第一章

1. 数据是数据库中存储的基本对象。

2. 数据库是长期储存在计算机内、有组织的、可共享的大量数据的集合。数据库数据具有永久存储、有组织和可共享三个基本特点。

3. 数据模型是对现实世界数据特征的抽象。通俗地讲数据模型就是现实世界的模拟。数据模型是数据库系统的核心和基础 。

4. 数据库管理系统的基本功能

- 数据定义功能。DDL，对数据对象的组成与结构进行定义。
- 数据组织、存储和管理。目标是提高存储空间利用率和方便存取，提供多种存取方法（索引查找、hash 查找、顺序查找等）来提高存取效率
- 数据操纵功能。DML，实现对数据库的查询和插删改
- 数据库的事务管理和运行管理。保证数据的安全性、完整性、并发使用及故障后的系统修复。
- 数据库的建立和维护功能。初始数据的输入、转换；数据库的转储、恢复；数据库的重组织和性能监视、分析功能。
- 其它功能。DBMS 与其他软件的通信，DBMS 与 DBMS/文件系统的数据转化，异构数据库之间的互访与互操作。

5. 数据库系统（阶段）的特点

（1）数据结构化

存取数据的方式灵活，存取粒度可是一个或一组数据项，一个或一组记录。而文件系统只能是记录。

（2）数据的共享性高，冗余度低且易扩充

数据面向整个系统，可被多个用户、应用共享。减少数据冗余、数据的不相容性与不一致性。这也使得数据库系统弹性大，易于扩充，可以适应各种用户的要求。

（3）数据独立性高

- 物理独立性是指用户的应用程序与数据库中数据的物理存储是相互独立的。
- 逻辑独立性是指用户的应用程序与数据库的逻辑结构是相互独立的。
- 数据独立性由数据库管理系统的二级映像功能来保证

（4）数据由数据库管理系统统一管理和控制

- 数据的安全性（Security）保护。
- 数据的完整性（Integrity）检查。
- 并发（Concurrency）控制。
- 数据库恢复（Recoery）。

6. 常用的数据模型：

- 层次模型（Hierarchical Model）。IBM 的 IMS 系统
- 网状模型（Network Model）。DBTG 系统/CODASYL 系统、IDMS、DMS、IDS/2、IMAGE
- 关系模型（Relational Model)）
- 面向对象数据模型（Object Oriented Data Model）
- 对象关系数据模型（Object Relational Data Model）。关系模型+面向对象模型
- 半结构化数据模型（Semistruture Data Model）。如 XML

其中层次模型和网状模型统称为结构化模型。

7. 数据库的三级模式、二级映像、两个独立性

（1）模式（schema）/逻辑模式（逻辑结构设计阶段。建立基本表等）

- 定义：模式是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。
- 模式实际是数据库数据在逻辑级上的视图。一个数据库只有一个模式。它以某一种数据模型（关系模型）为基础，统一综合考虑了所有用户的需求，并将这些需求有机地结合成一个逻辑整体。

（2）外模式（external schema) /子模式 /用户模式（也在逻辑结构设计阶段。建立视图等）

- 定义：外模式是数据库用户（包括应用程序员和最终用户）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。
- 外模式通常是模式的子集，一个数据库可以有多个外模式。反映了不同的用户的应用需求、看待数据的方式、对数据保密的要求。对模式中同一数据，在外模式中的结构、类型、长度、 保密级别等都可以不同

（3）内模式（internal schema）/存储模式（物理结构设计阶段，建立索引等）

- 定义：内模式是数据物理结构和存储方式的描述，是数据在数据库内部的组织方式一个数据库只有一个内模式。

（4）三级模式是对数据的三个抽象级别，二级映象在数据库管理系统内部实现这三个抽象层次的联系和转换。正是这两层影响保证了数据库中的数据能够具有较高的逻辑独立性和物理独立性。

（5）外模式-模式映象模式描述的是数据的全局逻辑结构。外模式描述的是数据的局部逻辑结构。同一个模式可以有任意多个外模式。这些映象定义通常包含在各自外模式描述中 。
（6）外模式-模式映象保证数据的逻辑独立性。当模式改变时，数据库管理员对外模式 / 模式映象作相应改变，使外模式保持不变。应用程序是依据数据的外模式编写的，从而应用程序不必修改，保证了数据与程序的逻辑独立性，简称数据的逻辑独立性。
（7）模式-内模式映象定义了数据全局逻辑结构与存储结构之间的对应关系。数据库中模式-内模式映象是唯一的 。该映象定义通常包含在模式描述中 。
（8）模式-内模式映象保证数据的物理独立性。当数据库的存储结构改变了（例如选用了另一种存储结构），数据库管理员修改模式 / 内模式映象，使模式保持不变。 从而应用程序不受影响。保证了数据与程序的物理独立性， 简称数据的物理独立性。

（9）数据库的二级映像保证了数据库外模式的稳定性，从底层保证了应用程序的稳定性，除非应用需求本身发生变化，否则应用程序一般不需要修改

8. 两类数据模型：概念模型；逻辑模型和物理模型

- 概念模型也称信息模型，它是按用户的观点来对数据和信息建模，主要用于数据库设计。
- 逻辑模型是按计算机系统的观点对数据建模，主要用于 DBMS 的实现。常见的逻辑模型有层次模型、网状模型、关系模型、面向对象数据模型、对象关系数据模型、半结构化数据模型。
- 物理模型是对数据最底层的抽象，它描述数据在系统内部的表示方式和存取方法，或在磁盘或磁带上的存储方式和存取方法，是面向计算机系统的。

9. 数据模型的组成元素：数据结构、数据操作、数据的完整性约束条件

第二章 关系数据库

1. 候选码（Candidate key）（属性组）：若关系中的某一属性组的值能唯一地标识一个元组，而其子集不能（最小属性组），则称该属性组为候选码。

另有：主码（属性组）主属性（诸属性）全码（All-key）（候选码是所有属性）

2. **规则 2.1** 实体完整性规则 若属性（指一个或一组属性）A 是基本关系 R 的主属性，则属性 A 不能取空值 （null alue）（或重复值？）。所谓空值就是“不知道”或“不存在”或“无意义”的值。

3. **规则 2.2** 参照完整性规则 若属性（或属性组）F 是基本关系 R 的外码它与基本关系 S 的主码 Ks 相对应（基本关系 R 和 S 不一定是不同的关系）， 则对于 R 中每个元组在 F 上的值必须为：

- 或者取空值（F 的每个属性值均为空值）（参照关系的外码必不是自身的主码）
- 或者等于 S 中某个元组的主码值

4. 用户定义的完整性就是针对某一具体关系数据库的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求。

5. 关系代数是一种抽象的查询语言，它用对关系的运算来表达查询

第三章 关系数据库标准语言 SQL

1. SQL，结构化查询语言，是一种高度非过程化的语言，是关系数据库的标准语言。是集数据查询（data query）、数据定义语言（DDL），数据操纵语言（DML），数据控制语言（DCL）功能于一体的关系数据语言。

2. SQL 的特点

（1）综合统一

SQL 集查询（DQL）、数据定义语言（DDL），数据操纵语言（DML），数据控制语言（DCL）功能于一体的关系数据语言，可以独立完成数据库生命周期中的全部活动：

- 定义和修改、删除关系模式，定义和删除视图，插入数据，建立数据库
- 对数据库中的数据进行查询和更新
- 数据库重构和维护
- 数据库安全性、完整性控制，以及事务控制

- 嵌入式 SQL 和动态 SQL 定义

（2）高度非过程化

- 非关系数据模型的数据操纵语言“面向过程”，必须指定存取路径。
- SQL 只要提出“做什么”，而无需指明“怎么做”，因此无须了解存取路径。 存取路径的选择以及 SQL 的操作过程由系统自动完成。

（3）面向集合的操作方式

- 非关系数据模型采用面向记录的操作方式，操作对象是一条记录
- SQL 采用集合操作方式。不仅操作对象、查找结果可以是元组的集合。而且一次插入、删除、更新操作的对象可以是元组的集合

（4）以同一种语法结构提供多种使用方式

- SQL 是独立的语言，用户可以在终端键盘上直接键入 SQL 命令对数据库进行操作
- SQL 又是嵌入式语言，SQL 能够嵌入到高级语言（例如 C，C++，Jaa）程序中，供程序员设计程序时使用
- 而在两种不同的使用方式下，SQL 的语法结构基本上是一致的。

（5）语言简洁，易学易用

- SQL 功能极强，完成核心功能只用了 9 个动词。

3. 视图的作用

（1）视图能够简化用户的操作

（2）视图可以更清晰地表达查询

（3）视图使用户能以多种角度看待同一数据

（4）视图能够对机密数据提供安全保护

（5）视图对重构数据库提供了一定程度的逻辑独立性

4. SQL 语言：第三章、4.2.4 授权（授予与收回，权限、用户、角色）、4.4 审计、5.2 参照完整性（拒绝执行、级联操作、设置控制）、5.3 用户定义的完整性（属性上、元组上）、5.4 完整性约束命名子句、8.1 嵌入式 SQL（ESQL）

第四章 数据库安全性

1. 数据库的安全性是指保护数据库以防止不合法使用所造成的数据泄露、更改或破坏。

2. 计算机系统的三类安全问题

（1）非授权用户对数据库的恶意存取和破坏

（2）数据库中重要或敏感的数据被泄露

（3）安全环境的脆弱性

3. 实现数据库安全性控制的常见方法及技术

（1）用户身份鉴别

包括：静态口令鉴别、动态口令鉴别、生物特征鉴别、智能卡鉴别

（2）多层存取控制（自主存取控制 DAC、强制存取控制 MAC）

（3）视图

（4）审计

（5）数据加密（存储加密、传输加密）

第五章 数据库完整性

1. 数据库的完整性（integrity）是指数据的正确性（correctness）和相容性（compat-ability）。

- 数据的正确性是指数据是符合现实世界语义，反映了当前实际状况的 （用户定义的完整性）
- 数据的相容性是指数据库同一对象在不同关系表中的数据是符合逻辑的（实体完整性和参照完整性）

2. 数据库完整性的类型

实体完整性、参照完整性、用户定义的完整性

（1）实体完整性规则 若属性（指一个或一组属性）A 是基本关系 R 的主属性，则属性 A 不能取空值 （null alue）（或重复值？）。所谓空值就是“不知道”或“不存在”或“无意义”的值。

（2）参照完整性规则 若属性（或属性组）F 是基本关系 R 的外码它与基本关系 S 的主码 Ks 相对应（基本关系 R 和 S 不一定是不同的关系）， 则对于 R 中每个元组在 F 上的值必须为：

- 或者取空值（F 的每个属性值均为空值）（参照关系的外码必不是自身的主码）
- 或者等于 S 中某个元组的主码值

（3）用户定义的完整性就是针对某一具体关系数据库的约束条件，反映某一具体应用所涉

及的数据必须满足的语义要求。

3. 数据库完整性约束条件（完整性规则）

第六章 关系数据理论

1. 规范化：一个低一级范式的关系模式通过模式分解可以转换为若干个高一级范式的关系模式的集合，这种过程就叫规范化。

2. 范式理论及其应用

3. 极小函数依赖集的概念 P193

第七章 数据库设计

1. 数据库设计的基本步骤、理论、方法、特点

（一）特点

1. 数据库建设的基本规律 —— 三分技术，七分管理，十二分基础数据

2. 结构（数据）设计和行为（处理）设计相结合（数据库设计与应用系统设计相结合）

（二）理论

- ☐ 关系规范化理论、函数依赖理论
- ☐ 它要求多方面的知识和技术。主要包括：
  - ➢ 计算机的基础知识
  - ➢ 软件工程的原理和方法
  - ➢ 程序设计的方法和技巧
  - ➢ 数据库的基本知识
  - ➢ 数据库设计技术
  - ➢ 应用领域的知识

（三）方法

1. 手工试凑法

2. 规范设计法

- 新奥尔良（New Orleans）方法
- 基于 E-R 模型的数据库设计方法
- 3NF（第三范式）的设计方法
- 面向对象的数据库设计方法
- 统一建模语言（UML）方法

（四）步骤

按照结构化系统设计的方法，数据库设计分 6 个阶段

1. 需求分析 （用户需求分析报告）

- 是整个设计过程的基础和地基
- 首先准确了解和分析用户需求（包括数据与处理）
- 需求分析做得是否充分与准确，决定了构建数据库的速度和质量
- 若做的不好则可能导致数据库设计返工重做

2. 概念结构设计

- 是整个设计过程的关键
- 通过对用户需求进行综合、归纳与抽象，形成一个独立于具体数据库管理系统的概念模型（E-R 图）

3. 逻辑结构设计 （外模式）

- 将概念结构转换为某个 DBMS 所支持的数据模型（关系模型），并对其进行优化

4. 物理结构设计 （内模式）

- 为逻辑数据结构选取一个最适合应用环境的物理结构（包括存储结构和存取方法，如索引）

5. 数据库实施

- 运用 DBMS 提供的数据库语言及其宿主语言，根据逻辑设计和物理设计的结果构建数据库
- 编写与调试应用程序
- 组织数据入库并进行试运行

6. 数据库运行和维护

- 经过试运行后即可投入正式运行
- 在运行过程中必须不断对其进行评估、调整与修改

btw：需求分析和概念设计独立于任何数据库管理系统（DBMS），逻辑设计和物理设计与选用的数据库管理系统（DBMS）密切相关

2. 概念结构设计与 ER 图绘制

3. E-R 图（视图）集成的方法

在开发一个大型信息系统时，最经常采用的策略是自顶向下地进行需求分析，然后再自底向上地设计概念结构。即首先设计各子系统的分 E-R 图，然后将它们集成起来，得到全局 E-R 图。

一般说来，E- R 集成可以有两种方式：

- 多个分 E-R 图一次集成；
- 逐步集成，用累加的方式一次集成两个分 E-R 图。

无论采用哪种方式，E-R 图的集成都要分两步：

- 合并。解决各分 E-R 图之间的冲突（属性冲突、命名冲突、结构冲突），将分 E-R 图合并起来生成初步 E-R 图。
- 修改和重构。利用分析方法和规范化理论，消除不必要的冗余，生成基本 E-R 图

3. 逻辑结构设计与 ER 图转换

4. 物理设计的内容——存储结构和存取方法

- 为关系模式选择存取方法（建立存取路径）—— 索引（B+树、Hash）和聚簇
- 设计关系、索引等数据库文件的物理存储结构 —— 指定数据的存放位置和存储结构。确定关系、索引、聚簇、日志、 备份等的存储安排和存储结构，确定系统配置等。

5. 数据库维护的内容

（1）数据库的转储和恢复

（2）数据库性能的监督、分析和改进

（3）数据库的重组织与重构造

（4）数据库的安全性、完整性控制

第八章 数据库编程

1. 存储过程的优点

（1）运行效率高。因为不像解释执行的 SQL 语句那样，提出请求时才进行语法分析和优化。提供了在服务器端快速执行 SQL 的途径。

（2）降低了客户机和服务器之间的通信量 。客户机只要像服务器发送调用存储过程的名字和参数。并且只有最终的处理结果才返回客户端。

（3）方便实施企业规则。把企业规则的运算程序写成存储过程放入服务器中当要变化时只要修改存储过程即可，无需修改其他应用程序。

2. ODBC 的工作流程

（1）配置数据源

配置数据源有两种方法：

- 运行数据源管理工具来进行配置
- 使用 Driver Manager 提供的 ConfigDsn 函数来增加、修 改或删除数据源

（2）初始化环境

由 Driver Manager 来进行控制 ，并配置环境属性

（3）建立连接

应用程序调用 SQLAllocHandle 分配连接句柄， 通过 SQLConnect、SQLDriverConnect 或 SQLBrowseConnect 与数据源连接

（4）分配语句句柄

处理任何 SQL 语句之前，应用程序还需要首先分配一个语句句柄。语句句柄含有具体的 SQL 语句以及输出的结果集等信息

（5）执行 SQL 语句

应用程序处理 SQL 语句的两种方式：

- 预处理（SQLPrepare、SQLExecute 适用于语句的多 次执行）
- 直接执行（SQLExecdirect）

（6）结果集处理

应用程序可以通过 SQLNumResultCols 来获取结果集中的列数，通过 SQL DescribeCol 或是 SQLColAttrbute 函 数来获取结果集每一列的名称、数据类型、精度和范围

第九章 关系查询处理和查询优化

关系数据库管理系统查询处理的步骤

1. 查询分析

查询分析的任务：对查询语句进行扫描、词法分析和语法分析

- 词法分析：从查询语句中识别出语言符号，如 SQL 关键词、属性名、关系名等。等
- 语法分析：进行语法检查和语法分析，判断是否符合 SQL 语法规则

2. 查询检查

查询检查的任务：语义分析、符号名转化、安全性检查、完整性初步检查

（1）对合法的查询语句进行语义检查，即根据数据字典中有关的模式定义检查语句中的数据库对象，如关系名、属性名是否存在和有效。

（2）如果是对视图的操作，则要用视图消解方法把对视图的操作转换成对基本表的操作

（3）根据数据字典中的用户权限和完整性约束定义对用户的存取权限进行检查。这里的完整性检查知识初步的、静态的检查。

（4）检查通过后把 SQL 查询语句转换成内部表示，即等价的关系代数表达式。这个过程中要把数据库对象的外部名称转换为内部表示。

（5）关系数据库管理系统一般都用查询树，也称为语法分析树来表示扩展的关系代数表达式。

3. 查询优化

（1）查询优化：选择一个高效执行的查询处理策略

（2）查询优化分类

- 代数优化/逻辑优化：指关系代数表达式的优化。即按照一定规则，对关系代数表达式进行等价变换，改变操作的次序和组合，使查询执行更高效。
- 物理优化：指存取路径和底层操作算法的选择

（3）选择依据

- 基于规则(rule based)
- 基于代价(cost based)
- 基于语义(semantic based)

（4）实际关系数据库管理系统中的查询优化器都综合运用了这些优化技术。

4. 查询执行

（1）依据优化器得到的执行策略生成查询执行计划

（2）由代码生成器(code generator)生成执行查询计划的代码

（3）两种执行方法

- 自顶向下
- 自底向上

第十章 数据库恢复技术

1.事务的特性

事务的 ACID 特性——原子性（Atomicity）、一致性（Consistency）、隔离（Isolation）、持续性（Durability ）

（1）原子性

事务是数据库的逻辑工作单位，事务中包括的诸操作要么都做，要么都不做

（2）一致性

- 事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态
- 一致性状态：数据库中只包含成功事务提交的结果 ，就说数据库处于一致性状态
- 不一致性状态：数据库系统运行中发生故障，有些事务尚未完成就被迫中断，这些未完成事务对数据库所做的修改有一部分已写入物理数据库，这时数据库就处于一种不正确的状态

（3）隔离性

一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对其他并发事务是隔离的，并发执行的各个事务之间不能互相干扰

（4）持续性（永久性）

指的是一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。 接下来的其他操作或故障不应该对其执行结果有任何影响。

2.数据库运行可能产生的基本故障及其类别

(1) 事务内部的故障

有的是可以通过事务程序本身发现的(例见书 P295) ；有的是非预期的，不能由事务程序处理的。

(2) 系统故障

称为软故障，是指造成系统停止运转的任何事件，使得系统要重新启动。

(3) 介质故障

称为硬故障，指外存故障。如磁盘损坏、磁头碰撞、瞬时强磁场干扰等。介质故障破坏数据库或部分数据库，并影响正在存取这部分数据的所有事务

(4) 计算机病毒

计算机病毒是一种人为的故障或破坏，是一些恶作剧者研制的一种计算机程序。可以繁殖和传播，造成对计算机系统包括数据库的危害

(5) 故障小结

各类故障，对数据库的影响有两种可能性：

- 一是数据库本身被破坏
- 二是数据库没有被破坏，但数据可能不正确，这是由于事务的运行被非正常终止造成的。

3.数据库恢复的实现技术——建立冗余数据——数据转储（backup） 、登记日志文件（logging）

数据转储

(1) 转储是指数据库管理员定期地将整个数据库复制到磁带、磁盘或其他存储介质上保存起的过程。备用的数据文本称为后备副本(backup)或后援副本。

(2) 数据库遭到破坏后可以将后备副本重新装入，重装后备副本只能将数据库恢复到转储时的状态，要想恢复到故障发生时的状态，必须重新运行自转储以后的所有更新事务

(3) 数据转储有两种方式（海量、增量），分别可以在两种状态下进行（动态、静态）。因此数据存储方式可以分为 4 类：

转储方式	转储状态	
	动态转储	静态转储
海量转储	动态海量转储	静态海量转储
增量转储	动态增量转储	静态增量转储

登记日志文件

(1) 日志文件(log file)是用来记录事务对数据库的更新操作的文件

(2) 日志文件的作用

- 事务故障恢复和系统故障恢复必须用日志文件。
- 在动态转储方式中必须建立日志文件，后备副本和日志文件结合起来才能有效地恢复数据库。

第十一章 并发控制

1.并发操作带来的数据不一致性及封锁技术（举例说明）

（一）并发操作带来的数据不一致性包括丢失修改、不可重复读和读“脏”数据。（lost update，non-repeatable read，dirty read）。结合书 P310 实例理解。

（二）封锁技术

1.排它锁（Exclusive Locks，简记为 X 锁）

- 排它锁又称为写锁
- 若事务 T 对数据对象 A 加上 X 锁，则只允许 T 读取和修改 A，其它任何事务都不能再对 A 加任何类型的锁，直到 T 释放 A 上的锁
- 保证其他事务在 T 释放 A 上的锁之前不能再读取和修改 A

2. 共享锁（Share Locks，简记为 S 锁）

- 共享锁又称为读锁
- 若事务 T 对数据对象 A 加上 S 锁，则事务 T 可以读 A 但不能修改 A，其它事务只能再对 A 加 S 锁，而不能加 X 锁，直到 T 释放 A 上的 S 锁
- 保证其他事务可以读 A，但在 T 释放 A 上的 S 锁之前不能对 A 做任何修改

1.常用意向锁：意向共享锁(Intent Share Lock，简称 IS 锁）、意向排它锁(Intent Exclusive Lock，简称 IX 锁）、共享意向排它锁(Share Intent Exclusive Lock，简称 SIX 锁)

2.IS 锁：如果对一个数据对象加 IS 锁，表示它的后裔结点拟（意向）加 S 锁。

□ 例如：事务 T1 要对 R1 中某个元组加 S 锁，则要首先对关 系 R1 和数据库加 IS 锁

3.IX 锁：如果对一个数据对象加 IX 锁，表示它的后裔结点拟 （意向）加 X 锁。

□ 例如：事务 T1 要对 R1 中某个元组加 X 锁，则要首先对关系 R1 和数据库加 IX 锁

4.SIX 锁：如果对一个数据对象加 SIX 锁，表示对它加 S 锁，再加 IX 锁，即 SIX = S + IX。

□ 例：对某个表加 SIX 锁，则表示该事务要读整个表（所以要对该表加 S 锁），同时会更新个别元组（所以要对该表加 IX 锁）

5.意向锁的相容矩阵

T_1 \ T_2	S	X	IS	IX	SIX	–
S	Y	N	Y	N	N	Y
X	N	N	N	N	N	Y
IS	Y	N	Y	Y	Y	Y
IX	N	N	Y	Y	N	Y
SIX	N	N	Y	N	N	Y
–	Y	Y	Y	Y	Y	Y

Y=Yes，表示相容的请求　　N=No，表示不相容的请求

(a) 数据锁的相容矩阵