Relational Theory
Part 2

UAlbany ICSI 410
Fall 2016

Much of the material in these slides
is taken directly from

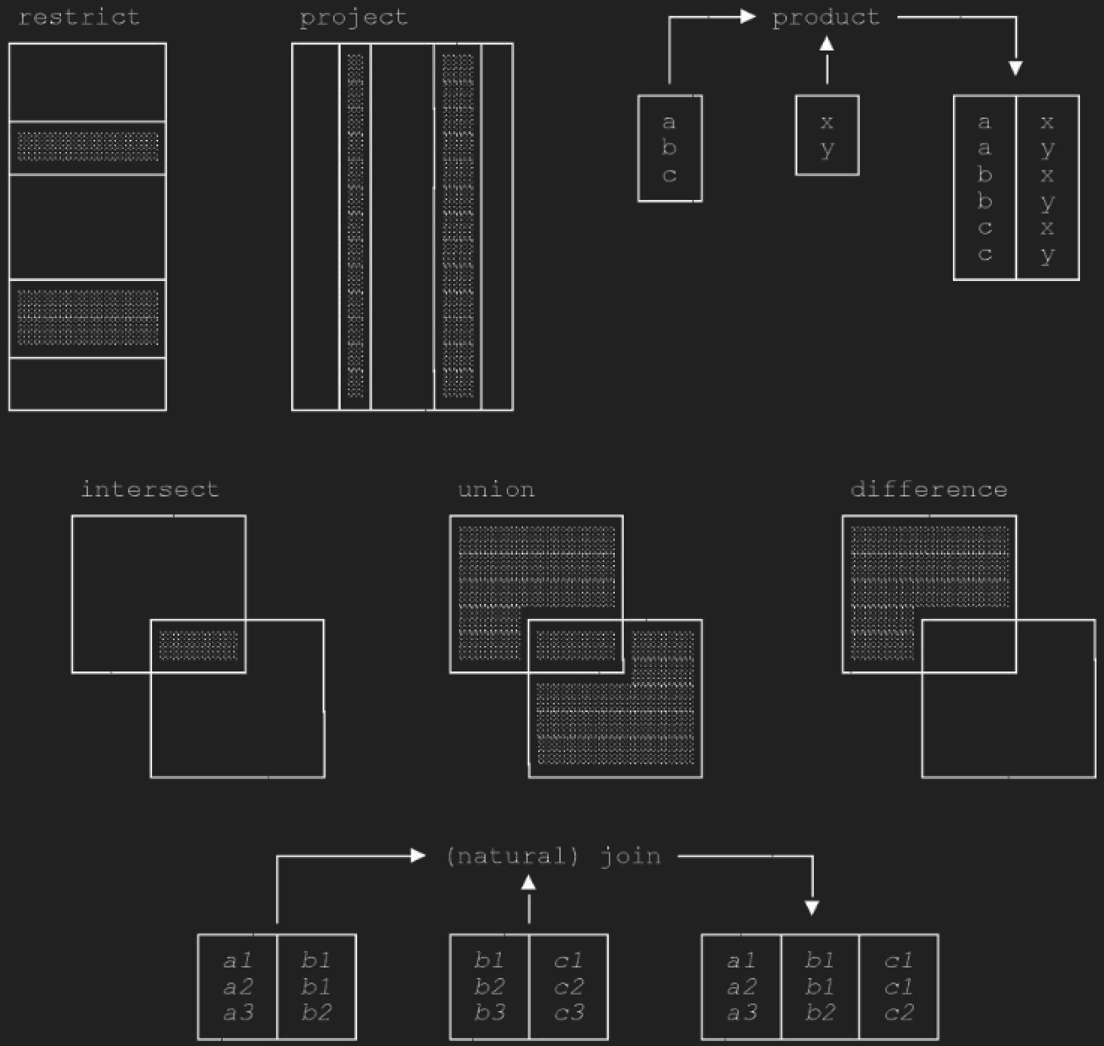
SQL and Relational Theory by C.J. Date

and

Database System Concepts by Silbershatz et al.

Relational Algebra

(Continued)



Relational Algebra: Operations

RENAME:

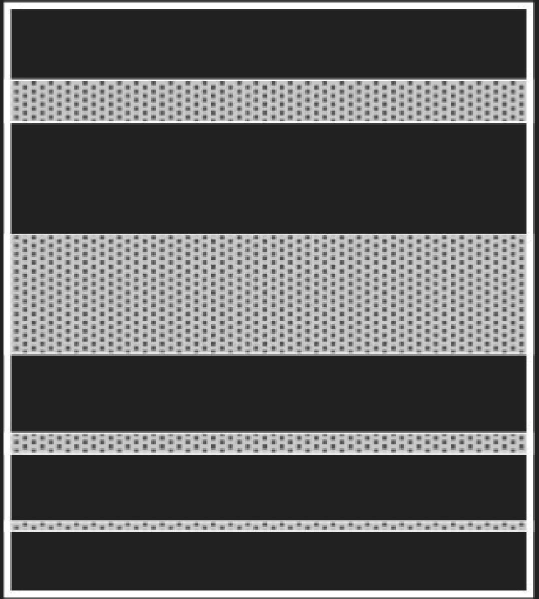
- Denoted by lowercase Greek letter rho (ρ)
- Unlike relations in the database, the results of relational-algebra expressions do not have a name that we can use to refer to them.
- It is useful to be able to give them names.

Relational Algebra: Operations

SELECT (aka RESTRICT):

- Denoted by lowercase Greek letter sigma (σ)
- "Selects" tuples that satisfy a given predicate.
- The predicate appears as a subscript to σ
- The argument relation is in parentheses after the σ

SELECT



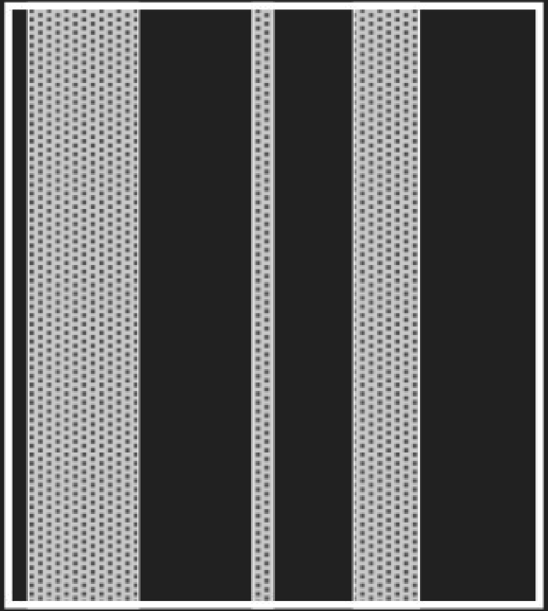
- NOTE: The term "select" in relational algebra has a different meaning than the one used in SQL. In relational algebra, the term "select" corresponds to SQL's WHERE.

PROJECT

Relational Algebra: Operations

PROJECT:

- Denoted by uppercase Greek letter pi (Π)
- Unary operation that returns its argument relation, with certain attributes left out.
- Since a relation (body) is a set, any duplicate rows are eliminated.
- We list those attributes that we wish to appear in the result as a subscript to Π.
- The argument relation is in parentheses after the Π.

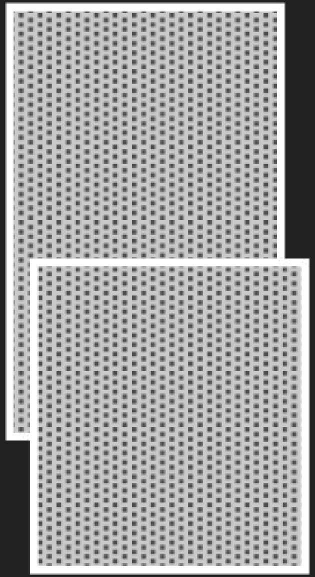


UNION

Relational Algebra: Operations

UNION:

- Denoted, as in set theory, by $\cup$.
- Binary operation that returns the set theory union of the bodies of the two argument relations.
- We must make sure that unions are taken between *compatible* relations. (More on this in a later slide.)

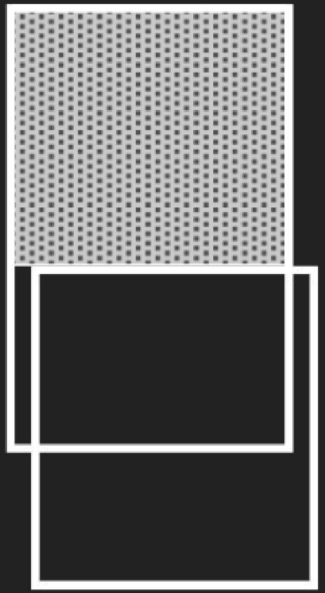


DIFFERENCE

Relational Algebra: Operations

Set-Difference:

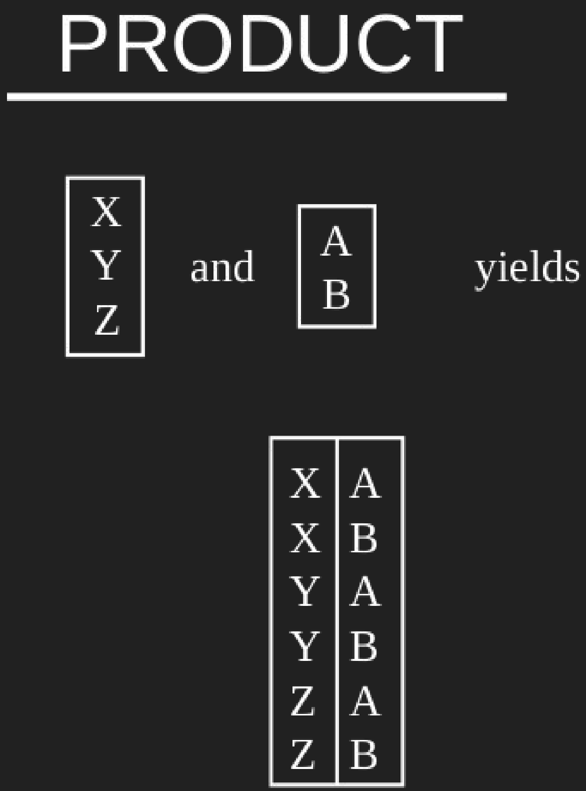
- Denoted by $-$
- Binary operation that allows us to find tuples that are in one relation but not another.
- $r - s$ produces a relation containing those tuples that are in *r* but not *s*.
- As with the union operation, we must insure that set differences are taken between compatible relations.



Relational Algebra: Operations

PRODUCT:

- Denoted by a cross (X)
- Binary operation that allows us to combine information from (any) two relations.
- Recall that a relation is a subset of the Cartesian product of a set of attribute domains.



X
Y
Z

A
B

X	A
X	B
Y	A
Y	B
Z	A
Z	B

Relational Algebra: Operations

The operations that we covered so far allow us to give us a complete definition of an expression in the relational algebra.

- $E_1 \cup E_2$
- $E_1 - E_2$
- $E_1 \times E_2$
- $\sigma_P(E_1)$
- $\Pi_S(E_1)$
- $\rho_x(E_1)$

These fundamental operations of the relational algebra are sufficient to express any relational-algebra query.

However, if we restrict ourselves to just these operations, certain common queries are lengthy to express.

Relational Algebra: Operations

The operations that we covered so far allow us to give us a complete definition of an expression in the relational algebra.

- *Set-intersection*
- *Natural-Join*
- *Theta-Join*
- *Left outer join*
- *Right outer join*
- *Full outer join*

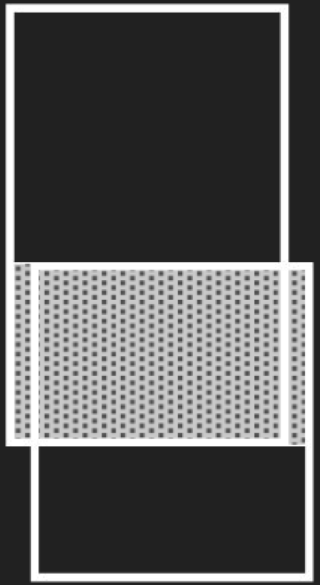
The next set of slides will cover these operations that do not add any power to the algebra, but which simplify common queries.

INTERSECTION

Relational Algebra: Operations

Set Intersection:

- Definition (from Date)
    - Let relations *r1* and *r2* be of the same type (have the same heading)
    - Then their intersection *r1* INTERSECT *r2* is a relation of the same type
        - With body consisting of all tuples *t* such that *t* appears in both *r1* and *r2*.

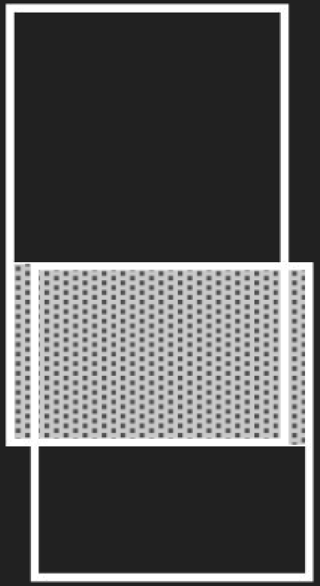


INTERSECTION

Relational Algebra: Operations

Set Intersection:

- Recall that Set Intersection adds no power to the relational algebra.
- It can be expressed using just one operation from the fundamental set of operators.



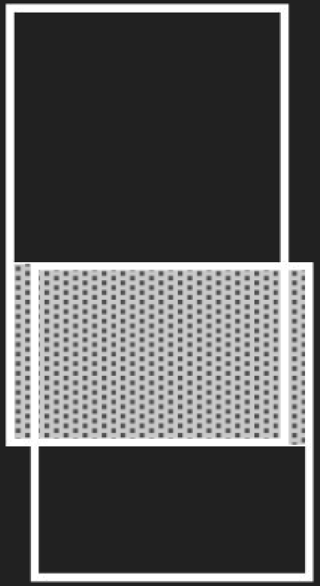
INTERSECTION

Relational Algebra: Operations

Set Intersection:

- Recall that Set Intersection adds no power to the relational algebra.
- It can be expressed using just one operation from the fundamental set of operators.

  Which one?



INTERSECTION

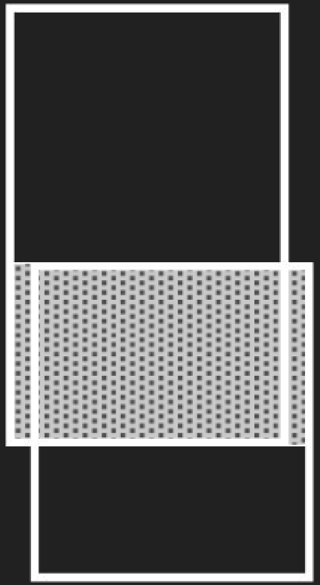
Relational Algebra: Operations

Set Intersection:

- Recall that Set Intersection adds no power to the relational algebra.
- It can be expressed using just one operation from the fundamental set of operators.

Which one?

The Difference Operator: $r - s$



Relational Algebra: Operations

INTERSECTION

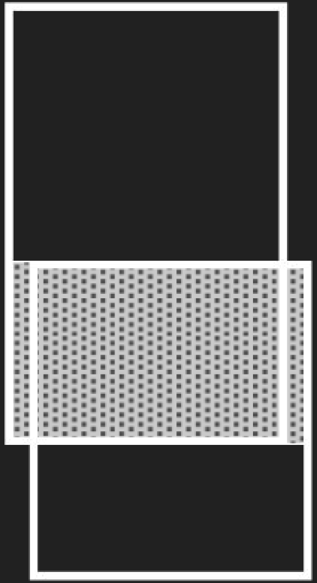
Set Intersection:

- Recall that Set Intersection adds no power to the relational algebra.
- It can be expressed using just one operation from the fundamental set of operators.

Which one?

The Difference Operator: $r - s$

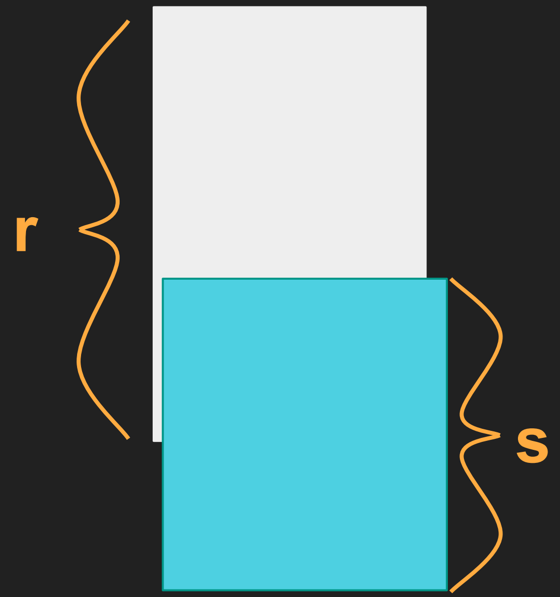
$$r \cap s = r - (r - s)$$



Relational Algebra: Operations

INTERSECTION

Set Intersection:

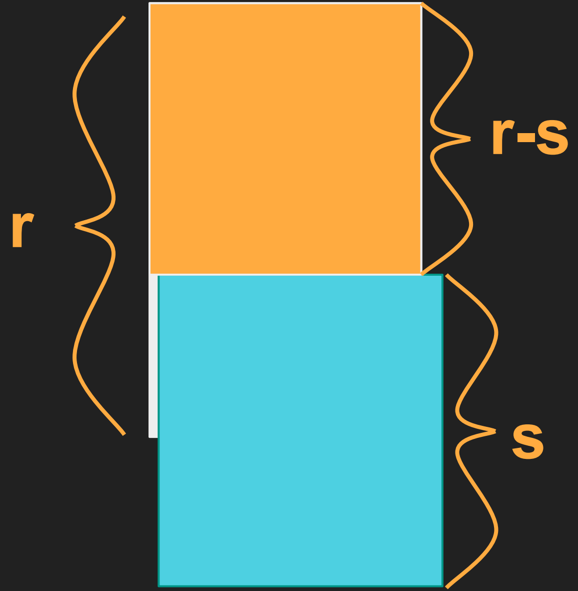


$$r \cap s = r - (r - s)$$

Relational Algebra: Operations

INTERSECTION

Set Intersection:

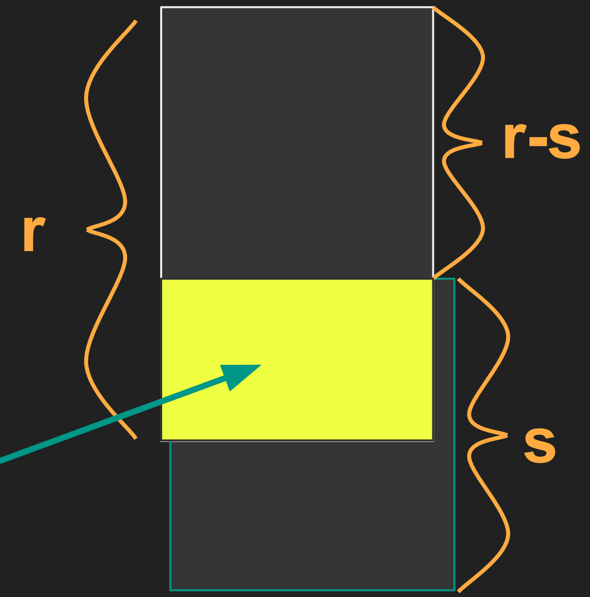


$$r \cap s = r - (r - s)$$

INTERSECTION

Relational Algebra: Operations

Set Intersection:



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

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

CUSTOMER_ACCOUNT_BALANCES		
cust_id	account_id	avail_balance
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

CUSTOMER_TYPES	
cust_id	cust_type_cd
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	I
10	B
11	B
12	B

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

RESTRICT CUSTOMER_TYPES tuples to individuals

$$\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES})$$

CUSTOMER_ACCOUNT_BALANCES		
cust_id	**account_id**	**avail_balance**
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

CUSTOMER_TYPES	
cust_id	**cust_type_cd**
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	I
10	B
11	B
12	B

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

RESTRICT CUSTOMER_TYPES tuples to individuals

$$\sigma_{\text{cust\_type\_cd} \ = \ \text{“I”}}(\text{CUSTOMER\_TYPES})$$

CUSTOMER_ACCOUNT_BALANCES

cust_id	account_id	avail_balance
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

CUSTOMER_TYPES

cust_id	cust_type_cd
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	I
10	B
11	B
12	B

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

RESTRICT CUSTOMER_TYPES tuples to individuals

$$\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES})$$

CUSTOMER_ACCOUNT_BALANCES		
cust_id	**account_id**	**avail_balance**
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

cust_id	cust_type_cd
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	I

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

PROJECT on cust_id

$$\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(\text{CUSTOMER\_TYPES}))$$

CUSTOMER_ACCOUNT_BALANCES		
cust_id	**account_id**	**avail_balance**
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

cust_id	cust_type_cd
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	I

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

PROJECT on cust_id

$$\pi_{cust\_id}(\sigma_{cust\_type\_cd \ = \text{“I”}}(\text{CUSTOMER\_TYPES}))$$

CUSTOMER_ACCOUNT_BALANCES		
cust_id	**account_id**	**avail_balance**
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

cust_id	cust_type_cd
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	I

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

PROJECT on cust_id

$$\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(\text{CUSTOMER\_TYPES}))$$

CUSTOMER_ACCOUNT_BALANCES		
cust_id	**account_id**	**avail_balance**
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

cust_id
1
2
3
4
5
6
7
8
9

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

CUSTOMER_ACCOUNT_BALANCES		
cust_id	account_id	avail_balance
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

cust_id
1
2
3
4
5
6
7
8
9

$\pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

SELECT tuples
from CUSTOMER_ACCOUNT_BALANCES
with avail_balance greater than 1,000

$\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES})$

CUSTOMER_ACCOUNT_BALANCES		
cust_id	account_id	avail_balance
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

cust_id
1
2
3
4
5
6
7
8
9

$\pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

SELECT tuples
from CUSTOMER_ACCOUNT_BALANCES
with avail_balance greater than 1,000

$$\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES})$$

CUSTOMER_ACCOUNT_BALANCES		
cust_id	**account_id**	**avail_balance**
1	1	1057.75
1	2	500
1	3	3000
2	4	2258.02
2	5	200
3	7	1057.75
3	8	2212.5
4	10	534.12
4	11	767.77
4	12	5487.09

cust_id
1
2
3
4
5
6
7
8
9

$$\pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))$$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

cust_id	account_id	avail_balance
1	1	1057.75
2	4	2258.02
3	7	1057.75
3	8	2212.5
4	12	5487.09

cust_id
1
2
3
4
5
6
7
8
9

SELECT tuples
from CUSTOMER_ACCOUNT_BALANCES
with avail_balance greater than 1,000

$\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES})$

$\pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

cust_id	account_id	avail_balance
1	1	1057.75
2	4	2258.02
3	7	1057.75
3	8	2212.5
4	12	5487.09

cust_id
1
2
3
4
5
6
7
8
9

PROJECT cust_id

$\pi_{\text{cust\_id}}(\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES}))$

$\pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

cust_id	account_id	avail_balance
1	1	1057.75
2	4	2258.02
3	7	1057.75
3	8	2212.5
4	12	5487.09

cust_id
1
2
3
4
5
6
7
8
9

PROJECT cust_id

$$\pi_{cust\_id}(\sigma_{avail\_balance > 1000}(CUSTOMER\_ACCOUNT\_BALANCES))$$

$$\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES))$$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

PROJECT cust_id

$$\Pi_{cust\_id}(\sigma_{avail\_balance > 1000}(CUSTOMER\_ACCOUNT\_BALANCES))$$

cust_id
1
2
3
3
4

cust_id
1
2
3
4
5
6
7
8
9

$$\Pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES))$$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

cust_id
1
2
3
3
4

cust_id
1
2
3
4
5
6
7
8
9

PROJECT cust_id

$$\pi_{cust\_id}(\sigma_{avail\_balance\ >\ 1000}(CUSTOMER\_ACCOUNT\_BALANCES))$$

$$\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES))$$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

cust_id
1
2
3
4

cust_id
1
2
3
4
5
6
7
8
9

PROJECT cust_id

$$\Pi_{\text{cust\_id}}(\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES}))$$

$$\Pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))$$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

cust_id
1
2
3
4

cust_id
1
2
3
4
5
6
7
8
9

Take the INTERSECTION of the two relations

$$\Pi_{\text{cust\_id}}(\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES})) \cap \Pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))$$

Relational Algebra: Operations

cust_id	cust_id
1	1
2	2
3	3
4	4
	5
	6
	7
	8
	9

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

Take the INTERSECTION of the two relations

$$\pi_{cust\_id}(\sigma_{avail\_balance\ >\ 1000}(CUSTOMER\_ACCOUNT\_BALANCES)) \cap \pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES))$$

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

Take the INTERSECTION of the two relations

$$\pi_{cust\_id}(\sigma_{avail\_balance > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES})) \cap \pi_{cust\_id}(\sigma_{cust\_type\_cd \; = \text{“I”}}(\text{CUSTOMER\_TYPES}))$$

cust_id
1
2
3
4

cust_id
1
2
3
4
5
6
7
8
9

Relational Algebra: Operations

INTERSECTION Example

Required Predicate:

customer_id ***i*** belongs to an individual with an available balance greater than 1,000

cust_id
1
2
3
4

$$\Pi_{\text{cust\_id}}(\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES})) \cap \Pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))$$

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with an available balance greater than 1,000

CUSTOMER_ACCOUNT_BALANCES

cust_id	account_id	avail_balance
1	1	1057.75
1	2	500.00
1	3	3000.00
2	4	2258.02
2	5	200.00
3	7	1057.75
3	8	2212.50
4	10	534.12
4	11	767.77
4	12	5487.09
5	13	2237.97
6	14	122.37
6	15	10000.00
7	17	5000.00
8	18	3487.19
8	19	387.99
9	21	125.67
9	22	9345.55
9	23	1500.00
10	24	23575.12
10	25	0.00
11	27	9345.55
12	28	38552.05
13	29	50000.00

CUSTOMER_TYPES

cust_id	cust_type_cd
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	I
10	B
11	B
12	B

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with an available balance greater than 1,000

We can pick up from here.

$\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES})$

CUSTOMER_ACCOUNT_BALANCES		
cust_id	**account_id**	**avail_balance**
1	1	1057.75
1	3	3000.00
2	4	2258.02
3	7	1057.75
3	8	2212.50
4	12	5487.09
5	13	2237.97
6	15	10000.00
7	17	5000.00
8	18	3487.19
9	22	9345.55
9	23	1500.00
10	24	23575.12
11	27	9345.55
12	28	38552.05
13	29	50000.00

cust_id
1
2
3
4
5
6
7
8
9

$\pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))$

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with an available balance greater than 1,000

We will need to do a PRODUCT, therefore we need to RENAME the columns in one table:

$$\sigma_{avail\_balance > 1000}(CUSTOMER\_ACCOUNT\_BALANCES)$$

CUSTOMER_ACCOUNT_BALANCES		
cust_id	account_id	avail_balance
1	1	1057.75
1	3	3000.00
2	4	2258.02
3	7	1057.75
3	8	2212.50
4	12	5487.09
5	13	2237.97
6	15	10000.00
7	17	5000.00
8	18	3487.19
9	22	9345.55
9	23	1500.00
10	24	23575.12
11	27	9345.55
12	28	38552.05
13	29	50000.00

cust_id
1
2
3
4
5
6
7
8
9

$$\rho_{CTYPES(ct\_cust\_id)}(\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES)))$$

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with an available balance greater than 1,000

Now we can get the PRODUCT

$\sigma_{\text{avail\_balance > 1000}}(\text{CUSTOMER\_ACCOUNT\_BALANCES}) \times$

$\rho_{\text{CTYPES(ct\_cust\_id)}}(\Pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd = “I”}}(\text{CUSTOMER\_TYPES})))$

CUSTOMER_ACCOUNT_BALANCES

cust_id	account_id	avail_balance
1	1	1057.75
1	3	3000.00
2	4	2258.02
3	7	1057.75
3	8	2212.50
4	12	5487.09
5	13	2237.97
6	15	10000.00
7	17	5000.00
8	18	3487.19
9	22	9345.55
9	23	1500.00
10	24	23575.12
11	27	9345.55
12	28	38552.05
13	29	50000.00

CTYPES

ct_cust_id
1
2
3
4
5
6
7
8
9

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with an available balance greater than 1,000

Now we can get the PRODUCT

$$\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES}) \times \rho_{\text{CTYPES(ct\_cust\_id)}}(\Pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES})))$$

cust_id	account_id	avail_balance	ct_cust_id
1	1	1057.75	1
1	3	3000.00	1
2	4	2258.02	1
3	7	1057.75	1
3	8	2212.50	1
4	12	5487.09	1
5	13	2237.97	1
6	15	10000.00	1
7	17	5000.00	1
8	18	3487.19	1
9	22	9345.55	1
9	23	1500.00	1
10	24	23575.12	1
11	27	9345.55	1
12	28	38552.05	1
13	29	50000.00	1
1	1	1057.75	2
1	3	3000.00	2
2	4	2258.02	2
3	7	1057.75	2
3	8	2212.50	2
4	12	5487.09	2
5	13	2237.97	2
6	15	10000.00	2
7	17	5000.00	2
8	18	3487.19	2
9	22	9345.55	2
9	23	1500.00	2
10	24	23575.12	2
11	27	9345.55	2
12	28	38552.05	2
13	29	50000.00	2
1	1	1057.75	3
1	3	3000.00	3
2	4	2258.02	3
3	7	1057.75	3
3	8	2212.50	3
4	12	5487.09	3
5	13	2237.97	3
6	15	10000.00	3
7	17	5000.00	3
8	18	3487.19	3
9	22	9345.55	3
9	23	1500.00	3
10	24	23575.12	3
11	27	9345.55	3
12	28	38552.05	3
13	29	50000.00	3
1	1	1057.75	4
1	3	3000.00	4
2	4	2258.02	4
3	7	1057.75	4
3	8	2212.50	4
4	12	5487.09	4
5	13	2237.97	4
6	15	10000.00	4
7	17	5000.00	4
8	18	3487.19	4
9	22	9345.55	4
9	23	1500.00	4
10	24	23575.12	4
11	27	9345.55	4
12	28	38552.05	4
13	29	50000.00	4
1	1	1057.75	5
1	3	3000.00	5
2	4	2258.02	5
3	7	1057.75	5
3	8	2212.50	5
4	12	5487.09	5
5	13	2237.97	5
6	15	10000.00	5
7	17	5000.00	5
8	18	3487.19	5
9	22	9345.55	5
9	23	1500.00	5
10	24	23575.12	5
11	27	9345.55	5
12	28	38552.05	5
13	29	50000.00	5
1	1	1057.75	6
1	3	3000.00	6
2	4	2258.02	6
3	7	1057.75	6
3	8	2212.50	6
4	12	5487.09	6
5	13	2237.97	6
6	15	10000.00	6
7	17	5000.00	6
8	18	3487.19	6
9	22	9345.55	6
9	23	1500.00	6
10	24	23575.12	6
11	27	9345.55	6
12	28	38552.05	6
13	29	50000.00	6
1	1	1057.75	7
1	3	3000.00	7
2	4	2258.02	7
3	7	1057.75	7
3	8	2212.50	7
4	12	5487.09	7
5	13	2237.97	7
6	15	10000.00	7
7	17	5000.00	7
8	18	3487.19	7
9	22	9345.55	7
9	23	1500.00	7
10	24	23575.12	7
11	27	9345.55	7
12	28	38552.05	7
13	29	50000.00	7
1	1	1057.75	8
1	3	3000.00	8
2	4	2258.02	8
3	7	1057.75	8
3	8	2212.50	8
4	12	5487.09	8
5	13	2237.97	8
6	15	10000.00	8
7	17	5000.00	8
8	18	3487.19	8
9	22	9345.55	8
9	23	1500.00	8
10	24	23575.12	8
11	27	9345.55	8
12	28	38552.05	8
13	29	50000.00	8
1	1	1057.75	9
1	3	3000.00	9
2	4	2258.02	9
3	7	1057.75	9
3	8	2212.50	9
4	12	5487.09	9
5	13	2237.97	9
6	15	10000.00	9
7	17	5000.00	9
8	18	3487.19	9
9	22	9345.55	9
9	23	1500.00	9
10	24	23575.12	9
11	27	9345.55	9
12	28	38552.05	9
13	29	50000.00	9

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with an available balance greater than 1,000

We now need to select the rows where (cust_id = ct_cust_id)

$$\sigma_{\text{cust\_id} = \text{ct\_cust\_id}}(\sigma_{\text{avail\_balance} > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES}) \times \rho_{\text{CTYPES(ct\_cust\_id)}}(\Pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd} = \text{“I”}}(\text{CUSTOMER\_TYPES}))))$$

cust_id	account_id	avail_balance	ct_cust_id
1	1	1057.75	1
1	3	3000.00	1
2	4	2258.02	1
3	7	1057.75	1
3	8	2212.50	1
4	12	5487.09	1
5	13	2237.97	1
6	15	10000.00	1
7	17	5000.00	1
8	18	3487.19	1
9	22	9345.55	1
9	23	1500.00	1
10	24	23575.12	1
11	27	9345.55	1
12	28	38552.05	1
13	29	50000.00	1
1	1	1057.75	2
1	3	3000.00	2
2	4	2258.02	2
3	7	1057.75	2
3	8	2212.50	2
4	12	5487.09	2
5	13	2237.97	2
6	15	10000.00	2
7	17	5000.00	2
8	18	3487.19	2
9	22	9345.55	2
9	23	1500.00	2
10	24	23575.12	2
11	27	9345.55	2
12	28	38552.05	2
13	29	50000.00	2
1	1	1057.75	3
1	3	3000.00	3
2	4	2258.02	3
3	7	1057.75	3
3	8	2212.50	3
4	12	5487.09	3
5	13	2237.97	3
6	15	10000.00	3
7	17	5000.00	3
8	18	3487.19	3
9	22	9345.55	3
9	23	1500.00	3
10	24	23575.12	3
11	27	9345.55	3
12	28	38552.05	3
13	29	50000.00	3
1	1	1057.75	4
1	3	3000.00	4
2	4	2258.02	4
3	7	1057.75	4
3	8	2212.50	4
4	12	5487.09	4
5	13	2237.97	4
6	15	10000.00	4
7	17	5000.00	4
8	18	3487.19	4
9	22	9345.55	4
9	23	1500.00	4
10	24	23575.12	4
11	27	9345.55	4
12	28	38552.05	4
13	29	50000.00	4
1	1	1057.75	5
1	3	3000.00	5
2	4	2258.02	5
3	7	1057.75	5
3	8	2212.50	5
4	12	5487.09	5
5	13	2237.97	5
6	15	10000.00	5
7	17	5000.00	5
8	18	3487.19	5
9	22	9345.55	5
9	23	1500.00	5
10	24	23575.12	5
11	27	9345.55	5
12	28	38552.05	5
13	29	50000.00	5
1	1	1057.75	6
1	3	3000.00	6
2	4	2258.02	6
3	7	1057.75	6
3	8	2212.50	6
4	12	5487.09	6
5	13	2237.97	6
6	15	10000.00	6
7	17	5000.00	6
8	18	3487.19	6
9	22	9345.55	6
9	23	1500.00	6
10	24	23575.12	6
11	27	9345.55	6
12	28	38552.05	6
13	29	50000.00	6
1	1	1057.75	7
1	3	3000.00	7
2	4	2258.02	7
3	7	1057.75	7
3	8	2212.50	7
4	12	5487.09	7
5	13	2237.97	7
6	15	10000.00	7
7	17	5000.00	7
8	18	3487.19	7
9	22	9345.55	7
9	23	1500.00	7
10	24	23575.12	7
11	27	9345.55	7
12	28	38552.05	7
13	29	50000.00	7
1	1	1057.75	8
1	3	3000.00	8
2	4	2258.02	8
3	7	1057.75	8
3	8	2212.50	8
4	12	5487.09	8
5	13	2237.97	8
6	15	10000.00	8
7	17	5000.00	8
8	18	3487.19	8
9	22	9345.55	8
9	23	1500.00	8
10	24	23575.12	8
11	27	9345.55	8
12	28	38552.05	8
13	29	50000.00	8
1	1	1057.75	9
1	3	3000.00	9
2	4	2258.02	9
3	7	1057.75	9
3	8	2212.50	9
4	12	5487.09	9
5	13	2237.97	9
6	15	10000.00	9
7	17	5000.00	9
8	18	3487.19	9
9	22	9345.55	9
9	23	1500.00	9
10	24	23575.12	9
11	27	9345.55	9
12	28	38552.05	9
13	29	50000.00	9

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with an available balance greater than 1,000

cust_id	account_id	avail_balance	ct_cust_id
1	1	1057.75	1
1	3	3000.00	1
2	4	2258.02	2
3	7	1057.75	3
3	8	2212.50	3
4	12	5487.09	4
5	13	2237.97	5
6	15	10000.00	6
7	17	5000.00	7
8	18	3487.19	8
9	22	9345.55	9
9	23	1500.00	9

We now need to select the rows where (cust_id = ct_cust_id)

$$\sigma_{\text{cust\_id = ct\_cust\_id}}(\sigma_{\text{avail\_balance > 1000}}(\text{CUSTOMER\_ACCOUNT\_BALANCES}) \times \rho_{\text{CTYPES(ct\_cust\_id)}}(\Pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd = “I”}}(\text{CUSTOMER\_TYPES}))))$$

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with an available balance greater than 1,000

cust_id	account_id	avail_balance	ct_cust_id
1	1	1057.75	1
1	3	3000.00	1
2	4	2258.02	2
3	7	1057.75	3
3	8	2212.50	3
4	12	5487.09	4
5	13	2237.97	5
6	15	10000.00	6
7	17	5000.00	7
8	18	3487.19	8
9	22	9345.55	9
9	23	1500.00	9

Finally, we project on account_id:

$$\pi_{\text{account\_id}}(\sigma_{\text{cust\_id = ct\_cust\_id}}(\sigma_{\text{avail\_balance > 1000}}(\text{CUSTOMER\_ACCOUNT\_BALANCES}) \times \rho_{\text{CTYPES(ct\_cust\_id)}}(\pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd = "I"}}(\text{CUSTOMER\_TYPES})))))$$

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with an available balance greater than 1,000

Finally, we project on account_id:

$$\pi_{\text{account\_id}}(\sigma_{\text{cust\_id = ct\_cust\_id}}(\sigma_{\text{avail\_balance > 1000}}(\text{CUSTOMER\_ACCOUNT\_BALANCES}) \times \rho_{\text{CTYPES(ct\_cust\_id)}}(\pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd = “I”}}(\text{CUSTOMER\_TYPES})))))$$

account_id
1
3
4
7
8
12
13
15
17
18
22
23

Relational Algebra: Operations

Another Example

Required Predicate:

account_id ***a*** belongs to an individual with
an available balance greater than 1,000

$$\pi_{account\_id}(\sigma_{cust\_id = ct\_cust\_id}(\sigma_{avail\_balance > 1000}(CUSTOMER\_ACCOUNT\_BALANCES) \times \rho_{CTYPES(ct\_cust\_id)}(\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES)))))$$

When working with the relational model, it is very common to combine information from multiple tables.

Operators are defined to *simplify* the expression of these types of queries.

Relational Algebra: Operations

The JOIN operations (from Silberschatz)

- Allow the combining of two relations by merging pairs of tuples, one from each relation, into a single tuple.
- There are a number of different ways to join relations.

JOIN

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

Relational Algebra: Operations

JOIN operations (from Silberschatz, with Date's terminology)

- Output Relation Schema Convention:
  - We do not repeat those attributes that appear in the schemas of both relations.
  - The ordering of attributes is as follows:
    - First the attributes common to the schemas of both relations
    - Second those attributes unique to the schema of the first relation
    - Finally, those attributes unique to the schema of the second relation.

JOIN

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

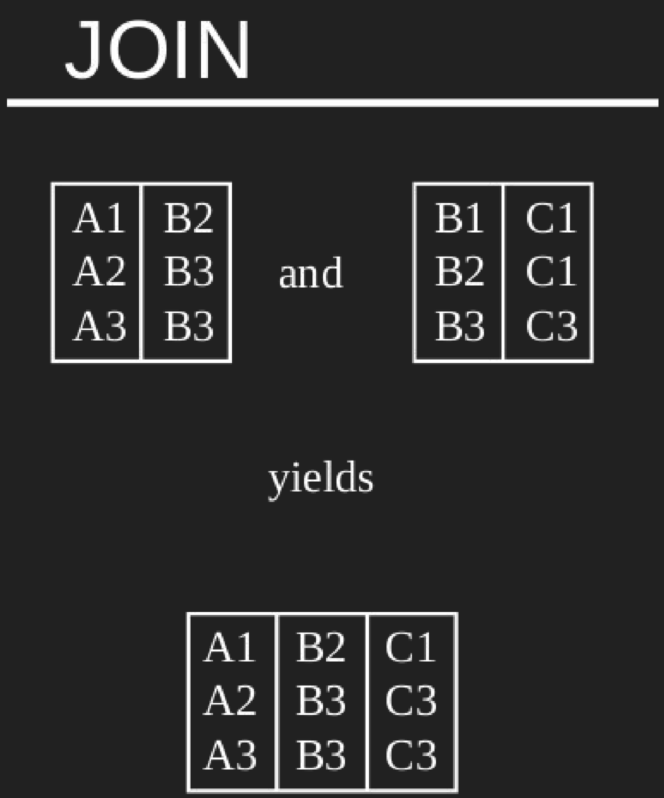
yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

Relational Algebra: Operations

JOIN operations (from Silberschatz, with Date's terminology)

- Output Relation Schema Convention:
  - We do not repeat those attributes that appear in the schemas of both relations.
  - The ordering of attributes is as follows:
    - First the attributes common to the schemas of both relations
    - Second those attributes unique to the schema of the first relation
    - Finally, those attributes unique to the schema of the second relation.



JOIN

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

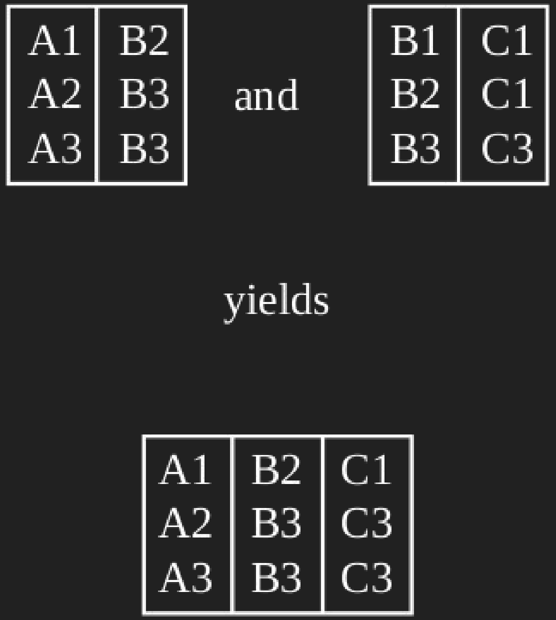
Note that in this convention, the order of the attributes matters. Therefore, we are using the term "schema" instead of "heading." The difference is headings are based on ***sets*** whereas schemas are based on ***sequences***.

Relational Algebra: Operations

The JOIN operations (from Beaulieu)

- Queries against a single relation are certainly not rare, but you fill find that most of your queries will require two, three, or even more relations.
- JOIN operations use a set of attributes as the *bridge* between relations, thereby allowing columns from both relations to be included in the result relation.

JOIN



A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

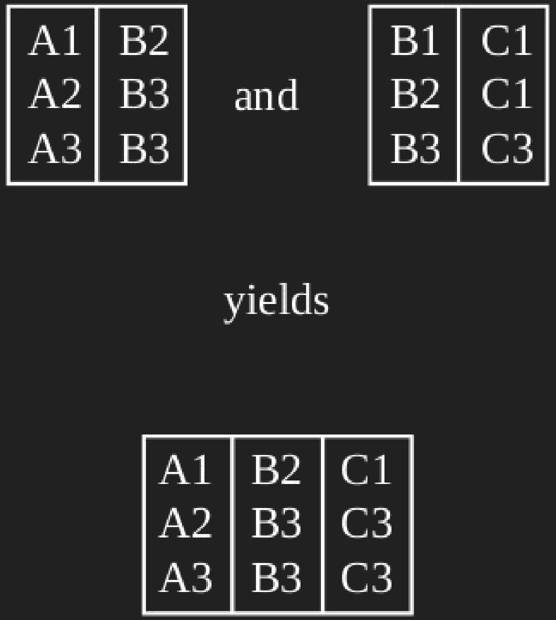
A1	B2	C1
A2	B3	C3
A3	B3	C3

Relational Algebra: Operations

The JOIN operations (from Date)

- **Joinability:**
  - Relations *r1* and *r2* are joinable if and only if attributes with the same heading are of the same type (meaning they are in fact the very same attribute).
  - Equivalently, relations *r1* and *r2* are joinable if and only if the set theory union of the headings of *r1* and *r2* is itself a legal heading.

JOIN



A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

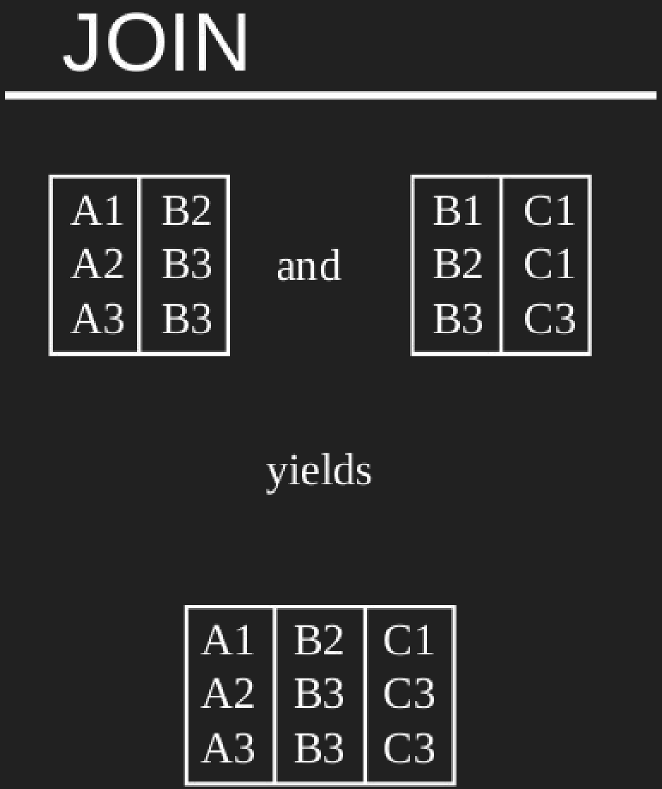
yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

Relational Algebra: Operations

The JOIN operations (from Date)

- **Joinability:**
  - Relations *r1* and *r2* are joinable if and only if attributes with the same heading are of the same type (meaning they are in fact the very same attribute).
  - Equivalently, relations *r1* and *r2* are joinable if and only if the set theory union of the headings of *r1* and *r2* is itself a legal heading.



JOIN

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

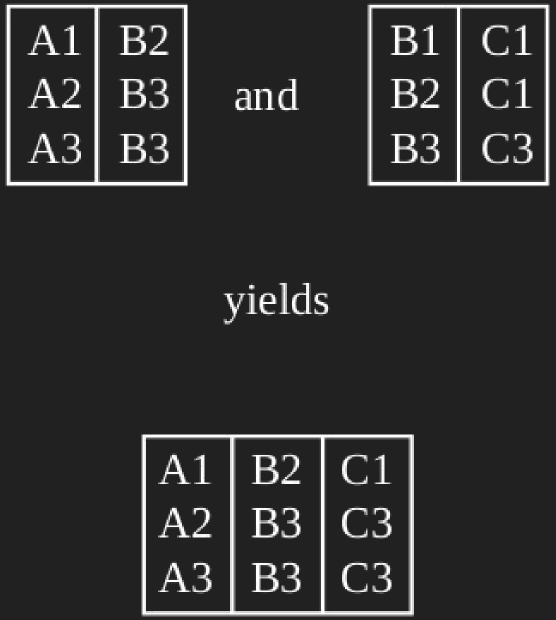
Recall: The heading is a *set* of attributes where *no two attributes have the same attribute name*.

Relational Algebra: Operations

JOIN

The JOIN operations (from Date)

- **Joinability:**
    - Relations *r1* and *r2* are joinable if and only if attributes with the same heading are of the same type (meaning they are in fact the very same attribute).
    - Equivalently, relations *r1* and *r2* are joinable if and only if the set theory union of the headings of *r1* and *r2* is itself a legal heading.



A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

Recall: The heading is a *set* of attributes where *no two attributes have the same attribute name*.

Why do the types of those attributes whose names are common between the relations determine whether the union of the heading is a valid heading?

Relational Algebra: Operations

JOIN

NATURAL JOIN (from Silbershatz)

- Usually, a query that involves a Cartesian product includes a selection operation on the result of the Cartesian product.
- The *natural join* is a binary operation that allows us to *combine certain selections and a Cartesian product into one operation*.

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

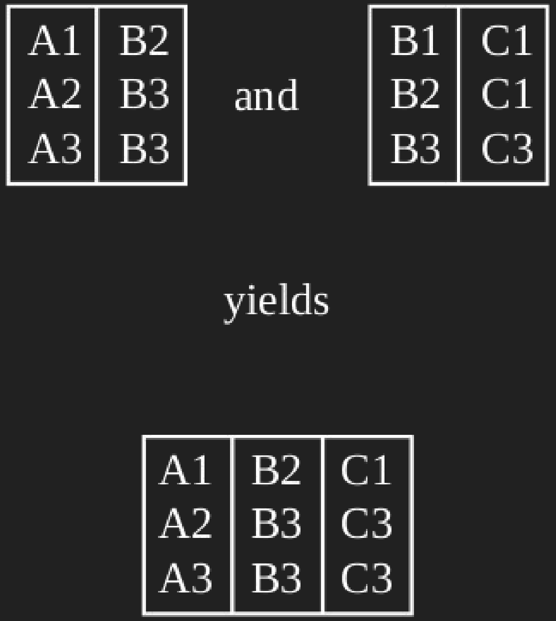
A1	B2	C1
A2	B3	C3
A3	B3	C3

Relational Algebra: Operations

NATURAL JOIN (from Silbershatz)

- Denoted by the join symbol ⋈
- The natural-join operation
  1. Forms a Cartesian product of its two arguments
  2. Performs a selection forcing equality on those attributes that appear in both relation schemas

JOIN



A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

Relational Algebra: Operations

NATURAL JOIN (from Silbershatz)

- Denoted by the join symbol ⋈
- The natural-join operation
    1. Forms a Cartesian product of its two arguments
    2. Performs a selection forcing equality on those attributes that appear in both relation schemas

JOIN

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

Relational Algebra: Operations

JOIN

NATURAL JOIN (from Silbershatz)

- Denoted by the join symbol $\bowtie$
- The natural-join operation
  1. Forms a Cartesian product of its two arguments
  2. Performs a selection forcing equality on those attributes that appear in both relation schemas

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

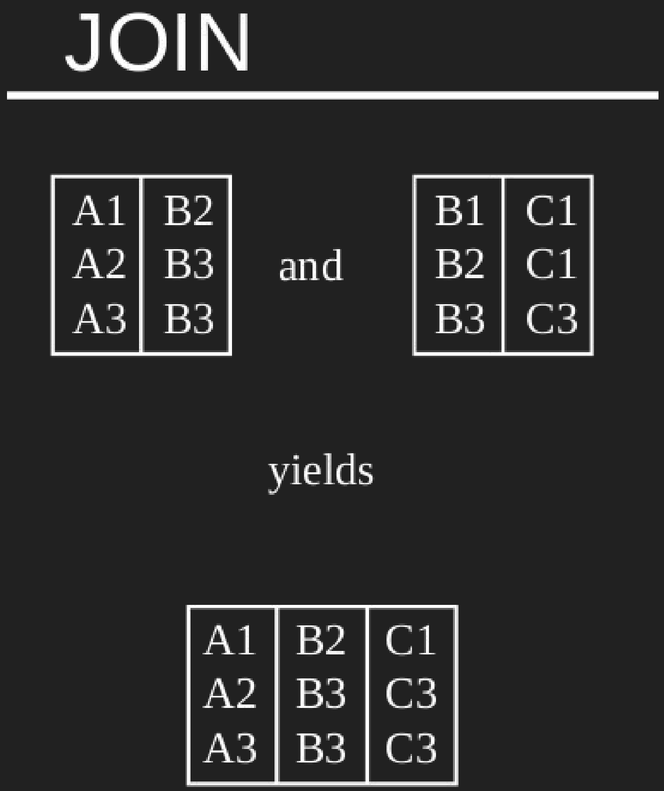
Remember this query expression???

$$\pi_{\text{account\_id}}(\sigma_{\text{cust\_id = ct\_cust\_id}}(\sigma_{\text{avail\_balance > 1000}}(\text{CUSTOMER\_ACCOUNT\_BALANCES}) \times \rho_{\text{CTYPES(ct\_cust\_id)}}(\pi_{\text{cust\_id}}(\sigma_{\text{cust\_type\_cd = “I”}}(\text{CUSTOMER\_TYPES})))))$$

Relational Algebra: Operations

NATURAL JOIN (from Silbershatz)

- Denoted by the join symbol $\bowtie$
- The natural-join operation
    1. Forms a Cartesian product of its two arguments
    2. Performs a selection forcing ***equality*** on those attributes that appear in both relation schemas



JOIN

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

Remember this query expression???

$$\pi_{account\_id}(\sigma_{cust\_id\ =\ ct\_cust\_id}(\sigma_{avail\_balance\ >\ 1000}(CUSTOMER\_ACCOUNT\_BALANCES) \times \rho_{CTYPES(ct\_cust\_id)}(\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES)))))$$

Relational Algebra: Operations

NATURAL JOIN (from Date)

- Definition:
  - Let *r1* and *r2* be *joinable*.
  - Then their *natural join* (or just *join* for short)

$$r1 \bowtie r2$$

is a relation with

a. heading the set theory union of the headings of *r1* and *r2*

b. body the set of all tuples *t* such that *t* is the set theory union of a tuple from *r1* and a tuple from *r2*.

JOIN

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

Relational Algebra: Operations

NATURAL JOIN (from Date)

- Easily the most important join operation.
  - So much so that the unqualified term “join” is taken almost invariably to mean the natural join specifically.

JOIN

A1	B2
A2	B3
A3	B3

and

B1	C1
B2	C1
B3	C3

yields

A1	B2	C1
A2	B3	C3
A3	B3	C3

Relational Algebra: Operations

NATURAL JOIN Example

$$\pi_{account\_id}(\sigma_{cust\_id\ =\ ct\_cust\_id}(\sigma_{avail\_balance\ >\ 1000}(CUSTOMER\_ACCOUNT\_BALANCES) \times \rho_{CTYPES(ct\_cust\_id)}(\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES)))))$$

Relational Algebra: Operations

NATURAL JOIN Example

$$\pi_{account\_id}(\sigma_{cust\_id\ =\ ct\_cust\_id}(\sigma_{avail\_balance\ >\ 1000}(CUSTOMER\_ACCOUNT\_BALANCES) \times \rho_{CTYPES(ct\_cust\_id)}(\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES)))))$$

BECOMES

Relational Algebra: Operations

NATURAL JOIN Example

$$\pi_{account\_id}(\sigma_{cust\_id = ct\_cust\_id}(\sigma_{avail\_balance > 1000}(CUSTOMER\_ACCOUNT\_BALANCES) \times \rho_{CTYPES(ct\_cust\_id)}(\pi_{cust\_id}(\sigma_{cust\_type\_cd = \text{“I”}}(CUSTOMER\_TYPES)))))$$

BECOMES

$$\pi_{account\_id}(\sigma_{(avail\_balance > 1000) \text{ AND } (cust\_type\_cd = \text{“I”})}(CUSTOMER\_ACCOUNT\_BALANCES \bowtie CUSTOMER\_TYPES))$$

Relational Algebra: Operations

NATURAL JOIN Example

$$\pi_{\text{account\_id}}(\sigma_{(\text{avail\_balance} > 1000) \text{ AND } (\text{cust\_type\_cd} = \text{“I”})}(\text{CUSTOMER\_ACCOUNT\_BALANCES} \bowtie \text{CUSTOMER\_TYPES}))$$

Relational Algebra: Operations

NATURAL JOIN

$\text{CUSTOMER\_ACCOUNT\_BALANCES} \bowtie \text{CUSTOMER\_TYPES}$

CUSTOMER_ACCOUNT_BALANCES

cust_id	account_id	avail_balance
1	1	1057.75
1	2	500.00
1	3	3000.00
2	4	2258.02
2	5	200.00
3	7	1057.75
3	8	2212.50
4	10	534.12
4	11	767.77
4	12	5487.09
5	13	2237.97
6	14	122.37
6	15	10000.00
7	17	5000.00
8	18	3487.19
8	19	387.99
9	21	125.67
9	22	9345.55
9	23	1500.00
10	24	23575.12
10	25	0.00
11	27	9345.55
12	28	38552.05
13	29	50000.00

CUSTOMER_TYPES

cust_id	cust_type_cd
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	I
10	B
11	B
12	B

Relational Algebra: Operations

NATURAL JOIN

CUSTOMER_ACCOUNT_BALANCES $\bowtie$ CUSTOMER_TYPES

Note that we no longer need to rename attributes, as we did when using taking the Cartesian Product.

CUSTOMER_ACCOUNT_BALANCES

cust_id	account_id	avail_balance
1	1	1057.75
1	2	500.00
1	3	3000.00
2	4	2258.02
2	5	200.00
3	7	1057.75
3	8	2212.50
4	10	534.12
4	11	767.77
4	12	5487.09
5	13	2237.97
6	14	122.37
6	15	10000.00
7	17	5000.00
8	18	3487.19
8	19	387.99
9	21	125.67
9	22	9345.55
9	23	1500.00
10	24	23575.12
10	25	0.00
11	27	9345.55
12	28	38552.05
13	29	50000.00

CUSTOMER_TYPES

cust_id	cust_type_cd
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	I
10	B
11	B
12	B

Relational Algebra: Operations

NATURAL JOIN

CUSTOMER_ACCOUNT_BALANCES $\bowtie$ CUSTOMER_TYPES

cust_id	account_id	avail_balance	cust_type_cd
1	1	1057.75	I
1	2	500.00	I
1	3	3000.00	I
2	4	2258.02	I
2	5	200.00	I
3	7	1057.75	I
3	8	2212.50	I
4	10	534.12	I
4	11	767.77	I
4	12	5487.09	I
5	13	2237.97	I
6	14	122.37	I
6	15	10000.00	I
7	17	5000.00	I
8	18	3487.19	I
8	19	387.99	I
9	21	125.67	I
9	22	9345.55	I
9	23	1500.00	I
10	24	23575.12	B
10	25	0.00	B
11	27	9345.55	B
12	28	38552.05	B

Relational Algebra: Operations

NATURAL JOIN

cust_id	account_id	avail_balance	cust_type_cd
1	1	1057.75	I
1	2	500.00	I
1	3	3000.00	I
2	4	2258.02	I
2	5	200.00	I
3	7	1057.75	I
3	8	2212.50	I
4	10	534.12	I
4	11	767.77	I
4	12	5487.09	I
5	13	2237.97	I
6	14	122.37	I
6	15	10000.00	I
7	17	5000.00	I
8	18	3487.19	I
8	19	387.99	I
9	21	125.67	I
9	22	9345.55	I
9	23	1500.00	I
10	24	23575.12	B
10	25	0.00	B
11	27	9345.55	B
12	28	38552.05	B

$\sigma_{\text{(avail\_balance > 1000) AND (cust\_type\_cd = “I”)}}$ (CUSTOMER_ACCOUNT_BALANCES $\bowtie$ CUSTOMER_TYPES)

Relational Algebra: Operations

NATURAL JOIN

cust_id	account_id	avail_balance	cust_type_cd
1	1	1057.75	I
1	2	500.00	I
1	3	3000.00	I
2	4	2258.02	I
2	5	200.00	I
3	7	1057.75	I
3	8	2212.50	I
4	10	534.12	I
4	11	767.77	I
4	12	5487.09	I
5	13	2237.97	I
6	14	122.37	I
6	15	10000.00	I
7	17	5000.00	I
8	18	3487.19	I
8	19	387.99	I
9	21	125.67	I
9	22	9345.55	I
9	23	1500.00	I
10	24	23575.12	B
10	25	0.00	B
11	27	9345.55	B
12	28	38552.05	B

$\sigma_{(avail\_balance > 1000) \text{ AND } (cust\_type\_cd = \text{“I”})}$ (CUSTOMER_ACCOUNT_BALANCES $\bowtie$ CUSTOMER_TYPES)

Relational Algebra: Operations

NATURAL JOIN

cust_id	account_id	avail_balance	cust_type_cd
1	1	1057.75	I
1	3	3000.00	I
2	4	2258.02	I
3	7	1057.75	I
3	8	2212.50	I
4	12	5487.09	I
5	13	2237.97	I
6	15	10000.00	I
7	17	5000.00	I
8	18	3487.19	I
9	22	9345.55	I
9	23	1500.00	I

$$\sigma_{(\text{avail\_balance} > 1000) \text{ AND } (\text{cust\_type\_cd} = \text{“I”})} (\text{CUSTOMER\_ACCOUNT\_BALANCES} \bowtie \text{CUSTOMER\_TYPES})$$

Relational Algebra: Operations

NATURAL JOIN

cust_id	account_id	avail_balance	cust_type_cd
1	1	1057.75	I
1	3	3000.00	I
2	4	2258.02	I
3	7	1057.75	I
3	8	2212.50	I
4	12	5487.09	I
5	13	2237.97	I
6	15	10000.00	I
7	17	5000.00	I
8	18	3487.19	I
9	22	9345.55	I
9	23	1500.00	I

$\pi_{\text{account\_id}}(\sigma_{(\text{avail\_balance} > 1000) \text{ AND } (\text{cust\_type\_cd} = \text{“I”})}(\text{CUSTOMER\_ACCOUNT\_BALANCES} \bowtie \text{CUSTOMER\_TYPES}))$

Relational Algebra: Operations

NATURAL JOIN

cust_id	account_id	avail_balance	cust_type_cd
1	1	1057.75	I
1	3	3000.00	I
2	4	2258.02	I
3	7	1057.75	I
3	8	2212.50	I
4	12	5487.09	I
5	13	2237.97	I
6	15	10000.00	I
7	17	5000.00	I
8	18	3487.19	I
9	22	9345.55	I
9	23	1500.00	I

$$\pi_{\text{account\_id}}(\sigma_{(\text{avail\_balance} > 1000) \text{ AND } (\text{cust\_type\_cd} = \text{“I”})}(\text{CUSTOMER\_ACCOUNT\_BALANCES} \bowtie \text{CUSTOMER\_TYPES}))$$

Relational Algebra: Operations

NATURAL JOIN

account_id
1
3
4
7
8
12
13
15
17
18
22
23

$\pi_{\text{account\_id}}(\sigma_{(\text{avail\_balance} > 1000) \text{ AND } (\text{cust\_type\_cd} = \text{“I”})}(\text{CUSTOMER\_ACCOUNT\_BALANCES} \bowtie \text{CUSTOMER\_TYPES}))$

Relational Algebra: Operations

NATURAL JOIN

$\pi_{account\_id}(\sigma_{(avail\_balance > 1000) \text{ AND } (cust\_type\_cd = “I”)} \text{ CUSTOMER\_ACCOUNT\_BALANCES} \bowtie \text{CUSTOMER\_TYPES}))$

account_id
1
3
4
7
8
12
13
15
17
18
22
23

$\pi_{account\_id}(\sigma_{cust\_id = ct\_cust\_id}(\sigma_{avail\_balance > 1000}(\text{CUSTOMER\_ACCOUNT\_BALANCES}) \times \rho_{CTYPES(ct\_cust\_id)}(\pi_{cust\_id}(\sigma_{cust\_type\_cd = “I”}(\text{CUSTOMER\_TYPES})))))$

account_id
1
3
4
7
8
12
13
15
17
18
22
23

Relational Algebra: Operations

NATURAL JOIN

<u>Requested Predicate:</u>

Employee with
first name **fname** and
last name **lname** works in
the department named **dname**.

DEPARTMENT	
dept_id	**name**
1	Operations
2	Loans
3	Administration

EMPLOYEE		
fname	**lname**	**dept_id**
Michael	Smith	3
Susan	Barker	3
Robert	Tyler	3
Susan	Hawthorne	1
John	Gooding	2
Helen	Fleming	1
Chris	Tucker	1
Sarah	Parker	1
Jane	Grossman	1
Paula	Roberts	1
Thomas	Ziegler	1

Relational Algebra: Operations

NATURAL JOIN

Requested Predicate:

Employee with
first name **fname** and
last name **lname** works in
the department named **dname**.

We take the NATURAL JOIN of the two relations:

$$\text{EMPLOYEE} \bowtie \text{DEPARTMENT}$$

DEPARTMENT

dept_id	name
1	Operations
2	Loans
3	Administration

EMPLOYEE

fname	lname	dept_id
Michael	Smith	3
Susan	Barker	3
Robert	Tyler	3
Susan	Hawthorne	1
John	Gooding	2
Helen	Fleming	1
Chris	Tucker	1
Sarah	Parker	1
Jane	Grossman	1
Paula	Roberts	1
Thomas	Ziegler	1

Relational Algebra: Operations

NATURAL JOIN

Requested Predicate:

Employee with
first name **fname** and
last name **lname** works in
the department named **dname**.

We take the NATURAL JOIN of the two relations:

EMPLOYEE ⋈ DEPARTMENT

fname	lname	dept_id	name
Michael	Smith	3	Administration
Susan	Barker	3	Administration
Robert	Tyler	3	Administration
Susan	Hawthorne	1	Operations
John	Gooding	2	Loans
Helen	Fleming	1	Operations
Chris	Tucker	1	Operations
Sarah	Parker	1	Operations
Jane	Grossman	1	Operations
Paula	Roberts	1	Operations
Thomas	Ziegler	1	Operations

Relational Algebra: Operations

NATURAL JOIN

Requested Predicate:

Employee with
first name **fname** and
last name **lname** works in
the department named **dname.**

We don't need the dept_id attribute:

$\Pi_{\text{fname,lname,name}}$ (**EMPLOYEE ⋈ DEPARTMENT**)

fname	lname	dept_id	name
Michael	Smith	3	Administration
Susan	Barker	3	Administration
Robert	Tyler	3	Administration
Susan	Hawthorne	1	Operations
John	Gooding	2	Loans
Helen	Fleming	1	Operations
Chris	Tucker	1	Operations
Sarah	Parker	1	Operations
Jane	Grossman	1	Operations
Paula	Roberts	1	Operations
Thomas	Ziegler	1	Operations

Relational Algebra: Operations

NATURAL JOIN

Requested Predicate:

Employee with
first name **fname** and
last name **lname** works in
the department named **dname.**

We don’t need the dept_id attribute:

$\Pi_{fname,lname,name}$ **(EMPLOYEE ⋈ DEPARTMENT)**

fname	lname	dept_id	name
Michael	Smith	3	Administration
Susan	Barker	3	Administration
Robert	Tyler	3	Administration
Susan	Hawthorne	1	Operations
John	Gooding	2	Loans
Helen	Fleming	1	Operations
Chris	Tucker	1	Operations
Sarah	Parker	1	Operations
Jane	Grossman	1	Operations
Paula	Roberts	1	Operations
Thomas	Ziegler	1	Operations

Relational Algebra: Operations

NATURAL JOIN

Requested Predicate:

Employee with
first name **fname** and
last name **lname** works in
the department named **dname.**

We don't need the dept_id attribute:

$\Pi_{\text{fname,lname,name}}$ (**EMPLOYEE ⋈ DEPARTMENT**)

fname	lname	name
Michael	Smith	Administration
Susan	Barker	Administration
Robert	Tyler	Administration
Susan	Hawthorne	Operations
John	Gooding	Loans
Helen	Fleming	Operations
Chris	Tucker	Operations
Sarah	Parker	Operations
Jane	Grossman	Operations
Paula	Roberts	Operations
Thomas	Ziegler	Operations

Relational Algebra: Operations

NATURAL JOIN

Requested Predicate:

Employee with
first name **fname** and
last name **lname** works in
the department named **dname.**

Finally, we rename the attribute "name" to "dname":

fname	lname	name
Michael	Smith	Administration
Susan	Barker	Administration
Robert	Tyler	Administration
Susan	Hawthorne	Operations
John	Gooding	Loans
Helen	Fleming	Operations
Chris	Tucker	Operations
Sarah	Parker	Operations
Jane	Grossman	Operations
Paula	Roberts	Operations
Thomas	Ziegler	Operations

$$\rho_{\text{EMP\_DEPT(fname,lname,dname)}}(\Pi_{\text{fname,lname,name}}(\textbf{EMPLOYEE} \bowtie \textbf{DEPARTMENT}))$$

Relational Algebra: Operations

NATURAL JOIN

Requested Predicate:

Employee with
first name **fname** and
last name **lname** works in
the department named **dname.**

EMP_DEPT		
fname	**lname**	**dname**
Michael	Smith	Administration
Susan	Barker	Administration
Robert	Tyler	Administration
Susan	Hawthorne	Operations
John	Gooding	Loans
Helen	Fleming	Operations
Chris	Tucker	Operations
Sarah	Parker	Operations
Jane	Grossman	Operations
Paula	Roberts	Operations
Thomas	Ziegler	Operations

$$\rho_{\text{EMP\_DEPT(fname,lname,dname)}}(\Pi_{\text{fname,lname,name}}(\textbf{EMPLOYEE} \bowtie \textbf{DEPARTMENT}))$$

Relational Algebra: Operations

THETA JOIN (from Silbershatz)

- The theta join operation is a *variant of the natural join* operation that allows us to *combine a selection and a Cartesian product* into a single operation.

$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$$

Where θ is a predicate on the attributes in the set union of the schemas of *r* and *s*.

Relational Algebra: Operations

THETA JOIN (from Stack Overflow “Difference between a theta join, equijoin and natural join”)

- A ***theta join*** allows for arbitrary comparison relationships (such as ≥).
- An ***equijoin*** is a theta join using the equality operator.
- A ***natural join*** is an equijoin on attributes that have the same name in each relationship.

Relational Algebra: Operations

THETA JOIN (from Stack Overflow "Difference between a theta join, equijoin and natural join")

- **Natural Join** = the join is made on all columns with the same name; it removes duplicate columns from the result, as opposed to all other joins.
- **Theta Join** = this is the general join everybody uses because it allows you to specify the condition (the ON clause in SQL). You can join on pretty much any condition you like, for example on Products that have the first 2 letters similar, or that have a different price. In practice, this is rarely the case - in 95% of the cases you will join on an equality condition, which leads us to:
- **Equi Join** = the most common one used in practice. Theta Join using only the equality operator.
- **Non-equi Join** = when you join on a condition other than "=".

Relational Algebra: Operations

THETA JOIN (from Stack Overflow “Difference between a theta join, equijoin and natural join”)

- **Natural Join** = the join is made on all columns with the same name; it removes duplicate columns from the result, as opposed to all other joins.
- **Theta Join** = this is the general join everybody uses because it allows you to specify the condition (the ON clause in SQL). You can join on pretty much any condition you like, for example on Products that have the first 2 letters similar, or that have a different price. In practice, this is rarely the case - in 95% of the cases you will join on an equality condition, which leads us to:
- **Equi Join** = the most common one used in practice. Theta Join using only the equality operator.
- **Non-equi Join** = when you join on a condition other than "=".

Equi Join and **Non-equi Join** are subsets of the general theta join.

Natural Join is also a theta join but the condition (the theta) is implicit.

Relational Algebra: Operations

THETA JOIN

$$\pi_{account\_id}(\sigma_{cust\_id\ =\ ct\_cust\_id}(\sigma_{avail\_balance\ >\ 1000}(CUSTOMER\_ACCOUNT\_BALANCES) \times \rho_{CTYPES(ct\_cust\_id)}(\pi_{cust\_id}(\sigma_{cust\_type\_cd\ =\ \text{“I”}}(CUSTOMER\_TYPES)))))$$

BECOMES

$$\pi_{account\_id}(CUSTOMER\_ACCOUNT\_BALANCES \bowtie_{(avail\_balance\ >\ 1000)\ AND\ (cust\_type\_cd\ =\ \text{“I”})} CUSTOMER\_TYPES)$$

Relational Algebra: Operations

THETA JOIN

$$\pi_{\text{account\_id}}(\text{CUSTOMER\_ACCOUNT\_BALANCES} \bowtie_{(\text{avail\_balance} > 1000) \text{ AND } (\text{cust\_type\_cd} = \text{“I”})} \text{CUSTOMER\_TYPES})$$

Relational Algebra: Operations

THETA JOIN Example

Requested Predicate:

Employee ID **emp_id** belongs to an employee that started at the bank before their supervisor.

EMP		
emp_id	**start_date**	**sup_emp_id**
1	2001-06-22	*null*
2	2002-09-12	1
3	2000-02-09	1
4	2002-04-24	3
5	2003-11-14	4
6	2004-03-17	4
7	2004-09-15	6
8	2002-12-02	6
9	2002-05-03	6
10	2002-07-27	4
11	2000-10-23	10

Relational Algebra: Operations

THETA JOIN Example

Requested Predicate:

> Employee ID **emp_id** belongs to an employee that started at the bank before their supervisor.

First, let's RENAME the relation:

$$\rho_{sub}(EMP)$$

$$\rho_{sup}(EMP)$$

EMP		
emp_id	**start_date**	**sup_emp_id**
1	2001-06-22	*null*
2	2002-09-12	1
3	2000-02-09	1
4	2002-04-24	3
5	2003-11-14	4
6	2004-03-17	4
7	2004-09-15	6
8	2002-12-02	6
9	2002-05-03	6
10	2002-07-27	4
11	2000-10-23	10

Relational Algebra: Operations

THETA JOIN Example

Requested Predicate:

Employee ID **emp_id** belongs to an employee that started at the bank before their supervisor.

First, let's RENAME the relation:

ρ_{sub} (EMP)

ρ_{sup} (EMP)

SUB		
emp_id	start_date	sup_emp_id
1	2001-06-22	*null*
2	2002-09-12	1
3	2000-02-09	1
4	2002-04-24	3
5	2003-11-14	4
6	2004-03-17	4
7	2004-09-15	6
8	2002-12-02	6
9	2002-05-03	6
10	2002-07-27	4
11	2000-10-23	10

SUP		
emp_id	start_date	sup_emp_id
1	2001-06-22	*null*
2	2002-09-12	1
3	2000-02-09	1
4	2002-04-24	3
5	2003-11-14	4
6	2004-03-17	4
7	2004-09-15	6
8	2002-12-02	6
9	2002-05-03	6
10	2002-07-27	4
11	2000-10-23	10

Relational Algebra: Operations

THETA JOIN Example

Requested Predicate:

> Employee ID **emp_id** belongs to an employee that started at the bank before their supervisor.

Now we can do the theta join:

SUB		
emp_id	**start_date**	**sup_emp_id**
1	2001-06-22	*null*
2	2002-09-12	1
3	2000-02-09	1
4	2002-04-24	3
5	2003-11-14	4
6	2004-03-17	4
7	2004-09-15	6
8	2002-12-02	6
9	2002-05-03	6
10	2002-07-27	4
11	2000-10-23	10

SUP		
emp_id	**start_date**	**sup_emp_id**
1	2001-06-22	*null*
2	2002-09-12	1
3	2000-02-09	1
4	2002-04-24	3
5	2003-11-14	4
6	2004-03-17	4
7	2004-09-15	6
8	2002-12-02	6
9	2002-05-03	6
10	2002-07-27	4
11	2000-10-23	10

$$\rho_{\text{sub}}(\text{EMP}) \bowtie_{((\text{sub.sup\_emp\_id} = \text{sup.emp\_id}) \text{ AND } (\text{sub.start\_date} < \text{sup.start\_date}))} \rho_{\text{sup}}(\text{EMP})$$

Relational Algebra: Operations

THETA JOIN Example

Requested Predicate:

Employee ID **emp_id** belongs to an employee that started at the bank before their supervisor.

sub.emp_id	sub.start_date	sub.superior_emp_id	sup.emp_id	sup.start_date	sup.superior_emp_id
3	2000-02-09	1	1	2001-06-22	*null*
8	2002-12-02	6	6	2004-03-17	4
9	2002-05-03	6	6	2004-03-17	4
11	2000-10-23	10	10	2002-07-27	4
13	2000-05-11	4	4	2002-04-24	3
16	2001-03-15	4	4	2002-04-24	3

Now we can do the theta join:

$$\rho_{\text{sub}}(\text{EMP}) \bowtie_{((\text{sub.sup\_emp\_id} = \text{sup.emp\_id}) \text{ AND } (\text{sub.start\_date} < \text{sup.start\_date}))} \rho_{\text{sup}}(\text{EMP})$$

Relational Algebra: Operations

THETA JOIN Example

Requested Predicate:

Employee ID **emp_id** belongs to an employee that started at the bank before their supervisor.

sub.emp_id	**sub.start_date**	**sub.superior_emp_id**	**sup.emp_id**	**sup.start_date**	**sup.superior_emp_id**
3	2000-02-09	1	1	2001-06-22	*null*
8	2002-12-02	6	6	2004-03-17	4
9	2002-05-03	6	6	2004-03-17	4
11	2000-10-23	10	10	2002-07-27	4
13	2000-05-11	4	4	2002-04-24	3
16	2001-03-15	4	4	2002-04-24	3

We can now PROJECT on sub.emp_id:

$$\pi_{\text{sub.emp\_id}} \left(\rho_{\text{sub}} (\text{EMP}) \bowtie_{((\text{sub.sup\_emp\_id} = \text{sup.emp\_id}) \text{ AND } (\text{sub.start\_date} < \text{sup.start\_date}))} \rho_{\text{sup}} (\text{EMP})\right)$$

Relational Algebra: Operations

THETA JOIN Example

Requested Predicate:

> Employee ID **emp_id** belongs to an employee that started at the bank before their supervisor.

sub.emp_id
3
8
9
11
13
16

We can now PROJECT on sub.emp_id:

$$\pi_{\text{sub.emp\_id}} (\rho_{\text{sub}} (\text{EMP}) \bowtie_{((\text{sub.sup\_emp\_id} = \text{sup.emp\_id}) \text{ AND } (\text{sub.start\_date} < \text{sup.start\_date}))} \rho_{\text{sup}} (\text{EMP}))$$

Relational Algebra: Operations

THETA JOIN Example

Requested Predicate:

> Employee ID **emp_id** belongs to an employee that started at the bank before their supervisor.

sub.emp_id
3
8
9
11
13
16

And we finish up with a RENAME:

$$\rho_{\text{EMP\_BEFORE\_SUP(emp\_id)}}(\pi_{\text{sub.emp\_id}}\,(\rho_{\text{sub}}(\text{EMP}) \bowtie_{((\text{sub.sup\_emp\_id} = \text{sup.emp\_id})\ \text{AND}\ (\text{sub.start\_date} < \text{sup.start\_date}))} \rho_{\text{sup}}(\text{EMP})))$$

Relational Algebra: Operations

THETA JOIN Example

Requested Predicate:

> Employee ID **emp_id** belongs to an employee that started at the bank before their supervisor.

EMP_BEFORE_SUP
emp_id
3
8
9
11
13
16

And we finish up with a RENAME:

$$\rho_{\text{EMP\_BEFORE\_SUP(emp\_id)}}(\Pi_{\text{sub.emp\_id}}(\rho_{\text{sub}}(\text{EMP}) \bowtie_{((\text{sub.sup\_emp\_id} = \text{sup.emp\_id}) \text{ AND } (\text{sub.start\_date} < \text{sup.start\_date}))} \rho_{\text{sup}}(\text{EMP})))$$

Relational Algebra: Operations

The **OUTER JOIN** operations (from Silbershatz)

- The outer-join operation is an extension of the join operation to deal with *missing information*.
- Some tuples in either or both of the relations being joined could be “lost” *when the join condition is not met*.
- The outer join operation works in a manner similar to the natural join operation, but preserves those tuples that would be lost in a join by *creating tuples in the result containing **null** values*.

Relational Algebra: Operations

NULLS (from Silbershatz)

- A null value indicates that the value does not exist (or is not known)
- An unknown value may be either
    - Missing (the value exists, but we do not have it)
    - Not known (we do not know whether or not the value actually exists)
- Null values are difficult to handle, and it is preferable not to resort to them.

(from Date)

- Nulls--and the entire theory of **three-valued logic** on which they are based--are fundamentally misguided and have no place in a clean formal system such as the relational model is intended to be.

Relational Algebra: Operations

NULLS (from Silbershatz)

- A null value indicates that the value does not exist (or is not known)
- An unknown value may be either
  - Missing (the value exists, but we do not have it)
  - Not known (we do not know whether or not the value actually exists)
- Null values are difficult to handle, and it is preferable not to resort to them.

(from Date)

We will cover three-valued logic, and how to deal with nulls, in the next lecture.

- Nulls--and the entire theory of **three-valued logic** on which they are based--are fundamentally misguided and have no place in a clean formal system such as the relational model is intended to be.

Relational Algebra: Operations

The **OUTER JOIN** operations (from Silbershatz)

- There are three forms of the outer-join operation:
  - Left outer join ⟕
  - Right outer join ⟖
  - Full outer join ⟗

Relational Algebra: Operations

The **OUTER JOIN** operations (from Silbershatz)

- There are three forms of the outer-join operation:
  - Left outer join ⟕
  - Right outer join ⟖
  - Full outer join ⟗

All three forms of outer join compute the join and add extra tuples to the result of the join.

Relational Algebra: Operations

⟕

Left outer join (from Silbershatz)

- Takes all tuples in the left relation that did not match with any tuple in the right position
- pads the tuples with null values for all other attributes from the right relation
- adds them to the result of the join.

All information from the left relation is present in the result of the left outer join.

Relational Algebra: Operations

⟖

Right outer join (from Silbershatz)

- Takes all tuples in the right relation that did not match with any tuple in the left position
- pads the tuples with null values for all other attributes from the left relation
- adds them to the result of the join.

All information from the right relation is present in the result of the left outer join.

Relational Algebra: Operations

$⟗$

Full outer join (from Silbershatz)

- Does both the left and right outer join operations
- pads tuples from the left relation that did not match any from the right relation.
- pads tuples from the right relation that did not match any from the left relation.
- adds the padded tuples to the result of the join.

All information from both relations is present in the result of the full outer join.

Relational Algebra: Operations

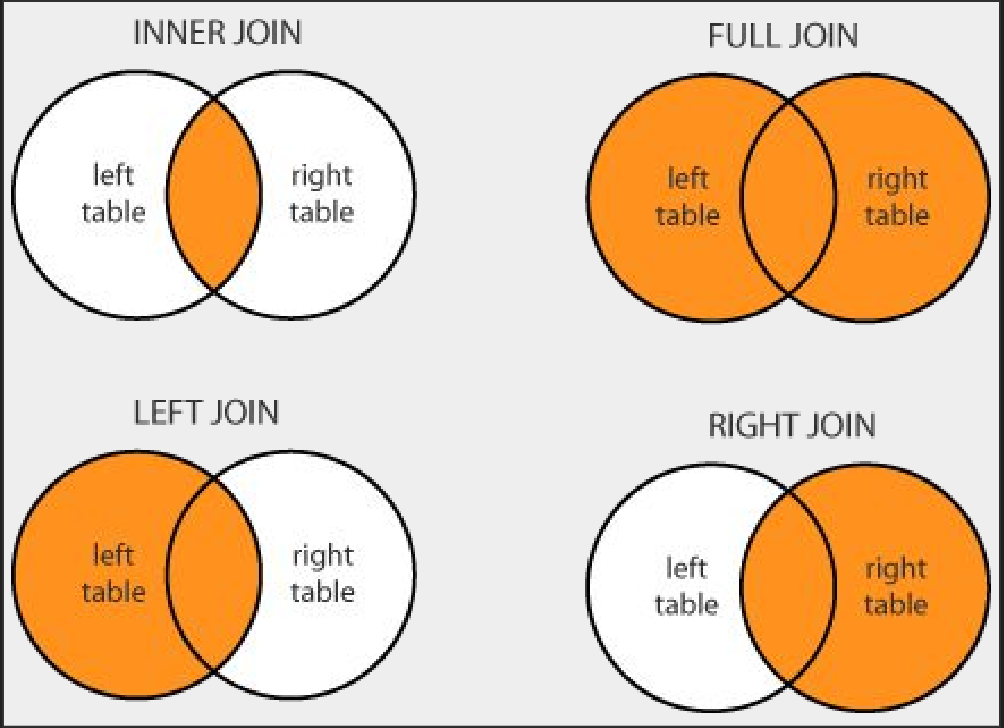


Image from http://www.dofactory.com/sql/join

Relational Algebra: Operations

⟕

Left outer join Example

Requested Predicate:

> Branch named **name** is in city **city** along with the business with customer ID **cust_id**

BRANCH_NAME_CITY	
name	**city**
Headquarters	Waltham
Woburn Branch	Woburn
Quincy Branch	Quincy
So. NH Branch	Salem

BUSINESS_CUST_CITY	
cust_id	**city**
10	Salem
11	Wilmington
12	Salem
13	Quincy

Relational Algebra: Operations

Left outer join Example

Requested Predicate:

> Branch named **name** is in city **city** along with the business with customer ID **cust_id**

BRANCH_NAME_CITY	
name	**city**
Headquarters	Waltham
Woburn Branch	Woburn
Quincy Branch	Quincy
So. NH Branch	Salem

BUSINESS_CUST_CITY	
cust_id	**city**
10	Salem
11	Wilmington
12	Salem
13	Quincy

BRANCH_NAME_CITY ⟕ BRANCH_CUST_CITY

Relational Algebra: Operations

Left outer join Example

Requested Predicate:

Branch named **name** is in city **city** along with the business with customer ID **cust_id**

name	city	cust_id
Headquarters	Waltham	*null*
Woburn Branch	Woburn	*null*
Quincy Branch	Quincy	13
So. NH Branch	Salem	10
So. NH Branch	Salem	12

BRANCH_NAME_CITY ⟕ BRANCH_CUST_CITY

Relational Algebra: Operations

Right outer join Example

Requested Predicate:

> Business with customer ID **cust_id** is in city **city** along with the branch named **name**

BRANCH_NAME_CITY	
name	**city**
Headquarters	Waltham
Woburn Branch	Woburn
Quincy Branch	Quincy
So. NH Branch	Salem

BUSINESS_CUST_CITY	
cust_id	**city**
10	Salem
11	Wilmington
12	Salem
13	Quincy

Relational Algebra: Operations

Right outer join Example

Requested Predicate:

> Business with customer ID **cust_id** is in city **city** along with the branch named **name**

BRANCH_NAME_CITY	
name	**city**
Headquarters	Waltham
Woburn Branch	Woburn
Quincy Branch	Quincy
So. NH Branch	Salem

BUSINESS_CUST_CITY	
cust_id	**city**
10	Salem
11	Wilmington
12	Salem
13	Quincy

BRANCH_NAME_CITY ⟖ BRANCH_CUST_CITY

Relational Algebra: Operations

Right outer join Example

Requested Predicate:

Business with customer ID **cust_id** is in city **city** along with the branch named **name**

name	cust_id	city
So. NH Branch	10	Salem
null	11	Wilmington
So. NH Branch	12	Salem
Quincy Branch	13	Quincy

BRANCH_NAME_CITY ⟖ BRANCH_CUST_CITY

Relational Algebra: Operations

Full outer join Example

Requested Predicate:

City **city** is home to the branch named **name** and to the business with customer ID **cust_id**

BRANCH_NAME_CITY	
name	**city**
Headquarters	Waltham
Woburn Branch	Woburn
Quincy Branch	Quincy
So. NH Branch	Salem

BUSINESS_CUST_CITY	
cust_id	**city**
10	Salem
11	Wilmington
12	Salem
13	Quincy

Relational Algebra: Operations

Full outer join Example

Requested Predicate:

City **city** is home to the branch named **name** and to the business with customer ID **cust_id**

BRANCH_NAME_CITY	
name	**city**
Headquarters	Waltham
Woburn Branch	Woburn
Quincy Branch	Quincy
So. NH Branch	Salem

BUSINESS_CUST_CITY	
cust_id	**city**
10	Salem
11	Wilmington
12	Salem
13	Quincy

BRANCH_NAME_CITY ⟗ BRANCH_CUST_CITY

Relational Algebra: Operations

Full outer join Example

Requested Predicate:

> City **city** is home to the branch named **name** and to the business with customer ID **cust_id**

name	**city**	**cust_id**
Headquarters	Waltham	*null*
Woburn Branch	Woburn	*null*
Quincy Branch	Quincy	13
So. NH Branch	Salem	10
So. NH Branch	Salem	12
null	Wilmington	11

BRANCH_NAME_CITY ⟗ BRANCH_CUST_CITY

Relational Algebra: Operations

Full outer join Example

Requested Predicate:

City **city** is home to the branch named **name** and to the business with customer ID **cust_id**

name	city	cust_id
Headquarters	Waltham	*null*
Woburn Branch	Woburn	*null*
Quincy Branch	Quincy	13
So. NH Branch	Salem	10
So. NH Branch	Salem	12
null	Wilmington	11

BRANCH_NAME_CITY ⟗ BRANCH_CUST_CITY

Note: the above output relation attribute scheme is from Postgres. RelaX outputs two columns for city.

Relational Algebra: Division

$T = \pi_{format,movieID}$ Tape(id,format,movieId) $F = \pi_{format}$ Format(format,extra_Ch))

1	VHS	7
2	DVD	7
4	VHS	55
5	VHS	1
8	HQ	7
11	VHS	25
17	DVD	25

VHS	0.0
DVD	1.0
HQ	1.5

Result:

1	7
2	7
8	7

Find movies which are available in all formats

Relational Division

Informally T ./. F is the set of all tuples r of T projected on attributes not belonging to F such that $\{(r)\} \times F \subseteq T$

Relational Algebra: an operator based on table predicates

Relational Division T . /. F

- Simulates universal quantifier for finite sets
- In order to divide T by F, the attributes of F must be a subset of the attributes of T: $\Sigma(F) \subset \Sigma(T)$
- Signature of T ./. F is $D = \Sigma(T) \setminus \Sigma(F)$

$$T \;./.\; F = \{ t' \mid t' \in \pi_D(T) \;\wedge$$
$$(\forall s \in F)(\exists t \in T)\;\; \pi'_{\Sigma(F)}(t) = s \wedge \pi'_D(t) = t' \}$$

π' denotes the projection of a row as opposed to π, which is defined on tables.

$\pi_D(T) = \{(7), (55), (1), (25)\}$

F = {VHS,DVD, HQ}

let t' be (55), for s = (DVD) there is

no tuple (DVD, 55) in T. t' = (7) is the only one which qualifies

From http://www.inf.fu-berlin.de/lehre/SS05/19517-V/FolienEtc/dbs05-08-RAlgebra-6.pdf

Relational Algebra Division

T ./. F may be defined in terms of other relational operators

$$T \;./.\; F = \underbrace{\pi_D (T) \setminus (\pi_D \underbrace{(\pi_D (T) \times F) \setminus T}_{\text{The "missing" tuples of T}})}_{\text{Building the complement}}$$

$$\boxed{D = \Sigma(T) \setminus \Sigma(F)}$$

Proof: Assignment

Property of relational division:

Let $D = \Sigma(T) \setminus \Sigma(F)$,

if D contains the key of T and $|F| > 1$ then $T \;./.\; F = \varnothing$

From http://www.inf.fu-berlin.de/lehre/SS05/19517-V/FolienEtc/dbs05-08-RAlgebra-6.pdf

Relational Algebra: Operations

DIVISION Example

Requested Predicate:

> Postal code **postal_code** has customers of each customer type.

postal_code_cust_type	
postal_code	**cust_type_cd**
01940	I
01801	I
02169	I
02451	I
03079	I
01887	I
02458	I
03079	B
01887	B
02169	B

cust_type
cust_type_cd
I
B

Relational Algebra: Operations

DIVISION Example

Requested Predicate:

> Postal code **postal_code** has customers of each customer type.

postal_code_cust_type ÷ cust_type

postal_code_cust_type	
postal_code	cust_type_cd
01940	I
01801	I
02169	I
02451	I
03079	I
01887	I
02458	I
03079	B
01887	B
02169	B

cust_type
cust_type_cd
I
B

Relational Algebra: Operations

DIVISION Example

Requested Predicate:

> Postal code **postal_code** has customers of each customer type.

postal_code_cust_type ÷ cust_type

postal_code_cust_type	
postal_code	cust_type_cd
01940	I
01801	I
02169	I
02451	I
03079	I
01887	I
02458	I
03079	B
01887	B
02169	B

cust_type
cust_type_cd
I
B

Relational Algebra: Operations

DIVISION Example

Requested Predicate:

Postal code **postal_code** has customers of each customer type.

postal_code_cust_type ÷ cust_type

postal_code
02169
03079
01887

Relational Algebra: Operations

This completes the slides on the relational model and the algebra.

We will return to these concepts later in the semester, particularly when we discuss how data modeling techniques and the inner workings of relational database systems, such as the query optimizer.