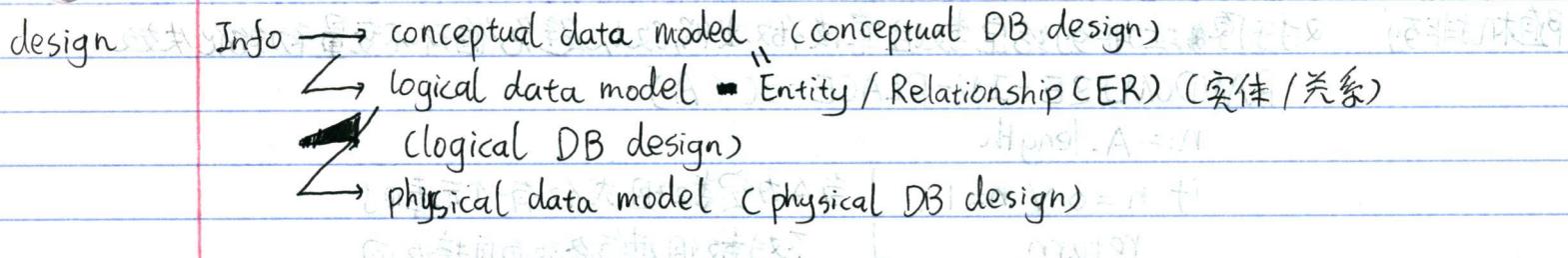


# Data Base - P1



概念数据模型 (conceptual data model), ER 模型中的最高级别 (the highest level ER model)

包含最少的细节信息, 但是建立了包含在模型集中要素的总体范围

model set

contain the least granular details, but establish the overall scope of what included in the

描述了系统的结构 (protray the system architecture)

对于一个总体范围更小的系统, 信息已不足以画出模型图

for a system of smaller scope, may not be necessary to draw

主要目的是建立逻辑ER模型间主数据实体的结构上的元数据共性

establish structural metadata commonality

for the master data entities between the set of logical ER models

另外可用于形成ER模型间的共性, 作为数据模型整合的基准

form commonality relationships between ER models as a basis for data model integration

逻辑数据模型 (logical data model), 比概念模型包含更多信息, 通常基于一个概念数据模型

注意并不强制需要一个概念模型, 特别是总体范围仅限于建立一个特别的信息系统

not require a conceptual ER model, especially if

the scope of the logical ER model includes only development of a distinct information system

相比概念模型, 在主数据实体外, 定义了操作和事务数据实体 (operational and transactional)

同时包含了每个数据实体建立的细节, 以及数据实体间关系的建立

detail of each data entity are developed, relationships between data entities are established

注意, 逻辑数据模型与实际实现无关 (developed independently of specific database management system)

物理数据模型 (physical data model), 基于一个逻辑数据模型建立 (developed from logical ER model)

通常而言, 物理数据模型是建立一个数据库实例

normally developed to be instantiated as a data base

必须包含足够多的技术细节以生成和实现一个实际的数据库

must show enough technology detail to produce and implement the actual database

由于实际的系统差异, 物理数据模型是依赖于技术的

technology dependent since each database management system is somewhat different

# Database - P2

E-R 模型 (Entity Relationship Model), 也称实体-关系模型, 用于描述现实世界的概念模型  
ER 模型中的基本要素: 实体 (entity), 关系 (relationship), 属性 (attribute)

rule of thumb E-R 模型中的要素与自然语言语法存在一个大略地对应关系

自然语言 natural language	E-R 模型结构 (E-R structure)	类似于
语法语法 grammar structure		
普通名词 common noun (普通名词)	entity type (实体类型)	类型 class
专有名词 proper noun (专有名词)	entity (实体)	实例 instance
及物动词 transitive verb (及物动词)	relationship type (关系类型)	方法 method
不及物动词 intransitive verb (不及物动词)	attribute type (属性类型)	函数 function
形容词 adjective (形容词)	attribute for entity (实体属性)	属性 property
副词 adverb (副词)	attribute for relationship (关系属性)	

E-R 图 (Entity Relationship Diagram, E-R Diagram)

根据定义好的 E-R 的模型, 依据指定的规则进行构图

实体 (entity), 指可以被唯一地认知的客观存在的物体, 在 E-R 图中以矩形表示

a thing capable of an independent existence

entity

实体可以从一个复杂的域中抽象出的概念

abstraction from the complexities of a domain, exist logically

也将这种情形区分为 entity type, 即一种分类 (category)

entity set

实体集合 (entity set), 与 entity type 一样是一种分类

但区别在其定义是依据实际情况而定的

defined at a particular point in time

如 1, 2, 0, -11 或任意  $x \in \mathbb{Z}$ , 都是整数的实例, 即 entity

而自然数  $\mathbb{N}$ , 正整数  $\mathbb{N}^+$ , 可以看作是整数的抽象概念, 即 entity type

而集合  $\{x \in \mathbb{Z} \mid 1 \leq x\}$  则可以看作是 entity set

attributes

强实体 (strong entity), 可以独立地由其自身的属性定义 (defined solely by its own)

弱实体 (weak entity), 指对另一个实体 (可以是强实体或依赖于强实体的弱实体)

有很强的依赖关系,

即全部或部分主键, 从其依赖的强实体获得

weak entity

# Database - P3

5.9 - 读取与写入

关系

(relationship), 又称联系，反映信息世界中实体内部或实体之间的关联

注意在 E-R 模型中，通常指实体之间的关系

capture how entity is related to one another

实体内部的联系通常指组成实体的各属性间的联系

实体间联系通常指不同实体集之间的联系

在 E-R 图中，用菱形表示实体间的关系，并用无向边连接相关实体

在无向边上标注联系的类型，可以看作关系相对于实体的唯一性

so. 1:1, 一对-一的联系 (one-to-one)，对于实体 A 与 B 之间的 1:1 关系

表示对于每一个 A 的实例，有至多一个 B 的实例与之相关联

对于每一个 B 的实例，有至多一个 A 的实例与之相关联

注意 1:1 关系与 bijective 映射的区别在于允许出现未指派的值

1:N, 一对多的联系 (one-to-many)，对于实体 A 与 B 之间的 1:N 关系

表示对于每一个 A 的实例，有 0 个或若干个 B 的实例与之相关联

对于每一个 B 的实例，有至多一个 A 的实例与之相关联

M:N, 多对多的关系 (many-to-many)，对于实体 A 与 B 之间的 M:N 关系

表示对于每一个 A (或 B) 的实例，有 0 个或若干个 B (或 A) 的实例与之相关联

但是特别注意，这种记法无法区分是否允许对实体的实例并不具有一种定义好的关系

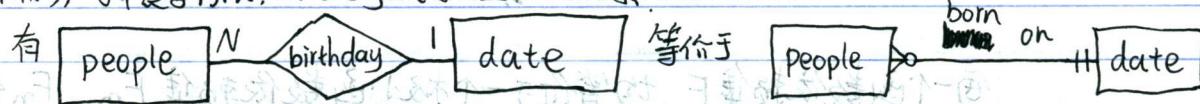
即 N 或者 N 是否包含 =0 的情形（其实可以以虚线表示）

将这种记法为 Chen's notation for entity-relationship modeling

基数 (cardinality). 定义了两个实体或实体集之间关系的数量属性

define the numerical attribute of the relationship between two entities or entity sets

可采用另一种表示方法，如对于人和生日的关系



有对应关系在两种表示方法之间

zero or one



one and only one



one and only one

one and only one

one and only one



one and only one



基数限制 (cardinality constraint)，指应用于关系时的最小或最大数

the minimum or maximum number apply to a relationship

# Database - P4

## 属性

(attribute) 实体和关系所具有的某一特性 (property or characteristic)

实体和关系可以由若干属性来刻画，而属性不能脱离实体和关系单独存在

有时为了区分实体与关系的属性，会将关系的属性称为描述属性 (descriptive attribute)

在 E-R 图中，属性用椭圆表示。

并且通过无向边与相应的实体或关系相连

由于属性是相对于实体或关系而言的

所以通常而言属性连接有且仅有一个实体或关系，即使不同实体关系有相同名称的属性



通常所指的属性为简单属性 (simple attribute)，即属性的值不可分割的

即 attribute value is atomic and can't be further divided

而派生属性 (derived attribute)，指可通过其他属性的值计算或生成

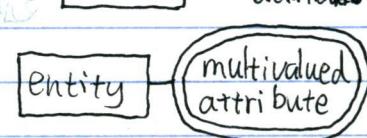
calculated or otherwise derived from another attribute

在 E-R 图中用虚线椭圆表示

多值属性 (multi-value attribute)，指属性可以指很多个值

在 E-R 图中用双层椭圆表示

通常情况下：除非特别说明，否则默认属性是单值的 (single-value attribute)



键 (key) 单属性或多个属性的组合，用于唯一地指向实体集中的一一个实例或关系表中的一条记录

an attribute or combination of attributes

uniquely define / identify an entity in an entity set or a record in a relational table

Database Object	type	subset	constraint	description
V1 (主键) primary key		{ primary key }	unique, not null	a candidate key uniquely identify entity set
unique key could be primary key	natural key	{ candidate key/s } { attributes containing NULL }	unique	may or may not be NULL is primary key in OLTP may be primary key in OLAP
as a primary key as a primary key (复合主键)	surrogate key	{ primary key }	unique, not null	artificial key generated internally in OLAP acting as substitute of primary key in OLTP
composite key	superkey	{ primary key }	unique, not null	primary key made up of multiple attributes
		{ candidate key/s }, { candidate key/s } { with extraneous }	N.A.	Can be used to identify a database record may contain extra attributes not necessary to uniquely identify record used to uniquely identify a db record without any extraneous data
	candidate key	{ PK, { alternate key/s } }	N.A.	not null and unique, a minimal superkey
	alternate key	{ { candidate key } - PK }	N.A.	candidate key but not primary key
	candidate key with extraneous data		N.A.	如果 <key, data> 可以 identify 但是 <key> 也能 identify 且 data 部分仅为 extraneous data

# Database - P5

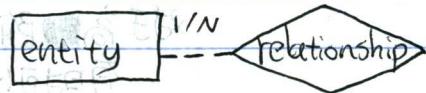
可选关系

(optional relationship), 即对于与实体集联系的某一关系 R

实体集中任一实体实例可以拥有 0 个关系 R 的实例

在 Chen's notation 中以虚线表示,

对应于 Information Engineering style (IE)



外键

(foreign key), 指单个属性或一组多个属性是另一个实体或关系中的主键

注意: 外键的值必须可以匹配参照中的一个实例

an attribute or combination of attribute in an entity / relationship

whose value match a primary key in another entity / relationship

a key in one table uniquely identify a row of another table (parent)

外键在母表 (parent table) 必须是唯一的, 但在子表 (child table) 中表现为复合主键

即外键在子表中表现为某一序偶  $(k_1, k_2, \dots, k_n)$  的值必须唯一.

且  $k_1, k_2, \dots, k_n$  必须在表中作为主键出现.

母表中的主键在子表中可以多次出现 (not-unique in child table)

参照完整性 (referential integrity), 指外键的值或者存在于母表的主键, 或者为 NULL

maintained as the value in foreign key in child table

is present as a value in primary key in parent table, else NULL

注意在实际数据库实现中, 可能会为了运行效率而不设置外键约束

外键约束 (constraint) 保证关系数据库无法插入无效数据.

通常外键约束蕴含了非空约束 (Not NULL constraint)

关联实体

(associative entity), 用于处于实际应用中的多对多关系, 也称关联表 (associative table)

implementation of a base relation to resolve many-to-many relationship

这种关系通常也可以看作一种实体

在 ER 图中, 关联实体用 表示.

关联实体通过引用主键的方式 映射到两个或更多其他表

map two or more tables together by referencing the primary keys of each data table

所以关联实体通常具有一组外键 (foreign key)

而这一组外键通常或者直接作为关联实体的主键, 即复合主键 (composite key)

或者作为 alternate key 或 extraneous data 出现

在这种情况下必须另外确定关联实体的主键  $FK_1, FK_2, \dots, FK_n$

关联实体或者表现为一个定义在多个外键集合上的关系,  $R \subseteq \{FK_1, FK_2, \dots, FK_n\}$

或者表现为一个定义在从包含外键的复合主键到一条记录的映射

$f: (PK \times FK_1, FK_2, \dots, FK_n) \rightarrow T$

-- 对应的

# Database - P6

## 弱关系

(weak relationship), 通常用于表示弱实体 (weak entity) 与某所依赖的强实体 (或依赖于其他强实体的弱实体) 的关系。即弱关系特指这种依赖关系 (dependent of strong entity) 在 E-R 图中用双层菱形框表示且与弱实体相连的一侧使用双实线。

## 递归关系

(recursive relationship), 指实体集中的某一实例与实体集中另一实例的关系。在 E-R 图中表现为关系的两侧连接同一实体。典型如以人作为实体集, 定义在其上的亲子关系。

复合属性 (composite attribute), 指实体的属性可以进一步分解为细节属性。或者说某一属性可由若干相互独立的部分构成, 或可拆分为若干相互独立的部分。

在 E-R 图中表现为属性连接在另一个属性上。典型如地址属性可以细分为省-市-街道。

## ER 模型 局限性

只用于关系型数据, only for relational data  
不能用于未结构化数据, not for unstructured data  
数据必须被清晰地描述为行、列, 或  
cleanly delineated into different fields, rows or column  
难以与已经存在的数据库整合  
difficulty integrating with an existing database

关系代数 (relational algebra): 是有明确语义 (well-founded semantics) 的一系列代数  
是一阶逻辑 (first-order logic) 的一个分支, 是闭合于运算下的关系的集合  
运算作用于一个或多个关系而生成一个关系。  
注意这里的“关系”扩展为  $n$  元组的集合, 即对于  $n$  个不同的属性,

定义在其上的关系  $R \subseteq A_1 \times A_2 \times \dots \times A_n = \{ (a_1, a_2, \dots, a_n) \mid a_i \in A_1, a_2 \in A_2, \dots, a_n \in A_n \}$

Well-founded semantics (有充分根据的语义). 用于定义如何从一组逻辑规则中得到一个结论。

First-order logic (-阶逻辑), 又称 first-order predicate calculus (-阶谓词演算), 量化理论  
与命题逻辑不同在于使用量化变量 (quantified variable), 并允许使用包含变量的语句

# Database - P7

关系代数运算

## 关系代数运算

关系代数中有六个原始运算，包括三个取自集合论的集合运算

且这三个运算与集合论中的运算是一样的

集合运算 (set operator), 包括三个取自集合的运算，但是与集合论中略有差异

并集 (set union), 差集 (set difference), 笛卡尔积 (Cartesian product)

在关系代数中，关系 R 定义为一个 n 个属性组成的 n 元组

即  $R = \{(a_1, a_2, \dots, a_n) \mid a_i \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}$

对于并集和差集，要求作用于运算的两个集合必须是并集相容的

并集相容 (union-compatible) 指两个关系必须拥有同样的属性集合

即如果 R 定义在属性集合  $A_1 \times A_2 \times \dots \times A_n$  上

则对于运算  $R \cup S, R - S$ , 其中的集合 S

S 也必须定义在同样的属性集合  $A_1 \times A_2 \times \dots \times A_n$  上

而由交集 (set intersection) 可由并集和差集定义

即  $R \cap S = R - (R - S)$

所以交集运算也必须是并集相容的

对于笛卡尔积，与集合论中的笛卡尔积定义略有不同

关系代数中的笛卡尔积要求两个关系拥有不相交的表头 (disjoint header)

即用于  $R \times S$  的关系 R 和 S 不能有相同的属性名 (no common attribute name)

在集中，元组被认为是浅表的 (shallow)

即  $R \times S = \{(a_1, \dots, a_n, b_1, \dots, b_m) \mid (a_1, \dots, a_n) \in R, (b_1, \dots, b_m) \in S\}$

而在关系代数中，关系 R 与关系 S 会被展平 (flattened) 并组装成新的元组

即  $R \times S = \{(a_1, a_2, \dots, a_n, b_1, \dots, b_m) \mid (a_1, \dots, a_n) \in R, (b_1, \dots, b_m) \in S\}$

即 n 元组关系与 m 元组关系的笛卡尔积为  $(n+m)$  元组

而  $R \times S$  的基数  $|R \times S| = |R| \times |S|$ , 这里与集合论中的定义相同

注意：在实际应用中， $R$  和  $S$  中的同名属性可以区分

如用关系名. 属性名加以区分，如  $R.name$  和  $S.name$

投影 (projection), 运算符为  $\pi$ , 是作用于关系 R 的一元运算 (unary operation)

记为  $\pi_{a_1, a_2, \dots, a_n} R$ , 其中  $a_1, a_2, \dots, a_n$  是一个属性名的集合

且  $\{a_1, a_2, \dots, a_n\} \subseteq \{r_1, r_2, \dots, r_m\}$ , 其中  $r_1, r_2, \dots, r_m$  为 R 的属性集合

即投影运算可理解为关系 R 中的元组被截取在属性集合  $\{a_1, a_2, \dots, a_n\}$  上

all tuple restricted to the set  $\{a_1, a_2, \dots, a_n\}$

语句中的属性名  
部分。即

在 SQL 中对应 select 语句，“default projection”会返回一个多重集 (multiset)

通过 DISTINCT 关键字，去除重复数据，

addition of DISTINCT keyword to eliminate duplicate data

# Database - P8

## 选择

(selection), 运算符为  $\sigma$ , 是作用于  $n$  元组关系  $R$  的一元运算

记为  $\sigma_P R$ , 其中  $P$  是一个从  $R$  中的一个属性为非逻辑对象的量化量化的 - 阶逻辑  
即  $P$  一个形如  $\forall P(a_i)$  的命题, 其中  $a_i$  是  $R$  的属性是  $P$  的 quantified variable over nonlogical object  
其结果为  $R$  中所有使  $P(a_i)$  为真的  $n$  元组

$$\text{即 } \sigma_{P(a_i)} R = \{ (a_1, a_2, \dots, a_n) \in R \mid P(a_i) \}.$$

这种选择也称为 ~~一阶~~ 一阶选择 (normal selection),

而通常所使用的形成为  $\sigma_\varphi R$ , 其中  $\varphi$  是一个命题公式 (propositional formula)

即  $\varphi$  为由在一阶选择中使用的原子逻辑 (atom) 以及逻辑运算符  $\vee, \wedge, \neg$  组成

即  $P$  形如一组命题公式的合取式  $\bigwedge_{i=1}^m P_i(a_1, a_2, \dots, a_n)$ .

$$\text{其结果 } \sigma_{\varphi=P_1 \wedge P_2 \wedge \dots \wedge P_m} R = \{ (a_1, a_2, \dots, a_n) \in R \mid P_1 \wedge P_2 \wedge \dots \wedge P_m \}.$$

注意在通常使用中, 常写作  $\sigma_{P_1, P_2, P_3, \dots, P_m} R$ .

这种记法与  $\sigma_{P_1} \wedge \sigma_{P_2} \wedge \sigma_{P_3} \wedge \dots \wedge \sigma_{P_m} R$  等价,

另外注意选择运算本身是右结合的, 即计算顺序为右向左

$$\text{即 } \sigma_{\varphi_1} \sigma_{\varphi_2} R \text{ 等价于 } \sigma_{\varphi_1} (\sigma_{\varphi_2} R)$$

选择运算也是可复合的, 即  $\sigma_{\varphi_1} (\sigma_{\varphi_2} R) = \sigma_{\varphi_1 \wedge \varphi_2} R$

即如果命题公式  $\varphi$  可拆分为命题公式  $\varphi_1, \varphi_2, \dots, \varphi_m$  的合取式

$$\text{即有 } \varphi = \varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_m.$$

$$\text{则选择运算 } \sigma_\varphi R = \sigma_{\varphi_1} (\sigma_{\varphi_2} (\dots (\sigma_{\varphi_m} R) \dots))$$

由于合取式  $\varphi_1 \wedge \varphi_2$  是可交换的, 则选择运算也是可交换的

$$\text{即 } \sigma_{\varphi_1} (\sigma_{\varphi_2} R) = \sigma_{\varphi_2} (\sigma_{\varphi_1} R)$$

也就是说改变选择运算的川序并不影响复合选择运算的结果

在 SQL 语言中, 选择运算对应于 Select 语句, 即  $\text{select * from } R$

即  $\sigma_{\varphi} R$  等价于语句  $\text{select * from } R \text{ where } \varphi$

## 重命名 (rename), 运算符为 $\rho$ , 是作用于 $n$ 元组关系 $R$ 的一元运算

记为  $\rho_{a/b} R$ , 其中  $b$  为关系  $R$  中一个现存的属性名,  $a$  为与关系  $R$  中除  $b$  以外不重名的属性

其结果为一个新的关系  $R'$ , 与  $R$  的差别仅为所有元组的属性  $b$  重命名为属性  $a$

identical to  $R$  except the  $b$  attribute in all tuples renamed to an  $a$  attribute

$$\text{即 } \rho_{b/a} R = R' (a_1, a_2, \dots, b, \dots, a_n) = R' (a_1, a_2, \dots, a, \dots, a_n)$$

重命名运算通常简单地用于对关系的属性或关系本身进行重命名, 以解决运算时的重复

Simply used to rename the attribute of a relation or the relation itself

注意重命名运算在 SQL 语言中没有通常的对应语句,

因为 SQL 中关系名的重复问题由编译器控制,

属性名的重复问题由表示方法 (关系名, 属性名) 解决

# Database - Pg

自然连接

(natural join), 运算符为  $\Delta$ , 是作用于  $m$  元关系  $R$  和  $n$  元关系  $S$  的二元运算 (binary operation) 记为  $R \Delta S$ , 其结果为在  $R$  和  $S$  中的公共属性上相等的所有元组的组合

set of all combinations of tuples in  $R$  and  $S$  equal on common attribute names  
连接也称为关系复合 (composition of relations)

在范畴论中的精确定义为纤维积

in the category theory, precisely the fiber product

最重要的运算符之一, 是逻辑中的 AND 在关系中的对应,

relational counterpart of logical AND

注意, 如果同一个变量出现在 AND 两侧的命题之中, 则变量表示相同对象  
且替换为值时必须使用相同的值

if the same variable appears in each of two predicates connected by AND  
that variable stands for the same thing  
and both appearances must be substituted by the same value

自然连接的意义 (semantics) 通常定义为

$$R \Delta S = \{ r \Delta s \mid r \in R \wedge s \in S \wedge \text{Fun}^{\text{RUS}}(r, s) \}$$

其中 Fun 是一个命题, 对于一个数学意义上的 ( $=$  元) 关系  $t$

当且仅当  $t$  是一个函数时为真  
iff  $t$  is function  
a predicate, true for relation  $t$  (mathematical sense)

通常而言要求关系  $R$  和  $S$  至少有一个公共属性 (at least one common attribute)

注意如果忽略这个约束, 且  $R$  和  $S$  没有公共属性, 则自然连接等同于笛卡尔积

if constraint omitted, and  $R$  and  $S$  have no common attribute

natural join become exactly Cartesian product

通过关系代数原始运算模拟自然连接: 对于  $m+k$  元关系  $R$  和  $n+k$  元关系  $S$ ,

$r_1, r_2, \dots, r_m$  为关系  $R$  的特有属性,  $s_1, s_2, \dots, s_n$  为关系  $S$  的特有属性,  $c_1, \dots, c_k$  为公共属性

首先对  $S$  中的公共属性  $c_1, \dots, c_k$  重命名为  $d_1, d_2, \dots, d_k$

且  $d_1, d_2, \dots, d_k$  不存在于  $r_1, r_2, \dots, r_m, s_1, s_2, \dots, s_n$  中

注意这里的

rename common attribute.  $T = P_{d_1/c_1, \dots, d_k/c_k}(S) = P_{d_1/c_1}(P_{d_2/c_2}(\dots(P_{d_k/c_k}(S))\dots))$

rename go

然后取  $R$  和  $S$  的笛卡尔积, 并选择相连接的元组

take the Cartesian product and select tuples that be joined 右结合的

attributes

$$U = \sigma_{c_1=d_1, \dots, c_k=d_k}(R \times S) = \sigma_{c_1=d_1}(\sigma_{c_2=d_2}(\dots(\sigma_{c_k=d_k}(R \times S))\dots))$$

get rid of renamed

最后取投影, 以删除重命名的属性

take projection

$$V = \pi_{r_1, \dots, r_m, c_1, \dots, c_k, s_1, \dots, s_n}(U)$$

# Database - P10

①连接

( $\theta$ -join), 与自然连接的运算符  $\bowtie$  相同, 也是作用于两个关系  $R$  和  $S$  的二元运算  
又称 ( $\Theta$ -join), 记为  $R \bowtie_{\theta} S$  或  $R \bowtie_{\theta} S$   
其中  $a, b$  为  $R, S$  中的属性名,  $\theta$  为常量.  $\theta$  为比较运算符  $\{<, \leq, =, \neq, \geq, >\}$   
其结果为  $R$  和  $S$  间满足  $\theta$  运算的所有元组组合  
通常  $\theta$  连接定义在  $R, S$  没有公共属性的情况下

通过关系代数原始运算模拟  $\theta$  连接

则  $\theta$  连接可视为对两个关系的笛卡尔积进行与  $\theta$  连接的条件  $\theta$  相同的选择

即  $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$ , 这里  $\theta$  为比较运算

当  $\theta$  运算的运算符为等号 “=” 时,  $\theta$  连接也称为相等连接 (equijoin)

所以自然连接也可以视为一类相等连接

在计算机语言中,  $\theta$  连接也可以由在两关系的自然连接上执行选择运算获得

所以对于支持自然连接和选择运算符的计算机语言, 可以不需要特别定义的  $\theta$  连接

注意在这种情况下, 如果关系  $R$  和  $S$  间没有公共属性

则自然连接将退化 (degenerate) 为笛卡尔积

在 SQL 语⾔实现中, 基于命题的连接称为内连接 (inner join)

joining on predicate usually called inner join

通过 on 关键字来定义选择的命题

注意: 通过构造笛卡尔积再进行选择在概念上是正确的

forming the flattened Cartesian product then filter rows is conceptually

但是实现中会使用更智能的数据结构来提高连接命令的速度

use more sophisticated data structure to speed up the join query

半连接

(semijoin), 运算符为左半连接  $\bowtie_L$  和右半连接  $\bowtie_R$ , 也是作用于两个关系  $R$  和  $S$  的二元运算

半连接的语义更形式化地定义为

$R \bowtie_L S = \{r \mid r \in R \wedge \exists s \in S (\text{Fun}(r, s))\}$  其中 Fun 的定义与自然连接相同

$R \bowtie_R S = \{s \mid s \in S \wedge \exists r \in R (\text{Fun}(r, s))\}$

可以基于自然连接通过关系代数原始运算模拟半连接  $C_1, \dots, C_k$  为  $R$  和  $S$  的公共属性

$R \bowtie_L S = \text{JL}_{C_1, C_2, \dots, C_k}(R \bowtie S)$ , 其中  $r_1, \dots, r_m$  为  $R$  的特有属性

$R \bowtie_R S = \text{JL}_{S_1, S_2, \dots, S_n}(R \bowtie S)$ , 其中  $s_1, s_2, \dots, s_n$  为  $S$  的特有属性

半连接也称为限制 (restriction)

# Database - P11

DEFINITION

## 反连接

(antijoin), 运算符为  $\Delta$ , 也是作用于两个关系 R 和 S 的二元运算  
其结果为在 S 中没有在公共属性上相等的元组的 R 中的元组

only tuple in R for which no tuple in S equal on their common attribute

反连接的意义可形式化地定义为

$$R \Delta S = \{r \mid r \in R \wedge \forall s \in S \text{ such that } \text{Func}(r) \neq \text{Func}(s)\}$$

$$\text{或 } R \Delta S = \{r \mid r \in R, \text{ no tuple } s \in S \text{ satisfy } \text{Func}(r) = \text{Func}(s)\}$$

反连接也可以定义为半连接的补集 (左半连接)

defined as the complement of the semijoin.

$$RP(R \Delta S) = R - R \times S$$

## 除法

(division), 运算符为  $\div$ , 作用于两个关系 R 和 S 的二元运算, 如  $R \div S$

其结果限制在关系 R 中特有的属性的元组,

restriction of tuples in R to the attribute name unique to R

且与 S 中元组的每一种组合在 R 中都存在, 这里忽略 S 的特有的属性

all combinations with tuples in S are present in R

除法的意义可形式化地定义为

$$R \div S = \{r_1, r_2, \dots, r_m \mid (r_1, r_2, \dots, r_m) \in R \wedge \forall s \in S \text{ such that } (r_1, r_2, \dots, r_m) \times s \in R\}$$

注意这里  $r_1, r_2, \dots, r_m$  为关系 R 的特有的属性

$c_1, c_2, \dots, c_n$  为关系 R 和 S 的公共属性

特别地要求关系 S 没有特有的属性, 即 S 的属性是 R 的属性的一个子集

否则结果将始终为空集

通过关系代数原语运算符以除法

首先投影 R 至仅有特有的属性  $r_1, r_2, \dots, r_m$ , 然后计算与 S 的笛卡尔积

$$A1: T = (\pi_{r_1, r_2, \dots, r_m} R) \times S$$

这一步得到所有可能的组合, 包括不在 R 中的部分

然后计算关系 T 与关系 R 的差集

$$A2: U = T - R$$

这一步得到所有应该存在于 R 但并不存在的元组

最后投影 U 和 R 到 R 的特有的属性, 并求两者差集

$$V = \pi_{r_1, r_2, \dots, r_m} U$$

$$W = (\pi_{r_1, r_2, \dots, r_m} R) - V$$

这一步从 R 中删除了没有包含所有元组的组合的部分

注意: 除法在 SQL 语言中并没有针对性的实现

most databases don't have division as a standard operation