Chapter 1

An introduction to relational databases and SQL
Objectives

Knowledge

- Identify the three main hardware components of a client/server system.
- Describe the way a client accesses the database on a server using these terms: application software, data access API, database management system, SQL query, and query results.
- Describe the way a relational database is organized using these terms: tables, columns, rows, cells, primary keys, and foreign keys.
- Identify the three types of relationships that can exist between two tables.
- Describe the way the columns in a table are defined using these terms: data type, null value, and default value.
- Describe the difference between DML statements and DDL statements.
Objectives (continued)

Knowledge

• Describe the difference between an action query and a SELECT query.
• List three coding techniques that can make your SQL code easier to read and maintain.
• Explain how views and stored procedures differ from SQL statements that are issued from an application program.
• Describe the use of a database driver.
A simple client/server system
The three hardware components of a client/server system

- Clients
- Server
- Network

Terms to know

- Local area network (LAN)
- Wide area network (WAN)
- Enterprise system
Client software, server software, and the SQL interface

Client
- Application software
- Data access API

Database server
- Database management system
- Database

SQL queries
Results
Server software

- Database management system (DBMS)
- The DBMS does the *back-end processing*

Client software

- Application software
- Data access API (application programming interface)
- The client software does the *front-end processing*

The SQL interface

- SQL queries
- *SQL* stands for *Structured Query Language*
Client/server system
- Processing is divided between client and server

File-handling system
- All processing is done by the clients
An application that uses an application server
A simple web-based system
Other client/server components

- Application servers store business components
- Web servers store web applications and web services
How web applications work

- Web browser on a client sends a request to a web server.
- Web server processes the request.
- Web server passes any data requests to the database server.
- Database server returns results to web server.
- Web server returns a response to the browser.
The Vendors table in an Accounts Payable database

<table>
<thead>
<tr>
<th>VENDOR_ID</th>
<th>VENDOR_NAME</th>
<th>VENDOR_ADDRESS1</th>
<th>VENDOR_ADDRESS2</th>
<th>VENDOR_CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>US Postal Service</td>
<td>Attn: Supt. Window Ser...</td>
<td>PO Box 7005</td>
<td>Madison</td>
</tr>
<tr>
<td>2</td>
<td>National Informatio...</td>
<td>PO Box 96621</td>
<td>NULL</td>
<td>Washington</td>
</tr>
<tr>
<td>3</td>
<td>Register of Copyright</td>
<td>Library Of Congress</td>
<td>NULL</td>
<td>Washington</td>
</tr>
<tr>
<td>4</td>
<td>Jobtrak</td>
<td>1990 Westwood Blvd St...</td>
<td>NULL</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>5</td>
<td>Newbrige Book Clu...</td>
<td>3000 Cindel Drive</td>
<td>NULL</td>
<td>Washington</td>
</tr>
<tr>
<td>6</td>
<td>California Chamber...</td>
<td>3255 Ramos Cir</td>
<td>NULL</td>
<td>Sacramento</td>
</tr>
<tr>
<td>7</td>
<td>Towne Advertiser</td>
<td>Kevin Minder</td>
<td>3441 W Macarthur Blvd</td>
<td>Santa Ana</td>
</tr>
<tr>
<td>8</td>
<td>BFI Industries</td>
<td>PO Box 9389</td>
<td>NULL</td>
<td>Fresno</td>
</tr>
<tr>
<td>9</td>
<td>Pacific Gas &amp; Elect...</td>
<td>Box 52001</td>
<td>NULL</td>
<td>San Francisco</td>
</tr>
<tr>
<td>10</td>
<td>Robbins Mobile Loc...</td>
<td>4669 N Fresno</td>
<td>NULL</td>
<td>Fresno</td>
</tr>
<tr>
<td>11</td>
<td>Bill Marvin Electric</td>
<td>4583 E Home</td>
<td>NULL</td>
<td>Fresno</td>
</tr>
<tr>
<td>12</td>
<td>City Of Fresno</td>
<td>PO Box 2069</td>
<td>NULL</td>
<td>Fresno</td>
</tr>
<tr>
<td>13</td>
<td>Golden Eagle Insur...</td>
<td>PO Box 85826</td>
<td>NULL</td>
<td>San Diego</td>
</tr>
<tr>
<td>14</td>
<td>Expedata Inc</td>
<td>4420 N. First Street, Suit...</td>
<td>NULL</td>
<td>Fresno</td>
</tr>
</tbody>
</table>
Terms to know

- Relational database
- Table
- Row
- Column
- Cell
- Primary key
- Composite primary key
- Non-primary key (unique key)
- Index
The relationship between two tables

Primary key

Foreign key
Terms to know

- Foreign key
- One-to-many relationship
- One-to-one relationship
- Many-to-many relationship
The columns of the Invoices table
Common Oracle data types

- CHAR, VARCHAR2
- NUMBER
- FLOAT
- DATE
Terms to know

- Data type
- Null value
- Default value
## Important events in the history of SQL

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>Dr. E. F. Codd develops the relational database model.</td>
</tr>
<tr>
<td>1978</td>
<td>IBM develops the predecessor to SQL.</td>
</tr>
<tr>
<td>1979</td>
<td>Relational Software, Inc. (later renamed Oracle) releases the first relational DBMS, Oracle.</td>
</tr>
<tr>
<td>1982</td>
<td>IBM releases their first RDBMS, SQL/DS (SQL/Data System).</td>
</tr>
<tr>
<td>1985</td>
<td>IBM released DB2 (Database 2).</td>
</tr>
<tr>
<td>1987</td>
<td>Microsoft releases SQL Server.</td>
</tr>
<tr>
<td>1989</td>
<td>ANSI publishes first SQL standards (ANSI/ISO SQL-89, or SQL1).</td>
</tr>
</tbody>
</table>
How knowing “standard SQL” helps you

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

How knowing “standard SQL” does not help you

- Any application requires changes when moved to another database.
First database releases

Oracle  1979  
DB2    1985  
SQL Server  1987  

Primary platforms

Oracle       Unix  
            OS/390 and z/OS  
DB2       Unix  
            OS/390 and z/OS  
SQL Server  Windows
SQL DML statements

- SELECT
- INSERT
- UPDATE
- DELETE

SQL DDL statements

- CREATE USER, TABLE, SEQUENCE, INDEX
- ALTER USER, TABLE, SEQUENCE, INDEX
- DROP USER, TABLE, SEQUENCE, INDEX
- GRANT
- REVOKE
A statement that creates a new user for a database

`CREATE USER ap IDENTIFIED BY ap`

A statement that grants privileges to a user

`GRANT ALL PRIVILEGES TO ap`
A statement that creates a new table

CREATE TABLE invoices
(
    invoice_id NUMBER NOT NULL,
    vendor_id NUMBER NOT NULL,
    invoice_number VARCHAR2(50) NOT NULL,
    invoice_date DATE NOT NULL,
    invoice_total NUMBER(9,2) NOT NULL,
    payment_total NUMBER(9,2) DEFAULT 0,
    credit_total NUMBER(9,2) DEFAULT 0,
    terms_id NUMBER NOT NULL,
    invoice_due_date DATE NOT NULL,
    payment_date DATE,
    CONSTRAINT invoices_pk
        PRIMARY KEY (invoice_id),
    CONSTRAINT invoices_fk_vendors
        FOREIGN KEY (vendor_id)
            REFERENCES vendors (vendor_id),
    CONSTRAINT invoices_fk_terms
        FOREIGN KEY (terms_id)
            REFERENCES terms (terms_id)
)
A statement that adds a new column to a table

```
ALTER TABLE invoices
ADD balance_due NUMBER(9,2)
```

A statement that deletes the new column

```
ALTER TABLE invoices
DROP COLUMN balance_due
```
A statement that creates an index on the table

```sql
CREATE INDEX invoices_vendor_id_index
  ON invoices (vendor_id)
```

A statement that deletes the new index

```sql
DROP INDEX invoices_vendor_id_index
```
A statement that creates a sequence

CREATE SEQUENCE invoice_id_seq
START WITH 115
INCREMENT BY 1
The Invoices base table

<table>
<thead>
<tr>
<th>INVOICE_ID</th>
<th>VENDOR_ID</th>
<th>INVOICE_NUMBER</th>
<th>INVOICE_DATE</th>
<th>INVOICE_TOTAL</th>
<th>PAYMENT_TOTAL</th>
<th>CREDIT_TOTAL</th>
<th>TERMS_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>34 QF58872</td>
<td>25-FEB-08</td>
<td>116.54</td>
<td>116.54</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>34 Q545443</td>
<td>14-MAR-06</td>
<td>1063.55</td>
<td>1063.56</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>110 P-0568</td>
<td>11-APR-06</td>
<td>20551.13</td>
<td>0</td>
<td>1200</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>110 P-0259</td>
<td>16-APR-06</td>
<td>26001.4</td>
<td>26001.4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>81 MARO1439</td>
<td>16-APR-06</td>
<td>936.93</td>
<td>936.93</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>122 985319-497</td>
<td>17-APR-06</td>
<td>2312.2</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>82 G73-24</td>
<td>17-APR-06</td>
<td>600</td>
<td>600</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>122 989319-487</td>
<td>18-APR-06</td>
<td>1927.54</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>122 989319-477</td>
<td>18-APR-06</td>
<td>2164.11</td>
<td>2164.11</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>122 989319-467</td>
<td>24-APR-06</td>
<td>2318.03</td>
<td>2318.03</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>122 989319-457</td>
<td>24-APR-06</td>
<td>3013.33</td>
<td>3013.33</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>122 989319-447</td>
<td>24-APR-06</td>
<td>3689.99</td>
<td>3689.99</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>122 989319-437</td>
<td>24-APR-06</td>
<td>2765.36</td>
<td>2765.36</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>122 989319-427</td>
<td>25-APR-06</td>
<td>2115.81</td>
<td>2115.81</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>121 976538</td>
<td>26-APR-06</td>
<td>313.55</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>
A SELECT statement that retrieves and sorts selected rows

```
SELECT invoice_number, invoice_date, invoice_total, payment_total, credit_total, 
    invoice_total - payment_total - credit_total AS balance_due
FROM invoices
WHERE invoice_total - payment_total - credit_total > 0
ORDER BY invoice_date
```

The result set for the SELECT statement

<table>
<thead>
<tr>
<th>INVOICE_NUMBER</th>
<th>INVOICE_DATE</th>
<th>INVOICE_TOTAL</th>
<th>PAYMENT_TOTAL</th>
<th>CREDIT_TOTAL</th>
<th>BALANCE_DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-0608</td>
<td>11-APR-08</td>
<td>20551.18</td>
<td>0</td>
<td>1200</td>
<td>19351.18</td>
</tr>
<tr>
<td>989319-497</td>
<td>17-APR-08</td>
<td>2312.2</td>
<td>0</td>
<td>0</td>
<td>2312.2</td>
</tr>
<tr>
<td>989319-487</td>
<td>18-APR-08</td>
<td>1927.54</td>
<td>0</td>
<td>0</td>
<td>1927.54</td>
</tr>
<tr>
<td>97/553B</td>
<td>26-APR-08</td>
<td>313.55</td>
<td>0</td>
<td>0</td>
<td>313.55</td>
</tr>
<tr>
<td>97/553</td>
<td>27-APR-08</td>
<td>904.14</td>
<td>0</td>
<td>0</td>
<td>904.14</td>
</tr>
<tr>
<td>97/522</td>
<td>30-APR-08</td>
<td>1962.13</td>
<td>0</td>
<td>200</td>
<td>1762.13</td>
</tr>
</tbody>
</table>
A SELECT statement that joins two tables

```sql
SELECT vendor_name, invoice_number, invoice_date, invoice_total
FROM vendors INNER JOIN invoices
    ON vendors.vendor_id = invoices.vendor_id
WHERE invoice_total >= 500
ORDER BY vendor_name, invoice_total DESC
```

The result set for the SELECT statement

<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>INVOICE_NUMBER</th>
<th>INVOICE_DATE</th>
<th>INVOICE_TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Federal Express Corporation</td>
<td>963253230</td>
<td>15-MAY-08</td>
<td>739.2</td>
</tr>
<tr>
<td>8 Ford Motor Credit Company</td>
<td>9982771</td>
<td>03-JUN-08</td>
<td>503.2</td>
</tr>
<tr>
<td>9 Franchise Tax Board</td>
<td>RTR-72-3662-X</td>
<td>04-JUN-08</td>
<td>1600</td>
</tr>
<tr>
<td>10 Fresno County Tax Collector</td>
<td>P02-88D77S7</td>
<td>06-JUN-08</td>
<td>856.92</td>
</tr>
<tr>
<td>11 IBM</td>
<td>Q545443</td>
<td>14-MAR-08</td>
<td>1083.58</td>
</tr>
<tr>
<td>12 Ingram</td>
<td>31359783</td>
<td>23-MAY-08</td>
<td>1575</td>
</tr>
<tr>
<td>13 Ingram</td>
<td>31361833</td>
<td>23-MAY-08</td>
<td>579.42</td>
</tr>
<tr>
<td>14 Malloy Lithographing Inc</td>
<td>0-2058</td>
<td>08-MAY-08</td>
<td>37966.19</td>
</tr>
<tr>
<td>15 Malloy Lithographing Inc</td>
<td>P-0259</td>
<td>16-APR-08</td>
<td>26881.4</td>
</tr>
<tr>
<td>16 Malloy Lithographing Inc</td>
<td>0-2060</td>
<td>08-MAY-08</td>
<td>23517.58</td>
</tr>
<tr>
<td>17 Malloy Lithographing Inc</td>
<td>P-0608</td>
<td>11-APR-08</td>
<td>20551.18</td>
</tr>
<tr>
<td>18 Malloy Lithographing Inc</td>
<td>0-2436</td>
<td>07-MAY-08</td>
<td>10976.06</td>
</tr>
<tr>
<td>19 Pollstar</td>
<td>77290</td>
<td>04-JUN-08</td>
<td>1750</td>
</tr>
</tbody>
</table>
A statement that adds a row to the Invoices table

```sql
INSERT INTO invoices
    (invoice_id, vendor_id, invoice_number, invoice_date,
     invoice_total, terms_id, invoice_due_date)
VALUES
    (invoice_id_seq.NEXTVAL, 12, '3289175', '18-JUL-08',
     165, 3, '17-AUG-08')
```
A statement that changes one value in one row

```
UPDATE invoices
SET credit_total = 35.89
WHERE invoice_number = '367447'
```

A statement that changes one value in multiple rows

```
UPDATE invoices
SET invoice_due_date = invoice_due_date + 30
WHERE terms_id = 4
```
A statement that deletes a selected invoice

DELETE FROM invoices
WHERE invoice_number = '4-342-8069'

A statement that deletes all paid invoices

DELETE FROM invoices
WHERE invoice_total - payment_total - credit_total = 0
Terms to know

- Query
- Result table (result set)
- Calculated value
- Join
- Inner join
- Outer join
- Action query
A CREATE VIEW statement for a view

```
CREATE VIEW vendors_min AS
    SELECT vendor_name, vendor_state, vendor_phone
    FROM vendors
```

The virtual table for the view

<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>VENDOR_STATE</th>
<th>VENDOR_PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 US Postal Service</td>
<td>WI</td>
<td>(800) 555-1205</td>
</tr>
<tr>
<td>2 National Information Data Ctr</td>
<td>DC</td>
<td>(301) 555-8950</td>
</tr>
<tr>
<td>3 Register of Copyrights</td>
<td>DC</td>
<td>NULL</td>
</tr>
<tr>
<td>4 Jobtrak</td>
<td>CA</td>
<td>(800) 555-8725</td>
</tr>
<tr>
<td>5 Newbrige Book Clubs</td>
<td>NJ</td>
<td>(800) 555-9980</td>
</tr>
<tr>
<td>6 California Chamber Of Commerce</td>
<td>CA</td>
<td>(916) 555-6670</td>
</tr>
<tr>
<td>7 Towne Advertiser's Mailing Svcs</td>
<td>CA</td>
<td>NULL</td>
</tr>
</tbody>
</table>
A SELECT statement that uses the view

```sql
SELECT *
FROM vendors_min
WHERE vendor_state = 'CA'
ORDER BY vendor_name
```

The result set

<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>VENDOR_STATE</th>
<th>VENDOR_PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASC Signs</td>
<td>CA</td>
<td>NULL</td>
</tr>
<tr>
<td>Abbey Office Furnishings</td>
<td>CA</td>
<td>(559) 555-8300</td>
</tr>
<tr>
<td>American Express</td>
<td>CA</td>
<td>(800) 555-3344</td>
</tr>
<tr>
<td>Aztek Label</td>
<td>CA</td>
<td>(714) 555-9000</td>
</tr>
<tr>
<td>BFI Industries</td>
<td>CA</td>
<td>(559) 555-1551</td>
</tr>
<tr>
<td>Bertelsmann Industry Svcs. Inc</td>
<td>CA</td>
<td>(805) 555-0584</td>
</tr>
<tr>
<td>Bill Jones</td>
<td>CA</td>
<td>NULL</td>
</tr>
</tbody>
</table>
Terms to know

- View
- Virtual table
A SELECT statement that’s difficult to read

```sql
select invoice_number, invoice_date, invoice_total, payment_total, credit_total, invoice_total - payment_total - credit_total as balance_due from invoices where invoice_total - payment_total - credit_total > 0 order by invoice_date
```

A SELECT statement with a readable style

```sql
SELECT invoice_number, invoice_date, invoice_total, payment_total, credit_total,
       invoice_total - payment_total - credit_total
       AS balance_due
FROM invoices
WHERE invoice_total - payment_total - credit_total > 0
ORDER BY invoice_date
```
SELECT statement with a block comment
/*
Author: Joel Murach
Date: 8/22/2008
*/
SELECT invoice_number, invoice_date, invoice_total, invoice_total - payment_total - credit_total AS balance_due
FROM invoices

A SELECT statement with a single-line comment
-- The fourth column calculates the balance due
SELECT invoice_number, invoice_date, invoice_total, invoice_total - payment_total - credit_total AS balance_due
FROM invoices
Coding recommendations

- Capitalize all keywords.
- Use lowercase for the other code.
- Separate the words in names with underscores.
- Start each clause on a new line.
- Break long clauses into multiple lines.
- Indent continued lines.
- Use comments only for code that is hard to understand.
- Make sure that the comments are correct and up-to-date.
A CREATE PROCEDURE statement

CREATE OR REPLACE PROCEDURE update_invoices_credit_total
(
    invoice_number_param VARCHAR2,
    credit_total_param NUMBER
)
AS
BEGIN
    UPDATE invoices
    SET credit_total = credit_total_param
    WHERE invoice_number = invoice_number_param;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
END;
/

A statement that executes the stored procedure

CALL update_invoices_credit_total('367447', 35.89)
A CREATE FUNCTION statement

CREATE OR REPLACE FUNCTION avg_invoice_total
(
    vendor_id_param INTEGER
)
RETURN NUMBER
AS
    avg_invoice_total_var NUMBER(9,2);
BEGIN
    SELECT AVG(invoice_total)
    INTO avg_invoice_total_var
    FROM invoices
    WHERE vendor_id = vendor_id_param;

    RETURN avg_invoice_total_var;
END;
/

A statement that uses the function

```sql
SELECT vendor_id, invoice_total, 
    avg_invoice_total(vendor_id)
FROM invoices
ORDER BY vendor_id
```

The result set

<table>
<thead>
<tr>
<th>VENDOR_ID</th>
<th>INVOICE_TOTAL</th>
<th>AVG_INVOICE_TOTAL(VENDOR_ID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>34</td>
<td>1083.58</td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td>116.54</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>224</td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>224</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>116</td>
</tr>
<tr>
<td>6</td>
<td>48</td>
<td>856.92</td>
</tr>
<tr>
<td>7</td>
<td>72</td>
<td>85.31</td>
</tr>
</tbody>
</table>
Terms to know

- PL/SQL (Procedure Language/SQL)
- Stored procedure
- Function (user-defined function or UDF)
Common options for accessing Oracle data

- Java application
  - JDBC
  - Java driver
  - Oracle

- .NET application
  - ADO.NET
  - .NET driver
  - Oracle

- Visual Basic 6 application
  - ADO
  - OLE DB
  - Oracle
Two commonly used Oracle JDBC drivers

- Thin driver
- OCI driver
Terms to know

- Data access model
- JDBC (Java Database Connectivity)
- ADO.NET (ActiveX Data Objects)
- Database driver
The readme file for Oracle JDBC drivers
C:\oracle\product\10.2.0\server\jdbc\readme.txt

The JAR file for the Oracle 10g Express JDBC driver
C:\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar
A Java class that gets data from an Oracle database

```java
import java.sql.*;
import java.text.NumberFormat;

public class DBTestApp
{
    public static void main(String args[])
    {
        // Load the database driver
        // NOTE: This block is for Oracle 10g (JDBC 3.0),
        // but not for Oracle 11g (JDBC 4.0)
        try
        {
            Class.forName("oracle.jdbc.OracleDriver");
        }
        catch(ClassNotFoundException e)
        {
            e.printStackTrace();
        }
    }
}
```
The Java class (continued)

// define common JDBC objects
Connection connection = null;
Statement statement = null;
ResultSet rs = null;
try
{

// Connect to the database
String dbUrl =
    "jdbc:oracle:thin:@localhost:1521:XE";
String username = "ap";
String password = "ap";
connection = DriverManager.getConnection(
    dbUrl, username, password);
The Java class (continued)

// Execute a SELECT statement
statement = connection.createStatement();
String query =
    "SELECT vendor_name, invoice_number,
    invoice_total " +
    "FROM vendors INNER JOIN invoices " +
    "  ON vendors.vendor_id =
    invoices.vendor_id " +
    "WHERE invoice_total >= 500 " +
    "ORDER BY vendor_name, invoice_total DESC";
rs = statement.executeQuery(query);
The Java class (continued)

    // Display the results of a SELECT statement
    System.out.println(
        "Invoices with totals over 500:\n");
    while(rs.next())
    {
        String vendorName =
            rs.getString("vendor_name");
        String invoiceNumber =
            rs.getString("invoice_number");
        double invoiceTotal =
            rs.getDouble("invoice_total");

        NumberFormat currency =
            NumberFormat.getCurrencyInstance();
        String invoiceTotalString =
            currency.format(invoiceTotal);
The Java class (continued)

```java
    System.out.println(
        "Vendor:     " + vendorName + "\n" +
        "Invoice No: " + invoiceNumber + "\n" +
        "Total:      " + invoiceTotalString +
        "\n");
    }
}

try {
    }
}

```

```java
    catch(SQLException e)
    {
        e.printStackTrace();
    }
```
The Java class (continued)

```java
finally
{
    try
    {
        if (rs != null)
            rs.close();
        if (statement != null)
            statement.close();
        if (connection != null)
            connection.close();
    }
    catch(SQLException e)
    {
        e.printStackTrace();
    }
}
```