

$$\underline{AB} \rightarrow C$$

$$\underline{AB} \rightarrow ABC$$

$$\boxed{\underline{K} \rightarrow R} \quad \text{1NF}$$

LECTURE 19 OF 42

Intro to Web Databases Discussion: Online DBs

Friday, 07 March 2008

William H. Hsu

Department of Computing and Information Sciences, KSU

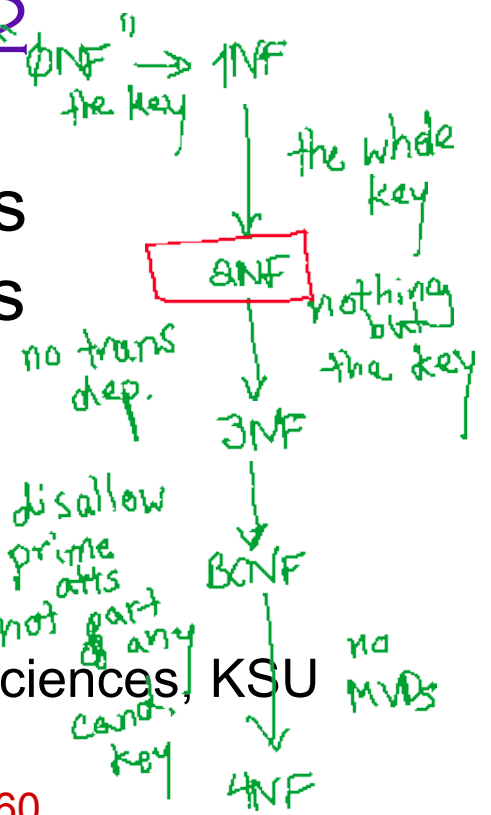
KSOL course page: <http://snipurl.com/va60>

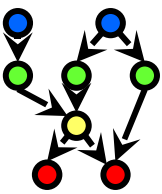
Course web site: <http://www.kddresearch.org/Courses/Spring-2008/CIS560>

Instructor home page: <http://www.cis.ksu.edu/~bhsu>

Reading for Next Class:

Second half of Chapter 8, Silberschatz *et al.*, 5th edition





FIRST NORMAL FORM

- Domain is atomic if its elements are considered to be indivisible units
 - ★ Examples of non-atomic domains:
 - ⇒ Set of names, composite attributes
 - ⇒ Identification numbers like CS101 that can be broken up into parts
- A relational schema R is in first normal form if the domains of all attributes of R are atomic
- Non-atomic values complicate storage and encourage redundant (repeated) storage of data

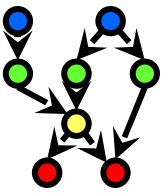
Day	Dishes
Mon	Main Entrée Side Dessert

- ★ Example: Set of accounts stored with each customer, and set of owners stored with each account
- ★ We assume all relations are in first normal form (and revisit this in Chapter 9)

Day	M	E	S	D

Day	Slot/type	Item
Mon	Main	

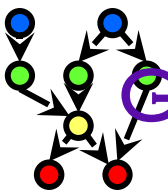




FIRST NORMAL FORM (CONT'D)

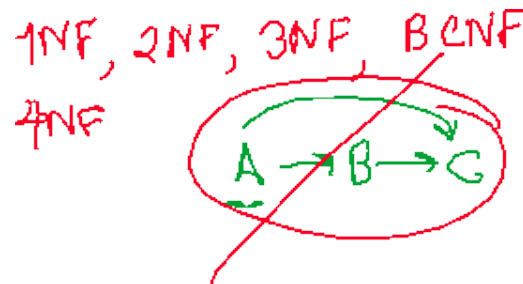
- Atomicity is actually a property of how the elements of the domain are used.
 - ★ Example: Strings would normally be considered indivisible
 - ★ Suppose that students are given roll numbers which are strings of the form *CS0012* or *EE1127*
 - ★ If the first two characters are extracted to find the department, the domain of roll numbers is not atomic.
 - ★ Doing so is a bad idea: leads to encoding of information in application program rather than in the database.

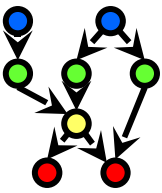




GOAL – DEVISE A THEORY FOR THE FOLLOWING

- Decide whether a particular relation R is in “good” form.
- In the case that a relation R is not in “good” form, decompose it into a set of relations $\{R_1, R_2, \dots, R_n\}$ such that
 - * each relation is in good form
 - * the decomposition is a lossless-join decomposition
- Our theory is based on:
 - * functional dependencies
 - * multivalued dependencies

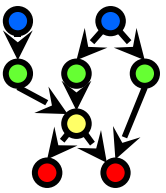




FUNCTIONAL DEPENDENCIES

- Constraints on the set of legal relations.
- Require that the value for a certain set of attributes determines uniquely the value for another set of attributes.
- A functional dependency is a generalization of the notion of a *key*.





FUNCTIONAL DEPENDENCIES (CONT.)

- Let R be a relation schema

$$\alpha \subseteq R \text{ and } \beta \subseteq R$$

- The functional dependency

$$\alpha \rightarrow \beta$$

holds on R if and only if for any legal relations $r(R)$, whenever any two tuples t_1 and t_2 of r agree on the attributes α , they also agree on the attributes β . That is,

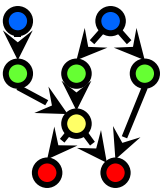
$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

- Example: Consider $r(A, B)$ with the following instance of r .

1	5
3	7

- On this instance, $A \rightarrow B$ does **NOT** hold, but $B \rightarrow A$ does hold.





FUNCTIONAL DEPENDENCIES (CONT.)

- K is a superkey for relation schema R if and only if $K \rightarrow R$
- K is a candidate key for R if and only if
 - ★ $K \rightarrow R$, and
 - ★ for no $\alpha \subset K$, $\alpha \rightarrow R$
- Functional dependencies allow us to express constraints that cannot be expressed using superkeys. Consider the schema:

bor_loan = (customer_id, loan_number, amount).

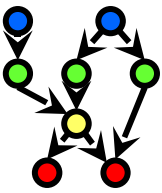
We expect this functional dependency to hold:

loan_number → amount

but would not expect the following to hold:

amount → customer_name

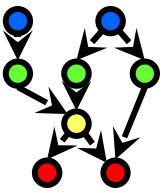




USE OF FUNCTIONAL DEPENDENCIES

- We use functional dependencies to:
 - ★ test relations to see if they are legal under a given set of functional dependencies.
 - ⇒ If a relation r is legal under a set F of functional dependencies, we say that r satisfies F .
 - ★ specify constraints on the set of legal relations
 - ⇒ We say that F holds on R if all legal relations on R satisfy the set of functional dependencies F .
- Note: A specific instance of a relation schema may satisfy a functional dependency even if the functional dependency does not hold on all legal instances.
 - ★ For example, a specific instance of *loan* may, by chance, satisfy *amount* → *customer_name*.

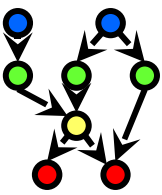




FUNCTIONAL DEPENDENCIES (CONT.)

- A functional dependency is trivial if it is satisfied by all instances of a relation
 - ★ Example:
 - ⇒ $customer_name, loan_number \rightarrow customer_name$
 - ⇒ $customer_name \rightarrow customer_name$
 - ★ In general, $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$

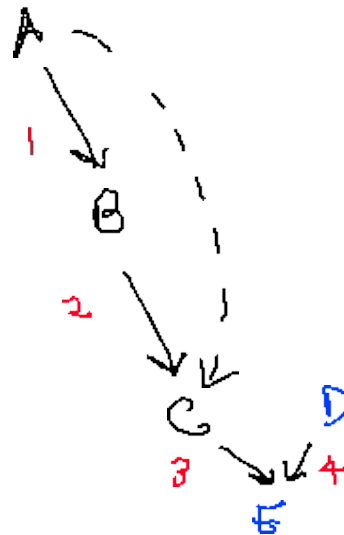


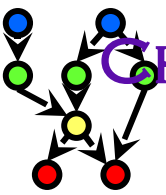


CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES

- Given a set F set of functional dependencies, there are certain other functional dependencies that are logically implied by F .
 - ★ For example: If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that $A \rightarrow C$
- The set of all functional dependencies logically implied by F is the *closure* of F .
- We denote the *closure* of F by F^+ .
- F^+ is a superset of F .

$$\frac{A \rightarrow C \quad CD \rightarrow E}{AD \rightarrow E}$$

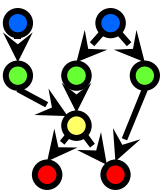




CHAPTER 8: APPLICATION DESIGN AND DEVELOPMENT

- User Interfaces and Tools
- Web Interfaces to Databases
- Web Fundamentals
- Servlets and JSP
- Building Large Web Applications
- Triggers
- Authorization in SQL
- Application Security

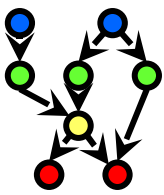




USER INTERFACES AND TOOLS

- Most database users do *not* use a query language like SQL.
 - ★ Forms
 - ★ Graphical user interfaces
 - ★ Report generators
 - ★ Data analysis tools (see Chapter 18)
- Many interfaces are Web-based
- Back-end (Web server) uses such technologies as
 - ★ Java servlets
 - ★ Java Server Pages (JSP)
 - ★ Active Server Pages (ASP)

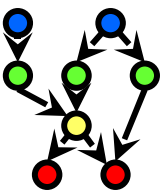




THE WORLD WIDE WEB

- 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





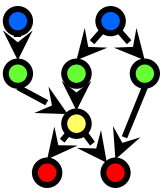
A FORMATTED REPORT

Acme Supply Company, Inc. Quarterly Sales Report

Period: Jan. 1 to March 31, 2005

Region	Category	Sales	Subtotal
North	Computer Hardware	1,000,000	1,500,000
	Computer Software	500,000	
	All categories		
South	Computer Hardware	200,000	600,000
	Computer Software	400,000	
	All categories		
Total Sales			2,100,000





WEB INTERFACES TO DATABASES

Why interface databases to the Web?

2. 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
 - ★ Examples: banks, airline and rental car reservations, university course registration and grading, an so on.





WEB INTERFACES TO DATABASE (CONT.)

1. Dynamic generation of documents

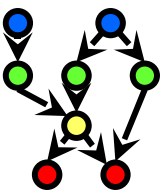
* Limitations of static HTML documents

- ⇒ Cannot customize fixed Web documents for individual users.
- ⇒ Problematic to update Web documents, especially if multiple Web documents replicate data.

* Solution: Generate Web documents dynamically from data stored in a database.

- ⇒ Can tailor the display based on user information stored in the database.
 - ◆ E.g. tailored ads, tailored weather and local news, ...
- ⇒ Displayed information is up-to-date, unlike the static Web pages
 - ◆ E.g. stock market information, ..





UNIFORM RESOURCES LOCATORS

- In the Web, functionality of pointers is provided by Uniform Resource Locators (URLs).

- URL example:

<http://www.bell-labs.com/topics/book/db-book>

- * 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.

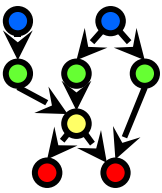
- The local identification can be:

- ⇒ 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>

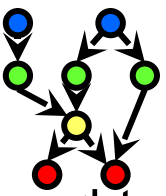




HTML AND HTTP

- HTML provides formatting, hypertext link, and image display features.
- 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



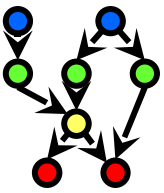


SAMPLE HTML SOURCE TEXT

```
<html> <body>
<table border cols = 3>
  <tr> <td> A-101 </td> <td> Downtown </td> <td> 500 </td> </tr>
  ...
</table>
<center> The <i>account</i> relation </center>

<form action="BankQuery" method=get>
  Select account/loan and enter number <br>
  <select name="type">
    <option value="account" selected> Account
    <option value="Loan">          Loan
  </select>
  <input type=text size=5 name="number">
  <input type=submit value="submit">
</form>
</body> </html>
```





DISPLAY OF SAMPLE HTML SOURCE

A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900

The *account* relation

Select account/loan and enter number

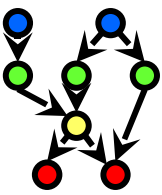
Account



⋮

submit

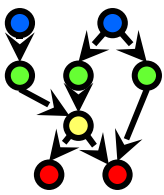




CLIENT SIDE SCRIPTING AND APPLETS

- 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

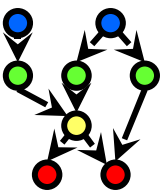




CLIENT SIDE SCRIPTING AND SECURITY

- 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.

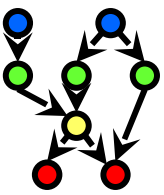




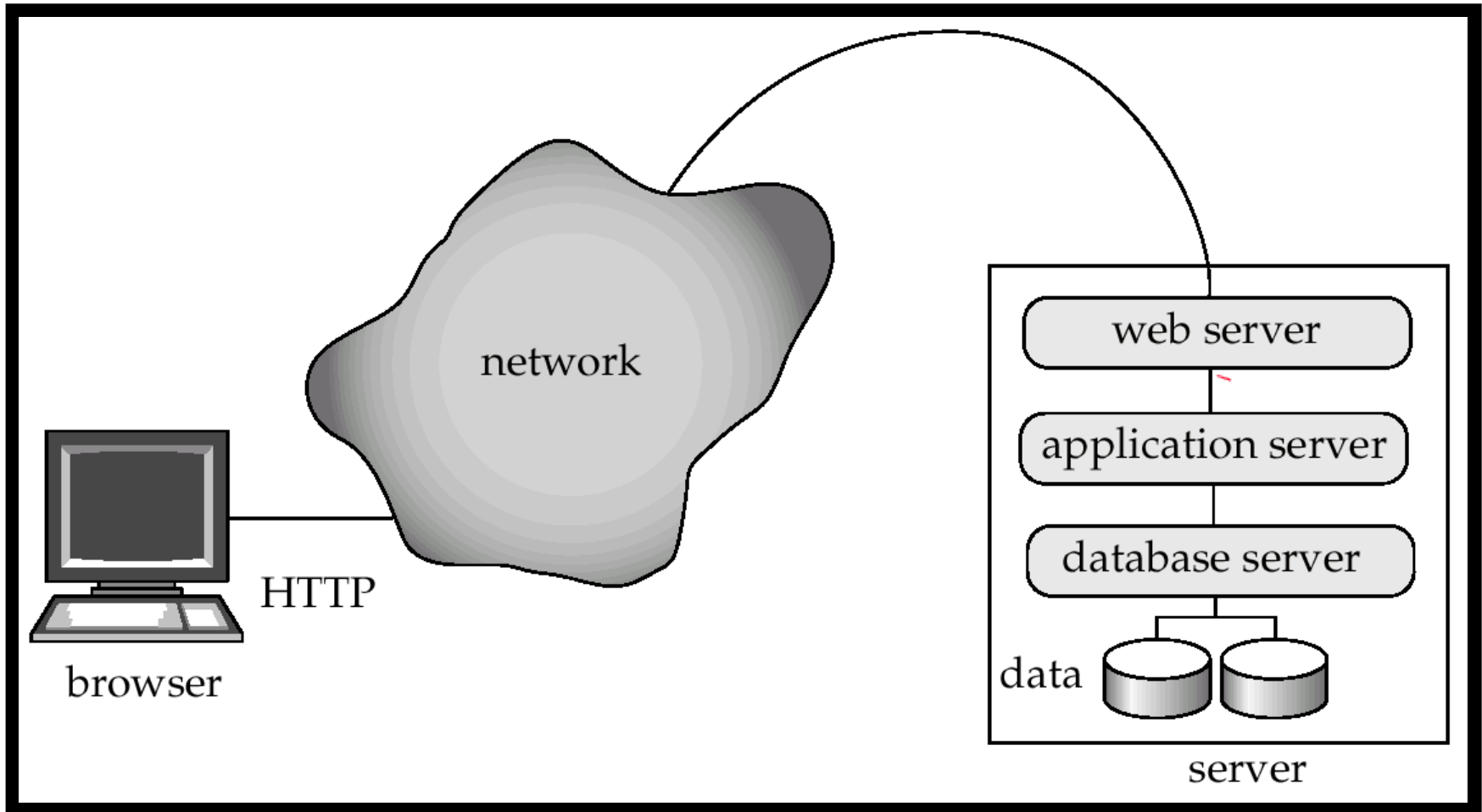
WEB SERVERS

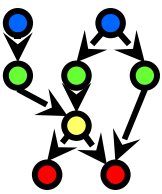
- A Web server can easily serve as a front end to a variety of information services.
- The document name in a URL may identify an executable program, that, when run, generates a HTML document.
 - ★ When a HTTP server receives a request for such a document, it executes the program, and sends back the 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





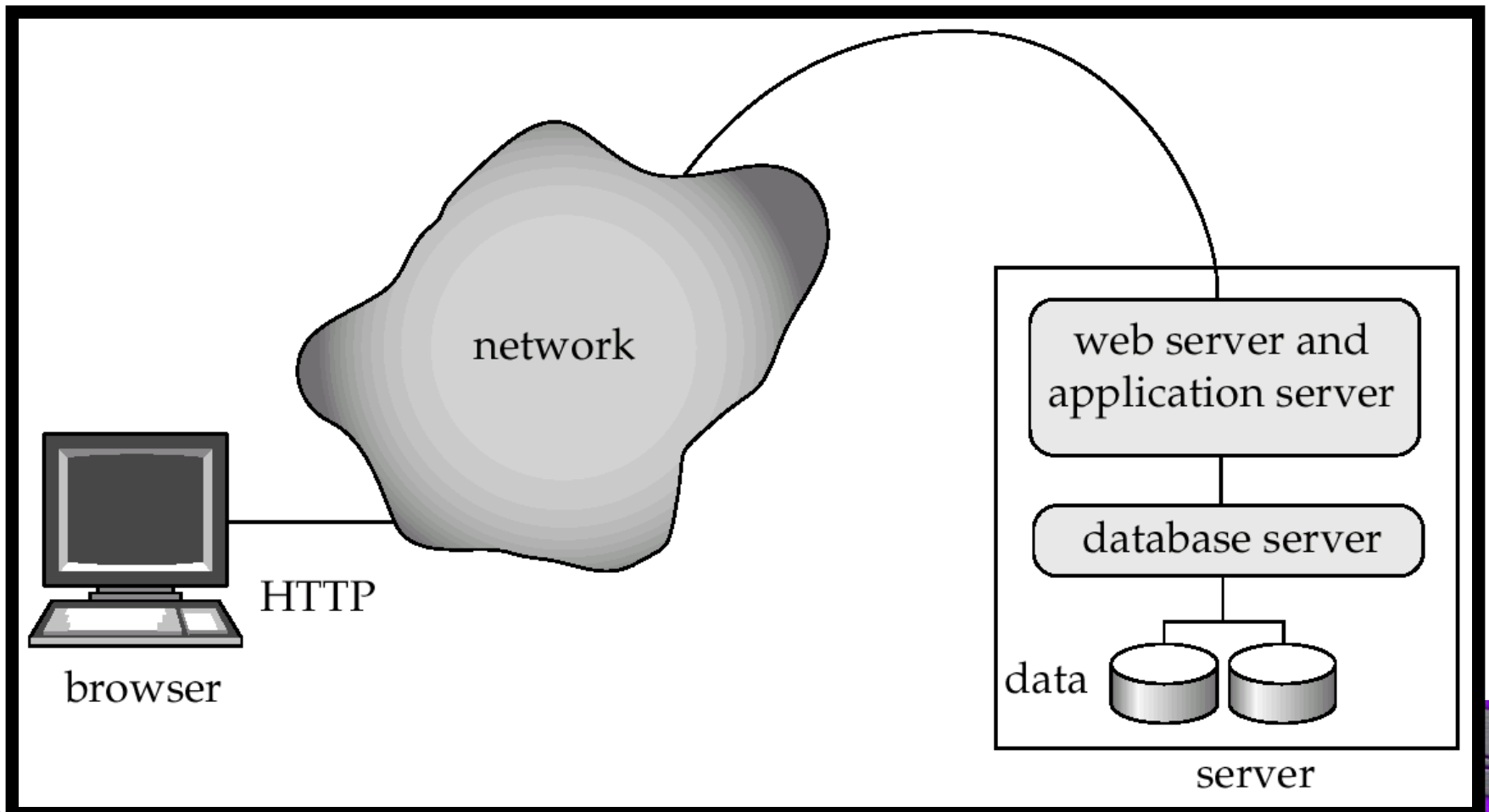
THREE-TIER WEB ARCHITECTURE

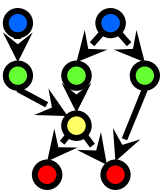




TWO-TIER WEB ARCHITECTURE

- Multiple levels of indirection have overheads
- ★ Alternative: two-tier architecture

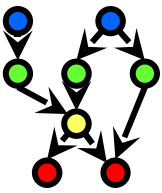




HTTP AND SESSIONS

- 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

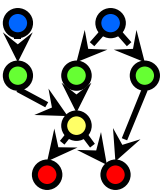




SESSIONS AND COOKIES

- A cookie is a small piece of text containing identifying information
 - ★ Sent by server to browser on first interaction
 - ★ 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

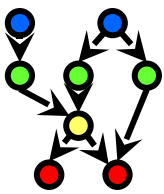




SERVLETS

- Java Servlet specification defines an API for communication between the Web server and application program
 - ★ E.g. methods to get parameter values and to send HTML text back to client
- Application program (also called a servlet) is loaded into the Web server
 - ★ Two-tier model
 - ★ Each request spawns a new thread in the Web server
 - ⇒ thread is closed once the request is serviced
- Servlet API provides a `getSession()` method
 - ★ Sets a cookie on first interaction with browser, and uses it to identify session on further interactions
 - ★ Provides methods to store and look-up per-session information
 - ⇒ E.g. user name, preferences, ..

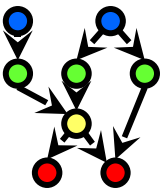




EXAMPLE SERVLET CODE

```
Public class BankQuery(Servlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
    result)
        throws ServletException, IOException {
        String type = request.getParameter("type");
        String number = request.getParameter("number");
        ...code to find the loan amount/account balance ...
        ...using JDBC to communicate with the database..
        ...we assume the value is stored in the variable balance
        result.setContentType("text/html");
        PrintWriter out = result.getWriter( );
        out.println("<HEAD><TITLE>Query Result</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("Balance on " + type + number + "=" + balance);
        out.println("</BODY>");
        out.close ( );
    }
}
```

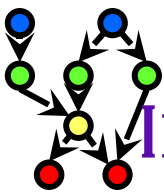




SERVER-SIDE SCRIPTING

- Server-side scripting simplifies the task of connecting a database to the Web
 - ★ Define a 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, Server-side Javascript, ColdFusion Markup Language (cfml), PHP, Jscript
 - ★ General purpose scripting languages: VBScript, Perl, Python

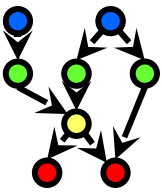




IMPROVING WEB SERVER PERFORMANCE

- Performance is an issue for popular Web sites
 - ★ May be accessed by millions of users every day, thousands of requests per second at peak time
- Caching techniques used to reduce cost of serving pages by exploiting commonalities between requests
 - ★ At the server site:
 - ⇒ Caching of JDBC connections between servlet requests
 - ⇒ Caching results of database queries
 - ◆ Cached results must be updated if underlying database changes
 - ⇒ Caching of generated HTML
 - ★ At the client's network
 - ⇒ Caching of pages by Web proxy

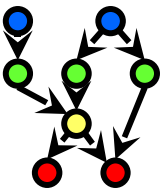




TRIGGERS

- A **trigger** is a statement that is executed automatically by the system as a side effect of a modification to the database.
- To design a trigger mechanism, we must:
 - ★ Specify the conditions under which the trigger is to be executed.
 - ★ Specify the actions to be taken when the trigger executes.
- Triggers introduced to SQL standard in SQL:1999, but supported even earlier using non-standard syntax by most databases.

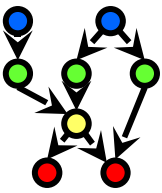




TRIGGER EXAMPLE

- Suppose that instead of allowing negative account balances, the bank deals with overdrafts by
 - ★ setting the account balance to zero
 - ★ creating a loan in the amount of the overdraft
 - ★ giving this loan a loan number identical to the account number of the overdrawn account
- The condition for executing the trigger is an update to the *account* relation that results in a negative *balance* value.





TRIGGER EXAMPLE IN SQL: 1999

```
create trigger overdraft-trigger after update on account  
referencing new row as nrow  
           for each row  
when nrow.balance < 0  
begin atomic  
    insert into borrower  
        (select customer-name, account-number  
         from depositor  
         where nrow.account-number =  
                depositor.account-number);  
    insert into loan values  
        (n.row.account-number, nrow.branch-name,  
         – nrow.balance);  
    update account set balance = 0  
    where account.account-number = nrow.account-number  
end
```

