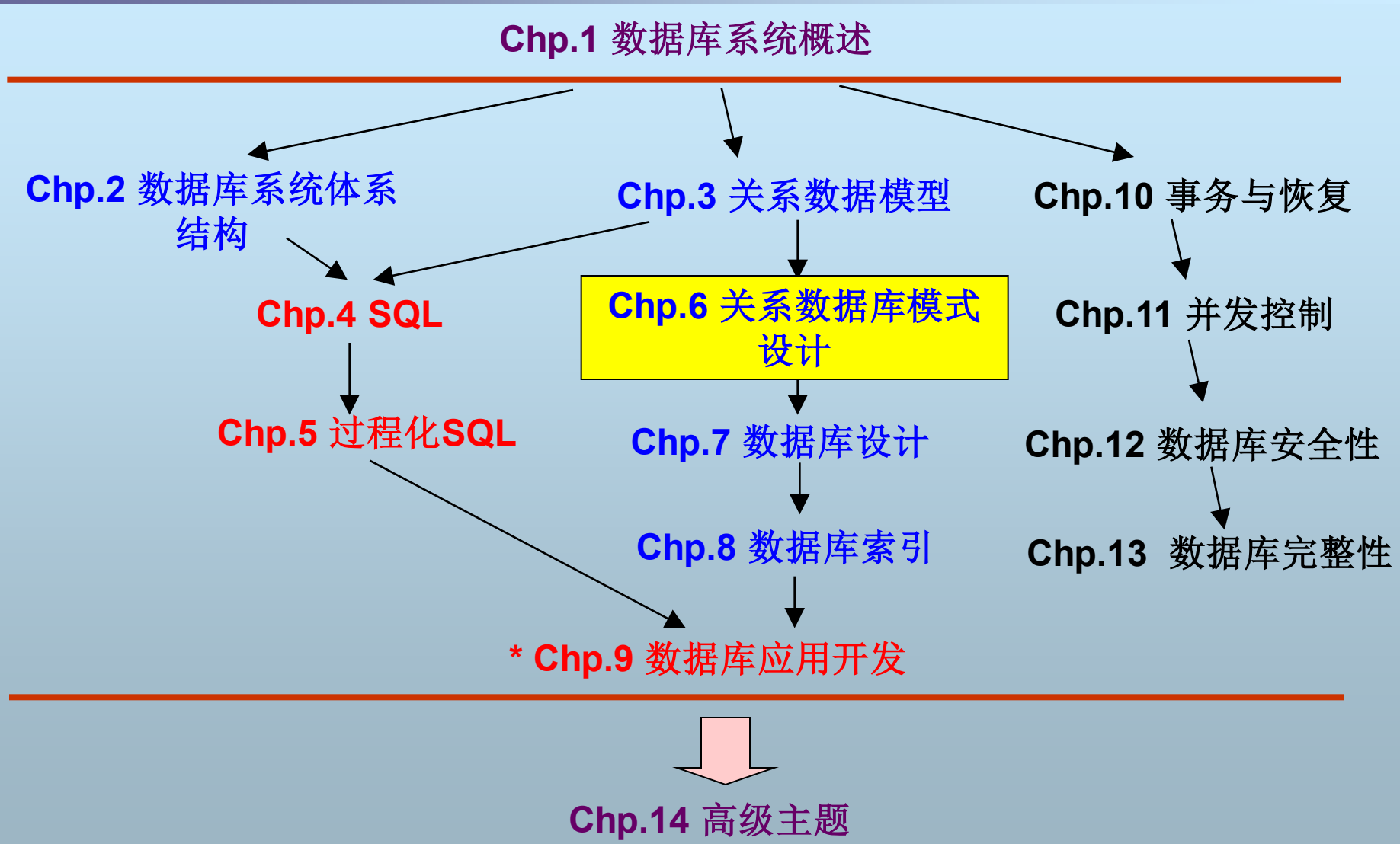
第6章 关系数据库模式设计

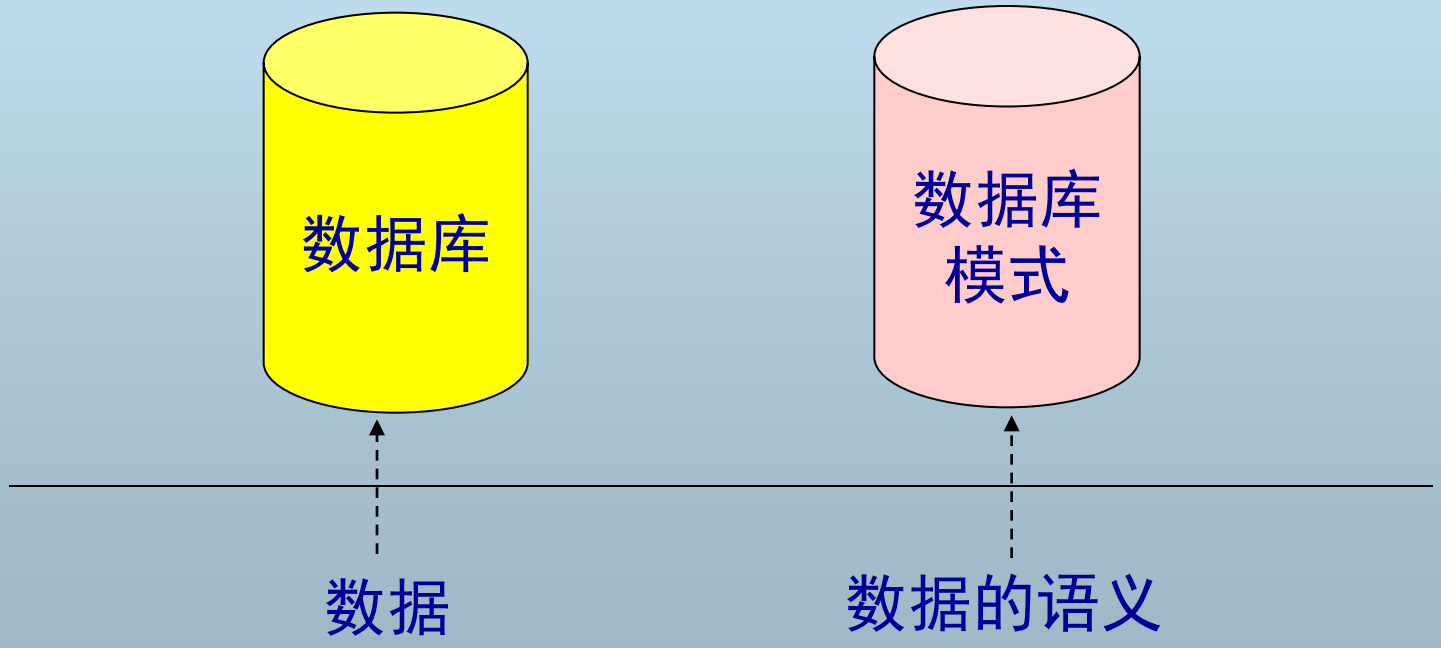
金培权（jpq@ustc.edu.cn）

课程知识结构


Chp.1 数据库系统概述
Chp.2 数据库系统体系结构
Chp.3 关系数据模型
Chp.10 事务与恢复
Chp.4 SQL
Chp.6 关系数据库模式设计
Chp.11 并发控制
Chp.5 过程化SQL
Chp.7 数据库设计
Chp.12 数据库安全性
Chp.8 数据库索引
Chp.13 数据库完整性
* Chp.9 数据库应用开发
Chp.14 高级主题

数据库模式

- 数据库模式是数据库中全体数据的逻辑结构和特征的描述



举例

学号	姓名	年龄
001	张三	20
002	李四	21
003	王五	22

学生(学号:char, 姓名:char, 年龄:int)

模式

数据库

问题的提出

- 如何把现实世界表达成数据库模式？
- 针对一个具体应用，应该如何构造一个适合于它的数据库模式？

- 这是数据库的逻辑设计问题——关系数据库的模式设计理论是数据库逻辑设计的理论基础

本章主要内容

- 关系模式的设计问题
- 函数依赖
- 关系模式的分解
- 关系模式的范式

一、关系模式的设计问题

- 关系模式设计不规范会带来一系列的问题
  - 数据冗余
  - 更新异常
  - 插入异常
  - 删除异常

示例关系模式R

示例关系模式 R(Tname, Addr, C#, Cname)

一个教师只有一个地址和一个联系电话

一个教师可教多门课程

一门课程只有一个任课教师

因此R的主码是（C#）

Tname	Addr	Tel	C#	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

1、问题（1）：数据冗余

- 教师T1教了三门课程，他的地址和电话被重复存储了2次

Tname	Addr	Tel	C#	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

2、问题（2）：更新异常

- 如果T1的地址变了，则需要改变3个元组的地址；若有一个未更改，就会出现数据不一致。但DBMS无法获知这种不一致

Tname	Addr	Tel	C#	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

3、问题（3）：插入异常

- 如果要增加一名教师，但他还未带课，则C#和Cname为空，但由于C＃是主码，为空违反了实体完整性，所以这名教师将无法插入到数据库中

Tname	Addr	Tel	C#	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

4、问题（4）：删除异常

- 如果教师T3现在不带课了，则需将T3的元组删去，但同时也把他的姓名和地址电话信息删掉了

Tname	Addr	Tel	C#	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

5、如何解决？

- 方法：模式分解
  - 方法1：R分解为
    - R1(Tname, Addr, Tel)
    - R2(C#,Cname)
  - 方法2
    - R1(Tname, Addr, Tel, C#)
    - R2(C#, Cname)
  - 方法3
    - R1(Tname, Addr, Tel)
    - R2(Tname ,C#, Cname)

授课信息丢失了

R1中问题依然存在

基本解决问题，但又带来联接查询代价

5、如何解决？

- 到底什么样的模式才最佳？怎么分解才能达到要求？标准是什么？如何实现？

二、函数依赖

- 什么是函数依赖
- 码的定义
- 最小函数依赖集

回顾：关系模式的形式化定义：

R（U，D，dom，F）

R为关系模式名，U是一个属性集，D是U中属性的值所来自的域，Dom是属性向域的映射集合，F是属性间的依赖关系

1、什么是函数依赖？

- 函数依赖是指一个关系模式中一个属性集和另一个属性集间的多对一关系
  - 例如选课关系SC(S#, C#, Score)
  - 存在由属性集{S#, C#}到属性集{Score}的函数依赖
    - 对于任意给定的S#值和C#值，只有一个Score值与其对应
    - 反过来，可以存在多个S#值和C#值，它们对应的Score值相等

2、基本概念

- 函数依赖（FD，Functional Dependency）的形式化定义
  - 设关系模式R(A1,A2,...,An)或简记为R(U)，X和Y是U的子集。r是R的任意一个实例（关系），若r的任意两个元组t1、t2，由t1[X]=t2[X]可导致t1[Y]=t2[Y]，即如果X相等则Y也相等，则称Y函数依赖于X或称为X函数决定Y，记作 X→Y
    - 即R的X属性集上的值可唯一决定R的Y属性集上的值
    - 也即对于R的任意两个元组，X上的值相等，则Y上的值也必相等
  - FD是相对于关系模式而言的，因此关系模式R的所有实例都要满足FD

2、基本概念

- 例如
  - Student关系模式中，{S#} →{Sname} （单个属性可去掉括号，简写成 S# →Sname ）
  - SC关系模式中， {S#,C#} →{Score}
- FD是否成立，唯一办法是仔细考察应用中属性的含义。FD实际上是对现实世界的断言。数据库设计者在设计时把应遵守的函数依赖通知DBMS，则DBMS会自动检查关系的合法性
  - 对于关系模式 R(Tname, Addr, C#, Cname)
    - 若一门课只能有一个教师，则有{C#} →{Tname}
    - 若一门课可有多个教师任教，则{C#} →{Tname}不成立
    - 因此FD是与具体应用相关的

2、基本概念

- 关系模式的形式化定义：
  - R（U，D，Dom，F）
    - R为关系模式名，U是一个属性集，D是U中属性的值所来自的域，Dom是属性向域的映射集合，F是属性间的依赖关系
  - FD是关系模式的一部分

3、平凡FD和不平凡FD

- 模式设计的首要问题是确定关系模式的最小函数依赖集
  - 给定一个函数依赖集S，若能找到一个远小于S的函数依赖集T，则DBMS只要实现T就可实现S中的所有函数依赖
- 平凡FD和不平凡FD
  - $X \rightarrow Y$，且$Y \subseteq X$，则$X \rightarrow Y$是平凡FD，否则是不平凡FD
- 平凡FD没有什么实际意义，消除平凡FD是缩小函数依赖集大小的一个简单方法

4、函数依赖集的闭包

- 函数依赖的逻辑蕴含
  - 设F是关系模式R的一个函数依赖集，X和Y是R的属性子集，若从F的函数依赖中能推出X→Y，则称F逻辑蕴含X→Y，记作F⊨X→Y
- 函数依赖集的闭包
  - 被函数依赖集F逻辑蕴含的函数依赖的全体构成的集合称为F的闭包，记做F^+

（1）函数依赖的推理规则

- Armstrong公理，可以从给定的函数依赖中推出新的函数依赖
  - 自反律（Reflexity）：若$B \subseteq A$，则$A \rightarrow B$成立
  - 增广律（Augmentation）：若$A \rightarrow B$，则$AC \rightarrow BC$（AC表示$A \cup C$）
  - 传递律（Transitivity）：若$A \rightarrow B$， $B \rightarrow C$，则$A \rightarrow C$
  - 自含律（Self_Determination）： $A \rightarrow A$
  - 分解律（Decomposition）：若$A \rightarrow BC$，则$A \rightarrow B$，且$A \rightarrow C$
  - 合并律（Union）：若$A \rightarrow B$， $A \rightarrow C$，则$A \rightarrow BC$
  - 复合律（Composition）：若$A \rightarrow B$， $C \rightarrow D$，则$AC \rightarrow BD$

（1）函数依赖的推理规则

- R(A, B, C, D, E, F)
- F={A→BC, B→E, CD→EF}
- AD→F对于函数依赖集F是否成立？
  - A→BC（已知）
  - A→C （分解律）
  - AD→CD（增广律）
  - CD→EF（已知）
  - AD→EF（传递律）
  - AD→F（分解律）

（2）码的形式化定义

- 设关系模式R(U)，F是R的一个FD集，X是U的一个子集，若
  - X→U ∈ F+，则X是R的一个超码，如果同时
  - 不存在X的真子集Y，使得Y→U成立，则X是R的一个候选码
- R(Tname, Addr, C#, Cname)
  - F={Tname→Addr, C#→Cname, C#→Tname}
  - C#→{Tname,Addr,C#,Cname}
  - 所以C#是候选码，若C#→Tname不成立，则候选码为{Tname, C#}

5、属性集的闭包

- 函数依赖集的闭包计算很麻烦
- 给定一个函数依赖集F，如何判断函数依赖X→Y是否可以从F中推出？
- 属性集的闭包
  - 设F是属性集U上的一个FD集，X是U的子集，则称所有用Armstrong推理规则推出的函数依赖X→A中所有A的集合，称为属性集X关于F的闭包，记做X^+
- X→Y能由Armstrong推理规则推出的充要条件是$Y \subseteq X^+$

（1）例子

- 关系模式R(A,B,C, D)
- F={A→B, B→C, B→D, A→D }
  - A^+=ABCD
  - B^+=BCD
  - C^+=C
  - D^+=D
- 不用计算F^+，就可知$A \rightarrow CD \in F^+$

6、最小函数依赖集

- 函数依赖集的等价和覆盖
  - 设S1和S2是两个函数依赖集，若$S1^+ \subseteq S2^+$,则称S2是S1的覆盖（或S2覆盖S1）
    - DBMS只要实现S2中的函数依赖，就自动实现了S1中的函数依赖
  - 若S2是S1的覆盖，且S1是S2的覆盖，则称S1与S2等价
    - DBMS只要实现任意一个FD集，就可自动实现另一个FD集

（1）定义

- 当且仅当函数依赖集F满足下面条件时，F是最小函数依赖集：
  - F的每个FD的右边只有一个属性
  - F不可约：F中的每个$X \rightarrow Y$，$F-\{X \rightarrow Y\}$与F不等价
  - F的每个FD的左部不可约：删除左边的任何一个属性都会使F转变为一个不等价于原来的F的集合

（2）举例

- Student(S#,Sname,Age, Sex)
  - F1={S#→Sname, S#→age, S#→sex}是最小函数依赖集
  - F2={S#→{S#,Sname}, S#→age, S#→sex}不是最小函数依赖集【右边不是单属性】
  - F3={S#→Sname, {S#,Sname}→age, S#→sex}不是最小函数依赖集【左部可约】
  - F4={S#→S#, S#→Sname, S#→age, S#→sex}不是最小函数依赖集【FD可约】

（3）求最小函数依赖集

- R(A,B,C,D)，F={A→BC, B→C, A→B, AB→C, AC→D}
  - 将右边写出单属性并去除重复FD（分解律）
    - F={A→B, A→C, B→C, A→B, AB→C, AC→D}
    - F={A→B, A→C, B→C, AB→C, AC→D}
  - 消去左部冗余属性
    - A→C， AC→D可推出A→AC， A→D，因此可去除AC→D中的C
    - A→C，可推出AB→BC可得AB→C，所以AB →C中的B是冗余属性
    - F={A→B, A→C, B→C, A→C, A→D}
={A→B, A→C, B→C, A→D}
  - 消去冗余函数依赖
    - A→C冗余，因为可由A→B， B→C推出
    - F={A→B, B→C, A→D}