



国家精品课程教材
高等院校信息技术系列教材

数据库系统 原理教程

(第2版)

陈 红 王 珊 张 孝 / 编著

- 教学课件
- 教学大纲
- 教学计划

清华大学出版社



日期: /

1. 数据模型

组成要素 { 数据结构
数据操作
数据的约束条件

分类 { 层次数据模型
网状数据模型
关系数据模型

三级模式结构: ^{在物理结构存储} 内模式 模式 ^{用户看到} 外模式

数据库 \Rightarrow 人工管理、文件系统、数据库系统
 \Rightarrow 结构: DB, DBMS, DBS

事物 \rightarrow 概念模型 (ER模型) \rightarrow 数据模型 { 网状
层次
关系

日期: /

二、关系数据模型

1. 关系操作: 增删改查、传统复合操作、专门关系操作.

• 存储路径对用户透明(不可见)

• students:

sid	Lname	fname	class	phone
		分量		

} \Rightarrow 标题/属性名 关系模式

} \Rightarrow 元组, 记录 { 基数 = 2 }

} 一个关系

属性、字段

维数(维度) = 5

关系类型: 基本关系、查询表、视图表

并没有实际存在数据库中

\rightarrow 从基表中取几列, 没有实际存在DB中

基本关系的性质:

① 非结构属性: 列中各值来自同一域

② 属性名唯一.

③ 列的顺序无所谓

④ 任意两元组不完全相同

⑤ 行的顺序无所谓

⑥ 分量值唯一. \rightarrow 分量不可再分, 取原子值

⑦ 关系名唯一

内容 \Rightarrow 值, 名称 \Rightarrow 型

日期: /

关系的完整性规则

① 第一范式: 不允许多值, 异域。每个属性不可再分

② 基于内容存取 \rightarrow 不能选取“第几行”, 选取“姓名=?”的元组

③ 行唯一性: 行不能完全一样

④ 实体完整性规则: 主键属性不空 \Rightarrow 修改或插入可能破坏主键不空 \rightarrow 因为主键空 $\frac{1}{2}$ 就称主键空

空值: 缺如, 尚未定义的值

超键: 唯一标识任意两行的属性或属性集 \rightarrow 语义上

每个至少一个

键(key, 候选码): 本身是超键但其任何子集非超键, 不含多余属性

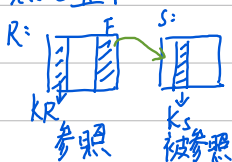
主属性: 键中的属性

主键: 键中选一个

主键属性: 主键里的属性

\Rightarrow R插入, SM删除, S, R修改有破坏

⑤ 参照完整性: F 或为空或等于 S 中某个元组的主码值



F 是 R 的外键

日期: /

关系代数

运算是封闭的

运算优先级: 投影 > 选择 > 笛卡尔积 > 连接 > 除法 > 交 > 并 > 差

单目运算只一个关系参与 对元组参与运算, 兼容要求

运算的前提 \Rightarrow 兼容表 (两表标题相同, 目相同, 相应属性域意义相同)

笛卡尔积: 行数 $|S \times R| = |S| \times |R| \rightarrow S$ 中每个与 R 中每个相接, 不管重复
列数 $|P \times Q| = |P| + |Q| \rightarrow$ 两表的列拼到一起

投影: 选取一些列, 去掉重复的元组, 显示选取的列

$S_1 := \Pi_{S\#, SP, SD}(S)$ 例: 查询学生表中有哪些系

赋值 选取哪些列

选择: 选取符合条件的元组 $\sigma_C(R) = \{t | t \in R \wedge C(t) = T\}$

符合的条件 (可以是复合的逻辑表达式)

例: 查询课程 C_1 的间接先修课

$C' := C$

$P := \sigma_{C.C\# = 'C_1'} \wedge C.C\# = C'.C\# (C \times C')$

$Q := \Pi_{C'.C\#}(P)$

日期: 自然连接

连接: 选修卡积中同名属性上分量相同的元组, 其它不要

有多同名则分量分别对应相同

例 查询选修 C_2 的学生姓名与成绩
(不在课程号表上)

$\pi_{S.N., G} (\sigma_{C.C\# = 'C_2'} (S \bowtie SC))$
姓名 成绩

除: $R: A_1 \dots A_n B_1 \dots B_n$ $S: B_1 \dots B_n$ S 中也可包含不在 R 中的属性, 不管组成的

T 是 R 有, S 没有的列, T 去掉重复的行后形成 K , 将 K 与 S 求笛卡积

k_i 代表 K 中元组, 若 k_i 的所有连接都在 R 中 则 k_i 保留为结果

例: $R: \begin{matrix} A & B & C \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_1 \\ a_1 & b_1 & c_2 \end{matrix}$ $S: \begin{matrix} C \\ c_1 \\ c_2 \end{matrix}$ $\Rightarrow R \div S = \begin{matrix} A & B \\ a_1 & b_1 \end{matrix}$

例: 选修了全部课程的学生学号

$\pi_{S\#, C\#} (SC) \div \pi_{C\#} (C)$

日期: /

运算依赖 \Rightarrow 基本: 并差、笛卡尔积、选择、投影

可被基本表示: 交、连接、除

其他关系运算:

θ 连接: 笛卡尔积中选取满足 $A_i \theta B_j$ 的元组

$\begin{matrix} \text{R的属性} & & \text{S的属性} \\ & \nwarrow & \nearrow \\ & A_i \theta B_j & \\ & \downarrow & \\ & > < = \dots & \end{matrix}$

当 θ 取 "=" 称等值连接 \rightarrow 相比自然连接只比较一列且不会去掉重复的

\downarrow
列数与笛卡尔积一致

日期:

/

SQL语言

非过程化语言 → 不管底层如何操作

不区分大小写

ABNF范式 → | 选择, { } 重复 0~∞ 次, [] 可选项

数据类型: { 布尔型 → T/F/UNKNOWN

字符型 → { 定长 CHAR(n)

变长 VARCHAR(n) → 可变量度 max 255 个字节

位类型 → { BIT(n)

BIT VARYING

定点数 → { INTEGER → 4字节 正负整数

SMALLINT → 2字节

NUMERIC → [precision] 精确数

DECIMAL

浮点 FLOAT 非精确数

时期时间 → { DATE

TIME

大对象型 → 内容多

创建数据库: CREATE SCHEMA 数据库名称

日期: /

标识符名 → 与c相同但不能长于128字符

域相关:

① 创建域: CREATE DOMAIN 域名称 [As] 类型

[DEFAULT] [CHECK]

↓
缺省值

↓
对域中值的约束

② 删除域: DROP DOMAIN 域名称 [RESTRICT] [CASCADE]

有人依赖该域则不删

↑
用创建时 As 后的类型代替域进行约束

日期: /

表相关

① 创建表: CREATE TABLE 名称

{ {列名 + 类型/域 + 约束} } ^{可重复} → 主键列必须 NOT NULL

[主键(主键组) PRIMARY KEY()]

{外键: FOREIGN KEY() REFERENCES 表名(列名)}

{表级约束} → NOT NULL 不空, UNIQUE 不可重复, DEFAULT 缺省值

[ON DELETE [CASCADE | SET DEFAULT | SET NULL | NO ACTION]]

[ON UPDATE [CASCADE | SET DEFAULT | SET NULL | NO ACTION]]

② 删表: DROP TABLE 表名 [RESTRICT | CASCADE]

型与值全删

有域或视图等依赖则不删

级联删除

③ 修改表: ALTER TABLE 表名 → 一般不可删列

[ADD 新列名]

[DROP 约束]

[MODIFY 列名, 类型]

定义取值范围

CHECK (VALUE IN (SELECT 列名 FROM 表名))

VALUE BETWEEN 1 AND 5

日期: /

索引相关

保存在字典(DD)

目录, 含有指针, 指向对应的元组

排序 降序
ASC/DESC

① 创建索引: CREATE INDEX 名称 ON 表名 (列名 [次序] [列名 [次序]])

[UNIQUE] [CLUSTER]

每个索引唯一对应一个记录

聚簇索引: 索引项与物理顺序相同依照索引重排表

② 删索引: DROP INDEX 名称

日期:

功能选择和投影

SELECT语句

所有的,默认,不重复

SELECT [ALL / DISTINCT] 无重复

所有列 $\leftarrow \{^* | \text{目标列表达式}, [AS \text{ 新名称}]\}$

FROM 表名 [别名]

[WHERE 条件]

[GROUP BY [HAVING]] 分组

[ORDER BY] 排列

目标列表表达式 \Rightarrow $\begin{cases} \text{Salary/12} \\ \text{表名.列名 (多个表)} \\ \text{'csl' 创建一列值为 'csl' } \end{cases}$

WHERE 条件表达式 → 对行进行筛选

集合函数不出现在WHERE中

→ { Salary > 100

(NOT BETWEEN)

Salary BETWEEN a AND b $\rightarrow a \leq \text{Salary} \leq b$

IN/NOT IN (a, b)

address LIKE/NOT LIKE '%abcd%'

要含有 'abcd' 的

转义符: ESCAPE 转义

通配符 % 表多个字符
_ 表单个字符

例:找第n个符合%的 \Rightarrow LIKE `'_1%%'` ESCAPE `'\'`

通配

空查找: IS NULL / IS NOT NULL

日期:

/ SELECT 语句最后一个语句

排序: ORDER BY 语句

也可用列号
→ 列名 [ASC/DESC], 列名 [ASC/DESC]

可多列名一起排

→ 在最后一步操作

集合函数: COUNT

忽略空值

SUM

AVG

MAX

MIN

[DISTINCT | ALL]

忽略重复值

COUNT 可用 * 统计元组数

列名 [AS 名称]

分组 GROUP BY 列名 [HAVING]: 按选定列分组, 只显示满足 HAVING 的
→ 至少包含一个集合函数, 否则可放到 where 中

先 where, 后 Group, 再 COUNT... 再 HAVING

SELECT 后的列必须在 Group 后的列之中, 要求每组占一行

日期: /

复杂查询

1. 连接查询

FROM 表1 [别名], 表2 [别名] → 求笛卡尔积

WHERE 连接条件 { 比较运算符

同名列要指明各自表名称

Between a and b

T₁.a = T₂.a → 相同一列相等, 等值连接

{ FROM S NATURAL JOIN SC = 自然连接

{ FROM S JOIN SC ON 连接条件

{ FROM S JOIN SC USING 列名 指明在哪一列等值连接

日期: /

2. 子查询: select 语句嵌套

{ 可用于 WHERE 和 HAVING, INSERT, UPDATE, DELETE 中
 不可用于 ORDER BY 中

放到 () 中作为表达式的一部分

{ 子查询的 SELECT 后跟单个列/表达式, 除非用 EXISTS.
 返回单值/单列多行/多列多行

比较运算符

IN

EXISTS

WHERE 列 = (子查询)

WHERE 列 IN (子查询)
列在子查询出的集合中

不相关子查询, 先子查询, 后父查询
内层独立于外层

相关子查询: 内层使用外层的 X, 对 X 的每一个取值执行内外程序, 确定 X 是否显示
类似 for 外层 m 行, 内层 n 行 则共做 $m \times n$ 次

日期: /

不等号 \Rightarrow "<>"

量化比较谓词: SOME, ANY, ALL.

WHERE 列 \in {SOME | ANY | ALL} (子查询) \rightarrow 可以返回集合
 有个就行 全部

EXISTS: 判断子查询返回是否为空

WHERE [NOT] EXISTS (子查询) \Rightarrow 子查询为空 WHERE 返回真

NOT EXISTS 实现 { 全称: 转为双重否定 $\forall x P \rightarrow \neg \exists x \neg P$
 除法: 用 NOT EXISTS 表否定

日期: /

高级 SQL:

1. 并 UNION: 集合 UNION [ALL] 集合

默认去重复, 加 ALL 不去重, 全显示

2. 交 INTERSECT: Q_1 INTERSECT [ALL] Q_2

例 $Q_1: aabbbcc$
 $Q_2: aaabbbcd$

↓
加 ALL 则各元素出现次数
为 Q_1, Q_2 最少次数

↓
加 ALL $\Rightarrow aabbbcc$

3. 差 EXCEPT

例 $Q_1: aaabbbcd$
 $Q_2: aabbbcc$

↓
加 ALL $\Rightarrow \{a, d\}$

不加 ALL $\Rightarrow \{d\}$

4. FROM \Rightarrow { 表名 [别名] (列名) } \rightarrow 对前几列改名, 在 SELECT 中直接用其别名
子查询

连接后的表: JOIN { 左外连接: LEFT OUTER JOIN

右外连接

外连接 FULL OUTER JOIN

不能实现传递闭包.

不能实现流计.

日期: /

SQL 的更新语句

INSERT :

一次插入一个元组

INSERT

INTO 表名 (列名)

VALUES (常量)/子查询 → 有多个会插入多个元组

↓
没写出来取空值

UPDATE 仅可更新一个表, 但可以更新多行多列

UPDATE 表名

SET 列名 = 表达式, 列名 = 表达式, 更新为

WHERE 条件 → 对哪些元组更新

↓
可子查询

DELETE 删除满足条件的行

DELETE

FROM 表名

WHERE 条件 → 不加 WHERE 删除表中所有数据

日期: /

视图

动态结果, 虚表, 视图上可以建立视图, 在视图中插入直接插到基本表中

定义视图 `CREATE VIEW 视图名 列名` } 将子查询的内容显示为视图
`AS`
`(SELECT)`
`[WITH CHECK OPTION]` } 更新插入时要满足 `SELECT` 中的 `WHERE`
且子句禁止迁移 (进出视图)

{ 水平视图 → 只是去掉基本表的某些行
垂直视图 → 只显示部分列
分组视图 → 用 `GROUP BY` 定义
连接视图 → 由连接运算得到

删除视图: `DROP VIEW 名称 [RESTRICT | CASCADE]`

↓
级联删除, 所有关联一起删

日期: /

视图查询: 将视图定义时的查询和现有的查询条件结合到一起对表查询
视图消解

集函数自身不作自身参数 且不能在 WHERE 中

视图更新 → 转化为对表的更新 (必须能够转化为对基本表的修改)
且满足表定义时的要求

视图的优缺点:

{ 数据独立性 → 上层应用只关心自己的视图

{ 更新有局限
结构有局限
性能有开销

视图物化.

/

SQL的完整性

过程性完整性约束 \rightarrow 触发器 \Rightarrow 在修改时起作用

非过程性完整性约束

数据取值 → { 非空 NOT NULL
唯一性 UNIQUE
约束取值范围 CHECK

域约束 → 创建域时对域进行约束

实体完整性 → 定义主键时, 主键列不为空 → 自动支持
→ 插入和更新不破坏

参照完整性 → 外键 → 自动支持插入和删除破坏

- 删父表值破坏完整性

默认 NO ACTION 拒绝操作
CASCADE
SET DEFAULT 子表置为缺省值
SET NULL 子表置空

⇒ 在定义表的末尾加上
ON DELETE
ON UPDATE

- 子表中插入破坏完整性 \rightarrow 拒绝操作

用户自定义的完整性 → CREATE ASSERTION 名称
CHECK ()

日期: /

触发器

触发事件 → insert, delete, update 语句

触发时机

触发动作

事件发生前/后检查

语句: CREATE TRIGGER 名称 {BEFORE | AFTER}

{INSERT | DELETE | UPDATE [OF 列名]}

ON 表名 [REFERENCING corr_name-def]

每行检查

FOR EACH ROW / EACH STATEMENT → 每语句检查

WHERE 条件

已命名语句 / BEGIN ATOMIC 语句 END

corr_name-def → 用于新旧对比

↓

OLD ROW AS 名称

NEW ROW AS 名称

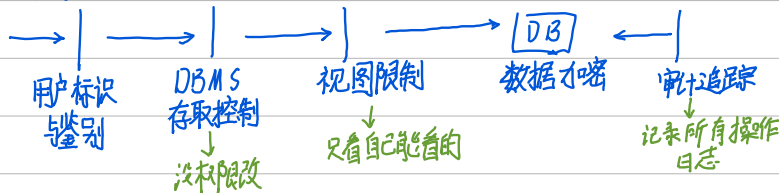
OLD TABLE AS 名称

NEW TABLE AS 名称

日期: /

数据库安全

安全模型:



用户标识与鉴别:

身份认证 → { 静态口令 → 用户设定
动态口令
生物特征
智能卡

存取控制子系统

最小特权策略

所有者 - 创建者 - 所有权 + 特权

{ 封闭系统 → 授予部分特权 → 只规定允许的
开放系统 → 去除部分特权 → 只规定不允许的

日期: /

存取控制的方式:

{ 自主的: 用户有权限且可将权限的子集转授他人

用户分为: 系统用户, 对象创建者, 一般用户, 公共用户

问题是 \rightarrow 无意泄露问题

强制存取控制: 用户与对象都有密级

主体

客体

敏感性标志

主 \geq 客 可读

客 \leq 客 可写

基于角色存取控制:

每种角色对应相应权限, 用户通过角色与权限关联

授权: $\text{GRANT } \{ \text{ALL } \text{PRIVILEGES} \}$

\rightarrow 权力 (增、删、改、查)

ON TABLE 名称

TO PUBLIC 用户名/角色名

[WITH GRANT OPTION] \rightarrow 允许转授

回收: $\text{REVOKE } [\text{GRANT OPTION FOR}]$

\checkmark 回收转授权

日期: /

角色创建: CREATE ROLE 名称

角色删除: DROP ROLE 名称

角色转授: GRANT 角色1

TO 角色2

WITH ADMIN OPTION → 可转授

审计: 操作全都记录下来到审计日志中

* 恢复日志 (只记录更新操作)

事件 { 服务器事件

系统权限

语句事件

模式对象事件

加密 → 方法 { 替换 → 字符一个一个换成别的
置换 → 改变顺序

统计数据库 → 不查具体一个元组, 只查询平均值等

日期: /

事务管理

事务 → 数据库操作中的原子操作

多个SQL语句 全做/全不做, 不可分割

并发控制, 恢复的基本单位

定义事务 → BEGIN TRANSACTION 语句

COMMIT → 正常结束

ROLLBACK → 异常终止, 回滚所有操作 → 整个事务相当于没操作

事务的特征: { 原子性 (A)

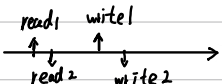
一致性 (C)

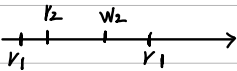
隔离性 (I): 不被其它事务干扰

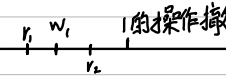
持久性 (D): 对数据改变是永久的

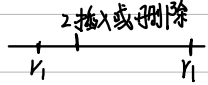
破坏 { 多事务并发执行
事务强行停止

不一致性:

丢失修改:  2 对数据的更改将 1 的覆盖

不可重复读:  1 读两次结果不同

读脏数据:  1 的操作撤销

幻像:  数据神秘多或少

日期: /

并发控制方法 → { 规定时间戳
共享资源加锁

封锁: 用之前加锁, 限制外人使用

→ 类型 { 排它锁, X型, 写
共享锁, S型, 读

→ 时机 { 事务开始前 / 操作开始前 → 加锁
操作结束后 / 事务结束后 → 解锁

封锁协议:

{ 一级封锁协议: 操作前 加X 事务结束 ⇒ 防丢失修改
对读操作不加锁 加在读数据

二级封锁协议: 在一级的基础上加上 读前 加S 读后 ⇒ 防读脏

{ 三级封锁协议: 在一级的基础上加上 读前 加S 事务后 ⇒ 防重复读

日期: /

并发调度可串行性: 并发结果与某一顺序串行结果一致

充分条件

两段锁协议: 锁动作分成加和解两阶段, 一旦解锁则不可加锁

可能造成死锁: 都等对方释放

处理死锁: \rightarrow { 预防: 对数据申请规定顺序
诊断 \rightarrow 操作超过规定完成时间
等待图法 \rightarrow 存在回路
解除: 撤销最小代价事务

活锁: T_1 锁 $R \rightarrow T_2$ 锁 $R \rightarrow T_3$ 锁 $R \rightarrow T_1$ 解锁 \rightarrow 给 T_3 , T_2 永远在等待导致饥饿

封锁粒度 \Rightarrow 粒度 \uparrow , 并发度 \downarrow , 开销 \downarrow

日期: /

多粒度封锁 → 允许每个结点独立加锁, 结点加显式锁后其后裔也要加隐式锁

意向锁 → 对上级加的, 加上 IS 后代表其后代将要加 S

\overbrace{IX}
 SIX

相容与否 (V/X)

	IS	IX	S	SIX	X
IS	✓	✓	✓	✓	X
IX	✓	✓	X	X	X
S	✓	X	✓	X	X
SIX	✓	X	X	X	X
X	X	X	X	X	X

隔离级别: (-致性级别): 对不一致接受的程度

↑
用设置

未提交读 → 低 → 可读另一事物未提交的数据, 可读脏, 可幻像
提交读 → 可以读脏, 幻像, 不允许脏修改, 不可重复读
可重复读 → 一次可以幻像
可串行读 → 高 → 不允许错误

日期: /

规范化和函数依赖

异常: { 冗余: 重复数据太多
更新异常: 修改很复杂, 需对多行修改
插入异常: 无法插入某实体
删除异常: 删除时另一个实体信息也没了

↑
存在函数依赖

函数依赖: $R(U)$, $X, Y \subseteq U$, X 确定则 Y 确定 \rightarrow 由语义范畴概念决定

属性集 决定因素

不由当前值决定

非平凡函数依赖: $X \rightarrow Y$ 但 $Y \not\subseteq X$
平凡函数依赖: $X \rightarrow Y$ 且 $Y \subseteq X$
 $X \rightarrow Y, Y \rightarrow X$ 称 $X \leftrightarrow Y$

部分函数依赖: $X \rightarrow Y$ 且 $\exists X' \rightarrow Y, X' \subset X$ 记 $X \twoheadrightarrow Y$
完全函数依赖: $X \rightarrow Y, X \subset X'$ 都有 $X' \rightarrow Y$, 记 $X \twoheadrightarrow Y$

虚线 X 为单属性一定成立

传递函数依赖: $X \rightarrow Y (Y \not\subseteq X), Y \rightarrow X, Y \rightarrow Z$ 则 $X \twoheadrightarrow Z$

日期:

关系模式: $R(U, D, dom, F) \rightarrow R(U, F)$

↑ 属性 ↑ 域 ↑ 依赖关系

属性集合 属性向域的映射集合

候选键: X 是 $R(U, F)$ 的属性有 $X \xrightarrow{+} U$, 无多余属性, 存在且不唯一

超键: $X \twoheadrightarrow U$

主键: 候选键中人为选一个

全键: 整个 U 为键

日期: /

范式

$R \in XNF$, R 为第 X 范式

规范化过程

$1NF \subset 2NF \subset 3NF \subset BCNF$

1NF { 不含多值属性和内部结构

用

值取多个域

所有候选键

2NF { 消除非主属性对键的部分函数依赖

投影分解

例 { $S \rightarrow SD, S \rightarrow SL,$

$SD \rightarrow SL, (S, C) \xrightarrow{F} G$

$(S, C) \xrightarrow{P} SD, (S, C) \xrightarrow{P} SL,$

取决定因素拿出来独立成表

拆为 $\{(S, C) \rightarrow G\}$

$\begin{cases} S \rightarrow SD \\ SD \rightarrow SL \\ S \rightarrow SL \end{cases}$

3NF { 消除非主属性对键的传递函数依赖

例 $S \rightarrow SD, SD \rightarrow SL$
 $S \rightarrow SL$

拆为: $\{S \rightarrow SD$

$\{SD \rightarrow SL\}$

BCNF { $R \in 1NF$ 且每个决定因素包含键

不存在主属性对键依赖

日期:

→ 能推出所有关系 (完备性)

阿氏公理: F 是 R 的依赖集, X, Y 是 R 属性子集, F 中推出 $X \rightarrow Y$ 称 F 逻辑蕴涵 $X \rightarrow Y$

定理: { 自反: $Y \subseteq X$ 则 $X \rightarrow Y$
增广: $X \rightarrow Y$ 则 $XZ \rightarrow YZ$
传递: $X \rightarrow Y, Y \rightarrow Z$ 则 $X \rightarrow Z$

推论 { 合并: $X \rightarrow Y, X \rightarrow Z$, 则 $X \rightarrow YZ$
分解: $X \rightarrow YZ$ 则 $X \rightarrow Y, X \rightarrow Z$
伪传递: $X \rightarrow Y, WY \rightarrow Z$ 则 $WX \rightarrow Z$
复合: $X \rightarrow Y, W \rightarrow V$ 则 $XW \rightarrow YV$

F 闭包: $F^+ = \{X \rightarrow Y \mid \text{所有基于阿氏公理推出}\}$

属性集 X 闭包: $X_F^+ = \{A \mid X \rightarrow A \text{ 由 } F \text{ 用阿氏推出}\}$

求 X_F^+ → 每个关系只用一次, 层层外推到不变为止
判断超键 X^+ 包含 U 则 X 是 R 超键

日期: /

函数依赖集等价 $F^+ = G^+ \iff F \subseteq G^+ \text{ 且 } G \subseteq F^+$

判断 F 中的每个 $X \rightarrow Y$ 是否在 G^+ 中满足 \rightarrow 计算 X^+ 看 Y 是否在其中

性质 $\Rightarrow \begin{cases} G \subseteq F \text{ 则 } G^+ \subseteq F^+ \\ (F^+)^+ = F^+ \end{cases}$

最小函数依赖集 \rightarrow 函数关系最少的

等价模式分解:

$\begin{cases} \text{分解后子模式属性集与原模式相同} \rightarrow \text{属性分解后并一起仍等于原来} \\ \text{无损连接} \rightarrow \text{分解后连接起来与原来相同} \\ \text{保持函数依赖} \rightarrow \text{分解后函数依赖并一起与原来相同} \end{cases}$
 \downarrow
缺一不可

日期: /

数据库系统设计

ER模型

