Assignment Operation

- Assign a relation expression to a variable

$$temp1 \leftarrow \pi_{R-S}(r)$$

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

$$result = temp1 - temp2$$

- Helps in writing sequential programs
- Difference from `rename` operation?

Generalized Projection

- Use arithmetic functions in projection list
- $\pi_{F1, F2, \ldots, Fn}(E)$
- $\pi_{\text{name, they\_owe - I\_owe}}(\text{friends})$
- $\pi_{\text{name, (they\_owe - I\_owe) as actual\_money\_owed}}(\text{friends})$

name	they_owe	i_owe
Avi	200	150
Rachel	100	250

name	actual_money_owed
Avi	50
Rachel	-150

Aggregate Functions

- Input → collection of values, output → single value
- `sum, avg, count, min, max`
- Aggregate functions can operate on **multisets**: multiple occurences of same value

Aggregate Function example

Query: Total salary paid to employees

$\mathcal{G}_{\texttt{sum(salary)}}(\texttt{employee})$

1200

Query: No. of employees

$\mathcal{G}_{\texttt{count(emp\_name)}}(\texttt{employee})$

6

emp_name	dept_name	salary
Fraust	Resouces	2000
Hugo	Testing	1300
Rao	Development	1200
Vanessa	Testing	2000
Chen	Resouces	1200
Wayne	Development	1400

employee relation

All aggregate functions take a "distinct" variation (since they work on multisets)

Query: No. of departments

$\mathcal{G}_{\texttt{count-distinct(dept\_name)}}(\texttt{employee})$

3

Grouping with aggregate functions

- If multiple values of an attribute, can group by it

Query: Total salary in each department

More sophisticated. Divide first by groups of department

$_{\texttt{dept\_name}}\mathcal{G}_{\texttt{sum(salary)}}(\texttt{employee})$

dept_name	sum(salary)
Resouces	3200
Testing	3300
Development	2600

emp_name	dept_name	salary
Fraust	Resouces	2000
Hugo	Testing	1300
Rao	Development	1200
Vanessa	Testing	2000
Chen	Resouces	1200
Wayne	Development	1400

employee relation

Query: Average & maximum salary in each department

$_{\texttt{dept\_name}}\mathcal{G}_{\texttt{avg(salary) as avg\_salary, max(salary) as max\_salary}}(\texttt{employee})$

dept_name	avg_salary	max_salary
Resouces	1600	2000
Testing	1650	2000
Development	1300	1400

Outer Join

- Extends natural join
- Deals with missing information
- `employee ⋈ employee_personal`
    - leaves out non-matching tuples

emp_name	dept_name	salary	street	city
Fraust	Resouces	2000	Mesa	Palo Alto
Hugo	Testing	1300	Walnut	North Manchester
Rao	Development	1200	Main	Oakland
Chen	Resouces	1200	Market	Carbondale
Wayne	Development	1400	Oak	Miami

- Outer join can make up these

emp_name	dept_name	salary
Fraust	Resouces	2000
Hugo	Testing	1300
Rao	Development	1200
Vanessa	Testing	2000
Chen	Resouces	1200
Wayne	Development	1400

`employee` relation

emp_name	street	city
Fraust	Mesa	Palo Alto
Hugo	Walnut	North Manchester
Rao	Main	Oakland
Jenna	Mesa	Palo Alto
Chen	Market	Carbondale
Wayne	Oak	Miami

`employee_personal` relation

Outer Join types

- Three types: ⟕, ⟗, ⟖

emp_name	dept_name	salary	street	city
Fraust	Resouces	2000	Mesa	Palo Alto
Hugo	Testing	1300	Walnut	North Manchester
Rao	Development	1200	Main	Oakland
Chen	Resouces	1200	Market	Carbondale
Wayne	Development	1400	Oak	Miami
Vanessa	Testing	2000	*null*	*null*

employee ⟕ employee_personal

emp_name	dept_name	salary	street	city
Fraust	Resouces	2000	Mesa	Palo Alto
Hugo	Testing	1300	Walnut	North Manchester
Rao	Development	1200	Main	Oakland
Chen	Resouces	1200	Market	Carbondale
Wayne	Development	1400	Oak	Miami
Jenna	*null*	*null*	Mesa	Palo Alto

employee ⟖ employee_personal

emp_name	dept_name	salary	street	city
Fraust	Resouces	2000	Mesa	Palo Alto
Hugo	Testing	1300	Walnut	North Manchester
Rao	Development	1200	Main	Oakland
Chen	Resouces	1200	Market	Carbondale
Wayne	Development	1400	Oak	Miami
Jenna	*null*	*null*	Mesa	Palo Alto
Vanessa	Testing	2000	*null*	*null*

employee ⟗ employee_personal

Null values

- “Non-existent” or “unknown” values
- Should be avoided if possible
- Arithmetic operation with null gives null
- Comparison (≤, <, >, ≥, =, ≠, …) will give *unknown*
- With booleans (like used in select, which itself is used a lot)
  - and: (true **and** unknown) = unknown, (false **and** unknown) = false, (unknown **and** unknown) = unknown
  - or: (true **or** unknown) = true, (false **or** unknown) = unknown, (unknown **or** unknown) = unknown
  - not: (**not** unknown) = unknown