Relational Model

⚠6 basic relational algebra operators:

Select: σ

Project: Π

Union: ∪

Set difference: –

Cartesian product: x

Rename: ρ

Additional Operations:

Set intersection

Natural join

Division

Assignment

i.e.

- Attributes: *ID, name, dept_name, salary*
- Schema: *instructor* (*ID, name, dept_name, salary*)
- Instance: rows in the relation

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Primary key is one of the selected candidate keys

Foreign key constraint: value in one relation must appear in another

⚠ Some operation examples:

Select [查询]

Relation r

A	*B*	*C*	*D*
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{A=B \wedge D > 5}(r)$

A	*B*	*C*	*D*
α	α	1	7
β	β	23	10

Example of selection:

$\sigma_{branch\_name="Perryridge"}(account)$

Tips: ∧ (and), ∨ (or), ¬ (not)

$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$

Project [项目]

Relation r

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Example: to eliminate the *branch_name* attribute of *account* (branch_name, account_name, balance)

$\Pi_{account\_number,\ balance}\ (account)$

Union [并集]

Relations r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cup s$:

A	B
α	1
α	2
β	1
β	3

Example: to find all customers with either an account or a loan

$\Pi_{customer\_name}(depositor) \cup \Pi_{customer\_name}(borrower)$

Set Difference [差集]

Relations r, s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r - s$

A	B
α	1
β	1

Cartesian-Product [笛卡尔积]

Relations r, s

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

$r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Rename [重命名]

Example:

$$\rho_x(E)$$

returns the expression E under the name X

If a relational-algebra expression E has arity n, then

$$\rho_{x(A_1,A_2,\ldots,A_n)}(E)$$

Set-Intersection [交集]

Relation r, s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cap s$

A	B
α	2

Note: $r \cap s = r - (r - s)$

Natural-Join [自然连接]

Relations r, s

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	∈

s

r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Example:

$R = (A, B, C, D)$

$S = (E, B, D)$

- Result schema = (A, B, C, D, E)
- $r \bowtie s$ is defined as:

$$\Pi_{r.A,\, r.B,\, r.C,\, r.D,\, s.E}(\sigma_{r.B = s.B \wedge r.D = s.D}(r \times s))$$

Division [分配]

Relations r, s

r

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
∈	6
∈	1
β	2

s

B
1
2

$r \div s$:

A
α
β

(Not require)

Relations r, s

r

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

s

D	E
a	1
b	1

$r \div s$

A	B	C
α	a	γ
γ	a	γ

Assignment [分配]

Example: Write $r \div s$ as

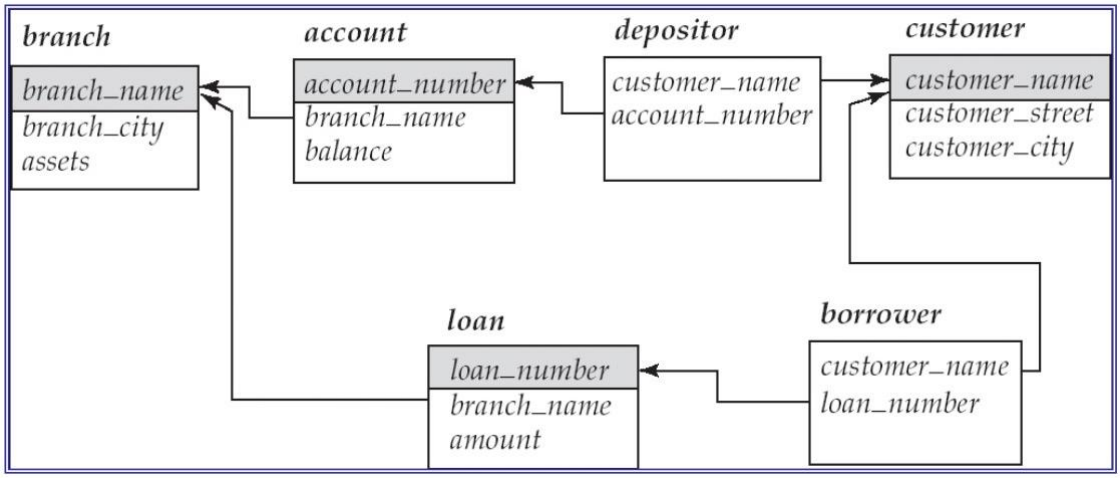
$$temp1 \leftarrow \Pi_{R-S}(r)$$
$$temp2 \leftarrow \Pi_{R-S}((temp1 \times s) - \Pi_{R-S,S}(r))$$
$$result = temp1 - temp2$$

The result to the right of the ← is assigned to the relation variable on the left of the ←.

May use variable in subsequent expressions.

⚠ Quiz of relational model operation:

Here is a banking relational model example:



(1)

Find all loans of over $1,200

$$\sigma_{amount > 1,200}(loan)$$

思路：直接就是一个模糊查询。

(2)

Find the names of all customers who have a loan at the Perryridge branch.

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$$

OR

$$\Pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number}((\sigma_{branch\_name = \text{“Perryridge”}}(loan)) \times borrower))$$

思路：直接筛选一波是"Perryridge"名字的人的贷款并与 borrower 形成笛卡尔积（通过 loan_number 筛选连接）并得出最后 customer_name。

(3)

Find the largest account balance

- Strategy:
  - Find those balances that are *not* the largest
    - Rename *account* relation as *d* so that we can compare each account balance with all others
  - Use set difference to find those account balances that were *not* found in the earlier step.

$$\Pi_{balance}(account) - \Pi_{account.balance}(\sigma_{account.balance < d.balance}(account \times \rho_{d}(account)))$$

思路：创建临时表与原表对比找出不是最大的，最后再利用差集求出最大的值。

(4)

Find the names of all customers who have both a loan and an account at bank.

$$\Pi_{customer\_name}(borrower) \cap \Pi_{customer\_name}(depositor)$$

思路：找到位于 borrower 以及 depositor 中的用户并求交集。

(5)

Find the name of all customers names, their load numbers and loan amount

$$\Pi_{customer\_name,\ loan\_number,\ amount}(borrower \bowtie loan)$$

思路：把贷款人和借款人 natural join 后取需要的元素即可。

(6)

Find names of all customers who have an account from both the "Downtown" and the "Uptown" branches.

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"Downtown"}}(depositor \bowtie account)) \cap \Pi_{customer\_name}(\sigma_{branch\_name = \text{"Uptown"}}(depositor \bowtie account))$$

思路：将储存用户和账户 natural join 后用 branch_name='*****'筛选，最后将条件求交集。

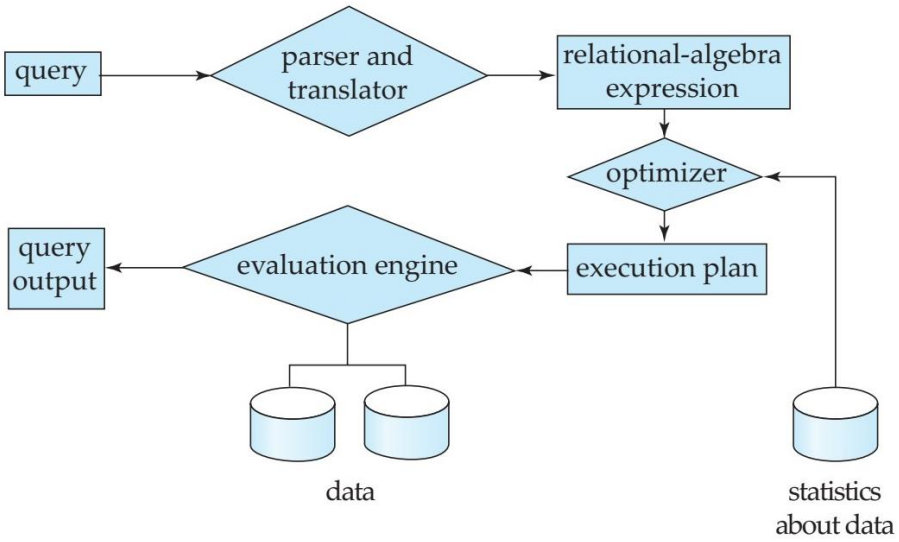
(7)

Find names of all customers who have an account at all branches located in Brooklyn city.

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account) \div \Pi_{branch\_name}(\sigma_{branch\_city = \text{"Brooklyn"}}(branch))$$

思路：先将 account 与 depositor 进行 natural join 取 customer_name 以及 branch_name 元素，后通过 branch_city=='Brooklyn'在 branch 中筛选出实例并取 branch_name 元素，最后将前者对后者进行 devision 操作达到通过 branch_name 铆合并去除 branch_name 的作用。

⚠ Query Processing:



3 main steps:

Parsing and translation

将查询转化为内部关系代数形式，然后解析器检查语法并验证关系。

Optimisation

关系代数拥有多种等价形式，每一个关系代数都可以使用算法进行优化，选择等价算法中成本最低的那一个，需求指定统一的评判标准来量化优化（如何衡量查询成本、如何组合用于单个操作的算法以求出一个

完整表达式的值、如何找到估计成本最低的评估计划)。

Evaluation

设计计划，执行计划，得出结果（评估查询步骤时间复杂度）

▲ Measures of Query Cost：

~disk access (ignore)

~CPU (real systems can't ignore, but we can)

~network communication (must be considered in parallel systems)

~response time (hard to estimate)

~total resource consumption (easy to minimize)

More specific:

- Disk cost can be estimated as:
  - Number of seeks (average-seek-cost)
  - Number of blocks read (average-block-read-cost)
  - Number of blocks written (average-block-write-cost)
- For simplicity, use the **number of block transfers** *from/to disk and the* **number of seeks** as the cost measures
  - Assume for simplicity that write cost is same as read cost
  - t_T – time to transfer one block
  - t_S – time for one seek
  - Cost for **b** block transfers plus **s** seeks

    $b * t_T + s * t_S$
- t_S and t_T depend on where data is stored; with 4 KB blocks:
  - High end magnetic disk: t_S = 4 msec and t_T =0.1 msec
  - SSD: t_S = 20-90 microsec and t_T = 2-10 microsec for 4KB
  - msec (millisecond or ms); microsec (microsecond or μs); 1ms=1,000μs