January 16, 2013 (Week 2)

- Overview of the three query languages that we are using: relational algebra for sets, relational algebra for bags, SQL
- cross product, join, self join, outer join
  - lecture and then activity 2
  - demo 1 will take place during the activity 2 time
- embedded SQL (i.e., using SQL in a programming language)
  - lecture
  - assignment 2 requires that you use embedded SQL

Which features are in these languages?

Language	Operators
relational algebra for sets works on sets; produces sets	$\sigma, \pi, \cup, \cap, -, \times, \bowtie, \div$ $plus\ renaming\ \rho$
relational algebra for bags works on bags; produces bags	$\sigma, \pi, \cup, \cap, -, \times, \bowtie, \div$ $plus\ renaming\ \rho$
extended operators for relational algebra for bags	duplicate elimination: δ aggregation operators grouping: γ extended project (arithmetic) sorting: τ outerjoin: $\overset{\circ}{\bowtie}_L$ or $\overset{\circ}{\bowtie}_R$ or $\overset{\circ}{\bowtie}$
SQL works on bags; produces bags	all features of extended relational algebra for bags

What did we cover in Week 1?

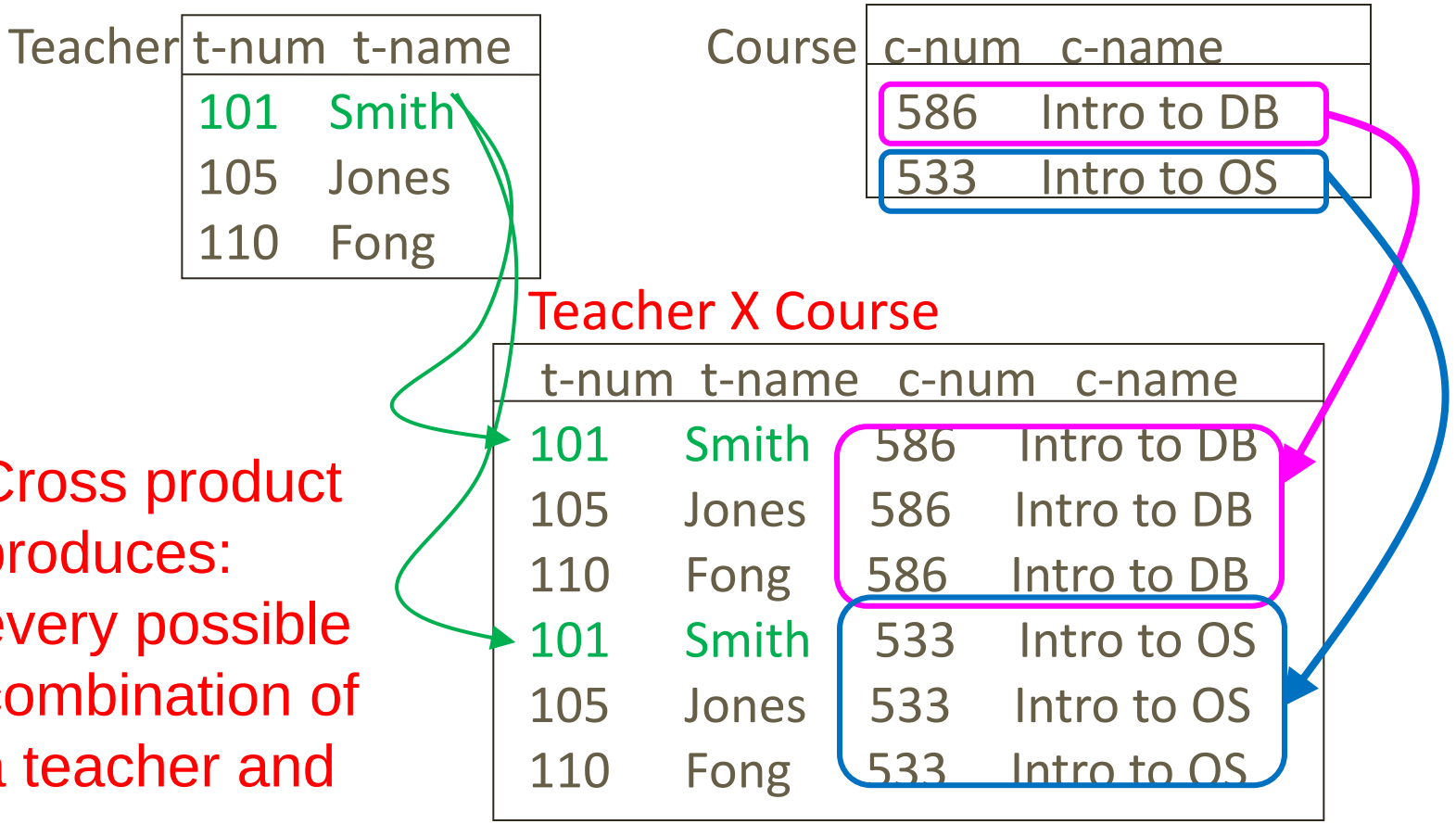
Language	Operators
relational algebra for sets works on sets; produces sets	$\sigma, \pi, \cup, \cap, -, \times, \bowtie, \div$ $plus\ renaming\ \rho$
relational algebra for bags works on bags; produces bags	$\sigma, \pi, \cup, \cap, -, \times, \bowtie, \div$ $plus\ renaming\ \rho$
extended operators for relational algebra for bags features equivalent to $\sigma, \pi, \cup, \cap, -, \times$ for sets	duplicate elimination: δ aggregation operators grouping: γ extended project (arithmetic) sorting: τ outerjoin: $\bowtie_L$ or $\bowtie_R$ or $\bowtie$
SQL works on bags; produces bags	all features of extended relational algebra for bags

What will we cover in Week 2 (tonight)?

Language	Operators
relational algebra for sets works on sets; produces sets	$\sigma, \pi, \cup, \cap, -, \times, \bowtie, \div$ $plus\ renaming\ \rho$
relational algebra for bags works on bags; produces bags	$\sigma, \pi, \cup, \cap, -, \times, \bowtie, \div$ $plus\ renaming\ \rho$
extended operators for relational algebra for bags join, left outer join, right outer join, full outer join	duplicate elimination: δ aggregation operators grouping: γ extended project (arithmetic) sorting: τ outerjoin: $\overset{\circ}{\bowtie}_L$ or $\overset{\circ}{\bowtie}_R$ or $\overset{\circ}{\bowtie}$
SQL works on bags; produces bags	all features of extended relational algebra for bags

Cross product, join, self join, outer join

X cross product operator
produces every possible combination



Cross product produces: every possible combination of a teacher and a course

Cross Product an operator from set theory

Suppose.. A = {a, b, c} B = {1, 2}

then in *set theory*, the cross product is defined as:

A X B = {(a, 1), (b, 1), (c, 1), (a, 2), (b, 2), (c, 2)}

A X B is a set consisting of pairs (2-tuples) where each pair consists of an element from A and an element from B

Cross Product in Set Theory

Suppose.. A = {a, b, c} B = {1, 2} C = {x, y} then

A X B = {(a, 1), (b, 1), (c, 1), (a, 2), (b, 2), (c, 2)}

and (A X B) X C =

{((a,1),x), ((b,1),x), ((c,1),x), ((a,2),x), ((b,2),x), ((c,2),x),
((a,1),y), ((b,1),y), ((c,1),y), ((a,2),y), ((b,2),y), ((c,2),y)}

Cross Product in Relational Algebra vs. Set Theory

Given A = {a, b, c} B = {1, 2} C = {x, y}

then (A X B) X C, in *set theory,* =

{((a,1),x), ((b,1),x), ((c,1),x), ((a,2),x), ((b,2),x), ((c,2),x),

((a,1),y), ((b,1),y), ((c,1),y), ((a,2),y), ((b,2),y), ((c,2),y)}

Codd simplified it in *relational algebra* to:

{(a,1,x), (b,1,x), (c,1,x), (a,2,x), (b,2,x), (c,2,x), (a,1,y), (b,1,y),
(c,1,y), (a,2,y), (b,2,y), (c,2,y)}

by eliminating parentheses….”flattening” the tuples.

Same slide with color eliminated

Given A = {a, b, c} B = {1, 2} C = {x, y} with the cross product

(A X B) X C in *set theory* =

{((a,1),x), ((b,1),x), ((c,1),x), ((a,2),x), ((b,2),x), ((c,2),x),
((a,1),y), ((b,1),y), ((c,1),y), ((a,2),y), ((b,2),y), ((c,2),y)}

(A X B) X C in relational algebra (as implified by Codd):

{(a,1,x), (b,1,x), (c,1,x), (a,2,x), (b,2,x), (c,2,x), (a,1,y), (b,1,y),
(c,1,y), (a,2,y), (b,2,y), (c,2,y)}

by eliminating parentheses….”flattening” the tuples.

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number} = \text{Account}}(\text{Account} \times \text{Deposit})$

Cross product followed by select.

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number} = \text{Account}}(\text{Account} \times \text{Deposit})$

notice the columns

Number	Owner	Balance	Type	Account	T-id	Date	Amount

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number =Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number =Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number =Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Yes! Place in query answer.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number =Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Yes! Place in query answer.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number =Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number =Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number = Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

All combinations fail! →

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number = Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number = Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number =Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Yes!

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number = Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
104	M. Jones	1000.00	checking	104	3	10/29/00	1000.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number = Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
104	M. Jones	1000.00	checking	104	3	10/29/00	1000.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

The first three fail.

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number = Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
104	M. Jones	1000.00	checking	104	3	10/29/00	1000.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Yes! Place in query answer.
Final answer:

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

$\sigma_{\text{Number = Account"}}$(Account X Deposit)

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
104	M. Jones	1000.00	checking	104	3	10/29/00	1000.00
105	H. Martin	10,000.00	checking	105	4	11/02/00	10,000.00

⋈ join operator (defined using σ and X)

Account	Number	Owner	Balance	Type

Deposit	Account	Transaction-id	Date	Amount

Check	Account	Check-number	Date	Amount

$$\sigma_{\text{A.Number=Deposit.Account}}(\text{Account X Deposit})$$

is equivalent to

$$\text{Account} \bowtie_{\text{A.Number=Deposit.Account}} \text{Deposit}$$

Basic Theta Join ≡ INNER JOIN

This query with INNER JOIN:

```
SELECT      C.Name, S.Name
FROM        Customer C INNER JOIN Salesperson S ON
            C.Salesperson = S.Number;
```

is equivalent to this query:

```
SELECT      C.Name, S.Name
FROM        Customer  C JOIN Salesperson S ON
                        C.Salesperson = S.Number;
```

And equivalent to this one – with the join condition in WHERE:

```
SELECT      C.Name, S.Name
FROM        Customer C, Salesperson S
WHERE       C.Salesperson = S.Number;
```

Equivalent queries: SQL and relational algebra

Customer(Number, Name, Address, CRating, CAmount, CBalance, Salesperson)
Salesperson(Number, Name, Address, Office)

```
SELECT  DISTINCT C.Name, S.Name
FROM    Customer C JOIN Salesperson S ON C.Salesperson = S.Number
WHERE C. CRating < 6;
```

$$\pi_{C.Name,S.Name}(\sigma_{C.CRating<6}(\text{Customer C} \bowtie_{C.Salesperson=S.Number} \text{Salesperson S}))$$

```
SELECT  DISTINCT C.Name, S.Name
FROM    Customer C, Salesperson S
WHERE C.Salesperson = S.Number AND C.CRating < 6;
```

$$\pi_{C.Name,S.Name}(\sigma_{C.CRating<6 \text{ AND } C.Salesperson=S.Number}(\text{Customer C X Salesperson S}))$$

Join .. with all six comparators

- Student $\bowtie_{\text{advisor=number}}$ Faculty
- Student S $\bowtie_{\text{S.age < F.age}}$ Faculty F
- Student S $\bowtie_{\text{S.salary} \geq \text{F.salary}}$ Faculty F etc.
- Join is sometimes called “theta-join” or “θ-join” where the θ represents any of the 6 comparators (<, >, =, ≠, ≤, ≥)
  (In PostgreSQL the 6 comparators are (<, >, =, != or <>, >=, <=).)
- The most common join (with equality) is called equi-join.

Extensions to the FROM clause - Joins

There are a number of join types that can be expressed in the WHERE clause:

- inner join (the regular join)
- cross join (this is a cross product)
- natural join (don't use this in practice)
- left outer join
- right outer join
- full outer join

the words "inner" and "outer" are not needed.

Consider this join

sailors

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	40
85	Art	3	25.5
95	Bob	3	63.5

reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

What is the query answer:

```
SELECT sname, day
FROM   sailors JOIN
       reserves
       ON
(sailors.sid = reserves.sid)
```

Which sailors are omitted from this join? Why?

sailors

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	40
85	Art	3	25.5
95	Bob	3	63.5

reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

What is the query answer:

```
SELECT sname, day
FROM   sailors JOIN
       reserves
       ON
(sailors.sid = reserves.sid)
```

What if you want ALL sailors in the answer?

- You can use a LEFT OUTER JOIN

Regular (inner join):
SELECT *
FROM sailors JOIN reserves USING (sid)

Left outer join
SELECT *
FROM sailors LEFT JOIN reserves USING (sid)

You get all sailors-reserves pairs that join plus all sailors that don't join.

What about the attribute values for boats for sailors that don't match? They are the null value.

JOIN vs. LEFT JOIN in PostgreSQL

```
select *
from   sailors S JOIN
       reserves R using (sid)
```

sid	sname	rating	age	bid	day
22	Dustin	7	45	102	1998-10-10
22	Dustin	7	45	103	1998-10-08
22	Dustin	7	45	104	1998-10-07
31	Lubber	8	55.5	102	1998-11-10
31	Lubber	8	55.5	103	1998-11-06
31	Lubber	8	55.5	104	1998-11-12
64	Horatio	7	35	101	1998-09-05
64	Horatio	7	35	102	1998-09-08
74	Horatio	9	35.5	103	1998-09-08
22	Dustin	7	45	101	1998-10-10

```
select *
from   sailors S LEFT JOIN
       reserves R using (sid)
```

sid	sname	rating	age	bid	day
22	Dustin	7	45	101	1998-10-10
22	Dustin	7	45	102	1998-10-10
22	Dustin	7	45	103	1998-10-08
22	Dustin	7	45	104	1998-10-07
29	Brutus	1	33	NULL	NULL
31	Lubber	8	55.5	102	1998-11-10
31	Lubber	8	55.5	103	1998-11-06
31	Lubber	8	55.5	104	1998-11-12
32	Andy	8	25.5	NULL	NULL
58	Rusty	10	35.5	NULL	NULL
64	Horatio	7	35	101	1998-09-05
64	Horatio	7	35	102	1998-09-08
71	Zorba	10	16	NULL	NULL
74	Horatio	9	35.5	103	1998-09-08
85	Art	3	25.5	NULL	NULL
95	Bob	3	63.5	NULL	NULL

(INNER) JOIN vs. LEFT OUTER JOIN

Customer

Number	Name	Address	CRating	CAmount	CBalance	Salesperson
1	smith	xxx	5	1,000	1,000	101
2	jones	yyy	7	5,000	4,000	101
3	wei	zzz	10	10,000	10,000	<null>

Salesperson

Number	Name	Address	Office
101	johnson	aaa	23
102	miller	bbb	26

Customer C INNER JOIN Salesperson S
on C.Salesperson = S.Number gives us:

1 smith xxx 5 1,000 1,000 101 101 johnson aaa 23

1 jones yyy 7 5,000 4,000 101 101 johnson aaa 23

Customer C LEFT OUTER JOIN Salesperson S
on C.Salesperson = S.Number gives us:

1 smith xxx 5 1,000 1,000 101 101 johnson aaa 23

1 jones yyy 7 5,000 4,000 101 101 johnson aaa 23

3 wei zzz 10 10,000 10,000 <null> <null> <null> <null> <null>

(INNER) JOIN vs. RIGHT OUTER JOIN

Customer

Number	Name	Address	CRating	Camount	Cbalance	Salesperson
1	smith	xxx	5	1,000	1,000	101
2	jones	yyy	7	5,000	4,000	101
3	wei	zzz	10	10,000	10,000	<null>

Salesperson

Number	Name	Address	Office
101	johnson	aaa	23
102	miller	bbb	26

Customer C INNER JOIN Salesperson S
on C.Salesperson = S.Number gives:

1 smith xxx 5 1,000 1,000 101 101 johnson aaa 23

1 jones yyy 7 5,000 4,000 101 101 johnson aaa 23

Customer C RIGHT OUTER JOIN Salesperson S
on C.Salesperson = S.Number gives:

1 smith xxx 5 1,000 1,000 101 101 johnson aaa 23

1 jones yyy 7 5,000 4,000 101 101 johnson aaa 23

<null><null><null><null><null><null><null> 102 miller bbb 26

(INNER) JOIN vs. FULL OUTER JOIN

Customer

Number	Name	Address	CRating	CAmount	CBalance	Salesperson
1	smith	xxx	5	1,000	1,000	101
2	jones	yyy	7	5,000	4,000	101
3	wei	zzz	10	10,000	10,000	<null>

Salesperson

Number	Name	Address	Office
101	johnson	aaa	23
102	miller	bbb	26

Customer C INNER JOIN Salesperson S
on C.Salesperson = S.Number gives us:

1 smith xxx 5 1,000 1,000 101 101 johnson aaa 23
1 jones yyy 7 5,000 4,000 101 101 johnson aaa 23

Customer C FULL OUTER JOIN Salesperson S
on C.Salesperson = S.Number gives us:

1 smith xxx 5 1,000 1,000 101 101 johnson aaa 23
1 jones yyy 7 5,000 4,000 101 101 johnson aaa 23
3 wei zzz 10 10,000 10,000 <null><null><null><null><null>
<null><null><null><null><null><null><null> <null> 102 miller bbb 26

INNER JOIN vs OUTER JOIN

an INNER (regular) JOIN includes only those customers that have salespersons; an INNER JOIN includes only the matches

a LEFT OUTER JOIN includes all matches plus all Customers that don't have a Salesperson ….

a RIGHT OUTER JOIN includes all matches plus all Salespersons that are not assigned to any customers

a FULL OUTER JOIN will include all of these!

LEFT OUTER JOIN Exercise

Student

Number	Name	Age	Advisor
1	Bob	22	102
2	Sally	19	<null>
3	Ted	19	101

Faculty

Number	Name	Age
101	Todd	35
102	Beth	22
103	Anya	<null>

```
SELECT *
FROM Student S LEFT OUTER JOIN Faculty F
        ON S.Advisor = F.Number
```

Number	Name	Age	Advisor	Number	Name	Age

Another LEFT OUTER JOIN Exercise

Student

Number	Name	Age	Advisor
1	Bob	22	102
2	Sally	19	<null>
3	Ted	19	101

Faculty

Number	Name	Age
101	Todd	35
102	Beth	22
103	Anya	<null>

```
SELECT S.Name AS SName, F.Name AS FName
FROM Student S LEFT OUTER JOIN Faculty F
        ON S.Advisor = F.Number
WHERE F.Name <> 'Beth'
```

SName	FName

RIGHT OUTER JOIN Exercise

Student

Number	Name	Age	Advisor
1	Bob	22	102
2	Sally	19	<null>
3	Ted	19	101

Faculty

Number	Name	Age
101	Todd	35
102	Beth	22
103	Anya	<null>

```
SELECT *
FROM Student S RIGHT OUTER JOIN Faculty F
   ON S.Advisor = F.Number
```


FULL OUTER JOIN Exercise

Student

Number	Name	Age	Advisor
1	Bob	22	102
2	Sally	19	<null>
3	Ted	19	101

Faculty

Number	Name	Age	
101	Todd	35	
102	Beth	22	
103	Anya	<null>	

```
SELECT *
FROM Student S FULL OUTER JOIN Faculty F
   ON S.Age = F.Age
```


CROSS JOIN

A “CROSS JOIN” is a cross product

The following queries are equivalent:

```
SELECT    *
FROM      Customer, Salesperson;
```

```
SELECT    *
FROM      Customer CROSS JOIN Salesperson;
```

Write this query in relational algebra.

NATURAL JOIN

NATURAL JOIN … like a "macro" that joins tables with an equality check for all attributes with the same name.

Consider the following database

Course (CNumber, CName, Description)

Teacher (TNumber, TName, Phone)

Offering (CNumber, TNumber, Time, Days, Room)

Natural Join

List course name and teacher name for all course offerings. This query can be expressed with the NATURAL JOIN or with an INNER JOIN.

These two queries are equivalent.

```
SELECT      CName, TName
FROM        Course NATURAL JOIN Offering
                NATURAL JOIN Teacher;
```

```
SELECT      CName, TName
FROM        Course C, Offering O, Teacher T
WHERE       C.CNumber = O.CNumber AND
                O.TNumber = T.TNumber;
```

Because the join attributes have the same attribute names.

What if course name and teacher name had the same attribute: name?

Course (CNumber, Name, Description)

Teacher (TNumber, Name, Phone)

Offering (CNumber, TNumber, Time, Days, Room)

```
SELECT  *
FROM    Course NATURAL JOIN Offering  NATURAL JOIN Teacher;
```

would join O.CNumber to C.CNumber AND
O.TNumber AND C.Name to T.Name!!!

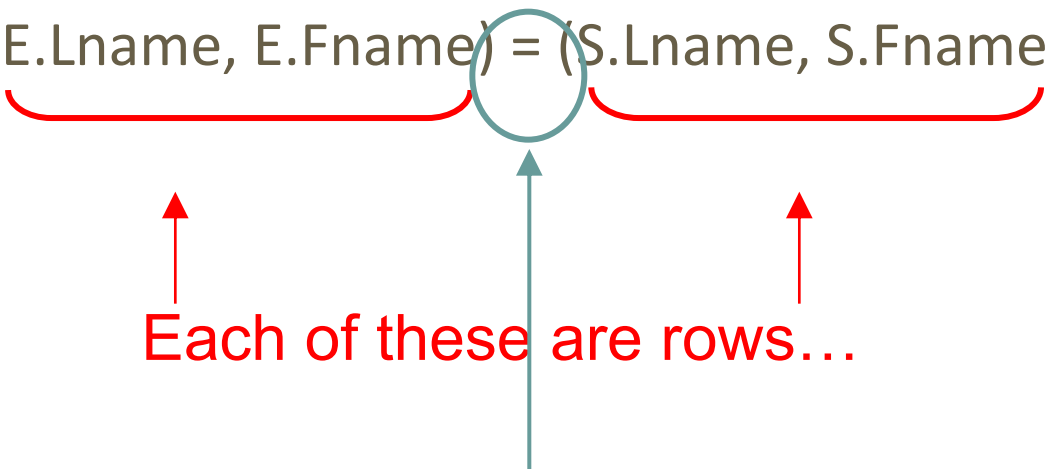
Natural join

- Joins two relations by checking for equality on all pairs of attributes with the same name.
- Eliminates duplicate columns from query answer.
- This is risky; your queries might change if you change your schema. (If you use natural join in SQL queries.)
  This is great for textbooks – queries are simpler.

row value constructors in SQL:1999

- A row value constructor allows you to create a row "on the fly" for example:

```
WHERE (E.Lname, E.Fname) = (S.Lname, S.Fname)
```



and the equality comparison is doing a pair-wise comparison on the two rows.

table value constructors in SQL:1999

- You can also create a table "on the fly"

  VALUES row-value-expr, …, row-value-expr

  Example:

  INSERT INTO movie_stars
  VALUES ("Rocky Horror Picture Show", 1977, "Curry, Tim"),
  ("Rocky V", 1984, "Stallone, Sylvester");

- The part shown in RED font is a table value constructor

What about row/table constructors in relational algebra?

If you need to have either a row or a relation, you can just define it and then use it in relational algebra.

You could say something like:

Let R = {("John", 5, "male"), ("Sue", 6, "female")}

or let R be the table

Name	Age	Gender
John	5	male
Sue	6	female

and then use R in expressions … like R X Student …or whatever