

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 Silberschatz et al.

Relational Algebra

(Continued)

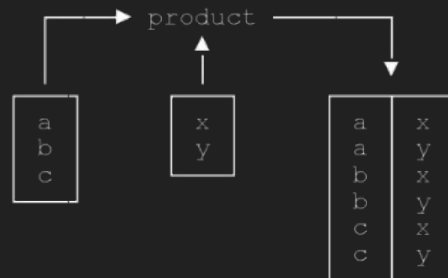
restrict



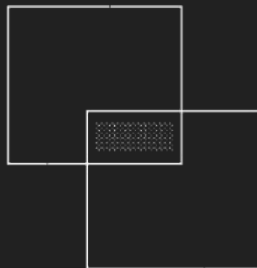
project



product



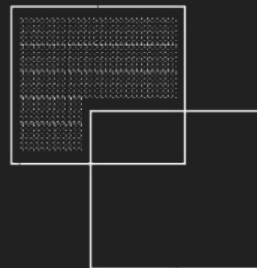
intersect



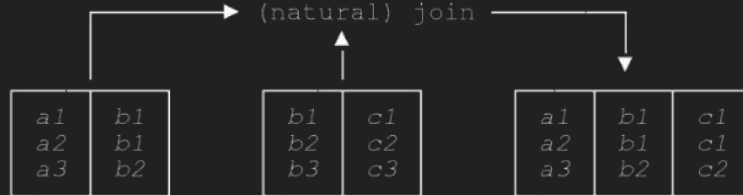
union



difference



(natural) join



Relational Algebra: Operations

RENAME:

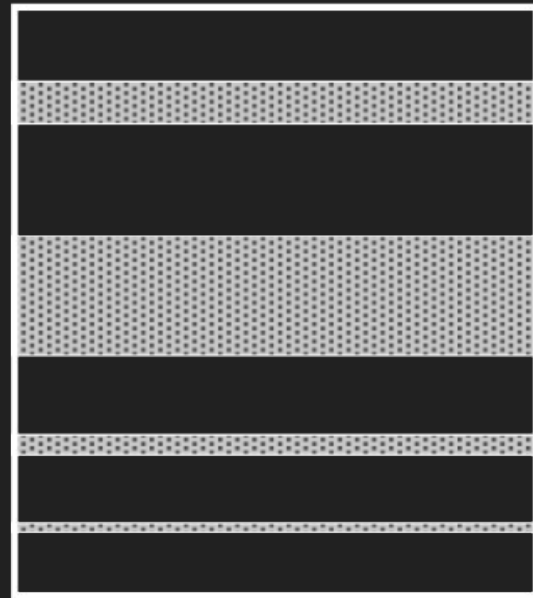
- Denoted by lowercase Greek letter rho (ρ)
- Unlike relations in the database, the results of relational-algebra expressions do not have a name that we can use to refer to them.
- It is useful to be able to give them names.

SELECT

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
-
- 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.

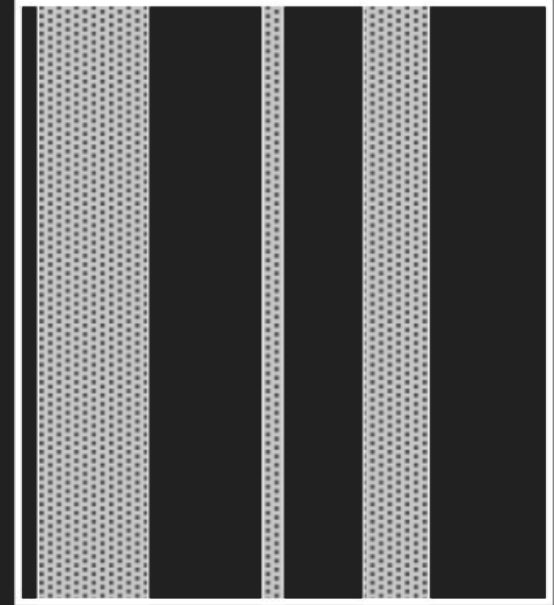


PROJECT

Relational Algebra: Operations

PROJECT:

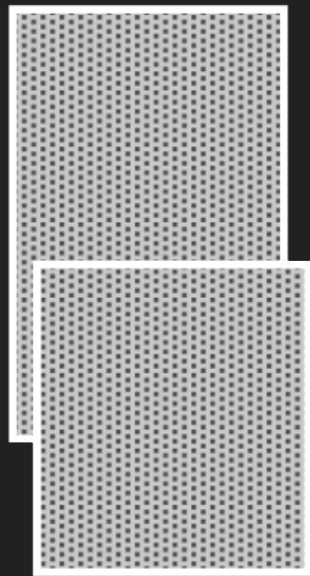
- Denoted by uppercase Greek letter pi (Π)
- Unary operation that returns its argument relation, with certain attributes left out.
- Since a relation (body) is a set, any duplicate rows are eliminated.
- We list those attributes that we wish to appear in the result as a subscript to Π .
- The argument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



Relational Algebra: Operations

UNION:

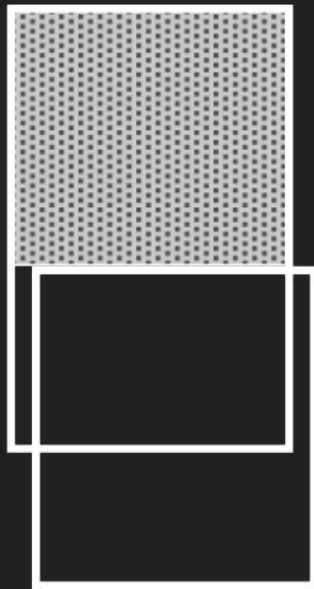
- Denoted, as in set theory, by \cup .
- Binary operation that returns the set theory union of the bodies of the two argument relations.
- We must make sure that unions are taken between *compatible* relations. (More on this in a later slide.)



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.

PRODUCT

X
Y
Z

and

A
B

yields

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*

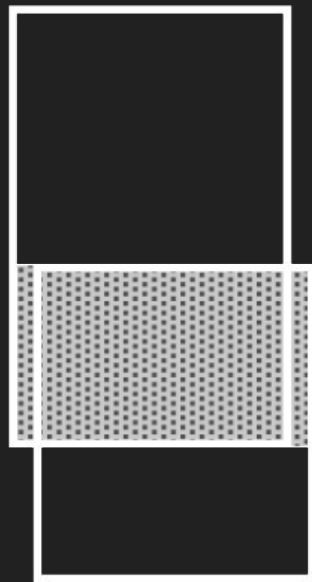
The next set of slides will cover these operations that do not add any power to the algebra, but which simplify common queries.

Relational Algebra: Operations

INTERSECTION

Set Intersection:

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

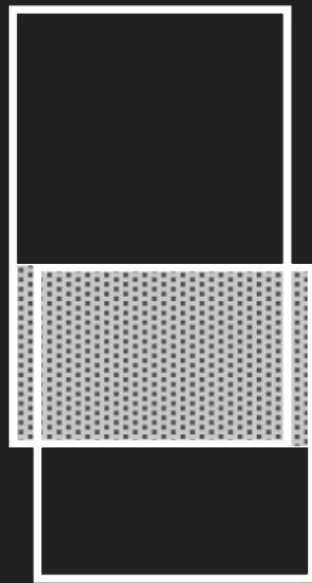


Relational Algebra: Operations

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.



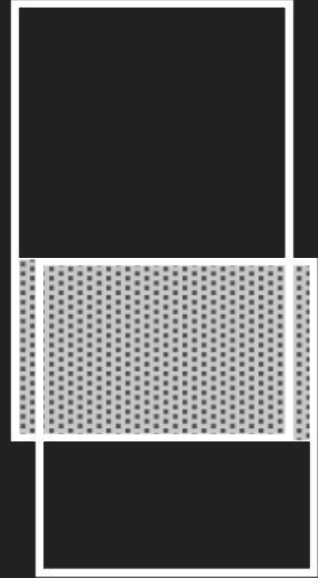
INTERSECTION

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

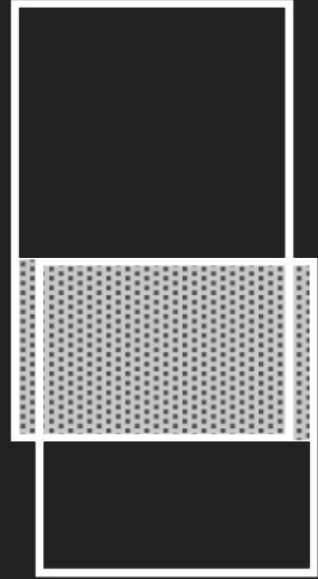
Relational Algebra: Operations

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: $r - s$



INTERSECTION

Relational Algebra: Operations

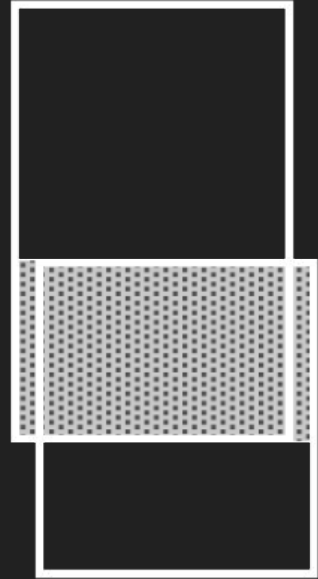
Set Intersection:

- 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: $r - s$

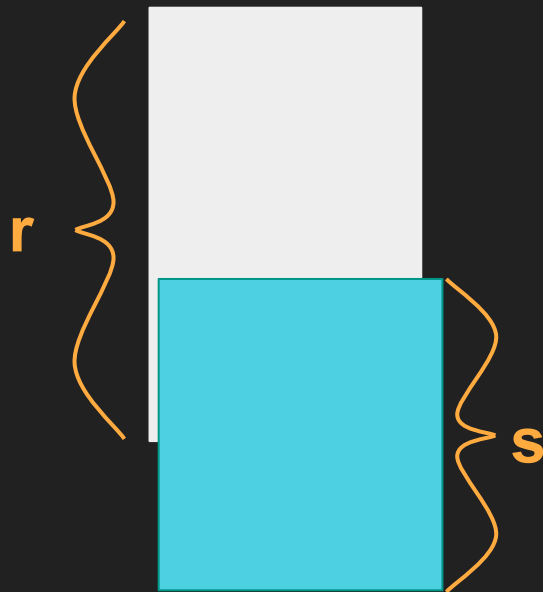
$$r \cap s = r - (r - s)$$



Relational Algebra: Operations

Set Intersection:

INTERSECTION

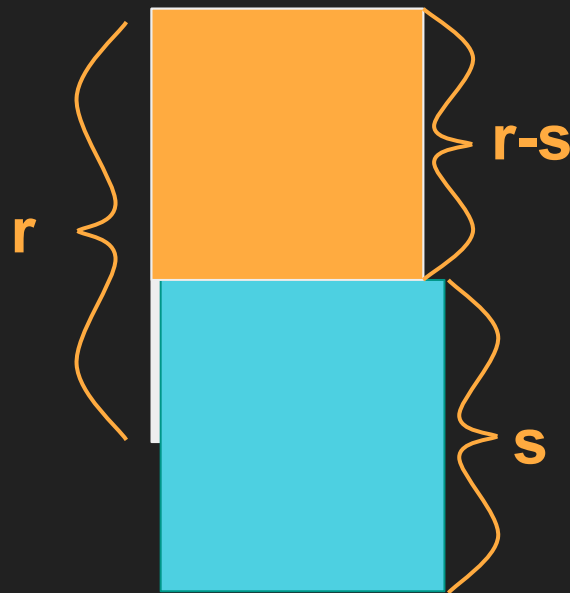


$$r \cap s = r - (r - s)$$

Relational Algebra: Operations

Set Intersection:

INTERSECTION

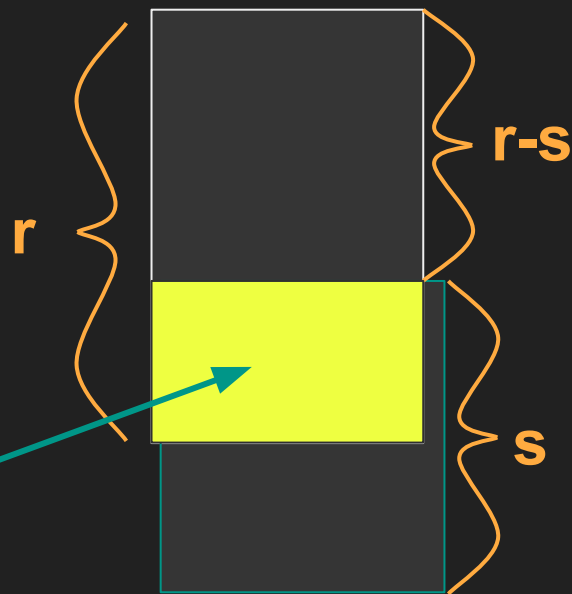


$$r \cap s = r - (r - s)$$

Relational Algebra: Operations

INTERSECTION

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

$\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))$

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
3	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

$\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))$

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
3	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

$\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))$

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
3	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

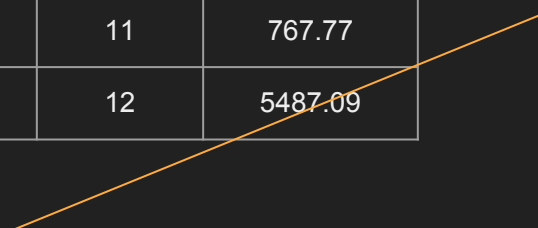
Required Predicate:

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

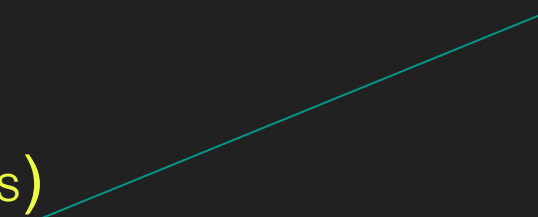
cust_id	account_id	avail_balance
1	1	1057.75
2	4	2258.02
3	7	1057.75
3	8	2212.5
4	12	5487.09

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

$\Pi_{\text{cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}))$

$\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))$



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
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
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

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

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

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

$$\Pi_{\text{cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES})) \cap \Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))$$

Relational Algebra: Operations

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

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

$$\Pi_{\text{cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES})) \cap \Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))$$

Relational Algebra: Operations

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

We can pick up from here.

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

$\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES})$

$\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))$

Relational Algebra: Operations

Another Example

Required Predicate:

account_id 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:

$\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES})$

$\rho_{\text{CTYPES(ct_cust_id)}}(\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES})))$

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

cust_id
1
2
3
4
5
6
7
8
9

Relational Algebra: Operations

Another Example

Required Predicate:

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

Now we can get the PRODUCT

$\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES})$ ✗

$\rho_{\text{CTYPES}}(\text{ct_cust_id})(\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES})))$

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
I	000001	2	I
I	000002	8	I
I	000003	9	E
I	000004	8	E
I	000005	82	E
I	000006	72	E
I	000007	82	E
I	000008	82	E
I	000009	82	E
I	000010	82	E
I	000011	82	E
I	000012	82	E
I	000013	82	E
I	000014	82	E
I	000015	82	E
I	000016	82	E
I	000017	82	E
I	000018	82	E
I	000019	82	E
I	000020	82	E
I	000021	82	E
I	000022	82	E
I	000023	82	E
I	000024	82	E
I	000025	82	E
I	000026	82	E
I	000027	82	E
I	000028	82	E
I	000029	82	E
I	000030	82	E
I	000031	82	E
I	000032	82	E
I	000033	82	E
I	000034	82	E
I	000035	82	E
I	000036	82	E
I	000037	82	E
I	000038	82	E
I	000039	82	E
I	000040	82	E
I	000041	82	E
I	000042	82	E
I	000043	82	E
I	000044	82	E
I	000045	82	E
I	000046	82	E
I	000047	82	E
I	000048	82	E
I	000049	82	E
I	000050	82	E
I	000051	82	E
I	000052	82	E
I	000053	82	E
I	000054	82	E
I	000055	82	E
I	000056	82	E
I	000057	82	E
I	000058	82	E
I	000059	82	E
I	000060	82	E
I	000061	82	E
I	000062	82	E
I	000063	82	E
I	000064	82	E
I	000065	82	E
I	000066	82	E
I	000067	82	E
I	000068	82	E
I	000069	82	E
I	000070	82	E
I	000071	82	E
I	000072	82	E
I	000073	82	E
I	000074	82	E
I	000075	82	E
I	000076	82	E
I	000077	82	E
I	000078	82	E
I	000079	82	E
I	000080	82	E
I	000081	82	E
I	000082	82	E
I	000083	82	E
I	000084	82	E
I	000085	82	E
I	000086	82	E
I	000087	82	E
I	000088	82	E
I	000089	82	E
I	000090	82	E
I	000091	82	E
I	000092	82	E
I	000093	82	E
I	000094	82	E
I	000095	82	E
I	000096	82	E
I	000097	82	E
I	000098	82	E
I	000099	82	E
I	000100	82	E
I	000101	82	E
I	000102	82	E
I	000103	82	E
I	000104	82	E
I	000105	82	E
I	000106	82	E
I	000107	82	E
I	000108	82	E
I	000109	82	E
I	000110	82	E
I	000111	82	E
I	000112	82	E
I	000113	82	E
I	000114	82	E
I	000115	82	E
I	000116	82	E
I	000117	82	E
I	000118	82	E
I	000119	82	E
I	000120	82	E
I	000121	82	E
I	000122	82	E
I	000123	82	E
I	000124	82	E
I	000125	82	E
I	000126	82	E
I	000127	82	E
I	000128	82	E
I	000129	82	E
I	000130	82	E
I	000131	82	E
I	000132	82	E
I	000133	82	E
I	000134	82	E
I	000135	82	E
I	000136	82	E
I	000137	82	E
I	000138	82	E
I	000139	82	E
I	000140	82	E
I	000141	82	E
I	000142	82	E
I	000143	82	E
I	000144	82	E
I	000145	82	E
I	000146	82	E
I	000147	82	E
I	000148	82	E
I	000149	82	E
I	000150	82	E
I	000151	82	E
I	000152	82	E
I	000153	82	E
I	000154	82	E
I	000155	82	E
I	000156	82	E
I	000157	82	E
I	000158	82	E
I	000159	82	E
I	000160	82	E
I	000161	82	E
I	000162	82	E
I	000163	82	E
I	000164	82	E
I	000165	82	E
I	000166	82	E
I	000167	82	E
I	000168	82	E
I	000169	82	E
I	000170	82	E
I	000171	82	E
I	000172	82	E
I	000173	82	E
I	000174	82	E
I	000175	82	E
I	000176	82	E
I	000177	82	E
I	000178	82	E
I	000179	82	E
I	000180	82	E
I	000181	82	E
I	000182	82	E
I	000183	82	E
I	000184	82	E
I	000185	82	E
I	000186	82	E
I	000187	82	E
I	000188	82	E
I	000189	82	E
I	000190	82	E
I	000191	82	E
I	000192	82	E
I	000193	82	E
I	000194	82	E
I	000195	82	E
I	000196	82	E
I	000197	82	E
I	000198	82	E
I	000199	82	E
I	000200	82	E
I	000201	82	E
I	000202	82	E
I	000203	82	E
I	000204	82	E
I	000205	82	E
I	000206	82	E
I	000207	82	E
I	000208	82	E
I	000209	82	E
I	000210	82	E
I	000211	82	E
I	000212	82	E
I	000213	82	E
I	000214	82	E
I	000215	82	E
I	000216	82	E
I	000217	82	E
I	000218	82	E
I	000219	82	E
I	000220	82	E
I	000221	82	E
I	000222	82	E
I	000223	82	E
I	000224	82	E
I	000225	82	E
I	000226	82	E
I	000227	82	E
I	000228	82	E
I	000229	82	E
I	000230	82	E
I	000231	82	E
I	000232	82	E
I	000233	82	E
I	000234	82	E
I	000235	82	E
I	000236	82	E
I	000237	82	E
I	000238	82	E
I	000239	82	E
I	000240	82	E
I	000241	82	E
I	000242	82	E
I	000243	82	E
I	000244	82	E
I	000245	82	E
I	000246	82	E
I	000247	82	E
I	000248	82	E
I	000249	82	E
I	000250	82	E
I	000251	82	E
I	000252	82	E
I	000253	82	E
I	000254	82	E
I	000255	82	E
I	000256	82	E
I	000257	82	E
I	000258	82	E
I	000259	82	E
I	000260	82	E
I	000261	82	E
I	000262	82	E
I	000263	82	E
I	000264	82	E
I	000265	82	E
I	000266	82	E
I	000267	82	E
I	000268	82	E
I	000269	82	E
I	000270	82	E
I	000271	82	E
I	000272	82	E
I	000273	82	E
I	000274	82	E
I	000275	82	E
I	000276	82	E
I	000277	82	E
I	000278	82	E
I	000279	82	E
I	000280	82	E
I	000281	82	E
I	000282	82	E
I	000283	82	E
I	000284	82	E
I	000285	82	E
I	000286	82	E
I	000287	82	E
I	000288	82	E
I	000289	82	E
I	000290	82	E
I	000291	82	E
I	000292	82	E
I	000293	82	E
I	000294	82	E
I	000295	82	E
I	000296	82	E
I	000297	82	E
I	000298	82	E
I	000299	82	E
I	000300	82	E
I	000301	82	E
I	000302	82	E
I	000303	82	E
I	000304	82	E
I	000305	82	E
I	000306	82	E
I	000307	82	E
I	000308	82	E
I	000309	82	E
I	000310	82	E
I	000311	82	E
I	000312	82	E
I	000313	82	E
I	000314	82	E
I	000315	82	E
I	000316	82	E
I	000317	82	E
I	000318	82	E
I	000319	82	E
I	000320	82	E
I	000321	82	E
I	000322	82	E
I	000323	82	E
I	000324	82	E
I	000325	82	E
I	000326	82	E
I	000327	82	E
I	000328	82	E
I	000329	82	E
I	000330	82	E
I	000331	82	E
I	000332	82	E
I	000333	82	E
I	000334	82	E
I	000335	82	E
I	000336	82	E
I	000337	82	E
I	000338	82	E
I	000339	82	E
I	000340	82	E
I	000341	82	E
I	000342	82	E
I	000343	82	E
I	000344	82	E
I	000345	82	E
I	000346	82	E
I	000347	82	E
I	000348	82	E
I	000349	82	E
I	000350	82	E
I	000351	82	E
I	000352	82	E
I	000353	82	E
I	000354	82	E
I	000355	82	E
I	000356	82	E
I	000357	82	E
I	000358	82	E
I	000359	82	E
I	000360	82	E
I	000361	82	E
I	000362	82	E
I	000363	82	E
I	000364	82	E
I	000365	82	E
I			

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)

$$\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \rho_{\text{CTYPES}(\text{ct_cust_id})}(\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))))$$

cust_id	account_id	avail_balance	ct_cust_id
BALANCES) X ES))))	1	000001	1
	1	000002	1
	1	000003	1
	1	000004	1
	1	000005	1
	1	000006	1
	1	000007	1
	1	000008	1
	1	000009	1
	1	000010	1
	1	000011	1
	1	000012	1
	1	000013	1
	1	000014	1
	1	000015	1
	1	000016	1
	1	000017	1
	1	000018	1
	1	000019	1
	1	000020	1
	1	000021	1
	1	000022	1
	1	000023	1
	1	000024	1
	1	000025	1
	1	000026	1
	1	000027	1
	1	000028	1
	1	000029	1
	1	000030	1
	1	000031	1
	1	000032	1
	1	000033	1
	1	000034	1
	1	000035	1
	1	000036	1
	1	000037	1
	1	000038	1
	1	000039	1
	1	000040	1
	1	000041	1
	1	000042	1
	1	000043	1
	1	000044	1
	1	000045	1
	1	000046	1
	1	000047	1
	1	000048	1
	1	000049	1
	1	000050	1
	1	000051	1
	1	000052	1
	1	000053	1
	1	000054	1
	1	000055	1
	1	000056	1
	1	000057	1
	1	000058	1
	1	000059	1
	1	000060	1
	1	000061	1
	1	000062	1
	1	000063	1
	1	000064	1
	1	000065	1
	1	000066	1
	1	000067	1
	1	000068	1
	1	000069	1
	1	000070	1
	1	000071	1
	1	000072	1
	1	000073	1
	1	000074	1
	1	000075	1
	1	000076	1
	1	000077	1
	1	000078	1
	1	000079	1
	1	000080	1
	1	000081	1
	1	000082	1
	1	000083	1
	1	000084	1
	1	000085	1
	1	000086	1
	1	000087	1
	1	000088	1
	1	000089	1
	1	000090	1
	1	000091	1
	1	000092	1
	1	000093	1
	1	000094	1
	1	000095	1
	1	000096	1
	1	000097	1
	1	000098	1
	1	000099	1
	1	000100	1
1	000101	1	
1	000102	1	
1	000103	1	
1	000104	1	
1	000105	1	
1	000106	1	
1	000107	1	
1	000108	1	
1	000109	1	
1	000110	1	
1	000111	1	
1	000112	1	
1	000113	1	
1	000114	1	
1	000115	1	
1	000116	1	
1	000117	1	
1	000118	1	
1	000119	1	
1	000120	1	
1	000121	1	
1	000122	1	
1	000123	1	
1	000124	1	
1	000125	1	
1	000126	1	
1	000127	1	
1	000128	1	
1	000129	1	
1	000130	1	
1	000131	1	
1	000132	1	
1	000133	1	
1	000134	1	
1	000135	1	
1	000136	1	
1	000137	1	
1	000138	1	
1	000139	1	
1	000140	1	
1	000141	1	
1	000142	1	
1	000143	1	
1	000144	1	
1	000145	1	
1	000146	1	
1	000147	1	
1	000148	1	
1	000149	1	
1	000150	1	
1	000151	1	
1	000152	1	
1	000153	1	
1	000154	1	
1	000155	1	
1	000156	1	
1	000157	1	
1	000158	1	
1	000159	1	
1	000160	1	
1	000161	1	
1	000162	1	
1	000163	1	
1	000164	1	
1	000165	1	
1	000166	1	
1	000167	1	
1	000168	1	
1	000169	1	
1	000170	1	
1	000171	1	
1	000172	1	
1	000173	1	
1	000174	1	
1	000175	1	
1	000176	1	
1	000177	1	
1	000178	1	
1	000179	1	
1	000180	1	
1	000181	1	
1	000182	1	
1	000183	1	
1	000184	1	
1	000185	1	
1	000186	1	
1	000187	1	
1	000188	1	
1	000189	1	
1	000190	1	
1	000191	1	
1	000192	1	
1	000193	1	
1	000194	1	
1	000195	1	
1	000196	1	
1	000197	1	
1	000198	1	
1	000199	1	
1	000200	1	
1	000201	1	
1	000202	1	
1	000203	1	
1	000204	1	
1	000205	1	
1	000206	1	
1	000207	1	
1	000208	1	
1	000209	1	
1	000210	1	
1	000211	1	
1	000212	1	
1	000213	1	
1	000214	1	
1	000215	1	
1	000216	1	
1	000217	1	
1	000218	1	
1	000219	1	
1	000220	1	
1	000221	1	
1	000222	1	
1	000223	1	
1	000224	1	
1	000225	1	
1	000226	1	
1	000227	1	
1	000228	1	
1	000229	1	
1	000230	1	
1	000231	1	
1	000232	1	
1	000233	1	
1	000234	1	
1	000235	1	
1	000236	1	
1	000237	1	
1	000238	1	
1	000239	1	
1	000240	1	
1	000241	1	
1	000242	1	
1	000243	1	
1	000244	1	
1	000245	1	
1	000246	1	
1	000247	1	
1	000248	1	
1	000249	1	
1	000250	1	
1	000251	1	
1	000252	1	
1	000253	1	
1	000254	1	
1	000255	1	
1	000256	1	
1	000257	1	
1	000258	1	
1	000259	1	
1	000260	1	
1	000261	1	
1	000262	1	
1	000263	1	
1	000264	1	
1	000265	1	
1	000266	1	
1	000267	1	
1	000268	1	
1	000269	1	
1	000270	1	
1	000271	1	
1	000272	1	
1	000273	1	
1	000274	1	
1	000275	1	
1	000276	1	
1	000277	1	
1	000278	1	
1	000279	1	
1	000280	1	
1	000281	1	
1	000282	1	
1	000283	1	
1	000284	1	
1	000285	1	
1	000286	1	
1	000287	1	
1	000288	1	
1	000289	1	
1	000290	1	
1	000291	1	
1	000292	1	
1	000293	1	
1	000294	1	
1	000295	1	
1	000296	1	
1	000297	1	
1	000298	1	
1	000299	1	
1	000300	1	
1	000301	1	
1	000302	1	
1	000303	1	
1	000304	1	
1	000305	1	
1	000306	1	
1	000307	1	
1	000308	1	
1	000309	1	
1	000310	1	
1	000311	1	
1	000312	1	
1	000313	1	
1	000314	1	
1	000315	1	
1	000316	1	
1	000317	1	
1	000318	1	
1	000319	1	
1	000320	1	
1	000321	1	
1	000322	1	
1	000323	1	
1	000324	1	
1	000325	1	
1	000326	1	
1	000327	1	
1	000328	1	
1	000329	1	
1	000330	1	
1	000331	1	
1	000332	1	
1	000333	1	
1	000334	1	
1	000335	1	
1	000336	1	
1	000337	1	
1	000338	1	
1	000339	1	
1	000340	1	
1	000341	1	

Relational Algebra: Operations

Another Example

Required Predicate:

account_id 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)

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

$\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES})) \times$
 $\rho_{\text{CTYPES}(\text{ct_cust_id})}(\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = "I"}(\text{CUSTOMER_TYPES}))))$

Relational Algebra: Operations

Another Example

Required Predicate:

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

Finally, we project on account_id:

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

$$\pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \\ \rho_{\text{CTYPES}}(\text{ct_cust_id})(\pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))))))$$

Relational Algebra: Operations

Another Example

Required Predicate:

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

Finally, we project on account_id:

$$\Pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \\ \rho_{\text{CTYPES}(\text{ct_cust_id})}(\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))))))$$

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

$$\Pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \rho_{\text{CTYPES(ct_cust_id)}}(\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))))))$$

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 will 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

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*.

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

NATURAL JOIN (from Silberschatz)

- 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*.

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 Silberschatz)

- 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

Relational Algebra: Operations

NATURAL JOIN (from Silberschatz)

- 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

Relational Algebra: Operations

NATURAL JOIN (from Silberschatz)

- 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

Remember this query expression???

$\Pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \bowtie \rho_{\text{CTYPES}(\text{ct_cust_id})}(\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = "I"}(\text{CUSTOMER_TYPES}))))))$

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 Silberschatz)

- 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

Remember this query expression???

$\pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \rho_{\text{CTYPES}}(\text{ct_cust_id})(\pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = "I"}(\text{CUSTOMER_TYPES}))))))$

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)

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

$$r1 \bowtie r2$$

is a relation with

- heading the set theory union of the headings of $r1$ and $r2$
- 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

$\pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \rho_{\text{CTYPES}}(\text{ct_cust_id}) (\pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = "I"}(\text{CUSTOMER_TYPES}))))))$

Relational Algebra: Operations

NATURAL JOIN Example

$\pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \rho_{\text{CTYPES}}(\text{ct_cust_id})(\pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = "I"}(\text{CUSTOMER_TYPES}))))))$

BECOMES

Relational Algebra: Operations

NATURAL JOIN Example

$\pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \rho_{\text{CTYPES}}(\text{ct_cust_id})(\pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = "I"}(\text{CUSTOMER_TYPES}))))))$

BECOMES

$\pi_{\text{account_id}}(\sigma_{(\text{avail_balance} > 1000) \text{ AND } (\text{cust_type_cd} = "I")})(\text{CUSTOMER_ACCOUNT_BALANCES} \bowtie \text{CUSTOMER_TYPES})$

Relational Algebra: Operations

NATURAL JOIN Example

$\pi_{\text{account_id}}(\sigma_{(\text{avail_balance} > 1000) \text{ AND } (\text{cust_type_cd} = "I")}(\text{CUSTOMER_ACCOUNT_BALANCES} \bowtie \text{CUSTOMER_TYPES}))$

Relational Algebra: Operations

NATURAL JOIN

CUSTOMER_ACCOUNT_BALANCES ⋈ 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 ⋈ **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 ⋈ **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
13	29	50000.00	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
13	29	50000.00	B

$\sigma_{(avail_balance > 1000) \text{ AND } (cust_type_cd = "I")}$ (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
13	29	50000.00	B

$\sigma_{(avail_balance > 1000) \text{ AND } (cust_type_cd = "I")}$ (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	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

$\sigma_{(avail_balance > 1000) \text{ AND } (cust_type_cd = "I")}$ (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	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

$\Pi_{\text{account_id}}(\sigma_{(\text{avail_balance} > 1000) \text{ AND } (\text{cust_type_cd} = \text{"I"})}(\text{CUSTOMER_ACCOUNT_BALANCES} \bowtie \text{CUSTOMER_TYPES}))$

account_id

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

$\Pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \rho_{\text{CTYPES}(\text{ct_cust_id})}(\Pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))))))$

account_id

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

Relational Algebra: Operations

NATURAL JOIN

Requested Predicate:

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:

EMPLOYEE ⋈ **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:

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

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:

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

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:

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

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”:

$\rho_{EMP_DEPT}(fname, lname, dname) (\pi_{fname, lname, name} (EMPLOYEE \bowtie DEPARTMENT))$

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**.

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

$\rho_{EMP_DEPT}(fname, lname, dname) (\Pi_{fname, lname, name} (EMPLOYEE \bowtie DEPARTMENT))$

Relational Algebra: Operations

THETA JOIN (from Silberschatz)

- 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 \geq).
- 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

$\pi_{\text{account_id}}(\sigma_{\text{cust_id} = \text{ct_cust_id}}(\sigma_{\text{avail_balance} > 1000}(\text{CUSTOMER_ACCOUNT_BALANCES}) \times \rho_{\text{CTYPES}}(\text{ct_cust_id})(\pi_{\text{cust_id}}(\sigma_{\text{cust_type_cd} = \text{"I"}}(\text{CUSTOMER_TYPES}))))))$

BECOMES

$\pi_{\text{account_id}}(\text{CUSTOMER_ACCOUNT_BALANCES} \bowtie_{(\text{avail_balance} > 1000) \text{ AND } (\text{cust_type_cd} = \text{"I"})} \text{CUSTOMER_TYPES}))$

Relational Algebra: Operations

THETA JOIN

$\pi_{\text{account_id}}(\text{CUSTOMER_ACCOUNT_BALANCES} \bowtie_{(\text{avail_balance} > 1000) \text{ AND } (\text{cust_type_cd} = \text{"I"})} \text{CUSTOMER_TYPES})$

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	<i>null</i>
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	<i>null</i>
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:

$\rho_{sub}(EMP)$

$\rho_{sup}(EMP)$

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

$$\rho_{sub}(EMP) \bowtie ((sub.sup_emp_id = sup.emp_id) \text{ AND } (sub.start_date < sup.start_date)) \rho_{sup}(EMP)$$

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	<i>null</i>
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:

$$\pi_{\text{sub.emp_id}} (\rho_{\text{sub}} (\text{EMP}) \bowtie ((\text{sub.sup_emp_id} = \text{sup.emp_id}) \text{ AND } (\text{sub.start_date} < \text{sup.start_date})) \rho_{\text{sup}} (\text{EMP}))$$

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

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:

$$\rho_{EMP_BEFORE_SUP(emp_id)}(\pi_{sub.emp_id}(\rho_{sub}(EMP) \bowtie ((sub.sup_emp_id = sup.emp_id) \text{ AND } (sub.start_date < sup.start_date)) \rho_{sup}(EMP)))$$

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:

$$\rho_{EMP_BEFORE_SUP(emp_id)}(\pi_{sub.emp_id}(\rho_{sub}(EMP) \bowtie ((sub.sup_emp_id = sup.emp_id) \text{ AND } (sub.start_date < sup.start_date)) \rho_{sup}(EMP)))$$

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

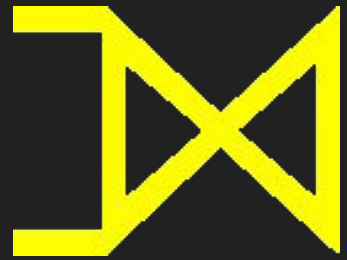


- Full outer join



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

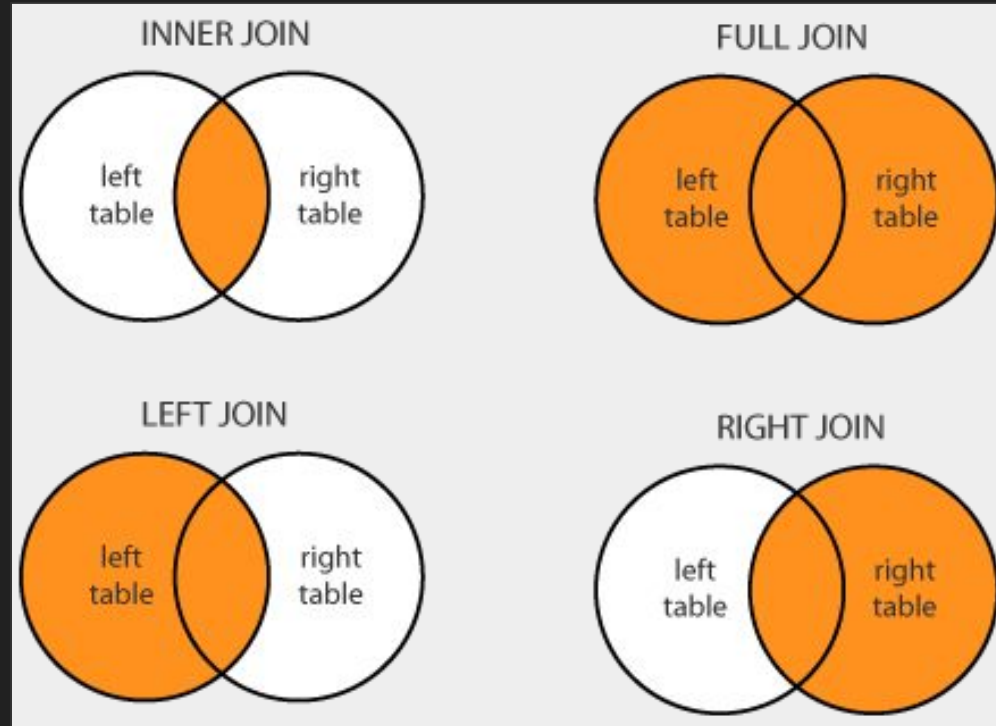


Image from <http://www.dofactory.com/sql/join>



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_NAME_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	<i>null</i>
Woburn Branch	Woburn	<i>null</i>
Quincy Branch	Quincy	13
So. NH Branch	Salem	10
So. NH Branch	Salem	12

BRANCH_NAME_CITY  **BRANCH_NAME_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_NAME_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
<i>null</i>	11	Wilmington
So. NH Branch	12	Salem
Quincy Branch	13	Quincy

BRANCH_NAME_CITY  **BRANCH_NAME_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_NAME_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	<i>null</i>
Woburn Branch	Woburn	<i>null</i>
Quincy Branch	Quincy	13
So. NH Branch	Salem	10
So. NH Branch	Salem	12
<i>null</i>	Wilmington	11

BRANCH_NAME_CITY  **BRANCH_NAME_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	<i>null</i>
Woburn Branch	Woburn	<i>null</i>
Quincy Branch	Quincy	13
So. NH Branch	Salem	10
So. NH Branch	Salem	12
<i>null</i>	Wilmington	11

BRANCH_NAME_CITY  **BRANCH_NAME_CITY**

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

Relational Algebra: Division

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

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 $T \div F$ 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 $T \div F$

- 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 $T \div F$ is $D = \Sigma(T) \setminus \Sigma(F)$

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

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

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

$$F = \{\text{VHS}, \text{DVD}, \text{HQ}\}$$

let t' be (55), for $s = (\text{DVD})$ there is

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

Relational Algebra Division

$T \div F$ may be defined in terms of other relational operators

$$T \div F = \pi_D(T) \setminus (\pi_D(\pi_D(T) \times F) \setminus T)$$

The "missing" tuples of T

Building the complement

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

Proof: Assignment

Property of relational division:

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

if D contains the key of T and $|F| > 1$ then $T \div F = \emptyset$



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 \div **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 \div **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
02169
03079
01887

postal_code_cust_type \div cust_type

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.