Insight into

DBMS:

Design and Implementation

Overview of the projects

孔令波
mlinking@126.com
+86 15010255486

SAP Labs Waterloo

Established Database R&D site going back 35+ years

~220 people

- Majority SAP Database Research and Development
- Work on HANA, Cloud-native databases, distributed databases, Edge Computing

Co-op & Grad Student intern programs

Strong academic collaborations

We are hiring!

- Full-time positions
- Student jobs - master/bachelor theses, graduate internships

ht

ht

ht

Outline of the chapters covered

- **Introduction**
- **Overview of the projects**
- **Demonstration of Development environment**
  - Watch and practice by yourself
- **My understanding about (R)DBMS**
  - History and D&I
- **SQL translation with 2 conversions**
  - SQL → RA (Relational Algebra)
  - RA → Sequence of File operations
- **Transaction control**
- **Deeper**
  - File, (R)DBMS, ERP, DW, Big Data (No SQL, SQL again)
  - SQL on MPP and Hadoop (Greenplum, **HAWQ**)

Overview of the projects

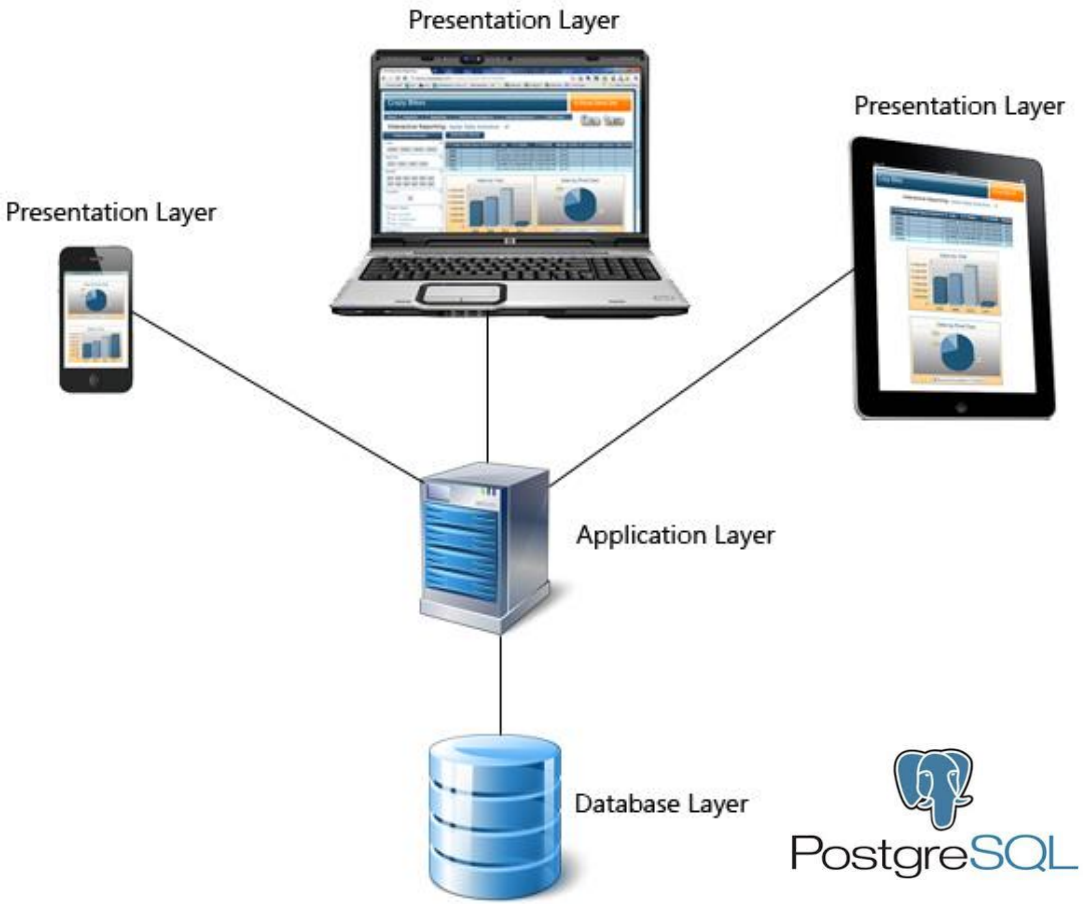
Overview of the projects

- **Basic (70)**
  1. Index File - 10
  2. Simulate multiple users - 10
  3. Try a Syntax parsing example - 20
  4. Lock table management - 20
  5. Debug **PostgreSQL/Greenplum/HAWQ** - 10
- **1 Web-based Application project (20)**
  - **Java/JSP/PHP/Python/R + PostgreSQL/Greenplum/HAWQ** - R.D.I.P
- **Advanced (10) PostgreSQL/Greenplum/HAWQ**
  - [transaction mechanism][SQL execution]

amazon JD.COM Alibaba Group 阿里巴巴集团 支付宝 Alipay.com CAINIAO菜鸟网络 天猫 TMALL.COM 淘宝网 Taobao.com 中国建设银行 China Construction Bank

Modern DBMS is important for IT age

- As backend service provider
  - 3 tier architecture is popular in many applications



Apache SOFTWARE FOUNDATION

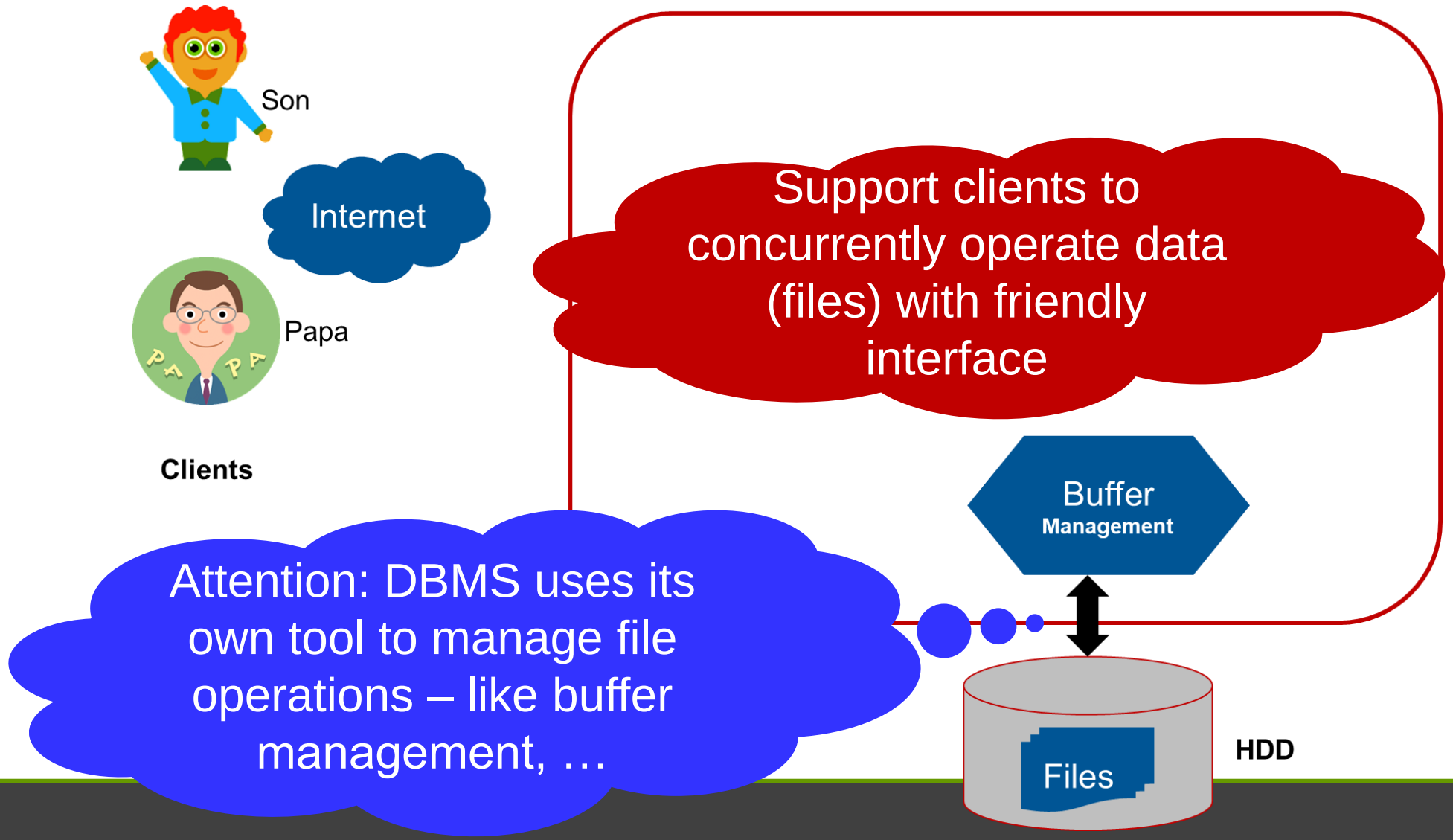
PostgreSQL ORACLE Microsoft SQL Server MySQL

By Modern DBMS? - Properties

- **Core is definitely**
  - Data Management (IDUS)
- **Top goal**
  - Concurrent data management for many users in a server or a distributed system
- **With 2 other properties**
  1. **Concurrent Data Management** by using files (which are supported by OS) – Deadlock and Data inconsistency
  2. **Friendly/Flexible data operations f**
     - Define specific data files
     - Insert, Delete, Update, Select s
     - Support some statistics
     - Provide flexible way for users to d
     - …

Select fname,lname
From employee, department
Where dno=dnumber
and
dname = 'Research'

Sketch of modern DBMS (More details later)



You can start some projects ASAP ba so far

Because related skills are easy for you to follow

My responsibility is to focus on the D&I of the core, and checking your work

- **Basic (70)**
  1. Index File
  2. Simulate multiple users
  3. Try a Syntax parsing example
  4. Lock table management
  5. Debug PostgreSQL/Greenplu
- **1 Web-based Application project (20)**
  - Java/JSP/PHP/Python/R + PostgreSQL/Greenplum/HAWQ - R.D.I.P
- **Advanced (10) in PostgreSQL/Greenplum/HAWQ**
  - [transaction mechanism][SQL execution]

Project requirements

Project 1: Index File [10 pts]

- You all have learned how to use file to store and access data – in fact this is the early version of Data Management System

- If the tuples/records are not so huge – 10 thousands, the performance seems OK. However millions of thousands of xxx? – Index File is needed!
- By index, you have learned many index data structures – AVL tree (named after inventors Adelson-Velsky and Landis), **B tree** (Balanced), KD-tree (K-dimensional) etc. Your practice is MM (Main Memory) version!

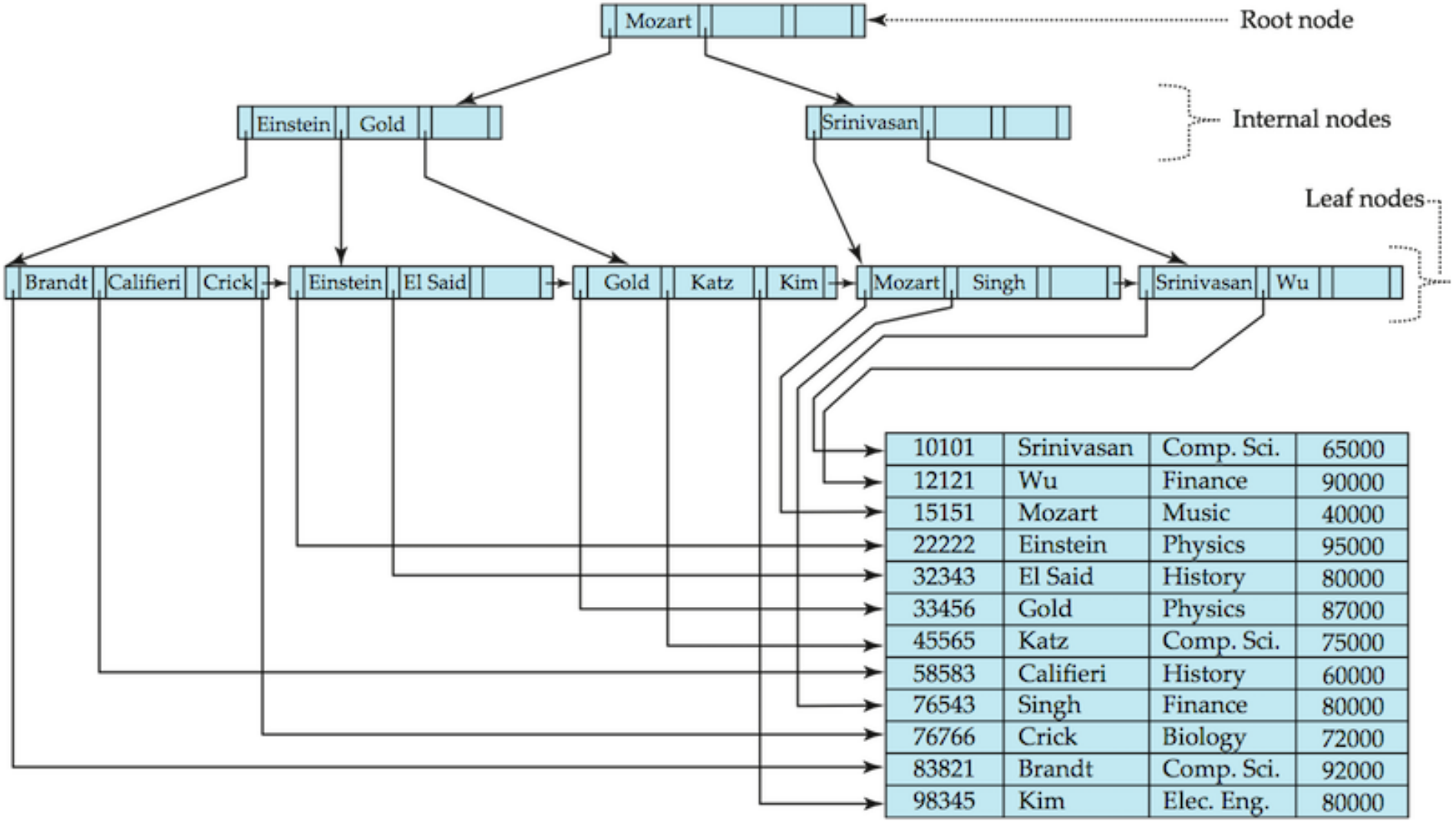
You are given

- On-line retails – 541909 records
  - https://archive.ics.uci.edu/ml/datasets/online+retail

	A	B	C	D	E	F	G	H
1	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
2	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010/12/1 8:26	2.55	17850	United Kingdom
3	536365	71053	WHITE METAL LANTERN	6	2010/12/1 8:26	3.39	17850	United Kingdom
4	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010/12/1 8:26	2.75	17850	United Kingdom
5	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010/12/1 8:26	3.39	17850	United Kingdom
6	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010/12/1 8:26	3.39	17850	United Kingdom
7	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010/12/1 8:26	7.65	17850	United Kingdom
8	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010/12/1 8:26	4.25	17850	United Kingdom
9	536366	22633	HAND WARMER UNION JACK	6	2010/12/1 8:28	1.85	17850	United Kingdom
10	536366	22632	HAND WARMER RED POLKA DOT	6	2010/12/1 8:28	1.85	17850	United Kingdom
11	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010/12/1 8:34	1.69	13047	United Kingdom
12	536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	2010/12/1 8:34	2.1	13047	United Kingdom
13	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	2010/12/1 8:34	2.1	13047	United Kingdom

We are interested to know

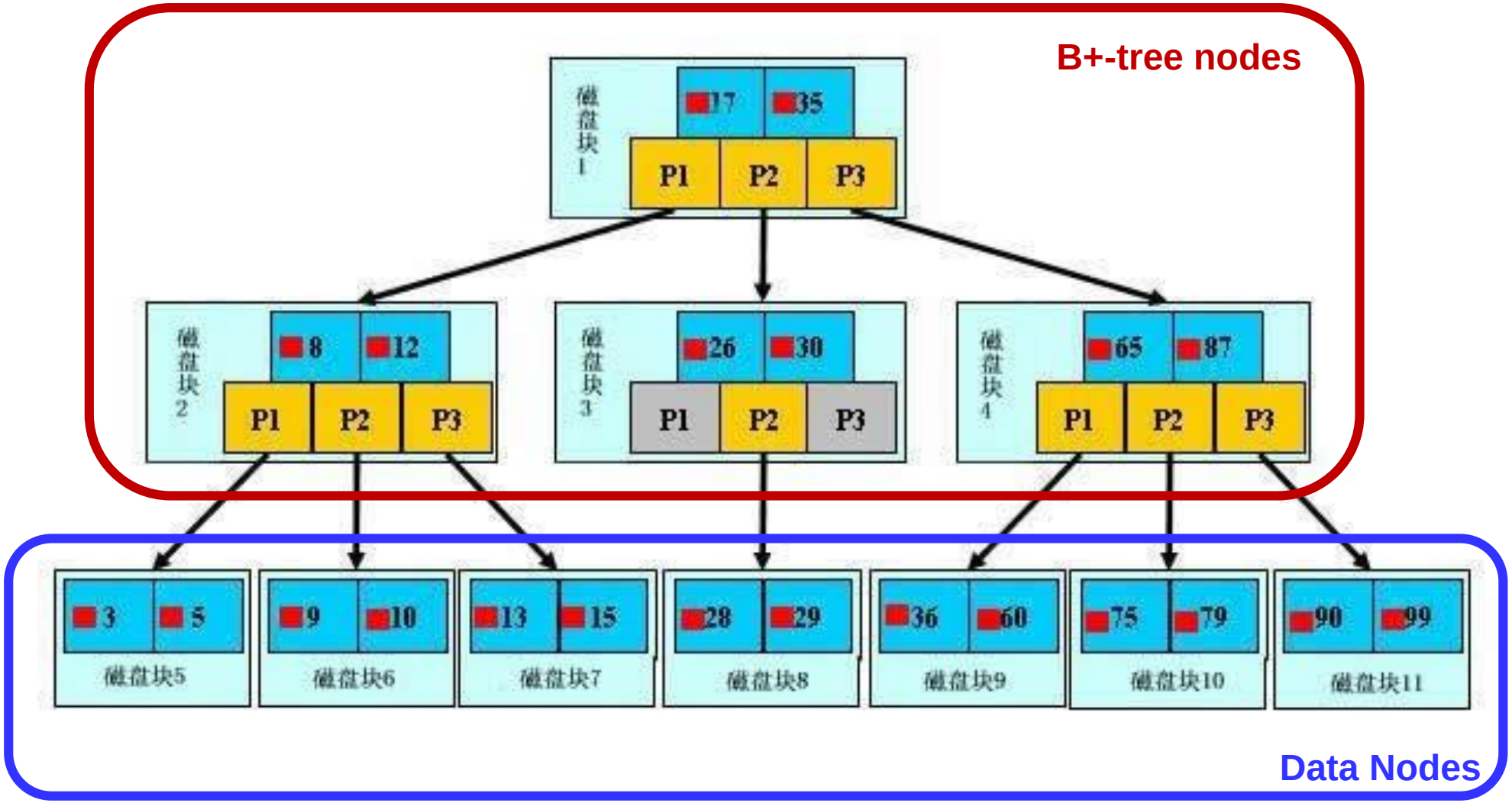
- The transactions which buy some specific product
  - White Metal Lantern
- How to find the invoice number? – "**White Metal Lantern**"
  - Location of the record should be connected with/embedded into the index
- You're required to
  1. Direct traverse the file to count the number of transactions containing "White Metal Lantern"
  2. **B+-tree** index (MM version) to finish 1
     - ✓ Finish the performance comparison of 1 and 2
  3. Design and Implement an Index File s.t
     - ✓ Efficient to **reconstruct** from the Index File



http://flacro.me/b-tree-index/

BLOCK?

- **In modern OS, PAGING based main memory management is popularly taken**
  - Your (machine code version) program is split into fixed size pages
    - 4KByte is the default size of page in Windows
  - The swapping of your program into/out of MM is based on PAGEs
  - To ensure the consistency of FILE management on HDD, BLOCK is presented for file organization in HDD
    - Its role is just like PAGE in MM management
    - All files are mapped to BLOCKs
    - Therefore, BLOCK size is usually same as PAGE size
      - ✓ Not MUST – maybe P=4KB, while B=8KB as in Windows NT

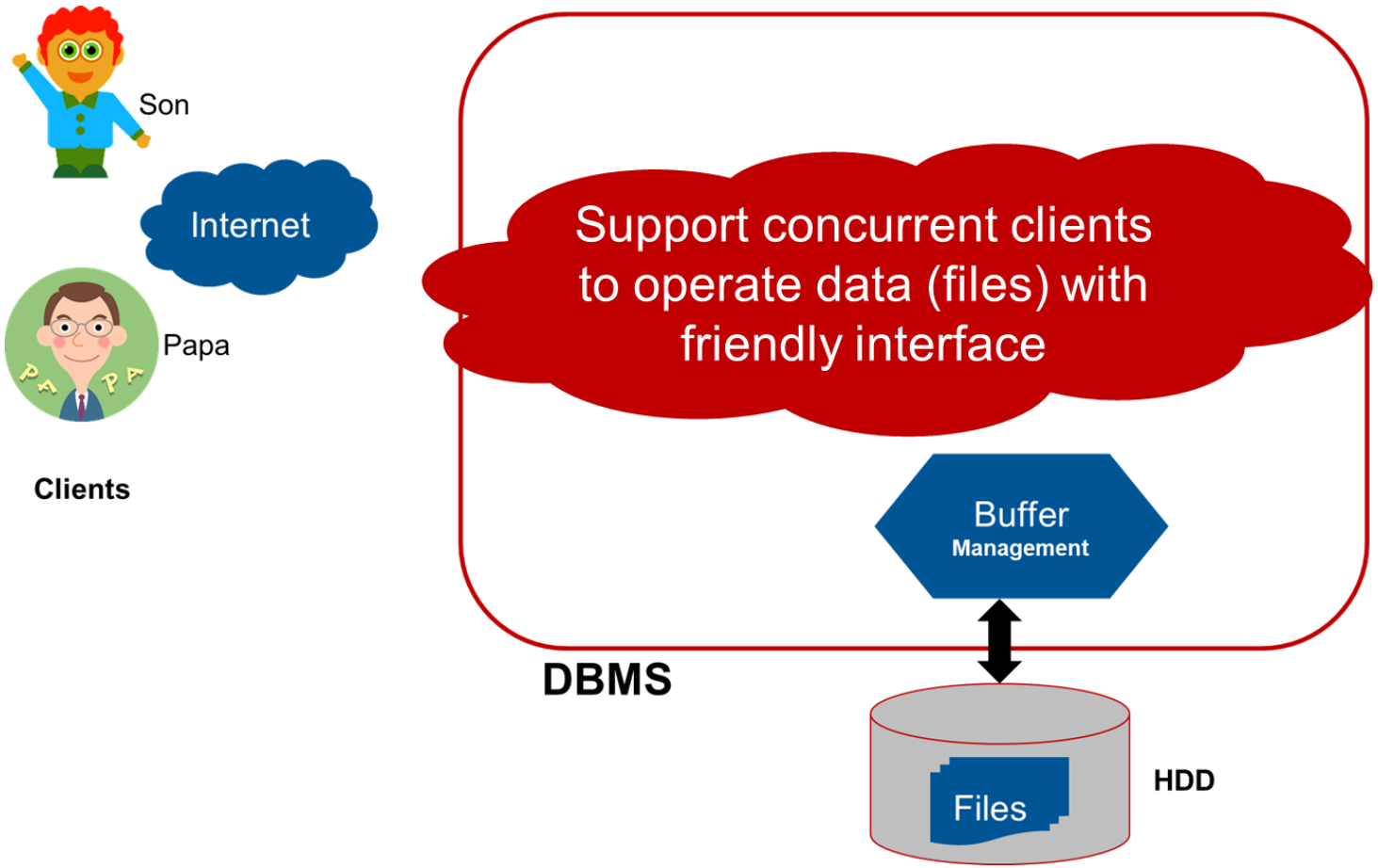


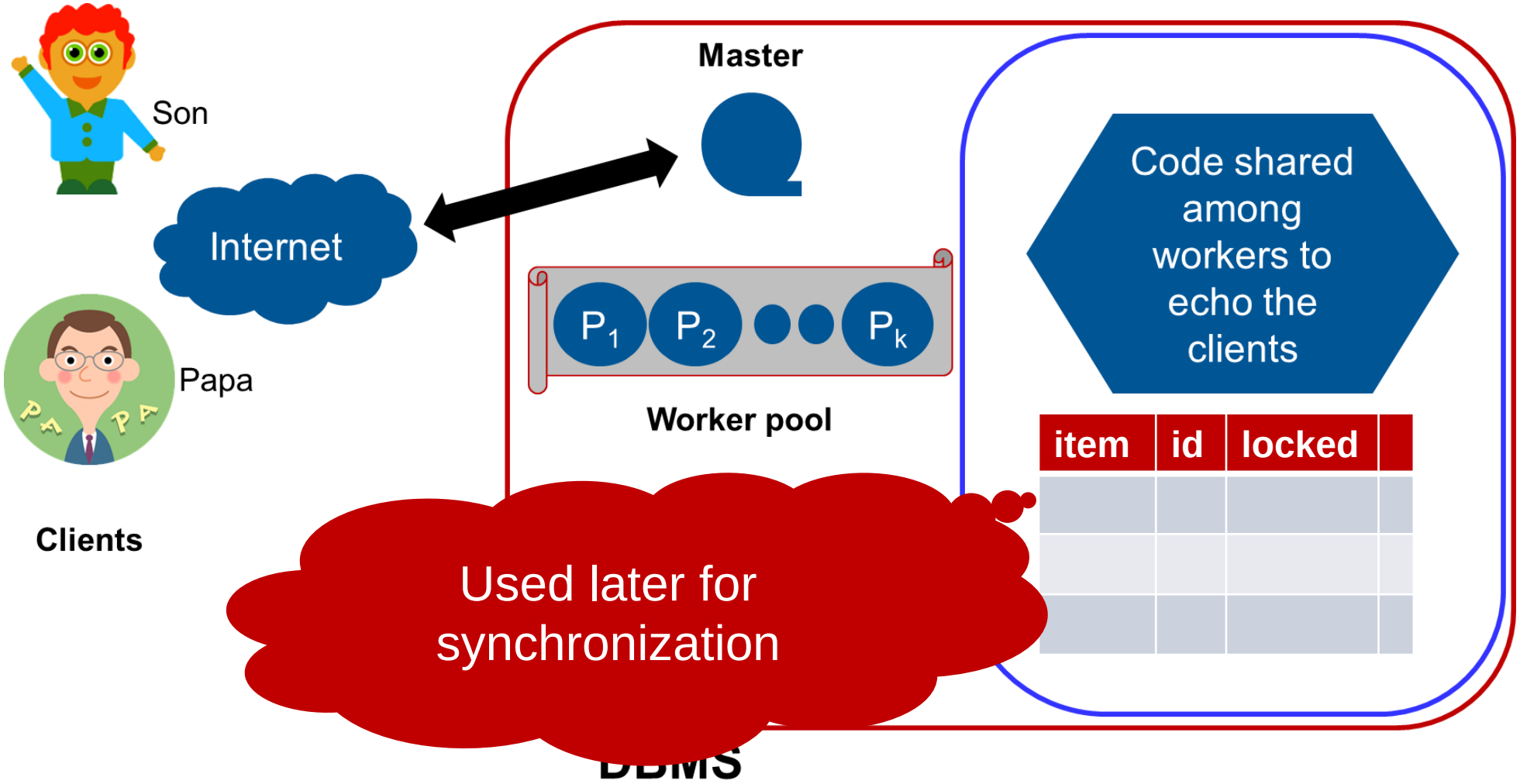
https://www.jianshu.com/p/0371c9569736

Project requirements

- **Project 2 [10 pts]: Simulate Concurrent Users**
  - By using multi-**process** programming skills, you're required to construct a simulation system
    - + C/S structure (Networking)
    - + **Process** pool (Server side)
    - + Support hundreds of clients
  - Logic of the simulation
    - Each client has same logic
      - ✓ Send words "Access **<u>some character</u>**" – random letter from A to Z.
        - » For example: "Access X"
    - Server based on **Process** Pool
      - ✓ Echo "Client x" + received words after random sleep
        - » After 5 ms echo "Client 12 for X"

- **If there is no available thread in the server pool, client should wait for a random sleep**




Son
Internet
Papa
Clients
Master
P_1
P_2
P_k
Worker pool
Code shared among workers to echo the clients
item
id
locked
Used later for synchronization
DBMS

- **You are required further to compare the performance (time to finish all the clients' requests) with different number of processes in the pool**
  - 10, 30, 50, 70, 100
  - Draw the comparison

Multi-Processed Programming with Python

- 如果你打算编写多进程的服务程序，Unix/Linux无疑是正确的选择。由于Windows没有fork调用，难道在Windows上无法用Python编写多进程的程序?
- 由于Python是跨平台的，自然也应该提供一个跨平台的多进程支持。multiprocessing模块就是跨平台版本的多进程模块。
  - multiprocessing模块提供了一个Process类来代表一个进程对象

```python
from multiprocessing import Process
import os

# 子进程要执行的代码
def run_proc(name):
    print('Run child process %s (%s)...' % (name, os.getpid()))

if __name__=='__main__':
    print('Parent process %s.' % os.getpid())
    p = Process(target=run_proc, args=('test',))
    print('Child process will start.')
    p.start()
    p.join()
    print('Child process end.')
```

MultiProcess Pool

- 如果要启动大量的子进程，可以用进程池的方式批量创建子进程：

```
from multiprocessing import Pool
import os, time, random

def long_time_task(name):
    print('Run task %s (%s)...' % (name, os.getpid()))
    start = time.time()
    time.sleep(random.random() * 3)
    end = time.time()
    print('Task %s runs %0.2f seconds.' % (name, (end - start)))

if __name__=='__main__':
    print('Parent process %s.' % os.getpid())
    p = Pool(4)
    for i in range(5):
        p.apply_async(long_time_task, args=(i,))
    print('Waiting for all subprocesses done...')
    p.close()
    p.join()
    print('All subprocesses done.')
```

子进程

- 很多时候，子进程并不是自身，而是一个外部进程。我们创建了子进程后，还需要控制子进程的输入和输出。
- subprocess模块可以让我们非常方便地启动一个子进程，然后控制其输入和输出。
- 例子演示了如何在Python代码中运行命令nslookup www.python.org，这和命令行直接运行的效果是一样的：

```
import subprocess

print('$ nslookup www.python.org')
r = subprocess.call(['nslookup', 'www.python.org'])
print('Exit code:', r)
```

如果子进程还需要输入，则可以通过communicate()方法输入：

```
import subprocess

print('$ nslookup')
p = subprocess.Popen(['nslookup'], stdin=subprocess.PIPE, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
output, err = p.communicate(b'set q=mx\npython.org\nexit\n')
print(output.decode('utf-8'))
print('Exit code:', p.returncode)
```

进程间通信

- Process之间肯定是需要通信的，操作系统提供了很多机制来实现进程间的通信。Python的multiprocessing模块包装了底层的机制，提供了Queue、Pipes等多种方式来交换数据。
- 以Queue为例，在父进程中创建两个子进程，一个往Queue里写数据，一个从Queue里读数据：

ask.csdn.net › questions

python-PyCharm-执行程序PermissionError: [WinError 5] 拒绝 ...

2019年1月9日 - ... in put if not self._sem.acquire(block, timeout): PermissionError: [WinError 5] 拒绝访问。 Process finished with exit code 0. 而直接在cmd中， ...

python-多进程在运行的时候只有一个子进程会运行，怎么解决 ... 2020年5月13日

图片-安装tensorflow的问题？——CSDN问答频道 2018年3月28日

ask.csdn.net站内的其它相关信息

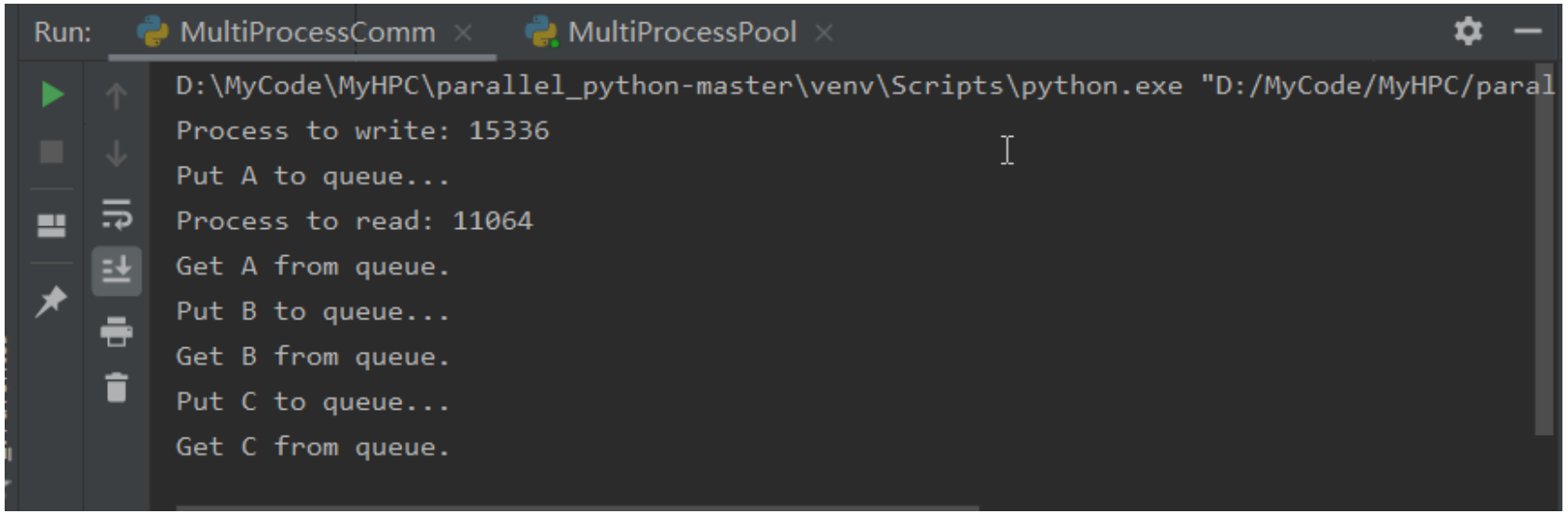
```python
from multiprocessing import Process, Queue
import os, time, random

# 写数据进程执行的代码:
def write(q):
    print('Process to write: %s' % os.getpid())
    for value in ['A', 'B', 'C']:
        print('Put %s to queue...' % value)
        q.put(value)
        time.sleep(random.random())

# 读数据进程执行的代码:
def read(q):
    print('Process to read: %s' % os.getpid())
    while True:
        value = q.get(True)
        print('Get %s from queue.' % value)

if __name__=='__main__':
    # 父进程创建Queue，并传给各个子进程:
    q = Queue()
    pw = Process(target=write, args=(q,))
    pr = Process(target=read, args=(q,))
    # 启动子进程pw，写入:
    pw.start()
    # 启动子进程pr，读取:
    pr.start()
    # 等待pw结束:
    pw.join()
    # pr进程里是死循环，无法等待其结束，只能强行终止:
    pr.terminate()
```

导入Manager

from multiprocessing import Process,Manager

将q = Queue() 改为 q = Manager().Queue()

编辑于：2019.06.16 11:16



Project requirements

- **Project 5 [10 pts]: Debug PostgreSQL**
  - http://hsqldb.org/
  - Very helpful to follow source code to understand the D&I of many softwares
    - Minix/Linux for OS
    - HyperSQL, H2, **PostgreSQL** , MySQL for DBMS

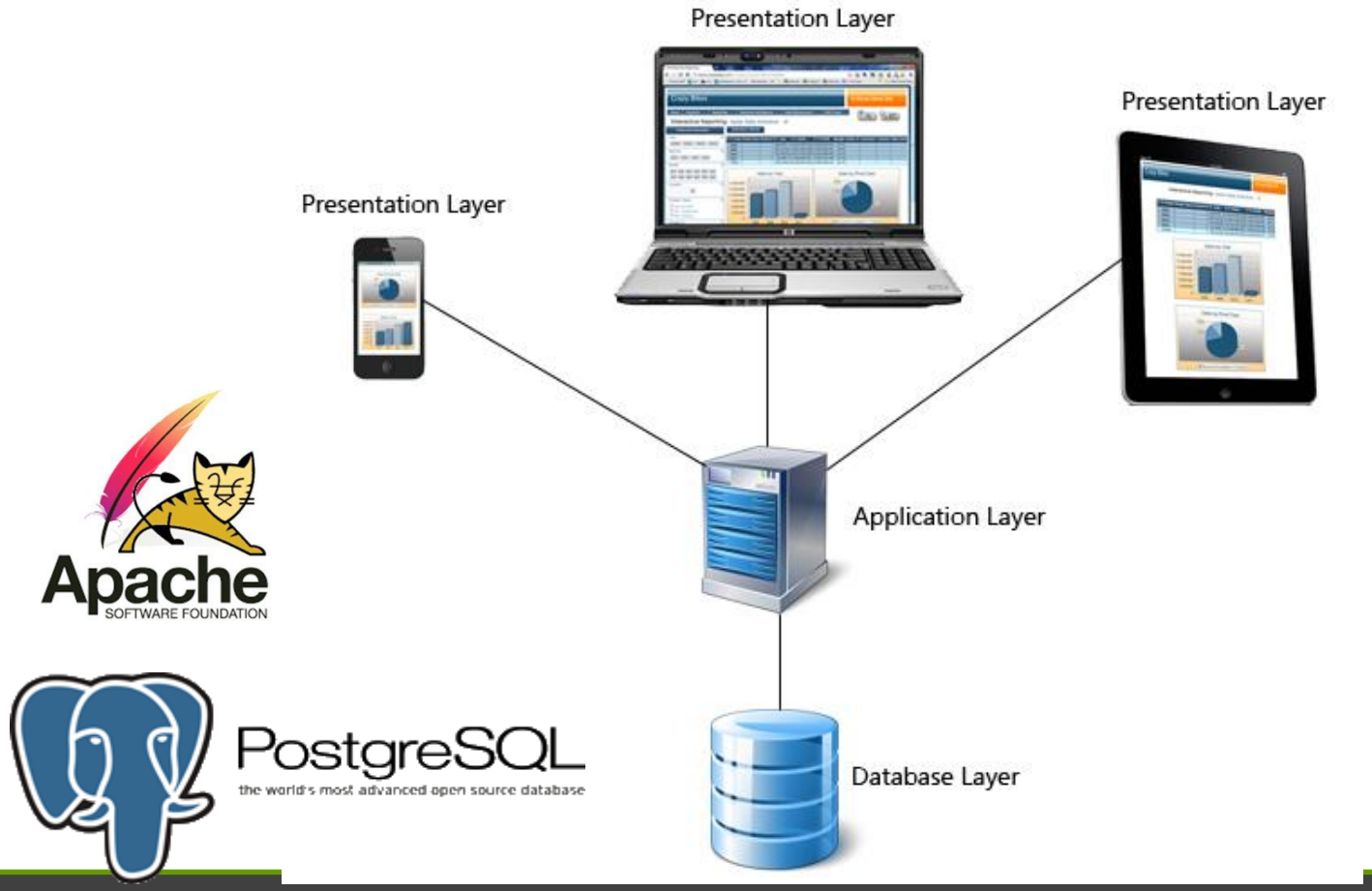
- **I prepared some slides to demonstrate how to use VS+Perl etc. to build/compile the source code of PostgreSQL**
- **Tasks of Project 5**
  1. Use PostgreSQL
     - With the given dataset – "Online Retail.xlsx" – 541909 records
       - ✓ https://archive.ics.uci.edu/ml/datasets/Online+Retail
       - ✓ Try to load it into PostgreSQL
         - » Define the table same as xlsx? Redundant or not – you'll learn later in DBMS Lecture part?
       - ✓ Try SQL sentences
         - » I.D.U.S

Tasks of Project 5

2. Debug **PostgreSQL**
   a) Change server prompt to **your name**
      - ✓ lbkong>>
   b) Cut the SQL processing in PostgreSQL (Client and Server), and just **echo** the inputted SQL
      - ✓ lbkong>> your input is: SELECT * From students
   c) Try to **print out the parse tree of a simple SQL** supported by PostgreSQL, like “SELECT * from students”
      - ✓ Of course you should create a “students” table first, and input some students records
      - ✓ Name, studentID, Memo (for 40 words)

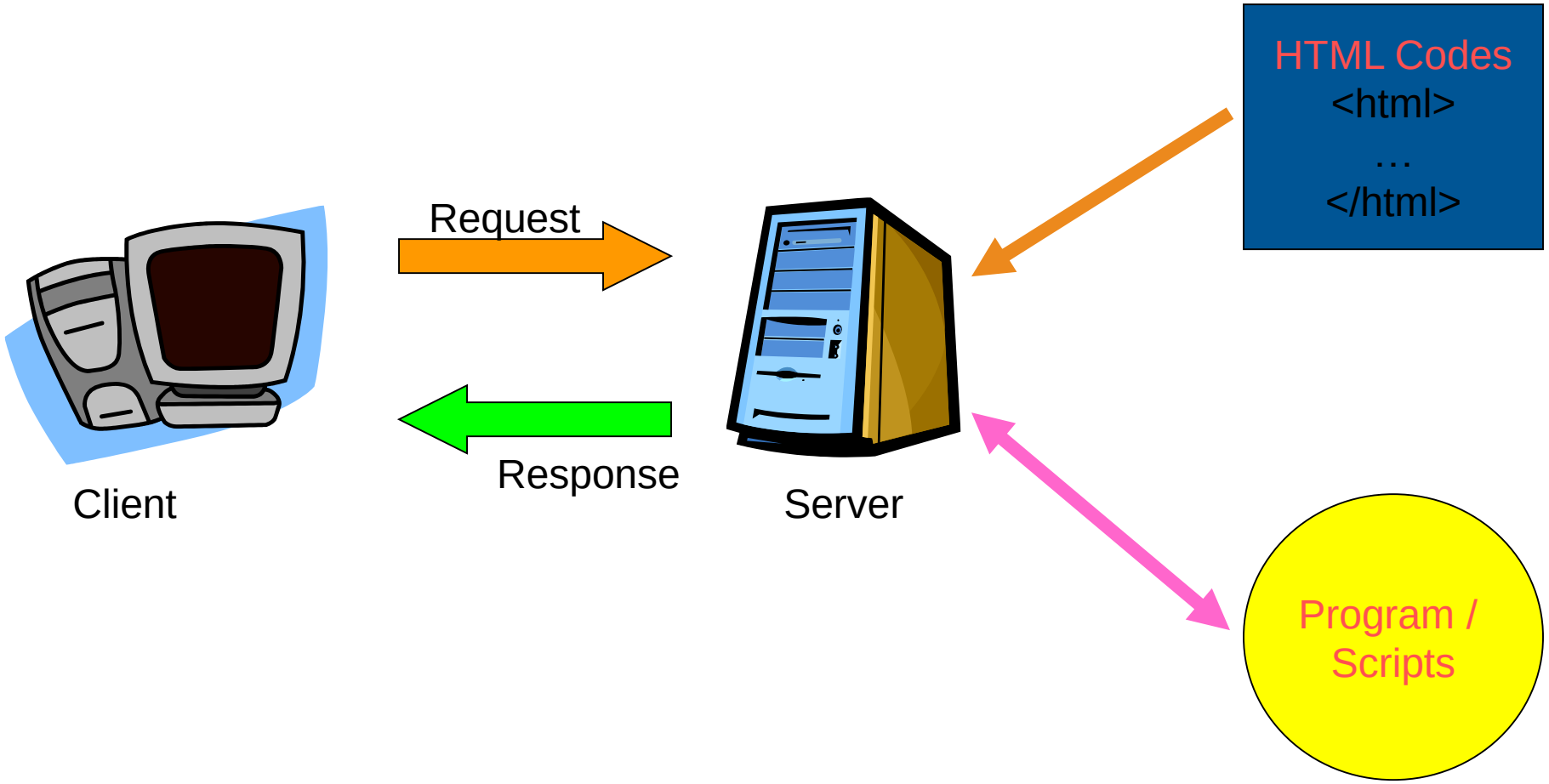
Java/JSP/PHP/Python/R + PostgreSQL/Greenplum/HAWQ - R.D.I.P

- **Web Application[20 pts]: such as JSP/Tomcat+ PostgreSQL**
  - DBMS is the critical software supporting our modern society



Browser is the popular frontend in 3-tier

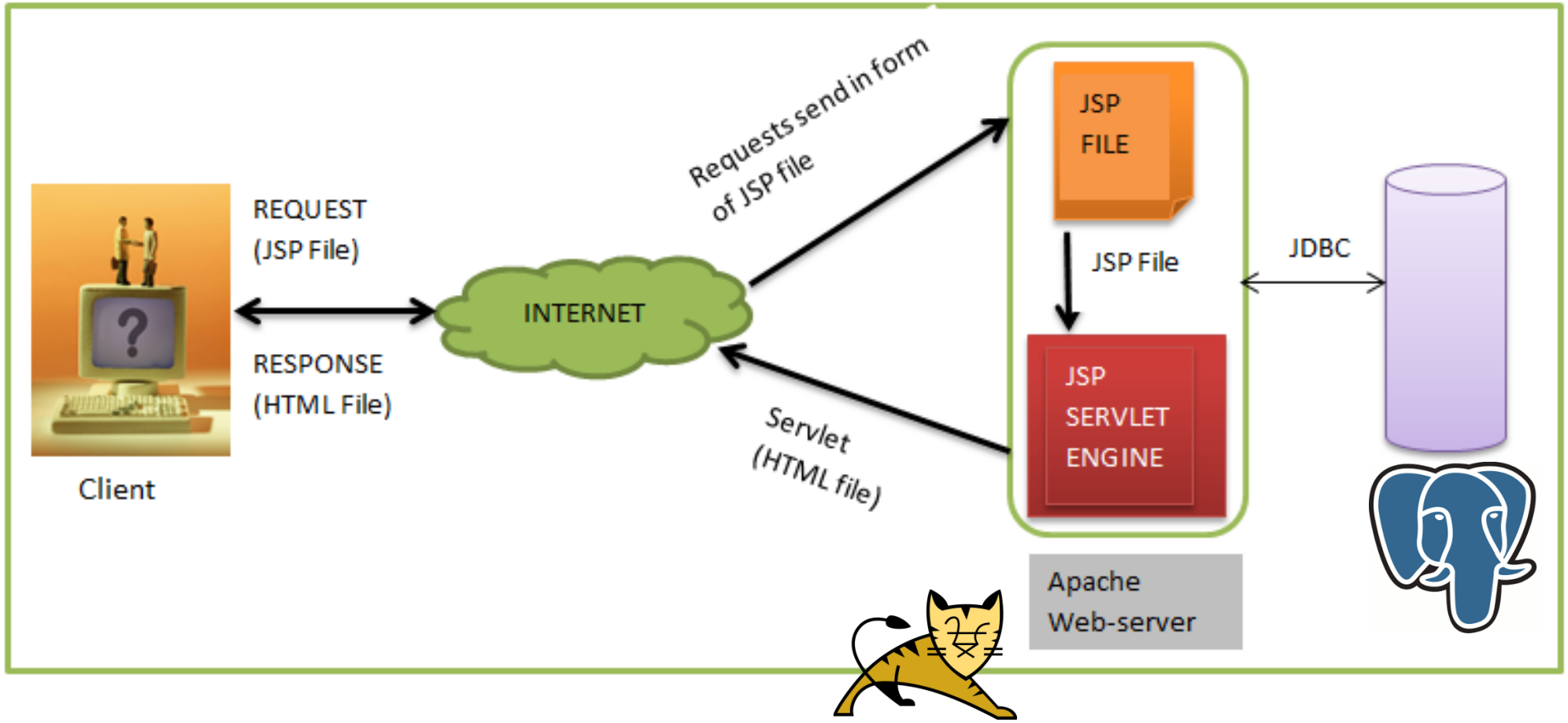
- The HTTP Request/Response Model



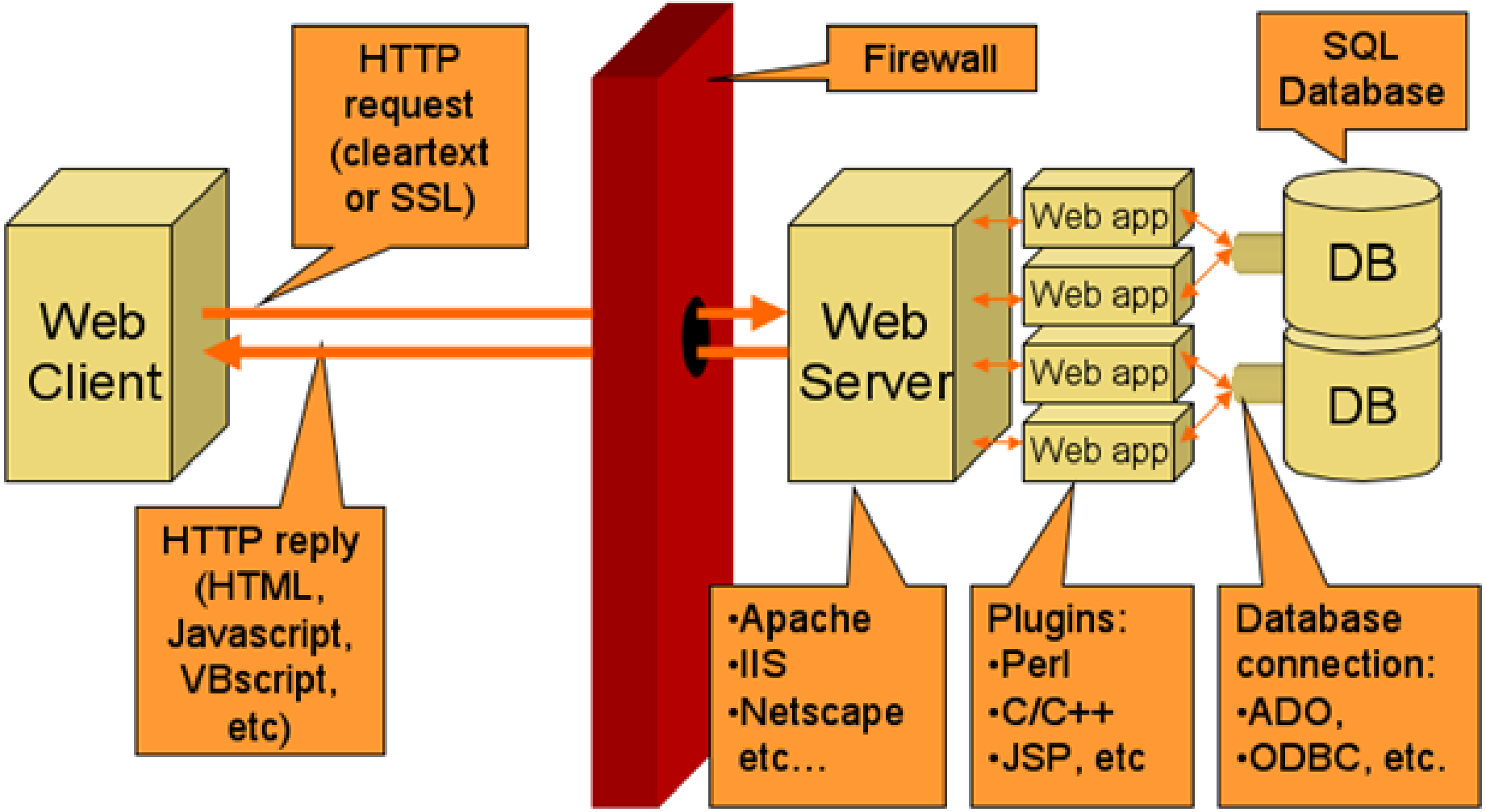
Dynamic page idea is powerful – improve more interaction through Web pages!

- **Dynamic? – [Web pages call functions for powerful service]**
  - The server could compose a webpage **on the fly** according to the client's request
    - The string returned to client is a complete webpage
- **Before xSPs (ASP, JSP) or PHP/Python…, the traditional way to produce dynamic webpages is CGI – Common Gateway Interface**
  - a program which could produce HTML file
    - It follows some standard by which World Wide Web servers may access external programs so that data is returned automatically in the form of a web page
  - You can use C to implement a **CGI** [Common Gate Interface] program which is called by web server

- JSP (JavaServer Pages) is one of those to support dynamic



- Now many techniques for dynamic – JSP is one of them



http://www.cs.columbia.edu/~hgs/teaching/ap/slides/autoc

Clarify some concepts

I hope you could understand the difference by yourself

- **Taxonomy**

	embedded in HTML	separate
server	SSI, ASP, PHP, **JSP**, CFM	server API (NSAPI), **cgi**, servlets
client	**JavaScript**	Java applets, plug-in

CGI example in C

- **CGI is the server-side program to produce HTML to client.**
- **So, we can also use CGI to produce a web page with FORM**

```
#  include  <stdio.h>
#  include  <stdlib.h>
void main(){
    printf( "Content-type:text/html;charset=utf-8\n\n");
   printf( " <html> \n ");
   printf( " <meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\" />  \n ");
   printf( " <title> Your first web page </title> \n ");
   printf( " <body> \n ");

    printf(" <form action=\"http://localhost/cgi-bin/Untitled6.cgi\">");
    printf(" <p>Pls input two numbers for multiplication.</p>");
    printf(" <input name=\"m\" size=\"5\">");
    printf(" <input name=\"n\" size=\"5\"> <br/>");
    printf(" <input type=submit value=\“OK\" ></form> ");

   printf( " </body>  ");
   printf( " </html> ");
}
```

Sample Servlet

- Servlet works like a CGI

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends
     HttpServlet {

   public void doGet(HttpServletRequest
     request, HttpServletResponse
     response)
   throws IOException, ServletException
   {
 response.setContentType("text/html");
```

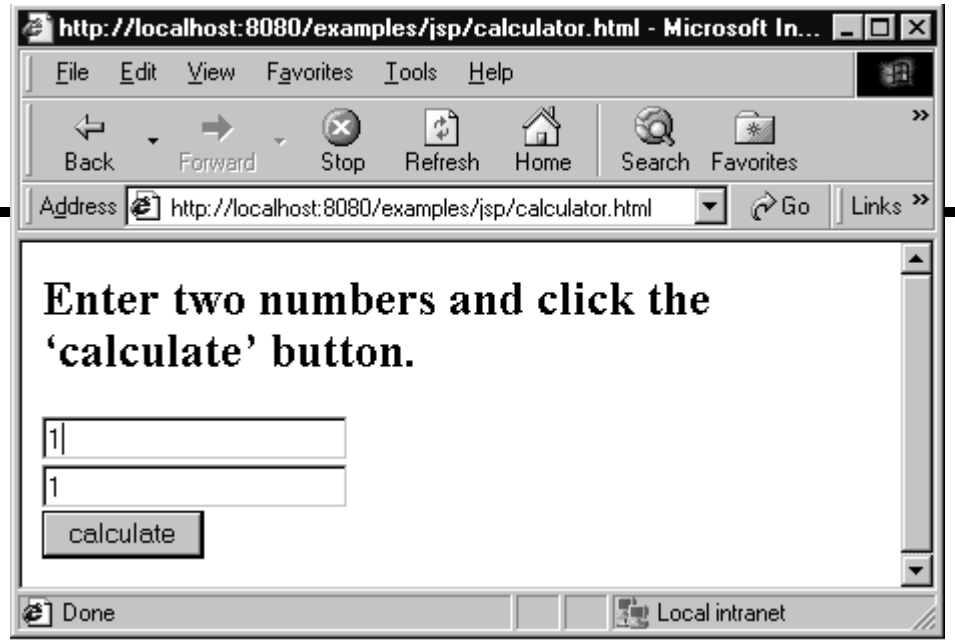
Hello World!

```
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<body>");
out.println("<head>");
out.println("<title>Hello World!</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Hello World!</h1>");
out.println("</body>");
out.println("</html>");
   }
```

JavaScript example – executed in client

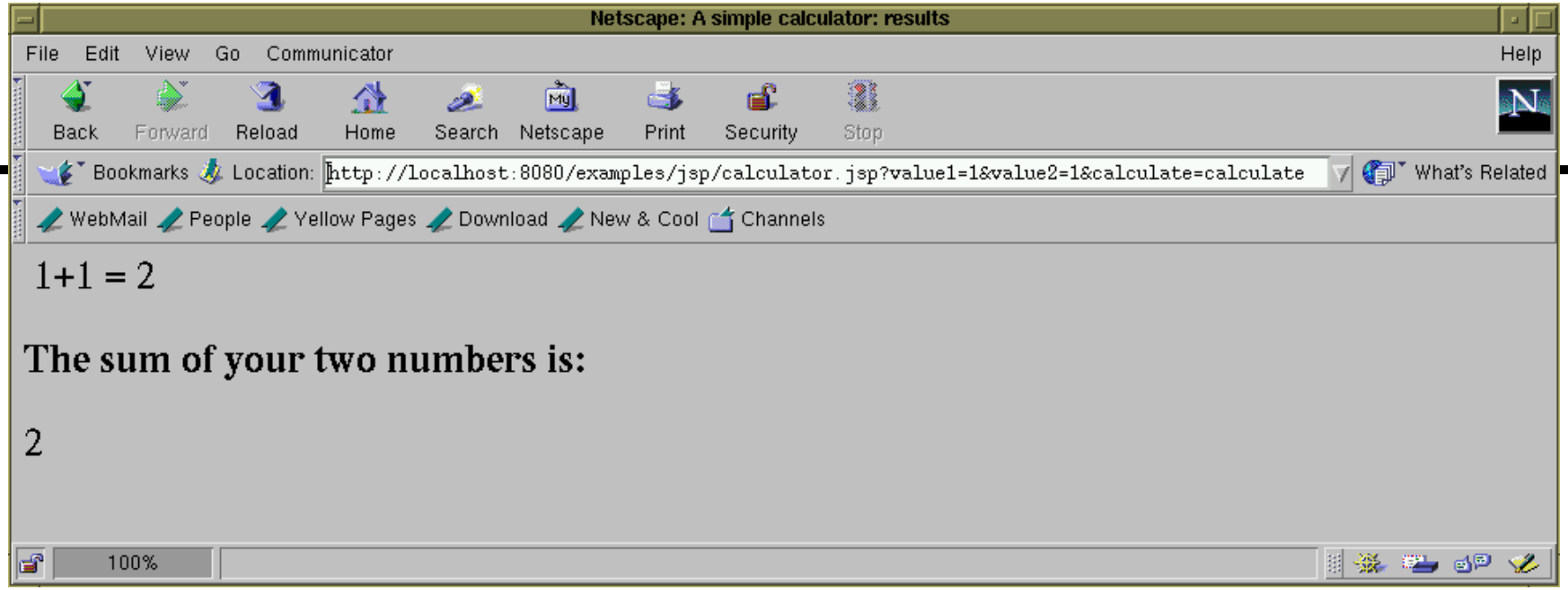
```
<!DOCTYPE html>
<!-- JSExAlertWrite.html -->
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Example: Functions alert() and document.write()</title>
  <script>
    alert("Hello, world!");
  </script>
</head>
<body>
  <h1>My first JavaScript says:</h1>
  <script>
    document.write("<h2><em>Hello world, again!</em></h2>");
    document.write("<p>This document was last modified on "
        + document.lastModified + ".</p>");
  </script>
</body>
</html>
```

JSP example – Calculator




```
<html>
<head></head>
<body>
<p>Enter two numbers and click the
'calculate' button.</p>

<form action="calculator.jsp" method="get">
<input type=text name=value1><br>
<input type=text name=value2 ><br>
<input type=submit name=calculate value=calculate>
</form>
</body>
</html>
```

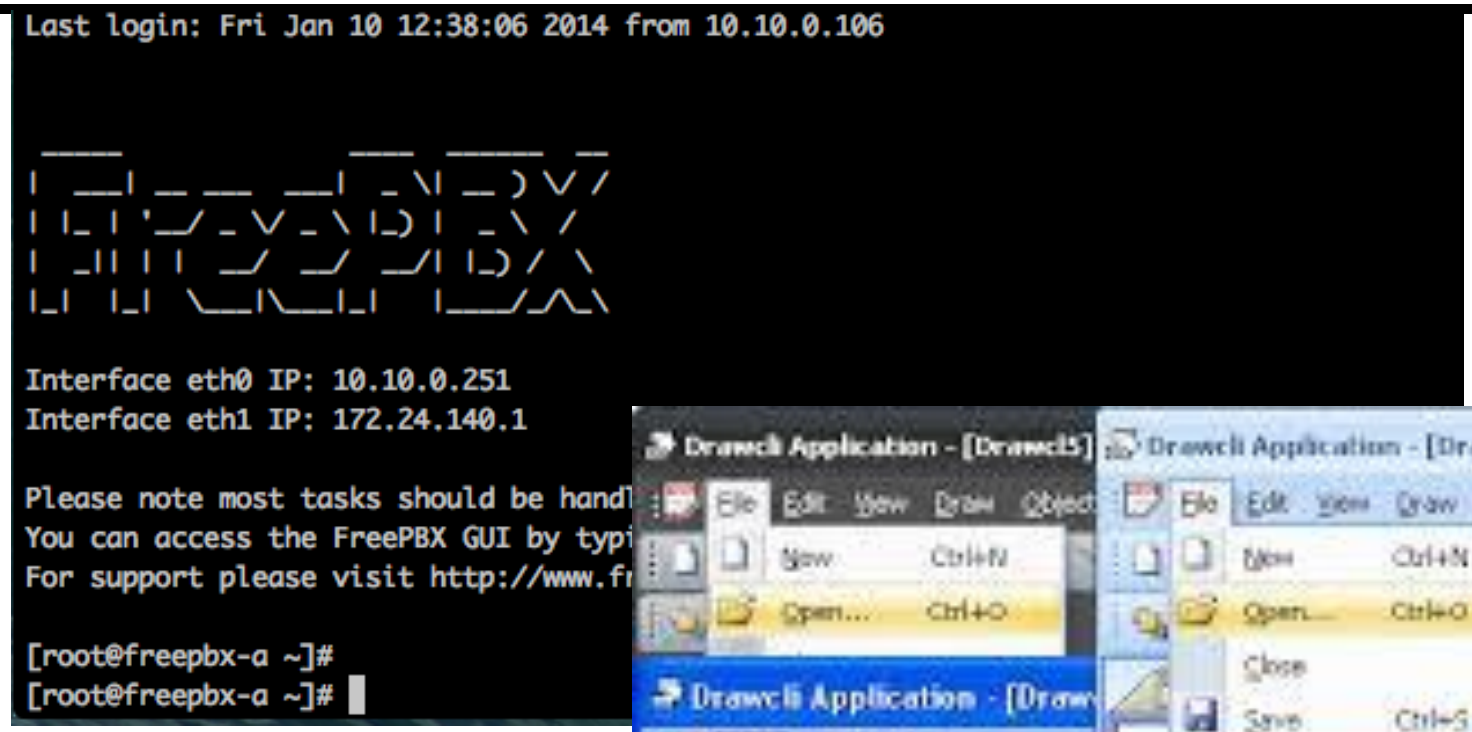
Calculator.html

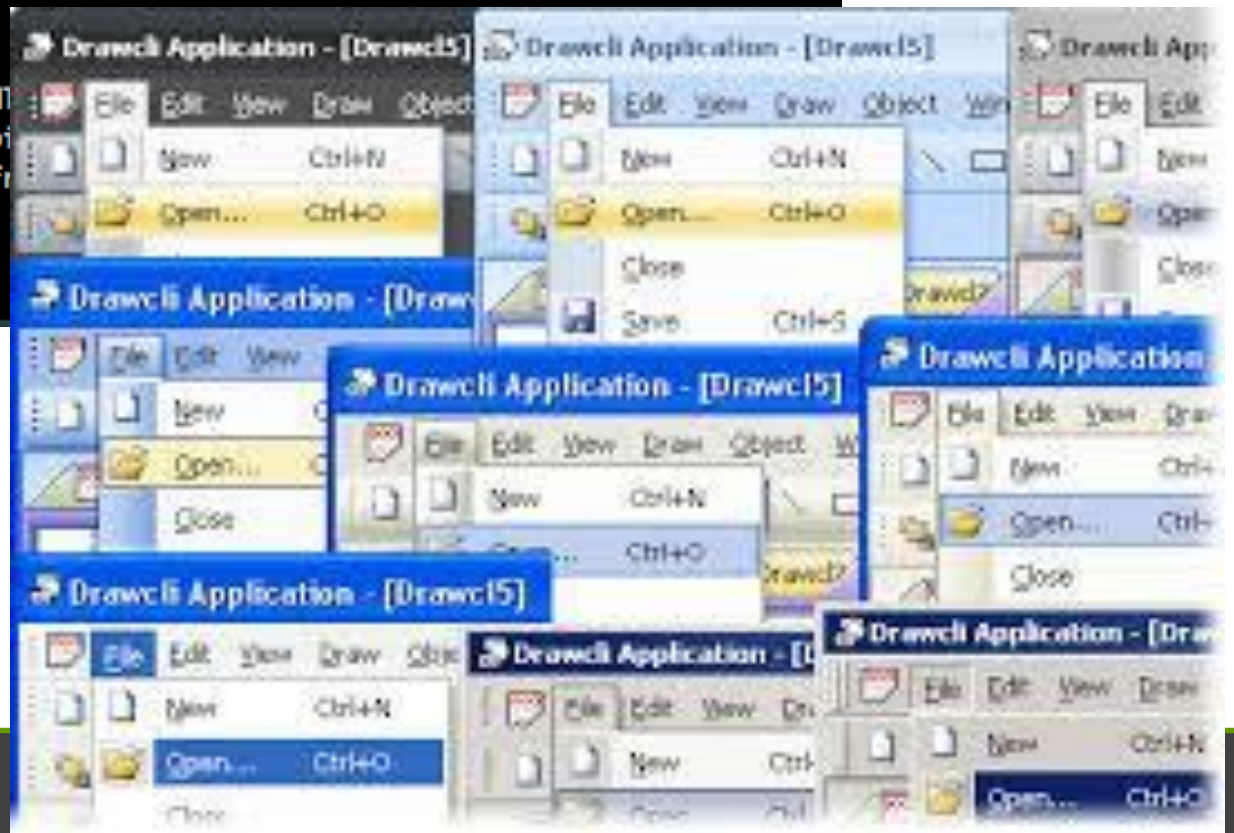



```
<html>
<head><title>A simple calculator: results</title></head>
<body>
<%-- A simpler example 1+1=2 --%>
1+1 = <%= 1+1 %>
<%-- A simple calculator --%>
<h2>The sum of your two numbers is:</h2>
<%= Integer.parseInt(request.getParameter("value1")) +
    Integer.parseInt(request.getParameter("value2")) %>
</body>
</html>
```

Calculator.jsp

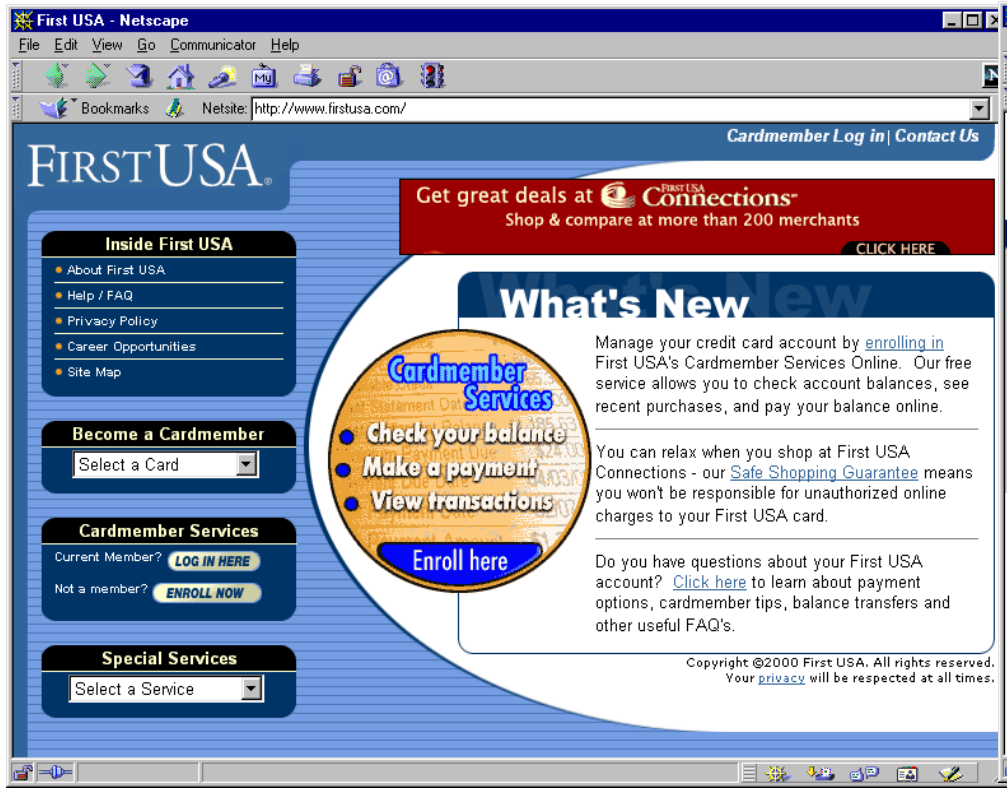
CLI → GUI (C/S) → B/S (3 tiers)

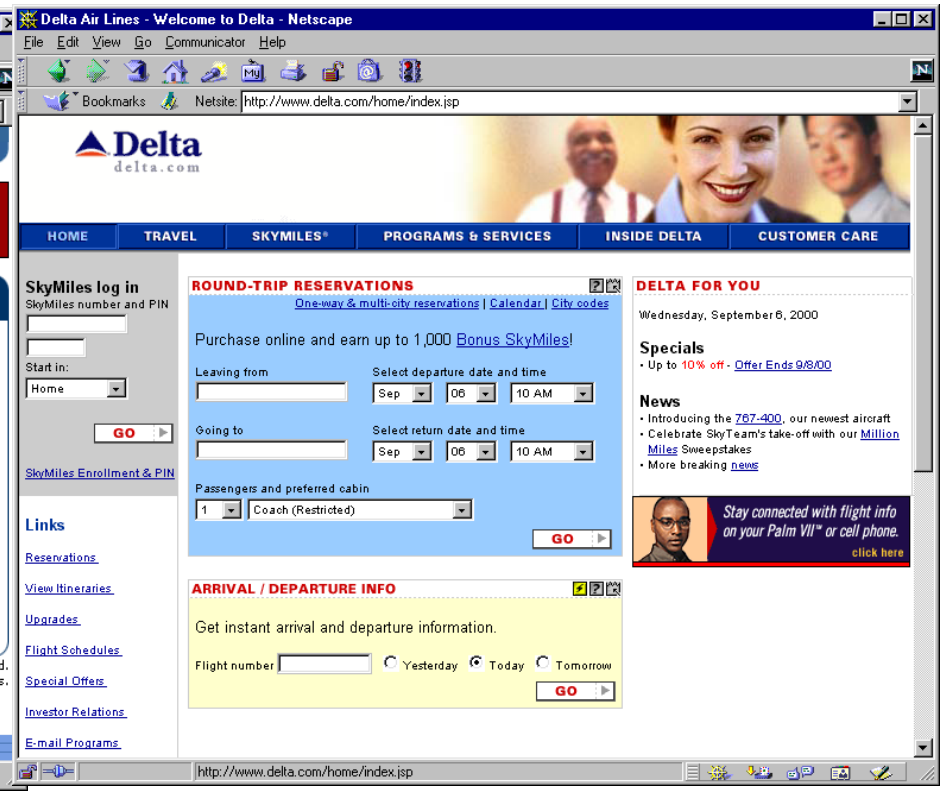




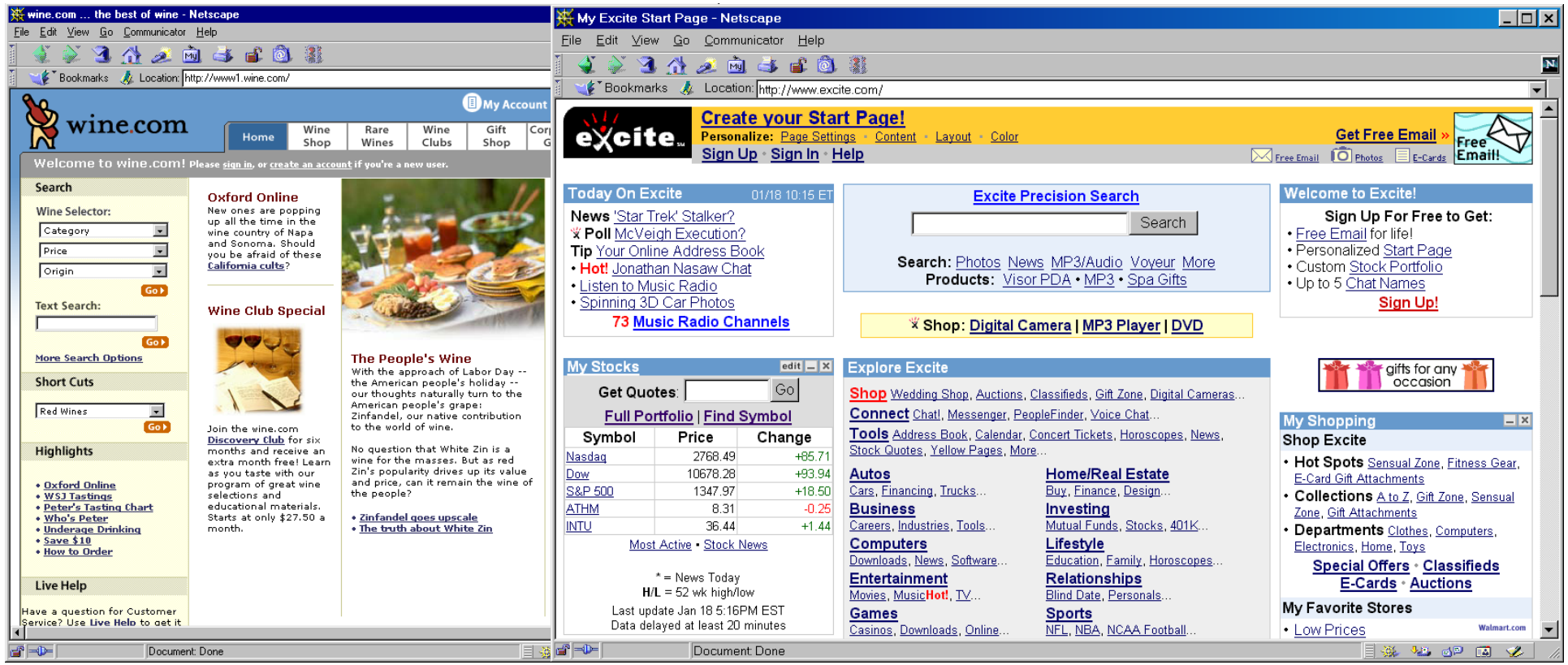
JSP/Servlets in the Real World

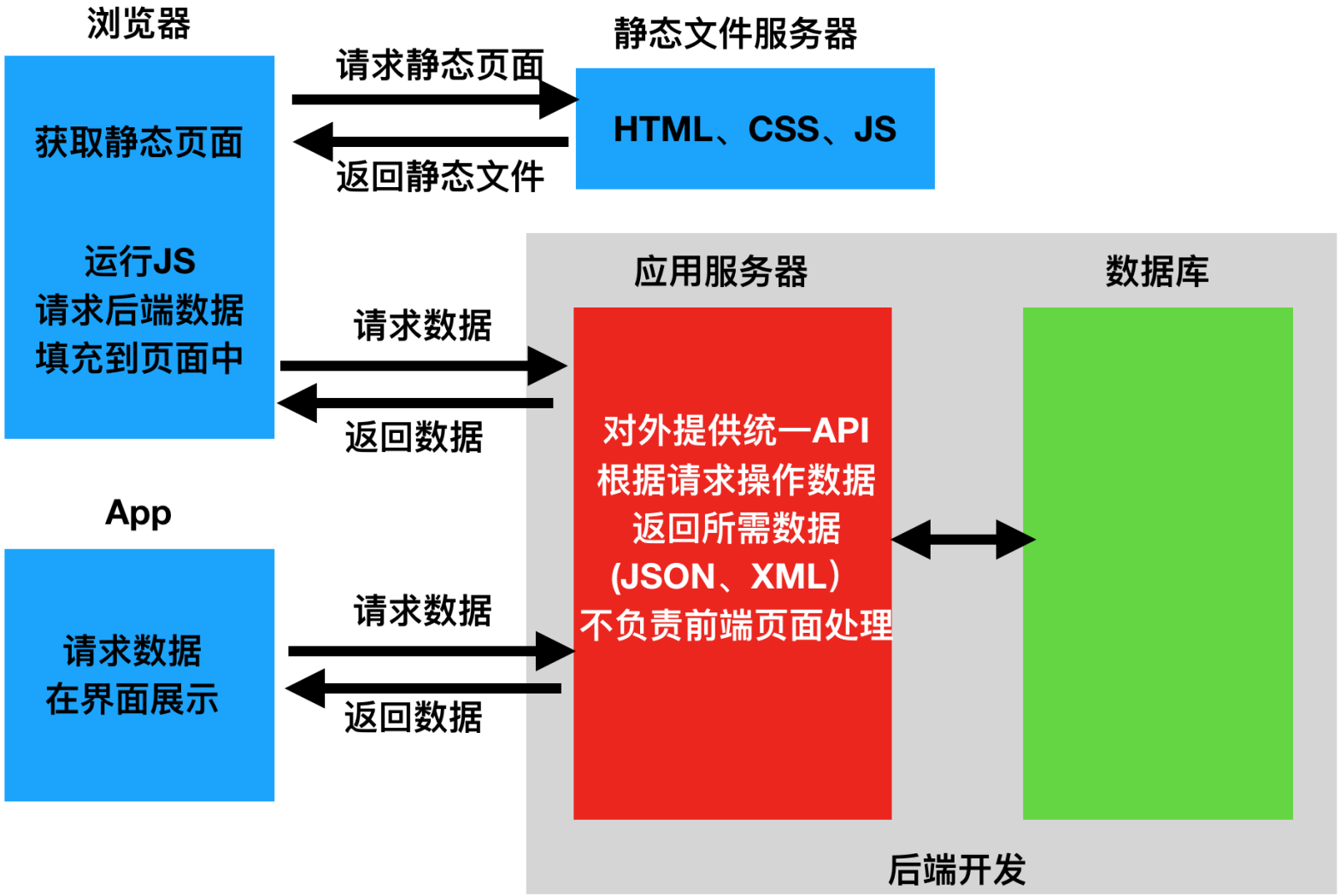
- **Delta Airlines**: entire Web site, including real-time schedule info
- **First USA Bank**: largest credit card issuer in the world; most on-line banking customers

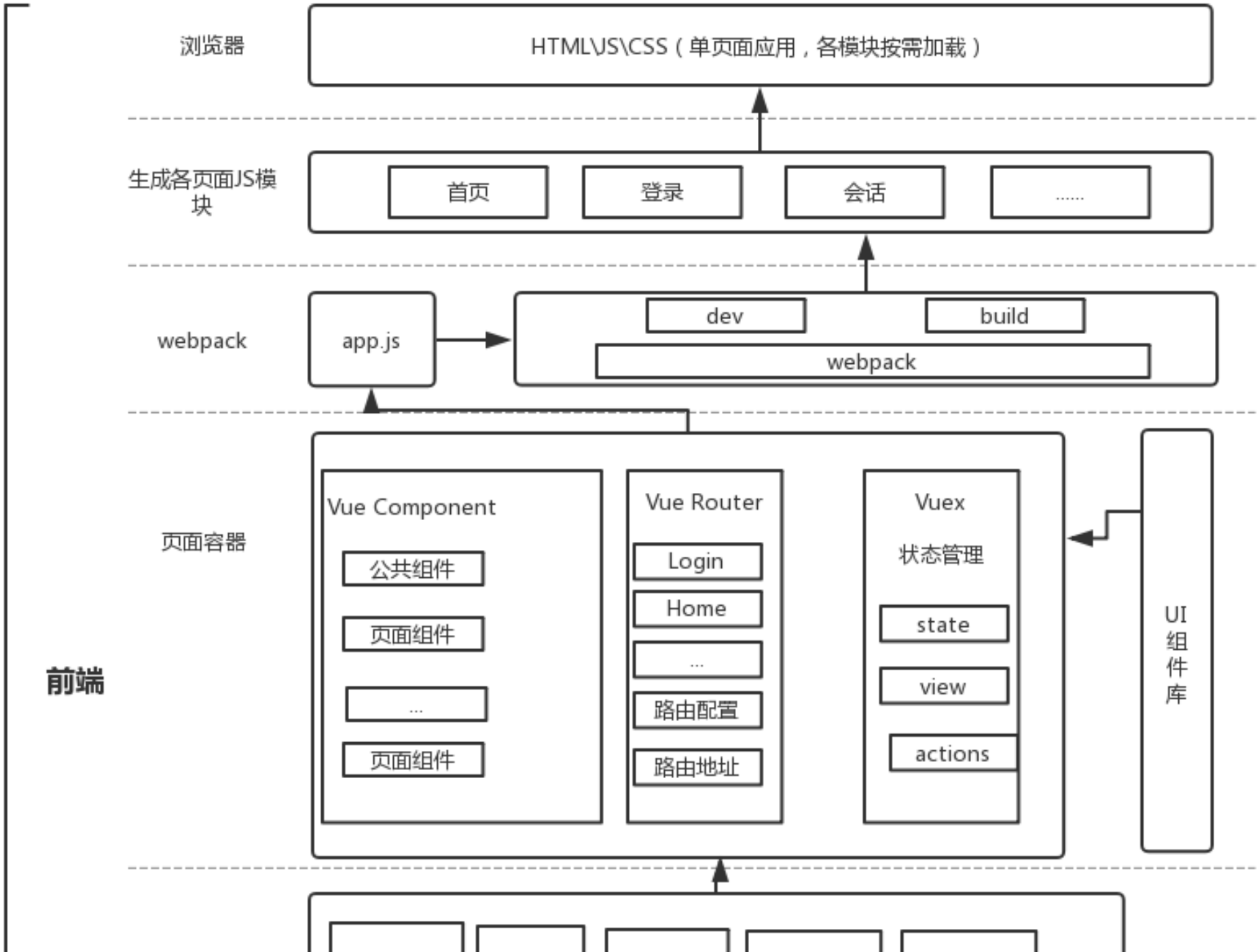


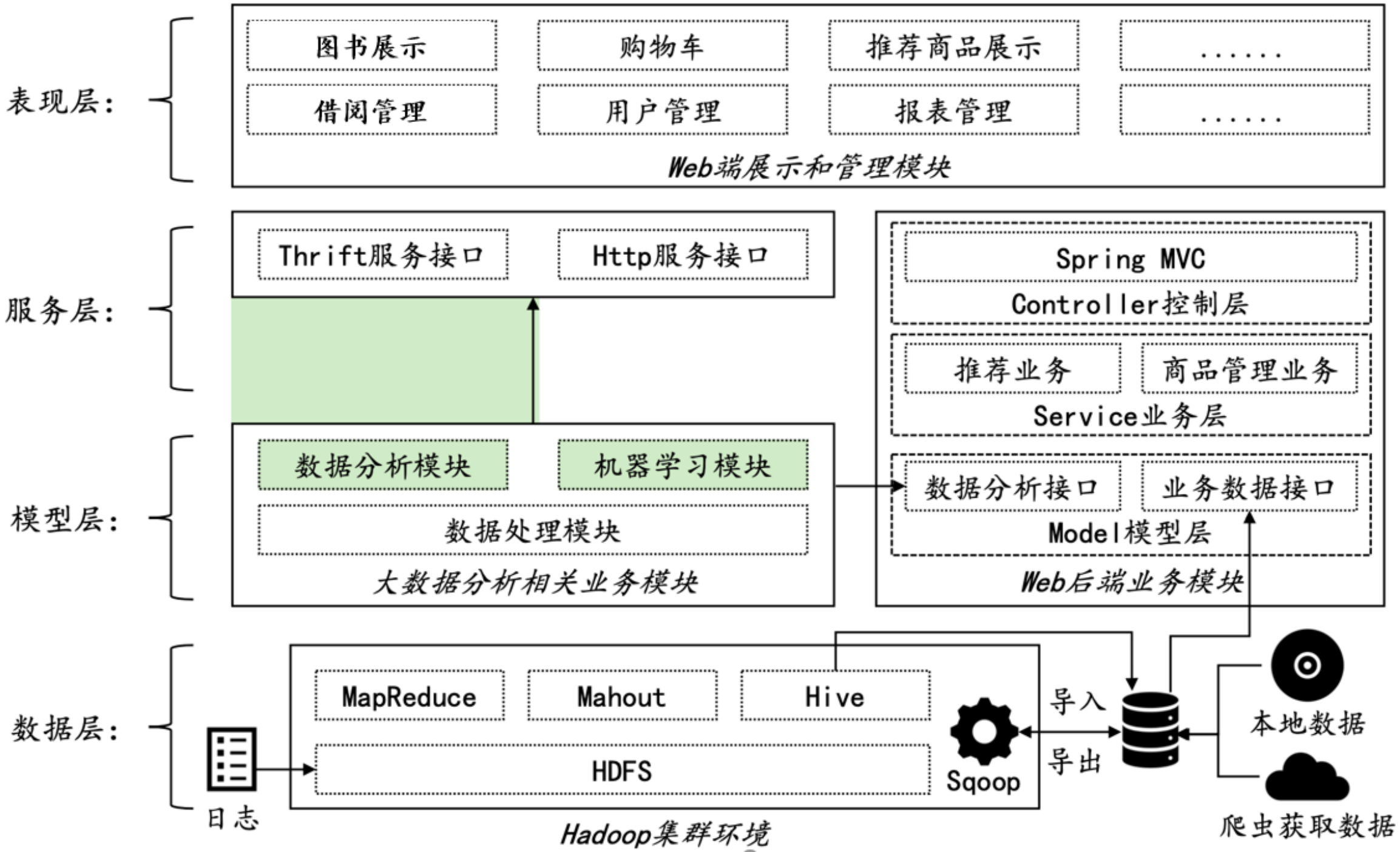


- **Excite**: one of the top five Internet portals; one of the ten busiest sites on the Web




浏览器
获取静态页面
运行JS
请求后端数据
填充到页面中
请求静态页面
返回静态文件
静态文件服务器
HTML、CSS、JS
请求数据
返回数据
应用服务器
数据库
对外提供统一API
根据请求操作数据
返回所需数据
(JSON、XML)
不负责前端页面处理
App
请求数据
在界面展示
请求数据
返回数据
后端开发


浏览器
HTML\JS\CSS（单页面应用，各模块按需加载）
生成各页面JS模块
首页
登录
会话
.....
webpack
app.js
dev
build
webpack
页面容器
Vue Component
公共组件
页面组件
...
页面组件
Vue Router
Login
Home
...
路由配置
路由地址
Vuex
状态管理
state
view
actions
UI组件库
前端


表现层：
图书展示
购物车
推荐商品展示
.....
借阅管理
用户管理
报表管理
.....
Web端展示和管理模块
服务层：
Thrift服务接口
Http服务接口
Spring MVC
Controller控制层
推荐业务
商品管理业务
Service业务层
模型层：
数据分析模块
机器学习模块
数据处理模块
大数据分析相关业务模块
数据分析接口
业务数据接口
Model模型层
Web后端业务模块
数据层：
MapReduce
Mahout
Hive
HDFS
Sqoop
导入
导出
本地数据
日志
爬虫获取数据
Hadoop集群环境

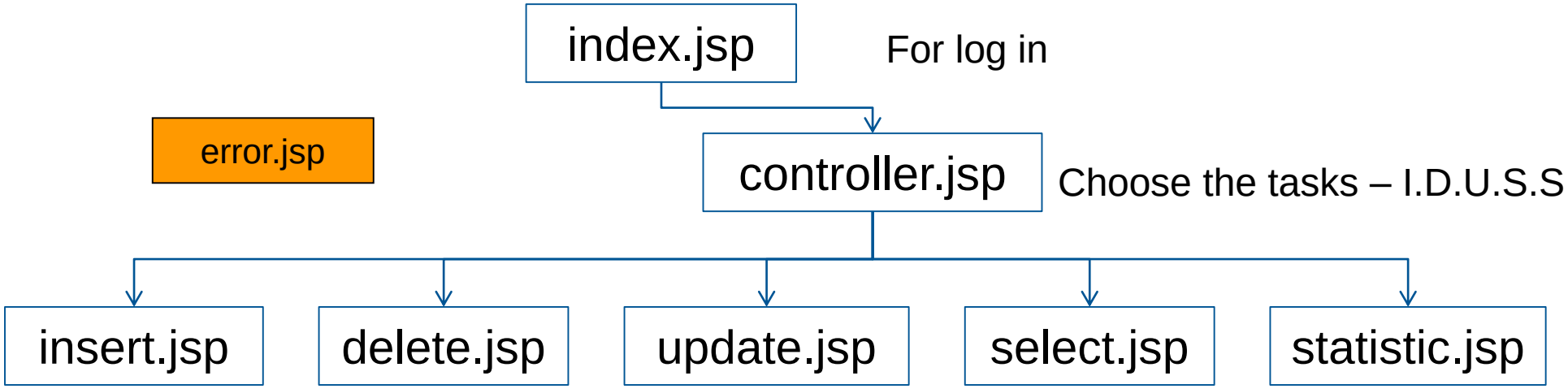
Requirement for Project 5

- The JSP based Web app should support
  - Log in module
  - I.D.U.S
  - Some simple statistics like AVG, Top 100, …
  - Try transaction commit

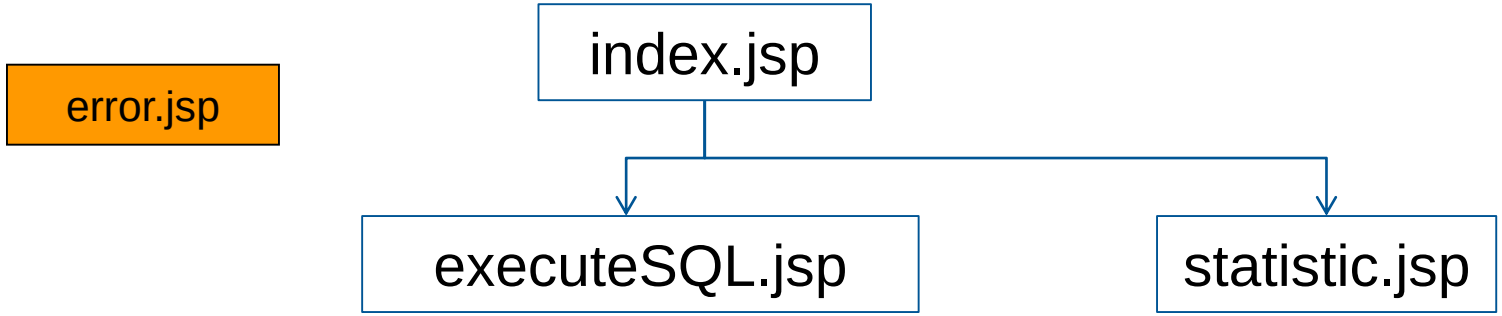
It's your responsibility to learn JSP and finish this "Web Application"

- You can choose your own data (or "on-line retail") to finish your Web app
  - Data should be kept in PostgreSQL
    - ✓ Show the db design in document
      - » You can use PowerDesigner to design and try database first
    - ✓ How to connect PostgreSQL is your responsibility

Hints



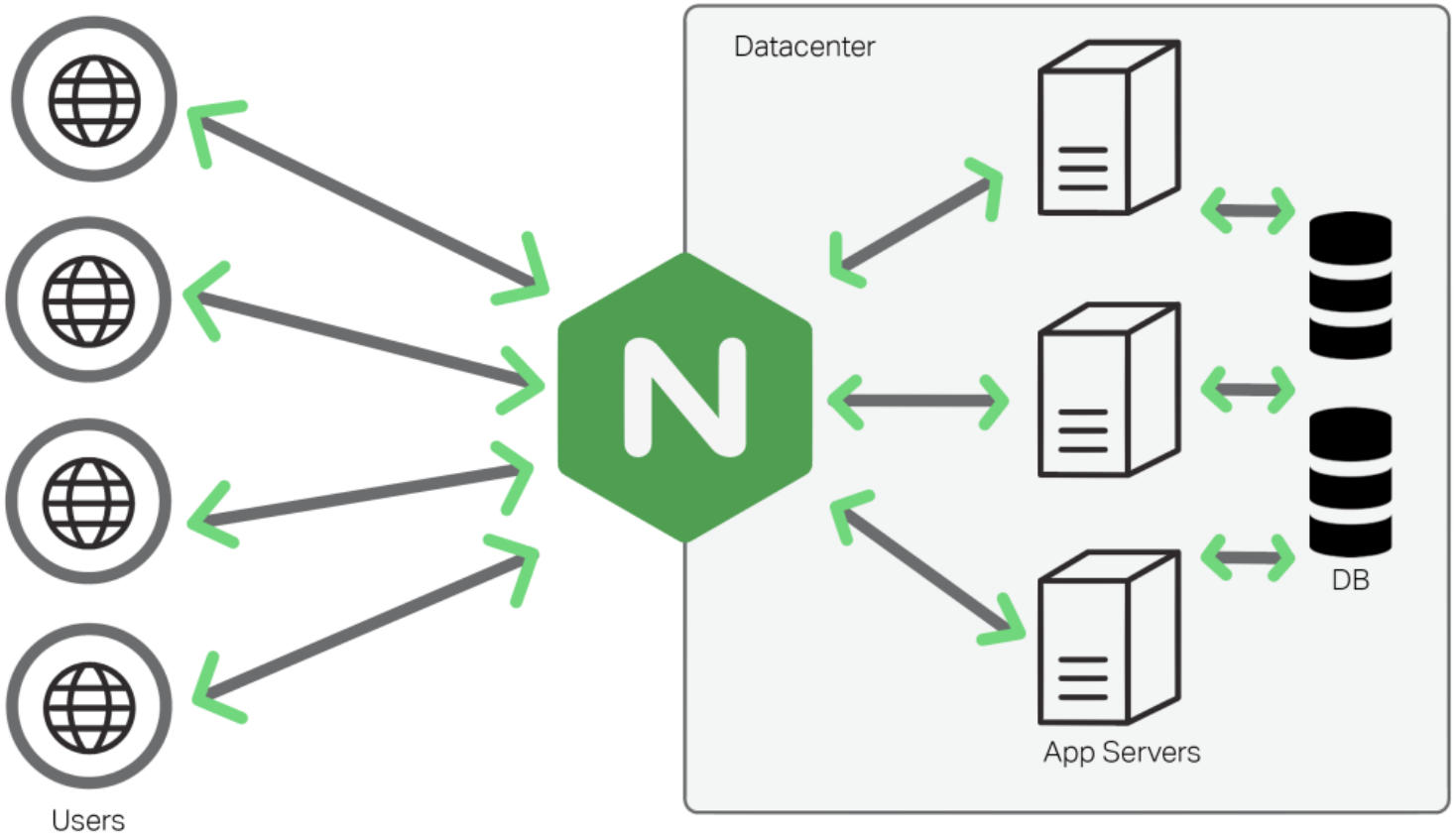
OR



You input SQL, and execute it
Show the result

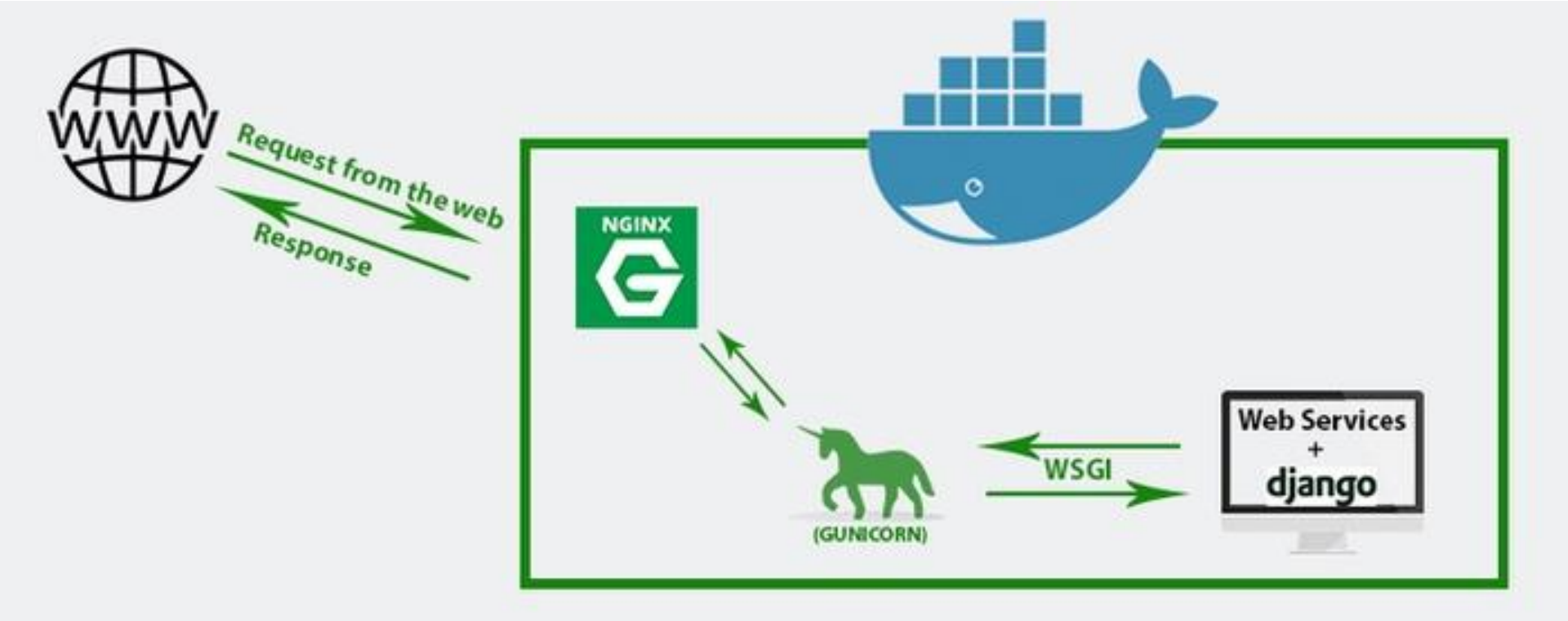
More complex architecture
In Big Data age

- + Nginx



More complex architecture
In Big Data age

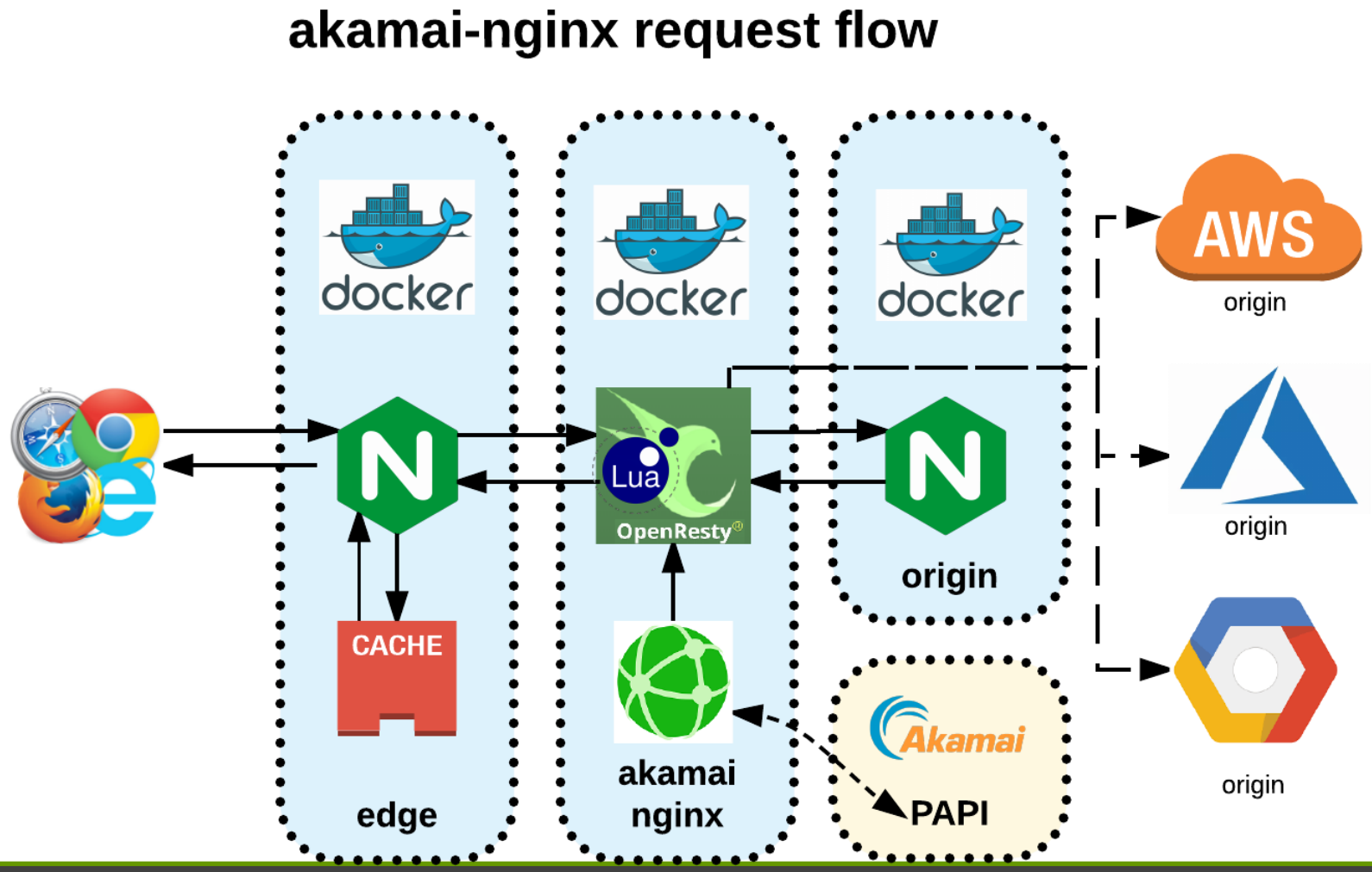
- Linux + Docker, Nginx, Django etc.



https://www.codementor.io/samueljames/nginx-setting-up-a-simple-proxy-server-using-docker-and-python-django-f7hy4e6jv

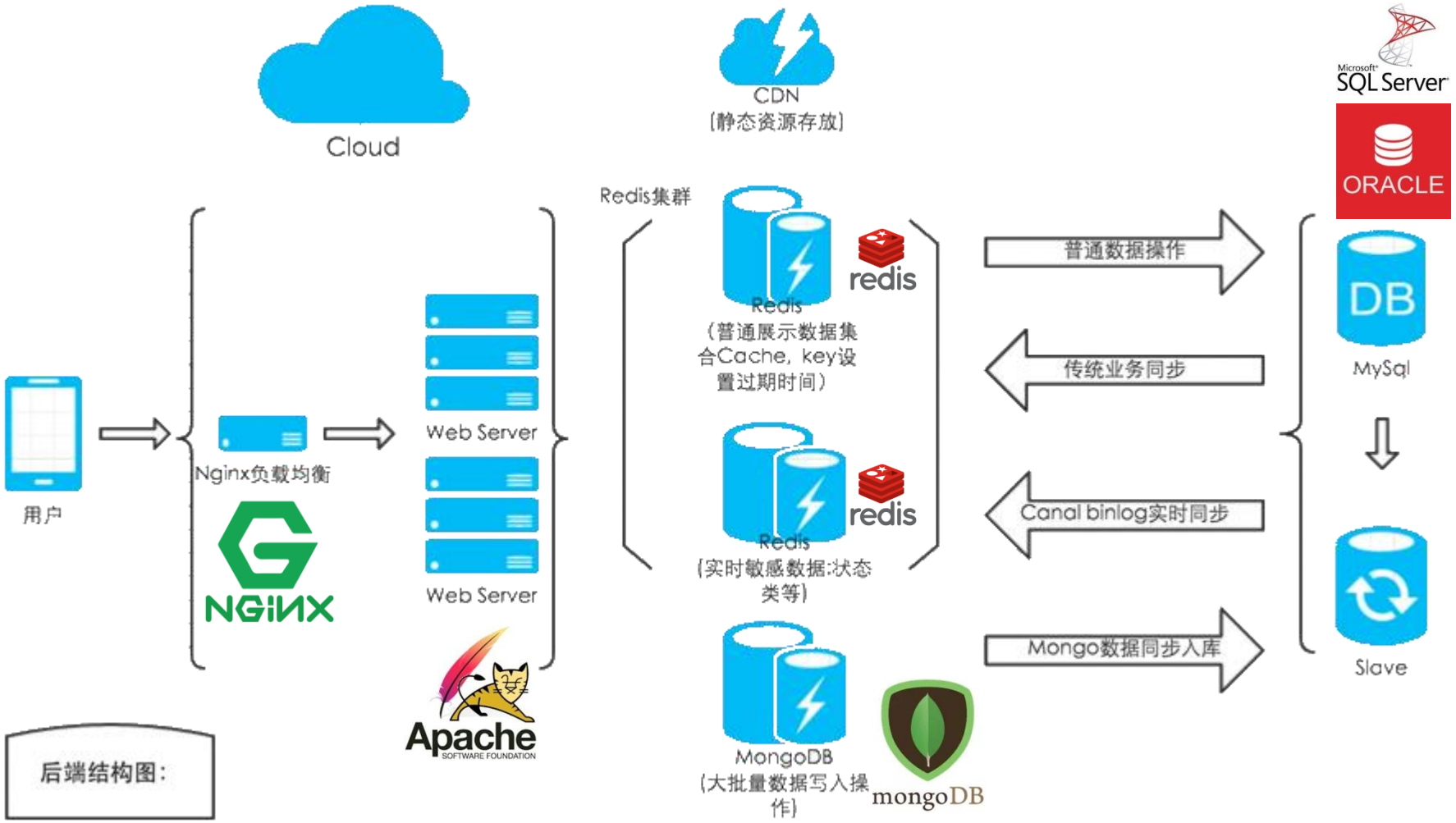
More complex architecture
In Big Data age

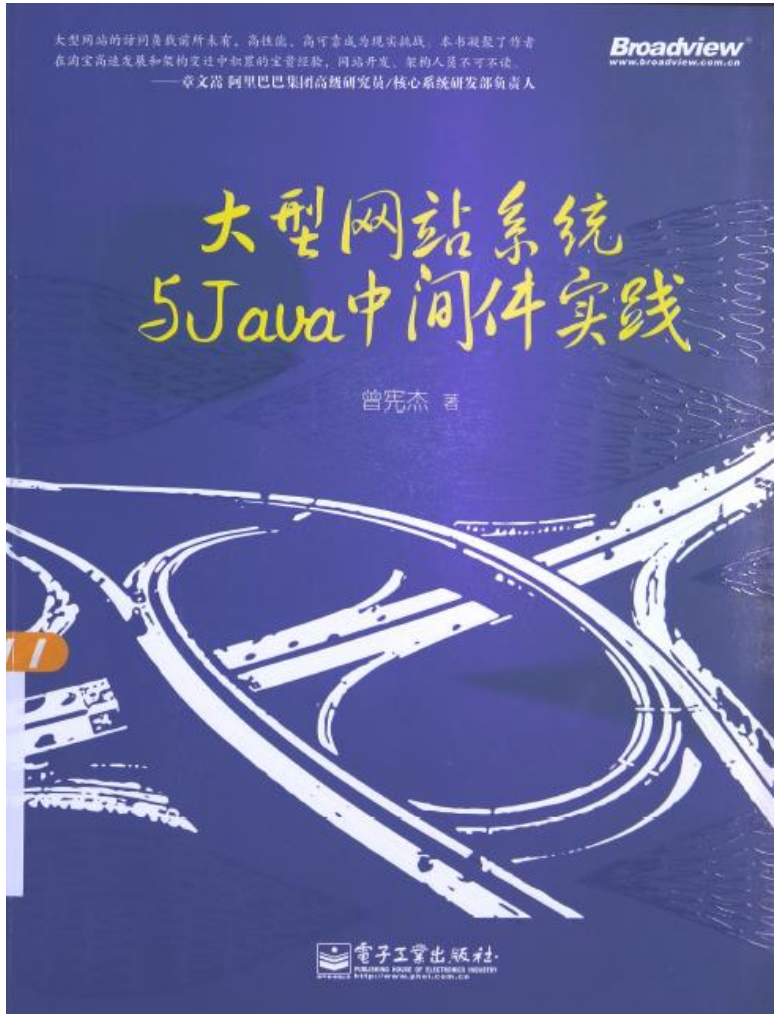
- Linux + Docker, Nginx, Lua etc.



Akamai PAPI is called once, to generate equivalent LUA config

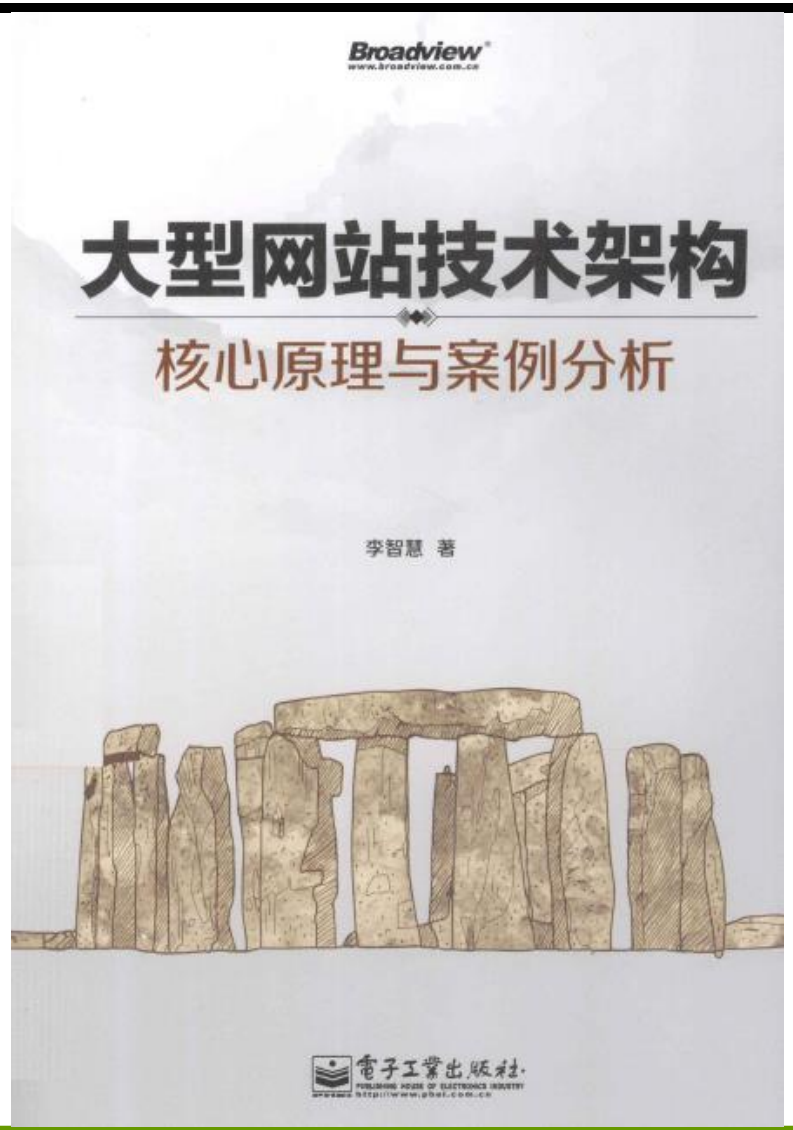
https://github.com/wyvern8/akamai-nginx

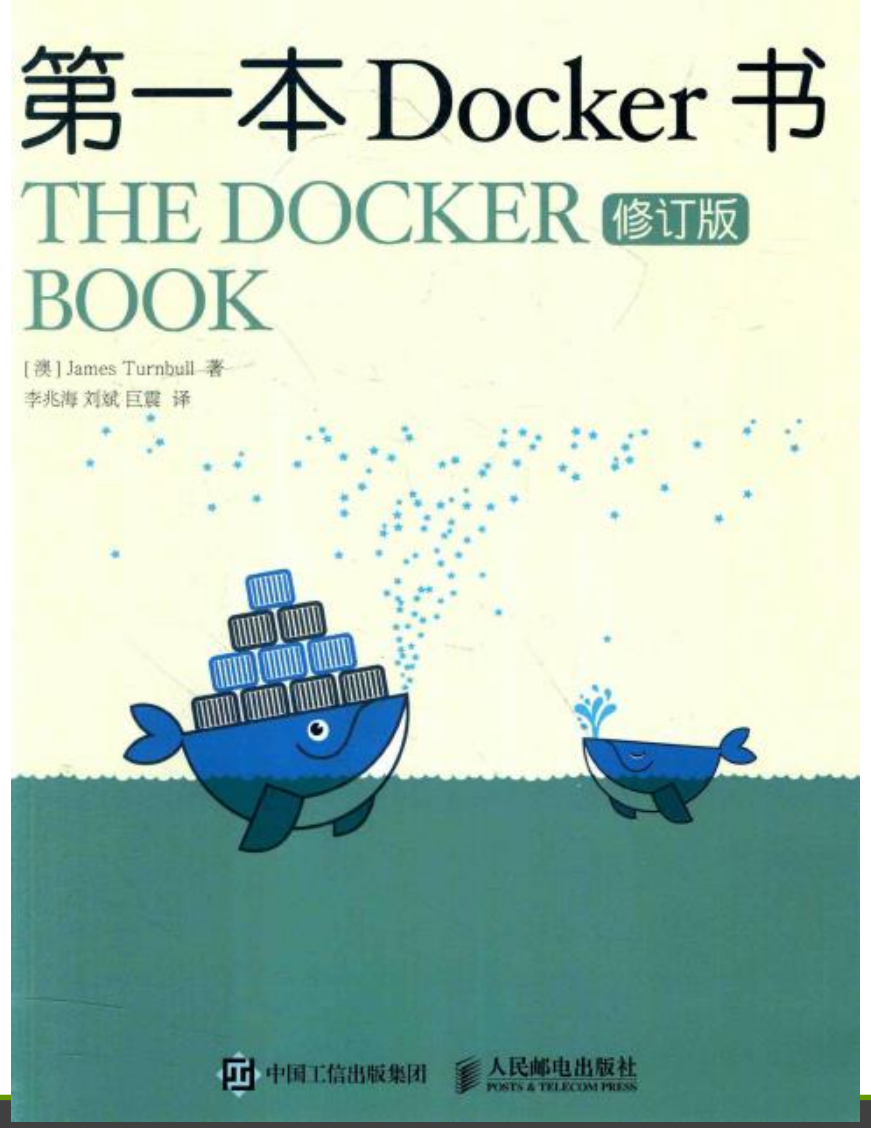

Cloud
CDN
(静态资源存放)
Redis集群
用户
Nginx负载均衡
NGINX
Web Server
Web Server
Apache
SOFTWARE FOUNDATION
Redis
(普通展示数据集合Cache，key设置过期时间)
redis
Redis
(实时敏感数据:状态类等)
redis
MongoDB
(大批量数据写入操作)
mongoDB
普通数据操作
传统业务同步
Canal binlog实时同步
Mongo数据同步入库
Microsoft SQL Server
ORACLE
DB
MySql
Slave
后端结构图:



- **大型网站系统与Java 中间件实践**

- **然后再看看下面这本书——更详细的技术细节**
  - 亿级流量网站架构核心技术


Broadview
www.broadview.com.cn
大型网站技术架构
核心原理与案例分析
李智慧 著
電子工業出版社



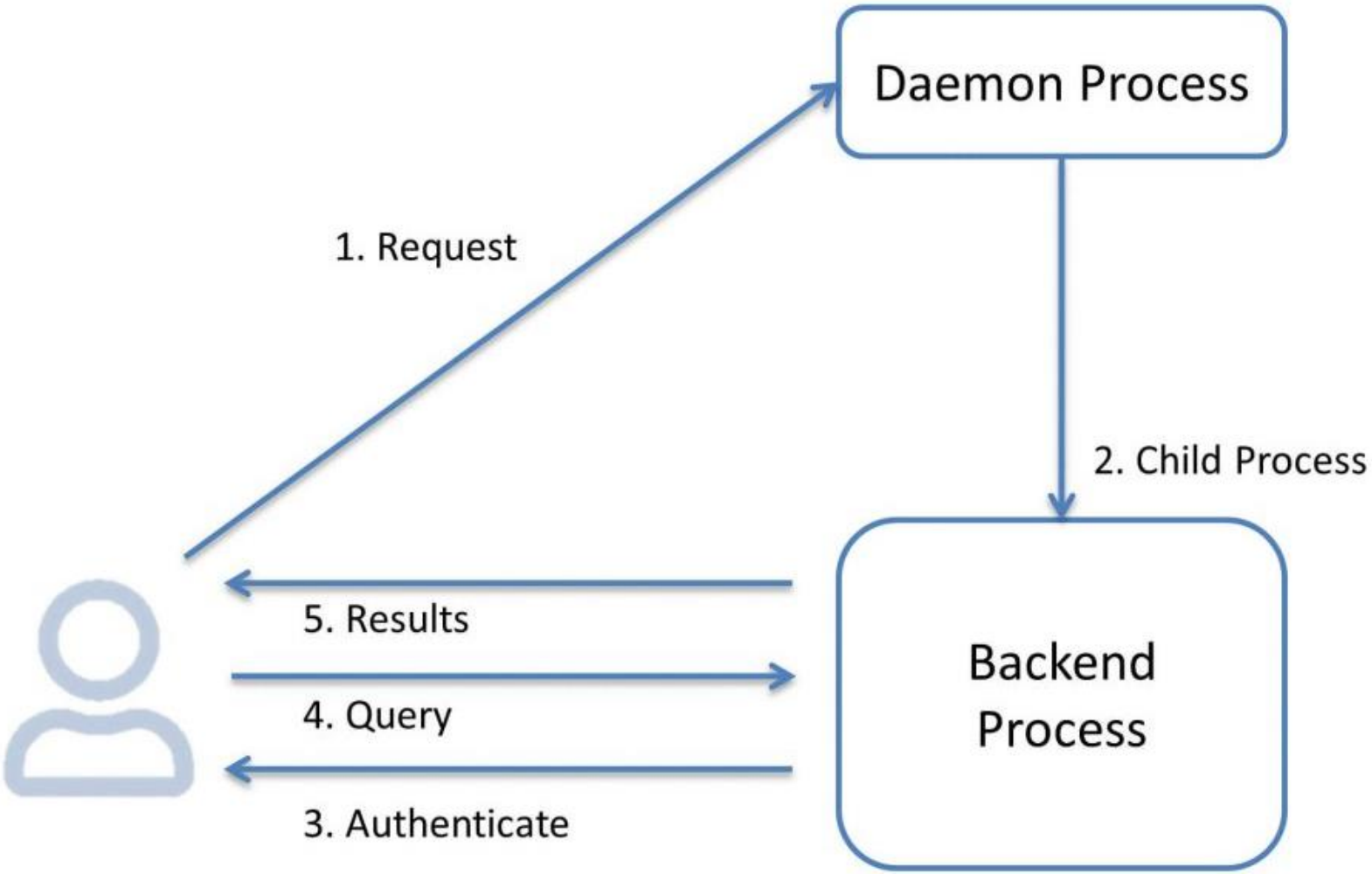
- 第一本Docker 书
- The Docker Book

The way this semester to evaluate 100 pts based on

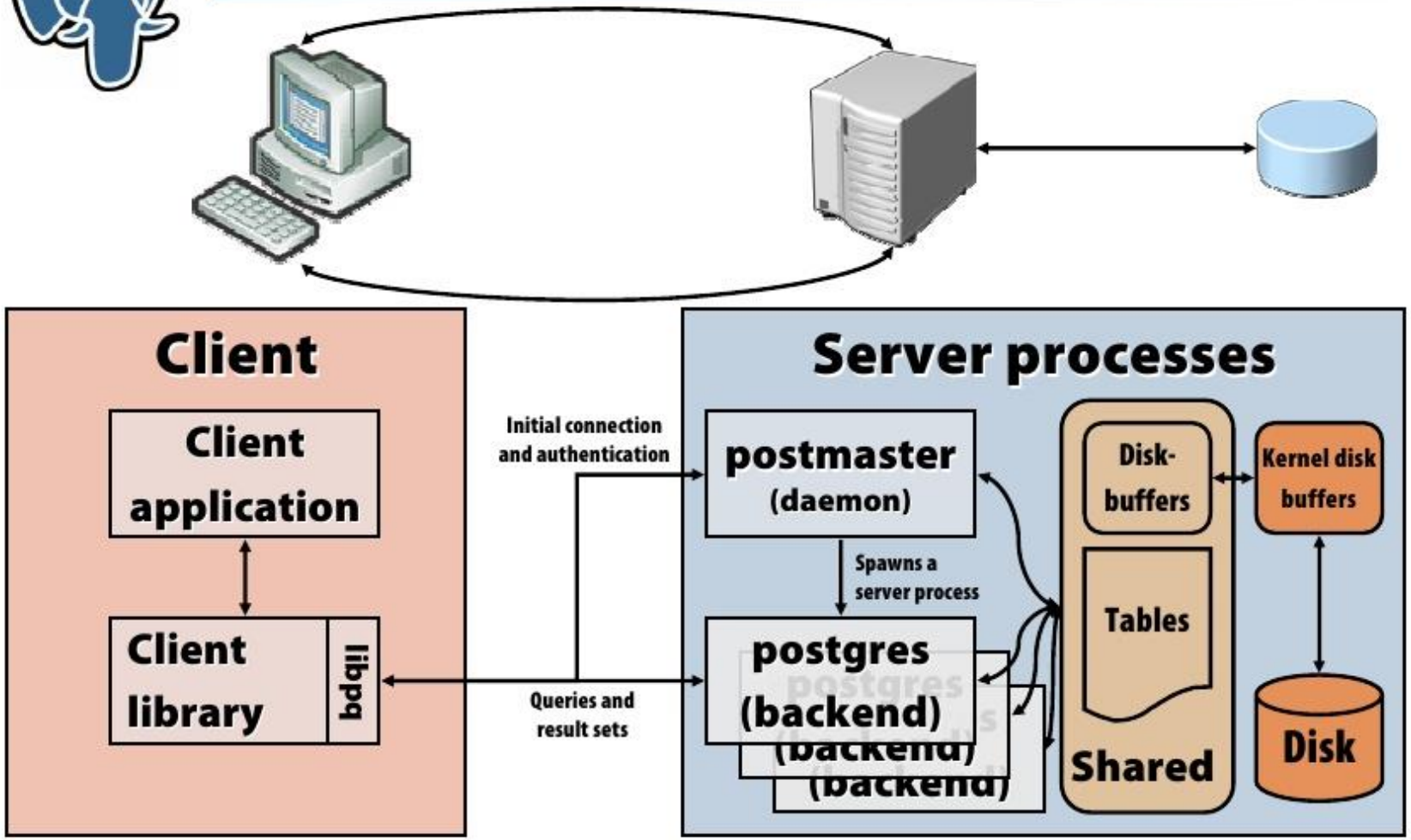
- Basic (70)
  1.
  2.
  3. Try a Syntax parsing example
  4. Lock table management
  5.
-

- Understanding [multi-process]

- Advanced (10) in PostgreSQL/Greenplum/HAWQ
  - [transaction mechanism][SQL execution]


Daemon Process
1. Request
2. Child Process
5. Results
4. Query
3. Authenticate
Backend Process

Overall architecture



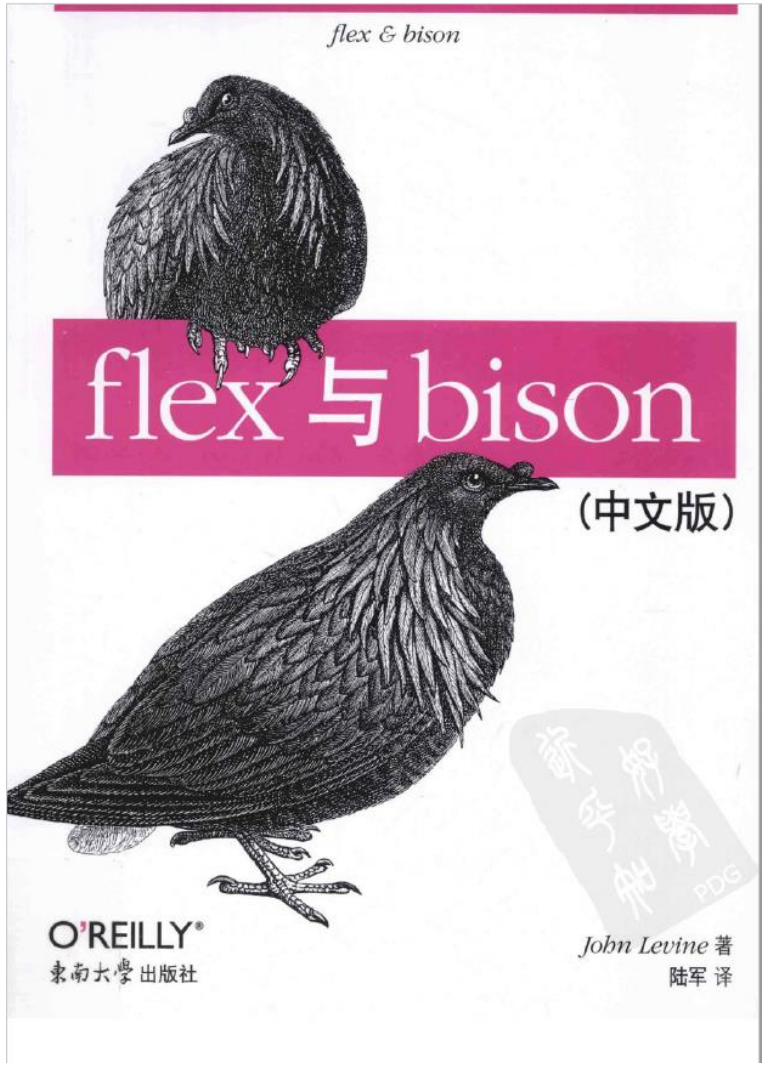

```
23961  Ss     0:05.64 /usr/local/bin/postmaster                (postgres)
23963  S      0:01.13 postmaster: stats buffer process         (postgres)
23966  S      0:03.24 postmaster: stats collector process      (postgres)
36324  I      0:00.43 postmaster: oddbjorn testdb [local] idle (postgres)
36428  I      0:00.23 postmaster: oddbjorn testdb [local] idle (postgres)
```

The way this semester to evaluate 100 pts based on

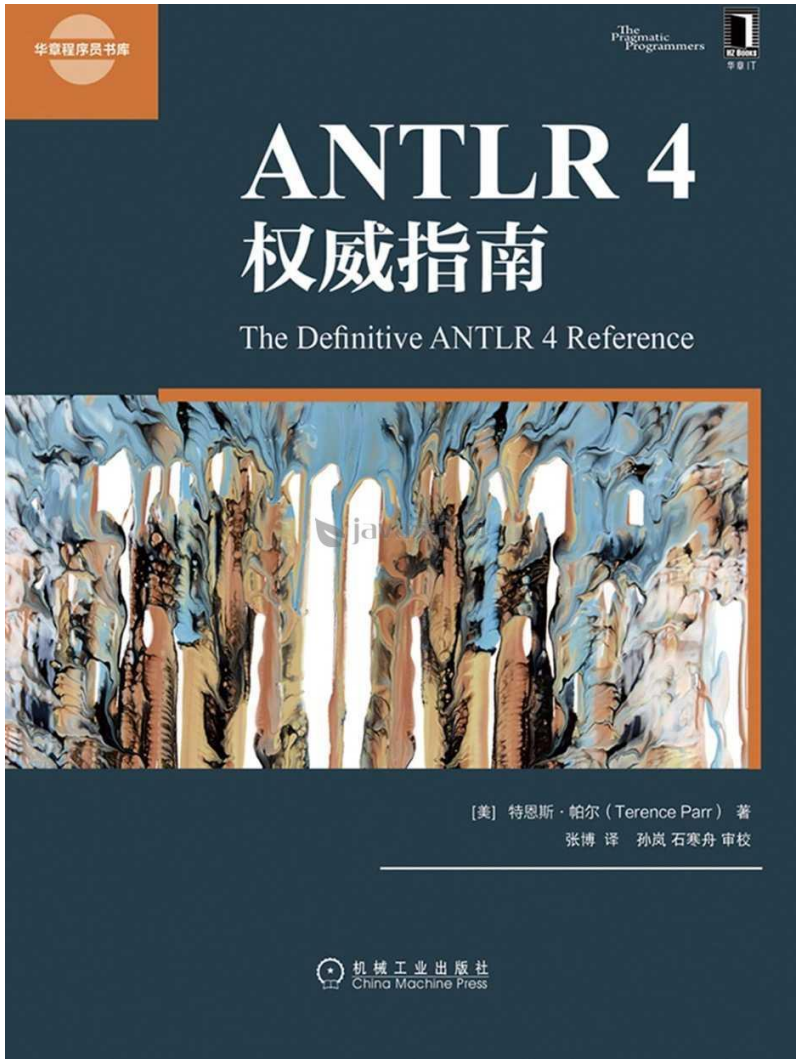
- Basic (70)
  1.
  2.
  3. Try a Syntax parsing example
  4. Lock table management
  5.
-

Later

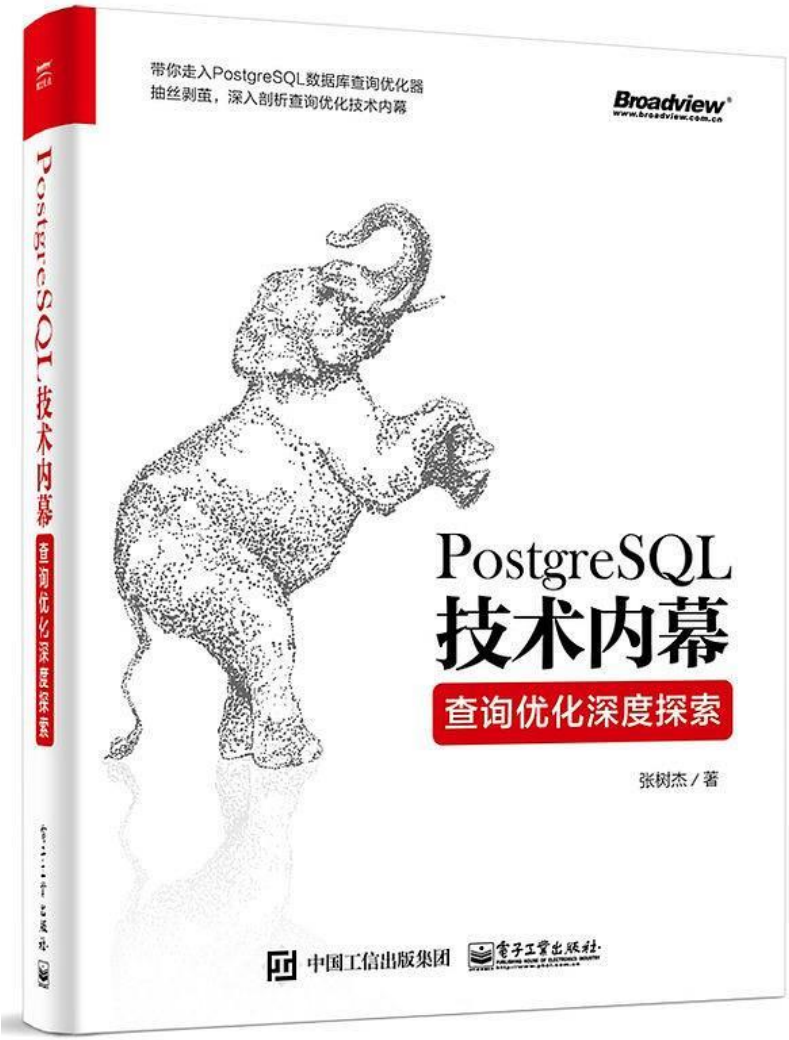
- Advanced (10)] in PostgreSQL/Greenplum/HAWQ
  - [transaction mechanism][SQL execution



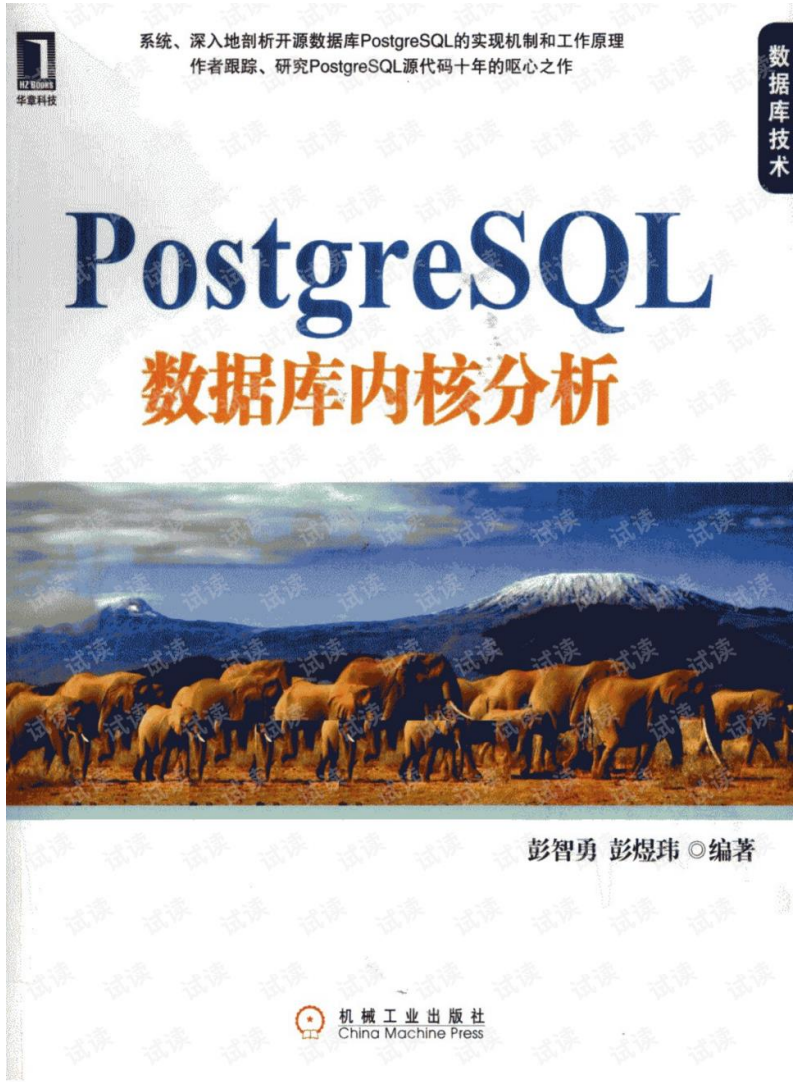
- flex & bison - Unix Text Processing Tools
- *John Levine*
- O'Reilly Media
- 2009



- ANTLR 4权威指南
- 作者: Terence Parr
- 出版社: 机械工业出版社
- 译者: 张博
- 出版年: 2017-5-1
- 页数: 262
- 定价: 69元
- 丛书: 华章程序员书库
- ISBN: 9787111566489



- PostgreSQL技术内幕：查询优化深度探索
- 作者: 张树杰
- 出版社: 电子工业出版社
- 出版年: 2018-6
- 定价: 79
- ISBN: 9787121341489



- **PostgreSQL 数据库内核分析**
- 作者: 彭智勇 / 彭煜玮
- 出版社: 机械工业出版社华章公司
- 出版年: 2012-1