

# 第一章 - 引言

---



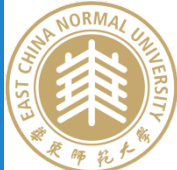
## ➤ 数据库系统应用

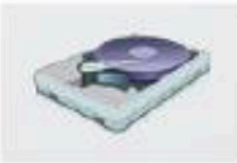
- ✓ 航空
- ✓ 银行
- ✓ 销售
- ✓ 电子商务
- ✓ 生产制造业
- ✓ 人力资源

➤ 生活中的各方面都会涉及数据库，成为所有企业**不可或缺**的组成部分

➤ 数据库可能非常大（1 GB, 1 TB or 1 PB?）

# 数据量度量



TERABYTE	10 的 12 次方	一块 1TB 硬盘		200,000 照片或 mp3 歌曲
PETABYTE	10 的 15 次方	两个数据中心机柜		16 个 Blackblaze pod 存储单元
EXABYTE	10 的 18 次方	2,000 个机柜		占据一个街区的 4 层数据中心
ZETTABYTE	10 的 21 次方	1000 个数据中心		纽约曼哈顿的 1/5 区域
YOTTABYTE	10 的 24 次方	一百万个数据中心		特拉华州和罗德岛州

## ➤ DBS包含哪些信息？

- ✓ 有联系的**数据集**
- ✓ 处理数据的**程序**

## ➤ DBS成为一种既方便又高效存取数据信息的途径

➤ 定义(p1): 数据库系统由一个**互相关联的数据集**和**一组用以访问这些数据的程序**组成。

## ➤ 20世纪60年代，大学数据库实例

- ✓ 保存**信息**在操作系统文件中，包括教师、学生、院系、课程等信息
- ✓ 对文件进行操作的**应用程序**，包括增加教师、学生、课程，计算学生平均成绩等程序

## ➤ 早期，数据库应用直接构建在文件处理系统（file processing system）之上

- ✓ **一个明显的弊端**：应用程序直接处理文件，**紧耦合**

- 1. 数据的冗余和不一致
- 2. 数据访问困难
  - ✓ 对于每个新任务需要编写新的应用程序处理
- 3. 数据孤立
  - ✓ 多个文件具有不同的格式 (比如: txt, doc, bin格式等等)
- 4. 完整性问题
  - ✓ 完整性约束“固化”(buried)在程序代码中
  - ✓ 很难通过修改程序来体现新的约束

## ➤ 5. 原子性问题

- ✓ 故障会导致部分更新，使数据库处于不一致状态

## ➤ 6. 多用户并发访问

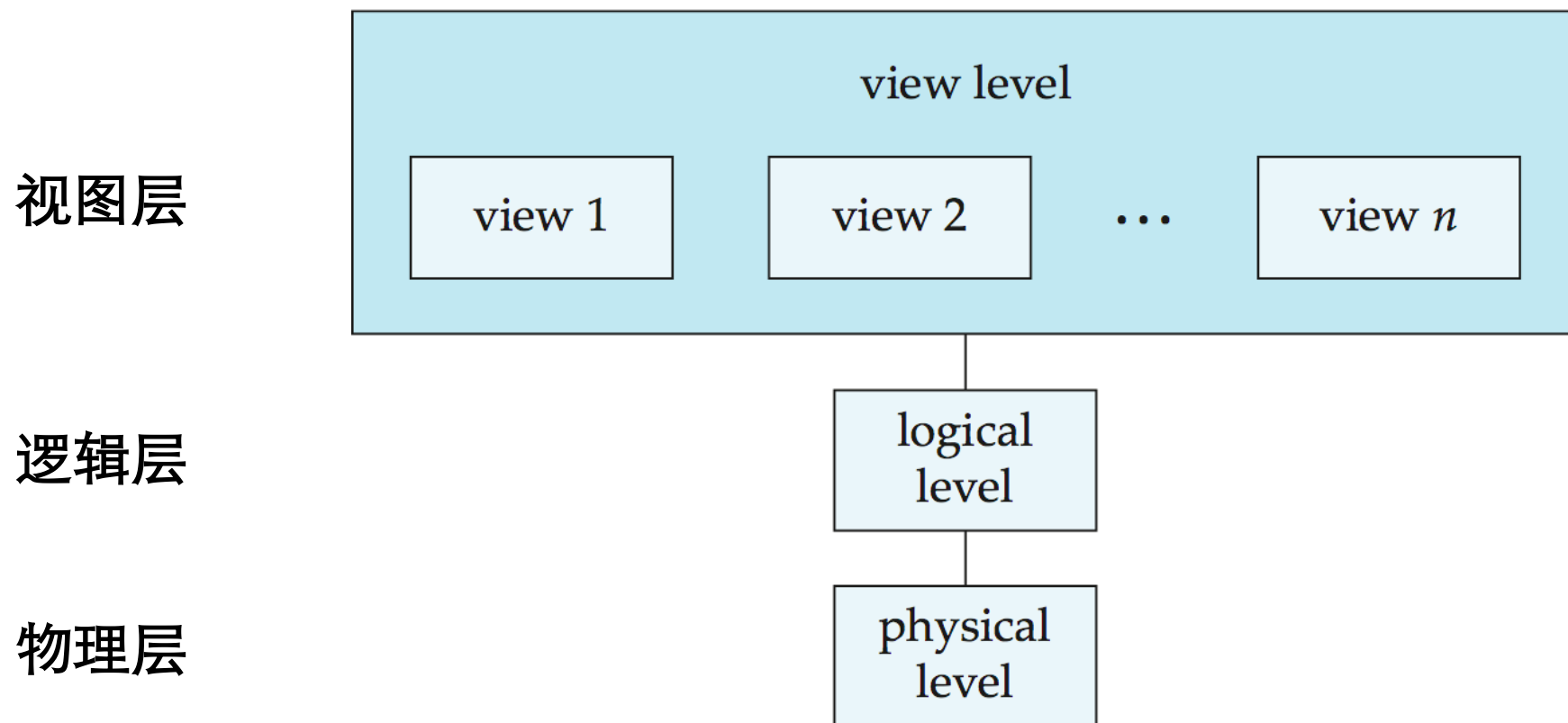
- ✓ 提高性能
- ✓ 可能带来不一致

## ➤ 7. 安全性问题

- ✓ 并非数据库系统的所有用户都可以访问所有数据

**数据库系统可以解决上述所有问题**

## ➤ 数据抽象的三个层次





- **物理层 (Physical level)** : 描述数据实际上是怎样存储的
- **逻辑层 (Logical level)** : 描述存储在数据库中的数据, 以及数据之间的关系, 降低耦合 (数据库管理员)
  - ✓ 为上层应用屏蔽了复杂的底层存储细节, 即物理数据独立性

```
type instructor = record
```

```
  ID : char(5);
```

```
  name : char(20);
```

```
  dept_name : char(20);
```

```
  salary : numeric(8, 2);
```

```
end;
```

- **视图层 (View level)** : 视图可以隐藏信息, 使用户仅访问数据库的一部分, 提高安全性 (数据库普通用户)

- 与程序设计语言中的**类型**和**变量**（**类**和**对象**）类似
- **模式（Schema）** – 数据库的总体设计
  - ✓ 例：数据库包含客户和账户的信息以及它们的联系
  - ✓ 与程序语言中的**类型声明**相似
  - ✓ **物理模式**: 在物理层描述数据库的设计
  - ✓ **逻辑模式**: 在逻辑层描述数据库的设计
  - ✓ **子模式**: 描述数据库的不同视图
  - ✓ **物理数据独立性**: 应用程序不依赖于物理模式
- **实例（Instance）** – 特定时刻存储在数据库中的信息的集合
  - ✓ 与**变量的值**相似

- 多种概念工具的集合，用于描述数据库：
  - ✓ 数据
  - ✓ 数据联系
  - ✓ 数据语义
  - ✓ 一致性约束
- 关系模型：表（行列）用于表示数据和数据间的联系
- 实体-联系（E-R）模型：实体、联系；数据库设计
- 基于对象的数据模型：E-R模型增加了封装、对象等
- 半结构化数据模型：用XML来表示半结构化数据
- 历史的数据模型，模型和底层实现联系耦合度强
  - ✓ 层次数据模型
  - ✓ 网状数据模型

## ► 表格形式

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

# 关系数据库例子



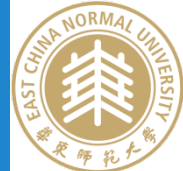
<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.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

(a) The *instructor* table

<i>dept_name</i>	<i>building</i>	<i>budget</i>
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

# 数据定义语言(DDL)



## ➤ 定义数据库模式

例: **create table** *instructor* (  
                    *ID*                    **char**(5),  
                    *name*                **varchar**(20),  
                    *dept\_name* **varchar**(20),  
                    *salary*             **numeric**(8,2))

## ➤ DDL生成表模版保存在数据字典中

## ➤ 数据字典包含关系模型元数据 (关于数据的数据)

- ✓ 表信息、视图、索引

- ✓ 完整性约束

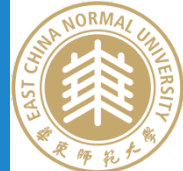
  - 主键约束

  - 参照完整性 (外键)

- ✓ 用户授权

- ✓ etc.

# 数据操纵语言 (DML)



- 用户访问和操纵按照某种适当的数据模型组织起来的数据
  - ✓ DML是一种查询 (query) 语言
- 两类数据操纵语言
  - ✓ 过程化DML – what and how
  - ✓ 声明式DML – what
- SQL是最广泛使用的声明式查询语言
  - ✓ 输入为1~N个表，仅返回一个表

## ➤ SQL: 广泛使用的非过程化（声明式）语言

- ✓ 例:检索编号为 22222的教师姓名

```
select  name
from    instructor
where   instructor.ID = '22222'
```

- ✓ 例:检索 Physics系的教师编号和所在办公楼

```
select instructor.ID, department.building
from    instructor, department
where   instructor.dept_name = department.dept_name and
         department.dept_name = 'Physics'
```

## ➤ 应用程序通过以下访问数据库

- ✓ 扩展宿主语言，允许使用嵌入式SQL（需要DML预编译器）
- ✓ 通过应用程序接口（e.g., ODBC/JDBC），将SQL查询发送到数据库



## 设计数据库结构的过程:

### ➤ 概念设计

- ✓ 需求决定 – 数据库中需要记录哪些数据和它们之间的联系？  
(第七章) 主要运用E-R模型来分析

### ➤ 逻辑设计 – 确定数据库模式，找到一个好的模式

- ✓ 如何将概念设计阶段的数据和联系映射到数据库系统的实现数据模型中？i.e., 设计表结构、主外键、范式（3NF、BCNF）等（第八章）

### ➤ 物理设计 – 确定数据库的物理布局（第十章）

## ► Discussion: 如下逻辑设计有问题吗?

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

## ➤ 设计数据库的规范化理论 (第八章), 避免:

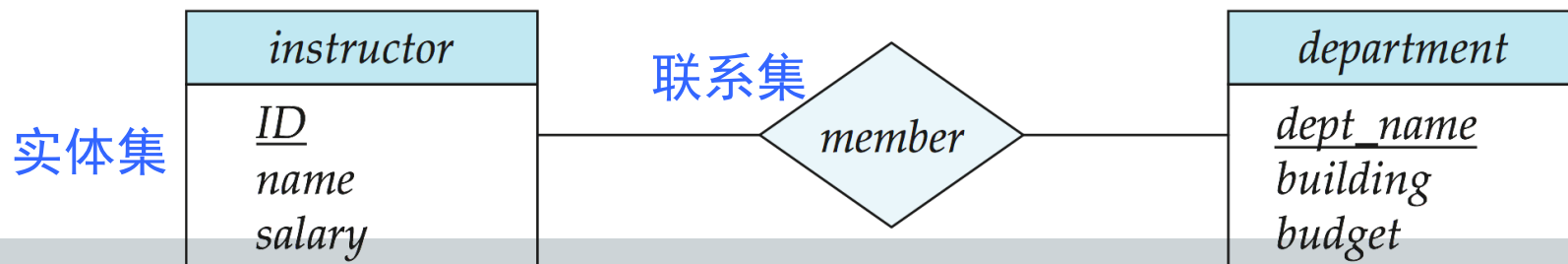
- ✓ 信息重复
- ✓ 缺乏表达某些信息的能力

## ➤ 实体-联系模型 (第七章)

### ✓ 将数据库需求建模为实体和联系的集合

- 实体: 区别于其他对象的一件“事情” 或一个“物体”
  - 由属性集合描述
- 联系: 几个实体之间的关联关系

### ✓ 以实体-联系图 (E-R图), 基于UML的符号表示:

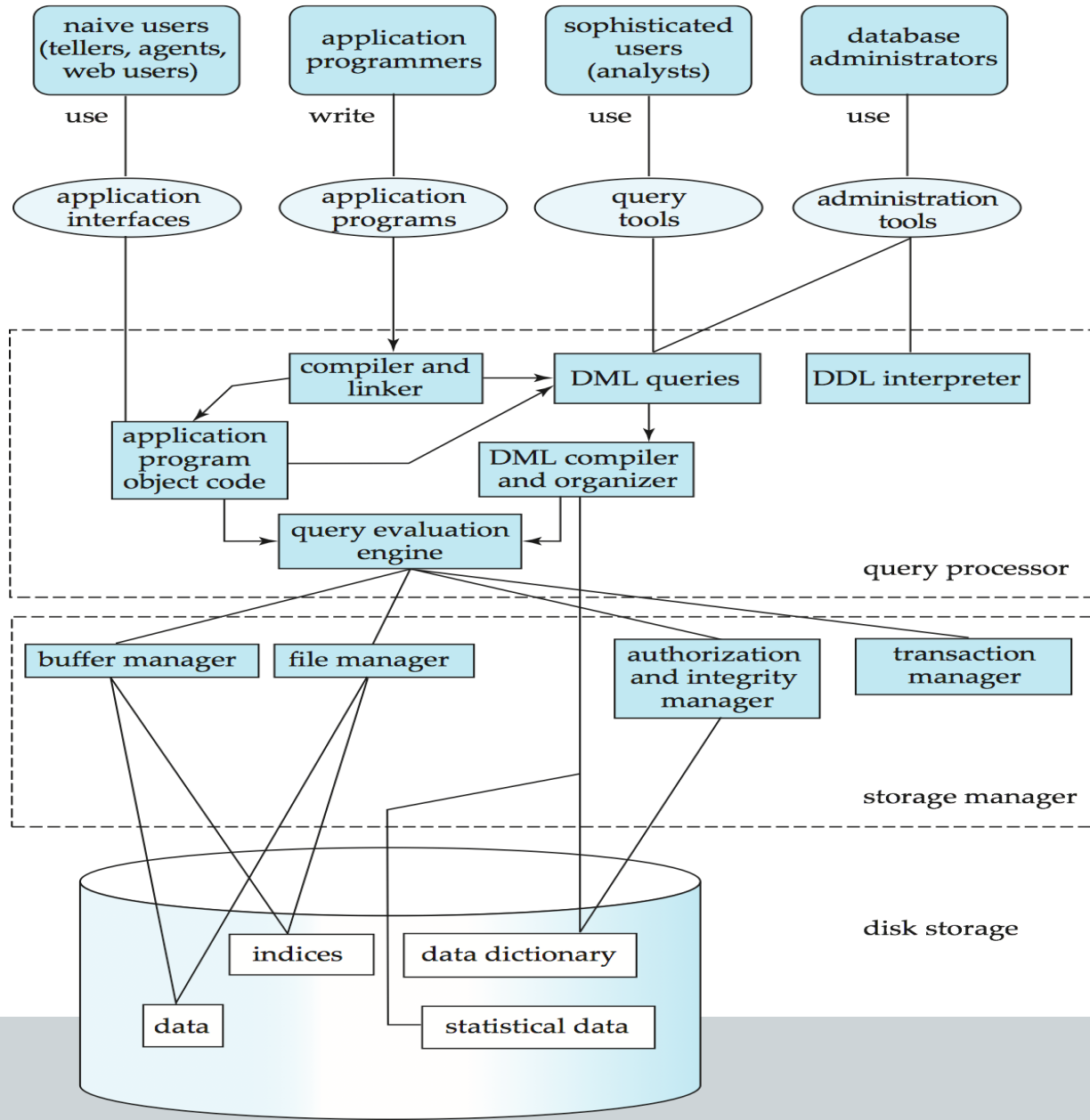


# 数据库系统内部



查询处理器

存储管理器



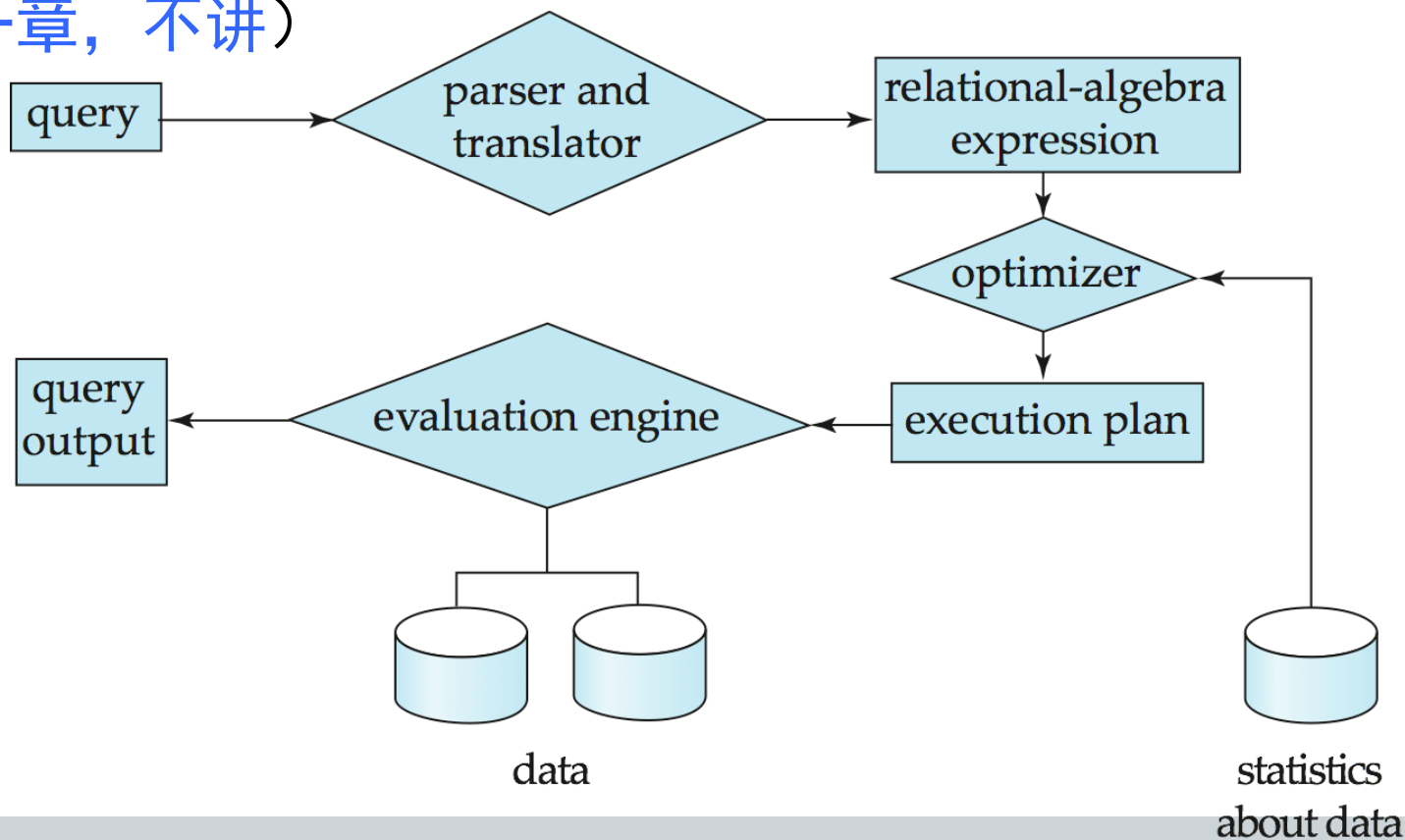
- **存储管理器**是数据库系统中负责在数据库中**存储的底层数据与应用程序**、以及向系统提交的查询之间**提供接口**的部件（**第十章**）
- 存储管理器**负责**:
  - ✓ 与文件管理器交互
  - ✓ 数据库中数据的**存储、检索和更新**
- 存储管理器**包括**:
  - ✓ 权限及完整性管理器 (authorization and integrity manager)
  - ✓ 事务管理器 (transaction manager) (**第十二章**)
  - ✓ 文件管理器 (file manager)
  - ✓ 缓冲区管理器 (buffer manager): 缓冲区替换策略

1. SQL (DDL解释和DML编译)

2. 查询优化

3. 查询执行引擎

(第十一章, 不讲)



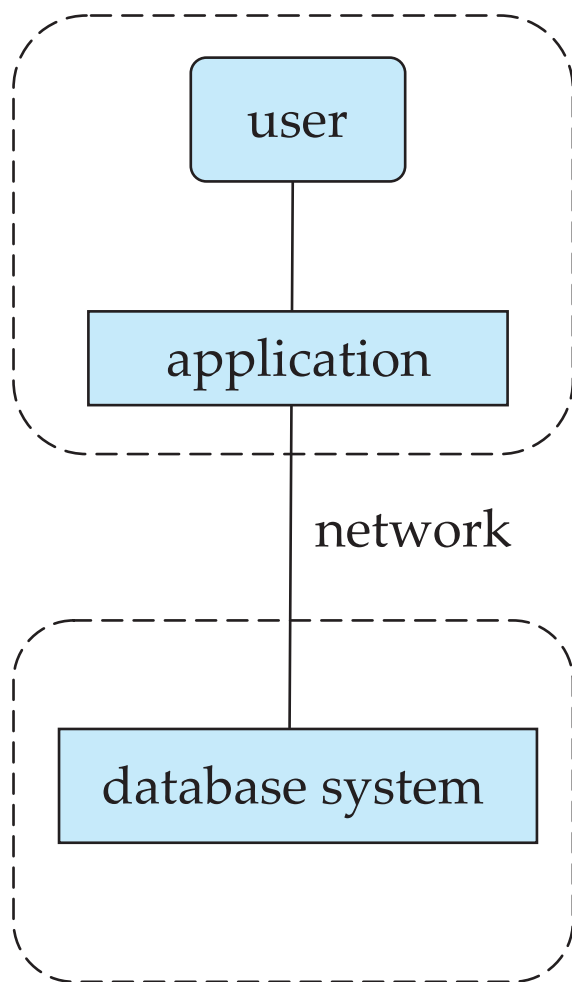
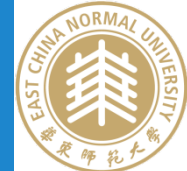
- 定义：事务是数据库应用中完成单一逻辑功能的操作集合（p13）
  - ✓ 原子性(Atomicity)：事务要么发生，要么不发生
  - ✓ 事务操作完成后，需保证数据一致性(Consistency)
  - ✓ 隔离性(Isolation)：事务运行期间需独占数据（锁机制）
  - ✓ 即使有系统故障，数据也要保证持久性(Durability)
  
- 事务管理器保证数据库在发生系统故障时也能保证数据库处于正确一致的状态（第十二章）
  - ✓ 并发控制管理器控制并发事务间的交互，保证数据库的一致性
  - ✓ 恢复管理器负责事务回滚、故障恢复

数据库系统的体系结构很大程度上取决于数据库系统所运行的计算机系统:

- 集中式: 数据库运行于本地
- 客户-服务器 (C/S) 系统:
  - ✓ 两层体系结构: 前端 (业务逻辑)、数据后台, 小型应用
  - ✓ 三层体系结构: 前端、业务逻辑、数据后台, 大型应用
- 并行数据库系统: 高端大型计算机 (神威太湖之光)
  - ✓ 使用多个处理器和磁盘提升处理速度和I/O速度
  - ✓ 1024处理器/机柜, 40,960 CPU处理器 (申威26010) , 260 cores/处理器
- 分布式数据库系统: 通过广域网络连接
  - ✓ 数据共享、局部自治、提高数据可用性

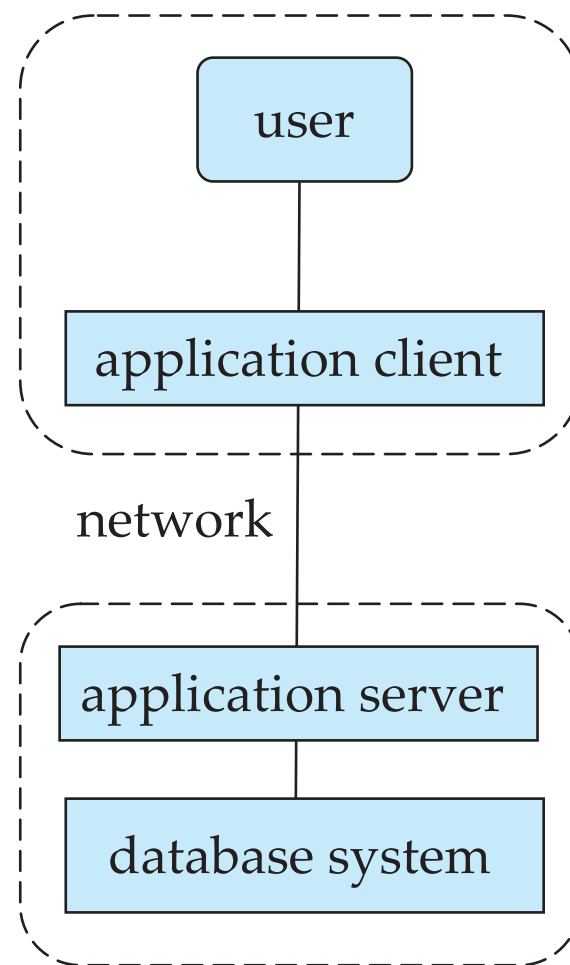


# 两层和三层C/S系统体系结构



(a) Two-tier architecture

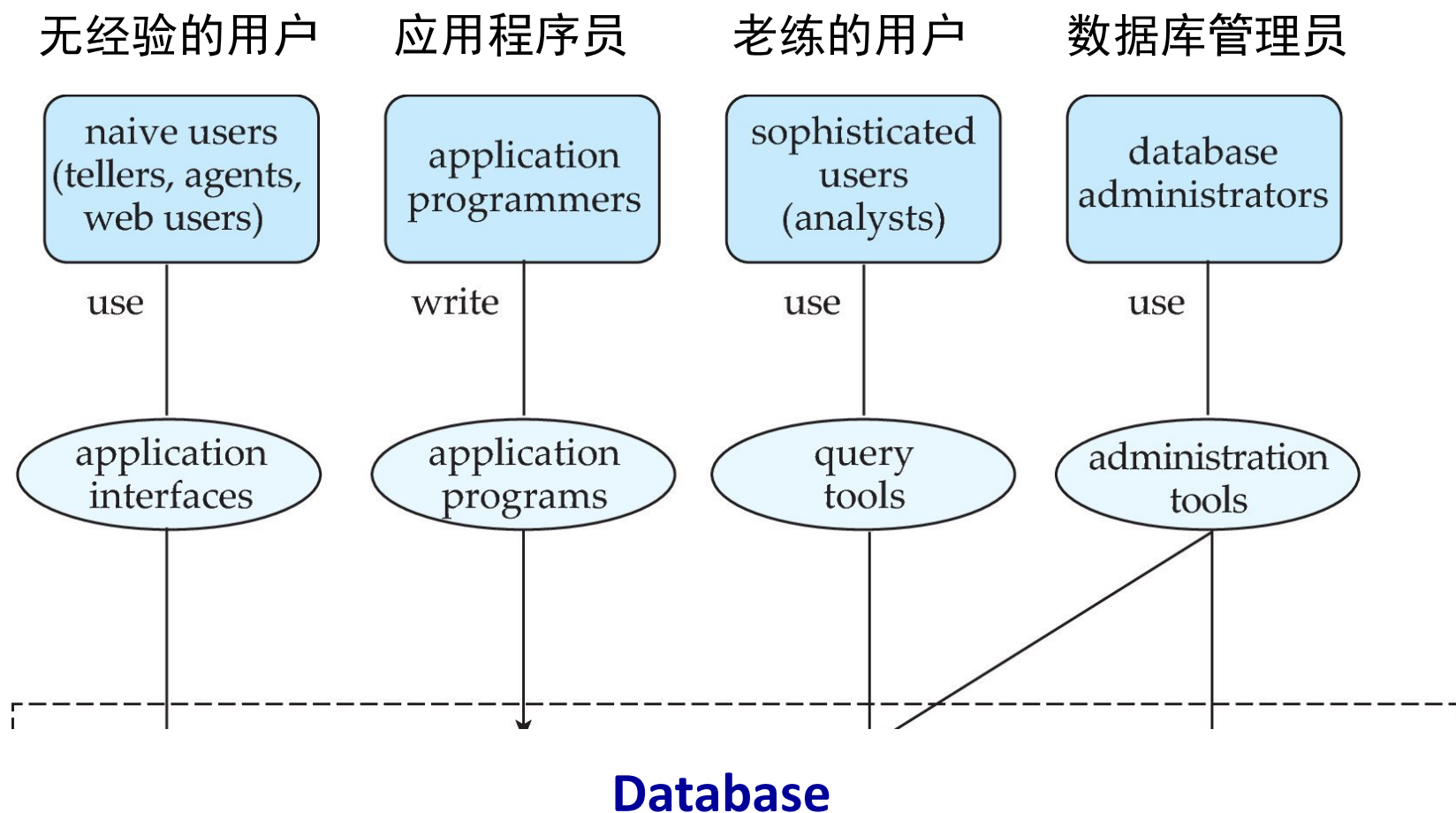
client



(b) Three-tier architecture

server

## ➤ 数据库可服务于多种用户



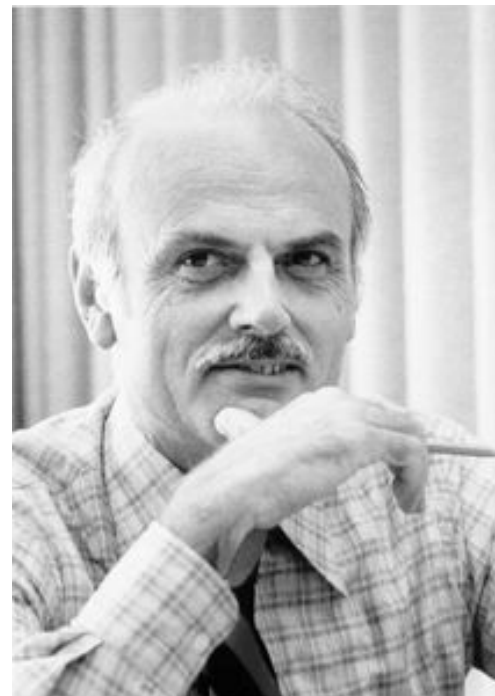
## ➤ 1950s -1960s初:

- ✓ 使用磁带存储数据
  - 顺序访问
- ✓ 打孔卡片输入



## ➤ 1960s末- 1970s:

- ✓ 硬盘存储，允许直接访问数据
  - 网状/层次数据模型
- ✓ Edgar Frank "Ted" Codd提出了关系数据模型
  - 提出非过程化的查询方法，屏蔽所有查询实现细节
  - 获得图灵奖



Codd, 1923-2003

## ➤ 1980s:

- ✓ 关系模型数据库进入商业系统应用，如IBM DB2、Oracle、Ingres，SQL 成为工业标准
- ✓ 初步研究并行和分布式数据库系统、面向对象数据库系统

## ➤ 1990s:

- ✓ 互联网的兴起对数据库系统提出更高的要求：可用性更好、处理速度更快、可靠性更高

## ➤ Early 2000s:

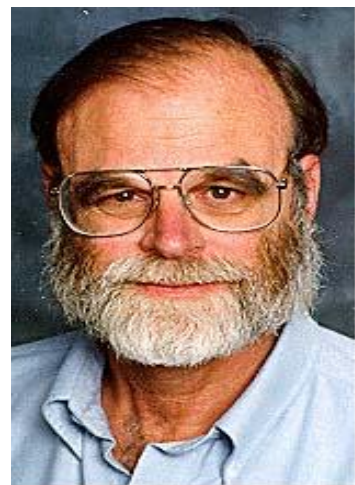
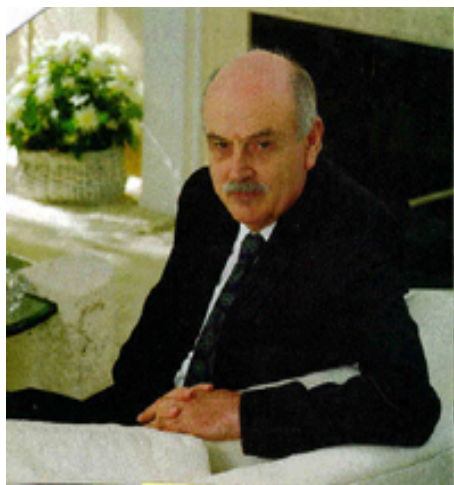
- ✓ XML的兴起和Xquery查询语言技术
- ✓ 开源数据库系统的应用，如PostgreSQL（UCB）和MySQL

## ➤ Late 2000s （NoSQL DBS）

- ✓ 大数据存储Google BigTable, Amazon Dynamo, Cassandra

## ► 图灵奖历史上的四位数据库专家：

- ✓ 1973年, Charles W. Bachman: “网状数据库之父”
- ✓ 1981年, Edgar F. Codd: “关系数据库之父”
- ✓ 1998年, James Gray: 数据库与事务处理
- ✓ 2014年, Michael Stonebraker: 现代数据库概念与实践



# What is NoSQL



## ➤ Stands for **Not Only SQL**

- ✓ Distributed, fault-tolerant architecture
- ✓ No fixed schema (formally described structure)
- ✓ No joins (typical in databases operated with SQL)
- ✓ Class of **non-relational** data storage systems, e.g. BigTable, Dynamo, PNUTS/Sherpa, ..
- ✓ All NoSQL offerings **relax one or more of the ACID properties**
- ✓ It's not a replacement for a RDBMS but compliments it



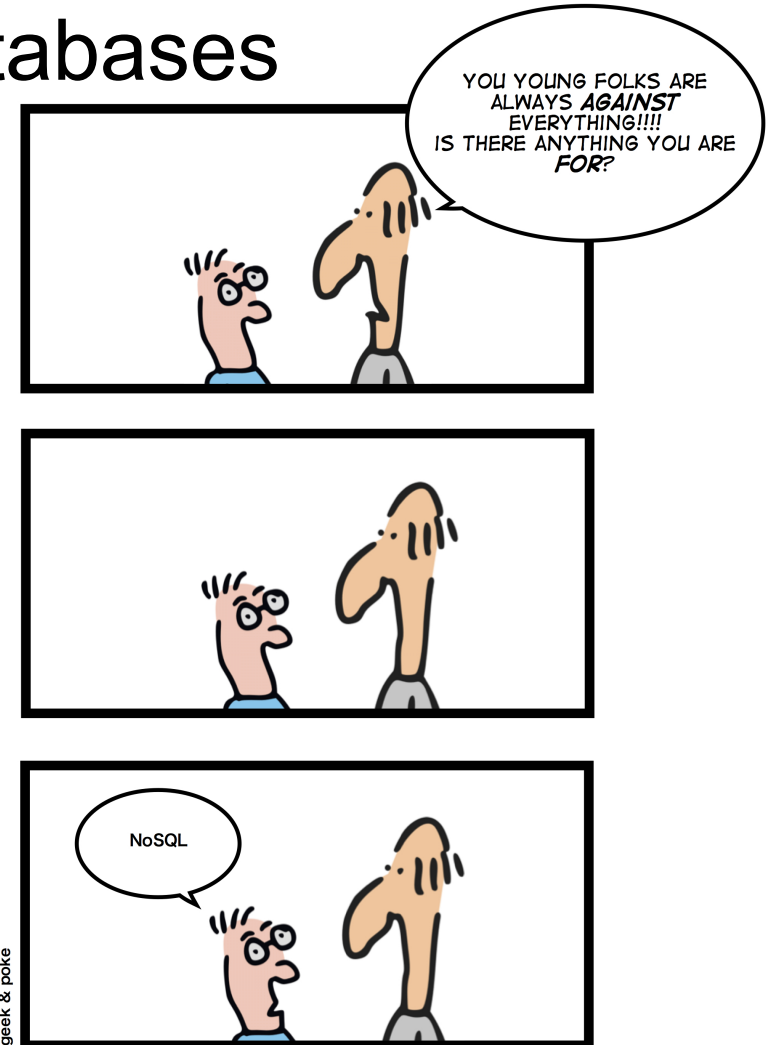
# NoSQL Products/Projects



➤ <http://www.nosql-database.org>

➤ lists **over 225** NoSQL Databases

- ✓ Cassandra
- ✓ CouchDB
- ✓ Hadoop & HBase
- ✓ MongoDB
- ✓ StupidDB
- ✓ Redis
- ✓ Hypertable
- ✓ etc.





# Where NoSQL is Used?



- Google (BigTable, LevelDB)
- LinkedIn (Voldemort)
- Facebook (Cassandra)
- Twitter (Hadoop/HBase, FlockDB, Cassandra)
- Netflix (SimpleDB, Hadoop/HBase, Cassandra)
- CERN, 欧洲核子研究组织 (CouchDB)



- 什么是数据库系统？
- 什么是数据模型？
- DML、DDL定义？
- 数据库设计的过程有哪些？
- 存储管理器与查询管理器部件包括哪些？
- 什么是事务？ACID特性是什么？为什么要进行事务管理？
- 数据库体系结构有哪些？

## ➤ p20-21 (四道题)

✓ 1.9, 1.11, 1.12, 1.15

✓ 请大家**独立**完成作业，有些题没有标准答案 😊

# Thanks for your attention!

---

Fei Xu (徐飞)

Associate Professor

Phone No.: 021-62233309

Email: [fxu@cs.ecnu.edu.cn](mailto:fxu@cs.ecnu.edu.cn)



2019 Fall 数据库系统原理