《全国计算机等级考试四级教程—数据库工程师》

第一章　引 论

1、 数据库技术产生于 20 世纪 60 年代，是信息系统的核心技术和重要基础；

2、 计算机科学与技术学科划分为四个专业方向：计算机科学（CS）；计算机工程（CE）；软件工程（SE）；信息技术（IT）。

1．1 基本概念

1．1．1 信息与数据

1、 信息、物质、能量是组成客观世界并促进社会发展的三大基本要素；

2、 信息（Information）--是客观世界事物的存在方式和运动状态的反映，是对事物之间相互联系、相互作用的描述。信息具有可感知、可存储、可加工、可传递和可再生的自然属性。

3、 数据（Data）--是描述现实世界事物的符号记录，是用物理符号记录下来的可以识别的信息。不同的物理符号体现出数据的不同表现形式。

4、 信息与数据间存在固有联系，数据是信息的符号表示，或称为载体。信息则是数据的语义解释，是数据的内涵，信息以数据的形式表现出来，并为人们理解和接受。

5、 数据处理（Data Processing）--是指对数据进行分类、收集、组织、存储，进而从已数据出发，抽取或推导出新的数据，这些数据表示了新的信息。

6、 数据管理（Data Management）--是指对数据的分类、收集、组织、编码、存储、检索和维护，是数据处理业务的重要环节。

7、 数据处理与数据管理的区别在于，数据处理除了具有数据管理功能外，还可通过数据管理得到的数据进一步深加工，从中获取新的数据和信息。

1．1．2 数据库系统

1、 数据库（DB，DataBase）--是长期存储在计算机内有组织的、大量的、共享的数据集合；

2、 数据库管理系统（DBMS，Database Management System）--是指在计算机系统中，位于用户与操作系统之间的数据管理系统软件，是数据库系统的核心。

3、 （广义的）数据库系统（DBS，DataBase System）--是指在计算机系统中引入数据库后的软硬件系统构成，DBS 一般分成三个层次：（1）计算机硬件平台；（2）系统软件和应用软件；（3）用户；在不引起混淆和歧义的情况下，数据库系统简称为数据库。

4、 （狭义的）　　—是由数据库和数据库管理系统组成的软件系统，主要为用户提供数据存储和查询、插入、修改、删除、更 8 有新等数据管理功能。

5、 数据库应用系统（DBAS，DataBase Application System）—是由数据库、数据库管理系统、数据库应用程序组成的软件系统，它面向具体应用领域，提供了更为复杂的数据处理功能。

6、 数据库技术—是研究数据库的结构、存储、设计、管理和使用的一门计算机应用学科。

7、 数据库技术与其它计算机科学有密切关系：

（1） 数据库技术以文件系统为基础发展而来，DBMS 需要操作系统的支持，数据库以文件形式存储在外部存储上的；

（2） 数据库与数据结构的关系很密切，数据库技术不仅用到数据结构中的链表、树、图等知识，各种数据模型本身就属于复杂数据结构；

（3） 主流的关系数据库系统，其理论基础是关系数据模型，而该模型是在离散数学集合论中“关系”这一基本概念上发展起来的；

（4） 当用户访问数据库，DBMS 对用户提交的查询操作类似于，计算机编译系统对程序的编译过程；

（5） 开发一些大型的 DBS 或 DBMS 的过程，要遵循软件工程的开发模式。

1．2 数据模型

1．2．1 数据模型概念

1、数据模型（Data Model）--对现实世界的特征抽象。是数据库系统的形式框架，是用来描述数据的一组概念和定义，包括描述数据、数据联系、数据操作、数据语义以及数据一致性的概念工具；

2、数据模型应满足 3 个条件：（1）能够比较真实地模拟现实世界；（2）容易为人们所理解；（3）便于在计算机上实现。

3、 数据模型的组成：

（1） 数据结构：用于描述系统的静态特征，从语法角度表述了客观世界中数据对象本身的结构和数据对象之间的关联关系，是刻画一个数据模型性质最重要的方面。在数据库系统中，通常按照数据结构的类型来区分、命名各种数模，如层次、网状、关系数模。

（2） 数据操作：用于描述系统的动态特征，是一组对数据库中各种数据对象允许执行的操作和操作规则组成的集合。数据操作可以是检索、插入等，数模必须定义这些操作的确切含义、操作符号、操作规则以及实现操作的数据库语言。

（3） 数据完整性约束：是一组完整性规则的集合，它定义了数模必须遵守的语义约束，也规定了数据库中数据内部及数据之间联系所必须满足的语义约束。它限定了数据库的状态以及状态的变化，以便维护数据的正确性、有效性。

1．2．2 数据模型分类

1、 用数据模型这一概念来描述数据库的结构和语义，通过现实世界—信息世界—机器世界（建模过程）的抽象转换过程构建数据库，并根据模型所定义的规范去管理和使用数据。

2、 建模过程：（1）将现实世界的数据对象抽象为信息世界中的某一信息结构；（2）再将信息结构转换为机器世界中某一具体 DBMS 支持的数据模型，并存储于计算机中。

3、 数据模型分类：（由上到下）

（1） 概念数据模型（概念模型）：按用户的观点对数据和信息进行建模，是现实世界到信息世界的第一层抽象，强调其语义表达功能，易于用户理解，是用户与设计人员交流的语言，主要用于数据库设计。最常用的是实体—联系模型。

（2） 数据结构模型（表示型/实现型）：是机器世界中与具体 DBMS 相关的数据模型，包括关系模型、网状模型和层次模型

（3） 物理数据模型：属底层数据模型，描述数据的实际存储方式。

1．3 数据视图与模式结构

1．3．1 数据视图与数据抽象

1、 数据视图：指从某个角度看到的客观世界数据对象的特征，是对数据对象某一方面特征的描述。

2、 数据抽象：是一种数据描述和数据库设计原则，是指专注于数据对象的某方面特征，而忽略其他特征。

3、 集和值：集是指对某一类数据的结构和属性的说明，值是集的一个具体赋值；

4、 数据模式：对数据库中数据某方面结构和特征的描述，它仅涉及集的描述，不涉及具体的值。

1．3．2 三级模式结构（从数据库管理系统角度）

1、 数据库三级模式结构—外部级、概念级和内部级，分别定义了外模式、模式和内模式，用于从不同角度描述数据库结构。

2、 模式：

（1） 也称逻辑模式、概念模式；

（2） 对数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图；

（3） 模式不仅定义了数据的逻辑结构，还定义了数据之间的联系、与数据的关的安全性和完整性要求；

（4） 一个数据库只有一个模式，建立在某种数据结构模型基础上。

3、 外模式：

（1） 也称子模式、用户模式、用户视图；

（2） 是对数据库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。

（3） 一个数据库可以有多个外模式，每个外模式描述了某个特定用户所使用的局部数据的逻辑结构和特征，是与某一应用有关的数据的逻辑表示。

（4） 外模式还是保证数据安全的有力措施，每个用户只能看见和访问所对应的外模式中的数

据，其它数据对他是不可见的。

4、 内模式：

（1） 也称物理模式、存储模式；

（2） 是对数据库中数据的物理结构和存储方式的描述，代表了数据在数据库内部的表示方式和物理组织结构；

1．3．3 二级映象与数据独立性

1、 外模式/模式映象：

（1） 定义了数据库中不同用户的外模式与数据库逻辑模式之间的对应关系；

（2） 可有多个外模式/模式映象，对于每个外模式，需要一个外模式/模式映象来定义该外模式与模式之间的对应关系；

（3） 当模式发生变化时，只需调整外模式/模式间的映象关系，而外模式无需修改，保证了数据与应用程序的逻辑独立性，称为数据的逻辑独立性。

2、 模式/内模式映象：

（1） 定义了数据库中数据全局逻辑结构，与这些数据在系统中的物理存储组织结构之间的对应关系。

（2） 模式/内模式映象是唯一的；

（3） 当内模式发生变化时，只需调整模式/内模式映象关系，而模式无需修改，保证了数据库中的数据与应用程序间的物理独立性，称为数据的物理独立性。

1．4 数据库系统体系结构

1、 数据库系统体系结构（从用户角度）：是指数据库系统的组成构件、各构件的功能及各构件间的协同工作方式；

2、 分类：

（1） 集中式：全部数据和数据管理功能均集中在一台计算机上的数据库系统；包括单用户和主从式两种，单用户 DBS 是指系统由一个用户独占，不同机器间不能共享数据；主从式 DBS 是指一个主机带多个分时多用户的 DBS；

（2） 分布式：数据库中的数据在逻辑上是一个整体，但在物理上却可以分布在网络中不同数据管理节点上；

（3） 客户/服务器：将 DBMS 和数据库应用分开，网络中某些节点上的计算机专门执行 DBMS 功能，负责数据管理服务，称为数据库服务器；其他节点的计算机上安装 DBMS 的外围应用开发工具，支持用户的应用，主要负责数据表示服务，称为客户端；

（4） 并行式：硬件平台是并行计算机系统，使用多个 CPU 和多个磁盘进行并行数据处理和磁

盘访问操作，以提高执行速度；

（5） WEB 式： 由通过互联网连接起来的客户端、WEB 服务器、数据库服务器组成。

1．5 数据库管理系统

1．5．1 数据库管理系统的功能

（1） 数据定义功能：DBMS 提供了数据定义语言（DDL），用户利用 DDL 定义数据库对象的三级模式结构，描述数据库的结构特征。

（2） 数据操纵功能：DBMS 提供数据操纵语言（DML），用户利用 DML 对数据进行查询、插入、删除或更新；

（3） 数据库运行管理和控制功能

（4） 数据库的建立和维护功能

1．5．2 数据库系统的全局结构（图）

1、 DBS 可分为用户、人机交互界面、DBMS 和磁盘四个层次；

2、 用户可分为四类：数据库管理员 DBA；专业用户；应用程序员；终端用户；

3、 DBMS 可分为两部份：

（1） 查询处理器：面向用户查询请求；包括以下几个功能模块：DML 编译器、嵌入式 DML 的预编译器、DDL 编译器、查询执行引擎；

（2） 存储管理器：面向数据存储访问，包括以下几个功能模块：权限和完整性管理器、事务管理器、文件管理器、缓冲区管理器；

4、 磁盘存储的类型：

（1） 以数据库文件方式存储的应用数据；

（2） 数据字典；

（3） 为提高查询速度而设置的数据库引擎；

（4） DMS 运行时的统计分析数据；

（5） 日志信息。

1．6 数据库技术的发展和应用

1、 第一代 DBS：60 年代末 70 年代初，层次型和网状型 DBS；

2、 第二代 DBS：70 年代后期，关系数据库系统；

3、 新型 DBS：80 年代，分布式数据库系统；90 年代，面向对象数据库系统、网络数据库系统

第二章 数据库应用系统（DBAS）生命周期

2．1 数据库应用系统生命周期

2．1．1 软件工程与软件开发方法

1、 软件工程：指导计算机软件开发和维护的工程科学，它采用工程化的概念、原理、技术和方法，以及正确的项目管理技术，来开发和维护软件；它将系统化、规范化、定量化方法应用于软件的开发、操作和维护，也就是将工程化应用于软件生产；

2、 软件工程的目标：在给定成本、进度的前提下，开发出满足用户需求并具有下述特征的软件产品：可修改性、有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性。

3、 软件生命周期：（以及开发周期）指软件产品从考虑其概念开始，到该不再使用的整个时期（产品交付使用的整个时期），包括概念阶段、需求阶段、设计阶段、实现阶段、测试阶段、安装部署及交付阶段；

4、 软件项目管理：为了能使软件开发按预定的质量、进度和成本进行，而对成本、质量、进度、人员、风险等进行分析和有效管理的一系列活动。

5、 软件工程以关注软件质量为特征，由方法、工具和过程三部分组成；

6、 软件过程模型（软件开发模型）：是对软件过程的一种抽象表示，表示了软件过程的整体框架和软件开发活动各阶段间的关系，常见的有：瀑布模型、快速原形模型、增量模型和螺旋模型。

2．1．2 DBAS（面向某个特定领域，实现特定功能的计算机软件、硬件的集成体）软件组成

1、 数据库应用软件在内部可看作由一系列软件模块/子系统组成，这些模块/子系统可分成两类：

（1） 与数据访问有关的数据库事务模块：利用 DBMS 提供的数据库管理功能，以数据库事务方式直接对数据库中的各类应用数据进行操作，模块粒度较小；

（2） 与数据访问无直接关联的应用模块：在许多与数据处理有关的应用系统中，对数据库的访问只是整体中的一部分，其他功能则与数据库访问无直接关系，这部分模块粒度可以比较大。

2、 DBAS 设计开发的硬件方面：主要涉及根据系统的功能、性能、存储等需求选择和配置合适的计算机硬件平台，并与开发好的 DBAS 软件系统进行集成，组成完整的数据库应用系统；

2．1．3 DBAS 生命周期模型

1、 数据库应用系统的生命周期模型：（图，p17）

（1） 参照软件开发瀑布模型的原理，DBAS 的生命周期由项目规划、需求分析、系统设计、实现和部署、运行管理与维护等 5 个基本活动组成；

（2） 将快速原形模型和增量模型的开发思路引入 DBAS 生命周期模型，允许渐进、迭代地开发

DBAS；

（3） 根据 DBAS 的软件组成和各自功能，细化 DBAS 需求分析和设计阶段，引入了数据组织与存储设计、数据访问与处理设计、应用设计三条设计主线，分别用于设计 DBAS 中的数据库、数据库事务和应用程序；

（4） 将 DBAS 设计阶段细分为概念设计、逻辑设计、物理设计三个步骤，每一步的设计内容又涵盖了三条设计主线。

2．2 规划与分析（图 p18）

2．2．1 系统规划与定义

1、 定义：系统规划与分析是面向将要开发的 DBAS，通过了解用户实际需求，明确该系统需要实现的目标和任务，并从数据管理和数据处理的角度，确定系统中数据库软件的功能、性能范围；

2、 系统规划与定义包括：

（1） 任务陈述：描述所要开发的 DBAS 的总体目标；

（2） 确定任务目标；

（3） 确定系统范围和边界；

（4） 确定用户视图；

2．2．2 可行性分析

1、 可行性分析包括以下四方面：

（1） 经济可行性：对项目进行成本效益分析；DBAS 的成本主要包括：A、软硬件购置费用；B、系统开发费用；C、系统安装、运行、维护费用。

（2） 技术可行性：是根据用户提出的系统功能、性能及实现系统的各项约束条件，对系统软件、硬件、技术方案作出评估和选择建议；

A、 硬件可行性研究是分析 DBAS 的硬件平台环境和设置；

B、 软件可行性研究包括：对可用的 DBMS 和操作系统的选型评估，对中间件和开发环境的选型建议，对 DBAS 开发模式和编程语言的建议；

C、 技术方案的选择是根据系统技术需求，提出 DBAS 可能采用的合理技术方案和关键技术；

（3） 操作可行性：是论证是否具备 DBAS 开发所需的各类人员资源、软件资源、硬件资源和工作环境等，以及为支持 DBAS 开发如何去改进加强这几方面资源。

（4） 开发方案选择：目的是提出并评价实现系统的各种开发方案，从中选出一种适用于 DBAS 软件的开发方案；

2．2．3 项目规划

1、 项目规划是项目管理者对资源、成本和进度做出合理估算，并在此基础上制定切实可行的 DBAS

项目开发计划。

2、 项目规划包括以下内容：

（1） 确定项目的目标和范围；

（2） 根据 DBAS 软件开发模型，分解和定义整个项目包括的工作活动和任务；

（3） 估算完成该项目的规模和所需各种资源；

（4） 制定合理的 DBAS 项目计划

3、项目规划的结果应形成数据库应用系统项目计划文档，即项目计划书。

2．3 需求分析

1、 数据库应用系统需求是指用户对 DBAS 在功能、性能、行为、设计约束等方面的期望和要求；

2、 DBAS 需求分析是在已经明确的 DBAS 系统范围基础上，通过对应用问题的理解和分析，采用合适的工具和符号，系统地描述 DBAS 的功能特征、性能特征和约束，并形成需求规范说明文档；

3、 需求分析过程由需求获取、需求分析、需求描述和规范说明、需求验证等组成；

4、 DBAS 的需求分析包括：

（1） 数据需求分析；

（2） 数据处理需求分析；

（3） 务需求分析；

（4） 分析数据库系统在性能、存储、安全、备份与恢复等方面的要求；

2．3．1 数据与数据处理需求分析（承上分别介绍）

1、 数据需求分析：是从对数据组织与存储的设计角度，辨识应用领域所管理的各类数据项和数据结构，与数据处理需求分析结果一起，组成数据字典；包括 5 个部分（Def:是一种用户可以访问的记录数据库和应用程序元数据的目录，它详细描述系统中的全部数据）

2、 数据处理需求分析：是从数据访问和处理的角度，明确对各类数据项所需进行的数据访问操作，分析结果可表示为数据流图或事务规范；（DFD）

3、 事务规范包括：

（1）事务名称；（2）事务描述；（3）事务所访问的数据项；（4）事务用户；

2．3．2 业务规则需求分析

1、业务规则需求分析：是从 DBAS 高层目标和整体功能出发，分析系统或系统中一些大粒度子系统应具有的业务类型和功能，明确用户或外部系统与 DBAS 的交互模式；

2．3．3 性能（能做到什么程度）需求分析

1、 DBAS 的性能指标：

（1） 数据操作响应时间（或数据访问响应时间）：从提交请求到返回结果的时间；

（2） 系统吞吐量：指系统在单位时间内所完成的事务或查询的数量，单位为 TPS；（TPC 是指事务处理性能委员会）

（3） 允许并发访问的最大用户数：在保证响应时间的前提下，系统最多允许多少用户同时访问数据库；

（4） 每 TPS 代价值，用于衡量系统性价比的指标

2、 影响 DBAS 性能的因素：

（1） 系统硬件资源；

（2） 网络通信设备性能；

（3） 操作系统环境；

（4） 数据库的逻辑设计和物理设计质量，数据库配置参数；

（5） DBAS 的配置和性能；

（6） 数据库应用程序自身。

2．3．4 其它需求分析

1、 存储需求分析：是指估计 DBAS 系统需要的数据存储量，包括：（1）初始数据库大小；（2）数据库增长速度；存储总量估算可采用：根据数据字典中每个数据项的结构描述信息，估计每个数据项的容量，将所有数据项的容量累加；

2、 安全性需求分析：

（1） DBAS 系统应达到的安全控制级别；

（2） 各类用户的数据视图和视图访问权限；

（3） DBAS 应有的口令保护机制或其它安全认证机制，用以控制用户登录数据库系统。

3、 备份和恢复需求分析：

（1） DBAS 运行过程中备份数据库的时间和备份周期；

（2） 所需备份的数据是全部数据库数据，还是一部分；

（3） 备份方式是采用完全备份还是采用差异备份。

2．4 系统设计

2．4．1 概念设计（DB 概念模型设计和总体设计）

1、 数据库概念模型设计：是根据数据需求分析阶段得到的需求结果，分析辨识需要组织存储在数据库中的各类应用领域数据对象的特征及其相互之间关联关系，并采用概念数据模型表示出来，得到独立于具体 DBMS 的数据库概念模型；

2、 ER 方法：（1）选择局部应用；（2）分别设计各个局部 ER 图；（3）局部 ER 图合并；

3、 系统总体设计：

（1） 确定 DBAS 体系结构；

（2） 系统硬件平台和操作系统、数据库管理系统等系统软件的选型和配置；

（3） 应用软件结构设计（软件程序概要设计Ⅰ）

（4） 对需求分析阶段识别出的业务规则进行初步设计，细化业务规则流程，明确采用的关键技术和算法；

（5） 对系统采用的关键技术进行方案选型和初步设计。

2．4．2 逻辑设计

1、 数据库逻辑结构设计：指从数据库的概念模型出发，设计表示为逻辑模式的数据库逻辑结构。（即从 ER 图到关系模型）

（1） ER 图转换为初始关系模式；

（2） 对初始关系模式进行优化；

（3） 检查关系表对数据库事务的支持性；

（4） 确定关系模式的完整性约束；

（5） 从数据安全性和独立性出发，设计用户视图。

2、 应用程序概要设计（II）；（在Ⅰ上细化）

3、 数据库事务概要设计；

2．4．3 物理设计

1、 数据库物理结构设计：主要指数据文件在外存上的存储结构和存取方法，它依赖于系统具体的硬件环境、操作系统和 DBMS；

（1） 数据库逻辑模式调整；

（2） 选择或配置基本关系表的文件组织形式；

（3） 数据分布设计；

（4） 安全模式设计；

（5） 确定系统配置；

（6） 物理模式评估；

2、 数据库事务（相当于 VB 的事件）详细设计：根据事务流程，利用 SQL 语句、数据库访问接口，采用高级程序设计语言或 DBMS 提供的事务实现机制，设计数据库事务。

3、 应用程序详细设计：

2．5 实现与部署

1、 建立数据库结构；

2、 数据加载；

3、 事务和应用程序的编码及测试；

4、 系统集成、测试与试运行；

5、 系统部署；

2．6 运行管理与维护

2．6．1 日常维护

（1） 数据库的备份与恢复

（2） 完整性维护

（3） 安全性维护

（4） 存储空间管理

（5） 并发控制及死锁处理

2．6．2 系统性能监控和分析

1、 统计数据可以通过两种途径收集：

（1） 由 DBMS 本身自动收集和存储统计数据

（2） 通过监控系统得到

2．6．3 系统性能优化调整

1、 系统性能优化的手段有：数据查询调整与优化、索引调整、数据库摸式调整、 DBMS 和操作系统参数调整等。

2、 模式调整主要涉及逻辑模式调整，可以从下考虑：

（1） 已达到第三范式的基本表，不要进一步规范化为 BCNF；

（2） 在分布式数据库中，对一个基本表中某些频繁被访问的数据，可以按水平分区或垂直分区方式拆分基本表。

2．6．4 系统升级

1、 改进应用程序；

2、 数据库重组；

3、 DBMS 和 OS 版本升级

第 3 章 需求分析及功能建模方法

3．1 需求分析概述

3．1．1 需求分析概念

1、 所谓需求分析：就是对待开发的系统要做什么，完成什么功能的全面描述。

2、 需求分析的工作：通过对需求的调查、了解、观察和分析，通过对原始数据的收集、分类和抽象，并采用有效的技术、工具，对原始资料进行加工整理，描述开发目标、实现的功能及其相互关系等活动的集合；

3、 需求的定义：客户对一个待开发的系统在实现目标、完成功能、应达到的性能、安全性、可靠性等方面的期望和要求的集合；

4、 需求获取的困难：

（1） 软件功能复杂；

（2） 需求的可变性；

5、 需求分析阶段的主要任务：分析当前的业务流程，包括体系结构，各职能部门完成的主要任务、关系及其交流的信息。

6、 需求分析的结果通常以模型等建模工具和方法描述系统的信息流、功能结构及完成各功能需要的数据。

7、 功能模型和软件需求规格说明书是软件开发的依据，将指导后续的开发工作。

8、 需求分析工作是系统分析员与用户不断交互的过程中完成的。

3．1．2 系统分析员的职能

1、 系统分析员的主要要任务：是确定应用信息系统及软件产品应该达到的各项功能性要求和非功能性要求，即用户要做什么。

2、 系统分析员应该具备的素质：

（1） 获取需求的能力；

（2） 管理及沟通能力；

（3） 技术素养；

3．1．3 需求获取的方法

常用的几种获取需求的方法：（1）面谈；（2）实地观察；（3）问卷调查；（4）查阅资源；

3．1．4 需求分析过程

1、 标识问题：

（1） 需求分析的第一步，通过对问题的识别和标识获得所求解问题及其运行环境的理解；

（2） 标识问题从现行系统的业务流程做起，理解现行系统的业务流程；

（3） 在标识理解需求的同时，还要注意确定系统的人机界面；

2、建立需求模型：

（1） 模型是对现实原形所作的一种抽象，其本质是只关心与研究内容有关的因素，而忽略无

关的因素，其目的是把复杂的事物变得简单，便于认识和分析；

（2） 目前常用的模型方法主要有 DFD 数据流图和 IDEFO，都属于结构化分析方法，其特征是抽象和分解；

（3） 首先对应用领域进行全面的分析，发现并找出同类事物的本质，用抽象方法把这类事物的非主要方面剔除，把握住事物的内部规律或本质，就可以找到解决办法；然后采用自上而下逐步求精的方法对复杂的问题进行分解；

（4） 结构化分析及建模方法的主要优点：

（A） 不过早陷入具体的细节；

（B） 从整体或宏观入手分析问题；

（C） 通过图形化的模型对象直观地表示系统要做什么，完成什么功能；

（D） 图形化建模方法方便系统分析员理解和描述系统；

（E） 模型对象不涉及太多的技术术语，便于用户理解；

3、描述需求：

（1） 需求描述的目标：对软件项目功能性和非功能性的需求全面描述；

（2） 功能性需求：指需要计算机实际解决的问题或实现的具体功能，明确描述系统必须做什么，实现什么功能以及输入输出等；

（3） 非功能性需求：软件项目对实际运行环境的要求；

（4） 需求描述主要由需求模型和需求说明书组成，说明书侧重文字说明，内容如下：需求概述；功能需求；信息需求；性能需求；环境需求；其他需求；

（5） 在对需求进行分析过程中，系统分析员要经常考虑的问题：

（A） 描述的需求是完全的吗？

（B） 需求描述是正确的和一致的吗？

（C） 描述的这些需求是可行的、实际可操作的吗？

（D） 描述中的每一条需求都是客户需要的吗？

4、确认需求：

1、 评审委员会审核下列内容：功能需求；数据需求；性能；数据管理；其他需求。

3．2 DFD 建模方法

3．2．1 DFD 方法的基本对象

1、 数据流：具有名字且有流向的数据，用标有名字的箭头表示。

2、 处理：表示对数据的加工和变换，在图中用矩形框表示。

3、 数据存储：表示用数据库形式存储的数据，对其存取分别以指向或离开数据存储的箭头表示；

4、 数据源及数据终点：表示当前系统的数据来源和去向，其图形符号以平行四边形表示。

3．2．2 开发 DFD 图

1、 DFD 图采用自顶而下逐步细化的结构化分析方法表示目标系统；

2、 DFD 方法应以软件项目的功能为中心进行抽象和分解，以数据流的变换来分析数据对企业中各类业务活动的影响；

3．2．3 建模案例（p43）；

3．2．4 数据字典

1、 数据字典包括以下说明信息：

（1） 源点及终点词条描述；

（2） 数据流词条描述；

（3） 数据存储；

（4） 处理描述；

（5） 数据元素词条描述。

3．3 IDEF0 建模方法

3．3．1 概述

1、 IDEF0 的基本思想是结构化分析方法，强调自顶而下有控制地逐步地展开细节，全面地描述系统，且通过建模来理解一个系统。一个模型由图形文字说明、词汇表及相互的交叉引用表组成。

2、 IDEF 方法的优点：具有模型元素单一、语义丰富、更易于从全局角度分析考察问题，模型容易理解。

3．3．2 IDEF0 方法

1、基本元素

（1） 矩形：代表活动，活动名称标在矩形内，活动编号按要求标在矩形框右下角指定位置；

（2） 箭头：左边的输入箭头代表完成活动需要的数据、上方的控制箭头描述了影响活动的执行的事件或约束、右边的输出箭头说明由活动产生的结果及信息、下方进入的机制箭头表示实施该活动的物理手段或资源。

（3） 输入输出箭头描述活动是什么（what）、控制箭头描述为何这么做（why）、机制箭头表示如何做（how）。

2、IDEF0 模

（1） 一个 IDEF0 模型由一组图形组成，这些图形组成一个由父到子的层次结构图，这组图形把一个复杂事物按自顶向下逐步细化的方式分解成一个个简单的或多个组成部分；

3、 建模规则

（1） 矩形框：用动词为矩形内活动命名，每个矩形要至少有一个控制箭头和输出箭头，可以没有输入，但不可以同时没有输入和控制。

（2） 箭头：箭头代表数据约束，而不是代表流或顺序；

（3） 其他：

（A） ICOM 码：只有一端与矩形相连的箭头叫边界箭头，这些箭头表示父矩形框的输入、控制和输出。IDEF0 用专门的记号 ICOM 码来说明父子图中的箭头关系。子图中每个边界箭头的开端分别用字母 I、C、O、M 来标明是输入、控制、输出及机制，再用一个数字表示其在父矩形框中箭头的相对位置。

（B） 结点号：IDEF0 模型是一组有一定层次结构的图形，通常用结点号来标志图形或矩形框在层次图中的位置；

（C） 模型名：每个模型有一个名字，通常用名字代表主题，用子名字表示不同的模型。基本名字与子名字间用“/”隔开，如 A/B/C，A 是主题、B 是模型号、C 是结点号。

3．3．3 建模过程及步骤

1、 IDEF0 建模过程及步骤：

（1） 明确目的，确定范围：在建模前首先要明确目的和意图，确定问题域；

（2） 建立内外关系图 A-0 图：根据系统目标、功能建立内外关系图 A-0 图，以确定整个模型的内外关系，确定系统的边界；

（3） 构造顶层图：把 A-0 图分解成 3~6 个主要部分得到 A0 图，A0 图是模型真正的顶层图；

（4） 开发 IDEF0 层次结构图：对 A0 图中的每个矩形框进行分解，就形成了基本的图形层次结构。在分解时要列出所有的数据项和活动表，分解的次序采用以下原则：

（A） 保持在同一水平上进行分解，均匀的模型深度；

（B） 按困难程序进行选择；

（5） 写文字说明；

（6） 检查确认图形；

3．4 DFD 与 IDEF0 的比较

1、 DFD 与 IDEF0 共同点：都是结构化分析思想，强调自顶而下逐步求精的方法对现实世界建模，先抓住主要的问题，形成较高层次的抽象，再由粗到细、由表及里地逐步细化，将一个大问题分解成几个小问题，对这小问题再进行分析求解；

2、 DFD 与 IDEF0 区别：

（1） DFD 图用箭头（数据流）来描述数据移动的方向、数据处理及处理之间的数据依赖关系。

IDEF0 图也用箭头代表数据流，但在 IDEF0 中不是强调流或顺序，而是强调数据约束。

（2） 从表达形式上看，DFD 图与 IDEF0 图都是用箭头和处理表达一个企业或组织的业务流程。但 IDEF0 图的箭头不仅能够表示数据流，还可以表示控制流和说明处理或实施方式的一些约束；

（3） 从模型元素的组成上来看，DFD 模型由 4 种元素组成，即外部顶、数据流、数据存储和处理。而 IDEF0 模型元素的组成更加简单，只有 2 种元素组成，即箭头和活动；

（4） 从模型规范上来讲，IDEF0 方法更加规范；

（5） IDEF0 模型结构清楚，便于理解和沟通。

第四章 数据库概念设计及数据建模

4．1 数据库概念设计概述

4．1．1 数据库概念设计的任务

1、 定义和描述应用领域涉及的数据范围；

2、 获取应用领域或问题域的信息模型；

3、 描述清楚数据的属性特征；

4、 描述清楚数据之间的关系；

5、 定义和描述数据的约束；

6、 说明数据的安全性要求；

7、 支持用户的各种数据处理需求；

8、 保证信息模型方便地转换成数据库的逻辑结构，同时便于用户理解。

4．1．2 概念设计过程

1、 概念设计的依据：是需求分析阶段的文档，通过对这些文档的分析理解，构造出信息模型，编写数据库概念设计说明书，信息模型和数据库概念设计说明书是数据库逻辑设计的依据；

2、 概念设计的基本步骤：

（1） 确定实体集；

（2） 确定联系和联系类型；

（3） 建立由信息模型表示的企业模型；

（4） 确定实体集属性；

（5） 对信息模型优化。

3、 概念设计的方法（自顶向下、自底向上、逐步扩张、混合策略）

4．2 数据建模方法

1、 数据建模方法的共同特点是：

（1） 能够真实客观地描述现实世界中的数据及数据之间的关系；

（2） 组成模型的概念少，语义清楚，容易理解；

（3） 不同概念的语义不重叠，概念无多义性；

（4） 用图形方式描述数据，数据直观易懂，有利于数据库设计者和用户交流；

（5） 这种数据模型容易转换成数据库逻辑设计阶段需要的数据结构。

4．3 ER 建模方法

4．3．1 基本概念

1、 实体或实例：指客观存在并可相互区分的事物，可以是一个具体的人或物，也可以是抽象的事件或概念；

2、 实体集：表示一个现实的和抽象事物的集合，这些事物必须具有相同的属性或特征。

3、 属性：用于描述一个实体集的性质和特征；

4、 码：实体集中能惟一标识每一个实例的属性或属性组；（以及域）

5、 联系：描述现实世界中实体之间的关系。（1）一对一联系；（2）一对多联系；（3）多对多联系

4．3．2 ER 方法语法

1、 ER 方法中用矩形框表示实体集，矩形框内写上实体集的名称；

2、 ER 模型用菱形表示联系，联系名写在菱形框内；

3、 ER 模型中实体集的属性用椭圆或圆角矩形框表示，属性名字写在其中。

补充：1、ER 建模步骤：局部（以数据字典为依据，自底而上）（确定范围，识别实体、确定关系、定义属性）到全局。

2、全局（合并和重构）合并：消除冲突（属性，命名，结构）；冲突：消除冗余。

4．4 IDEF1X 建模方法

4．4．1 IDEF1X 概述

1、 IDEF0 侧重描述系统功能，被称为功能建模方法；IDEF1X 侧重分析、抽象和概括应用领域中的数据，称为数据建模方法；

2、 IDEF1X 方法具有丰富的语法和语义；

3、 实体集分为（1）独立标识符实体集；（2）从属标识符实体集；

4、 实体集之间的联系分为：（1）标定型联系；（2）非标定型联系；（3）分类联系；（4）不确定联系

4．4．2 IDEF1X 模型元素

1、 实体集：

（1） 实体集语义：如果一个实体集的每一个实例都能被惟一地标识，而不决定于它与其他实体的联系，那么该实体集称为独立实体集；（存在键区）否则就叫从属实体集；

（2） 实体集语法：IDEF1X 用矩形框来表示独立实体集，用圆角矩形框来表示从属实体集；

2、 联系：

（1） 联系语义：

（A） 标定型联系：一个“确定型联系”（一对多）中，如果子女实体集中的每个实例都是由它与双亲的联系而确定的，这个关系称为“标定型联系”（在子实体中做外键时，在键区，反之成立）；

（B） 非标定型联系：一个“确定型联系”中，如果子女实体集中的每一个实例都能被惟一地确认而无需了解与之相联系的双亲实体集的实例，这个问题关系叫“非标定型联系”。

（C） 分类联系：是两个或多个实体集之间的联系，且在这些实体集中存在一个一般实体集，它的每一个实例都恰好与一个且仅一个分类实体集的一个实例相联系。

（D） 不确定联系：一个非确定联系又称为多对多联系，这种联系关联的两个实体集之间，任一实体集的一个实例都将对应另一实体集的 0 个、1 个或多个实例。

（2） 联系的语法：

（A） 标定联系语法：在 IDEF1X 图中，联系的语法用直线表示，在一个标定型联系中，子女实体集总是一个从属实体集，用圆角矩形框表示；

（B） 非标定联系语法：如果两个实体集之间有关系，并且是一个非标定联系，就用一条虚线把它们连接起来。

（C） 分类联系语法：一般实体集的一个实例只能与分类实体集的一个实例相对应；

（D） 不确定联系 m:n 的语法：不确定联系用一个两端带有实心圆的线段描述，表示多对多的连接关系。

3、 属性

（1） 属性的语义：用来描述一类现实或抽象事物的特征或性质。一个属性的具体取值叫属性实例，它由属性的类型和值来定义。

（2） 属性的语法

（A） 主码和非主码属性语法：在一个实体集中属性要有惟一的名字，属性名由名词表示，主码属性名后加（PK）标注，被列在属性列表的顶端，并用水平线将主码和其他属性分开。

（B） 外码语法：在外码属性后加“FK”来识别由联系继承得到的外来属性。

4．4．3 建模过程

1、第一阶段：建模规划及准备

（1） 建模目标：

（A） 目标说明：回答将构造的模型完成什么功能，涉及的问题和数据范围，同时说明是一个当前系统模型还是待建模型。

（B） 范围说明：在建模初期要给出模型覆盖的问题范围；

（2） 建模计划

（A） 项目说明；

（B） 收集数据；

（C） 定义实体；

（D） 定义联系；

（E） 定义码属性；

（F） 定义非码属性；

（G） 确认模型；

（H） 评审验收。

（3） 组织队伍：包括项目负责人、建模者、信息源、课题专家、评审委员会

2、 第二阶段：定义实体集

（1） 目标是标识和定义应用领域中的实体集，方法是分类标识原始材料中的所有名词；

（2） 区别实体集名词和非实体集名词的方法，是否具有下列特征：

（A） 它能够被描述或说明吗？

（B） 有多少同类的实例吗？

（C） 每个实例可以被标识和区分吗？

3、 第三阶段：定义联系

（1） 标识实体集之间的联系：建立联系矩阵，联系矩阵由一个二维数组表示。把实体集沿水平和垂直两方向列出，分析两个实体间的联系，有联系就用“X”表示，不存在联系用“null”表示。联系只标识直接关系，不标识间接关系。

（2） 定义联系：包括表示依赖、命名联系、关于联系的说明；当实体集之间的依赖关系建立后，就可以命名联系了。联系的名字可以动词表示。原则必须是具体的、简明的和有意义的。

（3） 构造实体级数：实体级图的范围和数目，依赖于建模的规模和建模问题涉及的实体集数

目。

4、 第四阶段：定义健

（1） 分解不确定的联系：把实体级图中不确定的关系转换成确定的连接形式，把每一个不确定的联系转换成为两个确定的联系；

（2） 标识码属性：码属性是那些能够惟一识别实体集中每一个实例的属性；

（3） 迁移主码：把一个实体集的主码复制到其他有关实体集的过程，但要遵守以下规则：

（A） 在一个联系中，迁移总是从父到子或从一般实体集移向分类实体集；

（B） 主码属性才能被迁移，如主码由多个属性组成，则要全部迁移；

5、 第五阶段：定义属性

（1） 标识和定义非主属性；

（2） 建立属性的所有者；

（3） 确认属性的定义；

（4） 绘制局部数据视图；

（A） 实体集的名称和编号写在矩形框外的上面；

（B） 主码属性写在矩形框内水平线的上面并用“PK”标注；

（C） 外码属性写在矩形框内水平线的下面并用“FK”标注；

（D） 非主属性也可以写在矩形框内水平线的下面；

第五章 关系数据库逻辑设计

5．1 概述

5．2 基本概念

5．2．1 关系模型

1、 关系模型采用一个二维表格在计算机中组织、存储、处理和管理数据。

（1） 关系名（数据库名）：由字母数字组成；

（2） 属性名；

（3） 关系模式和关系：描述模式描述关系的静态结构，由模式名、关系模式所包含的属性及属性值所满足的条件组成模式定义。

（4） 元组：描述关系中的行；

（5） 域：它定义关系的每个属性取值的类型；

（6） 主码：能够惟一标识关系中每一个元组的属性或属性组；

（7） 关系的数学定义：关系模式是建立在集合集论的基础上的，用数学的概念定义关系有；

（A） 定义一：域是值的集合，同一个域中的值具有相同的数据类型；

（B） 定义二：笛卡尔积的定义（相当于一张二维表）

（C） 定义三：关系的定义（笛卡尔积的子集（二维表），域的顺序不能改）

（D） 当关系引用了属性名后关系具有以下属性：

[1] 不能有重复的元组；

[2] 元组上下无序；

[3] 按属性名引用时属性左右无序；

[4] 所有属性值都是原子项（不可再分）；

（8） 总结：关系是一张二维表，表中的一行被称为一个元组，一列称为属性，由一组域值组成。关系是元组的集合，关系中的每个元组在数学上被定义为这个关系所涉及的全部域值中笛卡儿积的一个元素。

5．2．2 关系数据库

1、 关系数据库是按照二维表组织和存储的相互关联的关系的集合，关系数据库模式是关系模式的集合；

5．2．3 关系的完整性

1、 关系的完整性（完整性约束）：是对关系的某种约束规则和关系满足的定义。通常这组约束规则用来限定和检查数据库所含实例的合法性和正确性；

2、 完整性约束分静态和动态两种，静态完整性（相当于逻辑的，真理的）约束是基于关系模式的，主要有主码、外码约束和域约束组成；动态完整性约束（基于现实）是基于企业的业务规则的。

3、 静态完整性约束规则：

（1） 主码约束：主码必须满足：

（A） 惟一性：在一个关系中不存在两个元组，它们具有相同的主码值；

（B） 最小性：不存在从组成主码的属性集中去掉一个属性，还仍能保持数据的惟一性；

候选码：符合主码条件，但没有被选为主码。

（2） 外码约束：（外码的定义：有两个关系 R 和 S，X 是 R 得属性组，非码；是 S 的码，则称 X 是 R 得外码）外码为空或码的值（否者查找失败）

（3） 用户定义的完整性：（性别只为男和女）

5．3 关系数据库设计理论

5．3．1 问题的提出

究竟一个关系数据库包含哪些属性是合理的，如何评价一个关系模式设计的优劣？（插入、更新、删除异常）

补充（关系代数）

1、 传统的集合运算。（并、交、差、笛卡尔积）；

2、 专门的关系运算。（选择、投影、连接、自然连接、除法）

5．3．2 函数依赖

函数依理论利用一个关系中属性之间的依赖关系评价和优化关系模式，以保证存储到数据库中的关系具有较好特性；

1、 函数依赖：

（1） 设 R（U）为一关系模式，X 和 Y 为属性全集 U 的子集，若对于 R（U）的任意一个可能的关系 r，r 中不可能存在两个元组在 X 上的属性值相等，而在 Y 上的属性值不等，则称“X 函数决定 Y”或“Y 函数依赖于 X”，并记作 X→Y，其中 X 称为决定因素，因为根据函数依赖定义，给定一个 X，就能惟一决定一个 Y。（翻译：只要 X 上的属性值相等，则 Y 上的属性只就相等）

（2） 这里讨论的函数关系与数学上的不同，是不能计算的，是一个关系中属性之间存在的依赖关系；它是一种语义范畴的概念，只能根据两个属性之间的语义来确定一个函数依赖是否存在。

2、 完全（f）与部分函数（p）依赖：

（1） 在关系模式 R（U）中，如果 X→Y 成立，并且对 X 的任何真子集 X’不能函数决定 Y，则称 Y 对 X 是完全函数依赖，被记作 X---f---→Y。

（2） 若 X→Y，但 Y 不完全函数依赖于 X，则称 Y 对 X 是部分函数依赖，记作 X--p→Y（即至少存在一个真子集 X’函数决定 Y）；

3、 传递函数依赖：

在关系 R（U）模式中，如果 X 决定 Y，（Y 不属于 X），Y 不决定 X，Y 决定 Z，则称 Z 对 X 传递函数依赖。

4、 平凡与非平凡函数依赖：

（1） 若 X 决定 Y，但 Y 属于 X，则称 X→Y 是平凡函数依赖，否则称非平凡函数依赖；

（2） 即平凡函数依赖，仅当其右边的属性集是左边属性集的子集时成立；（全属于）

（3） 非平凡函数依赖，仅当其右边的属性集至少有一个属性不属于左边有集合时成立；（部分）

（4） 完全非平凡函数依赖：仅当其右边的属性集中属性都不在左边的集合时成立；（都不属于）

5、 码：

（1） 在关系模式 R（U）中，K 为 R 的属性或属性组，若 K 完全函数决定 A1.A2….An，则 K 为

关系模式 R 的候选码（应理解为一个属性组），包含在候选码中的属性称为主属性，否则为非主属性；

（2） 若一个关系的候选码不止一个，则选定其中一个作为关系 R 的主码；

（3） 关系的码属性除了必须完全函数决定关系的所有其他属性外，还必须满足最小化规则，即在关系模式 R（U）中，不存在一个 K 的真子集能够函数决定 R 的其他属性。

6、 函数依赖的推理规则：（Armstrong 公理及推论）

（1） 自反律：若 Y（包含于）X（包含于）U，则 X→Y 成立；

（2） 增广律：若 X→Y，且 Z（包含于）U，则 XZ→YZ 成立；

（3） 传递律：若 X→Y，Y→Z，则 X→Z 成立；

推论

（4） 合并规则：若 X→Y，X→Z 成立，则 X→YZ；

（5） 分解规则：若 X→Y 和 Z（包含于）Y 成立，则 X→Z 也成立；

（6） 伪传递规则：若 X→Y，YW→Z，则 XW→Z 成立；

7、 属性集闭包：

（1） 设 F 是属性集 U 上的函数依赖集，X 为 U 的一个子集，那么对于 F，属性集 X 关于 F 的闭包（用 X+表示）为：X+={A|X→A}（最大的 Y 的集合）

（2） 由属性集团包的定义可知，若想判断函数依赖 X→Y 是否成立，只要计算 X 关于函数依赖集 F 的闭包，若 Y 是 X 闭包中的一个元素则 X→Y 成立；

8、 确定关系的码：

（1） 利用迭代算法计算 X+，步骤如下：

（A） 选 X 作为闭包 X+的初值 X（0）；

（B） 由 X（i）计算 X（i+1）时，它是由 X（0）并上属性集合 A 所组成，其中 A 满足下列条件：Y（包含于）X（i），且 F 中存在函数依赖 Y→Z，而 A（包含于）Z。因为 U 是有穷的，所以会得到 X（i）=X（i+1），此时 X（i）为所求的 X+。

5．3．3 规范化设计方法

1、 第一范式：

（1） 定义：设关系模式 R（F，U），如果 R 的每一个属性都是不可分的数据项，则此关系模式为第一范式；

（2） 一个给定关系和第一范式（1NF）的区别：

（A） 一个关系中的数据按照行和列的形式组织，每个元组具有相同数目的属性个数，且每一个元组的属性值具有统一的数据类型和长度；元组或属性的排列与顺序无关，

每个元组必须通过一个属性或属性组惟一识别；

（B） 第一范式实际上对关系增加了一个约束，即关系中元组的每个属性都只取一个值，第一范式是对关系模式的基本要求，不满足第一范式的数据库就不是关系数据库。

2、 第二范式：

（1） 定义：若关系模式 R（F，U）是 1NF，且每个非主属性完全函数依赖于码，则称 R 为第二范式，即在 2NF 中不存在非主属性对码的部分依赖；

（2） 仅满足第一范式关系会存在种种问题，要消除必须用更高级的范式标准来设计，称为标准化；

（3） 具体做法是将大的关系分解成多个小的关系，使分解后的关系满足更高级范式的要求。

（4） 第二范式实际上对关系增加了一个约束，就是关系中的每一个属性必须完全依赖于主码（造成分表），即在第一范式的基础上，消除非主属性对主码的部分函数依赖可达到 2NF；

3、 第三范式：

（1） 定义：若关系 R（U，F）为第一范式，且不存在非主属性对主码的传递函数依赖，则称 R 为第三范式；

（2） 第三范式是在第二范式的基础上对关系又增加了一个约束，就是关系中的每一个非主属性必须只依赖于主码。即 2NF 的基础上，消除非主属性对主码的传递函数依赖可达到 3NF。

4、 改进的第三范式（BCNF）：

（1） 定义：如果关系模式 R 是 1NF，且每个属性既不存在部分函数依赖也不存在传递函数依赖于候选码，则称 R 是改进的第三范式（BCNF）。

5、 多值依赖与 4NF：

（1） 多值依赖：表示关系中属性（如 A、B、C）之间的依赖，对于 A 的每个值，都存在一个 B 或 C 的值的集合，而且 B 和 C 的值相互独立，记为：A→→B、A→→C

（2） 第四范式：如果关系模式 R 属于 1NF，对于 R 的每个非平凡的多值依赖 X→Y(Y 不属于 X)，X 含有候选码，则 R 是第四范式。即是从 BCNF 范式中消除主码内的独立依赖集（非平凡多值依赖）可达 4NF；

6、 连接依赖与 5NF

（1） 连接依赖：设关系模式 R，R 的属性子集为 R1、R2、R3、R4、R5、R6、R7….，当且仅当 R 的每个合法值等于 R1、R2、R3、R4、R5、R6、R7…的投影连接时，称 R 满足连接依赖；

（2） 第五范式：设 R 是一个满足 5NF 的关系模式，当且仅当 R 的每一个非平凡连接依赖都被 R 的候选码所蕴含，即从 4NF 中消除非候选码所蕴含的连接依赖为 5NF；

7、 总结：

（1） 范式表达了关系模式满足的条件，也是衡量关系模式设计优劣的标准；

（2） 利用范式进行规范化设计的目的是消除数据冗余，避免出现异常，使结构更合理；

（3） 规范化设计的基本过程是对关系进行的分解，消除属性间不合理的数据依赖，用一组等价的子关系代替原有的关系；

（4） 数据库规范化的程序越高，其关系表就越多，从而增加了表之间连接运算的代价，影响了数据库的执行速度和性能。所以通常关系模式规范化工作仅做到 3NF，这样既使关系中不合理的属性基本消除，规范化程度也不太高，保证数据库有较好的性能。

5．4 数据库模式设计

5．4．1 初始关系模式的设计

1、 把 ER 图转换成关系模式：

（1） 把 ER 模型中的每个实体集转换成一个同名的关系，实体集的属性就是关系的属性，实体集的码就是关系的码；

（2） 把 ER 模型中的每个联系转换成一个关系，与该联系相连的各实体集的码以及联系的属性转换成为关系的属性。

（A） 若联系为 1：1，则每个实体集的码均是该关系的候选码；

（B） 若联系为 1：n，则关系的码为 n 端实体集的码；

（C） 若联系为 m：n，则关系的码为各实体集码的组合；

（3） 合并具有相同码的关系

2、 检查确认对象：检查转换后的每个关系名和属性名是否符合数据库设计关于统一命名的约定；

5．4．2 优化关系模式

1、 模式分解原则：

（1） 分解具有无损连接性：分解后的关系能够恢复成原来的关系；

（2） 分解保持函数依赖：

（A） 无损连接和保持函数依赖是用于衡量一个模式分解是否导致原有模式中部分信息丢失的两个标准；

（B） 当一个关系被分解后会出现几种结果，既有无损连接，又能保持函数依赖是较理想的分解结果，意味着在分解的过程中没有丢失原有模式的任何信息；

（C） 一般情况下，分解到 3NF 就足够了，但在 3NF 关系下，仍存在一定程度上的更新异常或不一致的隐患，但与数据库性能比较起来是可以忽略的，因为在数据库设计过程中通过增加一些数据约束，就可以解决 3NF 引起的数据问题了。

3、 优化属性：确定各字段的类型和长度；

4、 确认模式满足需要：

5．4．3 数据完整性设计

1、 指定义数据库中存储的数据值满足的约束条件，通过对存储的数据值的约束维护关系的完整性。

2、 数据值满足条件分为：

（1） 域约束：限制指定列的取值及范围；

（2） 主码约束：定义每个关系的主码值不空，且惟一；

（3） 引用完整性约束：定义不同模式的属性间满足的条件，及一个关系模式中属性间可能满足的条件；

5．4．4 安全模式和外模式的设计

1、 根据选定的 DBMS 支持的安全控制特征来确定；

2、 根据不同用户对数据库存取特点定义相关的外模式；

第六章 存储技术与数据库物理设计

6．1 文件组织

6．1．1 数据库的物理结构

1、 数据库中的应用数据是以文件形式存储在外存上的，文件在逻辑上被组织成记录的序列，即每个 DB 文件可看作是逻辑记录的集合；

2、 一个文件在磁盘上占有一定的物理存储空间，文件中的每个逻辑记录被映射存储到某个特定的磁盘块上，一个文件在物理上可以看作是由存放文件记录的一系列磁盘块组成，称为物理文件；

3、 文件的逻辑记录与磁盘间的映射关系是由操作系统或 DBMS 来管理的，当需要对一个文件的逻辑记录进行操作时，先要根据这种映射关系找到该逻辑记录所在的磁盘块，然后再进行操作。

4、 从数据库物理结构角度需要解决如下 4 个问题：

（1） 文件的组织；

（2） 文件的结构；

（3） 文件的存取；

（4） 索引技术；

6．1．2 文件组织

1、 数据库与文件的对应关系

（1） 在外存中，数据库以文件形式组织，文件由逻辑记录组成，记录由多个域组成；

（2） 一个关系数据库包括一张或多张关系表，关系表与文件的对应关系有如下方式：

（A） 每张关系表单独用一个文件来存储，由 DBMS 通过 OS 的文件管理功能来管理；

（B） 现代中大型 DBMS 是由 OS 直接分配一块大的磁盘空间，DBMS 将该磁盘空间作为数据库磁盘文件直接管理，DB 的所有关系表都存储在该文件中；

（4） 关系表在逻辑上由一系列元组组成，元组由多个属性组成，每个元组可以用磁盘文件中的一个逻辑记录来存储，记录包括多个域，对应元组的多个属性；

2、文件记录格式：

（1） 数据库文件通常采用两种逻辑记录格式：定长记录格式和变长记录格式；

6．2 文件结构与存取

6．2．1 堆文件

1、 堆文件也称无序文件，记录随机在存储在文件物理空间是，新插入的记录存储在文件的末尾；

2、 堆文件常常用作存储那些将来使用，但目前不清楚如何使用的记录，为了实现文件记录的有效存取，堆文件经常与附加的存取路径一起使用；

3、 查找操行平均需要搜索（B+1）/2 个磁盘块，效率比较低；

4、 插入操作十分简单，先读文件头，找到最末磁盘地址，将最末磁盘块读入内存，将需插入的新记录写入磁盘块的末端，最后将修改过的磁盘块写回磁盘；

5、 删除比较复杂，可以先找到被删除记录所在的磁盘块，读入内存后在内存缓冲区删除记录，最后再写回磁盘；也可以在每个记录的磁盘空间增加一个删除标志位，当需要删除记录时，将标示位置 1；

6．2．2 顺序文件

1、 顺序文件按照文件记录在查询码上的取值的大小顺序排列各个记录；

2、 顺序文件的每个记录中有一个指针字段，根据查询码大小用指针将各个记录按序连接起来；

3、 文件建立时，应尽量使记录的物理顺序与查找码的顺序一致，以减少访问磁盘块的次数；

4、 根据查询条件对顺序文件进行查询时，如查询条件定义在查找码上，则使用二分法查找技术快速找到记录，如条件不在查找码上，则必须从头到尾依次扫描磁盘块，与堆文件一致，所以顺序文件的访问效率也不高；

5、 顺序文件插入工作包括定位和插入：

（1） 定位：在指针链中找到插入的位置，即插入记录在哪个记录的前面；

（2） 插入：如有自由空间，则在该位置插入新记录，如没有自由空间，则只能插入溢出块中，重新调整记录指针链关系，保证记录顺序；

6．2．3 聚集文件

1、 聚集文件是一种具有多种记录类型文件，存储了来自多个关系表的数据，每个关系表对应文件中的一种记录类型；

2、 当数据库中数据量效大时，对数据库查询需要多次访问磁盘文件，严重影响性能指标，为了降低多表操作时的磁盘访问次数，提高多表查询速度，可采用聚集文件；

3、 聚集文件将不同关系表中有关联关系的记录存储在同一磁盘块内，从而减少多表查询时磁盘块的访问次数，提高处理速度；

6．2．4 索引文件

是一种利用索引技术技术快速文件访问的文件组织和存取方法；

6．2．4 散列文件

是一种利用散列函数支持快速文件访问的文件组织和存取方法；

6．3 索引技术

6．3．1 基本概念

1、 索引技术：是一种快速文件访问技术，它将一个文件的每个记录在某个或某些域（属性）上的取值与该记录的物理地址直接联系起来，提供了一种根据记录域的取值快速访问文件记录的机制；它的关键是建立取值域到记录的物理地址的映射关系，这种映射关系叫索引；

2、 索引技术分类：

（1） 有序索引技术：利用索引文件（相当于书的目录）实现记录域（查找码）取值到记录物理地址间的映射关系，索引文件由索引记录组成，每个记录中记载一个索引项，索引项记录了某个特定的查找码值和具有该值的数据文件记录的物理地址；

（2） 散列技术：利用一个散列函数实现记录域取值到记录物理地址间的直接映射关系；

3、有序索引：有序索引作为基于索引文件的索引技术，需要考虑两个问题：

（1）如何组织索引文件中的索引记录；（2）如何从索引文件出发，访问数据文件中的数据记录；

（A） 当需要采用有序索引机制快速访问数据文件时，首先要为该数据文件建立一个索引文件，它是索引记录和索引项的集合；

（B） 索引文件建立的方法：首先选定某些记录域作为查找码，然后建立数据记录在查找码上的取值与物理地址间的映射关系，组成索引项。所有索引项作为索引记录存储在索引文件中，索引文件根据某个特定的查找码值的顺序组织为顺序文件；

（C） 一个数据文件可以有多个（提供的查询方式可多样）查找码和索引文件；

6．3．2 有序索引的分类及特点

1、 聚集索引与非聚集索引

（1） 对数据文件和它的一个特定的索引文件，如果数据文件中数据记录的排列顺序与索引文件中索引项的排列顺序相一致，则该索引文件称为聚集索引，否则称为非聚集索引；

（2） 在一个数据文件上除了建立一个聚集索引外，还可建立多个非聚集索引；

2、 稠密索引和稀疏索引

如果数据文件中的每个查找码都在索引文件中都对应一个索引记录，称为稠密索引，如果只一部分对应，则称为稀疏索引；

3、 主索引和辅索引

在数据文件包含主码的属性集上建立索引称为主索引，在非主码属性上建立的索引称为辅索引；

4、单层索引和多层索引

（1） 单层索引（线性索引）：索引项根据键值在索引文件中顺序排列，组织成一维线性结构，每个索引项直接指向数据文件中的数据记录；

（2） 当数据文件很大时，即使采用稀疏索引，建成的索引文件也很大，导致效率低下，为解决该问题，可对索引文件中的索引项本身再建立一级稀疏索引，组成 2 层索引结构；进一步地，可建立多层树型索引结构来快速定位；

6．4 散列技术

6．4．1 散列文件

1、 散列是一种快速查找技术，它利用定义在文件记录上的查找码，通过计算一个散列函数，以散列函数值作为记录的物理地址，实现对文件记录直接快速访问。

2、 首先指定文件记录的一个域作为查找码（散列域），然后定义一个查找码上的函数（散列函数），函数的输入为查找码值，输出为物理地址；

3、 一般使用桶作为基本的存储单位，一个桶可存放多个文件记录，物理地址可以是记录所在的桶号，散列函数的输出可以是桶号；

6．4．2 散列函数

1、 散列方法依赖于好的散列函数，它应该尽可能均匀地将查找码分布到各个桶中，具体要满足如下两个条件：

（1） 地址的分布是均匀的；

（2） 地址的分布是随机的；

6．4．3 桶溢出

1、 产生桶溢出的两个原因：

（1） 文件初始设计时，为文件记录预留的存储空间不足；

（2） 散列函数的均匀分布性不好；

2、 设计散列函数时，应根据文件大小决定物理空间，一般应有 20%余量，再设计合适的桶数目和桶大小，尽可能留有一些空闲桶，降低桶溢出的可能性；

3、 桶溢出的现象是难免的，需要 DBS 采用相应的桶溢出处理机制；

4、 散列方法的缺点：为了避免桶溢出。必须选一合适的散列函数，但这比较复杂，而且不象索引文件那样可以据数据记录变化动态调整。

6．5 数据字典

1、 数据字典（系统目录）中存储了数据库对象的各类描述信息和 DBMS 所需的控制信息，全称数据库元数据；

2、 数据库对象的各类描述信息：包括外模式、模式、内模式以及它们之间的映射的描述；

3、 DBMS 所需的控制信息：包括查询优化、安全性检查、用户权限验证等；

4、 数据字典主要包括：

（1） 关系模式信息；

（2） 与视图描述有关的信息；

（3） 关系的存储结构和存取方法信息；

（4） 完整性约束信息；

（5） 安全性有关信息；

（6） 数据库运行统计信息；

6．6 数据库物理设计

6．6．1 设计步骤和内容

1、 数据库物理结构设计：在具体的硬件环境、OS、DBMS 约束下，根据数据库逻辑设计结果，设计合适的数据库物理结构。目标是存储空间占用少、访问效率高和维护代价低；

2、 一旦选定了硬件平台、OS 和 DBMS，数据库的数据存储和存取方式等可用的物理模式也就随之确定了；

3、 数据库物理设计主要包括以下步骤：

（1） 数据库逻辑模式调整：将数据库逻辑模式及其视图转换为 DBMS 支持的基本表和视图，并利用 DBMS 提供的完整性机制设计业务规则；

（2） 文件组织与存取设计：配置基本表的文件组织形式，据实际情况为基本表设计合适的存取方法和路径；

（3） 数据分布设计：

（4） 安全模式设计：

（5） 确定系统配置：

（6） 物理模式评估：

6．6．2 数据库逻辑模式（是独立于具体的 DBMS 的）调整

1、 物理数据库设计首先需要根据数据库逻辑结构信息，设计目标 DBMS 平台支持的基本表的模式信息，这些模式信息代表了所要开发的具体目标数据库的结构，这个过程称为数据库逻辑模式调整，主要包括如下设计内容：

（1） 实现目标数据库基本表和视图：采用目标 DBMS 所支持的建表方法，设计基本表及其面向模型的完整性约束；

（2） 设计基本表业务规则；（如 check）

6．6．3 DB 文件组织与存取设计

1、分析事务的数据访问特性

（1） 使用事务-基本表交叉引用矩阵，分析系统内数据库事务对各个基本表的访问情况，确定事务访问了哪些基本表，对这些基本表执行了何种操作，并进一步分析各操作涉及到的基本表属性；

（2） 估计各事务的执行频率；

（3） 对每张基本表，汇总所有作用于该表上的各事务的操作频率信息；

2、 了解并选择数据库文件结构

（1） 如果数据库中的一个基本表中的数据量很少，并且操作非常频繁，该基本表可采用堆文件组织方式；

（2） 顺序文件支持基于查找码的顺序访问，也支持快速二分查找；

（3） 如果用户查询是基于散列域值的等值匹配，特别是如果访问顺序是随机的，散列文件比较合适。但散列文件组织不适合以下情况：

（A） 基于散列值域的非精确查询；

（B） 基于非散列域进行查询时；

（4） B-树和 B+树文件是实际数据库系统中使用非常广泛的索引文件结构，适合于定义在大数据量基本表上、基于查找码的等值查询等；

（5） 如果某此重要而频繁的用户查询经常需要进行多表连接操作，可考虑将这些基本表组织为聚集文件；

3、 设计存取路径：

（1） 为数据库文件设计合理的物理存储位置；

（2） 为基本表设计索引机制（加快了数据查询速度，减慢了数据更新速度）：索引可以提高文件存取速度，改善访问性能，但索引由 DBMS 管理，它的建立、维护需要一定的系统开销，数据的操作会引起索引的重新调整，还占用一定的存储空间，可根据如下原则决定是否为一个基本表建立索引：

（A） 对于经常需要查询、连接、统计操作，且数据量大的基本表可考虑建立索引，而对于经常执行插入、删除、更新操作或小数据量的基本表应尽量不建立索引；

（B） 一个基本表上除了可以建立一个聚集索引外，还可以建立多个非聚集索引，但索引越多，对表内数据更新所需的开销越大，对于一个更新频繁的表应少建或不建索引；

（C） 索引可以由用户根据需要随时创建或删除，以提高数据查询性能；

6．6．4 数据分布设计

1、不同类型数据的物理分布

（1） 各种数据在系统中的作用不同，使用的频率也不一样，应根据实际使用情况放在合适的物理介质上；

（2） 使用频率低但数据量大的，可以放在磁带中，而使用频繁，要求响应时间短的，必须放在支持直接存取的磁盘存储介质上；

2、 应用数据的划分和分布

（1） 根据数据的使用特征划分：可将基本表划分为频繁使用分区和非频繁使用分区，分别存放在不同的磁盘上，对前者可考虑建立 B+树等多层索引，而后者不建立或只建立单层索引；

（2） 根据时间、地点划分；

（3） 分布式数据库系统中的数据划分：

3、派生属性数据分布

（1） 派生属性指该属性的取值可根据表中其他属性的取值惟一确定；

（2） 对带有派生属性的基本表可采用两种实现方式：

（A） 将派生属性作为基本表内单独一列，称为派生列；

（B） 派生属性不出现在基本表中；

4、 关系模式的去规范化

（1） 在数据库物理设计阶段，可以对考虑数据库中某些 3NF、BCNF 模式是否可以降低其规范化程度，以提高查询效率，这称为关系模式的去规范化处理，但不满足 3NF 的关系模式又可能导致数据库访问异常，因此，设计基本表时，需在规范化和查询效率间权衡；

6．6．5 安全模式设计

1、系统安全设计

（1） 是指为数据库服务器合法用户分配用户名和口令，使其能够正常登录服务器访问所需的数据，还可采用基于 CA 认证的系统安全控制机制；

2、 数据安全设计

（1） 是指通过数据库系统视图机制和授权机制为用户对数据库对象访问的权限；

（2） 引用数据视图机制，只给用户需求的那部分数据访问权限，防止由合法用户造成信息泄密，另外数据视图还可以防止基本表发生改变时，影响用户的访问；

（3） 权限是允许用户对一给定的数据库对象可执行的操作；

（4） 数据库安全设计需要根据用户需求，采用授权机制，为用户分配合法访问的权限；

6．6．6 确定系统配置

1、 要根据实际应用系统的运行情况配置系统参数；

6．6．7 物理模式评估

1、 在设计过程中，通过对时间效率、空间效率、维护代价和用户要求权衡考虑，择优采用；

2、 评估物理数据库的方法完全依赖所选用的 DBMS，主要从定量估算各方案的存储空间、存取时间和维护代价入手；

第七章 数据库应用系统功能设计

7．1 软件体系结构与设计过程

7．1．1 软体体系结构

1、 软件体系结构又称软件架构，软件体系结构={构件，连接件，约束}。

2、 构件是组成系统的具有一定独立功能的不同粒度的程序模块、独立程序或软件子系统，是组成软件的系统元素；

3、 连接件将不同的构件连接起来，表示了构件间的相互作用；

4、 约束一般是对象连接时的规则，或指明了构件连接的条件。

5、 软件体系结构描述了软件系统的总体组织和层次结构、系统元素及其功能分配、全局控制、系统元素间的协调和交互、数据存取等；

7．1．2 软件设计过程

1、 概要设计

（1） 定义：是建立软件系统的总体结构和模块间的关系，定义各功能模块的接口，设计全局数据库、规定设计约束、制定组装测试计划；

（2） 一个好的概要设计要求是：良好的总体结构、功能模块间较低的耦合度和较高的内聚度，

并尽量降低模块接口的复杂性；

（3） 可以采用层次结构图表示软件总体结构，图中节点代表功能模块。

2、 详细设计

（1） 是细化概要设计产生的功能模块，形成可编程的程序模块，并用某种过程设计语言设计程序模块的内部细节，为编写软件代码提供依据。

（2） 可选用结构化设计方法、面向对象设计方法等；

3、 关于软件总体设计

（1） 一些大的 DBAS 可根据逐步抽象和层次化原则，将概要设计分解成两个步骤：

（A） 首先是软件总体结构设计，即对软件需求进行分解；

（B） 第二步是将每个子系统进一步划分为功能模块，定义各模块的数据结构、相互间交互关系；

7．2 DBAS 总体设计

7．2．1 系统总体设计

任务：是根据系统规划与分析结果，特别是技术可行性分析，以及系统需求规范，确定系统总体框架，作为后续设计活动的基础。

1、 确定 DBAS 体系结构

（1） 指将系统从功能、层次结构、地理分布等角度进行分解，划分为多个子系统。定义各子系统应实现的功能，设计全局控制，明确各子系统间的交互和接口关系；

（2） 可以从功能角度进行分解，也可以根据 DBAS 自身固有的层次结构特征进行分解；

（3） 将系统分解为多个子系统后，需选择和设计合适的系统体系结构，将这些子系统组织起来，并设计它们之间的交互关系；

（4） DBAS 体系结构可采用一些通用体系结构，也可根据 DBAS 所属的特定应用领域相关的体系结构。

2、软硬件造型和配置设计

（1） 总体设计阶段需要对系统的软硬件平台、存储设备、操作系统、数据库管理系统等作出合理的选择，并进行初步配置设计；

（2） 还需要选择系统开发采用的合适的中间件和开发工具，确定开发模式和开发语言；

3、应用软件总体设计

根据系统体系结构，确定相应的软件系统模块划分、功能分配，选择合适的软件体系结构；

4、业务规划初步设计

7．2．2 软件总体设计

1、 DBAS 软件包括 OS、DBMS、开发环境、中间件和应用软件；

2、 应用软件分为数据库事务和应用程序；

3、 数据库事务通过对数据库的直接操作实现数据管理和处理功能；

4、 应用程序一方面对数据库进一步加工处理，或从中抽取新信息实现复杂的数据处理功能；另一方面还可实现与数据库访问无关的功能；

5、 应用软件总体设计：

（1） 从数据流图、事务规范和业务规则需求分析结果出发，将系统分解为一系列子系统，分配相应功能，定义系统间协调交互机制；

（2） 进一步进行子系统结构设计，将各子系统从功能上划分为：数据库事务模块和应用程序模块；

（3） 确定子系统、应用程序模块、数据库事务间的全局控制和调用关系，并按体系结构框架组织起来。

6、总体设计得到的系统总体结构和分层模块结构，可以用模块结构图表示；

6、 模块结构图，是结构化程序设计中描述系统结构的一种图形化工具，它定义了模块的名字、功能和接口，并在模块结构图中反映出结构化设计思想。它只关心模块的外部特性，与模块内部流程无关，它由模块、调用、数据、控制和转接等于种基本符号组成；

7．2．3 客户/服务器体系结构

1、 基于 C/S 体系结构的 DBAS 将 DBMS 数据管理功能与数据库应用相分离，将 DBMS 数据库管理功能在客户端和服务器之间进行合理的分布和配置；

2、 数据库报务器完成 DBMS 的核心功能，而客户端负责完成用户交互功能，接收用户数据，生成并向数据库报务器发出数据操作请求，接收数据查询结果并通过客户端反馈给用户；

3、 两层 C/S 结构的特点是：

（1） DBAS 的数据管理和处理功能，被分解并分布在客户端和服务器上；

（2） 服务器能为多个客户端应用提供共享的数据管理功能；

（3） 客户端应用可通过网络访问多个不同数据源；

（4） 客户端除了完成人机交互功能外，还需要完成面向应用的数据处理功能，负荷重，属于典型的“胖客户端”；

4、 三层浏览器/服务器（B/S）结构是一种互联网环境下的新型数据库应用系统结构，它将数据处理功能分解并分布在表示层、功能层和数据层三层次上，分别由 WEB 浏览器、WEB 服务器和数据库服务器来实现，其特点是：

（1） 表示层位于客户端，由 WEB 浏览器实现，其功能单一，没有其他应用程序，属于典型的

“瘦客户端”；

（2） 功能层位于 WEB 服务器，实现面向具体应用领域的业务规则；

（3） 数据层位于数据库服务器，通过 DBMS 完成具体的数据存储和存取等数据管理功能；

7．3 概要设计

前言：DBMS 逻辑设计包括：DB 逻辑结构设计（前面已涉及）和 DBAS 概要设计（事务概要和应用程序概要）。事务概要注重事务本身的处理流程；事务详细则具体考虑 DBMS 相容性。

7．3．1 数据库事务概要设计

1、 如数据处理需求分析的结果是数据流图，则可将待设计的事务看作是程序，采用软件工程中面向数据流的程序设计方法，设计事务内部的数据处理流程和结构，也就是设计事务处理逻辑，过程包括：

（1） 从数据流图中识别出该事务对应的子数据流图；

（2） 确定子数据流图中的信息流类型，划定流界；

（3） 将子数据流图映射为事务的结构和处理流程，即事务逻辑；（事务的业务规则）

（4） 修正和细化事务设计，识别事务所访问的数据库对象和数据库用户；

2、 如数据处理需求分析的结果表示为事务规范，由于事务规范包括了事务名称、事务描述、访问的数据项、用户等信息，可直接从事务描述出发，根据具体应用领域的知识设计事务逻辑，得到事务概要结果；

3、 一个完整的事务概要设计包括：事务名称、访问的关系表及属性、事务处理逻辑（read、write 两个元操作，与 DBMS 无关）、事务用户；

4、 检查关系表对数据库事务的支持性：

（1） 对每一个事务，根据需求分析阶段的事务分析，列出该事务所访问的各个数据项；

（2） 列出事务访问的数据项所在的关系表和对应的属性；

（3） 如事务访问的数据项同时出现在多个表中，检查关联关系；

（4） 检查是否存在某些事务，访问的一些数据项未出现在任何关系表中；

7．3．2 应用软件概要设计

1、 应用软件概要设计，按照逐步求精、模块化、信息隐藏和功能细化原则，根据 DBAS 需求分析阶段得到的系统功能和业务规则描述，在总体设计结构基础上，将 DBAS 应用软件进一步细化为模块/子模块，组成软件的系统-子系统-模块-子模块层次结构，并对这些系统元素从结构、行为和数据三方面进行设计；

7．4 详细设计

7．4．1 数据库事务详细设计

1、 事务详细设计，是从事务概要设计得到的事务流程出发，在 DBMS 平台下，采用事务实现机制，和高级程序设计语言，利用 SQL 语句和数据库访问接口，在 DBMS 平台和开发环境下，进一步细化事务设计，设计具体的实现模式；

7．4．2 应用软件详细设计

1、根据概要设计中定义的各程序模块功能和输入输出数据需求，结合具体的设计环境和机制，设计各模块的内部处理流程和算法、数据结构、对外接口等；

7．5 人机界面设计

1、 人机界面设计原则：

（1） 用户应当感觉系统的运行始终在自己的控制之下，保持用户与人机界面间的双向交流；

（2） 当系统发生错误或程序运行时间较长时，用户界面应该为用户提供有意义的反馈信息；

（3） 应该忍受用户在使用过程中发生的各种操作错误，并能够方便地恢复过来，保证系统不受或少受影响；

（4） 应该遵循一定的标准和常规；

（5） 采取灵活多样的数据输入方式，尽量减少用户数据输入负担；

2、 人机界面设计最好采用原形迭代法：

（1） 初步设计

（2） 用户界面细节设计；

（3） 原形设计与改进；

第 8 章 关系数据库操作语言 SQL

8．1 SQL 支持的数据类型

8．1．1 数值型

1、 准确型

2、 近似型

8．1．2 字符串型

1、 普通编码字符串类型；

2、 统一编码字符串类型—Unicode 编码；

3、 二进制字符串类型；

8．1．3 日期时间类型

8．1．4 货币类型

8．2 定义和维护关系表

8．2．1 关系表的定义与删除

1、定义表

CREATE TABLE <表名> (<列名><数据类型>[列级完整性约束定义]{, <列名><数据类型>[列级完整性约束定义]…}[,表级完整性约束定义]（not null 和 default 不能在表级定义））

1、 列级完整性约束：

（1） NOT NULL：取值非空；

（2） DEFAULT：指定列的默认值，形式：DEFAULT 常量；

（3） UNIQUE：列取值不重复；

（4） CHECK：限制列的取值范围，形式：CHECK（约束表达式）；

（5） PRIMARY KEY：指定本列为主码；

（6） FOREIGN KEY：定义本列为引用其他表的外码；（foreign key 列名 references 外表名）

2、 删除表

DROP TABLE <表名>

8．2．2 修改表结构

ALTER TABLE <表名>（添加 add，删除 drop，修改 alter）

常用： column

8．3 数据操作语言

8．3．1 数据查询

1、查询语句的基本结构：

SELECT <目标列名序列> FROM <数据源> （into 列名 3）

{

1、普通选择：WHERE + 选择条件（p138）（结果是选出行）

2、排序（选出的结果排序）：ORDER BY（ASC/DESC）

3、分组:GROUP BY（对列值（属性值），相同为一组）；HAVING（依据什么条件来分组，相当于对于分组时的 WHERE）

4、聚合函数统计数据。

}

（1） 比较：SELECT A，B，C FROM TABLE_A WHERE A>30；

（2） 确定范围：WHERE A （NOT）BETWEEN 初始值 AND 结束值；

（3） 确定集合：WHERE A （NOT）IN （‘A1’，‘A2’….‘A3’）；

（4） 字符串匹配：WHERE A LIKE <匹配符>；

（5） 四种<匹配符>：

（A）_（下划线）：匹配任意一个字符；

（B）%（百分号）：匹配 0 个或多个字符；

（C）[ ]：匹配[ ]中的任意一个字符；

（D）[^]：不匹配[ ]中的任意一个字符；

（6） 涉及空值的查询：WHERE A IS （NOT）NULL；

（7） 多重条件查询：AND（条件必须全部为 TRUE，结果才为 TRUE），OR（任一条件为 TRUE，结果即为 TRUE）；

（8） 对查询结果进行排序：ORDER BY A [ASC（顺序） | DESC（逆序）]；

（9） 列别名：列名 AS 新列名；

（10） 消除取值相同的行：SELECT DISTINCT A FROM TABLE_A；

（11） 使用聚合函数统计数据：SQL 的聚合函数：

（A） COUNT（*）：统计表中元组的个数；

（B） COUNT（[ALL （全部）| DISTINCT（无重复）] <列名>）：统计本列非空列值的个数；

（C） SUM（列名）：计算列值的总和（必须是数值型列）；

（D） AVG（列名）：计算列值平均值（必须是数值型列）；

（E） MAX（列名）：求列最大值；

（F） MIN（列名）：求列最小值；

（12） 对查询结果进行分组计算：

（A） 使用 GROUP BY；

（B） 使用 HAVING 子句；

3、 连接查询；（对于多张表）

（1） 内连接：FROM 表 1 JOIN 表 2 ON （连接条件）；（

一个表 连接的根本原因在于，存在有多个表，记录了主码与不同行为的关系，而目的是需要统计主码与不同行为的总体情况）

（2） 自连接：一种特殊的内连接，相互连接的表在物理上是同一张表，但通过为表取别名的方法，在逻辑上分为两张表；

（3） 外连接：输出不满足连接条件的元组，格式：

FROM 表 1 LEFT|RIGHT（表一/二全，也就是说，有满足和不满足的） OUTER JOIN 表 2 ON （连接条件）

4、 查询语句的扩展：

（1） 合并多个结果集：SELECT 语句 1 UNION SELECT 语句 2……，使用 UNION 的两个基本规则：All 的作用

（A） 所有查询语句中列的个数和列的顺序必须相同；

（B） 所有查询语句中对应的数据类型必须兼容；

（2） 将查询结果保存到新表中：SELECT 查询列表序列 INTO 新表名 FROM 数据源；

（3） 使用 TOP 限制结果集行数：TOP n [percent] [WITH TIES]

（A） TOP n ：表示取查询结果的前 n 行；

（B） TOP n percent：表示取查询结果的前 n%行；

（C） WITH TIES：表示包括并列的结果；

（4） 使用 CASE 表达式：

（A） 简单 CASE 表达式：

（B） 搜索 CASE 表达式；

5、子查询：如果一个 SELECT 语句是嵌套在一个 SELECT、INSERT、UPDATE 或 DELETE 语句中，则称为子查询或内层查询，包含子查询的语句称为主查询或外层查询；

（1） 使用子查询进行基于集合的测试，形式：WHERE 表达式 [NOT] IN （子查询）；

（2） 使用子查询进行比较测试，形式：WHERE 表达式 比较运算符 （子查询）；

（3） 使用子查询进行存在性测试，形式：WHERE [NOT] EXISTS （子查询）；

8．3．2 数据修改

1、 添加数据：INSERT [INTO] 表名 VALUE 值列表；使用插入单行语句时要注意：

（1） 值列表中的值与列名表中的列按位置顺序对应，要求它们的数据类型必须一致；

（2） 如果[表名]后边没有指明列名，则值列表中的值的顺序必须与表中列的顺序一致，且每一列均有值；

2、 更新数据：形式 UPDATE 表名 SET [列名（值）=表达式] [WHERE 更新条件（选择行）]；

3、 删除数据:：形式 DELETE [FROM] 表名 [WHERE 删除条件]；

8．4 索引

1、 创建索引：CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED]

INDEX 索引名 ON 表名

（1） UNIQUE：表示要创建的索引是唯一索引；

（2） CLUSTERED：表示要创建的索引是聚集索引；

（3） NONCLUSTERED：表示要创建的索引是非聚集索引；（默认）

2、 删除索引：DROP INDEX 索引名；

8．5 视图

8．5．1 定义视图

1、 语法格式：CREATE VIEW 视图名 AS SELECT 语句 [WITH CHECK OPTION]

2、 需要注意下列几点：

（1） 在定义视图时要么指定全部视图列，要么全部省略不写。如果省略了视图列名，则视图的列名与查询语句的列名相同。但如下情况则要明确指出组成视图的所有列名：

A、 某个目标列不是单纯的属性名，而是计算函数或列的表达式；

B、 多表连接时选出了几个同名列作为视图的字段；

C、 需要在视图中为某个列选用新的更合适的列名。

（2） WITH CHECK OPTION 选项表示通过视图对数据进行增加、删除和更改操作时（以后）要保证对数据的操作结果要满足定义视图时指定的 WHERE 子句条件；

3、 视图通常用于查询数据，也可修改基本表中的数据，但不是所有的视力都可以这样。

4、 定义单源表视图—视图数据可只取自一个基本表的部分行、列，这样的视图行列与基本表行列对应，这样定义的视图一般可以进行查询和更改数据操作

5、 定义多源表视图—视图数据可以来自多个表中，这样定义的视图一般只用于查询，不用于修改数据。

6、 在已有视图上定义新视图—可以在视图上再建立视图，这时作为数据源的视图必须是已经建立好的。

7、 定义带表达式的视图—在定义基本表时，为减少数据库中的冗余数据，表中只存放基本数据，由基本数据经过各种计算派生出的数据一般是不存储的。所以定义视图时可以根据需要设置一些派生属性列，在这些派生属性列中保存经过计算的值。这些派生属性由于在基本表中并不实际存在，因此，也称它们为虚拟列。包含虚拟列的视图也称为带表达式的视图。

8、 含分组统计信息的视图—指定义视图的查询语句中含有 GROUP BY 子句，这样的视图只能用于查询，不能修改数据。

8．5．2 删除视图

1、 格式为：DROP VIEW <视图名>

8．5．3 视图的作用

1、 简化数据查询语句；

2、 使用户能从多角度看到同一数据；

3、 提高了数据的安全性；

4、 提供了一定程度的逻辑独立性（外模式）

第 9 章 事务调度与并发控制

9．1 事务与事务调度

9．1．1 事务的概念

1、 事务是构成数据库应用中一个独立逻辑工作单元的操作的集合，也是访问并可能更新数据库中各种数据项的一个程序执行单元。数据库系统通过执行各种事务实现对数据库数据的操作，管理和执行事务是 DBMS 的基本功能。

9．1．2 事务的特性（ACID 特性）

1、原子性（Atomicity）

一个事务对数据库的所有操作是一个不可分割的工作单元，这些操作要么全部执行，要么一个也不执行。

2、 一致性（Consistency）

当一个事务独立执行时，其执行结果应维护数据库的一致性，即数据库不会因事务执行而受到破坏。数据库满足全部完整性约束，处于正确的状态；

3、 隔离性（Isolation）

当多个事务并发执行时，系统应保证一个事务的执行结果不受其他事务的干扰，事务并发执行结果与这些事务串行执行时的结果是一样的；

4、 持久性（Durability）

一个事务一旦成功完成全部操作，则它对数据库的所有更新就永久地反映在数据库中，即使以后数据库发生了故障；

9．1．3 事务调度

1、 一个事务中各操作的执行顺序和执行时机一方面取决于事务自身内部逻辑，另一方面也受 DBMS 中事务调度机制的控制。当多个事务并发执行时，DBMS 必须采用合适的并发调度机制合理安排各个事务执行顺序，以保证事务的 ACID 特性。

2、 调度分为串行调度和并发调度，串行调度的特点是一个事务的所有操作都执行完后才开始执行另一事务，不存在事务操作的交叉执行；并发调度是指，在每个事务保持自身操作顺序的前提下，不同事务操作的交叉执行，DBMS 交叉执行来自多个事务的各个操作，以提高数据库系统的性能。

9．1．4 可串行化调度（对并行调度而言的）

1、 事务的串行调度能够产生正确的结果，但执行效率低，如果并发调度 S 等价于某一定义在 TS 上

的串行调度，那么 S 称为可串行化调度；

2、给定两个定义在事务集 TS 上的的调度 S 和 S'，如果可以通过交换 S 中一系列非冲突操作的执行顺序将 S 转换为 S'，则称 S 与 S' 是冲突等价。（操作冲突：如果两个操作 I1， I2 至少有一个对同一个数据项 Q 进行 write）P165

3、如果定义在事务 TS 上的并发调度 S 冲突等价于事务集 TS 上的某个串行调度 S'，则称 S 是冲突可串行的。

4、在引入冲突可串行概念后，判断一个并发调度是否正确可以归结为判断该调度是否冲突可串行的。

9．2 基于锁的并发控制技术

9．2．1 锁的概念

1、对数据库系统中每个可能被多个事务并发访问的数据项设置锁，锁代表了对该数据项的访问权限。即事务 T 在访问数据项 Q 前须向 DBMS 申请获得设置在 Q 上的锁，如成功，则 T 获得对 Q 的访问权，T 对 Q 操作完成后，释放所占用的锁，允许其他事务获得该锁并访问 Q，在 T 释放设置在 Q 上的锁前，其他事务不能访问 Q。

2、锁的类型有两种：

（1） 互斥锁（X 锁）：（Exclusive，写锁）若 T 获得 Q 上的 X 锁，则 T 可以对 Q 读写，其他事务不能再对 Q 进行任何操作，直到 T 释放 Q 上的锁；

（2） 共享锁（S 锁）：（Shared，读锁）若 T 获得 Q 上的 S 锁，则 T 可以对 Q 进行读取操作，但不可以修改，同时，允许其他事务再申请获得 Q 上的 S 锁，与 T 并行读取 Q，但在 T 释放 Q 上的 S 锁前，其他事务不能对 Q 做任何修改；

9．2．2 加锁协议

1、三级加锁协议：（保证数据一致性

（1） 1 级加锁协议要求事务 T 在修改数据项 Q 之前必须先对 Q 加 X 锁，直到事务结束才释放，事务结束包括正常结束和非正常结束，但事务如果只对 Q 读而不写，则不需对 Q 加锁；

（2） 2 级加锁协议是在 1 级加锁协议基础上，要求 T 在读取 Q 前必须先对其加 S 锁，读完后立即释放 S 锁；

（3） 3 级加锁协议是在 1 级加锁协议基础上，要求在读取 Q 前必须先对其加 S 锁，但需等到事务结束后才释放 S 锁。

9．2．3 两阶段锁协议（保证事务调度可串行性）

1、两阶段锁（2PL）基本原理如下：（在于“纯粹”二字）

（1） 每个事务的执行过程划分为两个阶段，加锁阶段和解锁阶段；

（2） 在加锁阶段，事务可以申请获得任何数据项上的任何类型的锁，但是不允许释放任何锁；

（3） 在解锁阶段，事务可以释放任何数据上的任何类型的锁，但是不能再申请任何的锁；

（4） 每个事务开始执行后就进入加锁阶段，当第一次释放锁后，即进入解锁阶段。

9．2．4 锁粒度

1、 施加 X 锁和 S 锁的数据项大小称为锁粒度。

2、 锁粒度越大，系统中可以被锁的数据项（锁之后的其他）就越少，事务的并发执行度也越低，但同时系统的开销也小，相反，当锁粒度越小时，事务的并发度高，但系统开销也较大；

9．3 死锁（*事务无法获得对需要访问的数据项的控制权）处理

9．3．1 死锁预防

1、 一次加锁法

该方法要求每个事务在开始时必须将需要访问的数据项全部加锁，否则不能执行下去，也就是要求事务必须一次性地获得对需要访问的全部数据项的访问权；

该方法的缺点是：

（1） 多个数据项会被一个事务长期锁定独占，导致其他事务无法及时访问这些数据项，降低了系统的并发程度；

（2） 由于很难事先精确知道每个事务在执行过程中需要加锁的全部数据项，只能扩大加锁范围，将事务执行时可能访问的所有数据项全部加锁，进一步降低了系统的并发程度；

2、 顺序加锁法

该方法对数据库中事务访问的所有数据项规定一个加锁顺序，每个事务在执行过程中必须按此顺序对所需数据加锁；

该方法的缺点：

（1） 数据库中需要加锁的数据项非常多，并且不断变化，维护这些数据项的加锁顺序很困难，代价非常大；

（2） 事务访问的数据项有时无法事先完全确定，有时很难要求事务按照固定的顺序对这些数据项进行加锁；

9．3．2 死锁检测与恢复

1、 死锁检测

（1） 可以利用事务等待图进行死锁检测，数据库系统出现死锁当且仅当事务等待图中包含回路，而且回路中的所有事务就是处于死锁的事务；

（2） 数据库并发控制子系统动态地构造和维护事务等待图，并周期地检测等待图，如图中有

回路，则说明系统中出现了死锁；

2、死锁恢复

（1） 当发现死锁存在时，系统可以通过死锁恢复机制将系统从死锁中解救出来，通常是选取一个或几个死锁事务，撤消这些事务，释放其所有的锁，消除事务等待图中的回路，从而解决了系统死锁问题；

（2） 如果决定撤消哪个事务或哪些事务，有两个原则：

A、 选择处于最多条回路交点处的事务；

B、 选择具有最少撤消代价的事务。

9．4 活锁（权限问题）处理

1、如果一个事务在系统不存在死锁的情况下，长期得不到 DBMS 的获批，处于长时间等待中的情况叫活锁，为了避免活锁，DBMS 可采用先来先服务的原则解决。

第 10 章 数据库的实施、运行和维护

10．1 数据库的实施（Def:根据数据库的逻辑结构设计和物理结构设计，在计算机的系统上建立实际的系统结构、导入数据并进行程序的调试）p175 图

10．1．1 定义数据库结构

1、 为了实现数据库的逻辑结构设计和物理结构设计结果，必须建立实际的数据库，即在确定了数据库的逻辑结构和物理结构后，开发人员使用具体的 DBMS 提供的数据定义语言（DDL）来严格描述数据库结构。

10．1．2 数据装载（是数据库实施阶段最重要的工作）

1、 完成了数据库定义后，还须装入各种实际数据；

2、 由于数据的来源不同，其组织方式、结构、格式会不同，可能出现源数据与新数据库结构不相容；

3、 我们可以先将源数据提取出来，存入计算机，然后分类转换，成为符合新数据库结构的数据，再存入数据库，具体步骤如下：

（1）筛选数据；（2）转换数据格式；（3）输入数据；（4）校验数据；（不相容情况）

4、 为完成初始数据的录入，通常需要设计一些数据录入子系统，由计算机辅助完成入库工作，对某些纸质数据或数据量少的数据，可由人工一条条进行录入；而对于数据量大的数据，可考虑采用批量数据装载程序来实现。

10．1．3 编写与调试应用程序

1、 只有当数据库的结构建立好后，才能开始应用程序的编写和调试；

2、 可使用模拟数据进行程序的调试。

10．1．4 数据库的试运行

1、 应用程序调试完成并已有一小部分数据入库，就可以开始数据库的试运行，也称联合调试；

2、 试运行十分重要，因为：

（1） 检测应用程序在接近真实的环境中运行是否符合设计要求；

（2） 检测系统设计的 5。

3、 试运行的工作主要有两个：

（1） 功能测试：运行数据库应用程序，执行各种操作，测试程序是否满足设计要求，找出不足，改进现有程序直到符合设计要求；

（2） 性能测试：测量系统的性能指标，分析是否符合设计目标。

10．2 数据库的运行和维护

1、 数据库设计并试运行后，如试运行结果符合设计目标，数据库就可以真正投入运行了，同时也标志着开发任务的基本结束和维护工作的开始；

2、 维护工作包括：

（1） 数据库的转储与恢复；

（2） 数据库安全性和完整性控制；

（3） 数据库性能的检测与改善；

（4） 数据库的重组（按照系统设计要求对数据库的存储空间进行安全调整）和重构（如逻辑结构等作必要调整）。

10．3 监控分析

1、 数据库的监控分析：指管理员借助相应工具在数据库运行过程中监测数据库系统的运行情况，掌握数据库当前或以往的负荷、配置、应用和其他相应信息，并对监测数据进行分析，分析数据库的性能参数和环境信息，评估系统的整体运行状态，为系统的安全运行和性能调优提供依据，并提出相应的改善措施，帮助管理人员尽早清除数据库的性能隐患；

2、 监控分析的目的：保证数据库系统安全、稳定地运行，以便在发现不正常的情况时，及时对系统进行维护；

3、 根据实现的方法不同，监控的机制分为：

（1） 自动监控机制；

（2） 手动监控机制。

4、根据监控的对象不同，监控分为：

（1） 对数据库架构体系的的监控；

（2） 对数据库性能的监控。

10. 4 空间管理

1、 在数据库运行过程中，对数据库空间使用情况，特别是空间的增长情况进行监控，并采取相应的措施对空间进行管理非常重要；

2、 空间管理主要包括（物理结构上的）：创建数据库空间，更改空间大小，删除空间，修改空间状态，新建、移动、关联数据文件等；

10. 5 参数调整

1、外部调整：（外部环境与数据库的性能有很大影响）

（1） CPU：当数据库操作对 CPU 的要求超过数据库服务器的 CPU 性能时，数据库性能就受到 CPU 的限制，使数据库操作变慢；如业务高峰时，CPU 的使用量仍然很低，说明服务器 CPU 资源充足；

（2） 网络：大量的 SQL 数据在网络上传输会导致网速变慢，调整网络设备，也可以一定程度上提高数据库的性能；

2、 调整内存分配

3、 调整磁盘 I/O（I/O 时间）

4、 调整竞争：（资源有限）

（1） 修改参数以控制连接到数据库的最大进程数；

（2） 减少调试进程的竞争；

（3） 减少多线程服务进程的竞争；

（4） 减少重做日志缓冲区竞争；

（5） 减少回滚段竞争。

10. 6 查询优化

1、 合理使用索引：索引是数据库中重要的数据结构，根本目的就是为了提高查询效率，使用原则如下 ：

（1） 经常在索引中作为条件被使用的列，应为其建立索引；

（2） 频繁进行排序或分组（即进行 group by 或 order by 操作）的列，应为其建立索引；

（3） 一个列的值域很大时，应为其建立索引；

（4） 如果待排列的列有多个，应在这些列上建立复合索引；

（5） 可以使用系统工具来检查索引的完整性，必要时进行修复。

2、 避免或简化排序：因为磁盘排序的开销很大，当能够利用索引自动以适当的次序产生输出时，优

化器就可以避免不必要的排序步骤，以下是一些影响因素：

（1） 由于现有的索引不足，导致排序时索引中不包括一个或几个等待排序的列；

（2） group by 或 order by 子句中列的次序与索引的次序不一样；

（3） 排序的列来自不同的表。

为了避免不必要的排序，就要正确地增建索引，合理地合并数据库表。如排序不可避免，那么应试图简化它。

3、 消除对大型表行数据的顺序存取：在嵌套查询中，对表的顺序存取对查询效率可能产生致命的影响，解决方法就是对连接的列进行索引。还可以使用并集来避免顺序存取。

4、 避免相关子查询：查询嵌套层次越多，效率越低，应尽量避免子查询，如不可避免，那么要在子查询中过滤尽量多的行；

5、 避免困难的正规表达式：避免含 MATCHES 和 LINK 关键字的正规表达式；

6、 使用临时表加速查询：把表的一个子集进行排序并创建临时表，有时能加速查询；

7、 用排序来取代非顺序磁盘存取；

8、 不充分的连接条件；

9、 存储过程；

10、 不要随意使用游标；

11、 事务处理。

第 11 章 故障管理

11．1 事务

1、事务是数据库的逻辑控制单位，是操作数据的一个程序执行单元。

2、为了保证数据的完整性，要求数据库系统维护事务具有如下性质：

（1） 原子性：事务是一个不可分割的工作单位，事务中的操作要么都做，要么都不做；

（2） 一致性：事务执行的结果必须使数据库从一个一致的状态变到另一个一致的状态；

（3） 隔离性：一个事务内部的操作及使用的数据对于其他并发事务是隔离的；

（4） 持续性：一个事务提交后，它对数据库中数据的改变是永久性的，即使系统可能出现故障，也不会对其它执行的结果有任何影响。

11．2 故障的种类（4 类）及解决方法（冗余）

11.2.1 事务内部故障

1、预期的事务内部故障：

通过事务程序本身发现的事物内部故障，可以通过将事务回滚，撤销其对数据库的修改，从而使数据库回到一致性的状态；

2、非预期的事务内部故障：

（1）由于事务内部故障大部分属于此类，所以事务故障仅限指此类故障；

（2）事务故障表明事务没有提交或撤销就结束了，因此数据库可能处于不正确的状态，因此，恢复事务必须强行回滚事务，在保证该事务对其他事务没有影响的条件下，利用日志文件撤销其对数据库的修改，使数据库恢复到该事务运行之前的效果；

（3）事务故障恢复是由系统自动完成的，对用户是透明的。

11.2.2 系统故障（软故障）

1、指数据库在运行过程中，由于硬件故障、数据库软件及操作系统的漏洞、突然停电等情况，导致系统停止运转，所有正在运行的事务以非正常方式终止，需要系统重新启动的一类故障；

2、系统故障导致内存中的内容丢失，而在硬盘上的内容仍然完好；从而导致数据库的数据可以处于不正确的状态；

3、要消除这些事务对数据库的影响，保证数据库中数据的一致性，办法就是在计算机系统重新启动后，对于未完成的事务可能已经写入数据库的内容，回滚所有未完成的事务写的结果，以保证数据库中数据的一致性；对于已完成的事务可能部分或全部留在缓存区的结果，需要重做所有已提交的事务，以将数据库真正恢复到一致状态。

4、一句话，当数据库发生系统故障时，容错对策是在重新启动系统后，撤销（UNDO）所有未提交的事务，重做（REDO）所有已提交的事务。

11.2.3 介质故障（硬故障）

1、指数据库在运行过程中，由于磁盘损坏、天灾人祸等情况，使用数据库中的数据部分或全部丢失的一类故障；

2、介质故障的容错对策采用两种方式：

（1）软件容错：

是使用数据库备份及事务日志文件，通过恢复技术，恢复数据库到备份结束时的状态；

（2）硬件容错：

目前常用的方法是采用双物理存储设备，最完全的方式是设计两套相同的数据库系统同时工作，数据的变化也同步，空间有一定距离，这样当发生损坏性的自然现象时，由于两套数据库系统具有空间距离，因此同时发生破坏的概率几乎为零，达到数据库的完全安全。

11.2.4 计算机病毒故障

1、计算机病毒是一种恶意的计算机程序，在对计算机系统造成破坏的同时也可对数据库系统造成破坏（主要破坏数据库文件）；

2、可以通过设立防火墙预防，杀毒软件查杀已感染的文件和数据库备份来解决；

11．3 数据库恢复技术概述

1、 恢复机制涉及两个关键问题：

（1） 如何建立冗余数据；

（2） 如何利用这些冗余数据实施数据库恢复。

2、 最常用的建立冗余数据技术是数据备份和登录日志文件，他们通常是结合起来使用的。

11．4 数据转储

1、 数据转储—指数据库管理员（DBA）定期拷贝数据库，并将拷贝得到的数据库放到其他介质中的过程。

2、 DBA 可在数据库系统发生故障后，利用这些副本恢复数据库，但此时恢复的数据库只能回到转储时的状态，要想恢复到故障前的状态，需要参考日志文件，重新运行转储后到故障前的所有事务才可以；

3、 静态转储和动态转储

（1） 静态转储：在静态转储过程中系统不能运行其他事务，不允许在转储期间对数据库的任何存取、修改活动。

（2） 动态转储：允许转储操作和用户事务并发执行；

（3） 静态转储虽然保证了数据的有效性，但却是以降低数据库的可用性为代价；而动态转储虽然提高了数据库的可用性，但数据库的有效性却得不到保证。

（4） 为了能保证数据的有效性，而又不降低可用性，就需要引入日志文件，用它记录转储期间各事务对数据库的修改活动，然后使用动态转储的备份副本加上日志文件就可将数据库恢复到某一时刻的正确状态。

3、 几种数据转储机制

（1） 完全转储：对所有数据库进行备份，需占用较多时间和空间，可作为系统失败时恢复数据库的基础；

（2） 增量转储：只复制上次备份后变化的文件；

（3） 差量转储：对最近一次数据库完全备份以来发生的数据变化进行备份，优点是速度快，占用较少的时间和空间。

4、 多种转储方法结合使用

（1） 仅采用完全转储；

（2） 完全转储加增量转储；

（3） 完全转储加差量转储

11．5 登记日志文件

11．5．1 日志文件的格式和内容

日志文件是记录每个事务对数据库更新操作的文件，数据库系统在运行过程中，DBMS 负责将所有事务的更新操作登记到日志文件中，也就是说日志文件是系统自动维护的。

1、以记录为单位的日志文件：其内容包括每个事务的开始标记、结束标记和所有更新操作；每个日志记录的内容包括：事务标识、操作类型、操作对象、更新前数据的旧值，和更新后数据的新值；(p188，表 11-2)

2、数据块为单位的日志文件：将更新前的整个数据块和更新后的整个数据块全部放在了日志文件中；

11．5．2 日志文件的作用

1、事务故障恢复和系统故障恢复必须使用日志文件

（1）故障恢复的两个基本操作：UNDO 和 REDO

（A） UNDO 的作用是撤销事务，具体步骤：

（a） 反向扫描日志文件，找到需要撤销的事务的更新操作；

（b） 对事务的更新操作执行逆操作；

（c） 继续反向查找该事务的其他更新操作，并执行相应的逆操作；

（d） 重复执行步骤（C），直至遇到该事务开始记录。

（B） REDO 的作用是重做事务，具体步骤：

（a） 正向扫描日志文件，找到需要重做的事务的更新操作；

（b） 对事务重新执行日志文件登记的操作，即将日志文件中“更新后的值”写入数据库；

（c） 继续正向查找该事务的其他更新操作，并重新执行，将日志文件中“更新后的值”写入数据库；

（d） 重复执行步骤（C），直至遇到该事务的提交记录。

（3） 事务故障恢复：只需把相应的事务作撤销 UNDO 即可；

（4） 系统故障恢复：① 事务已经开始，但没有提交，也就是日志文件中有 begin transaction，但没有 commit，此种直接 undo；2、完成所以操作并提交，但没有真正写入数据库，也就是日志是完整的，此种不仅 undo，还要 redo）（注意数据是在内存缓冲区修改的）

（A） 正向扫描日志文件，找到系统故障前发生的所有事务，如果该事务没有完成，将其事务标记加入撤销队列，如果该事务已经完成，则将其事务标记加入重做队列；

（B） 对撤销队列中的所有事务作撤销操作 UNDO；

（C） 对重做队列中的所有事务作重做操作 REDO。

2、在动态转储方式中必须建立日志文件

3、 在静态转储方式中，也可以建立日志文件

11．5．3 登记日志文件的原则

1、 登记的次序严格按并行事务执行的时间次序；

2、 必须先写日志文件，后写数据库

11．6 具有检查点的恢复技术

11．6．1 检查点的作用

检查点最大限度地减少数据库完全恢复时所必须执行的日志部分；

11．6．2 检查点的引入

1、在日志文件中增加一类新的记录—检查点记录，增加一个“重新开始文件”，并让恢复子系统在登录日志文件期间动态地维护日志

2、检查点记录的内容：

（1） 建立检查点时刻所有正在执行的事务清单；

（2） 这些事务最近一个日志记录的地址。

3、“重新开始文件”用来记录各个检查点在日志文件中的地址；

4、 动态维护日志文件的方法是周期性地执行如下操作：建立检查点、保存数据库状态，具体步骤：

（1） 将当前日志缓冲中的所有日志记录写入磁盘的日志文件上；

（2） 在日志文件中写入一个检查点记录；

（3） 将当前数据缓冲的所有数据记录写入磁盘的数据库中；

（4） 把检查点记录在日志文件中的地址写入一个“重新开始文件”；

5、 恢复子系统可以定期或不定期地建立检查点来保存数据库状态；

6、 使用检查点方法可以改善恢复效率，事务在检查点前已经提交，则不必执行 REDO 操作。

11．6．3 恢复的步骤

1、 从“重新开始文件”中找到最后一个检查点记录在日志文件中的地址，由该地址在日志文件中找到最后一个检查点记录；

2、 由该检查点记录得到检查点建立时刻所正在执行的事务清单(ACTIVE-LIST)，需要执行 UNDO 操作的事务建立 UNDO-LIST 队列，需要执行 REDO 操作的事务建立 REDO-LIST 队列；（把全部 ACTIVE-LIST 中的全放在放在 UNDO-LIST 中，REDO-LIST 默认为空）

3、 从检查点开始正向扫描日志文件，如有新开始的事务，则放入 UNDO 队列，如有提交事务，则放入 REDO 队列；

4、 对 UNDO 队列中的每个事务执行 UNDO 操作，对 REDO 队列中的事务进行 REDO 操作。

11．7 数据库镜像

11．7．1 数据库镜像的引入

为了避免介质故障对数据库可用性的影响，许多数据库系统都提供了数据库镜像的功能

11．7．2 数据库镜像简介

1、 数据库镜像是一种用于提高数据库可用性的解决方案，它根据 DBA 要求，自动把整个数据库或其中的关键数据复制到另一个磁盘中。

2、 数据库镜像的优点：（不同磁盘上有不同服务器，它们相互侦查，出问题时，好的主动接管出问题的服务器）

（1） 数据库镜像提供完整或接近完整的数据冗余，增强数据保护功能；

（2） 发生灾难时，数据库镜像可快速使数据库的备用副本提供服务，使数据不会丢失，提高数据库的可用性；

（3） 提高镜像数据库在升级期间的可用性。

11．7．3 数据库镜像分类

1、 双机互备援模式：就是两台主机均为工作机，两台工作机均为信息系统提供支持，并互相监视对方的运行情况，当一台主机出现异常时，另一主机则主动接管异常机的工作，保证信息系统能够不间断运行。工作机的切换时机：

（1） 系统软件或应用软件造成服务器宕机；

（2） 服务器没有宕机，系统软件或应用软件工作不正常；

（3） SCSI 卡损坏，造成服务器与磁盘陈列无法存取数据；

（4） 服务器内硬件损坏，造成服务器宕机；

（5） 服务器不正常关机

2、双机热备份模式：一台主机为工作机，另一台主机为备份机，在系统正常运行的情况下，工作机为信息系统提供支持，备份机监视工作机的运行情况。当工作机异常时，备份机主动接管工作机工作，从而保证信息系统不间断提供服务。

11．7．4 工作方式

1、 在“数据库镜像会话”中，主体服务器和镜像服务器作为“伙伴”进行通信和协作。这两个伙伴在会话中扮演互补的角色：“主体角色”和“镜像角色”，拥有主体角色的称为“主体服务器”，拥有镜像角色的称为“镜像服务器”。

2、 数据库镜像涉及尽快将对主体数据库执行的每项插入、更新和删除操作重做到镜像数据库中。重做通过将每个活动事务日志记录发送到镜像服务器来完成，这会尽快将日志记录按顺序应用到镜像数据库中，这样，每当数据库更新时，DBMS 将自动保证镜像数据与主体数据的一致性；

3、 在出现介质故障时，可由镜像数据库继续提供使用，同时 DBMS 将自动利用镜像磁盘数据进行主

数据库的恢复，不需关闭系统。

4、 一旦出现介质故障，通常使用一个“角色切换”的过程来互换主体服务器和镜像服务器。

5、 由于数据库镜像是通过复制数据实现的，在实际应用中，用户只选择对关键数据和日志文件进行镜像，而不是对整个数据库进行镜像。

11．8 RAID 的恢复技术

1、 RAID—廉价冗余磁盘陈列，它是由多块磁盘构成的一个整体，但并不是简单的磁盘容量叠加，而是相对于其他存储设备在容量、管理、性能、可靠性和可用性上都有了进一步的提高。尤其独特的是，当从这些磁盘中抽出一块，利用其他磁盘上的信息，可以恢复出这块磁盘的信息；

2、 RAID 系统可以连接在主机系统上，作为其存储数据的介质，与一般存储设备不同的是，它具有设备虚拟化的能力。即 RAID 系统内部可以包含很多个磁盘驱动器，但在主机系统是看不到的，主机系统主能通过一个子系统 RAID 控制器与这些磁盘构成虚拟设备进行交互。

3、 RAID 的冗余技术：

（1） 镜像冗余：把所有的数据拷贝到其他设备上，但额外开销很大，需要更多磁盘、控制器和电缆；

（2） 校验冗余：通过对成员磁盘的数据执行异或（XOR）操作，得到其校验值，并存放在另外的校验磁盘上。该技术实现起来稍显复杂，但它占用的磁盘比镜像少。

第 12 章 SQL Server 2000 数据库管理系统

12．1 SQL Server 2000 概述

服务是数据库完成所需功能的基础，SQL Server 2000 提供了四种服务：

1、SQL Server

SQL Server 服务是 SQL Server2000 最核心的服务，它直接管理和维护数据库，负责处理所有来自客户端的 Transact-SQL（SQL Server 使用的数据库语言）语句并管理服务器上构成数据库的所有文件，同时还负责处理存储过程，并将执行结果返回给客户端；

2、SQL Server Agent

对需要定期进行的工作，SQL Server2000 提供了代理的功能，根据系统管理员预先设定好的计划自动执行相应的功能。同时它还能对管理员设定好的错误等特定事件自动报警，且能通过电子邮件方式把系统存在的各种问题发送给指定的用户，帮助管理员对系统进行监视和管理；

3、Distributed Transaction Coordinator(DTC)

分布式事务协调器是一个事务管理器，在 DTC 支持下，客户可以在一个事务中访问不同服务器上的数据库。DTC 能够保证一个事务中的所有操作在所有服务器上全部成功，或者，当在某个服务器

上不成功时，确保所有服务器上的操作均被撤销，使全部服务器均回到事务开始前的状态；

4、Microsoft Search

提供了全文检索服务，能够对字符数据进行搜索。

12．2 SQL Server 2000 的安装

12．2．1 安装前的准备

1、SQL Server2000 的版本

SQL Server2000 共有企业版、标准版、开发版和个人版四个版本。

（1） 企业版：支持 SQL Server2000 中全部功能，适合于作为大型数据库服务器使用；

（2） 标准版：支持许多 SQL Server2000 功能，但在服务器扩展性、大型数据库支持、数据仓库、WEB 站点方面的能力较弱，适合于作为小工作组或部门的数据库使用；

（3） 开发版：支持企业版的全部功能，但只能作为开发和测试系统使用。不能作为生产服务器使用；

（4） 个人版：适合在移动环境中作业的用户，并且所动作的应用程序需要本地数据存储。

2、选择合适的操作系统

版本	操作系统要求
企业版	Windows NT Server 4.0 或以上、Windows 2000Server 或以上
标准版	Windows NT Server 4.0 或以上、Windows 2000Server 或以上
开发版	Windows 98、Windows 2000 Professional、Windows XP Professional、Windows 2000 Server
个人版	Windows 2000 Professional、Windows XP Professional 和所有其他的 Windows 2000

12．2．2 安装及安装选项

1、 放入安装光盘；

2、 选择“SQL Server2000 组件”—>“安装数据库服务器” —>“本地计算机” —>“创建新的 SQL Server 实例” —>输入合适的用户名和公司名—>“软件许可证协议” —>“服务器和客户端工具” —>“实例名” —>“安装类型” —>“选择组件” —>

12．2．3 测试安装

12．3 SQL Server 2000 常用工具简介

12．3．1 企业管理器

企业管理器是 SQL Server2000 的主要图形化管理工具，它提供了一个遵从 Microsoft 管理控制台风格的用户界面。在企业管理器中几乎可以完成所有管理工作；

12．3．2 查询分析器

查询分析器是一个图形化的查询工具，用户可以编写和执行 SQL 语句，并查看执行结果，它具

有以下特点：

1、 用不同的颜色标识 Transact-SQL 语法中不同含义的单词，提高语句的易读性；

2、 对象浏览器工具使用户可以轻松地查找数据库中的对象和对象结构；

3、 选择要操作的数据库；

4、 选择要执行的语句，可以让查询分析器只执行选中的 SQL 语句，若不选中任何语句，则是执行文本编辑器中的全部语句；

5、 可将文本编辑器中编写的语句保存起来，以备以后使用，也可打开保存好的包含 SQL 语句的文件进行编辑或执行。

12．4 创建和管理数据库

12．4．1 系统数据库

1、系统数据库：安装完成后，SQL Server 建立的系统数据有：

（1） master：最重要的系统数据库，记录了所有的系统级信息，包括登录帐号、系统配置、数据库属性等信息；

（2） msdb：提供对自动执行任务的支持；

（3） model：样板数据库，其中包含所有用户数据库的公共信息；

（4） tempdb：临时数据库，用于存储用户创建的临时表、用户声明的变量以及用户定义的游标数据等。当用户离开 SQL Server 时，系统自动删除 tempdb 数据库中所创建的对象并释放所占用的空间。

2、用户示例数据库：安装完 SQL Server 后，系统建立了两个用户示例数据库供用户学习使用：

（1）Pubs：主要存放关于出版商、作者及图书等信息；

（2）Northwind：主要存放关于产品、订单、客户信息。

12．4．2 SQL Server 数据库的构成

1、 SQL Server 的数据库由两种文件组成：数据文件和日志文件。数据文件用于存放数据，日志文件用于存放对数据的操作记录。

2、 在考虑数据库的空间分配时，需了解如下规则：

（1） 所有数据库都包含一个主数据文件与一个或多个日志文件，还可以包含零个或多个辅助数据文件；

（2） 在创建用户数据库时，包含系统表的 model 数据库自动被复制到新建数据库中；

（3） 在 SQL Server 2000 中，数据的存储单位是页，一个数据页是一块 8KB 的连续磁盘空间；

（4） 在 SQL Server 中，不允许表中的一行数据存储在不同的数据页上，且一行的数据大小不能超过一个数据页的大小；

3、数据文件和日志文件的作用

（1） 数据文件：用于存放数据库数据，数据文件又分为：主数据文件和辅助数据文件

（A） 主数据文件：主数据文件的推荐扩展名是.mdf，它包含数据库的系统信息，并可存放用户数据库的数据，每个数据库只包含一个主数据文件；

（B） 辅助数据文件：辅助数据文件的推荐扩展名是.ndf，当数据库数据量很大时，可能需要多个辅助数据文件，这些文件可以存放在不同的磁盘上，以便利用多个磁盘上的存储空间，并提高数据存取的并发性。

（C） 两种数据文件对用户是透明的，系统会选用最高效的方法来使用这些数据文件。

（2） 日志文件：主要记录对数据库数据的修改操作。日志文件的推荐扩展名为.ldf，它包含用于恢复数据库的日志记录，每个数据库必须至少有一个日志文件，也可以有多个。

4、创建数据库时的其他属性

（1） 文件名及其位置：每个数据库的数据文件和日志文件都具有一个逻辑文件名和物理的存放位置；

（2） 初始大小：可以指定每个数据文件和日志文件的初始大小，两者最小都是 512KB；

（3） 增长方式：当数据库的空间用完后，系统是否可自动扩大数据库的空间；

（4） 最大大小：指文件增长的最大空间限制，默认是无限制。

12．4．3 创建数据库

1、 使用企业管理器创建数据库

2、 使用 Transact-SQL 语句创建数据库

12．4．4 删除数据库

1、 使用企业管理器删除数据库

2、 使用 Transact-SQL 语句删除数据库

12．5 Transact-SQL 简介

12．5．1 Transact-SQL 语言基础知识

1、 注释

（1） 单行注释：以“--”为开始的一行；

（2） 块注释：以/* 注释 */的块。

2、 变量

（1） 变量的种类：全局变量和局部变量，全局变量以@@开始，局部变量以@开始；全局变量是由系统提供且预先声明的变量，用户一般只能查看不能修改全局变量的值。局部变量是用户用以保存特定类型的单个数据值的对象；

（2） 变量的声明与赋值：

（A） 变量名最多可以包含 128 个字符，使用 DECLARE 语句声明一个局部变量后，这个变量的值将被初始化为 NULL；

（B） 变量赋值语句格式：SET @局部变量名=值 或 表达式；

（C） 变量赋值语句格式：SELECT @局部变量名=值 或 表达式；

12．5．2 流程控制语句

1、BDGIN…END 语句：用于定义一个语句块，格式如下：

```
BEGIN
   语句 1
   语句 2
   …
 END
```

BDGIN…END 语句块一般与流程控制语句 IF…ELSE 或 WHILE 一起使用的。

2、IF…ELSE 语句：

```
IF 布尔表达式
     语句块 1
[ ELSE
     语句块 2]
```

3、WHILE 语句：

```
WHILE 布尔表达式
  循环体语句块
```

12．6 数据传输

12．6．1 DTS 功能概述

1、 DTS 提供了许多传输数据的工具，主要有：

（1） 导入/导出向导：它可以实现不同数据源之间的数据传输以及数据传输过程中的数据转换；

（2） DTS 设计器：此工具用于建立带有工作流和事件驱动逻辑的较为复杂的数据转换操作；

2、 DTS 的源数据和目的数据可以是异构的数据库数据。

12．6．2 利用 DTS 向导实现数据传输

第 13 章 数据库对象

13．1 存储过程(PROCEDURE)

13．1．1 存储过程基本概念

1、 在关系数据库中，SQL 语言是应用程序和数据库管理之间的主要编程接口；

2、 使用 SQL 语言编写代码时，可用两种方法存储和执行代码：

（1） 在客户端存储代码，并创建向数据库管理系统发送 SQL 命令，并处理返回结果的应用程序；（现编现用）

（2） 将这些发送的 SQL 语句存储在数据库管理系统中，这些存储在数据库管理系统中的 SQL 语句就是存储过程，（编好套用）然后再创建执行存储过程并处理返回结果的应用程序。

3、 使用存储过程的好处：

（1） 模块化程序设计：只需创建一次存储过程并将其存储在数据库中，以后就可以在应用程序中多次调用存储过程；

（2） 提高性能：系统在创建存储过程时对其进行分析和优化，并在第一次执行时进行语法检查和编译，编译好的代码放入内存中，以后再执行此存储过程时，只需直接执行内存中的代码，从而提高代码的执行效率；

（3） 减少网络流量：一个需要数百行 SQL 代码完成的操作现在只需一条执行存储过程的代码即可实现，因此，不再需要在网络中发送这些多语句；

（4） 可作为安全机制使用：执行存储过程和执行 SQL 语句权限有区别

13．1．2 创建和执行存储过程

1、 创建存储过程的 SQL 语句为：CREATE PROCEDURE，语法格式为：

```
CREATE PROCEDURE 存储过程名
 [{@ 参数名 数据类型}[=default][OUTPUT]
  ][,…n]
  AS
     SQL 语句[…n]
```

其中：

（1） default:表示参数的默认值。如果定义了默认值，则在执行存储过程时，可以不必指定该参数的值，默认值必须是常量或 NULL；

（2） OUTPUT（有返回值）：表明参数是输出参数，该选项的值可以返回给存储过程的调用者。

2、执行存储过程的 SQL 语句是 EXECUTE，语法格式：

```
EXECUTE（相当于 call） 存储过程名 [实参[，OUTPUT][，…n]]
```

3、 执行有多个输入参数的存储过程时，参数的传递方式有两种：

（1） 按参数位置传递值:指执行存储过程的 EXEC 语句中的实参的排列顺序必须与定义存储过程时定义的参数的顺序一致;

（2） 按参数名传递值:指执行存储过程的 EXEC 语句中要指明定义存储过程时指定的参数的名字以及此参数的值，而不关心参数的定义顺序。

4、 注意:

（1） 在执行有输出参数的存储过程时，执行语句中的变量名的后边要加上 OUTPUT 修饰符;

（2） 在调用有输出参数的存储过程时，与输出参数对应的是一个变量，此变量用于保存输出参数返回的结果;

13．2 用户自定义函数(FUNCTION)

13．2．1 基本概念

1、 用户定义函数可以扩展数据操作的功能，它在概念上类似于一般的程序设计语言中定义的函数。

13．2．2 创建和调用标量函数

标量函数—返回单个数据值的函数;

1、定义标量函数

```
CREATE FUNCTION [拥有者名.] 函数名
  ([{@ 参数名[AS]标量数据类型[=default]}[,…n]])
  RETURNS 返回值类型
  [AS]
BEGIN
    函数体
    RETURN 标量表达式
END
```

（1） 同存储过程一样，函数的参数也可以有默认值。

（2） 标量表达式：指定标量函数返回的标量值。

2、 调用标量函数:

当调用标量函数时，必须提供至少由两部分组成的名称：函数拥有者名和函数名（写出就行）。可在任何允许出现表达式的 SQL 语句中调用标量函数，只要类型一致;

13．2．3 创建和调用内嵌表值函数

1、 内嵌表值函数，没有函数体

2、多语句表值函数（相当于标量函数和内嵌表值函数的结合体）

多语句表值函数的返回值是一个表，因此对多语句表值函数的使用也是放在 SELECT 语句的 FROM

子句中。

```
CREATE FUNCTION [拥有者名.] 函数名
([{@ 参数名[AS]标量数据类型[=default]}[,…n]])
RETURNS 返回变量 TABLE <表定义>
[AS]
BEGIN
    函数体
    RETURN
END
<表定义>::=({列定义|表约束}[,…n])
```

13．3 触发器(TRIGGER)

13．3．1 触发器基本概念

1、 触发器是一种特殊的存储过程，其特殊性在于它不需要由用户来调用，而是当用户对表中的数据进行 UPDATE、INSERT 或 DELETE 操作时自动触发执行；用于保证业务规则和数据完整性。

2、 触发器常用于下列场合：

（1） 完成比 CHECK 的约束更复杂的数据约束；

（2） 为保证数据库性能而维护的非规范化数据；

（3） 实现复杂的业务规则，可使业务的处理任务自动进行。

13．3．2 创建触发器

1、 创建触发器的语句：CREATE TRIGGER，语法为：

```
CREATE TRIGGER 触发器名称
 ON {表名| 视图名}
 [WITH ENCRYPTION]
 {FOR |AFTER|INSTEAD OF}{[INSERT][, ][DELETE][, ][UPDATE]}
AS
[{IF UPDATE (column) …}]
SQL 语句
```

（1） 触发器名称在数据库中必须是惟一的；

（2） ON 子句用于指定在其上执行触发器的表；

（3） AFTER：指定触发器只有在引发的 SQL 语句中指定的操作都已成功执行，并且所有的约束

检查也成功完成后，才执行此触发器，这种触发器称为后触发型触发器；

（4） FOR：作用同 AFTER；

（5） INSTEAD OF：指定执行触发器而不是执行引发触发器执行的 SQL 语句，从而替代触发语句的操作，这种触发器称为前触发型触发器；

（6） INSTERT、DELETE 或 UPDATE 是引发触发器执行的操作，若同时指定多个操作，则各操作之间用逗号分隔；

2、 创建触发器时，需要注意：

（1） 在一个表上可以建立多个名称不同、类型各异的触发器，每个触发器可由三个操作引发；

（2） 大部分 Transact-SQL 语句都可用在触发器中，但也有一些限制；

（3） 在触发器定义中，可以使用 IF UPDATE 子句测试在 INSERT 和 UPDATE 语句中是否对指定字段有影响；

（4） 通常不要在触发器中返回任何结果。

3、 创建后触发型触发器

使用 FOR 或 AFTER 选项定义的触发器为后触发的触发器，即只有在引发触发器执行语句中指定的操作都已完成执行，并且所有的约束检查也成功完成后，才执行的触发器；

4、 创建前触发型触发器

使用 INSTEAD OF 选项定义的触发器为前触发型触发器，在该触发器中，指定执行触发器而不是执行引发触发器执行的 SQL 语句，从而替代引发语句的操作。

13．4 查看、修改及删除对象

13．4．1 查看对象

对于创建好的存储过程、函数可通过企业管理器和查询分析器查看这些对象的代码；

13．4．2 修改对象

1、 修改存储过程：ALTER PROCEDURE

```
ALTER PROC 存储过程名
[{@参数名 数据类型}[=default][OUTPUT]][,…n]
AS
  SQL 语句[…n]
```

修改与定义的语句基本一致，只将 CREARE PROC 改成 ALTER PROC；

2、修改用户自定义函数：

修改与定义的语句基本一致，只将 CREARE FUNCTION 改成 ALTER FUNCTION；

3、修改触发器：

修改与定义的语句基本一致，只将 CREARE TRIGGER 改成 ALTER TRIGGER；

13．4．3 删除对象

1、 删除存储过程：

DROP PROCEDURE {存储过程名}[，…n]；

2、删除用户自定义函数

DROP FUNCTION {[拥有者名.]函数名}[,…n]

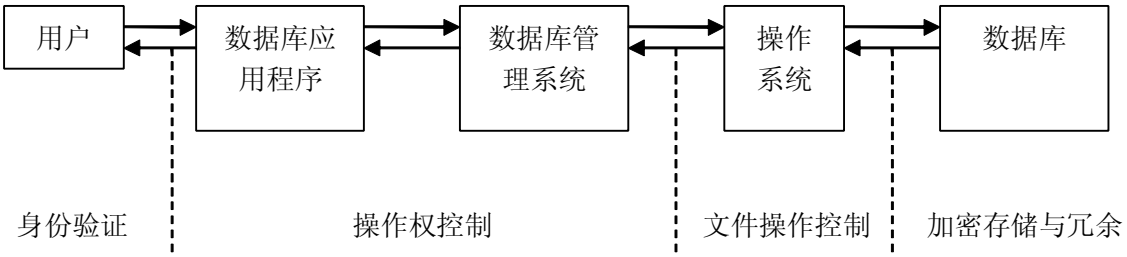
3、删除触发器

DROP TRIGGER {触发器名}[,…n]

第 14 章 安全管理

14．1 安全控制

14．1．1 安全控制模型



14．1．2 数据库权限的种类及用户的分类

1、 权限的种类

（1） 对 DBMS 进行维护的权限；

（2） 对数据库中的对象和数据进行操作的权限。

（A） 对数据库对象的权限，包括创建、删除和修改数据库对象；

（B） 对数据库数据的操作权限，包括对表、视图数据的增、删、改、查权限。

2、 数据库用户的分类

（1） 数据库系统管理员（sa）：在数据库中具有全部的权限；

（2） 数据库对象拥有者：对其所拥有的对象具有一切权限；

（3） 普通用户：只具有对数据库数据的增、删、改、查权限。

14．2 SQL Server 的安全控制

1、 用户访问 SQL Server 数据库中的数据中，必须经过三个认证过程

（1） 身份认证：验证用户是否有连接到数据库服务器的“连接权”；

（2） 验证用户是否数据库的合法用户；

（3） 验证数据库用户是否具有要进行的操作的操作权限。

2、SQL Server 的用户有两种类型：

（1） Windows 授权用户：来自 Windows 的用户或组；

（2） SQL 授权用户：来自于非 Windows 的用户，也将这种用户称为 SQL 用户。

3、SQL Server 为不同用户类型提供不同的安全认证模式：

（1） Windows 身份验证模式：允许 Windows NT 或 Windows 2000 用户连接到 SQL Server，在这种模式下，SQL Server 将通过 Windows 来获得用户信息，并对账号和密码进行重新验证，当使用 Windows 身份验证模式时，用户必须先登录到 Windows，然后再登录到 SQL Server；

（2） 混合验证模式：表示 SQL Server 接受 Windows 授权用户和 SQL 授权用户。

14．3 管理 SQL Server 登录账户

14．3．1 系统内置的登录账户

1、BUILTIN\Administrators：是一个 Windows 组账户，表示所有的 Windows Administrators（系统管理员）组中的用户都可以登录到 SQL Server，此组中的成员同是具有 SQL Server 的系统管理员权限；

2、Sa：SQL Server 验证模式的系统管理员账户；

3、域名\Administrators: Windows 的系统管理员同时也是 SQL Server 的合法用户，并且具有 SQL Server 的系统管理员权限。

14．3．2 建立登录账户

1、使用企业管理器建立登录账户

2、使用系统存储过程建立登录账户

（1） 建立 SQL Server 身份验证的登录账户：

sp_addlogin

[@loginame=]’ login’ [,[@passwd=]’ password’][,[@defdb=]’ database’]

其中：

（A）[@loginame=]’ login’ ：登录账户名；

（B）[@passwd=]’ password’ ：登录密码；

（C）[@defdb=]’ database’ ：连接的数据库。

（2） 建立 Windows 身份验证的登录账户

sp_grantlogin [@loginame=]’ login’

其中：[@loginame=]’ login’ 为要添加的 Windows NT 用户或组的名称，Windows NT 组和

用户必须用 Windows NT 域名限定，格式为“域\用户”

14．3．3 删除登录账户

1、 使用企业管理器删除登录账户；

2、 使用系统存储过程删除登录账户：

（1）删除 SQL Server 身份验证的登录账户：

sp_droplogin [@loginame=]’ login’

其中：[@loginame=]’ login’：将被删除的登录账户名；

（2）删除 Windows 身份验证的登录账户

sp_revokelogin [@loginame=]’ login’

其中：[@loginame=]’ login’为要删除的 Windows NT 用户或组的名称，Windows NT 组和用户必须用 Windows NT 域名限定，格式为“域\用户”

14．4 管理数据库用户

14．4．1 建立数据库用户

1、 使用企业管理器建立数据库用户；

2、 使用系统存储过程建立数据库用户：

sp_adduser[@ loginame=]’ login’ [,[@name_in_db=]’ user’][,[@grpname=]’ group’]

其中：

（1)[@ loginame=]’ login’：登录账户名，login 必须是已有的 SQL Server 登录账户或 Windows NT 用户名

（2）[@name_in_db=]’ user’：新数据库用户名，如没有指定，则 user 与 login 名相同；

（3）[@grpname=]’ group’：角色名，新用户自动地成为此角色的成员，group 必须是当前数据库已有的角色。

14．4．2 删除数据库用户

1、使用企业管理器删除数据库用户；

2、使用系统存储过程建立数据库用户：

sp_dropuser[@name_in_db=]’ user’

14．5 管理权限

14．5．1 SQL Server 权限种类

1、 对象权限：指用户对数据库中的表、视图等对象中数据的操作权限，相当于数据库操作语言（DML）的语句权限；

2、 语句权限：相当于数据定义语言（DDL）的语句权限，专指是否允许执行：CREATE TABLE、CREATE

VIEW 等与创建数据库对象有关的操作；

3、 隐含权限：指数据库管理系统预定义的服务器角色、数据库角色、数据库拥有者和数据库对象拥有者所具有的权限

14．5．2 权限的管理

权限的管理包括：

（1） 授予权限：允许用户或角色具有某种操作权；

（2） 收回权限：不允许用户或角色具有某种操作权，或收回曾经授予的权限；

（3） 拒绝访问：拒绝某用户或角色具有某种操作权限；

1、 使用企业管理器管理数据库用户权限；

2、 使用企业管理器管理语句权限；

3、 使用 Transact-SQL 语句管理对象权限

（1） GRANT 语句：用于授权；

GRANT 对象权限名 [，…]ON{表名|视图名|存储过程名}

TO{数据库用户名|用户角色名}[，…]

（2） REVOKE 语句：用于收回权限；

REVOKE 对象权限名 [，…]ON{表名|视图名|存储过程名}

TO{数据库用户名|用户角色名}[，…]

（3） DENY 语句：用于拒绝权限。

DENY 对象权限名 [，…]ON{表名|视图名|存储过程名}

TO{数据库用户名|用户角色名}[，…]

4、使用 Transact-SQL 语句管理语句权限

（1）授权语句：

GRANT 对象权限名 [，…] TO {数据库用户名|用户角色名}[，…]

（2）收权语句；

REVOKE 对象权限名 [，…] FROM {数据库用户名|用户角色名}[，…]

（3）拒绝权限。

DENY 对象权限名 [，…] TO {数据库用户名|用户角色名}[，…]

14．6 角色（数据库中具有相同权限的一组用户）

14．6．1 固定的服务器角色

是在服务器级上定义，这些角色具有完成特定服务器级管理活动的权限，用户不能添加、删除

或更改固定的服务器角色。用户的登录账户可以添加到固定的服务器角色中，使其成为服务器角色中的成员，从而具有服务器角色的权限。

固定的服务器角色	描述
Sysadmin	可在 SQL Server 中进行任何活动，该角色的权限包含了所有其它固定的服务器角色权限
Serveradmin	配置服务器范围的设置
Setupadmin	添加和删除链接服务器，并执行某些系统存储过程
Securityadmin	管理服务器登录账户
Processadmin	管理在 SQL Server 实例中运行的进程
Dbcreator	创建、更改和删除数据库
Diskadmin	管理磁盘文件
bulkadmin	执行 BULK INSERT 语句

固定的服务器角色的成员是系统的登录账户，系统内置的

（1） BUILTIN\Administrators 组；

（2） Sa

（3） 域名\Administrators

自动是 sysadmin 角色中的成员。

1、 添加固定的服务器角色成员

（1） 用企业管理器实现；

（2） 用系统存储过程实现；

sp_addsrvrolemember[@loginame=]’ login’ [@rolename=]’ role’

其中：

[@loginame=]’ login’ ：添加到固定服务器角色的登录名称；

[@rolename=]’ role’ ：要将登录添加到的固定服务器角色的名称。

2、 删除固定的服务器角色成员

14．6．2 固定的数据库角色

是在数据库级别上定义，用户不能添加、删除或更改固定的服务器角色。用户的登录账户可以添加到固定的数据库角色中，使其成为成员，从而具有数据库角色的权限。

固定的数据库角色	描述
Db_owner	在数据库中拥有全部权限
Db_accessadmin	可以添加或删除用户 ID

Db_securityadmin	可以管理数据库角色和角色成员，并管理数据库中的语句权限和对象权限
Db_ddladmin	可以建立、修改和删除数据库对象（运行所有的 DDL 语句）
Db_backupoperator	可以进行数据库的备份、恢复操作
Db_datareader	可以查询数据库中所有用户表中的数据
Db_datawriter	可以更改数据库中所有用户表中的数据
Db_denydatareader	不允许查询数据库中所有用户表中的数据
Db_denydatawriter	不允许更改数据库中所有用户表中的数据
public	默认不具有任何权限，但用户可对此角色进行授权

其中 public 角色是一个特殊的角色：

（1） 数据库中的每个用户都自动地是 public 角色成员，用户不能从 public 角色中添加和删除成员；

（2） 用户可以对这个角色进行授权。

1、添加固定的数据库角色成员

（1） 用企业管理器实现；

（2） 用系统存储过程实现；

sp_addrolemember [@rolename=]’ role’ ,[@membername=]’ security_account’

其中：

[@rolename=]’ role’ ：当前固定数据库中的角色名称；

[@membername=]’ security_account’ ：添加到角色中的用户名。

2、删除固定的数据库角色成员

14．6．3 用户自定义的角色

属于数据库一级的角色，用户可根据实际工作职能定义一系列角色，并给每个角色授予合适的权限。只需将数据库用户放置到合适的角色中即可。

1、建立用户自定义库角色

（1） 用企业管理器实现；

（2） 用系统存储过程实现；

sp_addrole [@rolename=]’ role’ ,[@ownername=]’ owner’

其中：

[@rolename=]’ role’ ：新的角色名称；

[@ownername=]’ owner’ ：新角色的所有者。

3、 为用户定义的角色授权；

4、 添加和删除用户自定义角色成员

第 15 章 备份和恢复数据库

15．1 备份数据库

15．1．1 概述

1、备份数据库的作用

（1） 备份数据库就是将数据库数据和与数据库的正常运行有关的信息保存起来，以备恢复数据库时使用，其主要目的是为了防止数据的丢失。

（2） 另一作用是作为数据转移的一种方式；

2、备份时间

（1） 对于系统数据库应进行了修改后立即备份，对于用户数据库一般采用定期备份；

（2） 备份数据库要选在数据库操作少的时间进行，这样可减少对备份以及数据操作性能的影响。

15．1．2 备份设备

1、 SQL Server 将备份数据库的场所称为备份设备，它支持将数据库备份到磁带或磁盘上；

2、 备份方式有两种：

（1）先创建备份设备，然后将数据库备份到备份设备上（永久备份设备）；

（2）直接将数据库备份到物理文件上（临时备份设备）。

15．1．3 创建备份设备

备份设备在操作系统一级实际上是物理文件，只是备份设备必须要先创建好，然后才能使用

1、 用企业管理器创建备份设备

2、 使用系统存储过程创建备份设备

sp_addumpdevice[@devtype=]’ device_type’ ,[@logicalname=]’ logical_name’ , [@physicalname=]’ physical_name’

其中：

（1）[@devtype=]’ device_type’ ：为备份设备的类型，可选下列之一：

（A） Disk：使用磁盘文件作为备份设备；

（B） Pipe：使用命名管道作为备份设备；

（C） Tape：使用磁带设备。

（2）[@logicalname=]’ logical_name’ ：备份设备的逻辑名称，该逻辑名称用在 BACKUP 和 RESTORE 语句中；

（3）[@physicalname=]’ physical_name’ ：备份设备的物理名称。

15．1．4 备份类型

1、 完全备份

是将数据库中的全部信息进行备份，它是恢复的基线，在进行完全备份时，不但备份数据库的数据文件、日志文件，而且还备份文件的存储位置信息以及数据库中的全部对象及相关信息；

备份数据库是要消耗时间的，在进行备份数据库时，用户可以访问数据库，它将不影响数据库的备份，并且还可以将备份过程中发生的活动全部备份下来。

2、差异备份

是备份从最近的完全备份之后对数据库所作的修改，它以完全备份为基点，备份变化了的数据文件和日志文件以及数据库中其他被修改的内容，

在差异备份过程中，也允许用户访问数据库和对其操作，且在备份过程中的活动也一起备份下来；

3、事务日志备份

（1） 是备份从上次备份之后的日志记录，在默认情况下，事务日志备份完成后，要截断日志。

（2） 事务日志记录了用户对数据库进行的修改操作，为了避免记录越来越多，必须定期地将日志记录中不需要的部分清除掉，这种过程叫截断日志，备份日志是截断日志的一种方法；

（3） 如果要进行事务日志备份，必须将数据库的故障还原模型设置为“完全”方式或“大容量日志记录的”方式，因为在默认情况下的“简单”方式，只能进行完全备份和差异备份，不能进行事务日志备份，因为在该模式下系统自动定期将事务日志中不活动的部分清除。

（4） 如要对数据库进行事务日志备份，则必须先设置数据库的故障还原模型，否则在恢复时就会出错。这是因为不同的还原模型对日志的记录和维护方式是不一样的。

15．1．5 备份策略（三种）

1、完全备份

完全备份策略适合于数据库数据不是很大，而且数据更改不是很频繁的情况。可以几天或几周进行一次；

2、完全备份+日志备份

如不允许丢失太多数据，且不希望经常进行完全备份；

3、完全备份+差异备份+日志备份

该策略的好处是备份和恢复的速度都比较快，出现故障时丢失的数据也比较少；

15．1．6 实现备份

1、 使用企业管理器备份数据库

2、 使用 Transact-SQL 语句备份数据库

（1） 备份数据库的基本语法：

BACKUP DATABASE 数据库名

TO {<备份设备名>}|{DISK|TAPE}={‘物理备份文件名’}

[WITH [DIFFERENTIAL][[,]{INIT|NOINIT}]]

其中：

（A）<备份设备名>：将数据库备份到已创建好的备份设备名上；

（B）DISK|TAPE：将数据库备份到磁盘或磁带；

（D） DIFFERENTIAL：进行差异备份；

（E） INIT：本次备份数据库将重写备份设备，即覆盖掉本设备上以前进行的所有备份；

（F） NOINIT：本次备份数据库将追加到备份设备上，即不覆盖。

（2） 备份数据库日志的基本语法：

BACKUP LOG 数据库名

TO {<备份设备名>}|{DISK|TAPE}={‘物理备份文件名’}

[WITH [{INIT|NOINIT}] [{[,]NO_LOG|TRUNCATE_ONLY|NO_TRUNCATE}]]

其中：

（A） NO_LOG 和 TRUNCATE_ONLY：表示备份完日志后要截断不活动的日志；

（B） NO_TRUNCATE：表示备份完日志后不截断不活动的日志；

（C） 其他选项同备份数据库语句的选项。

15．1．7 备份媒体集

1、 当数据库很大时，有时一个备份设备的空间可能不能满足要求，这里就可以将数据库备份到多个不同的备份设备上，同时使用多个备份设备进行备份的为此设备就称为备份媒体集；

2、 使用媒体集与使用单一备份设备的方法一样，只需添加其他备份设备，系统会自动将这些设备作为一个备份媒体集使用；

3、 系统在使用备份媒体集时，基本是将备份所需的空间均衡地分担到每个备份设备上；

4、 如果以后要单独使用某个设备，则必须重新初始化备份媒体集，即删除不需要的设备后，重写并初始化即可，但注意的是，重新初始化后原备份媒体集上所备份的内容将全部丢失。

15．2 恢复数据库

15．2．1 恢复前的准备及恢复顺序

1、恢复前的准备

在恢复数据库前必须限制数据库的访问，一般选择“限制访问”中的“db_owner、dbcreator 或 sysadmin 的成员”，说明只有以上角色才可访问数据库；

2、恢复的顺序

(1) 恢复最近的完全数据库备份；

(2) 恢复完全数据库备份之后最近的差异数据库备份；

(3) 按日志备份的先后顺序恢复自最近的完全或差异数据库备份之后的所有日志。

15．2．2 实现恢复

1、用企业管理器恢复数据库

2、用 Transact-SQL 语句恢复数据库

RESTORE DATABASE 数据库名

FROM 备份设备名

[WITH FILE=文件号[,]NORECOVERY[，]RECOVERY]

其中：

（1） FILE=文件号：标识要还原的备份集，文件号为 1 表示备份设备上的第一个备份集，文件号为 2 表示备份设备上的第二个备份集；

（2） NORECOVERY：表明对数据库的恢复还没有完成，使用此项恢复的数据库是不可用的，但可以继续恢复后续的备份；

（3） RECOVERY：表明对数据库的恢复已完成，一般是在恢复数据库的最后一个备份时使用此项，此时恢复的数据库是可用的。

恢复日志的语法：

RESTORE LOG 数据库名 FROM 备份设备名 [WITH FILE=文件号[，]NORECOVERY[，]RECOVERY]

：

第 19 章 数据库新技术：

一、分布式数据库：

1、 分布式数据库概念：

分布式数据库系统（Def）：物理上分散，逻辑上集中的数据库系统。

分布式数据库：分布式数据库系统上个场地上的数据库的逻辑集合。

2、 分布式数据库的目标：12 个。

其中，最基本特征：本地自治，非集中式管理，高可用性。

分布透明性：位置独立性，数据分布独立性，数据复制独立性。

带来一定复杂性：分布式查询，事务管理。

3、 数据分布策略：先数据分片，再数据分配。

数据分片：对某一个关系进行分片是将关系划分为多个片段，这些片段包括足够的信息可以使关系重构。（四种基本方法：水平，垂直，导出，混合）

数据分配：将数据片分到各个场地去。（方法：集中，分割，全复制，混合式）

4、 分布式数据库的参考模式结构：6 种（p333）

5、 分布透明性：分片透明性（最高级别的）；位置透明性；局部数据模型透明性。

6、 分布式 DBMS：

组成：GDBMS（全局数据库）：是分布式 DBMS 核心，提供终端分布透明性，协调全局事务在各个场地的执行，为全局提供应用支持。

GDD（全局数据字典）：

LDBMS：提供场地自制能力。

CM：通信管理，各个场地之间传送数据和场地。

7、 分布式数据库的相关技术

1、分布式查询：是用户与分布式 DBS 的接口。

考虑以下策略（3 个）：操作执行的顺序，操作执行的算法，不同场地间数据流动的顺序。

1、 分布式事务管理：恢复控制和并发控制。

恢复控制：两阶段协议（1，协同者询问所有参与者事务是否可以提交；2， 协同者根据参与者的回答决定事务的提交）协调者发生故障时导致阻塞。

并发控制：封锁协议。

二、对象数据库（OODB）

1、面向对象数据库的基本概念：对象与对象标识（OID）、封装（状态（属性）+ 行为（方法））、类、继承性、滞后联编（在运行时，根据具体环境选择相应的实现算法，而不是在编译时）、对象嵌套（一个对象的属性也是一个对象）

2、面向对象技术和数据库技术（2 类）

1、面向对象的数据库系统：在面向对象程序设计语言中引入数据库技术。如 ObjectStore、O2 等。

1、持久程序化设计。（持久数据：创建这些数据的程序运行终止后，数据仍然存在于系统中。）

1、嵌入式 SQL 语言在映射不同数据库的时候需要 DBA 将类型转换，而持久化程序设计语言中，查询语言是完全集成到宿主语言中，

2、DBA 直接操纵持久数据。

2、持久语言的基本概念。

1、对象持久化。（普通的，瞬态对象标识）

2、对象标识和指针。（持久化指针）

3、持久对象的存储和访问。（名字、标识、聚集形式存放，用循环寻找）

2、对象-关系数据库：从关系数据库系统中自然的加入面向对象技术。如 MS 的 SQL Server\Oracle 的 Oracle8i 以及 IBM 的 DB2 UDB 等

特点：扩充数据类型、支持复杂对象、支持继承（under）的概念、提供通用的规则系统。

三、并行数据库

1、并行数据库共享结构（4 种：共享内存，共享磁盘，无共享，层次）p343

2、数据划分：

一维数据划分：轮转法、散列法、范围划分。

多维数据划分：BERD、CMD、MAGIC 等

2、 并行算法（将算法任务分配到各个处理器上）

并行排序：非范围的，并行外排序归并

并行连接：划分连接，分片-复制连接

其他关系操作：选择，消除重复，投影，聚集。