- Chap1 数据库系统绪论
- Chap2 关系数据库
- Chap4 数据库安全性
- Chap5 数据库完整性
- Chap7 数据库设计
- Chap8 数据库编程
- Chap9 查询优化
- Chap10 事务与数据恢复
- Chap11 并发控制

Chap1 数据库系统绪论

- 数据库系统概述
- 数据模型
- 数据库系统的结构
- 数据库系统的组成

数据库系统概述

1. 基本概念：数据、数据库、数据库管理系统、数据库系统。
2. 数据：数据库中存储的基本对象。
    - 分类：**结构化数据**、**半结构化数据**、**非结构化数据**。
3. 信息：具有**时效性**的，有一定含义的，有逻辑的、经过**加工处理的**、对决策有价值的**数据流**。
    - 与数据的联系：
        - **信息=数据+处理。**
        - **数据是符号化的信息**；**信息是语义化的数据。**
4. 知识：对信息使用**归纳、演绎、比较**等手段进行挖掘，使其有价值的部分沉淀下来，并于已存在的人类知识体系相结合，有价值的信息就转变成知识。
    - 知识是从**定量到定性**的过程中得以实现的、抽象的、逻辑的东西。
5. 数据库（DB）：是长期**存储在计算机内、有组织的、可共享的**大量**数据的集合。**
    - 基本特征：（1）按一定的**数据模型组织、描述和存储**；（2）可为**用户共享**；（3）**冗余度较小**；（4）**数据独立性较高**；（5）易扩展。
6. 数据库管理系统（DBMS）：是位于用户与操作系统之间的数据管理软件，是**基础软件**，是一个**大型复杂的软件系统**，可以帮助用户**定义、创建、维护**和**控制**数据库访问的软件系统。
    - 用途：（1）科学地**组织和存储**数据（自动管理）；（2）高效地**获取和维护**数据（提供工具和接口）。
    - 功能：
        - **数据定义**：提供数据定义语言（DDL）；
        - **数据组织、存储和管理**：**分类**组织、存储和管理数据、**确定**组织数据的**文件结构和存取方式**、提供**多种存取方法提高存取效率**；
        - **数据操纵**：提供数据操纵语言（DML），实现**增、删、改、查**；
        - **事务管理和运行管理**：**统一管理和控制**数据库的建立、运行和维护、保证数据的**安全性、完整性**、支持多用户对数据的**并发使用**、支持（自动、手动）**系统数据恢复。**

        - **建立和维护**数据库：初始化、数据装载和转换、备份、转储、恢复、重组织、性能监视与分析等。
7. 数据库系统（DBS）：
    - 构成：
        - 数据库：（1）底层的数据文件、文件结构和存取方式；（2）数据字典，即数据库结构/模式描述；
        - 数据库管理系统：DBMS及其应用开发工具；
        - 应用程序；
        - 数据库管理员（DBA）。
8. 常见管理信息系统（MIS）架构：（1）Client/Server(C/S)架构；（2）Browser/Server(B/S)架构。
    - （1）C/S架构：通常是两层，**胖**客户端；（2）B/S架构：通常是三层，**瘦**客户端。
9. 数据管理：对数据进行**分类、组织、编码、存储、检索和维护。**
    - 发展过程：**人工管理**阶段（1940s中期以前）-> **文件系统**阶段（1940s后期-50s中期）-> **数据库系统**阶段（1950s后期至今）。
10. 数据库系统的特点：（1）数据**结构化**；（2）数据**共享性高**、**冗余度低**且易扩成；（3）数据**独立性高**；（4）数据由DBMS**统一管理和控制。**

数据模型

1. 数据模型：**对现实世界数据特征的抽象。**
    - 层次：（1）**概念模型**；（2）**逻辑模型和物理模型。**
        - 概念模型：按**用户的观点**建模（需求分析阶段）；
        - 逻辑模型：按**计算机系统的观点**建模（系统分析阶段）；
        - 物理模型：对**数据最底层的抽象**（系统详细设计阶段）。
    - 抽象过程：**理解** -> **区分** -> **命名** -> **表达。**
2. 概念模型：是对**信息世界的建模**，是现实世界到信息世界的一个中间层次，是数据库设计的有力工具，是数据库**设计人员和用户之间**进行交流的语言。
3. 信息世界中的基本概念：
    - 实体：客观存在并可相互区别的事物。可以是具体的**人、事、物或抽象的概念**；
    - 属性：实体所具有的某一特性；
    - 码/键：**唯一标识**实体的属性集；
    - 实体型：用**实体名及其属性名集合**来抽象和刻画**同类实体**的结构/类；
    - 实体集：同一类型**实体的集合**称为实体集；
    - 联系：现实世界中**事物内部**以及**事物之间**的联系在信息世界中反映为实体（型）内部的联系和实体（型）之间的联系。
4. 数据模型的组成要素：（1）数据结构；（2）数据操作；（3）数据的**完整性约束。**
    - 数据结构：描述数据库的**组成对象及对象之间的联系**，是对**系统静态特性的描述**；
    - 数据操作：对数据库中各种对象（型）的实例（值）允许执行的**操作的集合**，包括**操作**及有关的**操作规则。**
    - 数据完整性约束：是一组**完整性规则的集合**，来**保证数据的正确、有效和相容。**
        - 完整性规则：给定的数据模型中**数据及其联系**所具有的制约和依存规则。
5. 常用的数据模型：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、图数据模型。
6. 层次模型：用**树形结构**来表示各类实体及实体间的联系。

- 特点：
  - 结点的父节点是唯一的，只能直接处理**一对多**的实体联系；
  - 任何记录值只有按其路径查看时，才能显出全部意义；
  - 没有子节点的记录值能脱离父节点记录值而存在。
- 优点：（1）简单清晰；（2）查询效率高；（3）提供良好的完整性支持。
- 缺点：（1）表示多对多联系不自然；（2）对增删操作限制多、程序复杂；（3）查询子节点必须通过父节点；（4）层次命令趋于程序化。

7. 网状模型：满足**一个结点具有零个或多个父结点**的层次联系的集合。
   - 与层次模型的联系：层次模型实际上**是网状模型的一个特例。**
8. 关系模型：数据的**逻辑结构是一张二维表**，它由行和列组成。
   - 数据结构：（1）关系；（2）元组；（3）属性；（4）主码/键；（5）域；（6）**分量**；（7）关系模式。
   - **关系必须是规范化的，满足一定的规范条件。**
     - 最基本的规范条件：**不允许表中还有表。**
   - 关系的**完整性约束**条件：（1）实体完整性；（2）参照完整性；（3）用户定义的完整性。
   - 优点：（1）建立在严格的数学概念基础上；（2）概念单一；（3）存取路径对用户透明。
   - 缺点：（1）查询效率不如**格式化数据模型（层次/网状模型）**；（2）查询优化增加了开发数据库管理系统的难度。

数据库系统的结构

1. 数据库系统结构：
   - 从**数据库应用开发人员角度**：**三级模式结构**，是数据库系统**内部的系统结构**；
   - 从**数据库最终用户角度**：（1）单用户结构；（2）分布式结构；（3）客户-服务器；（4）浏览器-应用服务器/数据库服务器多层结构等。
2. 数据库系统的**三级模式结构**：模式、外模式、内模式。
   - **模式**：数据库中**全体数据的逻辑结构和特征的描述。**
     - 特点：是所有用户的**公共数据视图**；一个数据库**只有一个模式**；是数据库系统模式结构的**中间层。**
   - **外模式**（用户模式）：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。
     - 特点：外模式通常是模式的**子集**；**一个数据库可以有多个外模式**；介于**模式与应用之间**（一个外模式可为多个应用所用，但**一个应用只能使用一个外模式**）。
     - 用途：（1）保证**数据库安全性**；（2）使每个用户**只能看见和访问对应外模式中的数据。**
   - **内模式**（存储模式）：是**数据物理结构和存储方式的描述。**
     - 特点：**一个数据库只有一个内模式**；内模式对用户是"**半透明**"的。
3. **二级映像**：在数据库管理系统**内部实现三个抽象层次的联系和转换。**
   - 类别：外模式/模式映像（通常定义在各自外模式的描述中）；模式/内模式映像（通常定义在模式中）。
     - 外模式/模式映像：当模式改变时，数据库管理员对**外模式/模式映像作相应改变，是外模式保持不变**，从而保证**数据的逻辑独立性。**
     - 模式/内模式：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/内模式映像是唯一的。**当数据库存储结构改变了，数据库管理员修改**模式/内模式映像**，使模式保持不变，从而保证**数据的物理独立性。**

- 作用：（1）保证了数据库**外模式的稳定性**；（2）从底层**保证了应用程序的稳定性**；（3）保证**数据与程序之间的独立性**；（4）数据的存取由数据库管理系统管理。

数据库系统的组成

1. 数据库系统的组成：数据库、数据库管理系统（及其开发工具）、应用程序、数据库管理员（硬件平台及数据库、软件、人员）。
2. 数据库系统对硬件资源的要求：（1）足够大的内存；（2）足够大的磁盘或磁盘阵列等设备；（3）较高的通信能力。
3. 软件：包括（1）**数据库管理系统**；（2）支持数据库管理系统允许的**操作系统**；（3）与数据库接口的**高级语言及其编译系统**；（4）以数据库管理系统为核心的**应用开发工具**；（5）为特定应用环境开发的**数据库应用系统。**
4. 人员：数据库管理员、系统分析员和数据库设计人员、应用程序员、最终用户。
    - 数据库管理员的职责：
        - 决定数据库中的**信息内容和结构**；
        - 决定数据库的**存储结构**和存储策略**；
        - 定义数据的**安全性要求**和**完整性约束条件**；
        - **监控数据库的使用和运行**；
        - **数据库的改进和重组。**

Chap2 关系数据库

- 关系数据结构及形式化定义
- 关系操作
- 关系的完整性
- 关系代数

关系数据结构及形式化定义

1. （候选）键/码：某个关系的一组属性所组成的集合，满足：
    1. 该属性集合始终能够确保在关系中**唯一标识一个元组**；
    2. 该属性集合的**任意真子集不满足条件1。**
    - 将满足上述第1个条件的属性集合称为**超键**，因此候选键可定义为**最小超键。**
2. 基本规范条件：**关系的每一个分量必须是一个不可分的数据项。**
3. 三类关系：（1）**基本关系**（基本表）；（2）**查询表**；（3）**视图表。**
4. 关系模式：是关系的**结构**，是对关系的描述。
    - 关系模式是稳定的；关系是关系模式在某一时刻的值，是随时间变化的。
5. 关系数据库：在一个给定的应用领域中，**所有关系的集合**构成一个关系数据库。
    - 型与值：关系数据库模式是**型**；若干关系模式在某一时刻对应的关系的集合是**值。**

关系操作

6. 基本操作：**选择、投影、并、差、笛卡儿积。**
    - 操作的对象是**集合。**
    - 其他操作：除、连接、交。
7. 关系数据语言：（1）关系代数语言（如ISBL）；（2）（元组/域）关系演算语言；（3）具有关系代数和关系演算双重特点的语言（如SQL）。

关系的完整性

8. 关系的两个**不变性**：**实体完整性**和**参照完整性。**
    - 关系模型必须满足的完整性约束条件，应由关系系统自动支持。
9. 用户定义的完整性：应用领域需要遵循的约束条件，体现了具体领域中的语义约束。

关系代数

10. 关系代数：用对关系的运算来表达查询。
    - 传统集合运算：交、并、差、笛卡儿积。
    - 专门关系运算：选择、投影、连接、除。
11. 投影运算特点：投影之后不仅取消了原关系中的某些列，而且还**可能取消某些元组（避免重复行）**。
12. 自然连接：一种**特殊的等值连接**，在**结果中把重复的属性列去掉。**
13. 悬浮元组：两个关系R和S在做**自然连接**时，一者中某些元组可能在另一者中**不存在公共属性上值相等的元组**，从而造成这些元组在操作时被舍弃了。这些被舍弃的元组称为**悬浮元组。**
14. 外连接：把悬浮元组也保存在结果关系中，其他属性上填空值。
    - 左外连接：只保存左边关系的悬浮元组；
    - 右外连接：只保存左边关系的悬浮元组；
15. 除运算：

> 给定关系$R(X,Y),S(Y,Z)$，设关系R除以关系S的结果为关系T，则T的定义为：
>
> $$R\div S=\{t_r[X]|t_r\in R\wedge \Pi_Y(S)\subset Y_x\},$$
> $$Y_x=\{t[Y]|t\in R,t[X]=x\}$$
>
> T包含了**所有在R但不在S中的属性及其值**，且**T的元组与S的元组的所有组合都在R中。**

    - 除运算经常用于求解"**查询...至少/全部的/所有的...**"问题。
    - 用基本操作表示：$R\div S=\Pi_X(R)-\Pi_X(R-\Pi_X(R)\times\Pi_Y(S))$。

Chap4 数据库安全性

- 数据库安全性概述
- 数据库安全性控制
- 视图机制
- 审计
- 数据加密
- 其他安全性保护

数据库安全性概述

1. 数据库安全性：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. 不安全因素：
    - **非授权用户**对数据库的恶意存取和破坏；
    - 数据库中重要或敏感的数据**被泄露**；
    - **安全环境**脆弱性。
3. 安全标准： TCSEC -> CTCPEC/FC/ITSEC/ -> CC。
4. TCSEC/TDI安全级别划分：$D<C1<C2<B1<B2<B3<A1$。

- $C2$级是**安全产品的最低档次。**

5. 等级保护/分级保护：
   - **非涉密**信息系统**等级保护**：用户自主保护级＜系统审计保护级＜安全标记保护级＜结构化保护级＜访问验证保护级。
     - 重要系统采用**第三、四级。**
   - **涉密**信息系统**分级保护**：秘密级＜机密级＜绝密级。

数据库安全性控制

6. 存取控制流程：**身份鉴别** -> **自主存取控制和强制存取控制**（/推理控制）-> **简单入侵检测。**
7. 用户身份鉴别：静态口令鉴别、动态口令鉴别、生物特征鉴别、智能卡鉴别、多因子认证。
8. 存取控制：
   - 机制组成：（1）定义用户权限；（2）合法权限检查。
   - 用户**权限定义**和**合法权限检查机制**一起组成了数据库管理系统的**存取控制子系统。**
   - 常用方法：
     - **自主存取控制（DAC）**：TCSEC/TDI $C2$ 级要求；
     - **强制存取控制（MAC）**：TCSEC/TDI $B1$ 级以上要求。
9. 自主存取控制方法：通过SQL的GRANT与REVOKE语句实现。
   - 定义的存取权限称为**授权。**
   - 只有系统超级用户才有权创建一个新的数据库用户。新建数据库用户有三种**角色**：（1）CONNECT；（2）RESOURCE；（3）DBA。
10. RBAC（Role-Based Access Control）安全原则：（1）最小权限原则；（2）责任分离原则；（3）数据抽象原则。
11. 强制存取控制方法：数据库管理系统所管理的全部实体被分为**主体**和**客体**两大类。
    - 主体是系统中的活动实体，客体是系统中的被动实体，受主体操纵。
12. 敏感度标记：**绝密（TS）>= 机密（S）>= 可信（C）>= 公开（P）。**
    - 对于主体称为许可证级别；对于客体称为密级。
    - 强制存取控制规则：
      - 下读：主体许可证级别不低于客体密级，才可进行读；
      - 上写：主体许可证级别不高于客体密集，才可进行写。
    - **标记与数据是不可分的整体。**

视图机制

13. 视图机制：把保密的数据对无权存取该数据的用户隐藏起来，对数据提供一定程度的安全保护。

审计

14. 审计：启用专门的审计日志，将用户对数据库的所有操作记录在上面。
    - $C2$以上安全级别的DBMS必须具有审计功能。
15. 涉密信息系统中的“三权分立”：系统管理员、安全保密员、安全审计管理员。

数据加密

16. 数据加密：包括存储加密和传输加密。
    - 存储加密：（1）透明存储加密（**内核级加密方式**）；（2）非透明存储加密。

- 传输加密：（1）链路加密；（2）端到端加密。

其他安全性保护

17. 其他安全性保护：推理控制、隐蔽信道、数据库脱敏、数据库防火墙等。

Chap5 数据库完整性

- 实体完整性
- 参照完整性
- 用户定义的完整性
- 完整性约束命名子句
- 断言
- 触发器

1. 数据库的完整性：（1）数据的**正确性**；（2）数据的**相容性。**
   - 为维护数据库的完整性，数据库管理系统必须：
     - 提供**定义**完整性约束的机制；
     - 提供完整性**检查**的方法；
     - **违约处理**；

实体完整性

2. 实体完整性检查和违约处理：
   - 检查**主码值是否唯一**，如果不唯一则拒绝插入或修改。
   - 检查**主码的各个属性是否为空**，只要有一个为空就拒绝插入或修改。

参照完整性

3. 可能破坏参照完整性的情况及违约处理：
   - **参照表插入元组**，若违约则**拒绝**；
   - **参照表修改外码值**，若违约则**拒绝**；
   - **被参照表删除元组**，若违约则**拒绝/级连删除/设置为空值**；
   - **被参照表修改主码值**，若违约则**拒绝/级连修改/设置为空值。**

用户定义的完整性

4. 完整性定义：
   - 定义**属性**上的约束条件：（1）列值非空（NOT NULL）；（2）列值唯一（UNIQUE）；（3）检查列值是否满足一个条件表达式（CHECK）。
   - 定义**元组**上的约束条件：设置不同属性之间取值的相互约束条件（CHECK）。
   - 违约处理：**拒绝执行。**

完整性约束命名子句

5. 完整性约束命名子句：CONSTRAINT<完整性约束条件名><完整性约束条件>
   - 修改：ALTER TBALE <表名> DROP/ADD CONSTRAINT <完整性约束条件名><完整性约束条件>。

断言

6. 断言：可以用于定义多个表的或聚集操作的较复杂的完整性约束。
    - 任何使断言**不为真值**的操作都会被**拒绝执行**。

触发器

7. 定义：

```
CREATE TRIGGER <触发器名>
{BEFORE|AFTER} <触发事件> [OF <列名>] ON <表名>
REFERENCING NEW|OLD ROW AS <变量名>
FOR EACH {ROW|STATEMENT}
[WHEN <触发条件>] <触发动作体>
```

    - 表的**拥有者**才可在表上创建触发器。触发器名和表名**必须在统一模式**下。触发器只能定义在基本表上，不能定义在视图上。
8. 触发器类型：（1）**行级**触发器（触发动作每行执行一次）；（2）**语句级**触发器（触发动作只执行一次）。
9. 触发动作体：可以是**匿名PL/SQL过程块**，也可以是对已创建**存储过程**的调用。

Chap7 数据库设计

- 数据库设计概述
- 需求分析
- 概念结构设计
- 逻辑结构设计
- 物理结构设计

数据库设计概述

1. 数据库设计：对于一个给定的应用环境，构造（分析、设计）**优化**的数据库**逻辑模式**和**物理结构**，并据此建立数据库及其应用系统，使之能够有效地存储和管理数据，满足各种用户应用需求，包括**信息管理要求**和**数据操作**要求。
2. 设计目标：为用户和各种应用系统提供一个**信息基础设施**和**高效**的运行环境。
3. 数据模型与概念模型：
    - **概念模型**：表达信息世界的模型，信息世界是对现实世界的理解与抽象；
    - **数据模型**：表达计算机世界的模型。
    - 现实世界 -> 信息世界（概念模型；概念层）-> 计算机世界（数据模型；逻辑层、物理层）。
4. 数据库设计基本步骤：需求分析 -> 概念结构设计 -> 逻辑结构设计 -> 物理结构设计 -> 数据库的实施和维护。
    - 需求分析和概念设计独立于数据库管理系统；
    - **逻辑设计**和**物理设计**与选用的数据库管理系统密切相关。

需求分析

5. 需求分析调查任务：（1）信息要求；（2）处理要求；（3）安全性与完整性要求。
6. 调查用户需求的步骤：调查组织机构情况 -> 调查各部门的业务活动情况 -> 协助用户明确对新系统的各种要求 -> 确定新系统的边界。
7. 数据字典：关于数据库中**数据的描述**，即元数据，不是数据本身。数据字典通过对数据项和数据结构的定义，来描述数据流、数据存储的逻辑内容。

- 内容：数据项、数据结构、数据流、数据存储、处理过程。
  - 数据项：不可再分的最小数据单位；
  - 数据结构：反应数据间的组合关系，由若干个数据项或数据结构组成；
  - 数据流：数据结构在系统内传输的路径；
  - 数据存储：数据结构停留或保存的地方，也是数据流来源和去向之一。

概念结构设计

8. E-R模型：世界由一组称作**实体**的基本对象和这些对象之间的**联系**构成的。
9. 概念结构设计：在深入理解需求的基础上，分析系统中存在的各种主要实体、属性和联系（关系）。
   - **主要实体**的发现、划分和定义；
   - **实体主要属性**的发现、分析和定义；
   - **实体之间联系**的发现、分析和定义。
     - **一对一、一对多、多对多。**

逻辑结构设计

10. 逻辑结构设计：将概念结构设计阶段的基本E-R图，转化为与选用数据库管理系统产品所支持的数据类型相符合的逻辑结构。
11. 数据模型的优化步骤：
    1. 确定数据依赖；
    2. 对各个关系模式之间的数据依赖进行极小化处理；
    3. 按照**数据依赖理论**对关系模式进行分析；
    4. 根据数据处理需求确定是否进行模式的合并与分解；
    5. 对关系模式进行必要分解，提高操作效率与空间利用率。

物理结构设计

12. 物理结构设计：为一个给定的逻辑数据模型选取一个**适合**应用要求的物理结构的过程，就是数据库的物理设计。
    - 物理结构：数据库在物理设备上的**存储结构**与**存取方法**称为数据库的物理结构，依赖于选定的数据库管理系统。
13. 数据库物理设计步骤：
    1. 建立数据库schema结构（**建库脚本**）；
    2. 确定数据库物理结构（存取方法和存储结构）；
    3. 对物理结构进行评价（**时间和空间效率**）；
    4. **必要的数据库反规范化冗余设计。**
14. 关系数据库物理设计内容：（1）为关系模式**选择存取方法（建立存取路径，即索引设计）**；（2）设计**关系**、**索引**等数据库文件的**物理存储结构。**
15. 数据库管理系统**常用存取方法**：（1）**B+树索引存取方法**；（2）**Hash索引存取方法**；（3）**聚簇存取方法。**
    - B+树索引选择一般规则：查询条件、聚集函数参数、连接条件、排序；
    - Hash索引选择一般规则：等值连接、等值条件。
16. 评价指标：（1）存储空间；（2）存取时间；（3）维护代价。

Chap8 数据库编程

- 嵌入式SQL
- 过程化SQL
- 存储过程和函数
- JDBC编程
- SQL注入

嵌入式SQL

1. 嵌入式SQL：将SQL语句嵌入程序设计语言（C/C++，Java，称为(宿)主语言）中。
    - 处理过程：**预编译。**
2. 嵌入式SQL与主语言之间的通信：
    1. 向主语言传递SQL语句**执行状态信息**，使主语言能够**控制程序流程**（通过**SQLCA**实现）；
    2. 主语言向SQL语句提供参数（通过**主变量**实现）；
    3. 将SQL查询结果交给主语言处理（通过**主变量和游标**实现）。

过程化SQL

3. 过程化SQL：对SQL的扩展，使其增加了**过程化语句功能。**
    - 基本结构：**块。**块之间可以相互嵌套，每个块完成一个逻辑操作。
    - 块结构：（1）定义部分；（2）执行部分[条件控制/循环控制/异常处理部分]；
4. 变量、常量定义：**只能在基本块中使用。**当基本块执行结束，定义就不复存在。
5. 命名块与匿名块：
    - **命名块**：编译后保存在数据库中，可反复调用，运行速度快（如**存储过程（procedure）和函数（function）**）；
    - **匿名块**：每次执行时都需要编译，不会存储在数据库中，也不能在其他过程化SQL中调用。

存储过程和函数

6. 存储过程：由过程化SQL语句书写，经编译和优化后存储在数据库服务器中，使用时只要调用即可。
    - 优点：（1）效率高；（2）降低了客户机与服务器间的通信量；（3）方便实施业务规则。
    - 用户接口：（1）创建；（2）执行；（3）修改；（4）删除。
7. 函数与存储过程的异同：
    - 相同点：都是**持久性存储模块**；
    - 不同点：**函数必须指定返回类型。**

JDBC编程

8. 主流数据库访问编程工具：**ODBC（Open Database Connectivity）、JDBC（Java Database Connectivity）。**
9. JDBC：JDBC API提供了一套**访问**任何**RDBMS**的**标准接口。**

SQL注入

10. 预防：**必须使用参数化的传递方式，调用**`setXXX()`**函数进行变量绑定。**
    - 根本解决方案：**检查用户输入，转义特殊字符。**

Chap9 查询优化

- 关系数据库系统的查询处理
- 关系数据库系统的查询优化
- 代数优化
- 物理优化

关系数据库系统的查询处理

1. 查询处理：将用户提交的查询语句转化为高效的**查询执行计划。**
    - 阶段：**查询分析、查询检查、查询优化、查询执行。**
2. 查询分析：词法分析与语法分析。
3. 查询检查：合法性检查、试图转换、**安全性检查（DAC/MAC定义的权限）**、完整性初步检查。
    - 一般使用**查询树（/语法分析树）**来表示扩展的关系代数表达式。
4. 查询优化：**代数优化**和**物理优化。**
    - 选择依据：基于规则/代价/语义。
5. 查询执行：由**代码生成器**生成依据优化器得到的执行计划的代码，加以执行并返回结果。
6. 选择操作的实现：（1）**全表扫描方法**；（2）**索引扫描方法。**
7. 连接操作的实现：（1）**嵌套循环算法**；（2）**排序-合并算法**；（3）**索引连接算法**；（4）**Hash Join算法。**
    - 适用条件：（1）适于**任何条件**；（2）仅适于**不等于**（'<>'）之外的条件；（3）是（1）的变种；（4）仅适于**等值连接。**

关系数据库系统的查询优化

8. 查询优化：通过某种**代价模型**计算出各种**查询执行策略**的**执行代价**，然后选择代价最小的执行方案。
    - 数据库执行代价：（1）**集中式数据库**主要考虑**I/O代价**；（2）**分布式数据库**则需要考虑**I/O**、**CPU**、**内存**及**通信**代价。

代数优化

9. 代数优化：**通过关系代数表达式的等价变换来提高查询效率。**
10. 选择与笛卡尔积的**交换律**：
    - $\sigma_F(E_1 \times E_2) = \sigma_F(E_1) \times E_2$：$F$中涉及的属性都是$E_1$中的属性；
    - $\sigma_F(E_1 \times E_2) = \sigma_{F_1}(E_1) \times \sigma_{F_2}(E_2)$：$F = F_1 \wedge F_2$，且$F_i$仅涉及$E_i$的属性；
    - $\sigma_F(E_1 \times E_2) = \sigma_{F_2}(\sigma_{F_1}(E_1) \times E_2)$：$F_1$仅涉及$E_1$中的属性，$F_2$涉及$E_1, E_2$的属性；
        - 使得**部分选择操作可在笛卡尔积前做。**
11. **查询树的启发式优化**：
    - **选择运算应尽可能先做**；
    - 把**投影运算和选择运算同时进行**；
    - 把**投影和其前或后的双目运算结合起来**；
    - 把某些**选择**同在它之前执行的**笛卡尔积**结合起来成为一个**连接运算**；
    - **找出公共子表达式。**

物理优化

12. 物理优化：**选择高效合理的操作算法或存取路径**，求得优化的查询计划。
13. 物理优化方法：（1）**基于规则的启发式优化**；（2）**基于代价估算的优化**；（3）二者结合。

Chap10 事务与数据恢复

- 事务的基本概念
- 数据库恢复概述
- 故障的种类
- 恢复的实现技术
- 恢复策略
- 具有检查点的恢复技术
- 数据库镜像

事务的基本概念

1. **事务**：用户定义的一个数据库操作序列，是一个**不可分割的工作单位。**
    - 事务与程序是两个概念。**事务是恢复和并发控制的基本单位。**
    - COMMIT/ROLLBACK。
2. 事务的**ACID**特性：（1）**原子性（A）**；（2）**一致性（C）**；（3）**隔离性（I）**；（4）**持续性（D）**。

数据库恢复概述

3. 数据库的恢复：由于**故障不可避免**，因此数据库管理系统必须具有**把数据库从错误状态恢复到某一已知的正确状态**（或称一致状态/完整状态）的功能，即数据库的恢复管理系统对故障的对策。

故障的种类

4. 故障的种类：（1）**事务内部故障**；（2）**系统故障**；（3）**介质故障**；（4）**计算机病毒。**
5. **事务内部故障**：如运算溢出、并发事务死锁、违反某些完整性规则等。
    - 恢复：**事务撤销（UNDO）**，即强行回滚（ROLLBACK）该事务。
6. **系统故障**（/软故障）：如系统断电、特定硬件错误、操作系统故障、数据库管理系统代码错误等。
    - 恢复：**强行撤销（UNDO）未完成事务、重做（REDO）已提交事务。**
7. **介质故障**（/硬故障）：如磁盘损坏、磁头碰撞、瞬时强磁场干扰等。
8. 恢复操作基本原理：**冗余。**

恢复的实现技术

9. 建立数据冗余：（1）**数据转储**；（2）**登记日志文件。**
10. 转储：DBA定期地将整个数据库文件复制到磁带、磁盘或其他存储介质上保存起来的过程。
    - **后备副本/后援副本**：即备用的数据文件。
    - 转储方法分类：（1）**静态/动态**转储；（2）**海量/增量**转储。
11. 静态/动态转储：
    - 静态转储：系统中**无运行事务时**进行的转储；
    - 动态转储：**与用户事务并发**的转储。
        - 利用动态转储副本进行故障恢复：需要建立**日志文件。**
12. 海量/增量转储：
    - 海量（/全量）转储：每次转储**全部数据库**；
    - 增量转储：每次只转储上次转储后**更新过的数据。**

13. 登记日志文件：记录事务对数据的更新操作的文件。
    - 格式：（1）以**记录**为单位；（2）以**数据块**为单位。
    - 用途：（1）进行**事务故障**恢复；（2）进行**系统故障**恢复；（3）协助后备副本进行介质故障恢复。
    - 原则：
        - 登记次序严格按照并发事务执行的时间次序。
        - **先写日志文件，后写数据库。**

恢复策略

14. 事务故障恢复步骤：
    1. **反向扫描日志文件**，找到故障事务的更新操作；
    2. 对该事务的更新操作**执行逆操作**；
    3. 继续反向扫描，寻找该事务其他更新操作，做同样处理；
    4. 如此处理，直至读到事务开始标记。
15. 系统故障恢复步骤：
    1. **正向扫描日志文件**，将故障前已提交的事务加入**重做队列**，将故障时未完成的事务加入**撤销队列**；
    2. 对撤销队列中的事务进行撤销（UNDO）处理；
        - **反向**扫描日志文件，对撤销事务每个更新操作执行逆操作。
    3. 对重做队列中的事务进行重做（REDO）处理。
        - **正向**扫描日志文件，对重做事务重新执行已登记的操作。
16. 介质故障恢复步骤：
    1. **重装数据库**，使之恢复到最近一次转储时的一致性状态；
    2. **装入有关日志文件副本**，重做（REDO）已完成的事务。
    - 介质故障恢复需要**数据库管理员介入。**

具有检查点的恢复技术

17. 检查点记录的内容：**检查点时刻**所有正在执行的事务清单，及这些事务最近的一个日志记录的地址。
    - 重新开始文件的内容：各个检查点记录在日志文件中的地址。
18. 动态维护日志文件的方法：**周期性**地**建立检查点**，**保存数据库状态。**
    1. 将当前**日志缓冲区**中的所有**日志记录**写入**磁盘**日志文件；
    2. 在日志文件中写入一个**检查点记录**；
    3. 将当前**数据缓冲区**中的所有**数据记录**写入**磁盘**数据库；
    4. 把记录点记录在日志文件中的**地址**写入一个**重新开始文件。**

数据库镜像

19. 数据库镜像：主从复制。用于有效地恢复**介质故障。**

Chap11 并发控制

- 并发控制概述
- 封锁
- 封锁协议

- 活锁和死锁
- 并发调度的可串行性
- 两段锁协议
- 封锁的粒度

并发控制概述

1. 并发控制：为了**保证事务的隔离性和一致性**，数据库管理系统需要对并发操作进行正确调度。
2. 数据不一致性：（1）**丢失修改**；（2）**不可重复读**；（3）**读“脏”数据。**
    - 丢失修改：事务T_1执行**Update**，事务T_2执行**Update**；
    - 不可重复读：事务T_1执行**Read**，事务T_2执行**Update/Delete/Insert**；
    - 读“脏”数据：事务T_1执行**Rollback Update**，事务T_2执行**Read。**
3. 主要技术：（1）**封锁**；（2）时间戳；（3）乐观控制法；（4）多版本并发控制。

封锁

4. 基本封锁类型：（1）**排他锁（写锁）X锁**；（2）**共享锁（读锁）S锁**；

封锁协议

5. 三级封锁协议：
    - 一级封锁协议：事务T在**修改数据**R前**必须先对其加X锁**，**直到事务结束**才释放X锁。
        - **读数据不需要加锁**；
        - **仅能解决丢失修改问题。**
    - 二级封锁协议：在一级封锁协议基础上，事务T在**读取数据**R前**必须先对其加S锁**，**读完后**即可释放S锁。
        - **可防止丢失修改和读“脏”数据**；
        - 由于**读完即释放S锁**，因此**不能保证可重复读。**
    - 三级封锁协议：在一级封锁协议基础上，事务T在**读取数据**R前**必须先对其加S锁**，**直到事务结束**才释放S锁。
        - **可防止丢失修改、不可重复读和读“脏”数据。**

活锁和死锁

6. 避免活锁：采取**先来先服务**策略。
7. 死锁：
    - 产生条件：（1）请求互斥的共享资源；（2）非抢占；（3）持续占有一个资源同时请求另一资源；（4）循环等待。
    - 预防：（1）**一次封锁法**；（2）**顺序封锁法**；
    - 诊断：（1）**超时法**；（3）**等待图法**；
    - 解除：**选择一个处理死锁代价最小的事务**，将其撤消。

并发调度的可串行性

8. 可串行化调度：多个事务的并发执行时正确的，当且仅当其结果与**按某一次序串行地执行这些事务时的结果相同**，称这种调度策略为**可串行化调度。**
    - **可串行性**是并发事务正确调度的准测，一个并发调度是正确调度，当且仅当它是可串行化的。
9. 冲突：调度中一对连续的操作，若交换两者顺序，则设计的事务中至少有一个事务的行为会改变，则称这一对动作是**冲突**的。

- 两个操作**可交换**当且仅当两个操作是**无冲突**的。
- 可交换的操作：
  - **不同事务对同一元素的两个读操作**；
  - **不同事务对不同元素的任意两个操作。**
- 不可交换的操作：
  - **同一事务的任意两个操作**；
  - **不同事务对同一元素的两个写操作**；
  - **不同事务对同一元素的一读一写操作。**

10. 冲突可串行化：一个调度Sc在**保证冲突操作的次序不变**的情况下，通过**交换两事务不冲突操作的次序**得到另一个调度Sc'，若Sc'是串行的，则称Sc是**冲突可串行化**的。
    - 冲突可串行化是可串行化的**充分非必要条件。**

两段锁协议

11. 两段锁协议：指所有事务必须**分两个阶段对数据项进行加锁和解锁。**
    - 通过**两段所协议**可以实现**并发调度可串行性**；
    - 事务遵循两段锁协议是可串行化调度的**充分非必要条件。**
    - 两个阶段：（1）**扩展阶段（获得封锁）**；（2）**收缩阶段（释放封锁）。**
12. 两段锁协议与死锁：
    - 预防死锁的**一次封锁法**遵循两段锁协议；
    - **遵循两段锁协议的事务可能发生死锁。**

封锁的粒度

13. 封锁粒度：封锁对象的大小。
    - 封锁对象：（1）**逻辑单元**（如属性值、集合、元组等）；（2）**物理单元**（如页、记录等）。
14. 封锁粒度、开销与并发度：封锁**粒度**越大，并发度**越小**，系统**开销**越小。
15. 多粒度树：以**树形结构表示多级封锁粒度。**根结点为整个数据库，表示最大的数据粒度。叶结点表示最小的数据粒度（如元组）。
16. 多粒度封锁协议：允许多粒度树中的每个结点被独立地加锁。
    - **显示加锁**：直接加到数据对象上的封锁；
    - **隐式加锁**：由于上级结点加锁而使该对象上锁。
17. 意向锁：对一个结点加意向锁，则说明该结点的**下层结点**正在被加锁。
    - 避免了低效的隐式锁检查。
    - 分类：（1）**IS锁**（后裔结点拟加S锁）；（2）**IX锁**（后裔结点拟加X锁）；（3）**SIX锁**（加S锁及IX锁）。
18. 锁相容矩阵：

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

19. 锁强度：**X>SIX>S=IX>IS。**