Insight into DBMS:

Design and Implementation

Chapter 3:

My Understanding of DBMS

孔令波

mlinking@126.com

+86 15010255486

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**)

Chapter 2: Overview of DBMS

- **My understanding about DBMS**
  - History of (R)DBMS
  - Overview of DBMS's Inside

Do you know the top goals of this S.S.E?

你们知道我们对你们的期望吗?

- **SHOWING YOU SOME LIFE-MAKING SKILLS (PROGRAMMING IS ONE)!**

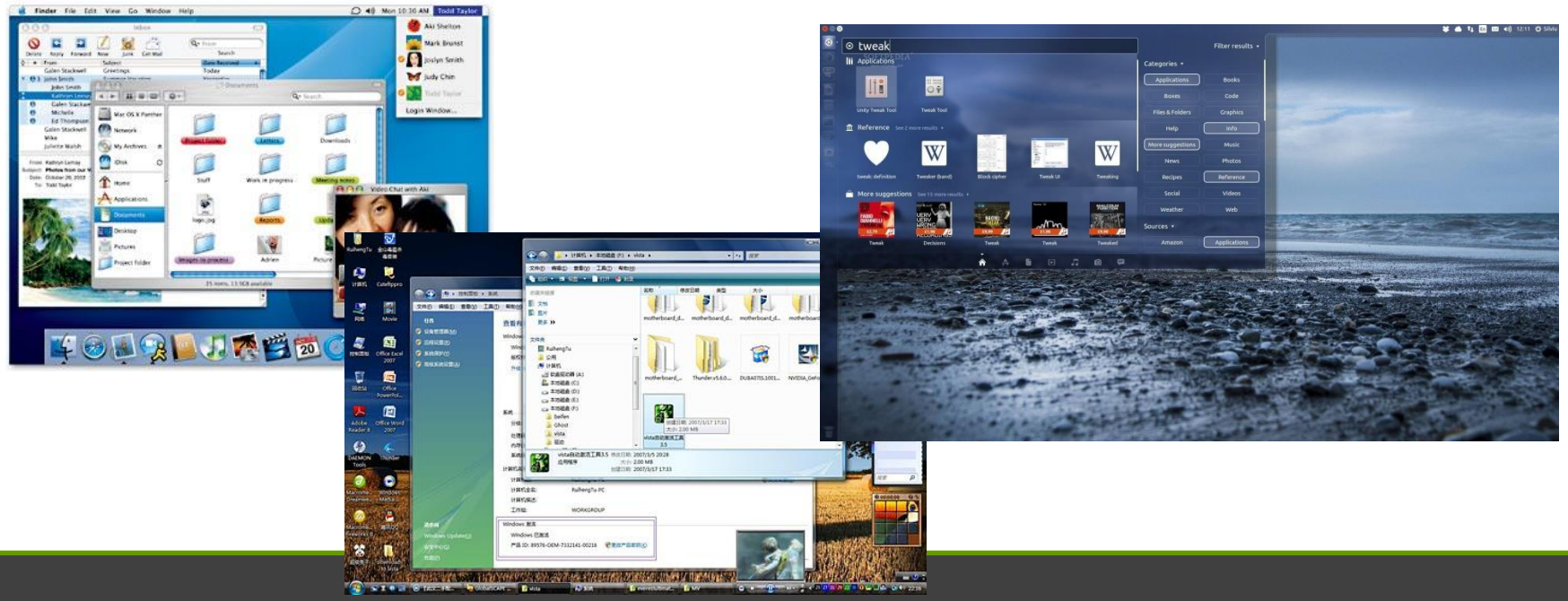
4th	**Programming** in IT companies	**Intern practice** **- Try yourself in companies**
3rd	**BI/Data Analytics**, SE, BA, Frontier topics, …	**Widen your ideas and skills**
2nd	DSA, **OS**, **DBMS**, Web **programming**	**Improving**
1st	Math, **Programming (with C)**, OO (Java)	**Fundamental programming**

- **The understanding of the design and implementation (D&I) of those classic softwares is helpful for us to improve our programming skills**
  - DBMS is one!
  - I benefit a lot from reading the source code of PostgreSQL written in C

- **Many problems we should consider carefully to cope with so as to promote yourself!**
  - **“只做简单的事, 是不能超拔出群的”**
  - The way to be outstanding is to do what others could not do
    - BUT, NEVER DO BAD THINGS! ☺

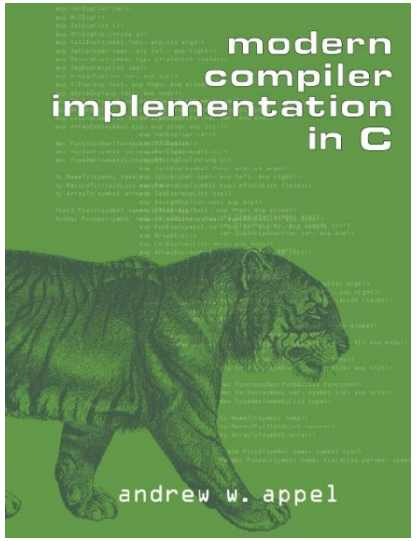
Learning from classics!

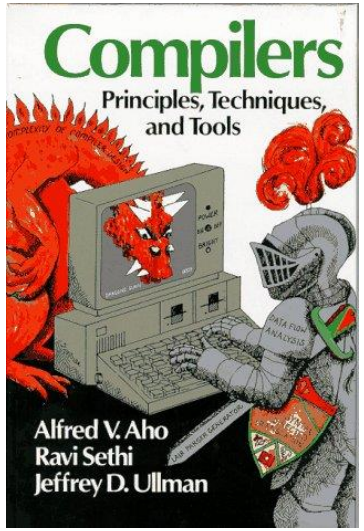
- **DBMS is one of the three classic softwares/programs for Computer Science**
  - **OS** (Operating System)
    - ➢ The **INDISPENSABLE** one for users to use computers
      - ✓ to support the CONCURRENT execution and of many programs and FRIENDLY interface for many users

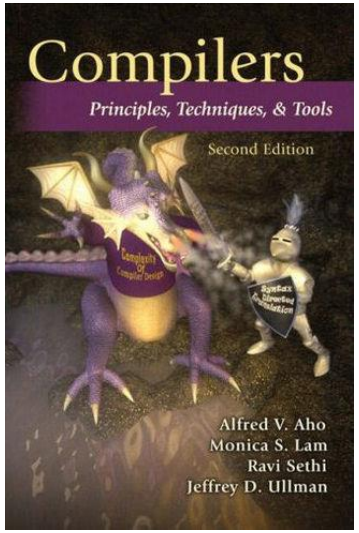


Learning from classics!

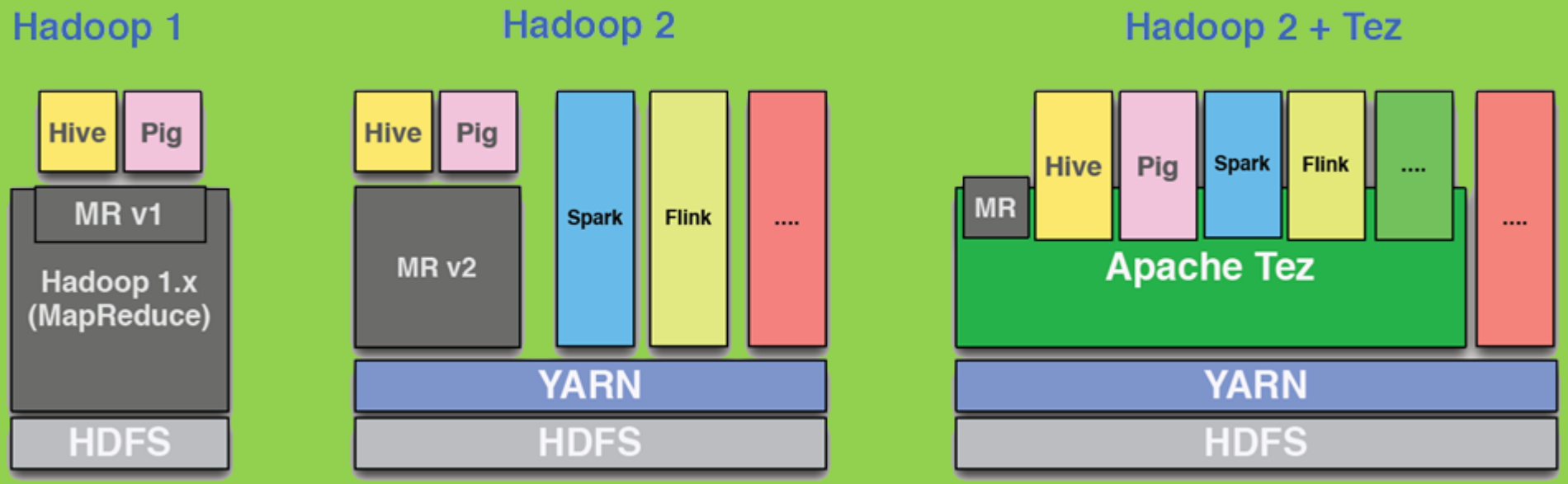
- **DBMS is one of the three classic softwares for Computer Science**
  - **Compiler**
    - Translator to convert source code into executable code (BIN)
      - ✓ EXE is popular for Windows
      - ✓ **a.out**（assembler and link editor output 汇编器和链接编辑器的输出）、**COFF**（Common Object File Format 通用对象文件格式）、**ELF**（Executable and Linking Format 可执行和链接格式） for UNIX/LINUX







- **Recently I believe there should be the 4th classic software**
  - Big Data
    - Hadoop/MapReduce, HIVE, Spark, HAWQ, etc.

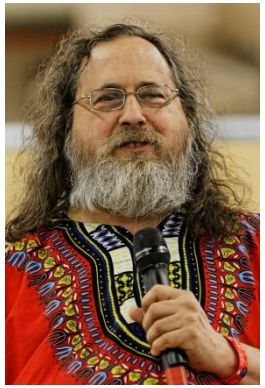


https://stackoverflow.com/questions/39411888/why-would-someone-run-spark-flink-on-tez

https://en.wikipedia.org/wiki/Richard_Stallman

https://en.wikipedia.org/wiki/Open-source_software_movement

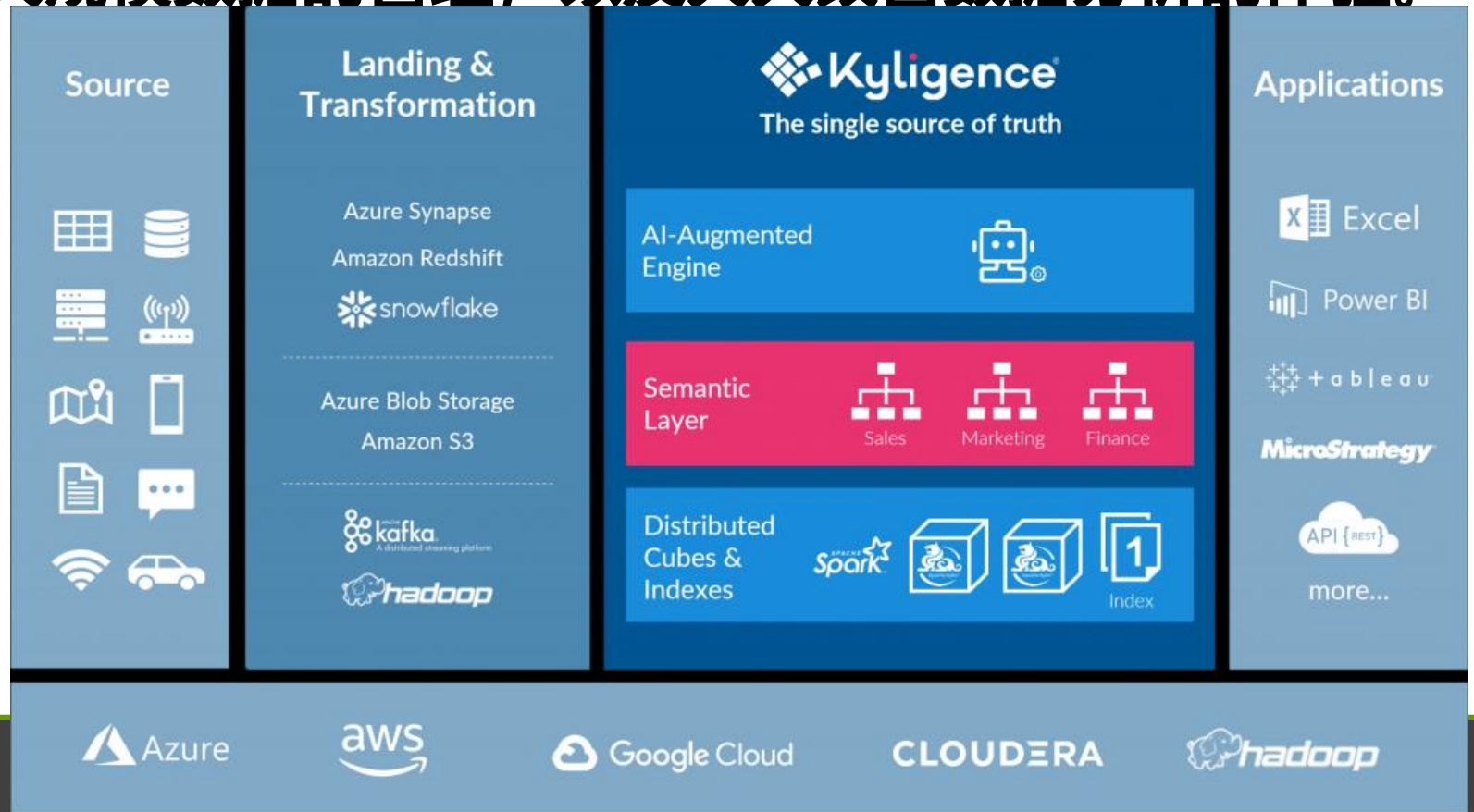
- **Thanks to OPEN SOURCE projects – we can learn amazing programming skills from talented guys**
  - You benefit, you share!
    - The Open Source Movement is branched from the free software movement which began in the late 1980s with the launching of the GNU project by Richard Stallman.
  - Many classic softwares
    - OS – BSD Unix, **Linux**, CentOS, **Android**, …
    - DBMS – **PostgreSQL**, **HyperSQL**, Cobase, …
    - Web server – **Apache**, Nginx, …
    - Big Data – Redis, **Hadoop**/MapReduce, **Spark**, …



Kylin – 2015年首个来自中国的 Apache 顶级项目，和 Hadoop，Spark 等处于一个级别

https://cn.kyligence.io/

- **Kyligence 的 AI 增强的数据平台为分析师和最终用户提供全组织统一的、整合过的及优化过的全局数据视图，不论是在数据中心还是多云部署，Kyligence 帮助企业发现和管理最有价值数据，进一步简化超大规模数据的管理，以及大大改善数据分析的体验。**



- **在实时数仓建设中，解决方案成熟，消息队列Kafka、Redis、Hbase鲜有敌手，几乎已成垄断之势。而OLAP的选择则制约整个实时数仓的能力。**
- **目前市面上主流的开源OLAP引擎包含不限于：Hive、HAWQ、Presto、Kylin、Impala、SparkSQL、Druid、Clickhouse、Greeplum等，可以说目前没有一个引擎能在数据量，灵活程度和性能上做到完美，用户需要根据自己的需求进行选型**
  - Kylin自身就是一个MOLAP系统，多维立方体（MOLAP Cube）的设计使得用户能够在Kylin里为百亿以上数据集定义数据模型并构建立方体进行数据的预聚合
  - 将数据先进行预聚合，然后把多维查询变成了key-value查询

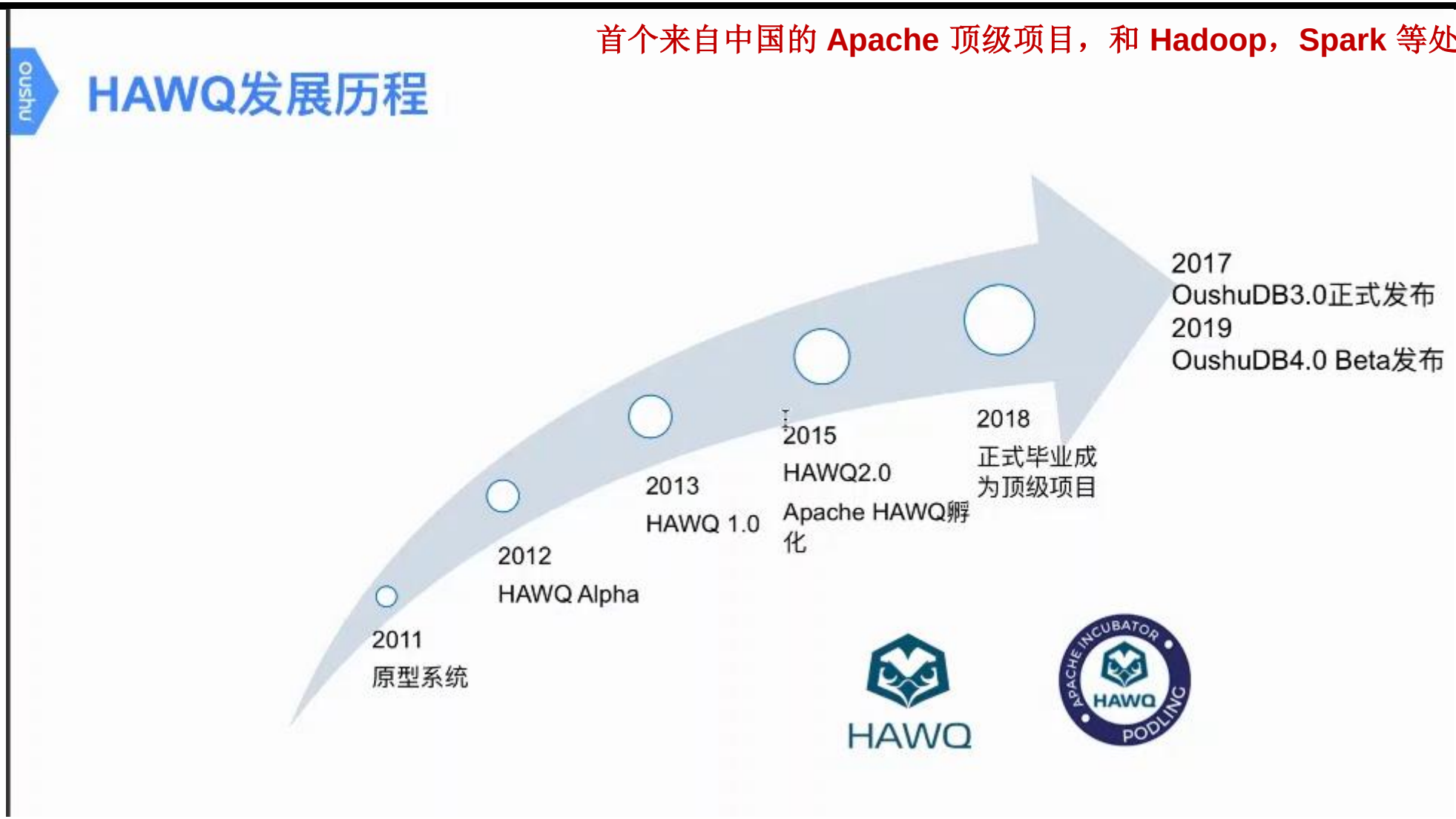
2016 年创业，HAWQ 是2012年就开始了(EMC)

http://hawq.apache.org/

http://www.oushu.io/ch/

2018年 – Apache 顶级项目 (Kylin 是2015年)

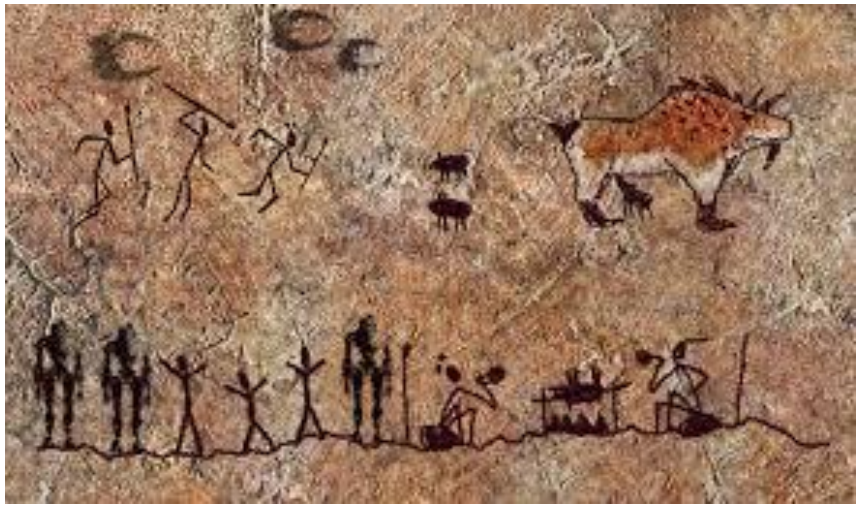
首个来自中国的 **Apache** 顶级项目，和 **Hadoop**，**Spark** 等处于一个级别



My understanding about DBMS

- **DBMS – Database Management System**
  - The kernel is definitely "**Data Management**"!
    - I.D.U.S
- **By that, a long history**
  - Early human records-clay tablets, hieroglyphics, cave paintings, paper records of family histories, treaties, inventories, and so on

Hieroglyphics [ˌhaiərəˈglifiks]
hieroglyphist [ˌhaiərəˈglifist]
n. 象形文字研究者,书写象形文字者



cave painting

Prehistoric stone painting

- All over the world – painting is the instinct!

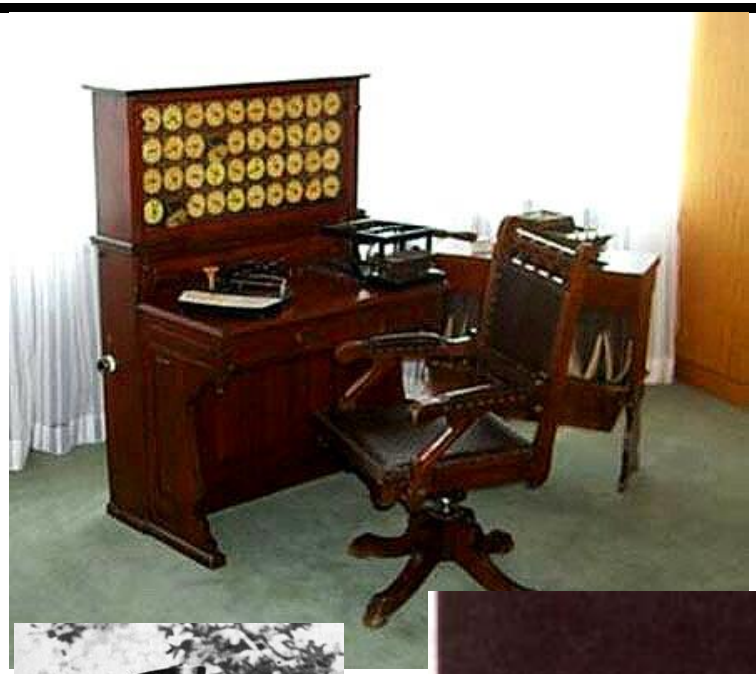
http://www.baike.com/wiki/%E9%98%BF%E5%B0%94%E5%A1%94%E5%B2%A9%E7%94%BB&prd=button_doc_jinru

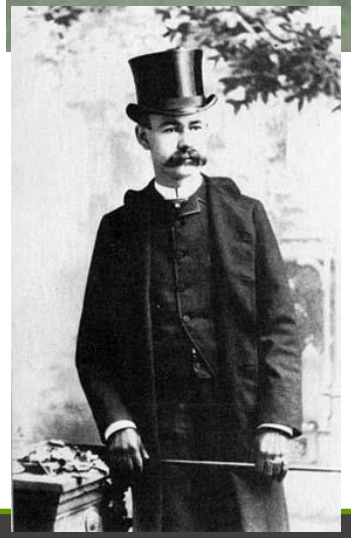
Oracle-bone inscription

Piles of paper files

Recently – before IT age

- **Hollerith used punched cards in 1890 US census**
  - Later **IBM**, the **BIG BLUE**!
- **Punched paper tape introduced in 1940s**





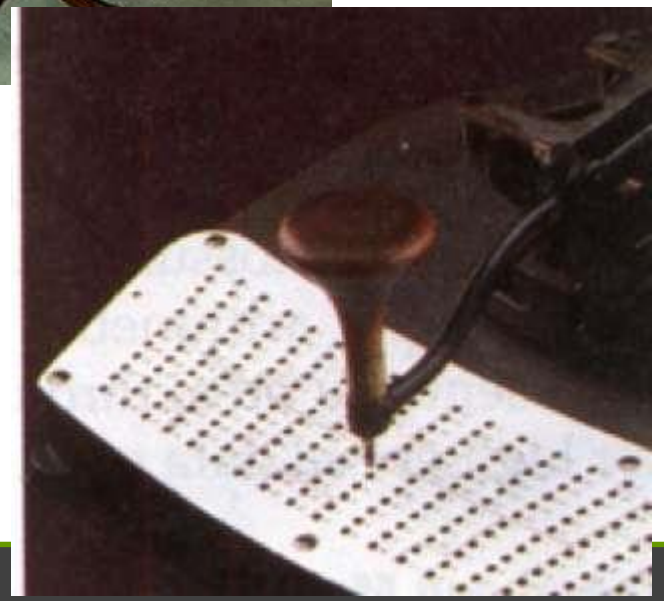


fig from www.columbia.edu/acis/history/census-tabulator.html

Brief History of Information Systems

- **Magnetic tape introduced about 1950-used in UNIVAC I**
  - UNIVersal Automatic Computer
  - Cards, paper tape, magnetic tape are sequential access devices
  - Used in sequential processing applications such as payroll

payroll - 必应词典

payroll 美 [ˈpeɪˌroʊl] 英 [ˈpeɪˌrəʊl]

na. 职工名册；发薪簿；应付薪金额；(计算机的)计算报表

n. （公司员工的）工资名单；（公司的）工资总支出

网络 工资表

变形 复数：payrolls；

例句：The general manager is trying to meet the payroll. 总经理在设法筹措款项支付工资.

查看更多释义

cn.bing.com/dict/payroll · 2019-9-18

So, by DATA MANAGEMENT now in IT age

- **FILE is the basis**
  - Device drivers encapsulate the "*evil*" details of physical devices and provide uniform APIs for programmers to manage data easily – stored as files in permanent storage media
  - APIs – C Interface
    - # include <stdio.h>
    - FILE *fopen(char *path, char *type);
      - ✓ int feof(FILE *stream);
      - ✓ int fseek(FILE *stream, long offset, int origin);
      - ✓ int fscanf(FILE *stream, const char *format, ...);
      - ✓ int fprintf(FILE *stream, const char *format, ...);
    - int remove(char *path);
    - int fclose(FILE *fp);

https://en.wikipedia.org/wiki/C_file_input/output

Basic File Terminology

TABLE 1.1 BASIC FILE TERMINOLOGY

TERM	DEFINITION
Data	"Raw" facts, such as a telephone number, a birth date, a customer name, and a year-to-date (YTD) sales value. Data have little meaning unless they have been organized in some logical manner. The smallest piece of data that can be "recognized" by the computer is a single character, such as the letter A, the number 5, or a symbol such as /. A single character requires one byte of computer storage.
Field	A character or group of characters (alphabetic or numeric) that has a specific meaning. A field is used to define and store data.
Record	A logically connected set of one or more fields that describes a person, place, or thing. For example, the fields that constitute a record for a customer named J.D. Rudd might consist of J.D. Rudd's name, address, phone number, date of birth, credit limit, and unpaid balance.
File	A collection of related records. For example, a file might contain data about vendors of ROBCOR Company; or a file might contain the records for the students currently enrolled at Gigantic University.

Here with C – Data M

You can finish I.D.U.S based on those IO functions – old way

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

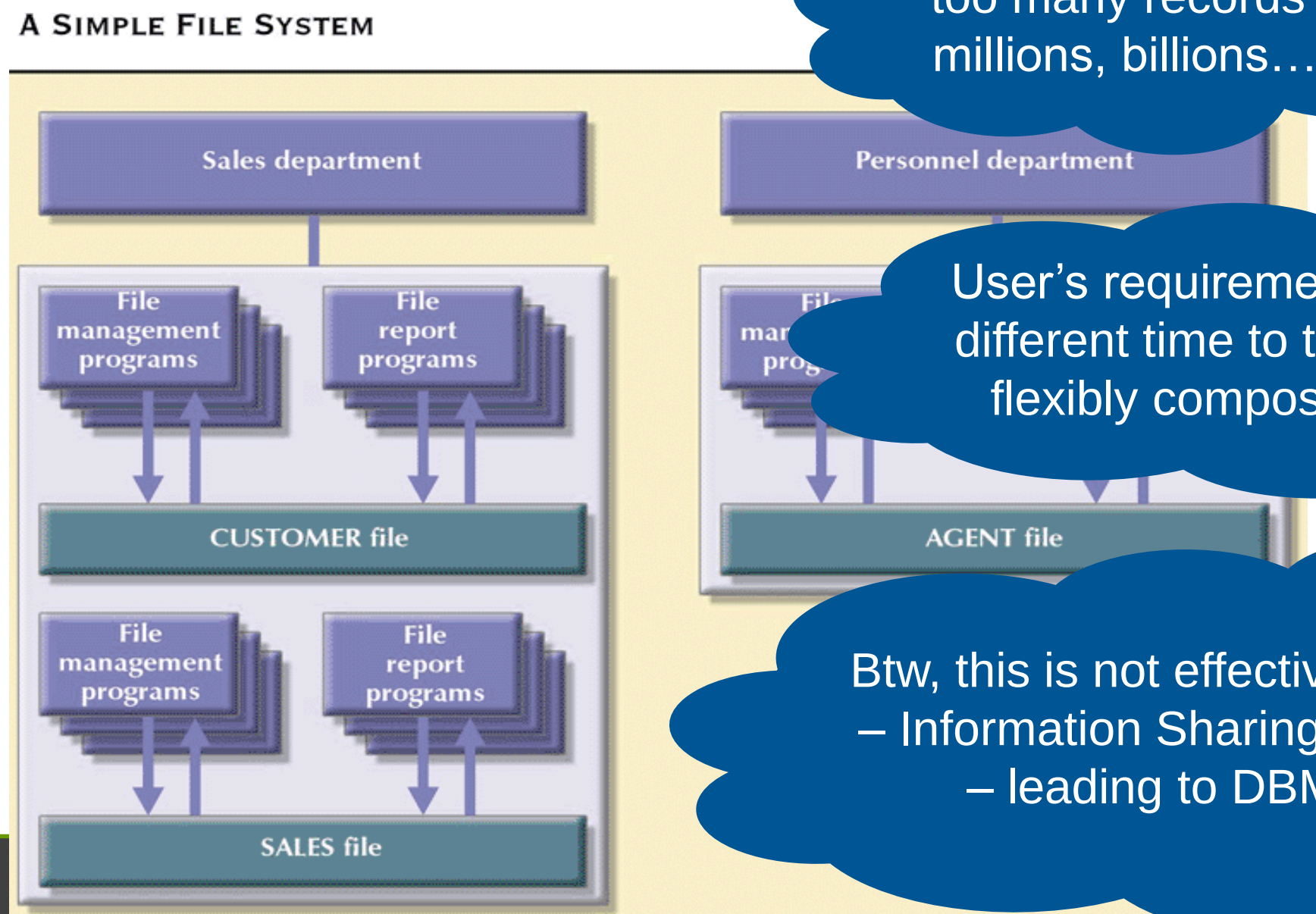
u know

- How to store the data shown in left?
- You define
  - define struct{
    int id;
    string name;
    string dept_name;
    int salary;
    } instructor;
- Use some IO functions
  - scanf()
  - printf()
  - seek()…

Old way to manage data – Simple File System

- Each file was used by its own application programs
- Each file was owned by individual or department who commissioned its creation

File based data management syste



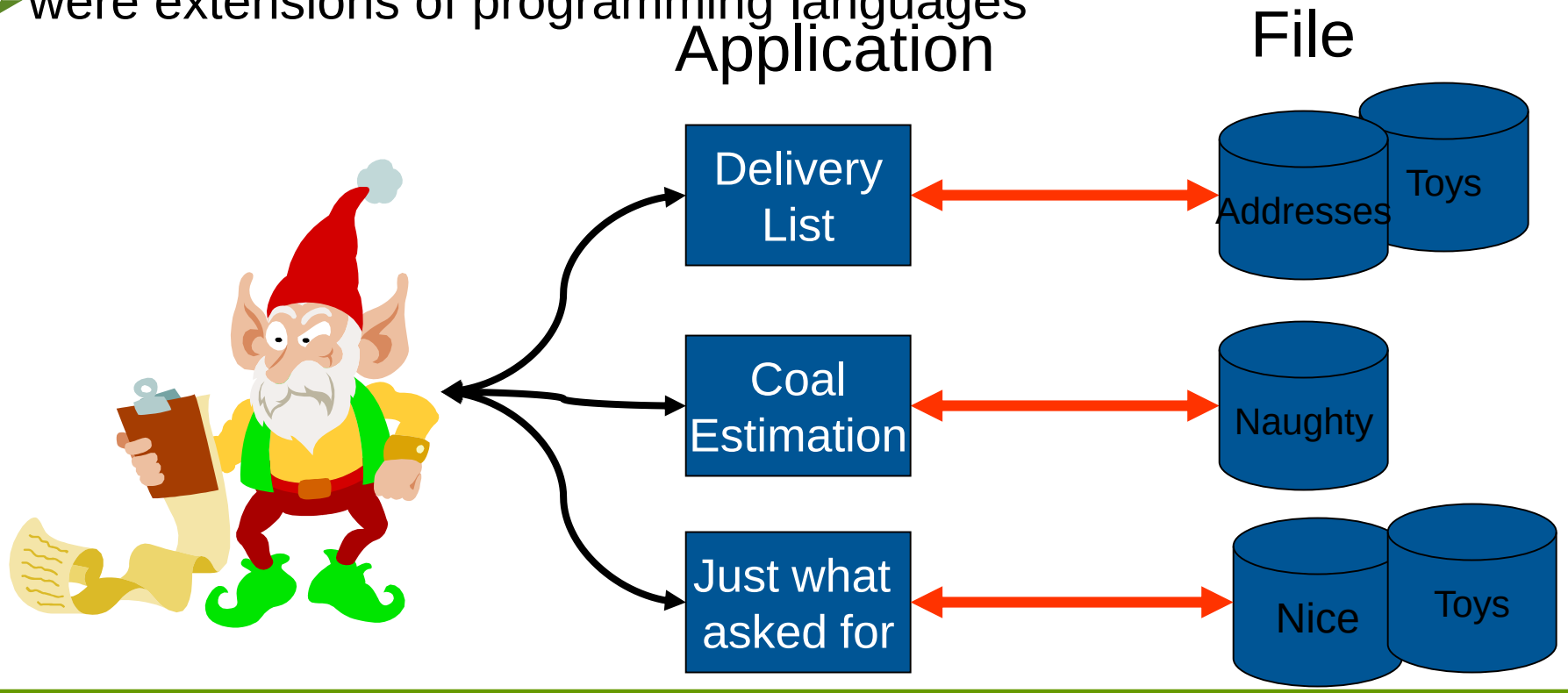
However, if there are too many records – millions, billions…?

User's requirements are quite different time to time? How to flexibly compose queries?

Btw, this is not effective for sharing – Information Sharing is important – leading to DBMS later

1950's and 1960's - file based data management system

- all applications were custom built for particular needs
  - Many similar/duplicative applications dealing with collections of business data
  - were extensions of programming languages

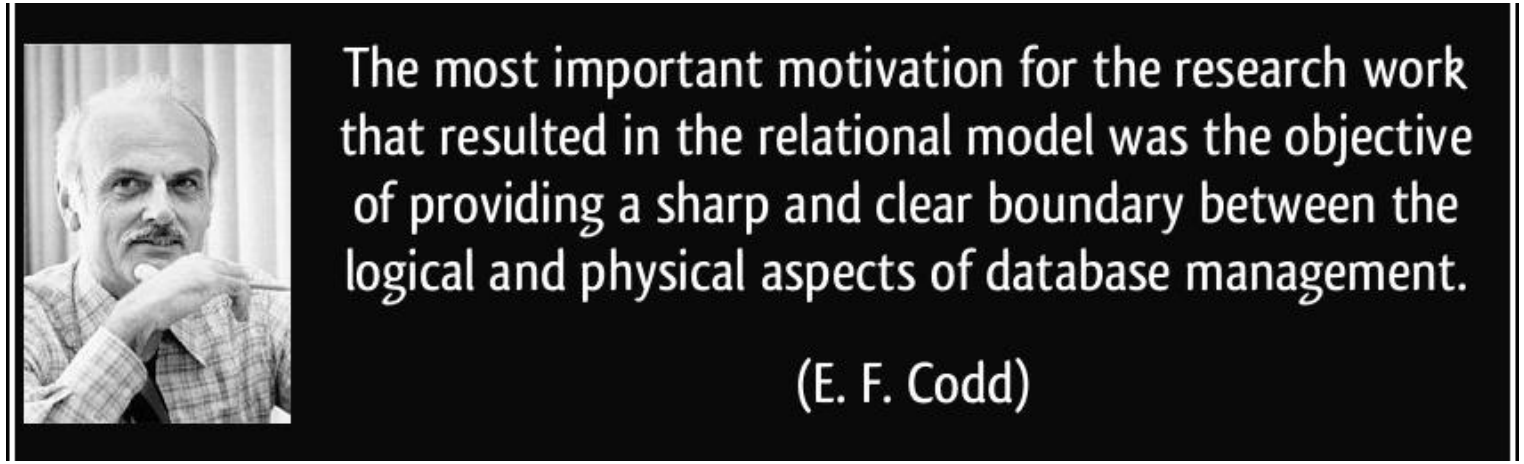


Problems with file processing systems

- **Inconsistent** data
- Inflexibility
- Limited data sharing
- Poor enforcement of standards
- Excessive program maintenance

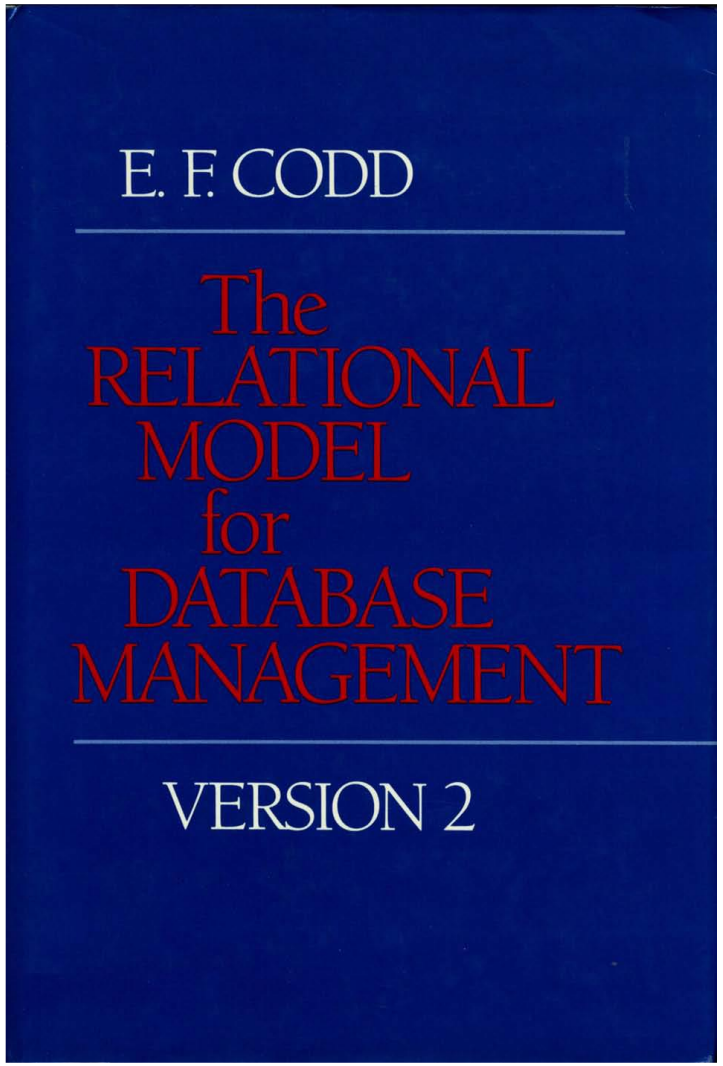
Edgar Frank Codd
Born: August 23, 1923
Died: April 18, 2003 (aged 79)

1970 - E. F. Codd and the Relational Model



- I prefer to use the top goal and 2 roles to indicate the benefit of DBMS
  - Top goal:
    - ✓ Support the **concurrent** **access of the data**
  - 2 roles:
    1. Concurrent data management by many users/programs
    2. Provide friendly/flexible interaction for users

https://zh.wikipedia.org/zh-hans/埃德加·科德



- The Relational Model for Database Management
- 作者: E. F. Codd
- 出版社: Addison Wesley Publishing Company
- 副标题: Version 2
- 出版年: 2000-5
- 页数: 538

https://book.douban.com/subject/3740120/

Example: Joining Two Tables

Enrolled

cid	grade	sid
Carnatic101	C	53666
Reggae203	B	53666
Topology112	A	53650
History105	B	53666

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Selection (σ)
- Projection (Π)
- Union ($\cup$)
- Intersection ($\cap$)
- Difference ($-$)
- Product ($\times$)
- Join ($\bowtie$)

- **Pre-Relational:**

  write a program

- **Relational SQL**

  Select name, cid from students s, enrolled e where s.sid = e.sid

- **Relational Algebra**

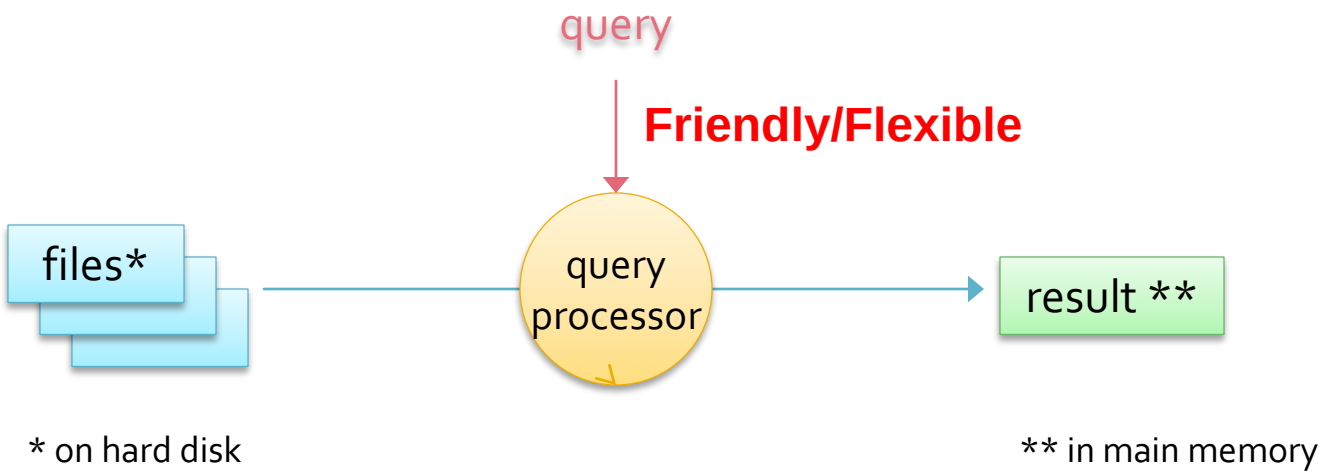
$$\pi_{name,cid}(Students \bowtie_{sid} Enrolled)$$

- **Relational Calculus**

{R | R.name = S.name ∧ R.cid = S.cid ∧
∃S(S∈Students∧ ∃E(E∈Enrolled ∧ E.sid = S.sid))}

Simple DBMS

- **Here is a naïve sketch of a Database Management System (DBMS)**
  - The query processor is responsible for accessing data and determining the result of a query

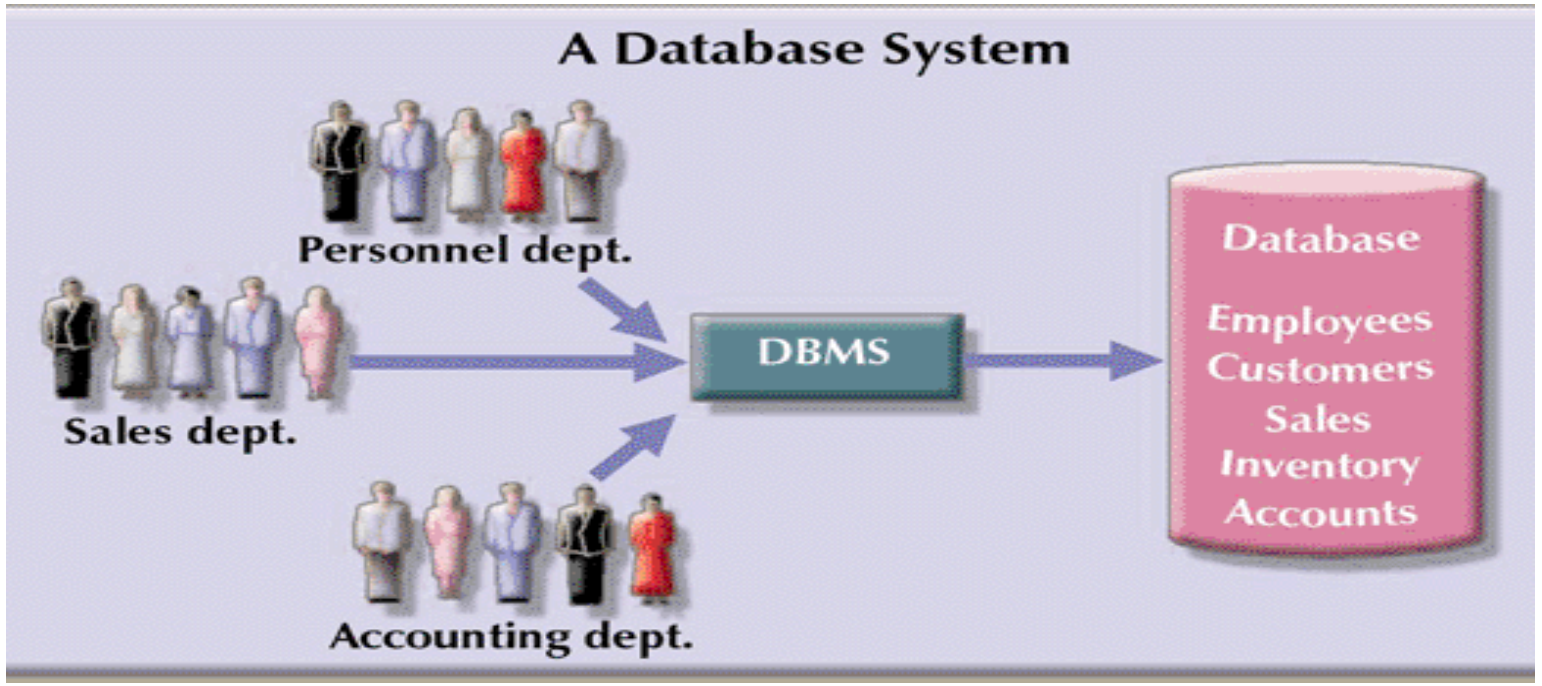


Concurrent Data Management

To show **How to ensure providing services for diverse branches** is the core of DBMS and this PP

Database vs. File System

- Problems inherent in file systems make usi[illegible]tem desirable
- File system: Many separate and unrelated files
- Database : Logically related data stored in a single logical data repository

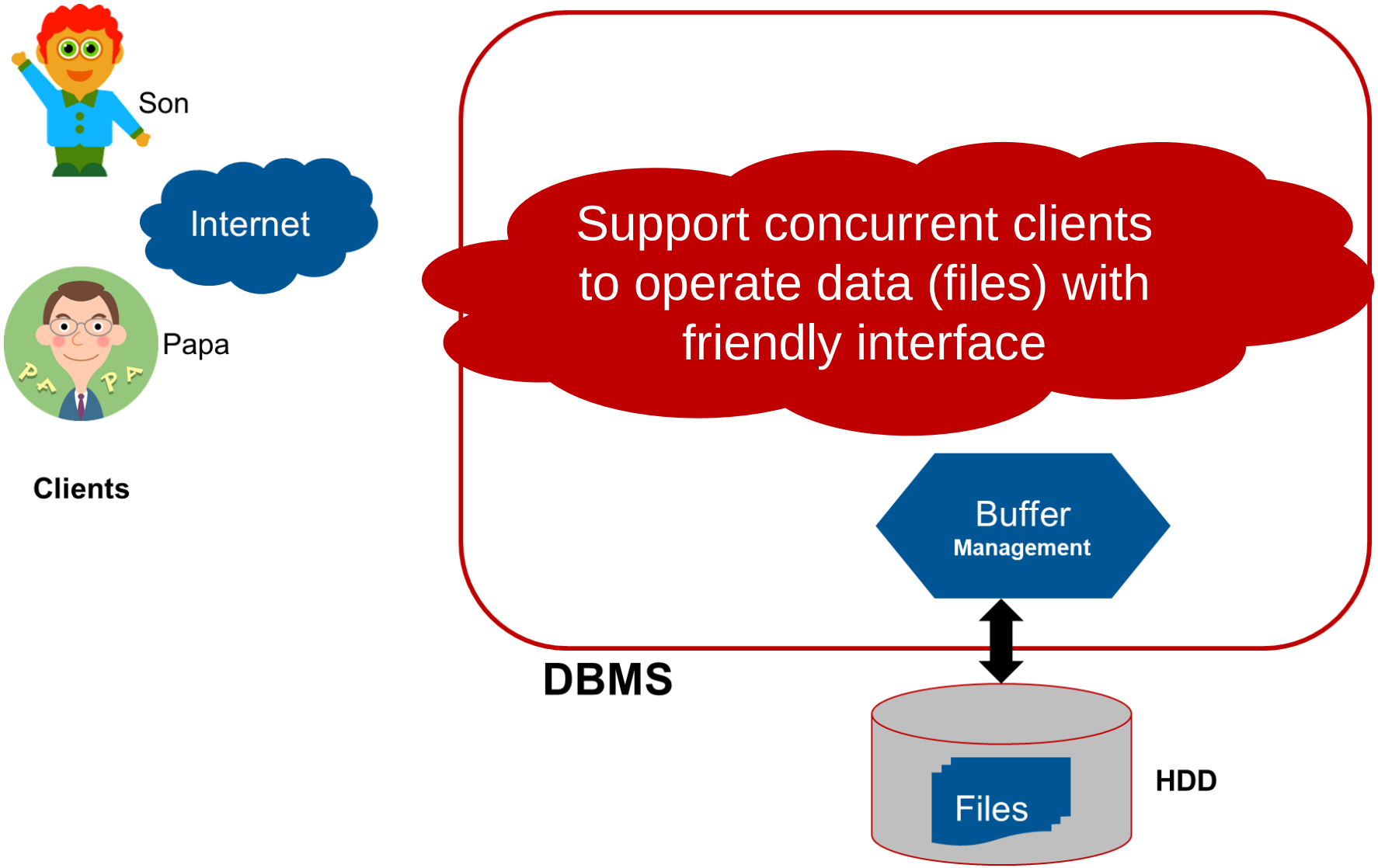


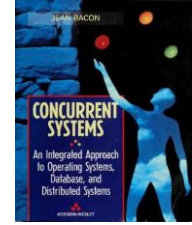
But, storing data is only the basis

- **Top goal of OS?**
  - Support the execution of many programs concurrently with limited resources
- **Top goal of DBMS**
  - Support the **concurrent** **management/access** of **data**
- **Logic to consider**
  - Basic tasks – One user?
    - how to represent data?
  - Advanced tasks – Many users?

2 roles of DBMS

- **I mentioned OS has 2 roles – manage resources and provide services for other programs**
  - Silbersatz's 2 roles are resource manager and friendly interface
- **You can also understand DBMS with following 2 roles**
  1. **Concurrent Data Management** by using files (which are supported by OS)
  2. **Friendly/Flexible data operations for users** to use
     - Define specific data files
     - Insert, Delete, Update, Select some specific records
     - Support some statistics
     - Provide flexible way for users to define their own processing
     - …


Son
Internet
Papa
Clients
Support concurrent clients to operate data (files) with friendly interface
Buffer
Management
DBMS
Files
HDD



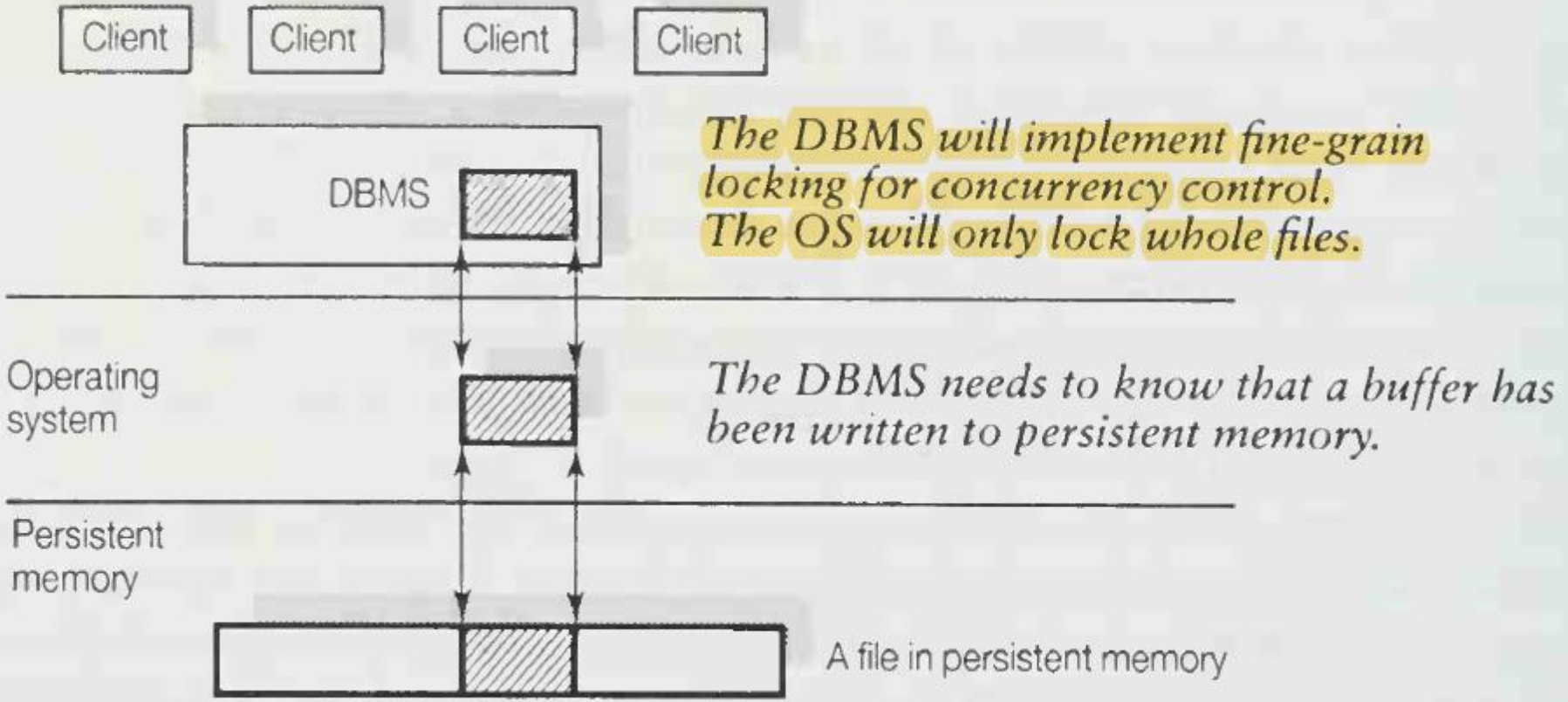


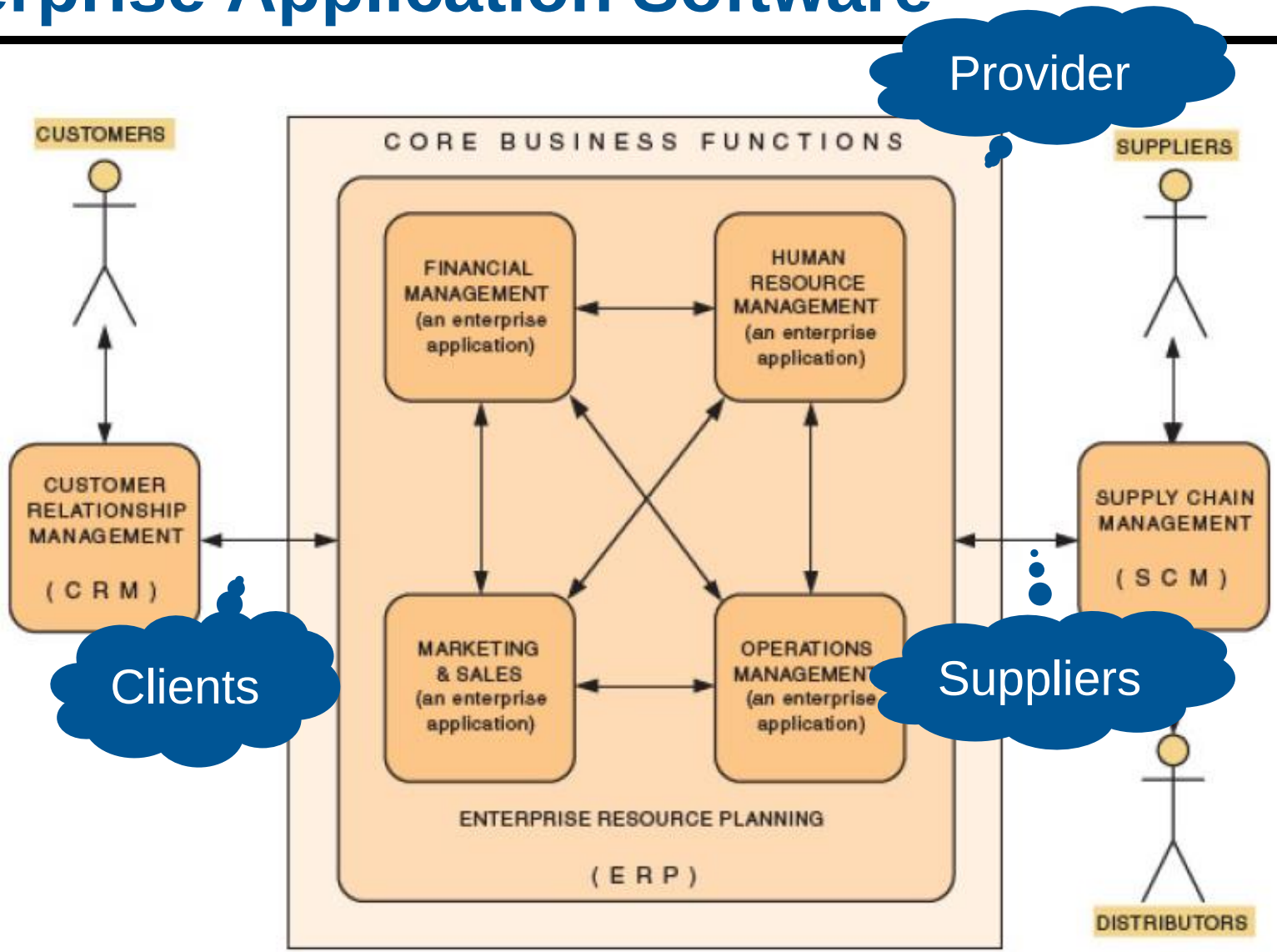
Figure 13.6
A DBMS using an operating system.

Operating systems may support shared or exclusive locks on whole files, but do not support fine granularity locking on small portions of files. This is done by the DBMS.

Modern DBMS is important for IT age

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

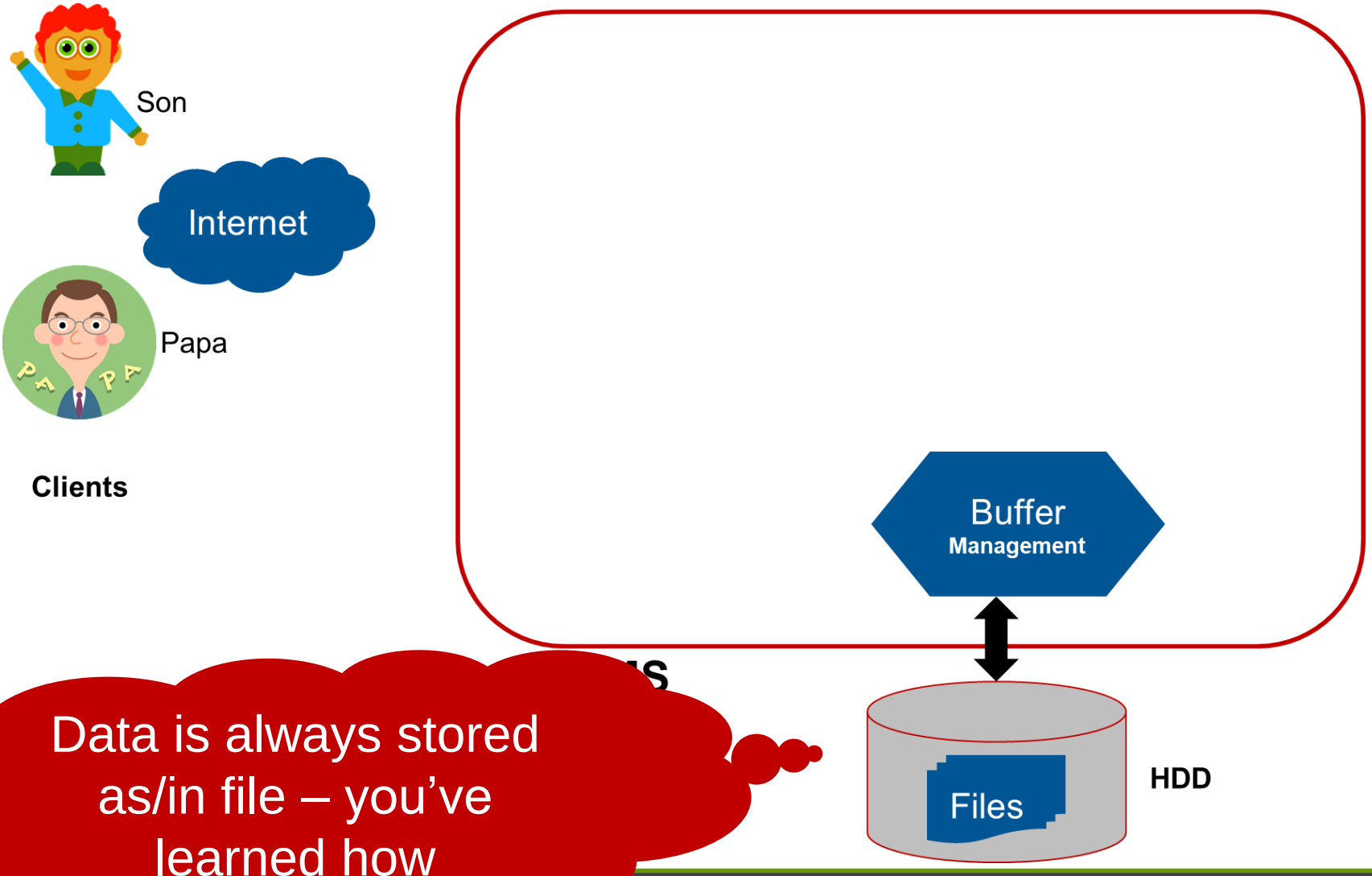
Great applications are constructed – Enterprise Application Software



Chapter 2: Overview of DBMS

- **My understanding about DBMS**
  - History of (R)DBMS
  - Overview of DBMS

Top goal: Concurrent access of the data



File + Index file is the basis to store data!

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

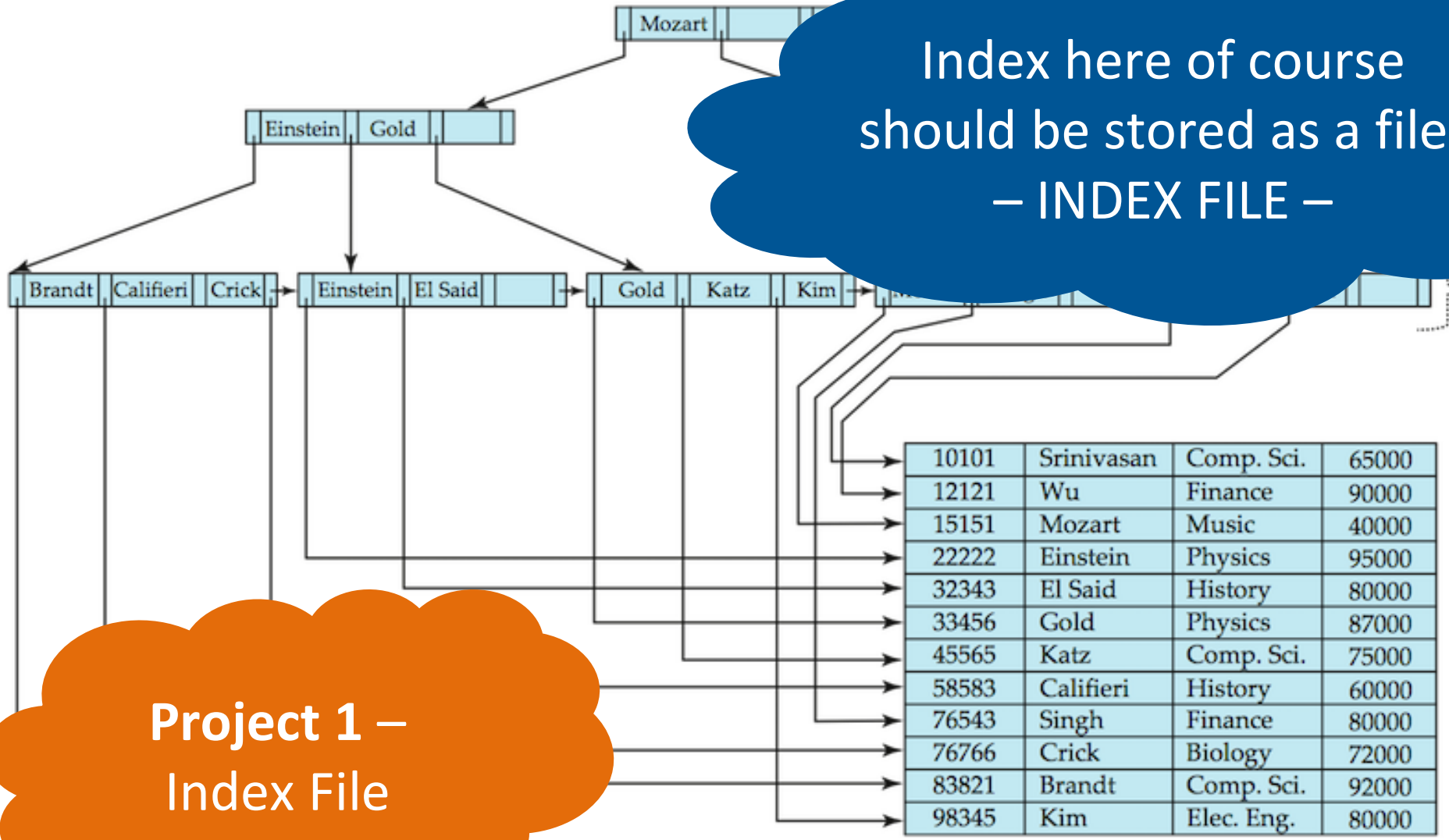
dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Figure 1.2 A sample relational database.

- **We need INDEXING for better search speed**
  - You have learned a lot of index data structures in DSA [Data-structures and Algorithms] course [**Main Memory version**]
    - List – array based list, linked list, 2-direction list …
    - Hashing table
    - Trees – binary tree, AVL (G.M. **A**delson-**V**elsky and E.M. **L**andis: a balanced binary tree) tree, B-tree, B^+-tree, B*-tree, A*-tree …
    - Graphs – linked graph, adjacent matrix representation

- **Among them, balanced trees are efficient for SEARCHING, therefore used as INDEXING for quick record search**



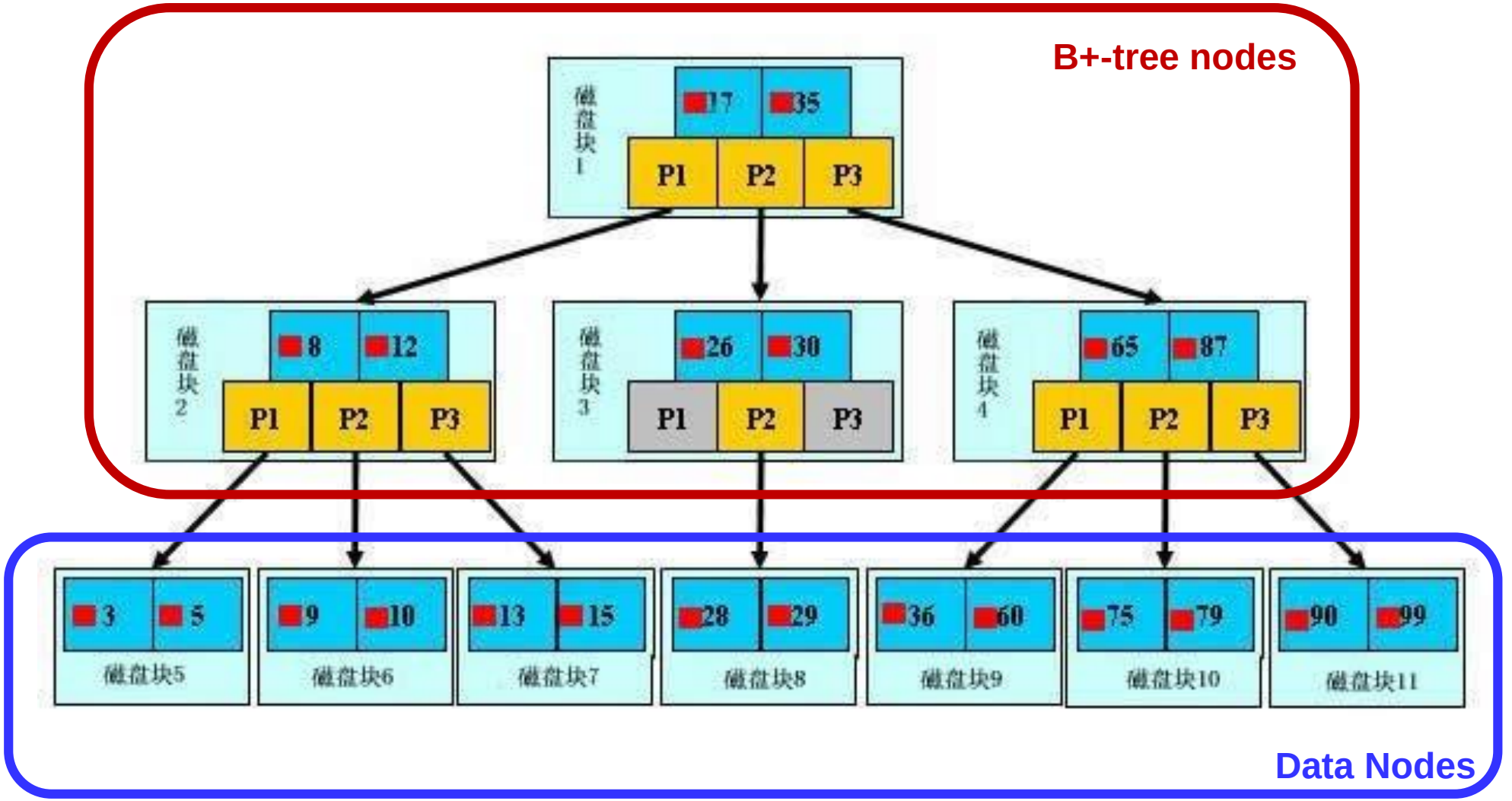
Index here of course should be stored as a file!
– INDEX FILE –

Project 1 – Index File

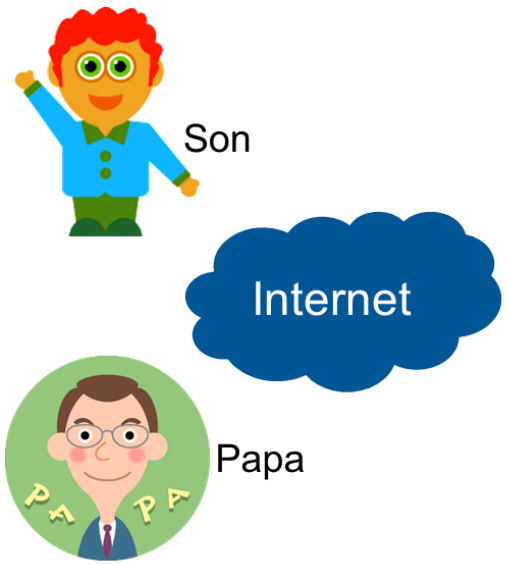
http://flacro.me/b-tree-index/

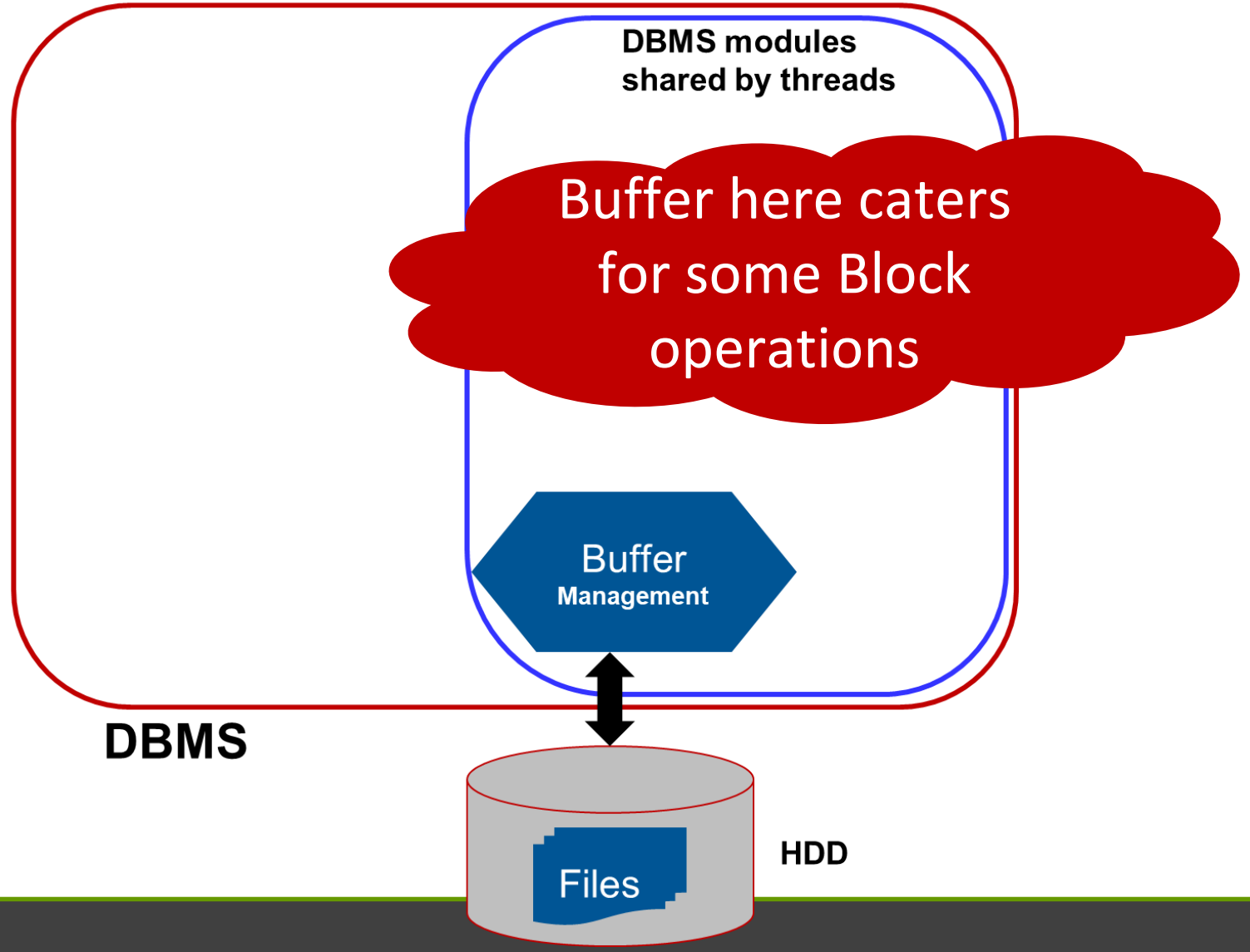
BLOCK?

- **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

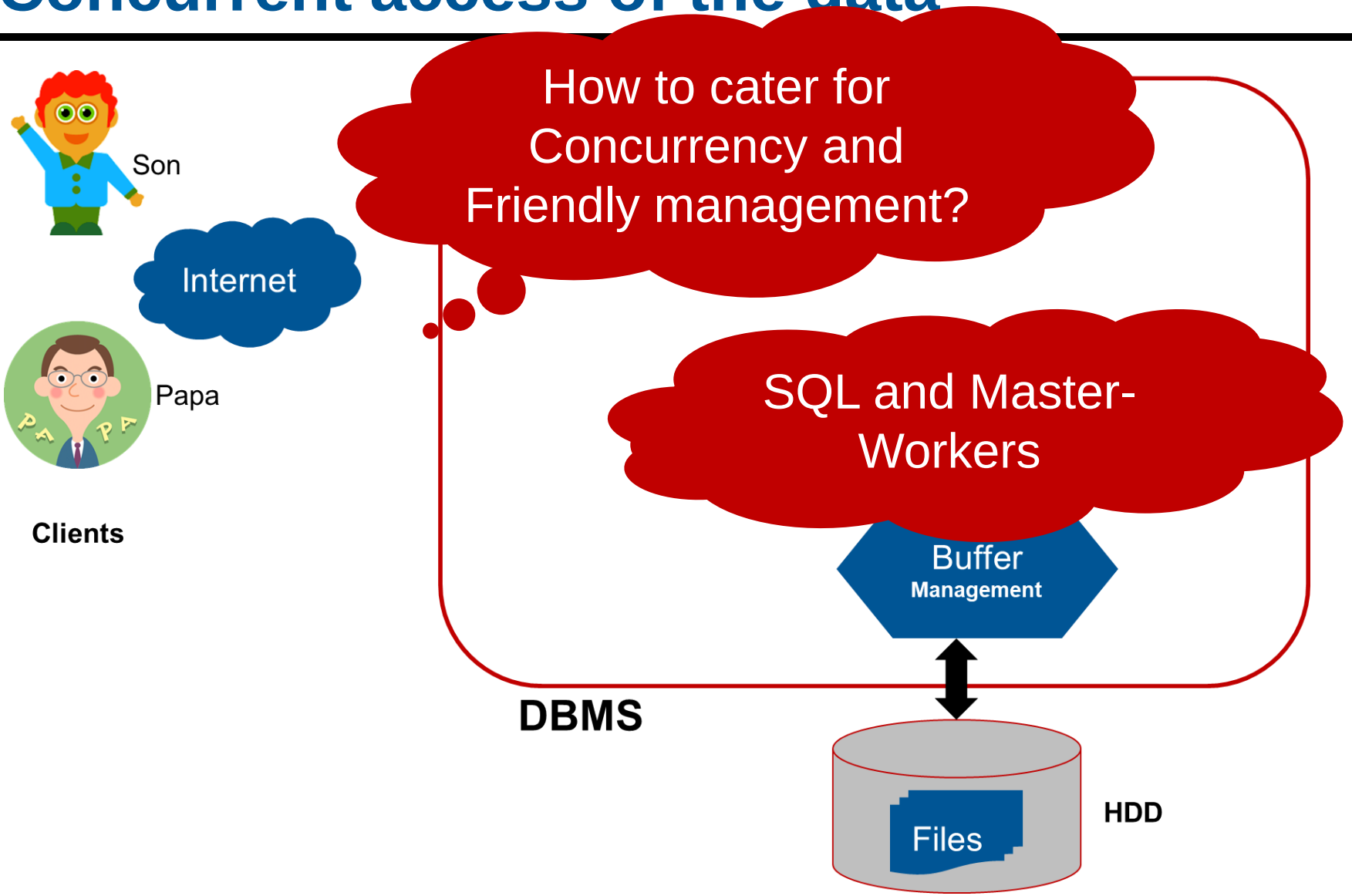


https://www.jianshu.com/p/0371c9569736


Son
Internet
Papa
Clients

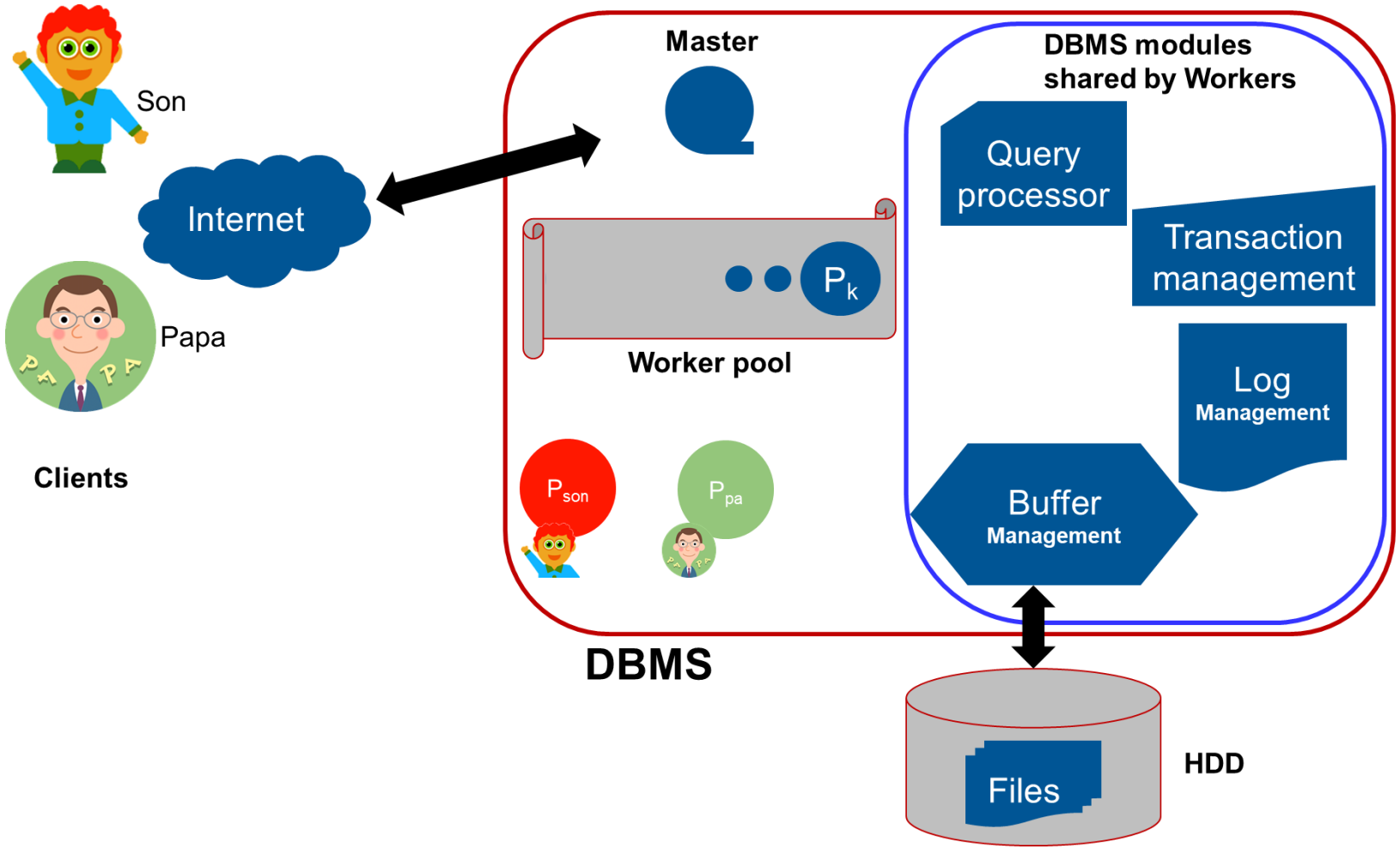

DBMS modules
shared by threads
Buffer here caters for some Block operations
Buffer
Management
DBMS
Files
HDD

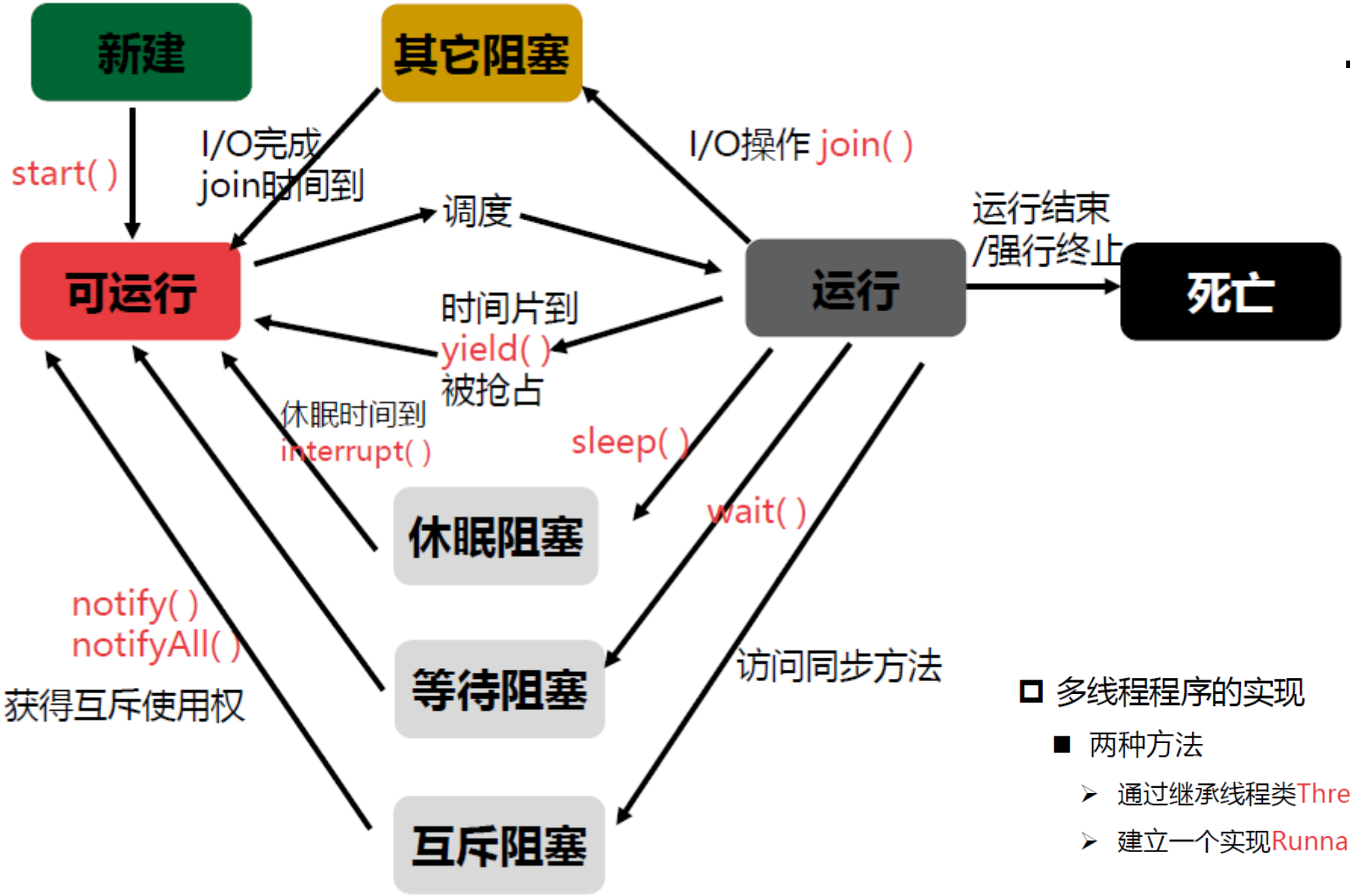
Top goal: Concurrent access of the data



Multi-Threaded is the popular way for Concurrency

- But PostgreSQL uses multi-process skills





- 多线程程序的实现
  - 两种方法
    - 通过继承线程类Thread来创建线程类；
    - 建立一个实现Runnable接口的类来创建线程

专业基础课

计算机体系结构　　算法导论

C语言　　操作系统

专业限选课

并行计算

(Parallel Computing)

百万年薪不是梦

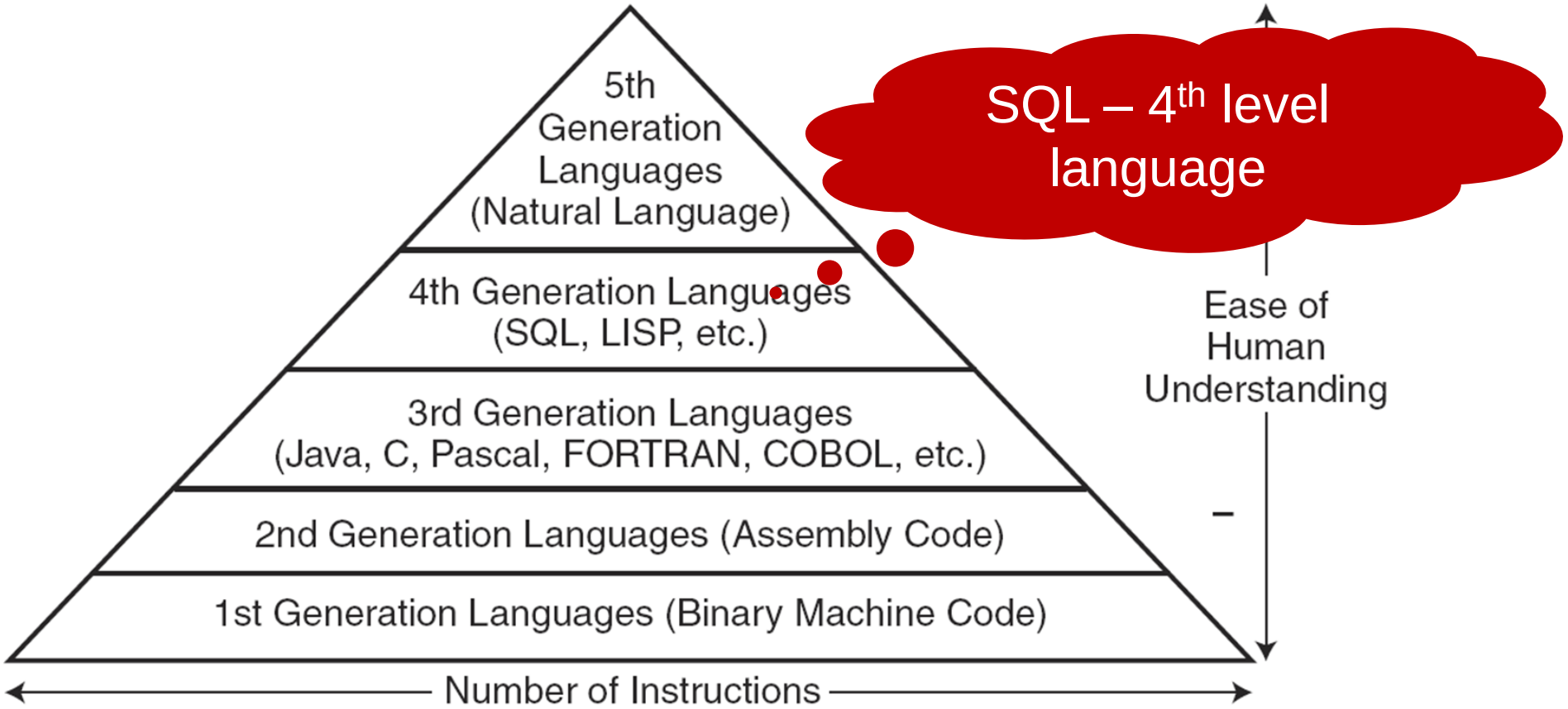
海量数据分析　　并行算法设计

大规模机器学习模型实现　　异构计算

商用信息检索系统构建

SQL is the Friendly Interface

- **Select name, cid from students s, enrolled e where s.sid = e.sid**



By SQL – Structured Query Language

- Originally, the users of DBMSs were programmers.
  - Accessing the stored data required writing a program in a programming language such as COBOL.
  - While these programs were often written to present a friendly interface to a nontechnical user, access to the data itself required the services of a knowledgeable programmer. **Casual access to the data was not practical**.
- SQL was invented at IBM in the 1970s, and became an **ANSI standard** in 1986 and an ISO standard in 1987;
- it is used today in a great many database management systems

- **SQL examples – friendly way to compose PARAMETERS for data management**

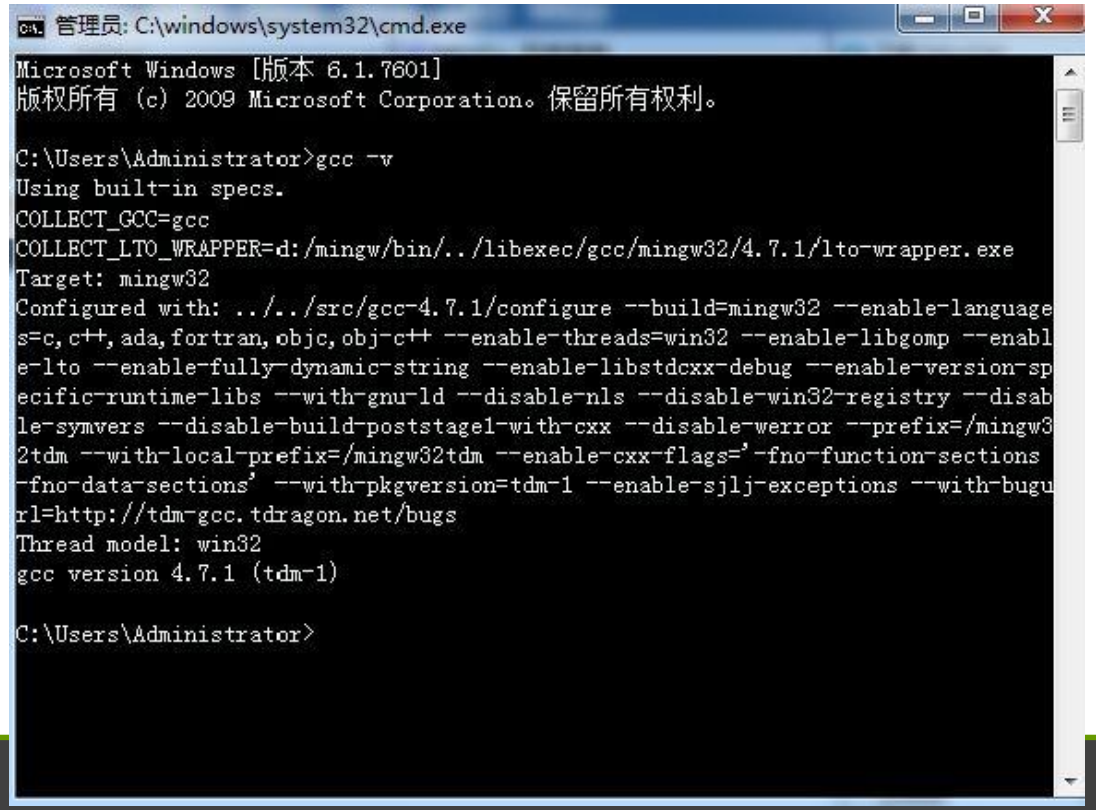
```
create table department
        (dept_name  varchar (20),
         building     varchar (15),
         budget       numeric (12,2),
         primary key (dept_name));
```

```
select name
from instructor
where dept_name = 'Comp. Sci.' and salary > 70000;
```

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name= department.dept_name;
```

Challenge: SQL's translation and execution

- By "Friendly/Flexible" data operations, it means the DBMS should provide users a way to compose operations
- Only file operations and fixed user-defined functions are not enough
  - Do you want "xx –l –x –h –d –o –s "?



- **To ease the common users (**
  from it), a friendly (easier)

You definitely know now that there should be a Translation from SQL to corresponding file operations (even own defined functions like AVG())

Friendly Interface – SQL

CREATE moneyNote (time
UPDATE moneyNote **SET** money–
SELECT AVG(money) **FROM** moneyNote
…

02221432

FILE System supported by OS

File

```
struct{
  time;
  money;}
```

File opera
learn in
- C
- Se
- R
- EOF
- Close

NB: file operation is based on BLOCK/PAGING idea – not like we think as RECORDS.
← OS IO/FILE Chapter

```
select name
from instructor
where dept_name = 'C
```

But how to get the file operation sequence by UNDERSTANDING SQL? – **TRANSLATOR – the kernel of modern DBMS**

- **How to derive the file operation sequence fro SQL?**
  - You've learned SET theory – **UNION**, **INTERSECTION**, …you can guess the file operations to get the result

			60000
45565	Katz		75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

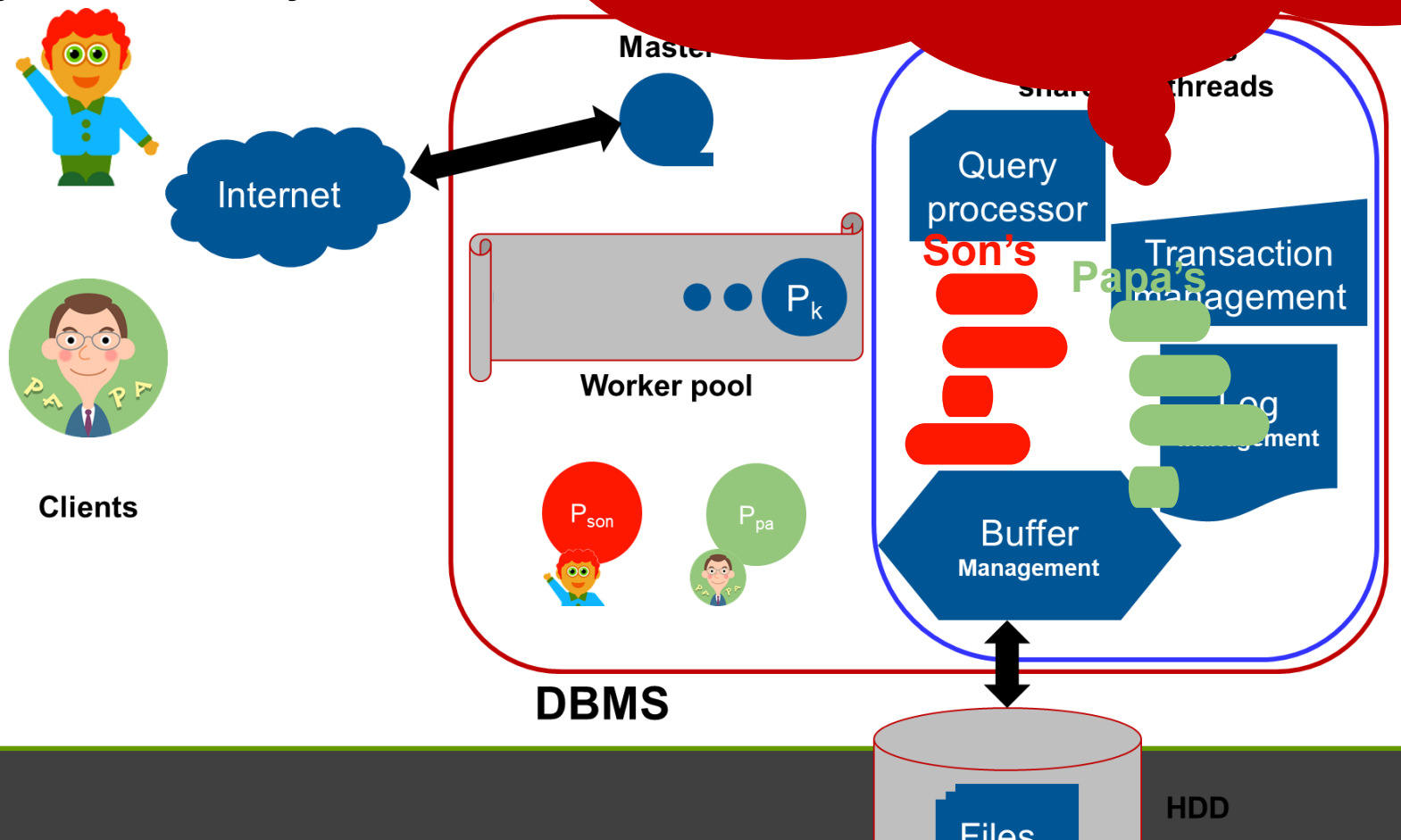
(a) The *instructor* table

  - **Res ←{};**
  - **While SCANning *INSTRUCTOR*** file
    - If the current RECORD's dept_name is 'Comp. Sci' and salary>70000
      - ✓ Res ← RECORD.name;
  - Return res

SQLs → File Operation s

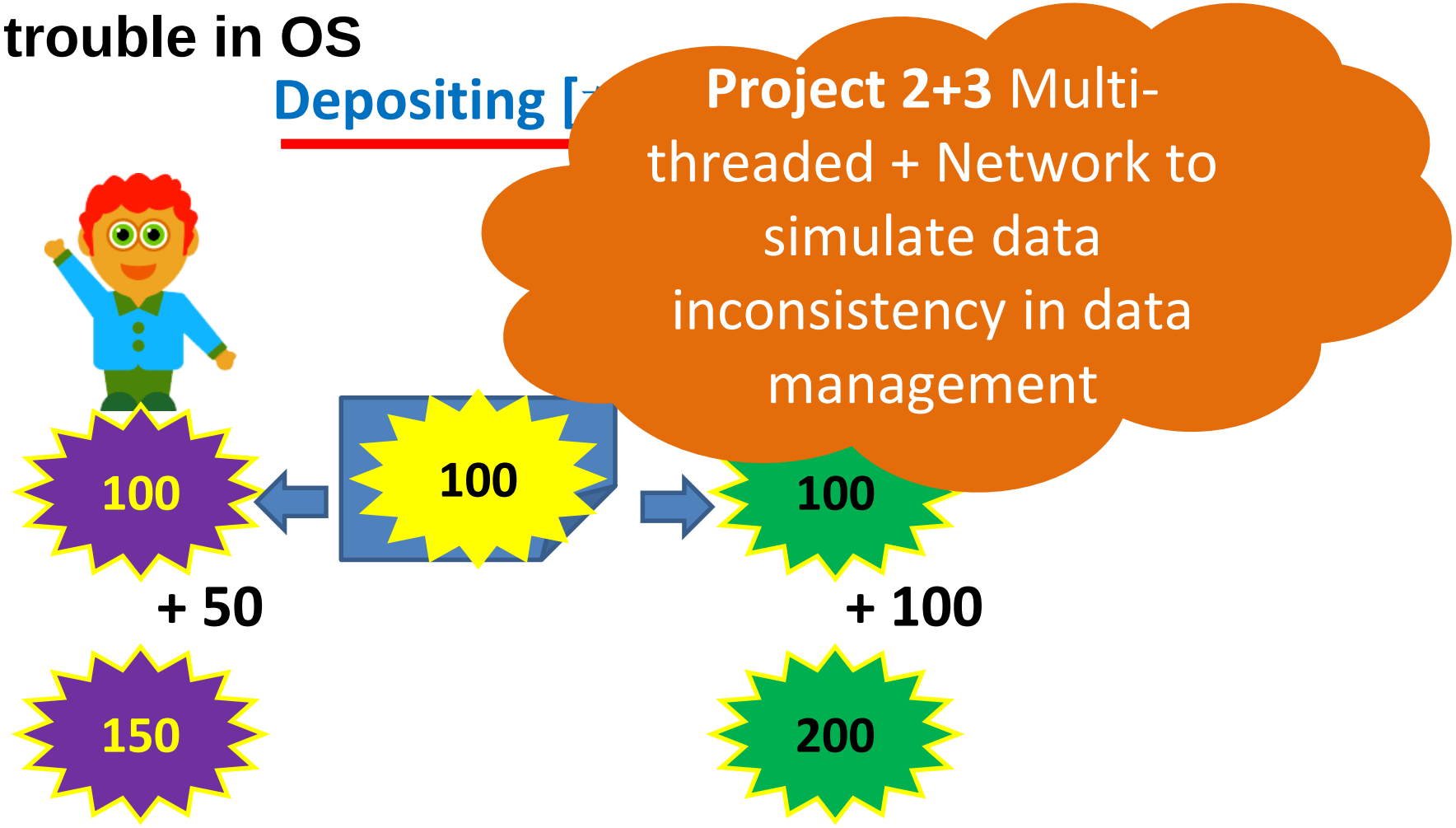
- SQLs from concurre
  into File operation se
  - They are usually interl

The File Operation Sequence for one client is usually treated as a **Transaction [事务]** by default (except when using COMMIT explicitly)



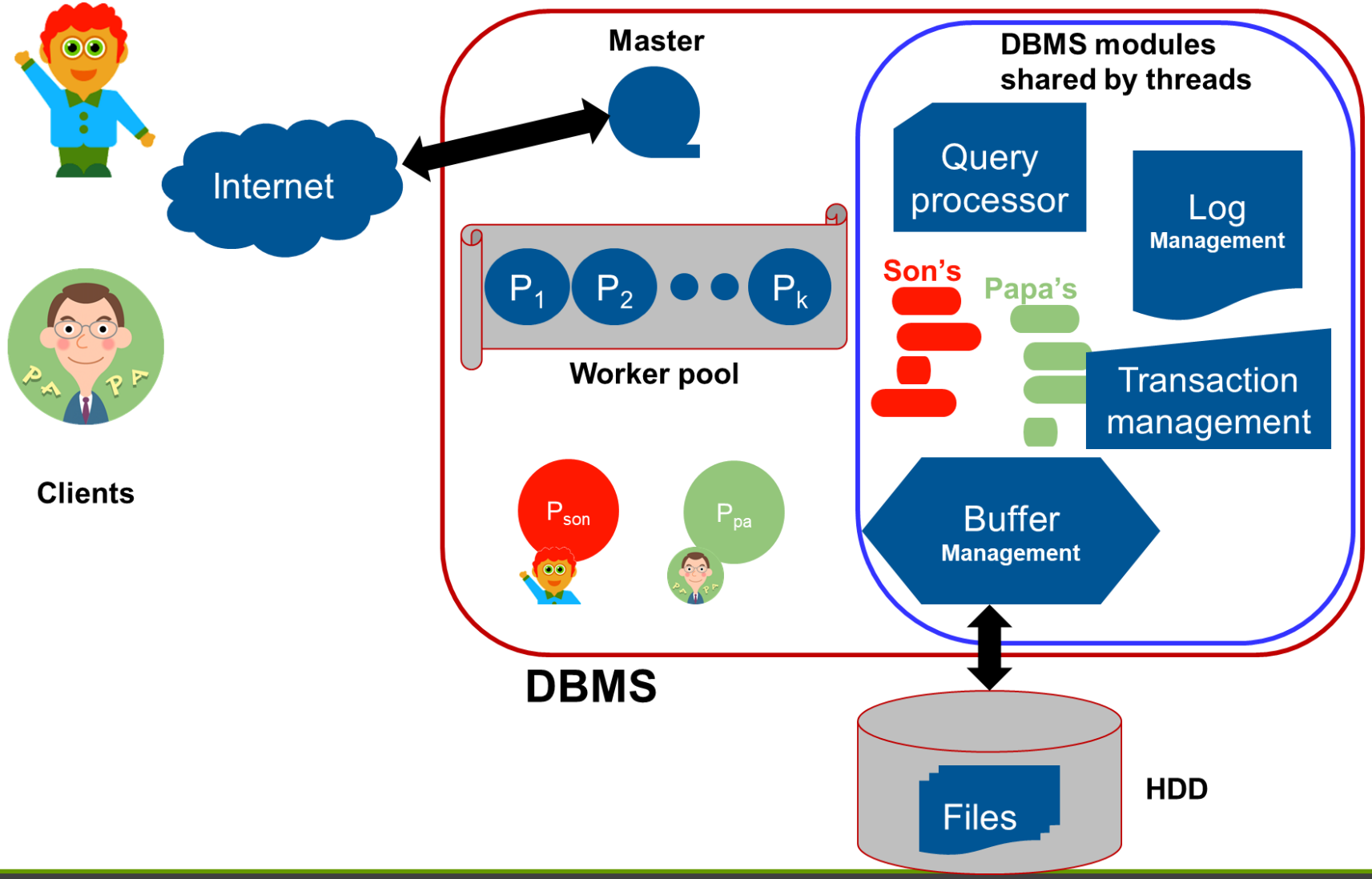
Challenge: Data Inconsistency

- By “concurrent” data management, it is similar with the risk of data inconsistency trouble in OS



- **But DBMS goes further – straight-foreword LOCK is not adequate**
  - **DEADLOCK**
- **DBMS uses more complicated ideas**
  - Two phase locking
  - Timestamp
  - MVCC
    - Multiversion concurrency control
      - https://en.wikipedia.org/wiki/Multiversion_concurrency_control
  - Snapshot

Log and Transaction Management



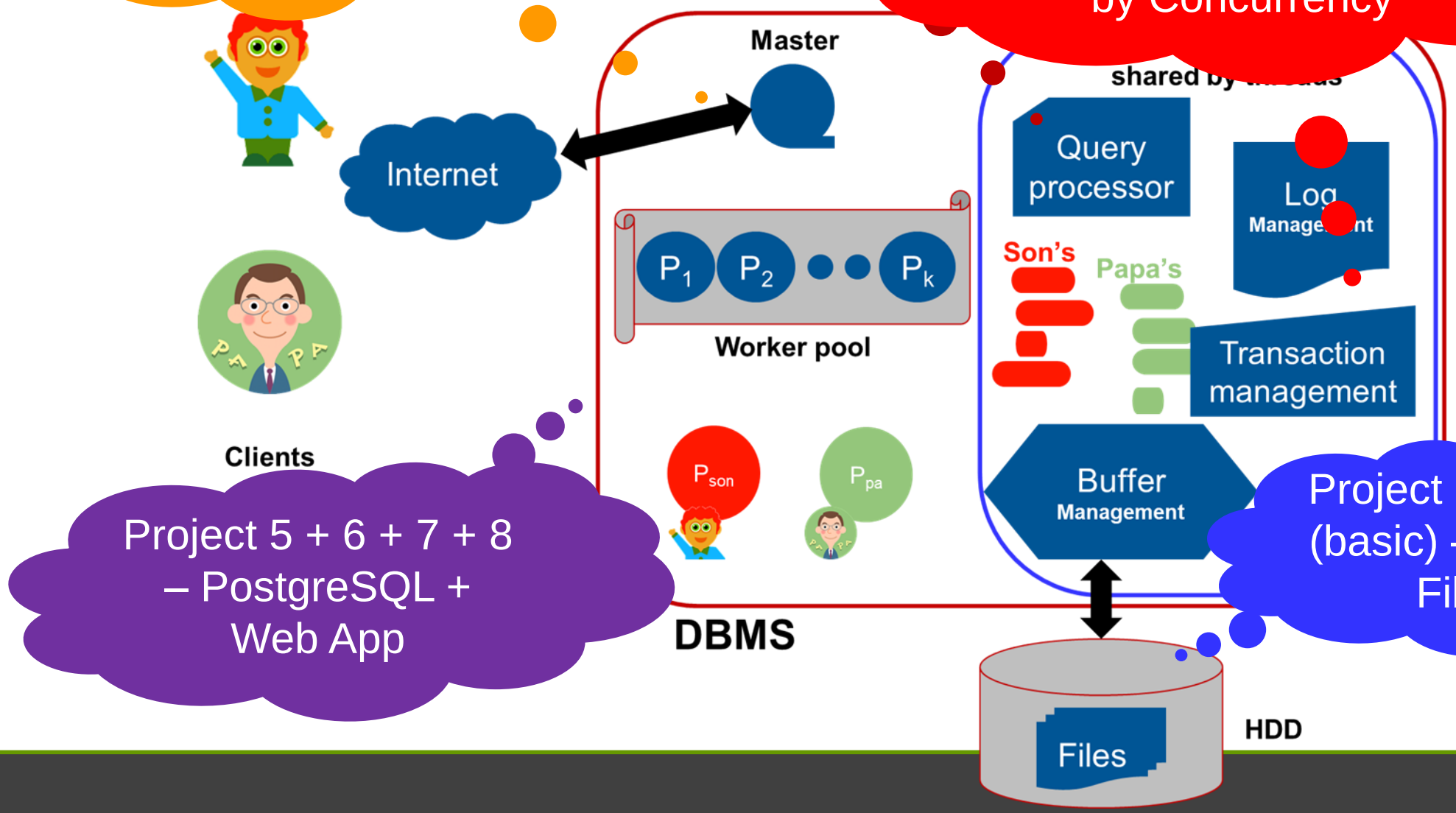
- **Relationship between Projects and DBMS Modules**

Project 2 – Worker Pool for Concurrency

Project 3 + 7 – SQL translator & Executi

Project 4 + 8 Log and Transaction for Data Inconsistency risk caused by Concurrency

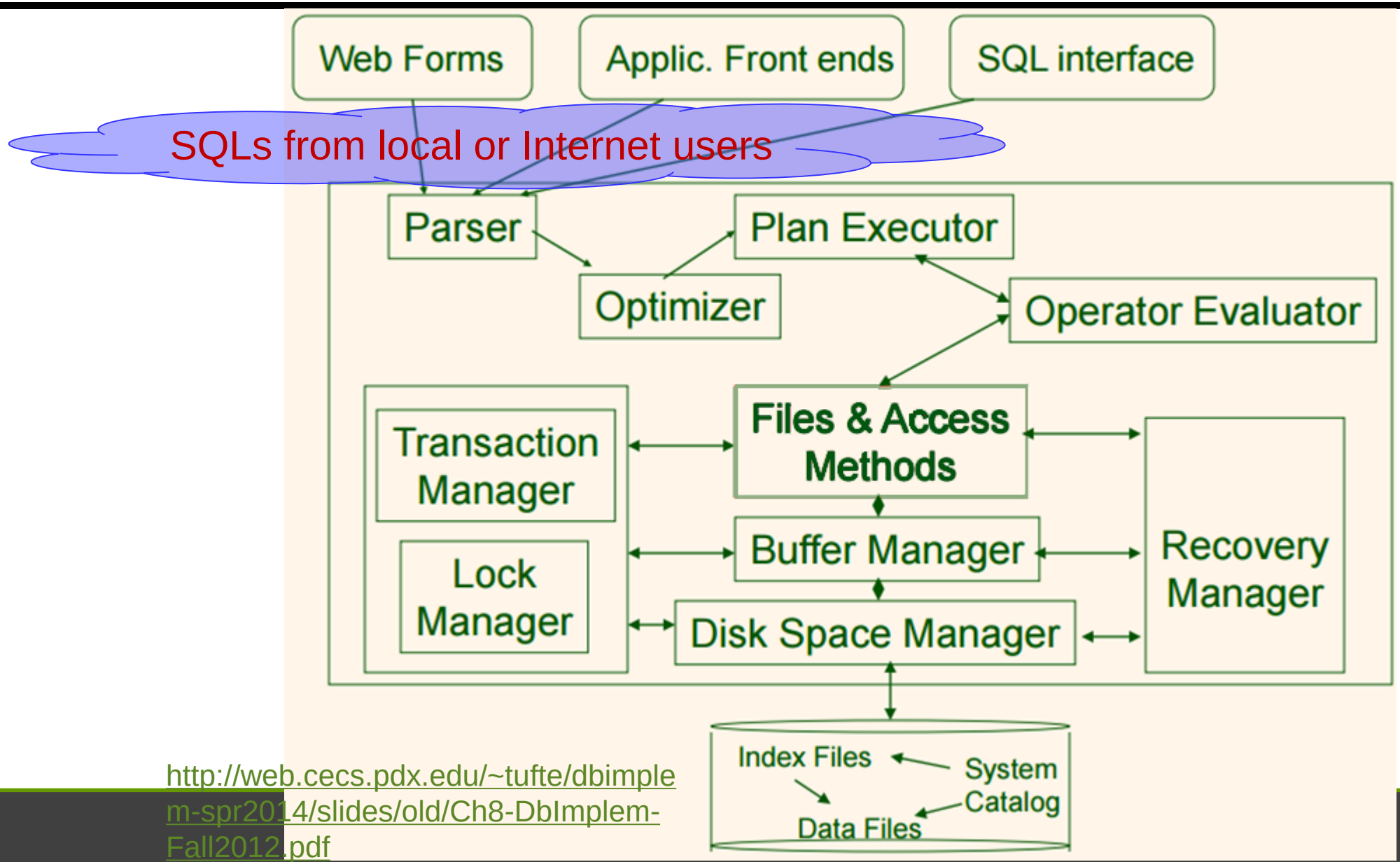
Sket modules of mode



Project 5 + 6 + 7 + 8 – PostgreSQL + Web App

Project 1 – DM (basic) – Index File

More detailed DBMS's modules



http://web.cecs.pdx.edu/~tufte/dbimplem-spr2014/slides/old/Ch8-DbImplem-Fall2012.pdf

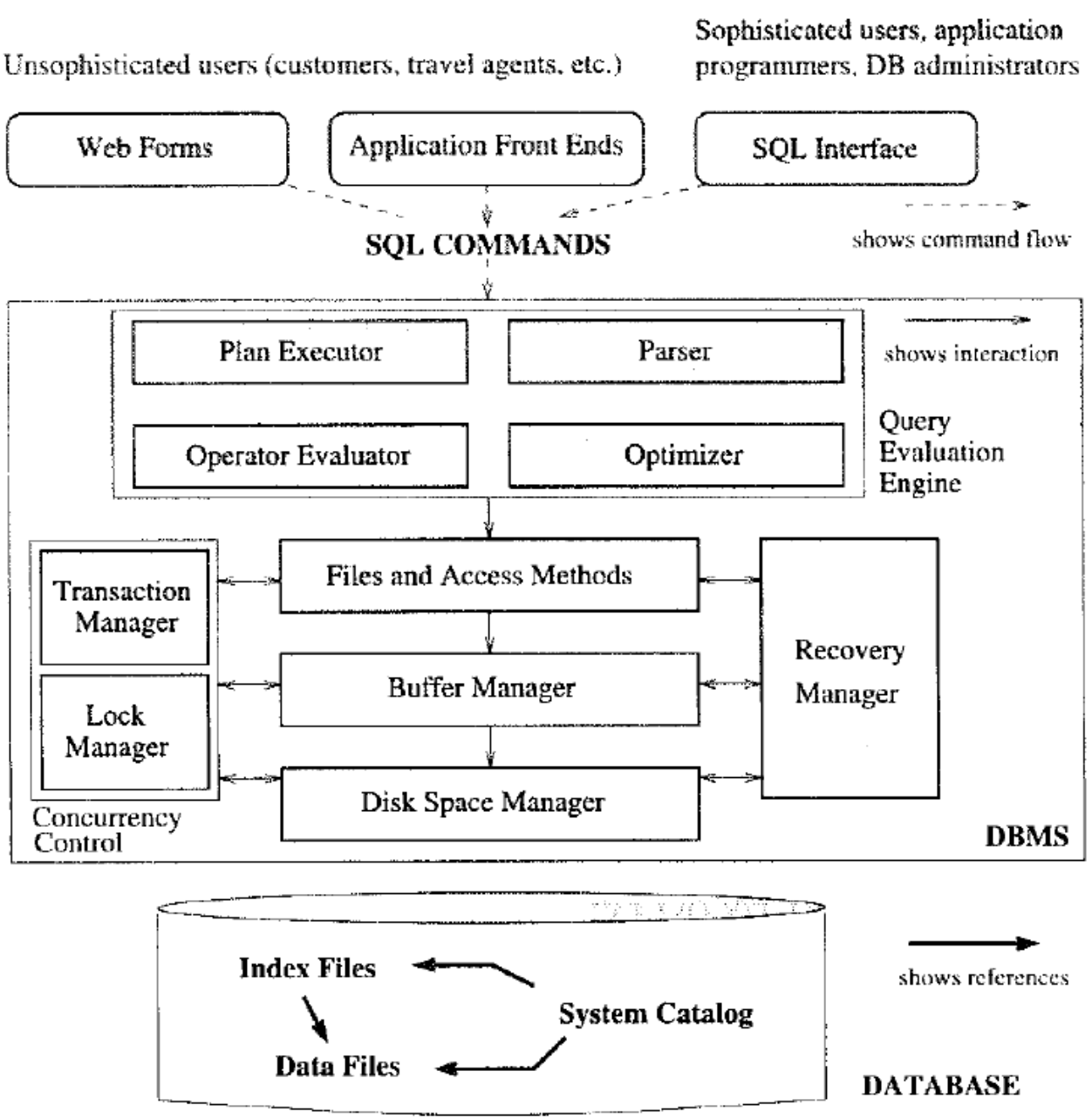


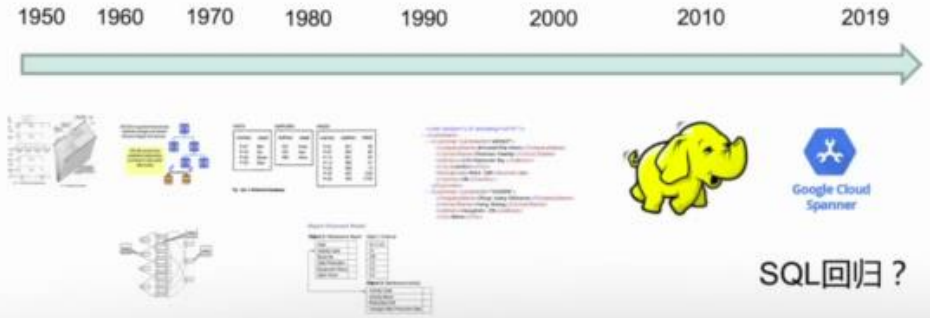
Figure 1.3 Architecture of a DBMS

Many other issues for D & I of modern DBMS

- **Data Safety**
  - **RAID** (Redundant Array of Independent Di
  - **Recovery**
    - Like sudden Power-Off?
- **Embedded into Programming Langu**
  - ODBC, **JDBC** (Java Database Connectivity)
- **Distributed DBMS, Big Data**
  - Definitely more complicated!
    - SQL execution → Data Inconsistency
  - Also new opportunities

Unlike many so-called Data Analytics Companies, D & I of core programs ~

SQL （分布式数据库）回归？



MR被其创造者Google 在2011年就放弃了

Cloud Spanner 的文章！

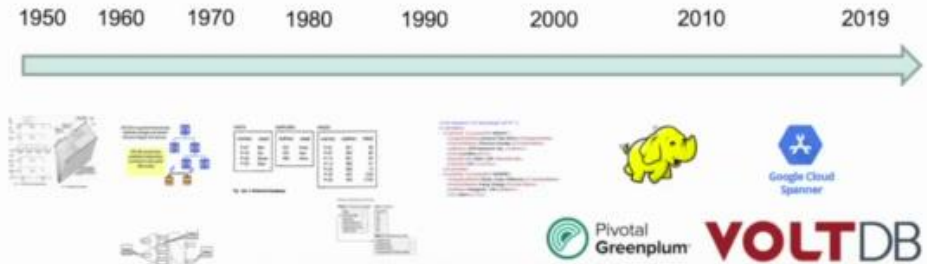
2008：MapReduce 是技术大倒退

作为一种数据处理模式，MapReduce 是一种大倒退。

- 模式（Schema）很有价值
- 实现糟糕，不支持 Access Method
- 高级访问语言很有价值

Google 内部2011年放弃MR，2015年公开放弃

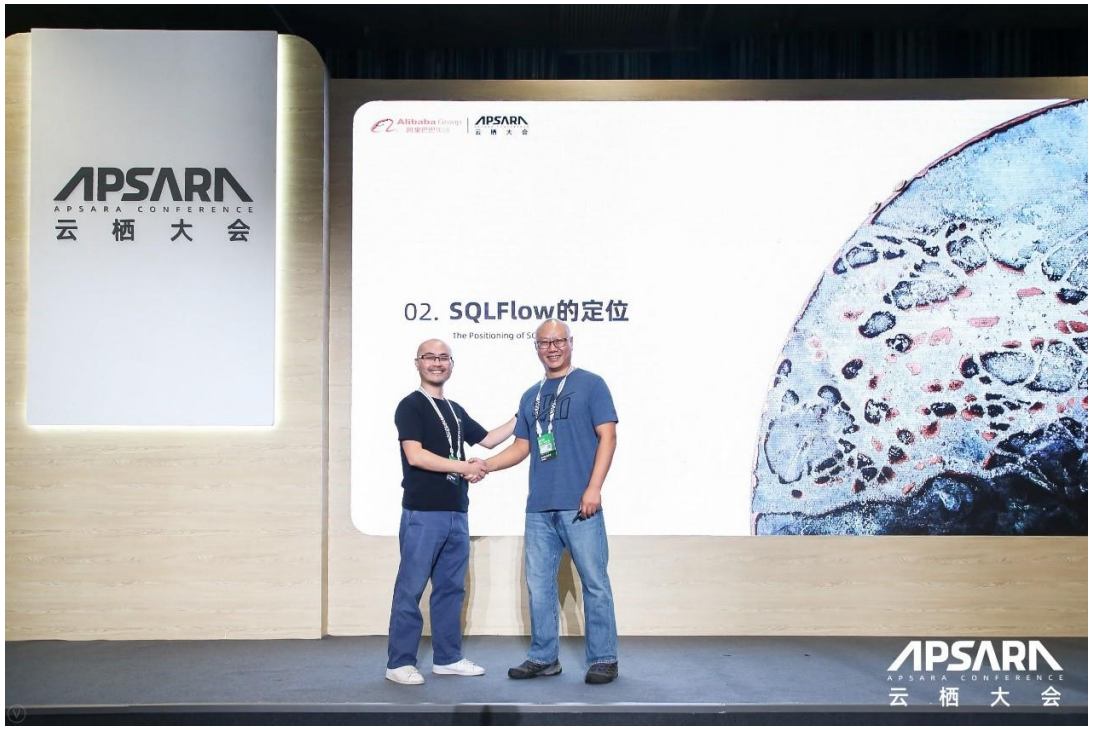
SQL （分布式数据库）从未离开



2019

王益 在蚂蚁金服 推出 SQLFlow + ElasticDL

https://zhuanlan.zhihu.com/p/87682668
https://developer.aliyun.com/article/719553 [原文]



项目官网：https://sqlflow.org
GitHub地址：https://github.com/sql-machine-learning/sqlflow
您也可以使用docker，运行文章中的汽车价格预测模型：docker run -p 8888:8888sqlflow/sqlflow:didi

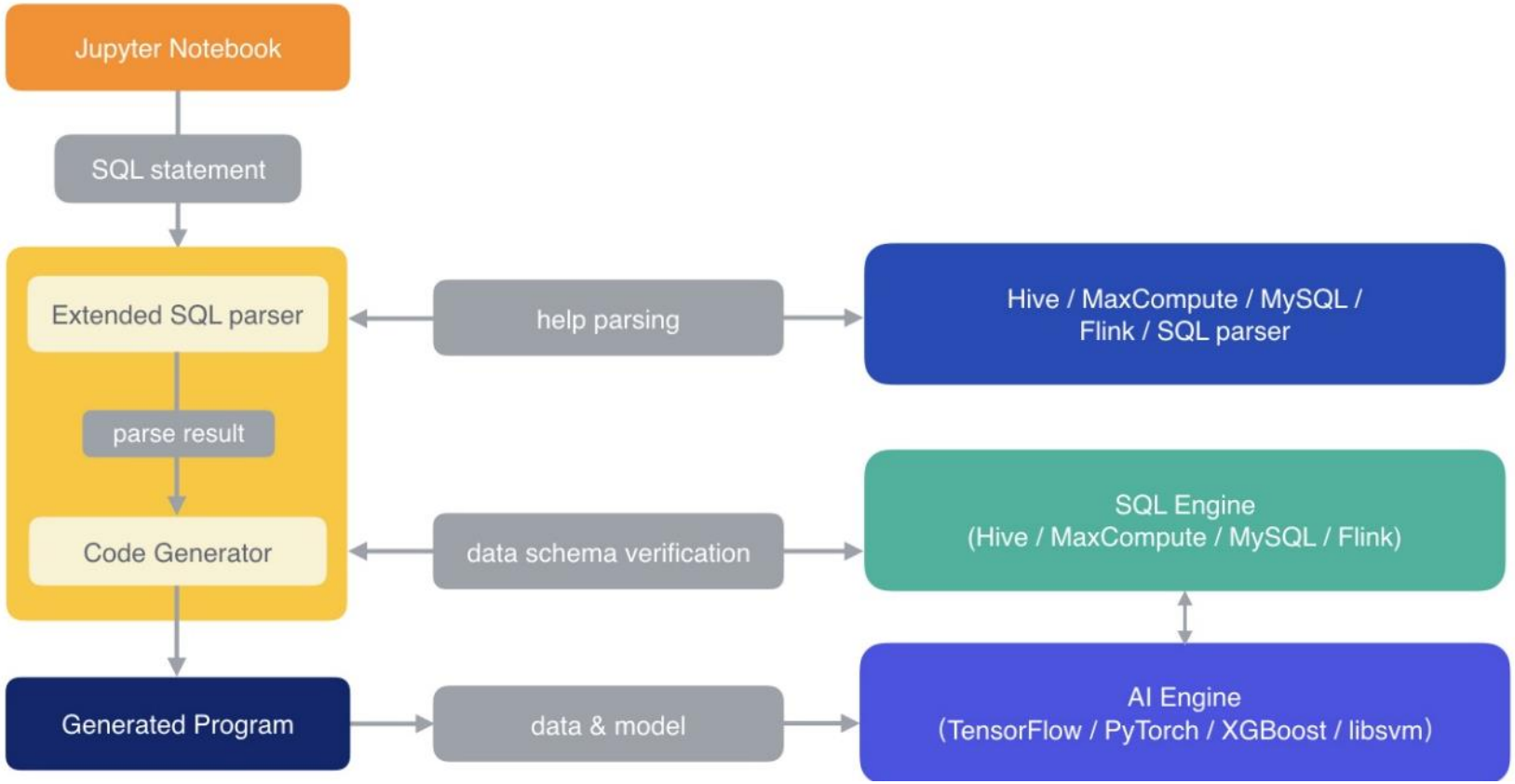
SQLFlow

https://developer.aliyun.com/article/719553 [原文]

SQLFlow Demo

for SIGMOD 2020

SQLFlow的技术架构



SQLFlow 和滴滴数据的整合逻辑