第一章 数据库系统引论

1.1 数据管理的发展

一、数据密集型应用与数据管理

1.数据（data）

数据的概念：Data是指对现实世界中事物或事物之间关系（通常称实体、实体的属性、实体间的联系）的一组描述符。数据可用数字、文本、图像...等类型/形式/格式来表示。
元数据（metadata）：元数据用于描述数据，是关于数据的数据。这是计算机科学中一类特殊的数据。
原始数据（raw data）vs 处理后的数据：原始数据一般是指对现实世界的观测值，如仪表设备的输出；广义地说，是由物理量转换来的符号（symbol）处理后的数据：原始数据需被输入到计算机中进行存储、处理或传输。数据处理通常是分阶段进行的，“原始”可以是一个相对概念，因此，来自某个阶段的“处理后的数据”可看作是下一个阶段的“原始数据”。

2.数据 vs 程序（program）

程序是规定计算机执行任务的一组指令
从这种意义上说，**数据**是指计算机可使用的、除程序代码（code）外的任何东西

3.数据 vs 信息 vs 知识

这三个术语常常交迭使用，三者的主要区别在于抽象的层次：**数据是抽象的最低层；信息其次；知识的抽象层次最高**
通过**数据处理与分析**（data processing and analysis），计算机系统将数据转换为信息或知识

4.数据密集型应用（data-intensive applications）

特点：数据量大（e.g., exceeding MB level）
持久数据（persistent data）
共享数据（shared data）

数据密集型应用中的核心技术是**数据管理**（data management），其主要任务：
数据组织与编码（data organization and coding）
数据存储、索引（data storage and indexing）
数据访问/检索/查询（data access/retrieval/query）
数据更新与维护（data updating and maintenance）
数据安全（data security）...

数据管理是数据处理的基础

5.数据管理方法

早期的**文件系统（file system）**方法的局限性：
①**数据分离与孤立**：跨文件的数据访问与数据处理难以实现
②**数据冗余**：数据存储与维护数据代价大；不一致性
③**程序 - 数据依赖性强**：数据结构定义靠应用代码实现，应用程序对数据文件也过分依赖，可维护性差
④**文件格式互不兼容**：不同编程语言定义的数据文件在格式上互不兼容，数据的综合处理难以实现
⑤**固定的应用程序（查询及报表）**：应用程序需事先设计好；即兴的（ad hoc） 查询与报表无法实现
⑥**无法提供数据管理及其辅助功能**：文件读写等操作；无数据共享访问、完整性、安全性、可恢复性，等

现在的**数据库（database, DB）**方法：
一个数据库是一个逻辑上相关的可共享数据集。数据库方法借助特殊的软件系统—**数据库管理系统（DBMS）**来实现数据管理。（**核心**）
数据库中既存储**业务数据**，又存储描述业务数据的**元数据**——称为**数据字典（data dictionary, DD）**或**数据目录（data catalog）**
DD使数据库具有**自描述性（self-describing）**，DBMS提供了**数据抽象（data abstraction）**、**程序—数据独立性（independence）**以及一系列**数据管理辅助功能**（见后文），形成了有效的数据管理方法。

二、数据库技术的发展历史

数据库以数据模型（data model）来分型、分代：

1.第一代：层次（hierarchical）& 网状（network）数据库

第一个DBMS：1964年，美国通用电器公司的Charles W. Bachman【1973年ACM图灵奖】等人开发的IDS (Integrated Data Store | 集成的数据存储)，奠定了**网状数据库**的基础
第一个商品化的层次DBMS：1960年代末，IBM公司推出的**层次数据库**管理系统IMS (Information Management System | 信息管理系统)

2.第二代：关系（relational）数据库

理论：1970年，IBM公司的 E.F. Codd【1981年ACM图灵奖】, “A relational model of data for large shared data banks”, Communication of the ACM, Vol. 13, No. 6 (1970) 奠定了关系数据库的理论基础
产品：1977前后，IBM公司的原型系统System R→商品化的关系数据库产品SQL/DS及DB2；
UC Berkeley的M. Stonebraker【2014年ACM图灵奖】等人于1974开始的原型系统INGRES，后来1980年初由INGRES公司进行商品化
1980后RDB技术&产品大发展！Jim Gray因在□事务管理等方面的贡献获1998年ACM图灵奖

3.第三代：后关系（post-relational）数据库

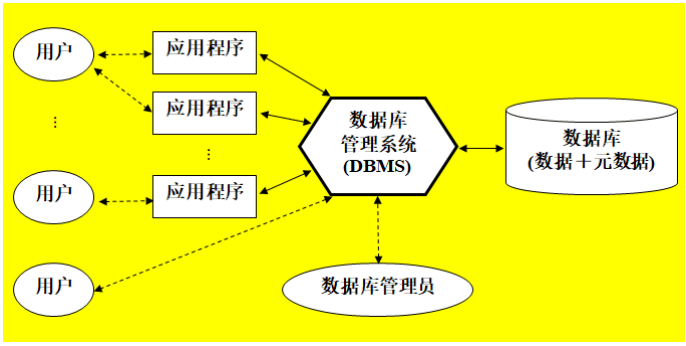
采用新的数据模型or扩充关系数据模型，产生新型数据库
e.g., object-oriented (OO), object-relational (OR), deductive/logical models …
用于管理复杂数据的高级数据库
e.g., semi-structured data, text, spatial, temporal, multimedia, statistical, scientific, engineering, □data streams, moving objects, Web-based (XML, RDF) databases …
面向数据分析的数据库（数据管理）技术：
e.g., data warehousing (DW) & online analytical processing (OLAP), data mining (DM) & knowledge discovery (KDD), online analysis mining (OLAM)

1.2 数据库系统

一、数据库系统

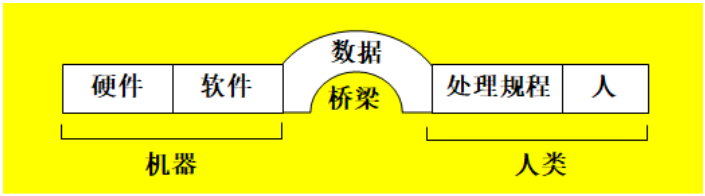
1.数据库系统（database system）

由数据库、数据库管理系统（DBMS）、数据库应用程序和创建、维护与使用数据库的人（people）所组成的系统。如下图所示：



2.DBMS环境（environment）

①**硬件**：运行DBMS软件和应用程序的计算机或网络
②**软件**：DBMS、应用程序、操作系统、甚至网络软件
③**数据**：即数据库，包含业务数据和元数据
④**处理规程（procedures）**：指支配数据库设计与使用的指令与规则，包括如何启动与停止DBMS、如何登录到DBMS、如何使用特定的DBMS工具或应用程序、如何为数据库建立后备（backup）、如何处理硬件或软件故障（failures）、如何维护数据库，等等
⑤**人**：相关人员，主要是数据库管理员和最终用户



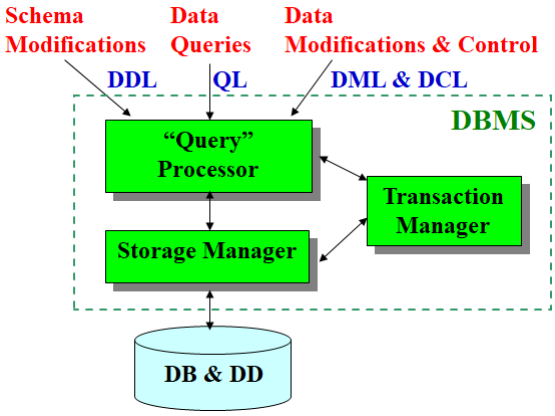
3.与数据库系统打交道的五类人员

(1) 数据管理员（data administrator, DA）：负责一个组织的数据资源的规划、政策与标准的制订、数据库的概念设计，以确保数据库开发最终能支持组织的目标
(2) 数据库设计员（database designer）：负责数据库的逻辑设计（模式）和物理设计（存储和存取技术等）
(3) 数据库管理员（database administrator, DBA）：负责数据库的物理实现、数据控制与系统运行维护，并确保数据库应用达到满意的性能
(4) 数据库应用开发员（application developer）：负责应用程序的设计与实现，以便为数据库最终用户提供所需的数据访问和数据操纵功能
(5) 数据库最终用户（end-user）：使用应用程序或数据库语言（如SQL）访问数据库的客户（clients）

二、DBMS

1.DBMS组成

查询处理器
存储管理器
事务管理器



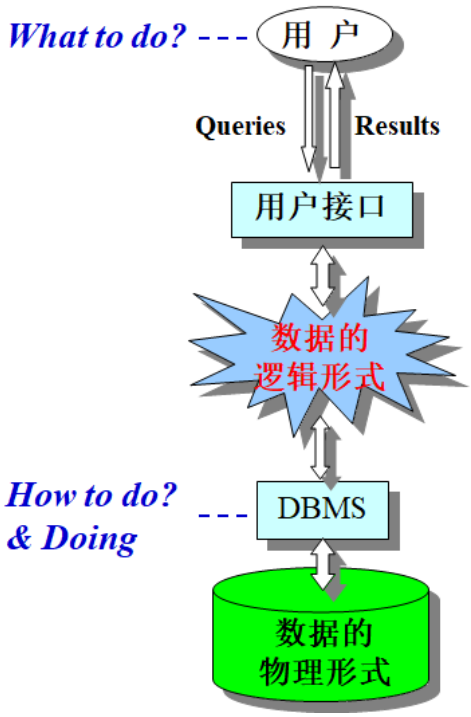
2.数据库语言

Data Definition Language, DDL
Query Language, QL
Data Manipulation Language, DML
Data Control Language, DCL

3.DBMS功能

1）提供高级的用户接口
交互式接口（interactive interface）
编程接口（programming interface）

用户接口的背后是数据库语言（database language），如关系数据库的结构化查询语言（SQL）



2）查询处理与优化
查询（queries）为广义的概念，如SQL语言中包括：SELECT, INSERT, DELETE, UPDATE语句
查询处理（processing）与优化（optimization）：语法检查，语义分析，制定执行策略，执行查询并返回结果

3）数据目录管理
数据目录/字典（ data catalog/dictionary, DD ）：存放元数据的（系统）数据库
元数据（metadata）： 数据定义信息，存储结构信息，其他管理信息

4）并发控制

由于数据库是共享的，即用户（提交的事务）是并发访问数据库的，因此，DBMS必须要有**并发控制（concurrency control）**机制，以协调并发事务的执行不发生"冲突"，同时确保数据完整性（data integrity）

5）数据库恢复

数据库系统可能会发生故障而导致数据库失效（failure），因此，DBMS必须要有**数据库恢复（database recovery）**机制，以保证数据库始终处于一致（consistency）状态

6）完整性约束检查

数据库中的数据必须遵守一定的约束才能保证其正确性，并向用户提供正确的信息。约束可分为语法的（syntactical）约束和语义的（semantic）约束，后者称为**完整性约束（integrity constraints）**
DBMS必须提供完整性约束检查功能

7）访问控制

在共享的数据环境中，DBMS必须控制不同用户对数据库的不同访问特权（privileges），以保证数据库的安全性——称为**访问控制（access control）**

1.3 数据抽象与数据独立性

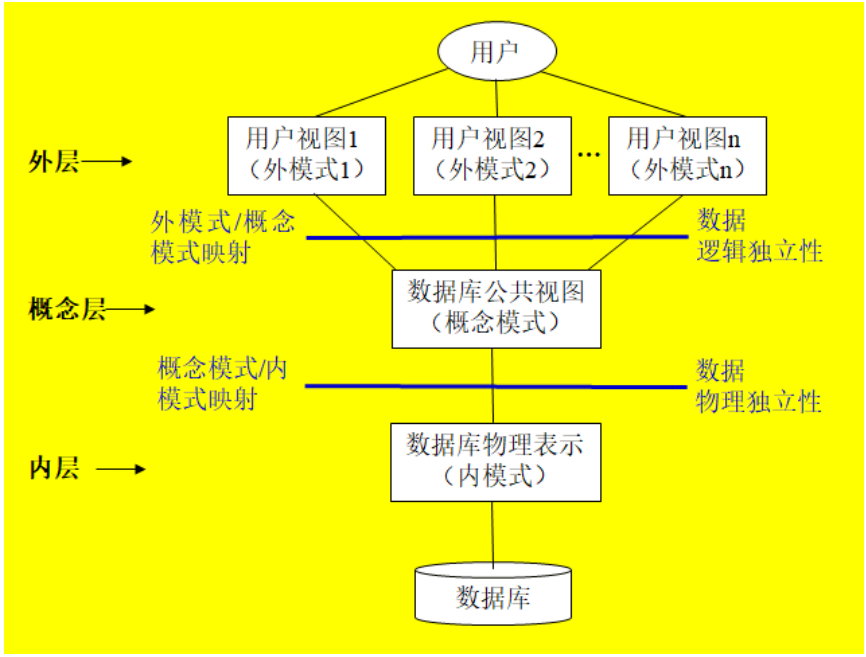
1.数据库模式（schema）vs 实例（instance）

区分数据库的描述（型）与数据库中数据（值）！
数据库模式：指数据库中全体数据的逻辑结构与特征的描述，也称数据库的内涵（intension）；
数据库实例：指数据库中特定时间点的数据，即数据库的特定状态（state），也称数据库的外延（extension）。一个模式可以有许多个实例。
数据库模式相对稳定，数据库实例经常变动！

2.ANSI-SPARC体系结构

1975年，美国国家标准学会（American National Standards Institute, ANSI）下属的标准规划与需求委员会（Standards Planning And Requirements Committee Architecture, SPARC）提出了称为**ANSI-SPARC Architecture**的DBMS抽象设计标准，如下页中图所示。
这个三层体系结构（three-level architecture）的**核心是三级模式+两级映射（mapping）**，其目标是将**数据库的物理表示**与**数据库的用户视图**进行分离，即提供**数据独立性（data independence）**。

3.ANSI-SPARC三层体系结构



外层（external level）是多个**外模式（external schema）**，每个外模式描述数据库中与特定用户相关的部分，即数据库的用户视图（user's view）
概念层（conceptual level）是一个**概念模式（conceptual schema）**，描述数据库中包含什么数据（以及数据之间的关系），即数据库公共视图（community view）
内层（internal level）是一个内模式（internal schema），描述数据库中数据是如何存储的，即数据库的物理表示（physical representation）

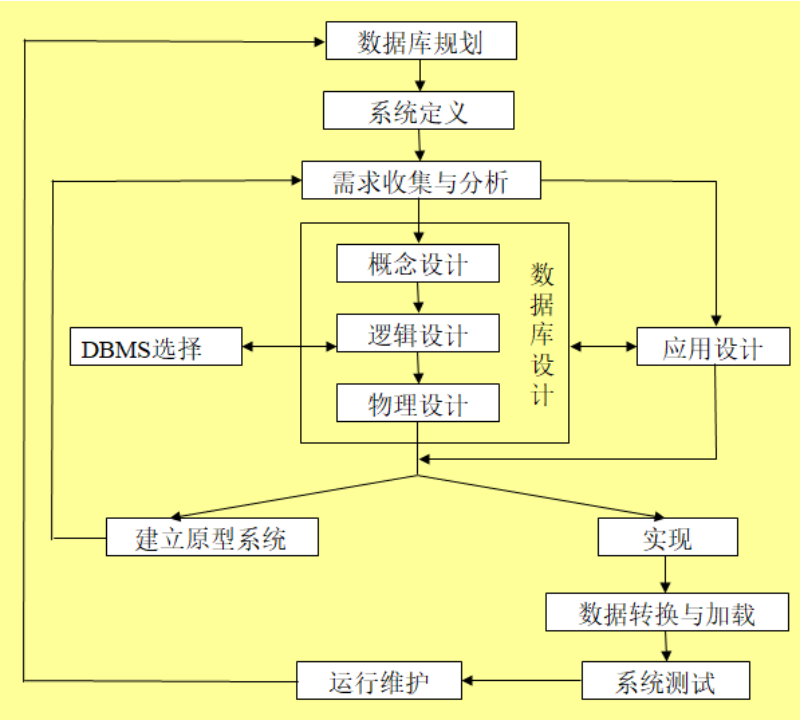
4.DBMS维护三级模式间的两种映射（mapping）

通过外模式与概念模式之间的映射机制来实现**逻辑（logical）独立性**——指外模式（以及外模式上运行的应用程序）对概念模式改变的抗扰性（immunity：adj. 不受影响的；有免疫力的）
通过概念模式与内模式之间的映射机制来实现**物理（physical）独立性**——指概念模式（和外模式）对内模式改变的抗扰性
数据独立性大大降低了数据库的使用与维护代价！

1.4 数据库的生命周期

数据库有生命周期（life cycle）

数据库系统是数据密集型应用（如一个组织的**信息系统**）中的基本和重要构件
信息系统的开发常用**信息工程方法（IEM）**Information Engineering Methodology： 一种面向数据的方法（data-oriented approach），即以数据为中心，注重对一个组织的业务目标的理解，在对业务过程进行分析和数据建模的基础上，分阶段开发信息系统的方法。
运用IEM进行信息系统开发过程中，数据库也有相应的生命周期，其各阶段如下页图所示。



阶段	主要活动
数据库规划	规划如何最有效和高效地实现生命周期的各阶段。
系统定义	规定数据库系统的范围与边界，包括用户、用户视图和应用领域。
需求收集与分析	为新的数据库系统收集与分析需求。
数据库设计	数据库的概念设计、逻辑设计与物理设计。
DBMS选择	为数据库系统选择一个合适的DBMS。
应用设计	设计访问与操纵数据库的用户接口与应用逻辑。
建立原型系统	为将实现的数据库系统构造一个原型，以便用户和设计人员进行评价。
实现	建立物理数据库定义与应用程序。
数据转换与加载	从旧系统加载数据到新系统，尽可能将现有应用与数据转换到新数据库。
系统测试	数据库系统错误测试，用户需求可满足性验证。
运行维护	监控与维护运行中的数据库；可能的数据库重构以满足新的需求。