Answer Queries Using Views

概要

- 背景
- 查询包含问题
- 使用视图回答查询
- 分析和总结

背景：使用物化视图加速查询

- Naïve方法：仅当以下条件满足时才使用物化视图
  - 查询与视图定义完全匹配 ×
  - 视图被完整物化 √ ← 我们保留此条件
- 利用物化视图的机会远不止于此
  - 视图与查询**逻辑等价**（A=B AND B=5 与 A=5 AND A=B）
  - 视图与查询的选择条件可能重叠（A>10 与 A>20）
  - 视图可作为查询中JOIN操作的输入（**R JOIN S** JOIN T → **V** JOIN T）
  - 查询的聚合操作可在视图上执行（GROUP BY A,B,C 与 GROUP BY A,B）
  - ……

概要

- 背景
- 查询包含问题
- 使用视图回答查询
- 分析和总结

基础问题: 查询包含与查询等价

- <u>**数据库理论**</u>领域的经典问题
- 查询包含：给定查询Q_1和Q_2，若对于任意数据库实例D都有$Q_1(D) \subseteq Q_2(D)$，则称Q_2包含Q_1，记为$Q_1 \sqsubseteq Q_2$。
- 查询等价：给定查询Q_1和Q_2，若$Q_1 \sqsubseteq Q_2$且$Q_2 \sqsubseteq Q_1$，则称Q_1与Q_2等价，记为$Q_1 \equiv Q_2$。

- 考虑只有一个视图的情形
  - 给定视图定义V_1和查询Q，在V_1包含Q时，可考虑使用V_1回答Q

查询包含问题：复杂度

- Q_1和Q_2均为**合取查询**时，查询包含问题被证明为**NP完全**问题
  - 合取查询：SPJ型查询，仅包含“属性=属性”或“属性=常量”型条件

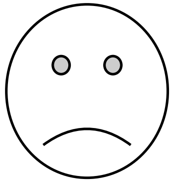
- 实际应用中通常不是瓶颈：
  - 问题规模通常较小
  - 一定**约束**下存在**多项式时间**解法

合取查询 (Conjunctive Query, CQ)

- 考虑表 r(a, b) 和 s(c, d, e)
- SPJ型查询，仅包含“属性=属性”或“属性=常量”型条件
  - `SELECT r.b, s.d FROM r JOIN s WHERE r.a = s.c AND s.e = 5`
- 理论分析中通常使用Datalog语言表示
  - `(X, Y) :- r(W, X), s(W, Y, 5)`
  - `<head> :- <body>`
  - body包含一个或多个<u>**子目标**</u>
  - r, s表示关系，W, X, Y表示变量，共享的变量可理解为JOIN
  - 翻译：如果在r和s中存在记录(W, X)和(W, Y, 5)，则将(X, Y)包含在结果中

包含映射

- **<u>包含映射</u>**：给定查询Q_1和Q_2，一个将Q_2中的**<u>变量</u>**映射到Q_1中的**<u>变量和常量</u>**的**<u>函数</u>**h被称为**<u>包含映射</u>**，如果：
  - 对于Q_2中的每个关系$R(x_1, x_2, \ldots)$，在Q_1中有$R(h(x_1), h(x_2), \ldots)$
  - $h\big(\boldsymbol{head}(Q_2)\big) = \boldsymbol{head}(Q_1)$，其中$\boldsymbol{head}(Q)$表示$Q$的头部变量



- 定理：$Q_1 \sqsubseteq Q_2 \Leftrightarrow$ 存在从Q_2到Q_1的包含映射
- 例：

Q_2:
```
(X, Y) :- r(W, X), s(W, Y, Z)
```

Q_1:
```
(X, Y) :- r(W, X), s(W, Y, 5)
```

CQ的查询包含问题

- 为什么这样一个看上去很简单的问题是NP完全的?
- 考虑布尔查询（头部无变量的查询）：
  - Q_2`() :- R(a, b), R(b, c), R(c, d), R(c, a)`
  - Q_1`() :- R(x, y), R(y, z), R(z, x)`
  - 问题：Q_2是否包含Q_1 $\Leftrightarrow$ 是否存在从Q_2到Q_1的包含映射?
- Three Color Problem
  - 给定图$G(V,E)$，其中$V = \{a,b,c,d\}$，$E = \{(a,b),(b,c),(c,d),(c,a)\}$
  - 是否可用三种不同的颜色$\{x,y,z\}$为图G染色?　　← NP完全问题
    - 使G中任意相邻两点颜色不同

解法：典型数据库

- **典型数据库**：给定CQ查询Q，其**典型数据库**$D[Q]$是Q的主体部分各关系实例化的结果，实例化过程中将**变量名**作为常量。
- 例：典型数据库
  - Q_1(y, x) :- R(y, x), S(x, x), R(x, w)
  - $D[Q_1] = \{\{(y,x),(x,w)\},\{(x,x)\}\}$

R

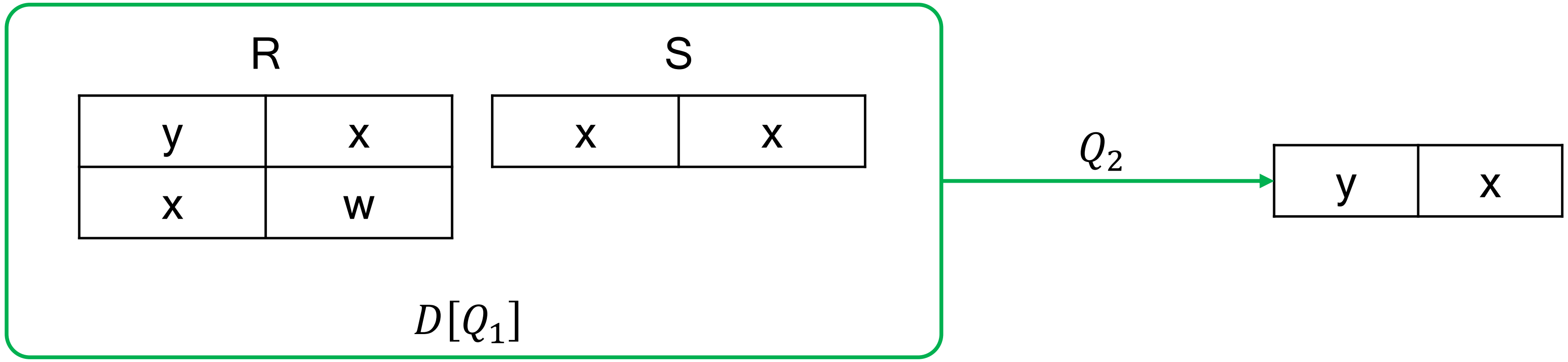
y	x
x	w

S

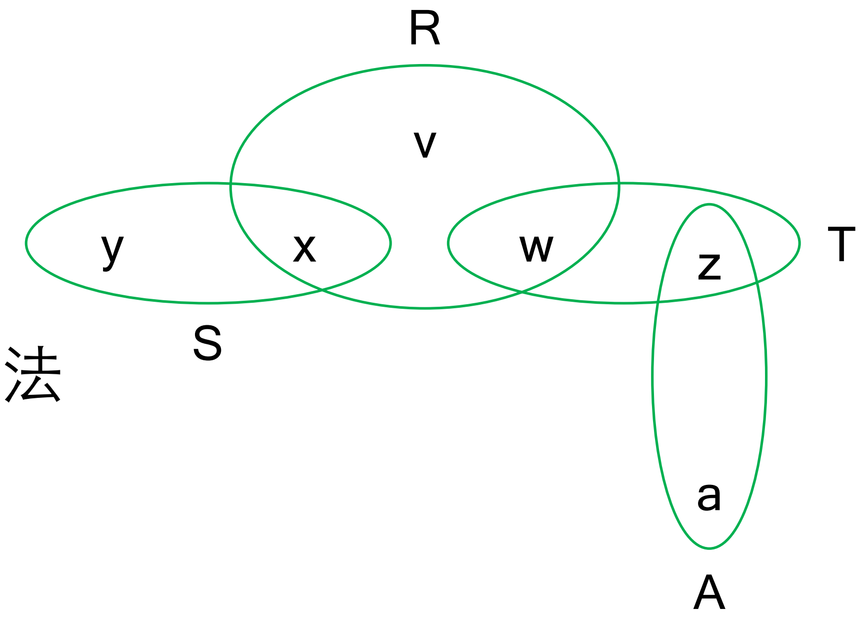
x	x

解法：典型数据库（续）

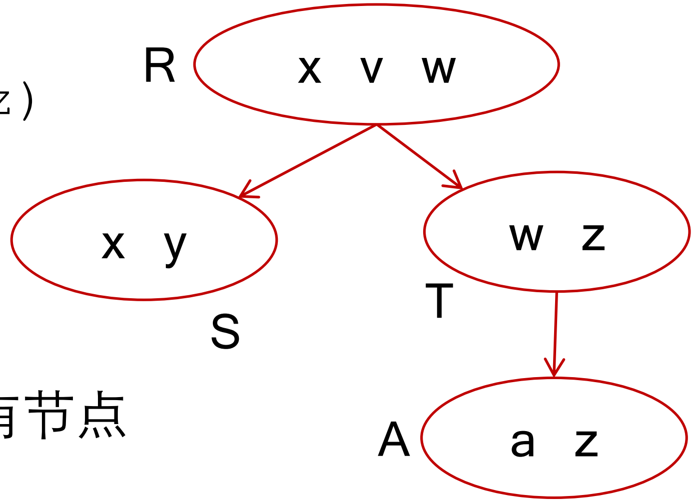
- 定理： $Q_1 \sqsubseteq Q_2 \Leftrightarrow \boldsymbol{head}(Q_1) \in Q_2(D[Q_1])$
- 解法：在Q_1的典型数据库上执行Q_2，若Q_1的头部元组在结果中，则Q_2包含Q_1
  - Q_1(y, x) :- R(y, x), S(x, x), R(x, w)
  - Q_2(x, y) :- R(x, y), S(y, z), R(z, w)



无环CQ查询

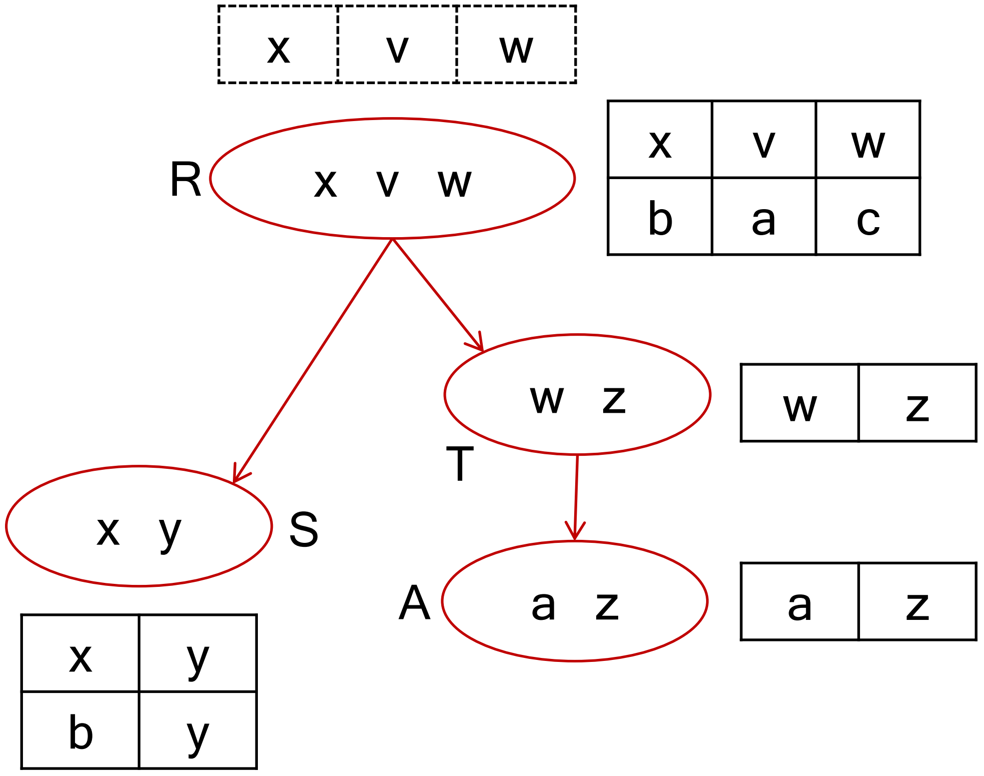


- 现实中环状JOIN较为少见*
  - 无环CQ查询的包含问题有多项式时间的解法
    - 基于超图分解
- 超图：一条边可连接多个节点
  - 边：关系，节点：属性
    - Q_1(y) :- R(x,v,w), S(x,y), T(w,z), A(a,z)
- 构建**消除树**
  - 从超图中不断移除“耳朵”边
    - 耳朵：除和另外一条边的公共节点之外的节点都是私有节点
  - 无环：可将超图消除至空



无环CQ查询：包含问题

- 问题：Q_2是否包含Q_1
- 解法：
  - 构建Q_2的消除树
  - 使用Q_1的主体部分实例化树节点
    - 变量名作为常量
  - 自底向上地计算**自然半连接**
  - 根节点不为空时，$Q_1 \sqsubseteq Q_2$



Q_2(y) :- R(x,v,w), S(x,y), T(w,z), A(a,z)

Q_1(y) :- R(x,v,w), S(x,y), T(w,z), A(a,z), S(b,y), R(b,a,c)

合取查询的并

- 如何处理SQL中的`UNION`操作?
  - 部分通过`OR`连接的条件可转化为`UNION`操作
- 可表示为**合取查询的并**
  - `SELECT id FROM t WHERE name = ‘alice’ OR name = ‘bob’`
  - `Q(x) :- t(x, ‘alice’)`
  - `Q(x) :- t(x, ‘bob’)`
- 定理：令Q_1为一个合取查询，$Q_2 = Q_2^1 \cup \cdots \cup Q_2^n$为一个合取查询的并。$Q_2$包含$Q_1$，**当且仅当**存在$1 \leq i \leq n$使得$Q_1 \sqsubseteq Q_2^i$。

等于条件之外

- 以上讨论限制CQ的选择条件为“属性=属性”或“属性=常量”
- 现实中比较（$=, <, >, \leq, \geq, \neq$）和非（¬）也很常见
  - `(w) :- R(u, v, w, x), S(x, y), ¬T(y, z), u < 10, v > 5`
- 加入此类条件会显著提高<u>**充要条件**</u>的问题复杂度
  - $Q_1 \sqsubseteq Q_2 \Leftrightarrow \ldots$
- 现实中<u>**充分条件**</u>足够满足需求
  - $\ldots \Rightarrow Q_1 \sqsubseteq Q_2$
- 充分条件：包含映射
  - 比较条件：考虑<u>**蕴含关系**</u>和<u>**等价类**</u>

包语义，分组聚合

- 上述讨论基于**集合语义**（消除重复）
- SQL基于**包语义**（默认不消除重复）
- 基于包语义的查询包含问题是**不可判定**的
- 幸运的是：大部分表都有主键
- 聚合函数
  - count，sum基于包语义；max，min基于集合语义
  - 一般处理方法：先考虑底层CQ，再考虑分组和聚合
    - GROUP BY A，B，C可用于计算GROUP BY A，B

外连接（OUTER JOIN）

- 以上讨论基于内连接（INNER JOIN）
- 现实中存在大量外连接操作
- 外连接可分解为SPJ型查询的组合（子查询和UNION）

概要

- 背景
- 查询包含问题
- 使用视图回答查询
- 分析和总结

问题定义

- 给定:
  - 数据库模型$\mathcal{R} = \{R_1, \dots, R_n\}$
  - 定义在$\mathcal{R}$上的视图定义集合$\mathcal{V} = \{V_1, \dots, V_m\}$
  - 对$\mathcal{R}$的查询Q
- 输出查询Q'（如存在），使得:
  - Q'的`FROM`子句只引用$\mathcal{V}$中的视图 ← 基础表也可作为视图
  - Q'等价于Q
- NP完全问题

例：使用视图回答查询

- 数据库
  - Movie(ID, title, year), Director(ID, director), Actor(ID, actor)
- 视图定义
  - `V1(I, T, Y) :- Movie(I, T, Y), Y > 1950`
  - `V2(I, D) :- Director(I, D), Actor(I, D)`
- 查询
  - `Q(I, T, Y) :- Movie(I, T, Y), Director(I, D), Actor(I, D), Y > 1980`
- 查询重写
  - `Q′(I, T, Y) :- V1(I, T, Y), V2(I, D), Y > 1980`

问题解法

- 定理结论：对于有n个子目标的合取查询Q，无需考虑子目标数超过n的重写
  - Naïve解法：枚举所有子目标数不超过n的重写，判断重写是否等价于Q

定理：当存在变量映射ϕ_i将$V_i \in \mathcal{V}$的子目标映射到Q的子目标时，称视图V_i对Q是**有意义的**。令$V_1, \ldots, V_k$表示对Q有意义的视图，$\boldsymbol{Y}_i = \boldsymbol{head}(V_i)$，

$$C(\boldsymbol{head}(Q)) \; :- \; V_1(\phi_1(\boldsymbol{Y}_i)), \ldots, V_k(\phi_k(\boldsymbol{Y}_{\boldsymbol{k}}))$$

当且仅当C的展开$C' \sqsubseteq Q$时，Q存在基于$\mathcal{V}$的重写。

- 实际可行的解法：使用上述定理及其它（启发式）规则剪枝。

问题解法（续）

- 桶算法
  - 为Q的每个子目标初始化一个桶
  - 将对每个子查询有意义的视图加入其对应的桶中
  - 计算所有桶的笛卡尔积，对产生的每条查询检测其是否与Q等价
- 存在其它更高效的算法
  - （下一步工作）

概要

- 背景
- 查询包含问题
- 使用视图回答查询
- 分析和总结

总结和分析

- 上面仅讨论了生成重写的方法
- 但是:
  - 可能存在多种重写
  - 可能直接从源表计算更高效
- 还是需要代价模型

- 进一步细化对具体算法的了解
  - 外连接，查询重写