

Microsoft

Developing Big Data Solutions

on Windows Azure



PREVIEW

patterns & practices

PREVIEW VERSION

Preface	4
Who This Guide Is For.....	5
Why This Guide Is Pertinent Now.....	5
A Roadmap for This Guide.....	5
Who's Who	8
Chapter 1: What is Big Data?	9
What Is Big Data?	9
What Is Microsoft HDInsight?	20
Summary.....	27
More Information.....	27
Chapter 2: Implementing Big Data Solutions with HDInsight	28
Overview of the Big Data Process	28
Determining Analytical Goals and Source Data.....	29
Planning the Infrastructure	32
Obtaining and Submitting Source Data	33
Processing the Data.....	39
Evaluating the Results	45
Tuning the Solution	49
Summary.....	49
More Information.....	50
Chapter 3: Using HDInsight for Business Analytics	51
Historical and Predictive Data Analysis	51
Combining HDInsight with Your Business Processes.....	53
HDInsight as a Data Collection, Analysis, and Visualization Tool	54
HDInsight as a Data Transfer, Data Cleansing, and ETL Mechanism	57
HDInsight as a Basic Data Warehouse or Commodity Data Store.....	60
HDInsight Integration with Enterprise Data Warehouses and BI.....	64
Analysis, Visualization, and Reporting Tools	72

Summary.....	77
Scenario 1: Basic Twitter Analysis	79
Introduction to Blue Yonder Airlines.....	79
Analytical Goals and Data Sources	79
HDInsight Infrastructure.....	80
Data Ingestion	83
Data Processing	100
Evaluating Results.....	105
Summary.....	111
Scenario 2 Placeholder	112
Scenario 3 Placeholder	112
Scenario 4 Placeholder	112
Appendix A: An Overview of Enterprise Business Intelligence.....	113

Preface

Do you know what visitors to your website really think about your carefully crafted content? Or, if you run a business, can you tell what your customers actually think about your products or services? Did you realize that your latest promotional campaign had the biggest effect on people aged between 40 and 50 living in Wisconsin (and, more importantly, why)?

Being able to get answers to these kinds of questions is increasingly vital in today's competitive environment, but the source data that can provide these answers is often hidden away; and when you can find it, it's very difficult to analyze successfully. It might be distributed across many different databases or files, be in a format that is hard to process, or may even have been discarded because it didn't seem useful at the time.

To resolve these issues, data analysts and business managers are fast adopting techniques that were commonly at the core of data processing in the past, but have been sidelined in the rush to modern relational database systems and structured data storage. The new buzzword is "Big Data," and it encompasses a range of technologies and techniques that allow you to extract real, useful, and previously hidden information from the often very large quantities of data that previously may have been left dormant and, ultimately, thrown away because storage was too costly.

In the days before Structured Query Language (SQL) and relational databases, data was typically stored in flat files, often in simple text format, with fixed width columns. Application code would open and read one or more files sequentially, or jump to specific locations based on the known line width of a row, to read the text and parse it into columns to extract the values. Results would be written back by creating a new copy of the file, or a separate results file.

Modern relational databases put an end to all this, giving us greater power and additional capabilities for extracting information simply by writing queries in a standard format such as SQL. The database system hides all the complexity of the underlying storage mechanism and the logic for assembling the query that extracts and formats the information. However, as the volume of data that we collect continues to increase, and the native structure of this information is less clearly defined, we are moving beyond the capabilities of even enterprise-level relational database systems.

Big Data encompasses techniques for storing vast quantities of structured, semi-structured, and unstructured data that is distributed across many data stores, and analyzing this data where it's stored instead of moving it all across the network to be processed in one location—as is typically the case with relational databases. The huge volumes of data, often multiple Terabytes or Petabytes, means that distributed processing is far more efficient than streaming all the source data across a network.

Big Data also deals with the issues of data formats by allowing you to store the data in its native form, and then apply a schema to it later, when you need to query it. This means that you don't inadvertently lose any information by forcing the data into a format that may later prove to be too restrictive. It also means that you can simply store the data now—even though you don't know how, when, or even whether it will be useful—safe in the knowledge that, should the need arise in the future, you can extract any useful information it contains.

Microsoft offers a Big Data solution called HDInsight, as both an online service in Windows Azure and as an on-premises mechanism running on Windows Server. This guide primarily focuses on HDInsight for Windows Azure, but explores more general Big Data techniques as well. For example, it provides guidance on configuring your storage clusters, collecting and storing the data, and designing and implementing real-time and batch queries using both the interactive query languages and custom map/reduce components.

Big Data solutions can help you to discover information that you didn't know existed, complement your existing knowledge about your business and your customers, and boost competitiveness. By using the cloud as the data store and Windows Azure HDInsight as the query mechanism you benefit from very affordable storage costs (at the time of writing 1TB of Windows Azure storage costs only \$95 per month), and the flexibility and elasticity of the "pay and go" model where you only pay for the resources you use.

Who This Guide Is For

This guide is aimed at anyone who is already (or is thinking of) collecting large volumes of data, and needs to analyze it to gain a better business or commercial insight into the information hidden inside. This includes business managers, data analysts, system administrators, and developers that need to create queries against massive data repositories.

Why This Guide Is Pertinent Now

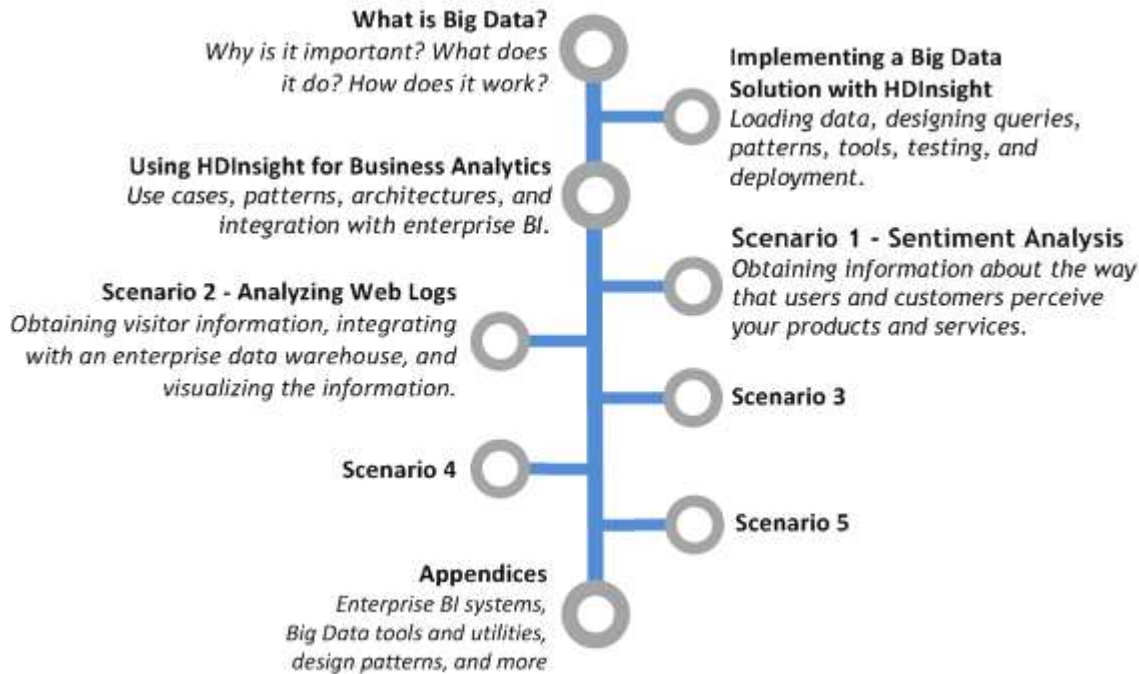
Businesses and organizations are increasingly collecting huge volumes of data that may be useful now or in the future, and they need to know how to store and query it to extract the hidden information it contains. This might be webserver log files, click-through data, financial information, medical data, user feedback, or a range of social sentiment data such as Twitter messages or comments to blog posts.

Big Data techniques and mechanisms such as HDInsight provide a mechanism to efficiently store this data, analyze it to extract useful information, and visualize the information in a range of display applications and tools. It is, realistically, the only way to handle the volume and the inherent semi-structured nature of this data.

No matter what type of service you provide, what industry or market sector you are in, or even if you only run a blog or website forum, you are highly likely to benefit from collecting and analyzing data that is easily available, often collected automatically (such as server log files), or can be obtained from other sources and combined with your own data to help you better understand your customers, your users, and your business; and to help you plan for the future.

A Roadmap for This Guide

This guide contains a range of information to help you understand where and how you might apply Big Data solutions. This is the map of our guide:



Chapter 1, “**What is Big Data?**” provides an overview of the principles and benefits of Big Data, and the differences between it and more traditional database systems, to help you decide where and when you might benefit from applying it. This chapter also discusses Windows Azure HDInsight, its place within Microsoft’s wider data platform, and provides a road map for the remainder of the guide.

Chapter 2, “**Implementing a Big Data Solution with HDInsight,**” explains how you can implement a Big Data solution using Windows Azure HDInsight. It contains general guidance for applying Big Data techniques by exploring in more depth topics such as the technical capabilities, design patterns, query construction, data visualization, and more. You will find much of the information about the design, development, and testing lifecycle of a Big Data solution useful, even if you choose not to use HDInsight as the platform for your own solution.

Chapter 3, “**Using HDInsight for Business Analytics,**” explores how, now that you understand how HDInsight works, you can use it within your own business processes, depending on your data analysis requirements and your existing business intelligence (BI) systems. For example, you might use HDInsight as a query tool and then visualize the results in a program such as Microsoft Excel. Alternatively, if you already have a data warehouse or an enterprise BI mechanism you may choose to use HDInsight as just another data source that integrates at different levels and feeds data into it.





The remaining chapters of the guide concentrate on specific scenarios for applying a Big Data solution, ranging from simple web server log file analysis to analyzing sentiment data such as users’ feedback and comments, and handling streaming data. Each chapter explores specific techniques and solutions relevant to the scenario, and shows an implementation based on Windows Azure HDInsight. The scenarios the guide covers are:

- Chapter 4, “**Scenario 1 - Sentiment Analysis**” covers obtaining information about the way that users and customers perceive your products and services by analyzing “sentiment” data. This is data such as emails, comments, feedback, and social media posts that refer to your products and services, or even to a specific topic or market sector that you are interested in investigating. In the example you will see how you can use Twitter data to analyze sentiment for a fictitious company.
 - Chapter 5, “**Scenario 2 - Analyzing Web Logs**” covers one of the most common scenarios for Big Data solutions: analyzing log files to obtain information about visitors or users and visualizing this information in a range of different ways. This information can help you to better understand trends and traffic patterns, plan for the required capacity now and into the future, and discover which services, products, or areas of a website are underperforming. In this example you will see how you can extract useful data and then integrate it with existing BI systems at different levels and by using different tools.
 - Chapter 6, **Scenario 3**
 - Chapter 7, **Scenario 4**
 - Chapter 8, **Scenario 5**
-

Finally, the appendices contain additional reference material that you may find useful as you explore the tools and utilities that are part of the Big Data frameworks.

Who's Who

This guide explores the use of Big Data solutions using Windows Azure HDInsight. A panel of experts comments on the design and development of the solutions for the scenarios discussed in the guide. The panel includes a business analyst, a data steward, a software developer, and an IT professional. The delivery of the scenario solutions can be considered from each of these points of view. The following table lists these experts.

	<p>Bharath is a business analyst who specializes in Big Data solutions and BI integration. He ensures that the solutions for each of the scenarios will work for the company and provide tangible benefits. He is a cautious person, for good reasons.</p> <p><i>"Implementing Big Data solutions can be a challenge, but the benefits this technology offers can help you to understand more about your company, products, and customers, and help you plan for the future".</i></p>
	<p>Jana is a data steward. She is responsible for the quality of the data held in the organization's databases and data warehouse. She uses her deep knowledge of the corporate data schemas to validate and correct errors in the data, such as removing duplication and enforcing common terminology, to ensure that it provides the correct results.</p> <p><i>"Business intelligence is a core part of all corporations today. So it's vital that they provide accurate, timely, and useful information, which means we must be able to trust the data we use."</i></p>
	<p>Markus is a senior software developer. He is analytical, detail-oriented, and methodical. He's focused on the task at hand, which is building successful business solutions and analytics systems. He knows that he's the person who's ultimately responsible for the code.</p> <p><i>"For the most part, a lot of what we know about software development can be applied to Big Data solutions. But there are always special considerations that are very important."</i></p>
	<p>Poe is an IT professional who's an expert in deploying and running database systems and applications, both in the cloud and in the corporate datacenter. Poe has a keen interest in practical solutions; after all, he's the one who gets paged at 03:00 when there's a problem.</p> <p><i>"I need to make sure our infrastructure, data storage systems, and applications perform well, are reliable, and are secure. This means we need consistent practices for management, monitoring, and back-up across all of our systems."</i></p>

If you have a particular area of interest, look for notes provided by the specialists whose interests align with yours.

Chapter 1: What is Big Data?

This chapter discusses the general concepts of Big Data, and introduces the scenarios you will explore in this guide. It also describes the ways that Big Data, and Microsoft's HDInsight for Windows Azure solution, differ from traditional data management and business intelligence (BI) techniques. The chapter sets the scene for the guide, and provides pointers to the subsequent chapters that describe the implementation of Big Data solutions suited to different market sectors, and to meet a range of data analysis requirements.

What Is Big Data?

The term “Big Data” is being used to encompass an increasing range of technologies and techniques. In essence, Big Data is data that is valuable but traditionally was not practical to store or analyze due to limitations of cost or the absence of suitable mechanisms. It is data, often produced at “fire hose” rate, that you don't know how to analyze at the moment but which may provide valuable in the future.

Big Data solutions aim to provide data storage and querying functionality for situations such as this that are, for a variety of reasons, beyond the capabilities of traditional database systems. They provide a mechanism for organizations to extract meaningful, useful, and often vital information from the vast stores of data that they are collecting.

Big Data is often described as a solution to the “three V's problem”:

- **Variety:** It's not uncommon for 85% of new data to not match any existing data schema. It may also be semi-structured or unstructured data. This means that applying schemas to the data before or during storage is no longer a practical option.
- **Volume:** Big Data solutions typically store and query hundreds of Terabytes of data, and the total volume is probably growing by ten times every five years. Storage must be able to manage this volume, be easily expandable, and work efficiently across distributed systems.
- **Velocity:** Data is being collected from many new types of devices from a growing number of users, and an increasing number of devices and applications per user. The design and implementation of storage must happen quickly and efficiently.

The quintessential aspect of Big Data is not the data itself; it's the ability to discover useful information hidden in the data. It's really all about the analytics that a Big Data solution can empower.

Many people consider Big Data to be a new way to do data warehousing when the volume of data exceeds the capacity or cost limitations for relational database systems. However, it can be difficult to fully grasp what Big Data really involves, what hardware and software it uses, and how and when it is useful. There are some basic questions, the answers to which will help you understand where Big Data solutions are useful and how you approach the topic in terms of implementing your own solutions. We'll work through these questions now.

Why Do I Need a Big Data Solution?

In the most simplistic terms, organizations need a Big Data solution to enable them to survive in a rapidly expanding and increasingly competitive market. For example, they need to know how their products and services are perceived in the market, what customers think of the organization, whether advertising campaigns are working, and which facets of the organization are (or are not) achieving their aims.

Organizations typically collect data that is useful for generating BI reports, and to provide input for management decisions. However, organizations are increasingly implementing mechanisms that collect other types of data such as “sentiment data” (emails, comments from web site feedback mechanisms, and tweets that are related to the organization's products and services), click-through data, information from sensors in users' devices such as location data, and website log files.

These vast repositories of data contain useful, and even vital, information that can be used for product and service planning, coordinating advertising campaigns, improving customer service, or as an input to reporting systems. This information is also very useful for predictive analysis such as estimating future profitability in a financial scenario, or for an insurance company to predict the possibility of accidents and claims. Big Data solutions allow you to store and extract all this information, even if you don't know when or how you will use the data at the time you are collecting it.

In addition, many organizations need to handle vast quantities of data as part of their daily operations. Examples include financial and reporting data, and medical data such as images and patients' notes. Processing, backing up, and querying all of this data becomes more complex and time consuming as the volume increases. Big Data solutions are designed to store vast quantities of data on distributed servers with automatic generation of replicas to guard against data loss, together with mechanisms for performing queries on the data to extract the information the organization requires.

What Problems Does Big Data Solve?

Primarily, the limitation with traditional database systems is due to the volume of data; Big Data solutions often work against data stores containing many hundreds of Terabytes or even Petabytes of data. The querying capabilities of existing relational database systems, data repositories, and data warehouse mechanisms often cannot manage these kinds of volumes of data, and cannot perform queries within a realistic timescale.

This isn't to say that relational databases have had their day. Continual development of the hardware and software for this core business function provides capabilities for storing very large amounts of data, replicating data across data stores, data warehousing and complex analysis, and many tools for visualizing the data to generate BI and comprehensive reports. However, Big Data solutions are optimized for the storage of huge volumes of data in a way that can dramatically reduce storage cost, while still being able to generate BI and comprehensive reports.



Big Data solutions typically target scenarios where there is a huge volume of unstructured or semi-structured data that must be stored and queried to extract business intelligence. Typically, 85% of data currently stored in Big Data solutions is unstructured or semi-structured.

Secondly, the data organizations collect is often not in a structured form that suits relational database systems. Some data, such as web server logs and responses to questionnaires may be preprocessed into the traditional row and column format. However, data such as emails, web site comments and feedback, and tweets are semi-structured or even unstructured data.

Deciding how to store this data using traditional database systems is problematic, and may result in loss of useful information if the data must be squeezed into a specific schema when it is stored.

How Is Big Data Different from Traditional Database Systems?

Traditional database systems typically use a relational model where all the data is stored using predetermined schemas, and linked using the values in specific columns of each table. There are some more flexible mechanisms, such as the ability to store XML documents and binary data, but the capabilities for handling these types of data are usually quite limited.

Big Data solutions do not force a schema onto the stored data. Instead, you can store almost any type of structured, semi-structured, or unstructured data and then apply a suitable schema only when you query this data.

Big Data solutions are optimized for storing vast quantities of data using simple file formats and highly distributed storage mechanisms. Each distributed node is also capable of executing parts of the queries that extract information. Whereas a traditional database system would need to collect all the data and move it to a central location for processing, with the consequent limitations of processing capacity and network latency, Big Data solutions perform the initial processing of the data at each storage node. This means that the bulk of the data does not need to be moved over the network, and it requires only a fraction of the central processing capacity to execute a query.



Relational databases systems require a schema to be applied when data is written, which may mean that some information hidden in the data is lost. Big Data solutions store the data in its raw format and apply a schema only when the data is read, which preserves all of the information within the data.

Figure 1 shows some of the basic differences between a relational database system and a Big Data solution in terms of storing and querying data. Notice how both relational databases and Big Data solutions use a cluster of servers; the main difference is where query processing takes place and how the data is moved across the network. In a Big Data solution only the results of the distributed query processing are passed across the cluster network to the node that will assemble them into a final results set.

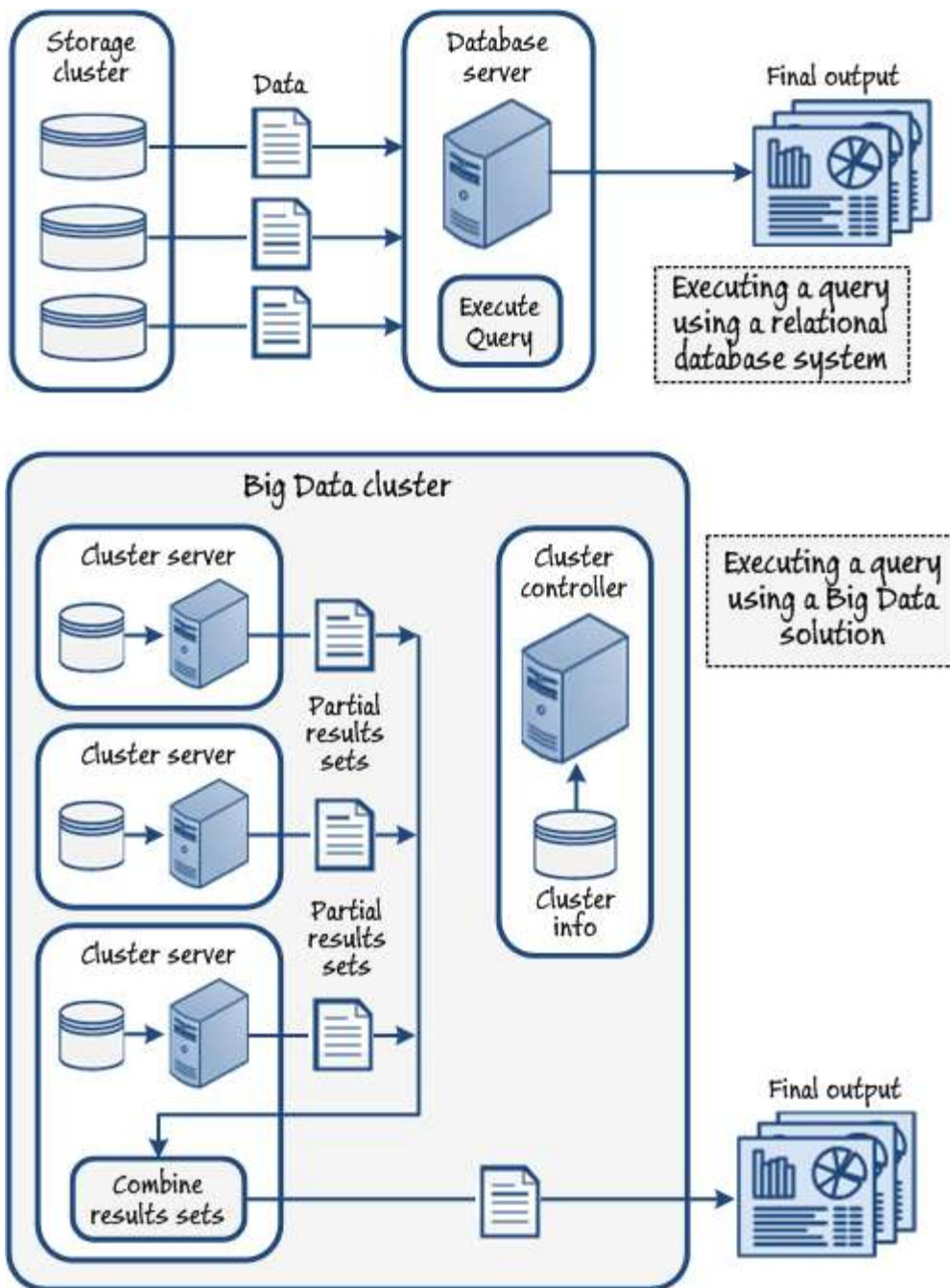


Figure 1
Some differences between relational databases and Big Data solutions

The servers in a cluster are typically co-located in the same datacenter and connected over a low-latency, high-bandwidth network. However, Big Data solutions can work well even without a high capacity network, and the servers can be more widely distributed, because the volume of data moved over the network is so much less than in a traditional relational database cluster.

The ability to work with highly distributed data and simple file formats also opens up opportunities for more efficient and more comprehensive data collection. For example, services and applications can store data in any of the predefined distributed locations without needing to preprocess it or execute queries that can absorb processing capacity. Data is simply appended to the files in the data store. Any processing required on the data is done when it is queried, without affecting the original data and risking losing valuable information.

The following table summarizes the major differences between a Big Data solution and existing relational database systems.

Feature	Relational database systems	Big Data solutions
Data types and formats	Structured	Semi-structured and unstructured
Data integrity	High - transactional updates	Low - eventually consistent
Schema	Static - required on write	Dynamic - optional on read and write
Read and write pattern	Fully repeatable read/write	Write once, repeatable read
Storage volume	Gigabytes to terabytes	Terabytes, petabytes, and beyond
Scalability	Scale up with more powerful hardware	Scale out with additional servers
Data processing distribution	Limited or none	Distributed across cluster
Economics	Expensive hardware and software	Commodity hardware and open source software

What Technologies Does Big Data Encompass?

Having talked in general terms about Big Data, it's time to be more specific about the hardware and software that it encompasses. The core of Big Data is an open source technology named Hadoop. Hadoop was developed by a team at the Apache Software Foundation. It is commonly understood to contain three main assets:

- The Hadoop kernel library that contains the common routines and utilities.
- The distributed file system for storing data files and managing clusters; usually referred to as HDFS™ (highly distributed filing system).
- The map/reduce library that provides the framework to run queries against the data.

In addition there are many other open source components and tools that can be used with Hadoop. The Apache Hadoop website lists the following:

- Avro™: A data serialization system.
- Cassandra™: A scalable multi-master database with no single points of failure.
- Chukwa™: A data collection system for managing large distributed systems.
- HBase™: A scalable, distributed database that supports structured data storage for large tables.
- HCatalog™: A table and storage management service for data created using Apache Hadoop.

- Hive™: A data warehouse infrastructure that provides data summarization and ad hoc querying.
 - Mahout™: A Scalable machine learning and data mining library.
 - Pig™: A high-level data-flow language and execution framework for parallel computation.
 - ZooKeeper™: A high-performance coordination service for distributed applications.
-

We'll be exploring and using some of these components in the scenarios described in subsequent chapters of this guide. For more information about all of them, see the [Apache Hadoop website](#).

However, this guide concentrates on Microsoft's implements a Big Data solution, named **HDInsight**, and specifically on the cloud based implementation of HDInsight in Windows Azure. You'll find a more detailed description of HDInsight later in this chapter. You will also see some general discussion of Big Data and Hadoop technologies and techniques throughout the rest of this guide.

How Does Big Data Work?

Big Data solutions are essentially a simple process based on storing data as multiple replicated disk files on a distributed set of servers, and executing a multistep query that runs on each server in the cluster to extract the results from each one and then combine these into the final results set.

The Data Cluster

Big Data solutions use a cluster of servers to store and process the data. Each member server in the cluster is called a **data node**, and contains a data store and a query execution engine. The cluster is managed by a server called the **name node**, which has knowledge of all the cluster servers and the files stored on each one. The name node server does not store any data, but is responsible for allocating data to the other cluster members and keeping track of the state of each one by listening for heartbeat messages.

To store incoming data, the name node server directs the client to the appropriate data node server. The name node also manages replication of data files across all the other cluster members, which communicate with each other to replicate the data. The data is divided into 64MB chunks and three copies of each data file are stored across the cluster servers in order to provide resilience against failure and data loss (the chunk size and the number of replicated copies are configurable for the cluster).



It's vital to maintain a backup of the name node server in a cluster. Without it, all of the cluster data may be lost.

In some implementations there is only a single name node server, which results in a single point of failure for the entire cluster. If the name node server fails, all knowledge of the location of the cluster servers and data may be lost. To avoid this, Big Data implementations usually contain a replication or backup feature for the name node server (often referred to as the **secondary name node**), which allows the cluster to survive a failure.

The Data Store

The data store running on each server in a cluster is a suitable distributed storage service such as HDFS or a compatible equivalent (Windows Azure HDInsight provides a compatible service over Windows Azure blob storage). A core principal of Big Data implementations is that data is co-located on the same server as the query execution engine that will process it in order to minimize bandwidth use and maximize query performance.

Windows Azure HDInsight (and some other virtualized environments) does not completely follow this principle because the data is stored in the datacenter storage cluster that is co-located with virtualized servers.

The data store in a Big Data implementation is usually referred to as a NoSQL store, although this is not technically accurate because some implementations do support a SQL-like query language. There are more than 120 NoSQL data store implementations available at the time of writing, but they can be divided into the following four basic categories:

- **Key/value stores.** These are data stores that hold data as a series of key/value pairs. The value may be a single data item or a complex data structure. There is no fixed schema for the data, and so these types of data store are ideal for unstructured data. An example of a key/value store is Windows Azure table storage, where each row has a key and a property bag containing one or more values. Key/value stores can be persistent or volatile.
- **Document stores.** These are data stores optimized to store structured, semi-structured, and unstructured data items such as JSON objects, XML documents, and binary data. An example of a document store is Windows Azure blob storage, where each item is identified by a blob name within a virtual folder structure.
- **Wide column or Column Family data stores.** These are data stores that do use a schema, but the schema can contain families of columns rather than actual columns. They are ideally suited to storing semi-structured data where some columns can be predefined but others are capable of storing differing elements of unstructured data. HBase running on HDFS is an example.
- **Graph data stores.** These are data stores that hold the relationships between objects. They are less common than the other types of data store, and tend to have only specialist uses.

NoSQL storage is typically much cheaper than relational storage, and usually supports a Write Once capability that allows only for data to be appended. To update data in these stores you must drop and recreate the relevant file. This limitation maximizes performance; Big Data storage implementations are usually measured by throughput rather than capacity because this is usually the most significant factor for both storage and query efficiency.

Modern data handling techniques, such as the Command Query Responsibility Separation (CQRS) and other patterns, do not encourage updates to data. Instead, new data is added and milestone records are used to fix the current state of the data at intervals. This approach provides better performance and maintains the history of changes to the data. For more information about CQRS see the patterns & practices guide [CQRS Journey](#).

The Query Mechanism

Big Data queries are based on a distributed processing mechanism called *MapReduce* that provides optimum performance across the servers in a cluster.

For a detailed description of the MapReduce framework and programming model, see MapReduce.org and Wikipedia.

The query uses two components, written in Java, which implement algorithms that perform a two-stage data extraction and rollup process. The **Map** component runs on each data node server in the cluster extracting data that matches the query, and optionally applying some processing or transformation to the data to acquire the required result set from the files on that server.

The **Reduce** component runs on one of the data node servers, and combines the results from all of the Map components into the final results set. Figure 2 shows a high-level overview of an example of this process that sums the values of the properties of each data item that has a key value of "A".

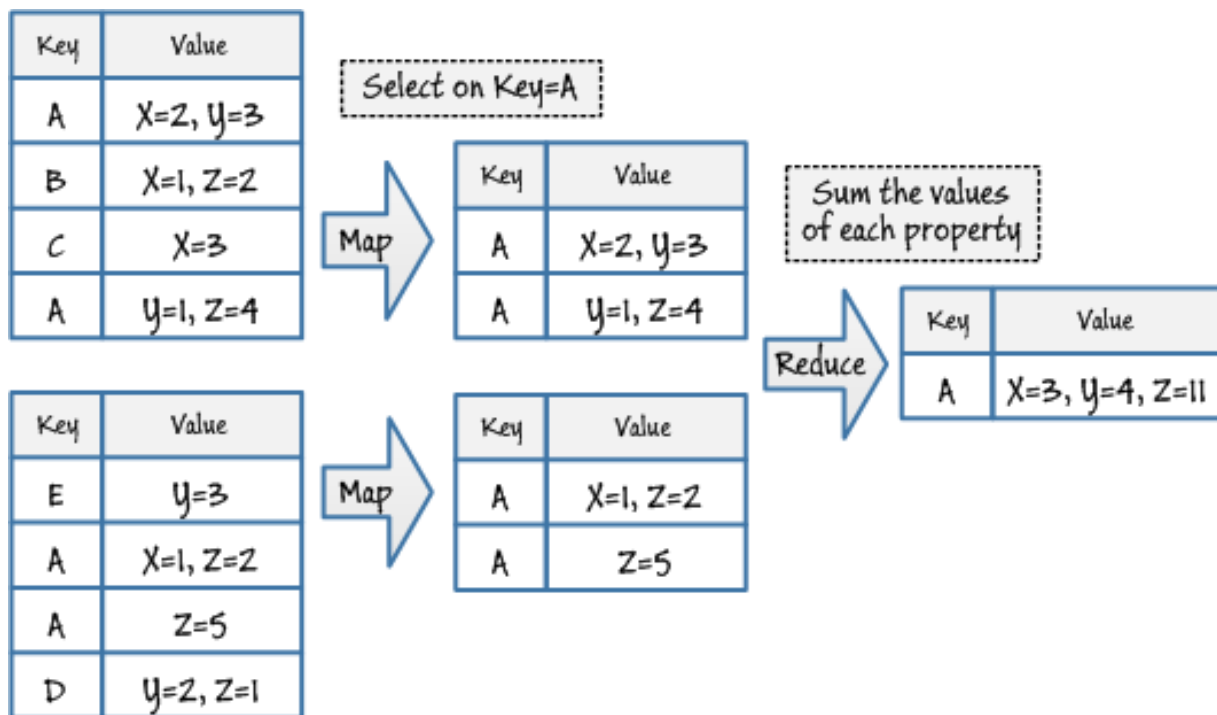


Figure 2

A high level view of the Big Data process for storing data and extracting information

Depending on the configuration of the query job, there may be more than one Reduce task running. The output from each Map task is stored in a common buffer, sorted, and then passed to one or more Reduce tasks. Intermediate results are stored in the buffer until the final Reduce task combines them all.

Although the core Hadoop engine requires Map and Reduce components it executes to be written in Java, you can use other techniques to create them in the background without writing Java code. For example you can use the tools named Hive and Pig that are included in most Big Data frameworks to write queries in a SQL-like or a

high-level language. You can also use the Hadoop streaming API to execute components written in other languages—see [Hadoop Streaming](#) on the Apache website for more details.

Chapter 3 of this guide contains more details of the way that Big Data solutions work, and guidance on using them. For example, it discusses topics such as choosing a data store and configuring a cluster, loading the data, the common patterns for querying data, and optimizing performance. It also describes the end-to-end process for using Big Data frameworks, and patterns such as ETL (extract-transform-load) for generating BI information.

What Are the Limitations of Big Data?

Modern relational databases are typically optimized for fast and efficient query processing using techniques such as Structured Query Language (SQL). Big Data solutions are optimized for reliable storage of vast quantities of data; the often unstructured nature of the data, the lack of predefined schemas, and the distributed nature of the storage usually preclude any optimization for query performance. Unlike SQL queries, which can use indexes and other intelligent optimization techniques to maximize query performance, Big Data queries typically require an operation similar to a table scan.



Big Data queries are batch operations that may take some time to execute. It's possible to perform real-time queries, but typically you will run the query and store the results for use within your existing BI tools and analytics systems.

Therefore, Big Data queries are typically batch operations that, depending on the data volume and query complexity, may take considerable time to return a final result. However, when you consider the volumes of data that Big Data solutions can handle, which are well beyond the capabilities of traditional data storage systems, the fact that queries run as multiple tasks on distributed servers does offer a level of performance that cannot be achieved by other methods. Unlike most SQL queries used with relational databases, Big Data queries are typically not executed repeatedly as part of an application's execution and so batch operation is not a major disadvantage.

Will Big Data Replace My Relational Database?

The relational database systems we use today and the new Big Data solutions are complementary mechanisms; Big Data is unlikely to replace the existing relational database. In fact, in the majority of cases, it complements and augments the capabilities for managing data and generating BI.

For example, it's common to use a Big Data query to generate a result set that is then stored in a relational database for use in the generation of BI, or as input to another process. Big Data is also a valuable tool when you need to handle data arriving very quickly, and which you can process later. You can dump the data into the Big Data storage cluster in its original format, and then process it on demand using a Big Data query that extracts the required result set and stores it in a relational database or makes it available for reporting. Figure 3 shows this approach in schematic form.

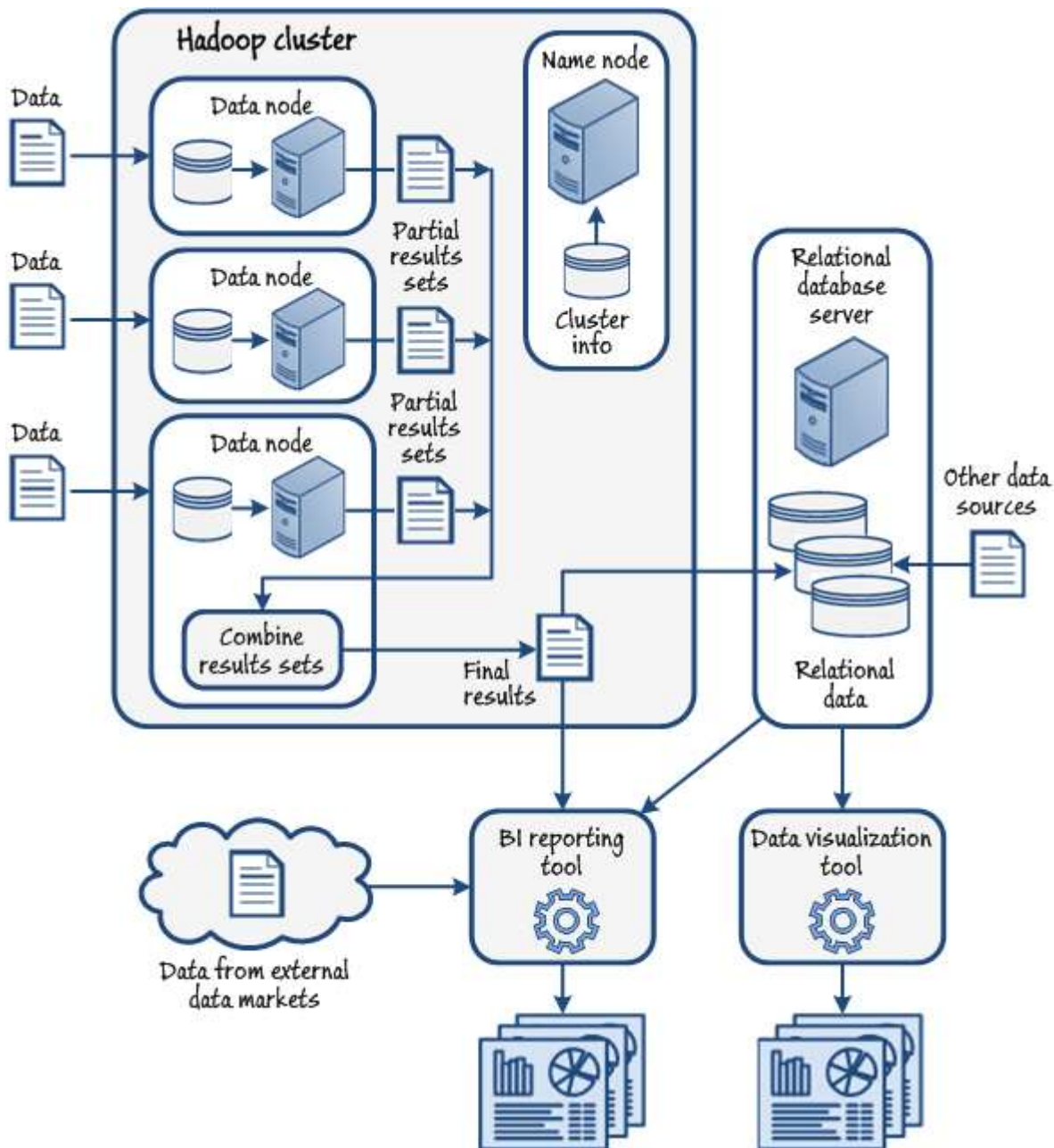


Figure 3
Combining Big Data with a relational database

Chapters 3 and 5 of this guide explore in more depth and demonstrate the integration of a Big Data solution with existing data analysis tools and enterprise BI systems. It describes the three main stages of integration, and explains the benefits of each approach in terms of maximizing the usefulness of your Big Data implementations.

Is Big Data the Right Solution for Me?

The first step in evaluating and implementing any business policy, whether it's related to computer hardware, software, replacement office furniture, or the contract for cleaning the windows, is to determine the results that you hope to achieve. Adopting a Big Data solution is no different.

The result you want from a Big Data solution will typically be better information that helps you to make data-driven decisions about the organization. However, to be able to get this information, you must evaluate several factors such as:

- **Where will the source data come from?** Perhaps you already have the data that contains the information you need, but you can't analyze it with your existing tools. Or is there a source of data you think will be useful, but you don't yet know how to collect it, store it, and analyze it?
- **What is the format of the data?** Is it highly structured, in which case you may be able to load it into your existing database or data warehouse and process it there? Or is it semi-structured or unstructured, in which case a Big Data solution that is optimized for textual discovery, categorization, and predictive analysis will be more suitable?
- **What are the delivery and quality characteristics of the data?** Is there a huge volume? Does it arrive as a stream or in batches? Is it of high quality or will you need to perform some type of data cleansing and validation of the content?
- **Do you want to combine the results with data from other sources?** If so, do you know where this data will come from, how much it will cost if you have to purchase it, and how reliable this data is?
- **Do you want to integrate with an existing BI system?** Will you need to load the data into an existing database or data warehouse, or will you just analyze it and visualize the results separately?

The answers to these questions will help you decide whether a Big Data solution is appropriate. As you saw earlier in this chapter, Big Data solutions are primarily suited to situations where:

- You have large volumes of data to store and process.
- The data is in a semi-structured or unstructured format, often as text files.
- The data is not well categorized; for example, similar items are described using different terminology such as a variation in country or region names, and there is no obvious key value.
- The data arrives rapidly as a stream, or in large batches that cannot be processed in real time.
- The data contains a lot of redundancy or duplication.
- The data cannot easily be processed into a format that suits existing database schemas without risking loss of information.
- You need to execute complex batch jobs on a very large scale, so that running the jobs in parallel is necessary.
- You want to be able to easily scale the system up or down on demand.

- You don't actually know how the data might be useful but you suspect that it might, be either now or in the future.

If you decide that you do need a Big Data solution, the next step is to evaluate and choose a platform. There are several you can choose from, some of which are delivered as cloud services and some that you run on your own on-premises or hosted hardware. In this guide you will see solutions implemented using Microsoft's cloud hosted solution, Windows Azure HDInsight.



Choosing a cloud-hosted Big Data platform makes it easy to scale out or scale in your solution without incurring the cost of new hardware or have existing hardware underused.

What Is Microsoft HDInsight?

Microsoft implements Big Data solutions through two implementations based on Hadoop, and called HDInsight. HDInsight is 100% Apache Hadoop compatible and is built on open source components in conjunction with HortonWorks, and is compatible with open source community distributions. All of the components are tested in typical scenarios to ensure that they work together correctly, and there are no versioning or compatibility issues. Developments in HDInsight are fed back into community through HortonWorks to maintain compatibility and to support the open source effort.

HDInsight is available in two forms:

- **Windows Azure HDInsight.** This is a service available to Windows Azure subscribers that uses Windows Azure clusters, integrates with Windows Azure storage. An ODBC connector is available to connect the output from HDInsight queries to data analysis tools.
- **Microsoft HDInsight Server.** This is a framework that you can install on Windows Server 2012 with Hyper-V. It integrates with Microsoft System Center 2012 to provide a familiar management and monitoring environment by using the Hadoop management pack.

For more information, see [Microsoft Big Data](#). To sign up for the Windows Azure service, go to [Windows Azure HDInsight](#). You can download HDInsight Server from the [Microsoft Download Center](#).

Figure 4 shows the main component parts of HDInsight, which can run on Windows Server 2012, in a Virtual Machine, or in Windows Azure.

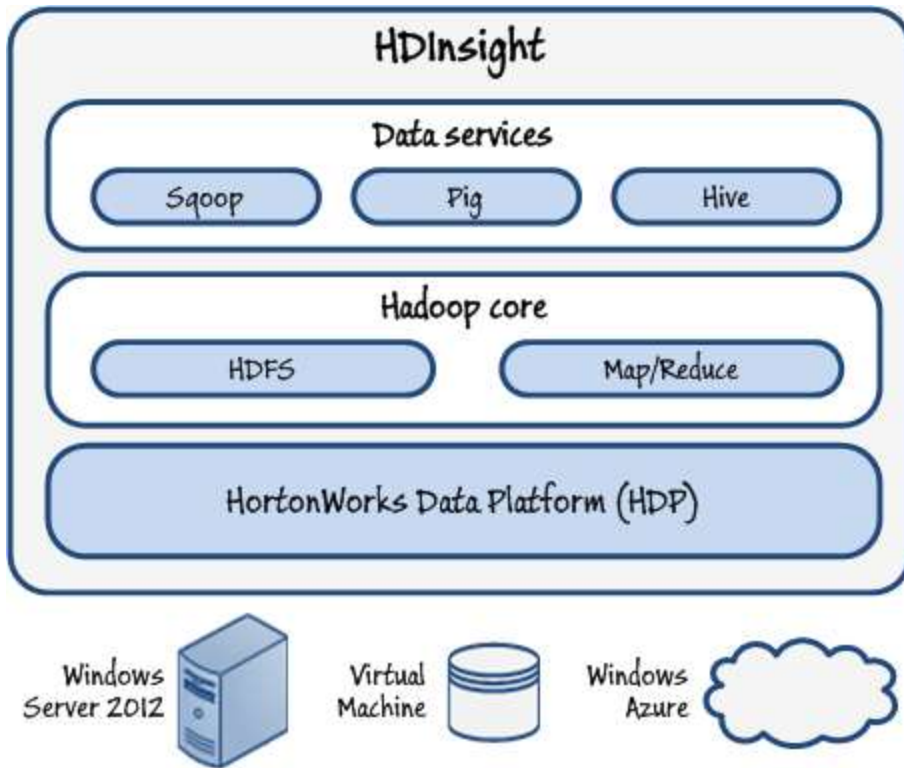


Figure 4
The main component parts of Microsoft HDInsight

HDInsight supports queries and Map Reduce components written in C# and JavaScript; and also includes a dashboard for monitoring clusters and creating alerts, the Talend Open Studio integration tool for connecting Hadoop to many other data systems, and the Apache HCatalog tool for sharing data between Hadoop and other systems.



HDInsight solutions can run in Windows Azure, on Windows Server 2012, and in a virtual machine located on-premises or in the cloud.

There is also a locally installable package available that you can use for developing, debugging, and testing your Big Data solutions. It runs on a single computer in Visual Studio, and provides multiple virtual data nodes. You can download the [Microsoft .NET SDK For Hadoop](#) from Codeplex or install it with NuGet. At the time of writing this contained a map/reduce implementation, Linq To Hive, and the WebHDFS Client.

The Data Store and Execution Platform

Big Data solutions store data as a series of files located within a familiar folder structure on disk. In Windows Azure HDInsight these files are stored in Windows Azure blob storage, on a cluster that you configure using the HDInsight portal page. In HDInsight Server the files are stored on disk in the usual way.

The execution platform is common across both Windows Azure HDInsight and HDInsight Server, and so working with it is a similar experience. The Windows Azure HDInsight portal and the HDInsight Service desktop provide an interactive environment in which you run the tools such as Hive and Pig.

You can use the Sqoop connector to import data from a SQL Server database or a Windows Azure SQL Database database into your cluster, or you can upload it using the JavaScript interactive console. It is also possible to perform queries against data held in Windows Azure blob storage when using Windows Azure HDInsight.

Chapter 2 of this guide provides more details of how you configure a cluster and upload data.

The Query and Analysis Tools

HDInsight contains implementations of the Hive and Pig tools. Hive allows you to overlay a schema onto the data when you need to run a query, and use a SQL-like language for these queries. For example, you can use the **CREATE TABLE** command to build a table by splitting the text strings in the data store using delimiters or at specific character locations, and then execute **SELECT** statements to extract the required data.

Pig allows you to create schemas and execute queries using a high level language usually referred to as Pig Latin. However, in HDInsight you can write Pig queries using the interactive console, and use languages such as JavaScript instead. It supports a fluent interface, which means that you can write queries such as **from("input.txt", "date, product, qty, price").select("product, price").run()** .

You can, of course, use other open source tools such as Mahout with HDInsight, even though they are not included as part of an HDInsight deployment. Mahout allows you to perform data mining queries that examine data files to extract specific types of information. For example, it supports recommendation mining (finding user's preferences from their behavior), clustering (grouping documents with similar topic content), and classification (assigning new documents to a category based on existing categorization).

HDInsight also supports writing queries that directly execute the Hadoop Map and Reduce components. However, unlike some other Big Data implementations, HDInsight allows you to write these components in C#, F#, and JavaScript. The tools and utilities convert the query into suitable Map and Reduce components using Java code, and generate a suitable command that is executed by Hadoop.

For more information about using HDInsight, a good place to start is the Microsoft TechNet library. You can see a list of articles related to HDInsight by searching the library using this URL:

<http://social.technet.microsoft.com/Search/en-US?query=hadoop>.

There are also several blogs that cover HDInsight, such as

<http://prologika.com/CS/blogs/blog/archive/tags/Hadoop/default.aspx>. In addition, there is a support forum for HDInsight at <http://social.msdn.microsoft.com/Forums/en-US/hdinsight/threads>.

The remaining chapters of this guide provide more details of how you perform queries and data analysis.

HDInsight and the Microsoft Data Platform

It may appear from what you've seen so far that HDInsight is a separate, stand-alone system for storing and analyzing large volumes of data. However, this is definitely not the case—it is a significant component of the Microsoft data platform; and part of the overall data acquisition, management, and visualization strategy.

Figure 5 shows an overview of the Microsoft data platform, and the role HDInsight plays within this. This figure does not include all of Microsoft's data-related products, and it doesn't attempt to show physical data flows. Instead, it illustrates the applications, services, tools, and frameworks that work together allow you to capture data, store it, and visualize the information it contains.

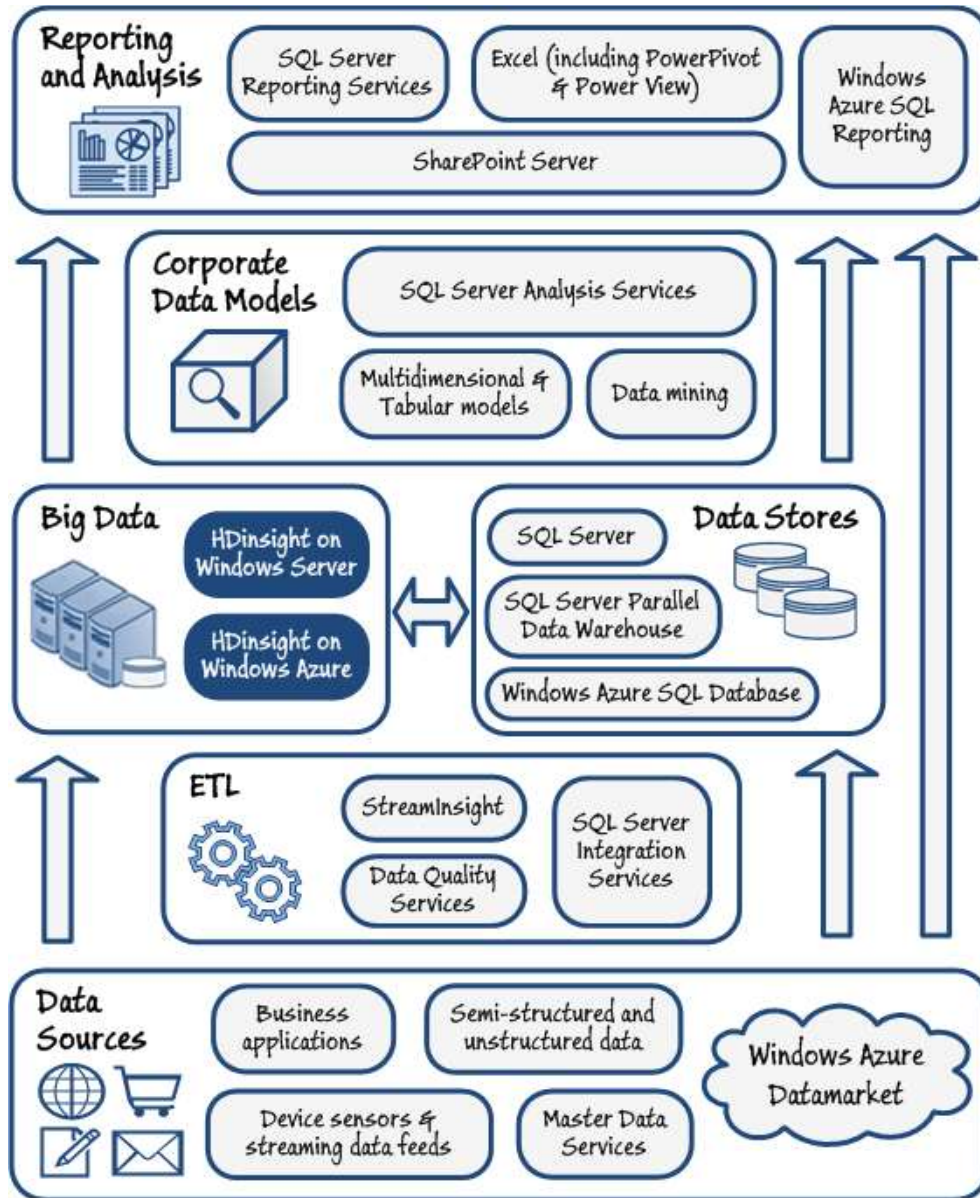


Figure 5
The Role of HDInsight as part of the Microsoft Data Platform

In Chapter 3 you will learn about the technical implementation of a Big Data solution using HDInsight, including the options for storing the data, the tools for performing queries, and the technicalities of connecting the data to BI systems and visualization tools.

As an example, consider the case where you want to simply load some unstructured data into Windows Azure HDInsight, combine it with data from external sources such as Windows Data Market, and then analyze and visualize the results using Microsoft Excel and Power View. Figure 6 shows this approach, and how it maps to elements of the Microsoft data platform.

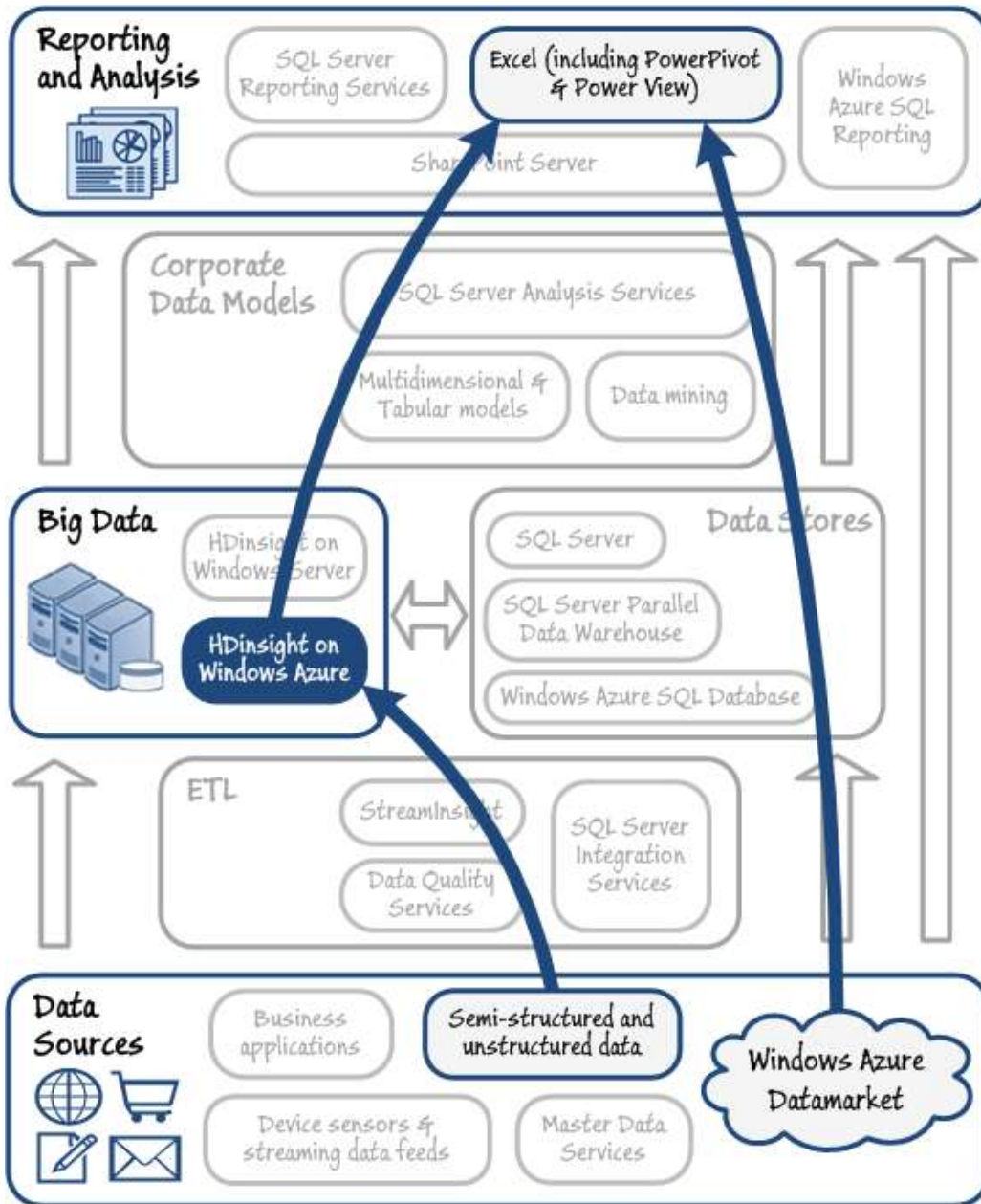


Figure 6
Simple data querying, analysis, and visualization using Windows Azure HDInsight

However, you may decide that you want to implement deeper integration of the data you store and query in HDInsight with your enterprise BI system. Figure 7 shows an example where streaming data collected from device sensors is fed through StreamInsight for categorization and filtering, and then dumped into a Windows Azure HDInsight cluster. The output from queries that are runs as a periodic batch jobs in HDInsight is integrated at the corporate model level with a data warehouse, and ultimately delivered to users through SharePoint libraries and web parts, and is available for use in reports and data analysis and visualization tools such as Microsoft Excel.

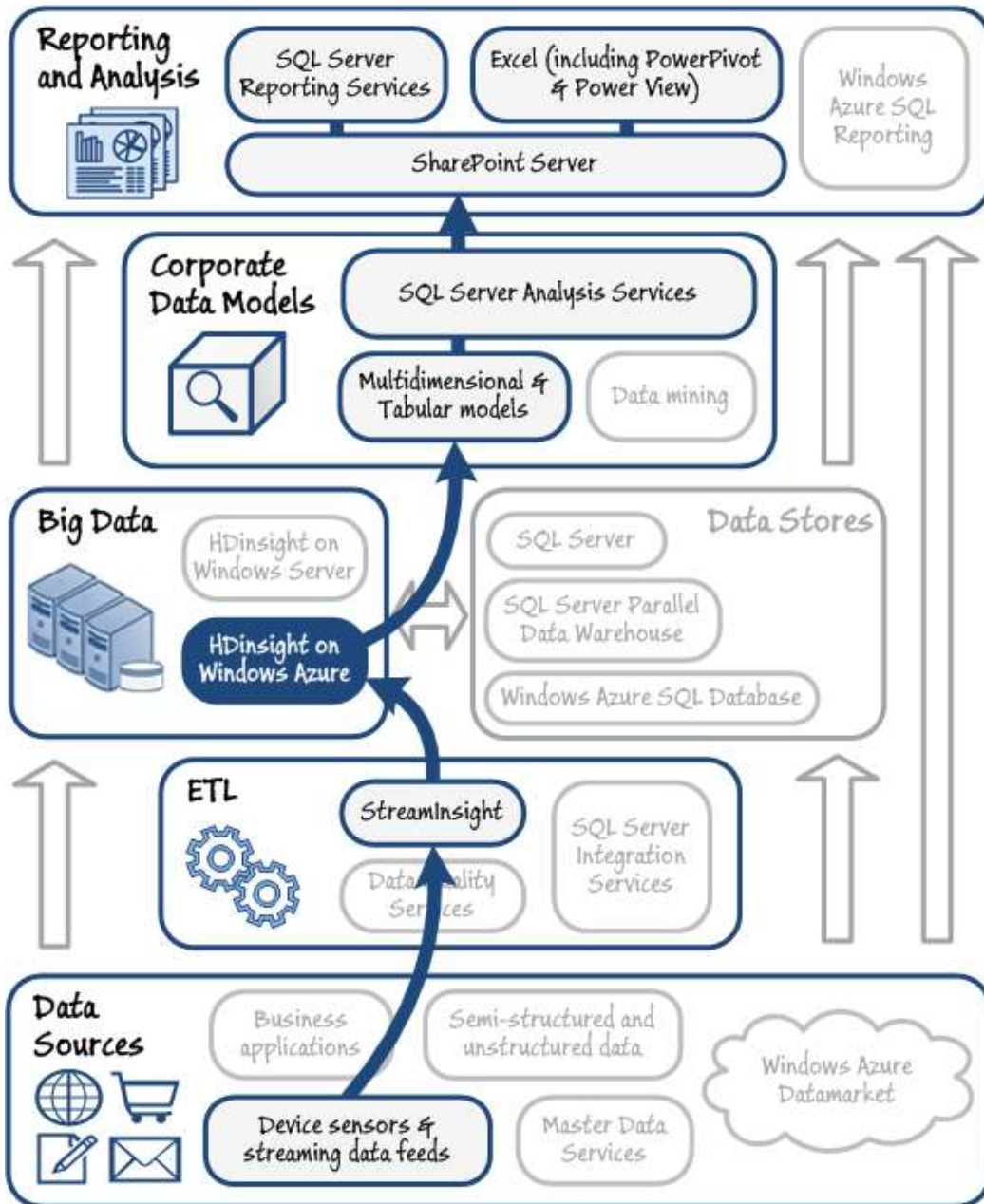


Figure 7
Integrating HDInsight with corporate models in an enterprise BI system

Alternatively, you may want to integrate HDInsight as a business data source for an existing data warehouse system, perhaps to produce a set of specific management reports on a regular basis. Figure 8 shows how you can take semi-structured or unstructured data and query it with HDInsight, validate and cleanse the results using Data Quality Services, and store them in your data warehouse tables ready for use in reports.

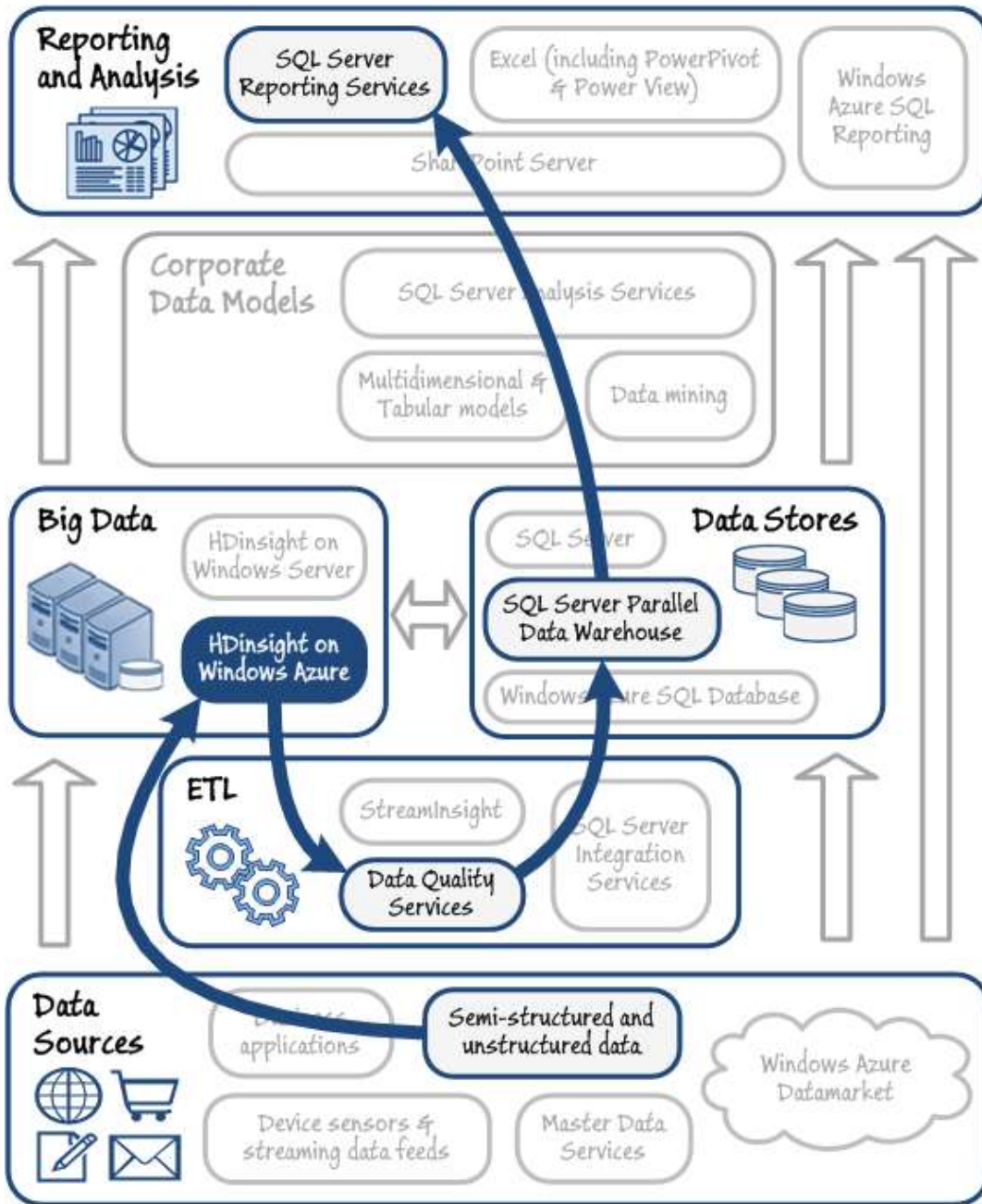


Figure 8
Querying, validating, cleansing, and storing data for use in reports

These are just three examples of the countless permutations and capabilities of the Microsoft data platform and HDInsight. Your own requirements will differ, but the combination of services and tools makes it possible to

implement almost any kind of Big Data solution using the elements of the platform. In the following chapters of this guide you will see many examples of the way that these tools and services work together.

Summary

In this introductory chapter you have discovered what a Big Data solution is, why you may need to implement one, and how it is different from the use of traditional relational databases and business intelligence tools. You have seen an overview of how Big Data solutions work, and a brief description of the different frameworks and platforms you can choose from.

This guide focuses on Microsoft's cloud based Big Data implementation, called Windows Azure HDInsight. However, HDInsight uses a 100% compatible implementation of the core Hadoop library and tools, as well as providing additional capabilities for writing queries in a range of languages.

Finally, the chapter introduced some of the typical scenarios for using a Big Data solution, and showed how Big Data solutions that use HDInsight fit into the Microsoft data platform. The scenarios are covered in detail on subsequent chapters of the guide. However, before then, the following chapter shows how to get started using Windows Azure HDInsight.

More Information

The official site for Apache Big Data solutions and tools is the [Apache Hadoop website](#).

For a detailed description of the MapReduce framework and programming model, see [MapReduce.org](#) and [Wikipedia](#).

For an overview and description of Microsoft HDInsight see [Microsoft Big Data](#).

To sign up for the Windows Azure service, go to [Windows Azure HDInsight](#).

You can download HDInsight Server from the [Microsoft Download Center](#).

The [Microsoft .NET SDK For Hadoop](#) is available from Codeplex or you can install it with NuGet.

The Microsoft TechNet library contains articles related to HDInsight. Search for these using the URL <http://social.technet.microsoft.com/Search/en-US?query=hadoop>.

There are also several blogs that cover HDInsight, such as <http://prologika.com/CS/blogs/blog/archive/tags/Hadoop/default.aspx>.

The official support forum for HDInsight is at <http://social.msdn.microsoft.com/Forums/en-US/hdinsight/threads>.

Chapter 2: Implementing Big Data Solutions with HDInsight

This chapter describes a general approach for implementing Big Data solutions with HDInsight. This includes an overview of planning, design, development, and testing considerations for a Big Data solution; and information about specific implementation decisions for HDInsight cluster and storage configuration, obtaining and ingesting source data, and processing the data to produce results for analysis and reporting.

Overview of the Big Data Process

Big Data implementation typically involves a common series of stages, irrespective of the type of data and the ultimate aims for obtaining information from that data. The stages are:

- **Determine the analytical goals and source data.** Before you start any data analysis project, it is useful to be clear about what it is you hope to achieve. You may have a specific question that you need to answer in order to make a critical business decision; in which case you must identify data that may help you determine the answer, from where it can be obtained, and if there are any costs associated with procuring it. Alternatively, you may already have some data that you want to explore to try to discern useful trends and patterns. Either way, understanding your goals will help you design and implement the solution that best supports those goals.
- **Plan and configure the infrastructure.** This involves setting up the Big Data software you have decided to use, or subscribing to an online service such as Windows Azure HDInsight. You may also consider integration options with existing database and BI infrastructure.
- **Obtain the data and submit it to HDInsight.** During this stage you decide how you will collect the data you have identified as the source, and how you will get it into HDInsight for processing. Often you will store the data in its raw format to avoid losing any useful contextual information it contains, though you may choose to do some pre-processing before storing it to remove duplication or to simplify it in some other way.
- **Process the data.** After you have started to collect and store the data, the next stage is to develop the processing solutions you will use to extract the information you need. While you can usually use Hive and Pig queries for even quite complex data extraction, you will occasionally need to create map/reduce components to perform more complex queries against the data.
- **Evaluate the results.** Probably the most important step of all is to ensure that you are getting what you expected, and that the results do make sense. Complex queries can be hard to write, and to get right first time. It's easy to make assumptions or miss edge cases that can skew the results quite considerably. Of course, it may be that you don't know what the expected result actually is (after all, the whole point of Big Data is to discover hidden information from the data) but you should make every effort to

validate the query results before making business decisions around them. In many cases, a business user who is familiar enough with the business context can perform the role of a *data steward* and review the results to verify that they are meaningful, accurate, and useful.

- **Tune the solution.** After you are convinced that your solution is working correctly, you should review the log files it creates, the processing techniques you use, and the implementation of the queries to ensure that they are executing in the most efficient way. It's possible to fine tune Big Data solutions to improve performance, reduce network load, and minimize the processing time by adjusting some parameters of the query and the execution platform or by compressing the data that does pass over the network.

Note that in many ways, data analysis is an iterative process; and you should take this approach when building a big data solution. In particular, given the large volumes of data and correspondingly long processing times typically involved in big data analysis, it can be useful to start by implementing a proof of concept iteration in which a small subset of the targeted data is used to validate the processing steps and results before proceeding with a full analysis. This enables you to test your big data processing design on a small HDinsight cluster or a single-node on-premise solution before scaling out to accommodate production-level data volumes.

Determining Analytical Goals and Source Data

Before embarking on a Big Data project, it is generally useful to think about what you hope to achieve and clearly define the analytical goals of the project. In some projects, there may be a specific question that the business wants to answer – such as “where should we open our new store?” In other projects, the goal may be more open-ended; for example to examine web site traffic and try to detect patterns and trends in visitors. Understanding the goals of the analysis can help you make decisions about the design and implementation of the solution, including the specific technologies to use and the level of integration with existing BI infrastructure.

Analytical Goals

Although every project has its own specific requirements, in general Big Data projects often fall into one of the following categories:

- **One-time analysis for a specific business decision.** For example, a company planning to expand by opening a new physical store, might use Big Data techniques to analyze demographic data for a shortlist of proposed store sites to determine the location that is likely to result in the highest revenue for the store. Alternatively, a charity planning to build water supply infrastructure in a drought-stricken area might use a combination of geographic, geological, health, and demographic statistics to identify the best locations to target.
- **Open “blue sky” exploration of “interesting” data.** Sometimes, the goal of Big Data analysis is simply to find out what you don't already know from the available data. For example, a business might be aware that customers are using Twitter to discuss its products and services, and want to explore the tweets to determine if any patterns or trends can be found relating to brand visibility or customer sentiment. There may be no specific business decision that needs to be made based on the data, but gaining a

better understanding of how customers perceive the business might inform business decision making in the future.

- **Ongoing reporting and BI.** In some cases, the Big Data solution will be used to support ongoing reporting and analytics, either in isolation or integrated with an existing enterprise BI solution. For example, a real estate business that already has a BI solution that enables analysis and reporting of its own property transactions across time periods, property types, and locations might be extended to include demographic and population statistics data from external sources. Integration patterns for HDInsight and enterprise BI solutions are discussed in Chapter 3.



In many respects, data analysis is an iterative process. It is not uncommon for an initial project based on open exploration of data to uncover trends or patterns that form the basis for a new project to support a specific business decision or to extend an existing BI solution.

Source Data

In addition to determining the analytical goals of the project, you must identify sources of data that can be used to meet those goals. Often, you will already know which data sources need to be included in the analysis; for example, if the goal is to analyze trends in sales for the past three years, you can use historic sales data from internal business applications or a data warehouse. However, in some cases you may need to search for data to support the analysis you need to perform; for example, if the goal is to determine the best location in which to open a new store, you may need to search for useful demographic data that covers the locations under consideration.

It is common in Big Data projects to combine data from multiple sources and create a “mash up” that enables you to analyze many different aspects of the problem in a single solution. For example, you might combine internal historic sales data with geographic data obtained from an external source to plot sales volumes on a map. You may then overlay the map with demographic data to try to correlate sales volume with particular geo-demographic attributes.

Common types of data source used in a Big Data solution include:

- **Internal business data from existing applications or BI solutions.** Often this data is historic in nature or includes demographic profile information that the business gathered from its customers. For example, you might use historic sales records to correlate customer attributes with purchasing patterns, and then use this information to support targeted advertising or predictive modeling of future product plans.
- **Log files.** Often applications or infrastructure services generate log data that can be useful for analysis and decision making with regard to managing IT reliability and scalability. Additionally, in some cases combining log data with business data can reveal useful insights about how IT services support the business. For example, you might use log files generated by Internet Information Services (IIS) to assess network bandwidth utilization, or to correlate web site traffic with sales transactions in an ecommerce application.

- **Sensors.** Increased automation in almost every aspect of life has led to a growth in the amount of data recorded by electronic sensors. For example, RFID tags in smart cards are now routinely used to track passenger progress through mass transit infrastructure, and sensors in plant machinery generate huge quantities of data in production lines. This availability of data from sensors enables highly dynamic analysis and real-time reporting.
 - **Social media.** The massive popularity of social media services such as Twitter, Facebook, LinkedIn, and others is a major factor in the growth of data volumes on the Internet. Many social media services provide application programming interfaces (APIs) that you can use to query the huge amount of data shared by users of these services and consume that data for analysis. For example, a business might use Twitter's query API to find tweets that mention the name of the company or its products, and analyze the data to determine how customers feel about the company's brand.
 - **Data feeds.** Many web sites and services provide data as a feed that can be consumed by client applications and analytical solutions. Common feed formats include RSS, ATOM, and OData, and the data sources themselves include blogs, news services, weather forecasts, and financial markets data.
 - **Government and special interest groups.** Many government organizations and special interest groups publish data that can be used for analysis. For example, the UK government publishes over 9000 downloadable datasets in a variety of formats, including statistics on population, crime, government spending, health, and other factors of life in the UK. Similarly, the US government provides census data and other statistics as downloadable datasets or in dBASE format on CD-ROM. Additionally, many international organizations provide data free of charge – for example, the United Nations makes statistical data available through its own web site and in the Windows Azure Datamarket.
 - **Commercial data providers.** There are many organizations that sell data commercially, including geographical data, historical weather data, economic indicators, and others. The Windows Azure Datamarket provides a central service through which you can locate and purchase subscriptions to these data sources.
-

When planning data sources to use in your Big Data solution, consider the following factors:

- **Availability.** How easy is it to find and obtain the data? You may have a specific analytical goal in mind, but if the data required to support the analysis is difficult (or impossible) to find you may waste valuable time trying to obtain it. When planning a Big Data project, it can be useful to define a schedule that allows sufficient time to research what data is available, and after an agreed deadline if the data cannot be found you may need to revise the analytical goals.
- **Format.** In what format is the data available, and how can it be consumed? Some data is available in standard formats and can be downloaded over a network or Internet API. In other cases, the data may only be available as a real-time stream that you must capture and structure for analysis. Later in the process, you will consider tools and techniques for consuming the data from its source and ingesting it into HDInsight, but even during this early stage you should identify the format and connectivity options for the data sources you want to use.

- **Relevance.** Is the data relevant to the analytical goals? You may have identified a potential data source and already be planning how you will consume it and ingest it into the analytical process. However, you should first examine the data source carefully and ensure that the data it contains is relevant to the analysis that needs to be performed.
 - **Cost.** You may determine the availability of a relevant dataset, only to discover that the cost of obtaining the data outweighs the potential business benefit of using it. This can be particularly true if the analytical goal is to augment an enterprise BI solution with external data on an ongoing basis, and the external data is only available through a commercial data provider.
-

Planning the Infrastructure

Having identified the data sources you need to analyze, you must design and configure the infrastructure required to support the analysis. Considerations for planning an HDInsight-based infrastructure for Big Data analysis include whether to use an on-premise or cloud-based HDInsight solution, which file storage option to use for HDInsight on Windows Azure, and how to integrate HDInsight with existing enterprise BI infrastructure.

Choosing an HDInsight Installation Option

HDInsight is available as an on-premise server application for Windows Server, or as a cloud service on Windows Azure. Consider the following guidelines when choosing between these options:

HDInsight on Windows Azure is a good choice when:

- You require HDInsight only for a specific period of time.
 - You require HDInsight for ongoing analysis, but the workload for HDInsight will vary, sometimes requiring a cluster with many nodes and sometimes not requiring any HDInsight services at all.
 - You want to avoid the cost in terms of capital expenditure, skills development, and time it takes to provision, configure, and manage on-premise servers for HDInsight.
-

HDInsight on Windows Server is a good choice when:

- You require ongoing HDInsight services with a predictable and constant level of scalability.
 - You have the necessary technical capability and budget to provision, configure, and manage your own HDInsight cluster.
 - The data you plan to analyze must remain on your own servers for compliance or confidentiality reasons.
-

Choosing a Storage Solution for HDInsight

HDInsight on Windows Azure supports the Hadoop file system (HDFS) natively, but also provides the ability to store data in on Azure Storage Vault (ASV) containers in the Windows Azure Blob Store. In general, using ASV offers significant advantages over native HDFS, including:

- The ability to store a larger volume of data than the storage attached to the HDInsight cluster members can hold.
- Low cost dynamic storage with geo-located redundancy.
- Persistence of files even after the HDInsight cluster is released.

In some cases, you might want to use native HDFS so that scripts written for other distributions of Hadoop or for HDInsight on Windows Server can be executed without modification. However, in most scenarios you should consider using ASV to host the source and output files for an HDInsight on Windows Azure cluster.

To use ASV with an HDInsight cluster, you must associate your Windows Azure Blob Store account with the cluster by specifying your storage account name and a passkey that can be used to provide secure access to the storage. After this association has been configured, you can reference specific locations in ASV in the form **asv://container/path**. You can use this syntax in the HDInsight interactive console and in scripts used to process data.

Obtaining and Submitting Source Data

When you have identified the sources of data you want to analyze and decided on the specific HDInsight Infrastructure design you plan to use, you can start to think about how you will obtain the data and submit it to HDInsight.

In some cases, such as when the data source is an internal business application or database, extracting the data into a text file that can be consumed by HDInsight is relatively straightforward; and in the case of external data sources from governments and commercial data providers, the data is often available for download in text format. However, in many cases you may need to extract data through a web service or other API.

Generally, you submit source data to HDInsight in the form of a text file – often delimited, but sometimes completely unstructured. Source files for data processing are uploaded to HDFS folders (or ASV containers), from where HDInsight map/reduce jobs can load and process them.

In some scenarios, you might want to transfer data directly to HDInsight from its source, while in other cases you might want to perform some pre-processing tasks on the source data to prepare it for processing. For example, you can improve performance by removing any unused fields or values from the source data before uploading it to HDInsight. Additionally, you might want to format some values, for example changing date strings to standard numerical date values. You may even want to perform some automated data validation and cleaning by using a technology like SQL Server Data Quality Services before submitting the data to HDInsight.

Tools and Technologies for HDInsight Data Ingestion

You can create custom applications or scripts to extract source data, or use off-the-shelf tools and platforms. Common tools on the Microsoft platform for extracting data from a source include Excel, SSIS, and StreamInsight. To upload the data to HDInsight, you can use a variety of file upload tools, or you can build a custom solution. If you are using ASV for storage in an HDInsight for Windows Azure cluster, you can use a tool such as *AzCopy*, which can be used as a standalone command line tool or as a library from your own code. Similarly, if your cluster is using native HDFS storage, you can use a tool that supports FTPS data transfer. For example, you could use *Curl* to upload files from the command line, or use *libCurl* programmatically.

Many HDInsight queries use a folder as a data source, not an individual file; so you can split your data across multiple files in the same folder and still process it in a single query. However, as a general rule, it is usually more efficient to consolidate your data in a single large file than to split it across multiple small files. HDInsight is most efficient when processing files greater than 256 MB as this is the block size used to distribute data for processing in parallel across multiple cluster nodes.

The Hadoop ecosystem also includes tools that are specifically designed to simplify the loading of particular kinds of data into HDFS. *Sqoop* provides a mechanism for transferring data from a relational database into HDFS files (and vice-versa), and *Flume* provides a framework for copying web server log files to an HDFS location for processing in HDInsight.

Common Data Load Patterns for HDInsight

This section describes some common use cases and for obtaining source data and uploading it to HDInsight. For each pattern, a suggested implementation is described.

Interactive Data Ingestion

Often, an initial exploration of HDInsight involves experimenting with a relatively small volume of data. In this case, there is no requirement for a complex, automated data ingestion solution, and the tasks of obtaining the source data, preparing it for processing, and uploading it to HDInsight can be performed interactively.

If the source data is not already available as an appropriately formatted file, you can use Excel to extract a relatively small volume of tabular data from a data source, reformat it as required, and save it as a delimited text file. Excel supports a range of data sources, including relational databases, XML documents, OData feeds, and the Windows Azure Datamarket. You can also use Excel to import a table of data from any web site, including an RSS feed. In addition to the standard Excel data import capabilities, you can use add-ins such as the Data Explorer to import and transform data from a wide range of sources.

After you have used Excel to prepare the data and save it as a text file, you can upload it to an HDInsight cluster by using the **fs.put()** command in the interactive console. This technique enables you to upload a file containing up to 5MB of data. If you need to upload a file larger than this, you can use a command line tool such as *AzCopy* if you are using ASV storage, or *Curl* if you are using native HDFS storage. This approach is shown in Figure 1.

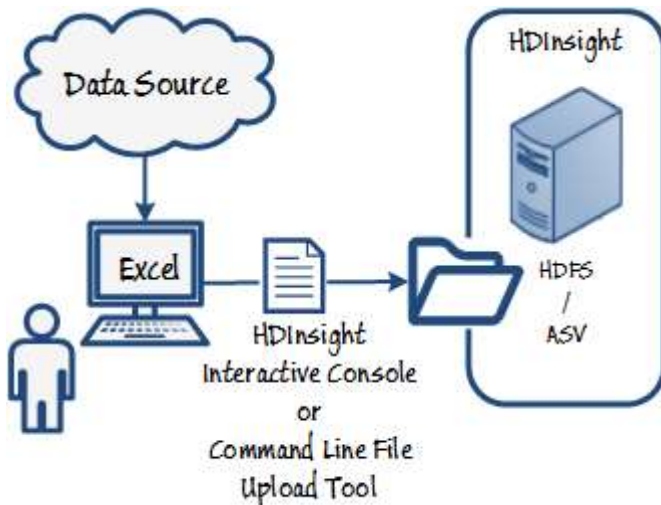


Figure 1
Interactive Data Ingestion

Automated Batch Upload to HDInsight

When a Big Data solution grows beyond the initial exploration stages, it is common to automate data ingestion so that data can be extracted from sources and loaded to the HDInsight cluster on a scheduled basis without human intervention. SSIS provides a platform for building automated ETL workflows that perform the necessary tasks to extract source data, apply transformations, and upload the resulting data to HDFS or ASV storage.

An SSIS solution consists of one or more *package*, each containing a *control flow* to perform a sequence of tasks. Tasks in a control flow can include calls to web services, FTP operations, file system tasks, automation of command line commands, and others. In particular, a control flow usually includes one or more *data flow* tasks, which encapsulate an in-memory, buffer-based pipeline of data from a source to a destination, with transformations applied to the data as it flows through the pipeline.

You can create an SSIS package that extracts data from a source and uploads it to an HDFS or ASV folder, and automate execution of the package using the SQL Server Agent service. The package usually consists of a data flow to extract the data from its source through an appropriate source adapter and then apply any required transformations. To get the data into the appropriate HDFS or ASV folder, you can use one of the following options:

- Procure or implement a custom destination adapter that programmatically writes the data to an HDFS folder or ASV container, and add it to the data flow.
- Use the data flow to download the data to a local file, and then use a command line task in the control flow to automate a file upload utility to upload the file to HDFS or ASV.

Figure 2 shows an automated batch upload architecture.

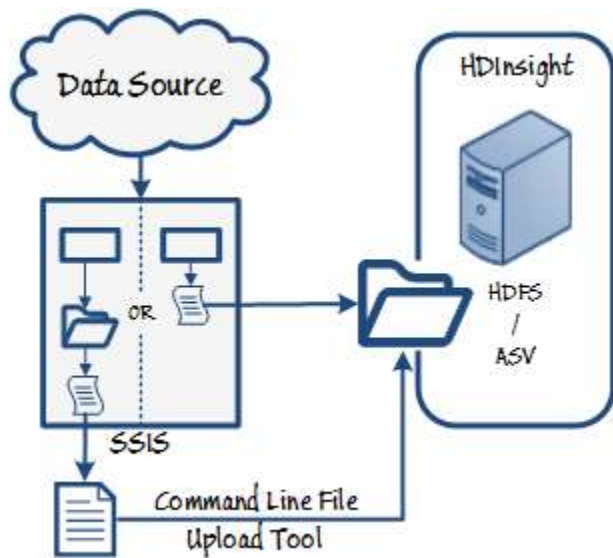


Figure 2
Automated batch upload with SSIS

For more information about creating ETL solutions with SSIS, see [SQL Server Integration Services](#) in SQL Server Books Online.

Real-Time Data Stream Capture

In some scenarios, the data that must be analyzed is only available as a real-time stream of events; for example, from a streaming web feed or electronic sensors. StreamInsight is a complex event processing (CEP) engine with a framework API for building applications that consume and process event streams. While HDInsight itself is optimized for batch processing of large volumes of data, you can use StreamInsight to capture events that are redirected to a real-time visualization solution and also captured for submission to HDInsight for historic analysis.

StreamInsight is available as an on-premise solution, or as a Windows Azure service. Regardless of the implementation you decide to use, a StreamInsight application consists of a standing query that uses LINQ syntax to extract events from a data stream. Events are implemented as classes or structs, and the properties defined for the event class provide the data values for visualization and analysis. You can create a StreamInsight application by using the Observable/Observer pattern, or by implementing input and output adapters from base classes in the StreamInsight framework.

When using StreamInsight to capture events for analysis in HDInsight, you can implement code to write the captured events to a local file for batch upload to the HDInsight cluster; or you can implement code that writes the event data directly to HDFS or ASV storage. Your code can append each new event to an existing file, create a new file for each individual event, or periodically create a new file based on temporal windows in the event stream.

Figure 3 shows a real time data stream capture architecture for HDInsight.

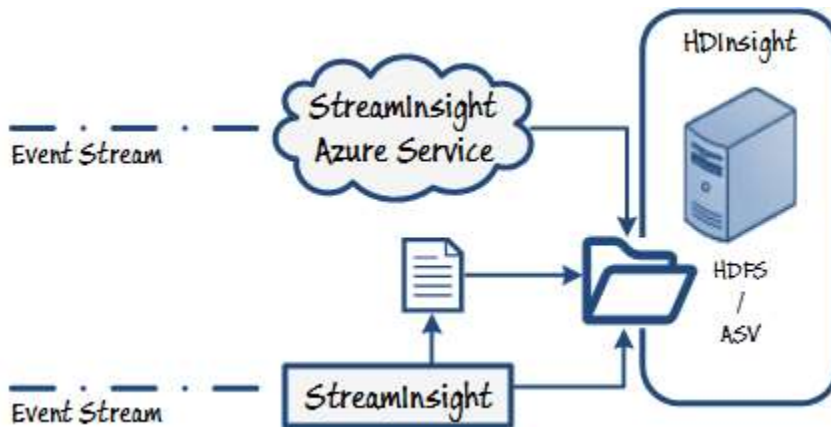


Figure 3
Real-time data stream capture with StreamInsight

For more information about developing StreamInsight applications, see the [Microsoft StreamInsight](#) on MSDN.

Options for Loading Relational Data into HDInsight

Source data for analysis is often obtained from a relational database, and to support this common scenario, you can use Sqoop to extract the data you require from a table, view, or query in the source database and saving the results as a file in HDFS or ASV. This approach makes it easy to transfer data from a relational database to HDInsight when your infrastructure supports direct connectivity from the HDInsight cluster to the database server. For example, if your database is hosted in Windows Azure SQL Database or a virtual machine in Windows Azure, you can use Sqoop to load data to an HDInsight for Windows Azure cluster. Similarly, you can use Sqoop to transfer data between an on-premise instance of SQL Server and an HDInsight for Windows Server cluster on the same network.

If direct network connectivity between the database server and HDInsight is not possible, then you can use SSIS to implement an automated batch upload solution as described previously.

Approaches for loading relational data into HDInsight are shown in Figure 4.

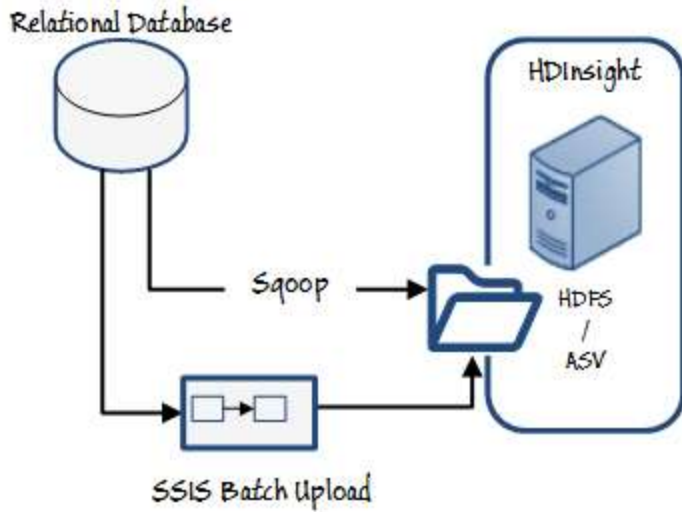


Figure 4
Loading relational data into HDInsight

Options for Loading Web Server Log Files to HDInsight

Analysis of web server log data is another common use case for HDInsight, and requires that log files be uploaded to HDFS. Flume is an open source project for an agent-based framework to copy web server logs to a central location, and is often used to load web server logs to HDFS for processing in Hadoop. Flume is not included in HDInsight, but can be downloaded from the Flume project web site and installed on Windows.

As an alternative to using Flume, you can use SSIS to implement an automated batch upload solution as described previously. Options for uploading web server log files are shown in Figure 5.

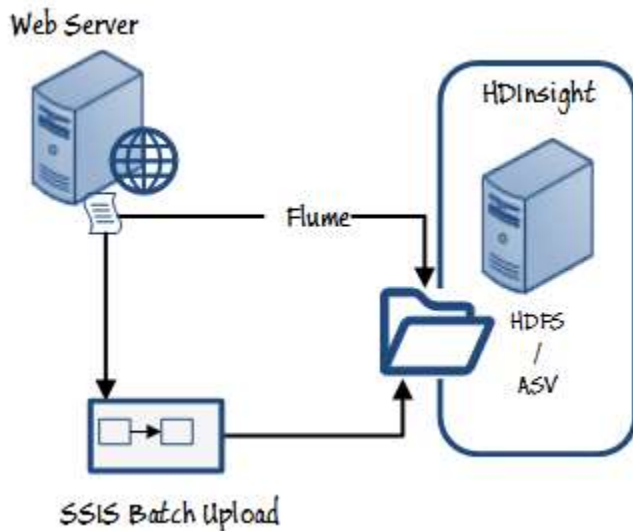


Figure 5
Loading web server logs into HDInsight

Considerations for Implementing Custom File Upload Code

In many of the patterns described above, the data is uploaded to an HDFS or ASV folder programmatically or by using a command line upload utility. There are numerous off-the-shelf tools and libraries available for uploading data to FTPS endpoints (as used by native HDFS storage) or to the Azure Blob Store (as used by ASV storage), and in most cases you should evaluate these tools and choose the one that best suits your requirements. In some cases however, you may want to implement a custom solution that gives you greater control over how the data is transferred to the destination. If you decide to implement a custom solution for uploading data to an HDInsight cluster, consider the following guidelines:

- General guidelines:
 - Implement your solution as a library that can be called programmatically from scripts or custom components in an SSIS data flow or a StreamInsight application. This will ensure that the solution provides flexibility for use in future Big Data projects as well as the current one.
 - Create a command line utility that uses your library so that it can be used interactively or called from an SSIS control flow.
 - Guidelines for uploading data to ASV:
 - Split the data into chunks that can be uploaded in parallel.
 - Be aware of Azure Blob Store size limitations
-

Processing the Data

After you have uploaded your source data to HDInsight, you can process it to generate query results for analysis. As described in chapter 1, all HDInsight processing consists of a map job in which multiple cluster nodes process their own subset of the data, and a reduce job in which the results of the map jobs are consolidated to create a single output. You can implement map/reduce code to perform the specific processing required for your data, or you can use a query interface such as Pig and Hive to write queries that are translated to map/reduce jobs and executed.

Many HDInsight processing solutions are incremental in nature and consist of multiple queries, each operating on the output of the previous one. For example, you might first use a custom map/reduce job to summarize a large volume of unstructured data, and then create a Pig script to restructure and group the data values produced by the initial map/reduce job. Finally, you might create Hive tables based on the output of the Pig script so that client applications such as Excel can easily consume the results.

Writing Map/Reduce Code

Fundamentally, all HDInsight processing is performed by running map/reduce batch jobs. You can implement map/reduce code in Java (which is the native language for map/reduce jobs in all Hadoop distributions) or in a number of additionally supported languages, including JavaScript, Python, C# and F#.

Map/Reduce code consists of two functions; a mapper and a reducer. The mapper is run in parallel on multiple cluster nodes, each node applying it to a subset of the data. The output from the mapper function on each node is then passed to the reducer function, which is run on one node to collate and summarize the results of the mapper function.

In most HDInsight processing scenarios, it is simpler and more efficient to use a higher-level abstraction such as Pig or Hive to generate the required map/reduce code on your behalf. However, you might want to consider creating your own map/reduce code when:

- You need to process completely unstructured data by parsing it and using custom logic to obtain structured information from it.
- You need to perform complex tasks that are difficult (or impossible) to express in Pig or Hive. For example, using an external geocoding service to convert latitude and longitude coordinates or IP addresses in the source data to geographical location names.

Processing Data with Pig

In the previous example, you saw how to use Pig to run an explicit map/reduce job. Pig is a query interface that provides a workflow semantic for processing data in HDInsight, and as well as using it to call custom map/reduce code, you can use Pig statements to execute implicit map/reduce jobs that are generated by the Pig interpreter on your behalf. Pig statements are expressed in a language named *Pig Latin*, and generally involve defining *relations* that contain data, either loaded from a source file or as the result of a Pig Latin expression on an existing relation. Relations can be thought of as result sets, and can be based on a schema (that you define in the Pig Latin statement used to create the relation) or completely unstructured.

You can run Pig Latin statements interactively in the interactive console or in a command line Pig shell named *Grunt*. You can also combine a sequence of Pig Latin statements in a script, which can then be executed as a unit. These Pig Latin statements are used to generate map/reduce jobs by the Pig interpreter, but the jobs are not actually generated and executed until you call either a DUMP statement (which is used to display a relation in the console, and is useful when interactively testing and debugging Pig Latin code) or a STORE statement (which is used to store a relation as a text file in a specified folder).

Pig is a good choice when you need to:

- Restructure source data by defining columns, grouping values, or converting columns to rows.
- Use a workflow-based approach to process data as a sequence of operations, which is often a logical way to approach many map/reduce tasks.

Pig Latin syntax has some similarities to SQL, and encapsulates many functions and expressions that make it easy to create a sequence of complex map/reduce jobs with just a few lines of simple code.

For more information about Pig Latin syntax, see [Pig Latin Reference Manual 2](#) on the Apache Hadoop site.

Processing Data with Hive

Pig provides a powerful abstraction layer over map/reduce, and enables you to perform complex processing of your source data to generate output that is useful for analysis and reporting. However, the syntax of Pig Latin, while similar to SQL in some respects, can be complex for non-programmers to master. Additionally, Pig scripts generally save their results as text files in HDFS or ASV, where they can easily be viewed interactively (for example in the HDInsight interactive JavaScript console), but can be difficult to consume from client applications or processes without copying the output files and importing them into client tools such as Excel.

Hive is another abstraction layer over map/reduce that addresses both of these problems by providing a query language called HiveQL that is syntactically very similar to SQL, and by supporting the ability to create tables of data that can be accessed remotely through an ODBC connection. In effect, Hive enables you to create an interface layer over map/reduce that can be used in a similar fashion to a traditional relational database; enabling business users to use familiar tools like Excel and SQL Server Reporting Services to consume data from HDInsight in a similar way as they would from a database system like SQL Server. Installing the ODBC driver for Hive on a client computer also installs an add-in for Excel, which provides a pane in which users can connect to an HDInsight cluster and submit HiveQL queries that return data to an Excel worksheet.

Hive is a good choice for data processing when:

- The source data is relatively structured, and can easily be mapped to a tabular schema.
- The processing you need to perform can be expressed effectively as HiveQL queries.
- You want to create a layer of tables through which business users can easily query source data or data generated by previously executed map/reduce jobs or Pig scripts.

You can use the Hive command line tool on the HDInsight cluster to work with Hive tables, or you can use the Hive interface in the HDInsight interactive console. You can also use the Hive ODBC driver to connect to Hive from any ODBC-capable client application.

Creating Tables with Hive

Hive uses tables to impose schema on data and provide a query interface for client applications. The key difference between Hive tables and those in traditional database systems, such as SQL Server, is that Hive adopts a “schema on read” approach that enables you to be flexible about the specific columns and data types that you want to project on top of your data. You can create multiple tables with different schema from the same underlying data, depending on how you want to use that data. The most important point to take away from this is that the table is simply a metadata schema that is imposed on data in underlying files.

You create tables by using the HiveQL **CREATE TABLE** statement, which in its simplest form looks similar to the equivalent statement in Transact-SQL. For example, the following code could be used to create a table named **mytable**:

HiveQL

```
CREATE TABLE mytable  
(col1 STRING,
```

```
col2 INT)
```

The primitive datatypes you can use for column in a Hive table are:

- TINYINT
- SMALLINT
- INT
- BIGINT
- BOOLEAN
- FLOAT
- DOUBLE
- STRING
- BINARY
- TIMESTAMP
- DECIMAL

In addition to these primitive types, you can define columns as ARRAY, MAP, STRUCT, and UNIONTYPE. For more information about Hive datatypes, see [Hive Data Types](#) on the Apache Hive language manual site.

As discussed previously, the table is simply a metadata definition that is imposed on data in underlying files. So where are the files for the **mytable** table? By default, Hive stores table data in the `/hive/warehouse/[table_name]` path in HDFS, so the previous code sample will create the table metadata definition and an empty directory at `/hive/warehouse/mytable`. Deleting the table by executing the statement **DROP TABLE mytable** will delete the metadata definition from the Hive database and also remove the `/hive/warehouse/mytable` directory.

You can specify an alternative path for a table by including the **LOCATION** clause in the **CREATE TABLE** statement. You can also specify the format for the files the table data will be stored in if you do not want to use the default control+A delimited (ASCII code 1) format. The following code sample creates the **mytable** table in a non-default folder and specifies that the data files for the table should be tab-delimited.

HiveQL

```
CREATE TABLE mytable
(col1 STRING,
 col2 INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE LOCATION '/user/hduser/mydata/mytable'
```

The ability to specify a non-default location for the table data is useful when you want to enable other applications or users to access the files outside of Hive. A simple way to load data into the table is to copy files of the appropriate format to the folder, and when the table is queried the schema defined in its metadata is automatically applied to the data in the files. An additional benefit of being able to specify the location is that this makes it easy to create a table for data that already exists in the folder (perhaps as the output from a

previously executed map/reduce job or Pig script) – after creating the table, the existing data in the folder can immediately be retrieved with a HiveQL query.

However, one consideration for using an existing location is that when the table is deleted, the folder it references will also be deleted – even if it already contained data files when the table was created. If you want to manage the lifetime of the folder containing the data files separately from the lifetime of the table, you must use the **EXTERNAL** keyword in the **CREATE TABLE** statement to indicate that the folder will be managed externally from Hive, as shown in the following code sample:

HiveQL

```
CREATE EXTERNAL TABLE mytable
(col1 STRING,
 col2 INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE LOCATION 'asv://mydata/mytable'
```

As a general guide, create external tables when the data is used by processes other than Hive or if the data files must be preserved even if the table is dropped. Additionally, if you want to store the data files in ASV instead of the native HDFS cluster storage, then you must use an external table definition. Conversely, use internal tables for data that is temporary or should have its lifecycle matched to the table. Additionally, if you plan to create a table from the results of a SELECT query against an existing table, the new table created from the query results must be internal.

Advanced options when creating a table include the ability to partition, skew, and cluster the data across multiple files and folders:

- You can use the **PARTITIONED BY** clause to create a subfolder for each distinct value in a specified column (for example, to store a file of daily data for each date in a separate folder). Partitioning can improve query performance as HDInsight will only scan relevant partitions in a filtered query.
- You can use the **SKEWED BY** clause to create separate files for each row where a specified column value is in a list of specified values. Rows with values not listed are stored in a single other file.
- You can use the **CLUSTERED BY** clause to distribute data across a specified number of subfolders (described as *buckets*) based on the values of specified columns using a hashing algorithm.

Loading Data into Tables

As previously stated, you can load data into Hive tables by simply copying data files into the appropriate folders. The table definition is purely a metadata schema that is applied to the data files in the folders when they are queried. This makes it easy to define tables in Hive for data that is generated by other processes and deposited in the appropriate folders when ready.

Additionally, you can use the HiveQL **LOAD** statement to load data from an existing file into a Hive table. This statement moves the file from its current location to the folder associated with the table. This technique is useful when you need to create a table from the results of a map/reduce job or Pig script that generates an

output file alongside log and status files, as it enables you to easily add the output data to a table without having to deal with the additional files that you do not want to include in the table.

Another way to add data to a table is to use a **CREATE TABLE** statement that includes a **SELECT** statement to query an existing table. The results of the **SELECT** statement are used to create data files for the new table. When you use this technique, the new table must be an internal table. Similarly, you can use an **INSERT** statement to insert the results of a **SELECT** query into an existing table.

Querying Tables with HiveQL

After you have created tables and loaded data files into the appropriate locations, you can start to query the data by executing HiveQL **SELECT** statements against the tables. As with all data processing on HDInsight, HiveQL queries are implicitly executed as map/reduce jobs to generate the required results. HiveQL **SELECT** statements are similar to SQL, and support common operations such as **JOIN**, **UNION**, and **GROUP BY**. For example, you could use the following code to query the **mytable** table described previously.

HiveQL

```
SELECT col1, SUM(col2) AS total
FROM mytable
GROUP BY col1
```

In addition to common SQL semantics, HiveQL supports the inclusion of custom map/reduce scripts embedded in a query through the **MAP** and **REDUCE** clauses, as well as custom user-defined functions (UDFs) implemented in Java. This extensibility enables you to use HiveQL to perform complex transforms to data as it is queried.

When designing an overall data processing solution with HDInsight, you can choose to perform complex processing logic in custom map/reduce components or Pig scripts as described earlier, and then create a layer of Hive tables over the results of the earlier processing that can be queried by business users who are familiar with basic SQL syntax. Alternatively, you can use Hive for all processing, in which case some queries may require scripts in **MAP** and **REDUCE** clauses or the use of custom UDFs to perform logic that is not possible in standard HiveQL functions. To help you decide on the right approach, consider the following guidelines:

- If the source data must be extensively transformed using complex logic before being consumed by business users, consider using custom map/reduce components or Pig scripts to perform most of the processing, and create a layer of simple to query Hive tables over the results.
- If the source data is already in an appropriate structure for querying, and only a few specific complex transforms are required by an experienced analyst with the necessary programming skills, consider using map/reduce scripts embedded in HiveQL queries to generate the required results.
- If queries will mostly be created by business users, but some complex logic is still regularly required to generate specific values or aggregations, consider encapsulating that logic in custom UDFs, as these will be simpler for business users to include in their HiveQL queries than a custom map/reduce script.

Evaluating the Results

After you have used HDInsight to process the data you can use it for analysis and reporting, which forms the foundation for business decision making. However, before making critical business decision based on the results of your queries, you must carefully evaluate them to ensure that they are:

- **Meaningful.** The results when combined and analyzed make sense, and the data values relate to one another in a meaningful way.
- **Accurate.** The results are correct, with no errors in source data or introduced by calculations or transformations applied during processing.
- **Useful.** The results are applicable to the business decision that they will support, and provide relevant metrics that help inform the decision making process.

Often, you will need to employ the services of a business user who intimately understands the business context for the data to “sanity check” the results and determine whether or not they fall within expected parameters. Depending on the complexity of the processing, you might also need to select a number of data inputs for spot-checking and trace them through the process to ensure that they produce the expected outcome.

When you are planning to use HDInsight to perform predictive analysis, it can be useful to evaluate the process against known values. For example, if your goal is to use demographic and historical sales data to determine the likely revenue for a planned future store, you can validate the processing model by using appropriate source data to predict revenue for an existing store and comparing the resulting prediction to the actual revenue value. If the results of the data processing you have implemented vary significantly from the actual known revenue, then it seems unlikely that the results for the planned future store will be reliable!

Viewing Results

To evaluate the results of your HDInsight data processing solution, you must be able to view them. There are many ways to implement reports and analytical visualizations from HDInsights results, and these are discussed in depth in Chapter 3. However, you can easily perform your initial evaluation of the results by using one or more of the techniques discussed in this section.

Viewing Data in the Interactive Console

The interactive console provides a simple way to view the results of HDInsights processing. You can view the contents of any file in HDFS or ASV by using the `#cat` command in the interactive console, and you can also create visualizations of data by loading the file data and using the JavaScript graph library to visualize it as shown in the following example code.

Interactive JavaScript

```
file = fs.read("./results/data")
data = parse(file.data, "col1, col2:long")
graph.pie(data)
```

These commands read the contents of the files in the `asv://results/data` folder into a variable, parse the variable into two columns (the first using the default `chararray` data type, the second containing a long integer), and display the data as a pie chart in the console; as shown in Figure 6.

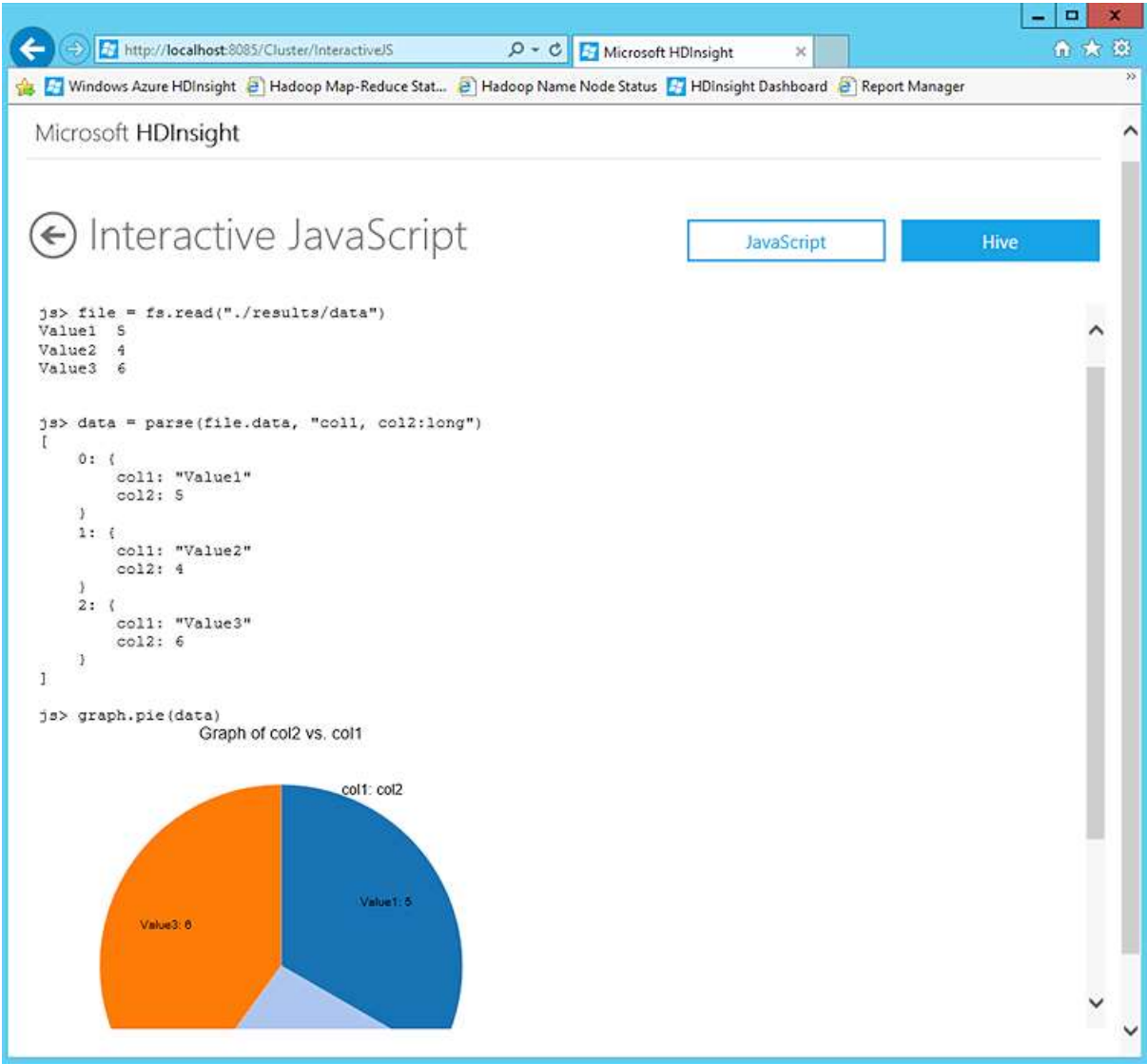


Figure 6
Viewing Data in the Interactive Console

If you have used Hive to create tables, you can also use the interactive console to execute HiveQL queries against those tables and display the results in the console, as shown in Figure 7.

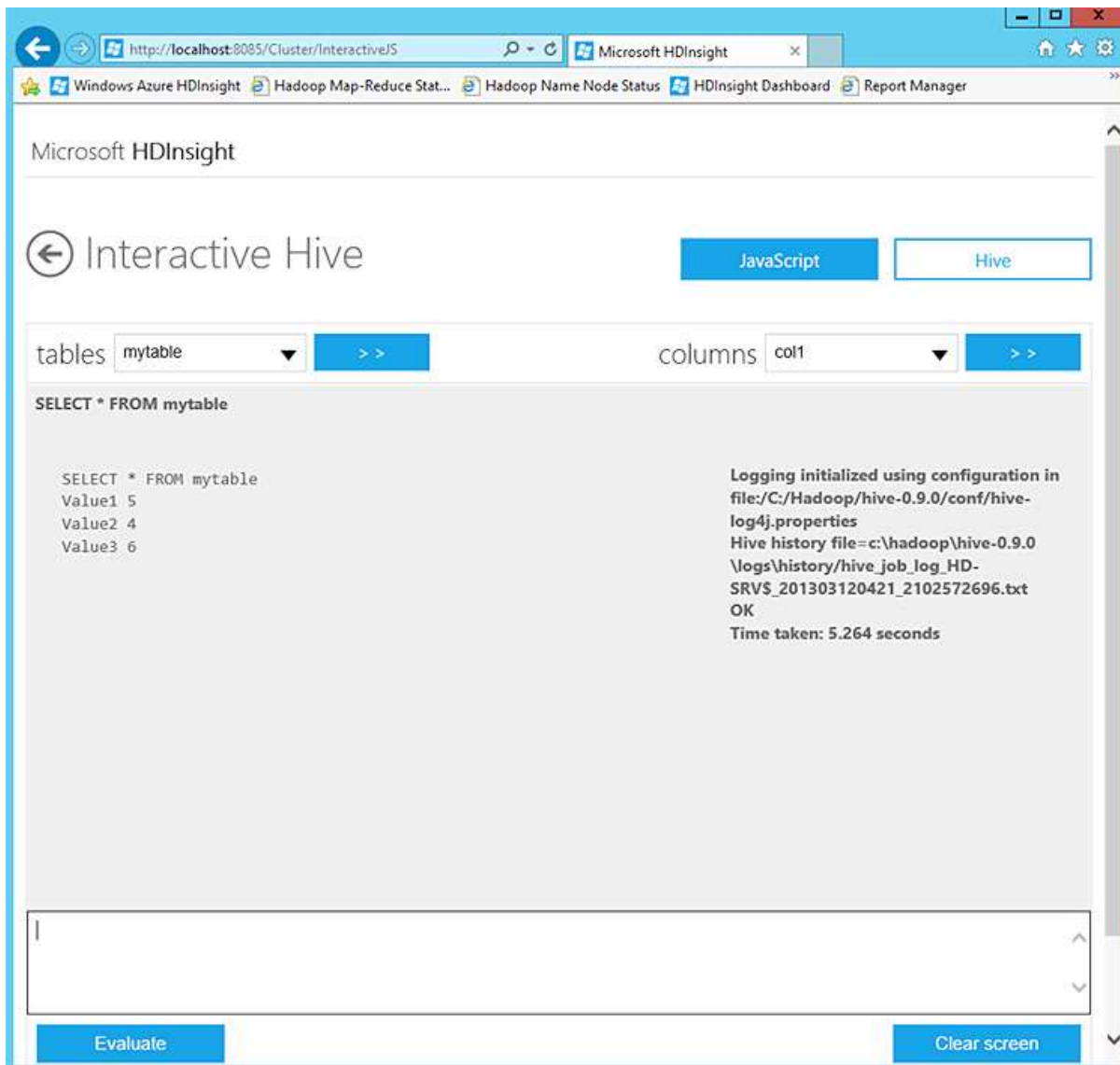


Figure 7
Executing a HiveQL query in the Interactive Console

Using the Data Explorer Add-In for Excel

While the interactive console provides a quick and easy way to view data produced by HDInsight processing, you might need to distribute processing results to business users for more detailed evaluation and exploration. To accomplish this, you can use the Data Explorer add-in for Excel, which enables you to import data from a wide range of sources into Excel for analysis and reporting.

The Data Explorer includes an option to import data from Hadoop, which enables users to connect to an HDInsight cluster and select one or more files on the HDFS file system. The add-in enables users to convert file data into tables, with specific column names and data types, before importing the data into a worksheet, where users can apply the full range of Excel functionality to the data. Figure 8 shows the Data Explorer add-in being used to import data from a file in an HDInsight cluster.

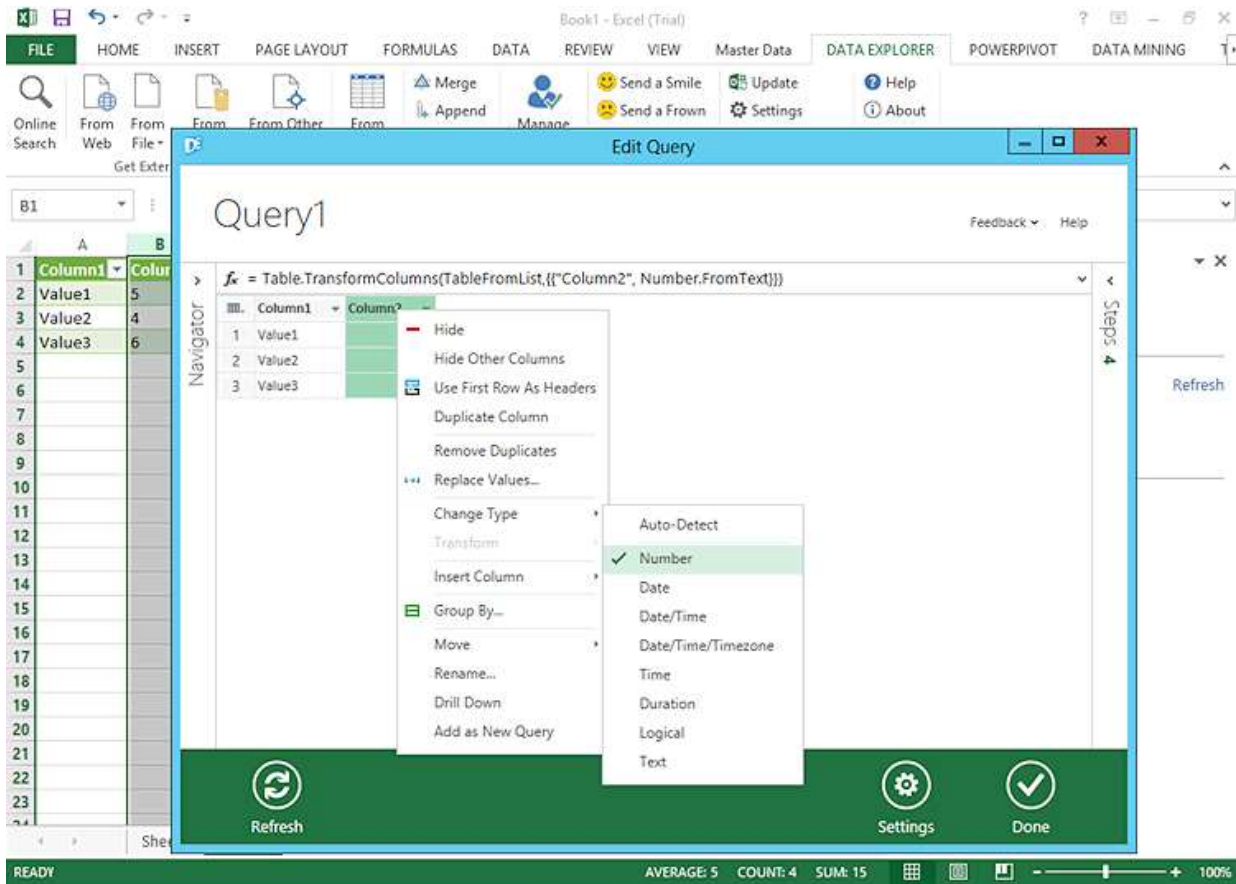


Figure 8
Using the Data Explorer Add-In with HDInsight

Accessing Hive Tables over ODBC

One of the main advantages of Hive is that it provides a querying experience that is similar to that of a relational database, which is a familiar technique for many business users. Additionally, the ODBC driver for Hive enables users to connect to HDInsight and execute HiveQL queries from familiar tools like Excel. By using the Hive add-in for Excel, users can connect to an HDInsight cluster over ODBC and use a query to retrieve a table of data into a worksheet, as shown in Figure 9.

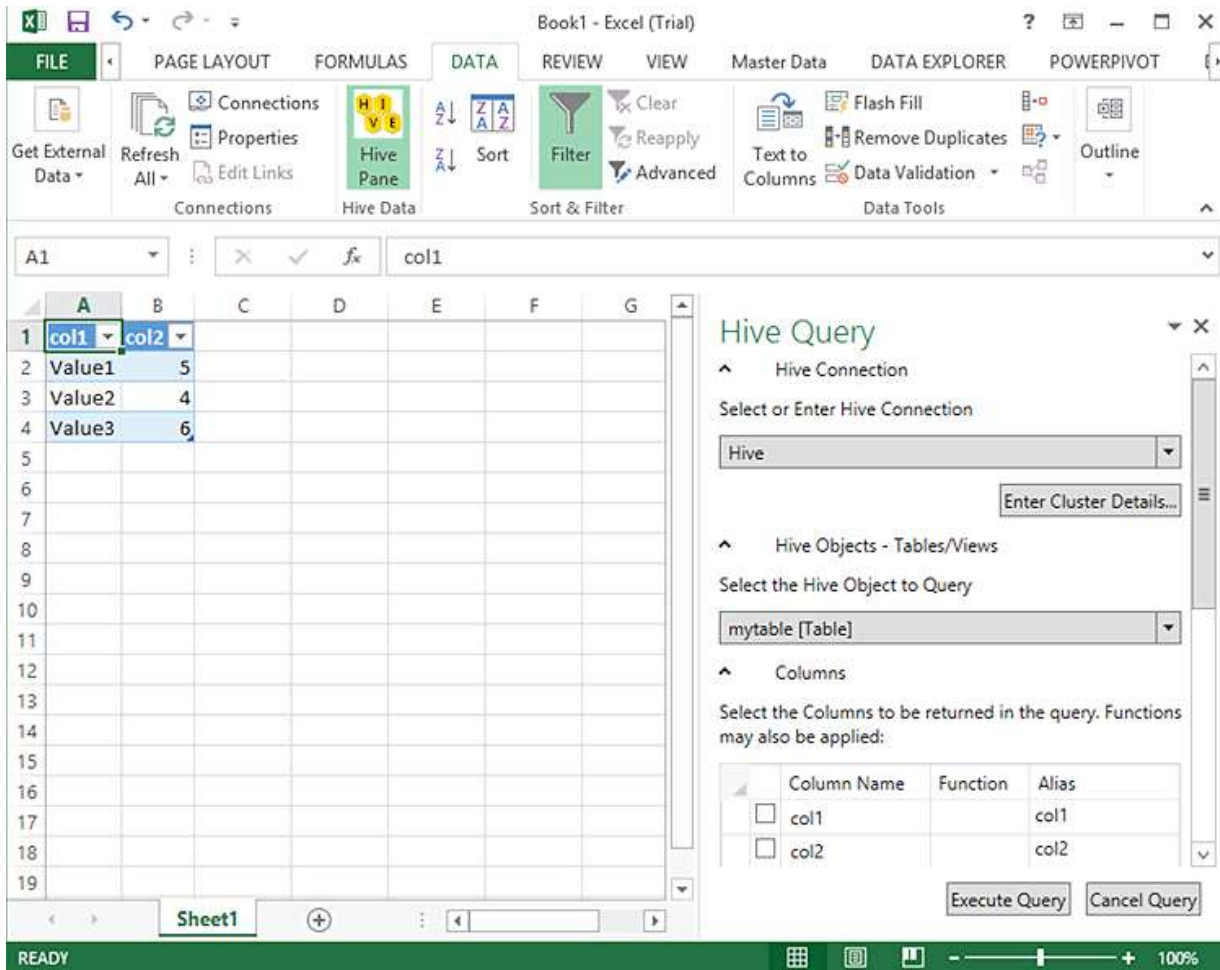


Figure 9
Accessing Hive over ODBC

Tuning the Solution

After you have built an HDInsight solution for big data processing, you can monitor its performance and start to find ways in which you can tune the solution to reduce query processing time or hardware resource utilization.

Summary

This chapter described the steps that are typically performed to create a Big Data solution with HDInsight. Not all projects will require all steps (for example, you may already have source data and not need to locate it), but by understanding the workflow described in this chapter, you can plan and implement effective solutions for analyzing Big Data.

Now that you know how to use HDInsight to ingest and process data, you can start to consider how you will use HDInsight as part of a business intelligence (BI) solution for Big Data in which the processed results can be

visualized in reporting and analytical tools. In the next chapter, a range of options for consuming processed data from HDInsight and integrating with other BI technologies will be discussed.

More Information

For more information about HDInsight, see the [Windows Azure HDInsight](#) web page.

Chapter 3: Using HDInsight for Business Analytics

In the previous chapter you learnt how Big Data solutions such as Microsoft HDInsight can help you discover vital information that may otherwise have remained hidden in your data—or even lost forever. This information can help you to evaluate your organization’s historical performance, discover new opportunities, identify operational efficiencies, increase customer satisfaction, and even predict likely outcomes for the future. It’s not surprising that Big Data is generating so much interest and excitement in what some may see as the rather boring world of data management!

In this chapter the focus moves away from the practicalities of implementing a Big Data solution to consider in more detail how you can consume the information you have learned how to extract. This means you need to think about how HDInsight fits into your existing business infrastructure. It may be that you just want to use it alongside your existing business intelligence (BI) systems, or you may want to deeply integrate it with these systems. The important point is that, irrespective of how you choose to use it, the end result is the same: some kind of analysis of the source data and meaningful visualization of the results.



Big Data solutions are not a replacement for your existing BI mechanisms. Instead, they complement these mechanisms and provide new opportunities for extracting information from your data.

This chapter explores how you can combine HDInsight with business analytics to make sense of huge volumes of semi-structured and unstructured data. Of course, many organizations already use data to improve decision making through their existing BI solutions that analyze and generate reports based on data generated by business activities and applications. Rather than seeking to replace traditional BI solutions, HDInsight provides a way to extend the value of your investment in BI by enabling you to incorporate a much wider variety of data sources that complement and integrate with existing data warehouse, analytical data models, and business reporting solutions.

Historical and Predictive Data Analysis

Most organizations already collect data from multiple sources. These might include line of business applications such as websites, accounting systems, and office productivity applications. Other data may come from interaction with customers, such as sales transactions, feedback, and reports from company sales staff. The data is typically held in one or more databases, or in a specially designed data warehouse system.

Having collected this data, organizations typically use it to perform the following kinds of analysis and reporting.

- **Historical analysis and reporting**, which is concerned with summarizing data to make sense of what happened in the past. For example, a business might summarize sales transactions by fiscal quarter and sales region, and use the results to create a report for shareholders. Additionally, business analysts

within the organization might explore the aggregated data by drilling down into individual months to determine periods of high and low sales revenue, or drilling down into cities to find out if there are marked differences in sales volumes across geographic locations. The results of this analysis can help inform business decisions, such as when to conduct sales promotions or where to open a new store.

- **Predictive analysis and reporting**, which is concerned with detecting data patterns and trends to determine what's likely to happen in the future. For example, a business might use statistics from historical sales data and apply it to known customer profile information to predict which customers are most likely to respond to a direct-mail campaign, or which products a particular customer is likely to want to purchase. This analysis can help improve the cost-effectiveness of the direct-mail campaign, or increase sales while building closer customer relationships through relevant targeted recommendations.

Both kinds of analysis and reporting involve taking source data, applying an analytical model to that data, and using the output to inform business decision making. In the case of historical analysis and reporting, the model is usually designed to summarize and aggregate a large volume of data to determine meaningful business measures (for example the total sales revenue aggregated by various aspects of the business, such as fiscal period and sales region).



Databases are the core of most organizations' data processing, and in most cases the purpose is simply to "run the operation" by, for example, storing and manipulating data to manage stock and create invoices. However, analytics and reporting is one of the fastest growing sectors in business IT as managers strive to learn more about their business.

For predictive analysis, the model is usually based on a statistical algorithm that categorizes clusters of similar data or correlates data attributes that influence one another to cause trends (for example, classifying customers based on demographic attributes, or identifying a relationship between customer age and the purchase of specific products).

The results of the analysis are typically consumed and visualized in the following ways.

- **Custom application interfaces.** For example, a custom application might display the data as a chart or generate a customer recommendation.
- **Business performance dashboards.** For example, you could use the PerformancePoint Services component of SharePoint Server to display key performance indicators (KPIs) as scorecards and summarized business metrics in a SharePoint Server site.
- **Reporting solutions** such as SQL Server Reporting Services or Windows Azure SQL Reporting. For example, business reports can be generated in a variety of formats and distributed automatically by email or viewed on demand through a web browser.
- **Analytical tools such as Microsoft Excel.** Information workers can explore analytical data models through PivotTables and charts, and business analysts can use advanced Excel capabilities such as

Power Pivot and Power View to create their own personal data models and visualizations, or use add-ins to apply predictive models to data and view the results in Excel.

You will see more about the tools for analytics and reporting later in this chapter. Before then you will explore different ways of combining a Big Data mechanism such as HDInsight with your existing business processes and data.

Combining HDInsight with Your Business Processes

Big Data solutions open up new opportunities for turning data into information. They can also be used to extend existing information systems to provide additional insights through analytics and data visualization. Every organization is different, and so there is no definitive list of way that you can use HDInsight as part of your own business processes. However, there are four general architectural models. Understanding these will help you to start making decisions on how best to integrate HDInsight with your organization, and with your existing BI systems and tools. The four different models are:

- **As a data collection, analysis, and visualization tool.** This model is typically chosen for handling data that you cannot process using existing systems. For example, you might collect feedback from customers through email, web pages, or external sources such as Facebook and Twitter, then analyze it to get a picture of user sentiment for your products. You might be able to combine this information with other data, such as demographic data that indicates population density and characteristics in each city where your products are sold.
 - **As a data transfer, data cleansing, and ETL mechanism.** HDInsight can be used to extract and transform data before you load it into your existing databases or data visualization tools. HDInsight solutions are well suited to performing categorization and normalization of data, and for extracting summary results to remove duplication and redundancy. This is typically referred to as an Extract, Transform, and Load (ETL) process.
 - **As a basic data warehouse or commodity storage mechanism.** HDInsight allow you to store both the source data and the results of queries executed over this data. You can also store schemas (or, to be precise, metadata) for tables that are populated by the queries you execute. These tables can be indexed, although there is no formal mechanism for managing key-based relationships between them. However, you can create data repositories that are robust and reasonably low cost to maintain, which is especially useful if you need to store and manage huge volumes of data.
 - **Integration with an enterprise data warehouse and BI system.** Enterprise-level data warehouses have some special characteristics that differentiate them from simple database systems, and so there are additional considerations for integrating with HDInsight. You can also integrate at different levels, depending on the way that you intend to use the data obtained from HDInsight.
-

Each of these basic models encompasses several more specific use cases. For example, when integrating with an enterprise data warehouse and BI system you need to decide at what level (report, corporate model, or data warehouse level) you will integrate the output from HDInsight.

The following sections of this chapter explore the four models listed above in more detail. You will see advice on when to choose each one, how you can use it to provide the information you need, and some of the typical use cases for each one.

The information in this chapter will help you to understand how you can integrate HDInsight with an enterprise BI system. However, a complete discussion of the factors related to the design and implementation of a data warehouse and BI solution is beyond the scope of this guide. Appendix A, “An Overview of Enterprise Business Intelligence” of this guide provides a high-level overview of the key features commonly found in an enterprise BI infrastructure. You will find this useful if you are not already familiar with such systems.

HDInsight as a Data Collection, Analysis, and Visualization Tool

Traditional data storage and management systems such as data warehouses, data models, and reporting and analytical tools provide a wealth of information on which to base business decisions. However, while traditional BI works well for business data that can easily be structured, managed, and processed in a dimensional analysis model, some kinds of analysis require a more flexible solution that can derive meaning from less obvious sources of data such as log files, email messages, tweets, and others.

There’s a great deal of useful information to be found in these less structured data sources, which often contain huge volumes of data that must be processed to reveal key data points. This kind of data processing is what Big Data solutions such as HDInsight were designed to handle. It provides a way to process large volumes of unstructured or semi-structured data to produce an output that can then be visualized directly or combined with other datasets.

If you do not intend to reuse the information from the analysis, but just want to explore the data, you may choose to consume it directly in an analysis or visualization tool such as Microsoft Excel. This standalone model is typically suited to the following scenarios:

- Handling data that you cannot process using existing systems.
- Collecting stream data that arrives at high rate or in large batches without affecting existing database performance.
- Collecting feedback from customers through email, web pages, or external sources such as Facebook and Twitter, then analyzing it to get a picture of user sentiment for your products.
- Combining information with other data, such as demographic data that indicates population density and characteristics in each city where your products are sold.
- Dumping data from your existing information systems so that you can work with it without interrupting other business processes or risking corruption of the original data.
- Trying out new ideas and validating processes before implementing them within the live system.



Combining your data with datasets available from Windows Azure Data Market or other commercial data sources can reveal useful information that might otherwise remain hidden in your data.

Architecture Overview

Figure 1 shows an overview of the architecture for a standalone HDInsight solution. The source data files are loaded into the HDInsight cluster, processed by one or more queries within HDInsight, and the output is consumed by the chosen reporting and visualization tools.

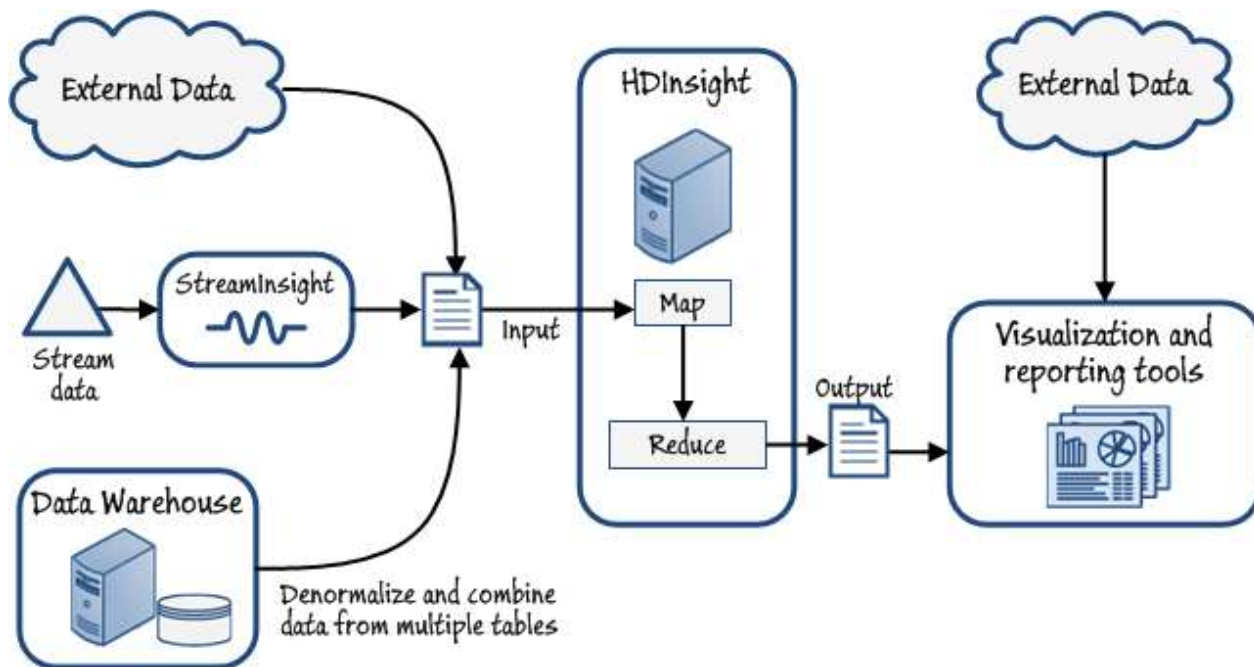


Figure 1

High-level architecture for the standalone data collection, analysis, and visualization model

Text files such as log files and compressed binary files can be loaded directly into the cluster storage, while stream data will usually need to be handled by a suitable mechanism such as StreamInsight. The output data may be combined with other datasets within your visualization and reporting tools to augment the information and provide comparisons, as you will see later in this section of the chapter.

When using HDInsight as a standalone query mechanism, you will often do so as an interactive process. For example, you might use the Data Explorer in Excel to submit a query to an HDInsight cluster and wait for the results to be returned – usually within a few seconds or even a few minutes. You can then modify and experiment with the query to optimize the information it returns.

However, keep in mind that all Big Data operations are batch jobs that are submitted to all of the servers in the cluster for parallel processing, and queries can often take minutes or hours to complete. For example, you might

use Pig to process an input file, with the results returned in an output file some time later—at which point you can perform the analysis by importing this file into your chosen visualization tool.

Data Sources for the Data Collection, Analysis, and Visualization Model

The input data for HDInsight can be almost anything, but for the standalone model they typically include the following:

- Social data, log files, sensors, and applications that generate data files.
- Datasets obtained from Windows Data Market and other commercial data providers.
- Internal data denormalized for experimentation and one-off analysis.
- Streaming data filtered or processed through SQL Server StreamInsight.

Notice that, as well as externally obtained data, Figure 1 includes data from within your organization’s existing database or data warehouse. HDInsight is an ideal solution when you want to perform offline exploration of existing data in a sandbox. For example, you may join several datasets from your data warehouse to create large denormalized datasets to act as the source for some experimental investigation or to test new analysis techniques. This avoids the risk of interrupting existing systems, affecting performance of your data warehouse system, or accidentally corrupting the core data.



Often you need to perform more than one query on the data in HDInsight to get the results into the form you need. It’s not unusual to base queries on the results of a preceding query; for example, using one to select and transform the required data and remove redundancy, one to summarize the data returned from the first query, and another to format the output as required. This iterative approach enables you to start with a large volume of complex and difficult to analyze data, and get it into a structure that you can consume directly from an analytical tool such as Excel—or, as you’ll see later, to use as the input into a managed BI solution.

Output Targets for the Data Collection, Analysis, and Visualization Model

The results from your HDInsight queries can be visualized using any of the wide range of tools that are available for analyzing data, combining it with other datasets, and generating reports. Typical examples for the standalone model are:

- HDInsight interactive reporting and charting.
- Visualization tools such as Excel, Power Pivot, and Power View.
- SQL Server Reporting Services using Report Builder.
- Custom or third party analysis and visualization tools.

You will see more details of these tools in the section “Analysis, Visualization, and Reporting Tools” at the end of this chapter.

Considerations for the Data Collection, Analysis, and Visualization Model

There are some important points to consider when choosing the standalone model for querying and visualizing data with HDInsight.

- This model is typically used when you want to:
 - Experiment with new types or sources of data.
 - Generate one-off reports or visualizations of external or internal data.
 - Monitor a data source using visualizations to detect changes or to predict behavior.
 - Combine with other data to generate comparisons or to augment the information.
 - You will usually choose the standalone model when you do not want to persist the results of the query after analysis, or after the required reports have been generated. It is typically used for one-off analysis tasks where the results are discarded after use; and so differs from the other models described in this chapter, in which the results are stored and reused.
 - Very large datasets are likely to preclude the use of an interactive analysis approach due to the time taken for the queries to run.
 - Data arriving as a stream, such as the output from sensors on an automated production line or the data generated by a GPS sensor in mobile device cannot be analyzed directly. A typical technique is to capture it using a stream processing technology such as StreamInsight, which feeds into an HDInsight cluster where it can be queried interactively or as a batch job at suitable intervals.
 - You are not limited to running a single query on the source data. You can follow an iterative pattern in which the data is passed through HDInsight multiple times, each pass refining the data until it is suitably prepared for use in your analytical tool. For example, a large unstructured file might be processed using custom Map/Reduce components to generate a smaller, more structured output file. This output could then be used in turn as the input for a Hive query that returns aggregated data in tabular form.
-

HDInsight as a Data Transfer, Data Cleansing, and ETL Mechanism

In a traditional business environment the data to power your reporting mechanism will usually come from tables in a database. However, it's increasingly necessary to supplement this with data obtained from outside your organization. This may be commercially available datasets, such as those available from Windows Data Market and elsewhere, or it may be data from less structured sources such as Twitter feeds, emails, log files, and more.

You will, in most cases, need to cleanse, validate, and transform this data before loading it into an existing database. Extract, Transform, and Load (ETL) operations can use Big Data solutions such as HDInsight to perform pattern matching, data categorization, de-duplication, and summary operations on unstructured or semi-structured data to generate data in the familiar rows and columns format that can be imported into a database table.

This data transfer, data cleansing, and ETL model is typically suited to the following scenarios:

- Extracting and transforming data before you load it into your existing databases or data visualization tools.
- Performing categorization and normalization of data, and for extracting summary results to remove duplication and redundancy.



ETL may not seem like the most exciting process in the world of IT, but it's the primary way that data stewards can ensure that the data is valid and contains correct values. ETL gets the data into the correct format for our database tables, while data cleansing is the gatekeeper that protects our tables from invalid, duplicated, or incorrect values.

Architecture Overview

Figure 2 shows an overview of the data transfer, data cleansing, and ETL model. Input data is transformed by HDInsight to generate the appropriate output format and data content, and then imported into the target data store. Analysis and reporting can then be done against the data store, often by combining the imported data with existing data in the data store.

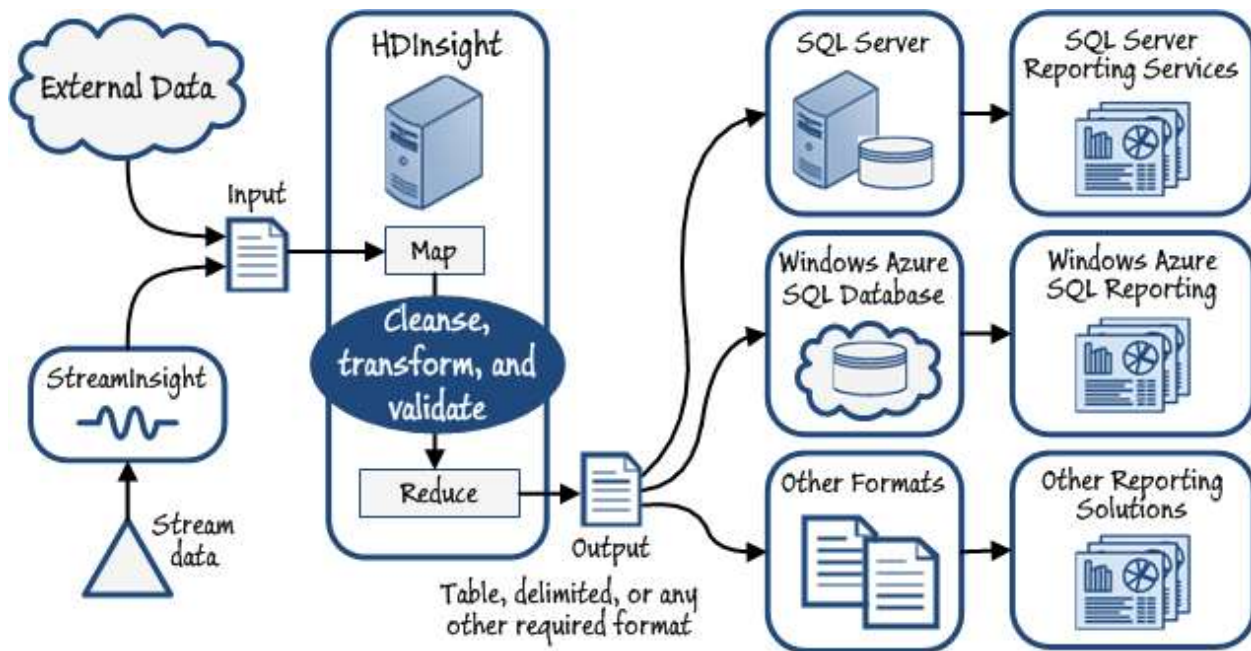


Figure 2
High-level architecture for the data transfer, data cleansing, and ETL model

The transformation process in HDInsight may involve just a single query, but it is more likely to require a multi-step process that uses custom Map and Reduce components or Pig scripts, followed by a Hive query in the final stage to generate the table format. However, the final format for import into a database may be something other than a table; for example, it may be a tab delimited file or some other format suitable for import into the target data store.

Data Sources for the Data Transfer, Data Cleansing, and ETL Model

Data sources for this model are typically external data that can be matched on a key to existing data in your data store so that it can be used to augment the results of analysis and reporting processes. Some examples are:

- Social data, log files, sensors, and applications that generate data files.
 - Datasets obtained from Windows Data Market and other commercial data providers.
 - Streaming data filtered or processed through SQL Server StreamInsight.
-

Output Targets for the Data Transfer, Data Cleansing, and ETL Model

This model is designed to generate output that is in the appropriate format for the target data store. Common types of data store are:

- A database such as SQL Server or Windows Azure SQL Database.
 - SharePoint server or another information management system.
 - A local or remote data repository in a custom format such as JSON objects.
 - Cloud data stores such as Windows Azure table or blob storage.
-

You may decide to use this approach even when you don't actually want to keep the results of the HDInsight query. You can load it into your database, generate the reports and analyses you require, and then delete the data from the database. You may need to do this even when you want to be able to run the report if the source data in HDInsight changes between each reporting cycle.

The results from your HDInsight queries will usually be visualized using the tools that are part of your data store, or are specifically designed to work with it. Typical examples for the data transfer, data cleansing, and ETL model are:

- SQL Server Reporting Services for SQL Server.
 - Windows Azure SQL Reporting for Windows Azure SQL Database.
 - Interactive analytical tools such as Excel, Power Pivot, and Power View
 - Business performance dashboards such as PerformancePoint Services in SharePoint Server
 - Custom or third party analysis and visualization tools
-

You will see more details of these tools in the section "Analysis, Visualization, and Reporting Tools" at the end of this chapter.

Considerations for the Data Transfer, Data Cleansing, and ETL Model

There are some important points to consider when choosing the data transfer, data cleansing, and ETL model.

- This model is typically used when you want to:

- Load stream data or large volumes of semi-structured or unstructured data from external sources into an existing database or information system.
 - Cleanse, transform, and validate the data before loading it.
 - Generate reports and visualizations that are regularly updated.
 - When the output is in table format, such as that generated by Hive, the data import process can use the Hive ODBC driver. Alternatively, you can use SQOOP to connect a relational database such as SQL Server or Windows Azure SQL Database to your HDInsight data store and export the results of a query into your database.
 - If the target for the data is not a database, you can generate a file in the appropriate format within the HDInsight query. This might be tab delimited format, fixed width columns, some other format for loading into Excel or a third-party application, or even for loading into Windows Azure storage through a custom data access layer that you create.
 - If the intention is to regularly update the target table or data store as the source data changes you will probably choose to use an automated mechanism such as a job scheduler to execute the query and data import processes. However, if it is a one-off operation you may decide to execute it interactively only when required.
 - Windows Azure table storage can be used to store table formatted data using a key to identify each row. Windows Azure blob storage is more suitable for storing compressed or binary data generated from the HDInsight query if you want to store it for reuse.
-

HDInsight as a Basic Data Warehouse or Commodity Data Store

Big Data solutions such as HDInsight can provide a robust, high performance, and cost-effective data storage and querying mechanism. Data is divided into blocks across the nodes in the cluster and replicated across the disks. Queries are distributed across the nodes for fast parallel query processing. HDInsight can also save the output from queries on the nodes in the cluster.

This combination of capabilities means that you can use HDInsight as a basic data warehouse. The low cost of storage when compared to most relational database mechanisms that have the same level of reliability also means that you can use HDInsight simply as a commodity storage mechanism for huge volumes of data, even if you decide not to transform it into Hive tables.



If you need to store vast amounts of data, irrespective of the format of that data, a Big Data solution such as HDInsight can reduce administration overhead and save money by minimizing the need for the high performance database servers and storage clusters used by traditional relational database systems. You may even choose to use a cloud hosted Big Data solution in order to reduce the requirements and running costs of on-premises infrastructure.

The data warehouse or commodity data store model is typically suited to the following scenarios:

- Store both the source data and the results of queries executed over this data in the familiar row and column format, using a range of data types for the columns that includes both primitive types (including timestamps) and complex types such as arrays, maps, and structures.
- Store schemas (or, to be precise, metadata) for tables that are populated by the queries you execute.
- Create indexes for tables, and partition tables based on a clustered index so that each has a separate metadata definition and can be handled separately.
- Create views based on tables, and create functions for use in both tables and queries.
- Create data repositories that are robust and reasonably low cost to maintain, which is especially useful if you need to store and manage huge volumes of data.
- Consume data directly from HDInsight in business applications, through interactive analytical tools such as Excel, or in corporate reporting platforms such as SQL Server Reporting Services.
- Store large quantities of data is a robust yet cost-effective way, for use now or in the future,

Architecture Overview

Figure 3 shows an overview of the architecture for the data warehouse or commodity data store model. The source data files may be obtained from external sources, but are just as likely to be internal data generated by your business processes and applications. For example, you might decide to use this model instead of using a locally installed data warehouse based on the traditional relational database model. However, as you will see, there are some limitations when using a Big Data solution such as HDInsight as a data warehouse.

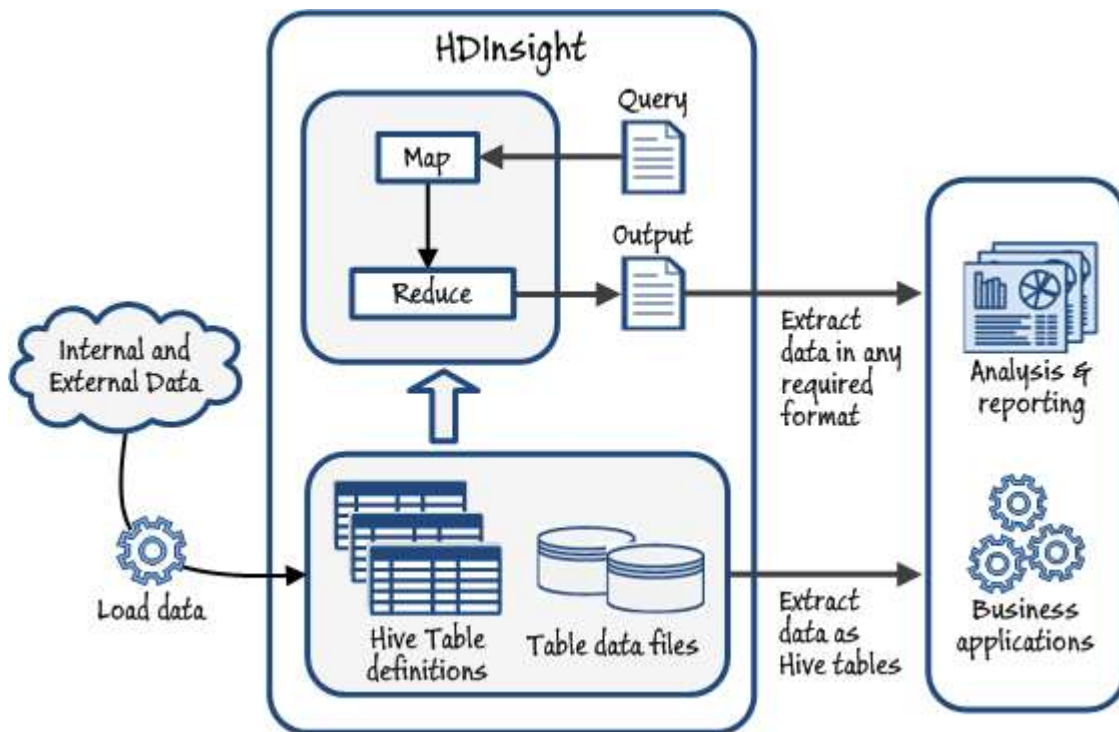


Figure 3
High-level architecture for the data warehouse or commodity data store model

This model is also suitable for use as a data store where you do not need to implement the typical data warehouse capabilities. For example, you may just want to minimize storage cost when saving large tabular format data files for use in the future, large text files such as email archives or data that you must keep for legal or regulatory reasons but you do not need to process, or for storing large quantities of binary data such as images or documents. In this case you simply load the data into the cluster without creating Hive tables for it.

You might also consider storing partly processed data where you have performed some translation or summary of the data but it is still in a relatively raw form that you want to keep in case it is useful in the future. For example, you might use Microsoft StreamInsight to allocate incoming positional data from a fleet of vehicles into separate categories or areas, and add some reference key to each item, then store the results ready for processing at a later date. Stream data typically generates very large files, and may arrive in rapid bursts, so using a separate solution such as HDInsight helps to minimize the load on your existing data management systems.



Big Data storage is robust because the data is replicated at least three times across the nodes of the cluster. It is worth considering Big Data implementations such as HDInsight, and especially cloud hosted implementations, just for storing data without actually processing or querying it. Effectively, use it just as cost-effective network attached storage (NAS).

Data Sources for the Data Warehouse or Commodity Data Store Model

Data sources for this model are typically data collected from internal and external business processes, but may also include reference data and datasets obtained from other sources that can be matched on a key to existing data in your data store so that it can be used to augment the results of analysis and reporting processes. Some examples are:

- Data generated by internal business processes, websites, and applications.
- Reference data and data definitions used by business processes.
- Datasets obtained from Windows Data Market and other commercial data providers.

If you adopt this model simply as a commodity data store rather than a data warehouse, you might also load data from other sources such as social data, log files, and sensors; or streaming data filtered or processed through SQL Server StreamInsight.

Output Targets for the Data Warehouse or Commodity Data Store Model

The main intention of this model is to provide the equivalent to an internal data warehouse system based on the traditional relational database model, and expose it as Hive tables. You can use these tables in a variety of ways, such as:

- By combining the datasets for analysis, and to generate reports and business information

- To generate ancillary information such as related items or recommendation lists for use in applications and websites.
 - For external access through web applications, web services, and other services.
 - To power information systems such as SharePoint server through web parts and the Business Data Connector (BDC).
-

If you adopt this model simply as a commodity data store rather than a data warehouse, you might use the data you store as an input for any of the models described in this chapter.

The data in an HDInsight data warehouse can be analyzed and visualized using any tools that can consume Hive tables. Typical examples are:

- SQL Server Reporting Services
 - SQL Server Analysis Services
 - Interactive analytical tools such as Excel, Power Pivot, and Power View
 - Custom or third party analysis and visualization tools
-

You will see more details of the reporting tools in the section “Analysis, Visualization, and Reporting Tools” at the end of this chapter. The section “Corporate Data Models” later in this chapter discusses SQL Server Analysis Services. You can also download a [case study](#) that describes using SQL Server Analysis Services with Hive.

Considerations for the Data Warehouse or Commodity Data Store Model

There are some important points to consider when choosing the standalone model for querying and visualizing data with HDInsight.

- This model is typically used when you want to:
 - Create a central point for analysis and reporting of big data by multiple users and tools.
 - Store multiple datasets for use by internal applications and tools.
 - Host your data in the cloud to benefit from reliability and elasticity, minimize cost, and reduce administration overhead.
 - Store both externally collected data and data generated by internal tools and processes.
 - Refresh the data at scheduled intervals or on demand.
- You can use Hive in HDInsight to:
 - Define tables that have the familiar row and column format, with a range of data types for the columns that includes both primitive types (including timestamps) and complex types such as arrays, maps, and structures.

- Load data from the file system into tables, save data to the file system from tables, and populate tables from the results of running a query.
- Create indexes for tables, and partition tables based on a clustered index so that each has a separate metadata definition and can be handled separately.
- Rename, alter and drop tables, and modify columns in a table as required.
- Create views based on tables, and create functions for use in both tables and queries.

For more details of how to work with Hive tables, see [Hive Data Definition Language](#) on the Apache Hive website.

- The main limitation of Hive tables compared to a relational database is that you cannot create restraints such as foreign key relationships that are automatically managed.
- Incoming data may be processed by any type of query, not just Hive, to cleanse and validate the data before converting it to table format.
- You can store the Hive queries and views within HDInsight so that they can be used in much the same way as the stored procedures in a relational database to extract data on demand. However, keep in mind that the core Hadoop engine is designed for batch processing by distributing data and queries across all the nodes in the cluster. To minimize response times you will probably need to pre-process the data where possible using queries within HDInsight, and store these intermediate results in order to reduce the time-consuming overhead of complex queries.
- You can use HDInsight ODBC connector in SQL Server to create linked servers. So, theoretically, you could write a Transact-SQL query that joins table in a SQL Server database to tables stored in an HDInsight data warehouse.

HDInsight Integration with Enterprise Data Warehouses and BI

You saw in Chapter 1 that Microsoft offers a set of applications and services to support end-to-end solutions for working with data. The Microsoft data platform includes all of the elements that make up an enterprise level BI system for organizations of any size.

Organizations that already use an enterprise BI solution for business analytics and reporting can extend their analytical capabilities by using HDInsight to add new sources of data to their decision making processes. This model is typically suited to the following scenarios:

- You have an existing enterprise data warehouse and BI system that you want to augment with data from outside your organization.
- You want to explore new ways to combine data to provide better insight into history and to predict future trends.

- You want to give users more opportunities for self-service reporting and analysis that combines managed business data and Big Data from other sources.

Architecture Overview

Enterprise BI is a topic in itself, and there are several factors that require special consideration when integrating a Big Data solution such as HDInsight with an enterprise BI system. If you are not already familiar with enterprise BI systems, it is worthwhile exploring Appendix A, “An Overview of Enterprise Business Intelligence” of this guide. It will help you to understand the main factors that make an enterprise BI system different from simple data management scenarios.

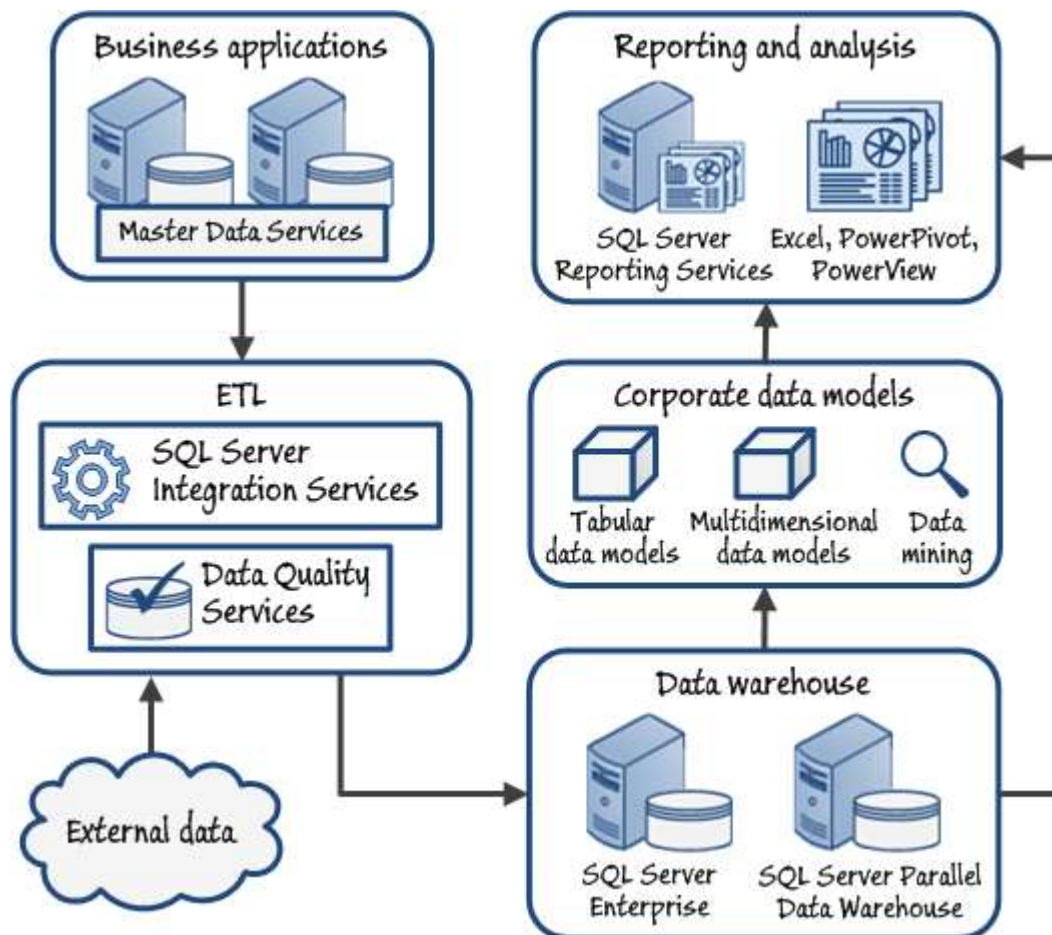


Figure 4

Overview of a typical enterprise data warehouse and BI implementation

Figure 4 shows an overview of a typical enterprise data warehouse and BI solution. Data flows from business applications and other external sources through an ETL process and into a data warehouse. Corporate data models are then used to provide shared analytical structures such as online analytical processing (OLAP) cubes or data mining models, which can be consumed by business users through analytical tools and reports. Some BI

implementations also enable analysis and reporting directly from the data warehouse, enabling advanced users such as business analysts and data scientists to create their own personal analytical data models and reports.

There are many technologies and techniques that you can use to combine HDInsight and existing BI technologies, but as a general guide there are three primary ways in which HDInsight can be used with an enterprise BI solution. Figure 5 shows an overview of these three levels of integration.

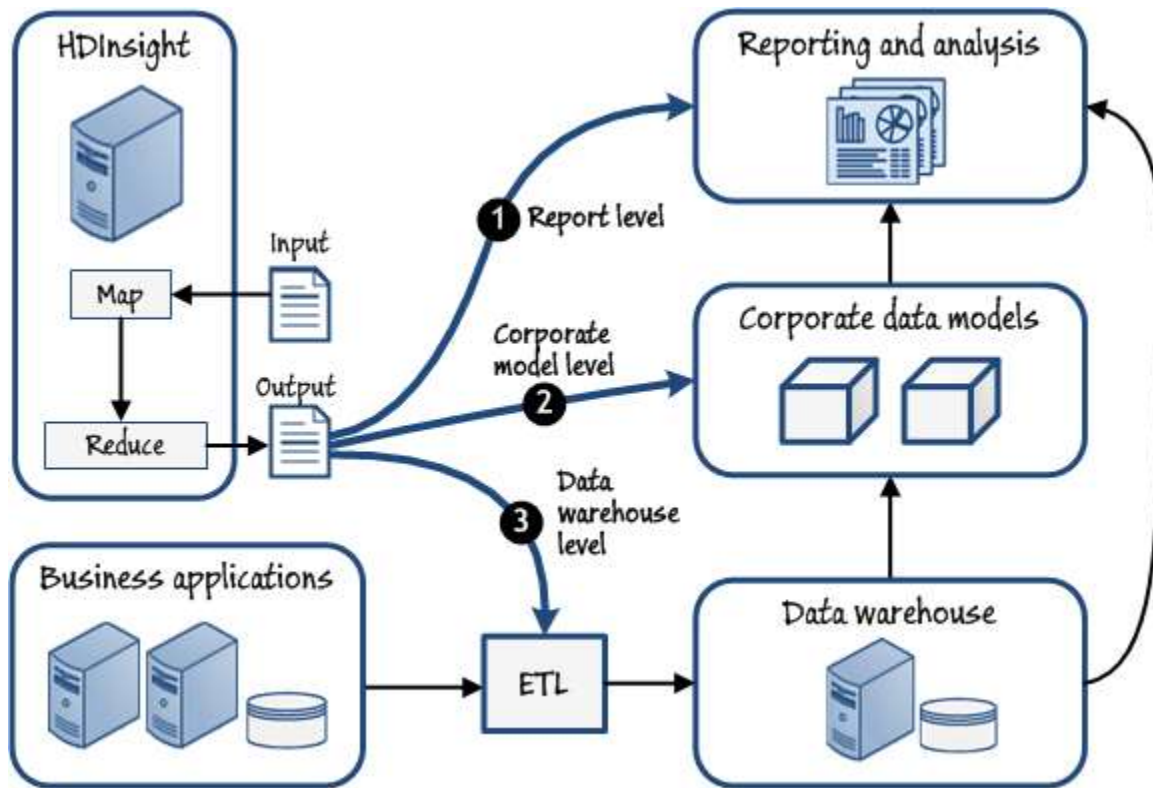


Figure 5
Three levels of integration for HDInsight with an enterprise BI system

The three integration levels shown in Figure 5 are:

1. **Report level integration.** Data from HDInsight is used in reporting and analytical tools to augment data from corporate BI sources, enabling the creation of reports that include data from corporate BI sources as well as HDInsight, and also enabling individual users to combine data from both solutions into consolidated analyses. This level of integration is typically used for creating mash ups, exploration of datasets and query capabilities, and for generating one off reports and visualizations.
2. **Corporate data model level integration.** HDInsight is used to process data that is not present in the corporate data warehouse, and the results of this processing are then added to corporate data models where they can be combined with data from the data warehouse and used in multiple corporate reports and analysis tools. This level of integration is typically used for exposing the data in specific formats to information systems, and for use in reporting and visualization tools.

3. **Data warehouse level integration.** HDInsight is used to prepare data for inclusion in the corporate data warehouse. The HDInsight data that has been loaded is then available throughout the entire enterprise BI solution. This level of integration is typically used to create standalone tables on the same database hardware as the enterprise data warehouse, which provides a single source of enterprise data for analysis, or to incorporate the data into a dimensional schema and populate dimension and fact tables for full integration into the BI solution.

The following sections describe these three integration levels in more detail to help you understand the implications of your choice. They also contain guidelines for implementing each one. However, keep in mind that you don't need to make just a single choice for all of your processes; you can use a different approach for each dataset that you extract from HDInsight, depending on the scenario and the requirements for that dataset.



Choose a suitable integration level for each dataset or data querying operation you carry out. You aren't limited to using the same one for every task that interacts with your Big Data solution.

Report Level Integration

Using HDInsight in a standalone non-integrated way, as described earlier in this chapter, can unlock information from currently unanalyzed data sources. However, much greater business value can be obtained by integrating the results from HDInsight queries with data and BI activities that are already present in the organization.

In contrast to the rigidly defined reports created by database administrators, a growing trend in BI is a “self-service” approach. In this approach the data warehouse provides some datasets based on data models and queries defined there, but the user selects the datasets and builds a custom report. The ability to combine multiple data sources in a personal data model enables a more flexible approach to data exploration that goes beyond the constraints of a formally managed corporate data warehouse. Users can augment reports and analysis of data from the corporate BI solution with additional data from HDInsight to create a mash up solution that brings data from both sources into a single, consolidated report.

You can use the following techniques to integrate output from HDInsight with enterprise BI data at the report level.

- Download the output files generated by HDInsight and open them in Excel, or import them into a database for reporting.
- Create Hive tables in HDInsight and consume them directly from Excel (including using Power Pivot) or from SQL Server Reporting Services (SSRS) by using the ODBC driver for Hive.
- Use Sqoop to transfer the results from HDInsight into a relational database for reporting. For example, copy the output generated by HDInsight to a Windows Azure SQL Database table and use Windows Azure SQL Reporting Services to create a report from the data.
- Use SQL Server Integration Services (SSIS) to transfer and, if required, transform HDInsight results to a database or file location for reporting. If the results are exposed as Hive tables you can use an ODBC

data source in an SSIS data flow to consume them. Alternatively, you can create an SSIS control flow that downloads the output files generated by HDInsight and uses them as a source for a data flow.

Corporate Data Model Level Integration

Integration at the report level makes it possible for advanced users to combine data from HDInsight with existing corporate data sources to create data mash ups. However, there may be some scenarios where you need to deliver combined information to a wider audience, or to users who do not have the time or ability to create their own complex data analysis solutions. Alternatively, you might want to take advantage of the functionality available in corporate data modeling platforms such as SQL Server Analysis Services (SSAS) to add value to the big data insights gained from HDInsight.



Integrating the output from HDInsight with your corporate data models allows you to use tools such as SQL Server Analysis Services to analyze the data and present it in a format this is easy for users who want to include it in reports, or perform deeper analysis on it.

By integrating data from HDInsight into corporate data models you can accomplish both of these aims, and use the data as the basis for enterprise reporting and analytics. You can use the following techniques to integrate HDInsight results into a corporate data model.

- Create Hive tables in HDInsight and consume them directly from a SQL Server Analysis Services (SSAS) tabular model by using the ODBC driver for Hive. SSAS in tabular mode supports the creation of data models from multiple data sources, and includes an OLE DB provider for ODBC, which can be used as a wrapper around the ODBC driver for Hive.
- Create Hive tables in HDInsight and then create a linked server in the instance of the SQL Server database source used by an SSAS multidimensional data model so that the Hive tables can be queried through the linked server and imported into the data model. SSAS in multidimensional mode can only use a single OLE DB data source, and the OLE DB provider for ODBC is not supported.
- Use Sqoop or SSIS to copy the data from HDInsight to a SQL Server database engine instance that can then be used as a source for an SSAS tabular or multidimensional data model.

You must choose between the multidimensional and the tabular data model when you install SQL Server Analysis Services, though you can install two instances if you need both models.

Data Warehouse Level Integration

Integration at the corporate data model level enables you to include data from HDInsight in a data model that is used for analysis and reporting by multiple users, without changing the schema of the enterprise data warehouse. However, in some cases you might want to use HDInsight as a component of the overall ETL solution used to populate the data warehouse, in effect using a source of Big Data that is queried through HDInsight just

like any other business data source in the BI solution, and consolidating the data from all sources to an enterprise dimensional model.



Full integration with a data warehouse at the data level allows HDInsight to become just another data source that is consolidated into the system, and can be used in exactly the same way as data from other sources. In addition, as with other data sources, it's likely that the data import process from HDInsight into the database tables will occur on a schedule, depending on the time taken to execute the queries and perform ETL tasks prior to loading it into the database tables, so that the data is as up to date as possible.

You can use the following techniques to integrate data from HDInsight into an enterprise data warehouse.

- Use Sqoop or the HDInsight connectors for SQL Server to copy data from HDInsight directly into a table in a SQL Server data warehouse, without integrating these tables into the dimensional model of the data warehouse.
- Use Sqoop or the HDInsight connectors for SQL Server to copy data from HDInsight to a staging database where it can be validated, cleansed, and conformed to the dimensional model of the data warehouse before being loaded into the fact and dimension tables.
- Create an SSIS package that reads the output file from HDInsight, or uses the ODBC driver for Hive to extract data from HDInsight, and then validates, cleanses, and transforms the data before loading it into the fact and dimension tables in the data warehouse.

For more information about enterprise BI systems, data warehousing, and dimensional models see Appendix A, "An Overview of Enterprise Business Intelligence."

Data Sources for the HDInsight Integration with Enterprise BI Model

The input data for HDInsight can be almost anything, but for the HDInsight integration with enterprise BI model they typically include the following:

- Social data, log files, sensors, and applications that generate data files.
- Datasets obtained from Windows Data Market and other commercial data providers.
- Streaming data filtered or processed through SQL Server StreamInsight.

Output Targets for the HDInsight Integration with Enterprise BI Model

The results from your HDInsight queries can be visualized using any of the wide range of tools that are available for analyzing data, combining it with other datasets, and generating reports. Typical examples for the HDInsight integration with enterprise BI model are:

- SQL Server Reporting Services.
- SharePoint server or another information management system.

- Business performance dashboards such as PerformancePoint Services in SharePoint Server.
- Interactive analytical tools such as Excel, Power Pivot, and Power View.
- Custom or third party analysis and visualization tools.

You will see more details of these tools in the section “Analysis, Visualization, and Reporting Tools” at the end of this chapter.

Considerations for the HDInsight Integration with Enterprise BI Model

There are some important points to consider when choosing the HDInsight integration with enterprise BI model for querying and visualizing data with HDInsight.

- This model is typically used when you want to:
 - Integrate external data sources with your enterprise data warehouse.
 - Augment the data in your data warehouse with external data.
 - Update the data at scheduled intervals or on demand.
- ETL processes in a data warehouse usually execute on a scheduled basis to add new data to the warehouse. If you intend to integrate HDInsight into your data warehouse so that it updates the information stored there, you must consider how you will automate and schedule the tasks of executing the HDInsight query and importing the results.
- You must ensure that data imported from HDInsight contains valid values, especially where there are typically multiple common possibilities (such as in street addresses and city names). You may need to use a data cleansing process such as Data Quality Services to force such values to the correct leading value.
- Most data warehouse implementations use slowly changing dimensions to manage the history of values that change over time. This often includes the values used as row keys, and so you must ensure that data imported into the data warehouse tables uses the correct surrogate key value. This means that you must:
 - Include the alternate key in the data model and use some complex logic to join the tables. If you simply join on the alternate key, some loss of data accuracy may occur.
 - Load the data into the data warehouse and conform it to the dimensional data model, including setting the correct surrogate key values.

For more information about leading values and surrogate keys in an enterprise BI system see Appendix A, “An Overview of Enterprise Business Intelligence.”

The following table summarizes the key considerations for deciding whether to adopt one of the three integration approaches described in this chapter. Each approach has its own benefits and challenges.

Integration Level	Typical Scenarios	Considerations
None	<p>No enterprise BI solution currently exists.</p> <p>The organization wants to evaluate HDInsight and Big Data analysis without affecting current BI and business operations.</p> <p>Business analysts in the organization want to explore external or unmanaged data sources that are not included in the managed enterprise BI solution.</p>	<p>Open-ended exploration of unmanaged data might produce a lot of business information, but without rigorous validation and cleansing the data might not be completely accurate.</p> <p>Continued experimental analysis may become a distraction from the day-to-day running of the business, particularly if the data being analyzed is not related to core business data.</p>
Report	<p>A small number of individuals with advanced self-service reporting and data analysis skills need to augment corporate data that is available from the enterprise BI solution with big data from other sources.</p> <p>A single report combining corporate data and some external data is required for one-time analysis.</p>	<p>It can be difficult to find common data values on which to join data from multiple sources to gain meaningful comparisons.</p> <p>Integrated data in a report can be difficult to share and reuse in different ways, though the ability to share PowerPivot workbooks in SharePoint Server may provide a solution for small to medium sized groups of users.</p>
Corporate Data Model	<p>A wide audience of business users must consume multiple reports that rely on data from both the corporate data warehouse and HDInsight. These users may not have the skills necessary to create their own reports and data models.</p> <p>Data from HDInsight is required for specific business logic in a corporate data model that is used for reports and dashboards.</p>	<p>It can be difficult to find common data values on which to join data from multiple sources.</p> <p>Integrating data from HDInsight into tabular SSAS models can be accomplished easily through the ODBC driver for Hive. Integration with multidimensional models is more challenging.</p>
Data Warehouse	<p>The organization wants a complete, managed BI platform for reporting and analysis that includes business application data sources as well as Big Data sources.</p> <p>The desired level of integration between Big Data from HDInsight and existing business data, and the required tolerance for data integrity and accuracy across all data sources, necessitates a formal dimensional model with conformed dimensions.</p>	<p>Data warehouses typically have demanding data validation requirements.</p> <p>Loading data into a data warehouse that includes slowly changing dimensions with surrogate keys can require complex ETL processes.</p>



One of the trickiest parts of full data warehouse integration is matching row keys for data imported from a Big Data solution. Not only must the key be in the correct format, it must also match the surrogate key. This key can differ based on the date that changes were made to the original entity. For example, a product may have more than one surrogate key over its lifetime, and the imported data must match the correct version of this key.

Analysis, Visualization, and Reporting Tools

The goal of any BI solution is to deliver information to business users through reporting and analytics. This information can be delivered in many ways, and through many tools. In most enterprise BI solutions, a combination of pre-prepared reports and interactive data analysis is used to deliver information to users, and tools such as SQL Server Reporting Services and Microsoft Excel are often used to accomplish this. Additionally, many organizations use SharePoint Server as a platform for publishing reports and sharing Excel workbooks.

There is a huge range of data analysis and BI tools available from Microsoft and from third parties that you can use with your data warehouse and HDInsight systems. Microsoft products include server-based solutions such as SQL Server Reporting Services and SharePoint Server, cloud-based solutions such as Windows Azure SQL Reporting, and desktop or Office applications such as Excel and Word.

This section briefly describes the analysis, visualization, and reporting tools listed in the previous sections of this chapter. It focuses on how they can be used in the context of the different models you have seen for combining HDInsight with your business processes.

HDInsight Interactive Reporting and Charting

The HDInsight JavaScript page of the interactive console allows you to run queries and issue commands using JavaScript. It also includes a JavaScript graphics library that you can use to create a range of different types of charts directly within the console pane. This provides an opportunity for quickly reviewing the output from a query to confirm it contains the dataset you require.

Figure 6 shows a simple example of charting hits against date for a set of web server log files.

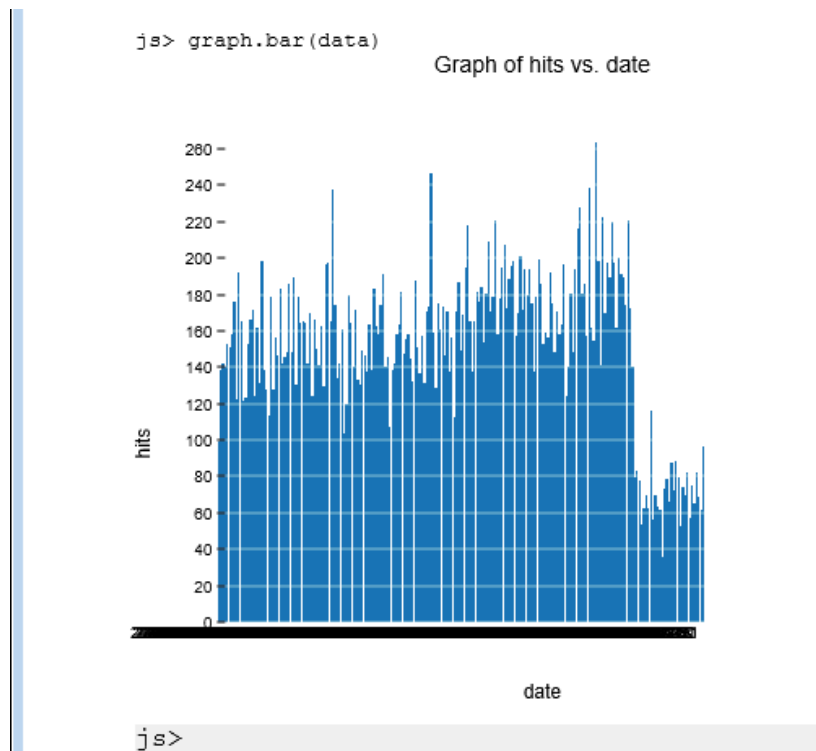


Figure 6

A bar chart generated in the HDInsight interactive console

Interactive Analytical Tools such as Excel, Power Pivot, and Power View

Microsoft Excel is one of the most widely used software applications in the world, and is commonly used as a tool for interactive data analysis and reporting. It supports a wide range of data import and connectivity options, including Data Explorer that connects to a Hive query in HDInsight using ODBC.

Excel also enables users to easily create PivotTables, and Power View reports and charts, from the results of analyzing large datasets. These tools are especially useful when you add value and insight by augmenting the results of your data analysis with external data. For example, you may perform an analysis of sentiment data based on a specific product by searching social media sites, reviews on the Internet, and feedback from customers to discover the popularity of a product in different areas. This geographically oriented data can be enhanced by subscribing to a demographic dataset in the Windows Azure Data Market from which socio-economic and population data may provide an insight into why some products sell particularly well in some locations but not others.

As well as datasets that you can download and use to augment your results, the Windows Azure Data Market includes a number of data services that you can use for data validation (for example, verifying that telephone numbers and postal codes are valid) and for data transformation (for example, looking up the country, state, and city for a particular IP address or longitude/latitude value).

Power Pivot

The growing awareness of the value of decisions based on proven data, combined with advances in data analysis tools and techniques, has resulted in an increased demand for versatile analytical data models that support ad-hoc analysis (the “self-service” approach). Power Pivot is an Excel-based technology that enables advanced users to create their own personal tabular data models and use them for analysis.

While corporate data models are formally designed and based on rigorously validated data (usually in the data warehouse), Power Pivot models can be based on data from many sources – including the output from HDInsight, “unmanaged” sources such as tables in a spreadsheet, and external data from the Windows Azure Data Market. Figure 7 shows a Power Pivot data model that combines sales data with demographic data from the Windows Azure Data Market.

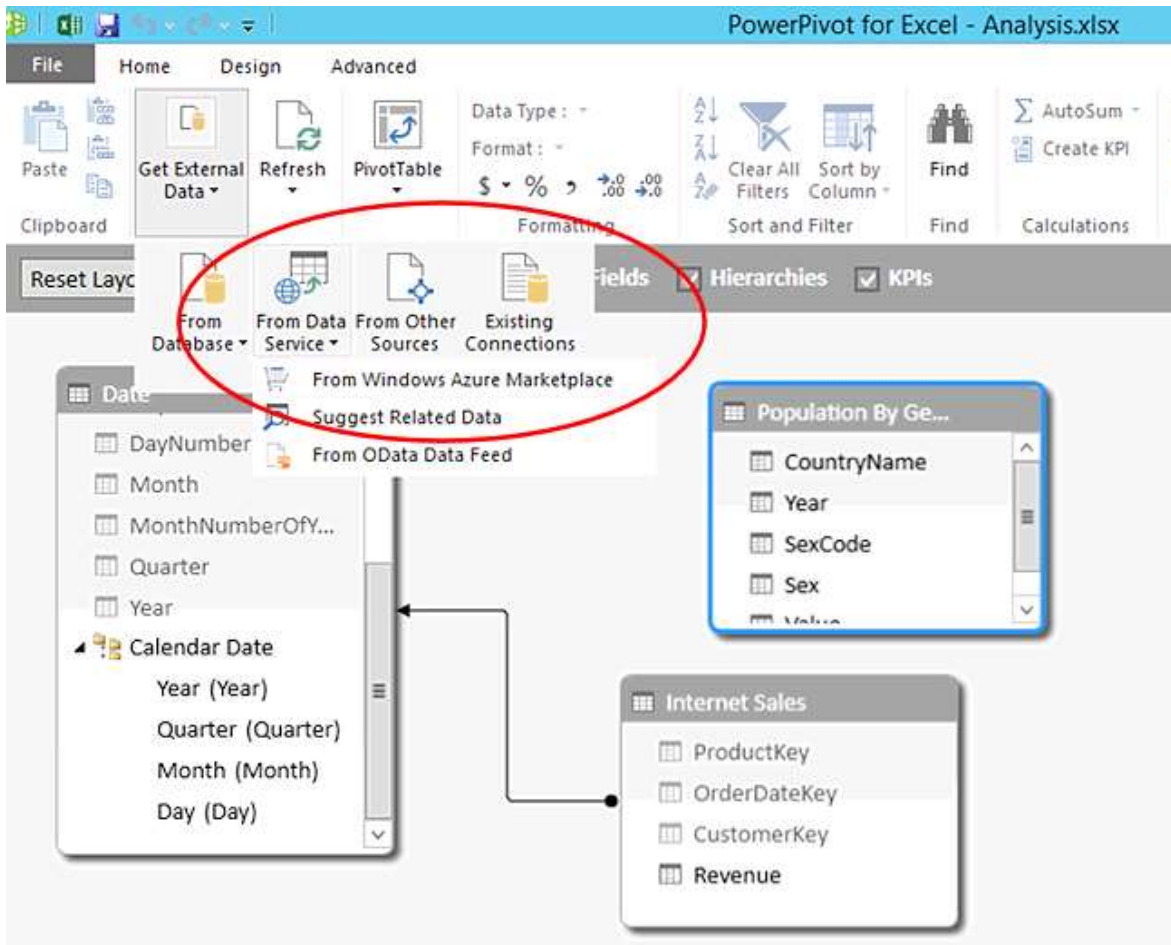


Figure 7
A Power Pivot data model

Power View

Power View is a data visualization technology that enables interactive, graphical exploration of data in a data model. Power View is available as a component of SQL Server 2012 Reporting Services when integrated with SharePoint Server, but is also available in Excel 2013. By using Power View, users can create reports that include multiple, interactive charts that make it easy to see relationships and trends in the data. Figure 8 shows how Power View can be used to visualize sales data in conjunction with demographic data from Windows Data Market.

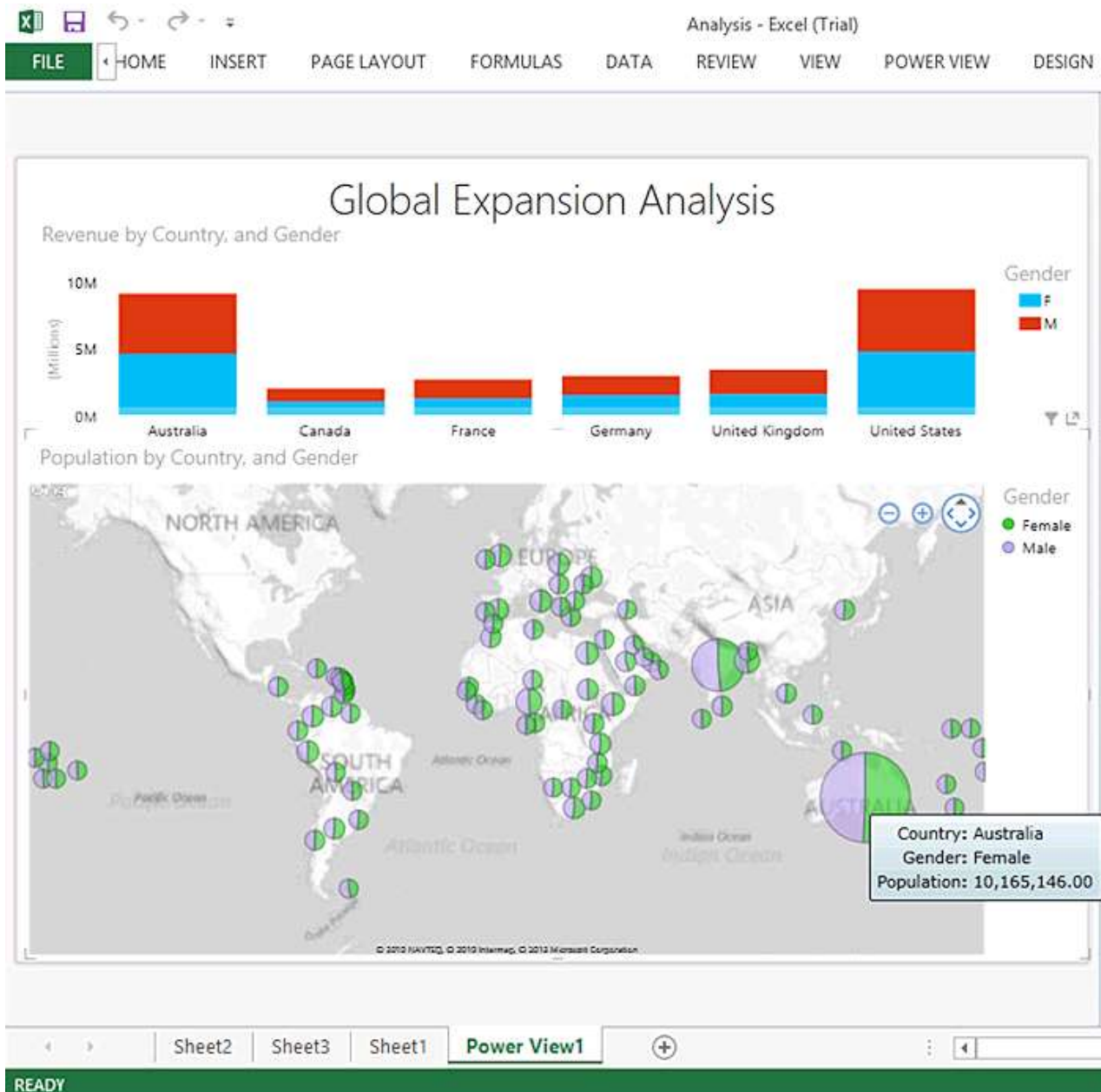


Figure 8
Visualizing data with Power View

SQL Server Reporting Services

SQL Server 2012 Reporting Services (SSRS) provides a platform for delivering reports that are based on data in data models or in data warehouse tables. Typically, standard business reports are authored by BI developers and published to a report server, though increasingly organizations are empowering advanced users such as business analysts to create their own reports in a self-service reporting environment. Figure 9 shows a report that is based on retail sales data delivered through SSRS.

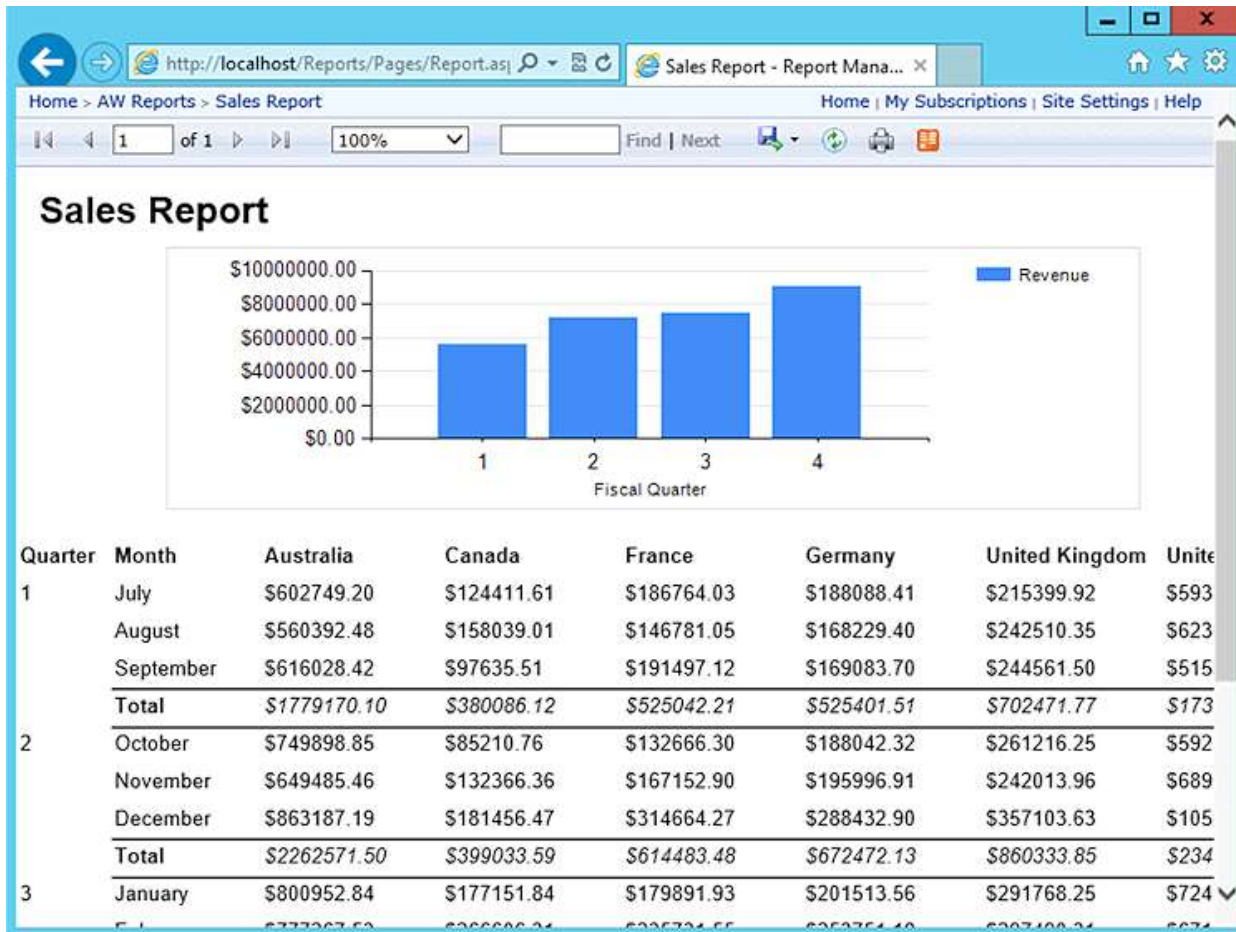


Figure 9

A SQL Server Reporting Services report

One of the key advantages of using SSRS is that reports can be exported to multiple formats, including Excel workbooks, Word documents, portable document format (PDF) files, and others. Windows Azure SQL Reporting Services offers similar capabilities in the cloud, with the restriction that report data can only be accessed from Windows Azure SQL Database. Additionally, an on-premise installation of SSRS enables the automatic distribution of reports by email.

Windows Azure SQL Reporting

If you have loaded your results from an HDInsight query into Windows Azure SQL Database you can use SQL Azure Reporting to generate a similar set of reports and charts to those available in SSRS. Some features of SSRS are not supported in Windows Azure SQL Reporting; for example, you cannot schedule report creation, use any data source other than SQL Database, send reports by email, or use customized report extensions. However, you can create reports in multiple formats for rendering in a browser, Report Viewer, and custom applications built with Windows Forms or other technologies.

For a comparison of SSRS and SQL Reporting see [Guidelines and Limitations for Windows Azure SQL Reporting](#).

Information Management Systems

In a business environment you may want to share information and display reports and charts over an internal network, or make them available publicly over the Internet. Many information management systems allow you to connect to a data source and assemble pages that contain data extracted from that data source.

For example, you can use the Business Data Connector (BDC) in SharePoint Server to connect to a data source and display the information in a variety of ways. You can also create web parts that consume the data. Typically you will import the data from HDInsight into a database and then connect to that database from SharePoint Server.

Business Performance Dashboards

Business decision makers often want to see information displayed in a dashboard style that combines data from several sources and presents an overview of all the relevant information. Typically the dashboard will also support the user drilling down into individual parts of the display to investigate individual items in more detail.

PerformancePoint Services, a component of SharePoint Server, provides an extensible architecture for monitoring data sources, filtering the data, generating scorecards and key performance indicators, and rendering information in the appropriate format. It will usually take the source data from the SharePoint data store, which can be populated from the results of HDInsight queries and from other business data sources.

Custom or Third Party Analysis and Visualization Tools

The output from an HDInsight query may be a text file, which you can open and view in any text editor. However, to make sense of the data you will usually want to generate more intuitive visualizations such as charts and reports. If you use Hive as the final step in your query chain the output is a table. In HDInsight you can view and analyze the data in this table using any tool that supports ODBC connections. Alternatively, you can write the query so that it generates a tab (or other character) delimited file and import this into any suitable visualization tool.

There are also many graphic libraries such as [D3.js](#) available for building your own data visualization tools and web applications.

Useful lists of analysis and visualization tools can be found at [Datavisualization.ch](#) (see [A Carefully Selected List of Recommended Tools](#)) and at [ComputerWorld](#) (see [30+ free tools for data visualization and analysis](#)).

Summary

In this chapter you have seen the main models, or general use cases, for using HDInsight as part of your business information and data management systems. There are, of course, many specific use cases that fall within those identified in this chapter. However, all are subcases of the models you have seen in this chapter, which are:

- As a data collection, analysis, and visualization tool for handling data that you cannot process using existing systems.

- As a data transfer, data cleansing, and ETL mechanism to extract and transform data before you load it into your existing databases or data visualization tools.
 - As a basic data warehouse or commodity storage mechanism to store both the source data and the results of queries executed over this data, or as a data repository that is robust and reasonably low cost to maintain.
 - Integration with an enterprise data warehouse and BI system at different levels, depending on the way that you intend to use the data obtained from HDInsight.
-

Each model has different advantages and suits different scenarios. There is no single approach that is best, and you must decide based on your own circumstances, the existing tools you use, and the ultimate aims you have for your Big Data implementation. There's nothing to prevent you from using it in all of the ways you have seen in this chapter!

You have also seen how analytics and data visualization are vitally important parts of most organizations' IT systems. Organizations typically require their databases to power the daily processes of taking orders, generating invoices, paying employees, and a myriad other tasks. However, accurate reporting information and analysis of the data the organization holds and collects from external sources is becoming equally vital to maintain competitiveness, minimize costs, and successfully exploit business opportunities.

Scenario 1: Basic Twitter Analysis

In this scenario, HDInsight is used to perform some basic analysis of data from Twitter. Social media analysis is a common Big Data use case, and this scenario demonstrates how to extract information from semi-structured data in the form of tweets. However, the goal of this scenario is not to provide a comprehensive guide to analyzing data from Twitter, but rather to show the entire Big Data process from start to finish, and demonstrate some of the techniques discussed in Chapter 1 of this guide. Specifically, this scenario demonstrates how to:

- Identify a source of data and define analytical goals.
 - Provision an HDInsight on Windows Azure cluster, and configure it to use the Azure Blob Store.
 - Ingest data into HDInsight interactively, by using SQL Server Integration Services, and by using StreamInsight.
 - Process data by using custom map/reduce code, Pig scripts, and Hive.
 - Review data processing results.
 - Tune data processing performance.
-

Introduction to Blue Yonder Airlines

This scenario is based on a fictitious company named *Blue Yonder Airlines*. The company is an airline serving passengers in the USA, and operates flights from its home hub at JFK airport in New York to Sea-Tac airport in Seattle and LAX in Los Angeles. The company has a customer loyalty program named *Blue Yonder Points*, and has recently started to use Twitter as a means of communicating with its customers. Customers send tweets to *@BlueYonderAirlines* and use the standard Twitter convention of including hashtags to denote key terms in their messages.

Analytical Goals and Data Sources

Since starting to use Twitter as a means of communication with passengers, customer service managers have observed a significant growth in the volume of messages exchanged with customers. Although there is no specific business decision under consideration, the customer services managers believe that some analysis of the tweets sent by customers may reveal important information about the issues that matter to customers and how they perceive the airline.

To support this hypothesis, business analysts at Blue Yonder Airlines intend to use the Twitter public APIs to collect daily samples of tweets that mention *@BlueYonderAirlines*, and load them to an HDInsight cluster for analysis.

Initially, the analytical goal is to explore the Twitter data and try to determine common hashtags and trends in the tweets sent by customers.

Twitter provides a number of public APIs that you can use to obtain data for analysis.

The Twitter REST-based search API provides a simple way to search for tweets that contain specific query terms. To return a collection of tweets that contain a search phrase, you simply specify a parameter named **q** with the appropriate search string, and optionally an **rpp** parameter to specify the number of tweets to be returned per page. For example, the following request returns up to 100 tweets containing the term “@BlueYonderAirlines” in Atom format.

`http://search.twitter.com/search.atom?q=@BlueYonderAirlines&rpp=100`

You can also request the results in RSS format, as shown in the following request:

`http://search.twitter.com/search.rss?q=@BlueYonderAirlines &rpp=100`

You can restrict the range of results by including **since** or **until** operators in the query term. For example, the following request returns tweets after 0:00 on March 1st 2013:

`http://search.twitter.com/search.rss?q=@BlueYonderAirlines since:2013-03-01&rpp=100`

For more sophisticated filtering, you can specify a **since_id** parameter with the ID of a tweet (usually the ID of the most recent tweet provided in the Atom or RSS response from a previous request).

If you want to analyze customer sentiment, the Twitter API includes semantic logic to categorize tweets and “positive” or “negative” based on key terms used in the tweet content. You can filter the results of a search based on sentiment by including an emoticon in the search string. For example, the following request returns up to 100 tweets that mention “@BlueYonderAirlines” and are positive in sentiment:

`http://search.twitter.com/search.rss?q=@BlueYonderAirlines :)&rpp=100`

Similarly, the following request returns tweets that are negative in sentiment:

`http://search.twitter.com/search.rss?q=@BlueYonderAirlines :(&rpp=100`

In addition to the REST API, the Twitter Streaming API offers a way to consume an ongoing real-time stream of tweets based on filtering criteria.

For more information about Twitter APIs, see [the developer documentation on the Twitter web site](#).

HDInsight Infrastructure

The IT department at Blue Yonder Airlines does not have the capital expenditure budget to purchase new servers for the Big Data project, especially as the hardware will only be required for the duration of the project. The most appropriate infrastructure is therefore to provision an HDInsight cluster on Windows Azure, so that the cluster can be released when no longer required and minimize the overall cost of the project.

To provision the cluster, a member of the IT department registers for the HDInsight on Windows Azure service at www.hadooponazure.com and creates the cluster named **blueyondr.cloudapp.net**. The default dashboard page for the cluster is shown in Figure 1.

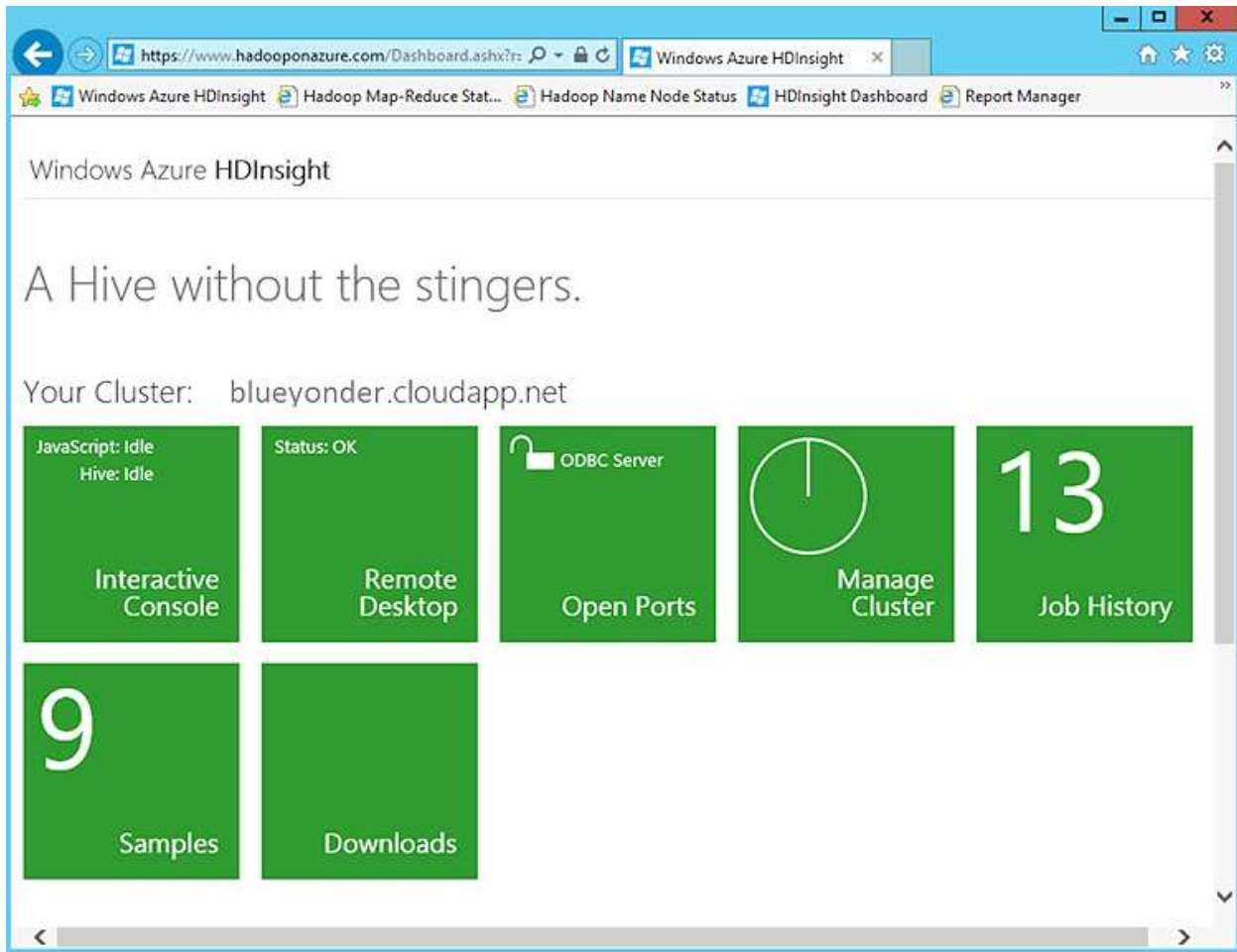


Figure 1
The HDInsight on Windows Azure Dashboard

Data Storage

The Twitter data will be gathered over a period of weeks to ensure a suitable sample for analysis, and the results will be retained after processing for visualization and exploration in Excel. To reduce the cost of HDInsight cluster resources, the source data and results will be stored in an Azure Blob Store. This ensures that the data is retained, even when the HDInsight cluster is released. The storage account is created in the Windows Azure management portal and named **BlueYonderBlobStore** as shown in Figure 2, and from here the keys used to access it can be displayed by clicking **Manage Keys**.

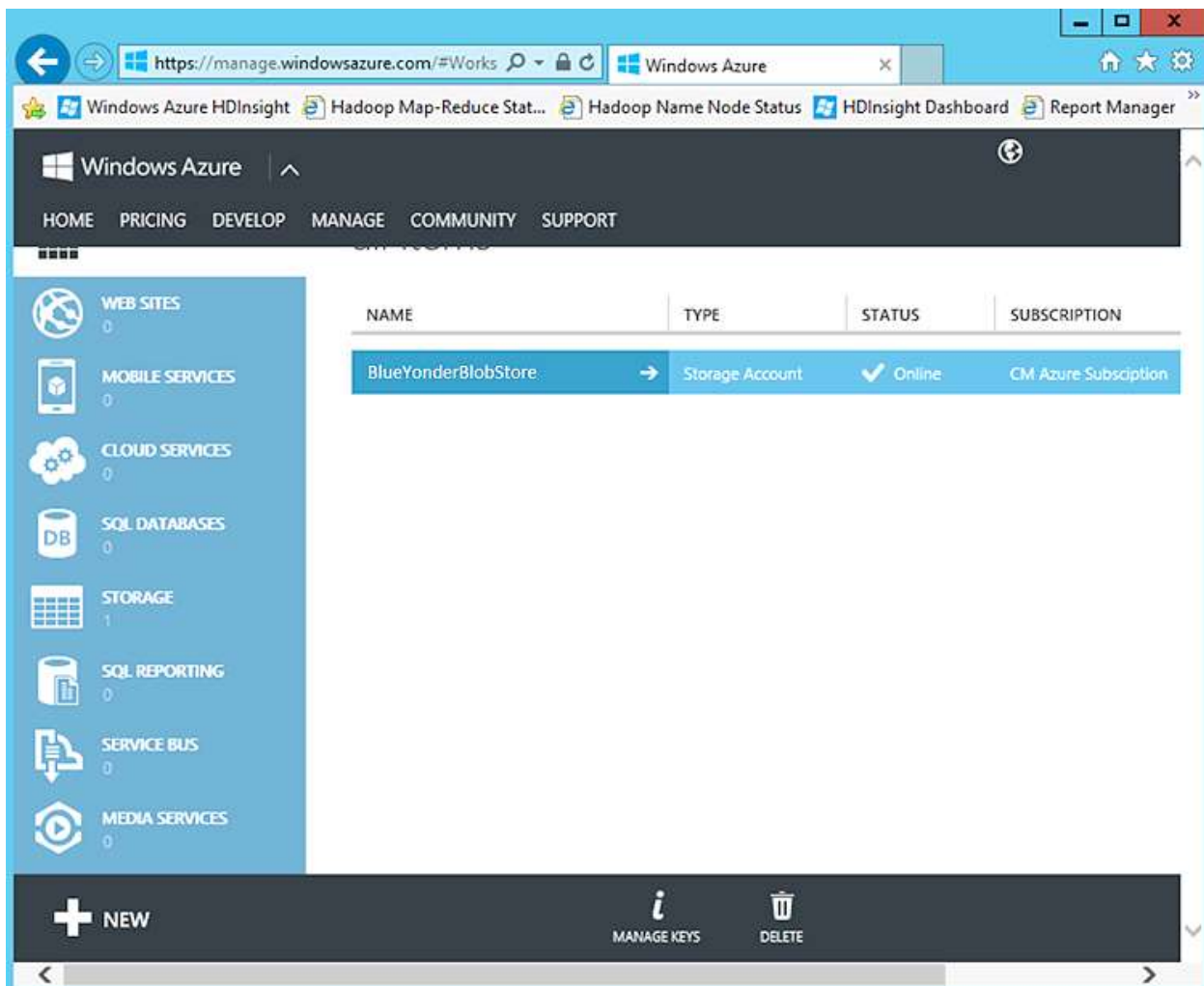


Figure 2
Configuring a storage account in Windows Azure

To enable the HDInsight cluster to use the Azure Blob Store as an Azure storage volume, the administrator clicks Setup ASV in the **Manage Cluster** section of the HDInsight dashboard and enters the storage account name and passkey, as shown in Figure 3.

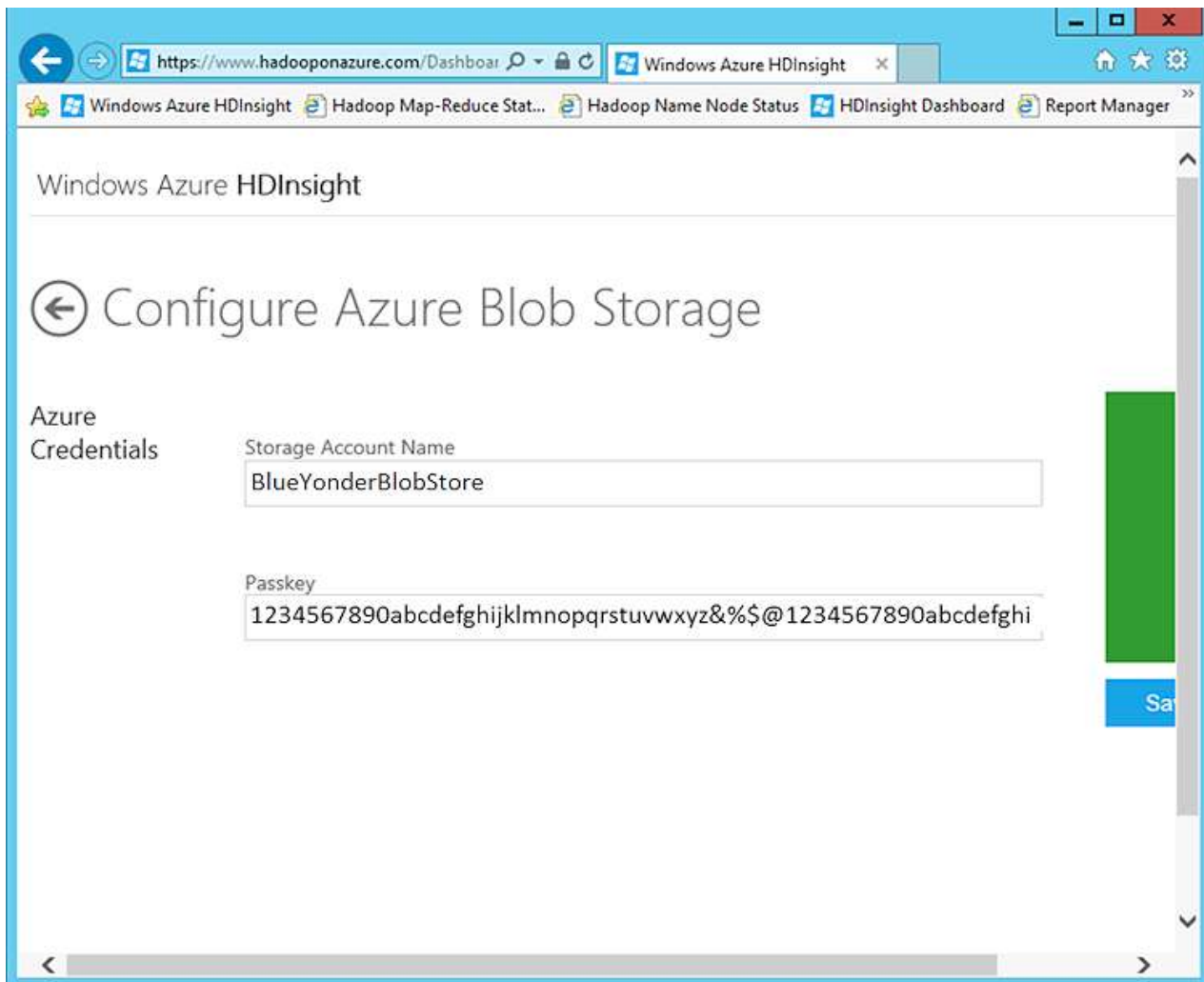


Figure 3
Configuring ASV storage for HDInsight

Data Ingestion

After the HDInsight cluster has been provisioned and its storage configured, the team can begin obtaining data from Twitter and uploading it to the Azure Blob Store, ready to be processed by HDInsight.

Interactive Data Ingestion

During the initial stage of the project, the team has decided to create a proof of concept with a small volume of data, which they can obtain interactively in Excel and upload to HDInsight through the interactive console.

To use Excel to extract a collection of tweets from Twitter, on the **Data** tab of the ribbon, you need to click **Get External Data**, and then in the drop-down list, click **From Web**. You can then specify a URL that corresponds to a REST-based search request as shown in Figure 4.

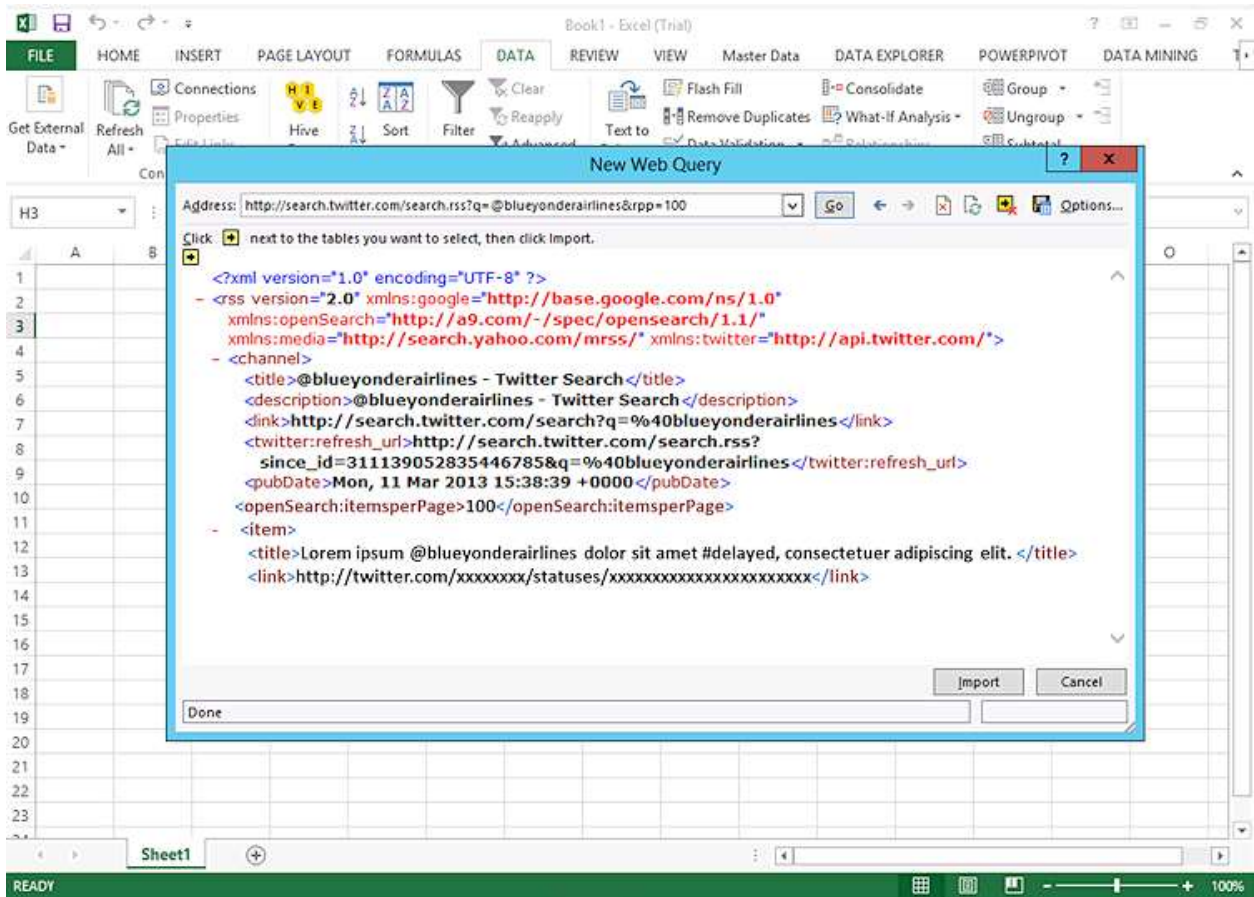


Figure 4
Using Excel to extract search results from Twitter

Importing the search results into the workbook results in a table of data with a row for each tweet returned by the query, as shown in Figure 5. After the data has been imported, you can remove any unnecessary columns and make formatting changes as required before saving the data as a delimited file. To simplify analysis, the business analysts at Blue Yonder Airlines have decided to include only the publication date, ID, title, and author of each tweet, and to format the publication date using the format DD-MM-YYYY. These changes are easy to make in Excel, and then the worksheet can be saved as a tab-delimited text file.

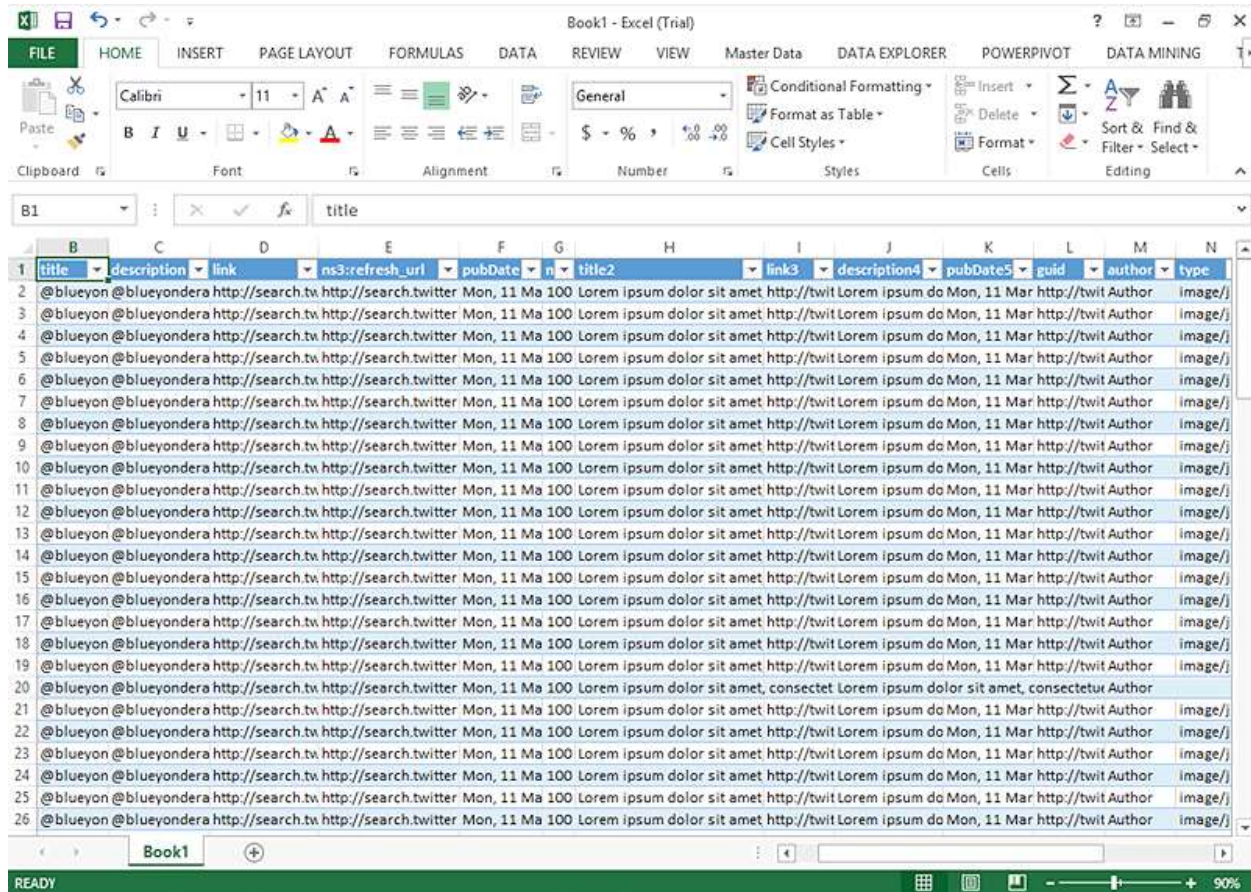


Figure 5
Twitter data in Excel

To upload the text file to the HDInsight cluster for analysis, a business analyst can use the **fs.put()** JavaScript command in the interactive console area of the HDInsight dashboard, and specify the source and destination paths as shown in Figure 6.

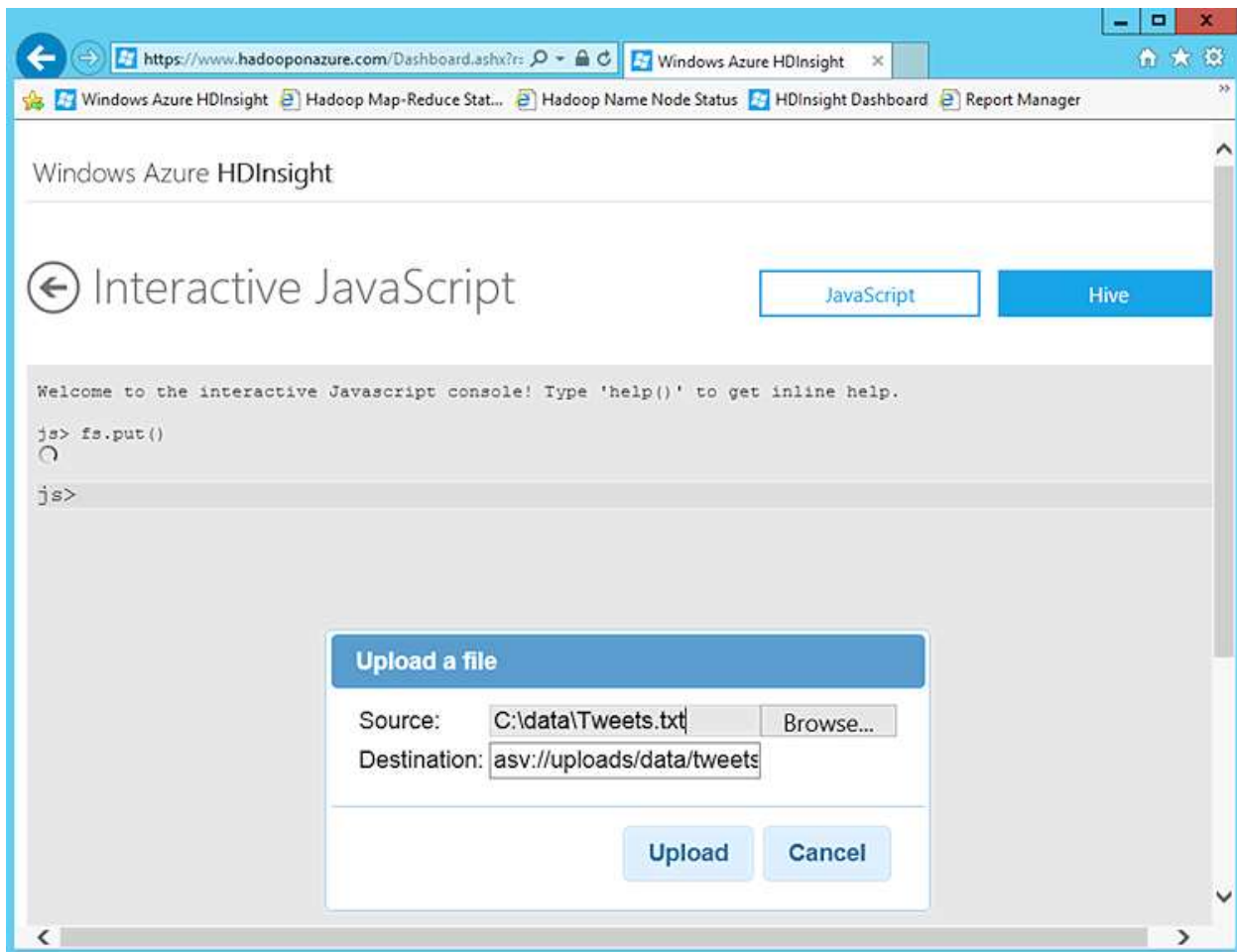


Figure 6
Uploading a file to HDInsight

Note that the destination path is prefixed with the **asv://** protocol indicator. Using this prefix uploads the data to the Windows Azure Blob Store associated with the cluster instead of the native HDFS file system. To view the contents of a folder in either system in the HDInsight interactive console, you can use the **#ls** command; and to view the contents of the file you can use the **#cat** command, as shown in Figure 7.

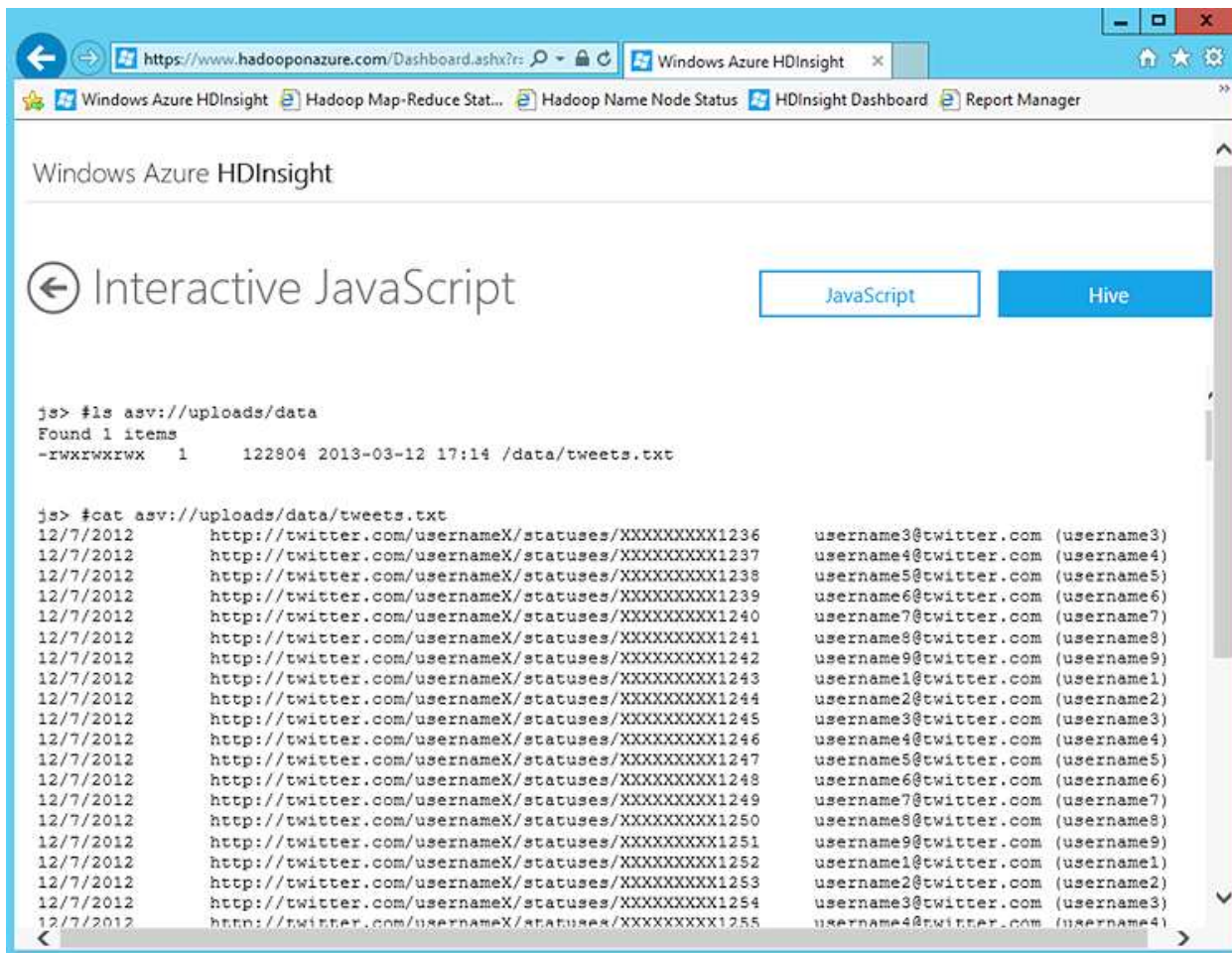


Figure 7
Browsing files in the HDInsight interactive console

The interactive console is suitable for uploading small files, but has an upper-limit on the file size. To support uploads of large files, and to improve the efficiency of data uploads, the developers at Blue Yonder Airlines decided to use a custom command line upload utility. The utility is run from the command line by using the following syntax:

```
BlobManager.UploadConsoleClient.exe source blob_store key container
```

The arguments for the BlobManager.UploadConsoleClient.exe utility are described in the following table:

Argument	Description
source	A local file or folder. If a folder is specified, all files in that folder are uploaded.
blob_store	The name of the Windows Azure Blob Store to which the data is to be uploaded
key	The passkey used to authenticate the connection to the Windows Azure Blob Store.
container	The name of the container in the Windows Azure Blob Store to which the data should be uploaded. The uploaded files will be placed in a folder named data within the specified container.

The upload utility splits the files into 4MB blocks and uses 16 I/O threads to upload 16 blocks (64MB) in parallel. After each block is uploaded, it is marked as committed and the next block is uploaded. When all of the blocks are uploaded, they are reassembled into the original files in the Windows Azure blob Store.

Automating Data Ingestion with SSIS

While interactively downloading data from Twitter and uploading it to HDInsight is appropriate for some initial exploration and testing, the project at Blue Yonder Airlines must obtain and upload data from Twitter every day for a period of a few weeks; making it preferable to automate the data ingestion process. To accomplish this, a developer has created an SSIS package that extracts the data from Twitter and uploads it to the Windows Azure Blob Store at a scheduled daily interval. The project contains a package named **Tweets.dtsx**, which defines the control flow shown in Figure 8.

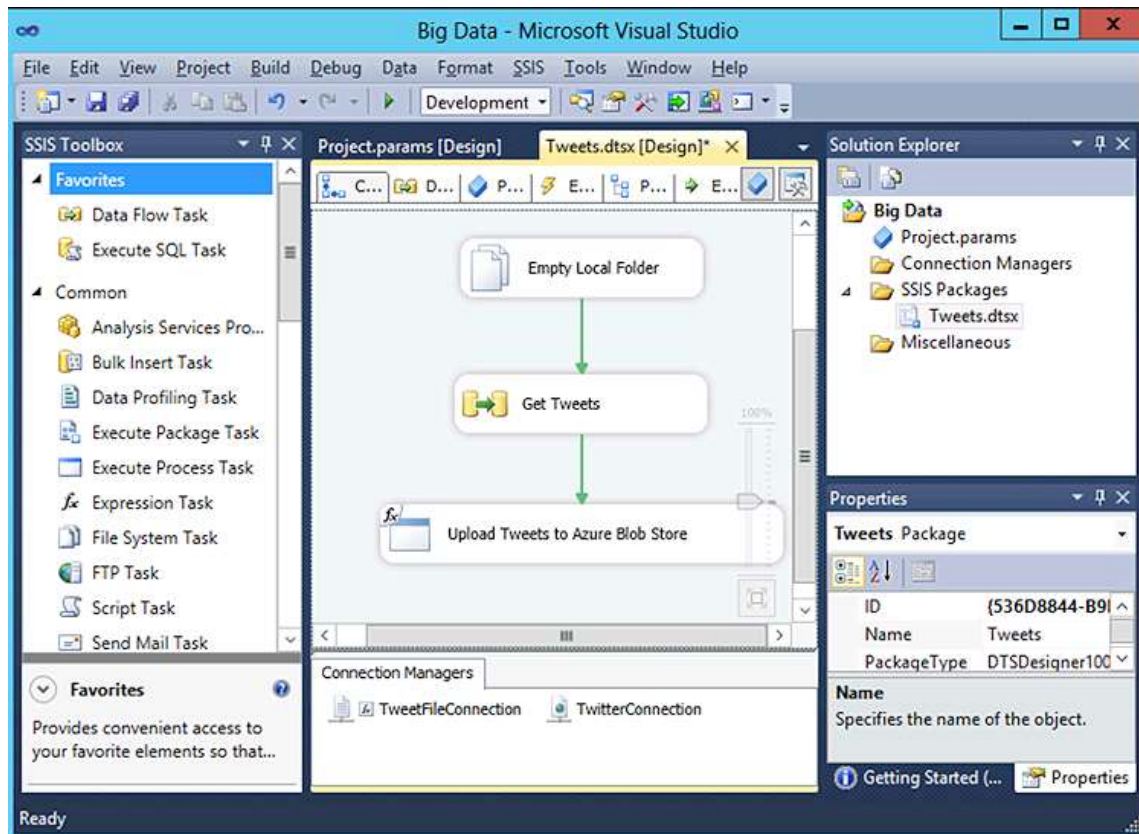


Figure 8
SSIS control flow for loading Twitter data to HDInsight

The SSIS project includes the following project-level parameters:

Parameter	Description
Query	The search term used to filter tweets from the Twitter search API.
LocalFolder	The local file system folder to which the search results will be downloaded from Twitter.
BlobstoreName	The name of the Windows Azure Blob Store associated with the HDInsight cluster.
BlobstoreKey	The passkey required to connect to the Windows Azure Blob Store.
BlobstoreContainer	The container in the Windows Azure blob store to which the data will be uploaded.

The control flow for the **Tweets.dtsx** package consists of the following tasks:

1. **Empty Local Folder:** A *File System* task that deletes any existing files in the folder defined by the **LocalFolder** parameter.
2. **Get Tweets.** A *Data Flow* task that extracts the data from Twitter and saves it as a local file in the folder defined by the **LocalFolder** parameter.
3. **Upload Tweets to Azure Blob Store:** An *Execute Process* task that uses the command line tool discussed previously to upload the local file to the Windows Azure Blob Store.

The package uses the following connection managers:

Connection Manager	Type	Connection String
TwitterConnection	HTTP	http://search.twitter.com/search.rss?rpp=100
TweetFileConnection	File	@[\$Project::LocalFolder] + @[System::ExecutionInstanceGUID] + ".txt" (This is an expression that uses the LocalFolder project parameter and the unique runtime execution ID to generate a filename such as C:\data\{12345678-ABCD-FGHI-JKLM-1234567890AB}.txt)

The **Get Tweets** data flow task is shown in Figure 9, and consists of a *Script* component named **Twitter Search** that implements a custom data source, and a *Flat File* destination named **Tweet File**.

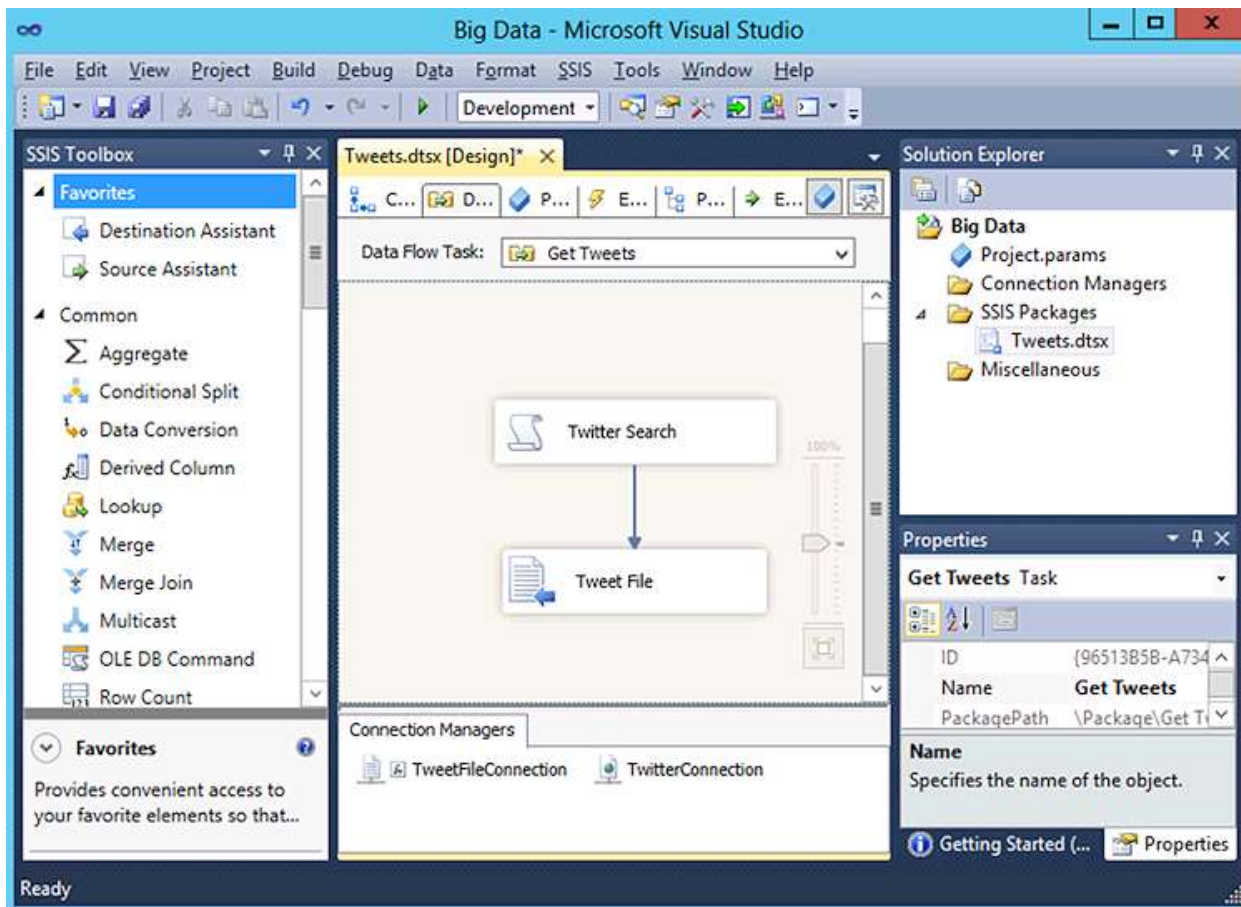


Figure 9
Data flow to extract data from Twitter

The **Twitter Search** script component is configured to use the **TwitterConnection** HTTP connection manager, and has the following output fields defined:

- TweetDate
- TweetID
- TweetAuthor
- TweetText

The Script component is granted read-only access to the **Query** parameter, and includes the following code:

```
C#
using System;
using System.Data;
using Microsoft.SqlServer.Dts.Pipeline.Wrapper;
using Microsoft.SqlServer.Dts.Runtime.Wrapper;
using System.Xml;
using System.ServiceModel.Syndication;
```

```

[Microsoft.SqlServer.Dts.Pipeline.SSISScriptComponentEntryPointAttribute]
public class ScriptMain : UserComponent
{
    private SyndicationFeed tweets = null;
    private XmlReader tweetReader = null;

    public override void PreExecute()
    {
        base.PreExecute();
        string today = DateTime.Today.ToString("yyyy=MM-dd");
        tweetReader = XmlReader.Create(Connections.Connection.ConnectionString
            + "&q=" + Variables.Query + " since:" + today);
        tweets = SyndicationFeed.Load(tweetReader);
    }

    public override void PostExecute()
    {
        base.PostExecute();
    }

    public override void CreateNewOutputRows()
    {
        if (tweets != null)
        {
            foreach (var tweet in tweets.Items)
            {
                Output0Buffer.AddRow();
                Output0Buffer.TweetDate = tweet.PublishDate.ToString("d");
                Output0Buffer.TweetID = tweet.Id;
                Output0Buffer.TweetAuthor = tweet.Authors[0].Email;
                Output0Buffer.TweetText = tweet.Title.Text;
            }
            Output0Buffer.SetEndOfRowset();
        }
    }
}

```

Note that the code dynamically generates a REST URI for the Twitter search API that includes the search term in the **Query** parameter and a filter to include only tweets since the start of the current day. The code also includes a reference to the **System.ServiceModel.Syndication** namespace, which includes classes that make it easier to work with RSS feeds.

The **PublishDate**, **Id**, first author **Email**, and **Title** fields from each tweet in the RSS results are assigned to the output fields and sent through the data flow to the **Tweet File** destination, which uses the **TweetFileConnection** connection manager to write the data to a tab-delimited file in the local file system.

After the data flow task has downloaded the data from Twitter, the **Upload Tweets to Azure Blob Store** Execute Process task in the control flow uses the custom BlobManager.UploadConsoleClient.exe console application described previously to upload the file to the Windows Azure Blob Store. The Execute Process task is configured to use the following expression for the **Arguments** property (the command line arguments that are passed to the executable):

```
@[$Project::LocalFolder] + " " + @[$Project::BlobstoreName] + " " +  
@[$Project::BlobstoreKey] + " " + @[$Project::BlobstoreContainer]
```

After the SSIS project is completed, it is deployed to an SSIS Catalog server and the **Tweets.dtsx** package is scheduled to run each day at a specified time with the following parameter values:

Parameter	Value
Query	@BlueYonderAirlines
LocalFolder	c:\data
BlobstoreName	BlueYonderBlobStore
BlobstoreKey	1234567890abcdefghijklmnopqrstuvwxyz&%\$@1234567890abcdefghijklmnopqrstuvwxyz==
BlobstoreContainer	uploads

This automatically retrieves and uploads data from Twitter at a scheduled time each day.

Capturing Real-Time Twitter Data with StreamInsight

The SSIS-based solution for ingesting Twitter data into HDInsight provides a way to capture a sample of tweets each day. However, if the analysis of twitter data proves useful to the business, a future iteration of the project might want to capture tweets in real-time and upload them to HDInsight in frequent batches for more detailed analysis. To accomplish this, the developers at Blue Yonder Airlines have investigated the Twitter streaming API and the use of Microsoft StreamInsight to capture tweet events from a Twitter stream and upload them to the Windows Azure Blob Store. The Twitter streaming API returns a stream of tweets in JSON format, which the StreamInsight application must serialize as events that are captured and written to a file. The file can then be uploaded to the Windows Azure Blob Store when a suitable batch of tweets has been captured.

To create the StreamInsight solution, the developers have created the following classes:

- **Tweet**. An event class that represents a tweet event:

```
C#  
[Serializable]  
public class Tweet  
{  
    public Int64 ID { get; set; }  
    public DateTime CreatedAt { get; set; }  
    public string Text { get; set; }  
    public string Source { get; set; }  
    public string UserLang { get; set; }  
}
```

```

public int UserFriendsCount { get; set; }
public string UserDescription { get; set; }
public string UserTimeZone { get; set; }
public string UserURL { get; set; }
public string UserName { get; set; }
public Int64 UserID { get; set; }
public Int64 DeleteStatusID { get; set; }
public Int64 DeleteUserID { get; set; }
}

```

- **Adapters.** *Input adapters* are used to submit events to the StreamInsight CEP engine, where they can be filtered by queries, and *output adapters* are used to consume and process the filtered events generated by the CEP engine. Adapters can be typed (that is, explicitly mapped to a specific event class) or untyped, and there are base classes for handling typed and untyped events based on three event shapes: *Point* events represent a single point in time, *Interval* events represent events with a fixed duration, and *Edge* events represent events with an unknown duration. All base classes for input and output adapters include **Start**, **Resume**, and **Dispose** events that you must implement, and in addition you must implement a method to submit events to StreamInsight in your input adapters, and an event to consume events in your output adapters. For the Twitter stream capture, the input adapter reads the JSON response from an HTTP request to the Twitter streaming API (with an appropriate track parameter that filters for tweets containing “@BlueYonderAirlines”), serializing each event to a Tweet class and submitting it to the StreamInsight CEP engine for processing. The output adapter reads each Tweet event in the CEP stream, displays it in the real-time user interface, and writes it to the text file.

C#

```

//INPUT ADAPTER
public class TweetInputAdapter : TypedPointInputAdapter<Tweet>
{
    string userName = "";
    string passWord = "";

    public TweetInputAdapter(TweetInputConfig config)
    {
        userName = config.Username;
        passWord = config.Password;
    }

    public override void Start()
    {
        ProduceEvents();
    }

    public override void Resume()
    {
        ProduceEvents();
    }
}

```

```

private void ProduceEvents()
{
    string q = "@BlueYonderAirlines";
    WebRequest request = HttpWebRequest.Create
        (https://stream.twitter.com/1.1/statuses/filter.json?track= + q);
    request.Credentials = new NetworkCredential(userName, passWord);
    var response = request.GetResponse();

    using (StreamReader streamReader = new
        StreamReader(response.GetResponseStream()))
    {
        while (AdapterState != AdapterState.Stopping)
        {
            string line = streamReader.ReadLine();
            JavaScriptSerializer s = new JavaScriptSerializer();
            Tweet tweet = s.Deserialize<Tweet>(line);
            PointEvent<Tweet> evt = CreateInsertEvent();
            evt.StartTime = DateTime.Now;

            //Get a tweet from the tweet reader object
            evt.Payload = tweet;

            // do we need to suspend?
            if (Enqueue(ref evt) == EnqueueOperationResult.Full)
            {
                ReleaseEvent(ref evt);
                Ready();
                // the engine will call Resume at some point
                return;
            }

            // we should do the same return value check for CTIs...
            if (EnqueueCtiEvent(DateTime.Now) == EnqueueOperationResult.Full)
            {
                ReleaseEvent(ref evt);
                Ready();
                // the engine will call Resume at some point
                return;
            }
        }
        // let the engine stop now.
        Stopped();
        return;
    }

    protected override void Dispose(bool disposing)
    {
        base.Dispose(disposing);
    }
}

```

```

    }
}

//OUTPUT ADAPTER
class TweetOutputAdapter : TypedPointOutputAdapter<Tweet>
{
    private string fname;

    public TweetOutputAdapter(TweetOutputConfig config)
    {
        fname = config.targetOutput;
    }

    public override void Start()
    {
        ConsumeEvents();
    }

    public override void Resume()
    {
        ConsumeEvents();
    }

    private void ConsumeEvents()
    {
        PointEvent<Tweet> currEvent = null;
        try
        {
            while (true)
            {
                if (AdapterState.Stopping == AdapterState)
                {
                    Stopped();
                    return;
                }
                if (DequeueOperationResult.Empty == Dequeue(out currEvent))
                {
                    Ready();
                    return;
                }
                else
                {
                    if (currEvent.EventKind == EventKind.Insert)
                    {
                        Tweet evtTweet = currEvent.Payload;
                        DateTime evtTime = currEvent.StartTime.DateTime;
                        string tweetData = evtTime.ToUniversalTime() + "\t" + evtTweet.ID
                            + "\t" + evtTweet.Text;
                    }
                }
            }
        }
        catch { }
    }
}

```

```

        // Write the tweet to the User Interface
        UIUpdater uu = new UIUpdater();
        uu.OnTweetCaptured(tweetData);

        //write the tweet to a file
        if (!File.Exists(fname))
        {
            using (StreamWriter outputFile = File.CreateText(fname))
            {
                outputFile.WriteLine(tweetData);
            }
        }
        else
        {
            using (StreamWriter outputFile = File.AppendText(fname))
            {
                outputFile.WriteLine(tweetData);
            }
        }
        ReleaseEvent(ref currEvent);
    }
}
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}

protected override void Dispose(bool disposing)
{
    base.Dispose(disposing);
}
}
}

```

- **Adapter Configurations.** Note that the adapter constructors in the previous code samples include an argument for a **config** object from which some initialization values are obtained. Adapter configuration classes are used to provide configuration values that can be used by adapters. In this case, the input adapter that retrieves tweets from Twitter and submits them to the CEP engine requires Twitter credentials, and the output adapter needs a target output destination that specifies the output file to which the tweet event data should be written:

C#

```

public class TweetInputConfig
{
    public string Username { get; set; }
    public string Password { get; set; }
}

```



```

public class TweetOutputConfig
{
    public string targetOutput { get; set; }
}

```

- **Adapter Factories.** Adapter factories are used by StreamInsight applications to instantiate adapters. As with adapters, the StreamInsight framework provides base classes for typed and untyped adapter factories. These base classes include **Create** and **Dispose** events, which you must implement. In this case, tweets are always treated as *Point* events, so only Point typed adapters have been implemented. The adapter factories therefore simply return an instance of a Point adapter with the appropriate configuration settings as shown here:

C#

```

public class TweetInputFactory:
    ITypedInputAdapterFactory<TweetInputConfig>
{
    public InputAdapterBase Create<TPayload>
        (TweetInputConfig configInfo, EventShape eventShape)
    {
        InputAdapterBase adapter = default(InputAdapterBase);
        if (eventShape == EventShape.Point)
        {
            adapter = new TweetInputAdapter(configInfo);
        }
        // add code for other event shapes as required
        return adapter;
    }

    public void Dispose()
    {
    }
}

public class TweetOutputFactory:
    ITypedOutputAdapterFactory<TweetOutputConfig>
{
    public OutputAdapterBase Create<TPayload>
        (TweetOutputConfig configInfo, EventShape eventShape)
    {
        OutputAdapterBase adapter = default(OutputAdapterBase);
        if (eventShape == EventShape.Point)
        {
            adapter = new TweetOutputAdapter(configInfo);
        }
        // add code for other event shapes as required
        return adapter;
    }

    public void Dispose()

```

```
{
}
}
```

After implementing the required classes, the Blue Yonder Airlines developers creates a simple Windows application that uses these classes to consume and process events through an instance of a StreamInsight server, which can be in-process or hosted externally as a service. A key element in using StreamInsight to process events is the creation of a query template that is used to select which events should be processed. The following code sample shows how to instantiate the classes described previously and process every tweet event:

C#

```
// connect to a StreamInsight server and create an application
server = Server.Create("StreamInsight");
var app = server.CreateApplication("app");

// Create the input adapter with Twitter credentials
var inputConfig = new TweetInputConfig { TwitterName = txtUserName.text,
                                         Password = txtPassword.Text };
// create the input stream
var inputStream = CepStream<Tweet>.Create("inputStream",
                                         typeof(TweetInputFactory), inputConfig, EventShape.Point);

// Create an output adapter instance with a file path
string targetFile = @"C:\Data\" + Guid.NewGuid() + ".txt";
var outputConfig = new TweetOutputConfig { targetOutput = targetFile };

// Create query and bind to the output adapter
CepStream<Tweet> myQuery = from evt in inputStream
                          where evt.DeleteStatusID == 0
                          select evt;

Query tweetsQuery = myQuery.ToQuery(app, "tweetsQuery", "",
                                     typeof(TweetOutputFactory), outputConfig,
                                     EventShape.Point,
                                     StreamEventOrder.FullyOrdered);

//Start querying
tweetsQuery.Start();
```

This code uses the query to filter a stream of events submitted by the input adapter so that deleted tweets are not passed to the output adapter. The tweets that are processed by the output adapter are displayed in the application user interface as shown in Figure 10, and they are also saved in a text file.

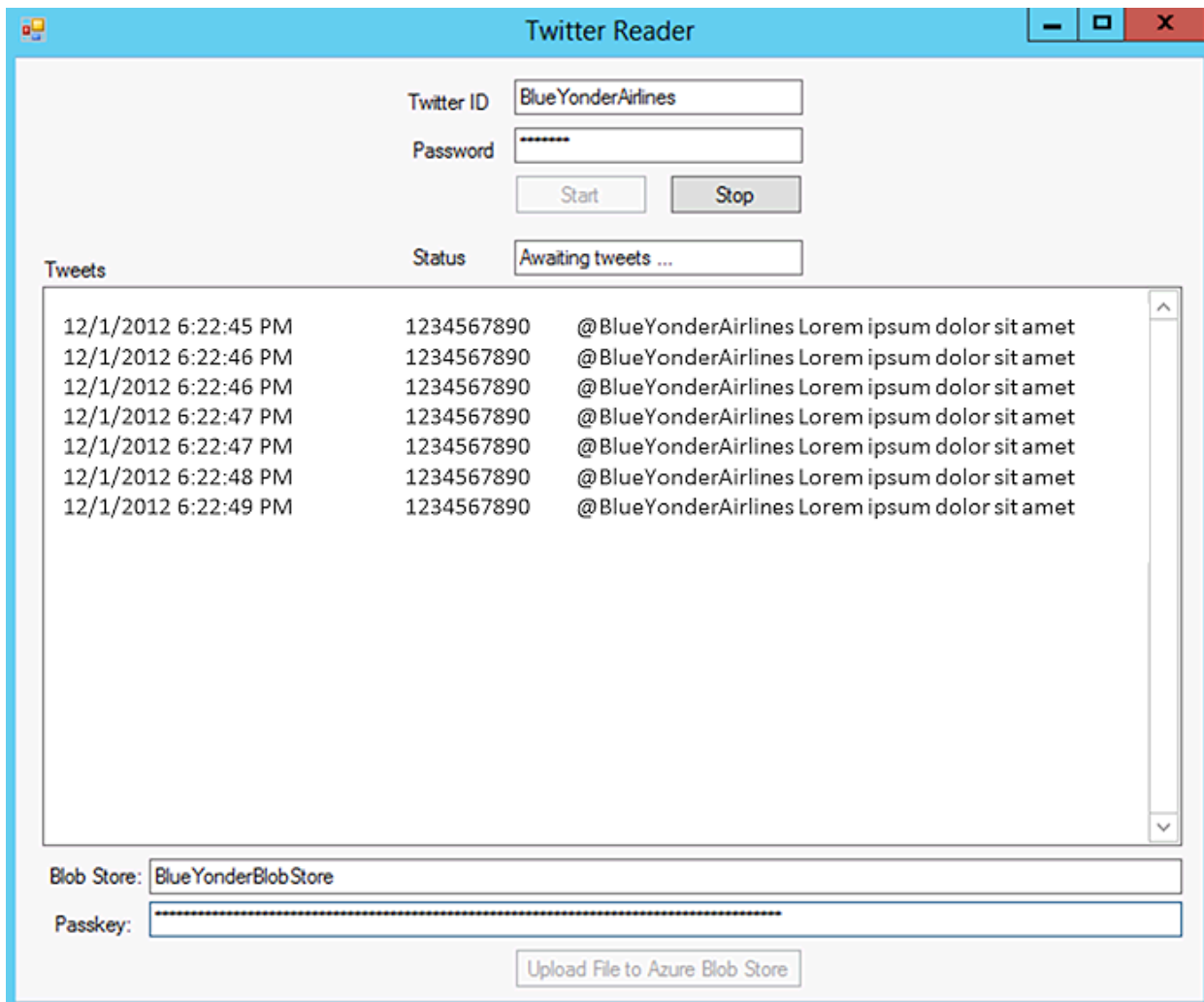


Figure 10
Capturing real-time Twitter data with StreamInsight

When a sufficient batch of tweets has been captured, the application uses the BlobManager library from the custom application described earlier to upload the text file to the Windows Azure Blob Store as shown in the following code:

```
C#
string conn = "DefaultEndpointsProtocol=https;AccountName="
             + txtBlobstoreName.Text + ";AccountKey=" + txtBlobstoreKey.Text;
BlobUploader u = new BlobUploader();
u.UploadBlob(conn, "uploads", "data/tweets.txt", targetFile, BlobUploadMode.Append);
```

Data Processing

Now that Blue Yonder Airlines has a solution for capturing the Twitter data and uploading it to the Windows Azure Blob Store, the developers can turn their attention to processing the data in HDInsight. As an initial step, it has been decided to parse the tweet text and identify the most commonly used hashtags.

Using Map/Reduce Code to Parse Text

Parsing the unstructured tweet text and identifying hashtags can be accomplished by implementing a custom map/reduce solution. The developers at Blue Yonder Airlines are more familiar with JavaScript than Java, and so have decided to implement the map/reduce code in a JavaScript script. The script consists of a **map** function that runs on each cluster node in parallel and parses the text input to create a set of key/value pairs with a value of 1 for each word prefixed by “#” found in the text. These value pairs are passed to the **reduce** function, which runs on a reducer to count the number of instances of each key – and so determine the number of times each hashtag was mentioned in the source data:

JavaScript

```
// Map function - runs on all nodes
var map = function (key, value, context) {
  // split the data into an array of words
  var hashtags = value.split(/[^\0-9a-zA-Z#]/);

  //Loop through the array, creating a value of 1 for each word beginning "#"
  for (var i = 0; i < hashtags.length; i++) {
    if (hashtags[i].substring(0, 1) == "#") {
      context.write(hashtags[i].toLowerCase(), 1);
    }
  }
};

//Reduce function - runs on reducer node(s)
var reduce = function (key, values, context) {
  var sum = 0;
  // Sum the counts of each tag found in the map function
  while (values.hasNext()) {
    sum += parseInt(values.next());
  }
  context.write(key, sum);
};
```

The script is uploaded to the cluster as tagcount.js, and executed by running the following Pig command from the interactive console in an HDInsight server.

```
pig.from("asv://uploads/data").mapReduce("./tagcount.js", "tag, count:long").orderBy
("count DESC").take(10).to ("asv://results/countedtags")
```

This command executes the map/reduce job on the files in the asv://uploads/data folder, sorts the results by the number of instances of each hashtag found in the data, filters the results to include only the top 10 hashtags, and stores the results in the asv://results/countedtags folder. After the job has completed, the

asv://results/countedtags folder contains a file named **part-r-00000** containing the top 10 most common hashtags and the total number of instances of each one as shown in Figure 11.

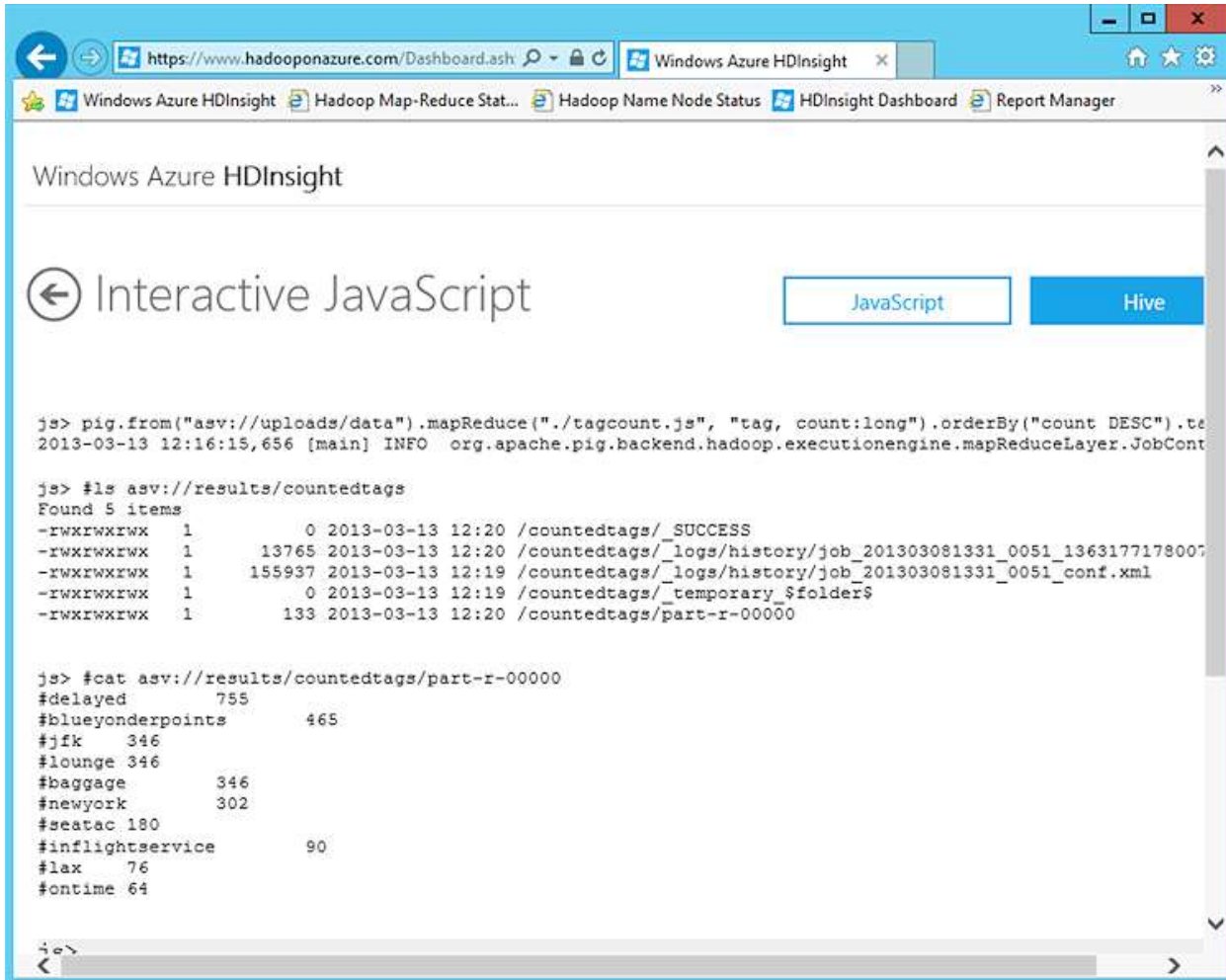


Figure 11
Output from a map/reduce job to find the top 10 hashtags

Using Pig to Group and Summarize Data Values

The top 10 hashtags provides some information about the topics that are important to Blue Yonder Airlines customers. However, it might be more useful to group the occurrences of these hashtags by date, and determine if there are any significant trends or patterns in the Twitter data.

Writing map/reduce code to group the hashtags by date is of course perfectly possible, but the code would be more complex than the basic script created previously. To simplify the development effort, the developers at Blue Yonder Airlines have decided to use Pig to group and summarize the tweets. Pig provides a workflow-based approach to data processing that is ideal for restricting and summarizing data, and a Pig Latin script to perform the aggregation is syntactically much easier to create than implementing the custom map/reduce code. Initially, the Pig Latin code uses the original tweets source data to group the number of mentions of each hashtag by date, and then combines this grouped relation with the top 10 tweets generated by the JavaScript map/reduce

job shown previously to create an output that includes the total count for each of the top 10 hashtags in the source data as well as a daily count for each hashtag.

Pig Latin

```
-- load tweets
Tweets = LOAD 'asv://uploads/data' AS (date, id, author, tweet);
-- split tweet into words
TweetWords = FOREACH Tweets GENERATE date, FLATTEN(TOKENIZE(tweet)) AS tag, id;
--filter words to find hashtags
Tags = FILTER TweetWords BY tag matches '#.*';
-- clean tags by removing trailing periods
CleanTags = FOREACH Tags GENERATE date, LOWER(REPLACE(tag, '\\.', '')) as tag, id;
-- group tweets by date and tag
GroupedTweets = GROUP CleanTags BY (date, tag);
-- count tag mentions per group
CountedTagMentions = FOREACH GroupedTweets GENERATE group, COUNT(CleanTags.id) as mentions;
-- flatten the group to generate columns
TagMentions = FOREACH CountedTagMentions GENERATE FLATTEN(group) as (date, tag), mentions;
-- load the top tags found by map/reduce previously
TopTags = LOAD 'asv://results/countedtags/part-r-00000' AS (toptag, totalcount:long);
-- Join tweets and top tags based on matching tag
TagMentionsAndTopTags = JOIN TagMentions BY tag, TopTags BY toptag;
-- get the date, tag, totalcount, and mentions columns
TagMentionsAndTotals = FOREACH TagMentionsAndTopTags GENERATE date, tag, totalcount,
mentions;
-- sort by date and mentions
SortedTagMentionsAndTotals = ORDER TagMentionsAndTotals BY date, mentions;
-- store the results as a file
STORE SortedTagMentionsAndTotals INTO 'asv://results/dailytagcounts';
```

This script is saved as pigscript.pig and executed from the Pig installation folder on the cluster in a remote desktop session as shown in Figure 12.

```
c:\apps\dist\pig-0.9.3-SNAPSHOT\bin>pig pignscript.pig
2013-03-13 12:39:54,187 [main] INFO org.apache.pig.Main - Logging error message
s to: c:\apps\dist\pig-0.9.3-SNAPSHOT\bin\pig_1363178394177.log
2013-03-13 12:39:54,584 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: hdfs://10.174.188.26:
9000
2013-03-13 12:39:55,038 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to map-reduce job tracker at: 10.174.188.26:901
0
2013-03-13 12:39:55,996 [main] WARN org.apache.pig.PigServer - Encountered Warn
ing USING_OVERLOADED_FUNCTION 1 time(s).
2013-03-13 12:39:55,996 [main] WARN org.apache.pig.PigServer - Encountered Warn
ing IMPLICIT_CAST_TO_CHARARRAY 2 time(s).
2013-03-13 12:39:56,008 [main] INFO org.apache.pig.tools.pigstats.ScriptState -
Pig features used in the script: HASH_JOIN, GROUP_BY, ORDER_BY, FILTER
2013-03-13 12:39:56,080 [main] INFO org.apache.pig.newplan.logical.rules.Column
PruneVisitor - Columns pruned for Tweets: 52
2013-03-13 12:39:56,567 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? fal
se
2013-03-13 12:39:56,616 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.CombinerOptimizer - Choosing to move algebraic foreach to comb
iner
2013-03-13 12:39:56,644 [main] INFO org.apache.pig.backend.hadoop.executionengi
```

Figure 12

Running a Pig Latin script from the Pig command line

When the script has completed successfully, the results are stored in a file named **part-r-00000** in the `asv://results/dailytagcounts` folder as shown in Figure 13.

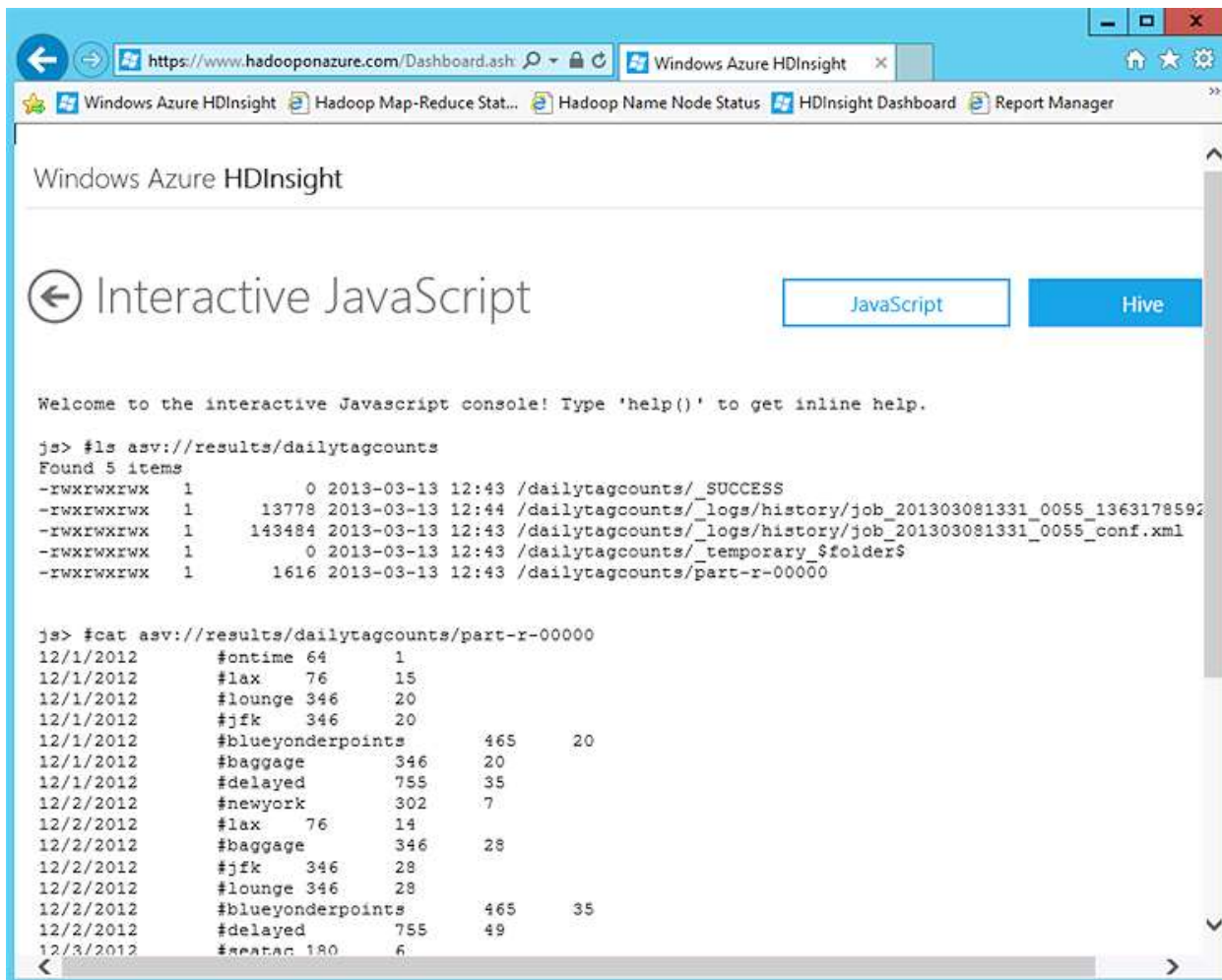


Figure 13
Grouped and counted hash tags produced by the Pig Latin script

Creating Tables with Hive

The output files generated by the JavaScript map/reduce script and the Pig Latin script contain useful information about the hashtags used by Blue Yonder Airlines customers, but to provide a more interactive analytical solution, the developers have decided to create Hive tables that can be queried to reveal further insights.

The following HiveQL statement is used to create an external table based on an empty folder in the Windows Azure Blob Store. The schema defined in the table reflects the values in the tab-delimited file generated by the Pig Latin script

HiveQL

```
CREATE EXTERNAL TABLE dailytwittertags
(tweetdate STRING,
tag STRING,
totalcount INT,
```

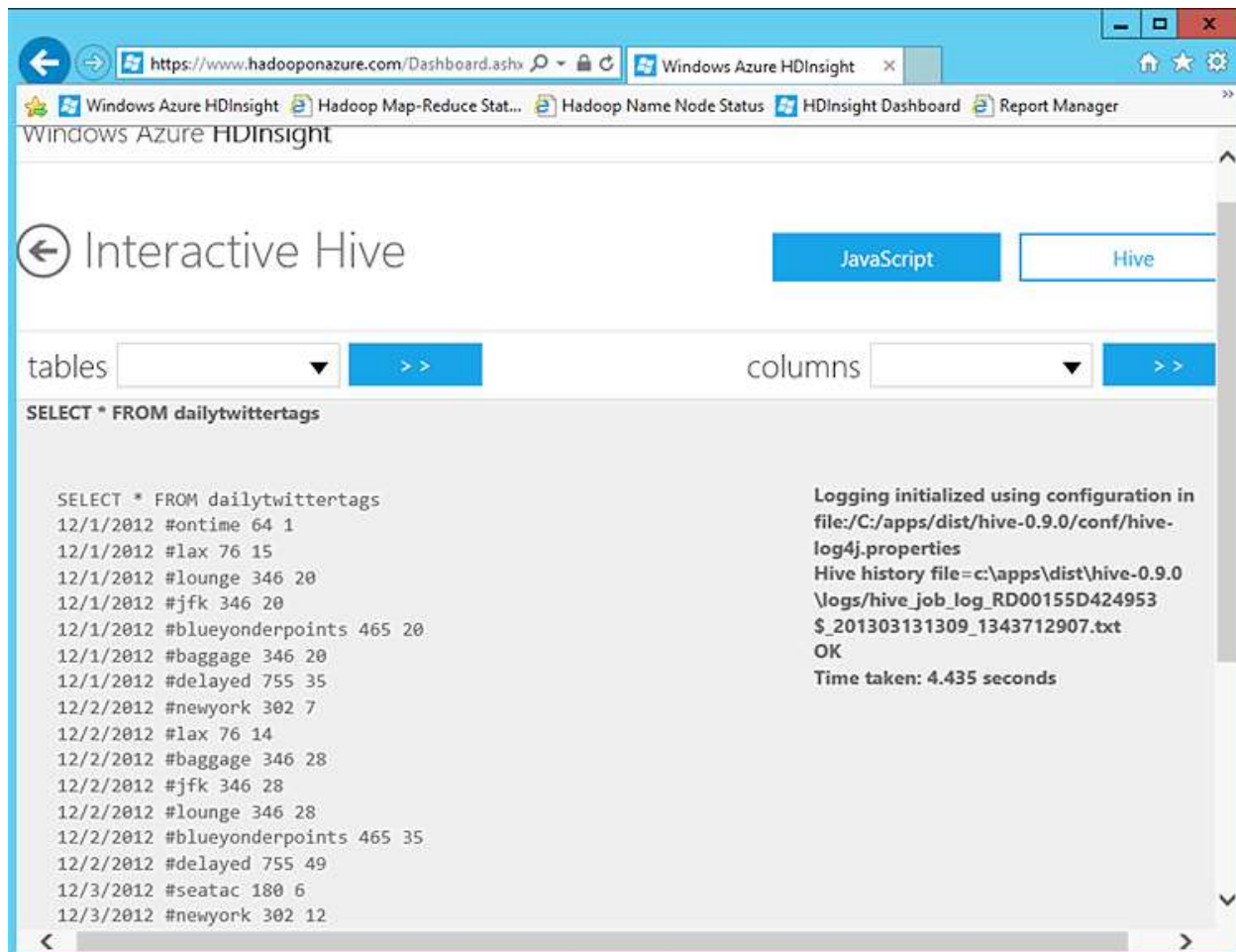


```
daycount INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE LOCATION 'asv://tables/dailytagcount'
```

To populate the table, the output file from the Pig Latin script is copied to the folder associated with the table by using the #cp command in the interactive JavaScript console.

```
#cp asv://results/dailytagcounts/part-r-00000 asv://tables/dailytagcounts/part-r-00000
```

Now that the folder associated with the table contains some data, you can query the table as shown in Figure 14.



The screenshot shows the Interactive Hive interface. At the top, there are tabs for 'JavaScript' and 'Hive'. Below that, there are dropdown menus for 'tables' and 'columns'. The main area displays the query 'SELECT * FROM dailytwittertags' and its results. The results are a list of tweets with their dates, hashtags, and counts. A log message on the right indicates that logging was initialized and the query took 4.435 seconds.

Date	Hashtag	Count	Other
12/1/2012	#ontime	64	1
12/1/2012	#lax	76	15
12/1/2012	#lounge	346	20
12/1/2012	#jfk	346	20
12/1/2012	#blueyonderpoints	465	20
12/1/2012	#baggage	346	20
12/1/2012	#delayed	755	35
12/2/2012	#newyork	302	7
12/2/2012	#lax	76	14
12/2/2012	#baggage	346	28
12/2/2012	#jfk	346	28
12/2/2012	#lounge	346	28
12/2/2012	#blueyonderpoints	465	35
12/2/2012	#delayed	755	49
12/3/2012	#seatac	180	6
12/3/2012	#newyork	302	12

Figure 14
Querying a Hive table based on the Pig Latin results

Evaluating Results

The data processing has now produced some results, so the developers at Blue Yonder Airlines have decided to invite some business analysts to review the results to validate their usefulness.

Reviewing Results in the Interactive Console

As a simple initial check of the results, a Business Analyst is invited to use the following commands in the interactive JavaScript console to display a pie chart of the 10 most commonly used hashtags:

Interactive JavaScript

```
file = fs.read("asv://results/countedtags/part-r-00000")
data = parse(file.data, "tag, count:long")
graph.pie(data)
```

The resulting pie chart shows the comparative frequency of each of the top 10 hashtags as shown in Figure 15.

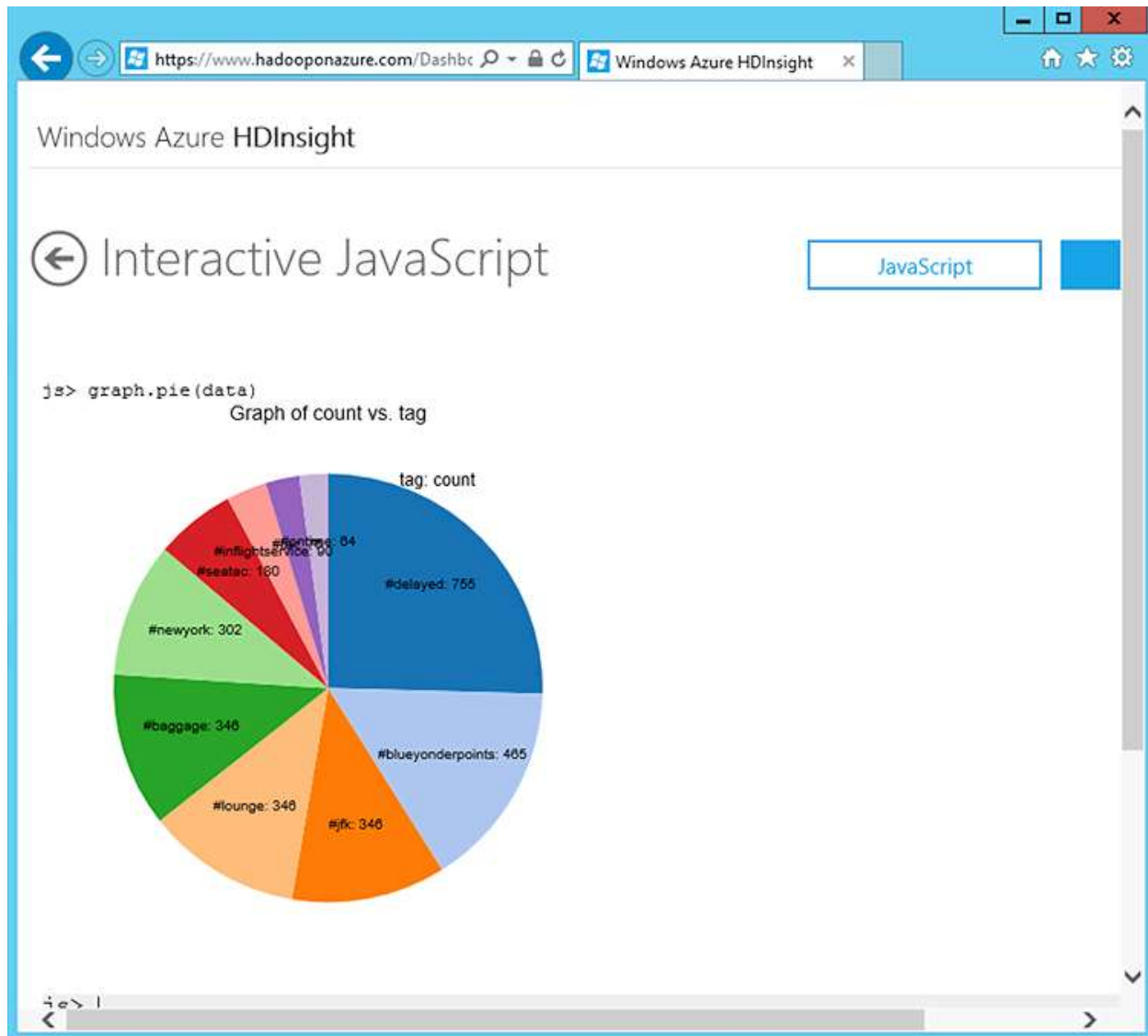


Figure 15
Displaying a graph of results in the interactive console

Using the Data Explorer in Excel

While the interactive explorer makes it easy to view small amounts of data, most business users are used to working in Excel, and this provides a more natural way for them to review the data processing results from HDInsight. Business Analysts at Blue Yonder Airlines already use the Data Explore add-in to import data into Excel for analysis, and decide to take advantage of the add-in's support for Hadoop in order to import the counted tags data.

To use the Data Explorer add-in import the data from HDInsight into Excel, the business analysts perform the following steps:

1. Click the **Data Explore** tab on the ribbon, and then in the **From Other Sources** drop-down list, click **From Hadoop File (HDFS)**.
2. Enter the name of the HDInsight cluster.
3. Browse the file system to find the file to be imported. In this case, the **part-r-0000** file generated by the JavaScript map/reduce code previously.
4. Convert the file contents into a table based on the appropriate column delimiter. In this case, the data contains two tab-delimited values (a tag and a count) for each row.
5. Rename columns and change data types as required. In this case, the columns are named Tag and Count, and the Count column is formatted as a Number as shown in Figure 16.

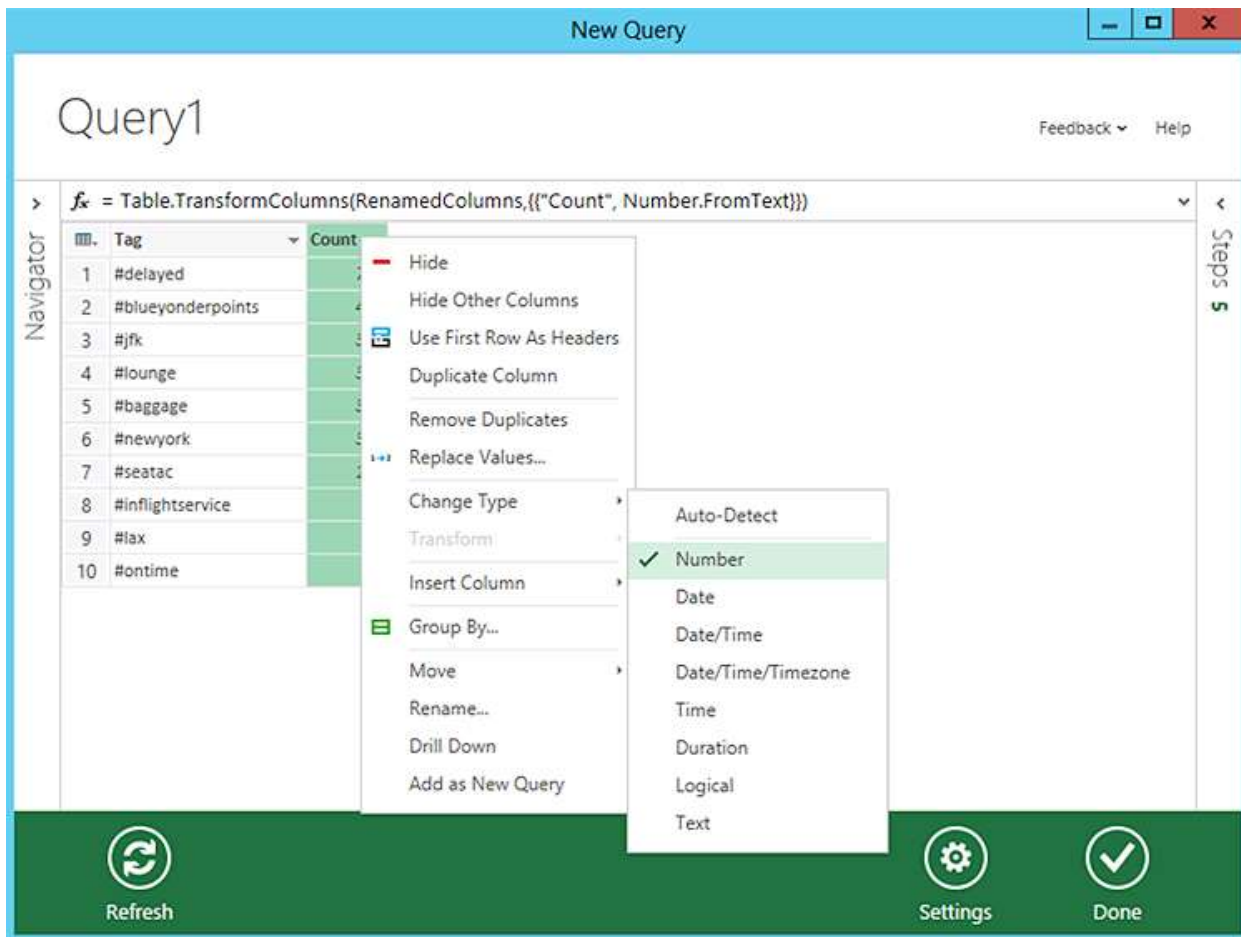


Figure 16

Importing data from HDInsight with the Data Explorer add-in for Excel

After the data has been imported, you can refine the Data Explorer query and use Excel’s built-in formatting and analytical capabilities to examine the data. For example, in Figure 17, a databar has been applied to make it easier to compare the relative hashtag counts.

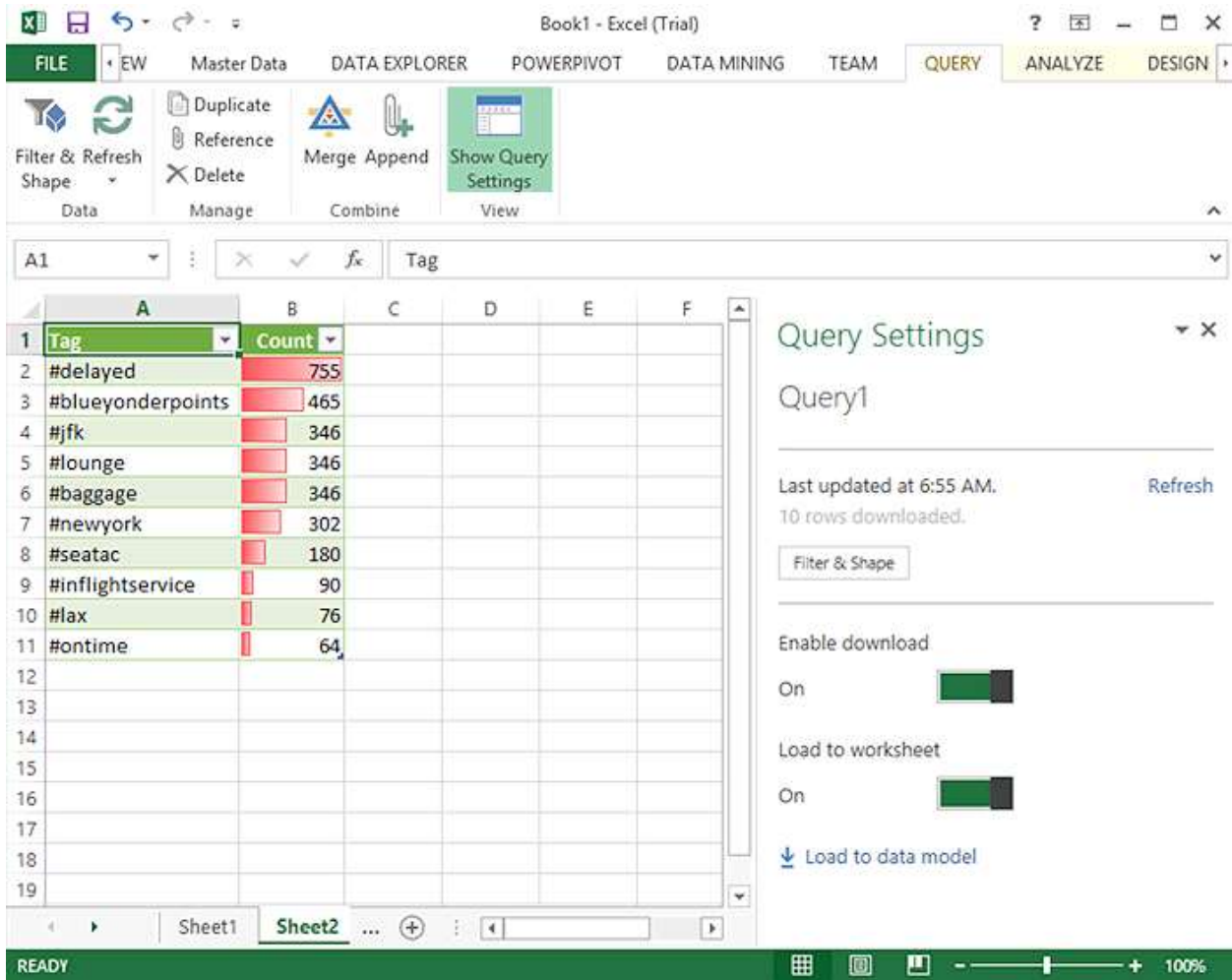


Figure 17
Analyzing data imported by the Data Explorer add-in for Excel

Querying Hive Tables from Excel

Previously, the developers at Blue Yonder Airlines defined Hive tables for the data in HDInsight. To consume these tables from Excel, the ODBC driver for Hive has been downloaded and installed on the computers belonging to the Business Analysis so that they can use the Hive Pane add-in for Excel to query the Hive tables in HDInsight and return results to Excel for analysis. Figure 18 shows the Hive pane being used to retrieve data from the **dailytwittertags** table created earlier.

The screenshot shows the Microsoft Excel interface with a Hive query pane open on the right. The main window displays a table with the following data:

tweetdate	tag	totalcount	daycount
12/1/2012	#ontime	64	1
12/1/2012	#lax	76	15
12/1/2012	#lounge	346	20
12/1/2012	#jfk	346	20
12/1/2012	#blueyonderpoints	465	20
12/1/2012	#baggage	346	20
12/1/2012	#delayed	755	35
12/2/2012	#newyork	302	7
12/2/2012	#lax	76	14
12/2/2012	#baggage	346	28
12/2/2012	#jfk	346	28
12/2/2012	#lounge	346	28
12/2/2012	#blueyonderpoints	465	35
12/2/2012	#delayed	755	49
12/3/2012	#seatac	180	6
12/3/2012	#newyork	302	12
12/3/2012	#lax	76	12
12/3/2012	#lounge	346	24

The Hive Query pane on the right shows the following configuration:

- Hive Connection:** Hive
- Hive Objects - Tables/Views:** dailytwittertags [Table]
- Columns:** tweetdate, tag

Figure 18
Querying a Hive table from Excel

After the data has been retrieved from Hive into Excel, the analyst can use the analytical capabilities of Excel to explore and visualize the information it contains as shown in Figure 19.

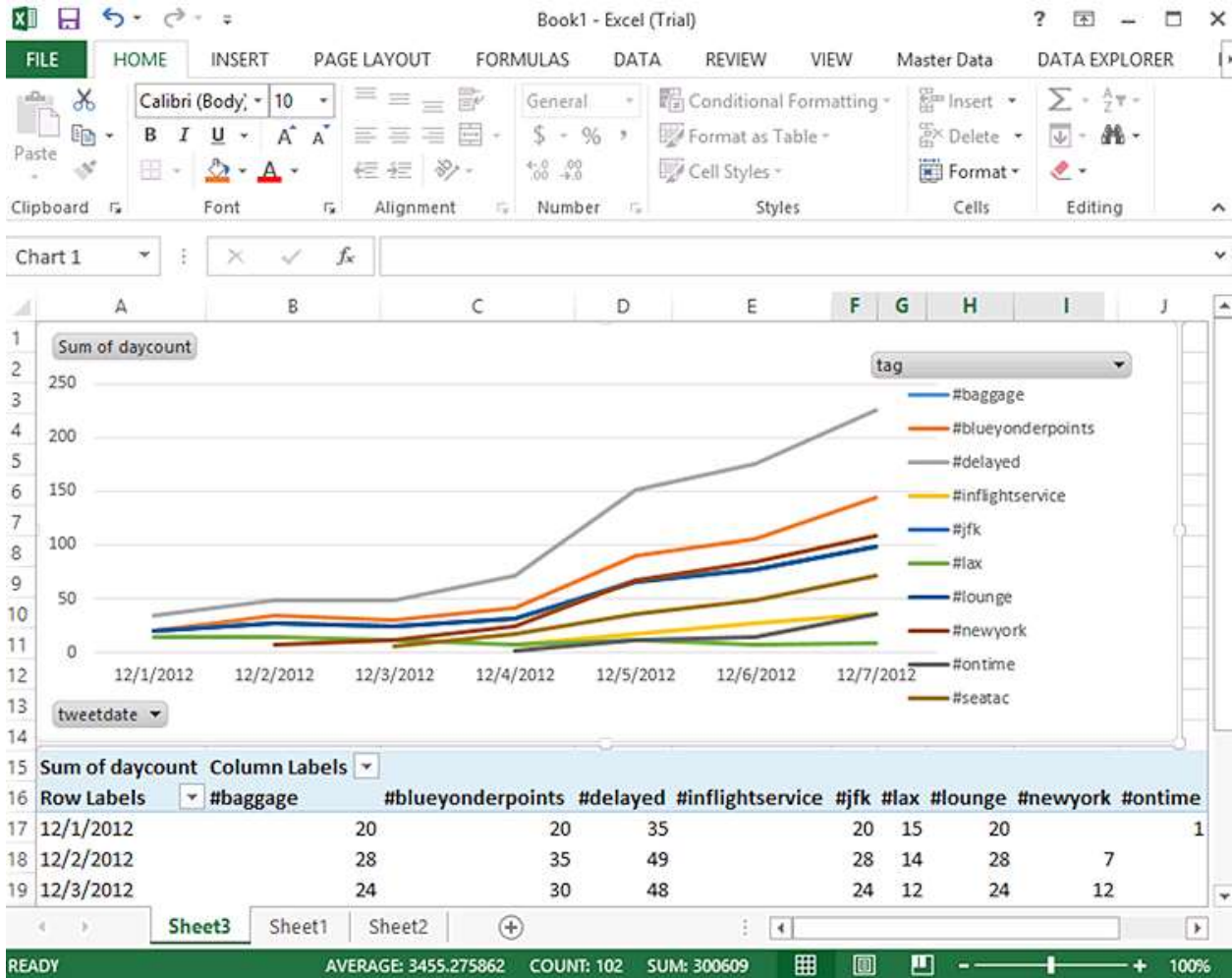


Figure 19
Analyzing the results of a Hive Query in Excel

Summary

In this scenario you have seen an example of a complete Big Data analysis process that includes obtaining source data and uploading it to a Windows Azure Blob Store, processing the data by using custom map/reduce code, Pig and Hive, and analyzing the results of the data processing in the HDInsight interactive console and Excel.

Scenario 2 Placeholder

Scenario 3 Placeholder

Scenario 4 Placeholder

Appendix A: An Overview of Enterprise Business Intelligence

Enterprise BI is a topic in itself, and a complete discussion of the design and implementation of a data warehouse and BI solution is beyond the scope of this guide. However, you do need to understand some of its specific features such as dimension and fact tables, slowly changing dimensions, and data cleansing if you intend to integrate a Big Data solution such as HDInsight with your data warehouse. This appendix provides an overview the main factors that make an enterprise BI system different from simple data management scenarios.

Even if you are already familiar with SQL Server as a platform for data warehousing and BI you may like to review the contents of this appendix before reading about integrating HDInsight with Enterprise BI in Chapter 3 and Scenario 2 in Chapter 5.

An enterprise BI solution commonly includes the following key elements:

- Business data sources.
- An extract, transform, and load (ETL) process.
- A data warehouse or departmental data marts.
- Corporate data models.
- Reporting and analytics.

Data flows from business applications and other external sources through an ETL process and into a data warehouse. Corporate data models are then used to provide shared analytical structures such as online analytical processing (OLAP) cubes or data mining models, which can be consumed by business users through analytical tools and reports.

Some BI implementations also enable analysis and reporting directly from the data warehouse, enabling advanced users such as business analysts and data scientists to create their own personal analytical data models and reports.

Figure 1 shows the architecture of a typical enterprise BI solution.

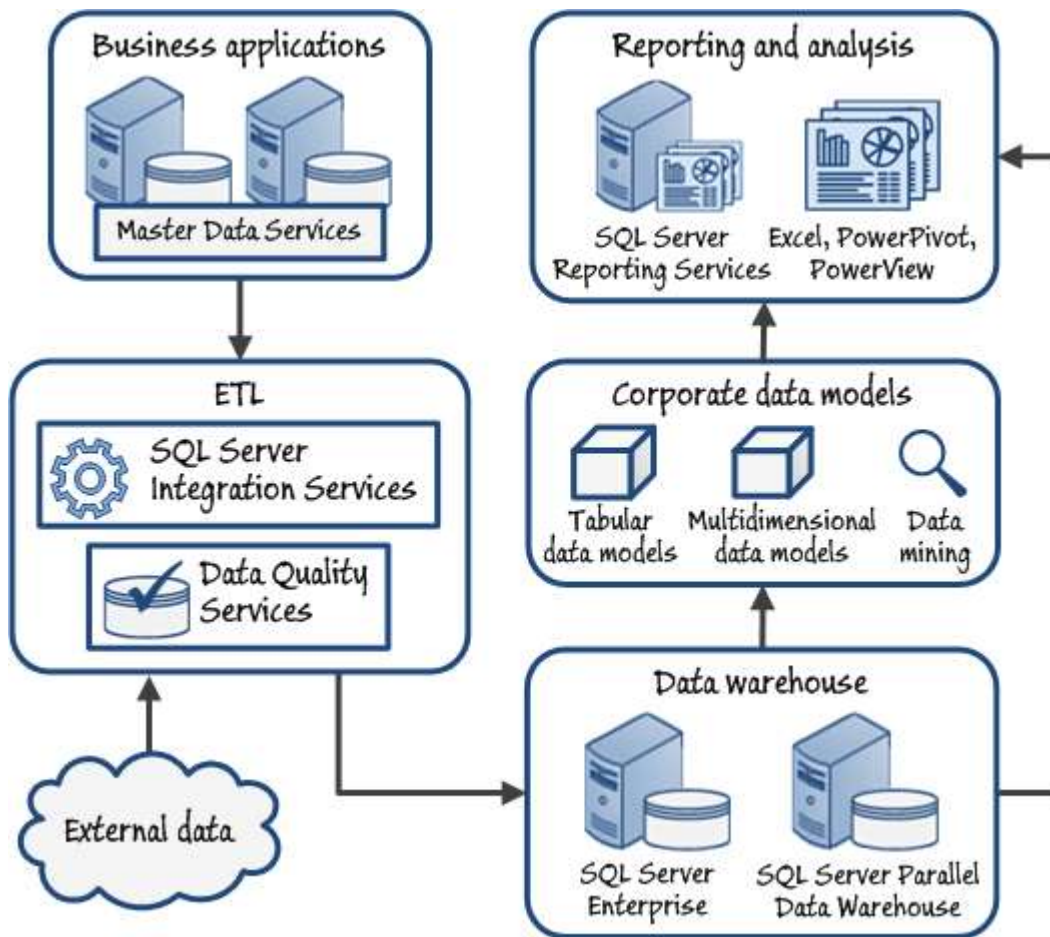


Figure 1
Overview of a typical enterprise data warehouse and BI implementation

Business Data Sources

Most of the data in an enterprise BI solution originates in business applications as the result of some kind of business activity. For example, a web retail business might use an e-commerce application to manage a product catalog and process sales transactions. In addition, most organizations use dedicated software applications for managing core business activities, such as a human resources and payroll system to manage employees, and a financial accounts application to record financial information such as revenue and profit.

Most business applications use some kind of data store, often a relational database. Additionally, the organization may implement a master data management solution to maintain the integrity of data definitions for key business entities across multiple systems. Microsoft SQL Server 2012 provides an enterprise-scale platform for relational database management, and includes Master Data Services – a solution for master data management.

ETL

One of the core components in an enterprise BI infrastructure is the ETL solution that extracts data from the various business data sources, transforms the data to meet structure, format, and validation requirements, and

loads the transformed data into the data warehouse. Typically, the ETL process performs an initial load of the data warehouse tables, and thereafter performs a regularly scheduled refresh cycle to synchronize the data warehouse with changes in the source systems. The ETL solution must therefore be able to detect changes in source systems in order to minimize the amount of time and processing overhead required to extract data, and intelligently insert or update rows in the data warehouse tables based on requirements to maintain historic values for analysis and reporting.

In most enterprise scenarios, the ETL process extracts data from business data sources and stores it in an interim staging area before loading the data warehouse tables. This enables the ETL process to consolidate data from multiple sources before the data warehouse load operation, and to restrict access to data sources and the data warehouse to specific time periods that minimize the impact of the ETL processing overhead on normal business activity.

SQL Server 2012 includes SQL Server Integration Services (SSIS), which provides a platform for building sophisticated ETL processes. An SSIS solution consists of one or more *packages*, each of which encapsulates a control flow for a specific part of the ETL process. Most control flows include one or more *data flows*, in which a pipeline-based, in-memory sequence of tasks is used to extract, transform, and load data. Figure 2 shows a simple SSIS data flow that extracts customer address data and loads it into a staging table.

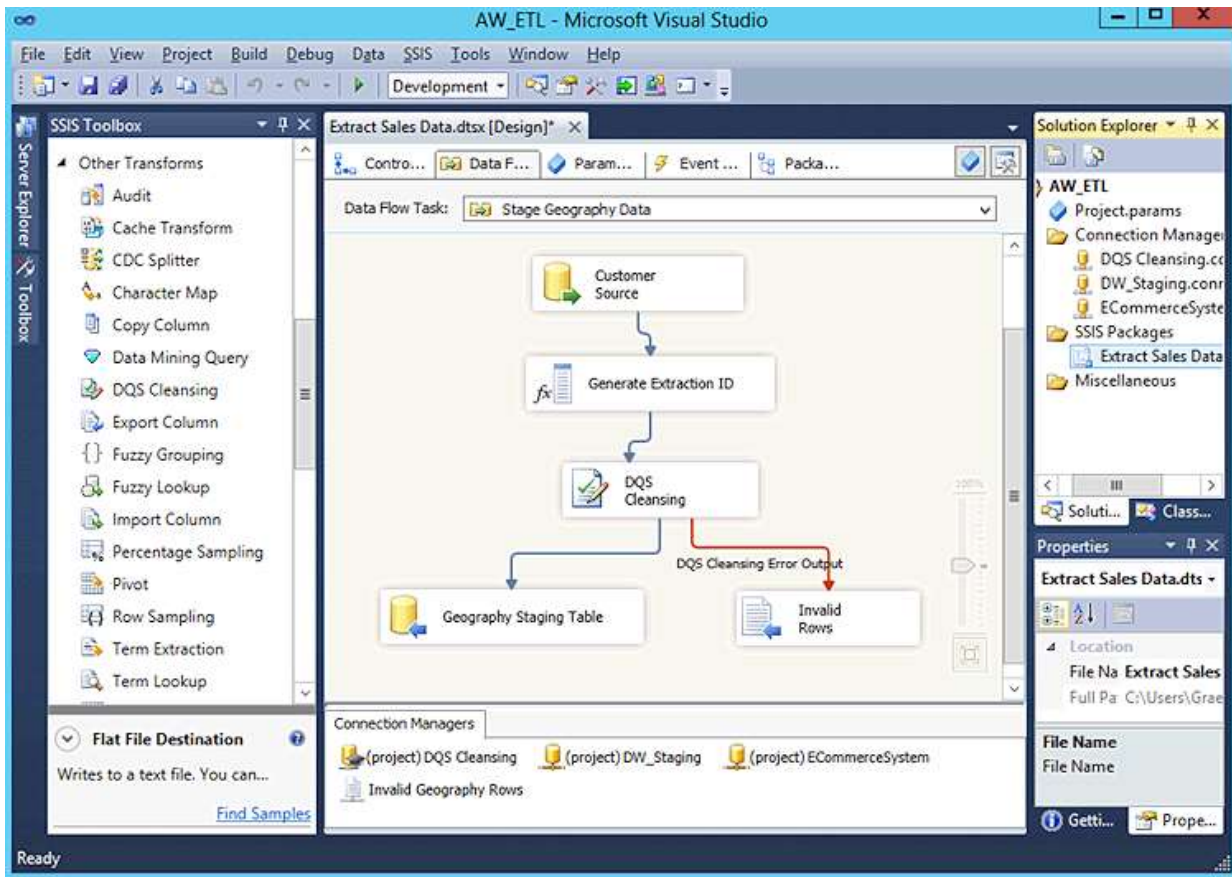


Figure 2
An SSIS data flow

Note that the data flow in Figure 2 includes a task named “DQS Cleansing”. This reflects the fact that an important requirement in an ETL solution is that the data being loaded into the data warehouse is as accurate and error-free as possible so that the data used for analysis and reporting is reliable enough to base important business decisions on.

To understand the importance of this, consider an e-commerce site in which users enter their address details in a free-form text box. This can lead to data input errors (for example, mistyping **New York** as **New Yok**) and inconsistencies (for example, one user might enter **United States**, while another enters **USA**, and a third enters **America**). While these errors and inconsistencies may not materially affect the operations of the business application (it is unlikely that any of these issues would prevent the delivery of an order), they may cause inaccuracies in reports and analyses. For example, a report showing total sales by country would include there separate values for the United States, which could be misleading when displayed as a chart.

To help resolve these kinds of data quality issue, SQL Server 2012 includes Data Quality Services (DQS). Business users with a deep knowledge of the data, who are often referred to as *data stewards*, can create DQS knowledge bases that define common errors and synonyms that should be corrected to the leading values. For example, Figure 3 shows the DQS knowledge base referenced by the data flow in Figure 2, in which **New Yok**, **New York City**, and **NYC** are all corrected to the leading value **New York**.

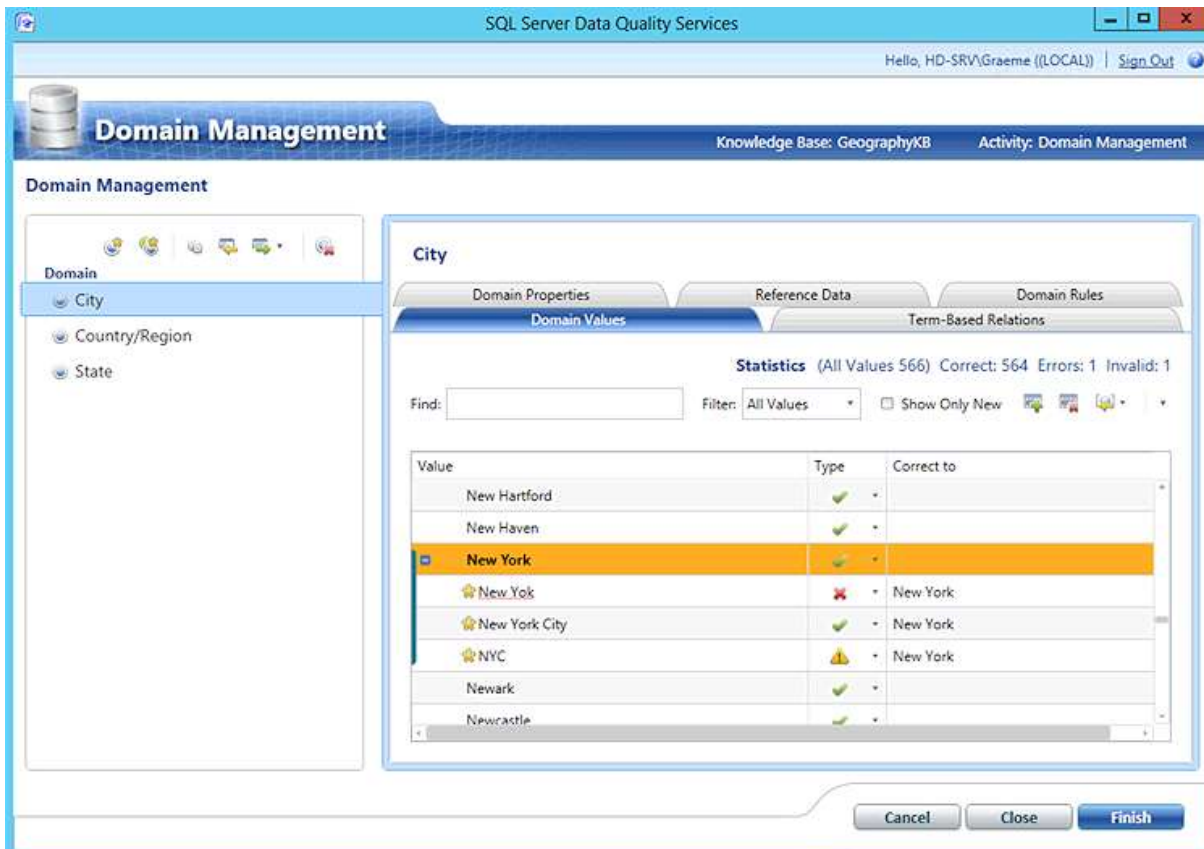


Figure 3
A DQS knowledge base

The Data Warehouse

A data warehouse is a central repository for the data extracted by the ETL process, and provides a single, consolidated source of data for reporting and analysis. In some organizations the data warehouse is implemented as a single database containing data for all aspects of the business, while in others the data may be factored out into smaller, departmental data marts.

Data Warehouse Schema Design

Regardless of the specific topology used, most data warehouses are based on a *dimensional* design in which numeric business measures such as revenue, cost, stock units, account balances, and so on are stored in business process-specific *fact tables*. For example, measures related to sales orders in the web retail business discussed previously might be stored in fact table named **FactInternetSales**. The facts are stored at a specific level of granularity, known as the *grain* of the fact table, that reflects the lowest level at which you need to aggregate the data. In the example of the **FactInternetSales** table, you could store a record for each order, or you could enable aggregation at a lower level of granularity by storing a record for each line item within an order. Measures related to stock management might be stored in a **FactStockKeeping** table, and measures related to manufacturing production might be stored in a **FactManufacturing** table.

Records representing the business entities by which you want to aggregate the facts are stored in *dimension* tables. For example, records representing customers might be stored in a table named **DimCustomer**, and records representing products might be stored in a table named **DimProduct**. Additionally, most data warehouses include a dimension table to store time period values, for example a row for each day to which a fact record may be related. The dimension tables are *conformed* across the fact tables that represent the business processes being analyzed. That is to say, the records in a dimension table (for example **DimProduct**) represent an instance of a business entity (in this case, a product), that can be used to aggregate facts in any fact table that involves the same business entity (for example, the same **DimProduct** records can be used to represent products in the business processes represented by the **FactInternetSales**, **FactStockKeeping**, and **FactManufacturing** tables).

To optimize query performance, a dimensional model uses minimal joins between tables. This approach results in a *denormalized* database schema in which data redundancy is tolerated in order to reduce the number of tables and relationships. The ideal dimensional model is a *star* schema in which each fact table is related directly to its dimension tables as shown in Figure 4.

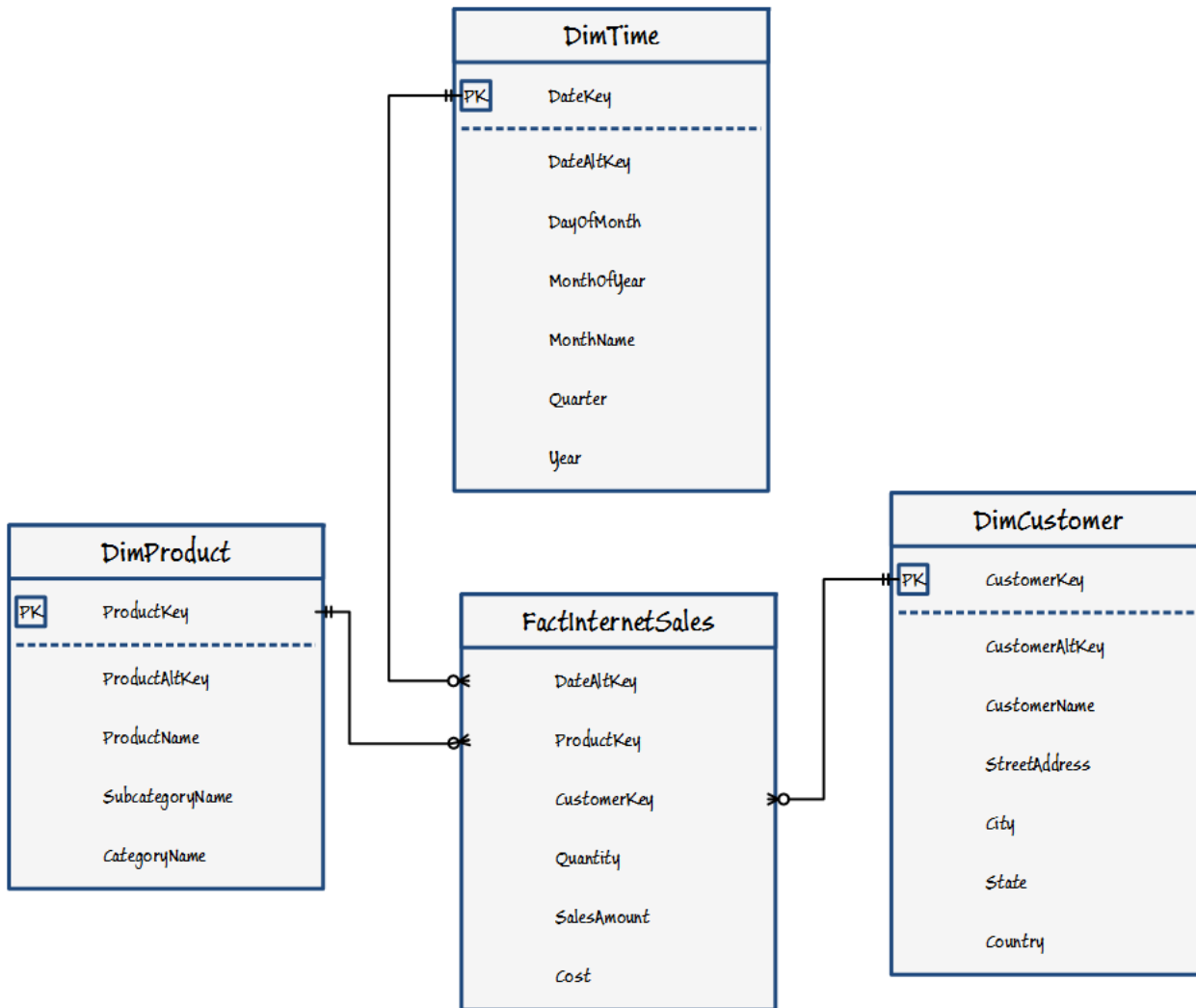


Figure 4
A star schema for a data warehouse

In some cases, there may be compelling reasons to partially normalize a dimension. One reason for this is to enable different fact tables to be related to attributes of the dimension at different levels of grain; for example, so that sales orders can be aggregated at the product level while marketing campaigns are aggregated at the product category level.

Another reason is to enable the same attributes to be shared across multiple dimensions; for example, storing geographical fields such as country, state, and city in a separate geography dimension table that can be used by both a customer dimension table and a sales region dimension table. This partially normalized design for a data warehouse is known as a *snowflake* schema, as shown in Figure 5.

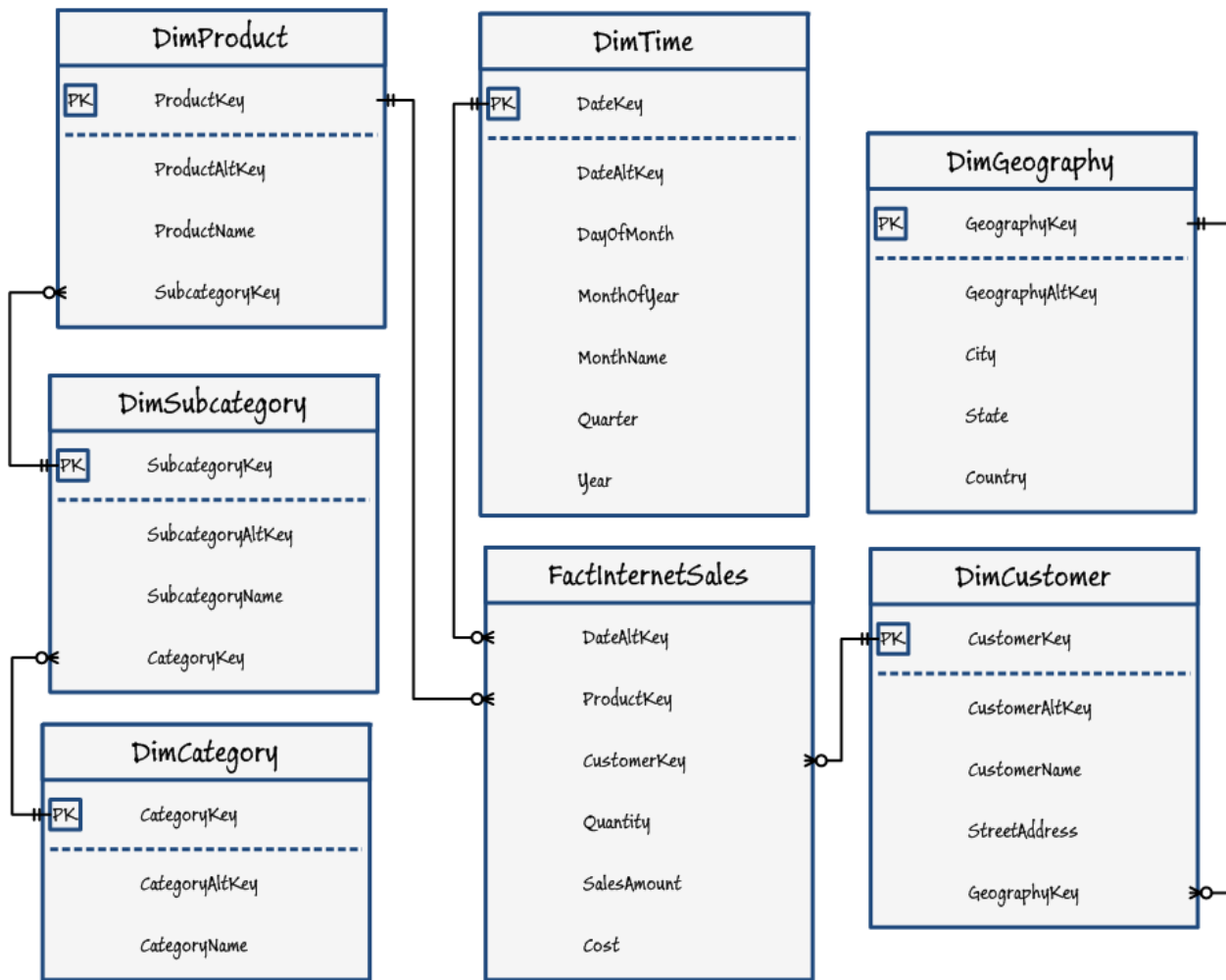


Figure 5
A snowflake schema for a data warehouse

Retaining Historical Dimension Values

Note that dimension records are uniquely identified by a key value. This key is usually a *surrogate* key that is specific to the data warehouse, and not the same as the key value used in the business data source where the dimension data originated. The original business key for the dimension records is usually retained in an *alternate* key column. The main reason for generating a surrogate key for dimension records is to enable the retention of historical versions of the dimension members.

For example, suppose a customer in New York registers with the web retailer discussed previously and makes some purchases. After a few years, the customer may move to Los Angeles and then make some more purchases. In the e-commerce application used by the web site, the customer's profile record is simply updated with the new address. However in the data warehouse, changing the address in the dim customer record would cause historical reporting of sales by city to become inaccurate – sales that were originally made to a customer in New York would now be counted as sales in Los Angeles. To avoid this problem, a new record for the

customer is created with a new, unique surrogate key value. The new record has the same alternate key value as the existing record, indicating that both records relate to the same customer, but at different points in time.

To keep track of which record represents the “active” instance of the customer at a specific time, the dimension table usually includes a Boolean column to indicate the current record, and often includes start and end date columns that reflect the period to which the record relates. This is shown in the following table.

CustKey	AltKey	Name	City	Current	Start	End
1001	212	Jesper Aaberg	New York	False	2008-01-01	2012-08-01
...
4123	212	Jesper Aaberg	Los Angeles	True	2012-08-01	NULL

This approach for retaining historical versions of dimension records is known as a *slowly changing dimension* (SCD), and it is important to understand that an SCD might contain multiple dimension records for the same business entity when implementing an ETL process that loads dimension tables, and when integrating analytical data (such as data obtained from HDInsight) with dimension records in a data warehouse.

Strictly speaking, the example shown in the table is a **type 2** slowly changing dimension. Some dimension attributes can be changed without retaining historical versions of the records, and these are known as **type 1** changes.

Implementing a Data Warehouse with SQL Server

SQL Server 2012 Enterprise Edition includes a number of features that make it a good choice for data warehousing. In particular, the SQL Server query engine can identify star schema joins and optimize query execution plans accordingly. Additionally, SQL Server 2012 introduces support for columnstore indexes, which use an in-memory compression technique to dramatically improve the performance of typical data warehouse queries.

For extremely large enterprise data warehouses that need to store huge volumes of data, Microsoft has partnered with hardware vendors to offer an enterprise data warehouse appliance. The appliance uses a special edition of SQL Server called Parallel Data Warehouse (PDW), in which queries are distributed in parallel across multiple processing and storage nodes in a “shared nothing” architecture. This technique is known as massively parallel processing (MPP), and offers extremely high query performance and scalability.

For more information about SQL Server Parallel Data Warehouse, see [Appliance: Parallel Data Warehouse \(PDW\)](#) on the SQL Server website.

Corporate Data Models

A data warehouse is designed to structure business data in a schema that makes it easy to write queries that aggregate data, and many organizations perform analysis and reporting directly from the data warehouse tables. However, most enterprise BI solutions include analytical data models as a layer of abstraction over the data warehouse to improve performance of complex analytical queries, and to define additional analytical structures such as hierarchies of dimension attributes. For example, a data model may aggregate sales by customer at the

country, state, city, and individual customer levels as well as exposing key performance indicators such as defining a score that compares actual sales revenue to targeted sales revenue.

These data models are usually created by BI developers and used as a data source for corporate reporting and analytics. Analytical data models are generically known as *cubes*, because they represent a multidimensional view of the data warehouse tables in which data can be “sliced and diced” to show aggregated measure values across multiple dimension members.

SQL Server 2012 Analysis Services (SSAS) supports the following two data model technologies:

- **Multidimensional databases.** Multidimensional SSAS databases support traditional OLAP cubes that are stored on disk, and are compatible with SSAS databases created with earlier releases of SQL Server Analysis Services. Multidimensional data models support a wide range of analytical features. A multidimensional database can also host data mining models, which apply statistical algorithms to datasets in order to perform predictive analysis.
- **Tabular databases.** Tabular databases were introduced in SQL Server 2012 and use an in-memory approach to analytical data modeling. Tabular data models are based on relationships defined between tables, and are generally easier to create than multidimensional models while offering comparable performance and functionality. Tabular models support most (but not all) of the capabilities of multidimensional models. Additionally, they provide an “upscale” path for personal data models created with PowerPivot in Microsoft Excel.

When you install SSAS, you must choose between multidimensional and tabular modes. An instance of SSAS installed in multidimensional mode cannot host tabular databases, and conversely a tabular instance cannot host multidimensional databases. You can however install two instances of SSAS (one in each mode) on the same server.

Figure 6 shows a multidimensional data model project in which a cube contains a **Date** dimension. The **Date** dimension includes two hierarchies, one for fiscal periods and another for calendar periods.

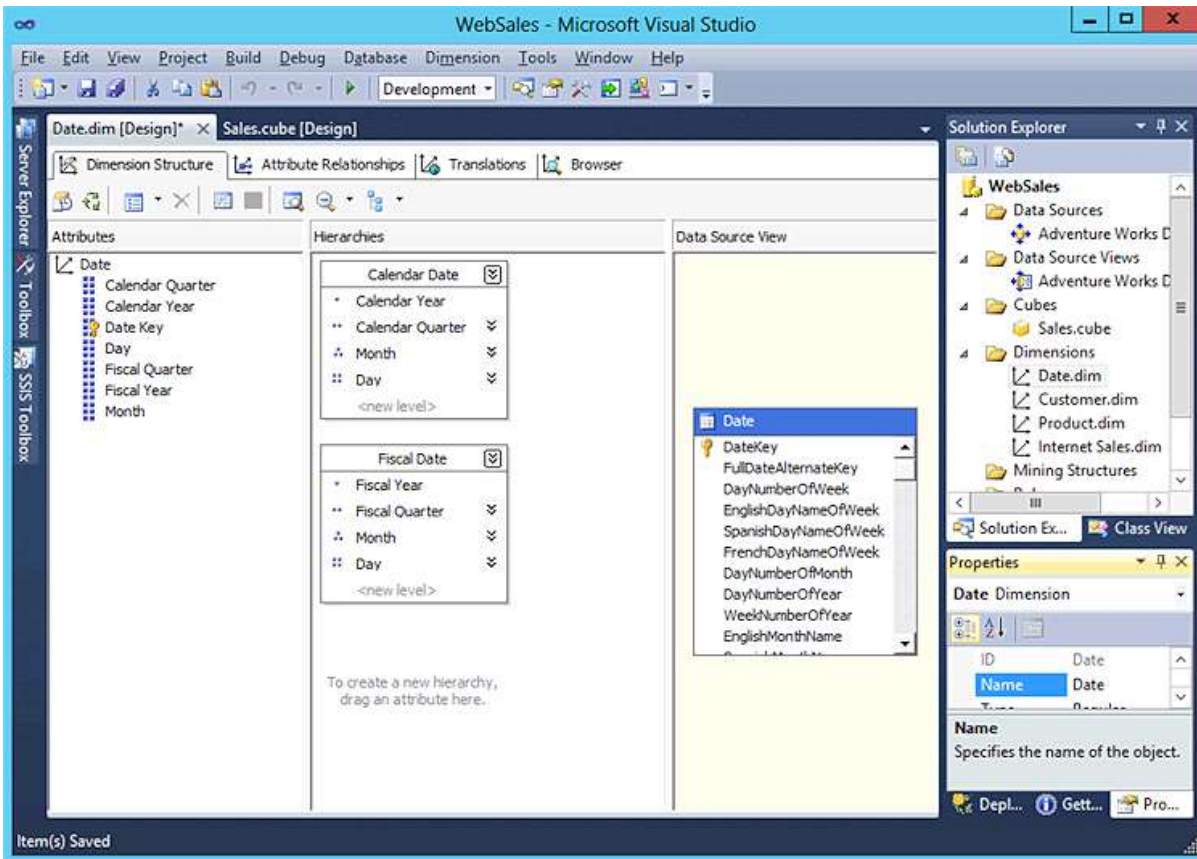


Figure 6
A multidimensional data model project

Figure 7 shows a tabular data model that includes the same dimensions and hierarchies as the multidimensional model in Figure 6.

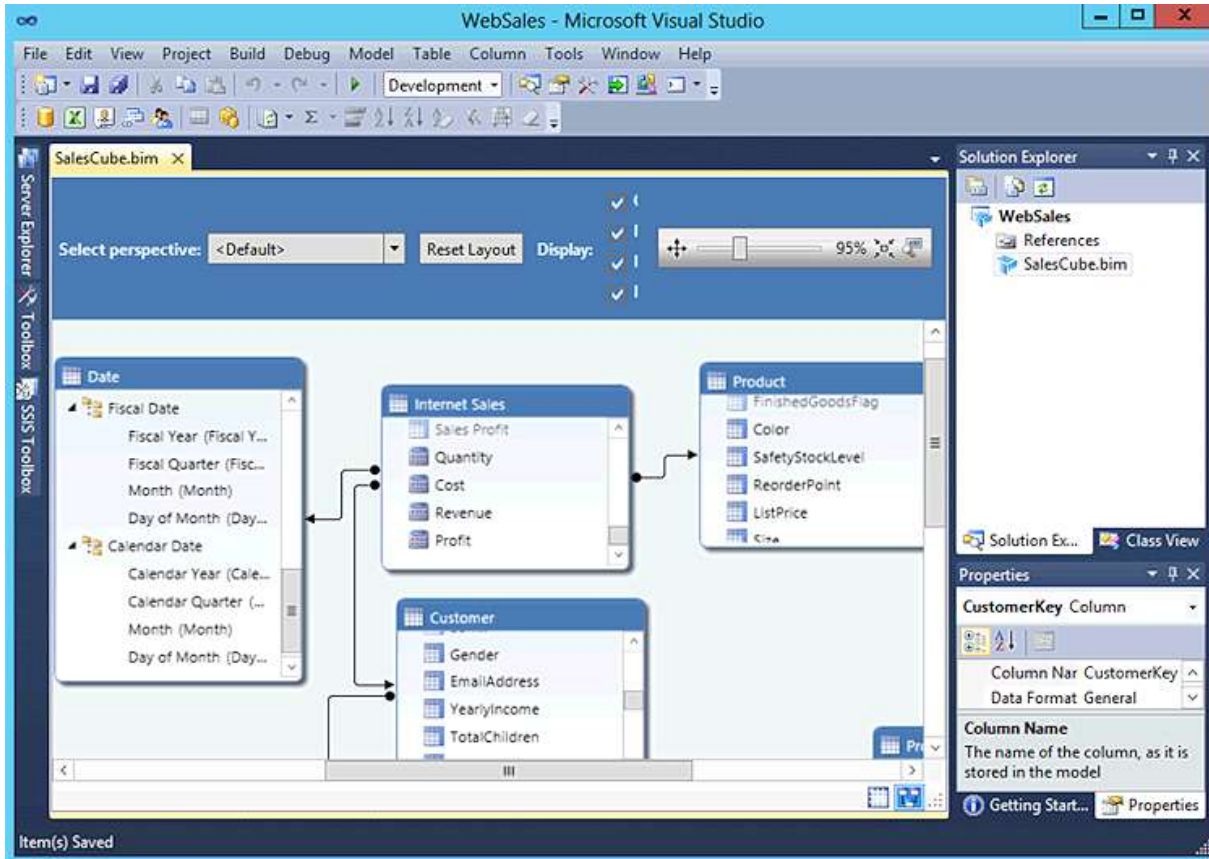


Figure 7
A tabular data model project

This appendix has described only the key aspects of data warehousing and BI that are necessary to understand how to integrate HDInsight into a BI solution. If you want to learn more about building BI solutions with Microsoft SQL Server and other components of the Microsoft BI platform, consider referring to “The Microsoft Data Warehouse Toolkit” (Wiley) or attending Microsoft Official Curriculum courses 10777 (“Implementing a Data Warehouse with Microsoft SQL Server 2012”), 10778 (“Implementing Data Models and Reports with Microsoft SQL Server 2012”), and 20467 (“Designing Business Intelligence Solutions with Microsoft SQL Server 2012”). You can find details of all Microsoft training courses at [Training and certification for Microsoft products and technologies](#) on the Microsoft.com Learning website.