Capítulo 1
Introducción a bases de datos relacionales y SQL

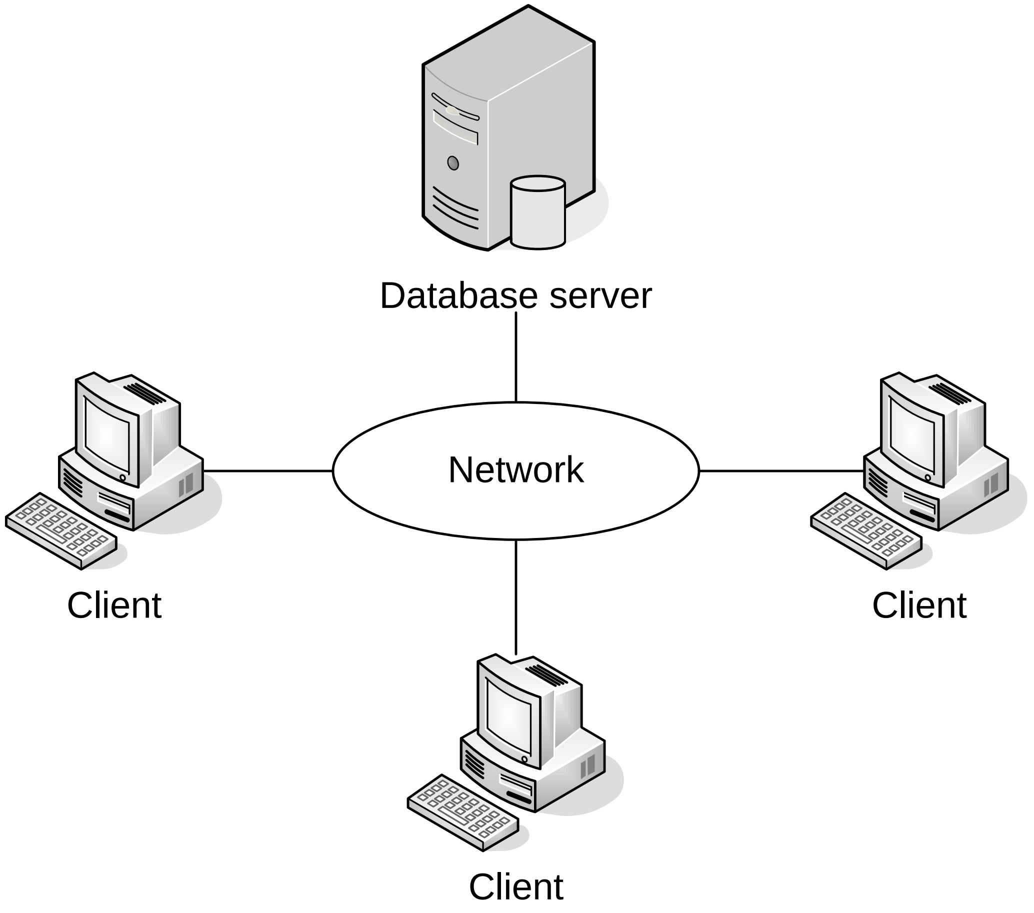
Objetivos
Conocimiento

- Identificar los tres componentes principales de hardware de un sistema cliente/servidor.
- Describa la forma en que un cliente accede a la base de datos en un servidor utilizando estos términos: software de aplicación, API de acceso a datos, sistema de administración de bases de datos, consulta SQL y resultados de consultas.
- Describa la forma en que se organiza una base de datos utilizando estos términos: tablas, columnas, filas y celdas.
- Describa cómo se relacionan las tablas de una base de datos relacional utilizando estos términos: clave principal y clave externa.
- Identifique los tres tipos de relaciones que pueden existir entre dos tablas.
- Describa la forma en que se definen las columnas de una tabla utilizando estos términos: tipo de datos, valor nulo, valor predeterminado y columna de identidad.

Objetivos (cont.)

- Describir la relación entre SQL estándar y Transact-SQL de Microsoft SQL Server.
- Describa la diferencia entre las instrucciones DML y las instrucciones DDL.
- Describa la diferencia entre una consulta de acción y una consulta SELECT.
- Enumere tres técnicas de codificación que pueden hacer que su código SQL sea más fácil de leer y mantener.
- Explique en qué se diferencian las vistas y los procedimientos almacenados de las sentencias SQL que se emiten desde un programa de aplicación.
- Describir el uso de objetos de comando, conexión y lector de datos cuando las aplicaciones .NET tienen acceso a una base de datos de SQL Server.

Un sistema cliente/servidor sencillo



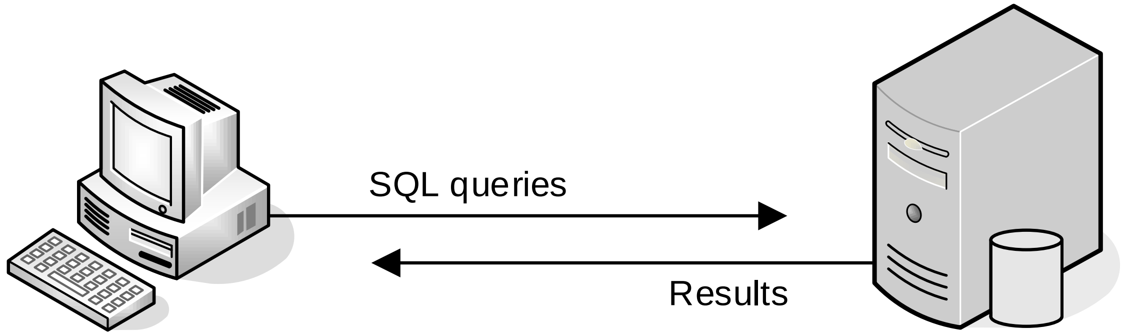
Los tres componentes de hardware de un sistema cliente/servidor

- Clientes
- Servidor
- Palabra de red

Términos que debes conocer

- Red de área local (LAN)
- Red de área extensa (WAN)
- Sistema empresarial

Software de cliente, software de servidor y la interfaz SQL



Client

Application software
Data access API

Database server

Database management system
Database

Software de servidor

- Sistema de gestión de bases de datos (DBMS)
- El DBMS realiza el procesamiento back-end

Software de cliente

- Aplicación
- API de acceso a datos (interfaz de programación de aplicaciones)
- El software cliente realiza el procesamiento front-end

La interfaz SQL

- El software de la aplicación se comunica con el DBMS mediante el envío de consultas SQL a través de la API de acceso a datos.
- Cuando el DBMS recibe una consulta, proporciona un servicio como devolver los datos solicitados (los resultados de la consulta) al cliente.
- SQL son las siglas de Structured Query Language, que es el lenguaje estándar para trabajar con una base de datos relacional.

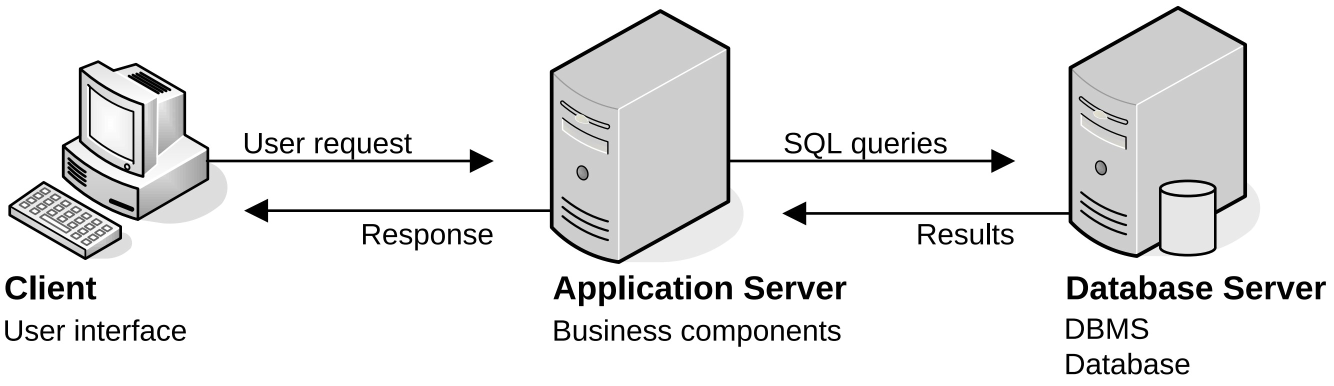
Sistema cliente/servidor

- El procesamiento se divide entre el cliente y el servidor

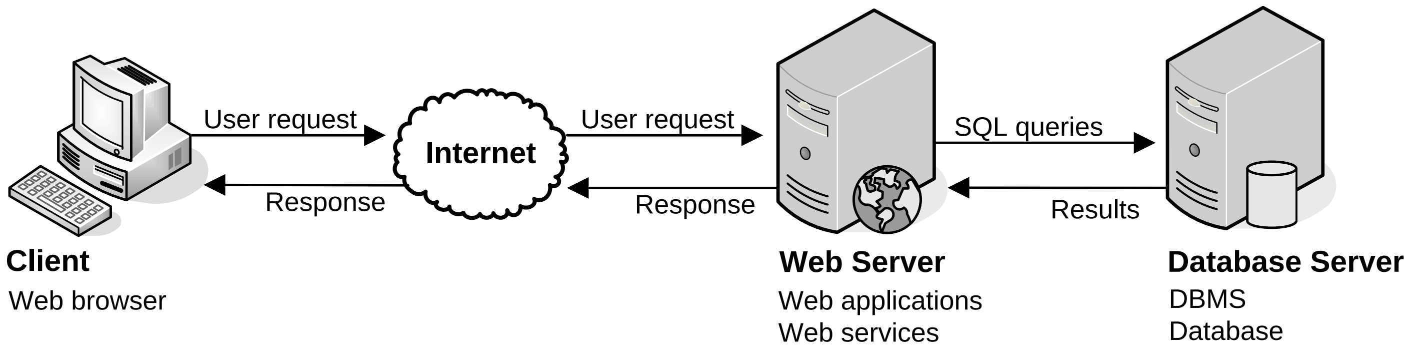
Sistema de manejo de archivos

- Todo el procesamiento se realiza en los clientes

Un sistema basado en Windows que utiliza un servidor de aplicaciones



Un sistema sencillo basado en la web



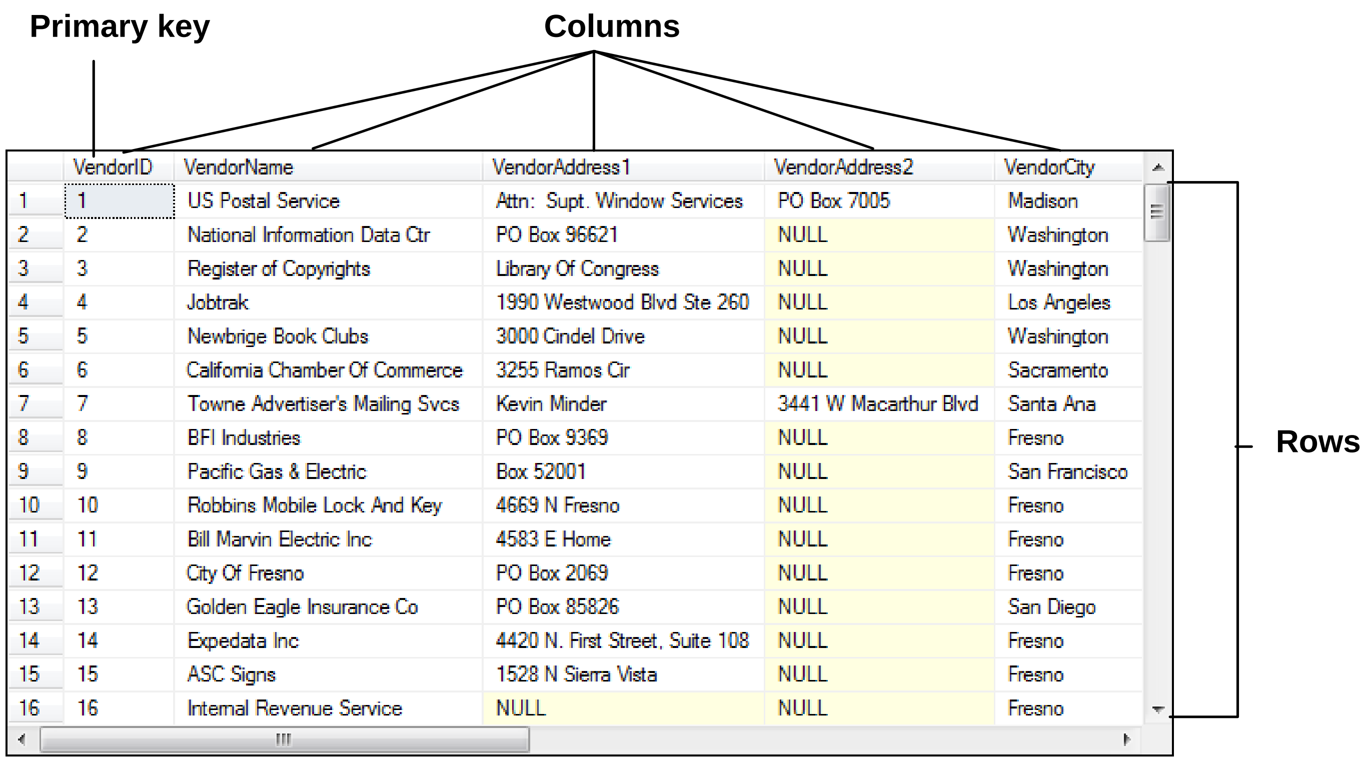
Otras arquitecturas de sistemas cliente/servidor

- Los servidores de aplicaciones almacenan componentes empresariales
- Los servidores web almacenan aplicaciones web y servicios web

Cómo funcionan las aplicaciones web

- Un explorador web de un cliente envía una solicitud a un servidor web.
- El servidor web procesa la solicitud.
- El servidor web pasa las solicitudes de datos al servidor de bases de datos.
- El servidor de bases de datos devuelve los resultados al servidor web.
- El servidor web devuelve una respuesta al navegador.

La tabla Proveedores en una base de datos de proveedores (AP)

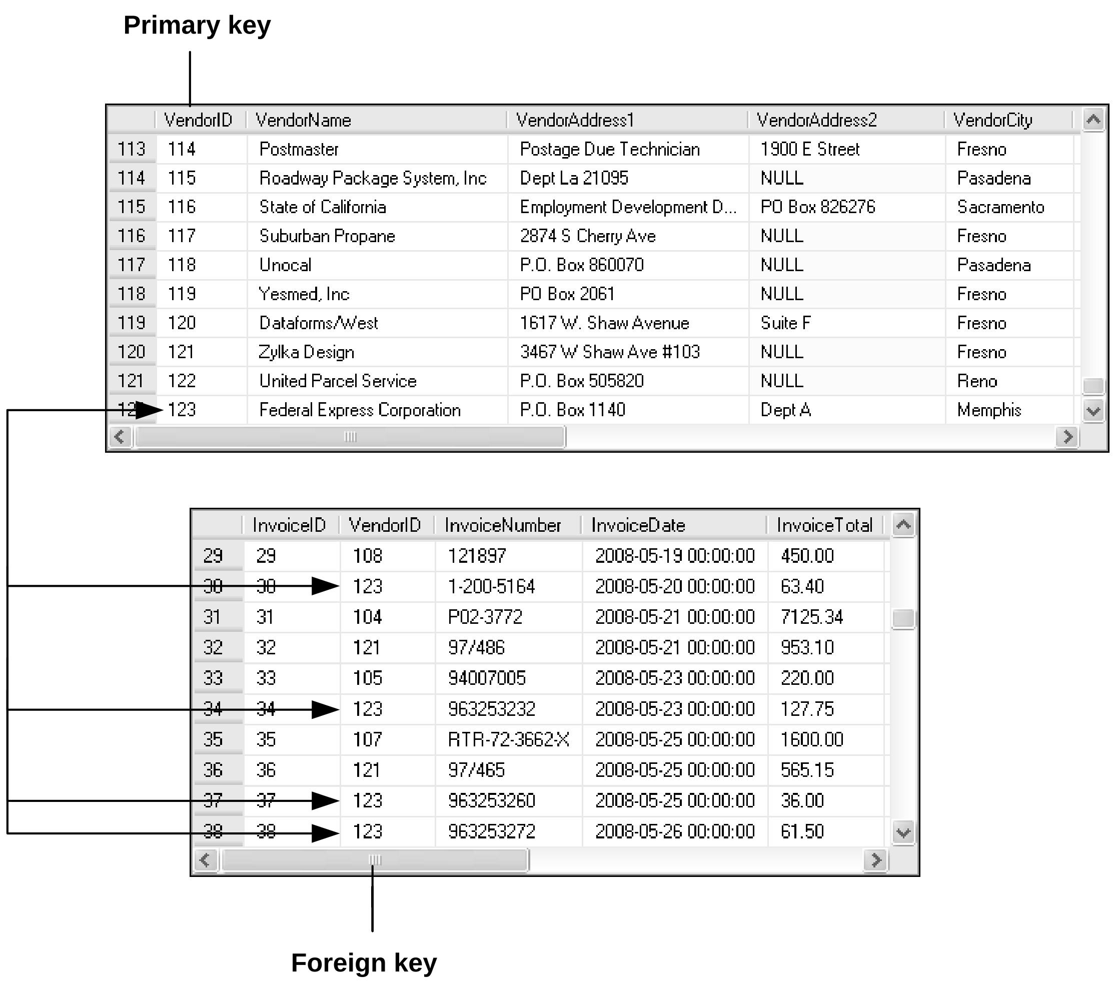


	VendorID	VendorName	VendorAddress1	VendorAddress2	VendorCity
1	1	US Postal Service	Attn: Supt. Window Services	PO Box 7005	Madison
2	2	National Information Data Ctr	PO Box 96621	NULL	Washington
3	3	Register of Copyrights	Library Of Congress	NULL	Washington
4	4	Jobtrak	1990 Westwood Blvd Ste 260	NULL	Los Angeles
5	5	Newbrige Book Clubs	3000 Cindel Drive	NULL	Washington
6	6	California Chamber Of Commerce	3255 Ramos Cir	NULL	Sacramento
7	7	Towne Advertiser's Mailing Svcs	Kevin Minder	3441 W Macarthur Blvd	Santa Ana
8	8	BFI Industries	PO Box 9369	NULL	Fresno
9	9	Pacific Gas & Electric	Box 52001	NULL	San Francisco
10	10	Robbins Mobile Lock And Key	4669 N Fresno	NULL	Fresno
11	11	Bill Marvin Electric Inc	4583 E Home	NULL	Fresno
12	12	City Of Fresno	PO Box 2069	NULL	Fresno
13	13	Golden Eagle Insurance Co	PO Box 85826	NULL	San Diego
14	14	Expedata Inc	4420 N. First Street, Suite 108	NULL	Fresno
15	15	ASC Signs	1528 N Sierra Vista	NULL	Fresno
16	16	Internal Revenue Service	NULL	NULL	Fresno

Terms

- Base de datos relacional
- Mesa
- Columna
- Fila
- Celda
- Clave principal
- Clave principal compuesta
- Clave no principal (clave única)
- Índice

La relación entre dos tablas



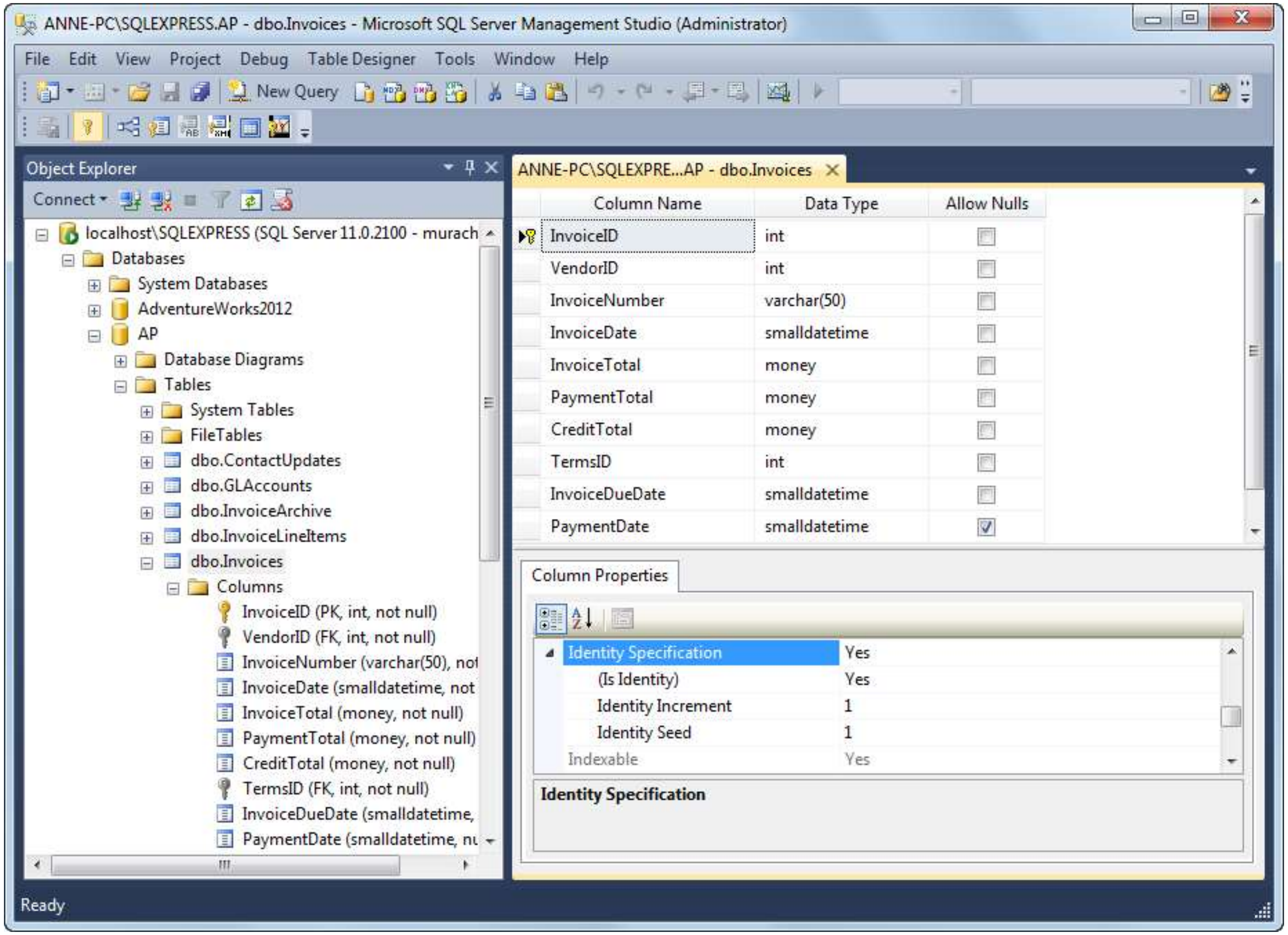
	VendorID	VendorName	VendorAddress1	VendorAddress2	VendorCity
113	114	Postmaster	Postage Due Technician	1900 E Street	Fresno
114	115	Roadway Package System, Inc	Dept La 21095	NULL	Pasadena
115	116	State of California	Employment Development D...	PO Box 826276	Sacramento
116	117	Suburban Propane	2874 S Cherry Ave	NULL	Fresno
117	118	Unocal	P.O. Box 860070	NULL	Pasadena
118	119	Yesmed, Inc	PO Box 2061	NULL	Fresno
119	120	Dataforms/West	1617 W. Shaw Avenue	Suite F	Fresno
120	121	Zylka Design	3467 W Shaw Ave #103	NULL	Fresno
121	122	United Parcel Service	P.O. Box 505820	NULL	Reno
122	123	Federal Express Corporation	P.O. Box 1140	Dept A	Memphis

	InvoiceID	VendorID	InvoiceNumber	InvoiceDate	InvoiceTotal
29	29	108	121897	2008-05-19 00:00:00	450.00
30	30	123	1-200-5164	2008-05-20 00:00:00	63.40
31	31	104	P02-3772	2008-05-21 00:00:00	7125.34
32	32	121	97/486	2008-05-21 00:00:00	953.10
33	33	105	94007005	2008-05-23 00:00:00	220.00
34	34	123	963253232	2008-05-23 00:00:00	127.75
35	35	107	RTR-72-3662-X	2008-05-25 00:00:00	1600.00
36	36	121	97/465	2008-05-25 00:00:00	565.15
37	37	123	963253260	2008-05-25 00:00:00	36.00
38	38	123	963253272	2008-05-26 00:00:00	61.50

Terms

- Clave foránea
- Relación de uno a varios
- Relación uno a uno
- Relación de varios a varios

Las columnas de la tabla Facturas


ANNE-PC\SQLEXPRESS.AP - dbo.Invoices - Microsoft SQL Server Management Studio (Administrator)

localhost\SQLEXPRESS (SQL Server 11.0.2100 - murach
- Databases
  - System Databases
  - AdventureWorks2012
  - AP
    - Database Diagrams
    - Tables
      - System Tables
      - FileTables
      - dbo.ContactUpdates
      - dbo.GLAccounts
      - dbo.InvoiceArchive
      - dbo.InvoiceLineItems
      - dbo.Invoices
        - Columns
          - InvoiceID (PK, int, not null)
          - VendorID (FK, int, not null)
          - InvoiceNumber (varchar(50), not
          - InvoiceDate (smalldatetime, not
          - InvoiceTotal (money, not null)
          - PaymentTotal (money, not null)
          - CreditTotal (money, not null)
          - TermsID (FK, int, not null)
          - InvoiceDueDate (smalldatetime,
          - PaymentDate (smalldatetime, nu

ANNE-PC\SQLEXPRE...AP - dbo.Invoices

Column Name	Data Type	Allow Nulls
InvoiceID	int	☐
VendorID	int	☐
InvoiceNumber	varchar(50)	☐
InvoiceDate	smalldatetime	☐
InvoiceTotal	money	☐
PaymentTotal	money	☐
CreditTotal	money	☐
TermsID	int	☐
InvoiceDueDate	smalldatetime	☐
PaymentDate	smalldatetime	✓

Column Properties

Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1
Indexable	Yes

Identity Specification

Common SQL Server data types

- bit
- int, bigint, smallint, tinyint
- money, smallmoney
- decimal, numeric
- float, real
- datetime, smalldatetime
- char, varchar
- nchar, nvarchar

Terms

- Data type
- Null value
- Default value
- Identity column

A comparison of relational databases and conventional file systems

Feature	File system	Relational database
Definition	Each program must define the file and the layout of the records within the file	Tables, rows, and columns are defined within the database and can be accessed by name
Maintenance	If the definition of a file changes, each program that uses the file must be modified	Programs can be used without modification when the definition of a table changes
Validity checking	Each program that updates a file must include code to check for valid data	Can include checks for valid data

A comparison of relational databases and conventional file systems (continued)

Feature	File system	Relational database
Relationships	Each program must provide for and enforce relationships between files	Can enforce relationships between tables using foreign keys; ad hoc relationships can also be used
Data access	Each I/O operation targets a specific record based on its relative position in the file or its key value	A program can use SQL to access selected data in one or more tables of a database

A comparison of relational databases and other database systems

Feature	Hierarchical database	Network database	Relational database
Supported relationships	One-to-many only	One-to-many, one-to-one, and many-to-many	One-to-many, one-to-one, and many-to-many; ad hoc relationships can also be used
Data access	Programs must include code to navigate through the physical structure of the database	Programs must include code to navigate through the physical structure of the database	Programs can access data without knowing its physical structure

A comparison of relational databases and other database systems (continued)

Feature	Hierarchical database	Network database	Relational database
Maintenance	New and modified relationships can be difficult to implement in application programs	New and modified relationships can be difficult to implement in application programs	Programs can be used without modification when the definition of a table changes

Important events in the history of SQL

Year	Event
1970	Dr. E. F. Codd developed the relational database model.
1978	IBM developed the predecessor to SQL, called Structured English Query Language (SEQUEL).
1979	Relational Software, Inc. (later renamed Oracle) released the first relational DBMS, Oracle.
1982	IBM released their first relational database system, SQL/DS (SQL/Data System).
1985	IBM released DB2 (Database 2).
1987	Microsoft released SQL Server.
1989	ANSI published the first set of standards (ANSI/ISO SQL-89, or SQL1).
1992	ANSI revised standards (ANSI/ISO SQL-92, or SQL2).
1999	ANSI published SQL3 (ANSI/ISO SQL:1999).

Important events in the history of SQL (continued)

Year	Event
2003	ANSI published SQL:2003.
2006	ANSI published SQL:2006.
2008	ANSI published SQL:2008.
2011	Information on these standards is not yet freely available.

How knowing “standard SQL” helps you

- Basic SQL statements are the same for all SQL *dialects*.
- Once you know one SQL dialect, you can easily learn others.

How knowing “standard SQL” does not help you

- Any non-trivial application will require modification when moved from one SQL database to another.

First database releases

Oracle	1979
DB2	1985
MySQL	2000
SQL Server	1987

Primary platforms

Oracle	Unix OS/390 and z/OS
DB2	OS/390 and z/OS Unix
MySQL	Unix Windows Mac OS
SQL Server	Windows

SQL DML statements

- SELECT
- INSERT
- UPDATE
- DELETE

SQL DDL statements

- CREATE DATABASE, TABLE, INDEX
- ALTER TABLE, INDEX
- DROP DATABASE, TABLE, INDEX

A statement that creates a new database

```
CREATE DATABASE AP;
```

A statement that creates a new table

```
CREATE TABLE Invoices
(InvoiceID      INT           NOT NULL IDENTITY
                PRIMARY KEY,
VendorID        INT           NOT NULL
                REFERENCES Vendors(VendorID),
InvoiceNumber   VARCHAR(50)   NOT NULL,
InvoiceDate     SMALLDATETIME NOT NULL,
InvoiceTotal    MONEY         NOT NULL,
PaymentTotal    MONEY         NOT NULL DEFAULT 0,
CreditTotal     MONEY         NOT NULL DEFAULT 0,
TermsID         INT           NOT NULL
                REFERENCES Terms(TermsID),
InvoiceDueDate  SMALLDATETIME NOT NULL,
PaymentDate     SMALLDATETIME NULL);
```

A statement that adds a new column to the table

```
ALTER TABLE Invoices
ADD BalanceDue MONEY NOT NULL;
```

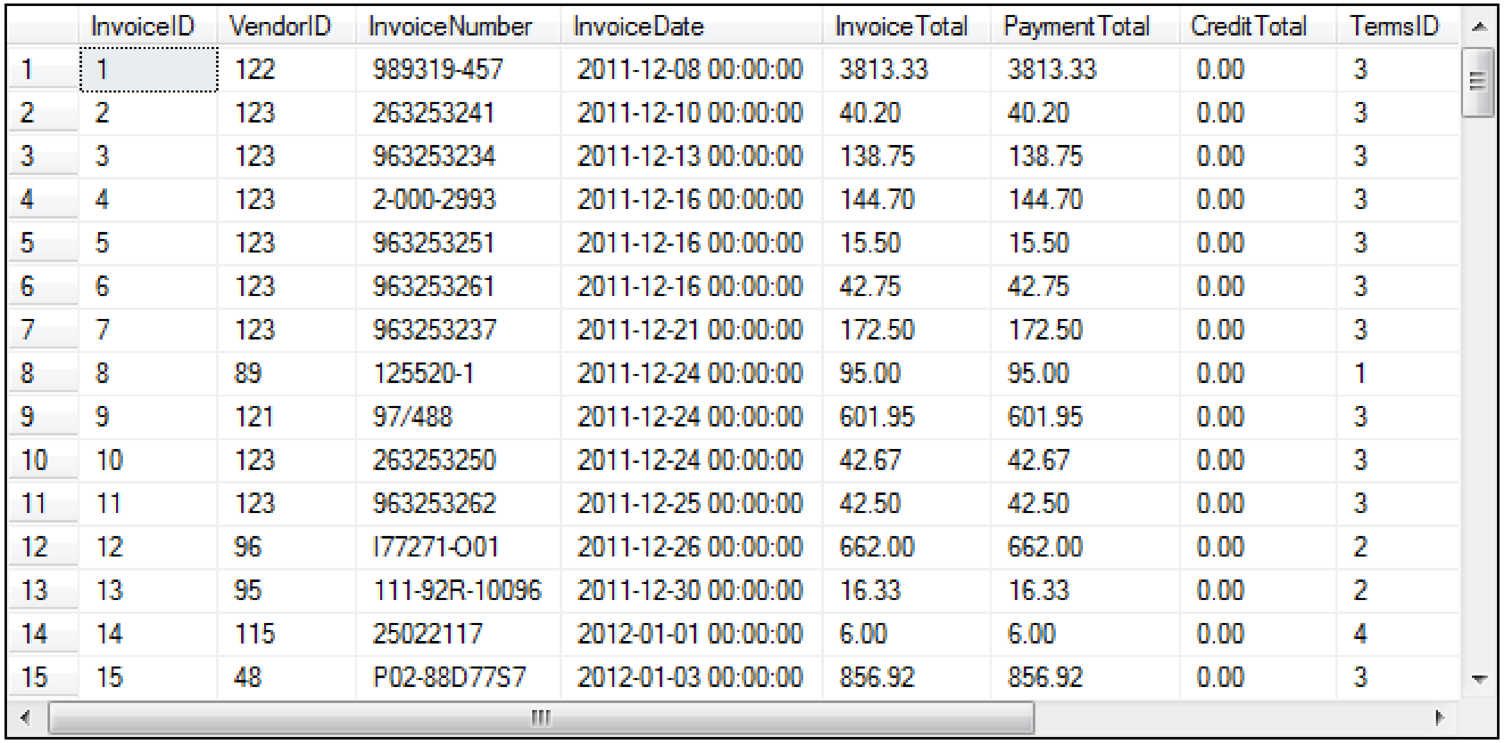
A statement that deletes the new column

```
ALTER TABLE Invoices
DROP COLUMN BalanceDue;
```

A statement that creates an index on the table

```
CREATE INDEX IX_Invoices_VendorID
    ON Invoices (VendorID);
```

The Invoices base table

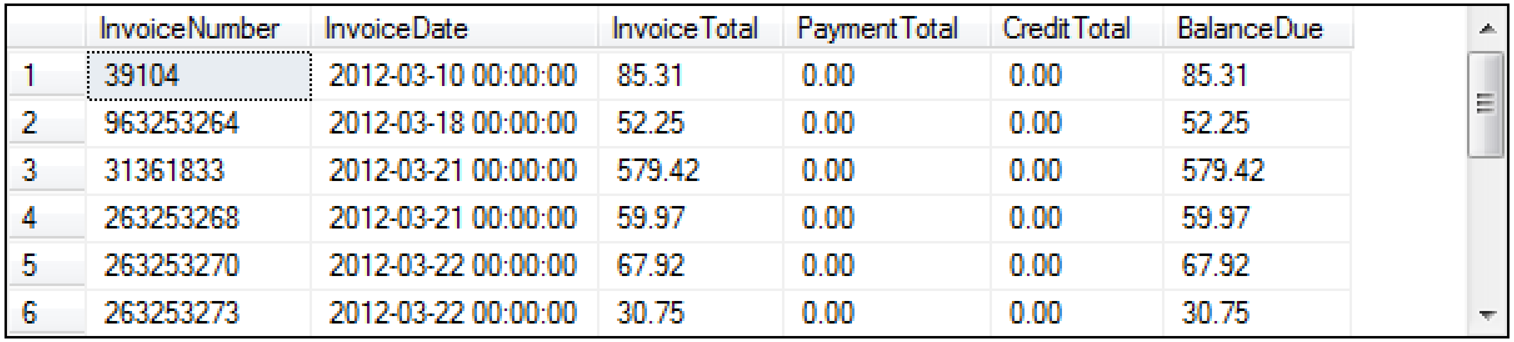


	InvoiceID	VendorID	InvoiceNumber	InvoiceDate	InvoiceTotal	PaymentTotal	CreditTotal	TermsID
1	1	122	989319-457	2011-12-08 00:00:00	3813.33	3813.33	0.00	3
2	2	123	263253241	2011-12-10 00:00:00	40.20	40.20	0.00	3
3	3	123	963253234	2011-12-13 00:00:00	138.75	138.75	0.00	3
4	4	123	2-000-2993	2011-12-16 00:00:00	144.70	144.70	0.00	3
5	5	123	963253251	2011-12-16 00:00:00	15.50	15.50	0.00	3
6	6	123	963253261	2011-12-16 00:00:00	42.75	42.75	0.00	3
7	7	123	963253237	2011-12-21 00:00:00	172.50	172.50	0.00	3
8	8	89	125520-1	2011-12-24 00:00:00	95.00	95.00	0.00	1
9	9	121	97/488	2011-12-24 00:00:00	601.95	601.95	0.00	3
10	10	123	263253250	2011-12-24 00:00:00	42.67	42.67	0.00	3
11	11	123	963253262	2011-12-25 00:00:00	42.50	42.50	0.00	3
12	12	96	I77271-O01	2011-12-26 00:00:00	662.00	662.00	0.00	2
13	13	95	111-92R-10096	2011-12-30 00:00:00	16.33	16.33	0.00	2
14	14	115	25022117	2012-01-01 00:00:00	6.00	6.00	0.00	4
15	15	48	P02-88D77S7	2012-01-03 00:00:00	856.92	856.92	0.00	3

A SELECT statement that retrieves and sorts selected columns and rows

```
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal,
    PaymentTotal, CreditTotal,
    InvoiceTotal - PaymentTotal - CreditTotal
    AS BalanceDue
FROM Invoices
WHERE InvoiceTotal - PaymentTotal - CreditTotal > 0
ORDER BY InvoiceDate;
```

The result set defined by the SELECT statement

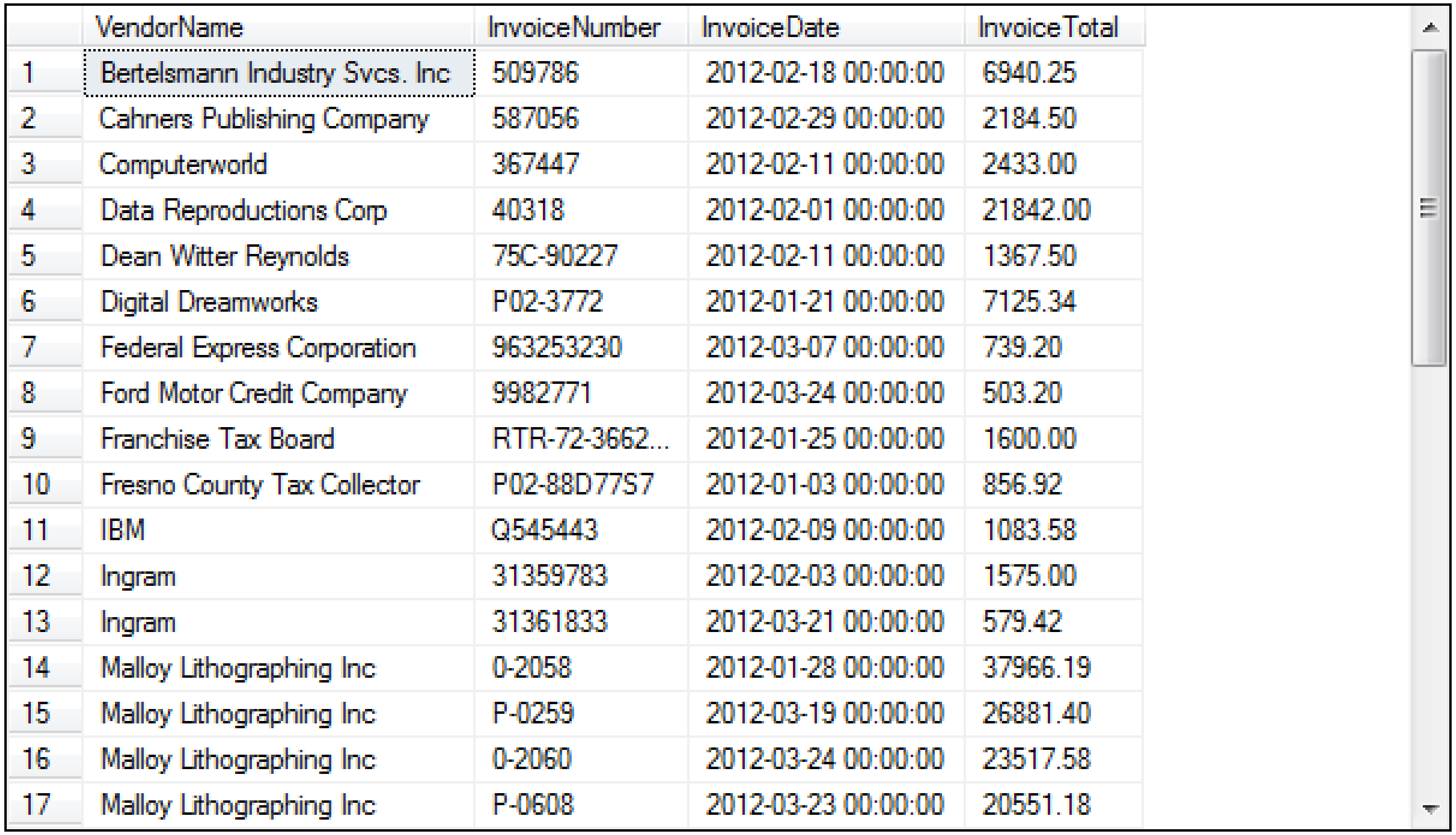


	InvoiceNumber	InvoiceDate	InvoiceTotal	PaymentTotal	CreditTotal	BalanceDue
1	39104	2012-03-10 00:00:00	85.31	0.00	0.00	85.31
2	963253264	2012-03-18 00:00:00	52.25	0.00	0.00	52.25
3	31361833	2012-03-21 00:00:00	579.42	0.00	0.00	579.42
4	263253268	2012-03-21 00:00:00	59.97	0.00	0.00	59.97
5	263253270	2012-03-22 00:00:00	67.92	0.00	0.00	67.92
6	263253273	2012-03-22 00:00:00	30.75	0.00	0.00	30.75

A SELECT statement that joins data from the Vendors and Invoices tables

```
SELECT VendorName, InvoiceNumber, InvoiceDate,
       InvoiceTotal
FROM Vendors INNER JOIN Invoices
    ON Vendors.VendorID = Invoices.VendorID
WHERE InvoiceTotal >= 500
ORDER BY VendorName, InvoiceTotal DESC;
```

The result set defined by the SELECT statement



	VendorName	InvoiceNumber	InvoiceDate	InvoiceTotal
1	Bertelsmann Industry Svcs. Inc	509786	2012-02-18 00:00:00	6940.25
2	Cahners Publishing Company	587056	2012-02-29 00:00:00	2184.50
3	Computerworld	367447	2012-02-11 00:00:00	2433.00
4	Data Reproductions Corp	40318	2012-02-01 00:00:00	21842.00
5	Dean Witter Reynolds	75C-90227	2012-02-11 00:00:00	1367.50
6	Digital Dreamworks	P02-3772	2012-01-21 00:00:00	7125.34
7	Federal Express Corporation	963253230	2012-03-07 00:00:00	739.20
8	Ford Motor Credit Company	9982771	2012-03-24 00:00:00	503.20
9	Franchise Tax Board	RTR-72-3662...	2012-01-25 00:00:00	1600.00
10	Fresno County Tax Collector	P02-88D77S7	2012-01-03 00:00:00	856.92
11	IBM	Q545443	2012-02-09 00:00:00	1083.58
12	Ingram	31359783	2012-02-03 00:00:00	1575.00
13	Ingram	31361833	2012-03-21 00:00:00	579.42
14	Malloy Lithographing Inc	0-2058	2012-01-28 00:00:00	37966.19
15	Malloy Lithographing Inc	P-0259	2012-03-19 00:00:00	26881.40
16	Malloy Lithographing Inc	0-2060	2012-03-24 00:00:00	23517.58
17	Malloy Lithographing Inc	P-0608	2012-03-23 00:00:00	20551.18

Terms

- Base table
- Result set
- Calculated value
- Query
- Join
- Inner join
- Outer join

Add a row to the Invoices table

```
INSERT INTO Invoices (VendorID, InvoiceNumber, InvoiceDate,
    InvoiceTotal, TermsID, InvoiceDueDate)
VALUES (12, '3289175', '4/18/2012', 165, 3, '5/18/2012');
```

Change the value of a column for a selected row

```
UPDATE Invoices
SET CreditTotal = 35.89
WHERE InvoiceNumber = '367447';
```

Change the value in a column for all rows that satisfy the search condition

```
UPDATE Invoices
SET InvoiceDueDate = InvoiceDueDate + 30
WHERE TermsID = 4;
```

Delete a selected invoice from the Invoices table

```
DELETE FROM Invoices
WHERE InvoiceNumber = '4-342-8069';
```

Delete all paid invoices from the Invoices table

```
DELETE FROM Invoices
WHERE InvoiceTotal – PaymentTotal – CreditTotal = 0;
```

A SELECT statement that's difficult to read

```
select invoicenumber, invoicedate, invoicetotal,
invoicetotal - paymenttotal - credittotal as balancedue
from invoices where invoicetotal - paymenttotal -
credittotal > 0 order by invoicedate
```

A SELECT statement that's coded with a readable style

```
Select InvoiceNumber, InvoiceDate, InvoiceTotal,
    InvoiceTotal - PaymentTotal - CreditTotal
    As BalanceDue
From Invoices
Where InvoiceTotal - PaymentTotal - CreditTotal > 0
Order By InvoiceDate;
```

A SELECT statement with a block comment

```
/*
Author: Bryan Syverson
Date: 8/22/12
*/
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal,
    InvoiceTotal - PaymentTotal - CreditTotal
    AS BalanceDue
FROM Invoices;
```

A SELECT statement with a single-line comment

```
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal,
    InvoiceTotal - PaymentTotal - CreditTotal
    AS BalanceDue
    -- The fourth column calculates the balance due
FROM Invoices;
```

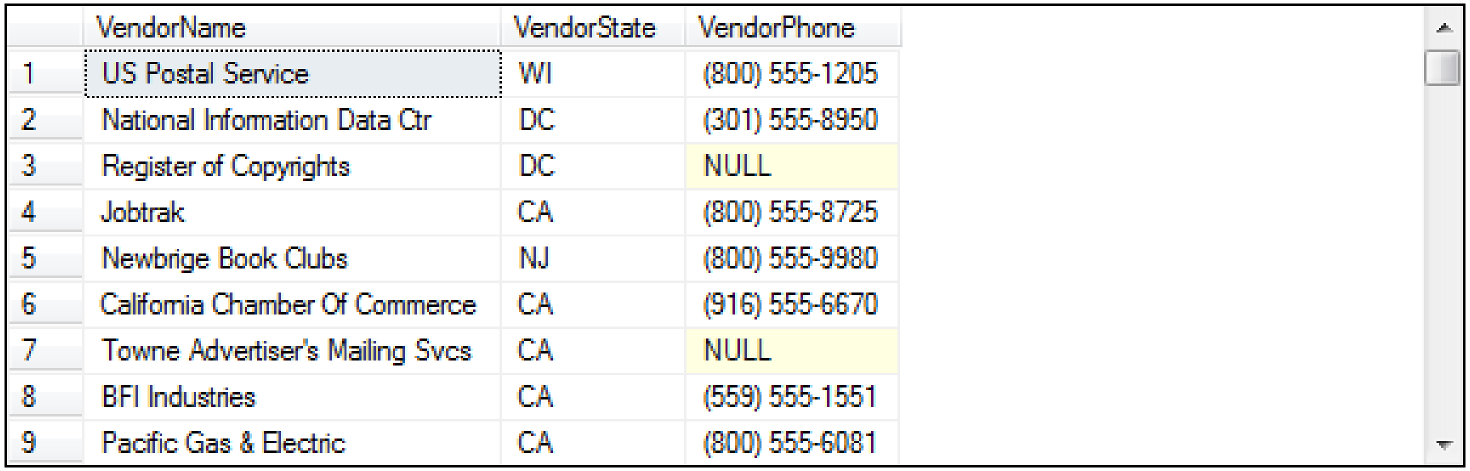
Recomendaciones de codificación SQL

- Comience cada nueva cláusula en una nueva línea.
- Divida las cláusulas largas en varias líneas y aplique sangría a las líneas continuas.
- Escriba en mayúscula la primera letra de cada palabra clave y cada palabra en los nombres de columnas y tablas.
- Termine cada instrucción con un punto y coma (;).
- Use comentarios solo para las partes del código que son difíciles de entender.

A CREATE VIEW statement for a view named VendorsMin

```
CREATE VIEW VendorsMin AS
     SELECT VendorName, VendorState, VendorPhone
     FROM Vendors;
```

The virtual table that's represented by the view

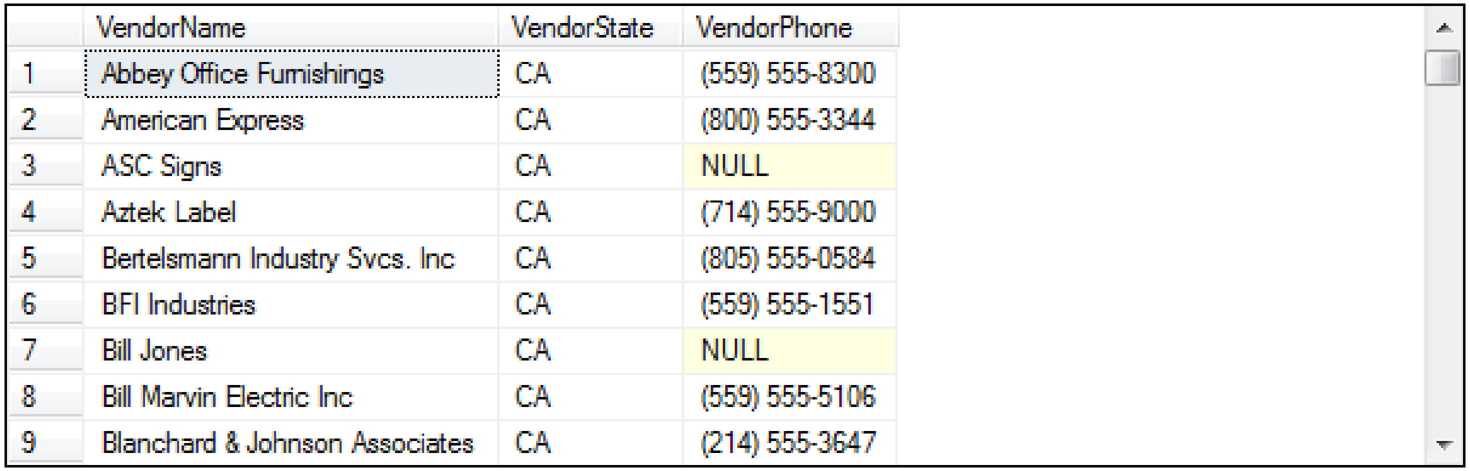


	VendorName	VendorState	VendorPhone
1	US Postal Service	WI	(800) 555-1205
2	National Information Data Ctr	DC	(301) 555-8950
3	Register of Copyrights	DC	NULL
4	Jobtrak	CA	(800) 555-8725
5	Newbrige Book Clubs	NJ	(800) 555-9980
6	California Chamber Of Commerce	CA	(916) 555-6670
7	Towne Advertiser's Mailing Svcs	CA	NULL
8	BFI Industries	CA	(559) 555-1551
9	Pacific Gas & Electric	CA	(800) 555-6081

A SELECT statement that uses the VendorsMin view

```
SELECT * FROM VendorsMin
WHERE VendorState = 'CA'
ORDER BY VendorName;
```

The result set that's returned by the SELECT statement



	VendorName	VendorState	VendorPhone
1	Abbey Office Furnishings	CA	(559) 555-8300
2	American Express	CA	(800) 555-3344
3	ASC Signs	CA	NULL
4	Aztek Label	CA	(714) 555-9000
5	Bertelsmann Industry Svcs. Inc	CA	(805) 555-0584
6	BFI Industries	CA	(559) 555-1551
7	Bill Jones	CA	NULL
8	Bill Marvin Electric Inc	CA	(559) 555-5106
9	Blanchard & Johnson Associates	CA	(214) 555-3647

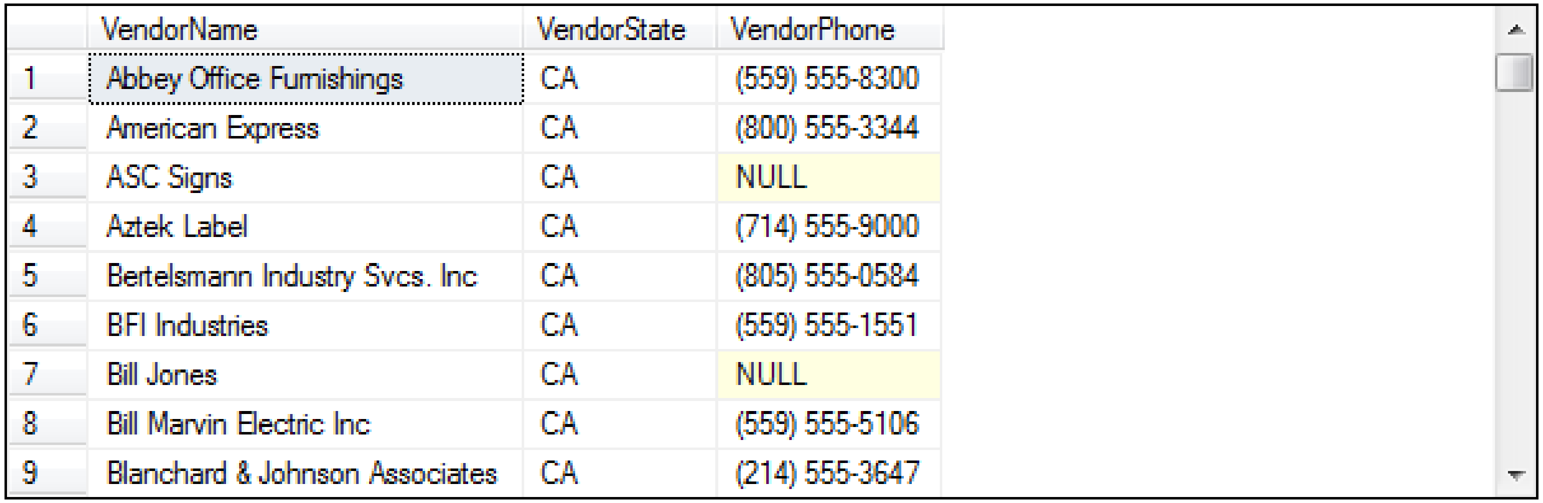
A CREATE PROCEDURE statement for a procedure named spVendorsByState

```
CREATE PROCEDURE spVendorsByState @State char(2) AS
    SELECT VendorName, VendorState, VendorPhone
    FROM Vendors
    WHERE VendorState = @State
    ORDER BY VendorName;
```

Instrucción que ejecuta el procedimiento almacenado spVendorsByState

```
EXEC spVendorsByState 'CA';
```

El conjunto de resultados

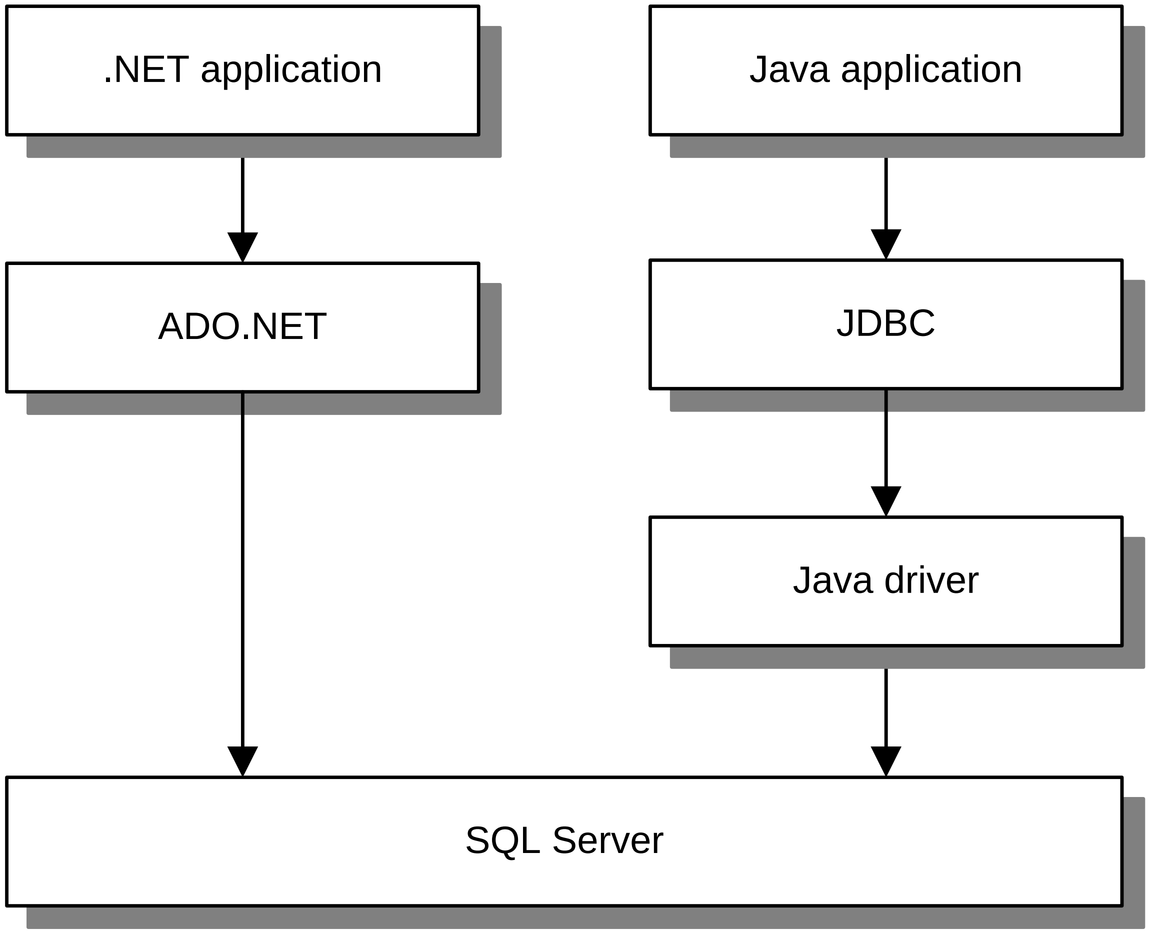


	VendorName	VendorState	VendorPhone
1	Abbey Office Furnishings	CA	(559) 555-8300
2	American Express	CA	(800) 555-3344
3	ASC Signs	CA	NULL
4	Aztek Label	CA	(714) 555-9000
5	Bertelsmann Industry Svcs. Inc	CA	(805) 555-0584
6	BFI Industries	CA	(559) 555-1551
7	Bill Jones	CA	NULL
8	Bill Marvin Electric Inc	CA	(559) 555-5106
9	Blanchard & Johnson Associates	CA	(214) 555-3647

Letra chica

- Procedimiento almacenado
- Lenguaje de control de flujo
- Detonante
- Función definida por el usuario (UDF)

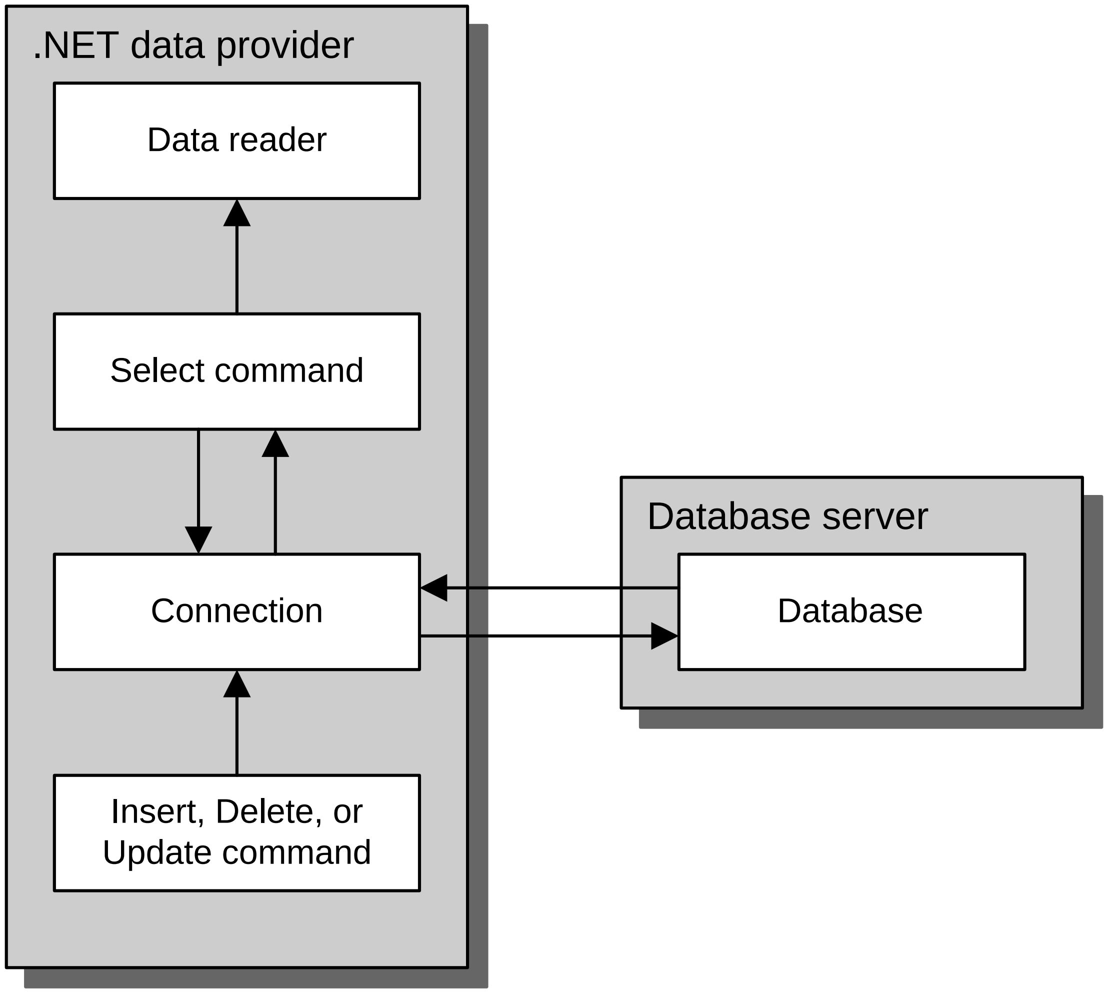
Opciones comunes para acceder a los datos de SQL Server


$$\begin{tikzcd}
\boxed{\text{.NET application}} \arrow[d] & \boxed{\text{Java application}} \arrow[d] \\
\boxed{\text{ADO.NET}} \arrow[dd] & \boxed{\text{JDBC}} \arrow[d] \\
& \boxed{\text{Java driver}} \arrow[d] \\
\boxed{\text{SQL Server}} & \boxed{\text{SQL Server}}
\end{tikzcd}$$

Letra chica

- Modelo de acceso a datos
- ADO.NET (para lenguajes .NET)
- JDBC (para Java)
- Controlador de base de datos

Basic ADO.NET objects in a .NET application



Letra chica

- Proveedor de datos .NET
- Objeto de comando
- Objeto de conexión
- Objeto lector de datos
- Arquitectura de datos desconectada

A Visual Basic function that uses ADO.NET to retrieve data from SQL Server

```
Public Shared Function GetVendor(
        vendorID As Integer) As Vendor
    Dim vendor As New Vendor

    ' Create the connection object
    Dim connection As New SqlConnection()
    connection.ConnectionString =
        "Data Source=localhost\SqlExpress;" &
        "Initial Catalog=AP;Integrated Security=True"

    ' Create the command object and set the connection,
    ' SELECT statement, and parameter value
    Dim selectCommand As New SqlCommand
    selectCommand.Connection = connection
    selectCommand.CommandText = "SELECT VendorID, " &
        "VendorName, VendorAddress1, VendorAddress2, " &
        "VendorCity, VendorState, VendorZipCode " &
        "FROM Vendors WHERE VendorID = @VendorID"
    selectCommand.Parameters.AddWithValue(
        "@VendorID", vendorID)
```

A Visual Basic function that uses ADO.NET to retrieve data from SQL Server (continued)

```
' Open the connection to the database
    connection.Open()

    ' Retrieve the row specified by the SELECT statement
    ' and load it into the Vendor object
    Dim reader As SqlDataReader =
        selectCommand.ExecuteReader
    If reader.Read Then
        vendor.VendorID = CInt(reader("VendorID"))
        vendor.VendorName = reader("VendorName").ToString
        vendor.VendorAddress1 =
            reader("VendorAddress1").ToString
        vendor.VendorAddress2 =
            reader("VendorAddress2").ToString
        vendor.VendorCity = reader("VendorCity").ToString
        vendor.VendorState = reader("VendorState").ToString
        vendor.VendorZipCode =
            reader("VendorZipCode").ToString
```

A Visual Basic function that uses ADO.NET to retrieve data from SQL Server (continued)

```
    Else
        vendor = Nothing
    End If
    reader.Close()

    ' Close the connection to the database
    connection.Close()

    Return vendor
End Function
```

A C# method that uses ADO.NET to retrieve data from SQL Server

```
public static Vendor GetVendor(int vendorID)
{
    Vendor vendor = new Vendor();

    // Create the connection object
    SqlConnection connection = new SqlConnection();
    connection.ConnectionString =
        "Data Source=localhost\\SqlExpress;" +
        "Initial Catalog=AP;Integrated Security=True";

    // Create the command object and set the connection,
    // SELECT statement, and parameter value
    SqlCommand selectCommand = new SqlCommand();
    selectCommand.Connection = connection;
    selectCommand.CommandText = "SELECT VendorID, " +
        "VendorName, VendorAddress1, VendorAddress2, " +
        "VendorCity, VendorState, VendorZipCode " +
        "FROM Vendors WHERE VendorID = @VendorID";
    selectCommand.Parameters.AddWithValue(
        "@VendorID", vendorID);
```

A C# method that uses ADO.NET to retrieve data from SQL Server (continued)

```
// Open the connection to the database
connection.Open();

// Retrieve the row specified by the SELECT statement
// and load it into the Vendor object
SqlDataReader reader = selectCommand.ExecuteReader();
if (reader.Read())
{
    vendor.VendorID = (int)reader["VendorID"];
    vendor.VendorName =
        reader["VendorName"].ToString();
    vendor.VendorAddress1 =
        reader["VendorAddress1"].ToString();
    vendor.VendorAddress2 =
       reader["VendorAddress2"].ToString();
    vendor.VendorCity =
       reader["VendorCity"].ToString();
    vendor.VendorState =
       reader["VendorState"].ToString();
```

A C# method that uses ADO.NET to retrieve data from SQL Server (continued)

```
            vendor.VendorZipCode =
                reader["VendorZipCode"].ToString();
        }
        else
        {
            vendor = null;
        }
        reader.Close();

        // Close the connection to the database
        connection.Close();

        return vendor;
    }
```