数据库系统概论（1）

- **第一章 引言（每年都考，就几分吧估计）**
  - **1.1、数据库系统概述（不重要）**
    - 数据库系统概论是关于数据库系统的一个学科领域。数据库系统概论主要涉及数据库的基本概念、特点以及数据模型等内容。数据库系统概论中包括了数据库的四个概念，即数据、数据库、数据库管理系统（DBMS）和数据库系统（DBS）。数据库是长期储存在计算机内、有组织的、可共享的大量数据的集合。数据库管理系统是位于用户与操作系统之间的一层数据管理软件，它主要用于科学地组织和存储数据、高效地获取和维护数据。数据库系统主要是由数据库、数据库管理系统（及其开发工具）、数据库管理员以及应用程序所构成的一套人机系统。
    - 数据库系统的特点包括数据的整体结构化、数据的共享性高且易扩充、数据独立性高以及数据由数据库管理系统统一管理和控制。数据的整体结构化是指数据库不再仅仅针对某一个应用，而是面向全组织，数据内部结构化且整体也是结构化的，数据之间具有联系。数据的共享性高使得数据可以被多个用户、多个应用共享使用，这有助于减少数据冗余、避免数据不相容性与不一致性，并使系统易于扩充。数据独立性高由数据库管理系统的二级映像功能来保证，其中包括物理独立性和逻辑独立性。物理独立性指用户的应用程序与数据库中数据的物理存储是相互独立的，当数据的物理存储改变了，应用程序不需要改变。逻辑独立性指用户的应用程序与数据库中数据的逻辑结构是相互独立的，当数据的逻辑结构改变了，应用程序也不需要改变。数据由数据库管理系统统一管理和控制，数据库管理系统提供数据的安全性保护、数据的完整性检查、并发控制和数据库恢复技术等功能。
    - 总结起来，数据库系统概论是关于数据库系统的一个学科领域，涉及了数据库的基本概念、特点以及数据模型等内容。数据库系统概论中介绍了数据库的四个概念，即数据、数据库、数据库管理系统（DBMS）和数据库系统（DBS）。数据库系统具有数据的整体结构化、数据的共享性高且易扩充、数据独立性高以及数据由数据库管理系统统一管理和控制等特点。
  - **1.2、数据库的四个概念（重点）**
    - **·数据（Data）： 数据是数据库中存储的基本对象，它是描述事物的符号记录。**
    - **·数据库（Database）： 数据库是长期储存在计算机内、有组织的、可共享的大量数据的集合。**
    - **·数据库管理系统（DBMS）： 数据库管理系统是位于用户与操作系统之间的一层数据管理软件，它是一个大型复杂的软件系统，它主要用于科学地组织和存储数据、高效地获取和维护数据。**
    - **·数据库系统（DBS）： 数据库系统主要是由数据库、数据库管理系统（及其开发工具）、数据库管理员以及应用程序所构成的一套人机系统。**
  - **1.3、 数据库系统的应用(不重要)**
    - •企业信息：

- □销售：用于存储客户、产品和购买信息。
- □会计：用于存储付款、收据、账户余额、资产和其他会计信息。
- •银行和金融
- □银行业：用于存储客户信息、账户、贷款，以及银行的交易记录。
- □信用卡交易：用于记录信用卡消费的情况和产生每月清单。
- ・大学：
- □用于存储学生信息、课程注册和成绩。(此外，还存储通常的单位信息，例如人力资源和会计信息等。)
- 综上所述，数据库已经成为当今几乎所有企业不可默认的组成部分，它不仅存储大多数企业都有的普通的信息，也存储各类企业特有的信息。

- **1.4、 数据库发展的历史**
  - ·20世纪最后的40年中，数据库的使用在所有的企业中都有所增长；
  - ·20世纪90年代末的互联网革命急剧地增加了用户对数据库的直接访问；
  - 参考教材29页1.13
- **1.5、 数据库系统的目标**
  - 数据库系统作为商业数据计算机化管理的早期方法而产生。
  - **早期文件处理系统的弊端：**
  - **•数据的冗余和不一致（data redundancy and inconsistency ）**
    - 由于文件和程序是在很长的一段时间内由不同的程序员创建的，不同文件可能有不同的结构，不同程序可能采用不同的程序设计语 言写成。此外，相同的信息可能在几个地方（文件）重复存储。例如，如果某学生有两个专业 （例如，音乐和数学），该学生的地址和电话号码就可能既出现在包含音乐系学生记录的文件 中，又出现在包含数学系学生记录的文件中。这种冗余除了导致存储和访问开销增大外.还可能导致数据不一致性(data inconsistency),即同一数据的不同副本不一致。例如，学生地址的更改可能在音乐系记录中得到反映而在系统的其他地方却没有。
  - **•数据访问困难(difficulty in accessing data)**
    - 假设大学的某个办事人员需要找出居住在某个特定邮编地区的所有学生的姓名，于是他要求数据处理部门生成这样的一个列表。由于原始系统的设 计者并未预料到会有这样的需求，因此没有现成的应用程序去满足这个需求。但是，系统中却有一个产生所有学生列表的应用程序。这时该办事人员有两种选择：一种是取得所有学生的列表并从中手工提取所需信息，另一种是要求数据处理部门让某个程序员编写相应的应用程序。 这两种方案显然都不太令人满意。假设编写了相应的程序，几天以后这个办事人员可能又需要 将该列表减少到只列出至少选课60学时的那些学生。可以预见，产生这样一个列表的程序又不存在，这个职员就再次面临前面那两种都不尽如人意的选择。 这里需要指出的是，传统的文件处理环境不支持以一种方便而高效的方式去获取所需数据。 我们需要开发通用的、能对变化的需求做出更快反应的数据检索系统。
  - **・数据孤立(data isolation)**

- 由于数据分散在不同文件中，这些文件又可能具有不同的格式，因此，编写新的应用程序来检索适当数据是很困难的。

- **· 完整性问题(integrity problem)**
  - 数据库中所存储数据的值必须满足某些特定的一致性约束 (consistency constraint) 。假设大学为每个系维护一个账户，并且记录各个账户的余额。我们还假设大学要求每个系的账户余额永远不能低于零。开发者通过在各种不同应用程序中加入适当的代码来强制系统中的这些约束。然而，当新的约束加入时，很难通过修改程序来体现这些新的约束。尤其是当约束涉及不同文件中的多个数据项时，问题就变得更加复杂了。
- **•原子性问题(atomicity problem)。**
  - 如同任何别的设备一样，计算机系统也会发生故障。一旦故障发生，数据就应被恢复到故障发生以前的一致的状态，对很多应用来说，这样的保证是至关重要的。让我们看看把A系的账户余额中的500美元转入B系的账户余额中的这样一个程序。假 设在程序的执行过程中发生了系统故障，很可能A系的余额中减去的500美元还没来得及存入B系的余额中，这就造成了数据库状态的不一致。显然，为了保证数据库的一致性，这里的借 和贷两个操作必须是要么都发生，要么都不发生。也就是说，转账这个操作必须是原子的——它要么全部发生要么根本不发生。在传统的文件处理系统中，保持原子性是很难做到的。
- **•并发访问异常(concurrent-access anomaly)**
  - 为了提高系统的总体性能以及加快响应速度，许多 系统允许多个用户同时更新数据。实际上，如今最大的互联网零售商每天就可能有来自购买者 对其数据的数百万次访问。在这样的环境中，并发的更新操作可能相互影响，有可能导致数据 的不一致。设A系的账户余额中有10 000美元，假如系里的两个职员几乎同时从系的账户中取 款(例如分别取出500美元和100美元)，这样的并发执行就可能使账户处于一种错误的(或不 一致的)状态。假设每个取款操作对应执行的程序是读取原始账户余额，在其上减去取款的金额.然后将结果写回。如果两次取款的程序并发执行，可能它们读到的余额都是10000美元， 并将分别写回9500美元和9900美元。A系的账户余额中到底剩下9500美元还是9900美元视哪个程序后写回结果而定，而实际上正确的值应该是9400美元。为了消除这种情况发生的可能性，系统必须进行某种形式的管理。但是，由于数据可能被多个不同的应用程序访问，这些程序相互间事先又没有协调，管理就很难进行。

- **1.6、 数据视图**
  - 数据库系统是一些互相关联的数据以及一组使得用户可以访问和修改这些数据的程序的集合。数 据库系统的一个主要目的是给用户提供数据的抽象视图，也就是说，系统隐藏关于数据存储和维护的 某些细节。
  - 1.6.1 数据抽象
    - 一个可用的系统必须能高效地检索数据。这种高效性的需求促使设计者在数据库中使用复杂的数 据结构来表示数据。由于许多数据库系统的用户并未受过计算机专业训练，系统开发人员通过如下几 个层次上的抽象来对用户屏蔽复杂性，以简化用户与系统的交互：
      - **· 物理层(physical level)**

- 最低层次的抽象，描述数据实际上是怎样存储的。物理层详细描述复杂的底层数据结构。

- · **逻辑层(logical level)**
  - 比物理层层次稍高的抽象，描述数据库中存储什么数据及这些数据间存 在什么关系。这样逻辑层就通过少量相对简单的结构描述了整个数据库。虽然逻辑层的简单结 构的实现可能涉及复杂的物理层结构，但逻辑层的用户不必知道这样的复杂性。这称作**物理数据独立性**(physical data independence) 数据库管理员使用抽象的逻辑层，他必须确定数据库中 应该保存哪些信息。
- · **视图层(view level)**
- 最高层次的抽象.只描述整个数据库的某个部分。尽管在逻辑层使用了比 较简单的结构，但由于一个大型数据库中所存信息的多样性，仍存在一定程度的复杂性。数据库系统的很多用户并不需要关心所有的信息，而只需要访问数据库的一部分。视图层抽象的定义正是为了使这样的用户与系统的交互更简单。系统可以为同一数据库提供多个视图。

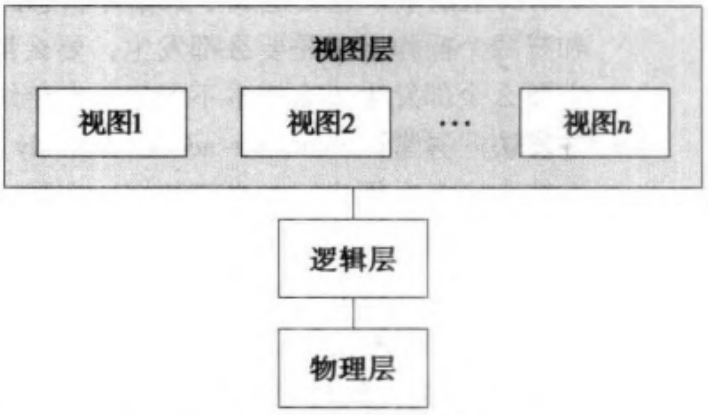

图 1-1　数据抽象的三个层次

- 1.6.2 实例与模式
  - 特定时刻存储在数据库中的信息的集合称作数据库的一个**实例(instance)**
  - 数据库的总体设计称作数据库**模式(schema)**
  - 数据库模式和实例的概念可以通过与用程序设计语言写出的程序进行类比来理解。数据库模式对应于程序设计语言中的变量声明(以及与之关联的类型的定义)。每个变量在特定的时刻会有特定的值，程序中变量在某一时刻的值对应于数据库模式的一个实例。 根据前面我们所讨论的不同的抽象层次，数据库系统可以分为几种不同的模式：
  - · **物理模式 (physical schema)**在物理层描述数据库的设计；
  - · **逻辑模式(logical schema)**则在逻辑层描述数据库的设计；
  - 数据库在视图层也可以有几种模式，有时称为**子模式(subschema)**,它描述了数据库的不同视图。
  - 在这些模式中，因为程序员使用逻辑模式来构造数据库应用程序，从其对应用程序的效果来看， 逻辑模式是目前最重要的一种模式。物理模式隐藏在逻辑模式下，并且通常可以在应用程序丝毫不受 影响的情况下被轻易地更改。应用程序如果不依赖于

物理模式，它们就被称为是具有**物理数据独立性 (physical data independence)**,因此即使物理模式改变了它们也无需重写。

- 1.6.3 数据模型
  - 数据库结构的基础是**数据模型(data model)**，数据模型是一个描述数据、数据联系、数据语义以及 一致性约束的概念工具的集合。数据模型提供了一种描述物理层、逻辑层以及视图层数据库设计的方式。数据模型可被划分为四类：
    - • **关系模型(relational model)** 关系模型用表的集合来表示数据和数据间的联系。每个表有多个 列，每列有唯一的列名。关系模型是基于记录的模型的一种。基于记录的模型的名称的由来是因为数据库是由若干种固定格式的记录来构成的。每个表包含某种特定类型的记录。每个记录类型定义了固定数目的字段(或属性)。表的列对应于记录类型的属性。关系数据模型是使用最 广泛的数据模型，当今大量的数据库系统都基于这种关系模型〉
    - • **实体-联系模型(entity-relationship model)** 实体-联系(E-R)数据模型基于对现实世界的这样 一种认识：现实世界由一组称作实体的基本对象以及这些对象间的联系构成。实体是现实世界中可区别于其他对象的一件“事情”或一个“物体”。实体-联系模型被广泛用于数据库设计。
    - • **基于对象的数据模型(object-based data model)** 面向对象的程序设计(特别是Java、C + +或C#) 已经成为占主导地位的软件开发方法。这导致面向对象数据模型的发展.面向对象的数据模型 可以看成是E-R模型增加了封装、方法(函数)和对象标识等概念后的扩展。
    - • **半结构化数据模型(semistructured data model)** 半结构化数据模型允许那些相同类型的数据项 含有不同的属性集的数据定义。这和早先提到的数据模型形成了对比：在那些数据模型中所有 某种特定类型的数据项必须有相同的属性集。可扩展标记语言(extensible Markup Language, XML)被广泛地用来表示半结构化数据。

- **1.7、 数据库语言**
  - 数据库系统提供**数据定义语言(data-definition language)**来定义数据库模式，以及**数据操纵语言(data-manipulation language)**来表达数据库的查询和更新。而实际上，数据定义和数据操纵语言并不是 两种分离的语言，相反地，它们简单地构成了单一的数据库语言(如广泛使用的SQL语言)的不同部分。
  - 1.7.1 数据定义语言
    - 数据库模式是通过一系列定义来说明的，这些定义由一种称作**数据定义语言(Data-Definibon H0~l Language, DDL)**的特殊语言来表达。存储在数据库中的数据值必须满足某些**一致性约束(consistency constraint)**，测试完整性约束的方法：**域约束(domain constraint)，参照完整性(referential integrity)，断言(assertion)，授权(authorization)**。
  - 1.7.2 数据操作语言
    - **数据操纵语言(Data-Manipulation Language, DML)**是这样一种语言，它使得用户可以访问或操纵那 些按照某种适当的数据模型组织起来的数据。有以下访问类型：
    - **· 对存储在数据库中的信息进行检索：** 、
    - **· 向数据库中插入新的信息。**

- **•从数据库中删除信息。**
- **•修改数据库中存储的信息。**
- 通常有两类基本的数据操纵语言：
  - **过程化DML(procedural DML)**要求用户指定需要什么数据以及如何获得这些数据。
  - **声明式DML(declarative DML)**(也称为非过程化DML)只要求用户指定需要什么数据，而不指 明如何获得这些数据。
- **查询(query)**是要求对信息进行检索的语句。DML中涉及信息检索的部分称作**查询语言(query language)** 。

- **1.8、关系数据库**
  - 关系数据库基于关系模型，使用一系列表来表达数据以及这些数据之间的联系。关系数据库也包括DML和 DDL。图1-2展示了 一个关系数据库示例.它由两个表组成：其一给出大学教师 的细节，其二给出大学中各个系的细节。 第一个表是instructor表，表示，例如，ID为22222的名 叫Einstein的教师是物理系的成员，他的年薪为95 000美元。 第二个表是department表，表示，例如，生物系在Watson大楼，经费预算为90000美元。

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) *instructor*表

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) *department*表

图 1-2　关系数据库的一个实例

  - 不难看出，表可以如何存储在文件中。例如，一个特殊的字符(比如逗号)可以用来分隔记录的不同属性，另一特殊的字符(比如换行符)可以用来分隔记录。对于数据库的开发者和用户，关系模型屏 蔽了这些低层实现细节。 我们也注意到，在关系模型中，有可能创建一些有问题的模式，比如出现不必要的冗余信息。例如，假设我们把系的budget存储为instructor记录的一个属性。那么，每当一个经费预算的值(例如，物H21理系的经费预算)发生变化时，这个变化必须被反映在与物理系相关联的所有教员记录中。