Relational Model

Dr. Odelu Vanga

Indian Institute of Information Technology Sri City

http://www.iiits.ac.in/people/regular-faculty/dr-odelu-vanga/

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- Set differences must be taken between *compatible* relations.
  - r and s must have the *same arity*
  - attribute domains of r and s must be compatible

Compatibility of R1 and R2

- arity(R1) = arity(R2)
- the corresponding attribute domains in R1 and R2 are the same

Tuple arity: the number of values in the sequence (including nulls)

Set Difference Operation – Example

- Relations *r*, *s*:

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

r – *s*:

A	B
α	1
β	1

Find all customers who have an account but no loan.

(**select** *customer-name* **from** *depositor*)
except
(select *customer-name* **from** *borrower)*

OR

```
select customer_name
from depositor
where customer_name not in
      (select customer_name
        from borrower);
```

Cartesian-Product Operation

- Notation $r \times s$
- Defined as:

$$r \times s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

Cartesian-Product Operation-Example

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 x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	19	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

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$

$r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	19	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

$\sigma_{A=C}(r \times s)$

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

from Clause - Cartesian

- The **from** clause corresponds to the Cartesian product operation of the relational algebra. It lists the relations to be scanned in the evaluation of the expression.
- Find the Cartesian product *borrower x loan*

  select *
  from *borrower, loan*

- Find the name, loan number and loan amount of all customers having a loan at the Perryridge branch.

  select *customer-name, borrower.loan-number, amount*
  from *borrower, loan*
  where *borrower.loan-number = loan.loan-number* **and**
  branch-name = 'Perryridge'

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.

Example:

$$\rho_{x}(E)$$

returns the expression *E* under the name *X*

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

$$\rho_{x\ (A1, A2, \ldots, An)}(E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to $A1, A2, \ldots, An$.

The Rename Operation

- The SQL allows renaming relations and attributes using the **as** clause:

  old-name **as** *new-name*

- Find the name, loan number and loan amount of all customers; rename the column name *loan-number* as *loan-id*.

select *customer-name, borrower.loan-number* **as** *loan-id, amount*
from *borrower, loan*
where *borrower.loan-number = loan.loan-number*

Rename - Tuple Variables

- Tuple variables are defined in the **from** clause via the use of the **as** clause.
- Find the customer names and their loan numbers for all customers having a loan at some branch.

  select *customer-name, B.loan-number, L.amount*
  from *borrower* **as** *B, loan* **as** *L*
  where *B.loan-number* = *L.loan-number*

- Find the names of all branches that have greater assets than some branch located in Brooklyn.

  select distinct *T.branch-name*
  from *branch* **as** *T, branch* **as** *S*
  where *T.assets* > *S.assets* **and** *S.branch-city* = 'Brooklyn'

Ordering the Display of Tuples

- List in alphabetic order the names of all customers having a loan in Perryridge branch

  select distinct *customer-name*
  from *borrower, loan*
  where *borrower.loan-number* = *loan.loan-number* **and**
  branch-name = 'Perryridge'
  order by *customer-name*

- We may specify **desc** for descending order or **asc** for ascending order, for each attribute; ascending order is the default.
  - E.g. **order by** *customer-name* **desc**

Duplicates

- In relations with duplicates, SQL can define how many copies of tuples appear in the result.
- ***Multiset*** versions of some of the relational algebra operators – given multiset relations r_1 and r_2:
  1. If there are c_1 copies of tuple t_1 in r_1, and t_1 satisfies selections σ_{θ}, then there are c_1 copies of t_1 in $\sigma_{\theta}(r_1)$.
  2. For each copy of tuple t_1 in r_1, there is a copy of tuple $\Pi_A(t_1)$ in $\Pi_A(r_1)$, where $\Pi_A(t_1)$ denotes the projection of the single tuple t_1.
  3. If there are c_1 copies of tuple t_1 in r_1 and c_2 copies of tuple t_2 in r_2, there are c_1 x c_2 copies of the tuple $t_1.\,t_2$ in r_1 x r_2

Duplicates (Cont.)

- Example: Suppose multiset relations r_1 (A, B) and r_2 (C) are as follows:

$$r_1 = \{(1, a)\ (2,a)\} \qquad r_2 = \{(2), (3), (3)\}$$

- Then $\Pi_B(r_1)$ would be $\{(\mathrm{a}), (\mathrm{a})\}$, while $\Pi_B(r_1)$ x r_2 would be

$$\{(a,2), (a,2), (a,3), (a,3), (a,3), (a,3)\}$$

- SQL duplicate semantics:

select $A_1,, A_2, \ldots, A_n$
from $r_1, r_2, \ldots, r_m$
where P

is equivalent to the ***multiset*** version of the expression:

$$\Pi_{A1,, A2, \ldots, An}(\sigma_P\ (r_1 \text{ x } r_2 \text{ x} \ldots \text{x } \mathrm{r}_m))$$

Example Queries

- Find the names of all customers who have a loan, an account, or both, from the bank

$$\prod_{customer\text{-}name}(borrower) \cup \prod_{customer\text{-}name}(depositor)$$

(select *customer-name* **from** *depositor*) **union (select** *customer-name* **from** *borrower)*

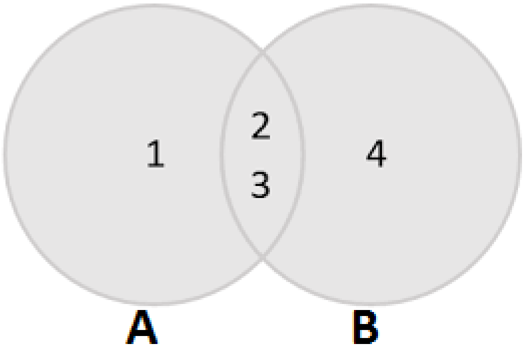
- Find the names of all customers who have a loan and an account at bank.

$$\prod_{customer\text{-}name}(borrower) \cap \prod_{customer\text{-}name}(depositor)$$

(**select** *customer-name* **from** *depositor*) **intersect (select** *customer-name* **from** *borrower)*

OR

select customer_name
from depositor
where customer_name **in**
(select customer_name
from borrower);



Example Queries

- Find the names of all customers who have a loan at the Perryridge branch.
  - Query 1

$$\Pi_{\text{customer-name}}(\sigma_{\text{branch-name} = \text{“Perryridge”}}$$
$$(\sigma_{\text{borrower.loan-number} = \text{loan.loan-number}}(\text{borrower x loan})))$$

  - Query 2

$$\Pi_{\text{customer-name}}(\sigma_{\text{loan.loan-number} = \text{borrower.loan-number}}($$
$$(\sigma_{\text{branch-name} = \text{“Perryridge”}}(\text{loan})) \text{ x borrower}))$$

Example Queries

- Find the largest account balance and rename *account* relation as *d*

Expression:

$$\Pi_{balance}(account) - \Pi_{account.balance}\,(\sigma_{account.balance\,<\,d.balance}\,(account\ x\ \rho_d\,(account)))$$

SQL query:

select balance
from account
where balance not in
(**select** account.balance
from account, account as d
where accountnt.balance < d.balance);

Example Queries

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

$$\Pi_{customer\text{-}name}\,(\sigma_{branch\text{-}name=\text{“}Perryridge\text{”}}\,(\sigma_{borrower.loan\text{-}number\,=\,loan.loan\text{-}number}(borrower\ x\ loan)))$$

- **(Try Own)** Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.

$$\Pi_{customer\text{-}name}\,(\sigma_{branch\text{-}name\,=\,\text{“}Perryridge\text{”}}\,(\sigma_{borrower.loan\text{-}number\,=\,loan.loan\text{-}number}(\text{borrower x loan})))\ -\ \Pi_{customer\text{-}name}(\text{depositor})$$

THANK YOU