8.4 Application programming interfaces

This section has been set as optional by your instructor.

Overview

An application programming interface (API) specifies the interaction between an application and a computer service, such as a database, email, or web service. If the application is written in a procedural language like C, the API specifies procedure calls. If the application is written in an object-oriented language like Java, the API specifies classes.

Hundreds of APIs are available, connecting major programming languages to commonly used services. Dozens of APIs are available for relational databases and other data sources. Each programming language supports different APIs for different services. Ex: Java supports JMS for message services, JSAPI for speech synthesis and recognition services, JavaMail for email services, and many others. Conversely, each service may have different APIs for different programming languages.

Leading database APIs include:

- ***SQL/Call Language Interface*** (***SQL/CLI***), an extension of the SQL standard
- ***ODBC*** supports many programming languages and data sources. ODBC was developed in parallel with SQL/CLI and conforms closely to the standard.
- ***JDBC*** for Java applications
- ***DB-API*** for Python applications
- ***ADO.NET*** for .NET applications. .NET is a Microsoft environment, and C# is the most popular .NET language.
- ***PDO*** for PHP applications. PHP is a widely used programming language for building dynamic websites.

MySQL connectors implement leading APIs between the MySQL database and major programming languages.

Table 8.4.1: Leading database APIs.

Acronym	Derivation	Original Sponsor	Initial release	Application language	MySQL Connector

ODBC	Open Database Connectivity	Microsoft	1992	Numerous	Connector/ODBC
JDBC	Java Database Connectivity	Sun Microsystems (now Oracle)	1997	Java	Connector/J
DB-API	Database API	Python Software Foundation	1996	Python	Connector/Python
ADO.NET	ActiveX Data Objects	Microsoft	2002	.NET languages	Connector/NET
PDO	PHP Data Objects	The PHP Group	2005	PHP	none

Feedback?

Terminology

*Use of the term **application programming interface**, or **API**, varies. The term always refers to the syntax and expected behavior of procedure or class declarations, as specified by a standards organization. In some cases, the term also refers to the implementation of the procedures or classes in complete code libraries.*

PARTICIPATION ACTIVITY 8.4.1: Database APIs.

1) Which API supports only one language ?

- ○ ODBC
- ◉ DB-API
- ○

Correct

DB-API is sponsored by the Python Software Foundation and supports Python only.

ADO.NET

2) Which API is often used with the C# language ?

- JDBC
- DB-API
- ADO.NET

Correct

ADO.NET is a Microsoft API designed for .NET languages. C# is the leading .NET language and commonly used with ADO.NET.

3) Database APIs support:

- Relational databases only
- Relational and non-relational databases only
- Databases and other data sources

Correct

The ODBC API supports file formats like .xlsx (Microsoft Excel) and .csv (comma-separated values), as well as databases.

Feedback?

Capabilities

At a high level, database APIs are similar and support common capabilities, including:

- **Manage connections**. A database connection identifies the database address and provides login credentials. A connection must be created and opened before queries are executed.
- **Prepare queries**. When a query is executed repeatedly, compiling once prior to execution is faster than compiling each time the query is executed. In database APIs, compiling prior to execution is often called 'preparing' a query.
- **Execute queries with single-row results**. Single-row queries are relatively easy to embed in API procedure calls. Procedure parameters transfer query results to program variables.
- **Execute queries with multiple-row results**. Queries that return multiple rows require special processing. A special API object, called a 'cursor' or 'reader', identifies a single row of the result table. The cursor advances through the result table as the application processes one row at a time.
- **Call stored procedures**. Stored procedures are executed with the API 'execute query' capability, like other SQL statements. Special API syntax matches stored procedure parameters with host language variables.

Specific syntax of the above capabilities varies greatly, depending on the API and host language. A JDBC code example appears below. A detailed description of DB-API for Python and MySQL

appears elsewhere in this material.

PARTICIPATION ACTIVITY 8.4.2: JDBC example.

1 2 3 4 5 6 7 2x speed

Procedural SQL

```
CREATE PROCEDURE FlightCount(IN airline VARCHAR(20), OUT quantity INT)
  SELECT COUNT(*)
  INTO quantity
  FROM Flight
  WHERE AirlineName = airline;
```

Java

```
import java.sql.*;
import java.util.Scanner;

public class JdbcDemo {
   public static void main(String[] args) throws SQLException {

      Connection reservation = DriverManager.getConnection("jdbc:mysql://localhost/Reserv
                                                           "samsnead", "b98$xm");
      CallableStatement flightCall = reservation.prepareCall("{ CALL FlightCount(?, ?) }"
      System.out.print("Enter airline name: ");
      Scanner scnr = new Scanner(System.in);
      String airlineName = scnr.nextLine();

      flightCall.setString("Airline", airlineName);
      flightCall.registerOutParameter("Quantity", Types.INTEGER);
      flightCall.executeQuery();
      int total = flightCall.getInt("Quantity");

      System.out.println("Airline: " + airlineName + "\nTotal flights: " + total);
      reservation.close();
   }
}
```

The Java variable total is assigned the Quantity value and displayed.

Captions

1. The FlightCount procedure is written in procedural SQL and stored in the Reservation database.
2. The java.sql namespace implements the JDBC API.
3. The reservation object opens a connection to the reservation database. close() releases the connection.
4. prepareCall() compiles CALL FlightCount and saves the statement in the flightCall object. ? characters are placeholders for parameters.
5. Java code inputs an airline name to the Java variable airlineName.
6. The airline parameter is set to airlineName, the quantity parameter is registered as an integer, and the query is executed.

7. The Java variable total is assigned the Quantity value and displayed.

Feedback?

PARTICIPATION ACTIVITY 8.4.3: JDBC API capabilities.

Refer to the animation above. Match the Java language element with the description.

import java.sql.*;	Makes JDBC classes available to the Java program. In Java, the import statement imports a class library into a program. The java.sql.* namespace includes classes that implement the JDBC API.	Correct
reservation	Identifies the database location and provides database access credentials. reservation is an object of the Connection class. A connection identifies a database for use with subsequent queries and specifies a database login and password.	Correct
flightCall	Contains an SQL stored procedure call. flightCall is an object of the CallableStatement class, which includes data and procedures necessary to call stored procedures.	Correct
prepareCall()	Compiles an SQL statement and associates the statement with a Java object. The term 'prepare' commonly means 'compile' in database APIs. prepareCall() compiles a text string representing an SQL statement and	Correct

	assigns the compiled statement to a Java object.	
getInt()	Returns the value of a stored procedure output parameter. getInt("Quantity") returns the value of the Quantity parameter after the FlightCount stored procedure executes.	**Correct**

Reset

Feedback?

Architecture

A database API is implemented in source code by API developers. The compiled code is distributed to application developers and linked to API calls when applications are compiled.

Database API implementations typically contain two software layers:

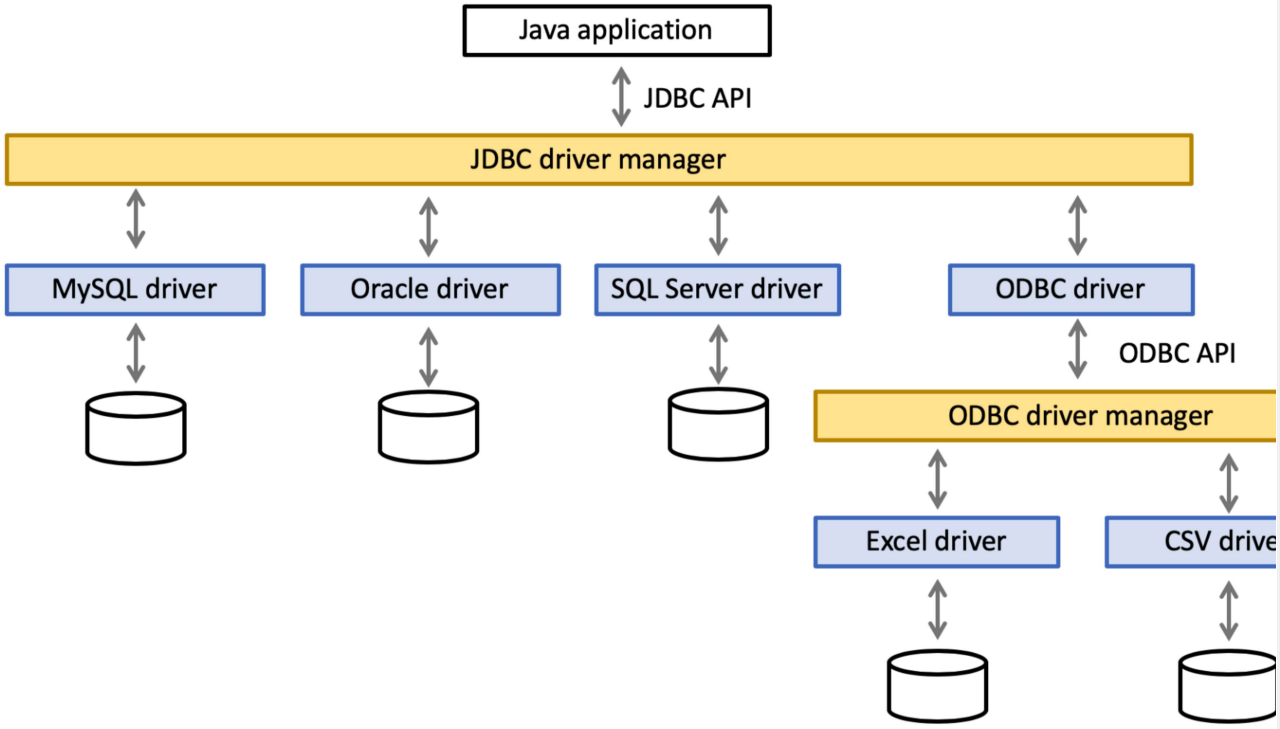
- A ***driver*** connects directly to the database. Although all relational databases support standard SQL, implementations vary somewhat, and most databases offer non-standard extensions. Consequently, a different driver is necessary for each database.
- The ***driver manager*** connects the application to the drivers. The application communicates with the driver manager using API procedure calls. The driver manager forwards each call to the correct driver. When one application connects to several databases, the driver manager selects a driver based on the active connection.

Many database APIs support non-relational data sources such as Excel spreadsheets (.xlsx files) and comma-separated data (.csv files). Each supported data source requires a different driver. Since many data sources do not support SQL capabilities such as transaction management, drivers to non-relational sources may not support the full API.

A primary API may support a secondary API. Ex: A JDBC implementation may include an ODBC driver, enabling access to any ODBC data source from JDBC. Supporting a secondary API is useful when a data source is not supported by the primary API. The secondary API introduces additional layers between application and data source, however, and therefore is inefficient.

PARTICIPATION ACTIVITY 8.4.4: API architecture.

1 2 3 4 2x speed



The ODBC driver for JDBC communicates with Excel and CSV data sources via the ODBC driver m

Captions

1. Java applications communicate via the JDBC API with the JDBC driver manager.
2. The driver manager communicates with each data source via a different driver.
3. If an API has no driver for a data source, the API may communicate indirectly via another API.
4. The ODBC driver for JDBC communicates with Excel and CSV data sources via the ODBC driver manager.

Feedback?

PARTICIPATION ACTIVITY 8.4.5: Architecture.

1) Each database requires a different driver manager.

- True
- False

Correct

Each database requires a different driver, but all drivers communicate with a single driver manager.

2) The driver manager communicates with all drivers using the same commands.

Correct

API developers must implement a different driver for each database to accommodate query language

True
False

differences across databases. The driver manager is unaware of differences between databases.

3) Every driver implements the full capabilities of the API.

- True
- False

Correct

Some data sources do not support all API capabilities. Ex: .csv files do not support transaction management. When a data source does not support an API capability, the driver cannot support the capability.

4) In the animation above, the secondary API is ODBC.

- True
- False

Correct

The primary API, JDBC, interacts directly with the application. The secondary API, ODBC, interacts with the primary API.

5) A secondary API can be layered on top of a third API, for a total of three API layers.

- True
- False

Correct

In principle, any API can act as a data source for another API, so APIs can be several layers deep. In practice, however, each layer introduces inefficiency. APIs rarely have more than two layers.

Feedback?

Exploring further:

- MySQL connectors

How was this section? **Provide feedback**