Database systems

- **database: a structured collection of data**
- **provides an abstract view of data**
  - separated from how it's stored in a file system
  - analogous to how file systems abstract from physical devices
- **uniform access to information**
- **provides centralized control**
- **can guarantee important properties**
  - consistency
  - security
  - integrity
- **can reduce redundancy, provide speed, efficiency**

CRUD

- **basic database operations:**

- **Create**
  - create a brand new record

- **Read**
  - read/ retrieve an existing record

- **Update**
  - change / modify / update all or part of an existing record

- **Delete**
  - guess what

ACID

- **critical properties of a database system:**

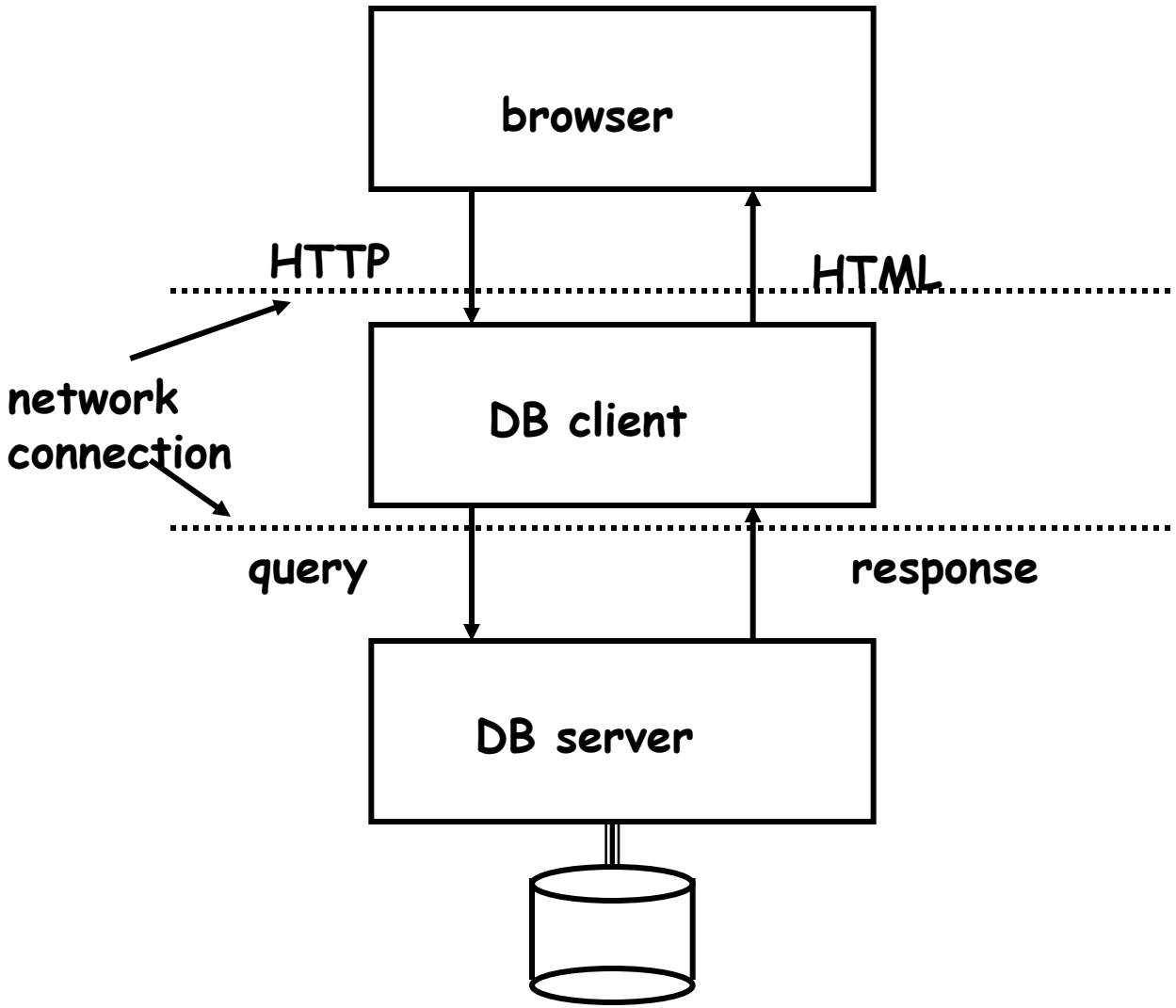
- **Atomicity**
  - all or nothing: all steps of a transaction are completed
  - no partially completed transactions

- **Consistency**
  - each transaction maintains consistency of whole database

- **Isolation**
  - effects of a transaction not visible to other transactions until committed

- **Durability**
  - changes are permanent, survive system failure
  - consistency guaranteed

Typical database system organization



Types of database systems

- **ordinary files**
  - sometimes ok, but this is not a database except in informal sense
    e.g., doesn't guarantee the ACID properties
- **relational / SQL**
  - MySQL, SQLite, Postgres, Oracle, DB2, …
  - tables, rows, attributes
  - very structured, organized
- **"NoSQL" (more accurately "non-relational")**
  - MongoDB, CouchDB, ...
  - collections, documents, fields
  - more intuitive, more flexible for some things
  - don't provide all the mechanisms and guarantees of SQL databases
  - may run better on clusters of servers
- **key-value & column stores**
  - Redis, Berkeley DB, memcached, BigTable, …

Relational Database Management Systems

- **e.g.: MySQL, Postgres, SQLite, Oracle, DB2, …**
- **a database is a collection of tables**
- **each table has a fixed number of columns**
  - each column is an "attribute" common to all rows
- **and a variable number of rows**
  - each row is a "record" that contains data

isbn	*title*	*author*	*price*
1234	*MySQL*	*DuBois*	*49.95*
4321	*TPOP*	*K & P*	*24.95*
2468	*Ruby*	*Flanagan*	*79.99*
2467	*Java*	*Flanagan*	*89.99*
2466	*Javascript*	*Flanagan*	*99.99*
1357	*Networks*	*Peterson*	*105.00*
1111	*Practical Ethics*	*Singer*	*25.00*
4320	*C Prog Lang*	*K & R*	*40.00*

Relational model

- **simplest database has one table holding all data**
  - *e.g.*, Excel spreadsheet
- **relational model: data in separate tables "related" by common attributes**
  - *e.g.*, **custid** in **custs** matches **custid** in **sales**
- **schema: content and structure of the tables**

```
books
     isbn    title    author    price
custs
     custid    name    adr
sales
     isbn    custid    date    price    qty
stock
     isbn    count
```

- **extract desired info by queries**
- **query processing figures out what info comes from what tables, extracts it efficiently**

Sample database

- **books** [isbn, title, author, price]

```
1234   MySQL              DuBois     49.95
4321   TPOP               K & P      24.95
2468   Ruby               Flanagan   79.99
2467   Java               Flanagan   89.99
```

- **custs** [custid, name, adr]

```
11    Brian    Princeton
22    Bob      Princeton
33    Bill     Redmond
44    Bob      Palo Alto
```

- **sales** [isbn, custid, date, price, qty]

```
4321    11     2012-02-28     45.00      1
2467    22     2012-01-01     60.00     10
2467    11     2012-02-05     57.00      3
4321    33     2012-02-05     45.00      1
```

- **stock** [isbn, count]

```
1234   100
4321    20
2468     5
2467     0
```

Retrieving data from a single table

- **SQL ("Structured Query Language") is the standard language for expressing queries**
  - all major database systems support it

- **general format:**

  select ***column-names*** **from** ***tables*** **where** ***condition*** **;**

```
select * from books;
select name, adr from custs;
select title, price from books where price > 50;
select * from books where author = "Flanagan";
select author, title from books where author like "F%";
select author, title from books order by author;
select author, count(*) from books group by author;
select author, count(*) as n from books group by author
    order by n desc;
```

- **result is a table**

Multiple tables and joins

- **if desired info comes from multiple tables, this implies a "join" operator to relate data in different tables**
  - in effect join makes a big table for later selection

```
select title, count from books, stock
   where books.isbn = stock.isbn;

select * from books, sales
   where books.isbn = sales.isbn
     and books.author like "F%";

select custs.name, books.title
  from books, custs, sales
    where custs.id = sales.custid
      and sales.isbn = books.isbn;

select price, count(*) as count from books
   where author like 'F%'
     group by author order by count desc;
```

MySQL

- **open source (?) relational database system**
  www.mysql.com
- **"LAMP"**
  - **L**inux
  - **A**pache
  - **M**ySQL
  - **P***: Perl, Python, PHP
- **command-line interface:**
  - connect to server using command interface

    ```
    mysql –h publicdb –u bwk –p
    ```

  - type commands, read responses

    ```
    show databases;
    use bwk;
    show tables;
    select now(), version(), user();
    source cmdfile;
    ```

- **these commands are specific to MySQL**

Creating and loading a table

- **create table**

```
create table books (
    isbn varchar(15) primary key,
    title varchar(35), author varchar(20),
    price decimal(10,2)
);
```

- **load table from file (tab-separated text)**

```
load data local infile "books" into table books
     fields terminated by "\t"
     ignore 1 lines;
```

- **fields have to be left justified.**
- **"terminated by" parameter must be a single character**
  - not whitespace: multiple blanks are NOT treated as single separator

- **can also insert one record at a time**

```
insert into books values('2464','AWK','Flanagan','89.99');
```

Other statements

- **generic SQL**
  - ought to be the same for all db systems
  - (though they are not always)

```
insert into sales
   values('1234','44','2008-03-06','27.95');
update books set price = 99.99
    where author = "Flanagan";
delete from books where author = "Singer";
```

- **MySQL-specific**
  - other db's have analogous but different statements

```
use bwk;
show tables;
describe books;
drop tables if exists books, custs;
```

SQLite: an alternative (www.sqlite.org)

- **small, fast, simple, embeddable**
  - no configuration
  - no server
  - single cross-platform database file
- **most suitable for**
  - embedded devices (cellphones)
  - web sites with modest traffic & rapid processing
    <100K hits/day, 10 msec transaction times
  - ad hoc file system or format replacement
  - internal or temporary databases
- **probably not right for**
  - large scale client server
  - high volume web sites
  - gigabyte databases
  - high concurrency
- **"SQLite is not designed to replace Oracle. It is designed to replace fopen()."**

Program interfaces to MySQL

- **original and basic interface is in *C***
  - about 50 functions
  - other interfaces build on this
  - most efficient access though query complexity is where the time goes
  - significant complexity in managing storage for query results
- **API's exist for most other languages**
  - Perl, Python, PHP, Ruby, ...
  - C++, Java, …
  - can use MySQL from Excel, etc., with ODBC module
- **basic structure for API's is**

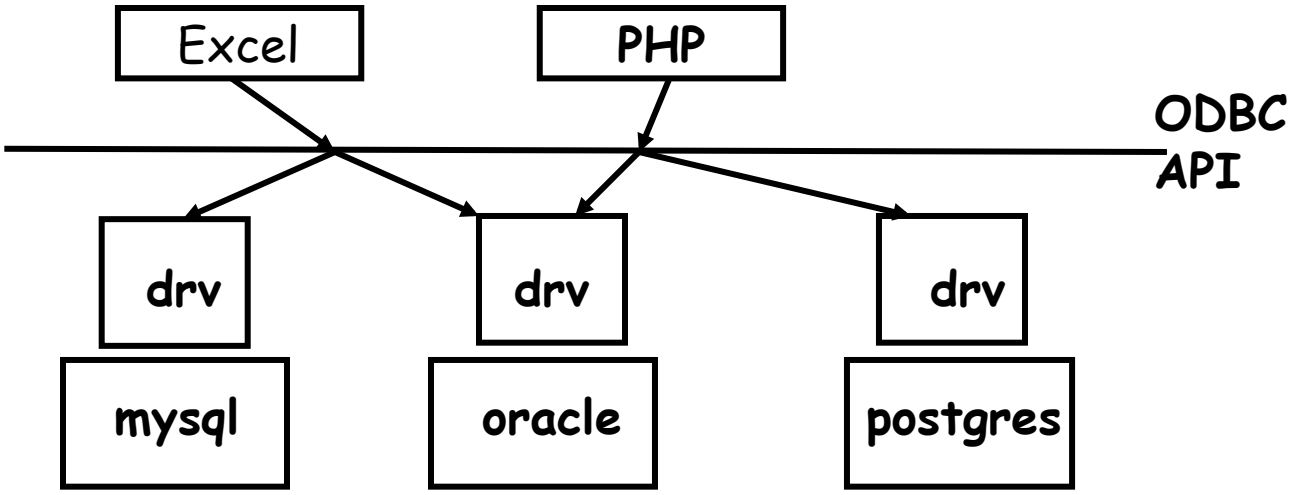
```
db_handle = connect to database
repeat {
        stmt_handle = prepare an SQL statement
        execute (stmt_handle)
        fetch result
} until tired
disconnect (db_handle)
```

Python version

```
import sys, fileinput, _mysql
def main():
   db = _mysql.connect(host="publicdb.cs.princeton.edu",
         user="bwk", db="bwk", passwd="xx")
   print "Enter query: ",
   q = sys.stdin.readline()
   while q != '':
      db.query(q)
      res = db.store_result()
      r = res.fetch_row()
      while len(r) != 0:
         print r
         r = res.fetch_row()
      print "Enter query: ",
      q = sys.stdin.readline()
main()
```

ODBC, JDBC, and all that

- **ODBC ("open database connectivity")**
  - Microsoft standard interface between applications and databases
  - API provides basic SQL interface
  - driver does whatever work is needed to convert
  - underlying database has to provide basic services
  - used for applications like Excel, Visual Basic, C/C++, ...
  - drivers exist for all major databases
  - makes applications relatively independent of specific database being used
- **JDBC is the same thing for Java**
  - passes calls through to ODBC drivers or other database software



MySQL access from Java (Connector/J JDBC package)

```
import java.sql.*;

public class mysql {
  public static void main(String args[]) {
   String url = "jdbc:mysql://publicdb.cs.princeton.edu/bwk";
   try {
      Class.forName("com.mysql.jdbc.Driver");
   } catch(java.lang.ClassNotFoundException e) {
      System.err.print("ClassNotFoundException: " + e.getMessage());
   }
   try {
      Connection con = DriverManager.getConnection(url, "bwk", ”xx");
      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery("select * from books");
      while (rs.next())
         System.out.println(rs.getString("title") + " "
                                    + rs.getString("author"));
      stmt.close();
      con.close();
   } catch(SQLException ex) {
      System.err.println("SQLException: " + ex.getMessage());
   }
  }
}
```

SQL injection

- one of the most common attacks on web servers
- malicious SQL statements within queries
  - can reveal database contents
  - and perhaps modify contents or do other damage

- if text from a form is handed directly to SQL engine,
  - the database is vulnerable

```
select * from books where author = 'something from a form';

select * from books where author = 'x';
 update books set price = $1.00 where author like 'K%'; --';
```

Defenses

- **always watch out for this**
- **don't try to roll your own with regular expressions**
  - it's too hard to get it right
- **use parameterized queries**
  - query is processed before insertion

```
cmd = "update people set name=%s where id=%s"
db.execute(cmd, (name, id))
```

- **details vary among systems (e.g., %s for MySQL, ? for SQlite)**

- **Django and other frameworks generally do this for you**

- **www.unixwiz.net/techtips/sql-injection.html**
- **www.bobby-tables.com**

Database design

- **two different possible table structures:**

```
books
   isbn   title   author   price
booktitle, bookauthor, bookprice
   isbn  title
   isbn  author
   isbn  price
```

- **they need different SQL queries:**

```
select title, author, price from books;
select title, author, price
  from booktitle, bookauthor,bookprice
    where booktitle.isbn = bookauthor.isbn
     and bookauthor.isbn = bookprice.isbn;
```

- **most of the program should be independent of the specific table organization**
  - shouldn't know or care which one is being used

```
getList(title, author, price)
```

"NoSQL" databases

- **intended for scalability, performance**
  - can be distributed easily
- **may not have fixed schema**
  - easier to reorganize or augment data organization than with SQL
- **no join operator: you have to do it yourself**
- **may not guarantee ACID properties**
  - "eventually consistent" instead
- **no standardization**
  - different access methods for different db's

MongoDB example (from flaskr)

```
from pymongo import Connection
db = Connection()['dbfile']
blog = db['blog']

def show_entries():
    entries = [dict(title=cur['title'], text=cur['text'])
                        for cur in blog.find()]
    return render_template('show_entries.html', entries=entries)

def add_entry():
    blog.insert({"title": request.form['title'],
                 "text": request.form['text']}) # BUG: injection?
    return redirect(url_for('show_entries'))
def clear():
    blog.remove()
    return redirect(url_for('show_entries'))
```

[see http://openmymind.net/2011/3/28/The-Little-MongoDB-Book/