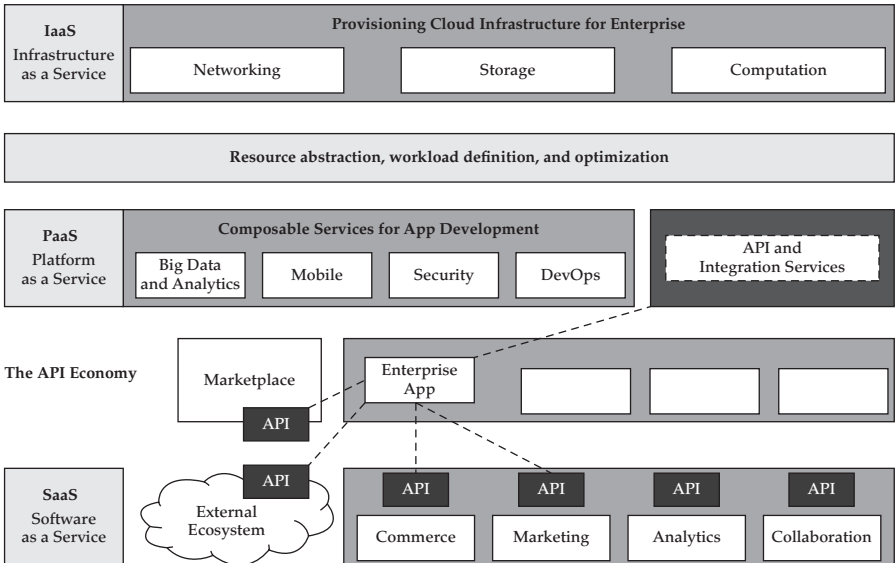


Figure 3-10 IBM’s cloud portfolio delivers a holistic approach to deeply integrated infrastructure, platform, and software as a service environments that is unrivalled in scalability, availability, and performance.

No other competitor in the cloud space can offer an equally rich as-a-service environment with solutions that are designed for the enterprise, and still remain open to emerging technologies from partners and third-party vendors. *Cloud* used to be a term that was surrounded by hype, but over the years, the as-a-service model has transformed business challenges that had remained unsolved for decades into new sources of opportunity and revenues; quite simply, this proves that cloud is beyond the hype. As competitors race to stitch together their own offerings, IBM customers are already designing apps and composing solutions with a Bluemix fabric, consuming software as a service functionality on top of a SoftLayer infrastructure that lives in the cloud.

This page intentionally left blank

4

The Data Zones Model: A New Approach to Managing Data

The Big Data opportunity: Is it a shift, rift, lift, or cliff? We think it is a matter of time until you are using one (ideally two) of these words to describe the effect that Big Data has had on your organization and its outcomes. Study after study proves that those who invest in analytics outperform their peers across almost any financial indicator that you can think of: earnings, stock appreciation, and more. Big Data, along with new technologies introduced by the open source community and enhancements developed by various technology vendors, is driving a dramatic shift in how organizations manage their analytic assets. And when it's done right, it has the effect of lifting those organizations' Analytics IQ, our measure for how smart an organization is about analytics.

A common challenge to any organization investing in its Analytics IQ has been providing common access to data across the organization when the data is being generated in various divisional and functional silos. In a social-mobile-cloud world, this question hasn't changed, but its answer certainly has. Historically, the answer to this question was to make the enterprise data warehouse (EDW) the center of the universe. The process for doing this had become fairly standardized too. *If companies want an EDW, they know what to do: Create a normalized data model to make information from different*

sources more consumable; extract required data from the operational systems into a staging area; map the data to a common data model; transform and move the data into a centralized EDW; and create views, aggregates, tiers, and marts for the various reporting requirements.

Over time, various challenges to this approach have given rise to some variations. For example, some organizations have attempted to move their data transformation and normalization processes into the warehouse, in other words, shifting from extract, transform, and load (ETL) to extract, load, and transform (ELT). Some organizations invested more heavily in operational data stores (ODSs) for more immediate reporting on recent operational and transaction-level data.

As organizations began to do more predictive analytics and modeling, they became interested in data that was not necessarily in the EDW: data sets whose volumes are too large to fit in the EDW, or untouched raw data in its most granular form. For example, consider a typical telecommunications (telco) EDW. Perhaps the most granular data in a telco is the call detail record (CDR). The CDR contains subscriber data, information about the quality of the signal, who was called, the length of the call, geographic details (location before, during, and at the end of the call), and more. Because this data is so voluminous—we're talking terabytes per day here—it can't possibly reside in the EDW. The data that ends up in the EDW is typically aggregated, which removes the finer details that a data scientist might require. This data is also likely to be range-managed; perhaps only six to nine months of the data is kept in the EDW, and then it's "rolled out" of the repository on some scheduled basis to make room for new data. As you can imagine, this approach makes it tough to do long-term historical analysis.

Analytics teams respond to such challenges by pulling down data directly from both operational systems and the data warehouse to their desktops to build and test their predictive models. At many organizations, this has created a completely separate path for data that is used for analytics, as opposed to data that is used for enterprise reporting and performance management.

With data center consolidation efforts, we've seen some organizations that make use of a specific vendor's warehousing technology load their EDWs with as much raw source system data as they can and then use the warehouse mostly for data preparation (this is where the ELT we mentioned earlier is used). As you can imagine, if an EDW is to be a trusted and highly

responsive repository of information, saddling it with data preparation tasks and stale data drives up costs. In fact, we've seen some clients appropriate up to 60 percent of these systems' resources to simply manage cold untouched data or data transformation work. Figure 4-1 summarizes such a traditional architecture for enterprise data management.

EDWs didn't become problematic overnight. These shifts have been a gradual evolution. The high-level architecture and the overall approach to EDWs have been fairly constant...until recently, that is, when Big Data changed things. To synergize and optimize a modern-era analytics architecture, you have to think in *zones*—separate areas for storing, processing, and accessing your information based on their particular characteristics and purposes. These zones foundationally sit on the polyglot environment discussed in Chapter 2, but they also need data refinery services and governance, real-time processing, and more. What's more, they need to be able to access the foundation (or leverage capabilities) off premise in the cloud, on premise, and more likely in a hybrid environment that includes both, with the capability to burst into the cloud as needed. If you're familiar with The Zone diet, which advocates balance with respect to the source of your daily caloric intake in the form of carbohydrates, proteins, and fats, it's the same kind of thing. The data zones model that we talk about in this chapter is designed to deliver a balanced calibration of the right technology, economic cost curves, and data preparation whose confluence boosts your organization's Analytics IQ.

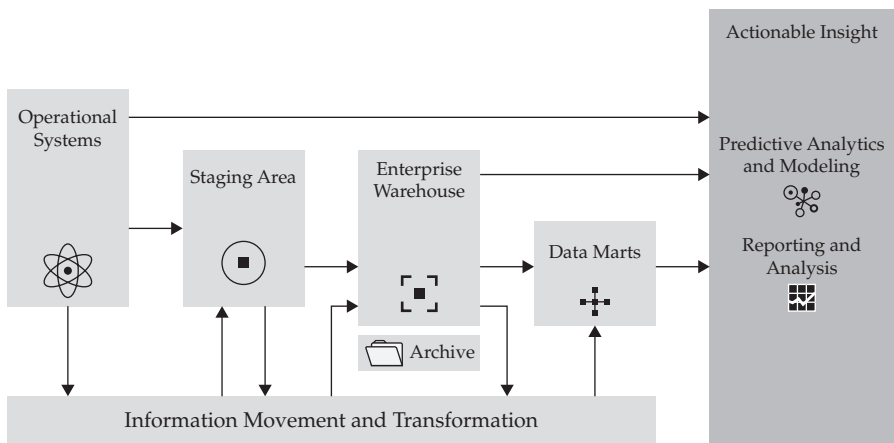


Figure 4-1 A traditional approach to enterprise data management

Challenges with the Traditional Approach

The biggest challenges that organizations are facing with the traditional EDW approach can be placed into three categories: *agility*, *cost*, and *depth of insight*.

Agility

Agility matters. It's critical in sports, politics, organizational effectiveness, and more. Agility is really a measurement of the ability to adapt or change direction in an efficient and effective manner to achieve a targeted goal. In fact, agility is so important that almost every role-playing game (RPG) has some sort of key performance indicator to measure a character's ability to carry out or evade an attack or to negotiate uneven terrain. Organizations are discovering that with traditional EDW approaches, it takes too long to make information available to the business. A common complaint from most business users is that any time they make a request for some additional data, the standard response is that it will take at least "several months" and "more than a million dollars"...and that's even *before* they share their specific requirements! But if you think about what it takes to make new information available in this kind of environment, responses like this are actually not as unreasonable as they might initially sound. At this point, we are beginning to feel a little sorry for operational teams because they really do get it from both ends: Developers don't want to wait on them (Chapter 2) and neither does the line of business (LOB). We think that IT ops is like parenthood, which is often a thankless job.

In the traditional "EDW is the center of the universe" architecture, if a LOB requires additional data to investigate a problem, several steps need to be taken. Of course, before even getting started, you have to figure out whether the required data already exists somewhere in the EDW. If it doesn't, the real work begins.

You start by figuring out how the new data should fit into the current normalized data model. You will likely need to change that model, which might involve some governance process in addition to the actual tasks of modifying the database schema. The next step (we hope) is to update the system's documentation so that you can easily identify this data if you get a similar request in the future. Of course, the database schema changes aren't in production yet; they are in some testing process to ensure that the required changes

didn't break anything. After you are confident that you can deliver on the LOB's request, you need to schedule these changes in production so that they don't impact any business-critical processes. What about total time? It depends on your environment, but traditionally, this isn't an hour or days measurement; in fact, weeks would be remarkably good. Our scenario assumes you have all the data lined up and ready to go—and that might not even be the case.

After you've taken care of the "where the data will go" part, you have to figure out where and how to get the data. This typically involves multiple discussions and negotiations with the source system owners to get the data feeds set up and scheduled based on the frequency and timeliness requirements of the business. You can then start working on how to map the source data to your new target model, after which you have to start building or updating your data integration jobs.

After all this is done, your new data is *finally* available. Only at this point can you start building queries, updating reports, or developing analytic models to actually leverage the new data. Tired yet? Sound familiar? What we are talking about used to be the status quo: The zones model this chapter introduces you to is about turning the status quo on its head, in the name of agility, insights, and economies of scale.

This is a major reason why organizations continue to see new databases pop up across various LOBs and continue to experience data sprawl challenges even after investing in the ideal EDW. Remember, just like developers, business users require agility too. If IT ops doesn't deliver it to them, they will go and seek it elsewhere. Agility is one of the reasons that investing in a data zones model is a great idea.

Cost

No one ever suggested that EDWs are cheap. In fact, an EDW is one of the most expensive environments for managing data. But this is typically a conscious decision, usually made for very good reasons (think back to the gold miner discussion we had in Chapter 1). High-end hardware and storage, along with more advanced, feature-rich commercial software, are used to provide the workload management and performance characteristics needed to support the various applications and business users that leverage this platform. These investments are part of a concerted effort to maintain the

reliability and resilience that are required for the critical, enterprise-level data that you have decided to manage in the EDW. In addition, as mentioned earlier, ensuring that you provide the consistent, trusted information that is expected from this investment drives up the effort and costs to manage this environment. Although EDWs are important, they are but a single technology within a polyglot environment.

Another thing about the “EDW is the center of the universe” approach is that organizations continuously have to add more and more capacity as the amount of data they are capturing and maintaining continues to rise and as they deliver additional reporting and analytic solutions that leverage that data. Companies are starting to realize that they will need to set aside significant amounts of additional capital budget every year just to support the current needs of the business, let alone the additional reporting or analytical requirements that continue to arise. Often, there isn’t a clear understanding of the impact of the new applications and analytics that are being developed by the LOB. As a result, organizations can unexpectedly run into capacity issues that impact the business and drive the need for “emergency” capital expenditures, which are usually more costly and don’t provide the flexibility to accommodate alternative approaches.

The cost issues all come down to this question: “Do you need to incur these high costs for *all* of your data, or are there more cost-effective ways to manage and leverage information across the enterprise?”

Depth of Insight

The third category of challenges that organizations face with the traditional EDW approach is related to the maturity and adoption of current approaches to enterprise data management. Initially, the organizations that invested in aggregating data for analysis and establishing an enterprise view of their business were able to drive competitive advantage. They were able to generate new insights about their customers and their operations, which provided significant business value. However, the early adopters are starting to see diminishing returns on these investments as their peers catch up.

Organizations are looking for new ways to deliver deeper, faster, and more accurate insights. They want to incorporate larger volumes of data into

their analyses. They want to be able to leverage different types of information that are not necessarily structured or suited for a typical relational database. And they want to be able to generate insights much more quickly, often in real time. The world simply doesn't fit into neat and tidy rows and columns anymore—in fact, it never did! There's a great deal of semistructured and unstructured data out there, waiting to be harvested, that can take an organization's Analytics IQ to the next level.

Unfortunately, the historic approach to managing enterprise information has not lent itself to addressing most of these requirements. And those that could potentially be addressed by current systems cannot usually be addressed cost effectively. In some cases, costs grow because IT tries to manage unstructured data as though it were structured data, essentially trying to push a square peg into a round hole.

While many organizations have tried to make the EDW the center of the universe, forward-looking, data-driven organizations are now starting to realize that this might no longer be the right approach, given business requirements for more rapid response times, the growing volumes and complexity of data, and competitive demands. A shift toward purpose-built components is required to more effectively manage costs and compete in today's market.

Next-Generation Information Management Architectures

Next-generation (NextGen) architectures for managing information and generating insight are not about a single technology. Despite what some people think, Big Data *does not* equal Hadoop. Hadoop does play a major role in NextGen architectures, but it's a player on a team. There are various technologies, along with alternative approaches to managing and analyzing information, that enable organizations to address the challenges we have discussed and to drive a whole new level of value from their information.

The most significant transformation affects how data is maintained and used in different zones, each supporting a specific set of functions or workloads and each leveraging a specific set of technologies. When you have these zones working together with data refinery services and information integration and governance frameworks, you'll have a *data reservoir*.

Prepare for Touchdown: The Landing Zone

One of the most critical zones in a NextGen analytics architecture is the “landing” zone. It is fair to say that most organizations have already put in place some type of staging or landing area for their data, but it is the Next-Gen approach that is going to dramatically change your business. In the past, data from operational systems was copied to a “dumb” staging area, which was often an FTP server. These files had to be extracted, transformed, and loaded into some structured data repository before the data could be accessed or used. An alternative attempt to streamline this rigid process involved putting the data files directly into a database. But this approach still required an understanding of the existing source schema and configuration efforts in the target database to match that schema.

The landing zone is an area where Hadoop can play a central role. Organizations are beginning to discover that by landing all of the data from their operational systems in Hadoop, they can just “dump” the data as is, without having to understand the schema in advance or to set up any table structures, but still maintain the ability to access and work with the data. So, in essence, Hadoop is enabling companies to create smarter, more functional landing zones.

Some organizations are leveraging this enhanced capability to transform the way they respond to requests for more data from the business. They are shifting from selectively extracting specific data from operational systems to copying *all* of the data from their operational systems to this new Hadoop-based landing zone. Then, when the business asks for additional data, they point directly to the landing zone to provide more immediate access to the data. When you know exactly what data is needed to address the specific business problem in question, the data can be transformed and moved into a more refined state or repository. For example, perhaps a raw data set was “discovered” in the refinery. A LOB user could “massage” this data and chose to transform it, leveraging parts of the IBM DataWorks catalog. At this point she could then spin up a dashDB analytics service in the cloud in minutes and start to investigate the data. This type of approach requires sophisticated exploration capabilities, as well as on premise and off premise modality.

If all data lands in Hadoop, the data in its rawest form can be made available to those groups of users who were becoming frustrated at being provided with only aggregated data related to their problem domain—which is the focus of the next section.

Into the Unknown: The Exploration Zone

Landing data in Hadoop is powerful—and it is not just because of the economic benefits. Hadoop provides the ability to actually do something with the data prior to it being transformed or structured. This is a key aspect of Hadoop. It also happens to be a key value proposition in IBM's Big Data and Analytics platform: Hadoop *is also* for discovery and analytics. Most vendors in the Hadoop space agree with this concept; however, there are two notable traditional warehouse vendors that go to great lengths to portray Hadoop solely as a data preparation platform and mostly ignore its potential as a platform to accelerate discovery and analytics.

Right after data lands in Hadoop, you can make the data accessible and start generating some value prior to defining a schema. (This is why Hadoop is generally classified as a *schema-later* paradigm, whereas the relational world is generally classified as a *schema-first* paradigm; see Chapter 2 for more details.) Because Hadoop easily accommodates any data shape you can think of, the analytics performed in Hadoop can be very broad and rich. For example, imagine being able to merge system-of-record (SoR) order details with system-of-engagement (SoE) data that includes free-form text (such as call center notes and description fields), or customer contact emails, web logs, call recordings, client-initiated video uploads for diagnostics, and so on. A customer care worker who has access to all of these assets is in a much better position to deliver effective client service and address the problem at hand—we call this a *360-degree view* of the client. Think about that telco example we referenced earlier. Data scientists want to reach deep into the data in an attempt to discover correlations or insights, build prediction models, and so on. A typical telco stores about nine months of rolling data in their EDW—and it's not at the granularity of the CDR either. It's typically some aggregated higher-level obscure data; this type of zone enables users to explore the raw data in its truest form—and that is why we call it the exploration zone.

Organizations are starting to discover that there is a lot of data that straddles the line between the structured and unstructured worlds. There is a lot of data flowing through organizations that consists of elements that, by themselves, might be considered structured, but are combined in semistructured containers, often in key/value (K/V) pairs in delimited text files, such as JSON, XML, or other similar formats. Some examples include log files,

clickstream sessionization details, machine- and sensor-generated data, third-party data feeds, and so on. And when you have all of this information available, you can start enabling knowledge workers to explore and navigate that data directly in a zone. This is where initial exploration often begins and is typically the best place for LOB analysts to discover what data is actually out there and might be available for further analysis.

You can start to see how a zone architecture that is polyglot based can help boost an organization’s Analytics IQ. Traditional relational reporting sees a structure built to store the data—the reporting is often repeatable. In contrast, in this zone, activity is interactive and exploratory and the data is the structure. In fact, you may not even know what you are looking for; in other words, you don’t even have a hypothesis to test. This is the methodology strongly associated with the RDBMS world. In this zone, data leads the way; you explore it all at will, and hope to discover or identify correlations you may have never even thought of before.

For example, Figure 4-2 shows clickstream data that can be used to perform sessionization analysis to better understand shopping cart abandonment.

You can see how the clickstream data in Figure 4-2 has some structure, yet this kind of data is perfect for Hadoop exploration. This figure shows how the Hadoop MapReduce framework can be used to categorize the web clicks for each visitor and also study those visitors who populated a shopping cart, waited a certain amount of time, and then abandoned it. Note how rich the data is—not only does it include the breadcrumb trail on the vendor’s web

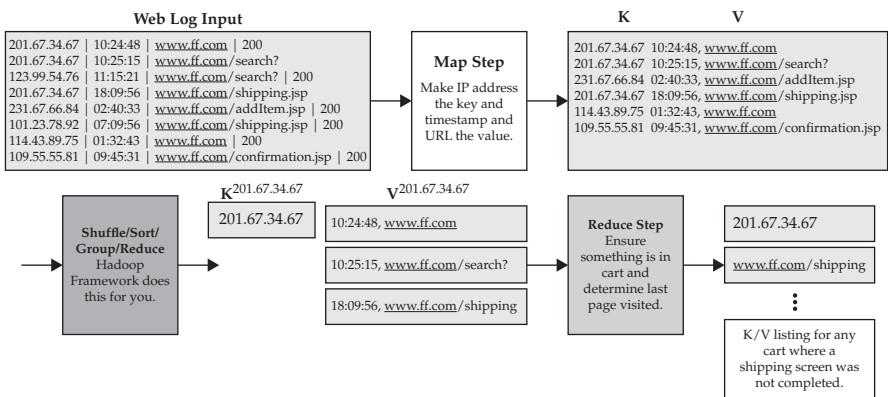


Figure 4-2 A clickstream from an online retailer

site, but it has spatial information (IP addresses) and temporal information (time stamps). Understanding data like this, across millions of users, could enable an organization to zoom in on why potential customers abandon their carts at the shipping calculation phase of their purchase. We've all been there—we thought we got a great online deal only to find that shipping was four times the price. Perhaps this organization could offer extra discount promotions. For example, it could give a point-in-time offer for free shipping if you spend \$100 or more today.

However, the greater value comes when you start combining all of these different types of data and when you start linking your customer's transactions to their path through your web site—this is called *pathing*. Data includes comments from call center notes or emails, insights from linking the daily output of different production facilities to the condition of production equipment, machine sensor data, and environmental data from weather feeds. When you consider that our example shows only a small portion of the input data available for analysis, you will understand where Big Data gets its name. You have *volume*, you have *variety*, and you have *velocity* (clickstreams are typically measured in terabytes per day).

The landing zone should really be the place where you capture and store the raw data coming from various sources. But what do you do with the metadata and other information extracted from your unstructured data? And where do you create these linkages and maintain some type of structure that is required for any real analysis? Guess what? Metadata is cool again! It's how you understand the provenance of your data; it's how you announce data assets to the community (those data sets that are refined, and those left raw for exploration). We cover this in Chapter 13.

A separate set of "exploration" zones is emerging to provide the aforementioned capabilities, enabling different groups of users across the organization to combine data in different ways and to organize the data to fit their specific business needs. It also provides a separate environment to test different hypotheses or discover patterns, correlations, or outliers. Such an environment lets the data lead the way. We call it a *sandbox*.

These sandboxes become virtualized and agile testing and learning environments where you can experiment with data from the landing zone in various combinations and schemas, incorporating additional information that has been generated from the less structured data. These data sets can be

more fluid and temporary in nature. Sandboxes provide a place where various groups of users can easily create and modify data sets that can be removed when they are no longer needed.

We are great proponents of sandboxes because they encourage us to fail, and fail fast. That caught you off-guard, right? Much can be learned from failure, and in yesterday's environments, because it cost so much to fail, we've become afraid of it. Some of the world's greatest accomplishments came after tremendous failures. Thomas Edison developed more than 10,000 prototypes of the lightbulb. James Dyson's iconic vacuum cleaner made it to market after 5,126 prototypes. And Steven Spielberg was rejected by the University of Southern California three times until he changed direction and started to make movies! Hadoop and cloud computing have changed the consequences of failure and afford new opportunities to learn and discover. This is why it's called a *sandbox*—go on and play, and if you find something, explore!

When leveraging Hadoop to host an exploration zone, the zone might be just another set of files that sits on the same cluster as the landing zone, or you can create different data sets for different user communities or LOBs. But you could also establish exploration zones on different clusters to ensure that the activities of one group do not impact those of another. What's more, you could leverage IBM BigInsights for Hadoop's multitenant capabilities that were designed for this very purpose (details in Chapter 6).

Of course, Hadoop *is not* the only place for doing exploration. Although Hadoop can be attractive from a cost and data preparation perspective, it might not provide the required performance characteristics, especially for deeper analytics. In-memory columnar technologies such as IBM's BLU Acceleration (learn about BLU in Chapter 8) are key instruments in any exploration architecture. Discovery via a managed service is key to a NextGen architecture since it provides an agile delivery method with LOB-like consumption—IBM Watson Analytics and dashDB are services we wholly expect to be part of such an architecture. Finally, there is an opportunity to leverage other fit-for-purpose environments and technologies, such as analytic appliances, instead of setting up exploration in the more expensive EDW counterparts.

Into the Deep: The Deep Analytic Zone

The need to go beyond Hadoop is even more important when you get into more complex analytical workloads that require higher performance. The

exploration zone is a great place to do your initial, high-level analysis. But when you want to take things to the next level, developing complex models and performing more advanced analytics, you will want to continue to leverage RDBMS technologies as part of your analytics architecture. That said, you should consider offloading these processes from your EDW to flatten the associated cost curves, optimize performance, and minimize the impact on other enterprise reporting and performance management applications that have been the bread and butter of the warehouse.

Different systems are proving to be more suited for these types of analytic workloads. Analytic appliances, such as the IBM PureData System for Analytics (powered by Netezza technologies), dashDB, and so on were designed from the *ground up* for analytics rather than standard query and reporting needs. These appliances typically provide alternative approaches to indexing data, such as not creating indexes at all! Not only do they store data in a way that is more suited to parallel processing, they deeply embed analytical functions into the core of their processing engines. They also often provide various accelerators, sometimes hardware based, to decompress and process queries in a manner that dramatically improves performance.

These appliances eliminate DBA activities that are typically required to optimize performance and simplify installation and upgrade processes, thereby reducing the overall cost of ownership. The cloud is having an impact here too: Clients want to be able to burst some of their analytical functions into the cloud or host some of this data off premise. IBM's dashDB is all about offering on premise, off premise, or hybrid round-trip analytics functionality for this very purpose. We think IBM is significantly ahead of the marketplace in this critical area. We know of no other vendor that enables you to seamlessly burst into the cloud and leverage round-trip APIs and interfaces for data that resides on premise or off premise.

Curtain Call: The New Staging Zone

Before the advent of NextGen information management architectures, landing and staging areas were more or less identical because they were simply a place to put data that was captured from source systems prior to being transformed or loaded into an EDW. However, with the added capabilities of Hadoop-based systems, much more is possible.

To make data from your different source systems available for reporting and analysis, you need to define a target data model and transform the data into the appropriate structure. In the past, a typical data transformation process involved one of two approaches. The first approach was to extract the data from where it had landed into a separate system to perform the transformation processes and then to deliver the transformed data to the target system, usually a data warehouse or mart. When a large volume of data is involved and many transformations are required, this process can take a long time, create a complex and fragile stack of dependencies, and more. Many companies have become challenged by the long batch processing times that are associated with this approach. In some cases, the available batch windows are exceeded; in other cases, the required transformations are not complete before the next day's processing is set to begin.

The other approach was to move the required data into the target data warehouse or mart and then to leverage the "in-database" processing capabilities to perform the transformations—this is sometimes appealing to clients whose transformation logic is SQL-rich since an RDBMS is very good at parallelizing data. Although this initially helps to speed up some data transformations, it can often increase the complexity and cost. It drives up the data storage and processing requirements of the data warehouse environment and impacts the performance of other critical EDW applications. Because of the drain on capacity, many companies have found themselves once again faced with challenges around processing times. Finally, when you consider IBM's Big SQL stand-alone MPP query engine for Hadoop (Chapter 6), you can see that, for the most part, the future of transformation is not in the EDW; it's too expensive and consumes too many high-investment resources.

Our point is that there are different classes (or tiers) of analytical processing requirements. Tiers that deliver the most performance are likely to cost more than those that can't meet speed-of-thought response times or stringent service level agreements (SLAs). If you are investing in a platinum analytics tier that provides the fastest performance for your organization, then we suggest you keep it focused on those tasks. Having a platinum tier prepare data or having it house data that is seldom queried isn't going to yield the best return on your investment. Keep in mind that the more transformation work you place in the EDW, the slower it will be for those LOB users who think they are leveraging a platinum-rated service.

Organizations are finding that they can include Hadoop in their architecture to assist with data transformation jobs. Instead of having to extract data from where it has landed to perform the transformations, Hadoop enables the data to be transformed where it is. And because they leverage the parallel processing capabilities of Hadoop (more on that in a bit), such transformations can often be performed faster than when they are done within the EDW. This provides companies with the opportunity to reduce their required batch window time frames and offload processing from a platinum-tiered system. Faster processing *and* reduced costs...that's a *potential* win-win scenario!

We want to say a couple of things here with respect to Hadoop as a data preparation platform and why we just said the word potential—you can think of this as an appetizer for Chapter 13. We aren't proposing that you toss out your integration tools. In fact, Gartner has been very public about Hadoop on its own being used as a data integration platform and how such an approach demands custom code, requires specialized skills and effort, and is likely to cost more. And the famous Dr. Ralph Kimball (the godfather of data warehousing) noted in a Cloudera web cast that Hadoop *is not* an ETL tool. Our own personal experience with a pharmaceutical client saw a mere two days of effort to compose a complex transformation with a set of integration services, compared to 30 days for a roll-your-own, hand-coding Hadoop effort—and the two-day effort ran faster too! So, here's the point: Data integration isn't just about the run time; it's about a graphical data flow interface, reusability, maintainability, documentation, metadata, and more. Make no mistake about it, the data integration tool that you select must be able to work with data in Hadoop's distributed file system (HDFS) as though it were any other data source and even be able to leverage its scalable parallel processing framework. But here's the thing: We've seen other parallel frameworks vastly outperform MapReduce for data transformation. And if the transformation is heavy on the SQL side, it actually might make sense to do the transformation in the EDW itself or in a separate, dedicated, parallel-processing data integration engine (not always, but sometimes). The IBM Information Server platform (and its associated as-a-service IBM DataWorks capabilities) is this kind of integration tool. It enables you to *compose and document transformations once* and then run them across a wide range of parallel processing frameworks, choosing the one that makes the most sense—Hadoop, a relational database,

a NoSQL database, or within its own processing framework. Hadoop is indeed a great contributor to data preparation, but it's just part of the solution.

Can you see the flow between zones at this point? Data from source systems finds its way into your organization through a "landing" zone, some of the information is combined in an "exploration" zone (i.e., sandboxes), and the information required for enterprise reporting and performance management is then transformed into the target data model and put into a "staging" zone, from which it can be loaded directly into the data warehouse or mart (or surfaced through Hadoop to the LOB). This staging zone is typically maintained in the same Hadoop system where the data has landed, but it has a structure that matches the target system rather than the source systems, like in the landing zone.

Having a lower-cost environment where data can be effectively processed and put into its target model provides substantial benefits. Not only does it streamline and lower the cost of transforming data, it also opens the door to new analytic possibilities.

You Have Questions? We Have Answers! The Queryable Archive Zone

Hadoop is also a great archiving area. And if all your data is staged in Hadoop as described earlier, you are effectively creating a *day zero archive*. Rolling data out of a traditional EDW requires effort. If all data in the EDW has been initially staged in Hadoop (and you maintain effective metadata), you can simply purge stale data from the EDW.

Pop quiz—what is one of the most popular features being requested for NoSQL data stores such as Hadoop? Full SQL access! To be fair, NoSQL today is really meant to convey the idea of "not only" SQL, but the majority of these systems still rely on low-level query or programming languages. They do not support the significant investments that most organizations have made in standards-based SQL skills and tool sets.

From a Hadoop perspective, organizations initially started leveraging the very basic SQL capabilities that are available through Apache Hive, and this enabled them to start querying Hadoop data by using commercial off-the-shelf (COTS) tools. Most of them found the performance and level of SQL support to be inadequate. An initiative codenamed "Stinger" was intended to make Hive faster and more SQL-complete, but it still has significant limitations.

There is also a movement by multiple vendors to create various SQL processing engines for Hadoop. IBM's offering is called *Big SQL*. We cover this capability in Chapter 6, but will say here that there is no other solution in the Hadoop space whose SQL support is as complete or as compatible with existing, standards-based interfaces. In fact, the IBM Big SQL technology was able to run *all* 22 TPC-H and *all* 99 TPC-DS queries without changes or workarounds. These are open data warehousing benchmarks that various Hadoop vendors are using as proof points for scalability and SQL compatibility. If you're looking into this space, you will find that no other vendor can match what IBM has done. Performance is even significantly better than other SQL solutions for Hadoop. IBM has been in the SQL parallelization and optimization game for almost half a century; in fact, IBM invented SQL!

The queryable archive zone was initially envisioned for the ability to find economies of scale and cost savings by leveraging the same reports and interfaces in Hadoop that you already have in place on the EDW (assuming that the data is in the same structure and data model). The initial interest in a queryable archive undoubtedly was because of the costs associated with the EDW and the sheer volume of data that needed to be maintained.

Before Hadoop, organizations took various steps to reducing these costs, such as archiving older data that is less frequently accessed. Such "cold" data is often put on lower-tier storage or removed from the system. Although such data can usually be restored if it is needed in the future, this typically is not a straightforward process, and it is often the case that organizations won't bother with it unless it becomes absolutely necessary. But with the ability to store data in a much less expensive environment where it can be accessed through standard SQL interfaces, the historical and detail-level data that accounts for the largest data volumes can now be offloaded from the EDW without requiring it to be "restored" whenever someone wants to access it.

Although SQL access to Hadoop plays a role in querying older data, it's also a rich and powerful interface for querying *any* data, in any zone, and that's where we are seeing the most traction. Hadoop gives you a lot of options!

Of course, the performance of queries running in Hadoop isn't likely to be equal to the performance of the same queries running in a relational store; however, for less frequently accessed data, this might be an acceptable compromise, given the cost savings. It all comes down to your SLA.

There are two ways to establish a queryable archive in Hadoop. The first approach is to follow standard archiving procedures, archiving the data into Hadoop instead of some alternative storage medium. Considering all the data storage cost models that are associated with Hadoop, we'll often refer to Hadoop as "the new tape." If you follow this approach, you can leverage your established processes for moving data to lower-cost storage. And because the data can easily be accessed with SQL, you can move data off the EDW much more aggressively. For example, instead of archiving only data that is older than seven years, you can archive data that is older than two years. And maybe you can start moving detail-level data even sooner. The IBM InfoSphere Optim product suite has been enhanced with Hadoop as a key repository for the archiving process, supporting HDFS as a repository for archived files. We discuss archiving, Hadoop, and information lifecycle management in Chapter 13.

The second approach for establishing a queryable archive in Hadoop involves creating the aforementioned *day zero archive*. Consider all the zones we've introduced so far. If you were to transform your overall data architecture to establish landing and staging zones, your staging zone could actually become a queryable archive zone. In other words, you could create your archive on day zero! If you create a load-ready staging zone, where the data structure and model matches, or can easily be mapped to the structure and model of your EDW, you essentially have a copy of most of the data in your EDW already in Hadoop—and in a queryable format. You can maintain that data there instead of archiving data from the EDW. You can then just delete unneeded older data from the EDW. The data will still reside in the Hadoop staging zone, which now is also your queryable archive zone. Even if you choose not to instantiate a mirror data model of the EDW, the data is still captured, and you can use metadata and SQL to retrieve it as needed.

Most organizations do not build their EDW from scratch, nor are they just getting started in this space. We have found that, in most cases, it is a gradual process and culture shift to get to a day zero archive architecture. We recommend starting small and learning "the ropes." Perhaps choose a limited subject area and perform an initial archiving of the data from your existing EDW. Most organizations already have multiple data transformation processes in place and cannot begin using new landing and staging zones, as outlined here, right away. But having a set of integration tools and services that works with HDFS

as well as it works with an RDBMS enables you to change the execution engine for the transformations without rewriting thousands of lines of code. You might prefer to start with the first approach outlined earlier (following standard archiving procedures), but moving toward the second approach (creating a day zero archive) should be part of your longer-term plan.

In Big Data We Trust: The Trusted Data Zone

As you have likely figured out by now, there are many opportunities to offload data and processes from the EDW, and these options can dramatically reduce costs and provide greater agility. An EDW should not be forced to do things that it wasn't designed to do efficiently. We'll go so far as to say that the suggestions we outline in this chapter free the EDW to do what it was really designed to do: provide a "trusted data" zone for your most critical enterprise information.

Although these other "zones" and technologies will help you to reduce costs and provide greater agility to your business, you still need a reliable, robust, well-performing environment where you can establish a harvested, consolidated view of information from across the enterprise. Whether it is to address regulatory compliance reporting, understand and report on the financial performance of your business, or make strategic decisions about where to invest your resources, there are always going to be certain aspects of an organization that rely on absolute truths and that require reliable and quick responses. This is why we believe our zone architecture enables EDWs to return to what they were truly designed for and what they do best.

So, as organizations evolve their architectures and put in place fit-for-purpose zones to address the different functional and nonfunctional requirements of various business needs, it will be important to have a "trusted data" zone. And this zone is best served by the more traditional EDW offerings that have been evolving over the past several decades.

A Zone for Business Reporting

One thing that will remain the same is the need to create different data structures and reports to support the specific needs of different business and functional units across an organization. Although there is value in creating an enterprise view, each unit needs to understand its operations and performance

on its own terms. Companies have historically addressed this by creating different data marts or separate views on the EDW to support these different business-reporting requirements.

This requirement will not go away, and the approach will remain fairly similar. Organizations will continue to maintain “business reporting” zones and will, at least for the immediate future, continue to rely on relational databases. The biggest advancement in this area is likely to be the use of in-memory technologies to improve query performance. Many business users have complained about the time it takes to run reports and get information about their business operations. To address this, some organizations are starting to leverage in-memory technologies as part of their operational systems so that they can report on those systems without impacting core operational processes. Others are starting to leverage in-memory databases for their actual data marts and improving response times, especially when reviewing business performance across multiple dimensions.

This is where the BLU Acceleration technology and the IBM managed dashDB service play such a crucial role in business reporting. They are designed from the ground up to be fast, to deliver incredible compression ratios so that more data can be loaded into memory, to be agile, and to be nearly DBA-free. With BLU Acceleration, you don’t spend time creating performance-tuning objects such as indexes or materialized views; you load the data and go. This keeps things agile and suits LOB reporting just fine.

Finally, for data maintained on premise, the addition of BLU Acceleration *shadow tables* in the DB2 Cancun Release further extends the reporting capabilities on operational data by enabling the creation of in-memory reporting structures over live transactional data without impacting transactional performance. This latest enhancement reinforces the theme of this chapter: There are multiple technologies and techniques to help you deliver an effective analytics architecture to your organization.

From Forecast to Nowcast: The Real-Time Processing and Analytics Zone

The need to process and analyze data in real time is not something new, and products for performing real-time analytics, such as the many complex event processing (CEP) engines that are offered by various vendors, have been available for a while. However, the costs, complexity, and limitations that are

often associated with “real-time” capabilities have often restricted the application of vanilla CEP technologies to the fringe use cases where the effort and investment required could be justified.

Big Data, along with some of its related evolving technologies, is now driving both the demand and the opportunity to extend these capabilities to a much broader set of mainstream use cases. As data volumes explode and organizations start dealing with more semistructured and unstructured data, some organizations are discovering that they can no longer afford to keep and store everything. And some organizations that used to just discard entire data sets are realizing that they might be able to get value from some of the information buried in those data sets. The data might be clickstreams generated from constantly growing web traffic, machine-generated data from device and equipment sensors being driven by the Internet of Things, or even system log files that are typically maintained for short periods of time and then purged.

New capabilities have emerged to address this need, enabling companies to process this data as it is generated and to extract the specific information required without having to copy or land all of the unneeded data or “noise” that might have been included in those feeds. And by incorporating analytic capabilities, organizations can derive true real-time insights that can be acted upon immediately. In addition, because there is so much focus on harvesting analytics from data at rest, people often miss the opportunity to perform additional analytics on the data in motion from what you learned through analytics on the data at rest. Both of these potential game changers underscore the need for a real-time processing and analytics zone.

The real-time processing and analytics zone, unlike some of the other zones, is not a place where data is maintained, but is more of a processing zone where data is analyzed or transformed as it is generated from various sources. Some organizations use this zone to apply transformations to data as it hits their doorsteps to build aggregated data structures on the fly. Whereas the raw data that is generated by those sources is the input, the output can be either a filtered set of that data, which has been extracted, or new data that was generated through analytics: in short, insight. This data can be put into one of the other zones for further analysis and reporting, or it can be acted upon immediately. Actions could be triggered directly from logic in the real-time processing and analytics zone or from a separate decision management application. This powerful zone is powered by IBM’s InfoSphere Streams technology, which we cover in Chapter 7.

Ladies and Gentlemen, Presenting... “The Data Zones Model”

In this chapter, we covered a NextGen information management architecture that we believe sets the bar for a more cost-efficient and impactful analytics infrastructure. We explained the zones, piece by piece, and show you the result in Figure 4-3. A good recipe has multiple components and steps. When these components are brought together in harmony and balance, their confluence is delightful to the palate. We have given you the various ingredients so that you can create your own dish. You might need a little more of one zone and less of another; each use case will be different, and not all of them will be to your taste. Analytics is a dish that can be served cold or hot—depending on your organization’s needs—so begin your journey. There’s no reason to wait.

As you can see, this zone architecture leverages both open source and proprietary technologies. Each has strengths and key benefits that it can deliver to the business as a whole for data-driven decision making. This new approach to managing data is not about rejecting all of the technologies and approaches that have been used in the past. It is more about evolving to use existing capabilities in a more purpose-driven, cost-effective manner, while taking advantage of new capabilities and techniques that make new approaches to analytics possible.

Although these technologies can provide a lot of new value, they can’t possibly address all of your data processing, management, and analytics

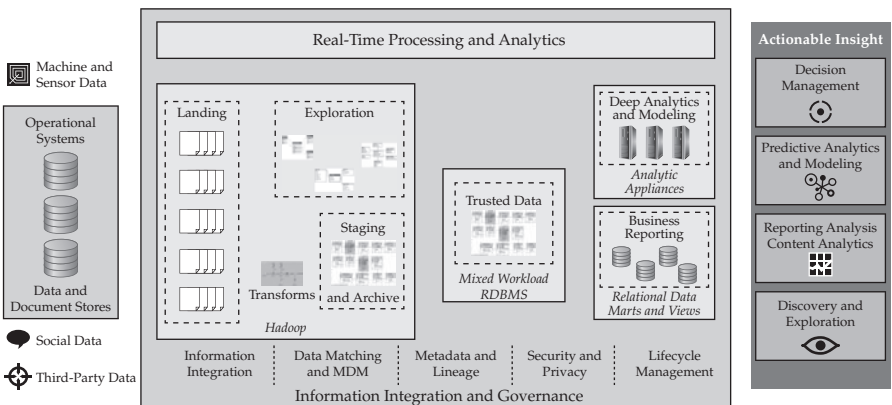


Figure 4-3 A next-generation zones reference architecture for Big Data and Analytics

needs, and a mission to completely replace all of your existing platforms or tools is not a good idea. It is important to emphasize that the evolution of these different zones for processing, managing, and analyzing data makes the need for information integration and governance capabilities even more critical. Some companies thought that the key to establishing trusted information is to put all the data in a single place—the EDW. But recently, organizations have begun to realize not only that data will continue to exist in different pockets across the organization, but that you can actually get more value at a lower cost by leveraging different zones and technologies. However, this makes it even more important to understand where your data is and where it came from. Your tools for integrating and securing data will need to work across multiple platforms and environments—this includes Hadoop, NoSQL, and SQL repositories. And there will be new challenges to match and link related data. This is where the concept of a *data reservoir* comes in.

With the data zones architecture, you have data and workloads assigned to platforms that are appropriate for the tasks at hand. Taken together, you can see all the data zones as the different repositories of your data reservoir. To manage the data in the reservoir (access control, data transformation, audit, and more), IBM provides a suite of information governance and integration tools and services that are designed to work with the data zones we've described in this chapter. These tools and services can be procured via traditional on-premise software—the IBM InfoSphere brand—and through a cloud service via the IBM DataWorks catalog.)

These services set the data reservoir architecture apart from a data lake, which we often hear discussed in Hadoop circles. It's easy to load Hadoop clusters with terabytes of data from many sources, but without proper management, any potential value quickly evaporates—sometimes we call that a data swamp. As we've discussed, Hadoop offers tremendous value as part of the data zone architecture, but by itself it does not have the capabilities needed to address all of your data processing, reporting and analytic needs, let alone ensure proper data governance. Taking the reservoir analogy a little further, IBM's data governance and integration capabilities are being made available as refinery services. In essence, you can filter your data as it pours into and travels through your reservoir through the appropriate refineries. IBM's refinery services are being provided as part of a deliberate strategic shift to ensure this architecture can work in cloud, on premise, or hybrid

cloud environments. The IBM DataWorks refinery services and principles of data governance are the focus of Part III of this book.

In addition, there will be continued focus on streaming capabilities, in-memory analytics, and other related capabilities to improve performance and enable greater handling of real-time data and analytic requirements. And, of course, expect advances in workload management capabilities as companies attempt to bring more types of workloads to lower-cost environments.

If you would like to read Part II - Watson Foundations and Part III - Calming the Waters: Big Data Governance, please click [here](#).

You have the unique opportunity to get a hard copy of the complete version of **Big Data Beyond the Hype. A Guide to Conversations for Today's Data Center.**

Additional Skills Resources

InfoSphere BigInsights for Hadoop Community

Rely on the wide range of IBM experts, programs, and services that are available to help you take your Big Data skills to the next level. Participate with us online in the InfoSphere BigInsights for Hadoop Community. Find whitepapers; videos; demos; BigInsights downloads; links to Twitter, blogs, and Facebook pages; and more.

Visit <https://developer.ibm.com/hadoop/>

Big Data University

Big Data University.com makes Big Data education available to everyone, and starts a journey of discovery to change the world! Big Data technologies, such as Hadoop and Streams, paired with cloud computing, can enable even students to explore data that can lead to important discoveries in the health, environmental, and other industries.

Visit <http://bigdatauniversity.com>

IBM Certification and Mastery Exams

Find industry-leading professional certification and mastery exams. New mastery exams are now available for InfoSphere BigInsights and InfoSphere Streams.

Visit www-03.ibm.com/certify/mastery_tests

Twitter

For the latest news and information as it happens, follow us on Twitter:

[@IBMBigData](#), [@Hadoop_Dev](#), and [@IBMAalytics](#)

developerWorks

On developerWorks, you'll find deep technical articles and tutorials that can help you build your skills to the mastery level. Also find downloads to free and trial versions of software to try today.

Visit www.ibm.com/developerworks/analytics/

Blogs

A team of experts regularly write blogs related to the full spectrum of Big Data topics. Bookmark the “Stream Computing” page and check often to stay on top of industry trends.

Visit www.ibmbigdatahub.com/technology/stream-computing or www.ibmbigdatahub.com/technology/all

This is part of The Big Data & Analytics Hub (ibmbigdatahub.com) that is populated with the content from thought leaders, subject matter experts, and Big Data practitioners (both IBM and third-party thinkers). The Big Data & Analytics Hub is your source for information, content, and conversation regarding Big Data analytics for the enterprise.

IBM Data Magazine

IBM Data magazine delivers substantive, high-quality content on the latest data management developments and IBM advances, as well as creates a strong community of the world’s top information management professionals. It vividly demonstrates how the smart use of data and information advances broad business success, providing the context that enables data management professionals at all levels to make more informed choices and create innovative, synchronized, agile solutions. The magazine’s clear, in-depth technical advice and hands-on examples show readers how to immediately improve productivity and performance. At the same time, expert commentary clearly articulates how advanced technical capabilities benefit the people and processes throughout an organization.

Visit <http://ibmdatamag.com>

IBM Redbooks

IBM Redbooks publications are developed and published by the IBM International Technical Support Organization (ITSO). The ITSO develops and delivers skills, technical know-how, and materials to IBM technical professionals, business partners, clients, and the marketplace in general.

Visit <http://ibm.com/redbooks>