Most database users do not use a query language like SQL

An application program acts as the intermediary between users and the database

- Applications split into
  - front-end
  - middle layer
  - backend

Front-end: user interface

- Forms
- Graphical user interfaces
- Many interfaces are Web-based
Application Architecture Evolution

- Three distinct era's of application architecture
  - mainframe (1960’s and 70’s)
  - personal computer era (1980’s)
  - We era (1990’s onwards)

Web Interface

- Web browsers have become the de-facto standard user interface to databases
  - Enable large numbers of users to access databases from anywhere
  - Avoid the need for downloading/installing specialized code, while providing a good graphical user interface
    - Javascript, Flash and other scripting languages run in browser, but are downloaded transparently
  - Examples: banks, airline and rental car reservations, university course registration and grading, and so on.
The World Wide Web

- The Web is a distributed information system based on hypertext.
- Most Web documents are hypertext documents formatted via the HyperText Markup Language (HTML)
- HTML documents contain
  - text along with font specifications, and other formatting instructions
  - hypertext links to other documents, which can be associated with regions of the text.
  - forms, enabling users to enter data which can then be sent back to the Web server

Uniform Resources Locators

- In the Web, functionality of pointers is provided by Uniform Resource Locators (URLs).
- URL example:
  - http://www.acm.org/sigmod
  - The first part indicates how the document is to be accessed
    - “http” indicates that the document is to be accessed using the Hyper Text Transfer Protocol.
  - The second part gives the unique name of a machine on the Internet.
  - The rest of the URL identifies the document within the machine.
- The local identification can be:
  - The path name of a file on the machine, or
  - An identifier (path name) of a program, plus arguments to be passed to the program
    - E.g., http://www.google.com/search?q=silberschatz
HTML and HTTP

- HTML provides formatting, hypertext link, and image display features
  - including tables, stylesheets (to alter default formatting), etc.
- HTML also provides input features
  - Select from a set of options
    - Pop-up menus, radio buttons, check lists
  - Enter values
    - Text boxes
  - Filled in input sent back to the server, to be acted upon by an executable at the server
- HyperText Transfer Protocol (HTTP) used for communication with the Web server

Web Servers

- A Web server can easily serve as a front end to a variety of information services.
- The document name in a URL may identify an executable program, that, when run, generates a HTML document.
  - When an HTTP server receives a request for such a document, it executes the program, and sends back the HTML document that is generated.
  - The Web client can pass extra arguments with the name of the document.
- To install a new service on the Web, one simply needs to create and install an executable that provides that service.
  - The Web browser provides a graphical user interface to the information service.
- Common Gateway Interface (CGI): a standard interface between web and application server
Three-Layer Web Architecture

- Multiple levels of indirection have overheads
- Alternative: two-layer architecture

Two-Layer Web Architecture
HTTP and Sessions

- The HTTP protocol is **connectionless**
  - That is, once the server replies to a request, the server closes the connection with the client, and forgets all about the request
  - In contrast, Unix logins, and JDBC/ODBC connections stay connected until the client disconnects
    - retaining user authentication and other information
  - Motivation: reduces load on server
    - operating systems have tight limits on number of open connections on a machine
- Information services need session information
  - E.g., user authentication should be done only once per session
- Solution: use a **cookie**

Sessions and Cookies

- A **cookie** is a small piece of text containing identifying information
  - Sent by server to browser
    - Sent on first interaction, to identify session
  - Sent by browser to the server that created the cookie on further interactions
    - part of the HTTP protocol
  - Server saves information about cookies it issued, and can use it when serving a request
    - E.g., authentication information, and user preferences
- Cookies can be stored permanently or for a limited time
Servlets

- Java Servlet specification defines an API for communication between the Web/application server and application program running in the server
  - E.g., methods to get parameter values from Web forms, and to send HTML text back to client
- Application program (also called a servlet) is loaded into the server
  - Each request spawns a new thread in the server
    - thread is closed once the request is serviced

Example Servlet Code

```java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PersonQueryServlet extends HttpServlet {
  public void doGet (HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HEAD><TITLE> Query Result</TITLE></HEAD>");
    out.println("<BODY>");
    ...... BODY OF SERVLET (next slide) ...
    out.println("</BODY>");
    out.close();
  }
}```
Example Servlet Code

```java
String persontype = request.getParameter("persontype");
String number = request.getParameter("name");
if(persontype.equals("student")) {
    ... code to find students with the specified name ...
    ... using JDBC to communicate with the database ...
    out.println("<table BORDER COLS=3> ");
    out.println(" <tr> <td>ID</td> <td>Name: </td> + " <td>Department</td> </tr>");
    for(... each result ...){
        ... retrieve ID, name and dept name
        ... into variables ID, name and deptname
        out.println("<tr> <td>" + ID + "</td>" + "<td>" + name + "</td>" + "<td>" + deptname + "</td> </tr>");
    }
    out.println("</table>");
} else {
    ... as above, but for instructors ...
}
```

Servlet Sessions

- Servlet API supports handling of sessions
  - Sets a cookie on first interaction with browser, and uses it to identify session on further interactions
- To check if session is already active:
  - if (request.getSession(false) == true)
    - .. then existing session
    - else .. redirect to authentication page
  - authentication page
    - check login/password
    - request.getSession(true): creates new session
- Store/retrieve attribute value pairs for a particular session
  - session.setAttribute("userid", userid)
  - session.getAttribute("userid")
**Servlet Support**

- Servlets run inside application servers such as
  - Apache Tomcat, Glassfish, JBoss
  - BEA Weblogic, IBM WebSphere and Oracle Application Servers
- Application servers support
  - deployment and monitoring of servlets
  - Java 2 Enterprise Edition (J2EE) platform supporting objects, parallel processing across multiple application servers, etc

---

**Server-Side Scripting**

- Server-side scripting simplifies the task of connecting a database to the Web
  - Define an HTML document with embedded executable code/SQL queries.
  - Input values from HTML forms can be used directly in the embedded code/SQL queries.
  - When the document is requested, the Web server executes the embedded code/SQL queries to generate the actual HTML document.
- Numerous server-side scripting languages
  - JSP, PHP
  - General purpose scripting languages: VBScript, Perl, Python
Java Server Pages (JSP)

- A JSP page with embedded Java code
  ```html
  <html>
  <head> <title> Hello </title> </head>
  <body>
  <% if (request.getParameter("name") == null) {
      out.println("Hello World");
  } else {
      out.println("Hello, " + request.getParameter("name");
  }%>
  </body>
  </html>
  ```

- JSP is compiled into Java + Servlets
- JSP allows new tags to be defined, in tag libraries
  - such tags are like library functions, can be used for example to build rich user interfaces such as paginated display of large datasets

PHP

- PHP is widely used for Web server scripting
- Extensive libraries including for database access using ODBC
  ```html
  <html>
  <head> <title> Hello </title> </head>
  <body>
  <?php if (!isset($_REQUEST['name'])) {
    echo "Hello World";
  } else {
    echo "Hello, " + $_REQUEST['name'];
  }?>
  </body>
  </html>
  ```
Client Side Scripting

- Browsers can fetch certain scripts (client-side scripts) or programs along with documents, and execute them in "safe mode" at the client site
  - Javascript
  - Macromedia Flash and Shockwave for animation/games
  - VRML
  - Applets
- Client-side scripts/programs allow documents to be active
  - E.g., animation by executing programs at the local site
  - E.g., ensure that values entered by users satisfy some correctness checks
  - Permit flexible interaction with the user.
    - Executing programs at the client site speeds up interaction by avoiding many round trips to server

Client Side Scripting and Security

- Security mechanisms needed to ensure that malicious scripts do not cause damage to the client machine
  - Easy for limited capability scripting languages, harder for general purpose programming languages like Java
- E.g., Java’s security system ensures that the Java applet code does not make any system calls directly
  - Disallows dangerous actions such as file writes
  - Notifies the user about potentially dangerous actions, and allows the option to abort the program or to continue execution.
Javascript

- Javascript very widely used
  - forms basis of new generation of Web applications (called Web 2.0 applications) offering rich user interfaces

- Javascript functions can
  - check input for validity
  - modify the displayed Web page, by altering the underling document object model (DOM) tree representation of the displayed HTML text
  - communicate with a Web server to fetch data and modify the current page using fetched data, without needing to reload/refresh the page
    - forms basis of AJAX technology used widely in Web 2.0 applications
    - E.g., on selecting a country in a drop-down menu, the list of states in that country is automatically populated in a linked drop-down menu

Example of Javascript used to validate form input

```html
<html><head>
<script type="text/javascript">
    function validate() {
        var credits=document.getElementById("credits").value;
        if (isNaN(credits)|| credits<=0 || credits>=16) {
            alert("Credits must be a number greater than 0 and less than 16");
            return false;
        }
    }
</script>
<body>
<form action="createCourse" onsubmit="return validate()">
    Title: <input type="text" id="title" size="20"><br />
    Credits: <input type="text" id="credits" size="2"><br />
    <input type="submit" value="Submit">
</form>
</body></html>
```
Application Architectures

- Application layers
  - Presentation or user interface
    - model-view-controller (MVC) architecture
      - model: business logic
      - view: presentation of data, depends on display device
      - controller: receives events, executes actions, and returns a view to the user
  - business-logic layer
    - provides high level view of data and actions on data
      - often using an object data model
    - hides details of data storage schema
  - data access layer
    - interfaces between business logic layer and the underlying database
    - provides mapping from object model of business layer to relational model of database

Business Logic Layer

- Provides abstractions of entities
  - e.g. students, instructors, courses, etc
- Enforces business rules for carrying out actions
  - E.g. student can enroll in a class only if she has completed prerequisites, and has paid her tuition fees
- Supports workflows which define how a task involving multiple participants is to be carried out
  - E.g. how to process application by a student applying to a university
  - Sequence of steps to carry out task
  - Error handling
    - e.g. what to do if recommendation letters not received on time
  - Workflows discussed in Section 26.2
Object-Relational Mapping

- Allows application code to be written on top of object-oriented data model, while storing data in a traditional relational database
  - alternative: implement object-oriented or object-relational database to store object model
    - has not been commercially successful
- Schema designer has to provide a mapping between object data and relational schema
  - e.g. Java class `Student` mapped to relation `student`, with corresponding mapping of attributes
  - An object can map to multiple tuples in multiple relations
- Application opens a session, which connects to the database
- Objects can be created and saved to the database using `session.save(object)`
  - mapping used to create appropriate tuples in the database
- Query can be run to retrieve objects satisfying specified predicates

Object-Relational Mapping and Hibernate (Cont.)

- The **Hibernate** object-relational mapping system is widely used
  - public domain system, runs on a variety of database systems
  - supports a query language that can express complex queries involving joins
    - translates queries into SQL queries
  - allows relationships to be mapped to sets associated with objects
    - e.g. courses taken by a student can be a set in Student object
  - See book for Hibernate code example
- The **Entity Data Model** developed by Microsoft
  - provides an entity-relationship model directly to application
  - maps data between entity data model and underlying storage, which can be relational
  - Entity SQL language operates directly on Entity Data Model
Web Services

- Allow data on Web to be accessed using remote procedure call mechanism
- Two approaches are widely used
  - **Representation State Transfer (REST)**: allows use of standard HTTP request to a URL to execute a request and return data
    - returned data is encoded either in XML, or in **JavaScript Object Notation (JSON)**
  - **Big Web Services**:
    - uses XML representation for sending request data, as well as for returning results
    - standard protocol layer built on top of HTTP
    - See Section 23.7.3

Accessing SQL From a Programming Language

- **API** (application-program interface) for a program to interact with a database server
- Application makes calls to
  - Connect with the database server
  - Send SQL commands to the database server
  - Fetch tuples of result one-by-one into program variables
- Various tools:
  - **JDBC** (Java Database Connectivity) works with Java
  - **ODBC** (Open Database Connectivity) works with C, C++, C#, and Visual Basic. Other API's such as ADO.NET sit on top of ODBC
  - **Embedded SQL**
### Embedded SQL

- The SQL standard defines embeddings of SQL in a variety of programming languages such as C, C++, Java, Fortran, and PL/1.
- A language to which SQL queries are embedded is referred to as a **host language**, and the SQL structures permitted in the host language comprise **embedded SQL**.
- The basic form of these languages follows that of the System R embedding of SQL into PL/1.
- **EXEC SQL** statement is used to identify embedded SQL request to the preprocessor:
  
  ```sql
  EXEC SQL <embedded SQL statement >;
  ```

  **Note:** this varies by language:
  - In some languages, like COBOL, the semicolon is replaced with END-EXEC.
  - In Java embedding uses `# SQL { .... };`

### ODBC

- Open DataBase Connectivity (ODBC) standard
  - standard for application program to communicate with a database server.
  - Application program interface (API) to
    - open a connection with a database,
    - send queries and updates,
    - get back results.
- Applications such as GUI, spreadsheets, etc. can use ODBC.
JDBC

- **JDBC** is a Java API for communicating with database systems supporting SQL.
- JDBC supports a variety of features for querying and updating data, and for retrieving query results.
- JDBC also supports metadata retrieval, such as querying about relations present in the database and the names and types of relation attributes.
- Model for communicating with the database:
  - Open a connection
  - Create a “statement” object
  - Execute queries using the Statement object to send queries and fetch results
  - Exception mechanism to handle errors

JDBC

1. Connect to the database
2. Issue query
3. Process results
4. Close connection
JDBC Connection
(the hardest part)

• import java.sql.*
• Connection conn =
  DriverManager.getConnection("jdbc:oracle
   :thin:@claros.cs.purdue.edu:1524:strep",
   "clifton", "password");
  – Contents of first argument vary by DBMS
• http://docs.oracle.com/database/121/TDPJD/toc.htm
• http://docs.oracle.com/javase/7/docs/api/

Executing SQL

• Build a statement
  – ps = conn.prepareStatement("select dummy,
    ? From dual");
  – ps.setString(1, "stuff");
• Execute the statement
  – ps.executeQuery();
  – Also executes for update
Working with Results

• ResultSet rs = ps.executeQuery();
• While ( ! rs.isAfterLast() ) {
    i = rs.getInt(1);
    s = rs.getString(2);
    rs.next();
}
• Can also update, delete, insert rows in the ResultSet
  – Explicitly save back
Updates: SQL Insert/Update Statements

- Statement execute method
  - `boolean typeOfResult = stmt.execute("insert into tab values ("a", 3")");`
  - `typeOfResult` indicates if a ResultSet
- Better: Statement `executeUpdate` method
  - `int executeUpdate("insert into tab values('a', 3")");`
  - Returns number of rows inserted or updated

Approach 2: Update in place: Edit a ResultSet

- `PreparedStatement ps = conn.prepareStatement("select * from tab",
  ResultSet.TYPE_SCROLL_INSENSITIVE,
  ResultSet.CONCUR_UPDATABLE,
  ResultSet.CLOSE_CURSORS_AT_COMMIT);
- ResultSet `rs = stmt.executeQuery();`
- `rs.updateString("s", "Chris");`
  - Updates the FirstName column at the current row in the ResultSet
- `rs.updateRow();`
  - Saves updates to the current row to the database
- `rs.moveToInsertRow();`
  - Edit data using `rs.update...`
  - `rs.insertRow();`
Batch Updates

• stmt.addBatch("insert into tab values('a', 3)");
  stmt.addBatch("insert into tab values('c', 4)");
• stmt.executeBatch();

SQLException

• Almost any of these can throw a SQLException
  – Start with getMessage()…
  – Can do some interesting things with errors when they occur (e.g., violation of constraints)
SQL Injection

- Suppose query is constructed using
  - "select * from instructor where name = "" + name + ""
- Suppose the user, instead of entering a name, enters:
  - 'X' or 'Y' = 'Y'
- then the resulting statement becomes:
  - "select * from instructor where name = "" + 'X' or 'Y' = 'Y" + ""
  - which is:
    - select * from instructor where name = 'X' or 'Y' = 'Y'
  - User could have even used
    - 'X'; update instructor set salary = salary + 10000; --
- Prepared statement internally uses:
  "select * from instructor where name = 'X' or \Y' = \Y"'
- **Always use prepared statements, with user inputs as parameters**
- Is the following prepared statement secure?
  - conn.prepareStatement("select * from instructor where name = "" + name + "")