第一章 概 论

这一章主要讨论与数据库相关联的基本问题及概念。从数据库定义出发讨论建立数据库所需要的基本技术；数据库系统的特点与功能；数据库系统的成分；数据库在计算机学科中的地位；数据库，特别是关系数据库形成的过程及数据库技术的发展趋势。从基本问题、特点等方面引出本书以后各章的内容。

1.1 数据库基本问题及概念

数据库这个词当前许多人都把它作为数据库系统的同义词在使用。然而，人们可以通过所讨论的内容很容易确定数据库一词在使用时的含意是指数据库系统还是指数据库。从原理上来讨论，严格地讲，数据库只是数据库系统的一个组成部分。为此，首先给出数据库系统的概括的定义：

大量的经过加工整理的、存储在称作数据库中的数据，由数据库管理系统管理，为多个不同的应用（或用户）共同使用的数据处理系统，可称为数据库系统。

由此可见，数据库系统有两个主要组成部分：**数据库管理系统**和**数据库**。其中，当然也包括了存放数据的存储介质及其设备和使用、维护数据库系统的人员。

数据库管理系统（DBMS）

数据库管理系统是数据库系统的一个软件，它允许一个或多个使用者对数据库中的**抽象数据**提出请求（包括询问和修改），并以合乎使用者要求的格式提供给使用者。抽象数据的含意是指数据由**数据模型**表示。所以，数据库管理系统必须与构成数据库的数据模型的特性相匹配。

数据库（DB）

数据库是指在辅助存储器中的存储数据。这些数据是现实世界中的一些有关信息，它们在特定的组织（企、事业）中能为多种应用（或用户）服务。它们以数据模型所确定的数据结构方式存储，并能以有效的存取方法为操纵数据的语言提供快速响应。

数据库系统作为一种数据处理系统这一观点来看，也属于一种程序设计系统。当前，这种系统均是在特定的操作系统支持下由数据库管理系统提供功能管理数据。使用者将需处理的数据（信息）构造成数据库，这属于数据库设计问题，是一种面向应用的程序设计；为了提供数据管理功能而完成数据库管理系统的设计，是属于计算机系统软件的设计问题，是一种面向计算机系统的程序设计。数据库设计是一种工程性的工作。粗略地讲，它首先对处理的信息（数据）进行需求分析，综合整理出处理对象的概念结构。尽管这种概念结构可以用数据模型的方法来抽象，但它可以独立于数据库管理系统，也就是说，它可以与数据库管理系统支持的数据模型不同，它可以不考虑对数据如何操作，但是却能明显地表示出某个特定组织如学校、工厂、医院……等部门的情况。实体-关联模型（E-R 模型）

是当前普遍使用的用于概念结构设计的模型。当完成概念结构设计后，根据实际情况再选定某一类数据库管理系统，完成数据库的具体设计。当然这两个过程是不断修整和不断完善的。早期的数据库系统都是基于网络数据模型和层次数据模型。基于这类DBMS的数据库的设计很大程度上取决于设计者的经验。当关系型数据库系统问世后，由于关系模型有坚实的数学基础，改变了只凭经验的局面，而可以用理论指导实践，使数据库的设计更加尽善尽美。

因此，数据库系统的研究包括了三个方面：

- 数据库管理系统的研究；
- 数据库理论的研究；
- 数据库设计方法及工具的研究。

数据库管理系统的研究包括了DBMS应具有什么样功能的原理性问题和如何实现的技术性问题。其中，当处理方法不同有：集中式、分布式、联邦式；或处理对象相异有：事务性数据、工程数据、多维数据（空间、时间）、知识性数据（规则，事实）时，其原理与技术各有不同的内容。当前，DBMS的研究已从集中式数据库管理系统向分布式数据库管理系统（DDBMS）、知识库管理系统（KBMS）等方面延伸到适应各种应用领域。数据库理论的研究围绕关系数据库理论、事务理论、逻辑与数据库（演绎数据库）、面向对象的数据库、知识库等方面，探索新思想的表达、提练、简化，最后使其为人们所理解；研究新算法以提高数据库效率。应该说理论研究对数据库技术的进展起到不可估量的作用，但还有更多内容需进一步开拓。上面已提到数据库设计是工程性工作，数据库设计得好或坏直接影响整个系统的效率和品质，且又因其难度大、耗时多，所以对数据库设计方法学的研究十分活跃。其核心是遵循软件工程的一般原则和方法，按照数据库系统自身的特点，用一种称为数据库工程的方法进行。面对这种方法，传统的手工设计方式已不能适应，而必须辅之以有效的综合性工具系统，用计算机来完成设计，即向自动化设计发展。这些工具系统中研究得较深入、实用程度较高的是需求分析工具。其它面向特定DBMS的物理设计辅助工具、逻辑设计辅助工具以及综合性的设计辅助工具也正在络绎不断地出现。

本书将重点讨论关系数据理论、关系数据库管理系统以及关系分布式数据库管理系统的原理与技术。

综上所述，数据库系统应具有：（1）管理持久数据的能力；（2）有效存取大量数据的能力；（3）众多使用者使用同一数据的能力。概括地说，数据库系统应具有：持久性、有效性、共享性三个主要特点。其中，（1）仅仅指出有一个“永久”存在的数据库，其中的数据是由数据库管理系统存取和管理；（2）是指与传统的文件系统相区别，数据库管理系统可以提供快速存取数据的任意部分的功能；（3）是指数据可以为许多使用者服务。

这些特点引出一些属于数据库系统的重要而且有用的概念。

数据模型

数据库的数据模型可以看作是一种形式化描述数据、数据之间的联系以及有关的语义约束规则的抽象方法。它规定数据如何结构化和一体化，以及规定允许对这种结构化数据进行何种操作。每一种数据模型都应有其相应的支持软件，DBMS提供一种通用的方法使用户能用易于理解的结构将需要管理的信息构造成数据库，不需用计算机的机器码

(bit)去看待存储的数据。

当前流行的数据库系统中有三种主要的数据模型:层次模型、网络模型、关系模型。第二章将详细讨论关系模型以及关于数据模型研究动向。

数据模型用一种抽象的方法提供多级抽象层，可从不同角度去看待数据库，如：从整体数据逻辑结构观察；从用户关心的数据结构去观察；从存储数据的方式去观察。这在 1.3 节中将详细讨论。通常,DBMS 不必从最低层(计算机原始机器码)去管理,而从较低层开始,即把数据看成是文件。

例 1.1 某公司的雇员信息可存放在文件中,每个雇员有：公司内部编号(雇员号)EMPNUM,姓名 ENAME,……,任职部门 DEPT,工资 SAL,家庭地址 ADD 等信息。每一个信息是文件的一个字段。每个雇员的信息由所有字段信息组成的记录所表示。其记录的结构形式如：

```
Record
    EMPNUM : [10];
    ENAME  : char[25];
      ⋮
    DEPT   : char[30];
    SAL    : [6]
    ADD    : char[30]
End.
```

雇员文件则由一系列记录组成。

在关系数据库中,较低层由一组记录组成的文件可抽象成用**关系**表示。这里,粗略地暂且把关系看成是一个数据库文件。它与传统文件相似由记录组成,所不同的是关系有**关系名**、**关系模式**和**值**的概念。例 1.1 的雇员文件用关系表示时则表示为：

EMP (EMPNUM, ENAME, …, DEPT, SAL, ADD)

其中 EMP 是雇员关系的关系名,EMP(EMPNUM,ENAME,……,DEPT,SAL,ADD)是雇员关系的关系模式,括号中的 EMPNUM,ENAME……等就是相应的字段名,在关系中常称为**属性**名,所以关系模式就是关系名和属性名的集合。雇员文件的每个记录表示某个具体雇员的信息。如(10,张光,……,系统开发部,500 元,街道口 18 号)则表示了名叫张光的雇员,其雇员号为 10,在系统开发部工作,工资 500 元,家住街道口 18 号。这个对于某一雇员的具体信息称为关系的值。或称**元组**。

上述的非形式化的讨论中关系与文件有时是同义的,但概念是不同的。且在数据库系统中使用方法亦不同。关系是文件的**抽象**,其中字段的数据类型不直接提及,记录的顺序也不具体规定。关系中记录称为元组。因此,文件是记录的列表,而关系是元组的集合。

数据库的有效存取

只要是与操作系统相联系的文件系统都具有文件存取功能,这是不足为奇的。但是,当需存取一个文件的特定部分数据时,数据库管理系统就有了明显的优势。

一般来说,数据库中某个特定记录(关系的元组)的查询/修改,都能在很短时间内完成而与文件的长度无关,即数据库的存取操作在独立于文件长度的常量时间内完成。有效

性的另一个含意是数据库管理系统具有导航作用，可从多个文件中找出所需要的组合信息。

例 1.2 设有两个关系：

EMP(ENUM, ENAME, DNAME)

DEPT(DNO, DNAME, MANAGER)

其中 EMP 如前所述且已简化；DEPT 为部门关系，有三个属性：部门号 DNO，部门名 DNAME，部门经理 MANAGER。

如需查找雇员张光所在部门经理，则需要在上述两个关系中经 DNAME 属性导航。首先在 EMP 关系中找到 ENAME＝“张光”的记录，查出他所在的 DNAME＝“供销部”；然后在 DEPT 关系中找 DNAME＝“供销部”的记录，查出 MANGER＝“李星”。如果建立了**好的文件存储结构**(如：采用索引技术、hash 技术…)就可以在很短时间内得到响应。此外，在关系数据库系统中采用了有效的**查询优化策略**，则可以在更短的时间取得响应。为了重点讨论关系数据库的问题，所以本书略去了有关文件物理存储结构这方面的叙述。第六章将讨论关于查询处理的细节。

数据库语言

一般的程序设计语言通常只有在程序运行时，程序的数据才存在；而数据库系统中，数据预先就存放在存储设备中，且定义(描述)一次后可以永远使用。其次，数据库系统中存放大量数据，对于一般事务性的数据处理来讲，对这些数据要完成的操作较简单，常用的是插入、删除、检索具体记录这类操作，它只要提供一定的内部函数的计算功能就足够了。因此，一般来讲，数据库语言是将数据定义和数据操作分开，分别称为**数据定义语言**(DDL)和**数据操纵语言**(DML)，且提供交互式使用方法。对于有些应用，为某一种任务需要完成某种计算任务，给出某些决策性处理、显示、打印等做更多的事情时，可以用传统的程序设计语言书写应用程序，而对数据库的操作由 DML 来做。因此，必须使 DML 语言能够纳入到程序设计语言中去。这时把 DML 称为**数据子语言**(DSL)，而把程序设计语言称为主语言(HL)。

数据定义语言负责描述和定义数据的各种特性。这里的数据指的是用特定数据模型抽象的形式表示的数据。在数据库不同的抽象层内有不同的**模式**(Schema)表示。所以，数据定义语言是对模式完成描述和定义的语言，它不是过程语言。

例 1.3 用数据定义语言定义例 1.2 中的雇员关系 EMP：

```
CREATE TABLE EMP ((ENUMINT(10), NONULL)
        ENAME CHAR(25), DNAME CHAR(30));
CREATE INDEX ENO ON EMP(ENUM);
```

这是 SQL 语言的数据定义语言。第一个 CREATE TABLE 描述(定义)了 EMP 关系，它的属性以及他们的物理实现：INT 为整数，CHAR 为固定长字符串。第二个 CREATE INDEX 是对 EMP 关系的雇员号上建立索引。

上述第一个语句是对**概念模式**的描述，第二个语句是对**存储模式**的描述，事实上还可以有对**用户模式**(**视图**)的描述。要注意的是：

· 数据定义语言对抽象层上各模式的定义随数据模型的不同而不同；

- 当设计数据库时或修改数据库时用数据定义语言定义数据库模式，而不是用来得到或修改数据本身(这里的数据指的是值)。

数据操纵语言是对数据库中数据(值)的操作，包括：检索、插入、删除、修改。它受数据模型和数据存储结构的影响，不同的数据操纵语言在语法结构上以及风格上亦有很大差异。其总结构可概括成：M(T,P)形式。其中，M 表示操作功能；T 表示操作目标；P 表示限定条件。基于关系模型的数据操作语言较直观地为用户提供操作，使用者不必知道存取时的内部细节，完全是一种说明性的语言；基于其它一些数据模型的操纵语言只是过程性的语言。

例 1.4 实例 1.2 中所讨论的查询："查找张光的经理"，用基于关系模型的语言 SQL 书写时，可由图 1.1 表示。

```
(1) SELECT MANAGER
(2) FROM EMP, DEPT
(3) WHERE EMP. ENAME= '张光'
(4)          AND EMP. DNAME=DEPT. DNAME
```

图 1.1 SQL 查询

现简单解释一下上面的 SQL 语句。第(1)行，告诉 DBMS 查询要求；第(2)行，说明要在 EMP 和 DEPT 两个关系中查找；第(3)行，说明雇员名为'张光'；最后一行，说明经理与雇员之间的联接是用，经理在 DEPT 关系中所在的部门与雇员在 EMP 关系中所在的部门相等这一事实来表示。

图 1.2 表示了用基于网络模型的语言 DML* 的简略文本书写上述查询的语句。

```
(1) EMP. ENAME:= '张光'
(2) FIND EMP RECORD BY CALC_KEY
(3) FIND OWNER OF CURRENT EMP_DEPTSET
(4) FIND FIRST MANAGER RECORD IN CURRENT DEP_MGR SET
(5) Print MAN. NAME
```

图 1.2 DML 查询

简单解释 DML 语言的含意：(1),(2)行告诉 DBMS 在 EMP 文件中查找"张光"的记录；(3) 行说明在当前 EMP_DEPT 系结构中查找系主(雇员)的部门；(4) 行在另一个当前的 DEP_MGR 系结构中找到经理；(5) 行查找且打印出"张光所在的部门列出的第一个经理"。还有些细节在图 1.2 中已省去。同时，行(5)的操作不是 DML 语句，而是 DML 嵌入的主语言中的一部分(主语言是一般的程序设计语言如，COBOL，PASCAL…等)。语言行中的"系结构"、"系主"是网络模型的术语。

由于 DML 语言是在文件中导航，比 SQL 复杂得多。对于 DML 语言的程序设计员来说要懂得更多的专业知识，书写语句时要知道文件结构细节，说明如何从一个记录到另一个记录的导航。从语句行数相比来看，SQL 也比 DML 简单得多，SQL 程序设计员只要回

---

* 这里的 DML 并非数据操纵语言的缩写，而是 CODASYL 的 DML。

答需要什么即可。这就是描述性语言更能得到推广的原因。第四章将详细介绍 SQL 语言的细节以及其它几个基于关系的数据库语言。对于 DML 这类用于网络模型数据库系统的过程性语言，本书不准备详细介绍，仅简单提一下，以示与关系语言的对照。

当数据操纵语言由主语言引用时，一般有两种方法(这主要取决于 DBMS 的特性)：一种是直接用 CALL 语句；一种是用预处理器处理数据操纵语句。这两种方法本质上没有太大区别。但是随着数据库技术的发展，将数据操纵语言和本语言集成，确是非常重要的。集成的 DML/主语言目前也有两种方法：一种是"面向对象"的方法，利用语言具有定义抽象数据类型(abstract data type)或类(classes)的能力来实现；一种是"逻辑"方法。前者是过程性的集成，后者是说明性的集成。这两种性质的语言各有其特色。总的来说，当所有其它因素相等的情况下，用户偏爱说明性语言；但是说明性语言实现时比过程性语言更困难，系统必须耗费昂贵的优化代价才能实现；要两者兼有之似乎不大可能。总之，进一步的讨论已不属本书范围了。

事务管理

数据库管理系统的一个重要功能是管理多个用户事务在数据库上同时操作。这里先给出用户事务(简称事务)的简单定义：**一个事务即是用户对数据库的一次原子操作，即要么全部执行、要么全部不执行**。有些数据库相当庞大，允许多台计算机同时对它操作，也可能多台计算机通过计算机网络连接分布在各个地方。

例 1.5 一个银行数据库系统，它既管理一些单位的工资，又管理其它有关银行的业务。该系统不仅有多个出纳机在各个分行为储户服务，也允许银行各部门职员对其操作。设某一储户正在银行取款，却在同一时刻该储户的单位正在为他发放工资，由银行职工在为他存款。这就形成了在同一时刻一个在存款和一个在取款，即形成了多个事务并发操作的情况。

多事务操作的典型情况是：(1) 多个事务的存取相关联，如例 1.4；(2) 亦可能互不关联，如各个储户同一时刻各自存取自身在银行内的帐号上的款项。有多个事务同时操作时必须进行协调，特别在第(1)种情况下，不加以控制则会出现不正确的余额值，其后果是不堪设想的。这样，数据库管理系统必须提供多个事务并发控制机制。同时，当数据库系统运行过程发生故障时仍能恢复数据库，即有一定的恢复机制保证多个事务对数据库的存取仍如没有故障发生时一样。这两种机制都是事务管理的基本内容。在第七章将对这些内容详尽讨论。第九章还将对分布式事务管理的问题以及分布式数据库管理系统的有关技术进行叙述。

数据库保护

数据库管理系统除了必须防止故障可能引起数据丢失以外，也必须阻止不合法的存取。例如，还是例 1.4 的银行数据库系统，应保证储户只能存自己帐号下的款项；显然，该储户不准随意修改自己帐号下的余额。因此 DBMS 要维护一特权表，说明每一个用户对每一对象有什么样的特权：读文件但不允许插入或修改数据；对某个属性不能查询；既可查询又可插入、修改。保证各种特权用户存取各自专有文件、文件中某个字段或数据库中某个子集。

此外，为了把部分数据隐蔽，让用户只看到该关心的那部分数据，DBMS 还提供视图

(view)机制。视图的作用在1.3节还要讨论。

例1.6 例1.1的EMP关系有如下属性：

EMP (ENAME, DNAME, SAL, ADD)

只有特定雇员可以对SAL属性操作，一般雇员只对其它三个属性进行存取。用SQL语言可以从EMP上定义SAFE_EMP视图：

```
CREATE VIEW SAFE_EMP AS
SELECT ENAME, DNAME, ADD
FROM EMP
```

这样可把SAFE_EMP视图看作是一个如下方式表示的关系：

SAFE_EMP(ENAME, DNAME, ADD)

它不包括SAL属性，可以为一般雇员所存取。如查"张光所在的部门"可以如下书写：

```
SELECT DNAME
FROM SAFE_EMP
WHERE ENAME='张光'
```

只有特定的雇员才对EMP关系进行存取，从某种程度上讲，也起到了保护SAL属性不被任意修改的可能。所以视图是对数据库数据起到保护作用。

对数据库存取的事务在终止时也还应对事务所操作的数据的完整性约束进行检验。这些涉及保护数据库中数据的问题将在第八章讨论。

数据独立性

对多种应用提供共享服务的数据库系统，对软件和硬件应有较强的适应性。通俗地理解是，在某些用户要求有所变化时，不应涉及到数据库系统有很大改变，或者数据库系统的某些变化不会涉及应用程序有很大的变动。根本的问题是，数据库中的数据是属于多个应用程序的，如何将数据与应用程序隔离，这是数据库技术研究的重要目标。在传统的文件系统中，数据是由特定的应用所私有的，数据与应用程序完全是依赖的关系。应用的改变或数据存储方式的改变均要重新设计应用程序，其代价和时间周期均很可观。要使数据不依赖于应用或称**数据具有独立性**，实质上是由文件系统过渡(发展)到数据库系统的具体标志。实现这一目标的首要出发点是，将数据库的数据模型化，上一节的数据模型就是这一概念：**用一种抽象的方法组织数据，并用多层次体系结构构造数据库**。

在计算机中数据的具体存储以信息位(bit)或字节(byte)表示；应用或用户涉及的数据如上节例中提到的是雇员、银行帐户等具体实体。为了使这些实体为多个应用使用，在用计算机处理时，两者之间可以用多层次抽象经过映射达到统一。这样，就能屏蔽部分细节，保留主要因素，使处理简单化。为了使应用程序对数据结构和存取方法有较高的独立性，数据库系统应提供数据的逻辑结构与物理存储结构，两者之间由映射维护其一致的表示。这样，当物理存储结构改动时不必修改逻辑结构。这种独立性称为**物理数据独立性**。在当前硬件迅猛发展的情况下，存储数据的物理介质及设备不断更新换代，存储方法、存取策略也不断改善。这些涉及到数据物理存储结构的改变往往影响应用程序的效率。具有物理数据独立性，使数据库的物理存储结构随技术发展不断加以调整，且不必修改应用程序。

此外，数据库是面向特定组织的整体数据的，它包含了该组织的全部要处理的对象。所以，抽象的逻辑结构应有总体逻辑结构和用户逻辑结构之分。前者是面向所有用户的；后者则是从某一用户观点出发从总体结构中抽象出来的，且可以有独立于总体逻辑结构的表示方法。两者之间亦由映射得到统一。用户逻辑结构从总体逻辑结构中分离出来是必要的。这样，当整体数据类型有所增加或修改时，增加新的应用时虽然会引起数据库的重新构造，但不会影响其它已有的用户逻辑结构，也就不会导致对已有的应用程序的修改（或修改不多）。这种总体逻辑结构的改变也尽量不影响用户程序，也是数据库系统中数据独立性的另一层次的表示。这种数据独立性称为**逻辑数据独立性**。

数据具有逻辑独立性和物理独立性时，数据处理系统具有了数据库系统的特色。图 1.3 表示了数据具有依赖性到具有保持独立性时的系统特点。

从文件系统发展到数据库系统是信息处理领域的重大变化。在文件系统阶段，人们关注的系统功能的设计，需在每个应用系统中花费很大精力去进行，且又严重依赖于数据。在数据库系统阶段，数据已结构化、一体化，DBMS 取代了文件系统中每个应用系统要考虑的问题，大量的程序设计由专业知识很强的系统程序设计工作者去完成。这样使用者只需熟悉 DBMS 所提供的接口，即可有效地实现各种功能。

1.2 数据库在计算机学科中的地位

讨论数据库在计算机学科中的地位，主要是讨论数据库与计算机其它学科的联系，从而可以看到数据库的重要作用。

数据库与其它学科的联系如图 1.3 所示。

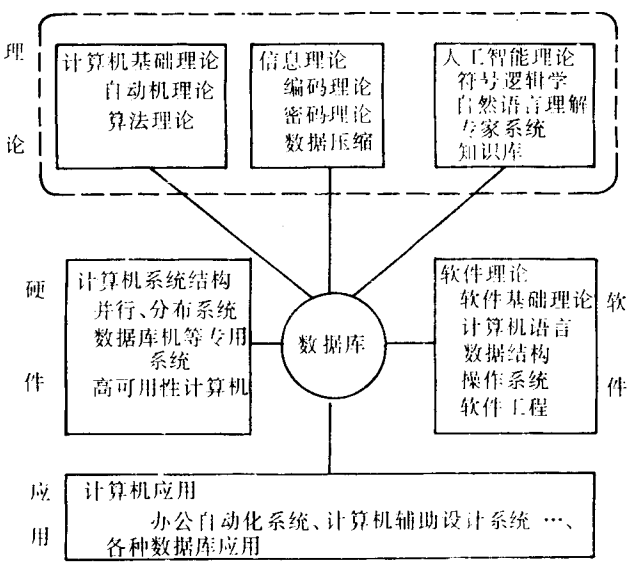

图 1.3 数据库与计算机其它学科的联系

计算机基础理论

在自动机理论中，对文字处理的部分匹配检索（对文字标题进行比较并输出等）等方面的许多行之有效的好方法对数据库系统来讲也是很适用的。此外，算法理论对计算机处理各种问题过程的复杂性的讨论，计算中时间开销和存储空间开销的评价，并针对这些问题提出相应的有效算法、并行处理理论等无疑对数据库系统是可以采用或可借鉴的。近期计算逻辑作为知识库、演译数据库的基础也更有紧密的联系。

信息处理

数据库系统本身就是信息处理的一种有效的技术的发展而来的，其中有关数据的一些相关技术如：编码理论、数据安全存储的保密处理技术、数据通信密码技术、大数据量存储时的压缩技术等，对数据库都是极有用的。

人工智能

人工智能与数据库的结合对于两者的有效应用与发展，乃至未来信息系统的建立，都是至关重要，且逐渐被大多数研究人员所接受，已各自在自己的领域内渗透。其中尤以知识表示与模拟、自然语言理解、数据库与知识库的互联等等方面。数据库技术对人工智能的最主要好处是有利于将人工智能系统规模提高到实用水平。可把数据库技术应用于知识管理、知识库设计、开发和提高知识库管理系统性能。人工智能技术对数据库技术的最大好处，体现在以数据库为基础的信息处理系统将增强表达推理能力。人工智能对数据库理论、功能、性能等方面均有贡献。

计算机系统结构

计算机性能的改善和硬件技术的进步，促进了数据库的发展，如：数据库机的研究、分布式数据库系统的研究。硬件性能的提高，存储容量的扩大和辅助存储器性能改善和容量的增加，促使数据库管理系统功能的实现更为方便。

反之，由于数据库在应用中得到广泛的推广，也带动了计算机硬件技术的发展，如：不断生产出高性能的器件，提高了系统的性能，也促进了高性能计算机体系结构的研究。

计算机软件

数据库管理系统和操作系统、高级程序设计语言都是属于计算机软件成分。它们之间的联系更多也更加直接。数据库管理系统是在操作系统支持下运行的，是操作系统数据处理的扩充；数据库语言与其它高级程序设计语言一样，都是语言，其性能和结构上本来就有相同之处，其原理与技术均可互相借鉴。对于嵌入式语言（数据操纵语言嵌入主语言）的主语言就是常用的 COBOL，PL/Ⅰ，C 等一类程序设计语言。如何使高级程序设计语言具有数据库操作功能，更需要将数据库技术与程序设计技术相结合。当前面向对象的程序设计语言，面向对象数据库的研究是极好实例。此外，数据库与数据结构联系也十分密切，数据库的物理存储方式和物理组织都实际上属于数据结构范畴，这方面技术的改进对数据库物理存储有很直接的效果，它们均会使数据库系统应用的效率有极明显的改善。在数据库设计时还需遵循软件工程方法来进行，其联系极为密切。

计算机应用领域

这是与数据库最密切的领域。正是由于应用的需要才开创了数据库的发展，也使应用不断扩展和推广，使数据库技术不断提高和完善。如办公自动化系统、决策支持系统、专家

系统等均将逐渐以数据库技术为核心。信息管理系统，包括事务管理系统、军事指挥系统、工程设计辅助计算机系统、生产制造计算机辅助工艺系统、柔性加工系统、计算机集成系统……等均离不开数据库技术。

总之，数据库系统在短短的时间里，在计算机领域中占有了重要的地位。它是计算机科学技术领域内不可分的一个部分。数据库技术的发展与计算机学科各领域的发展将互相促进，推动整个计算机事业的发展。

1.3 数据库系统的发展

数据处理以文件系统为中心，后转向用数据库技术的数据库系统，这起因于应用的需要，以及其它众多方面的进展，使数据库技术形成为一个学科领域。

纵观数据处理的历史，现代数据库系统萌芽于50年代后半期。在50年代初，数据处理进入文件处理阶段，实际上是数据库系统的前奏。1955年，W. C. McGee首先提出了源文件(Source File)概念，即可供多种目标共同利用的文件，或称公用文件，如：人事档案文件可以为不同管理所使用，行政管理、业务管理或财务管理……等。为了适合于这种处理的特点，他在IBM 702机上用磁带、磁盘技术开始对数据处理中的顺序处理、文件管理、输出表格等开发公用子程序。1959年，W. C. McGee以其工作总结提出了有历史性的论文[McGee 59]。论文阐述了通用性对数据处理的作用。使用公用子程序不仅可以减少编制程序的工作量，因而使软件费用降低，而且对于那些需要变更、扩展的系统设计过程来说，将更易于文件的编制。论文同时还提出：这种共用文件方式将减少对数据的处理和消除矛盾性。但需要具有控制功能；当包含有名目繁多的各类数据时，高效的维护和检索技术也是必要的；当然亦应有对共用文件的限制，即个人的数据安全性等问题，所以应具有权限级别的机制等问题。总之，论文提出了应给出共用文件的管理结构体系。

其时在许多应用场合也提出了关于希望使用共用数据的问题。如：美国空军在1950年开发的联机系统SAGE，提出了COMPOOL(COMmunication POOL)概念。SAGE用户用一个专门的语言JOVIAL可以从COMPOOL中存取数据，COMPOOL实际上是SAGE用户共同使用的具有集中控制的存放数据的场所。此后，用一台中央计算机的在线系统去联接多台终端的使用方式日益广泛。这些为数据库系统的出现作了准备。

从50年代后期起，有四个方面反映出数据库技术的形成过程。

1. 从COBOL源起的数据库系统的形成——CODASYL的活动

从1951年到1958年间，在数值计算方面由于开发了FORTRAN并获得成功，使人们认识到数据处理方面也应有所作为。1959年5月，美国国防部召集了有计算机厂家、用户和政府其它有关部门等的专家40余人，讨论开发事务处理通用语言，并组成了数据系统语言协会(The Conference On DAta SYstems Languages)简称CODASYL。该协会理事会下设近期、中期和长期三个委员会。其中，近期委员的任务是1959年9月应提出一个关于当时使用的事务处理软件优缺点的报告。然而，该委员会不仅完成了任务还在当年发表了一个新的语言：COBOL(Common Business Oriented Language)。1960年1月得到了CODASYL的承认，4月，美国政府出版总署公开出版该语言，即COBOL-60。随后，对其改进

和增加新功能后推出了 COBOL-61，其中融进了 McGee 1960 年在 IBM 1401 上开发的输出报告生成器 RPG(Report Program Generator)的内容。RPG 是当时较流行的一种商用系统。

1963 年 GE 公司推出了 IDS(Integrated Data Store)，这是该公司的 C. W. Bachaman 根据 McGee 的经验于 1962 年开始研制的。特别是他使用磁盘装置进行了表处理，并用于文件系统中[Bachaman 64]。可以说，这是数据库系统的先驱。为此，Bachaman 于 1973 年获得了美国计算机学会 ACM 颁发的图灵奖。

由于当时的商用系统 RPG，IDS 中增加了许多 COBOL 功能，促使 CODASYL 于 1965 年 6 月又对 IDS 进行研究，并对 COBOL 语法进行改进。为此，CODASYL 的 COBOL 协会专门成立了相应的组织，到 1967 年 5 月正式定名为数据库任务组 DBTG(Data Base Task Group)，对语言问题进行研究。1968 年发表了第一个工作报告：有数据库操作功能的 COBOL 的扩展[DBTG 68]。该报告提出了近期和长期工作目标：

近期

(1) 对 COBOL 语言追加在辅助存储器中对记录进行联接操作的功能，以适应数据处理。

(2) 引入 IDS 的主记录，即现在的记录的概念。

长期

(3) 将数据描述包含在数据文件中，因此，亦可以与应用程序分离。

(4) 对已成熟的一些典型的应用程序，对于任何级别的用户，只要适合其应用的需要给出变量即可使用。

(5) 对于文件或数据库的数据，应同时存储其存取过程，并以此过程读取数据。

(6) 应在现有操作系统具有保护文件功能的基础上，增加对数据的保护功能。

(7) 在辅助存储器上，存储管理不是由操作系统进行，而是由 COBOL 程序员进行管理。

(8) 对一个数据库，从不同的观点允许有多种角度的理解。

根据以上目标，DBTG 于 1969 年 10 月提出[DBTG 69]，由 ACM 公开发表，修改后于 1971 年 4 月提出[DBTG 71]。除完成了上述目标以外，实现了数据定义和数据操作相分离的结构，提出了存取控制和子模式等概念，使 COBOL 中含有数据库功能。

从 1971 年起在 CODASYL 之下又成立了新的组织：数据定义语言(DDL)委员会(Data Description Language Commit)，并提出了数据定义语言报告[DDL 73]，[DDL 78]。接着，DBTG 对子模式定义语言和数据操纵语言作为 COBOL 语言扩展内容作了研究并予完成，其最终文本包括在 COBOL-76 和 COBOL-78 中。

值得一提的是，IBM 公司在 DBTG 中的成员 Engles 曾在 1970 年 12 月提出了如下观点：数据库操纵语言应保证应用程序编写简单、确保数据一致性、以及应用程序与数据间的独立性等。若据这些观点，那么对[DBTG 71]中所制定的文本的许多方面必须加以改写。然而他的意见并未被 DBTG 采纳。

在数据库系统形成过程中 CODASYL 是有功绩的，它虽没有把当时世界上流行的事务处理语言完全统一，但它使它们得到了普及，且使 70 年代中期的大部分计算机系统中

均具有CODASYL方式的数据库系统。使CODASYL在数据库系统的发展中有了坚固的地位。当时也还有其它一些方式的数据库系统的出现，但没有CODASYL的影响大。人们为了区别，把CODASYL研制的数据库系统总称为CODASYL方式的数据库系统，或简称DBTG方案。

这就是有名的CODASYL DBTG方案形成的过程，随后该方案又做了不少的有益的工作。最具有DBTG方案的典型风格的商用系统是IDMS，一种网络模型的数据库系统，它可以运行在IBM 360/370，UNIVAC 7090，Siemens 4004，PDP-11/45上。成为当前数据库三大数据模型中非常有影响的一个方面。

CODASYL中期委员会和长期委员会的活动由于名字的多次更改，没有短期委员会那样"卓越"，但亦有两类重要成果：其一是1962年曾发表了Detab-X，一种表形式的语言，和信息代数(Information Algebra)[CODASYL 62]。有关信息代数的讨论直到70年代关系数据库理论兴起时才给了重新评价。其二是从1969年到1976年间曾有过三次关于数据库控制的报告，数据库控制系统的调查[System 69]、数据库控制系统功能分析[System 71]、数据库控制系统的选择[System 76]等。

2. 商用数据库管理系统的发展

上面以COBOL起源形成的数据库系统形成中提到过WcGee的RPG。其实，它是从1959年McGee在IBM 709机上开发的SHARE 9 PAC过渡的。RPG的功能被COBOL-60所包含后就失去了一定的使用价值。但是，它起到了先导作用。GE公司的IDS商用系统就是以McGee的9 PAC的经验中引伸的。除此以外，Postley也引用了RPG的经验在Informatics公司将IBM 7090上的GIRLS商品化为MARK Ⅳ。到1968年，在IBM System/360上运行的MARK Ⅳ，虽然不是数据库系统，却可以称作具有了现代数据库的特色。

其它商用产品有MITRE公司接受美国空军的委托开发了实验系统ADAM(Advanced Data Management System)；SCD公司开发了一个TDMS(Time-Shared Data Management System)，作为商品出售时称为CDMS。这些均是1966年的产品。这些商品形成了几乎完全是倒排方式的联机数据库系统。

得克萨斯大学计算中心以TDMS开发了远程文件管理系统RFMS(Remote File Management System)，由MRI公司(是该大学的协作单位)将其商品化推出，并于1969年正式出售。第1版完全倒排方式，1972年以后又有了层次方式与倒排方式相结合的版本。

特别值得提出的是Cincom System公司1968年开发的TOTAL系统，由于采用了较好的磁盘读取技术和完全适合于CODASYL方法，深受用户欢迎，很快在各种计算机上运行。

与此同时，IBM公司却迟迟没有对此项问题进行研究，直至1965年才开始提出了GIS(Generalized Information System)。后又于1969年发表了IMS(Information Management System)，它是一个DB/DC(Data Base/Data Communication)的庞大系统。IBM与洛克菲尔公司共同开发提出了基于自己的数据库模型：一种层次型数据结构为基础的数据模型及其相适应的语言DL/Ⅰ(Data Language/Ⅰ)。产品有极大的竞争力。人们把IMS作为当前流行的三大数据库模型之一——层次模型数据库的代表。

3. 关系方法的发展

1970 年 6 月 IBM 公司 San Jose 研究所的 E. F. Codd 在美国计算机学会会刊“Communication of the ACM”上发表了题为：A Relational Moded of Data for Shared Data Banks 的著名论文，为关系模型的讨论奠定了基础，并将数据库系统的研究推向新的历史时期，即只以应用为主转向理论研究发展的新时期。

关系模型的先驱或许可追索到 1962 年 CODASYL 发表的 Information Algebra，这个独立于计算机的数据处理理论基础由于当时缺乏实验条件未被重视，Codd 的论文发表，无疑也引起人们对它又引起关注；其次可追索的是 60 年代后期，人工智能研究的以二元关系为中心的实验系统提出的模型，经实验后提出关系数据文件[Levien 67]，LEAP 语言和数据结构[Rovner 68]以及 IBM 公司内部的一个研究报告：信息检索的关系模型，也不无其影响；1968 年 D. L. Childs 关于集合论数据结构 STDS(Set-Theoretic Data Struction) 上的 n 元关系的研究[Childs 68a，Childs 68b]，在 IBM 7090 机上的实现。这些对 Codd 的论文都起到了背景作用。

继 1971 年后，Codd 又继续发表了多篇关于关系模型的文章，把数学方法引入到数据库系统，使数据库系统有了坚实的理论基础。Codd 本人作为关系数据库的创始人和奠基人，获得 1981 年 ACM 图灵奖。

如上所述，自 Codd 提出关系模型数据库方法以后，1974 年 IBM 就提出了 SQL (Structured Query Language)语言，这是一种基于关系方法实现对数据库存取的语言。大约从 1975—1979 的五年期间，IBM 公司 San Jose 实验室成功地研制了一个实现 SQL 语言的关系数据库系统原型 System R(在 IBM 370 上运行)。该系统中所采用的技术被引入 IBM 公司第一个完全支持关系方法的产品 SQL/DS 上，于 1981 年宣布配置在 DOS/VSE 上，1983 年配置在 VM/CMS 上运行。之后，又引入到 DB2(IBM Database 2)中，配置在 MVS 上运行。IBM 83 年 6 月推出 DB 2 产品，表明了关系模型数据库系统终于被人们所接受，并得到最终的肯定。

其它有影响的关系模型数据库系统有 1973—1975 年由美国伯克利加州大学研制的 INGRES(Interactive Graphics and Retrieval System)可运行在 UNIX 操作系统支持下的 DEC PDP-11 系列上。经几年完善后由美国关系技术公司作为产品推出。INGRES 在实现关系模型时采用了许多新颖的、有效的技术，如：数据独立性、并发恢复事务管理、安全性、完整性控制等，对以后研制关系模型数据库管理系统起到了穿针引线的作用。当今，更为流行和有影响的关系数据库管理系统还有 ORACLE 公司的 ORACLE。从 1977 年开始该公司就准备实现 Codd 的思想，准备研制横跨在大范围计算机上可运行的数据库系统，它采用 SQL 语言，用 C 语言开发，使 ORACLE 不仅使用方便、速度快且移植性好。1979 年开始推出 ORACLE 第一版，它为由核心和一组支持软件组成的软件包。从开始研制并经近 10 年的改进，到 1986 年推出第 5 版，增加了与各种语言、决策支持工具以及网络通信软件的接口，形成了开放式体系结构，是当前唯一可以通用于大、中、小和微型机上的产品，为不同类型计算机提供整体化和标准化软件环境，这是 ORACLE 的最大特点。

关系数据库系统的出现，促进了数据库的小型化和普及化，使得在微型机上配置数据库系统成为可能。较流行的 dBASE-Ⅱ(由 Ashton-Tate 公司于 1981 年推出)可在 8 位微

型机上运行，经过几年完善已形成 dBASE-Ⅲ、Ⅲplus、Ⅳ等系列，可分别运行于各类微机和高档微机上。有关微机上可运行的产品在此不一一叙述了。

4. 标准化和学术活动的影响

1969—1971 年期间，CODASYL 提案发表后，对数据库系统标准化问题引起很多讨论。1972 年美国标准化协会计算机信息处理部(ANSI/X3)下设的标准化计划委员会(SPARC)，成立了数据库控制系统研究班，分别于 1975 年和 1978 年发表了两个报告[ANSI/X3/SPARC 75]，[ANSI/X3/SPARC 78]。这些报告研究了当时数据库系统的现状，从整个信息系统的接口观点出发进行分析，对数据库系统开发和研究有很深影响。其中提出的数据库结构标准化建议，将数据库划分为三层：用户层(外层)、概念层、内层(存储层)已被人们所接受。CODASYL DDLC 也于 1978 年对三层数据库结构提出相应的规范。

从 1978 年开始对数据库语言的标准规范也着手标准化工作，由于关系数据库系统继 System R 以后，先后又开发了 SQL/DS 和 DB 2，使 SQL 语言深受用户喜爱。1986 年 10 月 6 日 ANSI 宣布 SQL 为关系数据库系统的标准化语言。

结构的标准化和语言的标准化无疑使数据库技术的发展有着积极的意义。

数据库技术的迅猛发展与学会、学术活动的研究和推动有密切关系。较有影响的是 ACM 下设的 SIGMOD(数据管理研究会)，自 1970 年起以每年一次召开关系数据库系统原理(PODS)学术会议，几乎成为研究关系方法的中心场地。此外，由 VLDB 董事会组织的国际性大型数据库会议 VLDB 学术会议，自 1975 年开始也是每年一次。这些学术活动对数据库技术的理论研究和系统实现都起到了不可估量的作用。进入 80 年代中、后期以来，随着数据库技术推向新的应用领域，各种学术会议更是风起云涌。

我国数据库技术的发展大致始于 70 年代中、后期，其标志是 1977 年 11 月在安徽省黄山召开的第一次数据库技术研讨会。这次会议是国家计委支持下由中国科技大学主办。随后中国计算机学会重新恢复活动后，自 1982 年起几乎每年由学会下设的数据库学组的领导下，召开一次全国性的学术交流。会议起到了交流我国数据库应用与有关科研成果，指导我国数据库学术发展的作用。经过十多年的努力，我国数据库技术水平从初期的学习、理解、试探阶段发展到消化、改造、创新阶段。推动我国数据库技术发展的一个重要因素是 1978 年以来不少学者或学生赴国外访问和学习，参加国际学术活动，他们吸收了国外数据库的新理论和新技术，并参与开发数据库软件，这就大大地缩短了我国与先进国家在这方面的差距。不过，差距仍毕竟存在，我们必须更加努力寻求发展我国数据库技术的合适的途径，将我国的数据库技术推向新水平。

第二章　数据库系统的数据模型

人类对错综复杂的现实世界进行某些特定问题研究时，常常需要将其抽象，提取主要因素，略去次要成分，经过归纳形成一个较清晰的轮廓进行具体研究。这样，就使复杂问题简化从而易于处理。这种方法就是基于了“模型”的方法。

在自然界中信息是事物运动状态的表现，普遍地存在于自然界、存在于人类社会和思维领域。信息处理随着人类社会的发展而日益发展，已成为极其重要的问题。对信息的研究已成为一特定问题的研究。信息须用一定方式如：文字、数字、图形、声音、……等具体表现出来。这时，信息又转化为一种具体形式或称为数据的形式来表示。所以，严格地讲信息和数据是有区别的：信息是抽象的，它不随具体的物理载体而变；而数据是具体的，它是通过种种物理载体记录下来的。如：某学术团体准备召开学术会议研究某一方面的问题。这一事件形成了“开学术会议”这样一个信息。把这个信息通知有关方面时，可以用广播、通过声音这一具体形式；或用文件、以文字形式向有关方面传送。“开学术会议”这一信息是从这两种不同形式的数据解释后得到的。尽管数据形式不同，但“开学术会议”这一信息的内容没有变。因而可以说信息是数据的内涵。这是说信息与数据是不同的。在人类社会中，信息不仅是一种消息，还有更广的含义。人们在生活中通过实践在自然界、社会和思维领域取得必要信息，对这些信息的分析处理、归纳推理，有了理性认识并产生相应的行动，改造世界。然后，把这种改造和效果作为新的补充信息反馈回来，重新进行分析处理、归纳和理性思维，检验是否需要调整原有认识，修正行动，直至达到某种改造的目的。如此往复循环，不断前进。也就是说，信息被以某种数据形式进行处理而得到解释以后，才可能弄清楚数据的内在意义，也就是又获得了信息（某种意义上讲信息即是一种知识）。前一次处理获得的信息可以作为下一次处理的数据，如图 2.1 所示，信息与数据之间是递归的。也就是说从处理这个角度来看信息与数据时，其区别有时就忽略了。

图 2.1　信息处理

当今世界中所谓信息处理是指对信息（各种形式的数据）进行收集、储存、加工、传播等这一系列活动的总和。其基本目的是要从大量的、杂乱无章的、难以理解的数据中寻找规律，抽取并推导出对于某些特定情况下是有价值、有意义的数据，籍以作为管理或决策使用。当前，计算机已经被人们认为是信息处理最有效的手段和工具，它完全可以实现图

2.1 所示的处理要求。

在信息处理领域内，处理的对象是大量的数据，所采用的模型通常称为数据模型。信息处理中涉及数据处理方式和数据管理方式。在用计算机实现数据处理的过程中，已发展到了基于数据库技术的数据管理方式。本书讨论的数据模型，是讨论信息在数据库系统的形式结构中怎样表示和处理。我们可把数据模型看作一种形式化描述数据与数据之间的联系的一种规则。

由于数据库系统本身包含了数据库与数据库管理系统两个主要的组成部分。因而，数据模型的研究也主要是环绕这两个方面。一类数据模型主要描述数据库中数据的**概念结构**，不涉及（或很少涉及）数据操作及存储，也就是说独立于具体的数据库管理系统的性能。这类数据模型有利于数据库的设计，是更抽象的描述。如实体_关联模型（E_R 模型）、语义模型等。另一类数据模型不仅描述数据库中数据的结构，还包括对数据的操作以及操作后的数据完整性的问题。数据库管理系统的设计完全依赖于这类数据模型。如当前流行的关系数据模型、网络数据模型、层次数据模型等。

本章讨论后一类数据模型，重点是关系数据模型的主要内容，然后简单介绍数据模型研究的动向。

2.1 数据模型

数据库系统试图管理现实世界中错综复杂的联系必须有一个抽象的过程。**数据模型**是以一种数学形式，将形形色色的、千变万化的事物抽象成计算机可以表示的形式。

现实世界中事物千态百姿，但它们总能相互区别，主要的方法是通过事物的**特征**。如人的特征是：姓名、性别、年龄、藉贯等；树的特征是：树学名、种类、生长期……等。我们可以选择某些特征来识别各事物间的区别。其次，事物是息息相关的，如人种树，人取树成材制造人所需的物品等。表示事物之间是有联系的。所以，首先将其抽象为数据库的反映的信息结构：**实体**（Entity）、**属性**（Attribute）和**关联**（Relationship）

实体

实体没有形式化的定义，而是根据给定的性质所定义。也就是说：**实体是客观存在的且可以区别的事物**。即可从另一个实体中识别出一个实体。例如：人是实体、树是实体。此外，还可以把艺术、思维等抽象的概念也称为实体，只要能把它们区分开。

属性

描述事物特性的抽象称为属性。如上述人的特征可以理解为人的属性。也就是属性刻划实体的特性。一个实体可以由多个属性刻划，也就是说实体可以用属性集表示。例 1.1 的雇员实体的属性有：雇员号、雇员名、雇员所在部门，……工资、家庭地址。

属性有**值域**，通常域是整数的集合，实数的集合、或字符的集合……等。如雇员号是 10 位整数，雇员名是字符其长为 20，……等。每个属性可从其域中取一定的**值**，如“张光”就是雇员名这个属性的值。所以可以理解为**值域是属性的值的变化范围**。

实体集

所有性质相同的一类实体组成实体集（Entity Sets）。从属性描述实体性质，属性有值