第 1 章　数据库技术概论

- 了解：数据与数据处理的概念、数据库技术的产生背景与发展概况、SQL Server 的特点、常用管理工具、T-SQL 语言的功能。
- 理解：数据库系统的组成与特点、数据独立性的概念、数据模型的概念。
- 掌握：关系模型的基本知识、关系数据库的设计方法、SQL Server 的数据类型及各种运算符和语句。

1.1　数据库技术的产生与发展

数据库技术是一门研究如何存储、使用和管理数据的技术，是计算机数据管理技术的最新发展阶段，它能把大量的数据按照一定的结构存储起来，在数据库管理系统的集中管理下实现数据共享。

人类在长期的社会生产实践中会产生大量数据，如何对数据进行分类、组织、存储、检索和维护成为迫切的实际需要，在计算机成为数据处理的工具之后，数据处理现代化成为可能。数据库系统的核心任务是数据管理，但并不是一开始就有数据库技术，它是随着数据管理技术的不断发展而逐步产生与发展的。

1. 人工管理阶段

20 世纪 50 年代中期以前，计算机主要应用于科学计算，虽然也有数据管理的问题，但这时的数据管理是以人工管理方式进行的。在硬件方面，外存储器只有磁带、卡片和纸带等，没有磁盘等可以直接存取的外存储器。在软件方面，只有汇编语言，没有操作系统，也没有对数据进行管理的软件，数据处理方式基本上是批处理。在此阶段，数据管理的特点如下：

（1）数据不保存。此阶段处理的数据量较少，一般不需要将数据长期保存，只是在计算时将数据随程序一起输入，计算完后将结果输出，而数据和程序则一起从内存中被释放。若再计算，则需重新输入数据和程序。

（2）由应用程序管理数据。由于系统没有专门的软件对数据进行管理，数据需要由应用程序自行管理。每个应用程序不仅要规定数据的逻辑结构，而且要设计数据的存储结构及输入/输出方法等，程序设计任务繁重。

（3）数据有冗余，无法实现共享。程序与数据是一个整体，一个程序中的数据无法被其他程序使用，因此程序与程序之间存在大量的重复数据，数据无法实现共享。

（4）数据对程序不具有独立性。由于程序对数据的依赖性，数据的逻辑结构或存储结构一旦有所改变，则必须修改相应的程序，这就进一步加重了程序设计的负担。

2. 文件管理阶段

20 世纪 50 年代后期至 60 年代中期，计算机开始大量用于数据管理。硬件上出现了可以直接存取的大容量外存储器，如磁盘、磁鼓等，这为计算机数据管理提供了物质基础。软件上出现了高级语言和操作系统。操作系统中的文件系统专门用于管理数据，这又为数据管理提供了技术支持。数据处理方式不仅有批处理，而且有联机实时处理。

数据处理应用程序利用操作系统的文件管理功能将相关数据按一定的规则构成文件，通过文件系统对文件中的数据进行存取和管理，实现数据的文件管理方式，其特点如下：

（1）数据可以长期保存。文件系统在程序和数据之间提供了一个公共接口，使应用程序采用统一的存取方法来存取和操作数据。数据可以组织成文件，能够长期保存、反复使用。

（2）数据对程序有一定的独立性。程序和数据不再是一个整体，而是通过文件系统把数据组织成一个独立的数据文件，由文件系统对数据的存取进行管理，程序员只需通过文件名来访问数据文件，不必过多考虑数据的物理存储细节，因此程序员可集中精力进行算法设计，从而大大减少了程序维护的工作量。

文件管理使计算机在数据管理方面有了长足的进步。时至今日，文件系统仍是一般高级语言普遍采用的数据管理方式。

3. 数据库管理阶段

20 世纪 60 年代后期，计算机用于数据管理的规模更加庞大，数据量急剧增加，数据共享的需求也更加强烈。同时，计算机硬件价格下降，软件价格上升，编制和维护软件所需成本相对增加，其中维护成本更高，这些都成为数据管理技术在文件管理的基础上发展到数据库管理的原动力。

数据库（Database，DB）是按照一定的组织方式存储起来的、相互关联的数据集合。在数据库管理阶段，由一种叫作数据库管理系统（Database Management System，DBMS）的系统软件来对数据进行统一的控制和管理，把所有应用程序中使用的相关数据汇集起来，按照统一的数据模型存储在数据库中，为各个应用程序所使用。在应用程序和数据库之间保持较高的独立性，数据具有完整性、一致性和安全性高等特点，并且具有充分的共享性，有效地减少了数据冗余。

4. 新型数据库系统

数据库技术的发展先后经历了层次数据库、网状数据库和关系数据库三个阶段。层次数据库和网状数据库可以看作是第一代数据库系统，关系数据库可以看作是第二代数据库系统。自 20 世纪 70 年代提出关系数据模型和关系数据库后，数据库技术得到了蓬勃发展，应用也越来越广泛。但随着应用的不断深入，占主导地位的关系数据库系统已不能满足新应用领域的需求。例如，在实际应用中，除了需要处理数字、字符数据的简单应用外，还需要存储并检索复杂的复合数据（如集合、数组、结构）、多媒体数据、计算机辅助设计绘制的工程图纸和地理信息系统（Geographic Information Systems，GIS）提供的空间数据等。对于这些复杂数据，关系数据库无法实现对它们的管理。正是这些实际应用中涌现出的问题促使数据库技术不断向前发展，出现了许多不同类型的新型数据库系统，下面就进行简要介绍。

（1）分布式数据库系统。分布式数据库系统（Distributed Database System，DDBS）是数据库技术与计算机网络技术、分布式处理技术相结合的产物。分布式数据库系统是系统中的数据地理上分布在计算机网络的不同节点，但逻辑上属于一个整体的数据库系统，它不同于将数

据存储在服务器上供用户共享存取的网络数据库系统，分布式数据库系统不仅能支持局部应用（访问本地数据库），而且能支持全局应用（访问异地数据库）。分布式数据库系统主要应用于航空、铁路、旅游订票系统，银行通存通兑系统，水陆空联运系统，跨国公司管理系统和连锁配送管理系统等。

（2）面向对象数据库系统。面向对象数据库系统（Object-Oriented Database System，OODBS）是将面向对象的模型、方法和机制与先进的数据库技术有机结合而形成的新型数据库系统。它从关系模型中脱离出来，强调在数据库框架中发展类型、数据抽象、继承和持久性。它的基本设计思想是：把面向对象语言向数据库方向扩展，使应用程序能够存取并处理对象；扩展数据库系统，使其具有面向对象的特征，提供一种综合的语义数据建模概念集，以便对现实世界中复杂应用的实体和联系建模。因此，面向对象数据库系统首先是一个数据库系统，具备数据库系统的基本功能；其次是一个面向对象的系统，针对面向对象程序设计语言的永久性对象存储管理而设计，充分支持完整的面向对象概念和机制。对一些特定应用领域（如 CAD 等），面向对象数据库系统能较好地满足其应用需求。

（3）多媒体数据库系统。多媒体数据库系统（Multimedia Database System，MDBS）是数据库技术与多媒体技术相结合的产物。随着信息技术的发展，数据库应用从传统的企业信息管理扩展到计算机辅助设计（CAD）、计算机辅助制造（CAM）、办公自动化（OA）、人工智能（AI）等多个应用领域。这些领域中处理的数据不仅包括传统的数字、字符等格式化数据，还包括大量多媒体形式的非格式化数据，如图形、图像、声音等。这种能存储和管理多媒体数据的数据库称为多媒体数据库。多媒体数据库系统主要应用于军事、医学病例管理、航天测控、商标管理、地理信息、数字图书馆和期刊出版系统等。

（4）数据仓库技术。随着信息技术的高速发展，数据库应用的规模、范围和深度不断扩大，一般的事务处理已不能满足应用的需要，企业需要在大量数据基础上的决策支持，数据仓库（Data Warehouse，DW）技术的兴起满足了这一需求。数据仓库作为决策支持系统（Decision Support System，DSS）的有效解决方案，涉及 3 个方面的技术内容：数据仓库技术、联机分析处理（On-Line Analytical Processing，OLAP）技术和数据挖掘（Data Mining，DM）技术。

数据仓库、联机分析处理和数据挖掘是作为 3 种独立的数据处理技术出现的。数据仓库用于数据的存储和组织，联机分析处理集中于数据的分析，数据挖掘则致力于知识的自动发现。它们都可以分别应用到信息系统的设计和实现中，以提高相应部分的处理能力。但是，由于这 3 种技术内在的联系性和互补性，将它们结合起来即是一种新的 DSS 架构。这一架构以数据库中的大量数据为基础，系统则由数据驱动。数据仓库技术应用遍及通信、零售业、金融和制造业等领域。

（5）内存数据库系统。内存数据库（Main Memory Database，MMDB）系统是实时系统和数据库系统的有机结合。它抛弃了磁盘数据管理的传统方式，基于全部数据都在内存中这一前提重新设计了体系结构，并且在数据缓存、快速算法、并行操作方面进行了相应的改进，所以数据处理速度比传统数据库的数据处理速度要快很多，一般都在 10 倍以上。内存数据库的最大特点是其“主拷贝”或“工作版本”常驻内存，即活动事务只与实时内存数据库的内存拷贝打交道。内存数据库系统目前广泛应用于航空、军事、电信、电力及工业控制等领域。

1.2 数据库系统

数据库系统（Database System，DBS）是指基于数据库的计算机应用系统。和一般的应用系统相比，数据库系统有其自身的特点，它涉及一些既相互联系又有区别的基本概念。

1.2.1 数据库系统的组成


数据库系统组成

数据库系统是一个计算机应用系统，它是把有关计算机硬件、软件、数据和人员组合起来为用户提供信息服务的系统。因此，数据库系统是由计算机系统、数据库及其描述机制、数据库管理系统和有关人员组成的具有高度组织性的整体。

1. 计算机硬件

计算机硬件系统是数据库系统的物质基础，是存储数据库及运行数据库管理系统的硬件资源，主要包括计算机主机、存储设备、输入/输出设备及计算机网络环境。

2. 计算机软件

数据库系统中的软件包括操作系统、数据库管理系统、数据库应用系统等。

数据库管理系统是数据库系统的核心软件之一，它提供数据定义、数据操纵、数据库管理、数据库建立和维护及通信等功能。数据库管理系统提供对数据库中的数据资源进行统一管理和控制的功能，将用户、应用程序与数据库数据相互隔离，是数据库系统的核心，其功能的强弱是衡量数据库系统性能优劣的主要指标。数据库管理系统必须运行在相应的系统平台上，有操作系统和相关系统软件的支持。

数据库管理系统功能的强弱随系统而异，大系统功能较强、较全，小系统功能较弱、较少。目前较流行的数据库管理系统有 Access、MySQL、SQL Server、Oracle 和 Sybase 等。

数据库应用系统是指系统开发人员利用数据库系统资源开发出来的、面向某一类实际应用的应用软件系统。从实现技术角度而言，它是以数据库技术为基础的计算机应用系统。

3. 数据库

数据库是指数据库系统中按照一定的方式组织的、存储在外部存储设备上的、能为多个用户共享的、与应用程序相互独立的相关数据集合。它不仅包括描述事物的数据本身，而且还包括相关事物之间的联系。

数据库中的数据往往不像文件系统那样只面向某一项特定应用，而是面向多种应用，可以被多个用户、多个应用程序共享。其数据结构独立于使用数据的程序，对数据的增加、删除、修改和检索由数据库管理系统进行统一管理和控制，用户对数据库进行的各种操作都是由数据库管理系统实现的。

4. 数据库系统的有关人员

数据库系统的有关人员主要有 3 类：最终用户、数据库应用系统开发人员和数据库管理员（Database Administrator，DBA）。最终用户指通过应用系统的用户界面使用数据库的人员，他们一般对数据库知识了解不多。数据库应用系统开发人员包括系统分析员、系统设计员和程序员。系统分析员负责应用系统的分析，他们和用户、数据库管理员相配合，参与系统分析；系统设计员负责应用系统设计和数据库设计；程序员则根据设计要求进行编码。数据库管理员是数据管理机构的一组人员，他们负责对整个数据库系统进行总体控制和维护，以保证数据库

系统的正常运行。

综上所述，数据库中包含的数据是存储在存储介质上的数据文件的集合；每个用户均可使用其中的数据，不同用户使用的数据可以重叠，同一组数据可以为多个用户共享；数据库管理系统为用户提供对数据的存储组织、操作管理等功能；用户通过数据库管理系统和应用程序实现对数据库系统的操作与应用。

1.2.2　数据库的结构体系

数据库系统的三级模式

为了有效地组织、管理数据，提高数据库的逻辑独立性和物理独立性，人们为数据库设计了一个严谨的结构体系，数据库领域公认的标准结构是三级模式及二级映射。三级模式包括外模式、概念模式和内模式；二级映射是概念模式/内模式的映射和外模式/概念模式的映射。这种三级模式与二级映射结构构成了数据库的结构体系，如图 1.1 所示。

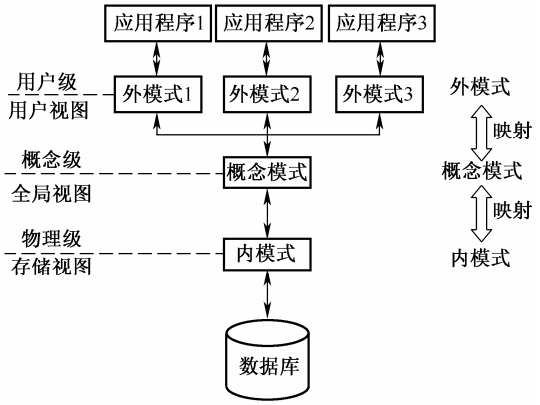

图 1.1　数据库的三级模式与二级映射

1. 数据库的三级模式

美国国家标准协会（American National Standards Institute，ANSI）的数据库管理系统研究小组于 1978 年提出了标准化的建议，将数据库结构体系分为三级：面向用户或应用程序员的用户级、面向建立和维护数据库人员的概念级、面向系统程序员的物理级。用户级对应外模式，概念级对应概念模式，物理级对应内模式，使不同级别的用户对数据库形成不同的视图。视图是指观察、认识和理解数据的范围、角度和方法，是数据库在用户眼中的反映。很显然，不同层次（级别）用户所看到的数据库是不相同的。

（1）概念模式。概念模式又称逻辑模式，或简称模式，对应于概念级。它是由数据库设计者综合所有用户的数据，按照统一的观点构造的全局逻辑结构，是对数据库中全部数据的逻辑结构和特征的总体描述，是所有用户的公共数据视图（全局视图）。它由数据库系统提供的数据定义语言（Data Definition Language，DDL）来描述和定义，体现并反映了数据库系统的整体观。

（2）外模式。外模式又称子模式或用户模式，对应于用户级。它是某个或某几个用户所看到的数据库的数据视图，是与某一应用有关的数据的逻辑表示。外模式是从概念模式导出的一个子集，包含概念模式中允许特定用户使用的那部分数据。用户可以通过外模式定义语言（外模式 DDL）来描述、定义对应于用户的数据记录（用户视图），也可以利用数据操纵语言（Data

Manipulation Language，DML）对这些数据记录进行操作。外模式反映了数据库的用户观。

（3）内模式。内模式又称存储模式或物理模式，对应于物理级。它是数据库中全体数据的内部表示或底层描述，是数据库最低一级的逻辑描述，它描述了数据在存储介质上的存储方式和物理结构，对应着实际存储在外存储介质上的数据库。内模式由内模式定义语言（内模式DDL）来描述和定义，它体现了数据库的存储观。

在一个数据库系统中只有唯一的数据库，因而作为定义、描述数据库存储结构的内模式和定义、描述数据库逻辑结构的模式也是唯一的，但建立在数据库系统之上的应用则是非常广泛多样的，所以对应的外模式不是唯一的，也不可能唯一。

2. 三级模式间的二级映射

数据库的三级模式是数据在 3 个级别（层次）上的抽象，使用户能够逻辑地、抽象地处理数据，而不必关心数据在计算机中的物理表示和存储方式，把数据的具体组织交给数据库管理系统去完成。为了实现这 3 个抽象级别的联系和转换，数据库管理系统在三级模式之间提供了二级映射，正是这二级映射保证了数据库中的数据具有较高的物理独立性和逻辑独立性。

（1）概念模式/内模式的映射。数据库中的概念模式和内模式都只有一个，所以概念模式/内模式的映射是唯一的。它确定了数据的全局逻辑结构与存储结构之间的对应关系。存储结构变化时，概念模式/内模式的映射也应有相应的变化，使其概念模式仍保持不变，即把存储结构变化的影响限制在概念模式之下，这使数据的存储结构和存储方法独立于应用程序，通过映射功能保证数据存储结构的变化不影响数据的全局逻辑结构的改变，从而不必修改应用程序，即确保了数据的物理独立性。

（2）外模式/概念模式的映射。数据库中的同一概念模式可以有多个外模式，对于每一个外模式，都存在一个外模式/概念模式的映射，用于定义该外模式和概念模式之间的对应关系。当概念模式发生改变时，如增加新的属性或改变属性的数据类型等，只需对外模式/概念模式的映射进行相应的修改，而外模式（即数据的局部逻辑结构）保持不变。由于应用程序是依据数据的局部逻辑结构编写的，所以应用程序不必修改，从而保证了数据与程序间的逻辑独立性。

1.2.3 数据库系统的特点

数据库系统的出现是计算机数据管理技术的重大进步，它克服了文件系统的缺陷，提供了对数据更高级、更有效的管理。

1. 数据结构化

在文件系统中，文件的记录内部是有结构的。例如，学生数据文件的每个记录是由学号、姓名、性别、出生年月、籍贯、简历等数据项组成的。但这种结构只适用于特定的应用，对其他应用并不适用。

在数据库系统中，每一个数据库都是为某一应用领域服务的。例如，学校信息管理涉及多个方面的应用，包括对学生的学籍管理、课程管理、学生成绩管理等，还包括教工的人事管理、教学管理、科研管理、住房管理和工资管理等，这些应用彼此之间都有着密切的联系。因此，在数据库系统中不仅要考虑某个应用的数据结构，还要考虑整个组织（即多个应用）的数据结构。这种数据组织方式使数据结构化了，这就要求在描述数据时不仅要描述数据本身，还要描述数据之间的联系。而在文件系统中，尽管其记录内部已有了某些结构，但记录之间没有联系。数据库系统实现整体数据的结构化，这是数据库的主要特点之一，也是数据库系统与文

件系统的本质区别。

2. 数据共享性高、冗余度低

数据共享是指多个用户或应用程序可以访问同一个数据库中的数据，而且数据库管理系统提供并发和协调机制，保证在多个应用程序同时访问、存取和操作数据库数据时不产生任何冲突，从而保证数据不遭到破坏。

数据冗余既浪费存储空间，又容易产生数据的不一致。在文件系统中，由于每个应用程序都有自己的数据文件，所以存在着大量重复的数据。

数据库从全局观念来组织和存储数据，数据已经根据特定的数据模型结构化。在数据库中，用户的逻辑数据文件和具体的物理数据文件不必一一对应，从而有效地节省了存储资源，减少了数据冗余，保证了数据的一致性。

3. 具有较高的数据独立性

数据独立性是指应用程序与数据库的数据结构之间相互独立。在数据库系统中，因为采用了数据库的三级模式结构，从而保证了数据库中数据的独立性。在数据存储结构改变时，不影响数据的全局逻辑结构，这样保证了数据的物理独立性。在全局逻辑结构改变时，不影响用户的局部逻辑结构和应用程序，这样就保证了数据的逻辑独立性。

4. 具有统一的数据控制功能

在数据库系统中，数据由数据库管理系统进行统一控制和管理。数据库管理系统提供了一套有效的数据控制手段，包括数据库安全性控制、数据库完整性控制、数据库的并发控制和数据库的恢复等，增强了多用户环境下数据库的安全性和一致性保护。

1.3　数据模型

数据库是现实世界中某种应用环境（一个单位或部门）所涉及的数据集合，它不仅要反映数据本身的内容，而且要反映数据之间的联系。由于计算机不能直接处理现实世界中的具体事物，所以必须将这些具体事物转换成计算机能够处理的数据。在数据库技术中，用数据模型（Data Model）来对现实世界中的数据进行抽象和表示。

1.3.1　数据模型的组成要素

一般而言，数据模型是一种形式化描述数据、数据之间的联系以及有关语义约束规则的方法，这些规则分为 3 个方面：描述实体静态特征的数据结构、描述实体动态特征的数据操作规则和描述实体语义要求的数据完整性约束规则。因此，数据结构、数据操作及数据的完整性约束也被称为数据模型的 3 个组成要素。

1. 数据结构

数据结构研究数据之间的组织形式（数据的逻辑结构）、数据的存储形式（数据的物理结构）、数据对象的类型等。存储在数据库中的对象类型的集合是数据库的组成部分。例如在教学管理系统中，要管理的数据对象有学生、课程、选课成绩等；在课程对象集合中，每门课程包括课程号、课程名、学分等信息，这些基本信息描述了每门课程的特性，构成在数据库中存储的框架，即对象类型。

数据结构用于描述系统的静态特性，是刻画一个数据模型性质最重要的方面。因此，在

数据库系统中，通常按照数据结构的类型来命名数据模型，如层次结构、网状结构和关系结构的数据模型分别命名为层次模型、网状模型和关系模型。

2. 数据操作

数据操作用于描述系统的动态特性，是指对数据库中的各种数据所允许执行的操作的集合，包括操作及有关的操作规则。数据库主要有查询和更新（包括插入、删除和修改等）两大类操作。数据模型必须定义这些操作的确切含义、操作符号、操作规则（如优先级）、实现操作的语言等。

3. 数据的完整性约束

数据的完整性约束是一组完整性规则的集合。完整性规则是给定的数据模型中数据及其联系所具有的约束和依存规则，用以限定符合数据模型的数据库状态和状态的变化，以保证数据的正确、有效和相容。

数据模型应该反映和规定数据必须遵守的、基本的、通用的完整性约束。此外，数据模型还应该提供定义完整性约束条件的机制，以反映具体涉及的数据必须遵守的、特定的语义约束条件，如学生信息中的“性别”只能为“男”或“女”，学生选课信息中的“课程号”的值必须取自学校已开设课程的课程号等。

1.3.2 数据抽象的过程

从现实世界中的客观事物到数据库中存储的数据是一个逐步抽象的过程，这个过程经历了现实世界、观念世界和机器世界 3 个阶段，对应于数据抽象的不同阶段采用不同的数据模型。首先将现实世界的事物及其联系抽象成观念世界的概念模型，然后再转换成机器世界的数据模型。概念模型并不依赖于具体的计算机系统，它不是数据库管理系统所支持的数据模型，它是现实世界中客观事物的抽象表示。概念模型经过转换成为计算机上某一数据库管理系统支持的逻辑数据模型。所以说，数据模型是对现实世界进行抽象和转换的结果，这一过程如图 1.2 所示。

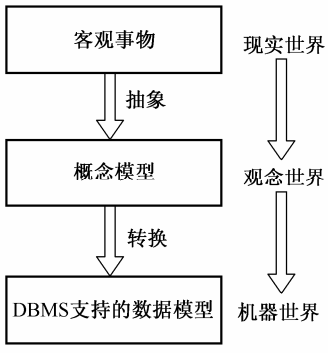

图 1.2 数据抽象的过程

1. 对现实世界的抽象

现实世界就是客观存在的世界，其中存在着各种客观事物及其相互之间的联系，而且每个事物都有自己的特征或性质。计算机处理的对象是现实世界中的客观事物，在对其实施处理的过程中，首先应了解和熟悉现实世界，从对现实世界的调查和观察中抽象出大量描述客观事

物的事实，再对这些事实进行整理、分类和规范，进而将规范化的事实数据化，最终实现数据库系统的存储和处理。

2. 观念世界中的概念模型

观念世界是对现实世界的一种抽象，通过对客观事物及其联系所进行的抽象描述构造出概念模型（Conceptual Model）。概念模型的特征是按用户需求观点对数据进行建模，表达了数据的全局逻辑结构，是系统用户对整个应用项目涉及数据的全面描述。概念模型主要用于数据库设计，它独立于现实世界的数据库管理系统，也就是说选择何种数据库管理系统不会影响概念模型的设计。

概念模型的表示方法很多，目前较常用的是实体联系模型（Entity Relationship Model），简称 E-R 模型。E-R 模型主要用 E-R 图来表示。

3. 机器世界中的逻辑模型和物理模型

机器世界是指现实世界在计算机中的体现与反映。现实世界中的客观事物及其联系在机器世界中以逻辑模型（Logical Model）描述。在选定数据库管理系统后，就要将 E-R 图表示的概念模型转换为具体的数据库管理系统支持的逻辑模型。逻辑模型的特征是按计算机实现的观点对数据进行建模，表达了数据库的全局逻辑结构，是设计人员对整个应用项目数据库的全面描述，逻辑模型服务于数据库管理系统的应用实现。通常也把数据的逻辑模型直接称为数据模型。数据库系统中主要的逻辑模型有层次模型、网状模型和关系模型。

物理模型（Physical Model）是对数据最底层的抽象，用以描述数据在物理存储介质上的组织结构，与具体的数据库管理系统、操作系统和硬件有关。

从概念模型到逻辑模型的转换是由数据库设计人员完成的，从逻辑模型到物理模型的转换是由数据库管理系统完成的，一般人员不必考虑物理实现细节，因而逻辑模型是数据库系统的基础，也是应用过程中要考虑的核心问题。

1.3.3　概念模型

E-R 模型的作用

当分析某种应用环境所需的数据时，首先要找出涉及的实体及实体之间的联系，进而得到概念模型，这是数据库设计的先导。

1. 实体与实体集

实体（Entity）是现实世界中任何可以相互区分和识别的事物，它既可以是能触摸的客观对象（如一位教师、一名学生、一种商品等），也可以是抽象的事件（如一场足球比赛、一次借书等）。

性质相同的同类实体的集合称为实体集（Entity Set），如一个学院的所有教师、一届世界杯足球赛的全部场次的比赛等。

2. 属性

每个实体都具有一定的特征或性质，这样才能区分不同的实体。例如，教师的编号、姓名、性别、职称等都是教师实体具有的特征；足球赛的比赛时间、地点、参赛队、比分、裁判姓名等都是足球赛实体的特征。实体的特征称为属性（Attribute），一个实体可用若干个属性来标识。

能唯一标识实体的属性或属性集称为实体标识符，如教师的编号可以作为教师实体的标识符。

3. 类型与值

属性和实体都有类型（Type）和值（Value）之分。属性类型就是属性名及其取值类型，属性值就是属性所取的具体值。例如，教师实体中的“姓名”属性，属性名“姓名”和取字符类型的值是属性类型，而“黎德瑟”“王德浩”等是属性值。每个属性都有特定的取值范围，即值域（Domain），超出值域的属性值则被认为无实际意义，如“性别”属性的值域为（男，女）、“职称”属性的值域为（助教，讲师，副教授，教授）等。由此可见，属性类型是个变量，属性值是变量所取的值，而值域是变量的取值范围。

实体类型（Entity Type）就是实体的结构描述，通常是实体名和属性名的集合；具有相同属性的实体有相同的实体类型。实体值是一个具体的实体，是属性值的集合。例如，教师实体类型是：

教师（编号，姓名，性别，出生日期，职称，基本工资，研究方向）

教师“王德浩”的实体值是：

（T6，王德浩，男，09/21/75，教授，4750，数据库技术）

由此可见，属性值所组成的集合表征为一个实体，相应的这些属性名的集合表征为一个实体类型，相同类型实体的集合称为实体集。

在 SQL Server 2019 中，用“表”来表示同一类实体，即实体集；用“记录”来表示一个具体的实体；用“字段”来表示实体的属性。显然，字段的集合组成一个记录，记录的集合组成一个表。实体类型则代表了表的结构。

4. 实体间的联系

实体之间的对应关系称为联系（Relationship），它反映了现实世界事物之间的相互关联。例如，图书和出版社之间的关联关系为：一个出版社可以出版多种书，同一种书只能在一个出版社出版。

实体间的联系是指一个实体集中可能出现的每一个实体与另一实体集中多少个具体实体存在联系。实体之间有各种各样的联系，归纳起来有 3 种类型：

（1）一对一联系。如果对于实体集 A 中的每一个实体，实体集 B 中至多只有一个实体与之联系，反之亦然，则称实体集 A 与实体集 B 具有一对一联系，记为 1:1。例如，一个工厂只有一个厂长，一个厂长只在一个工厂任职，则厂长与工厂之间的联系是一对一的联系。

（2）一对多联系。如果对于实体集 A 中的每一个实体，实体集 B 中可以有多个实体与之联系，反之，对于实体集 B 中的每一个实体，实体集 A 中至多只有一个实体与之联系，则称实体集 A 与实体集 B 具有一对多联系，记为 1:n。例如，一个公司有许多职员，但一个职员只能在一个公司就职，所以公司和职员之间的联系是一对多的联系。

（3）多对多联系。如果对于实体集 A 中的每一个实体，实体集 B 中可以有多个实体与之联系，而对于实体集 B 中的每一个实体，实体集 A 中也可以有多个实体与之联系，则称实体集 A 与实体集 B 之间有多对多的联系，记为 m:n。例如，一个读者可以借阅多种图书，任何一种图书可以被多个读者借阅，所以读者和图书之间的联系是多对多的联系。

5. E-R 图

概念模型是反映实体及实体之间联系的模型。在建立概念模型时，要逐一给实体命名以示区别，并描述它们之间的各种联系。E-R 图是用一种直观的图形方式建立现实世界中实体及其联系模型的工具，也是数据库设计的一种基本工具。

E-R 模型用矩形框表示现实世界中的实体，用菱形框表示实体间的联系，用椭圆框表示实体和联系的属性，实体名、联系名和属性名分别写在相应框内。对于作为实体标识符的属性，在属性名下画一条横线。实体与相应的属性之间、联系与相应的属性之间用线段连接。联系与其涉及的实体之间也用线段连接，同时在线段旁标注联系的类型（1:1、1:n 或 m:n）。

图 1.3 所示为学生信息系统中的 E-R 图，该图建立了学生、课程和学院 3 个不同的实体及其联系的模型。其中“课程号”属性作为课程实体的标识符（不同课程的课程号不同），“学号”属性作为学生实体的标识符，“学院编号”属性作为学院实体的标识符。联系也可以有自己的属性，如学生实体和课程实体之间的“选课”联系可以有“成绩”属性。

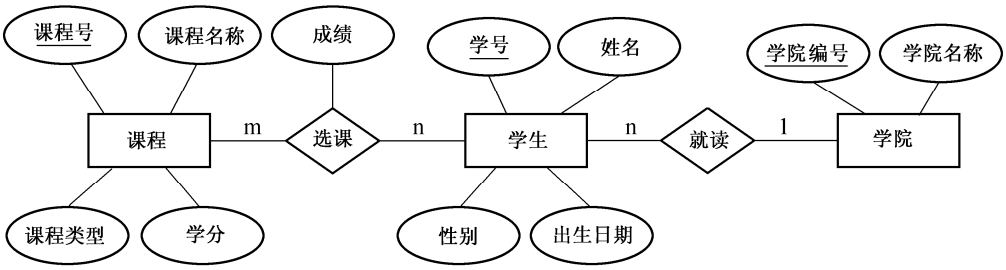

图 1.3　学生信息系统中的 E-R 模型

1.3.4　逻辑模型

E-R 模型只能说明实体间语义的联系，不能进一步说明详细的数据结构。在进行数据库设计时，总是先设计 E-R 模型，然后再把 E-R 模型转换成计算机能实现的逻辑数据模型，如关系模型。逻辑模型不同，描述和实现方法也不同，相应的支持软件（即数据库管理系统）也不同。在数据库系统中，常用的逻辑模型有层次模型、网状模型和关系模型 3 种。

1. *层次模型*

层次模型（Hierarchical Model）用树型结构来表示实体及其之间的联系。在这种模型中，数据被组织成由“根”开始的“树”，每个实体由根开始沿着不同的分枝放在不同的层次上。树中的每一个节点代表一个实体类型，连线则表示它们之间的关系。根据树型结构的特点，建立数据的层次模型需要满足以下两个条件：

（1）有一个节点没有父节点，这个节点即根节点。

（2）其他节点有且仅有一个父节点。

事实上，许多实体间的联系本身就是自然的层次关系。如一个单位的行政机构、一个家庭的世代关系等。

层次模型的特点是各实体之间的联系通过指针来实现，查询效率较高。但由于受到以上两个条件的限制，它能够比较方便地表示出一对一和一对多的实体联系，而不能直接表示出多对多的实体联系。对于多对多的联系，必须先将其分解为几个一对多的联系才能表示出来。因此，对于复杂的数据关系，实现起来较为麻烦，这就是层次模型的局限性。

采用层次模型来设计的数据库称为层次数据库。层次模型的数据库管理系统是最早出现的，它的典型代表是 IBM 公司在 1968 年推出的信息管理系统（Information Management System，IMS），这是世界上最早出现的大型数据库系统。

2. 网状模型

网状模型（Network Model）用以实体类型为节点的有向图来表示各实体及其之间的联系。其特点如下：

（1）可以有一个以上的节点无父节点。

（2）至少有一个节点有多于一个的父节点。

网状模型比层次模型复杂，但它可以直接用来表示多对多联系。然而由于技术上的困难，一些已实现的网状数据库管理系统（如 20 世纪 70 年代数据库系统语言协会下属的数据库任务组提出的 DBTG 系统）中仍然只允许处理一对多联系。

网状模型的特点是各实体之间的联系通过指针实现，查询效率较高，多对多联系也容易实现。但是当实体集和实体集中实体的数目都较多时，大量的指针使得管理工作相当复杂，对用户来说使用也比较麻烦。

3. 关系模型

与层次模型和网状模型相比，关系模型（Relational Model）有着本质的差别，它是用二维表格来表示实体及其相互之间的联系。在关系模型中，把实体集看成一个二维表，每个二维表称为一个关系。每个关系均有一个名字，称为关系名。

关系模型是由若干个关系模式（Relation Schema）组成的集合，关系模式就相当于前面提到的实体类型，它的实例称为关系（Relation）。例如，教师关系模式：教师（编号，姓名，性别，出生日期，职称，基本工资，研究方向），其关系实例如表 1.1 所示，表 1.1 就是一个教师关系。

表 1.1 教师关系

编号	姓名	性别	出生日期	职称	基本工资	研究方向
T1	黎德瑟	女	09/24/66	教授	5200	软件工程
T2	蔡理仁	男	11/27/83	讲师	3960	数据库技术
T3	张肆谦	男	12/23/91	助教	3450	网络技术
T4	李豆豆	男	01/27/73	副教授	4100	信息系统
T5	周武士	女	07/15/89	助教	3600	信息安全
T6	王德浩	男	09/21/75	教授	4750	数据库技术

一个关系就是没有重复行和重复列的二维表，二维表的每一行在关系中称为元组，每一列在关系中称为属性。教师关系的每一行代表一个教师的记录，每一列代表教师记录的一个字段。

虽然关系模型比层次模型和网状模型发展得晚，但其数据结构简单、容易理解，而且建立在严谨的数学理论基础之上，所以是目前比较流行的一种数据模型。自 20 世纪 80 年代以来，推出的数据库管理系统几乎都支持关系模型，本书讨论的 SQL Server 2019 就是一种关系数据库管理系统。

1.4 关系数据库

在关系数据库中，数据的逻辑结构是采用关系模型（即二维表格）来描述实体及其相互间的联系。关系数据库一经问世，即赢得了用户的广泛青睐和数据库开发商的积极支持，使其

迅速成为继层次数据库、网状数据库之后的一种崭新的数据组织方式，在数据库技术领域的市场占有量不断提高并最终占据统治地位。

1.4.1　关系数据库的基本概念

关系数据库的基本数据结构是关系，即平时所说的二维表格，在 E-R 模型中对应于实体集，而在数据库中又对应于表。因此二维表格、实体集、关系、表指的是同一个概念，只是使用的场合不同而已。

1. 关系

通常将一个没有重复行、重复列，并且每个行、列的交叉点只有一个基本数据的二维表格看成一个关系。二维表格包括表头和表中的内容，相应地，关系包括关系模式和记录的值，表包括表结构（记录类型）和表的记录，而满足一定条件的规范化关系的集合就构成了关系模型。

尽管关系与二维表格、传统的数据文件有相似之处，但它们之间又有着重要的区别。严格地说，关系是一种规范化了的二维表格。在关系模型中，对关系做了种种规范性限制，关系具有以下六个性质：

（1）关系必须规范化，每一个属性都必须是不可再分的数据项。规范化是指关系模型中每个关系模式都必须满足一定的要求，最基本的要求是关系必须是一个二维表格，每个属性值必须是不可分割的最小数据单元，即表中不能再包含表。例如，表 1.2 就不能直接作为一个关系。因为该表的“工资标准”一列有 3 个子列，这与每个属性不可再分割的要求不符。只要去掉“工资标准”项，而将“基本工资”“标准津贴”“业绩津贴”直接作为基本的数据项就可以了。

表 1.2　不能直接作为关系的表格示例

编号	姓名	工资标准		
		基本工资/元	标准津贴/元	业绩津贴/元
E1	张东	4350	2500	1780
E2	王南	3450	1350	1560
E3	李西	4245	2900	1870
E4	陈北	3780	2300	1780

（2）列是同质的（Homogeneous），即每一列中的分量是同一类型的数据，来自同一个域。

（3）在同一关系中不允许出现相同的属性名。

（4）关系中不允许有完全相同的元组。

（5）在同一关系中元组的次序无关紧要，也就是说，任意交换两行的位置并不影响数据的实际含义。

（6）在同一关系中属性的次序无关紧要，任意交换两列的位置也并不影响数据的实际含义，不会改变关系模式。

以上是关系的基本性质，也是衡量一个二维表格是否构成关系的基本要素。在这些基本要素中，属性不可再分割是关键，这构成关系的基本规范。

在关系模型中，数据结构简单、清晰，同时有严谨的数学理论作为指导，为用户提供了较为全面的操作支持，所以关系数据库成为当今数据库应用的主流。

2. 元组

二维表格的每一行在关系中称为元组（Tuple），相当于表的一个记录（Record）。每一行描述了现实世界中的一个实体。如表 1.1 中，每行描述了一个教师的基本信息。在关系数据库中，行是不能重复的，即不允许两行的全部元素完全对应相同。

3. 属性

二维表格的每一列在关系中称为属性（Attribute），相当于记录中的一个字段（Field）或数据项。每个属性有一个属性名，一个属性在其每个元组上的值称为属性值，因此，一个属性包括多个属性值，只有在指定元组的情况下属性值才是确定的。同时，每个属性有一定的取值范围，称为该属性的值域，如表 1.1 中的第 3 列，属性名是“性别”，取值是“男”或“女”，不是“男”或“女”的数据应被拒绝存入该表，对进入表的数据进行了约束。同样，在关系数据库中，列是不能重复的，即关系的属性不允许重复。属性必须是不可再分的，即属性是一个基本的数据项，不能是几个数据的组合项。

有了属性概念后，可以这样定义关系模式和关系模型：关系模式是属性名及属性值域的集合；关系模型是一组相互关联的关系模式的集合。

4. 关键字

关系中能唯一区分和确定不同元组的单个属性或属性组合，称为该关系的一个关键字。关键字又称为键（Key）或码。单个属性组成的关键字称为单关键字，多个属性组合的关键字称为组合关键字。需要强调的是，关键字的属性值不能取空值。空值就是不知道或不确定的值，因为空值无法唯一地区分、确定元组。

在表 1.1 所示的关系中，“性别”属性无疑不能充当关键字，“职称”属性也不能充当关键字，从该关系现有的数据分析，“编号”和“姓名”属性均可单独作为关键字，但“编号”作为关键字会更好一些，因为可能会有教师重名的现象，而教师的编号是不会相同的。这也说明，某个属性能否作为关键字，不能仅凭对现有数据进行归纳确定，还应根据该属性的取值范围进行分析判断。

关系中能够作为关键字的属性或属性组合可能不是唯一的。凡在关系中能够唯一区分、确定不同元组的属性或属性组合称为候选关键字（Candidate Key）。例如，表 1.1 所示关系中的“编号”和“姓名”属性都是候选关键字（假定没有重名的教师）。

在候选关键字中选定一个作为关键字，称为该关系的主关键字或主键（Primary Key）。关系中主关键字是唯一的。

5. 外部关键字

如果关系中某个属性或属性组合并非本关系的关键字，但却是另一个关系的关键字，则称这样的属性或属性组合为本关系的外部关键字或外键（Foreign Key）。在关系数据库中，用外部关键字表示两个表之间的联系。例如，在表 1.1 所示的教师关系中，增加“部门代码”属性，则“部门代码”属性就是一个外部关键字，该属性是“部门”关系的关键字，该外部关键字描述了“教师”和“部门”两个实体之间的联系。

1.4.2 关系运算

在关系模型中，数据是以二维表格的形式存在的，这是一种非形式化的定义。由于关系是属性个数相同的元组的集合，因此可以从集合论角度对关系进行集合运算。

利用集合论的观点，关系是元组的集合，每个元组包含的属性数目相同，其中属性的个数称为元组的维数。通常元组用小括号括起来的属性值表示，属性值间用逗号隔开。例如，（E1,张东,女）是三元组。

设 $A_1,A_2,\ldots,A_n$ 是关系 R 的属性，通常用 $R(A_1,A_2,\ldots,A_n)$来表示这个关系的一个框架，即 R 的关系模式。属性的名字唯一，属性 A_i 的取值范围 D_i（i=1,2,…,n）称为值域。

将关系与二维表进行比较可以看出两者存在简单的对应关系，关系模式对应一个二维表的表头，而关系的一个元组就是二维表的一行，很多时候甚至不加区别地使用这两个概念。例如，职工关系 R={(E1,张东,女),(E2,王南,男),(E3,李西,男),(E4,陈北,女)}，相应的二维表格表示形式如表 1.3 所示。

表 1.3　职工关系 R

编号	姓名	性别
E1	张东	女
E2	王南	男
E3	李西	男
E4	陈北	女

在关系运算中，并、交、差运算是从元组（即表格中的一行）的角度来进行的，沿用了传统的集合运算规则，也称为传统的关系运算。而连接、投影、选择运算是关系数据库中专门建立的运算规则，不仅涉及行而且涉及列，故称作专门的关系运算。

1. 传统的关系运算

传统的关系运算有并、差、交、广义笛卡尔积等运算。

（1）并（Union）。设 R、S 同为 n 元关系，且相应的属性取自同一个域，则 R、S 的并也是一个 n 元关系，记作 R∪S。R∪S 包含了所有分属于 R、S 或同属于 R、S 的元组。因为集合中不允许有重复元素，因此同时属于 R、S 的元组在 R∪S 中只出现一次。

（2）差（Difference）。设 R、S 同为 n 元关系，且相应的属性取自同一个域，则 R、S 的差也是一个 n 元关系，记作 R-S。R-S 包含了所有属于 R 但不属于 S 的元组。

（3）交（Intersection）。设 R、S 同为 n 元关系，且相应的属性取自同一个域，则 R、S 的交也是一个 n 元关系，记作 R∩S。R∩S 包含了所有同属于 R、S 的元组。

实际上，交运算可以通过差运算的组合来实现，如 A∩B=A-(A-B)或 B-(B-A)。

（4）广义笛卡尔积（Extended Cartesian Product）。设 R 是一个包含 m 个元组的 j 元关系，S 是一个包含 n 个元组的 k 元关系，则 R、S 的广义笛卡尔积是一个包含 m×n 个元组的 j+k 元关系，记作 R×S，并定义 $R\times S=\{(r_1,r_2,\ldots,r_j,s_1,s_2,\ldots,s_k)|(r_1,r_2,\ldots,r_j)\in R$ 且 $\{s_1,s_2,\ldots,s_k\}\in S\}$，即 R×S 的每个元组的前 j 个分量是 R 中的一个元组，而后 k 个分量是 S 中的一个元组。

【例 1.1】设 $R=\{(a_1,b_1,c_1),(a_1,b_2,c_2),(a_2,b_2,c_1)\}$，$S=\{(a_1,b_2,c_2),(a_1,b_3,c_2),(a_2,b_2,c_1)\}$，求 R∪S、R-S、R∩S、R×S。

根据运算规则，有以下结果：

$R\cup S=\{(a_1,b_1,c_1),(a_1,b_2,c_2),(a_2,b_2,c_1),(a_1,b_3,c_2)\}$

$R-S=\{(a_1,b_1,c_1)\}$

$R \cap S=\{(a_1,b_2,c_2),(a_2,b_2,c_1)\}$

$R \times S=\{(a_1,b_1,c_1,a_1,b_2,c_2),(a_1,b_1,c_1,a_1,b_3,c_2),(a_1,b_1,c_1,a_2,b_2,c_1),$
$(a_1,b_2,c_2,a_1,b_2,c_2),(a_1,b_2,c_2,a_1,b_3,c_2),(a_1,b_2,c_2,a_2,b_2,c_1),$
$(a_2,b_2,c_1,a_1,b_2,c_2),(a_2,b_2,c_1,a_1,b_3,c_2),(a_2,b_2,c_1,a_2,b_2,c_1)\}$

R×S 是一个包含 9 个元组的六元关系。

2. 专门的关系运算

专门的关系运算包括选择、投影、连接三种类型。

选择运算

（1）选择（Selection）。设 $R=\{(a_1,a_2,\ldots,a_n)\}$是一个 n 元关系，F 是关于$(a_1,a_2,\ldots,a_n)$的一个条件，R 中所有满足 F 条件的元组组成的子关系称为 R 的一个选择，记做 $\sigma_F(R)$，并定义 $\sigma_F(R)=\{(a_1,a_2,\ldots,a_n)|(a_1,a_2,\ldots,a_n)\in R$ 且$(a_1,a_2,\ldots,a_n)$满足条件 F}。

简而言之，对 R 关系按一定规则筛选一个子集的过程就是对 R 施加了一次选择运算。

投影运算

（2）投影（Projection）。设 $R(A_1,A_2,\ldots,A_n)$是一个 n 元关系，$\{i_1,i_2,\ldots,i_m\}$是$\{1,2,\ldots,n\}$的一个子集，并且 $i_1<i_2<\ldots<i_m$，定义 $\Pi(R)=R_1(A_{i_1},A_{i_2},\ldots,A_{i_m})$，即 $\Pi(R)$是 R 中只保留属性 $A_{i_1},A_{i_2},\ldots,A_{i_m}$ 的新关系，称 $\Pi(R)$是 R 在 $A_{i_1},A_{i_2},\ldots,A_{i_m}$ 属性上的一个投影，通常记作 $\Pi_{(A_{i_1},A_{i_2},\cdots,A_{i_m})}(R)$。

通俗地讲，关系 R 上的投影是从 R 中选择出若干属性列组成新的关系。

（3）连接（Join）。连接是从两个关系的笛卡尔积中选取属性间满足一定条件的元组，记做 $R \underset{A\theta B}{\bowtie} S$，其中 A 和 B 分别为 R 和 S 上维数相等且可比的属性组，θ 是比较运算符。连接运算从 R 和 S 的笛卡尔积 R×S 中选取关系 R 在 A 属性组上的值与关系 S 在 B 属性组上的值满足比较关系 θ 的元组。

连接运算中有两种常用的连接：等值连接和自然连接。θ 为等号“=”的连接运算，称为等值连接，它是从关系 R 与关系 S 的笛卡尔积中选取 A、B 属性值相等的那些元组。自然连接是一种特殊的等值连接，它要求关系 R 中的属性 A 和关系 S 中的属性 B 名字相同，并且在结果中把重复的属性去掉。一般的连接操作是从行的角度进行运算，但自然连接还需要取消重复列，所以是同时从行和列的角度进行运算。

在关系 R 和关系 S 进行自然连接时，选择两个关系在公共属性上的值相等的元组构成新的关系，此时，关系 R 中的某些元组可能在关系 S 中不存在公共属性上值相等的元组，造成关系 R 中这些元组的值在操作时被舍弃。由于同样的原因，关系 S 中的某些元组也有可能被舍弃。为了在操作时能保存这些将被舍弃的元组，提出了外连接（Outer Join）操作。

如果 R 和 S 进行自然连接时，把该舍弃的元组也保存在新关系中，同时在这些元组新增加的属性上填上空值（Null），这种连接就称为外连接。如果只把 R 中要舍弃的元组放到新关系中，那么这种连接称为左外连接；如果只把 S 中要舍弃的元组放到新关系中，那么这种连接称为右外连接；如果把 R 和 S 中要舍弃的元组都放到新关系中，那么这种连接称为全外连接。

【例 1.2】设有两个关系模式 R(A,B,C)和 S(B,C,D)，其中关系 R={(a,b,c),(b,b,f),(c,a,d)}，关系 S={(b,c,d),(b,c,e),(a,d,b),(e,f,g)}，分别求 $\Pi_{(A,B)}(R)$、$\sigma_{A=b}(R)$、$R \underset{R.A=S.B}{\bowtie} S$、R 和 S 自然连接、R 和 S 全外连接、R 和 S 左外连接、R 和 S 右外连接的结果。

根据连接运算的规则，结果如下：

$\Pi_{(A,B)}(R)=\{(a,b),(b,b),(c,a)\}$

$\sigma_{A=b}(R)=\{(b,b,f)\}$

$R \underset{R.A=S.B}{\bowtie} S=\{(a,b,c,a,d,b),(b,b,f,b,c,d),(b,b,f,b,c,e)\}$

R 和 S 自然连接=$\{(a,b,c,d),(a,b,c,e),(c,a,d,b)\}$

R 和 S 全外连接=$\{(a,b,c,d),(a,b,c,e),(c,a,d,b),(b,b,f,Null),(Null,e,f,g)\}$

R 和 S 左外连接=$\{(a,b,c,d),(a,b,c,e),(c,a,d,b),(b,b,f,Null)\}$

R 和 S 右外连接=$\{(a,b,c,d),(a,b,c,e),(c,a,d,b),(Null,e,f,g)\}$

【例 1.3】一个关系数据库由职工关系 E 和工资关系 W 组成，关系模式如下：

E（编号，姓名，性别）

W（编号，基本工资，标准津贴，业绩津贴）

写出实现以下功能的关系运算表达式：

（1）查询全体男职工的信息。

（2）查询全体男职工的编号和姓名。

（3）查询全体职工的基本工资、标准津贴和业绩津贴。

根据运算规则，写出关系运算表达式如下：

（1）对职工关系 E 进行选择运算，条件是“性别='男'”，关系运算表达式：

$\sigma_{性别='男'}(E)$

（2）先对职工关系 E 进行选择运算，条件是“性别='男'”，这时得到一个“男”职工关系，再对“男”职工关系在属性“编号”和“姓名”上做投影运算，关系运算表达式：

$\Pi_{(编号,\ 姓名)}(\sigma_{性别='男'}(E))$

（3）先对职工关系 E 和工资关系 W 进行连接运算，连接条件是“E.编号=W.编号”，这时得到一个职工工资关系，再对职工工资关系作投影计算，关系运算表达式：

$\Pi_{(编号,\ 姓名,\ 基本工资,\ 标准津贴,\ 业绩津贴)}(E \underset{E.编号=W.编号}{\bowtie} W)$

1.4.3　关系的完整性约束

为了防止不符合规则的数据进入数据库，数据库管理系统提供了一种对数据的监控机制，这种机制允许用户按照具体应用环境定义数据有效性和相容性条件。在对数据进行插入、删除、修改等操作时，数据库管理系统自动按照用户定义的条件对数据实施监控，使不符合条件的数据不能进入数据库，以确保数据库中存储的数据正确、有效、相容。这种监控机制称为数据完整性保护，用户定义的条件称为完整性约束条件。在关系模型中，数据完整性包括实体完整性（Entity Integrity）、参照完整性（Referential Integrity）和用户自定义完整性（User-defined Integrity）3 种类型。

1. 实体完整性

现实世界中的实体是可区分的，即它们具有某种唯一性标识。相应地，关系模型中以主关键字作为唯一性标识。主关键字中的属性（即主属性）不能取空值。如果主属性取空值，就说明存在某个不可标识的实体，即存在不可区分的实体，这与现实世界的应用环境相矛盾，因此这个实体一定不是一个完整的实体。

实体完整性就是指关系的主属性不能取空值，并且不允许两个元组的关键字的值相同。也就是说，一个二维表中没有两个完全相同的行，因此实体完整性也称为行完整性。

2. 参照完整性

现实世界中的实体之间往往存在某种联系，在关系模型中实体及实体间的联系都是用关系来描述的，这样就自然存在着关系与关系间的引用。

设 F 是关系 R 的一个或一组属性，但不是关系 R 的关键字，如果 F 与关系 S 的主关键字 Ks 相对应，则称 F 是关系 R 的外部关键字，并称关系 R 为参照关系（Referencing Relation），关系 S 为被参照关系（Referenced Relation）或目标关系（Target Relation）。

参照完整性就是定义外部关键字与主关键字之间的引用规则，即对于 R 中每个元组在 F 上的值必须取空值或等于 S 中某个元组的主关键字值。

3. 用户定义完整性

实体完整性和参照完整性适用于任何关系数据库系统。此外，不同的关系数据库系统根据其应用环境的不同，往往还需要一些特殊的约束条件。用户定义完整性就是针对某一具体关系数据库的约束条件，它反映某一具体应用所涉及的数据必须满足的语义要求，如规定关系中某一属性的取值范围。

1.4.4 关系数据库设计实例

前面介绍了关系模型和关系数据库的基本概念，下面以图 1.3 为例，按实体间不同的联系方式来分别讨论将 E-R 图转化为关系模型的一般方法，进而讨论一个关系数据库的实例。

1. 1:n 联系到关系模型的转化

在图 1.3 中，学院与学生的联系是一对多的联系。这种联系在进行关系模型转化时，把每个实体分别转化为一个关系，实体名作为关系名，实体属性作为关系属性，并在 1:n 联系的 n 方（本例是学生实体）中增加一个属性，该属性为与该实体相联系的另一个实体（本例是学院）的关键字，即学院编号属性。这样，根据学院与学生两个实体所转化的关系是：

学生（学号，姓名，性别，出生日期，编号），其中学号作为关键字；

学院（学院编号，学院名称），其中编号作为关键字。

对照图 1.3，在学生关系中增加了学院的关键字“学院编号”作为它的一个属性，引入该属性的意义在于描述这两个实体间的联系。从与之相联系的另一个实体引入的属性称为外部关键字，外部关键字描述的不是本实体的一个属性，而是描述本实体与另一个实体的联系。

2. m:n 联系到关系模型的转化

图 1.3 中学生与课程的联系是多对多的联系。对这样的联系进行关系模型转化时，需把两个实体独立地转化为两个关系，将实体名作为关系名，实体属性转化为关系属性。此外要单独设置一个关系来描述两个实体间的联系，其属性由两个实体的关键字及联系本身的属性组成。这样，根据学生和课程两个实体及其联系转化所得到的关系共有 3 个：

学生（学号，姓名，性别，出生日期），其中学号作为关键字；

课程（课程号，课程名称，课程类型，学分），其中课程号作为关键字；

选课（学号，课程号，成绩），其中学号和课程号的组合作为关键字。

3. 1:1 联系到关系模型的转化

其转化方法是将两个实体按上述实体转化方法分别转化为两个关系，并在每一个关系中

增加一个外部关键字，外部关键字由与本实体相联系的对方实体的关键字组成。图 1.3 中没有一对一的联系。

将一个 E-R 图中的每组联系的两个实体按上述方法分别转化为关系后，还需要对转化所得到的关系进行整理。如本例，学生实体既与学院实体有联系，也与课程有联系，上述转化过程中得到了两个不同的学生关系，像这种情况应取包含较多属性的关系作为最后结果。因此，根据图 1.3 转化的关系模型应该是：

学生（学号，姓名，性别，出生日期，编号）；

学院（学院编号，学院名称）；

课程（课程号，课程名称，课程类型，学分）；

选课（学号，课程号，成绩）。

4. 学生信息数据库

相应地，学生信息数据库（Student）中有学生信息表（StInfo）、课程信息表（CInfo）、选课信息表（SCInfo）和学院信息表（DInfo），它们在机器中的物理存储结构也有其相应的形式。各表的结构分别如表 1.4 至表 1.7 所示。表中的数据类型和宽度表示该属性所含数据的类型和长度，在 1.5 节中将详细介绍。

表 1.4　学生信息表的结构

列名	数据类型	宽度	说明	列名	数据类型	宽度	说明
StID	char	10	学号（主关键字）	StName	varchar	20	姓名
StSex	char	2	性别	Birthdate	datetime	8	出生日期
Class	varchar	30	班级名称	Telephone	varchar	20	联系电话
PSTS	char	4	政治面貌	Address	varchar	150	联系地址
DID	char	2	学院编号（外部关键字）	Resume	varchar	255	简历

说明：Student 数据库规定学生关系中的学号由 10 位数字组成，其中从左边数起的前两位表示所在学院，其后的两位表示专业，中间两位表示年级，再后两位表示班级，最后两位表示所在班级的学生编号。

表 1.5　课程信息表的结构

列名	数据类型	宽度	说明	列名	数据类型	宽度	说明
CNo	char	7	课程号（主关键字）	CName	varchar	30	课程名称
CType	char	4	课程类型	Credit	smallint	2	学分
CDes	varchar	100	备注				

表 1.6　选课信息表的结构

列名	数据类型	宽度	说明	列名	数据类型	宽度	说明
StID	char	10	学号（外部关键字）	Score	int	4	成绩
CNo	char	7	课程号（外部关键字）				

表 1.7　学院信息表的结构

列名	数据类型	宽度	说明	列名	数据类型	宽度	说明
DID	char	2	学院编号（主关键字）	DName	varchar	30	学院名称

由以上例子可知，关系模型中的各个关系模式不是随意组合在一起的，要使关系模型准确地反映事物及其之间的联系，需要进行关系数据库的设计。

1.5　SQL Server 2019 概述

SQL Server 2019 是由 Microsoft 公司开发和推广的关系数据库管理系统，是一个全面的数据库管理平台，为用户提供开发语言、数据类型、本地或云环境以及操作系统等方面的选择，可以满足成千上万用户的海量数据管理需求，快速构建相应的解决方案以实现私有云和公有云之间数据的扩展与应用的迁移，帮助用户进行各种数据分析、管理和操作。

1.5.1　服务器组件

SQL Server 2019 是一个功能全面整合的数据库平台，它包含了数据库引擎（Database Engine）、分析服务（Analysis Services）、报表服务（Reporting Services）、集成服务（Integration Services）等服务器组件。SQL Server 2019 的不同版本提供的服务器组件也会有所不同。

1. 数据库引擎

数据库引擎（SQL Server Database Engine，SSDE）是用于存储、处理和保护数据的核心服务。数据库引擎提供了受控访问和快速事务处理，以满足企业内最苛刻的数据应用程序的要求。数据库引擎还提供了大量的支持以保持高可用性。使用数据库引擎可以创建用于联机事务处理或联机分析处理的关系数据库。例如，创建数据库、创建表、数据查询、访问数据库等操作都是由数据库引擎完成的。一般来说，使用数据库系统实际上就是在使用数据库引擎。

2. 分析服务

SQL Server 分析服务（SQL Server Analysis Services，SSAS）是一种核心组件服务，支持对业务数据的快速分析，以及为商业智能应用程序提供联机分析处理（Online Analytical Processing，OLAP）和数据挖掘功能。

可以使用分析服务来设计、创建和管理包含来自多个数据源的详细数据和聚合数据的多维结构，这些数据源（如关系数据库）都存在于内置计算支持的单个统一逻辑模型中。

分析服务为统一的数据模型构建的大量数据提供快速、直观、由上至下的分析，这样可以采用多种语言向用户提供数据。分析服务使用数据仓库、数据集市、生产数据库和操作数据存储区来支持历史数据和实时数据分析。

3. 报表服务

SQL Server 报表服务（SQL Server Reporting Services，SSRS）是一个完整的基于服务器的报表平台，可以建立、管理、发布传统的基于纸张的报表或者交互的、基于 Web 的报表，而且可以将 SQL Server 和 Windows Server 的数据管理功能与 Office 应用系统相结合，实现信息的实时传递、转换，并展示数据的改变。

SQL Server 2019 报表服务不仅提供了对关系数据库的支持，而且对分析服务的多维数据也提供了支持，扩展了微软商业智能（BI）平台，以迎合那些需要访问商业数据的应用。报表服务是一个基于服务器的企业级报表环境，可借助 Web Services 进行管理。

4. 集成服务

SQL Server 集成服务（SQL Server Integration Services，SSIS）是一个数据集成平台，负责完成有关数据的提取、转换和加载等操作。其他三种服务就是通过 Integration Services 来进行联系的。除此之处，使用数据集成服务可以高效地处理各种各样的数据，例如：SQL Server、Oracle、Excel、XML 文档、文本文件等。

1.5.2　常用管理工具

SQLServer 系统环境

SQL Server 2019 系统提供了大量的管理工具，通过这些管理工具，可以快速、高效地管理系统。这些管理工具主要包括 SQL Server Management Studio、SQL Server 配置管理器、SQL Server Profiler 等以及大量的命令行实用工具。表 1.8 列举了用来管理 SQL Server 2019 的管理工具及功能。

表 1.8　SQL Server 2019 管理工具及功能

管理工具	功能
SQL Server Management Studio	用于编辑和执行查询，以及启动标准向导任务
SQL Server 配置管理器	管理服务器和客户端网络配置设置
SQL Server Profiler	提供用于监视 SQL Server 数据库引擎实例或 Analysis Services 实例的图形用户界面
数据库引擎优化顾问	是协助创建索引、索引视图和分区的最佳组合
SQL Server Business Intelligence Development Studio	用于包括 Analysis Services、Integration Service 和 Reporting Services 项目在内的商业解决方案的集成开发环境
Reporting Services 配置管理器	提供报表服务器配置的统一的查看、设置和管理方式
SQL Server 安装中心	安装、升级或更改 SQL Server 2019 实例中的组件

下面介绍 3 个重要工具。

1. SQL Server Management Studio

SQL Server Management Studio（SQL 服务器管理工作室，SSMS）是一个集成管理工具组。它将各种图形化工具和多功能的脚本编辑器组合在一起，完成访问、配置、控制、管理和开发 SQL Server 的所有工作，大大方便了技术人员和数据库管理员对 SQL Server 系统的各种访问。

单击“开始”→Microsoft SQL Server 2019→SQL Server Management Studio 18 命令启动 SSMS，系统将自动打开“连接到服务器”对话框，如图 1.4 所示。“连接到服务器”对话框中有“服务器类型”“服务器名称”“身份验证”3 个选项供用户进行选择。

“服务器类型”选项中的“数据库引擎”为默认选项，单击此选项中的下拉箭头，可以看到系统提供了数据库引擎、Analysis Services、Reporting Services、Integration Services 4 种服务。

图 1.4　“连接到服务器”对话框

“服务器名称”下拉列表列出所有可以连接的服务器的名称，可以在其中选择一个进行连接，或者输入“(local)”表示本地服务器。如果要连接到远程数据服务器，则需要输入服务器的 IP 地址。

“身份验证”下拉列表指定连接类型，可选择“Windows 身份验证”或“SQL Server 身份验证”。Windows 身份验证直接利用 Windows 用户的权限登录；SQL Server 身份验证要求用户输入用户名和密码。单击“连接”按钮，即可连接 SQL Server 服务器并打开 SSMS 主窗口，如图 1.5 所示。默认情况下主窗口包含“对象资源管理器”“已注册的服务器”和“对象资源管理器详细信息”三个组件。

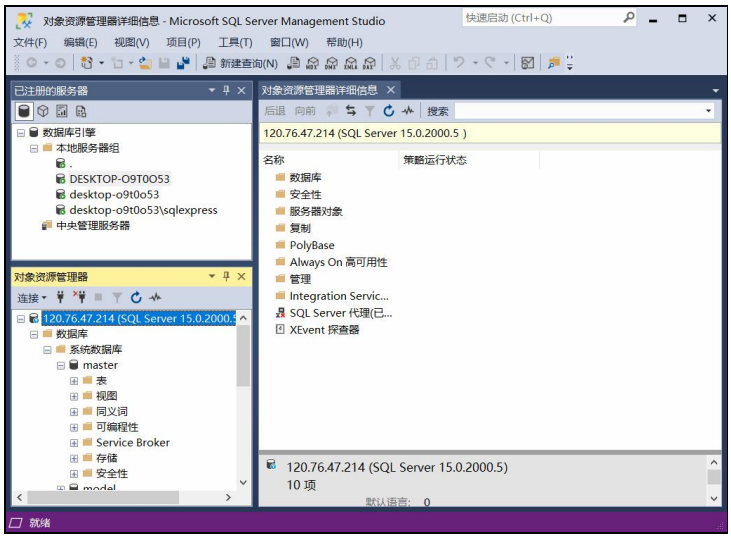

图 1.5　SQL Server Management Studio 主窗口

（1）对象资源管理器。SSMS 的左侧下窗格为“对象资源管理器”，它提供一个层次结构的用户界面，用于查看和管理每个 SQL Server 实例（即 SQL 服务器引擎）中的对象。用户可以直接通过“对象资源管理器”来操作数据库。如果“对象资源管理器”窗格不可见，可通过“视图”菜单中的选项打开。“对象资源管理器”的功能根据服务器的类型稍有不同，但一般都包括用于数据库的开发功能和所有服务器类型的管理功能。

例如，使用“对象资源管理器”查看 master 系统数据库的对象。只要在“对象资源管理

器”中展开“数据库”节点，选择系统数据库中的 master 数据库并展开，将列出该数据库中包含的所有对象，如表、视图、存储过程等。

（2）已注册的服务器。SSMS 可以管理多台服务器，系统使用“已注册的服务器”窗格来组织经常访问的服务器，该窗格中显示了所有已注册的 SQL Server 服务器，单击 SSMS 的“视图”→“已注册的服务器”命令打开。如果用户需要注册一个其他服务，可以右击“本地服务器”节点，在弹出的快捷菜单中选择“新建服务器注册”命令进行创建。

（3）对象资源管理器详细信息。“对象资源管理器详细信息”是 SSMS 的一个组件，它提供服务器中所有对象的表格视图，并显示一个用于管理这些对象的用户界面。默认情况下，“对象资源管理器详细信息”窗格在 SSMS 中是可见的。如果“对象资源管理器详细信息”窗格不可见，可通过“视图”菜单中的选项将其打开。

（4）查询编辑器。在 SSMS 主窗口中集成了“查询编辑器”，这样可以在对服务器进行图形化管理的同时编写 T-SQL 脚本。“查询编辑器”支持彩色代码关键字、可视化地显示语法错误、允许开发人员运行和诊断代码等功能。因此，“查询编辑器”具有很高的集成性和灵活性。

在 SSMS 主窗口的工具栏上，单击“新建查询”按钮，即可打开 SQL Server 2019 的“查询编辑器”，同时在 SSMS 的主菜单中显示“查询”菜单项，如图 1.6 所示。默认情况下，“查询编辑器”窗格中语句关键字显示为蓝色；语句参数和连接器显示为红色；系统函数为洋红色；运算符为深灰色　；无法确定的项，如列名和表名显示为黑色。

例如，查询学生数据库 Student 中学生的基本信息。只要在“查询编辑器”窗格输入以下命令：

```
USE Student
SELECT * FROM StInfo
WHERE StSex ='男'
```

单击工具栏中的“执行”按钮 执行(X)，该查询的结果如图 1.6 所示。

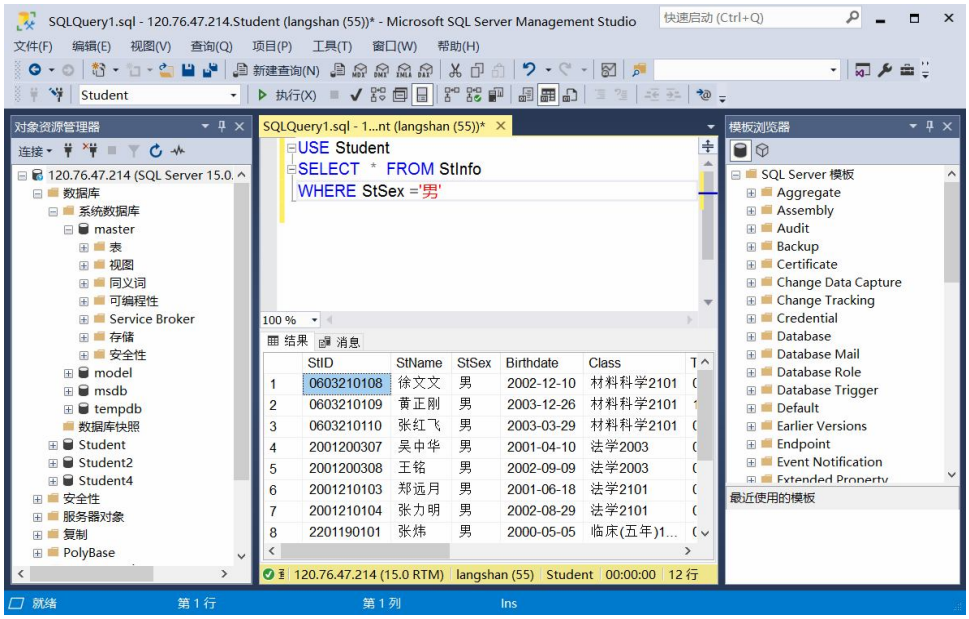

图 1.6　SQL Server Management Studio 集成环境操作效果

注意：SQL Server 2019 中的“查询编辑器”既可以工作在连接模式下，也可以工作在断开模式下。查询编辑器几乎可以处理任何数据源，允许设计 SELECT、INSERT、UPDATE 和 DELETE 等 DML 语句。

（5）模板资源管理器。在“模板资源管理器”中，提供了大量与 SQL Server 和分析服务相关的脚本模板。脚本模板提供了编写 T-SQL 语句的起点。模板实际上就是保存在文件中的脚本片段，可以在 SQL 查询视图中打开并且进行修改，使之适合需要。也就是说，使用模板资源管理器可以降低编写脚本的难度。

“模板资源管理器”窗格是可选的，打开后默认位于 SSMS 主窗口的右侧。用户可以在“模板资源管理器”中浏览可用模板，然后打开该模板以便将代码纳入“查询编辑器”窗口中，也可以创建自定义模板。如果“模板资源管理器”窗格不可见，可单击“视图”菜单中的选项或者按 Ctrl+Alt+T 组合键将其打开。

例如，想了解创建触发器的代码如何编写，只要在“模板资源管理器”窗格找到“Create Server Trigger”模板，双击打开创建触发器的模板便可进行学习。

（6）命令行实用工具。SQL Server 2019 不仅提供了大量的图形化工具，还提供了大量的命令行实用工具。通过这些命令，可以与 SQL Server 2019 进行交互，但不能在图形界面下运行，只能在 Windows 命令提示符下输入命令及参数执行（即相当于 DOS 命令）。这些命令行实用工具主要包括 bcp、dta、dtexec、dtutil 等。若想了解它们的功能、使用方法和其他更多的实用工具，读者可使用 SQL Server 2019 提供的联机帮助获取，这里不作介绍。

2. SQL Server 配置管理器

SQL Server 配置管理器（SQL Server Configuration Manager，SSCM）是一种工具，用于管理与 SQL Server 相关的服务、配置 SQL Server 使用的网络协议以及从 SQL 客户端计算机进行网络连接配置管理。

（1）管理与 SQL Server 相关的服务。在 Microsoft SQL Server 系统中，可以通过“SQL Server 配置管理器”或“计算机管理”工具查看和控制 SQL Server 的服务。

单击“开始”→“SQL Server 2019 配置管理器”命令打开 SSCM 窗口，如图 1.7 所示。在此窗口中用户可以配置每次启动数据库引擎时要使用的选项。

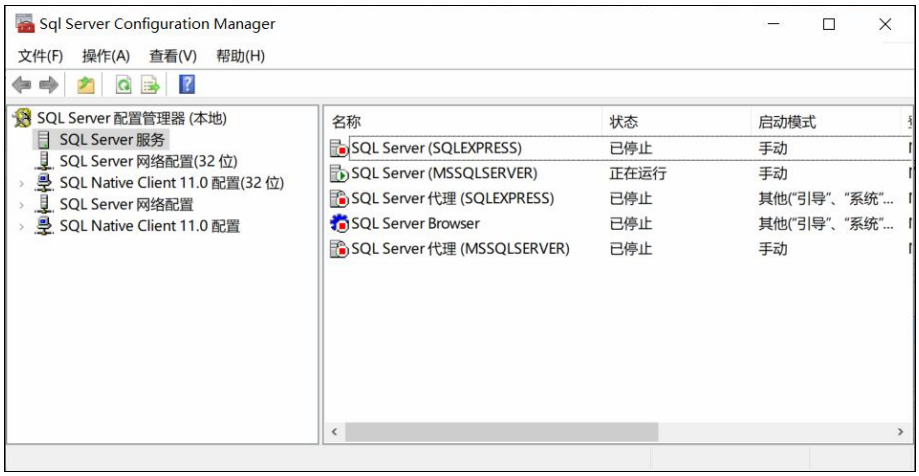

图 1.7　SQL Server Configuration Manager 窗口

通过右击某个服务名称，可以查看该服务的属性以及启动、停止、暂停、重新启动相应的服务。

注意：也可以按 Win+R 组合键，在运行对话框中输入 SQLServerManager15.msc，单击“确定”按钮打开配置管理器。还可以通过“控制面板”→“管理工具”→“计算机管理”选项打开“计算机管理”窗口，展开“服务”节点，单击“SQL Server 配置管理器”目录项，对 SQL Server 2019 的服务进行管理。在“服务”窗格中列出了所有系统中的服务，从列表中找到 9 种有关 SQL Server 2019 的服务，右击服务名称，在弹出的快捷菜单中选择“属性”命令进行配置。

（2）配置 SQL Server 使用的网络协议。在 SQL Server 配置管理器中，展开“SQL Server 网络配置”节点，单击“MSSQLSERVER 的协议”，在右侧详细信息窗格中将显示协议名称及其状态，如图 1.8 所示。用户可以“启用”和“禁用”相关的协议。

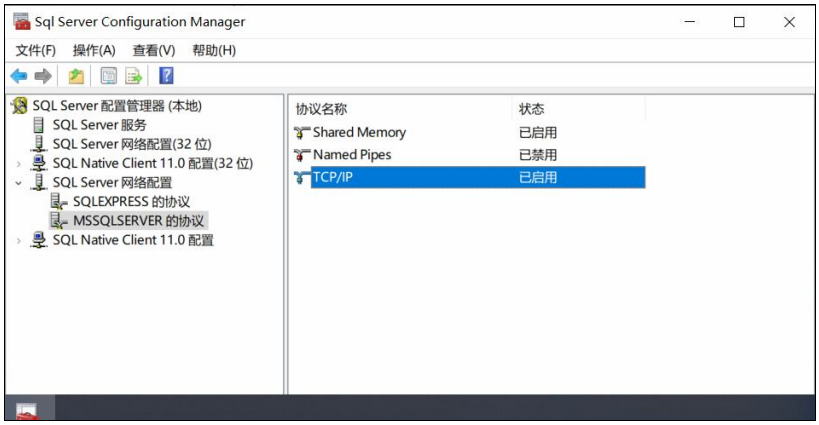

图 1.8　显示协议及其状态窗口

（3）客户端计算机管理网络连接配置。在 SQL Server 配置管理器左侧窗格中，展开“SQL Native Client 11.0 配置”节点，单击“客户端协议”，在右侧详细信息窗格中将显示客户端协议名称、所使用的协议顺序和启用状态，如图 1.9 所示。用户可以“启用”和“禁用”相关的协议。

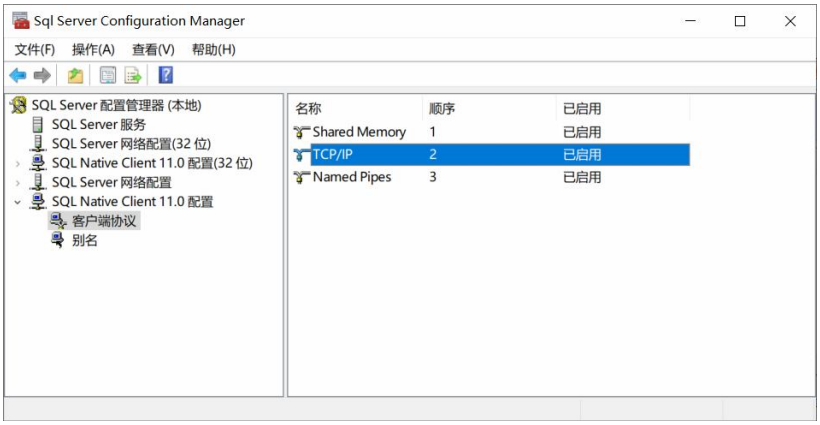

图 1.9　显示客户端协议名称、所使用的协议顺序和启用状态窗口

若要配置客户端所使用的协议顺序，只要在左侧窗格中右击“客户端协议”，在快捷菜单中选择“属性”命令，或者在右侧详细信息窗格中右击某个协议，在弹出的快捷菜单中选择“顺序”命令即可。

3．SQL Server Profiler

SQL Server Profiler（事件探查器）是图形化实时监视工具，它可以从服务器中捕获 SQL Server 事件，能帮助系统管理员监视数据库和服务器的行为，比如死锁的数量、致命的错误、跟踪 Transact SQL 语句和存储过程。可以把这些监视数据存入表或文件中，并在以后某一时间重新显示这些事件来一步一步地进行分析。

例如，若要对 SQL Server 2019 系统的运行过程进行摄录，使用 SQL Server Profiler 工具可以实现。在 SSMS 主窗口中，单击“工具”→Server Profiler 命令即可运行 SQL Server Profiler。

1.5.3 数据类型

在 SQL Server 中，表与视图中的列、变量、存储过程或函数中的参数和返回值等对象都具有数据类型。当指定了某个对象的数据类型时，也就定义了该对象所含的数据类型、存储值的长度、大小、数字精度（仅用于数字数据类型）和数值小数位数（仅用于数字数据类型）。

SQL Server 支持 4 种基本数据类型：数值数据类型、字符和二进制数据类型、日期时间数据类型、逻辑数据类型，用于各类数据值的存储、检索和解释。此外，还有其他一些数据类型，如可变数据类型、表类型等。

1．数值数据类型

SQL Server 提供了多种方法存储数值，SQL Server 的数值数据类型大致可分为 4 种基本类型。

（1）整数数据类型。有 4 种整数数据类型：bigint、int、smallint 和 tinyint，用于存储不同范围的整数值。

1）bigint 数据类型的存储长度为 8 字节，其中一个二进制位表示符号，其他 63 个二进制位表示大小，可存储取-2^{63}～$2^{63}-1$ 之间的整数。

2）int 数据类型的存储长度是 4 字节，其中一个二进制位表示符号，其他 31 个二进制位表示大小，取值范围是-2^{31}～$2^{31}-1$ 之间的整数。

3）smallint 数据类型的存储长度为 2 字节，其中一个二进制位表示符号，其他 15 个二进制位表示大小，取值范围是-2^{15}～$2^{15}-1$。

4）tinyint 数据类型的存储长度为 1 字节，取值范围是 0～255。

整数可以用较少的字节存储较大的精确数字，考虑到其高效的存储机制，只要有可能，对数值列应尽量使用整数。

（2）浮点数据类型。浮点数据用来存储系统所能提供的最大精度保留的实数数据。近似数字的运算存在误差，因此浮点数据不能用于需要精度固定的运算，如货币数据的运算。

1）real 数据类型可精确到第 7 位小数，表示范围为-3.40×10^{38}～3.40×10^{38}。每个 real 类型的数据占用 4 字节的存储空间。

2）float 数据类型可精确到第 15 位小数，表示范围为-1.79×10^{308}～1.79×10^{308}。每个 float 类型的数据占用 8 字节的存储空间。

3）float 数据类型可写成 float(n)的形式，n 指定 float 数据的精度，为 1～53 之间的整数，

默认值为 53。当 n 取 1～24 时，实际上定义了一个 real 类型的数据，系统用 4 个字节存储；当 n 取 25～53 时，系统认为是 float 类型的数据，用 8 个字节存储。

（3）精确数值数据类型。精确数值数据类型用于存储有小数点且小数点后位数确定的实数。SQL Server 支持两种精确的数值数据类型：decimal 和 numeric。这两种数据类型几乎是相同的，定义格式如下：

```
decimal[(p[, s])]
numeric[(p[, s])]
```

其中，p 指定精度，定义了总位数，即小数点左边和右边可以存储的十进制数字的最大个数；s 指定小数位数，即小数点右边可以存储的十进制数字的最大个数，0≤s≤p。有效存储空间为 2～17 个字节，使用最大精度时，有效值为$-10^{38}+1$～$10^{38}-1$。例如，decimal(10, 5)表示共有 10 位数，其中整数部分为 5 位，小数部分为 5 位。

（4）货币数据类型。除了 decimal 和 numeric 类型适用于货币数据的处理外，SQL Server 还专门提供了两种货币数据类型：money 和 smallmoney。

1）money 数据类型的存储长度是 8 字节，整数部分包含 19 位数字，小数部分包含 4 位数字，货币数据值介于-2^{63}～$2^{63}-1$ 之间。

2）smallmoney 数据类型与 money 数据类型类似，存储长度是 4 字节，取值范围为-2^{31}～$2^{31}-1$。

输入货币数据时必须在货币数据前加$符号，如果未提供该符号，值被当成浮点数，可能会损失值的精度。在显示货币值时，数值的小数部分仅保留两位有效位。

2. 字符和二进制数据类型

在 SQL Server 中字符和二进制数据类型是一种常用的基本数据类型。

（1）字符数据类型。字符数据类型用于存储汉字、各种字母、数字符号和特殊符号。输入字符型数据时要用单引号（'）将字符括起来。字符型数据有定长字符型（char）、变长字符型（varchar）和文本型（text）三种。

1）char 数据类型的定义形式为 char[(n)]，占用 n 个字节空间，n 的取值为 1～8000，即最多可存储 8000 个字符。通常采用 ANSI 字符集，即单个英文字符占用 1 字节，中文字符等会占用 2 个字节。在用 char(n)数据类型对数据表的列进行说明时，n 指示列长度。如果不指定 n，系统将长度默认为 1。输入字符的长度大于 n 时将会截掉超出的部分，输入字符的长度短于 n 时系统自动用空格填满剩余空间。

2）varchar 数据类型的定义形式为 varchar[(n)]，n 的取值为 1～8000。varchar 数据类型的结构与 char 数据类型一致，它们的主要区别是当输入 varchar 字符长度小于 n 时，剩余空间不用空格来填满，按输入字符的实际长度存储。若输入的数据超过 n 个字符，则截断后存储。varchar 类型数据所需要的存储空间要比 char 类型数据少一些，但 varchar 列的存取速度要比 char 列慢一些。

3）text 数据类型用于存储数据量庞大且长度会变的字符文本数据。text 列的长度可变，最多可包含 2^{31}-1 个字符。若用户要求表中的某列能存储 255 个字符以上的数据，可使用 text 数据类型。

SQL Server 允许使用多国语言，支持 Unicode 标准字符集。为此，SQL Server 提供了多字节的字符数据类型：nchar(n)、nvarchar(n)和 ntext。

nchar 可存放 Unicode 字符的固定长度字符类型，最大长度为 4000 个字符。

nvarchar 可存放 Unicode 字符的可变长度字符类型，最大长度为 4000 字符。

ntext 可存放 Unicode 字符的文本类型，最大长度为 2^{30}-1 个字符。

nchar、nvarchar 和 ntext 的用法分别与 char、varchar 和 text 相同，只是 Unicode 支持的字符范围更大，存储 Unicode 字符所需要的空间更大。

（2）二进制数据类型。SQL Server 二进制数据类型用于存储二进制数或字符串。与字符数据类型相似，在列中插入二进制数据时，用 0x 开头的两个十六进制数构成一个字节，例如输入 0x1AA5 代表十六进制数 1AA5。SQL Server 有 3 种有效二进制数据类型，即定长二进制类型 binary、变长二进制类型 varbinary 和大块二进制类型 image。

1）binary 数据类型的定义形式为 binary[(n)]，n 的取值为 1～8000，若不指定长度则 n 默认为 1。binary 数据用于存储二进制字符，如程序代码和图像数据。数据所需的存储空间为 n 个字节，若输入的数据不足 n 个字节，则补足后存储；若输入的数据超过 n 个字节，则截断超出部分后存储。

2）varbinary[(n)]数据类型与 binary 数据类型基本相同，n 的取值为 1～8000。通过存储输入数据的实际长度而节省存储空间，但存取速度比 binary 类型要慢。varbinary 数据类型的存储长度为实际数据长度+4 个字节。若输入的数据超过 n+4 个字节，则截断后存储。

3）image 数据类型与 text 数据类型类似，可存储 1～2^{31}-1 个字节的二进制数据。不同点在于 image 数据类型存储的是二进制数据而不是文本字符。

除非数据长度超过 8KB，一般宜用 varbinary 类型来存储二进制数据，建议列宽的定义不超过所存储的二进制数据可能的最大长度。image 数据列可以用来存储超过 8KB 的可变长度的二进制数据，如 Word 文档、Excel 电子表格、图像或其他文件。

注意：对于局部变量，text、ntext、image 数据类型无效。

3. 日期时间数据类型

日期时间数据类型用于存储日期和时间数据。SQL Server 支持 6 种日期时间数据类型：date、time、datetime、datetime2、smalldatetime 和 datetimeoffset。

（1）date 数据类型。存储用字符串表示的日期数据，可以表示 0001-01-01 到 9999-12-31（即公元元年 1 月 1 日到公元 9999 年 12 月 31 日）间的任意日期值，占用 3 个字节空间。数据格式为 YYYY-MM-DD，其中，YYYY 表示年份的 4 位数字（0001～9999），MM 表示月份的 2 位数字（01～12），DD 表示天数 2 位数字（01～31）。

（2）time 数据类型。以字符串形式记录一天中的某个时间，占用 5 个字节空间，取值范围为 00:00:00.0000000～23:59:59.9999999。数据格式为 hh:mm:ss[.nnnnnnn]，其中，hh 表示小时的 2 位数字（00～23），mm 表示分钟的 2 位数字（00～59），ss 表示秒的 2 位数字（00～59），n 为 0～7 位数字，表示秒的小数部分（0～9999999）。

（3）datetime 数据类型。用于存储从 1753-01-01 到 9999-12-31 的日期和时间数据，存储 8 字节空间。数据格式为 YYYY-MM-DD hh:mm:ss[.nnn]，其含义与 date 和 time 类型相同。对于定义为 datetime 数据类型的列，并不需要同时输入日期和时间，可省略其中一个。

（4）datetime2 数据类型。是 datetime 类型的扩展，取值范围 0001-01-01 到 9999-12-31，其数据范围更大，默认的小数精度更高。

（5）smalldatetime 数据类型。与 datetime 类型相似，取值范围 1900-01-01 到 2079-06-06，

精确到分钟，占用 4 个字节的存储空间。

（6）datetimeoffset 数据类型。用于定义 24 小时制和可识别时区的日内时间（世界标准时间值，UTC），占用 10 字节空间。数据格式为 hh:mm:ss[.nnnnnnn][{+|-}hh:mm]，其中，第 2 个 hh 表示时区偏移量（-14～+14），第 2 个 mm 表示分钟偏移值（00～59）。例如要存储北京时间 2011 年 11 月 11 日 12 点整，存储时该值将是 2011-11-11 12:00:00+08:00，即北京处于东八区，比 UTC 早 8 个小时。

4. 逻辑数据类型

SQL Server 的逻辑数据类型为 bit，适用于判断真假的场合，长度为 1 个字节。bit 数据类型取值为 1（真）、0（假）或 NULL。非 0 的数据被当成 1 处理，bit 列不允许建立索引，多个 bit 列可以占用同一个字节。如果一个表有不多于 8 个的 bit 列，SQL Server 将这些列合在一起用一个字节存储。如果表中有 9～16 个 bit 列，这些列将作为两个字节存储，更多列的情况依此类推。

5. uniqueidentifier 数据类型

uniqueidentifier 数据类型可存储 16 字节的二进制值，其作用与全局唯一标记符（Globally Unique IDentifier，GUID）一样。GUID 是唯一的二进制数：世界上的任何两台计算机都不会生成重复的 GUID 值。它的数据是形如 xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx 的字符串，共 36 个字符，其中每个 x 是一个十六进制数字，范围为 0～9 或 A～F，如 6F9619FF-8B86-D011-B42D-00C04FC964FF 是一个有效 uniqueidentifier 类型的数据值。每次调用 NEWID()函数可以生成一个全局唯一的 GUID 数据。GUID 主要用于多个节点、多台计算机的网络中，分配必须具有唯一性的标识符。

注意：SQL Server 会自动限制每个系统数据类型的值的范围，当插入数据库中的值超过了数据类型允许的范围时，SQL Server 就会报错。

1.6　Transact–SQL 简介

结构化查询语言（Structured Query Language，SQL）是一种关系数据库标准查询语言，每一个具体的数据库系统都对这种标准的 SQL 有一些功能上的调整（一般是扩展），语句格式也有个别变化，从而形成了各自不完全相同的 SQL 版本。Transact-SQL 就是 SQL Server 中使用的 SQL 版本。本节主要介绍 Transact-SQL 语言基础，数据定义和数据操作语句将在后续章节学习。

1.6.1　SQL 与 Transact-SQL

SQL 是一种介于关系代数与关系演算之间的语言。Transact-SQL（简称 T-SQL）是 Microsoft 公司在关系型数据库管理系统 SQL Server 中实现的一种计算机高级语言，是微软对 SQL 的扩展。

1. SQL 语言

SQL 语言最早是在 20 世纪 70 年代由 IBM 公司开发出来的，并被应用在 DB2 关系数据库系统中，主要用于关系数据库中的信息检索。

SQL 语言被提出后，由于其具有功能丰富、使用灵活、语言简洁易学等突出优点，在计算机工业界和计算机用户中备受欢迎。1986 年 10 月，美国国家标准协会（ANSI）的数据库

委员会批准了 SQL 作为关系数据库语言的美国标准。1987 年 6 月，国际标准化组织（ISO）将其采纳为国际标准，这个标准也称为 SQL 86。SQL 语言标准的出台使 SQL 语言作为标准关系数据库语言的地位得到了加强。随后，SQL 语言标准几经修改和完善，其间经历了 SQL 89、SQL 92、SQL 99，一直到 2003 年的 SQL 2003 等多个版本，每个新版本都较前面的版本有重大改进。随着数据库技术的发展，将来还会推出更新的标准。但是需要说明的是，公布的 SQL 语言标准只是一个建议标准，目前一些主流数据库产品也只达到了基本的要求，并没有完全实现这些标准。

按照 ANSI 的规定，SQL 语言被作为关系数据库的标准语言。SQL 语言中的语句可以用来执行各种各样的操作。目前流行的关系数据库管理系统，如 Access、SQL Server、Oracle、Sybase 等，都采用了 SQL 语言标准，而且很多数据库都对 SQL 语言中的语句进行了再开发和扩展。

尽管设计 SQL 语言的最初目的是查询，但 SQL 语言绝不仅仅是一个查询工具，它可以独立完成数据库的全部操作。根据其实现的功能可以将 SQL 语言划分为以下几类：

（1）数据查询语言（Data Query Language，DQL）：按一定的查询条件从数据库对象中检索符合条件的数据。

（2）数据定义语言（Data Definition Language，DDL）：用于定义数据的逻辑结构以及数据项之间的关系。

（3）数据操纵语言（Data Manipulation Language，DML）：用于更改数据库，包括增加新数据、删除旧数据、修改已有数据等。

（4）数据控制语言（Data Control Language，DCL）：用于控制对数据库中数据的操作，包括基本表和视图等对象的授权、完整性规则的描述、事务开始和结束控制语句等。

可见，SQL 语言是一种能够控制数据库管理系统并能与之交互的综合性语言。但 SQL 语言并不是一种像 C、Pascal 那样完整的程序设计语言，没有用于程序流程控制的 IF 语句、WHILE 语句等，它是一种数据库子语言；SQL 语言也并非严格的结构化语言，同 C、Pascal 这种高度结构化的程序设计语言相比，SQL 语言更像是英语句子，其中包括了不少用于提高可读性的词汇。

2. Transact-SQL

Transact-SQL 最早由 Sybase 公司和 Microsoft 公司联合开发，Microsoft 公司将其应用在 SQL Server 上，并将其作为 SQL Server 的核心组件，与 SQL Server 通信，并访问 SQL Server 中的对象。它在 ANSI SQL 92 标准的基础上进行了扩展，对语法也做了精简，增强了可编程性和灵活性，使其功能更为强大、使用更为方便。随着 SQL Server 2019 的应用普及，T-SQL 也越来越重要了。

T-SQL 对 SQL 的扩展主要包含以下 3 个方面：

（1）增加了流程控制语句。SQL 作为一种功能强大的结构化标准查询语言并没有包含流程控制语句，因此不能单纯使用 SQL 构造出一种最简单的分支程序。T-SQL 在这方面进行了多方面的扩展，增加了块语句、分支判断语句、循环语句和跳转语句等。

（2）加入了局部变量、全局变量等许多新概念，可以编写出更复杂的查询语句。

（3）增加了新的数据类型，处理能力更强。

为了使读者更好地了解和使用 T-SQL，这里对 T-SQL 的语法进行约定，如表 1.9 所示。

表 1.9　T-SQL 的语法约定及使用说明

约定	使用说明
大写	T-SQL 关键字
|（竖线）	分隔括号或大括号中的语法项。只能使用其中一项
{ }（大括号）	必选语法项。不要输入大括号
[ ]（方括号）	可选语法项。不要输入方括号
[, ... n]	指示前面的项可以重复 n 次。各项之间以逗号分隔
[... n]	指示前面的项可以重复 n 次。每一项由空格分隔
[;]	可选的 T-SQL 语句终止符
< >	对象名称

1.6.2　运算符与表达式

运算符是一种符号，通过运算符连接运算量构成表达式。简单表达式可以是一个常量、变量、列或标量函数。可以用运算符将两个或更多的简单表达式连接起来组成复杂表达式。运算符用来指定要在一个或多个表达式中执行的操作。


@和#标识符

1. 标识符

标识符是用户编程时使用的名字。每一个对象都用一个标识符来唯一地标识。对象标识符是在定义对象时创建的，该标识符随后用于引用该对象。标识符包含的字符数必须在 1～128 之间。标识符有两种类型：常规标识符和分隔标识符。

（1）常规标识符。又称为规则标识符，它的第一个字符必须是字母、下划线（_）、@符号或数字符号（#），后续字符可以为字母、数字、@符号、$符号、数字符号或下划线。在 SQL Server 中，某些处于标识符开始位置的符号具有特殊意义。例如，以符号@开头的标识符表示局部变量或参数；以#符号开头的标识符表示临时表或过程；以##符号开头的标识符表示全局临时对象。T-SQL 中的某些函数名称以@@符号开始，为避免混淆这些函数，建议用户不要使用以@@开始的标识符。

（2）分隔标识符。又称为界定标识符，包含在双引号（"）或方括号（[]）内的标识符就是分隔标识符。如果标识符是保留字或包含空格，则需要使用分隔标识符进行处理。例如，在 SELECT * FROM "My Table"命令中，标识符"My Table"有空格，所以使用双引号（"）分隔，即使用分隔标识符。

2. 常量与变量

在程序运行过程中不能改变其值的数据称为常量，相应地，在程序运行过程中可以改变其值的数据称为变量。

（1）常量。常量是表示特定数据值的符号，其格式取决于其数据类型。SQL Server 具有以下几种类型：字符串和二进制常量、日期时间常量、数值常量、逻辑数据常量、空值等。

1）字符串和二进制常量。字符串常量是用单引号括起来的字符系列。若字符串中本身又有单引号字符，则要用两个单引号来表示这个单引号，如 'China'、'O"Brien'、'X+Y=' 均为字符串常量。

在 SQL Server 中，字符串常量还可以采用 Unicode 字符串的格式，即在字符串前面用 N 标识，如 N'A SQL Server string' 表示字符串'A SQL Server string'为 Unicode 字符串。

二进制常量具有前缀 0x，并且是十六进制数字字符串，它们不使用引号，如 0xAE、0x12EF、0x69048AEFDD010E、0x（空串）为二进制常量。

2）日期时间常量。datetime 常量使用特定格式的字符日期值表示，用单引号括起来。表 1.10 列出了几种日期时间格式。

表 1.10 SQL Server 日期时间格式

输入格式	datetime 值	smalldatetime 值
Sep 3, 2021 1:34:34.122	2021-09-03 01:34:34.123	2021-09-03 01:35:00
9/3/2021 1PM	2021-09-03 13:00:00.000	2021-09-03 13:00:00
9.3.2021 13:00	2021-09-03 13:00:00.000	2021-09-03 13:00:00
13:25:19	1900-01-01 13:25:19.000	1900-01-01 13:25:00
9/3/2021	2021-09-03 00:00:00.000	2021-09-03 00:00:00

输入时，可以使用“/”“.”“-”作日期时间常量的分隔符。默认情况下，服务器按照 mm/dd/yy 的格式（即月/日/年的顺序）来处理日期类型数据。SQL Server 支持的日期格式有 mdy、dmy、ymd、myd、dym，用 SET DATEFORMAT 命令来设定格式。

对于没有日期的时间值，服务器将其日期指定为 1900 年 1 月 1 日。

3）数值常量。数值常量包括整型常量、浮点常量、精确数值常量、货币常量等。

- 整型常量由没有用引号括起来且不含小数点的一串数字表示，如 1894 和 2 为整型常量。
- 浮点常量主要采用科学记数法表示。如 101.5E5 和 0.5E-2 为浮点常量。
- 精确数值常量由没有用引号括起来且包含小数点的一串数字表示，如 1894.1204 和 2.0 为精确数值常量。
- 货币常量是以$为前缀的一个整型或实型常量数据，不使用引号，如$12.5 和 $542023.14 为货币常量。

4）逻辑数据常量。逻辑数据常量使用数字 0 或 1 表示，并且不使用引号。非 0 的数字当作 1 处理。

5）空值。在数据列定义之后，还需要确定该列是否允许空值（NULL），允许空值意味着用户在向表中插入数据时可以忽略该列值。空值可以表示整型、实型、字符型数据。

（2）变量。变量用于临时存放数据，变量中的值随着程序的运行而改变，变量有名字和数据类型两个属性。变量的命名使用常规标识符，即以字母、下划线（_）、@符号、数字符号（#）开头，后续接字母、数字、@符号、美元符号（$）、下划线（_）的字符序列，不允许嵌入空格或其他特殊字符。SQL Server 将变量分为全局变量和局部变量两类，其中全局变量由系统定义并维护。全局变量在名称前面加@@（两个@），局部变量的首字母为单个@。

1）局部变量。局部变量使用 DECLARE 语句定义，仅存在于声明它的批处理、存储过程或触发器中，处理结束后，存储在局部变量中的信息将丢失。

DECLARE 语句的语法格式：

```
DECLARE @<变量名> <数据类型> [, ...n ]
```

参数说明：

- @<变量名>：指定局部变量的名称。局部变量名必须以@符号开头，且必须符合标识符规则，最大长度为 30 个字符。
- <数据类型>：是任何由系统提供或用户定义的数据类型。用 DECLARE 定义的变量不能是 text、ntext 或 image 数据类型。
- [, ...n]：表示可以定义多个变量，变量之间以逗号分隔。

在使用 DECLARE 语句来声明局部变量时，必须提供变量名称及其数据类型。变量名前必须有一个@符号。一条 DECLARE 语句可以定义多个变量，各变量之间用逗号隔开，例如：

```
DECLARE @name varchar(30), @type int
```

局部变量的值使用 SELECT 或 PRINT 语句显示。局部变量的赋值可以通过 SELECT、UPDATE 和 SET 语句进行。例如，对于上面定义的局部变量@name 和@type，使用以下语句实现赋值与输出：

```
SET @name ='江山'          ' 通过 SET 赋值
SELECT @type =5            ' 通过 SELECT 赋值
PRINT @name                ' 将@name 的值“江山”显示在“消息”窗格
SELECT @type               ' 将@type 的值 5 显示在“结果”窗格
```

2）全局变量。全局变量通常被服务器用来跟踪服务器范围和特定会话期间的信息，不能显式地被赋值或声明。全局变量不能由用户定义，也不能被应用程序用来在处理器之间交叉传递信息。全局变量由系统提供，在某个给定的时刻，各用户的变量值互不相同。表 1.11 列出了 SQL Server 中常用的全局变量。

表 1.11　SQL Server 中常用的全局变量

变　量	说　明
@@rowcount	前一条命令处理的行数
@@error	前一条 SQL 语句报告的错误号
@@trancount	事务嵌套的级别
@@transtate	事务的当前状态
@@tranchained	当前事务的模式（链接的、非链接的）
@@servername	本地 SQL Server 的名称
@@version	SQL Server 和 OS 版本级别
@@spid	当前进程 id
@@identity	上次 INSERT 操作中使用的 identity 值
@@nestlevel	存储过程/触发器中的嵌套层
@@fetch_status	游标中上条 FETCH 语句的状态

3. 函数

函数是一组编译好的 T-SQL 语句，它们可以带一个或一组数值作为参数，也可不带参数，它返回一个数值、数值集合，或执行一些操作。函数能够重复执行一些操作，从而避免重写代码。

SQL Server 支持两种函数类型：内置函数和用户定义函数。

（1）内置函数。内置函数是一组预定义的函数，是 T-SQL 的一部分，按 T-SQL 中定义的方式运行且不能修改。在 SQL Server 中，函数主要用来获得系统的有关信息、执行数学计算和统计、实现数据类型的转换等。SQL Server 提供的函数包括字符串函数、数学函数、日期函数、系统函数等。

（2）用户定义函数。在 SQL Server 中，由用户定义的 T-SQL 函数即为用户定义函数。它将频繁执行的功能语句块封装到一个命名实体中，该实体可以由 T-SQL 语句调用。

4. 运算符

T-SQL 语言运算符共有 5 类，即算术运算符、位运算符、比较运算符、逻辑运算符和连接运算符。

（1）算术运算符。算术运算符用于数值型列或变量间的算术运算。算术运算符包括加（+）、减（-）、乘（*）、除（/）和取模（%）等。表 1.12 列出了所有的算术运算符及其可操作的数据类型。

表 1.12　算术运算符及其可操作的数据类型

算术运算符	数据类型
+、-、*、/	int、smallint、tinyint、numeric、decimal、float、real、money、smallmoney
%	int、smallint、tinyint

如果表达式中有多个算术运算符，则先计算乘、除和求余，然后计算加减。如果表达式中所有算术运算符具有相同的优先顺序，则执行顺序为从左到右。括号中的表达式比所有其他运算都要优先。算术运算的结果为优先级较高的参数的数据类型。

（2）位运算符。位运算符用于对数据进行按位与（&）、或（\）、异或（^）、求反（～）等运算。在 T-SQL 语句中进行整型数据的位运算时，SQL Server 先将它们转换为二进制数，然后再进行计算。其中与、或、异或运算符需要两个操作数，求反运算符仅需要一个操作数。表 1.13 列出了位运算符及其可操作的数据类型。

表 1.13　位运算符及其可操作的数据类型

位运算符	左操作数	右操作数
&	int、smallint、tinyint	int、smallint、tinyint、bigint
\	int、smallint、tinyint	int、smallint、tinyint、binary
^	binary、varbinary、int	int、smallint、tinyint、bit
～	无左操作数	int、smallint、tinyint、bit

1）做&运算时，只有当两个表达式中的两个位都为 1，结果中的位才被设置为 1，否则结果中的位被设置为 0。

2）做\运算时，如果两个表达式的任一位为 1，或者两个位均为 1，则结果的对应位被设置为 1；如果表达式中的两个位都不为 1，则结果中该位被设置为 0。

3）做^运算时，如果在两个表达式中，只有一个位为 1，则结果中该位被设置为 1；如果

两个位都为 0 或者都为 1，则结果中该位被设置为 0。

4）做～运算时，如果表达式的某位为 1，则结果中该位为 0，否则相反。

（3）比较运算符。比较运算符用来比较两个表达式的值，除了 text、ntext 或 image 数据类型外，可用于字符、数字或日期等其他类型的数据运算。SQL Server 中的比较运算符有大于（>）、小于（<）、大于等于（>=）、小于等于（<=）、等于（=）、不等于（!=、<>）、不小于（!<）、不大于（!>）等，通常出现在条件表达式中。

比较运算符的结果为布尔数据类型，其值为 TRUE、FALSE 和 UNKNOWN，如表达式 2=3 的运算结果为 FALSE。一般情况下，带有一个或两个 NULL 表达式的运算符返回 UNKNOWN。当 SET ANSI_ NULLS 为 OFF 且两个表达式都为 NULL 时，那么“=”运算符返回 TRUE。

（4）逻辑运算符。逻辑运算符有与（AND）、或（OR）、非（NOT）等，用于对某个条件进行测试，以获得其真实情况。逻辑运算符和比较运算符一样，返回 TRUE 或 FALSE 的布尔数据值。表 1.14 列出了逻辑运算符及其运算情况。

表 1.14　逻辑运算符及其运算情况

运算符	含义
AND	如果两个布尔表达式都为 TRUE，那么结果为 TRUE
OR	如果两个布尔表达式中的一个为 TRUE，那么结果就为 TRUE
NOT	对任何其他布尔运算符的值取反
LIKE	如果操作数与一种模式相匹配，那么值为 TRUE
IN	如果操作数等于表达式列表中的一个，那么值为 TRUE
ALL	如果一系列的比较都为 TRUE，那么值为 TRUE
ANY	如果一系列的比较中任何一个为 TRUE，那么值为 TRUE
BETWEEN	如果操作数在某个范围之内，那么值为 TRUE
EXISTS	如果子查询包含一些行，那么值为 TRUE

例如，NOT TRUE 为假；TRUE AND FALSE 为假；TRUE OR FALSE 为真。

逻辑运算符通常和比较运算符一起构成更为复杂的表达式。与比较运算符不同的是，逻辑运算符的操作数都只能是布尔型数据。

（5）连接运算符。连接运算符（+）用于两个字符串数据的连接，通常也称为字符串运算符。在 SQL Server 中，对字符串的其他操作通过字符串函数进行。字符串连接运算符的操作数类型有 char、varchar 和 text 等。例如，'Dr.'+'Computer'中的“+”运算符将两个字符串连接成一个字符串'Dr. Computer'。

5. 运算符的优先级别

不同运算符具有不同的运算优先级，在一个表达式中，运算符的优先级决定了运算的顺序。SQL Server 中各种运算符的优先顺序为：

()→～→^→&→\→*、/、%→+、-→NOT→AND →OR。

排在前面的运算符的优先级高于其后的运算符。在一个表达式中，先计算优先级高的运算，后计算优先级低的运算，相同优先级的运算按自左向右的顺序依次进行。

1.6.3 语句块和注释

在程序设计中，往往需要根据实际情况将需要执行的操作设计为一个逻辑单元，用一组T-SQL 语句实现。这就需要使用 BEGIN...END 语句将各语句组合起来。此外，对于程序中的源代码，为了方便阅读或调试，可在其中加入注释。

1. 语句块

用 BEGIN...END 来设定一个语句块，将 BEGIN...END 中的所有语句视为一个逻辑单元执行。

语法格式：

```
BEGIN
 <SQL 语句 | 语句块>
END
```

<SQL 语句 | 语句块>是任何有效的 T-SQL 语句或以语句块定义的语句分组。

在 BEGIN...END 中可嵌套另外的 BEGIN...END 来定义另一程序块。

2. 注释

有两种方法来声明注释：单行注释和多行注释。

（1）单行注释。在语句中，使用两个连字符“--”开头，则从此开始的整行或者行的一部分就成为了注释，注释在行的末尾结束。注释的部分不会被 SQL Server 执行。

（2）多行注释。多行注释方法是 SQL Server 自带的特性，可以注释跨越多行的代码，它必须用一对分隔符“/* */”将余下的其他代码分隔开。

注释并没有长度限制。SQL Server 文档禁止嵌套多行注释，但单行注释可以嵌套在多行注释中。

1.6.4 流程控制语句

T-SQL 提供了一些可以用于改变语句执行顺序的命令，称为流程控制语句。流程控制语句允许用户更好地组织存储过程中的语句，方便程序功能的实现。流程控制语句与常见的程序设计语言类似，主要包含以下几种。

1. 选择控制

根据条件来改变程序流程的控制叫作选择控制。T-SQL 中，IF…ELSE 语句是最常用的控制流语句，CASE 语句可以判断多个条件值，GOTO 语句无条件地改变流程，RETURN 语句会将当前正在执行的批处理、存储过程等中断。

（1）IF…ELSE 条件执行语句。通常是按顺序执行程序中的语句，但在许多情况下，语句执行的顺序和是否执行依赖于程序运行的中间结果。在这种情况下，必须根据条件表达式的值来决定执行哪些语句，这时，利用 IF...ELSE 结构可以实现这种控制。

语法格式：

```
IF <布尔表达式>
    <SQL 语句 | 语句块 1>          --条件表达式为真时执行
[ ELSE
    <SQL 语句 | 语句块 2> ]        --条件表达式为假时执行
```

参数说明：

- <布尔表达式>：是值为 TRUE 或 FALSE 的布尔表达式。
- <SQL 语句 | 语句块>：是 T-SQL 语句或语句块。IF 或 ELSE 条件只能影响一个 T-SQL 语句。若要执行多个语句，则必须使用 BEGIN…END 将其定义成语句块。

IF...ELSE 语句可以嵌套。两个嵌套的 IF…ELSE 语句可以实现 3 个条件分支。

（2）CASE 语句。如果有多个条件要判断，可以使用多个嵌套的 IF...ELSE 语句，但这样会造成程序的可读性差，此时使用 CASE 语句来取代多个嵌套的 IF...ELSE 语句更为合适。CASE 语句具有以下两种格式。

格式 1：简单 CASE 语句，将某个表达式与一组简单表达式进行比较以确定结果。

```
CASE <输入表达式>
    WHEN <表达式> THEN <结果表达式> [ ... n ]
    [ ELSE <结果表达式> ]
END
```

格式 2：CASE 搜索语句，CASE 计算一组逻辑表达式以确定结果。

```
CASE
  WHEN <布尔表达式> THEN <结果表达式> [ ... n ]
  [ ELSE <结果表达式>]
END
```

参数说明：

- <输入表达式>：在简单 CASE 语句中指定所要判断的表达式。
- WHEN <表达式>：在简单 CASE 语句中指定与<输入表达式>进行比较的表达式。
- THEN <结果表达式>：当<输入表达式>等于<表达式>或<布尔表达式>为 TRUE 时，执行 THEN 后面的<结果表达式>。
- ELSE <结果表达式>：当<输入表达式>不等于<表达式>或<布尔表达式>为 FALSE 时，执行 ELSE 后面的<结果表达式>。
- WHEN <布尔表达式>：指定搜索 CASE 格式时所计算的布尔表达式。
- 表明可以使用多个 WHEN 子句。

在格式 2 中，CASE 关键字后面没有表达式，多个 WHEN 子句中的布尔表达式依次执行，如果布尔表达式结果为 TRUE，则执行相应 THEN 关键字后面的表达式，执行完毕后跳出 CASE 语句。如果所有 WHEN 语句的布尔表达式为 FALSE，则执行 ELSE 子句中的表达式。

注意：*所有<结果表达式>的数据类型必须相同，或者必须是隐性转换。*

（3）GOTO 跳转语句。GOTO 语句允许程序的执行转移到标签处，跟随在 GOTO 语句之后的 T-SQL 语句被忽略，而从标签处继续处理，这增加了程序设计的灵活性。但是，GOTO 语句破坏了程序结构化的特点，使程序结构变得复杂且难以测试，建议尽量少使用。

语法格式：

```
GOTO <标签>
```

<标签>为 GOTO 语句处理的起点。<标签>的名称必须符合标识符规则。

（4）RETURN 语句。RETURN 语句可使程序从批处理、存储过程或触发器中无条件退出，不再执行本语句之后的任何语句。

语法格式：

```
RETURN [ <整数表达式> ]
```

<整数表达式>表示返回的整型值，可选项。如果没有指定返回值，SQL Server 系统会根据程序执行的结果返回一个内定状态值，如表 1.15 所示。

表 1.15 RETURN 命令返回的内定状态值

返回值	含义	返回值	含义
0	程序执行成功	-7	资源错误，如磁盘空间不足
-1	找不到对象	-8	非致命的内部错误
-2	数据类型错误	-9	已达到系统的极限
-3	死锁	-10、-11	致命的内部不一致性错误
-4	违反权限原则	-12	表或指针破坏
-5	语法错误	-13	数据库破坏
-6	用户造成的一般错误	-14	硬件错误

（5）WAITFOR 调度执行语句。WAITFOR 语句允许定义一个时间或者一个时间间隔，在定义的时间内或者经过定义的时间间隔时，其后的 T-SQL 语句会被执行。

语句格式：

```
WAITFOR { DELAY '<等待时间>' | TIME '<执行时间>' }
```

参数说明：

- DELAY '<等待时间>'：指定运行批处理、存储过程、事务的时间必须等待的时间，最长可达 24 小时。
- TIME '<执行时间>'：指定运行批处理、存储过程、事务的时间，表示 WAITFOR 语句完成的时间。

2. 循环控制

WHILE 语句根据条件表达式控制 T-SQL 语句或语句块重复执行的次数。条件为真（TRUE）时，在 WHILE 循环体内的 T-SQL 语句会一直重复执行，直到条件为假（FALSE）为止。在 WHILE 语句中可以使用 BREAK 与 CONTINUE 语句跳出循环。

语法格式：

```
WHILE <布尔表达式>
    <SQL 语句 | 语句块>
    [ BREAK | CONTINUE ]
```

参数说明：

- <布尔表达式>：返回值为 TRUE 或 FALSE。如果该表达式含有 SELECT 语句，必须用小括号将 SELECT 语句括起来。
- <SQL 语句 | 语句块>：T-SQL 语句或语句块。语句块定义应使用控制流关键字 BEGIN 和 END。
- BREAK：用于从最内层的 WHILE 循环中退出。将执行出现在 END 关键字后面的任何语句，END 关键字为循环结束标记。
- CONTINUE：使 WHILE 循环重新开始执行，忽略 CONTINUE 关键字后的任何语句。

在 WHILE 循环中，只要<布尔表达式>的值为 TRUE，就会重复执行循环体内的语句或语句块。

一、选择题

1．数据库系统与文件系统的主要区别是（　　）。

A．数据库系统复杂，而文件系统简单

B．文件系统只能管理程序文件，而数据库系统能够管理各种类型的文件

C．文件系统管理的数据量较少，而数据库系统可以管理庞大的数据量

D．文件系统不能解决数据冗余和数据独立性问题，而数据库系统可以解决

2．在关系数据库系统中，当关系的模型改变时，用户程序可以不变，这是因（　　）所致。

A．数据的物理独立性　　B．数据的逻辑独立性

C．数据的位置独立性　　D．数据的存储独立性

3．在数据库三级模式中，对用户所用到的那部分数据的逻辑描述是（　　）。

A．外模式　　B．概念模式

C．内模式　　D．逻辑模式

4．E-R 图用于描述数据库的（　　）。

A．概念模型　　B．数据模型

C．存储模型　　D．逻辑模型

5．以下对关系模型性质的描述中，不正确的是（　　）。

A．在一个关系中，每个数据项不可再分，它是最基本的数据单位

B．在一个关系中，同一列数据具有相同的数据类型

C．在一个关系中，各列的顺序不可以任意排列

D．在一个关系中，不允许有相同的字段名

6．已知两个关系：

职工（职工号，职工名，性别，职务，工资）

设备（设备号，职工号，设备名，数量）

其中“职工号”和“设备号”分别为职工关系和设备关系的关键字，则两个关系的属性中，存在一个外部关键字为（　　）。

A．设备关系的“职工号”　　B．职工关系的“职工号”

C．设备号　　D．设备号和职工号

7．在建立表时，将年龄字段值限制在 18～40 之间，这种约束属于（　　）。

A．实体完整性约束　　B．用户定义完整性约束

C．参照完整性约束　　D．视图完整性约束

8．下列标识符可以作为局部变量使用的是（　　）。

A．[@Myvar]　　B．My var　　C．@Myvar　　D．@My var

9．T-SQL 支持的一种程序结构语句是（　　）。

A．BEGIN…END　　B．IF…THEN…ELSE

C．DO CASE　　D．DO WHILE

10．字符串常量使用（　　）作为定界符。

A．单引号　　B．双引号

C．方括号　　D．大括号

11．SQL Server 中使用的 SQL 是（　　）。

A．ANSI-SQL　　B．PL/SQL

C．T-SQL　　D．MYSQL

12．字符（　　）可以用于 T-SQL 的注释。

A．～～　　B．//

C．/*　*/　　D．@@

13．下列数据类型中，（　　）所占字节最少。

A．tinyint　　B．float

C．char(30)　　D．datetime

14．SQL Server 组织数据采用（　　）。

A．层次模型　　B．网状模型

C．数据模型　　D．关系模型

15．下列属于实体属性的是（　　）。

A．红色　　B．汽车

C．盘子　　D．高铁

二、思考题

1．什么是数据库、数据库管理系统、数据库系统？它们之间有什么联系？

2．当前，主要有哪几种新型数据库系统？它们各有什么特点？用于什么领域？试举例说明。

3．什么是逻辑数据模型？目前数据库主要有哪几种逻辑数据模型？它们各有什么特点？

4．关系数据库中选择、投影、连接运算的含义是什么？

5．关键字段的含义是什么？它的作用是什么？

6．什么是 E-R 图？E-R 图是由哪几种基本要素组成的？这些要素如何表示？

7．简述 char 与 varchar 数据类型之间的区别、char 与 nchar 之间的区别。

8．如何定义局部变量？如何给局部变量赋值？