第1章 数据库导论

在计算机技术迅猛发展、互联网已经渗透到世界各个角落的今天，数据库技术始终处于中心地位。数据库技术主要研究如何存储、使用和管理数据，是计算机数据管理技术发展的新阶段，是计算机技术中发展最快、应用最广的技术之一。

当前，数据库技术已成为现代计算机信息系统和应用系统开发的核心技术，更是未来“信息高速公路”的支撑技术之一。数据库已成为计算机信息系统和应用系统的核心组成。全球顶级软件厂商在数据库领域进行了大量的深入研究，包括微软（SQL Server）、Oracle（Oracle+MySQL）、IBM（DB2）、SAP（Sybase+HANA）、Google（Spanner）、Facebook（RocksDB）、阿里巴巴（OceanBase）、Amazon（Aurora）等。数据库技术已从开始的网络/层次数据库到关系数据库，发展到面向对象数据库、分布式数据库、时序数据库，再到当今的 Nosql（KV 型数据库、文档型数据库、列式数据库、图数据库）与 NewSQL。

1.1 数据库概述

当今社会是一个信息化社会，信息是社会上各行各业的重要资源。数据是信息的载体，数据库是相互关联的数据的集合。数据库能利用计算机来保存和管理大量复杂的数据，快速而有效地为不同的用户和应用程序提供数据，帮助人们利用和管理数据资源，可以说数据库是计算机应用系统中一种专门管理数据资源的系统。

数据有多种形式，如文字、数码、符号、图形、图像及声音等，数据是所有计算机系统所要处理的对象。人们所熟知的一种处理办法是制作文件，即将处理过程编成程序文件，将所涉及的数据按程序要求组成数据文件，再用程序来调用，数据文件与程序文件保持着一定的关系。在计算机应用技术迅速发展的情况下，这种文件式管理方法逐渐显现出不足。例如，它使得数据通用性差、不便于移植、在不同文件中存储大量重复信息、浪费存储空间、更新不便等。而数据库系统能解决上述问题。数据库系统不是从具体的应用程序出发，而是立足于数据本身的管理，将所有数据保存在数据库中进行科学的组织。借助数据库管理系统，各种应用程序或应

用系统能方便地使用数据库中的数据。

百度百科将数据库（Database）定义为“按照数据结构来组织、存储和管理数据的建立在计算机存储设备上的仓库”，是长期储存在计算机内、有组织、可共享的数据集合。它不仅包括数据本身，而且包括相关数据之间的联系。数据库中的数据以一定的数据模型组织、描述和储存在一起，具有尽可能小的冗余度、较高的数据独立性和易扩展性等特点，并可在一定范围内为多个用户所共享。

简单地说，数据库就是一组经过计算机整理后的数据，存储在一个或多个文件中，而管理这个数据库的软件就称为数据库管理系统。

1.1.1 计算机数据管理的发展阶段

随着电子计算机软件和硬件技术的发展，数据处理过程发生了划时代的变革。数据库技术的发展又使数据处理跨入了一个崭新的阶段。数据处理技术的发展大致经历了以下三个阶段。

1. 人工管理阶段

人工管理阶段大约在 20 世纪 50 年代中期以前，这一阶段计算机主要用于科学计算，由于没有必要的软件（没有操作系统和管理数据的软件）、硬件环境的支持，用户只能直接在裸机上操作。用户的应用程序中不仅要设计数据处理方法，还要阐明数据在存储器上的存储方式和地址，此时的数据包含在程序中，如图 1-1 所示。

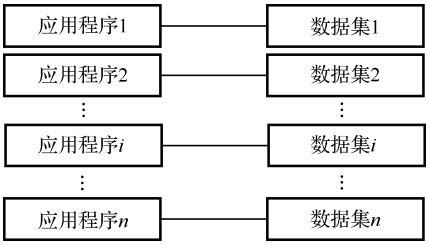

图 1-1　人工管理阶段数据与程序之间的关系

该阶段数据处理的特点：

（1）数据不保存在计算机中。

（2）数据面向程序，数据不能独立，数据和程序结合为一个不可分割的整体。程序依赖于数据，如果数据的类型、格式、输入/输出方式等逻辑结构或物理结构发生变化，必须对应用程序做出相应的修改。

（3）程序员必须对每个应用程序都实现数据的存储结构、存取方法、输入方式等，程序员编写应用程序时还要安排数据的物理存储，因此程序员负担很重。

（4）数据不共享，数据是面向应用的，不同应用的数据之间是相对独立、彼此无关的，即使两个不同应用涉及相同的数据，也必须各自定义。数据不仅高度冗余，而且不能共享。

2. 文件系统阶段

在文件系统阶段，计算机不仅用于科学计算（数据单一），还大量用于信息管理。大量的数据存储、检索和维护成为紧迫需求，出现了磁鼓、磁盘等直接存取数据的存储设备。软件系统中有了初级的操作系统，操作系统中有了专门管理数据的软件，一般称为文件管理系统。文件管理系统是一个独立的系统软件，是应用程序与数据文件之间的一个接口。文件管理系统把

有关的数据组织成一种文件，这种数据文件可以脱离程序而独立存在。在这一管理方式下，应用程序通过文件管理系统对数据文件中的数据进行加工处理。应用程序的数据具有一定的独立性，也比手工管理方式前进了一步，如图 1-2 所示。

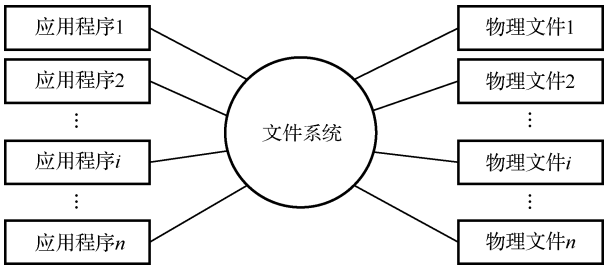

图 1-2　文件系统阶段数据与程序的关系

该阶段数据处理的特点：

（1）数据以文件的形式长期保存，数据可以长期保存在计算机中反复使用。对文件可进行检索、修改、插入、删除等操作。

（2）程序与数据间有一定的独立性，在存储上，数据文件可以脱离应用程序而独立存在，但逻辑上仅供该应用程序使用，即程序和数据间由软件提供的存取方法进行转换，数据存储发生变化不一定影响程序的运行。

（3）文件形式多样化。为了方便数据的存储和查找，人们研究了许多文件类型，有顺序文件、倒排文件、索引文件等。

（4）数据冗余量大，数据文件仍高度依赖于其对应的程序，不能被多个程序所通用。由于数据文件之间不能建立任何联系，因而数据的通用性仍然较差，冗余量大。

3. 数据库系统阶段

20 世纪 60 年代后期，计算机应用于管理的规模更加庞大，数据量急剧增加；计算机性能得到提高，更重要的是出现了大容量磁盘，存储容量大大增加且价格下降。在此基础上，有可能克服文件系统管理数据的不足，满足和解决实际应用中多个用户、多个应用程序共享数据的要求，使数据能为尽可能多的应用程序服务，从而出现了数据库。数据库的特点是数据不再只针对某一特定应用，而是面向全组织，具有整体的结构性，共享性高，因此冗余度小，具有一定的程序与数据间的独立性，并且实现了对数据进行统一的控制，如图 1-3 所示。

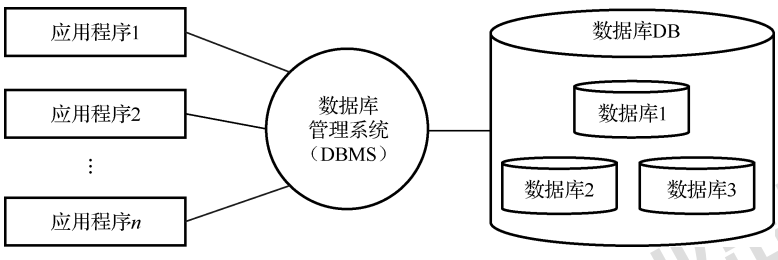

图 1-3　数据库系统阶段应用程序与数据之间的关系

该阶段数据处理的特点：

（1）数据共享是数据库系统区别于文件系统的最大特点之一，也是数据库系统技术先进性的重要体现。共享是指多用户、多种应用程序、多种语言互相覆盖地共享数据集合。

（2）面向全组织的数据结构化。数据库系统不再像文件系统那样从属于特定的应用，而是面向整个组织来组织数据，常常是按照某种数据模型，将整个组织的全部数据组织成为一个结构化的数据整体。它不仅描述了数据本身的特性，也描述了数据与数据之间的种种联系，使数据库能够描述复杂的数据结构。

（3）数据独立性。数据库技术的重要特征就是数据独立于应用程序而存在，数据与程序相互独立、互不依赖，不因一方的改变而改变另一方，从而大大简化了应用程序的设计与维护的工作量。

（4）可控数据冗余度。数据共享、结构化和数据独立性的优点使数据存储不必重复，不仅可以节省存储空间，而且从根本上保证了数据的一致性，这是有别于文件系统的重要特征。

（5）统一数据控制功能。数据库是系统中各用户的共享资源，因而计算机的共享一般是并发的，即多个用户同时使用数据库。因此，系统必须提供数据安全性控制、数据完整性控制、并发控制和数据恢复等功能。

1.1.2 数据库系统

数据库系统是由数据库及其管理软件组成的系统，是指在计算机系统中引入数据库后的系统，其构成部分主要有数据库、操作系统、数据库管理系统、应用开发工具、应用系统、数据库管理员和用户几部分。其中，数据库的建立、使用和维护的过程要由专门的人员来完成，这些人被称为数据库管理员（DataBase Administrator，DBA）。数据库系统结构如图 1-4 所示。

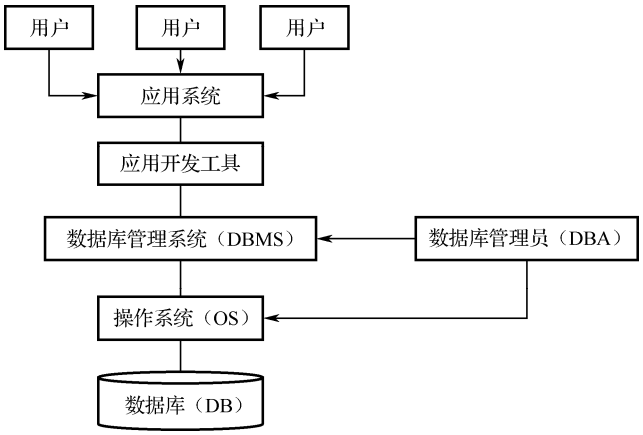

图 1-4　数据库系统结构

1. 数据库（DataBase，DB）

数据库（DB）是指数据库系统中集中存储的相互关联的数据集合。

它可以供用户共享，具有尽可能小的冗余度和较高的数据独立性，使得数据存储最优，数据最容易操作，并且具有完善的自我保护能力和数据恢复能力。

为了与输入、输出或中间数据加以区别，常把数据库数据称为“存储数据”、“工作数据”或“操作数据”。它们是某特定应用环境中进行管理和决策所必需的信息。

2. 用户（Users）

数据库系统中存在一组使用数据库的用户，即指存储、维护和检索数据的各类请求。主要有三类用户：终端用户、应用程序员和数据库管理员。

（1）终端用户也称最终用户、联机用户，是指从计算机联机终端存取数据库的人员。该用户使用数据库系统提供的终端命令语言、表格语言或菜单驱动等交互式对话方式来存取数据库中的数据。终端用户一般是不精通计算机和程序设计的各级管理人员、工程技术人员或各类科研人员。

（2）应用程序员也称为系统开发员，是指负责设计和编制应用程序的人员。该用户通常使用 VFP、PB 或 Oracle 等数据库语言设计和编写使用及维护数据库的应用程序来存取和维护数据库。

（3）数据库管理员（DBA），是指全面负责数据库系统的管理、维护和正常使用的人员。它可以是一个人或一组人。特别是对于大型数据库系统，数据库管理员极为重要，常设置有数据库管理员办公室，应用程序员是数据库管理员手下的工作人员。担任数据库管理员，不仅要具有较高的技术专长，还要具备较深的资历，并具有了解和阐明管理要求的能力。

数据库管理员的主要职责有：参与数据库设计的全过程，与用户、应用程序员、系统分析员紧密结合，设计数据库的结构和内容；决定数据库的存储与存取策略，使数据的存储空间利用率和存取效率均较优；定义数据的安全性和完整性；监督和控制数据库的使用和运行，及时处理运行程序中出现的问题；改进和重新构造数据库系统；等等。

3. 软件

软件是指负责数据库存取、维护和管理的软件系统，通常叫作数据库管理系统（Data Base Management System，DBMS）。数据库系统中各类用户对数据库的各种操作请求都是由 DBMS 来完成的，它是数据库系统的核心软件。DBMS 提供一种硬件层之上的对数据库的观察功能，并支持用较高层次的观点来表达用户的操作，使数据库用户不受硬件层细节的影响。DBMS 是在操作系统（OS）的支持下工作的。

4. 硬件

硬件是指存储数据库和运行数据库管理系统（包括操作系统）的硬件资源。它包括物理存储数据库的磁盘、磁鼓、磁带或其他外存储器及其附属设备、控制器、I/O 通道、内存、CPU 及其他外部设备等。

DBMS 在整个计算机层次结构中的地位如图 1-5 所示。

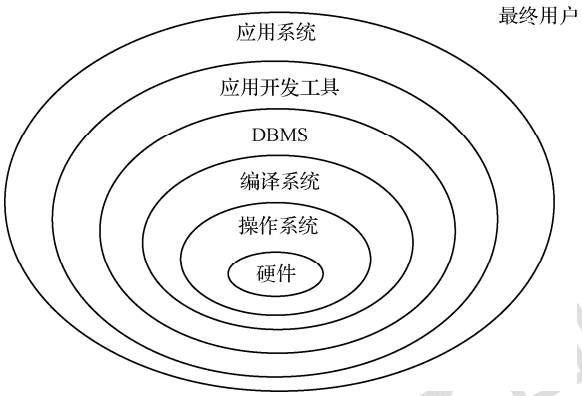

图 1-5　DBMS 在整个计算机层次结构中的地位

1.1.3 数据库管理系统

数据库管理系统（Database Management System，DBMS）是数据库系统的核心组成部分，是一种操纵和管理数据库的软件，用于建立、使用和维护数据库。用户通过 DBMS 访问数据库中的数据，数据库管理员也通过 DBMS 进行数据库的维护工作。用户在数据库系统中的一些操作，如数据定义、数据操作、数据库的运行管理及数据控制等都是通过数据库管理系统来实现的，如图 1-6 所示。

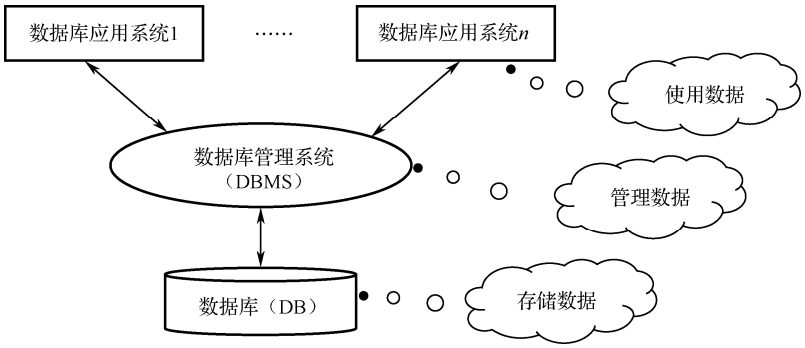

图 1-6　数据库管理系统

1. DBMS 的主要功能

DBMS 就是实现把用户意义下的抽象的逻辑数据处理通过三级模式转换成计算机中的具体的物理数据处理的软件，给用户带来很大的方便。

DBMS 是由许多“系统程序”所组成的一个集合。每个程序都有自己的功能，共同完成 DBMS 的一件或几件工作。

通常，DBMS 的主要功能包括以下 5 个方面。

（1）数据库定义。DBMS 提供相应的数据定义语言来定义数据库结构，刻画数据库的框架，并保存在数据字典中。数据字典是 DBMS 存取和管理数据的基本依据。

（2）数据存取。DBMS 提供数据操作语言来实现对数据库数据的基本存取操作：检索、插入、修改和删除。

（3）数据库运行管理。DBMS 提供数据控制功能，即数据的安全性、完整性和并发控制等，对数据库运行进行有效的控制和管理，以确保数据库数据正确有效和数据库系统的有效运行。

（4）数据库的建立和维护。其包括数据库初始数据的装入，数据库的转储、恢复、重组织，系统性能监视、分析等功能。这些功能大都由 DBMS 的实用程序来完成。

（5）数据通信。DBMS 提供处理数据的传输功能，实现用户程序与 DBMS 之间的通信。其通常与操作系统协作完成。

2. DBMS 的组成

DBMS 大多是由许多“系统程序”组成的一个集合。每个程序都有自己的功能，一个或几个程序一起完成 DBMS 的一件或几件工作。各种 DBMS 的组成因系统而异，一般说来，它由以下几个部分组成。

（1）语言编译处理程序，主要包括：数据描述语言翻译程序、数据操作语言处理程序、终端命令解释程序、数据库控制命令解释程序等。

（2）系统运行控制程序，主要包括：系统总控程序、存取控制程序、并发控制程序、完整性控制程序、保密性控制程序、数据存取和更新程序、通信控制程序等。

（3）系统建立、维护程序，主要包括：数据装入程序、数据库重组织程序、数据库系统恢复程序、性能监督程序等。

（4）数据字典。数据字典通常是一系列表，它存储着数据库中有关信息的当前描述。它能帮助用户、数据库管理员和数据库管理系统本身使用和管理数据库。

1.1.4　数据库系统结构

数据库系统有着严谨的体系结构。虽然各个厂家、各个用户使用的数据库管理系统产品类型和规模可能相差很大，但它们在体系结构上通常都具有相同的特征，即采用三级模式结构，并提供两种映像功能。

1. 数据库系统的三级模式

从数据库管理角度看，数据库系统通常采用三级模式结构，这是数据库系统内部的结构；从数据库最终用户角度看，数据库系统的结构分为集中式结构、分布式结构、客户/服务器结构，这是数据库系统外部的结构。

数据库系统的三级模式结构是指数据库系统由外模式、概念模式、内模式构成，如图 1-7 所示。

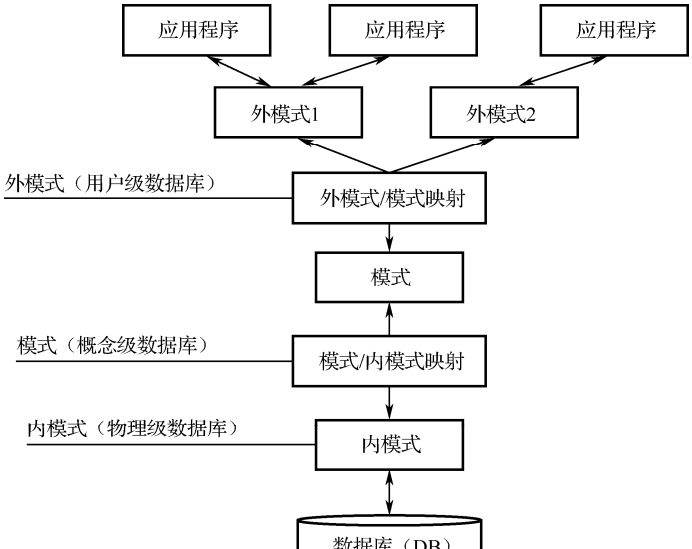

图 1-7　数据库系统的三级模式结构

1）内模式

内模式也称为存储模式，一个数据库只有一个内模式，它是整个数据库的底层表示。内模式是数据物理结构和存储方式的描述，是数据在数据内部的表示方式。

内模式定义的是存储记录的类型、存储记录的物理顺序、索引和存储路径等数据的存储组织。DBMS 提供了内模式描述语言来严格定义内模式。

2）模式

模式也称为逻辑模式或概念模式，是数据库全体数据的逻辑结构和特性的描述，是所有用户的公共数据视图。它仅仅涉及型的描述，不涉及具体的值。模式的一个具体值称为一个实例，同一个模式可以有很多实例。模式是相对稳定的，而实例是相对变动的，因为数据库中的数据是在不断更新的。模式反映的是数据的结构及其联系，而实例反映的是数据库某一时刻的状态。它是数据库模式结构的中间层，既不涉及数据的物理存储细节和硬件环境，也与具体的应用程序、所使用的应用开发工具及高级程序设计语言无关。

模式实际上是数据库在逻辑上的视图，一个数据库只有一个模式。定义模式时，不仅要定义数据的逻辑结构还要定义数据记录由哪些数据项组成，以及数据项的名字、类型、取值范围等，而且要定义数据之间的联系，定义与数据有关的安全性、完整性要求。

3）外模式

外模式也称为子模式或用户模式，它是数据库用户（包括应用程序员和最终用户）能够看见和使用的局部数据的逻辑结构和特征的描述，是数据库用户的数据视图，是与某一应用有关的数据的逻辑表示。

外模式一般是模式的子集。一个数据库可以有多个外模式。这样，不同的用户通过不同的外模式实现各自的数据视图，也能达到共享数据的目的。同一外模式可以被一个用户的多个应用程序所使用，但一个应用程序只能使用一个外模式。

外模式是保证数据库安全性的一个有力措施。每个用户只能看见和访问所对应的外模式中的数据，数据库的其余数据对该用户而言是不可见的。

2. 数据库的两级映射

数据库系统的三级模式是对数据的三个抽象级别，它把数据的具体组织留给 DBMS 管理，使用户能逻辑地、抽象地处理数据，而不必关心数据在计算机中的具体表示方式与存储方式。为了能够在内部实现这三个抽象层次的联系和转换，DBMS 在三级模式之间提供了两层映射。

1）外模式/模式映射

对应于同一个模式可以有任意多个外模式。它定义了某一个外模式和模式之间的对应关系，这些映射定义通常包含在各自的外模式中，当模式改变时，该映射要做相应的改变，以保证外模式保持不变，实现了数据与程序的逻辑独立性。

2）模式/内模式映射

它定义了数据逻辑结构和存储结构之间的对应关系，说明了逻辑记录和字段在内部是如何表示的。当数据库的存储结构改变时，可相应地修改该映射，从而使模式保持不变，保证了数据与程序的物理独立性。

正是这两层映射保证了数据库系统中的数据具有较高的逻辑独立性和物理独立性。

1.2 概念模型和数据模型

由于计算机不能直接处理现实世界中的具体事物，所以人们必须将具体事物转换成计算机能够处理的数据。在数据库中用数据模型来抽象、表示和处理现实世界中的数据。数据模型是描述数据、数据之间联系的结构模式。数据模型是数据库系统中用于提供信息表示和操作手段的形式构架。不同的数据模型提供了模型化数据和信息的不同工具，根据模型应用的不同目的，

可以将模型分为两类或两个层次：一个是概念模型（也称信息模型）；另一个是数据模型（层次模型、网状模型和关系模型）。前者按用户的观点对数据和信息建模，后者按计算机系统的观点对数据建模。

1.2.1　概念模型

数据库即是模拟现实世界中某应用环境（一个企业、单位或部门）所涉及的数据的集合，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。这个集合或者包含了信息的一部分（用用户视图模拟），或者包含了信息的全部（用概念视图模拟），而这种模拟是通过数据模型来实现的。

为了把现实世界中的具体事物抽象、组织为某一 DBMS 支持的数据模型，在实际的数据处理过程中：

（1）首先将现实世界的事物及联系抽象成信息世界的概念模型（信息模型）。

（2）然后抽象成计算机世界的数据模型。

概念模型并不依赖于具体的计算机系统，不是某一个 DBMS 所支持的数据模型，它是计算机内部数据的抽象表示，是概念模型；概念模型经过抽象，转换成计算机上某一 DBMS 支持的数据模型，概念模型的建立为现实世界过渡到计算机世界奠定了基础。

在数据处理中，数据加工经历了现实世界、信息世界和计算机世界三个不同的世界，经历了两级抽象和转换，如图 1-8 所示。

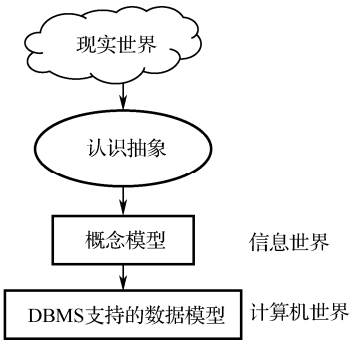

图 1-8　数据处理的抽象和转换过程

所以说，数据模型是现实世界的两级抽象的结果。

现实世界是存在于人们头脑之外的客观世界，现实世界中的事物及其相互间的联系反映到人们头脑中来，经过人们头脑的认识、选择、命名、分类等抽象工作之后，形成一些基本概念与关系，构成信息世界。信息世界是现实世界在人们头脑中的反映和抽象，它介于现实世界与计算机世界之间，起着承上启下的作用。现实世界的事物在信息世界中被抽象为“实体（Entity）”。

在信息世界中，常用的主要概念如下。

1. 实体（Entity）

实体：客观存在并且可以相互区别的“事物”称为实体。

实体可以是可触及的对象，如一名学生、一本书、一辆汽车；也可以是抽象的事件，如一堂课、一场比赛等。

2. 实体集（Entity Set）

实体集：不同值的同类型实体的集合称为实体集。

例如，所有的学生、所有的比赛等。

3. 属性（Attributes）

属性：实体的某一特性称为属性。一个实体可用若干个属性来刻画。也可以说，若干个属性值所组成的集合可表征一个实体（个体）。

例如，学生实体有学号、姓名、年龄、性别、系等方面的属性。

对于一个对象，它究竟具有哪些属性？这完全取决于考虑问题的角度。

例如，一个人，在人口统计中，关心他的姓名、年龄、性别、文化程度等属性；而在工资处理中，关心他的工资、奖金、水电费、房租等属性。

4. 实体和属性的型与值

1）属性的型与值

属性有“型”和“值”之分。

属性的型为属性名，如学号、姓名、年龄、性别、系是属性的型。

属性的值为属性的具体内容，如 990001、张立、20、男、计算机，这些属性值的集合表示了一个学生的实体值。

2）实体的型与值

实体的型（Entity Type）：若干个属性型组成的集合可以表示一个实体的型，简称实体型。

例如，学生（学号，姓名，年龄，性别，系）就是一个实体型。

实体的值：不同的实体有不同的属性内容（即属性值），如（990001，张立，20，男，计算机），这些属性值的集合表示了一个学生的实体值。

5. 联系（Relationship）

在现实世界中，事物内部及事物之间是有联系的，这些联系同样也要抽象和反映到信息世界中来，在信息世界中将被抽象为实体型内部之间的联系（即属性间的联系）和各种实体型之间的联系（也称实体之间的联系）。

1）一对一联系（1∶1）

如果对于实体集 *A* 中的每一个实体，实体集 *B* 中至多有一个与之相对应；反之，实体集 *B* 中的每一个实体，实体集 *A* 中也至多有一个实体与之对应，则称实体集 *A* 与实体集 *B* 具有一对一联系，记作 1∶1。例如，每个班级只有一个班长，班长和班级之间是一对一联系。

2）一对多联系（1∶*n*）

如果对于实体集 *A* 中的每一个实体，实体集 *B* 中有 *n* 个实体与之相对应；反之，如果对于实体集 *B* 中的每一个实体，实体集 *A* 中最多只有一个实体与之相对应，则称实体集 *A* 与实体集 *B* 具有一对多联系，记作 1∶*n*。例如，每个学生只能属于一个班级，每个班级可以有多名学生，班级和该班级中的学生之间是一对多联系。

3）多对多联系（*m*∶*n*）

如果对于实体集 *A* 中的每一个实体，实体集 *B* 中有 *n*（*n*≥0）个实体与之相对应；反之，如果对于实体集 *B* 中的每一个实体，实体集 *A* 也有 *m*（*m*≥0）个实体与之相对应，则称实体集 *A* 与实体集 *B* 具有多对多的联系，记作 *m*∶*n*。例如，每个教师可以上多门课程，每门课程又可以被多名教师授课，课程与教师之间是多对多联系。

两个实体型之间的三类联系如图 1-9 所示。

建立概念模型，就是用模型表示实体型及实体间的联系，概念模型的表示方法很多，其中最著名、最常用的是 P. P. S. Chen 于 1976 年提出的实体—联系方法（Entity-Relationship Approach）。该方法用 E-R 图描述现实世界的概念模型。

E-R 图中的 E 是英文单词 Entity 的缩写，表示实体的意思。这里所说的实体可以理解为现实世界中的事物，如高等院校中的院系、教师等。E-R 图中的 R 是英文单词 Relationship 的缩写，表示关系的意思。这里所说的关系可以理解为实体与实体之间的相互联系，如高等院校中院系与教师之间的相互联系。在 E-R 图中还涉及的一个概念是属性，它用来描述实体的特征。

例如，高等院校中院系的编号、名称；教师的姓名、编号、工资、所在院系等。

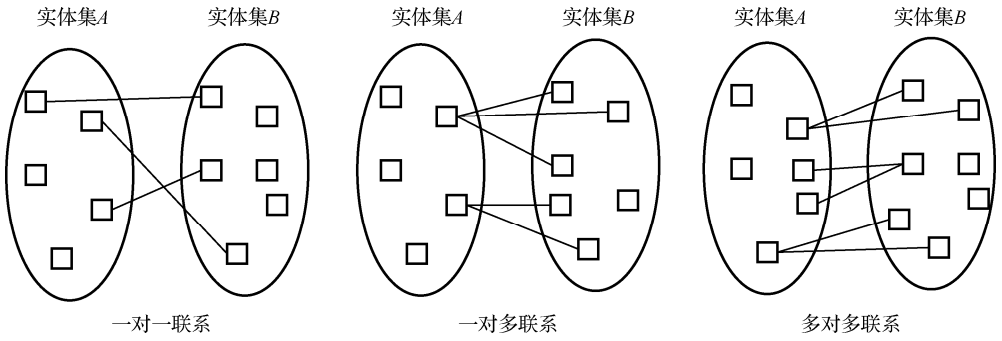

图 1-9　两个实体型之间的三类联系

在 E-R 图中共有三种符号：矩形、椭圆形（或者圆形）和菱形。其中，矩形表示实体，椭圆形或圆形表示属性，菱形表示关系。下面来看一下如何使用 E-R 图描述上面讲到的三种联系。

一对一联系（1∶1）：在高等院校中，校长和学校的关系就是一对一联系。每一个学校只有一名校长，一名校长只能管理一个学校。

图 1-10 描述了学校和校长的一对一联系。其中，矩形中的学校和校长表示实体；椭圆形中的学校编号、学校名称表示实体学校的属性，校长姓名、校长年龄、校长性别表示实体校长的属性；菱形中的管理表示学校和校长之间的关系，即学校是由校长来管理的。

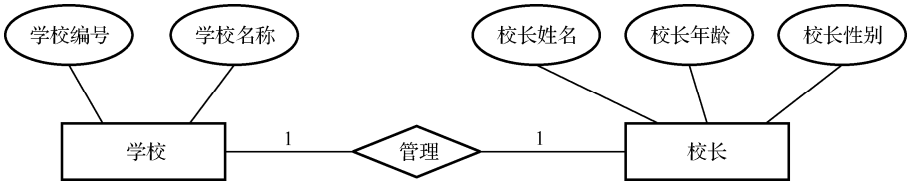

图 1-10　学校和校长的一对一联系

一对多联系（1∶n）：在高等院校中，院系和学生之间就是一对多联系。一个院系可以对应多个学生，而每一个学生只是其中某一个院系中的成员。

图 1-11 描述了学生和院系的一对多联系。其中，矩形中的院系和学生表示实体；椭圆形中的院系编号、院系名称表示实体院系的属性，学生编号、学生姓名、学生年龄表示实体学生的属性；菱形中的属于表示院系和学生之间的关系，即学生是属于某一个院系的。

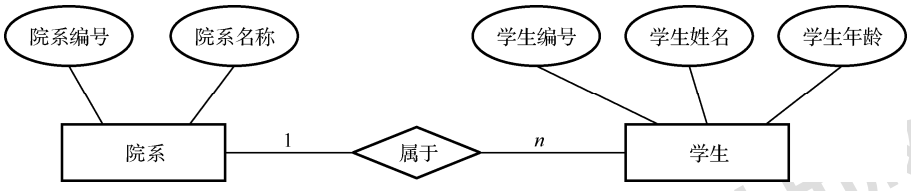

图 1-11　学生和院系的一对多联系

多对多联系（n∶m）：在高等院校中，课程与授课教师之间就是多对多联系。一门课程可以由几个不同的教师来讲授，一名教师也可以讲授多门不同的课程。

图 1-12 描述了教师和课程的多对多联系。其中，矩形中的课程和教师表示实体；椭圆形

中的课程编号、课程名称表示实体课程的属性，教师编号、教师姓名、教师职称表示实体教师的属性；菱形中的授课表示课程和教师之间的关系，即课程是由教师来讲授的。

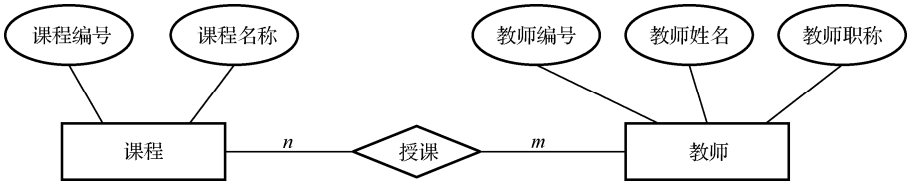

图 1-12　教师和课程的多对多关系

1.2.2　数据模型

数据世界也称计算机世界，它是现实世界中的客观事物及其联系经过信息抽象而在计算机中的表现形式。信息世界中的“实体”经过加工、编码抽象为计算机世界中的数据，存储在计算机中，即信息数据化。

在计算机世界中，概念模型被抽象为数据模型。数据模型是概念模型的数据化，实体型内部的联系抽象为同一记录内部各字段间的联系，实体型之间的联系抽象为记录与记录之间的联系。

在计算机世界中，常用的主要概念如下：

（1）字段（Field）。对应于属性的数据称为字段，也称为数据项。字段名往往和属性名相同。

例如，学生有学号、姓名、年龄、性别、系等字段。

（2）记录（Record）。对应于每个实体的数据称为记录。

例如，一个学生（990001，张立，20，男，计算机）为一条记录。

（3）文件（File）。对应于实体集的数据称为文件。

例如，所有学生的记录组成了一个学生文件。

现实世界是设计数据库的出发点，也是使用数据库的最终归宿。实体（概念）模型和数据模型是现实世界事物及其联系的两级抽象，如图 1-13 所示。而数据模型是实现数据库系统的根据。

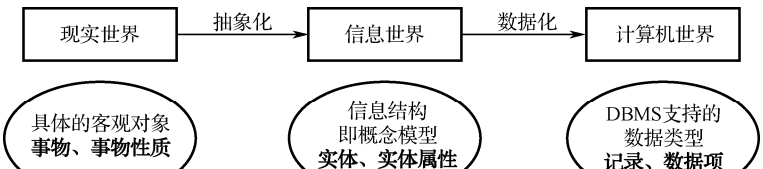

图 1-13　3 个世界的联系

概念模型是对信息世界的模型表示，而数据模型是对计算机世界的模型表示。在数据库中是用数据模型对现实世界进行抽象的，现有的数据库系统均是基于某种数据模型的。数据模型的好坏直接影响数据库的性能，数据模型的设计方法决定着数据库的设计方法。

数据库中最常见的数据模型有 3 种：层次模型（Hierarchical Model）、网状模型（Network Model）和关系模型（Relational Model）。

这 3 种数据模型的根本区别在于数据结构不同，即数据之间联系的表示方式不同。

（1）层次模型用“树结构”来表示数据之间的联系。

（2）网状模型用“图结构”来表示数据之间的联系。

（3）关系模型用“二维表”来表示数据之间的联系。

1. 层次模型

层次模型是数据库系统中最早出现的数据模型，采用层次模型的数据库的典型代表是 IBM 公司的数据库管理系统（Information Management System，IMS）。在现实世界中，许多实体之间的联系都表现出一种很自然的层次关系，如家族关系、行政机构等。

层次模型具有如下特征：

（1）有且仅有一个结点无父（双亲）结点，这个结点称为根结点。

（2）其他结点有且仅有一个父结点。

任何一个给定的记录值只有按其路径查看时，才能显出它的全部意义，没有一个子女记录值能够脱离双亲记录值而独立存在。

在层次模型中，结点层次从根开始定义，根为第一层，根的子结点为第二层，根为其子结点的父结点，同一父结点的子结点称为兄弟结点，没有子结点的结点称为叶结点。

在图 1-14 所示的抽象层次模型中，R1 为根结点；R2 和 R3 为兄弟结点，并且是 R1 的子结点；R4 和 R5 为兄弟结点，并且是 R2 的子结点；R3、R4 和 R5 为叶结点。其中，每个结点代表一个实体型，各实体型由上而下是 1∶*n* 联系。

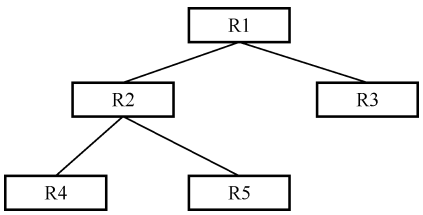

图 1-14　抽象层次模型

在树中，每个结点表示一个记录类型（实体型），结点间的连线表示记录类型间的关系，每个记录类型可包含若干字段，记录类型描述的是实体，字段描述实体的属性，各个记录类型及其字段都必须命名。

如果要存取某一记录类型的记录，可以从根结点起，按照有向树层次向下查找。

例如，下面是一个层次模型的例子，如图 1-15 所示。

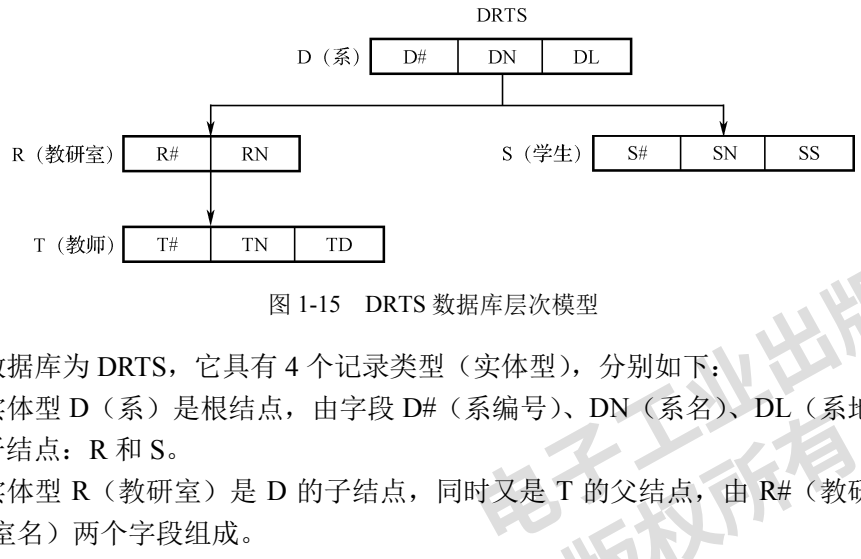

图 1-15　DRTS 数据库层次模型

层次数据库为 DRTS，它具有 4 个记录类型（实体型），分别如下：

（1）实体型 D（系）是根结点，由字段 D#（系编号）、DN（系名）、DL（系地点）组成，它有两个子结点：R 和 S。

（2）实体型 R（教研室）是 D 的子结点，同时又是 T 的父结点，由 R#（教研室编号）、RN（教研室名）两个字段组成。

（3）实体型 S（学生）由 S#（学号）、SN（姓名）、SS（成绩）3 个字段组成。

（4）实体型 T（教师）由 T#（职工号）、TN（姓名）、TD（研究方向）3 个字段组成。

S 与 T 是叶结点，它们没有子结点，由 D 到 R，R 到 T，由 D 到 S 均是一对多联系。

对应上述数据模型的一个值：该值是 D002 系（计算机系）记录值及其所有后代记录值组成的一棵树，D002 系有 3 个教研室子记录值（R001、R002、R003）和 3 个学生记录值（S63871、S63874、S63876），教研室 R001 有 3 个教师记录值（T2101、T17090、T3501）。

层次模型结构清晰，各结点之间联系简单，只要知道每个结点（除根结点以外）的双亲结点，就可知道整个模型结构，因此，画层次模型时可用无向边代替有向边。用层次模型模拟现实世界的层次结构的事物及其之间的联系是很自然的选择方式，如表示“行政层次结构”“家族关系”等。但层次模型的缺点是不能表示两个以上实体型之间的复杂联系和实体型之间的多对多联系。

2. 网状模型

在现实世界中，事物之间的联系更多地是非层次关系的。用层次模型表示非树形结构是很不直接的，网状模型则可以克服这一弊病，是用网络结构表示实体类型及其实体之间联系的模型。网状模型的典型代表是 DBTG（Data Base Task Group）系统，也称 CODASYL（Conference On Data Systems Language）系统。如图 1-16 所示为网状模型。

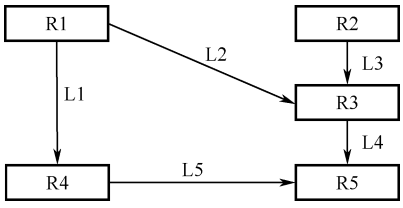

图 1-16　网状模型

在图 1-16 中，R1 与 R4 之间的联系被命名为 L1，R1 与 R3 之间的联系被命名为 L2，R2 与 R3 之间的联系被命名为 L3，R3 与 R5 之间的联系被命名为 L4，R4 与 R5 之间的联系被命名为 L5。R1 为 R3 和 R4 的父结点；R2 也是 R3 的父结点；R1 和 R2 没有父结点；R5 为 R3 和 R4 的子结点。

若用图来表示，网状模型是一个网络。在数据库中，满足以下两个条件的数据模型称为网状模型。

（1）允许一个以上的结点无父结点。

（2）至少有一个结点有多于一个的父结点。

网状模型中每个结点表示一个记录类型（实体型），每个记录类型可包含若干字段（实体的属性），结点间的连线表示记录类型（实体型）间的父子关系。

网状模型和层次模型在本质上是一样的。从逻辑上看，它们都是基本层次联系的集合，用结点表示实体，用有向边（箭头）表示实体间的联系；从物理结构上看，它们的每一个结点都是一个存储记录，用连接指针来实现记录之间的联系。当存储数据时这些指针就固定下来了，检索数据时必须考虑存取路径问题；更新数据时，涉及连接指针的调整，缺乏灵活性；系统扩充相当麻烦。网状模型中的指针更多，纵横交错，从而使数据结构更加复杂。

3. 关系模型

关系模型是数据库领域中最重要的一种数据模型。自 20 世纪 80 年代以来，计算机厂商推出的 DBMS 几乎都支持关系模型。非关系模型数据库系统产品也大都加上了关系接口。目前的数据库领域的研究工作也都是以关系方法为基础的。

关系模型要求关系必须是规范化的，即要求关系必须满足一定的规范条件。其中最基本的一条是：关系的每一个分量必须是一个不可分的数据项，即不允许表中还有表。

在关系模型中，实体和实体间的联系都是用关系来表示的。例如，学生、课程、学生与课

程之间的多对多联系在关系模型中可以表示如下：

学生（学号，姓名，性别，年龄，系，年级）；

课程（课程号，课程名，学分）；

选修（学号，课程号，成绩）。

关系模型与非关系模型不同，它是建立在严格的数学概念的基础上的。为了便于理解，这里主要讨论按照用户观点所应了解的关系模型。在用户看来，关系模型中数据结构就是一张二维表（如表 1-1、表 1-2 所示）。

表 1-1　学生登记表

学　号	姓　名	性　别	年　龄	系　号	年　级
950104	王小明	女	19	01	95
950206	黄大鹏	男	20	02	95
950508	张文斌	女	18	05	95
……	……	……	……	……	……

表 1-2　系信息表

系　号	系　名	办 公 室	主　任	电　话
01	计算机	教 209	张立	5585021
02	物理	教 501	李可	2334102
……	……	……	……	……
05	地质工程	教 301	陈鹏	5585206

表 1-1、表 1-2 是一张学生登记表和一张系信息表，它们由行和列组成。以上面两表为例，介绍关系模型中的一些术语如下。

（1）关系（Relation）：一个关系对应通常所说的一张二维表。

（2）元组（Tuple）：表中的一行即为一个元组。

（3）属性（Attribute）：表中的一列即为一个属性，给每一个属性起一个名称即属性名。表 1-1 有 6 列，对应 6 个属性（学号、姓名、性别、年龄、系号和年级）。

（4）域（Domain）：属性的取值范围，所以又称“值域”。

（5）分量：元组中的一个属性值。

（6）关系模式：对关系的描述，一般表示为：关系名（属性 1，属性 2，…，属性 n）。

（7）关键字或码（Key）：表中用来唯一确定（标识）一个元组的某个属性或属性组合，如表中学号。关键字必须唯一，但它的唯一性不是只对关系的当前元组构成来确定的，还要考虑元组构成的将来可能性，如表中姓名。一个关系中，关键字的值不能为空，即关键字的值为空的元组在关系中是不允许的。

① 候选关键字（Candidate Key）或候选码：如果一个关系中存在多个属性或属性组合都能用来唯一标识该关系的元组，则这些属性或属性组合都称为该关系的候选关键字或候选码。

② 主关键字（Primary Key）或主码：在一个关系的若干候选关键字中，指定作为关键字的属性或属性组合称为该关系的主关键字或主码。

③ 非主属性（Non Primary Attribute）或非码属性：关系中不组成码的属性均为非主属性

或非码属性。

④ 外部关键字（Foreign Key）或外键：当关系中某个属性或属性组合虽不是该关系的关键字或只是关键字的一部分，却是另外一个关系的关键字时，称该属性或属性组合为这个关系的外部关键字或外键。

⑤ 主表与从表：指以外键相关联的两个表；以外键作为主键的表称为主表，外键所在的表称为从表。

1）关系模型的操作

（1）操作：查询、插入、删除、修改。

前一种为检索，后三种为更新，关系模型数据操作的特点如下：

① 集合操作，操作对象和操作结果都是关系，即若干元组的集合。

② 存取路径对用户隐蔽，用户只要指出："干什么"或"找什么"，不必考虑"怎么找（干）"。

③ 存取路径由 RDBMS 自动选择，方便了用户，提高了数据的独立性。

（2）关系数据操作的理论标准为关系代数或关系演算。其中，关系演算又分为元组关系演算和域关系演算两种。关系代数、元组关系演算和域关系演算 3 种抽象语言在表达能力上是完全等价的。

（3）介于关系代数和关系演算之间的实用的代表性的关系操作语言是 SQL（Structured Query Language）。

2）完整性约束条件

完整性包括实体完整性、参照完整性、用户定义的完整性。

（1）实体完整性（Entity Integrity）。若属性 A 是基本关系 R 的一个主属性，则任何元组在 A 上的分量都不能为空。

实体完整性规定：主码的任何属性都不能为空。这是因为：

① 一个基本关系通常对应概念模型中的一个实体集或联系。

② 概念模型中的实体及联系都是可区分的，它们有某种唯一性标识，称为"码"。

③ 主码不能取空值。若取空值，则表明存在一个不以"码"为唯一性标识的实体。

（2）参照完整性（Referential Integrity）。若属性组 A 是基本关系 R1 的外码，它与基本关系 R2 主码 K 相对应，则 R1 中每个元组在 A 上的值必须为以下两种情况之一：

① 等于 R2 中某元组的主码值。

② 取空值（A 的每个属性值都是空值）。

例如，职工关系（职工编号，姓名，性别，年龄，身份证号码，部门编号）；部门关系（部门编号，部门名称，部门经理）。

参照完整性是对关系间引用数据的一种限制。

（3）用户定义的完整性。用户定义的完整性约束条件是某一具体数据库的约束条件，是用户自己定义的某一具体数据必须满足的语义要求。关系模型的 DBMS 应提供给用户定义它的手段和自动检验它的机制，以确保整个数据库始终符合用户所定义的完整性约束条件。

例如，职工关系（职工编号，姓名，性别，年龄，身份证号码，部门编号）。

3）关系模型的存储结构

在关系模型中，实体及实体间联系以"表"来表示，在数据库的物理组织中，表以文件形式存储。一个数据表对应一个操作系统文件。文件结构由系统自己设计。

4）关系模型的优缺点

优点：

（1）建立在严格的数学概念的基础上。

（2）概念单一，无论是实体还是实体间联系，都用“关系”表示。对数据的检索结果也是“关系”（即表），其数据结构简单、清晰，用户易懂易用。

（3）存取路径对用户透明，从而具有更高的数据独立性，更好的安全保密性，简化了程序员的工作。

缺点：查询效率不如非关系模型高。

1.3　关系数据库理论基础

1970 年，IBM 研究员 E. F. Codd 博士在刊物 *Communication of the ACM* 上发表了一篇名为 *A Relational Model of Data for Large Shared Data Banks* 的论文，提出了关系模型的概念，奠定了关系模型的理论基础。关系数据库是当前信息管理系统中最常用的数据库，关系数据库采用关系模式，应用关系代数的方法来处理数据库中的数据。

1.3.1　关系数据结构及形式定义

在关系模型中，无论是实体还是实体之间的联系都由单一的结构类型“关系”来表示。

1. 关系数据结构——笛卡儿积（Cartesian Product）

定义 1.1　设有一组域 $D_1, D_2, \cdots, D_n$，这些域可以部分或全部相同。域 $D_1, D_2, \cdots, D_n$ 的笛卡儿积定义为如下集合：

$$D_1 \times D_2 \times \cdots \times D_n = \{(d_1, d_2, \cdots, d_n) \mid d_i \in D_i, i = 1, 2, \cdots, n\}$$

式中，每一个元素（$d_1, d_2, \cdots, d_n$）称为一个 n 元组（或简称元组），元素中的每一个值 d_i 称为一个分量。

若干域的笛卡儿积具有相当多的元素，在实际应用中可能包含许多“无意义”的元素。人们通常感兴趣的是笛卡儿积的某些子集，笛卡儿积的子集就是一个关系。

两个集合 R 和 S 的笛卡儿积是元素对的集合，该元素对是通过选择 R 的某一元素（任何元素）作为第一个元素，S 的某一个元素作为第二个元素构成的，该乘积用 $R \times S$ 表示。笛卡儿积的结果可表示为一个二维表，表中的每行对应一个元组，表中的每列对应一个域。

例如，给出 3 个域：

D_1=导师集合　导师=王勇，张成

D_2=专业集合　专业=计算机应用专业，信息安全专业

D_3=研究生集合　学生=张燕，杨子，田军

则 D_1, D_2, D_3 的笛卡儿积为：

$D_1 \times D_2 \times D_3$={（王勇，计算机应用专业，张燕），（王勇，计算机应用专业，杨子），（王勇，计算机应用专业，田军），（王勇，信息安全专业，张燕），（王勇，信息安全专业，杨子），（王勇，信息安全专业，田军），（张成，计算机应用专业，张燕），（张成，计算机应用专业，杨子），（张成，计算机应用专业，田军），（张成，信息安全专业，张燕），（张成，信息安全专业，杨子），（张成，信息安全专业，田军）}

该笛卡儿积的基数为 2×2×3=12，也就是说 $D_1 \times D_2 \times D_3$ 一共有 2×2×3=12 个元组，这 12 个元组的总体可组成一张二维表，如表 1-3 所示。

表 1-3　D_1,D_2,D_3 的笛卡儿积

导　师	专　业	学　生
张毅	计算机应用专业	张燕
张毅	计算机应用专业	杨子
张毅	计算机应用专业	田军
张毅	信息安全专业	张燕
张毅	信息安全专业	杨子
张毅	信息安全专业	田军
张成	计算机应用专业	张燕
张成	计算机应用专业	杨子
张成	计算机应用专业	田军
张成	信息安全专业	张燕
张成	信息安全专业	杨子
张成	信息安全专业	田军

2. 关系（Relation）

定义 1.2　笛卡儿积 $D_1 \times D_2 \times \cdots \times D_n$ 的子集 R 称为在域 $D_1 \times D_2 \times \cdots \times D_n$ 上的一个关系（Relation），通常表示为

$$R(D_1, D_2, \cdots, D_n)$$

式中，R 表示关系的名称，n 称为关系 R 的元数或度数（Degree），而关系 R 中所含有的元组个数称为 R 的基数（Cardinal Number）。

关系是笛卡儿积的子集，所以关系也是一个二维表，表的每行对应一个元组，表的每列对应一个域。由于域可以相同，为了加以区分，必须为每列起一个名字，称为属性（Attribute），N 目关系必有 n 个属性。

例如，可以在表 1-3 的笛卡儿积中取出一个子集来构造一个关系。由于一个研究生只师从于一个导师，学习某一个专业，所以笛卡儿积中的许多元组是无实际意义的，从中取出有实际意义的元组来构造关系。给关系命名为 TS，属性名就取域名，即导师、专业和学生，则这个关系可以表示为：TS（导师，专业，学生）。

假设导师与专业是一对一的，即一个导师只有一个专业；导师与研究生是一对多的，即一个导师可以带多名研究生，而一名研究生只有一个导师，这样 TS 关系可以包含 3 个元组，如表 1-4 所示。

表 1-4　TS 关系

导　师	专　业	学　生
张毅	信息安全专业	张燕
张毅	信息安全专业	杨子
张成	信息安全专业	田军

假设学生不会重名（这在实际当中是不合适的，这里只是为了举例方便），则“学生”属性的每一个值都能唯一地标识一个元组，因此可以作为 SAP 关系的主码。

关系可以有 3 种类型：基本关系（通常又称为基本表或基表）、查询表和视图表。基本表是实际存在的表，它是实际存储数据的逻辑表示；查询表是查询结果对应的表；视图表是虚表，是由基本表或其他视图表导出的表，不对应实际存储的数据。

由上述定义可以知道，域 $D_1, D_2, \cdots, D_n$ 上的关系 R，就是由域 $D_1, D_2, \cdots, D_n$ 确定的某些元组的集合。

在关系模型中，对关系做了下列规范性限制：

（1）关系中不允许出现相同的元组。

（2）不考虑元组之间的顺序，即没有元组次序的限制。

（3）关系中每一个属性值都是不可分解的。

（4）关系中属性顺序可以任意交换。

（5）同一属性下的各个属性的取值必须来自同一个域，是同一类型的数据。

（6）关系中各个属性必须有不同的名字。

3. 关系模式

在数据库中要区分型和值。在关系数据库中，关系模式是型，关系是值。关系模式是对关系的描述，关系应描述两个方面内容：元组集合的结构（由哪些属性构成）；关系通常由赋予它的元组语义来确定，元组语义实际上是一个 n 目谓词（n 是属性集中属性的个数）。使 n 目谓词为真的笛卡儿积中的元素（或者说符合元组语义的那部分元素）的全体就构成了关系模式的关系。

定义 1.3　关系的描述称为关系模式（Relation Schema）。它可以形式化地表示为

$$R(U, D, \mathrm{DOM}, F)$$

式中，R 为关系名；U 为组成关系的属性名集合；D 为属性组 U 中属性所来自的域；DOM 为属性向域的映象集合；F 为属性间数据的依赖关系集合。

一般来说关系模式可以简记为：$R(U)$。

关系是关系模式在某一时刻的状态或内容。关系模式是静态的、稳定的，而关系是动态的、随时间不断变化的，因为关系操作在不断地更新着数据库中的数据。有时在实际生活中将关系和关系模式笼统地称为关系。

1.3.2　关系操作

关系模型由关系数据结构、关系操作、关系完整性约束 3 部分组成，上节已经介绍了关系数据结构，本节主要介绍关系操作。

1. 基本操作

关系模型中常用的关系操作包括查询（Query）操作及插入（Insert）操作、删除（Delete）和修改操作两大部分。

关系的查询表达能力很强，是关系操作中最主要的部分。查询操作又可以分为选择（Select）、投影（Project）、连接（Join）、除（Divide）、并（Union）、差（Except）、交（Intersection）、笛卡儿积等。

其中，选择、投影、并、差、笛卡儿积是 5 种基本操作，其他的操作都可以由这 5 种基本

操作来定义和导出。

关系操作是集合操作方式，即操作的对象和结果都是集合。这种操作方式也称为一次一集合（set-at-a-time）的方式。非关系数据模型的数据操作方式是一次一记录（record-at-a-time）方式。

2. 关系代数

关系数据操作就是关系的运算。关系的基本运算有两类：传统的集合运算（并、交、差等）和专门的关系运算（选择、投影、连接），关系数据库进行数据查询时经常需要进行几个基本运算的组合运算。

1）传统的集合运算

传统的集合运算是二目运算，包括并、交、差、笛卡儿积 4 种运算。设关系 R 和关系 S 具有相同的目 n（即两个关系都有 n 个属性），且相应的属性取自相同的域，t 是元组变量，$t \in R$，表示 t 是 R 的一个元组，可以定义并、交、差、笛卡儿积运算如下。

（1）并（Union）。设有关系 R、S（R、S 具有相同的关系模式），二者的“并”运算定义为

$$R \cup S = \{t \mid t \in R \vee t \in S\}$$

式中，“∪”为并运算符；t 为元组变量；结果 $R \cup S$ 为一个新的与 R、S 同类的关系，该关系是由属于 R 和 S 的元组构成的集合。

例如，合并两个相同结构的数据表，就是两个关系的并。

（2）差（Except）。设有关系 R、S（R、S 具有相同的关系模式），二者的“差”运算定义为

$$R-S=\{ t|t \in R \wedge t \notin S \}$$

式中，“-”为差运算符；t 为元组变量；结果 R-S 为一个新的与 R、S 同类的关系，该关系是由属于 R 但不属于 S 的元组构成的集合，即在 R 中减去与 S 中元组相同的那些元组。

例如，设有选修 C 语言的学生关系 R，选修计算机基础的学生关系 S。查询选修了 C 语言而没有选修计算机基础的学生，就可以使用差运算。

（3）交（Intersection）。设有关系 R、S（R、S 具有相同的关系模式），二者的“交”运算定义为

$$R \cap S=\{ t|t \in R \wedge t \in S \}$$

式中，“∩”为交运算符；结果 $R \cap S$ 为一个新的与 R、S 同类的关系，该关系是由属于 R 且属于 S 的元组构成的集合，即两者所有的相同的那些元组的集合。

例如，设有选修 C 语言的学生关系 R，选修计算机基础的学生关系 S。要查询既选修了 C 语言又选修了计算机基础的学生，就可以使用交运算。

（4）笛卡儿积（Cartesian Product）。这里的笛卡儿积是广义的笛卡儿积，两个分别为 n 目和 m 目的关系 R 和关系 S 的笛卡儿积是一个 $n+m$ 列的元组集合。若 R 有 k_1 个元组，S 有 k_2 个元组，则关系 R 和关系 S 的笛卡儿积有 $k_1 \times k_2$ 个元组。记作：$R \times S=\{\text{tr ts} \mid \text{tr} \in R \wedge \text{ts} \in S \}$

2）选择运算

选择运算是指选取关系中满足一定条件的元组。

选择运算的形式定义为：设有关系 R，选择条件（逻辑表达式）用 F 表示，则从关系 R 中选择出满足条件 F 的元组定义为

$$\sigma_F(R) =\{ t|t \in R \wedge F(t)= \text{true} \}$$

例如，从学生表中查询系别为“计算机系”的学生信息，使用的查询操作就是选择运算。

3）投影运算

投影运算是指选取关系中的某些列，并且将这些列组成一个新的关系。

投影运算的形式定义为：设有关系 R，其元组变量为 $t^k=<t_1, t_2, \cdots, t_k>$，那么关系 R 在其分量 $A_{i_1}, A_{i_2}, \cdots, A_{i_n}$（$n \leqslant k$, $i_1, i_2, \cdots, i_n$ 为 1 到 k 之间互不相同的整数）上的投影定义为

$$\prod_{i_1,i_2,\cdots,i_n}(R)=\{\, t \mid t=<t_{i_1}, t_{i_2}, \cdots, t_{i_n}> \wedge <t_1, t_2, \cdots, t_k> \in R \,\}$$

例如，从学生表中查询学生的学号和姓名信息，使用的查询操作就是投影运算。

4）连接运算

用笛卡儿积可以建立两个关系间的连接，但建立的关系是一个较为庞大的体系，而且不符合实际操作的需要。在实际问题当中，两个关系相互连接一般是满足某些条件的，所得到结果往往比较简单。因此，对于笛卡儿积可以做适当的限制，以适应实际应用的需要，这样就引入了连接运算（又称为θ连接）。

连接运算将两个关系拼接成一个更宽的关系，新关系中包含满足连接条件的元组。

连接运算的形式定义为：设有关系 R、S，同时 $i\theta j$ 是一个比较式，其中 i、j 分别为 R 和 S 中的域，θ 为算术比较符，此时关系 R、S 在域 i、j 上的θ连接定义为

$$R \overset{\triangleright}{}_{i\theta j}^{\triangleleft} S=\sigma_{i\theta j}(R \times S)$$

式中，“$\triangleright\triangleleft$”为连接运算符。该式说明，$R$ 与 S 的θ连接是 R 与 S 的笛卡儿积再加上限制 $i\theta j$ 而成，显然，$R \overset{\triangleright}{}_{i\theta j}^{\triangleleft} S$ 中元组的个数远远少于 $R \times S$ 的元组个数。

1.4　常用数据库介绍

当前，市场主流的数据库类型及其相互关系如图 1-17 所示。从图 1-17 可知，关系数据库与 Nosql 数据库并存。关系数据库是建立在关系模型基础上的数据库，其借助于集合代数等数学概念和方法来处理数据库中的数据。主流的 Oracle、MySQL 与 SQL Server 等都属于关系数据库。

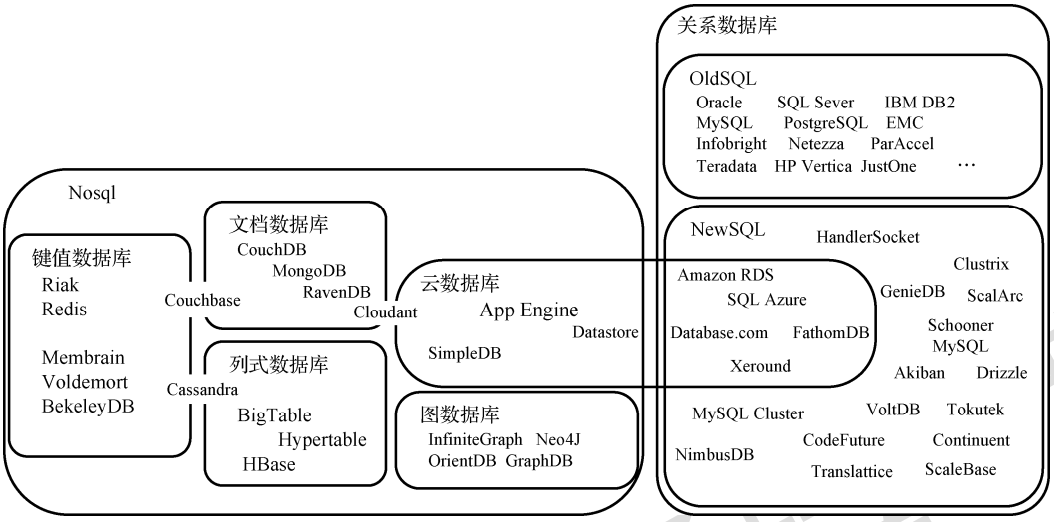

图 1-17　市场主流的数据库类型及其相互关系[1]

1　https://blog.csdn.net/CYLYBYXH/article/details/81029297

Nosql 数据库，全称为 Not Only SQL，意思是适用关系数据库时就使用关系数据库，不适用时也没有必要非使用关系数据库不可，可以考虑使用更加合适的数据存储方式。其主要分为临时性键值存储（Memcached、Redis）、永久性键值存储（ROMA、Redis）、文档数据库（MongoDB、CouchDB）、列式数据库（Cassandra、HBase），每种 Nosql 都有其特有的使用场景及优点。

下面先对主要的数据库进行介绍，然后对关系数据库与 Nosql 数据库进行对比分析。

1.4.1 Oracle

1977 年 6 月，Larry Ellison 与 Bob Miner 和 Ed Oates 创办了一家名为软件开发实验室（Software Development Laboratories，SDL）的计算机公司（Oracle 公司的前身）。当时，32 岁的 Larry Ellison，这个读了 3 家大学都没能毕业的辍学生，还只是一个普通的软件工程师。公司创立之初，Miner 是总裁，Oates 为副总裁，而 Ellison 因为一个合同的事情，还在另一家公司上班，第一位员工是 Bruce Scott。Ellison 和 Miner 预见到数据库软件的巨大潜力，于是，SDL 开始策划构建可商用的关系数据库管理系统（RDBMS）。当时 IBM 发表了“关系数据库”的论文，Ellison 以此造出新数据库，名为 Oracle。

1978 年，公司迁往硅谷，更名为“关系软件有限公司”（RSI）。RSI 在 1979 年的夏季发布了可用于 DEC 公司的 PDP-11 计算机上的商用 Oracle 产品，这个数据库产品整合了比较完整的 SQL 实现，其中包括子查询、连接及其他特性。当时美国中央情报局想买一套这样的软件来满足其业务需求，但在咨询了 IBM 公司之后发现 IBM 没有可用的商用产品，于是联系了 RSI，RSI 有了第一个客户。1983 年，为了突出公司的核心产品，RSI 再次更名为 Oracle 公司。

目前，Oracle 是世界领先的信息管理软件供应商和世界第二大独立软件公司。世界上的所有行业几乎都在应用 Oracle 技术，《财富》100 强中的 98 家公司都采用 Oracle 技术。Oracle 是第一个跨整个产品线（数据库、业务应用软件和应用软件开发与决策支持工具）开发和部署 100%基于互联网的企业软件公司。Oracle 的关系数据库也是世界上第一个支持 SQL 的数据库。

由于 Oracle 包括了几乎所有的数据库技术，因此被认为是未来企业级主选数据库之一。Oracle 主要有以下特点：

（1）对象/关系模型。Oracle 使用了对象/关系模型，也就是在完全支持传统关系模型的基础上，为对象机制提供了有限的支持。Oracle 不仅能够处理传统的表结构信息，而且能够管理由 C++、Smalltalk 及其他开发工具生成的多媒体数据类型，如文本、视频、图形和空间对象等。

（2）动态可伸缩性。Oracle 引入了连接存储池和多路复用机制，提供了对大型对象的支持，当需要支持一些特殊数据类型时，用户可以创建软件插件来实现。Oracle 8 采用了高级网络技术，提供共享池和连接管理器来提高系统的可括性，容量可从几吉字节到几百太字节，可允许 10 万个用户同时并行访问，Oracle 的数据库中每个表可以容纳 1000 列，能满足目前数据库及数据仓库应用的需要。

（3）系统的可用性和易用性。Oracle 提供了灵活多样的数据分区功能，一个分区可以是一个大型表，也可以是索引易于管理的小块，可以根据数据的取值分区，有效地提高了系统操作能力及数据可用性，缓解了 I/O 瓶颈。Oracle 还对并行处理进行了改进，在位图索引、查询、排序、连接和一般索引扫描等操作中引入并行处理，提高了单个查询的并行度。

（4）系统的可管理性和数据安全功能。Oracle 提供了自动备份和恢复功能，改进了对大规模和更加细化的分布式操作系统的支持，加强了 SQL 操作复制的并行性。为了帮助客户有效地管理整个数据库和应用系统，Oracle 还提供了企业管理系统，数据库管理员可以从一个集中控制台拖放式图形用户界面管理 Oracle 的系统环境。

（5）对多平台的支持与开放性。网络结构往往含有多个平台，Oracle 可以运行于目前所有的主流平台上，如 SUN Solarise、Sequent Dynix/PTX、Intel NT、HP UX、DEC UNIX、IBM AIX 等。Oracle 的异构服务为同其他数据源及使用 SQL 和 PL/SQL 的服务进行通信提供了必要的基础设施。

（6）支持大量多媒体数据。

例如，二进制图形、声音、动画及多维数据结构等。

（7）提供了基于角色（Role）分工的安全保密管理。在数据库管理功能、完整性检查、安全性、一致性方面都有良好的表现。

1.4.2　MySQL

MySQL 是一个小型关系数据库管理系统，是一款最流行的开源数据库，开发者为瑞典 MySQL AB 公司。2008 年 1 月 16 日，MySQL 被 SUN 公司收购；2009 年，SUN 又被 Oracle 公司收购，对于 MySQL 的前途，没有任何人抱乐观的态度。目前，MySQL 被广泛应用于 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源代码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。

MySQL 的特性如下：

（1）使用 C 和 C++编写，并使用了多种编译器进行测试，保证源代码的可移植性。

（2）支持 AIX、FreeBSD、HP-UX、Linux、Mac OS、Novell Netware、OpenBSD、OS/2 Wrap、Solaris、Windows 等多种操作系统。

（3）为多种编程语言提供了 API。这些编程语言包括 C、C++、Eiffel、Java、Perl、PHP、Python、Ruby 和 Tcl 等。

（4）支持多线程，充分利用 CPU 资源。

（5）优化的 SQL 查询算法，可有效提高查询速度。

（6）既能作为一个单独的应用程序应用在客户端服务器网络环境中，也能作为一个库而嵌入到其他的软件中提供多语言支持，常见的编码如中文的 GB 2312 和 BIG5、日文的 Shift_JIS 等都可以用作数据表名和数据列名。

（7）提供 TCP/IP、ODBC 和 JDBC 等多种数据库连接途径。

（8）提供用于管理、检查、优化数据库操作的管理工具。

（9）可以处理拥有上千万条记录的大型数据库。

总体来说，MySQL 数据库具有以下主要特点：

（1）同时访问数据库的用户数量不受限制。

（2）可以保存超过 50000000 条记录。

（3）它是目前市场上现有产品中运行速度最快的数据库系统。

（4）用户权限设置简单、有效。

如今，包括 Siemens 和 Silicon Graphics 这样的国际知名公司也开始把 MySQL 作为其数据

库管理系统，这就更加证明了 MySQL 数据库的优越性能和广阔的市场发展前景。

1.4.3 SQL Server

SQL Server 诞生于 1988 年，第一个版本是由 Sybase 公司、Microsoft 公司和 Asbton-Tate 公司联合开发的，只能在 OS/2 上运行。后来，Asbton-Tate 公司退出了 SQL Server 的开发。1993 年，其发布了 SQL Server 4.2 For Windows NT Advanced Server 3.1。这个版本在市场上取得了一些进展，但它不是一个企业级的 RDBMS。1994 年，Sybase 和 Microsoft 停止合作，各自开发自己的 SQL Server。Microsoft 公司致力于 Windows NT 平台的 SQL Server 开发，而 Sybase 公司则致力于 UNIX 平台的 SQL Server 开发。1995 年，Microsoft 公司推出了 SQL Server 6.0。1996 年，其推出了 SQL Server 6.5。SQL Server 6.5 是一个速度快、功能强、易使用、价格低的优秀数据库。1998 年，其推出了 SQL Server 7.0，此版本是一个真正的企业级数据库。2000 年，Microsoft 公司推出了 SQL Server 2000，它是微软.NET 产品的重要组成部分，是大规模联机事务处理（OLTP）、数据仓库和电子商务应用程序的优秀数据库平台。

2005 年，Microsoft 公司又推出了 SQL Server 2005，这是数据库引擎的又一次重写，SQL Server 2005 增加了许多新功能和新技术，包括 Service Broker、通知服务、XQuery 等。此版本是具有里程碑性质的企业级数据库产品。它在企业级支持、商业智能应用、管理开发效率等诸多方面，较 SQL Server 2000 均有质的飞跃，是集数据管理与商业智能（BI）分析于一体的数据管理与分析平台。SQL Server 2005 将 SQL Server 进一步推向企业领域，真正走向成熟，与 Oracle、IBM DB2 形成了商业数据库的三足鼎立之势。之后，SQL Server 历经 2008、2008 R2、2012、2014、2016 各版本的持续投入和不断进化，直至 2017 年 10 月 2 日正式发布了最新版本 SQL Server 2017。

SQL Server 2017 的新特性如下。

（1）跨平台与容器化，支持 Linux 服务器。SQL Server 2017 的第一个不得不提的变化，不是一个具体功能，而是其运行环境的变革——支持 Linux 服务器。这是微软 SQL Server 系列产品首次正式在 Linux 上运行，并且提供完整的官方支持。这无疑大大拓宽了 SQL Server 的应用场景和客户群体。虽然 Windows Server 的授权并不算昂贵，但对于许多以 Linux 生态为主要技术栈的公司而言，并不会考虑申购和运维基于 Windows 的后端服务器——因此在技术选型时，SQL Server 可能第一时间就被排除在外了。当 SQL Server 2017 正式支持 Linux 后，这一障碍将不复存在，SQL Server 终于可以在新的战场和竞争对手展开竞争，无疑会非常有助于其市场份额的提升。

（2）图数据处理。向新兴的 Nosql 学习也是现代关系数据库发展的一个重要特征。如文档数据库善于处理的 JSON 数据，在 SQL Server 2016 中得到了存储、索引、查询等方面的全面支持。SQL Server 2017 与时俱进，又开始向 Neo4j 学习，大胆地引入了图数据的处理与支持。

（3）先进的机器学习功能。机器学习无疑是近年来的热词，也是现代数据应用不可或缺的组成部分。受益于 Revolution Analytics 的收购，SQL Server 2016 版本带来了突破性的 SQL Server R Services。而在 SQL Server 2017 中，则更进一步加入了另一个拥有强大 AI 生态的语言支持——Python。原有的 R Services 也与新引入的 Python 服务一起重命名为 Machine Learning Services（机器学习服务）。

（4）新特性之自适应查询处理。如果说数据库内机器学习给开发者应用带来了智能，那么SQL Server 的查询执行引擎本身是否也能变得更聪明、更智能呢？答案是肯定的，这也正是SQL Server 2017 的另一个发展方向，相关的一系列特性被称为自适应查询处理（Adaptive Query Processing）。

1.4.4 MongoDB

MongoDB（来自英文单词“Humongous”，中文含义为“庞大”）是可以应用于各种规模的企业、各个行业及各类应用程序的开源数据库。MongoDB 是一个基于分布式文件存储的数据库，是当前最成功的 Nosql 数据库之一，使用 C++语言编写，旨在为 Web 应用提供可扩展的高性能数据存储解决方案。在高负载的情况下，添加更多的结点，可以保证服务器性能。MongoDB 能够使企业更加具有敏捷性和可扩展性，各种规模的企业都可以通过使用 MongoDB 来创建新的应用，提高与客户之间的沟通效率，加快产品上市时间，以及降低企业成本。

MongoDB 将数据存储为一个文档，数据结构由键值（key=>value）对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档、数组及文档数组。相关信息存储在一起，通过 MongoDB 查询语言进行快速查询访问。MongoDB 使用动态模式，这意味着可以在不首先定义结构的情况下创建记录，如字段或其值的类型；可以通过添加新字段或删除现有记录来更改记录的结构（称之为文档）。该数据模型可以轻松地代表层次关系，存储数组和其他更复杂的结构。集合中的文档不需要具有相同的一组字段，数据的非规范化是常见的。 MongoDB 还设计了高可用性和可扩展性，并提供了即用型复制和自动分片功能。

与关系数据库相比，MongoDB 的优点如下：

（1）弱一致性（最终一致），更能保证用户的访问速度。举例来说，在传统的关系数据库中，一个 COUNT 类型的操作会锁定数据集，这样可以保证得到“当前”情况下的精确值。在某些情况下，如通过 ATM 查看账户信息时很重要，但对于 Wordnik 来说，数据是不断更新和增长的，这种“精确”的保证几乎没有任何意义，反而会产生很大的延迟。此时需要的是一个“大约”的数字及更快的处理速度。

（2）文档结构的存储方式，能够更便捷地获取数据。对于一个层级式的数据结构来说，如果要将这样的数据使用扁平式的表状的结构来保存，无论是在查询数据还是获取数据时都十分困难。

（3）内置 GridFS，支持大容量的存储。GridFS 是一个出色的分布式文件系统，可以支持海量的数据存储。内置了 GridFS 的 MongoDB 能够满足对大数据集的快速范围查询。

（4）内置 Sharding，提供基于 Range 的 Auto Sharding 机制：一个 Collection 可按照记录的范围，分成若干段，切分到不同的 Shard 上。

（5）第三方支持丰富。现在网络上的很多 Nosql 开源数据库都完全属于社区型的，没有官方支持，给使用者带来了很大的风险。而开源文档数据库 MongoDB 背后有商业公司 10gen 为其提供商业培训和支持。而且 MongoDB 社区非常活跃，很多开发框架都迅速提供了对 MongoDB 的支持。不少知名大公司和网站也在生产环境中使用 MongoDB，越来越多的创新型企业转而使用 MongoDB 作为与 Django、RoR 搭配的技术方案。

（6）性能优越。在使用场合下，千万级别的文档对象，近 10GB 的数据，对有索引的 ID 的查询不会比 MySQL 慢，而对非索引字段的查询，则是全面胜出。MySQL 实际无法胜任大

数据量下任意字段的查询，而 MongoDB 的查询性能让人惊讶，写入性能同样令人满意，对于同样写入百万级别的数据，MongoDB 比 CouchDB 要快得多，基本上在 10 分钟以下就可以解决。

1.4.5 Redis

随着互联网+和大数据时代的来临，传统的关系数据库已经不能满足中大型网站日益增长的访问量和数据量的需要，这时就需要一种能够快速存取数据的组件来缓解数据库服务 I/O 的压力，来解决系统性能上的瓶颈。Redis 是目前最好的缓存数据库。Redis 是一个开源的、高性能的、基于键值对的缓存与存储系统，通过提供多种键值数据类型来适应不同场景下的缓存与存储需求。同时，Redis 的诸多高层级功能使其可以胜任消息队列、任务队列等不同的角色。

2008 年，意大利的一家创业公司 Merzia（http://merzia.com）推出了一款基于 MySQL 的网站实时统计系统 LLOOGG（http://lloogg.com），然而没过多久该公司的创始人 Salvatore Sanfilippo 便开始对 MySQL 的性能感到失望，于是他决定亲自为 LLOOGG 量身定做一个数据库，并于 2009 年开发完成，这个数据库就是 Redis。国内如新浪微博、街旁和知乎，国外如 GitHub、Stack Overflow、Flickr、暴雪和 Instagram，都是 Redis 的用户。

1.4.6 SQLite

SQLite 是最流行的嵌入式数据库。嵌入式数据库有很多种，随着手机移动开发的流行，SQLite 嵌入式数据库异军突起，占领了手机嵌入式数据库的领导地位。SQLite 是一个完整的关系数据库，支持 SQL 标准，支持函数索引、外键、视图、触发器、ACID，扩展支持自定义函数、JSON、全文索引、GIS 等高级特性，可以说功能非常全，但是程序包不到 500KB，可以在几十万字节的内存上运行，是当前手机或掌上嵌入式设备存储结构化数据的最好选择。SQLite 是开源免费软件，同时也有收费功能，主要是支持加密、压缩等高级特性，这些功能对于数据安全要求比较高的业务非常有意义。

1.4.7 关系数据库与 Nosql 数据库比较

1. 关系数据库的优点

关系数据库作为应用广泛的通用型数据库，它的突出优势主要有以下几点：

（1）保持数据的一致性（事务处理）。

（2）由于以标准化为前提，数据更新的开销很小。

（3）可以进行 Join 等复杂查询。

（4）存在很多实际成果和专业技术信息（成熟的技术）。

这其中，能够保持数据的一致性是关系数据库的最大优势。在需要严格保证数据一致性和处理完整性的情况下，用关系数据库肯定是没有错的。但是有些情况不需要 Join 运算，对上述关系数据库的优点也没有什么特别需要，这时似乎也就没有必要拘泥于关系数据库了。

2. 关系数据库的缺点

关系数据库的性能非常高。但它毕竟是一个通用型的数据库，并不能完全适应所有的用途。具体来说它并不擅长以下处理：

（1）大量数据的写入处理。

（2）为有数据更新的表做索引或表结构（schema）变更。
（3）字段不固定时应用。
（4）对简单查询需要快速返回结果的处理。

3. Nosql 数据库的优点

Nosql 主要应用在基于多维关系模型的非结构化的存储及特有的使用场景中。
（1）高并发，大数据下读写能力较强。
（2）基本支持分布式，易于扩展，可伸缩。
（3）简单，弱结构化存储。

4. Nosql 数据库的缺点

（1）Join 等复杂操作能力较弱。
（2）事务支持较弱。
（3）通用性差。
（4）无完整约束，复杂业务场景支持较差。

1.5　小结

数据库管理系统作为数据管理最有效的手段，为高效、精确地处理数据创造了条件。本章主要对数据库的基础知识进行了讲述。从数据与信息、数据管理技术的发展、数据模型、关系数据库及主流数据库等方面进行了介绍。读者通过对本章的学习熟悉数据管理技术发展的阶段特征；掌握 3 种不同的数据模型，并对关系数据库的构成有所了解。

1.6　课后练习

一、单项选择题

1．在数据管理技术的发展过程中，经历了人工管理阶段、文件系统阶段和数据库系统阶段。在这几个阶段中，数据独立性最高的是（　　）阶段。

A．数据库系统　　B．文件系统　　C．人工管理　　D．数据项管理

2．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
C．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
D．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

3．数据库的概念模型独立于（　　）。

A．具体的机器和 DBMS　　B．E-R 图
C．信息世界　　D．现实世界

4．在数据库中，下列说法不正确的是（　　）。

A．数据库避免了一切数据的重复
B．若系统是完全可以控制的，则系统可确保更新时的一致性

C．数据库中的数据可以共享
D．数据库减少了数据冗余
5．（　　）是存储在计算机内有结构的数据的集合。
A．数据库系统　　B．数据库
C．数据库管理系统　　D．数据结构
6．在数据库中存储的是（　　）。
A．数据　　B．数据模型
C．数据及数据之间的联系　　D．信息
7．在数据库中，数据的物理独立性是指（　　）。
A．数据库与数据库管理系统的相互独立
B．用户程序与 DBMS 的相互独立
C．用户的应用程序与存储在磁盘上数据库中的数据是相互独立的
D．应用程序与数据库中数据的逻辑结构相互独立
8．数据库的特点之一是数据的共享，严格地讲，这里的数据共享是指（　　）。
A．同一个应用中的多个程序共享一个数据集合
B．多个用户、同一种语言共享数据
C．多个用户共享一个数据文件
D．多种应用、多种语言、多个用户相互覆盖地使用数据集合
9．数据库系统的核心是（　　）。
A．数据库　　B．数据库管理系统
C．数据模型　　D．软件工具
10．下列关于数据库系统的叙述正确的是（　　）。
A．数据库中只存在数据项之间的联系
B．数据库的数据项之间和记录之间都存在联系
C．数据库的数据项之间无联系，记录之间存在联系
D．数据库的数据项之间和记录之间都不存在联系

二、论述题

1．文件系统中的文件与数据库系统中的文件有何本质上的不同？
2．什么是数据库？
3．数据库管理系统有哪些功能？