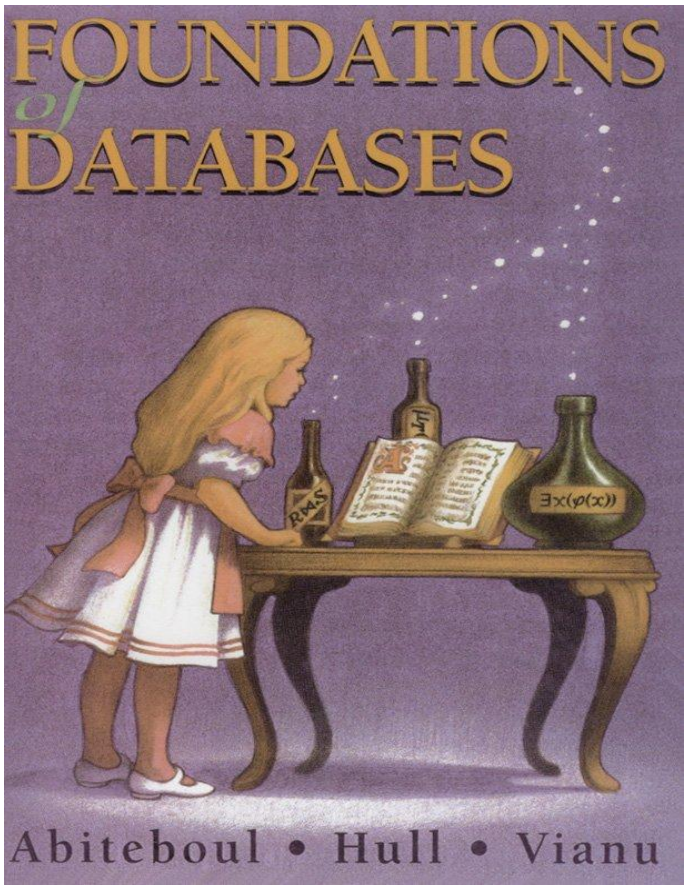
数据库基础教程讨论班

郭子正, 陈鹏宇

2023 年秋季学期

使用教材以及参考资料



- Abiteboul, Serge & Hull, Richard & Vianu, Victor. (1995). Foundations of Databases.
- M. Arenas et. al. Database Theory: Querying Data. 2022.
- 相关论文.

主要内容

关系数据库

Movies

Title	*Director*	*Actor*
The Trouble with Harry	Hitchcock	Gwenn
The Trouble with Harry	Hitchcock	Forsythe
The Trouble with Harry	Hitchcock	MacLaine
The Trouble with Harry	Hitchcock	Hitchcock
...	...	...
Cries and Whispers	Bergman	Andersson
Cries and Whispers	Bergman	Sylwan
Cries and Whispers	Bergman	Thulin
Cries and Whispers	Bergman	Ullman

Location

Theater	*Address*	*Phone Number*
Gaumont Opéra	31 bd. des Italiens	47 42 60 33
Saint André des Arts	30 rue Saint André des Arts	43 26 48 18
Le Champo	51 rue des Ecoles	43 54 51 60
...	...	...
Georges V	144 av. des Champs-Elysées	45 62 41 46
Les 7 Montparnassiens	98 bd. du Montparnasse	43 20 32 20

Pariscope

Theater	*Title*	*Schedule*
Gaumont Opéra	Cries and Whispers	20:30
Saint André des Arts	The Trouble with Harry	20:15
Georges V	Cries and Whispers	22:15
...	...	...
Les 7 Montparnassiens	Cries and Whispers	20:45

- 使用“表格”存储数据. 每个表叫做一个**关系** (relation), 关系的每一行叫做一个**元组** (tuple).
- 元组的每一列属性都从**值域** (domain) 中取值.
- 数据库**模式** (schema) 与数据库**实例** (instance) 有区别.

记号说明

- (至多) 可数的属性集合 **att**, 具有序关系 $\leq_{\mathbf{att}}$. 在描述元组时, 一般按照属性顺序.
- (至多) 可数的值域 **dom**. 常数从值域中取值. 使用 Dom(A) 描述属性 A 的值域.
- (至多) 可数的关系名称集合 **relname**.
- $sort(rel)$ 表示某关系 rel 的属性集合.$arity(rel) = |sort(rel)|$. 例如: sort(Movies)=$\{$Title,Director,Actor$\}$.arity(Movies)=3.
- 关系模式 Movies=(Title,Director,Actor)
- 元组表示示例:<The Trouble with Harry, Hitchhock, Gwenn>.
- 变量集合 **var**, 变量从值域中取值. **自由元组** (free tuple) 的属性值可以为变量或者常量.

不同视角[1]

Named and Conventional

$I(R) = \{f_1, f_2, f_3\}$

$$f_1(A) = a \quad f_1(B) = b$$
$$f_2(A) = c \quad f_2(B) = b$$
$$f_3(A) = a \quad f_3(A) = a$$

$I(S) = \{g\}$

$$g(A) = d$$

Unnamed and Conventional

$I(R) = \{\langle a, b\rangle, \langle c, b\rangle, \langle a, a\rangle\}$

$I(S) = \{\langle d\rangle\}$

Named and Logic Programming

$\{R(A:a, B:b),\ R(A:c, B:b),\ R(A:a, B:a),\ S(A:d)\}$

Unnamed and Logic Programming

$\{R(a, b),\ R(c, b),\ R(a, a),\ S(d)\}$.

- Named: 显式指定元组每一列的名称
- Unnamed: 使用序对表示元组. 属性顺序由全序关系唯一确定.
- 传统观点: 关系是元组的集合.
- 另一种观点: 关系是某种谓词.

[1] 将交替使用不同视角描述和探究问题. 可能会有不影响理解的记号滥用.

记号表 (仅供参考)

Constants	a, b, c
Variables	x, y
Sets of variables	X, Y
Terms	e
Attributes	A, B, C
Sets of attributes	U, V, W
Relation names (schemas)	R, S; $R[U], S[V]$
Database schemas	$\mathbf{R}, \mathbf{S}$
Tuples	t, s
Free tuples	u, v, w
Facts	$R(a_1, \ldots, a_n), R(t)$
Atoms	$R(e_1, \ldots, e_n), R(u)$
Relation instances	I, J
Database instances	$\mathbf{I}, \mathbf{J}$

主要内容

本节内容

我们介绍一些语言来刻画数据库上的查询. 不同的查询语言具有不同的语法和语义, 其表达能力也不尽相同.

三种查询范式

- Datalog: 一种带有函数谓词的逻辑程序.
- 关系演算 (relational calculus): 使用一阶谓词逻辑描述查询结果.
- 关系代数 (relational algebra): 定义一种由基础关系到最终结果的“操作序列”.

对查询语言进行扩展可以赋予其更强的表达能力. 语言的语义越强大, 能够表示的查询越多, 复杂性也随之增加.

主要内容

引入

Movies	*Title*	*Director*	*Actor*
	The Trouble with Harry	Hitchcock	Gwenn
	The Trouble with Harry	Hitchcock	Forsythe
	The Trouble with Harry	Hitchcock	MacLaine
	The Trouble with Harry	Hitchcock	Hitchcock
	...	...	...
	Cries and Whispers	Bergman	Andersson
	Cries and Whispers	Bergman	Sylwan
	Cries and Whispers	Bergman	Thulin
	Cries and Whispers	Bergman	Ullman

Location	*Theater*	*Address*	*Phone Number*
	Gaumont Opéra	31 bd. des Italiens	47 42 60 33
	Saint André des Arts	30 rue Saint André des Arts	43 26 48 18
	Le Champo	51 rue des Ecoles	43 54 51 60
	...	...	...
	Georges V	144 av. des Champs-Elysées	45 62 41 46
	Les 7 Montparnassiens	98 bd. du Montparnasse	43 20 32 20

Pariscope	*Theater*	*Title*	*Schedule*
	Gaumont Opéra	Cries and Whispers	20:30
	Saint André des Arts	The Trouble with Harry	20:15
	Georges V	Cries and Whispers	22:15
	...	...	...
	Les 7 Montparnassiens	Cries and Whispers	20:45

考虑如下查询:

- 谁是 “Cries and Whispers” 的导演?
- “Cries and Whispers” 在哪上演?
- Le Champo 的地址和电话是什么?
- 列出 Bergman 执导的电影名称以及上映的影院.

引入

例 2.1.1

列出 Bergman 执导的电影名称以及上映的影院.

解 2.1.2 (元组演算)

对于所有 $r_1 \in Movies, r_2 \in Pariscope, r_3 \in Location$, 如果
$r_1[Titles] = r_2[Titles]$ 并且
$r_2[Theater] = r_3[Theater]$ 并且
$r_1[Director] =$ "Bergman"
则输出 $< r_3[Theater], r_3[Address] >$.

引入

例 2.1.3

列出 Bergman 执导的电影名称以及上映的影院.

解 2.1.4

域演算 对于所有
$< x_{ti}, \text{“Bergman”}, x_{ac} > \in Movies, < x_{th}, x_{ti}, x_s > \in Pariscope,$
$< x_{th}, x_{ad}, x_p > \in Location,$
输出 $< Theater : x_{th}, Address : x_{ad} >$.

引入

例 2.1.5

列出 Bergman 执导的电影名称以及上映的影院.

解 2.1.6

基于规则的合取查询

$$ans(x_{th}, x_{ad}) \leftarrow Movies(x_{ti}, \text{“Bergman”}, x_{ac}), Pariscope(x_{th}, x_{ti}, x_s), Location(x_{th}, x_{ad}, x_p).$$

引入匿名变量

$$ans(x_{th}, x_{ad}) \leftarrow Movies(x_{ti}, \text{“Bergman”}, \_), Pariscope(x_{th}, x_{ti}, \_), Location(x_{th}, x_{ad}, \_).$$

Query-By-Example(QBE)

使用'_' 标记变量, 使用'P.' 标记输出.

Movies	*Title*	*Director*	*Actor*
	_The Seventh Seal	Bergman	

Pariscope	*Theater*	*Title*	*Schedule*
	_Rex	_The Seventh Seal	

Location	*Theater*	*Address*	*Phone number*
	P._Rex	P._1 bd. Poissonnière	

逻辑视角: 语法

定义 2.1.7 (Rule-based Conjunctive Query)

R 是数据库模式,**R** 上**基于规则的合取查询**指如下形式的表达式:

$$ans(u) \leftarrow R_1(u_1), \ldots, R_n(u_n)$$

其中 $u, u_1, \ldots, u_n$ 是自由元组. 箭头左边的部分叫做查询头 (head), 右边的部分叫做查询体 (body). 查询 q 中的变量记作 $var(q)$.

域限制

头部出现的变量必须在查询体中, 否则可能会导致无限的查询结果. 我们称这种情况是 "不安全的"(unsafe).

$$ans(x, y) \leftarrow R(x)$$

逻辑视角: 语义

定义 2.1.8 (赋值 (valuation))

给定 $V \subseteq \mathbf{var}$, **赋值** $\nu : V \to \mathbf{dom}$ 为 V 中每一个变量指定一个对应的常量.

定义 2.1.9 (数据库实例在查询下的像)

令 q 为某合取查询,$\mathbf{I}$ 是数据库模式 $\mathbf{R}$ 的某实例,$\mathbf{I}$ 在查询 q 下的**像**为

$$q(\mathbf{I}) = \{\nu(u) | \nu是var(q)上的赋值, \forall i \in [1, n], \nu(u_i) \in \mathbf{I}(R_i)\}$$

给定数据库实例, 查询的语义定义为该实例下查询的像.

合取查询性质

定义 2.1.10 (Active Domain)

- $adom(\mathbf{I})$: $\mathbf{I}$ 中的所有常量.
- $adom(q)$: q 中的所有常量.
- $adom(q, \mathbf{I})$: $adom(q) \cup adom(\mathbf{I})$.

定义 2.1.11 (单调)

$\forall \mathbf{I}, \mathbf{J} \in \mathbf{R}, \mathbf{I} \subseteq \mathbf{J} \implies q(\mathbf{I}) \subseteq q(\mathbf{J}))$.

定义 2.1.12 (可满足的 (satisfiable))

$\exists \mathbf{I}$ 使得 $q(\mathbf{I}) \neq \emptyset$

合取查询性质

定理 2.1.13

合取查询是单调可满足的.

证明.

$$ans(u) \leftarrow R_1(u_1), R_2(u_2), \ldots, R_n(u_n)$$

单调性: 注意到赋值函数的值域被扩大.
可满足性: 选取 $a \in \mathbf{dom}$ 但 $a \notin adom(q)$ 为某新的常量. 对任意关系 R_i 构造实例 $\mathbf{I}(R_i) = (adom(q) \cup \{a\})^{arity(R_i)}$. 令 ν 将 q 中各个变元赋值为 a. 显然 $\forall i, \nu(u_i) \in \mathbf{I}(R_i)$ □

是/非查询 (Yes-no Query)

例 2.1.14

有没有 Bergman 导演、在巴黎上映的电影?

解答

$$ans() \leftarrow Movies(x, \text{“Bergman”}, y), Pariscope(z, x, w)$$

如果返回 {<>}, 那么答案为 “是”, 如果返回 {}, 那么答案为 “否”.

合取演算: 语法

例 2.1.15

查询

$$ans(x_{th}, x_{ad}) \leftarrow \quad Movies(x_{ti}, \text{“Bergman”}, x_{ac}), Pariscope(x_{th}, x_{ti}, x_s), Location(x_{th}, x_{ad}, x_p).$$

等价于

$$\{x_{th}, x_{ad} | \exists x_{ti} \exists x_{ac} \exists x_s \exists x_p (Movies(x_{ti}, \text{“Bergman”}, x_{ac}), Pariscope(x_{th}, x_{ti}, x_s), Location(x_{th}, x_{ad}, x_p))\}.$$

合取演算: 语法

定义 2.1.16 ((Well-formed) Formula for Conjunctive Calculus)

假设 $\mathbf{R}$ 是数据库模式, 其上的一个合取演算式定义如下:

(a) $\mathbf{R}$ 上的一个 atom.

(b) 假设 φ, ψ 都是 formula, 那么 $\varphi \wedge \psi$ 也是.

(c) 假设 x 是变量,φ 是 formula, 那么 $\exists x\varphi$ 也是一个 formula.

自由变元

下列 φ 中的 x 是自由变元 (free variable).

- φ 是一个 atom (例:$\varphi = R(x)$).
- $\varphi = (\psi \wedge \xi)$, 其中 x 在 ψ 或 ξ 之一中是自由变元.
- $\varphi = \exists y\psi$, 其中 $x \neq y$, x 是 ψ 中的自由变元.

记 $free(\varphi)$ 为式中的自由变元. 变元若非自由则称受限 (bound).

合取演算: 语法

定义 2.1.17 (Conjunctive Calculus Query)

数据库模式 $\mathbf{R}$ 上的 conjunctive calculus query 为如下形式的表达式:

$$\{e_1, \ldots, e_m | \varphi\}$$

其中 φ 为 conjunctive calculus formula, $< e_1, \ldots, e_m >$ 为 free tuple, 其中的变元均为自由变元.

合取演算: 语义

合取演算的语义也通过赋值定义. 我们熟悉一下不同的记号.

赋值

$$\nu : a_i = \nu(x_i)$$

将赋值看成一种集合:

$$\{x_1/a_1, \ldots, x_n/a_n\}$$

I satisfies ϕ under ν

- $\varphi = R(u), \nu(u) \in \mathbf{I}(R)$; 或者
- $\varphi = (\psi \wedge \xi), \mathbf{I} \models \psi[\nu|_{free(\psi)}], \mathbf{I} \models \xi[\nu|_{free(\xi)}]$; 或者
- $\varphi = \exists\psi, \exists c \in \mathbf{dom}, \mathbf{I} \models \psi[\nu \cup \{x/c\}]$.

记作 $\mathbf{I} \models \varphi[\nu]$.

合取演算: 语义

合取演算查询的语义定义为实例在查询下的像:

实例在查询下的像

$$q(\mathbf{I}) = \{\nu(< e_1, \ldots, e_n >) | \mathbf{I} \models \varphi[\nu|_{free(\varphi)}]\}$$

两个合取演算式是**等价**的, 当且仅当:(1) 自由变元相同;(2) 任意实例上同时满足 (或不满足) 相同赋值.

合取演算查询的标准形式 (Normal Form)

$$\exists x_1, \ldots, x_m(R_1(u_1) \wedge \cdots \wedge R_n(u_n)).$$

任意合取演算式都等价于其标准形式.
Conjunctive Calculus Query $\Leftrightarrow$ Rule-based Conjunctive Query

讨论: 引入等号

例 2.1.18 (合取查询中的等号)

查询

$$ans(x_{th}, x_{ad}) \leftarrow \quad Movies(x_{ti}, \text{“Bergman”}, x_{ac}), Pariscope(x_{th}, x_{ti}, x_s), \\ Location(x_{th}, x_{ad}, x_p).$$

等价于

$$ans(x_{th}, x_{ad}) \leftarrow Movies(x_{ti}, x_d, x_{ac}), x_d = \text{“Bergman”}, \\ Pariscope(x_{th}, x_{ti}, x_s), Location(x_{th}, x_{ad}, x_p).$$

讨论: 引入等号

引入等号可能会带来麻烦.

例 2.1.19 (无限结果)

$$ans(x, y) \leftarrow R(x), y = z.$$

默认进行限制, 使等号中的变元等于常数或包含于某关系中.

例 2.1.20 (空结果)

设 $a \neq b$,

$$ans(x) \leftarrow R(x), x = a, x = b.$$

空查询 $q^{\emptyset}$.
基于规则的合取查询不会等价于空查询, 但是引入等号之后可能会.

复合查询

Extensional Databases(edb): 原始数据
Intensional Databases(idb): 中间结果

例 2.1.21 (复合查询)

假设 $\mathbf{R} = \{Q, R\}$, 查询如下:

$$S_1(x, z) \leftarrow Q(x, y), R(y, z, w)$$

$$S_2(x, y, z) \leftarrow S_1(x, w), R(w, y, v), S_1(v, z)$$

$$S_3(x, z) \leftarrow S_2(x, u, v), Q(v, z).$$

Q

1	2
2	1
2	2

R

1	1	1
2	3	1
3	1	2
4	4	1

S_1

1	3
2	1
2	3

S_2

1	1	1
1	1	3
2	1	1
2	1	3

S_3

1	2
2	2

复合查询

定义 2.1.22 (合取查询计划 (Conjunctive Query Program))

合取查询计划定义为如下形式的规则序列:

$$S_1(u_1) \leftarrow body_1$$

$$S_2(u_2) \leftarrow body_2$$

$$\vdots$$

$$S_m(u_m) \leftarrow body_m$$

其中各个 S_i 都不相同,$body_i$ 中只能包含 edb relation 以及 $S_1, \ldots, S_{i-1}$. 记 $[P(\mathbf{I})](S_i) = q_i([P(\mathbf{I})])$ 为 S_i 的内容.

复合查询

例 2.1.23 (空查询)

$$T(a, x) \leftarrow R(x)$$

$$S(x) \leftarrow T(b, x).$$

定理 2.1.24

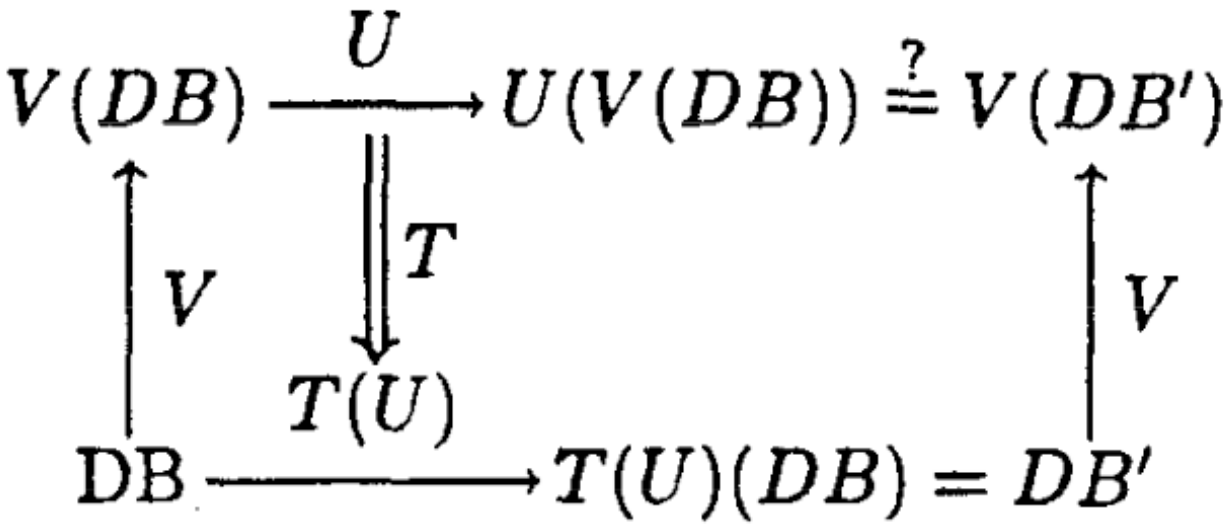
复合查询与 (可能带等式的) 合取查询等价.

视图

将查询的结果作为视图.

例 2.1.25 (视图传播问题[2])

对于 $q(\mathbf{I})$ 上的更新操作 $U(\cdot)$, 是否存在映射 $T(\cdot)$ 将视图上的操作映射到原始数据, 使得 $q(T(U)(\mathbf{I})) = U(q(\mathbf{I}))$?



[2] A.M.Keller, Algorithms for translating view updates to database updates for views involving selections, projections, and joins. PODS'85.

关系代数视角

The SPC Algebra - Selection, Projection, Cartisian Product.

例 2.1.26 (SPC Algebra)

$\pi_{2,3}(\sigma_{1=2}(\pi_4(\sigma_{1=5}(\sigma_{2=\text{“Bergman”}}(Movies) \times Pariscope)) \times Location))$.

The SPJR Algebra - Selection, Projection, Join, Renaming.

例 2.1.27 (SPJR Algebra)

$\pi_{Theater,Address}((\sigma_{Director=\text{“Bergman”}}(Movies) \bowtie Pariscope) \bowtie Location)$.

关系代数视角

Normal Form of SPC Algebra

$$\pi_{j_1,\ldots,j_n}(\{< a_1 >\} \times \cdots \times \{< a_m >\} \times \sigma_F(R_1 \times \cdots \times R_k))$$

一切 SPC Algebra 表达式都等价于其标准形式.

Normal Form of SPJR Algebra

$$\pi_{B_1,\ldots,B_n}(\{< A_1 : a_1 >\} \bowtie \cdots \bowtie \{< A_m : a_m >\} \bowtie \sigma_F(\delta_{f_1}(R_1) \bowtie \cdots \bowtie \delta_{f_k}(R_k)))$$

一切 SPJR Algebra 表达式都等价于其标准形式.

关系代数视角

定理 2.1.28 (语义等价性)

以下查询在语义上等价:

- *Rule-based Conjunctive Queries;*
- *Tableau Queries;*
- *Conjunctive Calculus Queries;*
- *Satisfiable SPC Algebra;*
- *Satisfiable SPJR Algebra.* □

语义增强: 关系并

例 2.1.29

哪里可以观看 “Annie Hall ” 或者 “Manhattan”?

解答 (Non-recursive Datalog)

$$ans(x_t) \leftarrow Pariscope(x_t, \text{“Annie Hall”}, x_s)$$

$$ans(x_t) \leftarrow Pariscope(x_t, \text{“Manhattan”}, x_s)$$

解答 (SPJRU)

$\pi_{Theater}(\sigma_{Title=\text{“Annie Hall”}} Pariscope) \cup \sigma_{Title=\text{“Manhattan”}}(Pariscope))$.

语义增强: 关系并

定理 2.1.30 (语义等价性)

以下查询在语义上等价:

- *Nonrecursive Datalog Program;*
- *SPCU Queries;*
- *SPJRU Queries.*

主要内容

引入

Movies

Title	*Director*	*Actor*
The Trouble with Harry	Hitchcock	Gwenn
The Trouble with Harry	Hitchcock	Forsythe
The Trouble with Harry	Hitchcock	MacLaine
The Trouble with Harry	Hitchcock	Hitchcock
...	...	...
Cries and Whispers	Bergman	Andersson
Cries and Whispers	Bergman	Sylwan
Cries and Whispers	Bergman	Thulin
Cries and Whispers	Bergman	Ullman

Location

Theater	*Address*	*Phone Number*
Gaumont Opéra	31 bd. des Italiens	47 42 60 33
Saint André des Arts	30 rue Saint André des Arts	43 26 48 18
Le Champo	51 rue des Ecoles	43 54 51 60
...	...	...
Georges V	144 av. des Champs-Elysées	45 62 41 46
Les 7 Montparnassiens	98 bd. du Montparnasse	43 20 32 20

Pariscope

Theater	*Title*	*Schedule*
Gaumont Opéra	Cries and Whispers	20:30
Saint André des Arts	The Trouble with Harry	20:15
Georges V	Cries and Whispers	22:15
...	...	...
Les 7 Montparnassiens	Cries and Whispers	20:45

考虑如下查询:

- 列出在 Gaumont Opera 上映但不由 Hitchcock 执导的电影.
- 列出 Hitchcock 执导但不参演的电影.
- 哪些电影的所有演员都在 Hitchcock 的指导下表演过?
- 哪些影院从不上映 Hitchcock 执导的电影?

引入否定：关系代数

- 在 SPCU、SPJRU 的基础上引入新运算符 “$-$”
  - named: $\{\sigma, \pi, \bowtie, \delta, \cup, -\}$
  - unnamed: $\{\sigma, \pi, \times, \cup, -\}$

例 2.2.1

列出在 Gaumont Opera 上映但不由 Hitchcock 执导的电影.

解 2.2.2 (关系代数)

$$\pi_{Title}\sigma_{Theater=\text{“Gaumont Opera”}}(Pariscope) - \pi_{Title}\sigma_{Director=\text{“Hitchcock”}}(Movies)$$

引入否定：nr-Datalog$^\neg$

- Nonrecursive Datalog 加入否定 (nr-Datalog$^\neg$)
  - 形如：$T \leftarrow R_1, \ldots, R_m, \neg S_1, \ldots, \neg S_n$

例 2.2.3

列出在 Gaumont Opera 上映但不由 Hitchcock 执导的电影.

解 2.2.4 (nr-Datalog$^\neg$)

$$R(x) \leftarrow Pariscope(\text{“Gaumont Opera”}, x, y)$$
$$S(x) \leftarrow Movies(x, \text{“Hitchcock”}, z)$$
$$ans(x) \leftarrow R(x), \neg S(x)$$

域限制[3]

所有变量必须存在于至少一个非否定的关系 R 中，否则可能会导致无限的查询结果. 我们称这种情况是 “不安全的”(unsafe).

$$ans(x) \leftarrow \neg R(x)$$

[3] 如不特殊说明，讨论的 nr-Datalog$^\neg$ 均满足域限制

引入否定：关系演算

- 在关系演算中，引入新运算符 “$\neg$”
- 由此衍生的新运算符：
  - $\vee$：或（$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \psi)$）
  - $\forall$：任意（$\forall x\varphi \equiv \neg\exists x\neg\varphi$）
  - $\rightarrow$：蕴含（$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$）
  - $\leftrightarrow$：等价（$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$）

例 2.2.5

列出在 Gaumont Opera 上映但不由 Hitchcock 执导的电影.

解 2.2.6 (关系演算)

$$\{x_t | \exists y_a \exists y_b Movies(x_t, \text{“Hitchcock”}, y_a) \wedge \neg Pariscope(\text{“Gaumont Opera”}, x_t, y_b)\}$$

“不安全的”查询

例 2.2.7

1. $\{x|\neg R(\text{“A”}, x)\}$
2. $\{x, y|R(\text{“A”}, x) \vee R(y, \text{“B”})\}$
3. $\{x|\forall y R(x, y)\}$

- 引入否定（$\neg$）后，会导致查询结果与值域（**dom**）相关
  - 如上述例 3 中，对于实例 $R = \{\langle 3, 1\rangle, \ldots, \langle 3, 5\rangle\}$
  - 在 **dom**$= \mathbb{N}^+$ 下的查询结果为 $\varnothing$
  - 在 **dom**$= \{1, 2, 3, 4, 5\}$ 下的查询结果为 $\{\langle 3\rangle\}$
- 即使不会导致无限的查询结果，其执行时间也可能是无限的
  - 如上述例 3 在 **dom**$= \mathbb{N}^+$ 下的查询执行需要验证所有的 $y \in \mathbb{N}^+$
- 需要引入带值域的查询概念，并分析查询的安全性

相对解释[4]

定义 2.2.8

设 $\mathbf{I}$ 是一个关系实例，活动域 $adom(\mathbf{I})$ 表示 $\mathbf{I}$ 中出现的值的集合，对于查询 $q = \{\vec{x}|\varphi\}$，$adom(q)$ 表示 q 中常量的集合。令 $adom(\mathbf{I}) \subseteq \mathbf{d} \subseteq \mathbf{dom}$，我们称 $\mathbf{I}$ **相对域** $\mathbf{d}$ **在赋值** ν **下满足** φ，记作 $\mathbf{I} \models_{\mathbf{d}} \varphi[\nu]$，若：

1. $\varphi = R(x_1, \ldots, x_n)$ is an atom and $\langle \nu(x_1), \ldots, \nu(x_n) \rangle \in \mathbf{I}(R)$;
2. $\varphi = (s = s')$ is an equality atom and $\nu(s) = \nu(s')$;
3. $\varphi = (\psi \wedge \xi)$ and $\mathbf{I} \models_{\mathbf{d}} \psi \left[\nu|_{\text{free }(\psi)}\right]$ and $\mathbf{I} \models_{\mathbf{d}} \xi \left[\nu|_{\text{free }(\xi)}\right]$;
4. $\varphi = (\psi \vee \xi)$ and $\mathbf{I} \models_{\mathbf{d}} \psi \left[\nu|_{\text{free }(\psi)}\right]$ or $\mathbf{I} \models_{\mathbf{d}} \xi \left[\nu|_{\text{free }(\xi)}\right]$;
5. $\varphi = \neg\psi$ and $\mathbf{I} \models_{\mathbf{d}} \psi[\nu]$ (i.e., $\mathbf{I} \models_{\mathbf{d}} \psi[\nu]$ does not hold);
6. $\varphi = \exists x\psi$ and for some $c \in \mathbf{d}, \mathbf{I} \models_{\mathbf{d}} \psi[\nu \cup \{x/c\}]$;
7. $\varphi = \forall x\psi$ and for each $c \in \mathbf{d}, \mathbf{I} \models_{\mathbf{d}} \psi[\nu \cup \{x/c\}]$.

[4] relativized interpretation

相对解释

定义 2.2.9

令 $q = \{x_1, \ldots, x_n | \varphi\}$，则 q 在实例 $\mathbf{I}$ 上相对域 $\mathbf{d}$ 的**像**为[5]：

$$q_{\mathbf{d}}(\mathbf{I}) = \{\langle \nu(x_1), \ldots, \nu(x_n)\rangle | \mathbf{I} \models_{\mathbf{d}} \varphi[\nu], \nu\text{的值域} \subseteq \mathbf{d}\}$$

例 2.2.10

$q = \{x | \forall y R(x, y)\}$，$\mathbf{I} = \{\langle 3, 1\rangle, \ldots, \langle 3, 5\rangle\}$，$\mathbf{dom} = \mathbb{N}^+$

- $q_{adom}(\mathbf{I}) = q_{adom(q,\mathbf{I})}(\mathbf{I}) = \{\langle 3\rangle\}$
- $q_{\{1,2,3,4,5,6\}}(\mathbf{I}) = \varnothing$
- $q_{\mathbf{dom}}(\mathbf{I}) = \varnothing$

[5] 更一般地，对于任意 $\mathbf{d} \subseteq \mathbf{dom}$，$q_{\mathbf{d}}(\mathbf{I}) = q_{\mathbf{d} \cup adom(q,\mathbf{I})}(\mathbf{I})$

[5] $adom(q, \mathbf{I}) = adom(q) \cup adom(\mathbf{I})$

域独立

定义 2.2.11

若查询 q 满足：对于任意关系实例 $\mathbf{I}$ 和 $\forall \mathbf{d}, \mathbf{d}' \subseteq \mathbf{dom}$，$q_{\mathbf{d}}(\mathbf{I}) = q_{\mathbf{d}'}(\mathbf{I})$，则称查询 q 在实例 $\mathbf{I}$ 上是**域独立的**.

例 2.2.12

$$\{x_t | \exists y_a \exists y_b Movies(x_t, \text{“Hitchcock”}, y_a) \\ \wedge \neg Pariscope(\text{“Gaumont Opera”}, x_t, y_b)\}$$

- 若查询 q 是域独立的，则对 $\forall \mathbf{d} \subseteq \mathbf{dom}$:

$$q_{adom}(\mathbf{I}) = q_{\mathbf{d}}(\mathbf{I}) = q_{\mathbf{dom}}(\mathbf{I})$$

语义等价性

定理 2.2.13 (语义等价性)

以下查询在语义上等价：

- 引入否定的关系代数（*named*、*unnamed*）
- *nr-Datalog*$^{\neg}$（单关系输出）
- 域独立的关系演算查询
- 活动域语义下的关系演算查询

域安全范式

将关系演算式转换为**域安全范式 (safe-range normal form, SRNF)**

- **替换变量名**，使得
  - 不存在不同的量词（$\exists$、$\forall$）限制相同变量
  - 不存在既有自由出现又有受限出现的变量
- **去除全称量词**：$\forall \vec{x} \varphi \Rightarrow \neg \exists \vec{x} \neg \varphi$
- **去除蕴含**：$\varphi \to \psi \Rightarrow \neg \varphi \vee \psi$，“$\leftrightarrow$”类似
- **否定下推**：
  - $\neg \neg \psi \Rightarrow \psi$
  - $\neg \left(\psi_1 \vee \cdots \vee \psi_n \right) \Rightarrow \left(\neg \psi_1 \wedge \cdots \wedge \neg \psi_n \right)$
  - $\neg \left(\psi_1 \wedge \cdots \wedge \psi_n \right) \Rightarrow \left(\neg \psi_1 \vee \cdots \vee \neg \psi_n \right)$
- **平坦化**，使得语法树上 $\wedge(\vee, \exists)$ 的孩子不是 $\wedge(\vee, \exists)$

域限制

ALGORITHM 5.4.3 (Range restriction (rr))

Input: a calculus formula φ in SRNF

Output: a subset of the free variables of φ or[3] $\perp$

begin
case φ **of**

$$
\begin{aligned}
R(e_1, \ldots, e_n) &: rr(\varphi) = \text{the set of variables in } \{e_1, \ldots, e_n\};\\
x = a \text{ or } a = x &: rr(\varphi) = \{x\};\\
\varphi_1 \wedge \varphi_2 &: rr(\varphi) = rr(\varphi_1) \cup rr(\varphi_2);\\
\varphi_1 \wedge x = y &: rr(\varphi) = \begin{cases} rr(\psi) & \text{if } \{x, y\} \cap rr(\psi) = \emptyset,\\ rr(\psi) \cup \{x, y\} & \text{otherwise;} \end{cases}\\
\varphi_1 \vee \varphi_2 &: rr(\varphi) = rr(\varphi_1) \cap rr(\varphi_2);\\
\neg\varphi_1 &: rr(\varphi) = \emptyset;\\
\exists\vec{x}\varphi_1 &: \textbf{if } \vec{x} \subseteq rr(\varphi_1)\\
&\qquad \textbf{then } rr(\varphi) = rr(\varphi_1) - \vec{x}\\
&\qquad \textbf{else return } \perp
\end{aligned}
$$

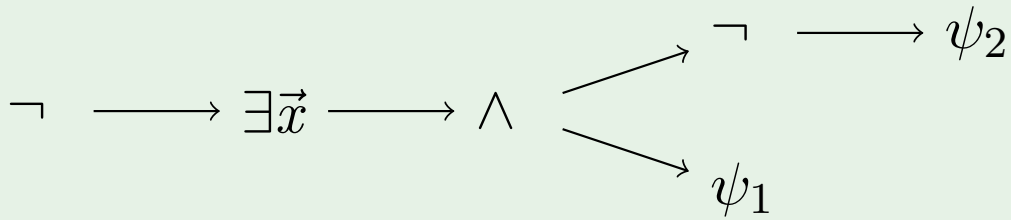
end case
end ■

域限制

例 2.2.14

$$\psi = \forall \vec{x}(\psi_1(\vec{x}) \to \psi_2(\vec{y}))$$

- SRNF：$\psi' = \neg\exists\vec{x}(\psi_1(\vec{x}) \wedge \neg\psi_2(\vec{y}))$



- $rr(\psi)$ 存在当且仅当 $rr(\psi_1) = \vec{x}, rr(\psi_2) \neq \bot$
- 此时，$rr(\psi) = \varnothing$

域安全的关系演算

定义 2.2.15

一个关系演算查询 $\{\vec{x}|\varphi\}$ 是“域安全[6]”的若 $rr(\mathrm{SRNF}(\varphi)) = free(\varphi)$

定理 2.2.16 (语义等价性)

以下查询在语义上等价：

- 引入否定的关系代数（*named*、*unnamed*）
- *nr-Datalog*$^\neg$（单关系输出）
- 域独立的关系演算查询
- 活动域语义下的关系演算查询
- **域安全的关系演算查询**

推论 2.2.17

域安全的关系演算查询是域独立的.

[6] safe range

域独立的含并查询

定义 2.2.18

仅由 $\wedge, \vee, \exists$ 构成，且可能存在等号的关系演算查询的域独立性是容易判断的，其中域独立的查询称为“正存在（演算）查询[7]”

定理 2.2.19

正存在查询与含并的合取查询语义等价.

[7] positive existential (calculus) query

聚集函数

- 聚合函数：sum, count, max, min, avg, ...
- 不能简单地通过投影来实现聚合函数 (会遗漏重复值)

例 2.2.20

关系模式 $Sales[Title, Date, Attendance]$ 下，求每个电影的总观影人数

- 错误表示 (非正式)：

$$\{\langle x_t, s\rangle | x_t \in \pi_{Title}(Sales) \wedge s = \mathrm{sum}\{\pi_{Attendance}\sigma_{Title=x_t}(Movies)\}\}$$

- 正确表示 (非正式)：

$$\{\langle x_t, s\rangle | x_t \in \pi_{Title}(Sales) \wedge s = \sum_{\langle x_t, x_d, x_a\rangle \in \sigma_{Title=x_t}(Movies)} x_a\}$$

- 关系代数表示

$$\pi_{Title,\mathbf{sum}(Attendance)}(Sales)$$

聚集函数

定义 2.2.21

聚集函数 f 是一个函数族 $f_1, f_2, \ldots$，对于关系模式 S（$arity(S) \geq i$），$f_i : inst(S) \to \mathbb{N}$，且 f_i 的函数值只依赖 S 的第 i 列.

例 2.2.22

关系模式 $Sales[Title, Date, Attendance]$ 下，求每个电影的总观影人数

$$\{\langle x_t, s\rangle | \exists x_d, x_a Sales(x_t, x_d, x_a) \wedge s = \mathbf{sum}_2\{\langle y_d, y_a\rangle | Sales(x_t, y_d, y_a)\}\}$$

- 可以将聚集项 $(f_i\{\vec{x}|\psi\})$ 引入关系演算式中 (ψ 中可能含有聚集项)
- 域安全的带聚集项的关系演算与带聚集的关系代数表达能力相同

讨论：无限数据库的有限表示

例 2.2.23

$\mathbf{dom} = \mathbb{R}$，$R$ 表示以 $(0,0),(4,0),(4,3),(0,3)$ 为顶点的矩形区域：

$$\langle x, y\rangle \in R \iff (x, y) \in [0,4] \times [0,3]$$

- 对于有限数据库，任意有限域 $\mathbf{d} \subseteq \mathbf{dom}$
- 有限占位符集合 C[8]：$\forall c \in C$ 代替 $\mathbf{dom} - \mathbf{d}$ 的一个子集
- 用 **“含占位元组”**$t = \langle t_1, \ldots, t_n\rangle \in (\mathbf{d} \cup C)^n$ 表示 $\mathbf{dom}^n$ 中的元组

定理 2.2.24

对有限数据库上的任意演算查询 $\{\vec{x}|\varphi\}$，可以用包含最多 $|var(\varphi)|$ 种不同的占位符的元组有限地表示其查询结果，即使其查询结果是无限的.

- 有限数据库上的任意演算查询结果是递归的

[8] $C \cap \mathbf{dom} = \varnothing$

讨论：无限数据库的有限表示

- 将元组 $\langle t_1, \ldots, t_n \rangle$ 看作 $\langle x_1 = t_1, \ldots, x_n = t_n \rangle$
- **广义元组：** $\langle \zeta_1(x_1), \ldots, \zeta_n(x_n) \rangle$
  - $\zeta_i(x_i)$ 表示对 x_i 的限制条件（$1 \leq i \leq n$）
- **广义关系实例：** 广义元组的集合

例 2.2.25

$\mathbf{dom} = \mathbb{R}$，$R$ 表示以 $(0,0), (4,0), (4,3), (0,3)$ 为顶点的矩形区域：

$$R = \{\langle 0 \leq x \leq 4, 0 \leq y \leq 3 \rangle\}$$

主要内容

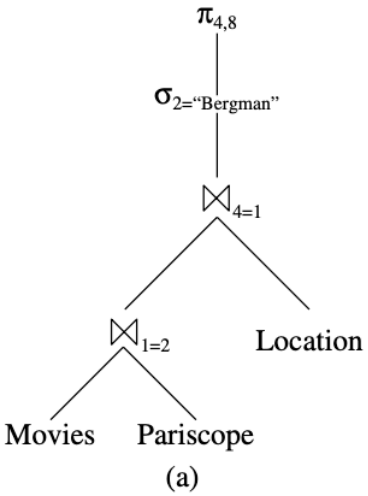
引入

如何更快地求解查询?

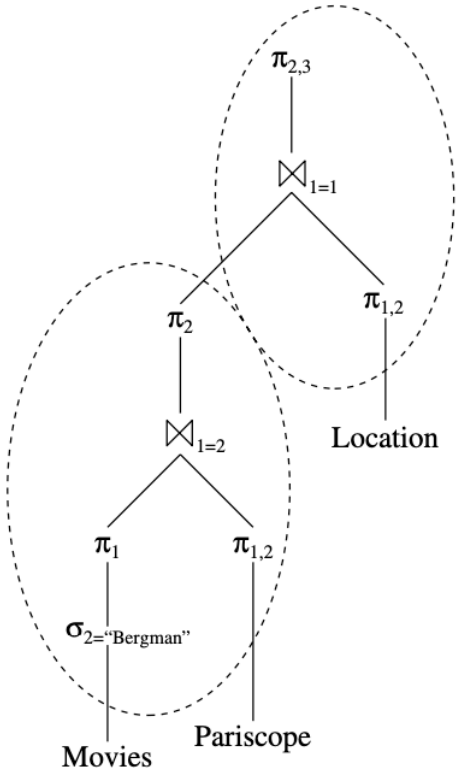
- 物理层面: 使用索引 (B-树, 哈希, ...)/ 使用缓存 (更好的换页策略等) / ...
- 逻辑层面: 查询改写 / ...

我们本次着重讨论逻辑层面的优化.

查询树/查询改写



(a)



(b)

$$
\begin{array}{lcl}
\sigma_F(\sigma_{F'}(q)) & \leftrightarrow & \sigma_{F \wedge F'}(q) \\
\pi_{\vec{j}}(\pi_{\vec{k}}(q)) & \leftrightarrow & \pi_{\vec{l}}(q) \\
\sigma_F(\pi_{\vec{l}}(q)) & \leftrightarrow & \pi_{\vec{l}}(\sigma_{F'}(q)) \\
q_1 \bowtie q_2 & \leftrightarrow & q_2 \bowtie q_1 \\
\sigma_F(q_1 \bowtie_G q_2) & \rightarrow & \sigma_F(q_1) \bowtie_G q_2 \\
\sigma_F(q_1 \bowtie_G q_2) & \rightarrow & q_1 \bowtie_G \sigma_{F'}(q_2) \\
\sigma_F(q_1 \bowtie_G q_2) & \rightarrow & q_1 \bowtie_{G'} q_2 \\
\pi_{\vec{l}}(q_1 \bowtie_G q_2) & \rightarrow & \pi_{\vec{l}}(q_1) \bowtie_{G'} q_2 \\
\pi_{\vec{l}}(q_1 \bowtie_G q_2) & \rightarrow & q_1 \bowtie_{G'} \pi_{\vec{k}}(q_2)
\end{array}
$$

Sideways Information Passing

为了优化

$$\pi_{\overrightarrow{j}}(\sigma_F(R_1 \times \cdots \times R_n))$$

可以将只涉及单个关系的选择下推，并对自然连接重新排序：

$$(\ldots(R_{i_k} \bowtie R_{i_{k+1}}) \bowtie \cdots \bowtie R_{i_n})$$

最好的自然连接顺序是什么？
或者，如何高效求解下列合取查询？

$$ans(z) \leftarrow \underbrace{R_1(u_1), \ldots, R_n(u_n)}_{\text{relation atoms}}, \underbrace{C_1, \ldots, C_m}_{\text{constraint atoms}}$$

Sideways Information Passing

利用 atom 之间的共同信息优化连接顺序, 按照顺序从左到右做连接.

例 2.3.1

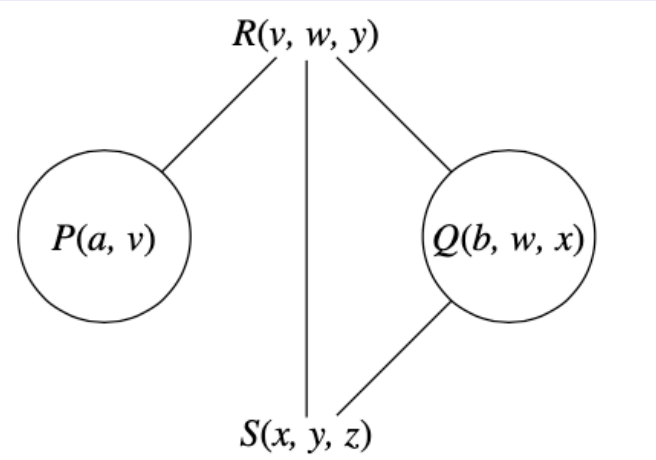
$ans(z) \leftarrow P(a, v), Q(b, w, x), R(v, w, y), S(x, y, z), v \leq x.$

一些直觉 (一)

- 自带常数的变量相当于被选择过, 因此可能基数较小, 尽可能先处理.
- 如果共享变量, 那么集中处理相当于先施加选择再连接, 避免先做笛卡尔积.
- 如果 constraint 中的变量之前都出现过, 那么尽早处理能够对现在当前的中间结果大小.

Sideways Information Passing

The SIP Graph



- 关系名作为顶点
- 共享变量的顶点邻接
- 标出出现常数的关系

Sideways Information Passing

基于 SIP 的启发式算法

当前处理的 atom A_j 满足:

- 包含常数, 或
- 与之前出现的某关系共享变量, 或
- A_j 是 constraint, 并且其中的变量在之前都出现过.

例 2.3.2 (利用 sip 策略优化之后的查询)

- $P(a, v), Q(b, w, x), v \leq x, R(v, w, y), S(x, y, z)$
- $P(a, v), R(v, w, y), S(x, y, z), v \leq x, Q(b, w, x)$
- $Q(b, w, x), R(v, w, y), P(a, v), S(x, y, z), v \leq x$

Sideways Information Passing

一些直觉 (二)

- 如果某个曾经出现过的变量 v 在之后再未出现, 那么就可以被遗忘. 可以直接投影掉中间结果中对应的列.

例 2.3.3

$$ans(z) \leftarrow P(a, v), R(v, w, y), \underbrace{S(x, y, z)}_{\text{在此之后 y 可忽略}}, \underbrace{v \leq x}_{\text{在此之后 v 可忽略}}, Q(b, w, x).$$

Query Decomposition: Join Detatchment

拆分连接.

注意记号差别

此处使用元组变量. 例如,p_1 是 P 中的一个元组, 条件 $C = (p_1.a = p_2.b)$.

例 2.3.4 (使用枢纽变量)

$$\begin{aligned} ans(z) \leftarrow & P_1(p_1), \cdots, P_m(p_m), \overbrace{C_1, \cdots, C_k, T}^{\text{只与}t,q,p_1,\cdots,p_m\text{有关}}, \\ & Q(q), \\ & R_1(s_1), \cdots, R_n(s_n), \underbrace{D_1, \cdots, D_l}_{\text{只与}r_1,\cdots,r_n\text{有关}} \end{aligned}$$

等价于

$$\begin{aligned} temp(q) \leftarrow & Q(q), R_1(r_1), \cdots, R_n(r_n), D_1, \cdots, D_l \\ ans(t) \leftarrow & P_1(p_1), \cdots, P_m(p_m), temp(q), C_1, \cdots, C_k, T. \end{aligned}$$

Query Decomposition: Tuple Substitution

替换元组, 缩小连接规模以减少中间过程大小.

例 2.3.5

for each r **in** R_i **do**

$$ans(s) + \leftarrow R_1(s_1), \cdots, R_{i-1}(s_{i-1}), R_{i+1}(s_{i+1}), \cdots, R_n(s_n), (C_1, \cdots, C_m, T)[s_i/r]$$

The Homomorphism Theorem

启发式算法基于局部性质，如何找到全局最优解？

定义 2.3.6 (查询的包含)

若 $\forall \mathbf{I}, q(\mathbf{I}) \subseteq q'(\mathbf{I})$，则称 q 包含于 q'，记为 $q \subseteq q'$. 若 $q \subseteq q', q' \subseteq q$，则称 q 与 q' 等价，记为 $q \equiv q'$.

例 2.3.7

$$q_0(x, y) \leftarrow R(x, y)$$
$$q_1(x, y) \leftarrow R(x, y_1), R(x_1, y_1), R(x_1, y)$$
$$q_2(x, y) \leftarrow R(x, y_1), R(x_1, y_1), R(x_1, y_2), R(x_2, y_2), R(x_2, y)$$
$$q_\omega(x, y) \leftarrow R(x, y_1), R(x_1, y)$$

可以看出，$q_0 \subseteq q_1 \subseteq q_2 \subseteq q_\omega$.

如何证明两个查询的包含/等价关系？

The Homomorphism Theorem

记号

赋值 (valuation): 将变量映射为常量.
替换 (substitution:) 将变量映射为常量或者变量.

利用变量替换, 我们证明查询的包含关系:

Homomorphism

若映射 θ 将 q' 中出现的变量替换为 q 中出现的变量, 则称 θ 为从 q 到 q' 的同态.

定理 2.3.8 (Homomorphism Theorem)

合取查询 $q \subseteq q'$ 当且仅当存在从 q 到 q' 的同态.

证明留作练习.

The Homomorphism Theorem

如何证明查询等价？根据同态定理，我们有

推论 2.3.9

$q \equiv q'$ 当且仅当存在从 q 到 q' 以及 q' 到 q 的同态.

对值域的限制

推论在有限值域下不成立，例如，当 R 的每一个变量的值域都恰好为 $\{a\}$ 时，下列查询等价但互相不存在同态：

$$q(x) \leftarrow R(x, a)$$

$$q'(y) \leftarrow R(a, y)$$

利用同态优化查询

将某个查询替换为使用“规模更小”的等价查询.

$$ans(x, y, z) \leftarrow R(x_2, y_1, z), R(x, y_1, z_1), R(x_1, y, z_1), R(x, y_2, z_2), R(x_2, y_2, z)$$

记号回顾:Tableau Query

将合取查询属于相同关系的变元写进同一张表格. 我们的优化目标等价于尽可能多地消去表格里的行.

R	A	B	C
u_1	x_2	y_1	z
u_2	x	y_1	z_1
u_3	x_1	y	z_1
u_4	x	y_2	z_2
u_5	x_2	y_2	z
u	x	y	z

- u_4 可被消除,$\theta(y_2) = y_1$
- u_5 可被消除,$\theta(z_2) = z_1$

计算复杂性

不同意义下的复杂性

- 数据复杂性 (Data Complexity): 只改数据, 不改变查询本身.
- 查询复杂性 (Query/Expression Complexity): 只可以改查询.
- 联合复杂性 (Combined Complexity): 都可以改.

定理 2.3.10 (查询包含的难度)

给定合取查询 q, q', 以下问题是 (联合复杂性下)*NP*-完全的:

- 判断是否 $q \subseteq q'$.
- 判断是否 $q \equiv q'$.

计算复杂性

合取查询总是可满足的，带等号的合取查询虽然不一定可满足，但是可满足性能够判断. 对于更加一般的关系演算，其可满足性是不可判定的.

定理 2.3.11 (查询可满足性)

关系演算可满足性问题是递归可枚举但非递归的.

证明.

构造从 PCP 到该问题的归约. □

计算复杂性

推论 2.3.12

(1) 关系演算的包含与等价问题不是图灵可识别的.

(2) 关系演算的域独立性不是图灵可识别的.

证明.

对 (1), 判断 $q - q'$ 的可满足性.
对 (2), 构造域不独立的查询 ψ, 判断查询 ϕ 的可满足性相当于判断 $\phi \wedge \psi$ 的域独立性. □

计算复杂性

很不幸，一般来说无法用图灵机找到最优化的查询.

合取查询求值

——但是我们可以优化某些比较简单的查询.

合取查询求值

合取查询的复杂性

数据复杂性是多项式时间. 查询复杂性可能是指数.

定理 2.3.13

在联合复杂性下, 给定 t, 投影-连接查询 q, 数据库实例 $\mathbf{I}$, 判断 $t \in q(\mathbf{I})$ 是否成立是 *NP*-难的.

R_i	A_i	A_{i+1}
	0	a
	0	b
	1	a
	1	b
	a	0
	a	1
	b	0
	b	1

(a)

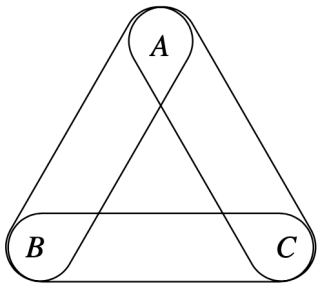
R_n	A_n	A_1
	0	a
	0	b
	1	a
	1	b
	a	0
	a	1
	b	0
	b	1

(b)

合取查询求值

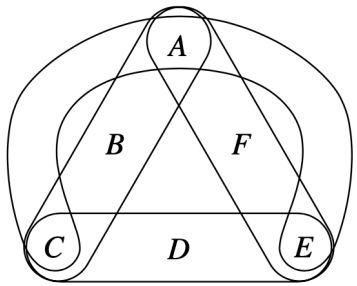
查询困难的原因: 包含环, 中间查询结果可能会指数爆炸.

例 2.3.14 (环)



$R_1[AB], R_2[BC], R_3[AC]$

(a)



$S_1[ABC], S_2[CDE], S_3[AFE], S_4[ACE]$

(b)

考虑无环的合取查询? 组成一棵树, 避免了指数爆炸.

定理 2.3.15

如果数据库模式 $\mathbf{R}$ 无环, 那么对于任意的数据库实例 $\mathbf{I}$, 查询 $\pi_X(\bowtie \mathbf{I})$ 可以在 (输入以及输出大小的) 多项式时间内完成.

主要内容

本节内容

合取查询应用非常广泛, 也是近几十年数据库理论研究的重要对象之一.

合取查询执行在联合复杂性意义上是 NP-难的.

- 细分合取查询联合复杂性
- 合取查询高效执行算法

主要内容

一阶逻辑查询复杂性

问题 3.1.1 (一阶逻辑查询 (FO) 判定问题)

输入：一阶逻辑查询 φ、数据库实例 D 和赋值 η
输出：为真当且仅当 $(D, \eta) \models \varphi$

定理 3.1.2

一阶逻辑查询判定问题是 *PSpace*-完全问题

一阶逻辑查询判定算法

Algorithm 1 EVALUATION(φ, D, η)

Input: An FO formula φ, a database D, and an assignment η for φ over D.
Output: `true` if $(D, \eta) \models \varphi$, and `false` otherwise.

1: **if** φ is of the form $R(\bar{x})$ **then return** $R(\eta(\bar{x})) \in D$
2: **else if** φ is of the form $(x_i = x_j)$ **then return** $\eta(x_i) = \eta(x_j)$
3: **else if** φ is of the form $(x_i = a)$ **then return** $\eta(x_i) = a$
4: **else if** φ is of the form $\neg\varphi'$ **then return** $\neg$EVALUATION(φ', D, η)
5: **else if** φ is of the form $\varphi' \vee \varphi''$ **then**
6: **return** EVALUATION$(\varphi', D, \eta) \vee$ EVALUATION(φ'', D, η)
7: **else if** φ is of the form $\exists x\, \varphi'$ **then**
8: **return** $\bigvee_{a \in \mathrm{Dom}(D)}$ EVALUATION$(\varphi', D, \eta[x/a])$
9: $\triangleright$ $\eta[x/a]$ extends η by setting $\eta(x) = a$.

引理 3.1.3

EVALUATION 算法的空间复杂度为 $O(|\varphi|^2 \cdot \log |D|)$

定理 3.1.4

数据复杂性下，一阶逻辑查询判定问题属于 *DLogSpace*

合取查询复杂性

问题 3.1.5 (合取查询 (CQ) 判定问题)

输入：合取查询 φ、数据库实例 D 和赋值 η
输出：为真当且仅当 $(D, \eta) \models \varphi$

定理 3.1.6

数据复杂性下，一阶逻辑查询判定问题属于 *DLogSpace*

合取查询复杂性

定理 3.1.7

合取查询判定问题是 *NP*-完全问题

证明.

复杂性上界（在 NP 内）是显然的.
对于给定的图 G 和正整数 k，构造如下的数据库实例 D 和查询 φ：

$$D = \{Node(v) | v \in V(G)\} \cup \{Edge(u, v) | (u, v) \in E(G), u \neq v\}$$

$$\varphi = \exists x_1, \ldots, x_k \bigwedge_{i=1}^{k} Node(x_i) \wedge \bigwedge_{\substack{1 \leq i,j \leq k \\ i \neq j}} Edge(x_i, x_j)$$

此时，$(D, \langle\rangle) \models \varphi$ 当且仅当图 G 有 k-团. □

参数化复杂性

定义 3.1.8 (参数化问题)

给定字母表 Σ，一个参数化问题可以表示为二元组 (L, κ)，其中 $L \subseteq \Sigma^*$ 表示问题识别的语言，多项式可计算函数 $\kappa : \Sigma^* \to \mathbb{N}$ 称为参数。

问题 3.1.9 (p-Clique)

输入：无向图 G，正整数 k
参数：k
输出：为真当且仅当图 G 有 k-团

问题 3.1.10 (p-CQ 判定问题)

输入：合取查询 φ、数据库实例 D 和赋值 η
参数：$|\varphi|$
输出：为真当且仅当 $(D, \eta) \models \varphi$

Fixed-Parameter Tractability

定义 3.1.11 (Fixed-Parameter Tractable(FPT))

一个参数化问题 (L, κ) 是固定参数可解 (属于 FPT) 的当且仅当存在一个多项式时间可计算的函数 $f\colon \mathbb{N} \to \mathbb{R}_0^+$ 使得对于 $\forall w \in \Sigma^*$，可在

$$O\left(f(\kappa(w)) \cdot poly(|w|)\right)$$

时间内判断 w 是否属于 L.

- 对于 CQ 查询，希望找到 $O(f(|\varphi|) \cdot poly(|D|))$ 的参数化算法
- 直观（暴力）算法的复杂性为 $O\left(|D|^{|\varphi|} \cdot poly(|D| + |\varphi|)\right)$

FPT 归约

定义 3.1.12 (FPT 归约)

参数化问题 (L_1, κ_1) 到 (L_2, κ_2) 的 FPT 归约是函数 $\Phi : \Sigma^* \to \Sigma^*$，使得存在多项式可计算的函数 f, g，对 $\forall w \in \Sigma^*$ 满足：

- $\Psi(w)$ 可在 $f(\kappa_1(w)) \cdot poly(|w|)$ 时间内计算
- $w \in L_1$ 当且仅当 $\Phi(w) \in L_2$
- $\kappa_2(\Phi(w)) \leq g(\kappa_1(w))$

W Hierarchy

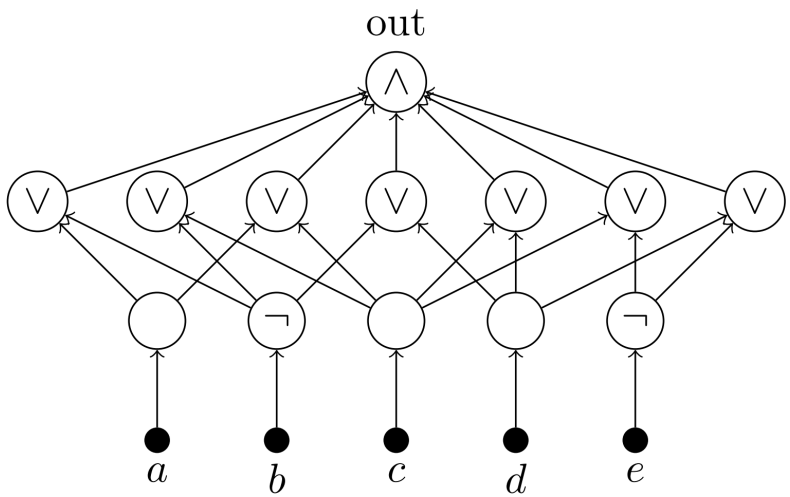
问题 3.1.13 (Weft-t Depth-d Weighted Circuit Satisfiability[9])

输入：和逻辑电路 C，每个输入到输出的路径上至多有 d 个门，其中至多 t 个扇入数大于等于 3

参数：k

输出：为真当且仅当可以将恰好 k 个输入设为真后，电路输出为真

- Weighted 2-SAT 问题可以 FPT 归约为 WCS[$C_{1,3}$]



[9] WCS[$C_{t,d}$]

W Hierarchy

定义 3.1.14 (W Hierarchy)

对于 $i \in \mathbb{N}$，一个参数化问题 (L, κ) 属于 $\mathrm{W}[i]$ 当且仅当存在常数 d，使得 (L, κ) 可以 FPT 归约到 $\mathrm{WCS}[C_{i,d}]$，$\mathrm{WH} = \bigcup_{i \geq 0} \mathrm{W}[i]$.

$$\mathrm{FPT} = \mathrm{W}[0] \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{WH}$$

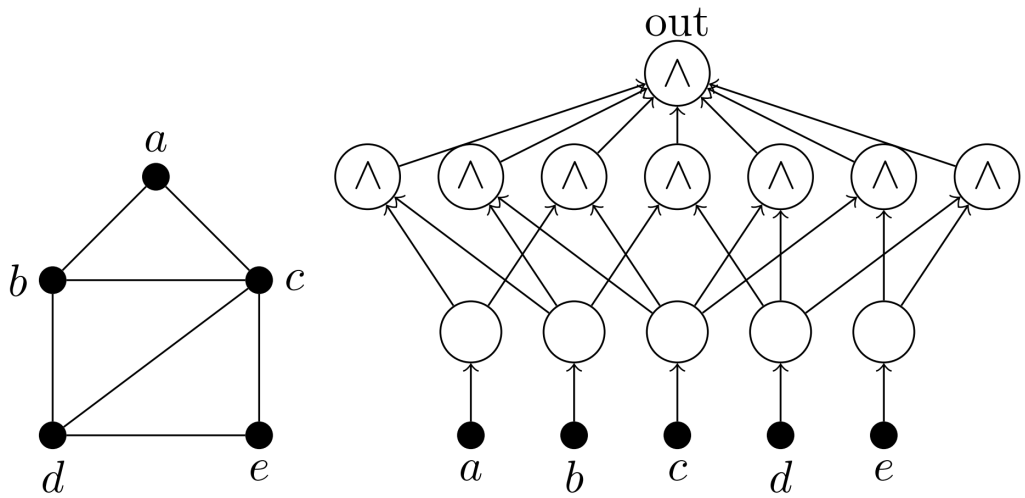
定理 3.1.15

若 $\mathrm{FPT} = \mathrm{W}[1]$，则 $\mathrm{NP} \subseteq \mathrm{DTIME}(2^{o(n)})$

CQ 的参数化复杂性

定理 3.1.16

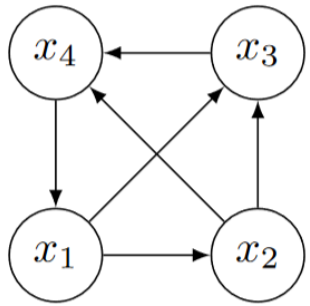
p-Clique 是 $W[1]$ 完全的



定理 3.1.17

p-CQ 判定问题是 $W[1]$ 完全的

CQ 求解困难的原因



$$\text{Answer} :- E(x_1, x_2), E(x_2, x_3), E(x_3, x_4), E(x_4, x_1), E(x_1, x_3), E(x_2, x_4)$$

- 关系间成环，求解时互相影响、依赖

无环 CQ

- 如何将 CQ 转化为图并判断其是否有环
  - 超图：每个点对应一个变量，每个关系连一条超边
  - 普通图：每个点对应一个变量，每个关系中的变量两两连边

Definition　Join Tree and Acyclic Hypergraph

Given a hypergraph $H = (V, E)$, a tree T is a *join tree* of H if

- the nodes of T are precisely the hyperedges in E and,
- for each node $v \in V$, the set of nodes of T in which v is an element forms a connected subtree of T.

Moreover, H is *acyclic* if H admits a join tree.

Definition　Acyclicity of CQs

An *acyclic conjunctive query* (ACQ) is a conjunctive query q such that its associated hypergraph H_q is acyclic.

无环 CQ 的识别

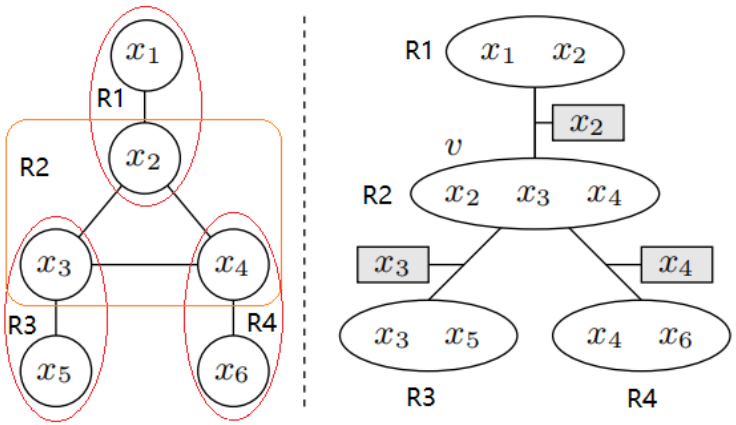
Proposition GYO Algorithm

A hypergraph $H = (V, E)$ is acyclic if and only if all of its vertices can be deleted by repeatedly applying the following two operations (in no particular order):

1. Delete a vertex that appears in at most one hyperedge.
2. Delete a hyperedge that is contained in another hyperedge.

定理 3.1.18

可以在线性时间判断一个 CQ 是否无环，并输出 join 树（若无环）



无环 CQ 的执行

- Boolean CQ(BCQ)：判断给定的 CQ 是否有解
- BCQ 比 CQ 判定问题一般

Algorithm YANNAKAKIS(q, D)

Input: A Boolean ACQ q and a database D
Output: $q(D)$

1: $T :=$ a join tree of H_q
2: $N :=$ the set of nodes of T
3: $r :=$ the root of T
4: **while** $N \neq \emptyset$ **do**
5: Choose $s \in N$ such that no child of s is in N
6: Compute $q_s(D)$
7: **if** s is a leaf of T **then**
8: $Q_s(D) := q_s(D)$
9: **else**
10: Let $s_1, \ldots, s_p$ be the children of s in T
11: $Q_s(D) := \bigcap_{i=1}^{p} \left(q_s(D) \ltimes Q_{s_i}(D)\right)$
12: $N := N - \{s\}$
13: **if** $Q_r(D) \neq \emptyset$ **then**
14: **return** `true`
15: **else**
16: **return** `false`

定理 3.1.19

Boolean Acyclic CQ(BACQ) 问题可以在 $O(|\varphi||D|\log|D|)$ 时间内解决

无环 CQ 的执行

- ACQ 的解的判定问题可在 $O(|\varphi||D|\log|D|)$ 时间内解决
- ACQ 的解的存在性问题可在 $O(|\varphi||D|\log|D|)$ 时间内解决
- ACQ 的解的求值问题
  - 查询结果可能是指数的
  - 解的枚举算法：总体多项式时间、多项式延迟
  - 自可约性：判定解的存在性 $\Rightarrow$ 多项式延迟枚举

- ACQ 以下：解的计数问题、均匀采样、随机顺序枚举....
  - free-connex：添加一条由自由变量构成的边后图仍然无环
- ACQ 以上：如果查询实际上“很接近”一棵树会怎样？
  - 树宽 (Treewidth)
  - (广义) 超树宽 ((generalized) Hypertree Width)

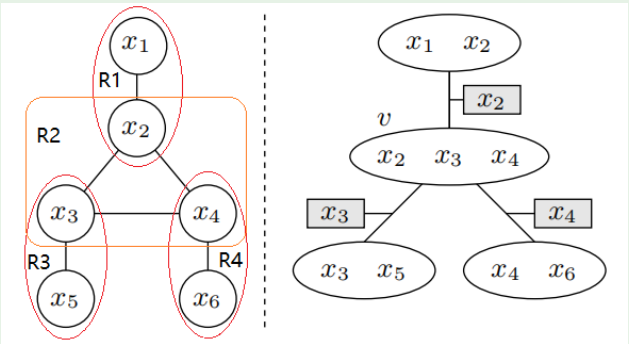
Tree Decomposition

Definition 7.2: Consider a graph $G = \langle V, E \rangle$. A tree decomposition of G is a tree structure T where each node of T is a subset of V, such that:

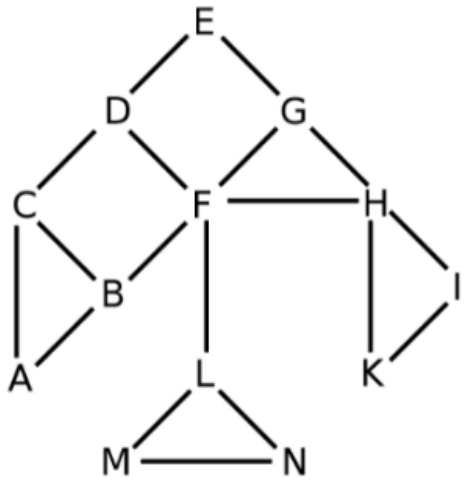
- The union of all nodes of T is V.
- For each edge $(v_1 \to v_2) \in E$, there is a node N in T such that $v_1, v_2 \in N$.
- For every vertex $v \in V$, the set of nodes of T that contain v form a subtree of T; equivalently: if two nodes contain v, then all nodes on the path between them also contain v (connectedness condition).

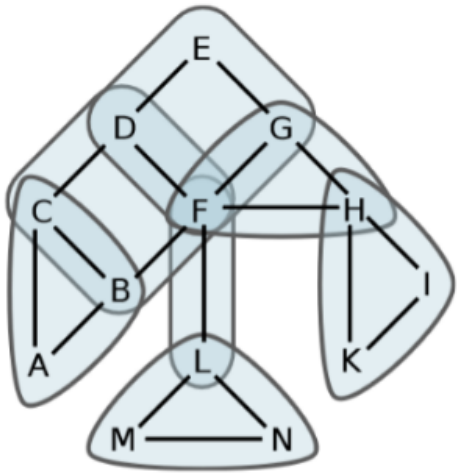
例 3.1.20 (Treewidth=2)

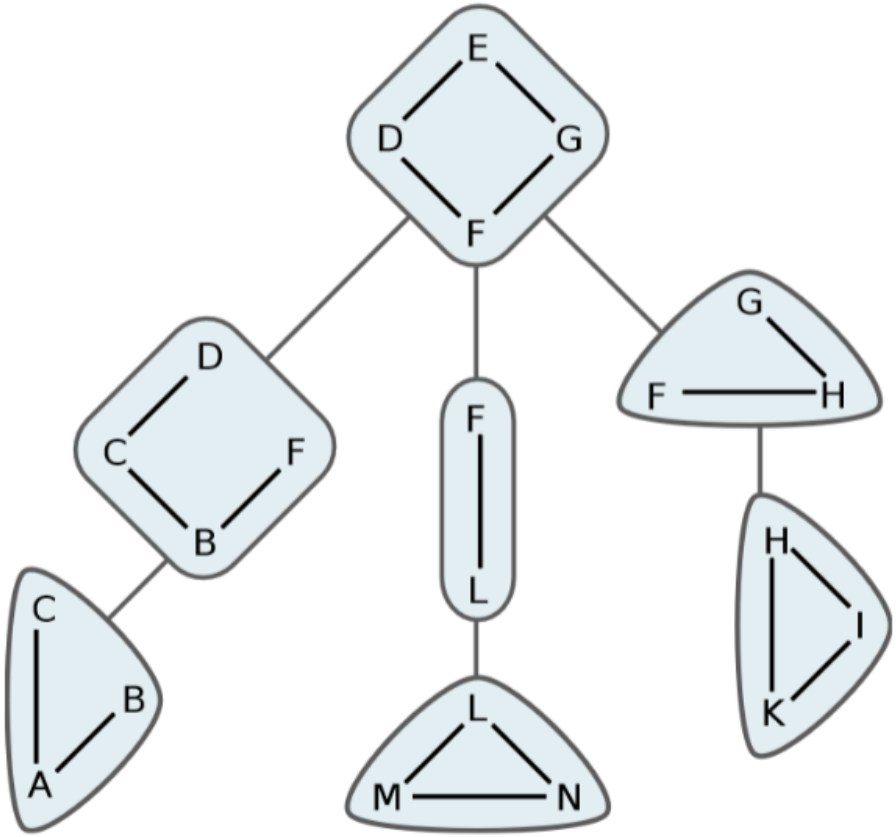
$$\begin{aligned}\varphi(x_1, x_5, x_6) \leftarrow & R_1(x_1, x_2), \\ & R_2(x_2, x_3, x_4), \\ & R_3(x_3, x_5), \\ & R_4(x_4, x_6)\end{aligned}$$



Tree Decomposition







Tree Width

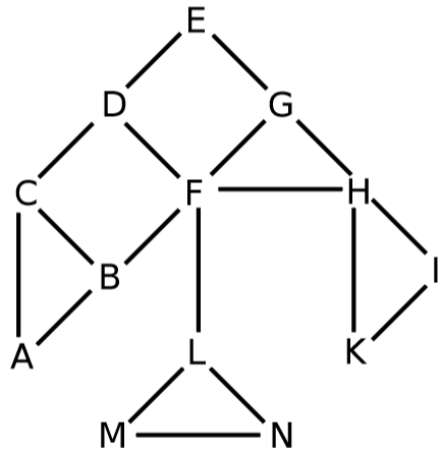
The treewidth of a graph defines how "tree-like" it is:

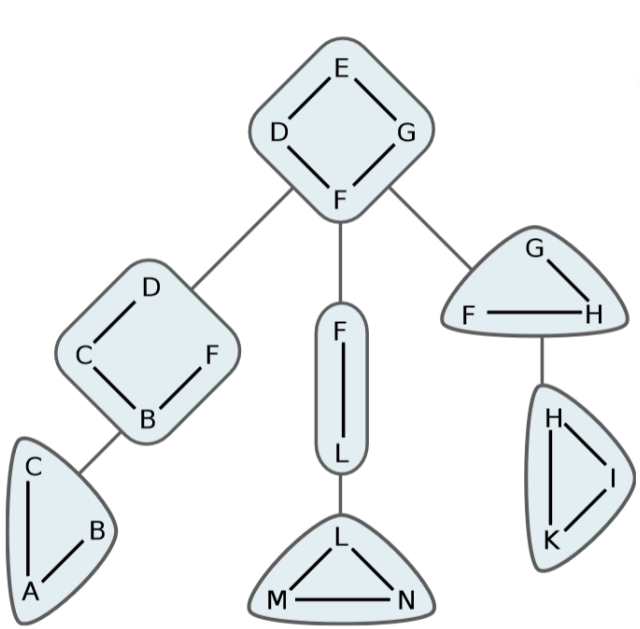
Definition 7.3: The width of a tree decomposition is the size of its largest bag minus one.
The treewidth of a graph G, denoted $\mathrm{tw}(G)$, is the smallest width of any of its tree decompositions.

Simple observations:

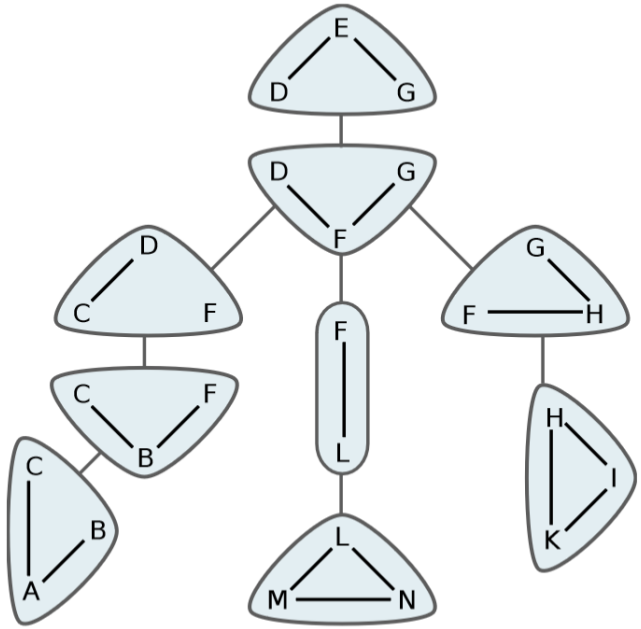
- If G is a tree, then we can decompose it into bags that contain only one edge
  $\leadsto$ trees have treewidth 1
- Every graph has at least one tree decomposition where all vertices are in one bag
  $\leadsto$ maximal treewidth = number of vertices -1

Tree Width





$\rightsquigarrow$ tree decomposition of width 3



$\rightsquigarrow$ tree decomposition of width 2 = treewidth of the example graph

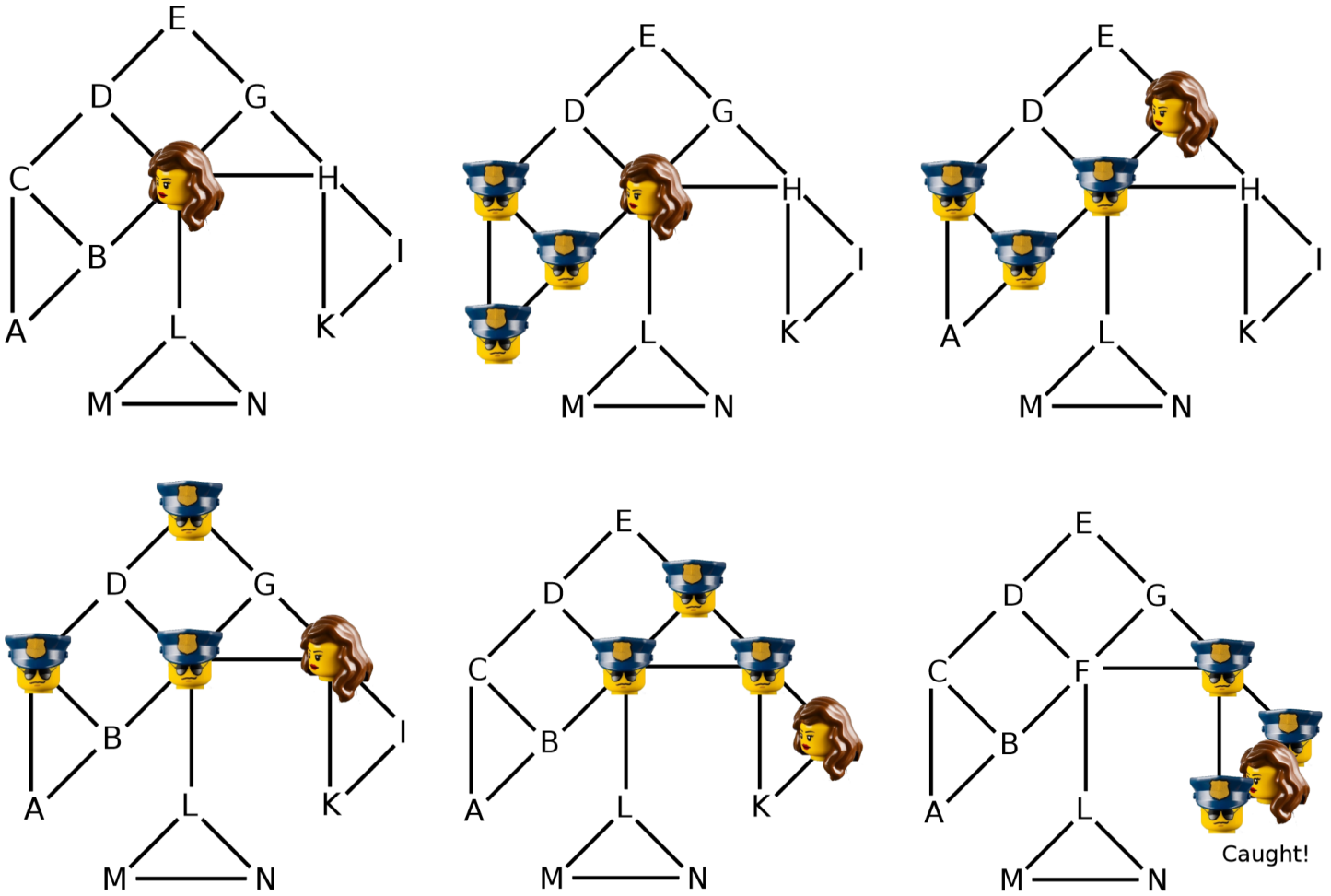
Tree Width via Games

Seymour and Thomas [1993] gave
an alternative characterisation of treewidth:

The Cops-and-Robber Game

- The game is played on a graph G
- There are k cops and one robber that may be positioned at vertices
- In the first turn, the robber places herself at an arbitrary vertex of the graph; the cops are all in a "helicopter" (i.e., not yet placed on any vertex)
- In each turn:
  - one of the cops can decide to "fly" to an arbitrary vertex in the graph
  - if the moving cop is already in the game, he is lifted from his vertex
  - before "landing" (i.e. positioning the cop at his new vertex), the target vertex is announced to the robber (the robber sees the helicopter approaching)
  - the robber can run along the edges of the graph, as far as she likes, as long as she does not use any vertex currently occupied by a cop
  - the moving cop arrives at his destination vertex
- The cops' goal is to catch the robber; the robber's goal is never to be caught

Tree Width via Games



Theorem 7.6 (Seymour and Thomas): A graph G is of treewidth $\leq k-1$ if and only if k cops have a winning strategy in the cops & robber game on G.

Summary of Tree Width

Graphs of bounded treewidth as a generalisation of (undirected) trees:

- Trees have treewidth 1
- Graphs of higher treewidth resemble trees with "thicker branches"
- It is (in theory) not hard to check if a graph has treewidth $\leq k$ for some k
- It is (in theory) not hard to answer BCQs whose primal graph has a bounded treewidth

Practically feasible only for lower treewidths

However, bounded treewidth does not generalise the notion of hypergraph acyclicity (acyclic families of hypergraphs may have unbounded treewidth)

Is there a better notion of tree-likeness for hypergraphs?

Query Width

Idea of Chekuri and Rajamaran [1997]:

- Create tree structure similar to tree decomposition
- But consider bags of query atoms instead of bags of variables
- Two connectedness conditions:
  (1) Bags that refer to a certain variable must be connected
  (2) Bags that refer to a certain query atom must be connected

Query width: least number of atoms needed in bags of a query decomposition

Theorem 8.1: Given a query decomposition for a BCQ, the query answering problem can be decided in time polynomial in the query width.

Theorem 8.2 (Gottlob et al. 1999): Deciding if a query has query width at most k is NP-complete.

In particular, it is also hard to find a query decomposition

$\rightsquigarrow$ Query answering complexity drops from NP to P ...
... but we need to solve another NP-hard problem first!

Hyper Tree Decomposition[10]

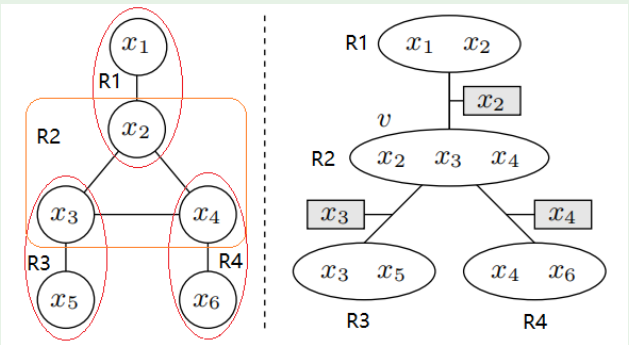
Definition 8.3: Consider a hypergraph $G = \langle V, E \rangle$. A hypertree decomposition of G is a tree structure T where each node n of T is associated with a bag of variables $B_n \subseteq V$ and with a set of edges $G_n \subseteq E$, such that:

- T with B_n yields a tree decomposition of the primal graph of G.
- For each node n of T:
  - (1) the vertices used in the edges G_n are a superset of B_n,
  - (2) if a vertex v occurs in an edge of G_n and this vertex also occurs in B_m for some node m below n in T, then $v \in B_n$.

The width to T is the largest number of edges in a set G_n.
The hypertree width of G, hw(G), is the least width of its hypertree decompositions.

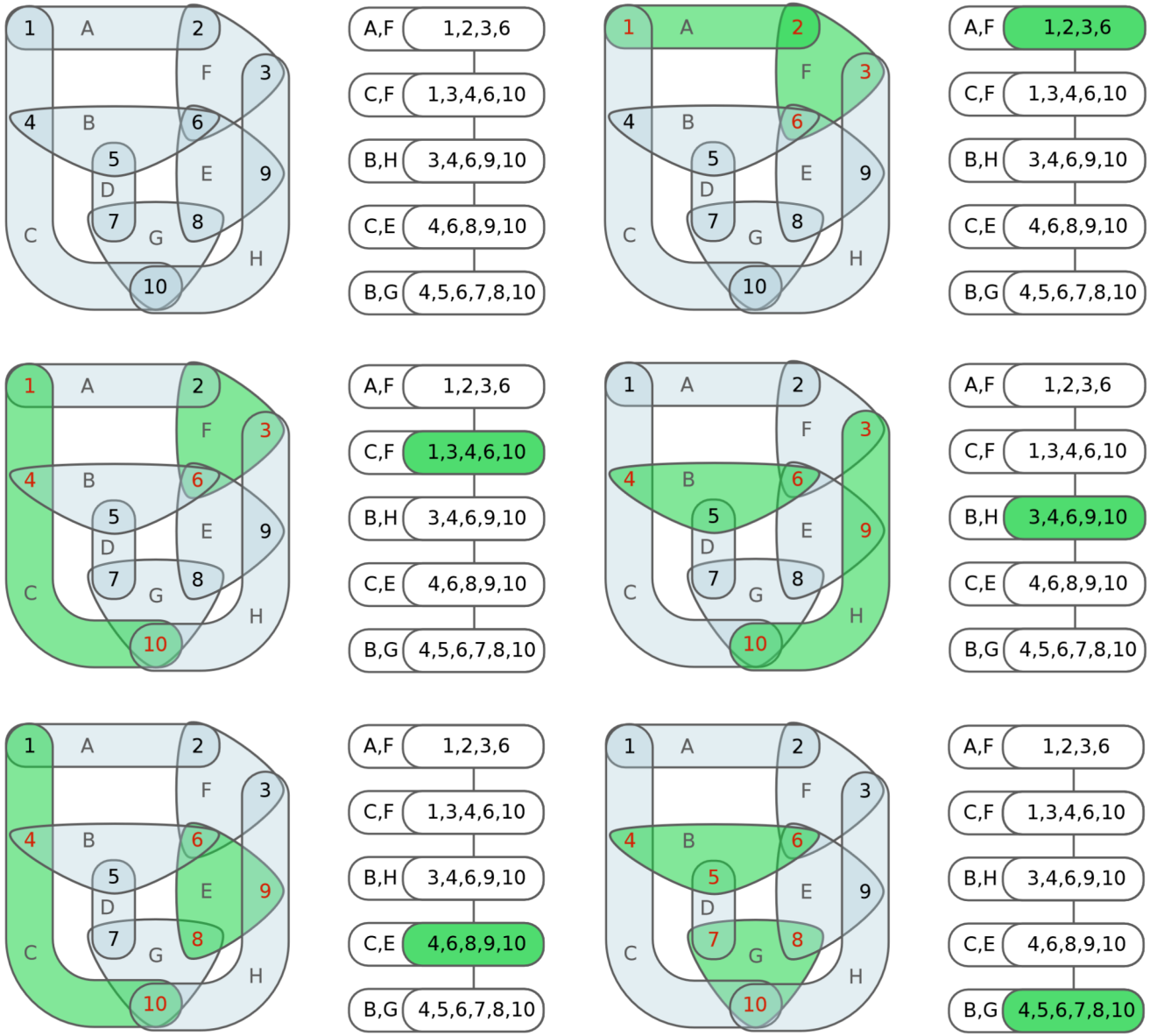
例 3.1.21 (Treewidth=2, Hyperwidth=1)

$$\varphi(x_1, x_5, x_6) \leftarrow R_1(x_1, x_2), \\ R_2(x_2, x_3, x_4), \\ R_3(x_3, x_5), \\ R_4(x_4, x_6)$$

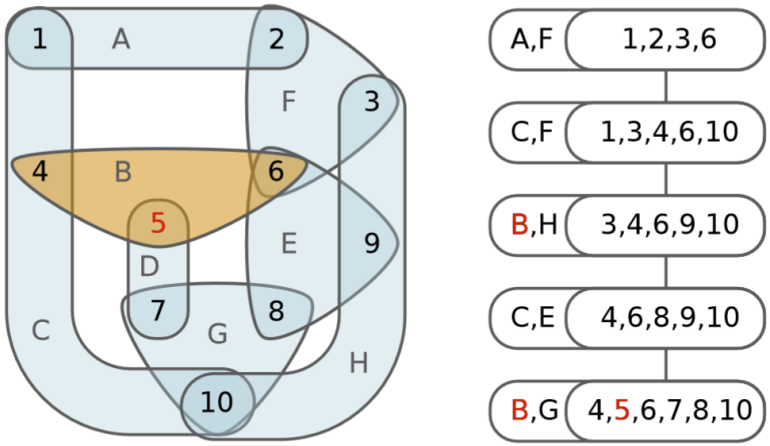


[1] (2) is the "special condition" : without it we get the generalized hypertree width

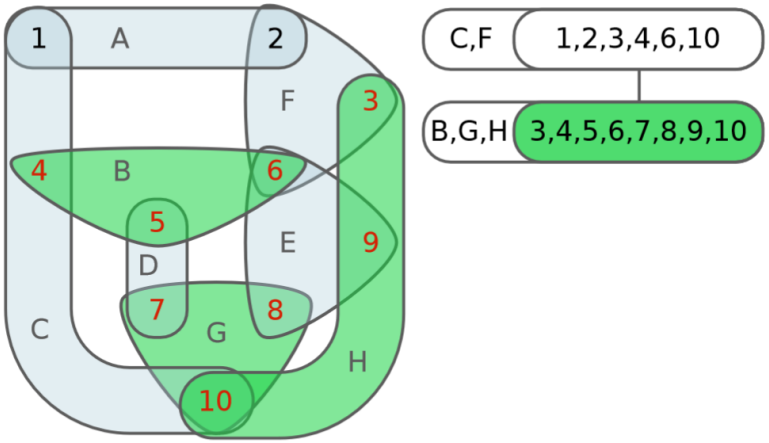
(Generalized) Hyper Tree Decomposition



(Generalized) Hyper Tree Decomposition



Special condition violated $\rightsquigarrow$ no hypertree decomposition
$\rightsquigarrow$ But generalised hypertree decomposition of width 2



Special condition satisfied $\rightsquigarrow$ hypertree decomposition of width 3

(Generalized) Hyper Tree Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width $\leq$ hypertree width $\leq$ query width
  (both inequalities might be $<$ in some cases)
- Acyclic graphs have hypertree width 1
- Deciding "query width $< k$?" is NP-complete
- Deciding "generalised hypertree width < 4?" is NP-complete
- Deciding "hypertree width $< k$?" is polynomial (LOGCFL)
- Hypertree decompositions can be computed in polynomial time if k is fixed

Theorem 8.9: For a BCQ of (generalised) hypertree width k, query answering can be decided in polynomial time, and is complete for LOGCFL.

Summary of (Hyper) Tree Width

Treewidth

Graph $G(Q) = (V, E)$

- $V :=$ variables in Q
- $(u, v) \in E$ if $u, v \in \bar{y}_i$ for some i
- $tw(G) :=$measures how close G is to a tree
- $tw(G) = 1 \leftrightarrow G$ acyclic

$$tw(Q) = tw(G(Q))$$

Hyper-Treewidth

Hyper-Graph $H(Q) = (V, \mathcal{E})$

- $V :=$ variables in Q
- $\mathcal{E} := \{\{v \mid v \in \bar{y}_i\} \mid i \in [n]\}$
- $htw(G) :=$ how close H is to acyclic
  - [Gottlob, Leone, Scarcello '02]

$$htw(Q) \leq tw(Q)$$

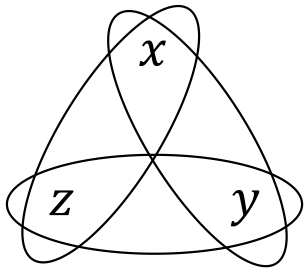
查询结果规模的上界

$$Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x)$$

设 $|R| = |S| = |T| = N$，结果的大小 $|Q|$ 会达到 N^3 吗？

- $|R \bowtie S| = O(N^2)$
- $R \bowtie S(x,y,z)$ 被 $T(z,x)$ “支配”（属性包含）
- $(R \bowtie S) \bowtie T$ 的连接结果大小为 $\max(|R \bowtie S|, |T|) = O(N^2)$

超图边覆盖问题



$$Q(x, y, z) = R(x, y) \bowtie S(y, z) \bowtie T(z, x)$$

- 超图边覆盖问题：求一个超边集 C，使得每个点（变量）都至少被一条 C 中的超边（关系）覆盖
- 显然地，$|Q| \leq \prod_{R \in C} |R|$

分数边覆盖问题

- 输入：超图 $G = (V, E)$
- 输出：权值函数 $c : E \to [0, 1]$，使得$\forall v \in V, \sum_{e_i \ni v} c(e_i) \geq 1$
- 边覆盖问题的可行解（C）也是可行的分数边覆盖（c）
  - $\forall e \in E, c(e) = \begin{cases} 1 & e \in C \\ 0 & e \notin C \end{cases}$
- 分数边覆盖问题可看作边覆盖问题条件“松弛”后的衍生问题

线性规划

- 超图 G（最小）边覆盖问题的整数线性规划：
  - $G = (V, E), |V| = n, |E| = m$
  - $V = \{v_1, \ldots, v_n\}, E = \{e_1, \ldots, e_m\}$

$$\text{minimize} \sum_{i=1}^{m} c_i$$

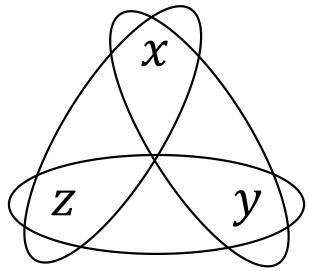
$$\text{s.t.} \sum_{e_i \ni v} c_i \geq 1 \text{ for all } v \in V$$

$$c_1, c_2, \ldots, c_m \in \{0, 1\}$$

- 松弛后的条件

$$c_1, c_2, \ldots, c_m \in [0, 1]$$

线性规划



- $V = \{x, y, z\}, E = \{(x, y), (y, z), (z, x)\}$

$$\begin{aligned} \text{minimize} \quad & c_1 + c_2 + c_3 \\ \text{s.t.} \quad & c_1 + c_2 \geq 1 \\ & c_2 + c_3 \geq 1 \\ & c_3 + c_1 \geq 1 \\ & c_1, c_2, c_3 \in \{0, 1\} \end{aligned}$$

- 松弛后：$c_1, c_2, c_3 \in [0, 1]$

AGM 界

定理 3.1.22 (AGM 界)

给定查询 $Q = \bowtie_{i=1}^{m} R_i$，构造超图 $G = (V, E)$，其中

- $V = \mathrm{var}(Q)$
- $E = \{\mathrm{var}(R_i) | 1 \leq i \leq m\}$

设 c 为 G 的一个分数边覆盖，则对任意数据库实例 $\mathcal{D}$，

$$|Q(D)| \leq \prod_{i=1}^{m} |R_i^D|^{c(\mathrm{var}(R_i^D))}$$

AGM 界

$$
\begin{aligned}
|Q(D)| &\leq \prod_{i=1}^{m} |R_i^D|^{c(\mathrm{var}(R_i))} \\
&= \prod_{i=1}^{m} 2^{c(\mathrm{var}(R_i^D))\log(|R_i^D|)} \\
&= 2^{\sum_{i=1}^{m} c(\mathrm{var}(R_i^D))\log(|R_i^D|)}
\end{aligned}
$$

- 因此，

$$
\begin{aligned}
\log|Q(D)| \leq \min \ & \sum_{i=1}^{m} \log(|R_i^D|) \cdot c_i \\
\text{s.t.} \ & \sum_{\mathrm{var}(R_i) \ni v} c_i \geq 1 \text{ for all } v \in \mathrm{var(Q)} \\
& c_1, c_2, \ldots, c_m \in [0, 1]
\end{aligned}
$$

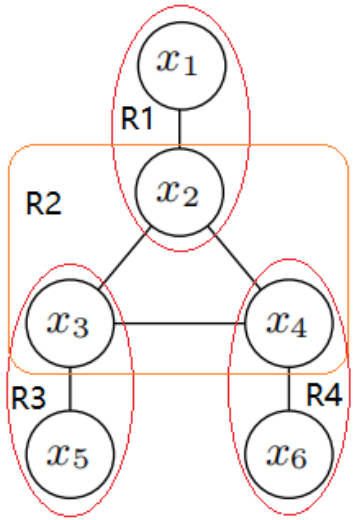
AGM 界的最优性

定理 3.1.23

给定查询 $Q = \bowtie_{i=1}^{m} R_i$，及其对应的超图 $G = (V, E)$，设 c^* 为 G 的最小分数边覆盖，则存在一个数据库实例 $\mathcal{D}$，使得

$$|Q(D)| = \prod_{i=1}^{m} |R_i^D|^{c^*(\mathrm{var}(R_i^D))}$$

超图独立集问题



$$Q(x_1, \ldots, x_6) = R_1(x_1, x_2) \bowtie R_2(x_2, x_3, x_4) \bowtie R_3(x_3, x_5) \bowtie R_4(x_4, x_6)$$

- （超图）独立集问题：求一个点集 I，使得 I 中的任意两个顶点都不在同一条（超）边上
- $\{x_1, x_5, x_6\}$ 是 G 的一个独立集
- 直观地，$\exists D, |Q(D)| \geq |\mathrm{adom}^D(x_1)| \cdot |\mathrm{adom}^D(x_5)| \cdot |\mathrm{adom}^D(x_6)|$
  - $\mathrm{adom}^D(x_i)$：D 中变量 x 的活动域（可能的取值集合）

线性规划

- 超图 G（最大）独立集问题的整数线性规划：
  - $G = (V, E), |V| = n, |E| = m$
  - $V = \{v_1, \ldots, v_n\}, E = \{e_1, \ldots, e_m\}$

$$\text{maximize} \sum_{i=1}^{n} d_i$$

$$\text{s.t.} \sum_{v_i \in e} d_i \leq 1 \text{ for all } e \in E$$

$$d_1, d_2, \ldots, d_n \in \{0, 1\}$$

- 松弛后的条件（分数独立集）

$$d_1, d_2, \ldots, d_n \in [0, 1]$$

对偶问题

- 最小分数边覆盖问题

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} c_i \\ \text{s.t.} \quad & \sum_{e_i \ni v} c_i \geq 1 \text{ for all } v \in V \\ & c_1, c_2, \ldots, c_m \in [0, 1] \end{aligned}$$

- 与最大分数独立集问题

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} d_i \\ \text{s.t.} \quad & \sum_{v_i \in e} d_i \leq 1 \text{ for all } e \in E \\ & d_1, d_2, \ldots, d_n \in [0, 1] \end{aligned}$$

- 互为对偶问题

强对偶定理

定理 3.1.24 (强对偶定理)

对于任意线性规划 P 及其对偶线性规划问题 P_D，若 P 有最优解 ρ^*，P_D 有最优解 τ^*，则

$$\rho^* = \tau^*$$

AGM 界最优性直观证明思路

- 最小分数边覆盖给出查询大小的上界

$$\forall D, |Q(D)| \leq \prod_{i=1}^{m} |R_i^D|^{c_i^*}$$

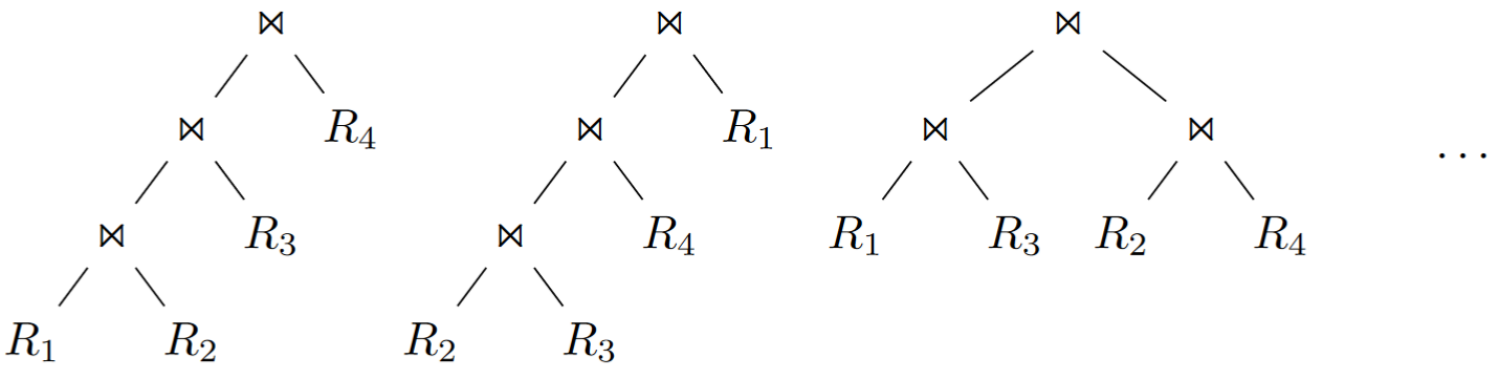
- 最大分数独立集给出查询大小的下界

$$\exists D, |Q(D)| \geq \prod_{i=1}^{n} |\mathrm{adom}^D(x_i)|^{d_i^*}$$

- 根据强对偶定理，这两个界相等

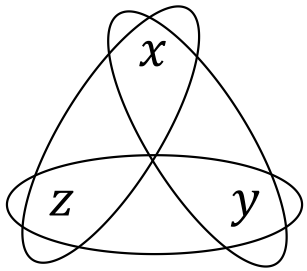
连接查询计划

$$Q = R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$$



“两两连接”查询计划树

坏情况



$$Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x)$$

- 其中

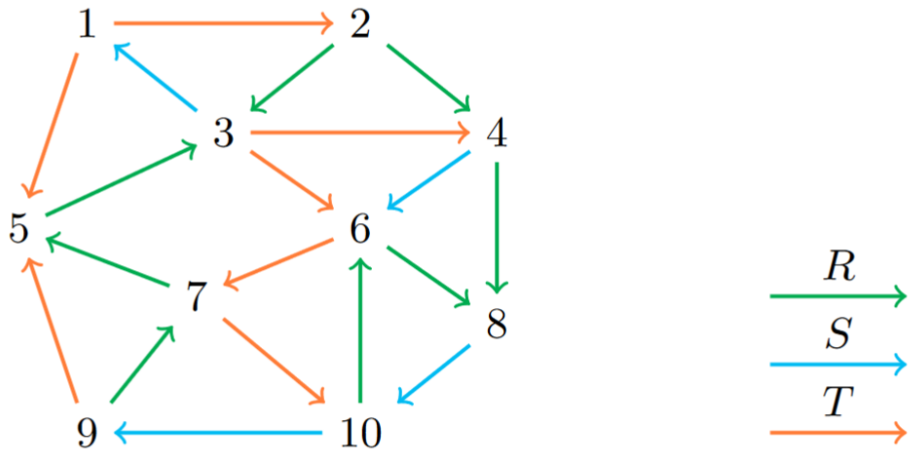
$$R = S = T = \left\{ \begin{array}{r} (1,1),(1,2),(1,3),\ldots,(1,\frac{N}{2}) \\ \\ (2,1),(3,1),\ldots,(\frac{N}{2},1) \end{array} \right\}$$

- 此时，$|R \bowtie S| = |S \bowtie T| = |T \bowtie R| = \frac{N^2}{4} = \Omega(N^2)$
- 然而，根据 AGM 界，$|R \bowtie S \bowtie T| \leq N^{1.5}$

坏情况下最优的查询算法（WCO）

- 希望找到一种数据复杂性为 $\widetilde{O}(|Q(D)|)$ 的查询执行算法
- Worst Case Optimized (WCO) Algorithm

WCO 算法：例子



$$Q(A,B,C) = R(A,B) \bowtie S(B,C) \bowtie T(C,A)$$

(1) Compute $L_1 := \pi_A(R \bowtie T)$.

(2) For each $a \in L_1$, $L_1 = \{5, 2, 6, 7, 4, 10\}$;

- compute the values b in $\pi_B(R \bowtie S)$ such that $(a, b) \in R$ and
- add the pairs (a, b) to L_2.

(3) For each $(a, b) \in L_2$, $L_2 = \{(5,3), (2,3), (2,4), (6,8), (4,8)\}$;

- compute the values c in $\pi_C(S \bowtie T)$ such that $(b, c) \in S$ and $(c, a) \in T$
- add the triples (a, b, c) to L_3.

(4) Return L_3. $L_3 = \{(5,3,1), (2,3,1)\}$.

WCO 算法：AEJoin

- Attribute Elimination Join (AEJoin)

Algorithm AEJoin(q, D)

Input: Join query q using attributes $A_1, \ldots, A_m$ and database D
Output: $q(D)$

$L_0 := \{()\}$ ▷ L_0 is the set with the empty tuple
for $i = 1, \ldots, m$ **do**
  $L_i := \emptyset$
  for each tuple $\bar{a} \in L_{i-1}$ **do**
    $V := \bigcap_{R \in \mathcal{R}(A_i)} \pi_{\{A_i\}}(R^D \ltimes \{\bar{a}\})$
    $L_i := L_i \cup (\{\bar{a}\} \times V)$
return L_m

- 将前 i 个变量的部分赋值保存在 L_i 中
- 对于 L_i 中的每个部分赋值分别计算第 $i+1$ 个变量的可能赋值
- 最后的 L_n 即为查询结果

WCO 算法：AEJoin

定理 3.1.25

对于 $\forall D$，*AEJoin* 的时间复杂度为

$$\widetilde{O}(nm \cdot |Q(D)|)$$

其中 $n = |\mathrm{var}(Q)|$ 表示 Q 中的变量数，$m = |\mathrm{atom}(Q)|$ 表示 Q 中的查询原子（关系）数，$|Q(D)|$ 表示查询结果数.

主要内容

本节内容

随意插入数据并无意义，且会使计算负担过重.

- 引入语义限制：完整性约束
- 范式：数据库模式设计指南

主要内容

动机

刻画数据库关系语义，帮助：

- 设计健壮/高效的数据库模式
- 检查/保证数据完整性
- 快速查询/查询优化算法

函数依赖/键值依赖

定义 4.1.1

设 U 是属性集合. 给定 $X, Y \subseteq U$,U 上的函数依赖记作 $X \to Y$.U 上的键值依赖记作 $X \to U$.
如果 $\forall s, t \in I, \pi_X(s) = \pi_X(t) \implies \pi_Y(s) = \pi_Y(t)$, 则称 I 满足 $X \to Y$, 记作 $I \models X \to Y$.

借助函数依赖的关系分解

给定数据库实例 I, 属性集合 U,I 满足 $X \to Y$, 记 $Z = U - XY$. 有 $I = \pi_{XY}(I) \bowtie \pi_{XZ}(I)$.

逻辑蕴含

假设数据库实例满足函数依赖集合 $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$，能否快速推断是否也满足 σ?

逻辑蕴含

给定属性集合 Σ, Γ, 如果对任意数据库实例 I, 当 $I \models \Sigma$ 时都有 $I \models \Gamma$ 时, 称 Σ 蕴含 Γ, 记作 $\Sigma \models \Gamma$.$\Sigma \models \Gamma$ 且 $\Gamma \models \Sigma$ 时, 称它们等价, 记作 $\Sigma \equiv \Gamma$.

例 4.1.2

设 $\Sigma_1 = \{A \to C, B \to C, CD \to E\}$, 有
$\Sigma_1 \models AD \to E, \Sigma_1 \models CDE \to C, \emptyset \models CDE \to C$.

逻辑蕴涵

函数依赖闭包

$\Sigma^* = \{X \to Y | \Sigma \models X \to Y\}$

属性的闭包

设 X 是某个属性, 它的闭包被定义为 $X^* = \{A | \Sigma \models X \to A\}$.

逻辑蕴涵

下列算法能够计算属性闭包

Input: a set Σ of fd's and a set X of attributes.

Output: the closure X^* of X under Σ.

1. $unused := \Sigma$;
2. $closure := X$;
3. repeat until no further change:
   if $W \to Z \in unused$ and $W \subseteq closure$ then
   i. $unused := unused - \{W \to Z\}$;
   ii. $closure := closure \cup Z$
4. output *closure*.

练习: 把算法优化至线性时间.

引理 4.1.3

$\Sigma \models X \to Y$ 当且仅当 $Y \subseteq X^*$.(根据定义可立即证明)

公理系统

小故事: 猫和老鼠

公理系统

- 命题与字符串之间存在对应关系. 例: 可以把命题 $a < b, b < c$ 表示成字符串 $s_1 =$ "$a < b$", $s_2 =$ "$b < c$".
- "公理": 一种字符串改写规则. 例: 根据不等号的传递性, 可以把字符串 s_1, s_2 改写为 $s_3 =$ "$a < c$".
- "证明": 按照公理规则不断改写字符串, 所产生的字符串序列. 某命题可被公理体系证明, 指它的字符串表示可以在上述某种字符串序列中找到. 例, 给定字符串 s_1, s_2, 在不等号传递性下可以证明 s_3.

公理的正确性与完备性

- 正确性 (soundness): 一切按照公理规则推出的命题都是真的.
- 完备性 (completeness): 一切真命题都能被公理规则推出.

Armstrong 公理系统

FD1: (reflexivity) If $Y \subseteq X$, then $X \to Y$.

FD2: (augmentation) If $X \to Y$, then $XZ \to YZ$.

FD3: (transitivity) If $X \to Y$ and $Y \to Z$, then $X \to Z$.

定理 4.1.4

Armstrong 公理系统是正确且完备的.

多值依赖 (Multivalued Dependency, mvd)

表能够分解为两个子表的连接.

定义 4.1.5

对于 $X, Y \subseteq U$, 数据库实例 I 满足多值依赖 $X \twoheadrightarrow Y$, 如果 $I = \pi_{XY}(I) \bowtie \pi_{X(U-Y)}(I)$.

例 4.1.6

Theater $\twoheadrightarrow$ Screen, Title

Showings	*Theater*	*Screen*	*Title*	*Snack*
	Rex	1	The Birds	coffee
	Rex	1	The Birds	popcorn
	Rex	2	Bladerunner	coffee
	Rex	2	Bladerunner	popcorn
	Le Champo	1	The Birds	tea
	Le Champo	1	The Birds	popcorn
	Cinoche	1	The Birds	Coke
	Cinoche	1	The Birds	wine
	Cinoche	2	Bladerunner	Coke
	Cinoche	2	Bladerunner	wine
	Action Christine	1	The Birds	tea
	Action Christine	1	The Birds	popcorn

连接依赖 (Join Dependency, jd)

表能够分解为子表的连接.

定义 4.1.7

数据库实例 I 满足连接依赖 $\bowtie [X_1, \ldots, X_n]$, 如果 $I = \bowtie_{j=1}^{n} \{\pi_{X_j}(I)\}$.

例 4.1.8

⋈[{Theater,Screen,Title},{Theater, Snack}]

Showings

Theater	*Screen*	*Title*	*Snack*
Rex	1	The Birds	coffee
Rex	1	The Birds	popcorn
Rex	2	Bladerunner	coffee
Rex	2	Bladerunner	popcorn
Le Champo	1	The Birds	tea
Le Champo	1	The Birds	popcorn
Cinoche	1	The Birds	Coke
Cinoche	1	The Birds	wine
Cinoche	2	Bladerunner	Coke
Cinoche	2	Bladerunner	wine
Action Christine	1	The Birds	tea
Action Christine	1	The Birds	popcorn

连接依赖 (Join Dependency, jd)

表能够分解为子表的连接.

定义 4.1.9

数据库实例 I 满足连接依赖 $\bowtie [X_1, \ldots, X_n]$, 如果 $I = \bowtie_{j=1}^{n} \{\pi_{X_j}(I)\}$.

例 4.1.10

$\bowtie$[{Snack,Distributor,Price},{Distributor, Theater},{Snack,Theater}]

SDT	*Snack*	*Distributor*	*Price*	*Theater*
	coffee	Smart	2.35	Rex
	coffee	Smart	2.35	Le Champo
	coffee	Smart	2.35	Cinoche
	coffee	Leclerc	2.60	Cinoche
	wine	Smart	0.80	Rex
	wine	Smart	0.80	Cinoche
	popcorn	Leclerc	5.60	Cinoche

连接依赖

定理 4.1.11

不存在连接依赖的公理体系.

但是存在同时含有多值依赖和函数依赖的公理体系.

MVD0: (complementation) If $X \twoheadrightarrow Y$, then $X \twoheadrightarrow (U - Y)$.

MVD1: (reflexivity) If $Y \subseteq X$, then $X \twoheadrightarrow Y$.

MVD2: (augmentation) If $X \twoheadrightarrow Y$, then $XZ \twoheadrightarrow YZ$.

MVD3: (transitivity) If $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$, then $X \twoheadrightarrow (Z - Y)$.

FMVD1: (conversion) If $X \to Y$, then $X \twoheadrightarrow Y$.

FMVD2: (interaction) If $X \twoheadrightarrow Y$ and $XY \to Z$, then $X \to (Z - Y)$.

The Chase

思想: 如果数据语义得到保证, 那么可以用来帮助简化查询.

例 4.1.12

	A	B	C	D
T	w	x	y	z′
	w′	x	y′	z
t	w	x	y	z

(a) The tableau query (T, t)

	A	B	C	D
T′	w	x	y	z′
	w′	x	y′	z
	w	x	y′	z
t	w	x	y	z

(b) (One) result of applying the jd $\bowtie[AB, BCD]$

	A	B	C	D
T″	w	x	y	z′
	w′	x	y	z
	w	x	y	z
t	w	x	y	z

(c) Result of applying the fd $A \to C$

最终得到 $(T, t) \equiv (\{t\}, t)$.

The Chase

在数据库实例满足某些约束的情况下，查询等价问题可能会变得更容易.

定义 4.1.13 (满足依赖的数据库实例)

$sat(\Sigma) = \{\mathbf{I} | \mathbf{I} \models \Sigma\}$.

定义 4.1.14 (查询在依赖下等价)

称 q_1 与 q_2 在依赖 Σ 下等价，当且仅当 $\forall \mathbf{I} \in sat(\Sigma), q_1(\mathbf{I}) \equiv q_2(\mathbf{I})$，记作 $q_1 \equiv_\Sigma q_2$.

Chase: 利用依赖下的查询改写规则，不断简化查询.

The Chase

两条改写规则

假设变量之间有某种顺序 (如:x<y). 查询为表格查询 (T, t).

- *fd rule:* 假设 $X \to Y$. 元组 $u, v \in T, \pi_X(u) = \pi_X(v), \pi_A(u) \neq \pi_A(v)$. 设 $x = u[A], y = v[A], x < y$, 把所有的 y 替换为 x, 剩下的不变.
- *jd rule:* 假设 $\bowtie [X_1, \ldots, X_n], u_1, dots, u_n \in T$, 某个 $u \in R$ 是自由元组, 且 $\forall i, \pi_{X_i}(u_i) = \pi_{X_i}(u)$. 在 T 中加入元组 u, 其余不变.

Chasing sequence

按照改写规则改写的查询可以排成一个序列

$$(T, t) = (T_0, t_0), \ldots, (T_i, t_i), \ldots$$

其中序列的某个查询可以通过上一个查询按照某种规则改写得到. 如果序列长度有限, 那么最后一个查询叫做改写的**结果**.

由于上述两条规则不引入新变量, 因此序列长度必然有限.

The Chase

Church-Rosser 性质

同一依赖限制下, 无论每步选取何种改写规则, 最终得到的查询改写结果都相同.

定义 4.1.15 (Chase)

如果 (T, t) 是 R 上的表格查询,Σ 是 fd 或者 jd 的属性集合, 那么 $chase(T, t, \Sigma)$ 是从 (T, t) 出发, 任意一种改写序列的结果.

The Chase

理想情况: 给定依赖集合, 每一步随机选取规则并化简, 直到序列终止, 之后利用其他技术 (同态定理/基数估计/...) 继续化简.

复杂性事实

当查询与数据库模式作为输入时, 决定能否对查询应用 jd 规则是 NP-难的. 但当数据库模式固定时, 问题是多项式时间可解决的.

利用依赖进行查询优化

另一个例子.

	A	B	C	D
T	w'	x	y'	z'
	w	x'	y'	z'
	w	x''	y''	z
t	w	x	y	z

(a) The tableau query (T, t) corresponding to q

	A	B	C	D
T''	w'	x	y'	z'
	w	x'	y'	z
	w	x''	y'	z
t''	w	x	y	z

(c) The tableau query $(T'', t'') = chase(T', t', \{B \to D, D \to C\})$

	A	B	C	D
T'	w'	x	y'	z'
	w	x'	y'	z'
	w	x''	y''	z
	w	x'	y''	z
	w	x''	y'	z'
t'	w	x	y	z

(b) The tableau query $(T', t') = chase(T, t, \{\bowtie[AB, ACD]\})$

	A	B	C	D
T'''	w'	x	y'	z
	w	x'	y'	z
t'''	w	x	y	z

(d) The tableau query $(T''', t''') = min(T'', t'')$

主要内容

引入

Movies

Title	*Director*	*Actor*
The Birds	Hitchcock	Hedren
The Birds	Hitchcock	Taylor
Bladerunner	Scott	Hannah
Apocalypse Now	Coppola	Brando

Showings

Theater	*Screen*	*Title*	*Snack*
Rex	1	The Birds	coffee
Rex	1	The Birds	popcorn
Rex	2	Bladerunner	coffee
Rex	2	Bladerunner	popcorn
Le Champo	1	The Birds	tea
Le Champo	1	The Birds	popcorn
Cinoche	1	The Birds	Coke
Cinoche	1	The Birds	wine
Cinoche	2	Bladerunner	Coke
Cinoche	2	Bladerunner	wine
Action Christine	1	The Birds	tea
Action Christine	1	The Birds	popcorn

$$Showings[Title] \subseteq Movies[Title]$$

包含依赖的公理化

IND1: (reflexivity) $R[X] \subseteq R[X]$.

IND2: (projection and permutation) If $R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$, then $R[A_{i_1}, \ldots, A_{i_k}] \subseteq S[B_{i_1}, \ldots, B_{i_k}]$ for each sequence $i_1, \ldots, i_k$ of distinct integers in $\{1, \ldots, m\}$.

IND3: (transitivity) If $R[X] \subseteq S[Y]$ and $S[Y] \subseteq T[Z]$, then $R[X] \subseteq T[Z]$.

定理 4.2.1

上述公理是正确 *(sound)* 且完备 *(complete)* 的.

包含依赖的公理化

想法: 比照 Armstrong 公理体系, 设计高效算法快速判定包含依赖.

定理 4.2.2

设 Σ 是 $\mathbf{R}$ 上的包含依赖集合, 且 $R_a[A_1, \ldots, A_m] \subseteq R_b[B_1, \ldots, B_m]$, 则 $\Sigma \models R_a[A_1, \ldots, A_m] \subseteq R_b[B_1, \ldots, B_m] \Leftrightarrow \exists R_{i_1}[\overrightarrow{C_1}], \ldots, R_{i_k}[\overrightarrow{C_k}]$ 使得

- $R_{i_j} \in \mathbf{R}, \forall j \in [1, k]$;
- $\overrightarrow{C_j}$ 是 R_{i_j} 中 m 个不同的属性序列;
- $R_{i_1}[\overrightarrow{C_1}] = R_a[A_1, \ldots, A_m]$;
- $R_{i_k}[\overrightarrow{C_k}] = R_b[B_1, \ldots, B_m]$;
-

包含依赖的公理化

想法: 比照 Armstrong 公理体系, 设计高效算法快速判定包含依赖.

ALGORITHM 9.1.6

Input: A set Σ of ind's over $\mathbf{R}$ and ind $R_a[A_1, \ldots, A_m] \subseteq R_b[B_1, \ldots, B_m]$.

Output: Determine whether $\Sigma \models R_a[A_1, \ldots, A_m] \subseteq R_b[B_1, \ldots, B_m]$.

Procedure: Build a set $\mathcal{E}$ of expressions of the form $R_i[C_1, \ldots, C_m]$ as follows:

1. $\mathcal{E} := \{R_a(A_1, \ldots, A_m)\}$.
2. Repeat until $R_b[B_1, \ldots, B_m] \in \mathcal{E}$ or no change possible:
   If $R_i[C_1, \ldots, C_m] \in \mathcal{E}$ and

$$R_i[C_1, \ldots, C_m] \subseteq R_j[D_1, \ldots, D_m]$$

   can be derived from an ind of Σ by one application of IND2, then insert $R_j[D_1, \ldots, D_m]$ into $\mathcal{E}$.
3. If $R_b[B_1, \ldots, B_m] \in \mathcal{E}$ then return *yes*; else return *no*. ■

一个坏消息

用公理化方法推断包含依赖是 PSPACE-难的.

主要内容

主要内容

引入

Links	*Line*	*Station*	*Next Station*
	4	*St.-Germain*	*Odeon*
	4	*Odeon*	*St.-Michel*
	4	*St.-Michel*	*Chatelet*
	1	*Chatelet*	*Louvre*
	1	*Louvre*	*Palais-Royal*
	1	*Palais-Royal*	*Tuileries*
	1	*Tuileries*	*Concorde*
	9	*Pont de Sevres*	*Billancourt*
	9	*Billancourt*	*Michel-Ange*
	9	*Michel-Ange*	*Iena*
	9	*Iena*	*F. D. Roosevelt*
	9	*F. D. Roosevelt*	*Republique*
	9	*Republique*	*Voltaire*

考虑如下查询:

- 从 Odeon 出发, 可到达哪些车站?
- 从 Odeon 出发, 能乘坐哪些线路?
- 能从 Odeon 到 Chatlet 吗?
- 任意车站之间都能互相到达吗?
- 线路图内有环吗?

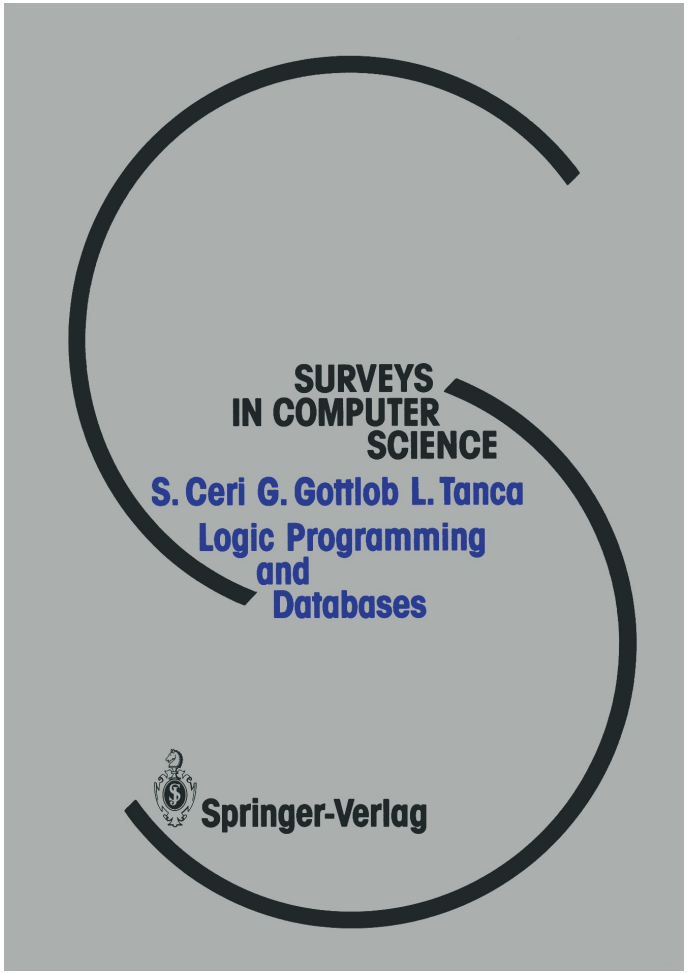
引入

之后会证明, 上述查询无法用关系代数表达.
原因: 无法表达递归.

动机: 演绎数据库

给定基本事实和推理规则, 自动完成推理 (程序自动分析, 早期人工智能等).

- 外延数据库 (extensional database,edb): 给定的基本事实.
- 规则集合: 定义推理的规则.
- 内涵数据库 (intensional database,idb): 根据规则推断出的事实.



动机: 演绎数据库

FATHER	
FATHER	*CHILD*
john	*jeff*
jeff	*margaret*
john	*anthony*
anthony	*bill*
anthony	*janet*

MOTHER	
MOTHER	*CHILD*
margaret	*annie*
mary	*jeff*
claire	*bill*
janet	*paul*

$ancestor(X,Y) :- parent(X,Y).$
$ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).$

ANCESTOR	
ANCESTOR	*DESCENDENT*
john	*jeff*
jeff	*margaret*
margaret	*annie*
john	*anthony*
anthony	*bill*
anthony	*janet*
mary	*jeff*
claire	*bill*
janet	*paul*
john	*margaret*
mary	*margaret*
jeff	*annie*
john	*bill*
john	*janet*
anthony	*paul*
john	*annie*
mary	*annie*
john	*paul*

语法

定义 5.1.1 (Datalog 语法)

Datalog 规则形式如下:

$$\underbrace{R_1(u_1)}_{\text{查询头}} \leftarrow \underbrace{R_2(u_2), \ldots, R_n(u_n)}_{\text{查询体}}$$

其中 R_i 是关系名,u_i 是自由元组. Datalog 程序指上述规则的有限集合.

语法

定义 5.1.2 (实例化)

实例化: 给定赋值函数 ν, 把 Datalog 规则中
$R_1(u_1) \leftarrow R_2(u_2), \ldots, R_n(u_n)$ 的所有自由变元 x 替换为 $\nu(x)$:

$$R_1(\nu(u_1)) \leftarrow R_2(\nu(u_2)), \ldots, R_n(\nu(u_n))$$

定义 5.1.3 (idb,edb,schema)

给定 Datalog 程序 P:

- idb(P): 出现在某规则头部的关系集合
- edb(P): 所有只出现在规则体的关系集合
- schema(P) = idb(P) ∪ edb(P)

语法

回顾开头的问题:

1. 从 Odeon 出发, 可到达哪些车站?
2. 从 Odeon 出发, 能乘坐哪些线路?
3. 能从 Odeon 到 Chatlet 吗?

解 5.1.4

查询 P_{metro}:

$$
\begin{aligned}
St\_Reachable(x, x) &\leftarrow \\
St\_Reachable(x, y) &\leftarrow St\_Reachable(x, z), Links(u, z, y) \\
Li\_Reachable(x, u) &\leftarrow St\_Reachable(x, z), Links(u, z, y) \\
Ans\_1(y) &\leftarrow St\_Reachable(Odeon, y) \\
Ans\_2(u) &\leftarrow Li\_Reachable(Odeon, u) \\
Ans\_3() &\leftarrow St\_Reachable(Odeon, Chatelet)
\end{aligned}
$$

语法

对于查询 P_{metro}:

$$
\begin{aligned}
St\_Reachable(x, x) &\leftarrow \\
St\_Reachable(x, y) &\leftarrow St\_Reachable(x, z), Links(u, z, y) \\
Li\_Reachable(x, u) &\leftarrow St\_Reachable(x, z), Links(u, z, y) \\
Ans\_1(y) &\leftarrow St\_Reachable(Odeon, y) \\
Ans\_2(u) &\leftarrow Li\_Reachable(Odeon, u) \\
Ans\_3() &\leftarrow St\_Reachable(Odeon, Chatelet)
\end{aligned}
$$

其中:

$$edb(P_{metro}) = \{Links\},$$

$$idb(P_{metro}) = \{St\_Reachable, Li\_Reachable, Ans\_1, Ans\_2, Ans\_3\}$$

对第二个规则的实例化为:

$$
\begin{aligned}
St\_Reachable(Odeon, Louvre) \leftarrow & St\_Reachable(Odeon, Chatelet), \\
& Links(1, Chatelet, Louvre)
\end{aligned}
$$

语义: 模型论解释

把 Datalog 程序看做一阶逻辑语句集合.

$$\rho : R_1(u_1) \leftarrow R_2(u_2), \ldots, R_n(u_n)$$

相当于约束

$$\Sigma_\rho : \forall x_1, \ldots, x_m(R_1(u_1) \leftarrow R_2(u_2) \wedge \cdots \wedge R_n(u_n))$$

对 Datalog 程序 $P = \{\rho_1, \ldots, \rho_n\}$, 定义 $\Sigma_P = \bigwedge_i \Sigma_{\rho_i}$.

定义 5.1.5 (模型)

Datalog 程序 P 的一个**模型**指的是满足 Σ_P 的数据库模式 sch(P) 上某实例.

程序 P 在某 edb(P) 实例 $\mathbf{I}$ 上的语义指 P 上包含 $\mathbf{I}$ 的 (基数) 最小模型.

语义: 模型论解释

例 5.1.6 (本节开头查询的结果)

Links	*Line*	*Station*	*Next Station*
	4	*St.-Germain*	*Odeon*
	4	*Odeon*	*St.-Michel*
	4	*St.-Michel*	*Chatelet*
	1	*Chatelet*	*Louvre*
	1	*Louvre*	*Palais-Royal*
	1	*Palais-Royal*	*Tuileries*
	1	*Tuileries*	*Concorde*
	9	*Pont de Sevres*	*Billancourt*
	9	*Billancourt*	*Michel-Ange*
	9	*Michel-Ange*	*Iena*
	9	*Iena*	*F. D. Roosevelt*
	9	*F. D. Roosevelt*	*Republique*
	9	*Republique*	*Voltaire*

$$
\begin{aligned}
St\_Reachable(x, x) &\leftarrow \\
St\_Reachable(x, y) &\leftarrow St\_Reachable(x, z), Links(u, z, y) \\
Li\_Reachable(x, u) &\leftarrow St\_Reachable(x, z), Links(u, z, y) \\
Ans\_1(y) &\leftarrow St\_Reachable(Odeon, y) \\
Ans\_2(u) &\leftarrow Li\_Reachable(Odeon, u) \\
Ans\_3() &\leftarrow St\_Reachable(Odeon, Chatelet)
\end{aligned}
$$

Ans_1	*Station*
	Odeon
	St.-Michel
	Chatelet
	Louvres
	Palais-Royal
	Tuileries
	Concorde

Ans_2	*Line*
	4
	1

Ans_3
$\langle\rangle$

语义: 模型论解释

模型论解释并未给出求解 Datalog 查询的方法.

- 如何判断某 Datalog 程序的模型论语义是否存在?
- 如何求解 Datalog 查询?

观察

只需考虑 $adom(P, \mathbf{I})$. 记 $\mathbf{B}(P, \mathbf{I})$ 是 sch(P) 上的实例, 且满足:

(1) $\forall R \in edb(P), R(u) \in \mathbf{B}(P, \mathbf{I}) \Leftrightarrow R(u) \in \mathbf{I}$

(2) $\forall R \in idb(P), R(u) \in adom(P, \mathbf{I}) \Rightarrow R(u) \in \mathbf{B}(P, \mathbf{I})$

则 $\mathbf{B}(P, \mathbf{I})$ 是 P 上包含 $\mathbf{I}$ 的模型.

证明留作练习.

语义: 模型论解释

定理 5.1.7

设 P 是 *Datalog* 程序,$\mathbf{I}$ 是 $edb(P)$ 上的实例,$\mathcal{X}$ 是 P 上包含 $\mathbf{I}$ 的模型集合. 则:

- $\cap\mathcal{X}$ 是 P 上包含 $\mathbf{I}$ 的最小模型, 从而定义了 $P(\mathbf{I})$ 的语义.
- $adom(P(\mathbf{I})) \subseteq adom(P, \mathbf{I})$.
- $\forall R \in edb(P), P(\mathbf{I})(R) = \mathbf{I}(R)$.

证明留作练习.

朴素 (但低效) 的算法

(在有限步内) 枚举 $\mathbf{B}(P, \mathbf{I})$ 的子集, 选出 P 上包含 $\mathbf{I}$ 的最小模型, 作为查询的答案输出.

为什么是最小模型?

如何理解数据库中不存在的记录, 不存在/不确定?

封闭世界假设 (CWA)

数据库准确记录了全部的信息. 如果 p 被记录或推断出则认为 p, 否则认为 $\neg p$.

最小模型恰好满足封闭世界假设. 但是该假设本身可能出现问题:

例 5.1.8 (封闭世界假设下的或与否定)

某数据库记录为 $\{p \vee q\}$, 那么 (由于 p, q 不在数据库中且无法推断出) 可以推断 $\neg p, \neg q$ 成立. 但是 $\{p \vee q, \neg p, \neg q\}$ 产生了不一致.

Herbrand 域

语义: 不动点解释

我们试图找到高效求解 Datalog 的方法.

定义 5.1.9 (直接结果)

设 $\mathbf{K}$ 是数据库实例, 且有规则

$$P : A \leftarrow A_1, \ldots, A_n$$

若 $A_i \in \mathbf{K}$, 则称 A 是 $\mathbf{K}, P$ 上的直接结果. 称 T_P 为直接结果算符, 定义 $T_P(\mathbf{K})$ 为所有 $\mathbf{K}, P$ 上直接结果的集合.

语义: 不动点解释

对于算符 T, 称:

- T 单调, 如果 $\forall \mathbf{I}, \mathbf{J}, \mathbf{I} \subseteq \mathbf{J} \implies T(\mathbf{I}) \subseteq T(\mathbf{J})$.
- $\mathbf{K}$ 是 T 的不动点, 如果 $T(\mathbf{K}) = \mathbf{K}$.

引理 5.1.10

设 P 是 *Datalog* 程序,

(1) 算符 T_P 是单调的.

(2) 数据库实例 $\mathbf{K}$ 是 Σ_P 的模型当且仅当 $T_P(\mathbf{K}) \subseteq \mathbf{K}$.

(3) T_P 的每个不动点都是 Σ_P 的模型, 但反之未必成立.

语义: 不动点解释

定理 5.1.11

给定 $P, \mathbf{I}, T_P$ 存在包含 $\mathbf{I}$ 的最小不动点 $\mathbf{K}$, 且 $\mathbf{K} = P(\mathbf{I})$.

证明.

首先证明 $P(\mathbf{I})$ 是 T_P 的不动点:

- 因为 $P(\mathbf{I})$ 是 P 的一个模型, 因此 $T_P(P(\mathbf{i})) \subseteq P(\mathbf{I})$.
- 由于 T_P 是单调的, 因此 $T_P(T_P(\mathbf{I})) \subseteq T_P(P(\mathbf{I}))$, 因此 $T_P(P(\mathbf{I}))$ 是 P 的一个模型. 又由于 $\mathbf{I} \subseteq T_P(P(\mathbf{I}))$, 因此 $T_P(P(\mathbf{I}))$ 是 P 上包含 $\mathbf{I}$ 的一个模型. 由于 $P(\mathbf{I})$ 最小, 因此 $P(\mathbf{I}) \subseteq T_P(P(\mathbf{I}))$.

注意到 T_P 包含 $\mathbf{I}$ 的不动点 $\mathbf{K}$ 均包含 $P(\mathbf{I})$, 因此 $P(\mathbf{I})$ 是满足上述要求的最小不动点. □

语义: 不动点解释

给定 edb(P) 上的实例 $\mathbf{I}$, 我们有

$$\mathbf{I} \subseteq T_P(\mathbf{I}) \subseteq T_P^2(\mathbf{I}) \subseteq \cdots \subseteq \mathbf{B}(P, \mathbf{I})$$

由于 $\mathbf{B}(P, \mathbf{I})$ 是有限的, 因此序列 $\{T_P^i(\mathbf{I})\}$ 在有限步内到达不动点. 记不动点为 $T_P^\omega(\mathbf{I})$.

语义: 不动点解释

Example 12.3.3 Recall the program P_{TC} for computing the transitive closure of a graph G:

$$T(x, y) \leftarrow G(x, y)$$
$$T(x, y) \leftarrow G(x, z), T(z, y).$$

Consider the input instance

$$\mathbf{I} = \{G(1, 2), G(2, 3), G(3, 4), G(4, 5)\}.$$

Then we have

$$T_{P_{TC}}(I) = I \cup \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\}$$
$$T^2_{P_{TC}}(I) = T_{P_{TC}}(I) \cup \{T(1, 3), T(2, 4), T(3, 5)\}$$
$$T^3_{P_{TC}}(I) = T^2_{P_{TC}}(I) \cup \{T(1, 4), T(2, 5)\}$$
$$T^4_{P_{TC}}(I) = T^3_{P_{TC}}(I) \cup \{T(1, 5)\}$$
$$T^5_{P_{TC}}(I) = T^4_{P_{TC}}(I).$$

Thus $T^{\omega}_{P_{TC}}(I) = T^4_{P_{TC}}(I)$.

语义: 不动点解释

定理 5.1.12

$T_P^\omega(\mathbf{I}) = P(\mathbf{I})$.

证明.

显然 $T_P^\omega(\mathbf{I})$ 是不动点, 只需证明其最小.
考虑 T_P 上任意包含 $\mathbf{I}$ 的不动点 $\mathbf{J}$, 有 $\mathbf{J} \supseteq T_P^0(\mathbf{I}) = \mathbf{I}$. 对两边不断取直接结果 T_P, 有 $\mathbf{J} = T_P^i(\mathbf{J}) \supseteq T_P^i(\mathbf{I})$, 因此 $\mathbf{J} \supseteq T_P^\omega(\mathbf{I})$, 从而 $T_P^\omega(\mathbf{I})$ 最小. □

使得 $T_P^i(\mathbf{I}) = T_P^\omega(\mathbf{I})$ 的最小整数 i 称作查询的阶段数, 记作 $stage(P, \mathbf{I})$.
显然 $stage(P, \mathbf{I}) \leq |\mathbf{B}(P, \mathbf{I})|$. 如果 $stage(P, \mathbf{I})$ 只与 P 有关, 则称查询 P 有界 (bounded).

语义: 不动点解释

回顾图的传递闭包查询:

$$T(x, y) \leftarrow G(x, y)$$
$$T(x, y) \leftarrow G(x, z), T(z, y).$$

例 5.1.13 (利用不动点方法求解 Datalog)

图的传递闭包查询可通过如下算法求解:

1. $T := G$;
2. while $q(T) \neq T$, do $T := q(T)$.

上述算法仍存在冗余计算步骤, 之后会讨论优化策略.

语义: 证明论解释

例 5.1.14 (证明树)

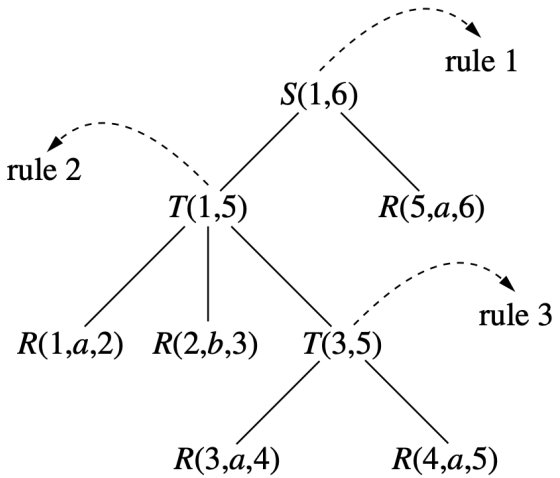
EXAMPLE 12.4.1 Consider the following program:

$$S(x_1, x_3) \leftarrow T(x_1, x_2), R(x_2, a, x_3)$$
$$T(x_1, x_4) \leftarrow R(x_1, a, x_2), R(x_2, b, x_3), T(x_3, x_4)$$
$$T(x_1, x_3) \leftarrow R(x_1, a, x_2), R(x_2, a, x_3)$$

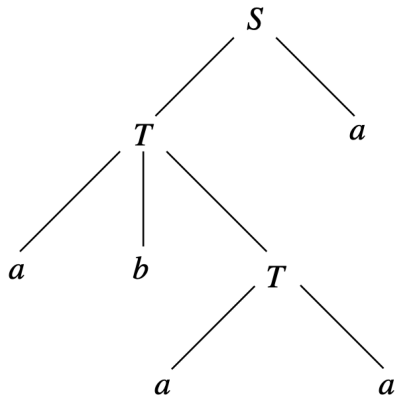
and the instance

$$\{R(1, a, 2), R(2, b, 3), R(3, a, 4), R(4, a, 5), R(5, a, 6)\}.$$

A proof tree of $S(1, 6)$ is shown in Fig. 12.3(a).



(a) Datalog proof



(b) Context-free derivation

语义: 证明论解释

语义: 证明论解释

语义: 证明论解释

静态程序分析

我们分析 Datalog 查询的下列性质:

- 可满足性
- 查询包含
- 有界性

定义 5.1.15 (内涵关系可满足性)

给定 Datalog 程序 P, 称一个内涵关系 T 可被 P 满足, 如果存在数据库实例 $\mathbf{I}$ 使得 $P(\mathbf{I})(T) \neq \emptyset$.

我们先考虑不含常数的 Datalog 查询.

引理 5.1.16

设 P 是某个不含常数的 *Datalog* 程序,$\mathbf{I}, \mathbf{J}$ 是两个实例,q 是某

主要内容

主要内容

主要内容

主要内容