

se trata como el resultado de un proceso generativo que incluye variables ocultas. LDA asume que hay un vocabulario fijo de palabras, y el número de temas latentes está predefinido y permanece constante. LDA asume que cada tema latente sigue una distribución de Dirichlet [30] sobre el vocabulario, y cada documento se representa como una mezcla aleatoria de temas latentes.

La figura 9-4 ilustra las intuiciones detrás de LDA. El lado izquierdo de la figura muestra cuatro temas construidos a partir de un corpus, donde cada tema contiene una lista de las palabras más importantes del vocabulario. Los cuatro temas de ejemplo están relacionados con el problema, la política, los nervios y el informe. Para cada documento, se elige una distribución sobre los temas, como se muestra en el histograma de la derecha. A continuación, se elige una asignación de tema para cada palabra del documento y se elige la palabra del tema correspondiente (discos de colores). En realidad, solo el documento

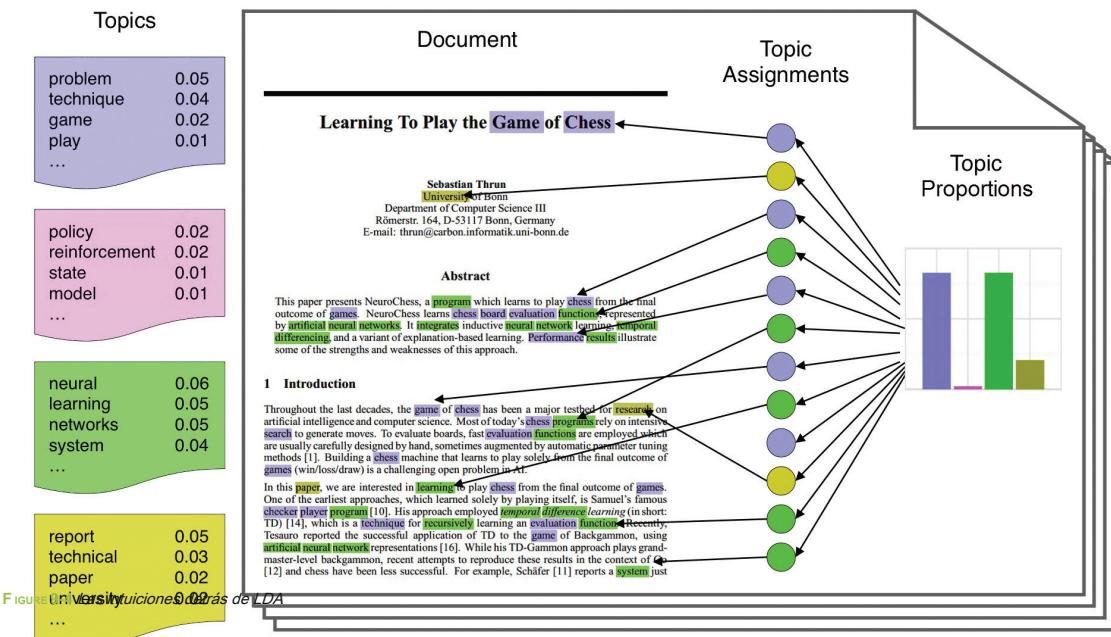
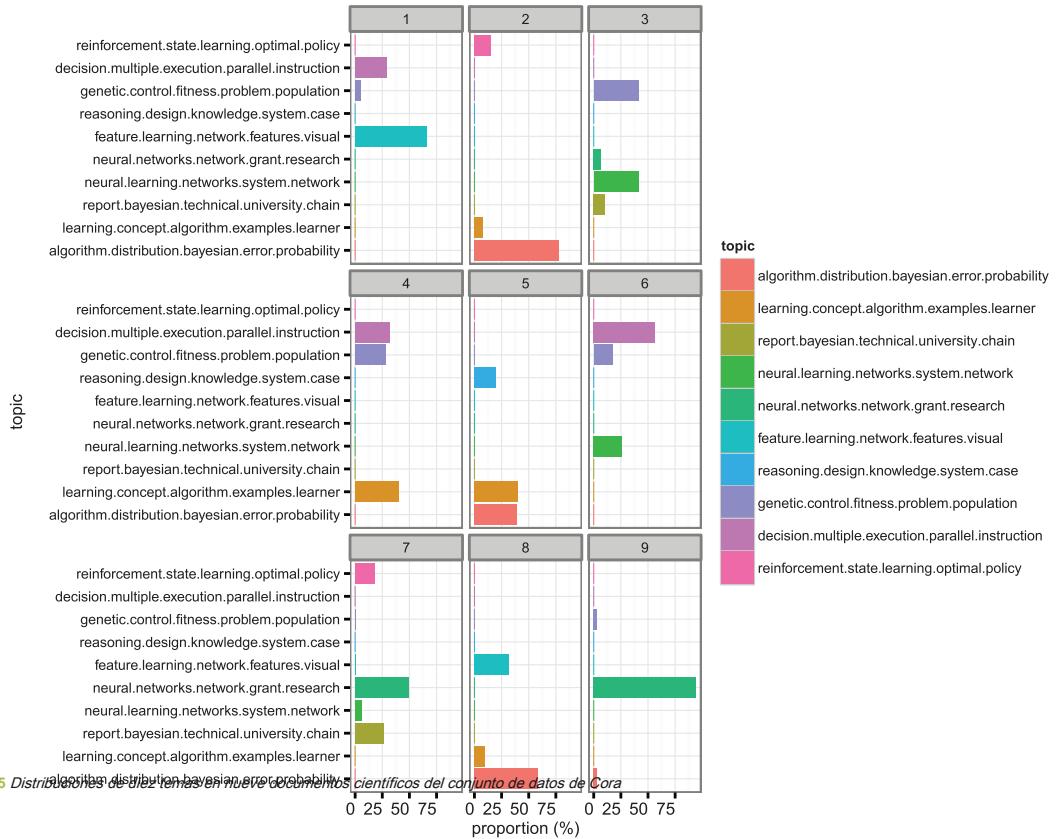


FIGURE 9-4 Intuiciones detrás de LDA

El lector puede consultar el artículo original [29] para conocer los detalles matemáticos de LDA. Básicamente, LDA puede verse como un caso de estimación bayesiana jerárquica con una distribución posterior a los datos de grupo, como documentos con temas similares.

Muchas herramientas de programación proporcionan paquetes de software que pueden realizar LDA sobre conjuntos de datos. R viene con un lida paquete [31] que tiene funciones integradas y conjuntos de datos de muestra. los lida El paquete fue desarrollado por el grupo de investigación de David M. Blei [32]. La figura 9-5 muestra las distribuciones de diez temas en nueve documentos científicos extraídos aleatoriamente de cora conjunto de datos del lida paquete. los cora El conjunto de datos es una colección de 2.410 documentos científicos extraídos del motor de búsqueda Cora [33].



El código que sigue muestra cómo generar un gráfico similar a la Figura 9-5 usando R y paquetes de complementos como lda y ggplot.

```

require ("ggplot2")
require ("reshape2")
require ("lda")

# cargar documentos y vocabulario
datos (cora.documents)
datos (cora.vocab)

theme_set (theme_bw ())

# Número de grupos de temas para mostrar
K <- 10

# Número de documentos para mostrar
N <- 9

```

```

resultado <- lda.collapsed.gibbs.sampler(cora.documents,
                                         K, ## Num clusters
                                         cora.vocab,
                                         25, ## Num iteraciones
                                         0.1,
                                         0.1,
                                         compute.log.likelihood = TRUE)

# Obtenga las palabras principales del grupo
top.words <- top.topic.words(resultado $ temas, 5, by.score = TRUE)

# construir proporciones de temas
topic.props <- t(resultado $ sumas_documento) / colSums(resultado $ sumas_documento)

document.samples <- sample(1: dim(topic.props)[1], N)
topic.props <- topic.props[document.samples,]

topic.props [is.na(topic.props)] <- 1 / K

colnames(topic.props) <- aplicar(top.words, 2, paste, collapse = "")

topic.props.df <- melt(data.frame(topic.props),
                       documento = factor(1:N),
                       variable.name = "tema",
                       id.vars = "documento")

qplot(tema, valor * 100, relleno = tema, estadística = "identidad",
      ylab = "proporción (%)", datos = topic.props.df,
      geom = "histograma") +
  tema(axis.text.x = element_text(angle = 0, hjust = 1, size = 12)) + coord_flip() +
  facet_wrap(~ documento, ncol = 3)

```

Los modelos de temas se pueden utilizar en el modelado de documentos, la clasificación de documentos y el filtrado colaborativo [29]. Los modelos de tema no solo se pueden aplicar a datos textuales, sino que también pueden ayudar a anotar imágenes. Así como un documento puede considerarse una colección de temas, las imágenes pueden considerarse una colección de características de la imagen.

9.7 Determinación de sentimientos

Además del TFIDF y los modelos de temas, el equipo de ciencia de datos puede querer identificar los sentimientos en los comentarios de los usuarios y las revisiones de los productos ACME. *Análisis de los sentimientos* se refiere a un grupo de tareas que utilizan estadísticas y procesamiento del lenguaje natural para identificar y extraer información subjetiva de los textos.

Los primeros trabajos sobre análisis de sentimientos se centraron en detectar la polaridad de las reseñas de productos de Epinions [34] y las reseñas de películas de InternetMovieDatabase (IMDb) [35] a nivel de documento. El trabajo posterior se ocupa del análisis de sentimientos a nivel de la oración [36]. Más recientemente, el enfoque se ha desplazado al nivel de frase [37] y formas de texto corto en respuesta a la popularidad de los servicios de microblogging como Twitter [38, 39, 40, 41, 42].

Intuitivamente, para realizar un análisis de sentimientos, uno puede construir manualmente listas de palabras con sentimientos positivos (como **brillante, asombroso, y espectacular**) y sentimientos negativos (como **horrible, estúpido, y horrible**). El trabajo relacionado ha señalado que se puede esperar que este enfoque logre una precisión de alrededor del 60% [35], y es probable que se supere mediante el examen de las estadísticas del corpus [43].

Los métodos de clasificación como Bayes ingenuo, como se presentó en el Capítulo 7, maximumentropía (MaxEnt) y máquinas de vectores de soporte (SVM) se utilizan a menudo para extraer estadísticas de corpus para el análisis de sentimientos. Investigaciones relacionadas han descubierto que estos clasificadores pueden obtener una precisión de alrededor del 80% [35, 41, 42] en el análisis de sentimientos sobre datos no estructurados. Uno o más de estos clasificadores se pueden aplicar a datos no estructurados, como reseñas de películas o incluso tweets.

El corpus de reseñas de películas de Pang et al. [35] incluye 2.000 reseñas de películas recopiladas de un archivo de IMDb del grupo de noticias rec.arts.movies.reviews [43]. Estas críticas de películas se han etiquetado manualmente en 1,000 críticas positivas y 1,000 críticas negativas.

Dependiendo del clasificador, es posible que los datos deban dividirse en conjuntos de entrenamiento y prueba. Como se vio anteriormente en el Capítulo 7, una regla práctica útil para dividir datos es producir un conjunto de entrenamiento mucho más grande que el conjunto de prueba. Por ejemplo, una división 80/20 produciría el 80% de los datos como conjunto de entrenamiento y el 20% como conjunto de prueba.

A continuación, se capacita a uno o más clasificadores sobre el conjunto de capacitación para aprender las características o patrones que residen en los datos. Las etiquetas de opinión en los datos de prueba están ocultas para los clasificadores. Después del entrenamiento, los clasificadores se prueban sobre el conjunto de pruebas para inferir las etiquetas de sentimiento. Finalmente, el resultado se compara con las etiquetas de opinión originales para evaluar el desempeño general del clasificador.

El código que sigue está escrito en Python usando la biblioteca Natural Language Processing Toolkit (NLTK) (<http://nltk.org/>). Muestra cómo realizar un análisis de sentimientos utilizando el clasificador ingenuo de Bayes sobre el corpus de reseñas de películas.

El código divide las 2,000 revisiones en 1,600 revisiones como el conjunto de entrenamiento y 400 revisiones como el conjunto de pruebas. El clasificador ingenuo de Bayes aprende del conjunto de entrenamiento. Los sentimientos en el conjunto de prueba se ocultan al clasificador. Para cada revisión en el conjunto de entrenamiento, el clasificador aprende cómo cada característica impacta el sentimiento del resultado. A continuación, se le da al clasificador el conjunto de pruebas. Para cada revisión del conjunto, predice cuál debería ser el sentimiento correspondiente, dadas las características de la revisión actual.

```
importar nltk.classify.util
de nltk.classify importar NaiveBayesClassifier de nltk.corpus importar
movietitis de colecciones importar defaultdict
```

```
importar numpy como np
```

```
# definir una división 80/20 para entrenamiento / prueba
SPLIT = 0.8
```

```
def word_feats (palabras):
    feats = defaultdict (lambda: False) para palabra en
    palabras:
        hazañas [palabra] = Verdadero
    volver hazañas
```

```
posids = movie_reviews.fileids ('pos')
```

```

negids = movie_reviews.fileids ('neg')

posfeats = [(word_feats (movie_reviews.words (fileids = [f])), 'pos')
             para f en posids]
negfeats = [(word_feats (movie_reviews.words (fileids = [f])), 'neg')
             para f en negids]

cutoff = int (len (posfeats) * SPLIT)

trainfeats = negfeats [: cutoff] + posfeats [: cutoff] testfeats = negfeats [cutoff:] +
posfeats [cutoff:]

print 'Entrenar en% d instancias \ nPrueba en% d instancias%' (len (trainfeats),
len (testfeats))

clasificador = NaiveBayesClassifier.train (trainfeats)
imprimir 'Precisión:', nltk.classify.util.accuracy (clasificador, testfeats)

classifier.show_most_informative_features ()

# preparar matriz de confusión

pos = [classifier.classify (fs) for (fs, l) in posfeats [cutoff:]] pos = np.array (pos)

neg = [clasificador.clasificar (fs) para (fs, l) en negfeats [cutoff:]] neg = np.array (neg)

imprimir 'Matriz de confusión:'
print '\t' * 2, 'Clase prevista' print '-' * 40

imprimir '| \ t% d (TP) \ t | \ t% d (FN) \ t | Clase real % (
(pos == 'pos'). sum (), (pos == 'neg'). sum ())
imprimir '-' * 40
imprimir '| \ t% d (FP) \ t | \ t% d (TN) \ t | % (
(neg == 'pos'). sum (), (neg == 'neg'). sum ())
imprimir '-' * 40

```

El resultado que sigue muestra que el clasificador de Bayes ingenuo se entrena en 1.600 instancias y se prueba en 400 instancias del corpus de la película. El clasificador alcanza una precisión del 73,5%. La mayoría de las características de información para las revisiones positivas del corpus incluyen palabras como **sobresaliente**, **vulnerable**, y **asombroso**; y palabras como **insultante**, **ridículo**, y **desinteresado** son las características más informativas para las reseñas negativas. Al final, el resultado también muestra la matriz de confusión correspondiente al clasificador para evaluar más el desempeño.

Entrenar en 1600 instancias Probar
en 400 instancias Precisión: 0,735

Características más informativas

excepcional = Verdadero	pos: neg	=	13,9: 1,0
insulting = True	neg: pos	=	13,7: 1,0
vulnerable = Verdadero	pos: neg	=	13,0: 1,0
ridículo = Verdadero	neg: pos	=	12,6: 1,0
uninvolving = Verdadero	neg: pos	=	12,3: 1,0
asombroso = Verdadero	pos: neg	=	11,7: 1,0
evita = Verdadero	pos: neg	=	11,0: 1,0
fascinación = Verdadero	pos: neg	=	10,3: 1,0
animadores = Verdadero	pos: neg	=	10,3: 1,0
símbolo = Verdadero	pos: neg	=	10,3: 1,0

Matriz de confusión:

Clase prevista		Clase real
195 (TP)	5 (FN)	
101 (FP)	99 (TN)	

Como se discutió anteriormente en el Capítulo 7, una **matriz de confusión** es un diseño de tabla específico que permite la visualización del desempeño de un modelo sobre el conjunto de prueba. Cada columna de rowand corresponde a una clase posible en el conjunto de datos. Cada celda de la matriz muestra el número de ejemplos de prueba para los cuales la clase real es la fila y la clase predicha es la columna. Los buenos resultados corresponden a números grandes en la diagonal principal (TP y TN) y elementos pequeños, idealmente cero, o ff-diagonales (FP y FN). La Tabla 9-7 muestra la matriz de confusión de la salida del programa anterior para el conjunto de pruebas de 400 revisiones. Debido a que un clasificador bien realizado debe tener una matriz de confusión con números grandes para TP y TN e idealmente números cercanos a cero para FP y FN, se puede concluir que el clasificador de Bayes ingenuo tiene muchos falsos negativos y no se desempeña muy bien en esta prueba. conjunto.

T PODER 9-7 ConfusionMatrix para el conjunto de pruebas de ejemplo

		Positivo	Negativo
Real	Positivo	195 (TP)	5 (FN)
	Negativo	101 (FP)	99 (TN)

El capítulo 7 ha introducido algunas medidas para evaluar el desempeño de un clasificador más allá de la matriz de confusión. La precisión y la memoria son dos medidas que se usan comúnmente para evaluar tareas relacionadas con el análisis de texto. Las definiciones de precisión y recuperación se dan en las ecuaciones 9-8 y 9-9.

$$\text{Precisión} = \frac{TP}{TP + FP} \quad (9-8)$$

$$\text{Recordar} = \frac{TP}{TP + FN} \quad (9-9)$$

Precisión se define como el porcentaje de documentos en los resultados que son relevantes. Si ingresando palabra clave **bTeléfono**, el motor de búsqueda devuelve 100 documentos, y 70 de ellos son relevantes, la precisión del resultado del motor de búsqueda es del 0,7%.

Recordar es el porcentaje de documentos devueltos entre todos los documentos relevantes del corpus. Si ingresando palabra clave **bTeléfono**, el motor de búsqueda devuelve 100 documentos, solo 70 de los cuales son relevantes y no devuelve 10 documentos relevantes adicionales, la retirada es $70 / (70 + 10) = 0,875$.

Por lo tanto, el clasificador ingenuo de Bayes de la Tabla 9-7 recibe una recuperación de $195 / (195 + 5) = 0,975$ y una precisión de $195 / (195 + 101) \approx 0,659$.

La precisión y la memoria son conceptos importantes, ya sea que la tarea se refiera a la recuperación de información de un motor de búsqueda o al análisis de texto en un corpus finito. Idealmente, un buen clasificador debería lograr precisión y recordar cerca de 1.0. En la recuperación de información, una puntuación de precisión perfecta de 1.0 significa que todos los resultados recuperados por una búsqueda fueron relevantes (pero no dice nada sobre si se recuperaron todos los documentos relevantes), mientras que una puntuación de recuperación perfecta de 1.0 significa que todos los documentos relevantes fueron recuperados por la búsqueda. (pero no dice nada acerca de cuántos documentos irrelevantes también se recuperaron). Por tanto, tanto la precisión como el recuerdo se basan en la comprensión y la medida de la relevancia. En realidad, es difícil para un clasificador lograr tanto una alta precisión como una alta recuperación. Para el ejemplo de la Tabla 9-7, el clasificador ingenuo de Bayes tiene una memoria alta pero una precisión baja. Por lo tanto, el equipo de ciencia de datos debe verificar la limpieza de los datos, optimizar el clasificador y encontrar formas de mejorar la precisión mientras se mantiene la alta recuperación.

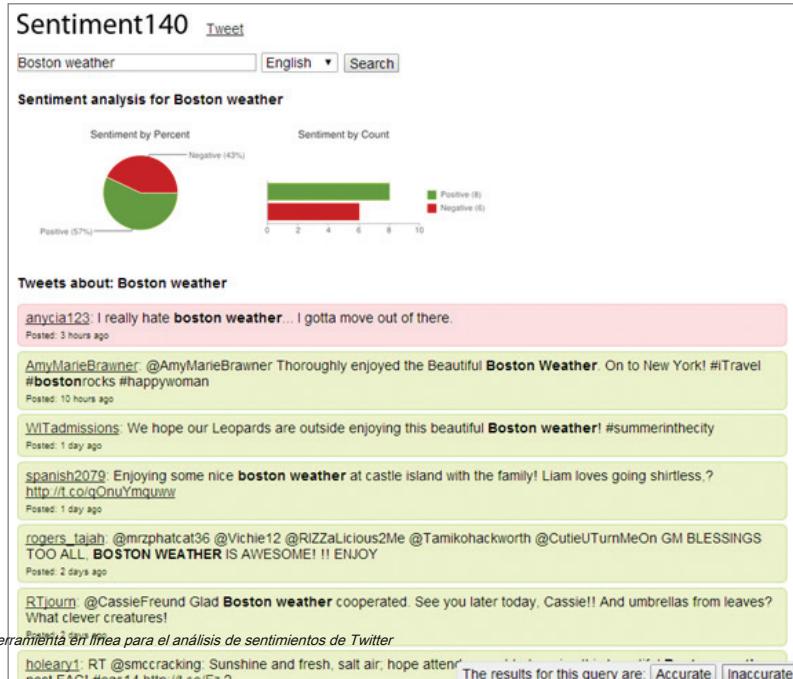
Los clasificadores determinan los sentimientos basándose únicamente en los conjuntos de datos en los que están capacitados. El dominio de los conjuntos de datos y las características de las características determinan lo que pueden aprender los clasificadores de conocimiento. Por ejemplo, *ligero* es una característica positiva para las reseñas en computadoras portátiles, pero no necesariamente para reseñas en carretillas o libros de texto. Además, los conjuntos de entrenamiento y prueba deben compartir características similares para que los clasificadores se desempeñen bien. Por ejemplo, los clasificadores capacitados en reseñas de películas generalmente no deben probarse en tweets o comentarios de blogs.

Tenga en cuenta que un nivel de sentimiento absoluto no es necesariamente muy informativo. En cambio, se debe establecer una línea de base y luego compararla con los últimos valores observados. Por ejemplo, una proporción del 40% de tweets positivos sobre un tema frente al 60% de negativos podría no considerarse una señal de que un producto no tiene éxito si otros productos exitosos similares tienen una proporción similar basada en la psicología de cuando las personas tuitean.

El ejemplo anterior demuestra cómo utilizar el ingenuo Bayes para realizar un análisis de sentimientos. El ejemplo se puede aplicar a los tweets en ACME **bTeléfono** y **bEbook** simplemente reemplazando el corpus de reseñas de películas con los tweets preetiquetados. También se pueden utilizar otros clasificadores en lugar de Bayes ingenuo.

El corpus de reseñas de películas contiene solo 2.000 reseñas; por lo tanto, es relativamente fácil etiquetar manualmente cada reseña. Para el análisis de sentimientos basado en grandes cantidades de datos de transmisión, como millones o miles de millones de tweets, es menos factible recopilar y construir conjuntos de datos de tweets que sean lo suficientemente grandes o etiquetar manualmente cada uno de los tweets para entrenar y probar uno o más clasificadores. Hay dos formas populares de lidar con este problema. La primera forma de construir datos preetiquetados, como se ilustra en un trabajo reciente de Go et al. [41] y Pak y Paroubek [42], es aplicar supervisión y usar emoticonos como :) y :(para indicar si un tweet contiene sentimientos positivos o negativos. Las palabras de estos tweets pueden, a su vez, usarse como pistas para clasificar los sentimientos de futuros tweets. Go et al. [41] utilizan métodos de clasificación que incluyen ingenuos Bayes, MaxEnt, y SVMover los conjuntos de datos de entrenamiento y prueba para realizar clasificaciones de sentimientos. Su demo está disponible en

<http://www.sentiment140.com>. La Figura 9-6 muestra los sentimientos resultantes de una consulta contra el término "clima de Boston" en un conjunto de tweets. Los espectadores pueden marcar el resultado como exacto o inexacto, y dicha retroalimentación se puede incorporar en el entrenamiento futuro del algoritmo.



Los emoticonos hacen que sea fácil y rápido detectar los sentimientos de millones o miles de millones de tweets. Sin embargo, el uso de emoticonos como único indicador de sentimientos a veces puede ser engañoso, ya que los emoticonos no necesariamente se corresponden.

La figura 9-7 contiene un amplio tweet mostrado en

ment.



Para abordar este problema, la investigación relacionada suele utilizar Amazon Mechanical Turk (MTurk) [44] para recopilar reseñas con etiquetas humanas. MTurk es un mercado de Internet de crowdsourcing que permite a las personas o empresas coordinar el uso de la inteligencia humana para realizar tareas que son difíciles de realizar para las computadoras. En muchos casos, se ha demostrado que MTurk recopila información humana mucho más rápido en comparación con los canales tradicionales, como las encuestas puerta a puerta. Para la tarea de análisis de sentimiento de ejemplo, el equipo de ciencia de datos puede publicar los tweets recopilados de la Sección 9.3 en MTurk como tareas de inteligencia humana (HIT). Luego, el equipo puede pedir a los trabajadores humanos que etiqueten cada tweet como positivo, neutral o negativo. El resultado se puede utilizar para entrenar a uno o más clasificadores o probar el desempeño de los clasificadores. La figura 9-8 muestra una tarea de muestra en MTurk relacionada con el análisis de sentimientos.

The screenshot shows the Amazon Mechanical Turk (AMT) interface. At the top, there are tabs for 'Your Account', 'HITS', 'Qualifications', and a notification that '284,274 HITS available now'. Below the tabs, there are search filters: 'Find HITS containing' and 'that pay at least \$ 0.00', with checkboxes for 'for which you are qualified' and 'require Master Qualification'. A timer indicates '00:00:00 of 30 minutes'. Buttons for 'Accept HIT' and 'Skip HIT' are present. Summary statistics show 'Total Earned: \$0.15' and 'Total HITS Submitted: 3'. Below this, a task description reads: 'Judge the Relevance and Sentiment of content about PayPal (206511)'. It specifies the requester as CrowdFlower, a reward of '\$0.05 per HIT', 12 HITS Available, and a duration of 30 minutes. Qualifications required are noted as 'HIT approval rate (%) is greater than 96'. The main task area is titled 'Task preview' and contains instructions and a post for judgment. The post text is: 'POST: RT @pesoexchanger Exchange your Paypal funds to CASH NOW! No more waiting for days! No more delays! Get money even on Holidays or... http://t.co/0JJqe3YatW'. A note below the post says 'Please pay close attention to the post.' A question asks 'What is the author's sentiment (feeling) throughout the post (outlined in red) as it relates to PayPal?'. Five radio button options are provided: 'Very Positive', 'Slightly Positive', 'Neutral', 'Slightly Negative', and 'Very Negative'. At the bottom of the task preview, there are 'Accept HIT' and 'Skip HIT' buttons, along with a link to report the HIT.

FIGURE 9-8 Amazonas Turco Mecánico

FAQ | Contact Us | Careers at Amazon | Developers | Press | Policies | Blog
©2005-2013 Amazon.com, Inc. or its Affiliates. An [amazon.com](#) company

9.8 Obtener conocimientos

Hasta ahora, este capítulo ha analizado varias tareas de análisis de texto, incluida la recopilación de texto, la representación de texto, TFIDF, modelos de temas y análisis de sentimientos. Esta sección muestra cómo ACME utiliza estas técnicas para obtener información sobre las opiniones de los clientes sobre sus productos. Para mantener el ejemplo simple, esta sección solo usa *bTeléfono* para ilustrar los pasos.

Correspondiente a la fase de recopilación de datos, el equipo de Data Science ha utilizado **teléfono** como palabra clave para recopilar más de 300 reseñas de un popular sitio web de reseñas técnicas.

Las 300 reseñas se visualizan como una nube de palabras después de eliminar las palabras vacías. UN *nube de palabras o nube de etiquetas* es una representación visual de datos textuales. Las etiquetas son generalmente palabras sueltas y la importancia de cada palabra se muestra con el tamaño de fuente o el color. La figura 9-9 muestra la nube de palabras creada a partir de las 300 revisiones. Las reseñas se han doblado previamente en mayúsculas y minúsculas y se han tokenizado en minúsculas, y las palabras vacías se han eliminado del texto. Una palabra que aparece con más frecuencia en la Figura 9-9 se muestra con un tamaño de fuente más grande. La orientación de cada palabra es solo con fines estéticos. La mayor parte del gráfico está ocupada por



FIGURE 9-9 Nube de palabras sobre las 300 reseñas de bPhone.

Afortunadamente, el popular sitio web de revisión técnica permite a los usuarios proporcionar calificaciones en una escala de uno a cinco cuando publican reseñas. El equipo puede dividir las revisiones en subgrupos utilizando esas calificaciones.

Para revelar más información, el equipo puede eliminar palabras como **teléfono**, **bPhone**, y **CUMBRE**, que no son muy útiles para el estudio. La investigación relacionada a menudo se refiere a estas palabras como **palabras vacías específicas del dominio**.

La figura 9-10 muestra la nube de palabras correspondiente a 50 reseñas de cinco estrellas extraídas de los datos. Tenga en cuenta que los tonos de gris son solo para fines estéticos. El resultado sugiere que los clientes están satisfechos con la *vendedor*, la *marca*, y el *producto*, y ellos *recomendar*b Llame a sus amigos y familiares.

La figura 9-11 muestra la nube de palabras de 70 reseñas de una estrella. Las palabras **sim** y **botón** ocurren con suficiente frecuencia que sería recomendable probar las revisiones que contienen estos términos y determinar qué se dice sobre los botones y las tarjetas SIM. Las nubes de palabras pueden revelar información útil más allá de los términos más destacados. Por ejemplo, el gráfico de la Figura 9-11 contiene extrañamente palabras como **robado** y

Venezuela. A medida que el equipo de Data Science investiga las historias detrás de estas palabras, descubre que estas palabras aparecen en reseñas de 1 estrella porque hay algunos vendedores no autorizados de Venezuela que venden productos robados.



FIGURE 9



FIGURE 9-11 Nube de palabras sobre reseñas de una estrella

TFIDF se puede utilizar para resaltar las palabras informativas en las reseñas. La Figura 9-12 muestra un subconjunto de revisiones en las que cada palabra con un tamaño de fuente más grande corresponde a un valor TFIDF más alto. Cada revisión se considera un documento. Con TFIDF, los analistas de datos pueden revisar rápidamente las revisiones e identificar qué aspectos se perciben para hacer de bPhone un buen producto o un mal producto.



FIGURE 9-12 Reseñas destacadas por valores TEDF

Los modelos de tema, como LDA, pueden clasificar las revisiones en temas. Las Figuras 9-13 y 9-14 muestran gráficos circulares de temas como resultados de la LDA. Estas cifras se producen con herramientas y tecnologías como Python, NoSQL y D3.js. La figura 9-13 visualiza diez temas creados a partir de reseñas de cinco estrellas. Cada tema se centra en un aspecto diferente que puede caracterizar las reseñas. El tamaño del disco representa el peso de una palabra. En un entorno interactivo, al pasar el mouse sobre un tema se muestran las palabras completas y sus pesos correspondientes.

La figura 9-14 visualiza diez temas de reseñas de una estrella. Por ejemplo, el tema de la parte inferior derecha contiene palabras como *botón*, *poder*, y *roto*, lo que puede indicar que bPhone tiene problemas relacionados con el botón y la fuente de alimentación. El equipo de ciencia de datos puede rastrear estas revisiones y averiguar si ese es realmente el caso.

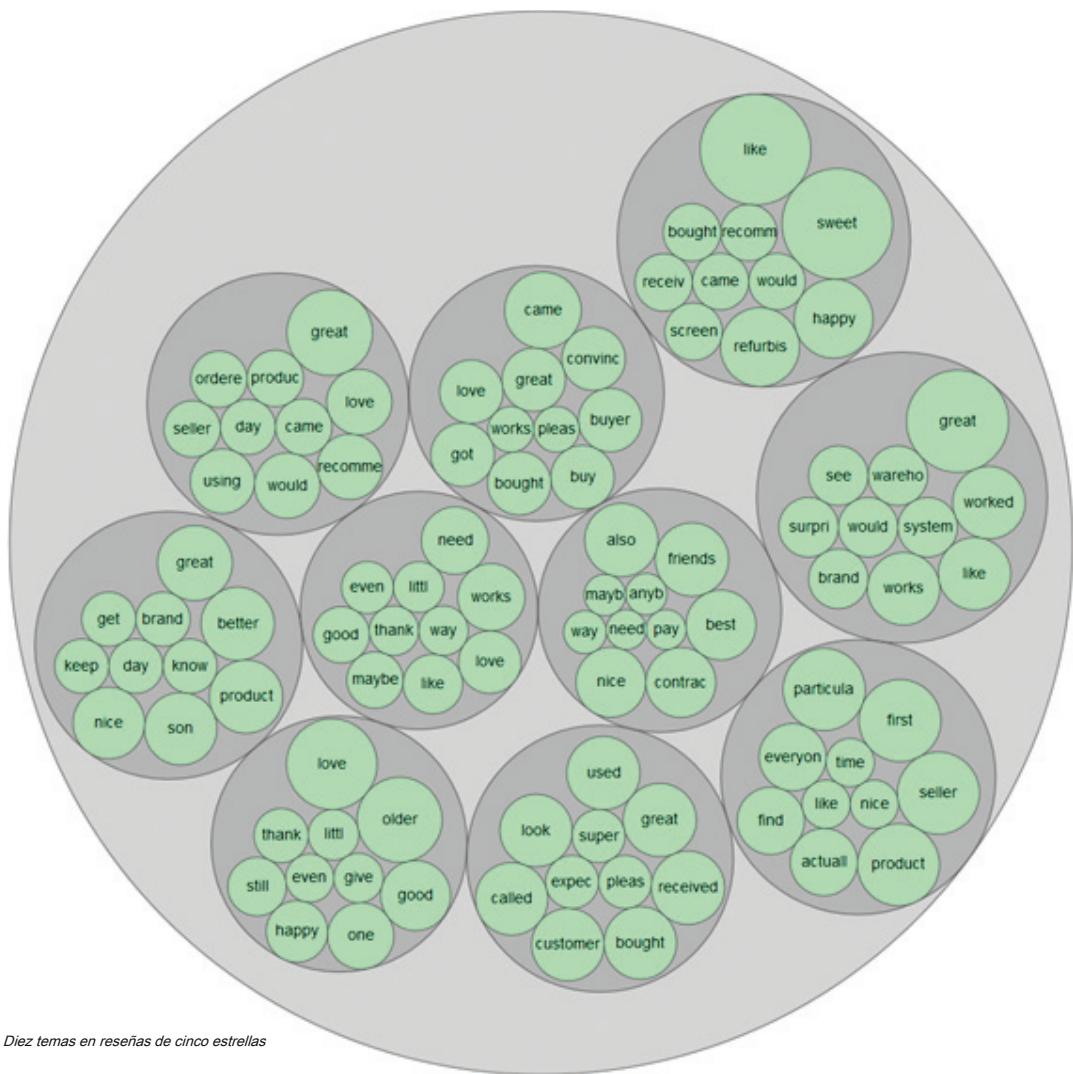


FIGURE 9-13 Diez temas en reseñas de cinco estrellas

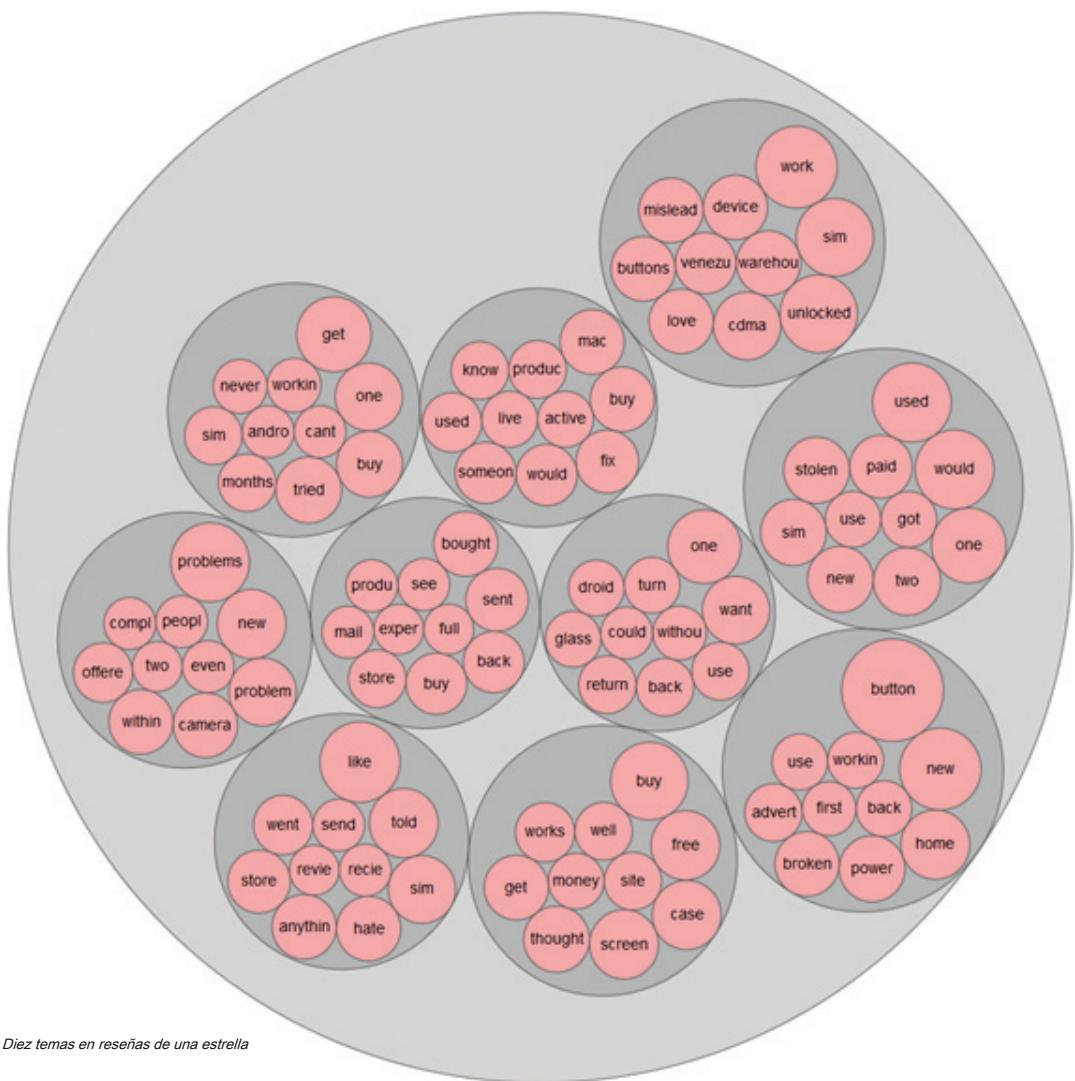


FIGURE 9-14 Diez temas en reseñas de una estrella

La figura 9-15 proporciona una forma diferente de visualizar los temas. Se extraen cinco temas de cinco estrellas y reseñas de una estrella, respectivamente. En un entorno interactivo, al pasar el mouse sobre un tema se resaltan las palabras correspondientes en este tema. Las capturas de pantalla de la Figura 9-15 se tomaron cuando

Tema 4 se resalta para ambos grupos. El peso de una palabra en un tema viene indicado por el tamaño del disco.

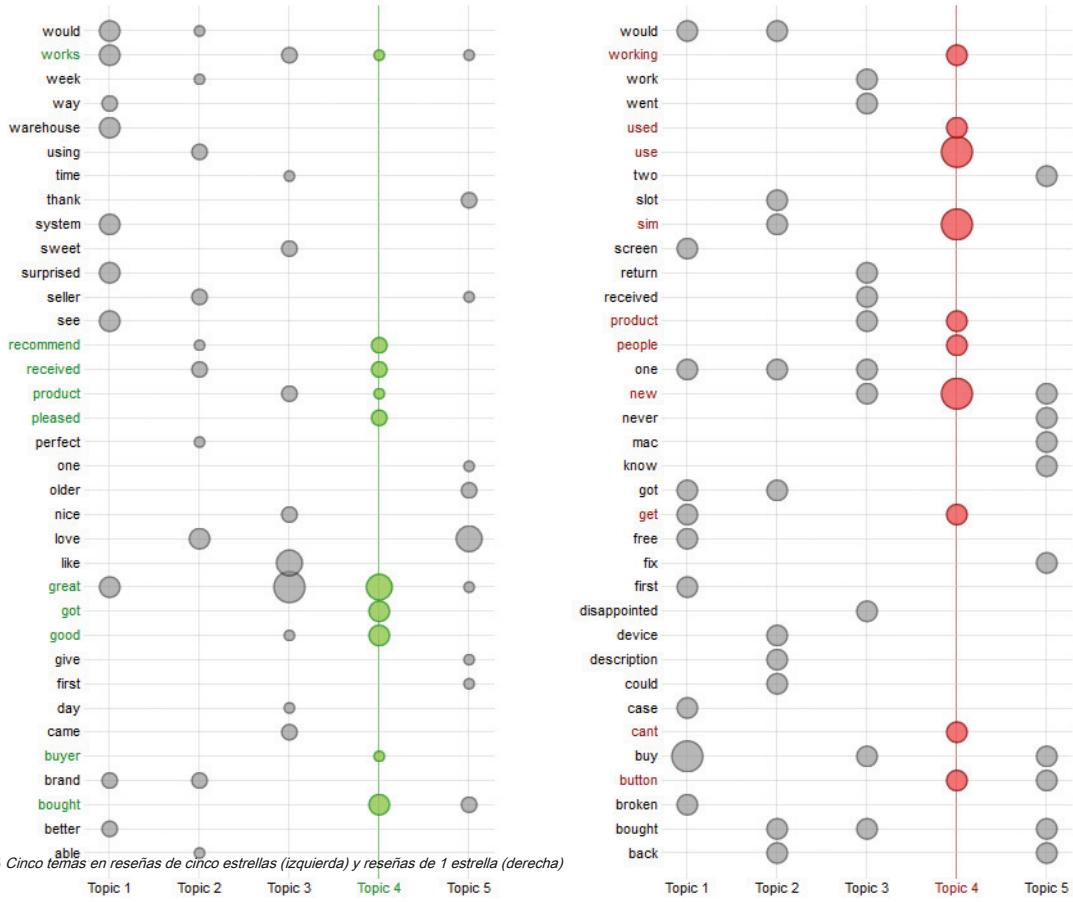


FIGURE 9-15 Cinco temas en reseñas de cinco estrellas (izquierda) y reseñas de 1 estrella (derecha)

El equipo de Data Science también ha realizado un análisis de sentimiento de más de 100 tweets del popular sitio de microblogging Twitter. El resultado se muestra en la Figura 9-16. El lado izquierdo representa sentimientos negativos y el lado derecho representa sentimientos positivos. Verticalmente, los tweets se han colocado aleatoriamente con fines estéticos. Cada tweet se muestra como un disco, donde el tamaño representa el número de seguidores del usuario que realizó el tweet original. El tono de color de un disco representa la frecuencia con la que se ha retuiteado este tweet. La cifra indica que la mayoría de los clientes están satisfechos con el bPhone de ACME.

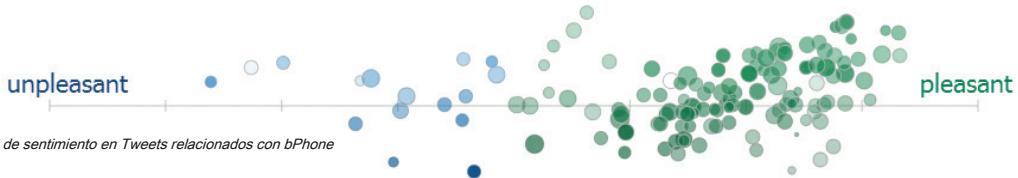


FIGURE 9-16 Análisis de sentimiento en Tweets relacionados con bPhone

Resumen

Este capítulo ha discutido varias subáreas del análisis de texto, incluido el análisis, la búsqueda y recuperación y la minería de texto. Con un ejemplo de gestión de marca, el capítulo habla de un proceso de análisis de texto típico: (1) recopilar texto sin procesar, (2) representar texto, (3) usar TFIDF para calcular la utilidad de cada palabra en los textos, (4) categorizar documentos por temas que utilizan modelos de temas, (5) análisis de sentimientos y (6) obtención de mayores conocimientos.

El análisis general del texto no es una tarea trivial. En correspondencia con el ciclo de vida analítico de datos, las partes de un proyecto de análisis de texto que consumen más tiempo a menudo no son la realización de estadísticas ni la implementación de algoritmos. Lo más probable es que el equipo pase la mayor parte del tiempo formulando el problema, obteniendo los datos y preparándolos.

Ejercicios

1. ¿Cuáles son los principales desafíos del análisis de textos?
2. ¿Qué es un corpus?
3. ¿Cuáles son las palabras comunes (como *a*, *y*, *de*) ¿llamado?
4. ¿Por qué no podemos usar TF solo para medir la utilidad de las palabras?
5. ¿Qué es una advertencia de las FDI? ¿Cómo aborda TFIDF el problema?
6. Nombra tres beneficios de usar TFIDF.
7. ¿Qué métodos se pueden utilizar para el análisis de sentimientos?
8. ¿Cuál es la definición de *tema* en modelos de temas?
9. Explique las ventajas y desventajas de precisión y recuperación.
10. Realice el modelado de temas LDA en el corpus Reuters-21578 utilizando Python y LDA. El NLTK tiene ya vienen con el corpus Reuters-21578. Para importar este corpus, ingrese el siguiente comentario en el indicador de Python:

```
desde nltk.corpus import reuters
```

El LDA ya ha sido implementado por varias bibliotecas de Python como gensim [45]. Utilice una de estas bibliotecas o implemente su propia LDA para realizar el modelado de temas en el corpus Reuters-21578.

- 11.** Elija un tema de su interés, como una película, una celebridad o cualquier palabra de moda. Luego recopila 100 tweets relacionados con este tema. Etiquételos manualmente como positivos, neutrales o negativos. Luego, divídilos en 80 tweets como el conjunto de entrenamiento y los 20 restantes como el conjunto de prueba. Ejecute uno o más clasificadores sobre estos tweets para realizar análisis de sentimientos. ¿Cuáles son la precisión y el recuerdo de estos clasificadores? ¿Qué clasificador se desempeña mejor que los demás?

Bibliografía

- [1] Dr. Seuss, "Green Eggs and Ham", Nueva York, NY, EE. UU., RandomHouse, 1960.
- [2] M. Steinbach, G. Karypis y V. Kumar, "Una comparación de técnicas de agrupación de documentos", *KDD Taller sobre minería de textos*, 2000.
- [3] "El Proyecto Penn Treebank", Universidad de Pennsylvania [en línea]. Disponible: <http://www.cis.upenn.edu/~treebank/home.html>. [Consultado el 26 de marzo de 2014].
- [4] Wikipedia, "Lista de API abiertas" [en línea]. Disponible: http://en.wikipedia.org/wiki/List_of_open_APIs. [Consultado el 27 de marzo de 2014].
- [5] ProgrammableWeb, "API Directory" [en línea]. Disponible: <http://www.programmableweb.com/apis/>. [Consultado el 27 de marzo de 2014].
- [6] Twitter, "Sitio de desarrolladores de Twitter" [en línea]. Disponible: <https://dev.twitter.com/>. [Consultado el 27 de marzo de 2014].
- [7] "Herramientas Curl y libcurl" [en línea]. Disponible: <http://curl.haxx.se/>. [Consultado el 27 de marzo 2014].
- [8] "XML Path Language (XPath) 2.0", World Wide Web Consortium, 14 de diciembre de 2010. [En línea]. Disponible: <http://www.w3.org/TR/xpath20/>. [Consultado el 27 de marzo de 2014].
- [9] "Gnip: la fuente de datos sociales", GNIP [en línea]. Disponible: <http://gnip.com/>. [Accedido 12 Junio de 2014].
- [10] "DataSift: Tome decisiones con datos sociales", DataSift [en línea]. Disponible: <http://cambio.de.datos.com/>. [Consultado el 12 de junio de 2014].
- [11] G. Salton y C. Buckley, "Enfoques de ponderación de términos en la recuperación automática de texto", en *Información Procesamiento y Gestión*, 1988, págs. 513-523.
- [12] GK Zipf, *El comportamiento humano y el principio de mínimo esfuerzo*, Reading, MA: Addison-Wesley, 1949.
- [13] ME Newman, "Leyes de poder, distribuciones de Pareto y ley de Zipf", *Física contemporánea*, vol. 46, No. 5, págs. 323-351, 2005.
- [14] Y. Li, D. McLean, ZA Bandar, JD O'Shea y K. Crockett, "Similitud de oraciones basada en redes semánticas y estadísticas de corpus", *Transacciones IEEE sobre conocimiento e ingeniería de datos*, vol. 18, no. 8, págs. 1138-1150, 2006.
- [15] WN Francis y H. Kucera, "Brown Corpus Manual", 1979. [En línea]. Disponible: <http://icame.uib.no/brown/bcm.html>.
- [dieciséis] "Evaluación crítica de la extracción de información en biología (BioCreative)" [en línea]. Disponible: <http://www.biocreative.org/>. [Consultado el 2 de abril de 2014].
- [17] JJ Godfrey y E. Holliman, "Switchboard-1 Release 2", Linguistic Data Consortium, Filadelfia, 1997. [En línea]. Disponible: <http://catalog.ldc.upenn.edu/LDC97S62>. [Consultado el 2 de abril de 2014].

- [18] P. Koehn, "Europarl: A Parallel Corpus for Statistical Machine Translation", *MT Cumbre*, 2005.
- [19] N. Seco, T. Veale y J. Hayes, "Una métrica de contenido de información intrínseca para la similitud semántica en WordNet", *ECAI*, vol. 16, págs. 1089–1090, 2004.
- [20] P. Resnik, "Uso del contenido de información para evaluar la similitud semántica en una taxonomía", En *Actas de la XIV Conferencia Conjunta Internacional sobre Inteligencia Artificial (IJCAI'95)*, vol. 1, págs. 448–453, 1995.
- [21] T. Pedersen, "Las medidas de contenido de información de similitud semántica funcionan mejor sin texto etiquetado con sentido", *Tecnologías del lenguaje humano: Conferencia anual de 2010 del Capítulo norteamericano de la Asociación de Lingüística Computacional*, págs. 329–332, junio de 2010.
- [22] CD Manning, P. Raghavan y H. Schütze, "Document and Query Weighting Schemes", en *Introducción a la recuperación de información*, Cambridge, Reino Unido, Cambridge University Press, 2008, pág. 128.
- [23] M. Porter, "Porter's English Stop Word List", 12 de febrero de 2007. [En línea]. Disponible: <http://bola de nieve. tartarus.org/algorithms/english/stop.txt>. [Consultado el 2 de abril de 2014].
- [24] M. Steinbach, G. Karypis y V. Kumar, "Una comparación de técnicas de agrupación de documentos", *KDD taller sobre minería de textos*, vol. 400, no. 1 de 2000.
- [25] T. Joachims, "Inferencia transductiva para la clasificación de texto utilizando máquinas de vectores de soporte", *ICML*, vol. 99, págs. 200–209, 1999.
- [26] P. Soucy y GW Mineau, "Un algoritmo KNN simple para la categorización de texto", *ICDM*, págs. 647–648, 2001.
- [27] B. Liu, X. Li, WS Lee y PS Yu, "Clasificación de texto por etiquetado de palabras", *AAAI*, vol. 4, págs. 425–430, 2004.
- [28] DM Blei, "Modelos temáticos probabilísticos", *Comunicaciones de la ACM*, vol. 55, no. 4, págs. 77–84, 2012.
- [29] DM Blei, AY Ng y MI Jordan, "Latent Dirichlet Allocation", *Revista de aprendizaje automático Investigación*, vol. 3, págs. 993–1022, 2003.
- [30] T. Minka, "Estimación de una distribución de Dirichlet", 2000.
- [31] J. Chang, "Ida: métodos de muestreo de Gibbs contraídos para modelos de tema", *GRÚA*, 14 de octubre de 2012. [En línea]. Disponible: <http://cran.r-project.org/web/packages/lda/>. [Consultado el 3 de abril de 2014].
- [32] DM Blei, "Software de modelado de temas" [en línea]. Disponible: <http://www.cs.princeton.edu/~blei/topicmodeling.html>. [Consultado el 11 de junio de 2014].
- [33] A. McCallum, K. Nigam, J. Rennie y K. Seymore, "Un enfoque de aprendizaje automático para la construcción de motores de búsqueda específicos de dominio", *IJCAI*, vol. 99, 1999.
- [34] PD Turney, "¿Me gusta o no? Orientación semántica aplicada a la clasificación no supervisada de reseñas", *Actas de la Asociación de Lingüística Computacional*, págs. 417–424, 2002.
- [35] B. Pang, L. Lee y S. Vaithyanathan, "Thumbs Up? Clasificación de sentimientos mediante técnicas de aprendizaje automático", *Procedimientos de EMNLP*, págs. 79–86, 2002.
- [36] M. Hu y B. Liu, "Minería y resumen de reseñas de clientes", *Actas de la Décima MCA Conferencia internacional SIGKDD sobre descubrimiento de conocimiento y minería de datos*, págs. 168–177, 2004.
- [37] A. Agarwal, F. Biadsy y KR McKeown, "Análisis de polaridad a nivel de frase contextual usando puntuación de efecto léxico y N-gramos sintácticos", *Actas de la 12a Conferencia del Capítulo Europeo de la Asociación de Lingüística Computacional*, págs. 24–32, 2009.

- [38] B. O'Connor, R. Balasubramanyan, BR Routledge y NA Smith, "From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series", *Actas de la Cuarta Conferencia Internacional sobre Weblogs y Redes Sociales, ICWSM '10*, págs. 122-129, 2010.
- [39] A. Agarwal, B. Xie, I. Vovsha, O. Rambow y R. Passonneau, "Análisis de sentimiento de los datos de Twitter", *En Actas del Taller de Idiomas en las Redes Sociales*, págs. 30-38, 2011.
- [40] H. Saif, Y. He y H. Alani, "Semantic Sentiment Analysis of Twitter", *Actas del 11 Conferencia Internacional sobre The SemanticWeb (ISWC'12)*, págs. 508-524, 2012.
- [41] A. Go, R. Bhayani y L. Huang, "Clasificación de sentimientos de Twitter mediante supervisión distante", *Informe del proyecto CS224N, Stanford*, págs. 1-12, 2009.
- [42] A. Pak y P. Paroubek, "Twitter como un corpus para el análisis de sentimientos y la minería de opiniones", *Actas de la Séptima Conferencia Internacional sobre Recursos y Evaluación del Lenguaje (LREC'10)*, págs. 19-21, 2010.
- [43] B. Pang y L. Lee, "Opinion Mining and Sentiment Analysis", *Fundamentos y Tendencias en Recuperación de información*, vol. 2, no. 1-2, págs. 1-135, 2008.
- [44] "Amazon Mechanical Turk" [en línea]. Disponible: <http://www.mturk.com/>. [Consultado el 7 de abril de 2014].
- [45] R. Řehůřek, "Biblioteca Python Gensim" [en línea]. Disponible: <http://radimrehurek.com/gensim/>. [Consultado el 8 de abril de 2014].

10

AdvancedAnalytics— Tecnología y herramientas: MapReduce y Hadoop

Conceptos clave

Hadoop

Ecosistema Hadoop

Mapa reducido

NoSQL

El Capítulo 4, "Teoría y métodos analíticos avanzados: agrupamiento", hasta el Capítulo 9, "Teoría y métodos analíticos avanzados: análisis de texto", cubrió varios métodos analíticos útiles para clasificar, predecir y examinar relaciones dentro de los datos. Este capítulo y el Capítulo 11, "Análisis avanzado: tecnología y herramientas: análisis en la base de datos", abordan varios aspectos de la recopilación, el almacenamiento y el procesamiento de datos estructurados y no estructurados, respectivamente. Este capítulo presenta algunas tecnologías y herramientas clave relacionadas con la biblioteca de software Apache Hadoop, "un marco que permite el procesamiento distribuido de grandes conjuntos de datos en grupos de computadoras usando modelos de programación simples" [1].

Este capítulo se centra en cómo Hadoop almacena datos en un sistema distribuido y cómo Hadoop implementa un paradigma de programación simple conocido como MapReduce. Aunque este capítulo hace algunas referencias específicas de Java, el único conocimiento prerequisito previsto es un conocimiento básico de programación. Además, los detalles específicos de Java para escribir un programa MapReduce para Apache Hadoop están más allá del alcance de este texto. Esta omisión puede parecer problemática, pero las herramientas del ecosistema Hadoop, como Apache Pig y Apache Hive, a menudo pueden eliminar la necesidad de codificar explícitamente un programa MapReduce. Junto con otras herramientas relacionadas con Hadoop, Pig y Hive se tratan en una parte de este capítulo que trata sobre el ecosistema Hadoop.

Para ilustrar el poder de Hadoop en el manejo de datos no estructurados, la siguiente discusión proporciona varios casos de uso de Hadoop.

10.1 Análisis de datos no estructurados

Antes de realizar el análisis de datos, los datos necesarios deben recopilarse y procesarse para extraer la información útil. El grado de procesamiento inicial y preparación de datos depende del volumen de datos, así como de lo sencillo que sea comprender la estructura de los datos.

Recuerde los cuatro tipos de estructuras de datos analizados en el Capítulo 1, "Introducción a Big Data Analytics":

- **Estructurado:** Un formato específico y coherente (por ejemplo, una tabla de datos)
- **Semiestructurada:** Un formato de autodescripción (por ejemplo, un archivo XML)
- **Cuasi estructurado:** Un formato algo inconsistente (por ejemplo, un hipervínculo)
- **No estructurado:** Un formato inconsistente (por ejemplo, texto o video)

Los datos estructurados, como las tablas del sistema de gestión de bases de datos relacionales (RDBMS), suelen ser el formato de datos más fácil de interpretar. Sin embargo, en la práctica sigue siendo necesario comprender los distintos valores que pueden aparecer en una determinada columna y lo que estos valores representan en diferentes situaciones (basándose, por ejemplo, en el contenido de las otras columnas del mismo registro). Además, algunas columnas pueden contener texto no estructurado u objetos almacenados, como imágenes o videos. Aunque las herramientas presentadas en este capítulo se centran en datos no estructurados, estas herramientas también se pueden utilizar para conjuntos de datos más estructurados.

10.1.1 Casos de uso

El siguiente material proporciona varios casos de uso de MapReduce. El paradigma MapReduce ofrece los medios para dividir una tarea grande en tareas más pequeñas, ejecutar tareas en paralelo y consolidar los resultados de las tareas individuales en el resultado final. Apache Hadoop incluye una implementación de software de MapReduce. Más adelante en este capítulo se proporcionan más detalles sobre MapReduce y Hadoop.

IBMWatson

En 2011, el sistema informático de IBM Watson participó en el programa de juegos de televisión de EE. UU. *Peligro contra dos de los mejores Peligro* campeones en la historia del programa. En el juego, a los concursantes se les proporciona una pista como "A él le gustan sus martinis agitados, no revueltos" y la respuesta correcta, formulada en forma de pregunta, sería: "¿Quién es James Bond?" Durante el torneo de tres días, Watson pudo derrotar a los dos concursantes humanos.

Para educar a Watson, se utilizó Hadoop para procesar diversas fuentes de datos, como enciclopedias, diccionarios, noticias, publicaciones y todo el contenido de Wikipedia [2]. Para cada pista proporcionada durante el juego, Watson tuvo que realizar las siguientes tareas en menos de tres segundos [3]:

- Deconstruir la pista proporcionada en palabras y frases
- Establecer la relación gramatical entre las palabras y las frases.
- Cree un conjunto de términos similares para usar en la búsqueda de respuesta de Watson
- Use Hadoop para coordinar la búsqueda de una respuesta en terabytes de datos
- Determinar posibles respuestas y asignar su probabilidad de ser correctas.
- Accionar el timbre
- Proporcionar una respuesta sintácticamente correcta en inglés.

Entre otras aplicaciones, Watson se está utilizando en la profesión médica para diagnosticar pacientes y proporcionar recomendaciones de tratamiento [4].

LinkedIn

LinkedIn es una red profesional en línea de 250 millones de usuarios en 200 países a principios de 2014 [5]. LinkedIn ofrece varios servicios gratuitos y basados en suscripción, como páginas de información de la empresa, ofertas de empleo, búsquedas de talentos, gráficos sociales de los contactos, fuentes de noticias personalizadas y acceso a grupos de discusión, incluido un grupo de usuarios de Hadoop. LinkedIn utiliza Hadoop para los siguientes propósitos [6]:

- Procesar registros de transacciones de la base de datos de producción diaria
- Examinar las actividades de los usuarios, como las vistas y los clics.
- Devolver los datos extraídos a los sistemas de producción
- Reestructurar los datos para agregarlos a una base de datos analítica
- Desarrollar y probar modelos analíticos

Yahoo!

A partir de 2012, Yahoo! tiene una de las implementaciones de Hadoop más grandes anunciadas públicamente en 42 000 nodos en varios clústeres que utilizan 350 petabytes de almacenamiento sin procesar [7]. Las aplicaciones Hadoop de Yahoo! Incluyen lo siguiente [8]:

- Creación y mantenimiento de índices de búsqueda
- Optimización del contenido de la página web

- Optimización de la ubicación de anuncios web
- Filtros de spam
- Análisis ad-hoc y desarrollo de modelos analíticos

Antes de implementar Hadoop, se tardaban 26 días en procesar los datos de registro de tres años. Con Hadoop, el tiempo de procesamiento se redujo a 20 minutos.

10.1.2 MapReduce

Como se mencionó anteriormente, el paradigma MapReduce proporciona los medios para dividir una tarea grande en tareas más pequeñas, ejecutar las tareas en paralelo y consolidar los resultados de las tareas individuales en el resultado final. Como su nombre lo indica, MapReduce consta de dos partes básicas: un paso de mapeo y un paso de reducción, que se detallan a continuación:

Mapa:

- Aplica una operación a un dato
- Proporciona una salida intermedia Reducir:
 - Consolida los resultados intermedios de los pasos del mapa
 - Proporciona el resultado final

Cada paso usa pares clave / valor, denotados como < clave, valor >, como entrada y salida. Es útil pensar en los pares clave / valor como un par ordenado simple. Sin embargo, los pares pueden adoptar formas bastante complejas. Por ejemplo, la clave podría ser un nombre de archivo y el valor podría ser el contenido completo del archivo.

La ilustración más simple de MapReduce es un ejemplo de recuento de palabras en el que la tarea consiste simplemente en contar el número de veces que aparece cada palabra en una colección de documentos. En la práctica, el objetivo de dicho ejercicio es establecer una lista de palabras y su frecuencia para fines de búsqueda o establecer la importancia relativa de cert

0-1 ilustra el

MapReduce proce

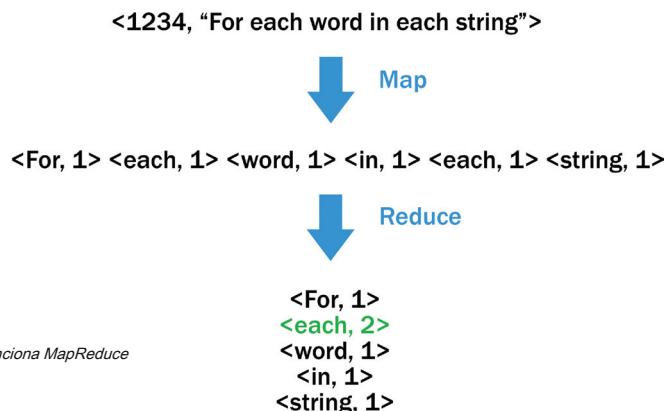


FIGURE 10-1 Ejemplo de cómo funciona MapReduce

En este ejemplo, el paso del mapa analiza la cadena de texto proporcionada en palabras individuales y emite un conjunto de pares clave / valor de la forma < palabra, 1>. Para cada clave única, en este ejemplo, palabra —El paso de reducción suma el 1 valores y salidas el < palabra, contar> pares clave / valor. Porque la palabra cada apareció dos veces en la línea de texto dada, el paso de reducción proporciona un par clave / valor correspondiente de < cada uno, 2>.

Cabe señalar que, en este ejemplo, la clave original, 1234, se ignora en el procesamiento. En una aplicación típica de recuento de palabras, el paso del mapa se puede aplicar a millones de líneas de texto y el paso de reducción resumirá los pares clave / valor generados por todos los pasos del mapa.

Ampliando el ejemplo del recuento de palabras, el resultado final de un proceso MapReduce aplicado a un conjunto de documentos podría tener el llave como un par ordenado y el valor como una tupla ordenada de longitud $2n$. A continuación, se muestra una posible representación de dicho par clave / valor:

<(nombre de archivo, fecha y hora), (palabra1,5, palabra2,7, ..., palabraN, 6)>

En esta construcción, la clave es el par ordenado nombre del archivo y fecha y hora. El valor consta de los n pares de palabras y sus recuentos individuales en el archivo correspondiente.

Por supuesto, un problema de recuento de palabras podría abordarse de muchas formas distintas a MapReduce. Sin embargo, MapReduce tiene la ventaja de poder distribuir la carga de trabajo en un grupo de computadoras y ejecutar las tareas en paralelo. En un recuento de palabras, los documentos, o incluso partes de los documentos, podrían procesarse simultáneamente durante el paso del mapa. Una característica clave de MapReduce es que el procesamiento de una porción de la entrada se puede realizar independientemente del procesamiento de las otras entradas. Por lo tanto, la carga de trabajo se puede distribuir fácilmente en un grupo de máquinas.

El contralmirante GraceHopper (1906-1992) de la Marina de los EE. UU., Pionero en el campo de las computadoras, una de las mejores explicaciones de la necesidad de utilizar un grupo de ordenadores. Comentó que durante la época preindustrial, los bueyes eran usados para jalar peso, pero cuando un buey no podía mover un tronco, la gente no intentaba criar un buey más grande; agregaron más bueyes. Su punto fue que a medida que crecían los problemas computacionales, en lugar de construir una computadora más grande, más potente y más cara, una mejor alternativa es construir un sistema de computadoras para compartir la carga de trabajo. Por lo tanto, en el contexto de MapReduce, una gran tarea de procesamiento se distribuiría entre muchas computadoras.

Aunque el concepto de MapReduce ha existido durante décadas, Google lideró el resurgimiento de su interés y adopción a partir de 2004 con el trabajo publicado por Dean y Ghemawat [9]. Este documento describe el enfoque de Google para rastrear la web y construir el motor de búsqueda de Google. Como se describe en el artículo, MapReduce se ha utilizado en lenguajes de programación funcional como Lisp, que obtuvo su nombre por ser capaz de procesar listas (procesamiento de listas).

En 2007, un caso de uso de MapReduce muy publicitado fue la conversión de 11 millones *New York Times* artículos de periódicos de 1851 a 1980 en archivos PDF. La intención era hacer que los archivos PDF estuvieran disponibles abiertamente para los usuarios en Internet. Después de desarrollar y probar el código de MapReduce en una máquina local, se generaron 11 millones de archivos PDF en un clúster de 100 nodos en aproximadamente 24 horas [10].

Lo que permitió que el desarrollo del código MapReduce y su ejecución procedieran fácilmente fue que el paradigma MapReduce ya se había implementado en Apache Hadoop.

10.1.3 Apache Hadoop

Aunque MapReduce es un paradigma simple de entender, no es tan fácil de implementar, especialmente en un sistema distribuido. La ejecución de un trabajo de MapReduce (el código de MapReduce se ejecuta con algunos datos específicos) requiere la gestión y coordinación de varias actividades:

- Los trabajos de MapReduce deben programarse en función de la carga de trabajo del sistema.
- Los trabajos deben ser monitoreados y administrados para garantizar que los errores encontrados se manejen adecuadamente para que el trabajo continúe ejecutándose si el sistema falla parcialmente.
- Los datos de entrada deben distribuirse por todo el clúster.
- El procesamiento por pasos de mapa de la entrada debe realizarse en todo el sistema distribuido, preferiblemente en las mismas máquinas donde residen los datos.
- Los resultados intermedios de los numerosos pasos del mapa deben recopilarse y proporcionarse a las máquinas adecuadas para reducir la ejecución de los pasos.
- La salida final debe estar disponible para que la utilice otro usuario, otra aplicación o quizás otro trabajo de MapReduce.

Afortunadamente, Apache Hadoop maneja estas actividades y más. Además, muchas de estas actividades son transparentes para el desarrollador / usuario. El siguiente material examina la implementación de MapReduce en Hadoop, un proyecto de código abierto administrado y licenciado por Apache Software Foundation [11].

Los orígenes de Hadoop comenzaron como un motor de búsqueda llamado Nutch, desarrollado por Doug Cutting y Mike Cafarella. Según dos artículos de Google [9] [12], en 2004 se agregaron versiones de MapReduce y el sistema de archivos de Google a Nutch. En 2006, Yahoo! contrató a Cutting, quien ayudó a desarrollar Hadoop basado en el código en Nutch [13]. El nombre "Hadoop" proviene del nombre del elefante de juguete de peluche del niño de Cutting que también inspiró el reconocido símbolo del proyecto Hadoop.

A continuación, se presenta una descripción general de cómo se almacenan los datos en un entorno Hadoop.

Hadoop Distributed File System (HDFS)

Basado en el sistema de archivos de Google [12], el sistema de archivos distribuido de Hadoop (HDFS) es un sistema de archivos que proporciona la capacidad de distribuir datos en un clúster para aprovechar el procesamiento paralelo de MapReduce. HDFS no es una alternativa a los sistemas de archivos comunes, como ext3, ext4 y XFS. De hecho, HDFS depende del sistema de archivos de cada unidad de disco para administrar los datos que se almacenan en el medio de la unidad. El Wiki de Hadoop [14] proporciona más detalles sobre las opciones y consideraciones de configuración de disco.

Para un archivo determinado, HDFS divide el archivo, por ejemplo, en bloques de 64 MB y almacena los bloques en todo el clúster. Por tanto, si el tamaño de un archivo es de 300 MB, el archivo se almacena en cinco bloques: cuatro bloques de 64 MB y un bloque de 44 MB. Si el tamaño de un archivo es menor de 64 MB, al bloque se le asigna el tamaño del archivo.

Siempre que sea posible, HDFS intenta almacenar los bloques de un archivo en diferentes máquinas para que el paso del mapa pueda operar en cada bloque de un archivo en paralelo. Además, de forma predeterminada, HDFS crea tres copias de cada bloque en el clúster para proporcionar la redundancia necesaria en caso de falla. Si una máquina falla, HDFS replica una copia accesible de los bloques de datos relevantes a otra máquina disponible. HDFS también es compatible con los racks, lo que significa que distribuye los bloques en varios racks de equipos para evitar que una falla completa del rack provoque un evento de datos no disponibles. Además, las tres copias de cada bloque permiten a Hadoop cierta flexibilidad para determinar qué máquina usar para el paso del mapa en un bloque en particular. Por ejemplo,

una máquina inactiva o subutilizada que contiene un bloque de datos para ser procesado se puede programar para procesar ese bloque de datos.

Para administrar el acceso a los datos, HDFS utiliza tres demonios de Java (procesos en segundo plano): NameNode, DataNode y SecondaryNameNode. Funcionando en una sola máquina, el **NameNode** daemon determina y rastrea dónde se almacenan los distintos bloques de un archivo de datos. Los **DataNode** daemon gestionan los datos almacenados en cada máquina. Si una aplicación cliente desea acceder a un archivo en particular almacenado en HDFS, la aplicación se pone en contacto con el NameNode, y el NameNode proporciona a la aplicación las ubicaciones de los distintos bloques para ese archivo. A continuación, la aplicación se comunica con los DataNodes adecuados para acceder al archivo.

Cada DataNode genera periódicamente un informe sobre los bloques almacenados en el DataNode y envía el informe al NameNode. Si uno o más bloques no son accesibles en un DataNode, el NameNode asegura que una copia accesible de un bloque de datos inaccesible se replica en otra máquina. Por motivos de rendimiento, NameNode reside en la memoria de una máquina. Debido a que NameNode es fundamental para el funcionamiento de HDFS, cualquier indisponibilidad o corrupción de NameNode da como resultado un evento de indisponibilidad de datos en el clúster. Por lo tanto, NameNode se ve como un único punto de falla en el entorno Hadoop [15]. Para minimizar la posibilidad de una falla en NameNode y mejorar el rendimiento, NameNode generalmente se ejecuta en una máquina dedicada.

Un tercer demonio, el **SecondaryNameNode**, proporciona la capacidad de realizar algunas de las tareas del NameNode para reducir la carga en el NameNode. Tales tareas incluyen actualizar la imagen del sistema de archivos con el contenido de los registros del sistema de archivos. Es importante tener en cuenta que el NameNode secundario no es un NameNode de respaldo o redundante. En el caso de una interrupción de NameNode, el NameNode debe reiniciarse e inicializarse con el último archivo de imagen del sistema de archivos y el contenido de los registros de ediciones. Las últimas versiones de Hadoop proporcionan una función HDFS de alta disponibilidad (HA). Esta función permite el uso de dos NameNodes: uno en estado activo y el otro en estado de espera. Si un NameNode activo falla, el NameNode en espera se hace cargo. Cuando se utiliza la función HDFS HA, no es necesario un NameNode secundario [16].

La Figura 10-2 ilustra un clúster de Hadoop con diez máquinas y el almacenamiento de un archivo grande que requiere tres bloques de datos HDFS. Además, este archivo se almacena mediante triple replicación. Las máquinas que ejecutan NameNode y SecondaryNameNode se consideran **nodos maestros**. Dado que los DataNodes toman sus instrucciones de los nodos maestros, las máquinas que ejecutan los DataNodes se denominan **nodos trabajadores**.

Estructuración de un trabajo de MapReduce en Hadoop

Hadoop proporciona la capacidad de ejecutar trabajos de MapReduce como se describe, en un alto nivel, en la Sección 10.1.2. Esta sección ofrece detalles específicos sobre cómo se ejecuta un trabajo de MapReduce en Hadoop. Un programa MapReduce típico en Java consta de tres clases: el controlador, el asignador y el reductor.

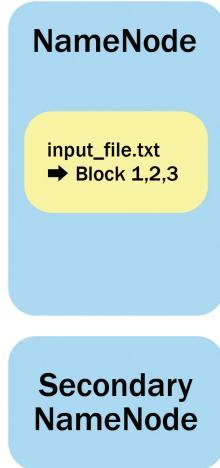
Los **controlador** proporciona detalles como la ubicación de los archivos de entrada, las disposiciones para agregar el archivo de entrada a la tarea de mapa, los nombres de las clases Java del asignador y reductor y la ubicación de la salida de la tarea de reducción. También se pueden especificar varias opciones de configuración de trabajos en el controlador. Por ejemplo, el número de reductores se puede especificar manualmente en el controlador. Estas opciones son útiles dependiendo de cómo se utilizará la salida del trabajo de MapReduce en el procesamiento posterior.

Los **mapeador** proporciona la lógica que se procesará en cada bloque de datos correspondiente a los archivos de entrada especificados en el código del controlador. Por ejemplo, en el ejemplo de MapReduce de recuento de palabras proporcionado anteriormente, una tarea de mapa se instancia en un nodo trabajador donde reside un bloque de datos. Cada tarea de mapa procesa un fragmento del texto, línea por línea, analiza una línea en palabras y emite < palabra, 1 > para cada palabra, independientemente de cuántas

veces palabra apelar
memoria (o caché)

nodo

Master Nodes



8 Worker Nodes across 2 Racks

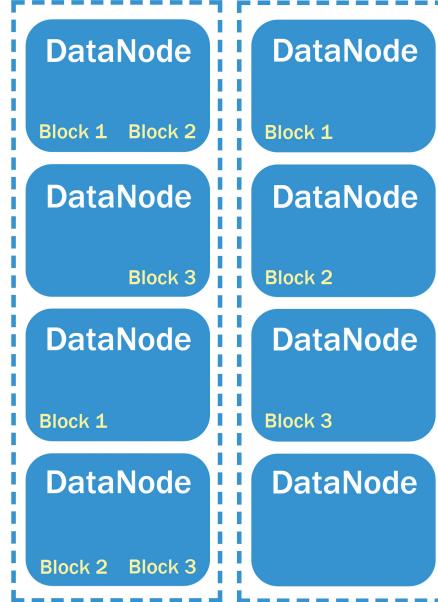


FIGURE 10-2 Un archivo almacenado en HDFS

A continuación, los pares clave / valor son procesados por el `shuf fil e y ordenar` funcionalidad basada en el número de reduceres a ejecutar. En este sencillo ejemplo, solo hay un reducer. Entonces, todos los datos intermedios se le pasan. A partir de las distintas salidas de la tarea de mapa, para cada clave única, se construyen matrices (listas en Java) de los valores asociados en los pares clave / valor. Además, Hadoop asegura que las claves se pasen a cada reductor en orden ordenado. En la Figura 10-3, < cada uno, (1,1)> es el primer par clave / valor procesado, seguido alfabéticamente por < Para, (1)> y el resto de los pares clave / valor hasta que el último par clave / valor se pase al reductor. El () denota una lista de valores que, en este caso, es solo una matriz de unos.

En general, cada reductor procesa los valores de cada clave y emite un par clave / valor según lo definido por la lógica de reducción. Luego, la salida se almacena en HDFS como cualquier otro archivo en, por ejemplo, bloques de 64 MB replicados tres veces en los nodos.

Consideraciones adicionales al estructurar un trabajo de MapReduce

La discusión anterior presentó los conceptos básicos de estructurar y ejecutar un trabajo MapReduce en un clúster de Hadoop. Varias características de Hadoop brindan funcionalidad adicional a un trabajo de MapReduce.

Primero un **combinador** es una opción útil para aplicar, cuando sea posible, entre la tarea de mapa y el `shuf fil e y sort`. Normalmente, el combinador aplica la misma lógica utilizada en el reductor, pero también aplica esta lógica en la salida de cada tarea de mapa. En el ejemplo del recuento de palabras, un combinador suma el número de apariciones de

cada palabra de am
conteo simple de palabras

ngle cadena en el

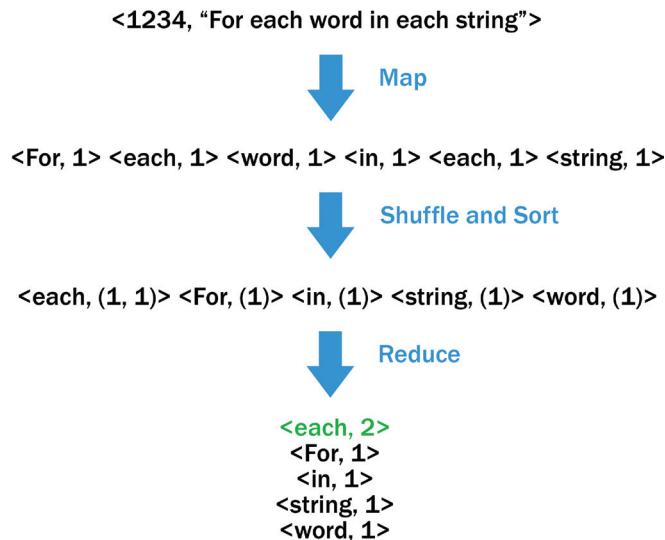


FIGURE 10-3 Shuffl e y s

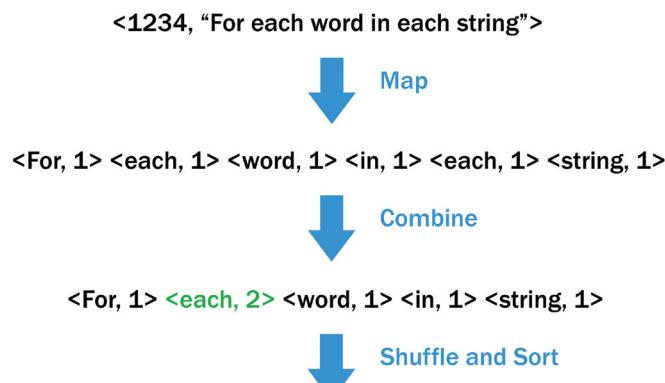


FIGURE 10-4 Usando un combinador

Por lo tanto, en un entorno de producción, en lugar de diez mil posibles < el, 1> pares clave / valor emitidos desde la tarea de mapa al Shuffl e y Sort, el combinador emite un < el, 10000> par clave / valor. El paso de reducción aún obtiene una lista de valores para cada palabra, pero en lugar de recibir una lista de hasta un millón de unidades lista (1,1,..., 1) para una clave, el paso de reducción obtiene una lista, como lista (10000,964 .. . , 8345), que puede ser tan largo como el número de tareas de mapa que se ejecutaron. El uso de un combinador minimiza la cantidad de salida de mapa intermedio que el reductor debe almacenar, transferir a través de la red y procesar.

Otra opción útil es la **particionador**. Determina los reductores que reciben llaves y la correspondiente lista o enviar cada palabra a otro reducir

itoner puede consonante

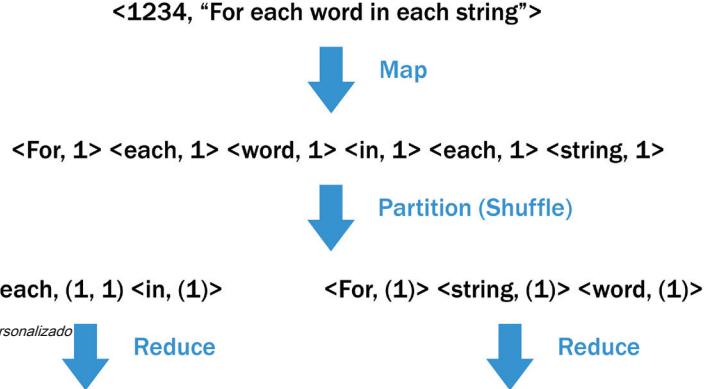


FIGURE 10-5 Usando un particionador personalizado

Como ejemplo más práctico, un usuario podría usar un particionador para separar la salida en archivos separados para cada año calendario para un análisis posterior. Además, se podría utilizar un particionador para garantizar que la carga de trabajo se distribuya uniformemente entre los reductores. Por ejemplo, si se sabe que algunas claves están asociadas con una gran mayoría de los datos, puede ser útil asegurarse de que estas claves vayan a reductores separados para lograr un mejor rendimiento general. De lo contrario, a un reductor se le podría asignar la mayoría de los datos y el trabajo de MapReduce no se completaría hasta que se complete la tarea de reducción de larga duración.

Desarrollar y ejecutar un programa HadoopMapReduce

Un enfoque común para desarrollar un programa HadoopMapReduce es escribir código Java utilizando una herramienta de entorno de desarrollo interactivo (IDE) como Eclipse [17]. En comparación con un editor de texto sin formato o una interfaz de línea de comandos (CLI), las herramientas IDE ofrecen una mejor experiencia para escribir, compilar, probar y depurar código. Un programa MapReduce típico consta de tres archivos Java: uno para el código del controlador, el código del mapa y el código de reducción. Se pueden escribir archivos Java adicionales para el combinador o el particionador personalizado, si corresponde. El código Java se compila y almacena como un archivo Java Archive (JAR). Este archivo JAR se ejecuta luego contra los archivos de entrada HDFS especificados.

Más allá de aprender la mecánica de enviar un trabajo de MapReduce, tres desafíos clave para un nuevo desarrollador de Hadoop son definir la lógica del código para usar el paradigma de MapReduce; aprender las clases, los métodos y las interfaces de Apache Hadoop Java; e implementar la funcionalidad de controlador, mapa y reducción en Java. Algo de experiencia previa con Java facilita que un nuevo desarrollador de Hadoop se concentre en aprender Hadoop y escribir el trabajo MapReduce.

Para los usuarios que prefieren utilizar un lenguaje de programación que no sea Java, existen otras opciones. Una opción es utilizar el **API de transmisión de Hadoop**, que permite al usuario escribir y ejecutar trabajos de Hadoop sin conocimiento directo de Java [18]. Sin embargo, es necesario tener conocimientos de algún otro lenguaje de programación, como Python, C o Ruby. Apache Hadoop proporciona Hadoop-streaming.jar para archivar eso

acepta las rutas HDFS para los archivos de entrada / salida y las rutas para los archivos que implementan el mapa y reducen la funcionalidad.

A continuación, se muestran algunas consideraciones importantes al preparar y ejecutar un trabajo de transmisión de Hadoop:

- Aunque la salida de `shuf` e `sort` se proporciona al reductor en orden de clasificación clave, el reductor no recibe los valores correspondientes como una lista; más bien, recibe pares clave / valor individuales. El código de reducción tiene un monitor de cambios en el valor de la clave y maneja apropiadamente la nueva clave.
- El mapa y el código de reducción ya deben estar en formato ejecutable, o el intérprete necesario ya debe estar instalado en cada nodo de trabajo.
- El mapa y el código de reducción ya deben residir en cada nodo de trabajador, o se debe proporcionar la ubicación del código cuando se envía el trabajo. En el último caso, el código se copia en cada nodo trabajador.
- Algunas funciones, como un particionador, aún deben escribirse en Java.
- Las entradas y salidas se manejan a través de `stdin` y `stdout`. `Stderr` también está disponible para rastrear el estado de las tareas, implementar la funcionalidad del contador e informar problemas de ejecución en la pantalla [18].
- Es posible que la API de transmisión no funcione bien como una funcionalidad similar escrita en Java.

Una segunda alternativa es utilizar **Tubos Hadoop**, un mecanismo que utiliza código C ++ compilado para el mapa y funcionalidad reducida. Una ventaja de usar C ++ son las extensas bibliotecas numéricas disponibles para incluir en el código [19].

Para trabajar directamente con datos en HDFS, una opción es utilizar la API de C (`libhdfs`) o la API de Java proporcionada con Apache Hadoop. Estas API permiten leer y escribir en archivos de datos HDFS fuera del paradigma típico de MapReduce [20]. Este enfoque puede ser útil cuando se intenta depurar un trabajo de MapReduce examinando los datos de entrada o cuando el objetivo es transformar los datos de HDFS antes de ejecutar un trabajo de MapReduce.

Otro negociador de recursos (YARN)

Apache Hadoop continúa experimentando un mayor desarrollo y actualizaciones frecuentes. Un cambio importante fue separar la funcionalidad MapReduce de la funcionalidad que administra la ejecución de los trabajos y las responsabilidades asociadas en un entorno distribuido. Esta reescritura a veces se llamaMapReduce 2.0, u otro negociador de recursos (YARN). YARN separa la administración de recursos del clúster de la programación y supervisión de los trabajos que se ejecutan en el clúster. La implementación de YARN hace posible que otros paradigmas distintos de MapReduce se utilicen en entornos Hadoop. Por ejemplo, un modelo Bulk Synchronous Parallel (BSP) [21] puede ser más apropiado para el procesamiento de gráficos que MapReduce [22]. Apache Hama, que implementa el modelo BSP, es una de varias aplicaciones que se están modificando para utilizar el poder de YARN [23].

YARN reemplaza la funcionalidad proporcionada anteriormente por los demonios JobTracker y TaskTracker. En versiones anteriores de Hadoop, un trabajo MapReduce se envía al demonio JobTracker. JobTracker se comunica con NameNode para determinar qué nodos trabajadores almacenan los bloques de datos necesarios para el trabajo MapReduce. JobTracker luego asigna un mapa individual y reduce las tareas al TaskTracker que se ejecuta en los nodos de trabajo. Para optimizar el rendimiento, cada tarea se asigna preferiblemente a un nodo trabajador que está almacenando un bloque de datos de entrada. TaskTracker se comunica periódicamente con JobTracker sobre el estado de sus tareas en ejecución. Si una tarea parece haber fallado, JobTracker puede asignar la tarea a un TaskTracker diferente.

10.2 El ecosistema de Hadoop

Hasta ahora, este capítulo ha proporcionado una descripción general de Apache Hadoop en relación con su implementación de HDFS y el paradigma MapReduce. La popularidad de Hadoop ha generado herramientas patentadas y de código abierto para hacer que Apache Hadoop sea más fácil de usar y proporcionar funciones y características adicionales. Esta parte del capítulo examina los siguientes proyectos de Apache relacionados con Hadoop:

- **Cerdo:** Proporciona un lenguaje de programación de flujo de datos de alto nivel
- **Colmena:** Proporciona acceso similar a SQL
- **Cuidador de elefantes:** Proporciona herramientas analíticas
- **HBase:** Proporciona lecturas y escrituras en tiempo real

Al enmascarar los detalles necesarios para desarrollar un programa MapReduce, Pig y Hive permiten al desarrollador escribir código de alto nivel que luego se traduce en uno o más programas MapReduce. Debido a que MapReduce está diseñado para el procesamiento por lotes, Pig andHive también está diseñado para casos de uso de procesamiento por lotes.

Una vez que Hadoop procesa un conjunto de datos, Mahout proporciona varias herramientas que pueden analizar los datos en un entorno Hadoop. Por ejemplo, un análisis de agrupamiento de k-medias, como se describe en el Capítulo 4, se puede realizar usando Mahout.

Al diferenciarse del procesamiento por lotes de cerdos y colmenas, HBase brinda la capacidad de realizar lecturas y escrituras en tiempo real de datos almacenados en un entorno Hadoop. Este acceso en tiempo real se logra en parte almacenando datos en la memoria y también en HDFS. Además, HBase no depende de MapReduce para acceder a los datos de HBase. Debido a que el diseño y el funcionamiento de HBase son significativamente diferentes de las bases de datos relacionales y las otras herramientas de Hadoop examinadas, se presentará una descripción detallada de HBase.

10.2.1 Cerdo

Apache Pig consta de un lenguaje de flujo de datos, Pig Latin, y un entorno para ejecutar el código Pig. El principal beneficio de usar Pig es utilizar el poder de MapReduce en un sistema distribuido, mientras se simplifican las tareas de desarrollo y ejecución de un trabajo de MapReduce. En la mayoría de los casos, es transparente para el usuario que un trabajo de MapReduce se está ejecutando en segundo plano cuando se ejecutan los comandos de Pig. Esta capa de abstracción sobre Hadoop simplifica el desarrollo de código contra datos en HDFS y hace que MapReduce sea más accesible para una audiencia más amplia.

Al igual que Hadoop, el origen de Pig comenzó en Yahoo! en 2006. Pig se transfirió a Apache Software Foundation en 2007 y tuvo su primer lanzamiento como un subproyecto de Apache Hadoop en 2008. A medida que Pig evoluciona con el tiempo, persisten tres características principales: facilidad de programación, optimización de código detrás de escena y extensibilidad de capacidades [24].

Con Apache Hadoop y Pig ya instalados, los conceptos básicos del uso de Pig incluyen ingresar al entorno de ejecución de Pig escribiendo **cerdo** en el símbolo del sistema y luego ingresando una secuencia de líneas de instrucción Pig en el gruñido rápido.

Aquí se muestra un ejemplo de comandos específicos de Pig:

```
$ cerdo
gruñido> registros = CARGAR '/user/customer.txt' AS
          (id_cliente: INT, primer_nombre: CHARARRAY,
```

```

last_name: CHARARRAY,
dirección_de_correo_electrónico: CHARARRAY);

gruñido> filtrados_records = FILTRAR registros
          BY email_address coincide con '. * @ lsp.com';

gruñido> ALMACENAR registros_filtrados EN '/ user / isp_customers';
gruñido> salir

```

PS

En la primera gruñido indicador, un archivo de texto es designado por la variable Pig registros con cuatro campos definidos: cust_id, primer nombre, apellido, y dirección de correo electrónico. A continuación, la variable filter_records se le asignan aquellos registros donde el dirección de correo electrónico termina con @ lsp.com para extraer los clientes cuya dirección de correo electrónico es de un proveedor de servicios de Internet (ISP) en particular. Utilizando la

TIENDA comando, los registros filtrados se escriben en una carpeta HDFS, isp_customers. Finalmente, para salir del entorno interactivo de Pig, ejecute el DEJAR mando. Alternativamente, estos comandos individuales de Pig podrían escribirse en el archivo filter_script.pig y envíe el tema en el símbolo del sistema de la siguiente manera:

```
$ cerdo filter_script.pig
```

Estas instrucciones de Pig se traducen, entre bastidores, en uno o más trabajos de MapReduce. Por lo tanto, Pig simplifica la codificación de un trabajo MapReduce y permite al usuario desarrollar, probar y depurar rápidamente el código Pig. En este ejemplo en particular, el trabajo MapReduce se iniciaría después de la TIENDA se procesa el comando. Antes de la TIENDA comando, Pig había comenzado a construir un plan de ejecución, pero aún no había iniciado el procesamiento de MapReduce.

Pig proporciona la ejecución de varias manipulaciones de datos comunes, como combinaciones internas y externas entre dos o más archivos (tablas), como se esperaría en una base de datos relacional típica. Escribir estas uniones explícitamente en MapReduce usando Hadoop sería bastante complicado y complejo. El cerdo también proporciona un AGRUPAR POR funcionalidad que es similar a la Agrupar por funcionalidad ofrecida en SQL. El capítulo 11 tiene más detalles sobre el uso Agrupar por y otras sentencias SQL.

Una característica adicional de Pig es que proporciona muchas funciones integradas que se utilizan fácilmente en el código de Pig. La tabla 10-1 incluye varias funciones útiles por categoría.

T PODER 10-1 Funciones de cerdo integradas

AVG	BinStorage ()	abdominales	ÍNDICE DE	AddDuration
CONCAT	JsonLoader	HACER_TECNO	ULTIMO_	Tiempo actual
			ÍNDICE DE	
CONTAR	JsonStorage	COS, ACOS	LCFORST	Días entre
COUNT_STAR	PigDump	Exp	INFERIOR	GetDay
DIFF	Almacenamiento de cerdos	PISO	REGEX_	GetHour
			EXTRAER	
				(continúa)

T PODER 10-1 Funciones de cerdo integradas (continuación)

Esta vacío	TextLoader	LOG, LOG10	REEMPLAZAR	GetMinute
MAX	HBaseStorage	ALEATORIO	STRSPLIT	GetMonth
MIN		REDONDO	SUBSTRING	GetWeek
TALLA		SIN, ASIN	PODAR	GetWeekYear
SUMA		SQRT	UCFIRST	GetYear
TOKENIZAR		TAN, ATAN	SUPERIOR	Minutos entre
				RestarDuración
				Hasta la fecha

Otras funciones y los detalles de estas funciones integradas se pueden encontrar en la pig.apache.org sitio web [25].

En términos de extensibilidad, Pig permite la ejecución de funciones definidas por el usuario (UDF) en su entorno. Por lo tanto, algunas operaciones complejas pueden codificarse en el idioma que elija el usuario y ejecutarse en el entorno Pig. Los usuarios pueden compartir sus UDF en un repositorio llamado Piggybank alojado en el sitio de Apache [26]. Con el tiempo, las UDF más útiles pueden incluirse como funciones integradas en Pig.

10.2.2 Colmena

De forma similar a Pig, Apache Hive permite a los usuarios procesar datos sin escribir explícitamente código MapReduce. Una diferencia clave de Pig es que el lenguaje Hive, HiveQL (Hive Query Language), se parece al StructuredQuery Language (SQL) en lugar de a un lenguaje de scripting.

Una estructura de tabla de Hive consta de filas y columnas. Las filas normalmente corresponden a algún detalle de registro, transacción o entidad particular (por ejemplo, cliente). Los valores de las columnas correspondientes representan los distintos atributos o características de cada fila. Hadoop y su ecosistema se utilizan para aplicar alguna estructura a los datos no estructurados. Por lo tanto, si una estructura de tabla es una forma adecuada de ver los datos reestructurados, Hive puede ser una buena herramienta para usar.

Además, un usuario puede considerar usar Hive si el usuario tiene experiencia con SQL y los datos ya están en HDFS. Otra consideración al usar Hive puede ser cómo se actualizarán o agregarán los datos a las tablas de Hive. Si los datos simplemente se agregarán a una tabla periódicamente, Hive funciona bien, pero si es necesario actualizar los datos en su lugar, puede ser beneficioso considerar otra herramienta, como HBase, que se discutirá en la siguiente sección.

Aunque el rendimiento de Hive puede ser mejor en ciertas aplicaciones que una base de datos SQL convencional, Hive no está diseñado para consultas en tiempo real. Una consulta de Hive se traduce primero en un trabajo de MapReduce, que luego se envía al clúster de Hadoop. Por lo tanto, la ejecución de la consulta debe competir por los recursos con cualquier otro trabajo enviado. Al igual que Pig, Hive está destinado al procesamiento por lotes. Nuevamente, HBase puede ser una mejor opción para las necesidades de consultas en tiempo real.

Para resumir la discusión anterior, considere usar Hive cuando existan las siguientes condiciones:

- Los datos encajan fácilmente en una estructura de tabla.
- Los datos ya están en HDFS. (Nota: los archivos que no son HDFS se pueden cargar en una tabla de Hive).
- Los desarrolladores se sienten cómodos con la programación y las consultas SQL.
- Existe el deseo de dividir los conjuntos de datos en función del tiempo. (Por ejemplo, las actualizaciones diarias se agregan a la tabla de Hive).
- El procesamiento por lotes es aceptable.

El resto de la discusión de Hive cubre algunos conceptos básicos de HiveQL. Desde el símbolo del sistema, un usuario ingresa al entorno interactivo de Hive simplemente ingresando **colmena**:

```
$ colmena
colmena>
```

Desde este entorno, un usuario puede definir nuevas tablas, consultarlas o resumir su contenido. Para ilustrar cómo usar HiveQL, el siguiente ejemplo de fi ne una nueva tabla de Hive para contener datos de clientes, cargar datos HDFS existentes en la tabla de Hive y consultar la tabla.

El primer paso es crear una tabla llamada cliente para almacenar los datos del cliente. Debido a que la tabla se completará a partir de un archivo HDFS delimitado por una pestaña existente ('\t'), este formato se especifica en la consulta de creación de la tabla.

```
colmena> crear cliente de mesa (
```

```
    cust_id bigint,
    cadena de first_name,
    cadena last_name,
    cadena de dirección de correo electrónico)
    formato de fila delimitado
    campos terminados por '\t';
```

Se ejecuta la siguiente consulta de HiveQL para contar el número de registros en la tabla recién creada, cliente. Debido a que la tabla está actualmente vacía, la consulta devuelve un resultado de cero, la última línea del resultado proporcionado. La consulta se convierte y se ejecuta como un trabajo de MapReduce, lo que da como resultado la ejecución de una tarea de mapa y una tarea de reducción.

```
colmena> seleccionar recuento (*) del cliente;
```

```
Total de trabajos de MapReduce = 1
```

Lanzamiento del trabajo 1 de 1
Número de tareas de reducción determinadas en tiempo de compilación: 1 Trabajo de inicio
= job_1394125045435_0001, URL de seguimiento =

```
http://pivhdsne: 8088 / proxy / application_1394125045435_0001 / Kill Command = / usr / lib
/gphd / hadoop / bin / hadoop job
```

```
- matar trabajo_1394125045435_0001
Información del trabajo de Hadoop para la etapa 1: número de mapeadores: 1,
número de reductores: 1
2014-03-06 12: 30: 23,542 Mapa de la etapa 1 = 0%, 2014-03-06 12: reducir = 0%
30: 36,586 Mapa de la etapa 1 = 100% , reducir = 0%,
CPU acumulada 1,71 segundos
2014-03-06 12: 30: 48,500 Mapa de la etapa 1 = 100% , reducir = 100%,
CPU acumulada 3,76 segundos
```

```
MapReduce Tiempo total acumulado de CPU: 3 segundos 760 mseg Trabajo finalizado =
job_1394125045435_0001
Trabajos de MapReduce lanzados:
Trabajo 0: Mapa: 1 Reducir: 1          CPU acumulada: 3,76 segundos      Lectura HDFS: 242
Escritura HDFS: 2 ÉXITO
Tiempo total de CPU de MapReduce invertido: 3 segundos 760 mseg OK
```

0

Al consultar tablas grandes, Hive supera y escala mejor que la mayoría de las consultas de bases de datos convencionales. Como se indicó anteriormente, Hive traduce las consultas de HiveQL en trabajos de MapReduce que procesan partes de grandes conjuntos de datos en paralelo.

Para cargar la tabla de clientes con el contenido del archivo HDFS, customer.txt, solo es necesario proporcionar la ruta del directorio HDFS al archivo.

colmena> cargar datos en la ruta '/user/customer.txt' en la tabla cliente;

La siguiente consulta muestra tres filas del cliente mesa.

```
colmena> seleccionar * del límite de clientes 3;
34567678      María      Jones      mary.jones@isp.com
897572388     Harry Schmidt      harry.schmidt@isp.com
89976576      Tom        Herrero      thomas.smith@otro_isp.com
```

A menudo es necesario unir una o más tablas de Hive basadas en una o más columnas. El siguiente ejemplo proporciona el mecanismo para unirse al cliente mesa con otra mesa, pedidos, que almacena los detalles sobre los pedidos del cliente. En lugar de colocar todos los datos del cliente en la tabla de pedidos, solo los correspondientes cust_id aparece en el pedidos mesa.

```
colmena> seleccione o.order_number, o.order_date, c. *
de pedidos o cliente de unión interna c en o.cust_id =
c.cust_id
donde c.email_address = 'mary.jones@isp.com';
```

Tareas totales de MapReduce = 1

Tarea de inicio 1 de 1

Número de tareas de reducción no especificadas. Estimado a partir del tamaño de los datos de entrada: 1 Trabajo inicial =
job_1394125045435_0002, URL de seguimiento =

```
http://pivhdsne: 8088 / proxy / application_1394125045435_0002 / Kill Command = / usr / lib
/gphd / hadoop / bin / hadoop job
```

- matar trabajo_1394125045435_0002

Información del trabajo de Hadoop para la Etapa 1: número de mapeadores: 2;

número de reductores: 1

```
2014-03-06 13: 26: 20,277 Mapa de la etapa 1 = 0%, 2014-03-06      reducir = 0%
13: 26: 42,568 Mapa de la etapa 1 = 50%,                                reducir = 0%,
```

CPU acumulada 4,23 segundos

```
2014-03-06 13: 26: 43,637 Mapa de la etapa 1 = 100%, , reducir = 0%,
CPU acumulada 4,79 segundos
```

```
2014-03-06 13: 26: 52,658 Mapa de la etapa 1 = 100%, , reducir = 100%,
CPU acumulada 7,07 segundos
```

MapReduce Tiempo total acumulado de CPU: 7 segundos onds 70 mseg

```
Trabajo finalizado = job_1394125045435_0002 Trabajos de
MapReduce iniciados:
Trabajo 0: Mapa: 2 Reducir: 1          CPU acumulada: 7,07 segundos      Lectura HDFS: 602
Escritura HDFS: 140 ÉXITO
Tiempo total de CPU de MapReduce invertido: 7 segundos 70 mseg OK

X234825811 201  3-11-15 17:08:43 34567678 Mary Jones mary.jones@isp.com 3-11-04 12:53:19
X234823904 201  34567678 Mary Jones mary.jones@isp.com
```

El uso de combinaciones y SQL en general se tratará en el Capítulo 11. Para salir del entorno interactivo de Hive, utilice dejar.

```
colmena> salir;
PS
```

Una alternativa a la ejecución en el entorno interactivo es recopilar las declaraciones de HiveQL en un script (por ejemplo, my_script.sql) y luego ejecute el archivo de la siguiente manera:

```
$ colmena -f my_script.sql
```

Esta introducción a Hive proporcionó algunos de los comandos y declaraciones básicos de HiveQL. Se anima al lector a investigar y utilizar, cuando sea apropiado, otras funciones de Hive, como tablas externas, explicar planes, particiones y INSERTAR EN comando para agregar datos al contenido existente de una tabla de Hive.

A continuación, se muestran algunos casos de uso de Hive:

- **Análisis exploratorio o ad-hoc de datos HDFS:** Los datos se pueden consultar, transformar y exportar a herramientas analíticas, como R.
- **Extrae o alimenta datos a sistemas de informes, paneles o repositorios de datos como HBase:** Las consultas de Hive se pueden programar para proporcionar dichos feeds periódicos.
- **Combinando datos estructurados externos que ya residen en HDFS:** Hadoop es excelente para procesar datos no estructurados, pero a menudo hay datos estructurados que residen en un RDBMS, como Oracle o SQL Server, que deben unirse con los datos que residen en HDFS. Los datos de un RDBMS se pueden agregar periódicamente a las tablas de Hive para consultar los datos existentes en HDFS.

10.2.3 HBase

A diferencia de Pig y Hive, que están destinados a aplicaciones por lotes, Apache HBase es capaz de proporcionar acceso de lectura y escritura en tiempo real a conjuntos de datos con miles de millones de filas y millones de columnas. Para ilustrar las diferencias entre HBase y una base de datos relacional, esta sección presenta detalles considerables sobre la implementación y el uso de HBase.

El diseño de HBase se basa en el artículo de 2006 de Google sobre Bigtable. Este documento describió a Bigtable como un "sistema de almacenamiento distribuido para administrar datos estructurados". Google utilizó Bigtable para almacenar datos específicos de productos de Google para sitios como Google Earth, que proporciona imágenes satelitales del mundo. Bigtable también se utilizó para almacenar resultados de rastreadores web, datos para la optimización de búsqueda personalizada y datos del flujo de clics del sitio web. Bigtable se creó sobre el sistema de archivos de Google. MapReduce también se utilizó para procesar

datos dentro o fuera de un Bigtable. Por ejemplo, los datos del flujo de clics sin procesar se almacenaron en una Bigtable. Periódicamente, se ejecutaba un trabajo MapReduce programado que procesaba y resumía los datos de flujo de clics recién agregados y adjuntaba los resultados a una segunda Bigtable [27].

El desarrollo de HBase comenzó en 2006. HBase se incluyó como parte de una distribución de Hadoop a finales de 2007. En mayo de 2010, HBase se convirtió en un proyecto de nivel superior de Apache. Más tarde, en 2010, Facebook comenzó a utilizar HBase para su infraestructura de mensajería de usuario, que acomodaba a 350 millones de usuarios que enviaban 15 mil millones de mensajes al mes [28].

Arquitectura y modelo de datos de HBase

HBase es un almacén de datos que está destinado a distribuirse en un grupo de nodos. Al igual que Hadoop y muchos de sus proyectos Apache relacionados, HBase se basa en HDFS y logra sus velocidades de acceso en tiempo real al compartir la carga de trabajo en una gran cantidad de nodos en un clúster distribuido. Una tabla HBase consta de filas y columnas. Sin embargo, una tabla HBase también tiene una tercera dimensión, versión, para mantener los diferentes valores de una intersección de filas y columnas a lo largo del tiempo.

Para ilustrar esta tercera dimensión, un ejemplo simple sería que para cualquier cliente en línea dado, se podrían almacenar varias direcciones de envío. Por lo tanto, la fila estaría indicada por un número de cliente. Una columna proporcionaría la dirección de envío. El valor de la dirección de envío se agregaría en la intersección del número de cliente y la columna de la dirección de envío, junto con una marca de tiempo correspondiente a la última vez que el cliente usó esta dirección de envío.

Durante un custo y mostrar la c seleccione el apropiado	víspela gallina t.
---	--------------------------

Checkout (Step 2 of 4)

Choose a shipping address:

		Last Used
<input type="checkbox"/>	1600 Pennsylvania Avenue NW Washington DC, 20500 USA	Edit Delete 15-Apr-2014
<input type="checkbox"/>	London SW1A 1AA, United Kingdom	Edit Delete 15-Mar-2014
<input type="checkbox"/>	आगरा, उत्तर प्रदेश 282001, India	Edit Delete 14-Feb-2014

Add a new address

FIGURE 10-6 Elegir una dirección de envío al finalizar la compra

Back

Continue

Por supuesto, además de la dirección de envío del cliente, se debe almacenar otra información del cliente, como la dirección de facturación, preferencias, créditos / débitos de facturación y beneficios para el cliente (por ejemplo, envío gratuito). Para este tipo de aplicación, se requiere acceso en tiempo real. Por lo tanto, el uso del procesamiento por lotes de Pig, Hive o Hadoop'sMapReduce no es un enfoque de implementación razonable. La siguiente discusión examina cómo HBase almacena los datos y proporciona acceso de lectura y escritura en tiempo real.

Como se mencionó, HBase está construido sobre HDFS. HBase usa una estructura clave / valor para almacenar el contenido de una tabla HBase. Cada valor son los datos que se almacenarán en la intersección de la fila, la columna y la versión. Cada clave consta de los siguientes elementos [29]:

- Longitud de la fila
- Fila (a veces llamada clave de fila)
- Longitud de la familia de columnas
- Familia de columnas
- Calificador de columna
- Versión
- Tipo de clave

los *fila* se utiliza como atributo principal para acceder al contenido de una tabla HBase. La fila es la base de cómo se distribuyen los datos en el clúster y permite una consulta de una tabla HBase para recuperar rápidamente los elementos deseados. Por lo tanto, la estructura o disposición de la fila debe diseñarse específicamente en función de cómo se accederá a los datos. En este sentido, una tabla HBase se construye expresamente y no está pensada para consultas y análisis ad-hoc generales. En otras palabras, es importante saber cómo se utilizará la tabla HBase. Esta comprensión del uso de la mesa ayuda a definir de manera óptima la construcción de la fila y la mesa.

Por ejemplo, si una tabla HBase va a almacenar el contenido de los correos electrónicos, la fila se puede construir como la concatenación de una dirección de correo electrónico y la fecha de envío. Debido a que la tabla HBase se almacenará en función de la fila, la recuperación de los mensajes de correo electrónico por una dirección de correo electrónico determinada será bastante eficiente, pero la recuperación de todos los mensajes de correo electrónico en un rango de fechas determinado llevará mucho más tiempo. La discusión posterior sobre regiones proporciona más detalles sobre cómo se almacenan los datos en HBase.

Una columna en una tabla HBase se designa mediante la combinación de *familia de columnas* y el *calificador de columna*. La familia de columnas proporciona una agrupación de alto nivel para los calificadores de columna. En el ejemplo anterior de dirección de envío, la fila podría contener el *número de orden*, y los detalles del pedido se pueden almacenar en la familia de columnas *pedidos*, utilizando los calificadores de columna como *dirección_envío*, *dirección_facturación*, *fecha_pedido*. En HBase, una columna se especifica como familia de columnas: calificador de columna. En el ejemplo, la columna *pedidos: dirección_envío* se refiere a la dirección de envío de un pedido.

UN *celda* es la intersección de una fila y una columna en una tabla. Los *versión*, a veces llamado el *marca de tiempo*, proporciona la capacidad de mantener diferentes valores para el contenido de una celda en HBase. Aunque el usuario puede definir un valor personalizado para la versión al escribir una entrada en la tabla, una implementación típica de HBase usa el valor predeterminado de HBase, la hora actual del sistema. En Java, esta marca de tiempo se obtiene con Sistema

.getCurrentTimeMillis (), el número de milisegundos desde el 1 de enero de 1970. Debido a que es probable que solo se requiera la versión más reciente de una celda, las celdas se almacenan en orden descendente de la versión. Si la aplicación requiere que las celdas se almacenen y recuperen en orden ascendente de su tiempo de creación, el enfoque es utilizar Long.MAX_VALUE -

`System.currentTimeMillis () en`

Java como número de versión. Long.MAX_VALUE corresponde al valor máximo que puede tener un entero largo en Java. En este caso, el almacenamiento y la clasificación siguen estando en orden descendente de los valores de la versión.

Tipo de clave se utiliza para identificar si una clave en particular corresponde a una operación de escritura en la tabla HBase o una operación de eliminación de la tabla. Técnicamente, una eliminación de una tabla HBase se logra con una escritura en la tabla. El tipo de clave indica el propósito de la escritura. Para eliminaciones, un marcador de lápida es

escrito en la tabla para indicar que todas las versiones de celda iguales o anteriores a la marca de tiempo especificada deben eliminarse para la fila y columna correspondientes familia: columna cali fi cador.

Una vez que se instala un entorno de HBase, el usuario puede ingresar al entorno de shell de HBase ingresando **shell hbase** en el símbolo del sistema. Una mesa HBase, mi mesa, luego se puede crear de la siguiente manera:

```
$ hbase shell
hbase> crear 'my_table', 'cf1', 'cf2',
          {SPLITS => ['250000', '500000', '750000']}
```

Dos familias de columnas, cf1 y cf2, se definen en la tabla. La opción especifica cómo se dividirá la tabla en función de la porción de fila de la clave. En este ejemplo, la tabla se divide en cuatro partes, llamadas **regiones**. Filas menores de 250000 se agregan a la primera región; filas de 250000 a menos de 500000 se agregan a la segunda región, y lo mismo ocurre con las divisiones restantes. Estas divisiones proporcionan el mecanismo principal para lograr el acceso de lectura y escritura en tiempo real. En este ejemplo, mi mesa se divide en cuatro regiones, cada una en su propio nodo trabajador en el clúster de Hadoop. Por lo tanto, a medida que aumenta el tamaño de la tabla o aumenta la carga de usuarios, se pueden agregar nodos de trabajo adicionales y divisiones de región para escalar el clúster de manera adecuada. Las lecturas y escrituras se basan en el contenido de la fila. HBase puede determinar rápidamente la región apropiada para dirigir un comando de lectura o escritura. Más adelante se discutirá más sobre las regiones y su implementación.

Solo las familias de columnas, no los calificadores de columnas, deben definirse durante la creación de la tabla HBase. Se pueden definir nuevos calificadores de columna siempre que se escriban datos en la tabla HBase. A diferencia de la mayoría de las bases de datos relacionales, en las que un administrador de base de datos necesita agregar una columna y definir el tipo de datos, las columnas se pueden agregar a una tabla HBase cuando surja la necesidad. Esta flexibilidad es uno de los puntos fuertes de HBase y, sin duda, es deseable cuando se trata de datos no estructurados. Con el tiempo, es probable que los datos no estructurados cambien. Por lo tanto, el nuevo contenido con nuevos calificadores de columna debe extraerse y agregarse a la tabla HBase.

Las familias de columnas ayudan a definir cómo se almacenará físicamente la tabla. Una tabla HBase se divide en regiones, pero cada región se divide en familias de columnas que se almacenan por separado en HDFS. Desde el símbolo del sistema de Linux, ejecutando **hadoop fs -ls -R / hbase** muestra cómo la tabla HBase, *mi mesa*, se almacena en HBase.

```
$ hadoop fs -ls -R / hbase
```

```
0 2014-02-28 16:40 / hbase / my_table / 028ed22e02ad07d2d73344cd53a11fb4 243 2014-02-28 16:40 /
hbase / my_table / 028ed22e02ad07d2d73344cd53a11fb4 /
      . regioninfo
0 2014-02-28 16:40 / hbase / my_table / 028ed22e02ad07d2d73344cd53a11fb4 /
      cf1
0 2014-02-28 16:40 / hbase / my_table / 028ed22e02ad07d2d73344cd53a11fb4 /
      cf2
0 2014-02-28 16:40 / hbase / my_table / 2327b09784889e6198909d8b8f342289 255 2014-02-28 16:40 /
hbase / my_table / 2327b09784889e6198909d8b8f342289 /
      . regioninfo
0 2014-02-28 16:40 / hbase / my_table / 2327b09784889e6198909d8b8f342289 /
      cf1
0 2014-02-28 16:40 / hbase / my_table / 2327b09784889e6198909d8b8f342289 /
      cf2
0 2014-02-28 16:40 / hbase / my_table / 4b4fc9ad951297efe2b9b38640f7a5fd 267 2014-02-28 16:40 /
hbase / my_table / 4b4fc9ad951297efe2b9b38640f7a5fd /
      . regioninfo
```

```

0 2014-02-28 16:40 / hbase / my_table / 4b4fc9ad951297efe2b9b38640f7a5fd /
    cf1
0 2014-02-28 16:40 / hbase / my_table / 4b4fc9ad951297efe2b9b38640f7a5fd /
    cf2
0 2014-02-28 16:40 / hbase / my_table / e40be0371f43135e36ea67edec6e31e3 267 2014-02-28 16:40 /
hbase / my_table / e40be0371f43135e36ea67edec6e31e3 /
    .regioninfo
0 2014-02-28 16:40 / hbase / my_table / e40be0371f43135e36ea67edec6e31e3 /
    cf1
0 2014-02-28 16:40 / hbase / my_table / e40be0371f43135e36ea67edec6e31e3 /
    cf2

```

Como puede verse, se han creado cuatro subdirectorios bajo / hbase / mytable. Cada subdirectorio se nombra tomando el hash de su respectivo nombre de región, que incluye las filas inicial y final. Debajo de cada uno de estos directorios están los directorios de las familias de columnas, cf1 y cf2 en el ejemplo, y el .regioninfo archivo, que contiene varias opciones y atributos sobre cómo se mantendrán las regiones. Los directorios de familias de columnas almacenan claves y valores para los calificadores de columna correspondientes. Los calificadores de columna de una familia de columnas rara vez deben leerse con los calificadores de columna de otra familia de columnas. La razón de las familias de columnas separadas es minimizar la cantidad de datos innecesarios que HBase tiene que filtrar dentro de una región para encontrar los datos solicitados. Solicitar datos de dos familias de columnas significa que se deben escanear varios directorios para extraer todas las columnas deseadas, lo que frustra el propósito de crear las familias de columnas en primer lugar. En tales casos, el diseño de la tabla puede ser mejor con una sola familia de columnas. En la práctica, el número de familias de columnas no debería ser superior a dos o tres. De otra manera,

Las siguientes operaciones agregan datos a la tabla usando el poner mando. De estos tres poner operaciones, datos1 y datos2 se ingresan en calificadores de columna, cq1 y cq2, respectivamente, en la familia de columnas cf1. El valor datos3 se ingresa en el calificador de columna cq3 en la familia de columnas cf2. La fila está designada por la tecla de fila 000700 en cada operación.

```
hbase> poner 'my_table', '000700', 'cf1: cq1', 'data1'
```

0 fila (s) en 0.0030 segundos

```
hbase> poner 'my_table', '000700', 'cf1: cq2', 'data2'
```

0 fila (s) en 0.0030 segundos

```
hbase> poner 'my_table', '000700', 'cf2: cq3', 'data3'
```

0 fila (s) en 0.0040 segundos

Los datos se pueden recuperar de la tabla HBase usando el obtener mando. Como se mencionó anteriormente, la marca de tiempo está predeterminada en milisegundos desde el 1 de enero de 1970.

```
hbase> obtener 'my_table', '000700', 'cf2: cq3'
```

COLUMNA	CELDA
cf2: cq3	marca de tiempo = 1393866138714, valor = data3

1 fila (s) en 0.0350 segundos

Por defecto, el comando `get` devuelve la versión más reciente. Para ilustrar, después de ejecutar un segundo `put` en la misma fila y columna, una `get` proporciona el valor añadido más reciente de datos 4.

```
hbase> put 'my_table', '000700', 'cf2: cq3', 'data4'
```

0 fila (s) en 0.0040 segundos

```
hbase> get 'my_table', '000700', 'cf2: cq3'
```

COLUMNA	CELDA
<code>cf2: cq3</code>	marca de tiempo = 1393866431669, valor = data4

1 fila (s) en 0.0080 segundos

Los `get` La operación puede proporcionar varias versiones especificando el número de versiones a recuperar. Este ejemplo ilustra que las celdas se presentan en orden de versión descendente.

```
hbase> get 'my_table', '000700', {COLUMN => 'cf2: cq3', VERSIONS => 2}
```

COLUMNA	CELDA
<code>cf2: cq3</code>	marca de tiempo = 1393866431669, valor = data4
<code>cf2: cq3</code>	marca de tiempo = 1393866138714, valor = data3

2 fila (s) en 1.0200 segundos

Una operación similar a la `get` es `scan`. Un `scan` recupera todas las filas entre un `STARTROW` y un `STOPROW`, pero excluyendo el `STOPROW`. Nota: si el `STOPROW` estaba configurado para 000700, solo una fila 000600 habría sido devuelta.

```
hbase> scan 'my_table', {STARTROW => '000600', STOPROW => '000800'}
```

FILA	COLUMNA + CELDA
000600	column = cf1: cq2, timestamp = 1393866792008, value = data5
000700	column = cf1: cq1, timestamp = 1393866105687, value = data1
000700	column = cf1: cq2, timestamp = 1393866122073, value = data2
000700	column = cf2: cq3, timestamp = 1393866431669, value = data4

2 fila (s) en 0.0400 segundos

La siguiente operación elimina la entrada más antigua de la columna `cf2: cq3` por fila 000700 especificando la marca de tiempo.

```
hbase> delete 'my_table', '000700', 'cf2: cq3', 1393866138714
```

0 fila (s) en 0.0110 segundos

Repetiendo lo anterior `get` La operación para obtener ambas versiones solo proporciona la última versión para esa celda. Después de todo, se eliminó la versión anterior.

```
hbase> get 'my_table', '000700', {COLUMN => 'cf2: cq3', VERSIONS => 2}
```

COLUMNA	CELDA
<code>cf2: cq3</code>	marca de tiempo = 1393866431669, valor = data4

1 fila (s) en 0.0130 segundos

Sin embargo, ejecutar un escanear operación, con la opción RAW establecida en cierto, revela que la entrada eliminada realmente permanece. La línea resaltada ilustra la creación de un marcador de lápida, que informa la obtener y escanear operaciones para ignorar todas las versiones de celda anteriores de la fila y columna en particular.

```
hbase> escanear 'my_table', {RAW => true, VERSIONS => 2,
                           STARTROW => '000700'}
```

FILA	COLUMNA + CELDA
000700	column = cf1: cq1, timestamp = 1393866105687, value = data1
000700	column = cf1: cq2, timestamp = 1393866122073, value = data2
000700	column = cf2: cq3, timestamp = 1393866431669, value = data4
000700	column = cf2: cq3, timestamp = 1393866138714, type = DeleteColumn column = cf2: cq3,
000700	timestamp = 1393866138714, value = data3

1 fila (s) en 0.0370 segundos

¿Cuándo se eliminarán permanentemente las entradas eliminadas? Para comprender este proceso, es necesario comprender cómo HBase procesa las operaciones y logra el acceso de lectura y escritura en tiempo real. Como se mencionó anteriormente, una tabla HBase se divide en regiones según la fila. Cada región es mantenida por un nodo trabajador. Durante un poner o Eliminar operación contra una región en particular, el nodo trabajador primero escribe el comando en un archivo de registro de escritura anticipada (WAL) para la región. The WAL asegura que las operaciones no se pierdan si falla un sistema. A continuación, los resultados de la operación se almacenan dentro de la RAM del nodo trabajador en un repositorio llamado MemStore [31].

Escribir la entrada en MemStore proporciona el acceso en tiempo real requerido. Cualquier cliente puede acceder a las entradas en la MemStore tan pronto como se escriban. A medida que el MemStore aumenta de tamaño o en intervalos de tiempo predeterminados, el MemStore ordenado se escribe (se borra) en un archivo, conocido como HFile, en HDFS en el mismo nodo trabajador. Una implementación típica de HBase actualiza MemStore cuando su contenido es ligeramente menor que el tamaño del bloque HDFS. Con el tiempo, estos archivos lavados se acumulan y el nodo trabajador realiza una

compactación menor que realiza una combinación ordenada de los distintos archivos lavados.

Mientras tanto, cualquier obtener o escanear las solicitudes que recibe el nodo trabajador examinan estas posibles ubicaciones de almacenamiento:

- MemStore
- Hfiles resultantes de los flashes de MemStore
- Archivos de compactaciones menores

Así, en el caso de un Eliminar operación seguida relativamente rápido por un obtener operación en la misma fila, el marcador de desecho se encuentra en elMemStore y las versiones anteriores correspondientes en los archivos H más pequeños o Hfiles fusionados previamente. Los obtener El comando se procesa instantáneamente y los datos apropiados se devuelven al cliente.

Con el tiempo, a medida que se acumulan los archivos H más pequeños, el nodo trabajador ejecuta un **compactación mayor** que fusiona los archivos H más pequeños en un archivo H grande. Durante la compactación principal, las entradas eliminadas y los marcadores de desecho se eliminan permanentemente de los archivos.

Casos de uso para HBase

Como se describe en el artículo Bigtable de Google, un caso de uso común para un almacén de datos como HBase es almacenar los resultados de un rastreador web. Usando el ejemplo de este documento, la fila com.cnn.www, por ejemplo, corresponde

a la URL de un sitio web, www.cnn.com. Una familia de columnas, llamada ancla, está de fi nido para capturar las URL del sitio web que proporcionan enlaces al sitio web de la fila. Lo que puede no ser una implementación obvia es que esas URL de sitios web de anclaje se utilizan como calificadores de columna. Por ejemplo, si Deportes Ilustrados .cnn.com proporciona un enlace a www.cnn.com, el cali fi cador de columna es sportsillustrated.cnn .com. Sitios web adicionales que proporcionan enlaces a www.cnn.com aparecen como calificadores de columna adicionales. El valor almacenado en la celda es simplemente el texto del sitio web que proporciona el enlace. Así es como puede verse el ejemplo de CNN en HBase siguiendo un obtener operación.

```
hbase> obtener 'web_table', 'com.cnn.www', {VERSIONS => 2}
```

COLUMNA	CELDA
ancla: sportsillustrated.cnn.com	marca de tiempo = 1380224620597, valor = cnn
ancla: sportsillustrated.cnn.com	marca de tiempo = 1380224000001, valor = cnn.com
ancla: edition.cnn.com	marca de tiempo = 1380224620597, valor = cnn

Se devuelven resultados adicionales para cada sitio web correspondiente que proporciona un enlace a [www.cnn .com](http://www.cnn.com). Finalmente, se requiere una explicación para usar com.cnn.www para la fila en lugar de [www.cnn .com](http://www.cnn.com). Al invertir las URL, los diversos sufijos (.com, .gov, o. red) que corresponden a los dominios de nivel superior de Internet se almacenan en orden. Además, la siguiente parte del nombre de dominio (cnn) se almacena en orden. Entonces, todos los cnn.com Los sitios web se pueden recuperar mediante un escaneo con el STARTROW de com.cnn y el apropiado STOPROW.

Este sencillo caso de uso ilustra varios puntos importantes. Primero, es posible llegar a mil millones de filas y millones de columnas en una tabla HBase. A febrero de 2014, se han identificado más de 920 millones de sitios web [32]. En segundo lugar, la fila debe definirse en función de cómo se accederá a los datos. Una tabla HBase debe diseñarse con un propósito específico en mente y un plan bien razonado sobre cómo se leerán y escribirán los datos. Por último, puede resultar ventajoso utilizar los calificadores de columna para almacenar realmente los datos de interés, en lugar de simplemente almacenarlos en una celda. En el ejemplo, a medida que se establecen nuevos sitios web de alojamiento, se convierten en nuevos calificadores de columna.

Un segundo caso de uso es el acceso al almacenamiento y búsqueda de mensajes. En 2010, Facebook implementó un sistema de este tipo utilizando HBase. En ese momento, el sistema de Facebook manejaba más de 15 mil millones de mensajes de usuario a usuario por mes y 120 mil millones de mensajes de chat por mes [33]. A continuación se describe el enfoque de Facebook para crear un índice de búsqueda para las bandejas de entrada de los usuarios. Usando cada palabra en el mensaje de cada usuario, se diseñó una tabla HBase de la siguiente manera:

- El rowwas se define como el ID de usuario.
- El cali fi cador de columna se estableció en una palabra que aparece en el mensaje.
- La versión era el ID de mensaje.
- El contenido de la celda era el conjunto de la palabra en el mensaje.

Esta implementación permitió a Facebook proporcionar la capacidad de autocompletar en el cuadro de búsqueda y devolver los resultados de la consulta rápidamente, con los mensajes más recientes en la parte superior. Siempre que los ID de mensajes aumenten con el tiempo, las versiones, almacenadas en orden descendente, garantizan que los correos electrónicos más recientes se devuelvan primero al usuario [34].

Estos dos casos de uso ayudan a ilustrar la importancia del diseño inicial de la tabla HBase en función de cómo se accederá a los datos. Además, estos ejemplos ilustran el poder de poder agregar nuevas columnas

agregando nuevos calificadores de columna, a pedido. En una implementación típica de RDBMS, las nuevas columnas requieren la participación de un DBA para alterar la estructura de la tabla.

Otras consideraciones de uso de HBase

Además de los aspectos de diseño de HBase presentados en las discusiones de casos de uso, las siguientes consideraciones son importantes para una implementación exitosa.

- **JavaAPI:** Anteriormente, se presentaron varios comandos y operaciones del shell de HBase. Los comandos de shell son útiles para explorar los datos en un entorno HBase e ilustrar su uso. Sin embargo, en un entorno de producción, la API de Java HBase podría utilizarse para programar las operaciones deseadas y las condiciones en las que ejecutar las operaciones.
- **Familia de columnas y nombres de calificadores de columna:** Es importante mantener la longitud de los nombres de las familias de columnas y los calificadores de columna lo más cortos posible. Aunque los nombres cortos tienden a ir en contra de la sabiduría convencional sobre el uso de nombres descriptivos y significativos, los nombres del nombre de la familia de la columna y el calificador de la columna se almacenan como parte de la clave de cada par clave / valor. Por lo tanto, cada byte adicional agregado a un nombre en cada fila puede sumarse rápidamente. Además, de forma predeterminada, se replican tres copias de cada bloque HDFS en el clúster de Hadoop, lo que triplica el requisito de almacenamiento.
- **De fi niendo filas:** La definición de la fila es uno de los aspectos más importantes del diseño de la tabla HBase. En general, este es el mecanismo principal para realizar operaciones de lectura / escritura en una tabla HBase. La fila debe construirse de tal manera que las columnas solicitadas se puedan recuperar fácil y rápidamente.
- **Evite crear filas secuenciales:** La tendencia natural es crear filas secuencialmente. Por ejemplo, si la clave de fila va a tener el número de identificación del cliente, y los números de identificación del cliente se crean secuencialmente, HBase puede encontrarse con una situación en la que todos los nuevos usuarios y sus datos se escriben en una sola región, que no distribuye la carga de trabajo en el clúster, según lo previsto [35]. Un enfoque para resolver tal problema es asignar al azar un prefijo al número secuencial.
- **Control de versiones:** Las opciones de la tabla HBase que se pueden definir durante la creación de la tabla o alterarse posteriormente controlan durante cuánto tiempo existirá una versión del contenido de una celda. Hay opciones para TimeToLive (TTL) después de las cuales se eliminarán las versiones anteriores. Además, hay opciones para el número mínimo y máximo de versiones a mantener.
- **Guardián del zoológico:** HBase utiliza Apache Zookeeper para coordinar y administrar las diversas regiones que se ejecutan en el clúster distribuido. En general, Zookeeper es "un servicio centralizado para mantener información de configuración, nombrar, proporcionar sincronización distribuida y proporcionar servicios grupales. Todos estos tipos de servicios son utilizados de una forma u otra por aplicaciones distribuidas". [36] En lugar de crear su propio servicio de coordinación, HBase utiliza Zookeeper. En relación con HBase, hay algunas consideraciones de configuración de Zookeeper [37].

10.2.4 Mahout

La mayor parte de este capítulo se ha centrado en procesar, estructurar y almacenar grandes conjuntos de datos utilizando Apache Hadoop y varias partes de su ecosistema. Una vez que un conjunto de datos está disponible en HDFS, el siguiente paso puede ser aplicar una técnica analítica presentada en los Capítulos 4 al 9. Herramientas como R son útiles para analizar conjuntos de datos relativamente pequeños, pero pueden sufrir problemas de rendimiento con los grandes conjuntos de datos almacenados en Hadoop. Para aplicar las técnicas analíticas dentro del entorno Hadoop, una opción es utilizar Apache

Cuidador de elefantes. Este proyecto de Apache proporciona bibliotecas Java ejecutables para aplicar técnicas analíticas de manera escalable a BigData. En general, amahout es una persona que controla a un elefante. ApacheMahout es el conjunto de herramientas que dirige a Hadoop, el elefante en este caso, a producir resultados analíticos significativos.

Mahout proporciona código Java que implementa los algoritmos para varias técnicas en las siguientes tres categorías [38]:

Clasificación:

- Regresión logística
- Bayes ingenuo
- Bosques aleatorios
- Modelos HiddenMarkov

Agrupación:

- Agrupación de dosel
- Agrupación de K-medias
- K-significa difusos
- Maximización de expectativas (EM)

Recomendados / filtrado colaborativo:

- Recomendados no distribuidos
- Filtrado colaborativo distribuido basado en elementos

Pivotal HD Enterprise con HAWQ

Los usuarios pueden descargar e instalar ApacheHadoop y las herramientas del ecosistema descritas directamente desde el www.apache.org sitio web. Otra opción de instalación es descargar distribuciones empaquetadas comercialmente de los diversos proyectos de Apache Hadoop. Estas distribuciones a menudo incluyen funciones de usuario adicionales, así como utilidades de administración de clústeres. Pivotal es una empresa que ofrece una distribución denominada Pivotal HD Enterprise, como se ilustra en la Figura 10-7.

Pivotal HD Enterprise incluye varios componentes de software Apache que se han presentado en este capítulo. El software Apache adicional incluye lo siguiente:

- **Oozie:** Administra los trabajos de Apache Hadoop actuando como un sistema de programación de flujo de trabajo
- **Sqoop:** Mueve datos de manera eficiente entre Hadoop y bases de datos relacionales
- **Canal artificial:** Recopila y agrega datos de transmisión (por ejemplo, datos de registro) La

funcionalidad adicional proporcionada por Pivotal incluye [39] lo siguiente:

- **Centro de comando** es una sólida herramienta de gestión de clústeres que permite a los usuarios instalar, configurar, configurar, monitorear y administrar componentes y servicios de Hadoop a través de una interfaz gráfica web. Simplifica la instalación, las actualizaciones y la expansión del clúster de Hadoop mediante un tablero integral con vistas instantáneas del estado del clúster y métricas clave de rendimiento. Los usuarios pueden ver información en vivo e histórica sobre el host, la aplicación,

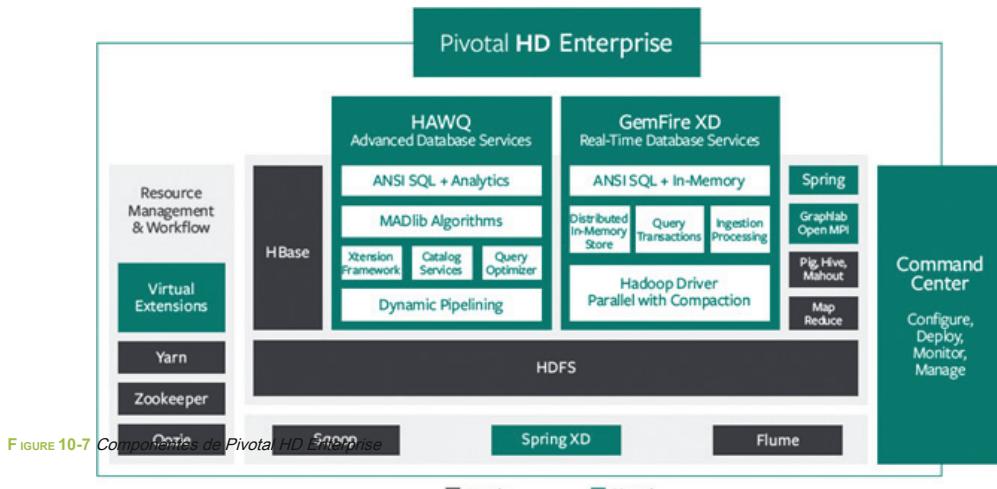


FIGURE 10-7 Componentes de Pivotal HD Enterprise

- **Graphlab onOpenMPI (interfaz de paso de mensajes)** es un muy usado y maduro

Marco de cálculo distribuido, de alto rendimiento y basado en gráficos que se adapta fácilmente a gráficos con miles de millones de vértices y aristas. Ahora puede ejecutarse de forma nativa dentro de un clúster de Hadoop existente, lo que elimina el costoso movimiento de datos. Esto permite a los científicos y analistas de datos aprovechar algoritmos populares como el rango de página, el filtrado colaborativo y la visión por computadora de forma nativa en Hadoop en lugar de copiar los datos en otro lugar para ejecutar el análisis, lo que alargaría los ciclos de la ciencia de datos. Combinado con los algoritmos de aprendizaje automático de MADlib para datos relacionales, Pivotal HD se convierte en la plataforma analítica avanzada líder para el aprendizaje automático en el mundo.

- **HadoopVirtualizationExtensions (HVE)** plug-ins hacen que Hadoop-aware del virtual

topología y escala los nodos de Hadoop de forma dinámica en un entorno virtual. Pivotal HD es la primera distribución de Hadoop que incluye complementos de HVE, lo que permite una implementación sencilla de Hadoop en un entorno empresarial. Con HVE, Pivotal HD puede ofrecer una escalabilidad verdaderamente elástica en la nube, aumentando las opciones de implementación local.

- **HAWQ (HAdoopWith Query)** agrega el poder expresivo de SQL a Hadoop para acelerar

proyectos de análisis de datos, simplifique el desarrollo al tiempo que aumenta la productividad, amplíe las capacidades de Hadoop y reduzca los costos. HAWQ puede ayudar a procesar las consultas de Hadoop más rápido que cualquier interfaz de consultas basada en Hadoop en el mercado al agregar recursos de procesamiento SQL en paralelo ricos y probados. HAWQ aprovecha los productos de análisis e inteligencia empresarial existentes y las habilidades de SQL existentes de la fuerza laboral para brindar una mejora del rendimiento de más de 100 veces a una amplia gama de tipos de consultas y cargas de trabajo.

10.3 NoSQL

NoSQL (No solo lenguaje de consulta estructurado) es un término que se utiliza para describir los almacenes de datos que se aplican a datos no estructurados. Como se describió anteriormente, HBase es una herramienta ideal para almacenar claves / valores en familias de columnas. En general, el poder de los almacenes de datos NoSQL es que a medida que aumenta el tamaño de los datos, la solución implementada puede escalar simplemente agregando máquinas adicionales al sistema distribuido. A continuación se proporcionan cuatro categorías principales de herramientas NoSQL y algunos ejemplos [40].

Almacenes de clave / valor contienen datos (el valor) a los que se puede acceder simplemente mediante un identificador dado (la clave). Como se describe en la discusión de MapReduce, los valores pueden ser complejos. En un almacén de clave / valor, no existe una estructura almacenada de cómo utilizar los datos; el cliente que lee y escribe en un almacén de claves / valores necesita mantener y utilizar la lógica de cómo extraer de manera significativa los elementos útiles de la clave y el valor. A continuación, se muestran algunos usos de los almacenes de clave / valor:

- Utilizando el ID de inicio de sesión de un cliente como clave, el valor contiene las preferencias del cliente.
- Utilizando un ID de sesión web como clave, el valor contiene todo lo que se capturó durante la sesión.

Almacenes de documentos son útiles cuando el valor del par clave / valor es un archivo y el archivo en sí es autodescriptivo (por ejemplo, JSON o XML). La estructura subyacente de los documentos se puede utilizar para consultar y personalizar la visualización del contenido de los documentos. Dado que el documento es autodescriptivo, el almacén de documentos puede proporcionar una funcionalidad adicional sobre un almacén de clave / valor. Por ejemplo, un almacén de documentos puede proporcionar la capacidad de crear índices para acelerar la búsqueda de documentos. De lo contrario, se deberían examinar todos los documentos del almacén de datos. Los almacenes de documentos pueden resultar útiles para lo siguiente:

- Gestión de contenidos de páginas web
- Análisis web de datos de registro almacenados

Tiendas de la familia de columnas son útiles para conjuntos de datos dispersos, registros con miles de columnas pero solo algunas columnas tienen entradas. El concepto de clave / valor todavía se aplica, pero en este caso, una clave está asociada con una colección de columnas. En esta colección, las columnas relacionadas se agrupan en familias de columnas. Por ejemplo, las columnas de edad, sexo, ingresos y educación pueden agruparse en una familia demográfica. Los almacenes de datos de familias de columnas son útiles en los siguientes casos:

- Para almacenar y mostrar entradas de blog, etiquetas y comentarios de los espectadores
- Para almacenar y actualizar varios contadores y métricas de páginas web

Bases de datos gráficos están pensados para casos de uso como redes, donde hay elementos (personas o enlaces a páginas web) y relaciones entre estos elementos. Si bien es posible almacenar gráficos como árboles en una base de datos relacional, a menudo resulta engorroso navegar, escalar y agregar nuevas relaciones. Las bases de datos de gráficos ayudan a superar estos posibles obstáculos y se pueden optimizar para recorrer rápidamente un gráfico (pasar de un elemento de la red a otro elemento de la red). A continuación se muestran ejemplos de implementaciones de bases de datos de gráficos:

- Redes sociales como Facebook y LinkedIn
- Aplicaciones geoespaciales como sistemas de entrega y tráfico para optimizar el tiempo para llegar a uno o más destinos.

La Tabla 10-2 proporciona algunos ejemplos de almacenes de datos NoSQL. Como suele ser el caso, la elección de un almacén de datos específico debe realizarse en función de los requisitos funcionales y de rendimiento. Un almacén de datos en particular puede proporcionar una funcionalidad excepcional en un aspecto, pero esa funcionalidad puede conllevar la pérdida de otra funcionalidad o rendimiento.

T PODER 10-2 Ejemplos de almacenes de datos NoSQL

Valor clave	Redis	redis.io
	Voldemort	www.project-voldemort.com/voldemort
Documento	CouchDB	couchdb.apache.org
	MongoDB	www.mongodb.org
Familia de columnas	Cassandra	cassandra.apache.org
	HBase	hbase.apache.org/
Grafico	FlockDB	github.com/twitter/flockdb
	Neo4j	www.neo4j.org

Resumen

Este capítulo examinó el paradigma MapReduce y su aplicación en el análisis de Big Data. Específicamente, examinó la implementación de MapReduce en Apache Hadoop. El poder de MapReduce se realiza con el uso del sistema de archivos distribuidos Hadoop (HDFS) para almacenar datos en un sistema distribuido. La capacidad de ejecutar un trabajo de MapReduce en los datos almacenados en un grupo de máquinas permite el procesamiento paralelo de petabytes o exabytes de datos. Además, al agregar máquinas adicionales al clúster, Hadoop puede escalar a medida que crecen los volúmenes de datos.

Este capítulo examinó varios proyectos de Apache dentro del ecosistema de Hadoop. Al proporcionar un lenguaje de programación de nivel superior, Apache Pig y Hive simplifican el desarrollo de código al enmascarar la lógica subyacente de MapReduce para realizar tareas comunes de procesamiento de datos como filtrar, unir conjuntos de datos y reestructurar datos. Una vez que los datos se acondicionan correctamente dentro del clúster de Hadoop, Apache Mahout se puede utilizar para realizar análisis de datos como agrupamiento, clasificación y filtrado colaborativo.

La fuerza de MapReduce en Apache Hadoop y los proyectos mencionados hasta ahora en el ecosistema de Hadoop se encuentran en entornos de procesamiento por lotes. Cuando se requiere procesamiento en tiempo real, incluyendo lectura y escritura, Apache HBase es una opción. HBase utiliza HDFS para almacenar grandes volúmenes de datos en todo el clúster, pero también mantiene los cambios recientes en la memoria para garantizar la disponibilidad en tiempo real de los datos más recientes. Mientras que MapReduce en Hadoop, Pig y Hive son herramientas de uso más general que pueden abordar una amplia gama de tareas, HBase es una herramienta algo más específica para un propósito. Los datos se recuperarán y escribirán en HBase de una manera bien entendida.

HBase es un ejemplo de los almacenes de datos NoSQL (No solo lenguaje de consulta estructurado) que se están desarrollando para abordar casos de uso específicos de Big Data. Mantener y recorrer gráficos de redes sociales son ejemplos de bases de datos relacionales que no son la mejor opción como almacén de datos. Sin embargo, las bases de datos relacionales y SQL siguen siendo herramientas poderosas y comunes y se examinarán con más detalle en el Capítulo 11.

Ejercicios

1. Investigue y documente casos de uso adicionales e implementaciones reales de Hadoop.
2. Compare y contraste Hadoop, Pig, Hive y HBase. Enumere las fortalezas y debilidades de cada conjunto de herramientas.

Investigue y resuma tres casos de uso publicados para cada conjunto de herramientas.

Los ejercicios del 3 al 5 requieren conocimientos de programación y un entorno Hadoop que funcione. El texto de la novela *Guerra y paz* se puede descargar desde <http://libros'en línea>

. library.upenn.edu/ y se utiliza como conjunto de datos para estos ejercicios. Sin embargo, se pueden sustituir fácilmente otros conjuntos de datos. Documente todos los pasos de procesamiento aplicados a los datos.

3. Utilice MapReduce en Hadoop para realizar un recuento de palabras en el conjunto de datos especificado.
4. Utilice Pig para realizar un recuento de palabras en el conjunto de datos especificado.
5. Utilice Hive para realizar un recuento de palabras en el conjunto de datos especificado.

Bibliografía

- [1] Apache, "Apache Hadoop", [en línea]. Disponible: <http://hadoop.apache.org/>. [Accedido 8 Mayo de 2014].
- [2] Wikipedia, "IBMWatson", [en línea]. Disponible: http://en.wikipedia.org/wiki/IBM_Watson. [Consultado el 11 de febrero de 2014].
- [3] D. Davidian, "IBM.com", 14 de febrero de 2011. [En línea]. Disponible: https://www-304.ibm.com/conexiones/blogs/davidian/tags/hadoop?lang=en_us. [Consultado el 11 de febrero de 2014].
- [4] IBM, "IBM.com", [en línea]. Disponible: http://www-03.ibm.com/innovation/us/watson/watson_in_healthcare.shtml. [Consultado el 11 de febrero de 2014].
- [5] LinkedIn, "LinkedIn", [en línea]. Disponible: <http://www.linkedin.com/about-us>. [Consultado el 11 de febrero de 2014].
- [6] LinkedIn, "Hadoop", [en línea]. Disponible: <http://data.linkedin.com/projects/hadoop>. [Consultado el 11 de febrero de 2014].
- [7] S. Singh, "http://developer.yahoo.com/", [en línea]. Disponible: <http://desarrollador.yahoo.com/blogs/hadoop/apache-hbase-yahoo-multi-tenancyhelm-again-171710422.html>. [Consultado el 11 de febrero de 2014].

- [8] E. Baldeschwieler, "http://www.slideshare.net", [en línea]. Disponible: http://www.slideshare.net/ydn/hadoop-yahoo-internet-scale-data-processing. [Consultado el 11 de febrero de 2014].
- [9] J. Dean y S. Ghemawat, "MapReduce: Procesamiento de datos simplificado en grandes conglomerados", [en línea]. Disponible: http://research.google.com/archive/mapreduce.pdf. [Consultado el 11 de febrero de 2014].
- [10] D. Gottfrid, "Autoservicio, diversión de supercomputación prorrteada", 1 de noviembre de 2007. [En linea]. Disponible: http://open.blogs.nytimes.com/2007/11/01/self-service-prorrteada-super-informática-divertido /. [Consultado el 11 de febrero de 2014].
- [11] "Apache.org", [en línea]. Disponible: http://www.apache.org/. [Consultado el 11 de febrero de 2014].
- [12] S. Ghemawat, H. Gobio ff y S.-T. Leung, "El sistema de archivos de Google", [en linea]. Disponible: http://static.googleusercontent.com/media/research.google.com/en/us/ archive / gfs-sosp2003.pdf. [Consultado el 11 de febrero de 2014].
- [13] D. Cutting, "Búsqueda libre: Rambilings sobre Lucene, Nutch, Hadoop y otras cosas ff", [en línea]. Disponible: http://cutting.wordpress.com. [Consultado el 11 de febrero de 2014].
- [14] "Configuración de disco de Hadoop Wiki", [en línea]. Disponible: http://wiki.apache.org/hadoop/DiskSetup. [Consultado el 20 de febrero de 2014].
- [15] "Wiki.apache.org/hadoop", [en línea]. Disponible: http://wiki.apache.org/hadoop/NameNode. [Consultado el 11 de febrero de 2014].
- [diecisésis] "HDFS High Availability" [en línea]. Disponible: http://hadoop.apache.org/docs/current / hadoop-yarn / hadoop-yarn-site / HDFSHighAvailabilityWithNFS.html. [Consultado el 8 de mayo de 2014].
- [17] Eclipse. [En línea]. Disponible: https://www.eclipse.org/downloads/. [Accedido 27 Febrero 2014].
- [18] Apache, "Hadoop Streaming", [en línea]. Disponible: https://wiki.apache.org/hadoop/HadoopStreaming. [Consultado el 8 de mayo de 2014].
- [19] "Hadoop Pipes", [en línea]. Disponible: http://hadoop.apache.org/docs/r1.2.1/api / org / apache / hadoop / mapred / pipe / package-summary.html. [Consultado el 19 de febrero de 2014].
- [20] "HDFS Design", [en línea]. Disponible: http://hadoop.apache.org/docs/stable1/hdfs_design.html. [Consultado el 19 de febrero de 2014].
- [21] "Tutorial BSP" [en línea]. Disponible: http://hama.apache.org/hama_bsp_tutorial . html. [Consultado el 20 de febrero de 2014].
- [22] "Hama", [en línea]. Disponible: http://hama.apache.org/. [Consultado el 20 de febrero de 2014].
- [23] "PoweredByYarn", [en línea]. Disponible: http://wiki.apache.org/hadoop/PoweredByYarn. [Consultado el 20 de febrero de 2014].
- [24] "Pig.apache.org", [en línea]. Disponible: http://pig.apache.org/.

- [25] "Pig", [en línea]. Disponible: <http://pig.apache.org/>. [Consultado el 11 de febrero de 2014].
- [26] "Piggybank", [en línea]. Disponible: <https://cwiki.apache.org/confluence/pantalla/PIG/PiggyBank>. [Consultado el 28 de febrero de 2014].
- [27] F. Chang, J. Dean, S. Ghemawat, WC Hsieh, DA Wallach, M. Burrows, T. Chandra, A. Fikes y RE Gruber Fay Chang, "Bigtable: A Distributed Storage System for Structured Data", [en línea]. Disponible: <http://research.google.com/archive/bigtable.html>. [Consultado el 11 de febrero de 2014].
- [28] K. Muthukkaruppan, "La tecnología subyacente de los mensajes", 15 de noviembre de 2010. [En línea]. Disponible: <http://www.facebook.com/notes/facebook-engineering/the-operating-technology-of-messages/454991608919>. [Consultado el 11 de febrero de 2014].
- [29] "Valor clave de HBase", [en línea]. Disponible: <http://hbase.apache.org/book/regions.arch.html>. [Consultado el 28 de febrero de 2014].
- [30] "Número de familias de columnas", [en línea]. Disponible: <http://hbase.apache.org/book/number.of.cfs.html>.
- [31] "HBase Regionserver", [en línea]. Disponible: <http://hbase.apache.org/book/Regionserver.arch.html>. [Consultado el 3 de marzo de 2014].
- [32] "Netcraft", [en línea]. Disponible: <http://news.netcraft.com/archives/2014/02/03/febrero-2014-web-server-survey.html>. [Consultado el 21 de febrero de 2014].
- [33] K. Muthukkaruppan, "La tecnología subyacente de los mensajes", 15 de noviembre de 2010. [En línea]. Disponible: <http://www.facebook.com/notes/facebook-engineering/the-operating-technology-of-messages/454991608919>. [Consultado en febrero de 2011 de 2014].
- [34] N. Spiegelberg. [En línea]. Disponible: <http://www.slideshare.net;brizzdotcom/mensajes-de-facebook-hbase>. [Consultado el 11 de febrero de 2014].
- [35] "HBase Rowkey", [en línea]. Disponible: <http://hbase.apache.org/book/rowkey.design.html>. [Consultado el 4 de marzo de 2014].
- [36] "Zookeeper", [en línea]. Disponible: <http://zookeeper.apache.org/>. [Consultado el 11 de febrero de 2014].
- [37] "Zookeeper", [en línea]. Disponible: <http://hbase.apache.org/book/zookeeper.html>. [Consultado el 21 de febrero de 2014].
- [38] "Mahout", [en línea]. Disponible: <http://mahout.apache.org/users/basics/algoritmos.html>. [Consultado el 19 de febrero de 2014].
- [39] "Pivotal HD", [en línea]. Disponible: <http://www.gopivotal.com/big-data/pivotal-hd>. [Consultado el 8 de mayo de 2014].
- [40] PJ Sadalage y M. Fowler, *NoSQL Distilled: una breve guía para el mundo emergente de Polyglot*, Upper Saddle River, Nueva Jersey: Addison Wesley, 2013.

11

AdvancedAnalytics— Tecnología y herramientas: Análisis en la base de datos

Conceptos clave

MADlib

Expresiones regulares

SQL

Funciones definidas por el usuario

Funciones de ventana

Análisis en la base de datos es un término amplio que describe el procesamiento de datos dentro de su repositorio. En muchos de los ejemplos de R anteriores, los datos se extrajeron de una fuente de datos y se cargaron en R. Una ventaja del análisis en la base de datos es que se elimina la necesidad de mover los datos a una herramienta analítica. Además, al realizar el análisis dentro de la base de datos, es posible obtener resultados casi en tiempo real. Las aplicaciones de análisis en la base de datos incluyen detección de fraudes en transacciones de tarjetas de crédito, recomendaciones de productos y selección de anuncios web adaptados a un usuario en particular.

La base de datos de código abierto apolular es PostgreSQL. Este nombre hace referencia a un importante lenguaje analítico en la base de datos conocido como **Lenguaje StructuredQuery (SQL)**. Este capítulo examina temas básicos y avanzados de SQL. Los ejemplos proporcionados de código SQL se probaron con Greenplumdatabase 4.1.1.1, que se basa en PostgreSQL 8.2.15. Sin embargo, los conceptos presentados son aplicables a otros entornos SQL.

11.1 Conceptos básicos de SQL

Una base de datos relacional, parte de un sistema de gestión de bases de datos relacionales (RDBMS), organiza los datos en tablas con re establecido utilizado para almacenar detalles

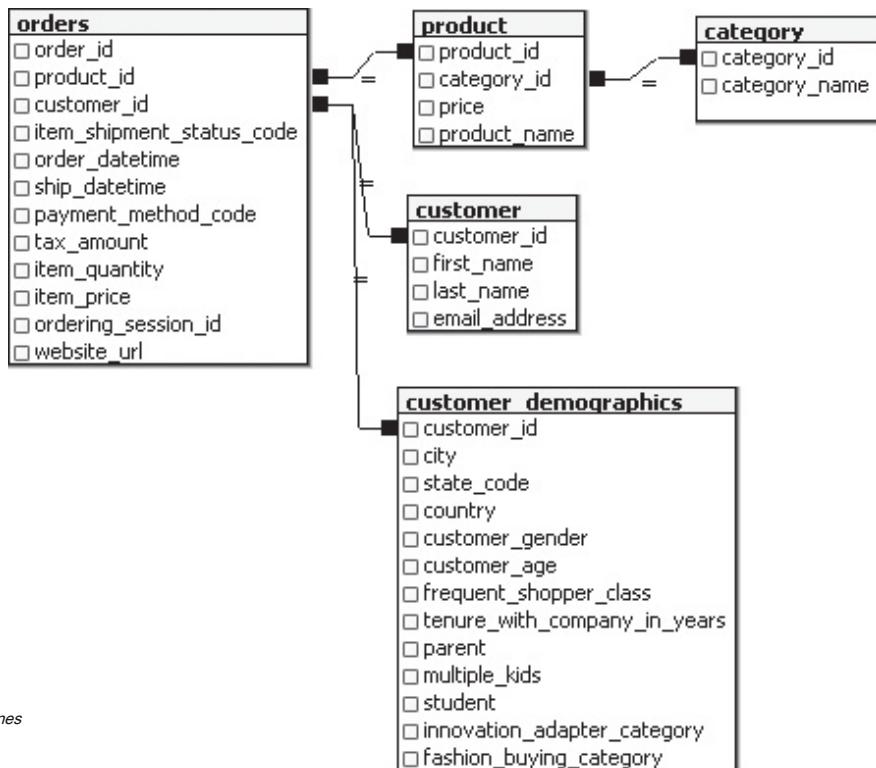


FIGURE 11-1 Diagrama de relaciones

La mesa **pedidos** contiene registros para cada transacción de pedido. Cada registro contiene elementos de datos como el **ID del Producto** ordenó el **Identificación del cliente** para el cliente que realizó el pedido, el **order_datetime**, y así. Las otras cuatro tablas proporcionan detalles adicionales sobre los artículos pedidos y el cliente. Las líneas entre las tablas en la Figura 11-1 ilustran las relaciones entre las tablas. Por ejemplo, el primer nombre, apellido y sexo de un cliente del **cliente** La tabla se puede asociar con un **pedidos** récord basado en la igualdad de **Identificación del cliente** en estas dos tablas.

Aunque es posible construir una mesa grande para guardar todos los detalles del pedido y del cliente, el uso de cinco mesas tiene sus ventajas. La primera ventaja es el ahorro de almacenamiento en disco. En lugar de almacenar el nombre del producto, que puede tener varios cientos de caracteres, en el **pedidos** mesa, mucho más corta **ID del Producto**, de quizás unos pocos bytes, se puede utilizar y almacenar en lugar del nombre del producto.

Otra ventaja es que los cambios y correcciones se realizan fácilmente. En este ejemplo, la mesa **categoría** se utiliza para categorizar cada producto. Si se descubre que se asignó una categoría incorrecta a un artículo de producto en particular, **categoría ID** en el **producto** la tabla necesita ser actualizada. Sin el **producto** y **categoría** tablas, puede ser necesario actualizar cientos de miles de registros en el **pedidos** mesa.

Una tercera ventaja es que los productos se pueden agregar a la base de datos antes de realizar cualquier pedido. De manera similar, se pueden crear nuevas categorías en previsión de que más adelante se agreguen líneas de productos completamente nuevas a las ofertas del minorista en línea.

En el diseño de una base de datos relacional, la preferencia es no duplicar datos como el nombre del cliente en varios registros. El proceso de reducir dicha duplicación se conoce como **normalización**. Es importante reconocer que una base de datos diseñada para procesar transacciones puede no estar necesariamente diseñada de manera óptima para propósitos analíticos. Las bases de datos transaccionales a menudo se optimizan para manejar la inserción de nuevos registros o las actualizaciones de los registros existentes, pero no se ajustan de manera óptima a la realización de consultas ad-hoc. Por lo tanto, al diseñar almacenes de datos analíticos, es común combinar varias de las tablas y crear una tabla más grande, aunque algunos datos puedan estar duplicados.

Independientemente del propósito de una base de datos, SQL se usa generalmente para consultar el contenido de las tablas de la base de datos relacional, así como para insertar, actualizar y eliminar datos. Una consulta SQL básica contra el **cliente** La tabla puede verse así.

```
SELECT first_name,
       apellido
  DE     cliente
 DÓNDE customer_id = 666730

primer nombre      apellido
Masón            Hu
```

Esta consulta devuelve la información del cliente para el cliente con un **Identificación del cliente** de 666730. Esta consulta SQL consta de tres partes clave:

- **SELECCIONE:** Especifica las columnas de la tabla que se mostrarán
- **DE:** Especifica el nombre de la tabla a consultar
- **DÓNDE:** Especifica el criterio o filtro a aplicar

En una base de datos relacional, a menudo es necesario acceder a datos relacionados de varias tablas a la vez. Para realizar esta tarea, la consulta SQL utiliza UNIRSE declaraciones para especificar las relaciones entre las múltiples tablas.

11.1.1 Uniones

Las uniones permiten al usuario de una base de datos seleccionar correctamente columnas de dos o más tablas. Según el diagrama de relaciones de la Figura 11-1, la siguiente consulta SQL proporciona un ejemplo del tipo más común de combinación: una combinación interna.

```
SELECCIONAR c.customer_id,
    o.order_id,
    o.product_id,
    o.item_quantity AS qty
DE      ordenes o
        INNER JOIN cliente c
            ON o.customer_id = c.customer_id
DONDE c.first_name = 'Mason'
    Y c.last_name = 'Hu'
```

customer_id	order_id	ID del Producto	cantidad
666730	51965-1172-6384-6923	33611	5
666730	79487-2349-4233-6891	34098	1
666730	39489-4031-0789-6076	33928	1
666730	29892-1218-2722-3191	33625	1
666730	07751-7728-7969-3140	34140	4
666730	85394-8022-6681-4716	33571	1

Esta consulta devuelve detalles de los pedidos realizados por el cliente Mason Hu. La consulta SQL une las dos tablas en el DE cláusula basada en la igualdad de la **Identificación del cliente** valores. En esta consulta, la específi

Identificación del cliente El programador no necesita conocer el valor para MasonHu; solo es necesario conocer el nombre completo del cliente.

Algunas funciones adicionales más allá del uso del UNIR INTERNAMENTE se introduce en esta consulta SQL. Alias o y C están asignados a tablas **pedidos** y **cliente**, respectivamente. Se utilizan alias en lugar de los nombres completos de las tablas para mejorar la legibilidad de la consulta. Por diseño, los nombres de columna especificados en el

SELECCIONE también se proporcionan en la salida. Sin embargo, el nombre de la columna generada se puede modificar con el COMO palabra clave. En la consulta SQL, los valores de **cantidad de objetos** se muestran, pero esta columna generada ahora se llama **cant**.

los UNIR INTERNAMENTE devuelve esas filas de las dos tablas donde el EN se cumple el criterio. De la consulta anterior en el **cliente** tabla, solo hay una fila en la tabla para el cliente Mason Hu. Porque el correspondiente **Identificación del cliente** porque Mason Hu aparece seis veces en el **pedidos** mesa, la

UNIR INTERNAMENTE consulta devuelve seis registros. Si el DÓNDE no se incluyó la cláusula, la consulta habría devuelto millones de filas para todos los pedidos que tenían un cliente coincidente.

Suponga que un analista desea saber qué clientes han creado una cuenta en línea pero aún no han realizado un pedido. La siguiente consulta usa un UNIÓN EXTERIOR DERECHA para identificar los primeros cinco

clientes, en orden alfabético, que no hayan realizado un pedido. La clasificación de los registros se realiza con PEDIR POR cláusula.

```
SELECCIONAR c.customer_id,
    c.first_name,
    c.last_name,
    o.order_id
DE      ordenes o
        DERECHO EXTERIOR UNIRSE al cliente c
            ON o.customer_id = c.customer_id
DONDE o.order_id ES NULL ORDER
BY c.last_name,
    c.first_name
LÍMITE 5
```

customer_id	first_name	apellido	Solicitar ID
143915	Abigail	Aaron	
965886	Audrey	Aaron	
982042	Carretero	Aaron	
125302	Daniel	Aaron	
103964	Emily	Aaron	

En la consulta SQL, un UNIÓN EXTERIOR DERECHA se utiliza para especificar que todas las filas de la tabla *cliente*, en el lado derecho (RHS) de la combinación, debe devolverse, independientemente de si hay una coincidencia *Identificación del cliente* en el *pedidos* mesa. En esta consulta, el DÓNDE La cláusula restringe los resultados a solo aquellos registros de clientes unidos donde no hay coincidencia *Solicitar ID*. NULO es una palabra clave SQL especial que denota un valor desconocido. Sin el DÓNDE cláusula, la salida también habría incluido todos los registros que tenían una coincidencia *Identificación del cliente* en el *pedidos* tabla, como se ve en la siguiente consulta SQL.

```
SELECCIONAR c.customer_id,
    c.first_name,
    c.last_name,
    o.order_id
DE      ordenes o
        DERECHO EXTERIOR UNIRSE al cliente c
            ON o.customer_id = c.customer_id
ORDER BY c.last_name,
    c.first_name
LÍMITE 5
```

Identificación del cliente	primer nombre	apellido	Solicitar ID
143915	Abigail	Aaron	
222599	Addison	Aaron	50314-7576-3355-6960
222599	Addison	Aaron	21007-7541-1255-3531
222599	Addison	Aaron	19396-4363-4499-8582
222599	Addison	Aaron	69225-1638-2944-0264

En los resultados de la consulta, el primer cliente, Abigail Aaron, no había realizado un pedido, pero el siguiente cliente, Addison Aaron, había realizado al menos cuatro pedidos.

Hay varios otros tipos de declaraciones de combinación. la IZQUIERDA COMBINACIÓN EXTERNA realiza la misma funcionalidad que el UNIÓN EXTERIOR DERECHA excepto que se consideran todos los registros de la tabla del lado izquierdo (LHS) de la combinación. UN ÚNETE A FULLOUTER incluye todos los registros de ambas tablas independientemente de si hay un registro coincidente en la otra tabla. UN ÚNETE CRUZADO combina dos tablas haciendo coincidir cada fila de la primera tabla con cada fila de la segunda tabla. Si las dos tablas tienen 100 y 1000 filas, respectivamente, el resultado ÚNETE CRUZADO de estas tablas tendrá 100.000 filas.

Los registros reales devueltos de cualquier operación de combinación dependen de los criterios establecidos en la DÓNDE cláusula. Por lo tanto, se debe tener una cuidadosa consideración al usar un DÓNDE cláusula, especialmente con combinaciones externas. De lo contrario, se puede deshacer el uso previsto de la combinación externa.

11.1.2 Establecer operaciones

SQL proporciona la capacidad de realizar operaciones de conjuntos, como uniones e intersecciones, en filas de datos. Por ejemplo, suponga que todos los registros del *pedidos* La mesa se divide en dos mesas. Los *orders_arch* La tabla, abreviatura de pedidos archivados, contiene los pedidos ingresados antes de enero de 2013. Los pedidos tramitados en o después de enero de 2013 se almacenan en el *orders_recent* mesa. Sin embargo, todos los pedidos de *ID del Producto* 33611 son necesarios para un análisis. Un enfoque sería escribir y ejecutar dos consultas independientes en las dos tablas. Los resultados de las dos consultas podrían fusionarse más tarde en un archivo o tabla independiente. Alternativamente, se podría escribir una consulta usando el UNIÓN TODOS operador de la siguiente manera:

```
SELECT id_cliente,
       Solicitar ID,
       order_datetime,
       ID del Producto,
       item_quantity AS cant.
DE      orders_arch
DÓNDE   product_id = 33611
UNIÓN   TODOS
SELECCIONE Identificación del cliente,
       Solicitar ID,
       order_datetime,
       ID del Producto,
       item_quantity AS cant.
DE      orders_recent
DONDE product_id = 33611 ORDER
BY order_datetime
```

Identificación del cliente	Solicitar ID	order_datetime	product_id	qty
643126	13501-6446-6326-0182	2005-01-02 19:28:08	33611	1
725940	70738-4014-1618-2531	2005-01-08 06:16:31	33611	1
742448	03107-1712-8668-9967	2005-01-08 16:11:39	33611	1

640847	73619-0127-0657-7016 2013-01-05 14:53:27 33611	1
660446	55160-7129-2408-9181 2013-01-07 03:59:36 33611	1
647335	75014-7339-1214-6447 2013-01-27 13: 02:10 33611	1
.	.	.
.	.	.

Los primeros tres registros de cada tabla se muestran en la salida. Debido a que los registros resultantes de ambas tablas se adjuntan juntos en la salida, es importante que las columnas se especifiquen en el mismo orden y que los tipos de datos de las columnas sean compatibles. UNIÓN TODOS fusiona los resultados de los dos SELECCIONE declaraciones independientemente de los registros duplicados que aparecen en ambos SELECCIONE declaraciones. si solo UNIÓN se utilizó, se eliminaría cualquier registro duplicado, basado en todas las columnas especificadas.

los INTERSECARSE El operador determina cualquier registro idéntico devuelto por dos SELECCIONE declaraciones. Por ejemplo, si uno quisiera saber qué artículos se compraron antes y después de 2013, la consulta SQL usando el INTERSECARSE operador sería este.

```
SELECT product_id
DE      orders_arch
INTERSECARSE
SELECT product_id
DE      orders_recent
```

ID del Producto
22
30
31
.
.
.

Es importante señalar que la intersección solo devuelve un **ID del Producto** si aparece en ambas tablas y devuelve exactamente una instancia de tal **ID del Producto**. Por lo tanto, la consulta solo devuelve una lista de ID de productos distintos.

Para contar la cantidad de productos que se pidieron antes de 2013 pero no después de ese momento, el EXCEPTO El operador se puede utilizar para excluir los ID de producto en el *orders_recent* tabla de los ID de producto en el *orders_arch* tabla, como se muestra en la siguiente consulta SQL.

```
SELECCIONAR CONTADOR (e. *)
DE      (SELECCIONAR product_id
        DE      orders_arch
        EXCEPTO
        SELECT product_id
        DE      orders_recent) e
```

13569

La consulta anterior usa la CONTAR función agregada para determinar el número de filas devueltas de una segunda consulta SQL que incluye el EXCEPTO operador. Esta consulta SQL dentro de una consulta a veces es

llamado a **subconsulta** o un **consulta anidada**. Las subconsultas permiten la construcción de consultas bastante complejas sin tener que ejecutar primero las piezas, volcar las filas en tablas temporales y luego ejecutar otra consulta SQL para procesar esas tablas temporales. Las subconsultas se pueden utilizar en lugar de una tabla dentro del DE cláusula o se puede utilizar en la DÓNDE cláusula.

11.1.3 Agrupación de extensiones

Anteriormente, el CONTAR() La función agregada se utilizó para contar el número de filas devueltas de una consulta. Estas funciones agregadas a menudo resumen un conjunto de datos después de aplicarle alguna operación de agrupación. Por ejemplo, puede ser conveniente conocer los ingresos por año o los envíos por semana. La siguiente consulta SQL usa la SUMA() función agregada junto con la AGRUPAR POR operador para proporcionar los tres principales artículos pedidos en función de **cantidad de objetos**.

```
SELECCIONE i.product_id,
           SUM (i.item_quantity) COMO total
  DE      orders_recent i
AGRUPAR POR i.product_id
ORDER BY SUM (i.item_quantity) DESC LIMIT 3
```

ID del Producto	total
15072	6089
15066	6082
15060	6053

AGRUPAR POR puede usar el ENROLLAR() operador para calcular subtotales y totales generales. La siguiente consulta SQL emplea la consulta anterior como una subconsulta en el DÓNDE cláusula para proporcionar el número de artículos pedidos por año para los tres artículos principales pedidos en general. Los ENROLLAR El operador proporciona los subtotales, que coinciden con la salida anterior para cada **ID del Producto**, así como el gran total.

```
SELECCIONE r.product_id,
           DATE_PART ('año', r.order_datetime) AS año, SUM (r.item_quantity)
                                         AS total
  DE      orders_recent r
DÓNDE  r.product_id IN (SELECCIONAR o.product_id
                         DE      orders_recent o
                         GROUP BY o.product_id
                         ORDEN POR SUMA (o.item_quantity) DESC LIMIT 3)

GRUPO  POR ROLLUP (r.product_id, DATE_PART ('año', r.order_datetime)) POR r.product_id,
ORDEN
           DATE_PART ('año', r.          order_datetime)
```

ID del Producto	año	total
15060	2013	5996
15060	2014	57
15060		6053
15066	2013	6030

15066	2014	52
15066		6082
15072	2013	6023
15072	2014	66
15072		6089
		18224

los CUBO operador amplía la funcionalidad del ENROLLAR operador proporcionando subtotales para cada columna especificada en el CUBO declaración. Modificar la consulta anterior reemplazando el ENROLLAR operador con el CUBO El operador da como resultado la misma salida con la adición de los subtotales para cada año.

SELECCIONE r.product_id,

```
    DATE_PART ('año', r.order_datetime) AS año, SUM (r.item_quantity)
                                                AS total
```

DE orders_recent r

DÓNDE r.product_id IN (SELECCIONAR o.product_id

```
    DE      orders_recent o
    GROUP BY o.product_id
    ORDEN POR SUMA (o.item_quantity) DESC LIMIT 3)
```

GRUPO POR CUBO (r.product_id, DATE_PART ('año', r.order_datetime)) POR r.product_id,

ORDEN

```
    DATE_PART ('año', r.          order_datetime
```

ID del Producto año total

15060	2013	5996
15060	2014	57
15060		6053
15066	2013	6030
15066	2014	52
15066		6082
15072	2013	6023
15072	2014	66
15072		6089
	2013	18049
	2014	175
		18224

← fila adicional

← fila adicional

Porque nulo Los valores en la salida indican las filas de subtotal y total general, se debe tener cuidado cuando nulo los valores aparecen en las columnas que se agrupan. Por ejemplo, nulo los valores pueden ser parte del conjunto de datos que se analiza. Los AGRUPAMIENTO() La función puede identificar qué filas con nulo los valores se utilizan para los subtotales o totales generales.

SELECCIONE r.product_id,

```
    DATE_PART ('año', r.order_datetime)           Como año,
    SUM (r.item_quantity)                      COMO total,
    AGRUPACIÓN (r.product_id)                  AS group_id,
    GROUPING (DATE_PART ('año', r.order_datetime)) AS group_year orders_recent r
```

DE

DÓNDE r.product_id IN (SELECCIONAR o.product_id
 DE orders_recent o
 GROUP BY o.product_id
 ORDEN POR SUMA (o.item_quantity) DESC LIMIT 3)

GRUPO POR CUBO (r.product_id, DATE_PART ('año', r.order_datetime)) POR r.product_id,
 ORDEN DATE_PART ('año', r. order_datetime)

ID del Producto	año	total	Identificación del grupo group_year
15060	2013	5996	0 0
15060	2014	57	0 0
15060		6053	0 1
15066	2013	6030	0 0
15066	2014	52	0 0
15066		6082	0 1
15072	2013	6023	0 0
15072	2014	66	0 0
15072		6089	0 1
	2013	18049	1 0
	2014	175	1 0
		18224	1 1

En la consulta anterior, *group_year* se establece en 1 cuando se calcula un total a través de los valores de *año*.

Similar, *Identificación del grupo* se establece en 1 cuando se calcula un total a través de los valores de *ID del Producto*.

La funcionalidad de ENROLLAR y CUBO se puede personalizar a través de GRUPO DE JUEGOS. La consulta SQL usando el CUBO El operador se puede reemplazar con la siguiente consulta que emplea CONJUNTOS DE AGRUPACIÓN para proporcionar los mismos resultados.

SELECCIONE r.product_id,
 DATE_PART ('año', r.order_datetime) AS año, SUM (r.item_quantity)
 AS total
 DE orders_recent r
 DÓNDE r.product_id IN (SELECCIONAR o.product_id
 DE orders_recent o
 GROUP BY o.product_id
 ORDEN POR SUMA (o.item_quantity) DESC LIMIT 3)

GRUPO POR CONJUNTOS DE AGRUPACIÓN ((r.product_id,
 DATE_PART ('año', r.order_datetime)),
 (r.product_id),
 (DATE_PART ('año', r.order_datetime)), ())

PEDIR POR r.product_id,
 DATE_PART ('año', r.order_datetime)

Los conjuntos de agrupación enumerados definen las columnas para las que se proporcionarán subtotales. El último conjunto de agrupaciones, (), especifica que el total general se proporciona en los resultados de la consulta. Por ejemplo, si solo se deseaba el gran total, la siguiente consulta SQL usando CONJUNTOS DE AGRUPACIÓN puede ser usado.

```

SELECCIONE r.product_id,
           DATE_PART ('año', r.order_datetime) AS año, SUM (r.item_quantity)
                                         AS total
DE      orders_recent r
DÓNDE   r.product_id IN (SELECCIONAR o.product_id
                         DE      orders_recent o
                         GROUP BY o.product_id
                         ORDEN POR SUMA (o.item_quantity) DESC LIMIT 3)

GRUPO    POR CONJUNTOS DE AGRUPACIÓN ((r.product_id,
                                         DATE_PART ('año', r.order_datetime)),
                                         ())
PEDIR POR r.product_id,
           DATE_PART ('año', r.order_datetime)

```

ID del Producto	año	total
15060	2013	5996
15060	2014	57
15066	2013	6030
15066	2014	52
15072	2013	6023
15072	2014	66
		18224

Porque el AGRUPAR POR la cláusula puede contener múltiples CUBO, ROLLUP, o especifiaciones de columna, pueden producirse conjuntos de agrupación duplicados. Los IDENTIFICACIÓN DEL GRUPO() la función identifica las filas únicas con un 0 y las filas redundantes con un 1, 2,.... Para ilustrar la función IDENTIFICACIÓN DEL GRUPO(), ambos ENROLLAR y CUBO se utilizan cuando sólo una especifiación **ID del Producto** está siendo examinado.

```

SELECCIONE r.product_id,
           DATE_PART ('año', r.order_datetime) AS año, SUM (r.item_quantity)
                                         COMO total,
                                         AS group_id
DE      orders_recent r
DÓNDE   r.product_id IN (15060)
GRUPO    POR ROLLUP (r.product_id, DATE_PART ('año', r.order_datetime)),
                                         CUBE (r.product_id, DATE_PART ('año', r.order_datetime)) POR r.product_id,
ORDEN
           DATE_PART ('año', r.order_datetime),
           IDENTIFICACIÓN DEL GRUPO()

```

ID del Producto	año	total	Identificación del grupo
15060	2013	5996	0
15060	2013	5996	1
15060	2013	5996	3
15060	2013	5996	4
15060	2013	5996	5
15060	2013	5996	6
15060	2014	57	0
15060	2014	57	1
15060	2014	57	2
15060	2014	57	3
15060	2014	57	4
15060	2014	57	5
15060	2014	57	6
15060		6053	0
15060		6053	1
15060		6053	2
	2013	5996	0
	2014	57	0
		6053	0

Filtrar en el **Identificación del grupo** los valores iguales a cero producen registros únicos. Este filtrado se puede lograr con el TENIENDO cláusula, como se ilustra en la siguiente consulta SQL.

```

SELECCIONE r.product_id,
    DATE_PART ('año', r.order_datetime) AS año, SUM (r.item_quantity)
        COMO total,
        IDENTIFICACIÓN DEL GRUPO() AS group_id
DE      orders_recent r
DÓNDE   r.product_id IN (15060)
GRUPO    POR ROLLUP (r.product_id, DATE_PART ('año', r.order_datetime)),
        CUBE (r.product_id, DATE_PART ('año', r.order_datetime)) GROUP_ID () = 0
TENIENDO
ORDEN    POR r.product_id,
        DATE_PART ('año', r.order_datetime),
        IDENTIFICACIÓN DEL GRUPO()
    
```

ID del Producto	año	total	Identificación del grupo
15060	2013	5996	0
15060	2014	57	0
15060		6053	0
	2013	5996	0
	2014	57	0
		6053	0

11.2 Análisis de texto en la base de datos

SQL ofrece varias funciones básicas de cadenas de texto, así como una función de búsqueda con comodines. Relacionado SELECCIONE declaraciones y sus resultados incluidos en los delimitadores de comentarios SQL, / ** /, incluyen lo siguiente:

```

SELECCIONAR SUBSTRING ('1234567890', 3,2)          /* returns '34'      */
SELECT '1234567890' LIKE '%7%' SELECT             /* returns True      */
'1234567890' LIKE '7%' SELECT '1234567890'        /* returns False     */
LIKE '_2%' SELECT '1234567890' LIKE '_3%'          /* returns True      */
SELECT '1234567890' LIKE '_3%'                   /* returns False     */
                                                /* returns True      */

```

This section examines more dynamic and flexible tools for text analysis, called **regular expressions**, and their use in SQL queries to perform pattern matching. Table 11-1 includes several forms of the comparison operator used with regular expressions and related SQL examples that produce a True result.

TABLE 11-1 Regular Expression Operators

~	Contains the regular expression (case sensitive)	'123a567' ~ 'a'
~*	Contains the regular expression (case insensitive)	'123a567' ~* 'A'
!~	Does not contain the regular expression (case sensitive)	'123a567' !~ 'A'
!~*	Does not contain the regular expression (case insensitive)	'123a567' !~* 'b'

More complex forms of the patterns that are specified at the RHS of the comparison operator can be constructed by using the elements in Table 11-2.

TABLE 11-2 Regular Expression Elements

	Matches item a or b (a b)
^	Looks for matches at the beginning of the string Looks
\$	for matches at the end of the string Matches any single
.	character
*	Matches preceding item zero or more times Matches
+	preceding item one or more times Makes the preceding
?	item optional Matches the preceding item exactly n
{n}	times
(continues)	

TABLE 11-2 Regular Expression Elements (Continued)

()	Matches the contents exactly
[]	Matches any of the characters in the content, such as [0-9] Matches a
\x	nonalphanumeric character named x Matches an escape string \y
\y	
To illustrate the use of these elements, the following SELECT statements include examples in which the comparisons are True or False.	
/* matches x or y ('x y')*/	
SELECT '123a567' ~ '23 b' SELECT	/* returns True */ /* returns
'123a567' ~ '32 b'	False */
/* matches the beginning of the string */	
SELECT '123a567' ~ '^123a'	/* returns True */
SELECT '123a567' ~ '^123a7'	/* returns False */
/* matches the end of the string */	
SELECT '123a567' ~ 'a567\$'	/* returns True */
SELECT '123a567' ~ '27\$'	/* returns False */
/* matches any single character */	
SELECT '123a567' ~ '2.a'	/* returns True */
SELECT '123a567' ~ '2..5'	/* returns True */
SELECT '123a567' ~ '2...5'	/* returns False */
/* matches preceding character zero or more times */	
SELECT '123a567' ~ '2*'/*	/* returns True */
SELECT '123a567' ~ '2*a'	/* returns True */
SELECT '123a567' ~ '7*a'	/* returns True */
SELECT '123a567' ~ '37*'/*	/* returns True */
SELECT '123a567' ~ '87*'/*	/* returns False */
/* matches preceding character one or more times */	
SELECT '123a567' ~ '2+'/*	/* returns True */
SELECT '123a567' ~ '2+a'	/* returns False */
SELECT '123a567' ~ '7+a'	/* returns False */
SELECT '123a567' ~ '37+'/*	/* returns False */
SELECT '123a567' ~ '87+'/*	/* returns False */
/* makes the preceding character optional */	
SELECT '123a567' ~ '2?'	/* returns True */

```
SELECT '123a567' ~ '2?a' SELECT          /* returns True */
'123a567' ~ '7?a' SELECT '123a567' ~  /* returns True */
'37?' SELECT '123a567' ~ '87?'          /* returns False */
```

/* Matches the preceding item exactly {n} times */

SELECT '123a567' ~ '5{0}' SELECT	/* returns True */
'123a567' ~ '5{1}' SELECT '123a567' ~	/* returns True */
'5{2}' SELECT '1235567' ~ '5{2}'	/* returns False */
SELECT '123a567' ~ '8{0}' SELECT	/* returns True */
'123a567' ~ '8{1}'	/* returns True */
	/* returns False */

/* Matches the contents exactly */

SELECT '123a567' ~ '(23a5)'	/* returns True */
SELECT '123a567' ~ '(13a5)'	/* returns False */
SELECT '123a567' ~ '(23a5)7*'	/* returns True */
SELECT '123a567' ~ '(23a5)7+'	/* returns False */

/* Matches any of the contents */

SELECT '123a567' ~ '[23a8]' SELECT	/* returns True */
'123a567' ~ '[8a32]' SELECT '123a567' ~	/* returns True */
'[(13a5)]' SELECT '123a567' ~ '[xyz9]'	/* returns True */
SELECT '123a567' ~ '[a-z]' SELECT '123a567'	/* returns False */
~ '[b-z]'	/* returns True */
	/* returns False */

/* Matches a nonalphanumeric */

SELECT '\$50K+' ~ '\\$' SELECT '\$50K+'	/* returns True */
~ '\+' SELECT '\$50K+' ~ '\\$\+'	/* returns False */

/* Use of the backslash for escape clauses

*/

/* \w denotes the characters 0-9, a-z, A-Z, or the underscore(_) */	/* */
SELECT '123a567' ~ '\w'	/* returns True */
SELECT '123a567+' ~ '\w'	/* returns True */
SELECT '+++++++' ~ '\w'	/* returns False */
SELECT '_' ~ '\w'	/* returns True */
SELECT '+' ~ '\w'	/* returns False */

Regular expressions can be developed to identify mailing addresses, e-mail addresses, phone numbers, or currency amounts.

/* use of more complex regular expressions */

SELECT '\$50K+' ~ '\\${0-9}*K\+'	/* returns True */
SELECT '\$50K+' ~ '\\${0-9}K\+'	/* returns False */
SELECT '\$50M+' ~ '\\${0-9}*K\+'	/* returns False */

```

SELECT '$50M+' ~ '\$[0-9]*(K|M)\+'          /* returns True */

/* check for ZIP code of form ##### #### */
SELECT '02038-2531' ~ '[0-9]{5}-[0-9]{4}' /* returns True */
SELECT '02038-253' ~ '[0-9]{5}-[0-9]{4}'  /* returns False */
SELECT '02038'      ~ '[0-9]{5}-[0-9]{4}'  /* returns False */

```

So far, the application of regular expressions has been illustrated by including the Boolean comparison in a SELECT statement as if the result of the comparison was to be returned as a column. In practice, these comparisons are used in a SELECT statement's WHERE clause against a table column to identify specific records of interest. For example, the following SQL query identifies those ZIP codes in a table of customer addresses that do not match the form ##### ####. Once the invalid ZIP codes are identified, corrections can be made by manual or automated means.

```

SELECT address_id,
       customer_id,
       city,
       state,
       zip,
       country
  FROM   customer_addresses
 WHERE zip !~ "[0-9]{5}-[0-9]{4}"

```

address_id	customer_id	city	state	zip	country
7	13	SINAI	SD	57061-0236	USA
18	27	SHELL ROCK	IA	S0670-0480	USA
24	37	NASHVILLE	TN	37228-219	USA
.					
.					
.					

SQL functions enable the use of regular expressions to extract the matching text, such as SUBSTRING(), as well as update the text, such as REGEXP_REPLACE().

```

/* extract ZIP code from text string */
SELECT SUBSTRING('4321A Main Street Franklin, MA 02038-2531' FROM '[0-9]{5}-[0-9]{4}')

```

02038-2531

```

/* replace long format zip code with short format ZIP code */
SELECT REGEXP_REPLACE('4321A Main Street Franklin, MA 02038-2531',
                      '[0-9]{5}-[0-9]{4}',
                      SUBSTRING(SUBSTRING('4321A Main Street Franklin, MA 02038-2531' FROM
                      '[0-9]{5}-[0-9]{4}'),1,5)
)

```

4321A Main Street Franklin, MA 02038

Regular expressions provide considerable flexibility in searching and modifying text strings. However, it is quite easy to build a regular expression that does not work entirely as intended. For example, a particular operation may work properly with a given dataset, but future datasets may contain new cases to be handled. Thus, it is important to thoroughly test any SQL code using regular expressions.

11.3 Advanced SQL

Building upon the foundation provided in the earlier parts of this chapter, this section presents advanced SQL techniques that can simplify in-database analytics.

11.3.1 Window Functions

In Section 11.1.3, several SQL examples using aggregate functions and grouping options to summarize a dataset were provided. A **window function** enables aggregation to occur but still provides the entire dataset with the summary results. For example, the RANK() function can be used to order a set of rows based on some attribute. Based on the SQL table, *orders_recent*, introduced in Section 11.1.2, the following SQL query provides a ranking of customers based on their total expenditures.

```
SELECT s.customer_id,
       s.sales,
       RANK()
       OVER (
           ORDER BY s.sales DESC ) AS sales_rank (SELECT
FROM      r.customer_id,
           SUM(r.item_quantity * r.item_price) AS sales orders_recent r
           FROM
           GROUP BY r.customer_id) s
```

customer_id	sales	sales_rank
683377	27840.00	1
238107	19983.65	2
661519	18134.11	3
628278	17965.44	4
619660	17944.20	5
.		
.		
.		

The subquery in the FROM clause computes the total sales for each customer. In the outermost SELECT clause, the sales are ranked in descending order. Window functions, such as RANK(), are followed by an OVER clause that specifies how the function should be applied. Additionally, the window function can be applied to groupings of a given dataset using the PARTITION BY clause. The following SQL query provides the customer rankings based on sales within product categories.

```
SELECT s.category_name,
       s.customer_id,
```

```

s.sales,
RANK()
OVER (
    PARTITION BY s.category_name
    ORDER BY s.sales DESC ) AS sales_rank (SELECT
FROM c.category_name,
r.customer_id,
SUM(r.item_quantity * r.item_price) AS sales orders_recent r
FROM
LEFT OUTER JOIN product p
ON r.product_id = p.product_id
LEFT OUTER JOIN category c
ON p.category_id = c.category_id
GROUP BY c.category_name,
r.customer_id) s
ORDER BY s.category_name,
sales_rank

```

category_name	customer_id	sales	sales_rank
Apparel	596396	4899.93	1
Apparel	319036	2799.96	2
Apparel	455683	2799.96	2
Apparel	468209	2700.00	4
Apparel	456107	2118.00	5
.			
.			
.			
Apparel	430126	2.20	78731
Automotive Parts and Accessories	362572	5706.48	1
Automotive Parts and Accessories	587564	5109.12	2
Automotive Parts and Accessories	377616	4279.86	3
Automotive Parts and Accessories	443618	4279.86	3
Automotive Parts and Accessories	590658	3668.55	5
.			
.			
.			

In this case, the subquery determines each customer's sales in the respective product category. The outer SELECT clause then ranks the customer's sales within each category. The provided portions of the SQL query output illustrate that the ranking begins at 1 for each category and demonstrate how the rankings are affected by ties in the amount of *sales*.

A second use of windowing functions is to perform calculations over a sliding window in time. For example, moving averages can be used to smooth weekly sales figures that may exhibit large week-to-week variation, as shown in the plot in Figure 11-2.

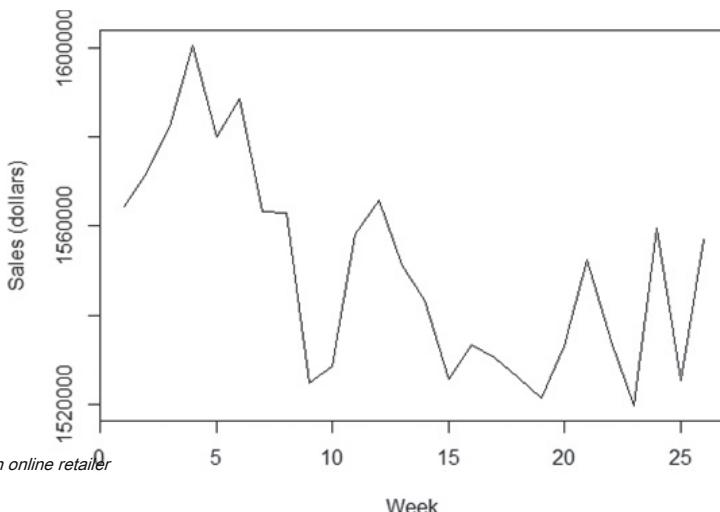


FIGURE 11-2 Weekly sales for an online retailer

The following SQL query illustrates how moving averages can be implemented using window functions:

```

SELECT year,
       week,
       sales,
       AVG(sales)
          OVER (
            ORDER BY year, week
            ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING) AS moving_avg sales_by_week
FROM
WHERE year = 2014
      AND week <= 26
ORDER BY year,
           week
  
```

year	week	sales	moving_avg
2014	1	1564539	1572999.333 ← average of weeks 1, 2, 3
2014	2	1572128	1579941.75 ← average of weeks 1, 2, 3, 4
2014	3	1582331	1579982.6 ← average of weeks 1, 2, 3, 4, 5
2014	4	1600769	1584834.4 ← average of weeks 2, 3, 4, 5, 6
2014	5	1580146	1583037.2 ← average of weeks 3, 4, 5, 6, 7
2014	6	1588798	1579179.6
2014	7	1563142	1563975.6
2014	8	1563043	1553665
2014	9	1524749	1547534.8

2014	10	1528593	1548051.6
2014	11	1558147	1545714.2
2014	12	1565726	1549404
2014	13	1551356	1548812.6
2014	14	1543198	1543820.2
2014	15	1525636	1536767.6
2014	16	1533185	1531662.2
2014	17	1530463	1527313.6
2014	18	1525829	1528787.8
2014	19	1521455	1532649
2014	20	1533007	1533370
2014	21	1552491	1532116
2014	22	1534068	1539713.6
2014	23	1519559	1538199.6
2014	24	1559443	1539086.2 ←average of weeks 22,23,24,25,26
2014	25	1525437	1540340.75 ←average of weeks 23,24,25,26
2014	26	1556924	1547268 ← average of weeks 24,25,26

The windowing function uses the built-in aggregate function AVG(), which computes the arithmetic average of a set of values. The ORDER BY clause sorts the records in chronological order and specifies which rows should be included in the averaging process with the current row. In this SQL query, the moving average is based on the current row, the preceding two rows, and the following two rows. Because the dataset does not include the last two weeks of 2013, the first moving average value of 1,572,999.333 is the average of the first three weeks of 2014: the current week and the two subsequent weeks. The moving average value for the second week, 1,579,941.75, is the sales value for week 2 averaged with the prior week and the two subsequent weeks. For weeks 3 through 24, the moving average is based on the sales from 5-week periods, centered on the current week. At week 25, the window begins to include fewer weeks because the following

ss against
the weekly sales fi

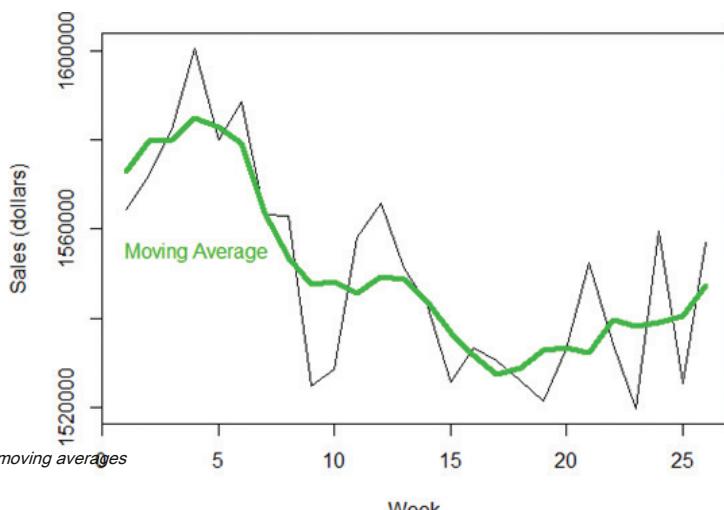


FIGURE 11-3 Weekly sales with moving averages

Built-in window functions may vary by SQL implementation. Table 11-3 [1] from the PostgreSQL documentation includes the list of general-purpose window functions.

TABLE 11-3 Window Functions

row_number()	Number of the current row within its partition, counting from 1.
rank()	Rank of the current row with gaps, same as row_number of its first peer.
dense_rank()	Rank of the current row without gaps; this function counts peer groups.
percent_rank()	Relative rank of the current row: $(\text{rank} - 1) / (\text{total rows} - 1)$.
cume_dist()	Relative rank of the current row: $(\text{number of rows preceding or peer with current row}) / (\text{total rows})$.
ntile(num_buckets integer)	Integer ranging from 1 to the argument value, dividing the partition as equally as possible.
lag(value any [, offset integer [, default any]])	Returns the value evaluated at the row that is offset rows before the current row within the partition; if there is no such row, instead return default. Both offset and default are evaluated with respect to the current row. If omitted, offset defaults to 1 and default to null.
lead(value any [, offset integer [, default any]])	Returns the value evaluated at the row that is offset rows after the current row within the partition; if there is no such row, instead return default. Both offset and default are evaluated with respect to the current row. If omitted, the offset defaults to 1 and the default to null.
first_value(value any)	Returns the value evaluated at the first row of the window frame.
last_value(value any)	Returns the value evaluated at the last row of the window frame.
nth_value(value any, nth integer)	Returns the value evaluated at the nth row of the window frame (counting from 1); null if no such row.
http://www.postgresql.org/docs/9.3/static/functions-window.html	

11.3.2 User-Defined Functions and Aggregates

When the built-in SQL functions are insufficient for a particular task or analysis, SQL enables the user to create user-defined functions and aggregates. This custom functionality can be incorporated into SQL queries in the same ways that the built-in functions and aggregates are used. User-defined functions can also be created to simplify processing tasks that a user may commonly encounter.

For example, a user-defined function can be written to translate text strings for female (F) and male (M) to 0 and 1, respectively. Such a function may be helpful when formatting data for use in a regression analysis. Such a function, fm_convert(), could be implemented as follows:

```
CREATE FUNCTION fm_convert(text) RETURNS integer AS 'SELECT CASE
```

```
    WHEN $1 = "F" THEN 0 WHEN $1
    = "M" THEN 1 ELSE NULL
```

```
END'
```

```
LANGUAGE SQL
```

```
IMMUTABLE
```

```
RETURNS NULL ON NULL INPUT
```

In declaring the function, the SQL query is placed within single quotes. The first and only passed value is referenced by \$ 1. The SQL query is followed by a LANGUAGE statement that explicitly states that the preceding statement is written in SQL. Another option is to write the code in C. IMMUTABLE indicates that the function does not update the database and does not use the database for lookups. The

IMMUTABLE declaration informs the database's query optimizer howbest to implement the function. The RETURNS NULL ON NULL INPUT statement specifies how the function addresses the case when any of the inputs are null values.

In the online retail example, the fm_convert() function can be applied to the *customer_gender* column in the *customer_demographics* table as follows.

```
SELECT customer_gender,
       fm_convert(customer_gender) as male
  FROM   customer_demographics
 LIMIT 5
```

customer_gender	male
M	1
F	0
F	0
M	1
M	1

Built-in and user-defined functions can be incorporated into user-defined aggregates, which can then be used as a window function. In Section 11.3.1, a window function is used to calculate moving averages to smooth a data series. In this section, a user-defined aggregate is created to calculate an ***Exponentially Weighted Moving Average (EWMA)***. For a given time series, the EWMA series is defined as shown in Equation 11-1.

$$\begin{aligned} & \text{for } t = 1 \\ & \text{EWMA}_t = y_t \\ & \text{for } t \geq 2 \\ & \text{EWMA}_t = \alpha \cdot y_t + (1 - \alpha) \cdot \text{EWMA}_{t-1} \end{aligned} \quad (11-1)$$

where $0 \leq \alpha \leq 1$

The smoothing factor, determines howmuchweight to place on the latest point in a given time series. By repeatedly substituting into Equation 11-1 for the prior value of the EWMA series, it can be shown that the weights against the original series are exponentially decaying backward in time.

To implement EWMA smoothing as a user-defined aggregate in SQL, the functionality in Equation 11-1 needs to be implemented first as a user-defined function.

```
CREATE FUNCTION ewma_calc(numeric, numeric, numeric) RETURNS numeric as
/* $1 = prior value of EWMA */ 
/* $2 = current value of series */
/* $3 = alpha, the smoothing factor */
'SELECT CASE
    WHEN $3 IS NULL                      /* bad alpha */
    OR $3 < 0
    OR $3 > 1 THEN NULL WHEN $1 IS NULL
    THEN $2 WHEN $2 IS NULL THEN $1          /* t = 1 */
    ELSE ($3 * $2) + (1-$3) *$1 END'
                                         /* y is unknown */ /* t >= 2 */
                                         /* */

*/
```

LANGUAGE SQL

IMMUTABLE

Accepting three numeric inputs as defined in the comments, the `ewma_calc()` function addresses possible bad values of the smoothing factor as well as the special case in which the other inputs are null.

The `ELSE` statement performs the usual EWMA calculation. Once this function is created, it can be referenced in the user-defined aggregate, `ewma()`.

```
CREATE AGGREGATE ewma(numeric, numeric)
(SFUNC = ewma_calc,
 STYPE = numeric,
 PREFUNC = dummy_function)
```

In the `CREATE AGGREGATE` statement for `ewma()`, `SFUNC` assigns the state transition function (`ewma_calc` in this example) and `STYPE` assigns the data type of the variable to store the current state of the aggregate. The variable for the current state is made available to the `ewma_calc()` function as the first variable, `$ 1`. In this case, because the `ewma_calc()` function requires three inputs, the `ewma()`

aggregate requires only two inputs; the state variable is always internally available to the aggregate. The `PREFUNC` assignment is required in the Greenplum database for use in a massively parallel processing (MPP) environment. For some aggregates, it is necessary to perform some preliminary functionality on the current state variables for a couple of servers in the MPP environment. In this example, the assigned

`PREFUNC` function is added as a placeholder and is not utilized in the proper execution of the `ewma()` aggregate function.

As a window function, the `ewma()` aggregate, with a smoothing factor of 0.1, can be applied to the weekly sales data as follows.

```
SELECT year,
       week,
       sales,
       ewma(sales, .1)
   OVER (
        ORDER BY year, week)
FROM     sales_by_week
WHERE year = 2014
```

AND week <= 26
ORDER BY year,
week

year	week	sales	ewma
2014	1	1564539	1564539.00
2014	2	1572128	1565297.90
2014	3	1582331	1567001.21
2014	4	1600769	1570377.99
2014	5	1580146	1571354.79
.	.	.	.
2014	23	1519559	1542043.47
2014	24	1559443	1543783.42
2014	25	1525437	1541948.78
2014	26		

Figure 11-4 incl

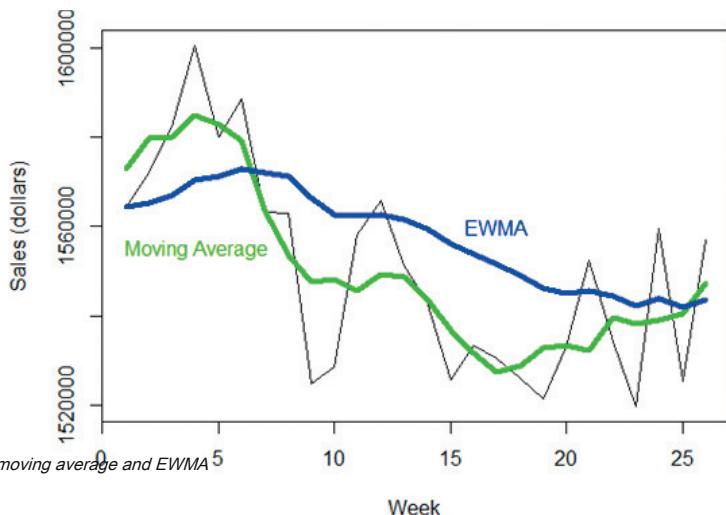


FIGURE 11-4 Weekly sales with moving average and EWMA

Increasing the value of the smoothing factor from 0.1 causes the EWMA to follow the actual data better, but the trade-off is that large fluctuations in the data cause larger fluctuations in the smoothed series. The user-defined aggregate, `ewma()`, is used in the SQL query in the same manner as any other window function with the specification of the OVER clause.

11.3.3 Ordered Aggregates

Sometimes the value of an aggregate may depend on an ordered set of values. For example, to determine the median of a set of values, it is common to first sort the values from smallest to largest and identify the median from the center of the sorted values.

The sorting can be accomplished by using the function

`array_agg()`. The following SQL query calculates the median of the weekly sales data.

```
SELECT (d.ord_sales[ d.n/2 + 1 ] +
       d.ord_sales[ (d.n + 1)/2 ]) / 2.0 as median
  FROM   (SELECT ARRAY_AGG(s.sales ORDER BY s.sales) AS ord_sales,
                COUNT(*) AS n
               FROM   sales_by_week s
              WHERE s.year = 2014
                AND s.week <= 26) d
```

median

1551923.5

In general, the function `ARRAY_AGG()` builds an array from a table column. Executing the subquery from the previous SQL query for just the first five weeks illustrates the creation of the array, denoted by the braces, and the sorted weekly sales within the array.

```
SELECT ARRAY_AGG(s.sales ORDER BY s.sales) AS ord_sales,
       COUNT(*) AS n
      FROM   sales_by_week s
     WHERE s.year = 2014
       AND s.week <= 5
```

ord_sales	n
{1564539,1572128,1580146,1582331,1600769}	5

Besides creating an array, the values can be concatenated together into one text string using the `string_agg()` function.

```
SELECT STRING_AGG(s.sales ORDER BY s.sales) AS ord_sales,
       COUNT(*) AS n
      FROM   sales_by_week s
     WHERE s.year = 2014
```

```
AND s.week <= 5
```

ord_sales	n
15645391572128158014615823311600769	5

However, in this particular example, it may be useful to separate the values with a delimiter, such as a comma.

```
SELECT STRING_AGG(s.sales, ',' ORDER BY s.sales) AS ord_sales,
       COUNT(*) AS n
  FROM sales_by_week s
 WHERE s.year = 2014
       AND s.week <= 5
```

ord_sales	n
1564539,1572128,1580146,1582331,1600769	5

Although the sorted sales appear to be an array, there are no braces around the output. So the displayed ordered sales are a text string.

11.3.4 MADlib

SQL implementations include many basic analytical and statistical built-in functions, such as means and variances. As illustrated in

this chapter, SQL also enables the development of user-defined functions and aggregates to provide additional functionality.

Furthermore, SQL databases can utilize an external library of functions. One such library is known as **MADlib**. The description file [2] included with the MADlib library download states the following:

MADlib is an open-source library for scalable in-database analytics. It offers data-parallel implementations of mathematical, statistical, and machine learning methods for structured and unstructured data.

The concept of Magnetic/Agile/Deep (MAD) analysis skills was introduced in a 2009 paper by Cohen, et al. [3]. This paper describes the components of MAD as follows:

- **Magnetic:** Traditional Enterprise Data Warehouse (EDW) approaches “repel” new data sources, discouraging their incorporation until they are carefully cleansed and integrated. Given the ubiquity of data in modern organizations, a data warehouse can keep pace today only by being “magnetic”: attracting all the data sources that crop up within an organization regardless of data quality niceties.
- **Agile:** Data Warehousing orthodoxy is based on long-range and careful design and planning. Given growing numbers of data sources and increasingly sophisticated and mission-critical data analyses, a modern warehouse must instead allow analysts to easily ingest, digest, produce, and adapt data rapidly. This requires a database whose physical and logical contents can be in continuous rapid evolution.
- **Deep:** Modern data analyses involve increasingly sophisticated statistical methods that go well beyond the rollups and drilldowns of traditional business intelligence (BI). Moreover, analysts often need to see both the forest and the trees in running these algorithms; they want to study enormous datasets without resorting to samples and extracts. The modern data warehouse should serve both as a deep data repository and as a sophisticated algorithmic runtime engine.

In response to the inability of a traditional EDW to readily accommodate new data sources, the concept of a *data lake* has emerged. A data lake represents an environment that collects and stores large volumes of structured and unstructured datasets, typically in their original, unaltered forms. More than a data depository, the data lake architecture enables the various users and data science teams to conduct data exploration and related analytical activities. Apache Hadoop is often considered a key component of building a data lake [4].

Because MADlib is designed and built to accommodate massive parallel processing of data, MADlib is ideal for Big Data in-database analytics. MADlib supports the open-source database PostgreSQL as well as the Pivotal GreenplumDatabase and Pivotal HAWQ. HAWQ is a SQL query engine for data stored in the HadoopDistributed File System(HDFS). ApacheHadoop and the Pivotal products were described in Chapter

10, "Advanced Analytics—Technology and Tools: MapReduce and Hadoop." MADlib version 1.6 modules [5] are described in Table 11-4.

TABLE 11-4 *MADlib Modules*

Generalized Linear Models	Includes linear regression, logistic regression, and multinomial logistic regression
Cross Validation	Evaluates the predictive power of a fitted model Solves
Linear Systems	dense and sparse linear system problems
Matrix Factorization	Performs low-rank matrix factorization and singular value decomposition
Association Rules	Implements the Apriori algorithm to identify frequent itemsets Implements
Clustering	k-means clustering
Topic Modeling	Provides a Latent Dirichlet Allocation predictive model for a set of documents
Descriptive Statistics	Simplifies the computation of summary statistics and correlations Conducts
Inferential Statistics	hypothesis tests
Support Modules	Provides general array and probability functions that can also be used by other MADlib modules
Dimensionality Reduction	Enables principal component analyses and projections
Time Series Analysis	Conducts ARIMA analyses
http://doc.madlib.net/latest/modules.html	

In the following example, MADlib is used to perform a k-means clustering analysis, as described in Chapter 4, "Advanced Analytical Theory and Methods: Clustering," on the web retailer's customers. Two

customer attributes—age and total sales since 2013—have been identified as variables of interest for the purposes of the clustering analysis. The customer's age is available in the *customer_demographics* table. The total sales for each customer can be computed from the *orders_recent* table. Because it was decided to include customers who had not purchased anything, a LEFT OUTER JOIN is used to include all customers. The customer's age and sales are stored in an array in the *cust_age_sales* table. The MADlib k-means function expects the coordinates to be expressed as an array.

```

/* create an empty table to store the input for the k-means analysis */
CREATE TABLE cust_age_sales (
    customer_id integer,
    coordinates float8[])

/* prepare the input for the k-means analysis */
INSERT INTO cust_age_sales (customer_id, coordinates[1], coordinates[2])
SELECT d.customer_id,
    d.customer_age,
    CASE
        WHEN s.sales IS NULL THEN 0.0 ELSE
        s.sales
    END
FROM   customer_demographics d
LEFT OUTER JOIN (SELECT r.customer_id,
    SUM(r.item_quantity * r.item_price) AS sales FROM
    orders_recent r
    GROUP BY r.customer_id) s
ON d.customer_id = s.customer_id

/* examine the first 10 rows of the input */
SELECT * from cust_age_sales order by
customer_id
LIMIT 10

customer_id      coordinates
1                {32,14.98}
2                {32,51.48}
3                {33,151.89}
4                {27,88.28}
5                {31,4.85}
6                {26,54}
7                {29,63}
8                {25,101.07}
9                {32,41.05}
10               {32,0}

```

Using the MADlib function, `kmeans_random()`, the following SQL query identifies six clusters within the provided dataset. A description of the key input values is provided with the query.

```
/*
K-means analysis

cust_age_sales - SQL table containing the input data
coordinates - the column in the SQL table that contains the data points
customer_id - the column in the SQL table
that contains the
    identifier for each point
km_coord - the table to store each point and its assigned cluster
km_centers - the SQL table to store the
centers of each cluster
l2norm - specifies that the Euclidean distance formula is used
25 - the maximum
number of iterations

0.001 - a convergence criterion
False(twice) - ignore
some options
6 - build six clusters

*/
```

```
SELECT madlib.kmeans_random('cust_age_sales', 'coordinates',
    'customer_id', 'km_coord', 'km_centers',
    'l2norm', 25, 0.001, False, False, 6)
```

```
SELECT *
FROM     km_coord
ORDER BY pid
LIMIT 10
```

pid	coords	cid
1	{1,1}:{32,14.98}	6
2	{1,1}:{32,51.48}	1
3	{1,1}:{33,151.89}	4
4	{1,1}:{27,88.28}	1
5	{1,1}:{31,4.85}	6
6	{1,1}:{26,54}	1
7	{1,1}:{29,63}	1
8	{1,1}:{25,101.07}	1
9	{1,1}:{32,41.05}	1
10	{1,1}:{32,0}	6

The output consists of the **km_coord** table. This table contains the coordinates for each point id (**pid**), the **customer_id**, and the assigned cluster ID (**cid**). The coordinates (**coords**) are stored as sparse vectors. Sparse vectors are useful when values in an array are repeated many times. For example, {1,200,3}:{1,0,1} represents the following vector containing 204 elements, {1,0,0,...,0,1,1,1}, where the zeroes are repeated 200 times.

The coordinates for each cluster center or centroid are stored in the SQL table **km_center**.

```
SELECT *
FROM     km_centers
```

ORDER BY coords

```
cid coords
6 {1,1}:{44.1131730722154,6.31487804161302}
1 {1,1}:{39.8000419034649,61.6213603286732}
4 {1,1}:{39.2578830823738,167.758556117954}
5 {1,1}:{40.9437092852768,409.846906145043}
3 {1,1}:{42.3521947160391,1150.68858851676}
2 {1,1}:{41.2411873840445,4458.93716141001}
```

Because the age values are similar for each centroid, it appears that the sales values dominated the distance calculations. After visualizing the clusters, it is advisable to repeat the analysis after rescaling, as discussed in Chapter 4.

Summary

This chapter presented several techniques and examples illustrating how SQL can be used to perform in-database analytics. A typical SQL query involves joining several tables, filtering the returned dataset to the desired records with a WHERE clause, and specifying the particular columns of interest. SQL provides the set operations of UNION and UNION ALL to merge the results of two or more SELECT statements or

INTERSECT to find common record elements. Other SQL queries can summarize a dataset using aggregate functions such as COUNT() and SUM() and the GROUP BY clause. Grouping extensions such as the CUBE and ROLLUP operators enable the computation of subtotals and grand totals.

Although SQL is most commonly associated with structured data, SQL tables often contain unstructured data such as comments, descriptions, and other freeform text content. Regular expressions and related functions can be used in SQL to examine and restructure such unstructured data for further analysis.

More complex SQL queries can utilize window functions to supply computed values such as ranks and rolling averages along with an original dataset. In addition to built-in functions, SQL offers the ability to create user-defined functions. Although it is possible to process the data within a database and extract the results into an analytical tool such as R, external libraries such as MADlib can be utilized by SQL to conduct statistical analyses within a database.

Exercises

1. Show that EWMA smoothing is equivalent to an ARIMA(0,1,1) model with no constant, as described in Chapter 8, "Advanced Analytical Theory and Methods: Time Series Analysis."
2. Referring to Equation (11-1), demonstrate that the assigned weights decay exponentially in time.
3. Develop and test a user-defined aggregate to calculate n factorial (n!), where n is an integer.
4. From a SQL table or query, randomly select 10% of the rows. Hint: Most SQL implementations have a random() function that provides a uniform random number between 0 and 1. Discuss possible reasons to randomly sample records from a SQL table.

Bibliography

- [1] PostgreSQL.org, "Window Functions" [Online]. Available: <http://www.postgresql.org/docs/9.3/static/functions-window.html>. [Accessed 10 April 2014].
- [2] MADlib, "MADlib" [Online]. Available: <http://madlib.net/download/>. [Accessed 10 April 2014].
- [3] J. Cohen, B. Dolan, M. Dunlap, J. Hellerstein, and C. Welton, "MAD Skills: New Analysis Practices for Big Data," in *Proceedings of the VLDB Endowment Volume 2 Issue 2, August 2009*.
- [4] E. Dumbill, "The Data Lake Dream," *Forbes*, 14 January 2014. [Online]. Available: <http://www.forbes.com/sites/edddumbill/2014/01/14/the-data-lake-dream/> . [Accessed 4 June 2014].
- [5] MADlib, "MADlib Modules" [Online]. Available: <http://doc.madlib.net/latest/modules.html>. [Accessed 10 April 2014].

12

The Endgame, or Putting It All Together

Key Concepts

Communicating and operationalizing an analytics project

Creating the final deliverables

Using a core set of material for different audiences

Comparing main focus areas for sponsors and analysts

Understanding simple data visualization principles

Cleaning up a chart or visualization

This chapter focuses on the final phase of the Data Analytics Lifecycle: operationalize. In this phase, the project team delivers final reports, code, and technical documentation. At the conclusion of this phase, the team generally attempts to set up a pilot project and implement the developed models from Phase 4 in a production environment. As stated in Chapter 2, “Data Analytics Lifecycle,” teams can perform a technically accurate analysis, but if they cannot translate the results into a language that resonates with their audience, others will not see the value, and significant effort and resources will have been wasted. This chapter focuses on showing how to construct a clear narrative summary of the work and a framework for conveying the narrative to key stakeholders.

12.1 Communicating and Operationalizing an Analytics Project

As shown in Figure 12-1, the final phase in the Data Analytics Lifecycle focuses on operationalizing the project. In this phase, teams need to assess the benefits of the project work and set up a pilot to deploy the models in a controlled way before broadening the work and sharing it with a full enterprise or ecosystem of users. In this context, a pilot project can refer to a project prior to a full-scale rollout of the new algorithms or functionality. Th

products, or servic

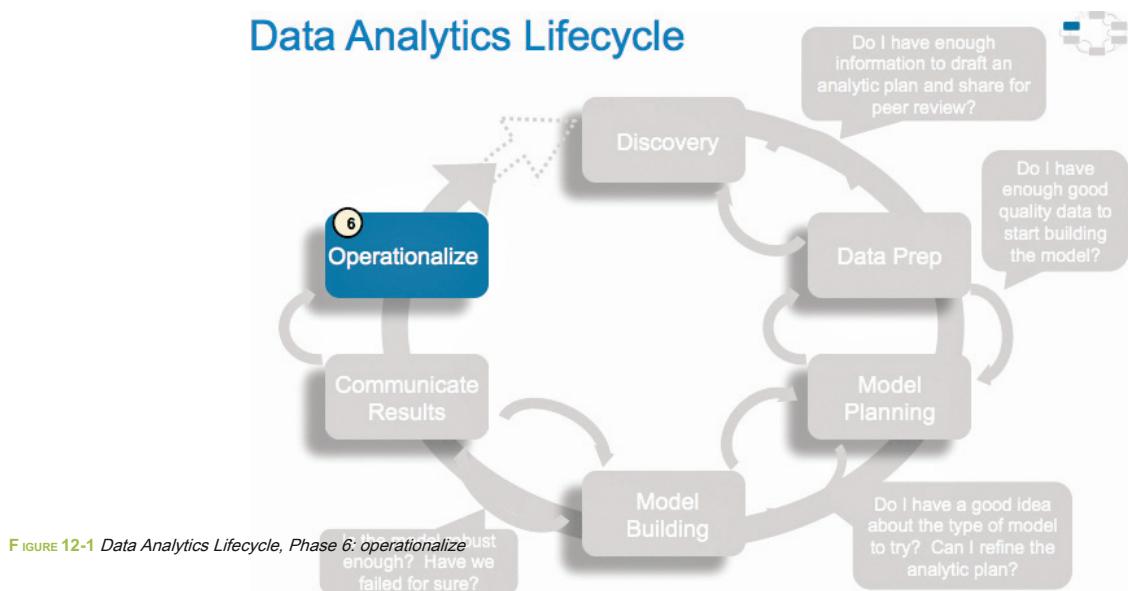


FIGURE 12-1 Data Analytics Lifecycle, Phase 6: operationalize Just enough? Have we failed for sure?

The team's ability to quantify the benefits and share them in a compelling way with the stakeholders will determine if the work will move forward into a pilot project and ultimately be run in a production environment. Therefore, it is critical to identify the benefits and state them in a clear way in the final presentations.

As the team scopes the effort involved to deploy the analytical model as a pilot project, it also needs to consider running the model in a production environment for a discrete set of products or a single line of business, which tests the model in a live setting. This allows the team to learn from the deployment and make adjustments before deploying the application or code more broadly across the enterprise. This phase can bring in a new set of team members—namely, those engineers responsible for the production environment who have a new set of issues and concerns. This group is interested in ensuring that running the model fits smoothly into the production environment and the model can be integrated into downstream processes. While executing the model in the production environment, the team should aim to detect input anomalies before they are fed to the model, assess run times, and gauge competition for resources with other processes in the production environment.

Chapter 2 included an in-depth discussion of the Data Analytics Lifecycle, including an overview of the deliverables produced by each of its major phases:

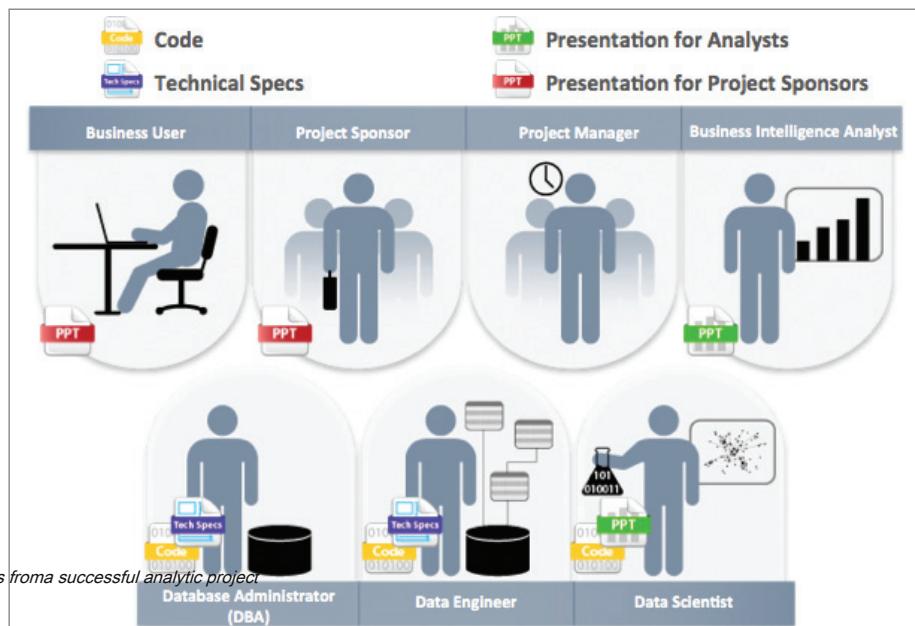


FIGURE 12-2 Key outputs from a successful analytic project

Following is a brief review of the key outputs for each of the main stakeholders of an analytics project and what they usually expect at the conclusion of a project:

- **Business User** typically tries to determine the benefits and implications of the findings to the business.
- **Project Sponsor** typically asks questions related to the business impact of the project, the risks and return on investment (ROI), and how the project can be evangelized within the organization and beyond.

- **Project Manager** needs to determine if the project was completed on time and within budget.
- **Business Intelligence Analyst** needs to know if the reports and dashboards hermanages will be impacted and need to change.
- **Data Engineer and Database Administrator (DBA)** typically need to share the code from the analytical project and create technical documents that describe how to implement the code.
- **Data Scientists** need to share the code and explain the model to their peers, managers, and other stakeholders.

Although these seven roles represent many interests within a project, these interests usually overlap, and most of them can be met with four main deliverables:

- **Presentation for Project Sponsors** contains high-level takeaways for executive-level stakeholders, with a few key messages to aid their decision-making process. Focus on clean, easy visuals for the presenter to explain and for the viewer to grasp.
- **Presentation for Analysts**, which describes changes to business processes and reports. Data scientists reading this presentation are comfortable with technical graphs (such as Receiver Operating Characteristic [ROC] curves, density plots, and histograms) and will be interested in the details.
- **Code** for technical people, such as engineers and others managing the production environment
- **Technical specifications** for implementing the code

As a rule, the more executive the audience, the more succinct the presentation needs to be for project sponsors. Ensure that the presentation gets to the point quickly and frames the results in terms of value to the sponsor's organization. When presenting to other audiences with more quantitative backgrounds, focus more time on the methodology and findings. In these instances, the team can be more expansive in describing the outcomes, methodology, and analytical experiments with a peer group. This audience will be more interested in the techniques, especially if the team developed a new way of processing or analyzing data that can be reused in the future or applied to similar problems. In addition, use imagery or data visualization when possible. Although it may take more time to develop imagery, pictures are more appealing, easier to remember, and more effective to deliver key messages than long lists of bullets.

12.2 Creating the Final Deliverables

After reviewing the list of key stakeholders for data science projects and main deliverables, this section focuses on describing the deliverables in detail. To illustrate this approach, a fictional case study is used to make the examples more specific. Figure 12-3 describes a scenario of a fictional bank, YoyoDyne Bank, which would like to embark on a project to do churn prediction models of its customers. *Churn rate* in this context refers to the frequency with which customers sever their relationship as customers of YoyoDyne Bank or switch to a competing bank.