数据库 01

2025 年 3 月 18 日

考纲内容：

- **绪论**
  数据库的 4 个基本概念，数据管理技术的产生和发展，数据建模、概念模型和数据模型的三要素，数据库系统的三级模式结构，数据库的两级映像与数据独立性，数据库系统的组成。
- **关系模型**
  关系模型的数据结构及形式化定义，关系操作，关系完整性，关系代数（传统的集合运算、专门的关系运算）。
- **关系数据库标准语言 SQL**
  数据定义、数据查询、数据更新、空值处理、视图。ss
- **数据库安全性**
  数据库安全性概述，数据库安全性控制。

1 绪论

1.1 数据库的基本概念

数据库系统的四个基本概念：

1. 数据 (Data)：描述事物的符号记录，是数据库中存储的基本对象。
2. 数据库 (Database, DB)：长期存储在计算机内、有组织的、可共享的大量数据的集合。数据库中的数据具有如下特点：
   - 永久存储
   - 有组织
   - 可共享
3. 数据库管理系统 (Database Management System, DBMS)：位于用户与操作系统之间的一层数据管理软件，是基础软件，是一个大型复杂的软件系统。
4. 数据库系统 (Database System, DBS)：由数据库、数据库管理系统（及其应用开发工具）、应用程序和数据库管理员（DBA）组成的存储、管理、处理和维护数据的系统。

1.2 数据管理技术的产生和发展

数据管理技术的发展经历了三个阶段：

1. **人工管理阶段**（20 世纪 50 年代中期以前）

- 数据不保存
- 应用程序管理数据
- 缺点：数据不可共享、数据冗余度大、数据不一致性

2. **文件系统阶段**（20 世纪 50 年代中期至 60 年代中期）
   - 数据可长期保存
   - 文件系统实现对数据的管理
   - 缺点：数据共享性差、数据冗余大、数据独立性差
3. **数据库系统阶段**（20 世纪 60 年代末至今）
   - 数据结构化
   - 数据共享性高，冗余度低
   - 数据独立性高
   - 由 DBMS 统一管理和控制

1.3 数据建模与数据模型

1.3.1 数据建模

数据建模是抽象、表示和处理现实世界中数据的方法和过程。

1.3.2 概念模型

概念模型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。
最常用的概念模型是实体-联系模型 (E-R 模型)，它由下列要素组成：

- 实体 (Entity)：客观存在并可相互区别的事物
- 属性 (Attribute)：实体所具有的某一特性
- 联系 (Relationship)：实体之间的关联

图 1: E-R 图示例：学生-课程关系

1.3.3 数据模型的三要素

数据模型是对现实世界数据特征的抽象，由三部分组成：

1. **数据结构**：描述数据库的组成对象及对象间的联系
2. **数据操作**：对数据库中各种对象实例允许执行的操作及操作规则
3. **数据约束**：保证数据库中数据满足特定语义规则的条件

按照抽象级别，数据模型可分为：

- **概念模型**：面向用户，如 E-R 模型
- **逻辑模型**：面向 DBMS，如层次模型、网状模型、关系模型、面向对象模型等
- **物理模型**：面向存储，描述数据在存储介质上的实际组织方式

1.4 数据库系统的三级模式结构

ANSI/SPARC 提出的三级模式结构包括：

1. 外模式 (External Schema)：也称为用户模式，是用户与数据库系统的接口，由若干外部视图组成。
2. 模式 (Schema)：也称为概念模式，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。
3. 内模式 (Internal Schema)：也称为存储模式，是数据物理结构和存储方式的描述，是数据在存储介质上的表示方式和存取方法。

注：此处三种模式，其实对应用户，软件开发者和数据库开发人员。

1.5 数据库的两级映像与数据独立性

1.5.1 两级映像

1. 外模式/模式映像：定义外模式与模式之间的对应关系，当模式改变时，对应的外模式/模式映像也需要改变。
2. 模式/内模式映像：定义模式与内模式之间的对应关系，当内模式改变时，对应的模式/内模式映像也需要改变。

1.5.2 数据独立性

1. 物理数据独立性：当数据库的内模式改变时，只需要修改模式/内模式映像，使模式保持不变，应用程序不受影响。
2. 逻辑数据独立性：当数据库的模式改变时，只需要修改外模式/模式映像，使外模式保持不变，应用程序不受影响。

数据独立性是数据库系统的重要特征，它保证了应用程序和数据库结构的相对独立，从而提高了数据库系统的可维护性和扩展性。

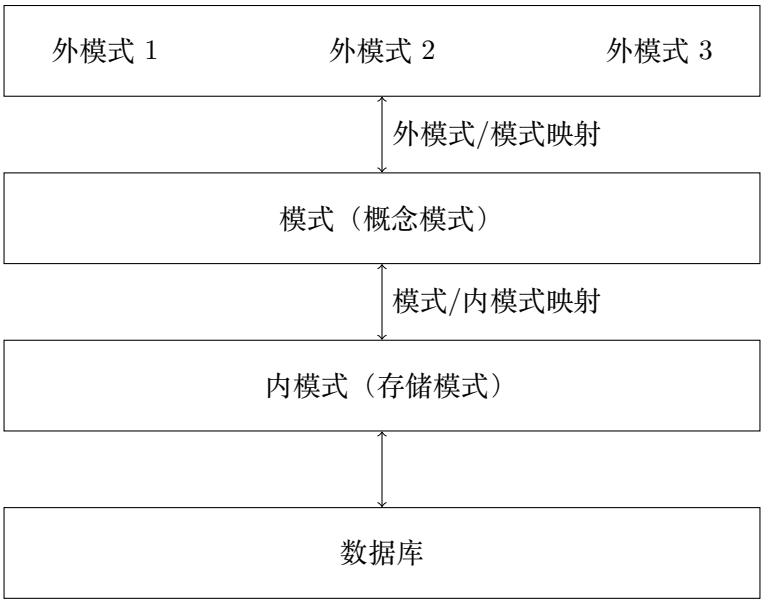


图 2: 数据库系统的三级模式结构

1.6 数据库系统的组成

数据库系统由以下几部分组成：

1. **硬件平台**：计算机、存储设备和网络设备等
2. **数据库**：存储在计算机中的数据集合
3. **DBMS**：管理数据库的软件
4. **应用程序**：为用户提供操作界面的程序
5. **数据库管理员 (DBA)**：负责数据库的规划、设计、维护和管理
6. **用户**：使用数据库的人，包括最终用户、应用程序员和 DBA

1.7 小结

- 数据库系统的四个基本概念：数据、数据库、数据库管理系统和数据库系统
- 数据管理技术的发展经历了人工管理、文件系统和数据库系统三个阶段
- 数据模型由数据结构、数据操作和数据约束三要素组成
- 数据库系统采用三级模式结构：外模式、模式和内模式
- 数据库的两级映像支持物理数据独立性和逻辑数据独立性
- 数据库系统由硬件、软件、数据、人员等组成部分构成

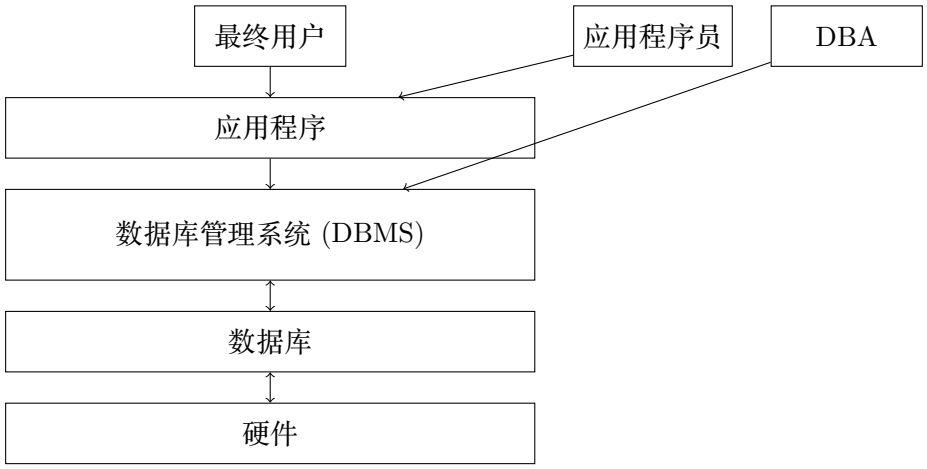


图 3: 数据库系统的组成

2 关系模型

2.1 关系模型的数据结构

关系模型是目前最重要的数据库模型，由 E.F.Codd 于 1970 年首先提出。关系模型的基本数据结构非常简单，就是关系，即二维表格结构。

2.1.1 关系的形式化定义

设有 n 个域 $D_1, D_2, \ldots, D_n$，它们的笛卡尔积为：

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

关系 (Relation)是笛卡尔积 $D_1 \times D_2 \times \ldots \times D_n$ 的子集，表示为 $R(A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n)$，其中：

- R 是关系名
- A_i 是属性名
- D_i 是域（属性的取值范围）
- n 是关系的目或度（Degree），表示关系的属性个数
- 关系中的每个元组（Tuple）对应表中的一行
- 关系模式（Relation Schema）：$R(A_1, A_2, \ldots, A_n)$

2.1.2 关系的性质

1. 列是同质的：每一列中的数据来自同一个域，是同一类型的数据

2. 不同的列可来自同一个域：不同属性可对应相同的域
3. 列的顺序无关紧要：列的次序可以任意交换
4. 行的顺序无关紧要：行的次序可以任意交换
5. 行列确定唯一的值：给定行号和列名后，表中的值唯一确定
6. 不允许表中有重复的行（元组）：任意两个元组在至少一个属性上取值不同
7. 每个分量必须是不可分的数据项：不允许表中的表（非规范化）

Example 1. 某大学学生关系 *Student* 的一个实例：

Sno	***Sname***	***Ssex***	***Sage***
201901	李勇	男	*20*
201902	刘晨	女	*19*
201903	王敏	女	*18*

关系模式：*Student(Sno, Sname, Ssex, Sage)*

2.1.3 关系模型中的基本概念

1. 候选键（Candidate Key）：能唯一标识关系中元组的最小属性集合。
2. 主键（Primary Key）：从候选键中选定的一个，用于唯一标识关系中的元组。
3. 外键（Foreign Key）：关系 R 的一个属性（或属性集），它不是 R 的主键，但是它在另一个关系 S 中是主键。
4. 主属性（Prime Attribute）：包含在任何一个候选键中的属性。
5. 非主属性（Non-prime Attribute）：不包含在任何候选键中的属性。

2.2 关系操作

关系模型的操作主要分为查询和更新两类。

2.2.1 查询操作

查询操作是关系数据库中最基本的操作，主要包括：

- 选择（Selection）：从关系中选取满足条件的元组
- 投影（Projection）：从关系中选取指定的列
- 连接（Join）：将两个关系按照共同属性组合成一个关系
- 除法（Division）：A ÷ B，求 A 中满足 B 中所有条件的元组
- 并（Union）：两个关系的并集

- 差（Difference）：两个关系的差集
- 交（Intersection）：两个关系的交集
- 笛卡尔积（Cartesian Product）：两个关系的所有可能组合

2.2.2 更新操作

更新操作包括：

- 插入（Insert）：向关系中添加元组
- 删除（Delete）：从关系中删除元组
- 修改（Update）：修改关系中的元组

2.3 关系完整性

关系模型中的完整性约束是保证数据库中数据正确性、有效性和相容性的规则，主要包括：

2.3.1 实体完整性（Entity Integrity）

实体完整性规则：关系的主键属性值不能为空（NULL）。
这保证了每个实体（即关系中的每一行）都能被唯一标识。

2.3.2 参照完整性（Referential Integrity）

参照完整性规则：如果关系 R 的外键 F 是关系 S 的主键，则关系 R 中每个元组在 F 上的取值必须是：

- 要么等于关系 S 中某个元组的主键值
- 要么为空值（如果允许外键取空值）

2.3.3 用户定义的完整性（User-defined Integrity）

用户定义的完整性是针对具体应用的约束条件，例如：

- 属性值的范围约束（例如年龄必须大于 0 且小于 120）
- 属性间的相互约束（例如入职日期必须晚于出生日期）

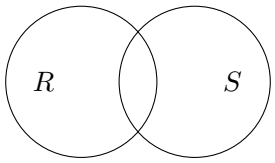
2.4 关系代数

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。

2.4.1 传统的集合运算

1. 并（Union）：$R \cup S = \{t | t \in R \vee t \in S\}$
2. 差（Difference）：$R - S = \{t | t \in R \wedge t \notin S\}$
3. 交（Intersection）：$R \cap S = \{t | t \in R \wedge t \in S\}$
4. 笛卡尔积（Cartesian Product）：$R \times S = \{(r, s) | r \in R \wedge s \in S\}$

需要注意的是，进行并、差、交运算的两个关系必须是同元（union-compatible）的，即它们必须具有相同的目（属性数）且对应的属性来自相同的域。



$R \cup S$: 两个圆的全部区域
$R \cap S$: 两个圆的交叉区域
$R - S$: 仅在 R 中的区域

图 4: 集合运算示意图

2.4.2 专门的关系运算

1. 选择（Selection）：$\sigma_F(R) = \{t | t \in R \wedge F(t) = \text{true}\}$

   选择操作是从关系 R 中选取满足给定条件 F 的元组。

2. 投影（Projection）：$\Pi_A(R) = \{t[A] | t \in R\}$

   投影操作是从关系 R 中选取指定的属性 A 组成新的关系。

3. 连接（Join）：

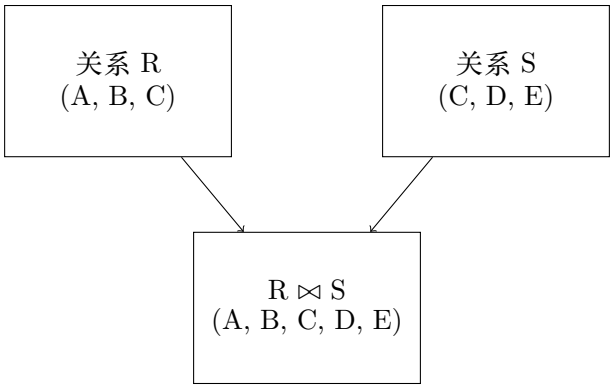
   - 自然连接（Natural Join）：$R \bowtie S = \{rs | r \in R \wedge s \in S \wedge r[A] = s[A]\}$，其中 A 是 R 和 S 共有的属性集合。
   - θ-连接（Theta Join）：$R \bowtie_\theta S = \{rs | r \in R \wedge s \in S \wedge \theta(r, s)\}$
   - 外连接（Outer Join）：保留在连接中无匹配的元组，缺少的属性值用 NULL 填充。

4. 除法（Division）：$R \div S = \{t[X] | t \in R \wedge \forall s \in S, ts \in R\}$

   除法操作用于回答” 对于 S 中的所有值，找出 R 中与之相关的所有值” 类型的查询。

Example 2. 考虑以下两个关系：

学生关系 *Student(Sno, Sname, Sage, Sdept)*
选课关系 *SC(Sno, Cno, Grade)*



自然连接：基于共同属性 C

图 5: 自然连接示例

以下关系代数表达式表示”查询所有选修了*'C1'* 课程的学生姓名”:
$\Pi_{Sname}(\sigma_{Cno='C1'}(Student \bowtie SC))$
执行过程：

1. 首先，*Student* 和 *SC* 进行自然连接（按 *Sno* 属性）
2. 然后，选择 *Cno='C1'* 的元组
3. 最后，投影出 *Sname* 属性

2.5 扩展的关系代数操作

除了基本的关系代数操作外，还有一些扩展操作：

1. 广义投影（Generalized Projection）：允许在投影列表中包含计算表达式
2. 聚集（Aggregation）：包括 COUNT、SUM、AVG、MAX、MIN 等聚合函数
3. 外连接（Outer Join）：保留在连接中无匹配的元组
4. 半连接（Semi-join）：$R \ltimes S = \Pi_R(R \bowtie S)$

2.6 小结

- 关系模型以简单的二维表格形式表示数据，具有形式化的数学基础
- 关系的基本特性包括列的同质性、行列无序性、元组的唯一性等
- 关系完整性约束包括实体完整性、参照完整性和用户定义的完整性
- 关系代数提供了一套形式化的操作，分为传统的集合运算和专门的关系运算
- 关系代数是关系数据库查询语言的理论基础，SQL 语言实现了关系代数的大部分功能