Lecture 7
More on Relational Algebra

CARTESIAN PRODUCT

- R x S: concatenates every tuple in R with every tuple in S
- Binary set operation
- Also known as CROSS PRODUCT or CROSS JOIN
- Relations don't have to be union-compatible

A	B
a1	b1
a2	b2

x

C	D
c1	d1
c2	d2
c3	d3

=

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a1	b1	c3	d3
a2	b2	c1	d1
a2	b2	c2	d2
a2	b2	c3	d3

CARTESIAN PRODUCT (Cont.)

- The relational model does not allow different columns to have the same name within the same schema
- What happens when we form a product of two relations with columns of the same name?
  - Default solution: the attributes are automatically prefixed with the relation name
  - Alternative solution: rename the attributes
    - E.g., suffix the attribute names with 1 and 2 before applying cartesian product
    - Climbs x Climbers attributes: (CId.1, RId, Date, Duration, CId.2, CName, Skill, Age)

Climbers:

CId.2	CName	Skill	Age
123	Edmund	EXP	80
214	Arnold	BEG	25
313	Bridget	EXP	33
212	James	MED	27

Climbs:

CId.1	RId	Date	Duration
123	1	10/10/88	5
123	3	11/08/87	1
313	1	12/08/89	5
214	2	08/07/92	2
313	1	06/07/94	3

CARTESIAN PRODUCT (Cont.)

- Cartesian product is typically used in conjunction with a selection

$$\sigma_{CId.1=CId.2}(Climbs \times Climbers)$$

CId.1	RId	Date	Duration	CId.2	CName	Skill	Age
123	1	10/10/88	5	123	Edmund	EXP	80
123	3	11/08/87	1	123	Edmund	EXP	80
313	1	12/08/89	5	313	Bridget	EXP	33
214	2	08/07/92	2	214	Arnold	BEG	25
313	1	06/07/94	3	313	Bridget	EXP	33

- What can you tell from the result above?

(THETA) JOIN

- A join of two relations: $R \infty_C S = \sigma_C (R \times S)$
  - c is called a *join condition*
- A general JOIN condition : <c> AND <c>…AND <c>
  - Each c is Ai θ Bi
  - θ is one of the comparison operators {=,<,≤,>,≥,≠}
- Example:

$$Climbs \infty_{CId.1=CId.2} Climbers$$

Can join conditions be connected using ORs or NOTs in relational algebra?

Yes

No

Can join conditions be connected using ORs or NOTs in relational algebra?

Yes	0%
No	0%

Can join conditions be connected using ORs or NOTs in relational algebra?

Yes	0%
No	0%

Product vs. Join

- (THETA) JOIN: only combinations of tuples satisfying the join condition appear in the result
- (CARTESIAN) PRODUCT: all combinations of tuples are included in the resulting relation
- Fewer tuples in JOIN than in PRODUCT (might be able to compute more efficiently)
- Tuples whose join attributes are null or for which the join condition is FALSE *do not* appear in the result

EQUIJOIN

- The condition in a theta join is almost always an equality or conjunction of equalities
- EQUIJOIN: the only comparison operator used is =

EQUIJOIN Example

Climbers(C2):

CId	CName	Skill	Age
123	Edmund	EXP	80
214	Arnold	BEG	25
313	Bridget	EXP	33
212	James	MED	27

Climbs(C1):

CId	RId	Date	Duration
123	1	10/10/88	5
123	3	11/08/87	1
313	1	12/08/89	5
214	2	08/07/92	2
313	1	06/07/94	3

$\mathit{Climbs} \infty_{C1.CId=C2.CId} \mathit{Climbers}$

C1.CId	**RId**	**Date**	**Duration**	**C2.CId**	**CName**	**Skill**	**Age**
123	1	10/10/88	5	123	Edmund	EXP	80
123	3	11/08/87	1	123	Edmund	EXP	80
313	1	12/08/89	5	313	Bridget	EXP	33
214	2	08/07/92	2	214	Arnold	BEG	25
313	1	06/07/94	3	313	Bridget	EXP	33

Natural JOIN

- In an EQUIJOIN $R \leftarrow R_1 \infty_c R_2$, the join attribute of R_2 appear redundantly in the result relation R
- In a Natural JOIN (denoted by * or ∞):
  - The redundant join attributes of R_2 are eliminated from R
  - The equality condition is implied and need not be specified

$\text{Climbs} \infty \text{Climbers}:$

CId	RId	Date	Duration	CName	Skill	Age
123	1	10/10/88	5	Edmund	EXP	80
123	3	11/08/87	1	Edmund	EXP	80
313	1	12/08/89	5	Bridget	EXP	33
214	2	08/07/92	2	Arnold	BEG	25
313	1	06/07/94	3	Bridget	EXP	33

Natural JOIN (Cont.)

- Natural JOIN is performed by equating all attribute pairs that have the same name in the two relations
- If this is not the case, a renaming operation is applied first

Exercise

Sailors

sid	sname	rating	age
22	Dustin	7	45
31	Lubber	8	55
58	Rusty	10	35

Boats

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

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Exercise (Cont.)

- Find the names of sailors who have reserved boat 103.
- Find the names of sailors who have reserved a red boat.
- Find the colors of boats reserved by Dustin.
- Find the names of sailors who have reserved at least one boat.
- Find the names of sailors who have reserved a red or a green boat.
- Find the names of sailors who have reserved a red and a green boat.
- Find the names of sailors with age over 20 who have not reserved a red boat.

An Example

Routes

RId	RName	Grade	Rating	Height
1	Last Tango	II	12	100
2	Garden Path	I	2	60
3	The Sluice	I	8	60
4	Picnic	III	3	400

Climbers

CId	Cname	Skill	Age
123	Edmund	EXP	80
214	Arnold	BEG	25
313	Bridget	EXP	33
212	James	MED	27

Climbs

CId	RId	Date	Duration
123	1	10/10/88	5
123	3	11/08/87	1
313	1	12/08/89	5
214	2	08/07/92	2
313	3	06/07/94	3

An Example (Cont.)

- How can we express queries such as “The CId's of climbers who have climbed **all** routes”?

1. Build a relation with all possible pairs of routes and climbers:

$$\text{Allpairs} \leftarrow (\pi_{CId}\text{Climbers}) \times (\pi_{RId}\text{Routes})$$

2. Compute the set of all (CId, RId) pairs for which climber CId has not climbed route RId:

$$\text{NotClimbed} \leftarrow \text{Allpairs} - \pi_{CId,RId}\text{Climbs}$$

An Example (Cont.)

3. $\pi_{CId}(\text{NotClimbed})$: the set of id's of climbers who have not climbed some route
4. The climbers who have climbed all routes are the ones who have not failed to climb some route:

$$\pi_{CId}\text{Climbers} - \pi_{CId}(\text{NotClimbed}) \equiv$$

$$\pi_{CId}\text{Climbers}$$

$$- \pi_{CId}((\pi_{CId}\text{Climbers} \times \pi_{RId}\text{Route}) - \pi_{CId,RId}\text{Climbs})$$

An Example (Cont.)

- Rather than write this long expression, it is easier to use the DIVISION operation
- We could write "Climbers who have climbed all routes" as

$$\pi_{CId,RId}\text{Climbs} \div (\pi_{RId}\text{Routes})$$

- What about "Routes that have been climbed by all climbers"?

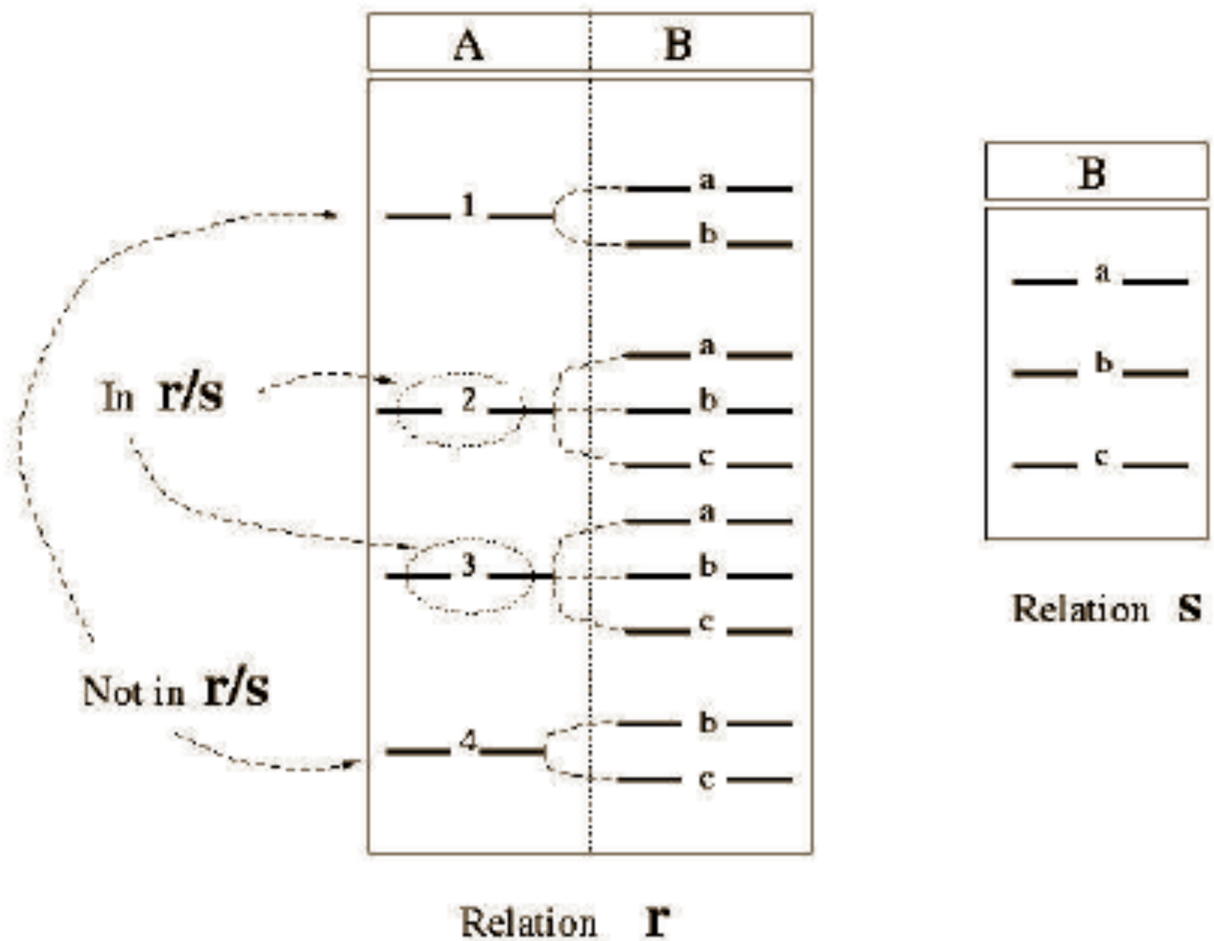
DIVISION

- Goal: Produce the tuples in one relation, r, that match all tuples in another relation, s
  - r (A_1, …, A_n, B_1, …, B_m)
  - s (B_1, …, B_m)
  - r/s, with attributes A_1, …, A_n is the set of all tuples <a> such that for every tuple <b> in s, <a, b> is in r
- Can be expressed in terms of projection, set difference, and product

More Details on DIVISION

- The DIVISION operation is applied to two relations R(Z) ÷ S(X), where X is a subset of Z
- Let Y = Z - X (i.e., Z = X ∪ Y), i.e., let Y be the set of attributes of R that are not attributes of S
- The result of DIVISION is a relation T(Y) that includes a tuple t if tuples t_R appear in R with t_R [Y] = t, and with t_R [X] = t_s *for every tuple* t_s in S
- For a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with *every* tuple in S

DIVISION Illustration



DIVISION Example 1

SSN_PNOS	ESSN	PNO
	123456789	1
	123456789	2
	666884444	3
	453453453	1
	453453453	2
	333445555	2
	333445555	3
	333445555	10
	333445555	20
	999887777	30
	999887777	10
	987987987	10
	987987987	30
	987654321	30
	987654321	20
	888665555	20

SMITH_PNOS	PNO
	1
	2

SSNS=SSN_PNOS ÷ SMITH_PNOS

SSNS	ESSN
	123456789
	453453453

DIVISION Example 2

R	A	B
	a1	b1
	a2	b1
	a3	b1
	a4	b1
	a1	b2
	a3	b2
	a2	b3
	a3	b3
	a4	b3
	a1	b4
	a2	b4
	a3	b4

S	**A**
	a1
	a2
	a3

T= R ÷ S

T	**B**
	b1
	b4

Exercise

- Sailors (sid, sname, rating, age)
- Boats (bid, bname, color)
- Reserves (sid, bid, day)

Exercise (Cont.)

- Find the names of sailors who have reserved all boats.
- Find the names of sailors who have reserved all boats called Interlake.

Renaming

- **Rename Operator**: ρ
- The general rename operation can be expressed by any of the following forms:
  - $\rho_{S\,(B_1,\, B_2,\, \ldots,\, B_n)}$ (R) : R is renamed to S with new column names B_1, B_2, …..B_n
  - ρ_S(R): relation R is renamed to S
  - $\rho_{(B_1,\, B_2,\, \ldots,\, B_n)}$ (R): R is a relation with new column names B_1, B_2, …..B_n

Example 1

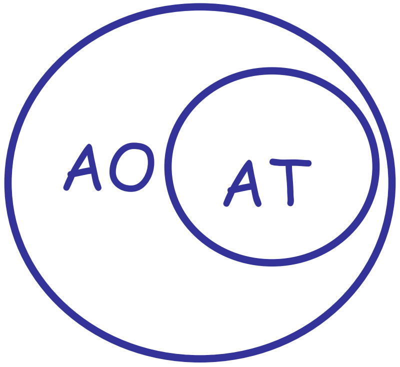
- Find the name of climber who has climbed a certain route at least once
  - AO $\longleftarrow \pi_{CId}$ Climbs
  - R $\longleftarrow$ Climbers ∞ AO
  - Res $\longleftarrow \pi_{CName}$ R

Example 2

- Find the names of climbers who have climbed the same route at least twice
  - ρ_{R1} Climbs
  - R2 $\leftarrow$ R1 $\infty_{R1.CId=Climbs.CId \text{ AND } R1.RId=Climbs.RId \text{ AND } R1.Date \neq Climbs.Date}$ Climbs
  - AT $\leftarrow$ π_{CId} R2
  - R3 $\leftarrow$ Climbers ∞ AT
  - Res $\leftarrow$ π_{CName} R3

Example 3

- Find the name of climber who has climbed a certain route exactly once



$EO \longleftarrow AO\text{-}AT$

$R \longleftarrow Climbers \infty EO$

$Res \longleftarrow \pi_{CName} R$

Complete Set of Relational Algebra Operations

- Complete set: select σ, project π , union $\cup$, set difference - , and cartesian product $\times$
- Any other relational algebra expression can be expressed by a combination of these five operations. For example:
  - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
  - $R \bowtie_{\langle\text{join condition}\rangle} S = \sigma_{\langle\text{join condition}\rangle}(R \times S)$

Recap of Relational Algebra Operations

- Select $\sigma_{<selection\ condition>} R$
- Project $\pi_{<attribute\ list>} R$
- Union $R \cup S$
- Intersection $R \cap S$
- Difference $R - S$
- Cross product $R \times S$
- Join $\mathrm{R} \bowtie_{<join\ condition>} S$
- Natural join $R \bowtie S$
- Division $R \div S$
- Rename $\rho_{<new\ schema>} R$