CIS 353 Database
Relational Model and Relational Algebra

The Relational Model

- Built around a single concept for modeling data: the relation or table.
  - A relational database is a collection of relations.
  - Each relation is a table with rows and columns.
- Supports high-level programming language (SQL).
  - Limited but very useful set of operations
- Most current DBMS are relational (Oracle, IBM DB2, MS SQL)

The Relational Model

CoursesTaken

Student	Course	Grade
Hermione Grainger	Potions	A
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

- Structure: Table (like an array of structs)
- Operations: Relational algebra (selection, projection, conditions, etc)
- Constraints: E.g., grades can be only {A, B, C, D, F}

Relations

- A relation is a two-dimensional table:
  - Relation ⇔ table
  - Attribute ⇔ column name
  - Tuple ⇔ row (not the header row)
- Database ⇔ collection of relations
- A relation has two parts:
  - Schema defines column heads of the table (attributes)
  - Instance contains the data rows (tuples, rows, or records) of the table

Student	Course	Grade
Hermione Grainger	Potions	A
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

CoursesTaken

Schema

CoursesTaken

Student	Course	Grade
Hermione Grainger	Potions	A
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

- The schema of a relation is the name of the relation followed by a parenthesized list of attributes

  CoursesTaken (Student, Course, Grade)
- A design in a relational model consists of a set of schemas.
- Such a set of schemas is called a relational database schema.

Relation and Schema

- Relation is a set of tuples
  - Order in which we present the tuples does not matter
- The attributes in a schema are also a set (not a list)
  - Schema is the same irrespective of order of attributes.

    CoursesTaken(Student, Grade, Course)
  - We specify a “standard” order when we introduce a schema

CoursesTaken

Student	Course	Grade
Hermione Grainger	Potions	A
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

Degree and Cardinality

Student	Course	Grade
Hermione Grainger	Potions	A
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

- Degree is the number of fields/attributes in schema
  - 3 in the table above
- Cardinality is the number of tuples in relation
  - 4 in the table above

Activity

Employee

ID	Name	SSN	Salary	Phone	Email
101	John	AA	50000	12	j@sw
102	Robin	BB	60000	13	r@yh
103	Alya	CC	35000	14	a@hm
104	Yusuf	DD	68000	15	y@ch
105	John	EE	62000	89	j@in
106	Raj	FF	45000	87	r@au
107	Jayant	GG	25000	45	j@us
108	John	HH	35000	15	j@de
109	Neil	II	25000	12	n@uk

Keys of Relations

Keys of Relations

Employee

ID	Name	SSN	Salary	Phone	Email
101	John	AA	50000	12	j@sw
102	Robin	BB	60000	13	r@yh
103	Alya	CC	35000	14	a@hm
104	Yusuf	DD	68000	15	y@ch
105	John	EE	62000	89	j@in
106	Raj	FF	45000	87	r@au
107	Jayant	GG	25000	45	j@us
108	John	HH	35000	15	j@de
109	Neil	II	25000	12	n@uk

Keys of Relations

- Uniquely identifying each tuple
  - Super Keys
  - Candidate Keys
  - Primary Keys
  - Alternate Keys

Employee

ID	Name	SSN	Salary	Phone	Email
101	John	AA	50000	12	j@sw
102	Robin	BB	60000	13	r@yh
103	Alya	CC	35000	14	a@hm
104	Yusuf	DD	68000	15	y@ch
105	John	EE	62000	89	j@in
106	Raj	FF	45000	87	r@au
107	Jayant	GG	25000	45	j@us
108	John	HH	35000	15	j@de
109	Neil	II	25000	12	n@uk

Super Keys

- A super set of keys
- Uniquely identify a tuple
- May contain attributes with NULL value
- A relation can have many super keys
- Super keys may contain additional attributes

Employee

ID	Name	SSN	Salary	Phone	Email
101	John	AA	50000	12	j@sw
102	Robin	BB	60000	13	r@yh
103	Alya	CC	35000	14	a@hm
104	Yusuf	DD	68000	15	y@ch
105	John	EE	62000	89	j@in
106	Raj	FF	45000	87	r@au
107	Jayant	GG	25000	45	j@us
108	John	HH	35000	15	j@de
109	Neil	II	25000	12	n@uk

Super Keys

Superkeys:

{ID}, {SSN}, {ID, Name},
{ID, SSN}, {ID, Phone},
{Name, Phone}, {ID, Email},
{Name, SSN, Phone},
{Name, Email},
{ID, SSN, Phone} . . . . .

Employee

ID	Name	SSN	Salary	Phone	Email
101	John	AA	50000	12	j@sw
102	Robin	BB	60000	13	r@yh
103	Alya	CC	35000	14	a@hm
104	Yusuf	DD	68000	15	y@ch
105	John	EE	62000	89	j@in
106	Raj	FF	45000	87	r@au
107	Jayant	GG	25000	45	j@us
108	John	HH	35000	15	j@de
109	Neil	II	25000	12	n@uk

Superkeys

- A superkey is defined as a subset of attribute types of a relation *R* with the property that no two tuples in any relation state should have the same combination of values for these attribute types
- A superkey specifies a **uniqueness constraint**
- A superkey can have redundant attribute types
  - Example: (ID, SSN, Phone)

Candidate Keys

- **Minimal** super keys

Superkeys:

{ID}, {SSN}, {ID, Name},
{ID, SSN}, {ID, Phone},
{Name, Phone}, {ID, Email},
{Name, SSN, Phone},
{Name, Email},
{ID, SSN, Phone} . . . . .

Employee

ID	Name	SSN	Salary	Phone	Email
101	John	AA	50000	12	j@sw
102	Robin	BB	60000	13	r@yh
103	Alya	CC	35000	14	a@hm
104	Yusuf	DD	68000	15	y@ch
105	John	EE	62000	89	j@in
106	Raj	FF	45000	87	r@au
107	Jayant	GG	25000	45	j@us
108	John	HH	35000	15	j@de
109	Neil	II	25000	12	n@uk

Candidate Keys

- Minimal super keys

Superkeys:

{ID}, {SSN}, {ID, Name},
{ID, SSN}, {ID, Phone},
{Name, Phone}, {ID, Email},
{Name, SSN, Phone},
{Name, Email},
{ID, SSN, Phone} . . . . .

Employee

ID	Name	SSN	Salary	Phone	Email
101	John	AA	50000	12	j@sw
102	Robin	BB	60000	13	r@yh
103	Alya	CC	35000	14	a@hm
104	Yusuf	DD	68000	15	y@ch
105	John	EE	62000	89	j@in
106	Raj	FF	45000	87	r@au
107	Jayant	GG	25000	45	j@us
108	John	HH	35000	15	j@de
109	Neil	II	25000	12	n@uk

Candidate Keys:

{ID}, {SSN}, {Name, Phone},
{Email}

(Candidate) Keys

- A key *K* of a relation scheme *R* is a superkey of *R* with the additional property that removing any attribute type from *K* leaves a set of attribute types that is no superkey of *R*
- A key does not have any redundant attribute types
- The key constraint states that every relation must have at least one key that allows uniquely identifying its tuples
- All super keys can't be candidate keys. All candidate keys are super keys

Primary Keys

- Candidate key with no NULL value
- Generally, never changed

Employee

ID	Name	SSN	Salary	Phone	Email
101	John	AA	50000	12	j@sw
102	Robin	BB	60000	13	r@yh
103	Alya	CC	35000	14	a@hm
104	Yusuf	DD	68000	15	y@ch
105	John	EE	62000	89	j@in
106	Raj	FF	45000	87	r@au
107	Jayant	GG	25000	45	j@us
108	John	HH	35000	15	j@de
109	Neil	II	25000	12	n@uk

Alternate Keys

- Candidate keys except the primary key

Employee

ID	Name	SSN	Salary	Phone	Email
101	John	AA	50000	12	j@sw
102	Robin	BB	60000	13	r@yh
103	Alya	CC	35000	14	a@hm
104	Yusuf	DD	68000	15	y@ch
105	John	EE	62000	89	j@in
106	Raj	FF	45000	87	r@au
107	Jayant	GG	25000	45	j@us
108	John	HH	35000	15	j@de
109	Neil	II	25000	12	n@uk

Primary Keys, and Alternative Keys

- The primary key is used to identify tuples in the relation, to establish connections to other relations, and for storage purposes
  - Entity integrity constraint: attribute types that make up the primary key should always satisfy a NOT NULL constraint
- Only one Candidate Key can be Primary Key
- Other candidate keys are then referred to as alternate keys

Foreign Keys: Referential Integrity

Student

S_ID	Name	Dept_Code	Credits
101	John	101	12
102	Robin	102	14
103	Alya	103	20
104	Yusuf	104	10

Dept

Dept_Code	Dept_Name
101	CSE
102	EEE
103	ECE
104	MECH

Foreign Keys

- A set of attribute types FK in a relation R_1 is a foreign key of R_1 if two conditions are satisfied (referential integrity constraint)
  - The attribute types in FK have the same domains as the primary key attribute types PK of a relation R_2
  - A value FK in a tuple t_1 of the current state r_1 either occurs as a value of PK for some tuple t_2 in the current state r_2 or is NULL

Relational Constraints

Domain constraint	The value of each attribute type A must be an atomic and single value from the domain.
Key constraint	Every relation has a key that allows uniquely identifying its tuples.
Entity integrity constraint	The attribute types that make up the primary key should always satisfy a NOT NULL constraint.
Referential integrity constraint	A foreign key FK has the same domain as the primary key PK attribute type(s) it refers to and either occurs as a value of PK or NULL.

Example Relational Data Model

SUPPLIER(SUPNR:integer, SUPNAME:string, SUPADDRESS:string, SUPCITY:string, SUPSTATUS:integer)

PRODUCT(PRODNR:integer, PRODNAME:string, PRODTYPE:string, AVAILABLE QUANTITY:integer)

SUPPLIES(*SUPNR*, *PRODNR*:integer, PURCHASE_PRICE:real, DELIV_PERIOD:integer)

PURCHASE_ORDER(PONR:integer, PODATE:date, *SUPNR*:integer)

PO_LINE(*PONR*:integer, *PRODNR*:integer, QUANTITY:integer)

Example Relational Data Model

Supplier

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
...				

Product

PRODNR	PRODNAME	PRODTYPE	AVAILABLE_QUANTITY
0119	Chateau Miraval, Cotes de Provence Rose, 2015	rose	126
0384	Dominio de Pingus, Ribera del Duero, Tempranillo, 2006	red	38
...			

Supplies

SUPNR	PRODNR	PURCHASE_PRICE	DELIV_PERIOD
21	0119	15.99	1
21	0384	55.00	2
...			

Purchase_Order

PONR	PODATE	SUPNR
1511	2015-03-24	37
1512	2015-04-10	94
...		

PO_Line

PONR	PRODNR	QUANTITY
1511	0212	2
1511	0345	4
...		

Recap

sid	fname	lname	login	age	GPA
C123456	John	Smith	smithx02@ece.abc.edu	19	3.3
C234561	Karen	Johns	johnsx05@cs.abc.edu	18	3.0
C345612	Mary	Anderson	anderx10@math.abc.edu	20	3.5
C456123	Helen	Henderson	hendex05@ece.abc.edu	18	2.9
C561234	John	Smith	smithx99@cs.abc.edu	19	3.8
C612345	Aidan	Cocke	cockex35@ece.abc.edu	18	3.3

Relational Query Languages

Query languages: Allow manipulation and retrieval of data from a database.

Relational model supports simple, powerful QLs:

- Strong formal foundation based on logic.
- Allows for optimization.

Query Languages != programming languages!

- QLs not intended to be used for complex calculations.
- QLs support easy, efficient access to large data sets.

Formal Relational Query Languages

Two mathematical Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:

- *Relational Algebra*: More operational (imperative), very useful for representing execution plans. (a procedural programming language)
- *Relational Calculus*: Lets users describe what they want, rather than how to compute it. (Non-operational, *declarative*, *basis for SQL*.)

Relational Algebra

- Operator takes in a relation and output a different relation
- Pure relational algebra has set semantics
  - No duplicate tuples in a relation instance
- Basic operators:
  - Selection (σ) Selects a subset of rows from relation.
  - Projection (π) Deletes unwanted columns from relation.
  - Cross-product (×) Allows us to combine two relations.
  - Set-difference (-) Tuples in reln. 1, but not in reln. 2.
  - Union (U) Tuples in reln. 1 and in reln. 2.
- Additional operators:
  - Intersection, join, renaming: Not essential, but (very!) useful.
- Since each operator returns a relation, operators can be composed!

Relational Algebra Operators: Unary

- Unary Operators: on **single relation**
- **Projection** (π): Retains only desired columns (vertical)
- **Selection** (σ): Selects a subset of rows (horizontal)
- **Renaming** (ρ): Rename attributes and relations.

Projection (π)

- Discards attributes that are not in *projection list*.
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
- No duplicates in result!
- *Result* relation can be the *input* for another relational algebra operation! (*Operator composition.*)

Projection (π)

Sailor (sid, sname, rating, age)

- **List the names and ratings of all the sailors**

Projection (π)

Sailor (sid, sname, rating, age)

- **List the names and ratings of all the sailors**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Projection (π)

Sailor (sid, sname, rating, age)

- **List the names and ratings of all the sailors**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Sailor

$\pi_{sname,rating}$ **(Sailor)**

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

Projection (π)

Sailor (sid, sname, rating, age)

- **List the age of all the sailors**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

age
35.0
55.5

Selection (σ)

- Selects rows that satisfy *selection condition*.
- *Schema* of result identical to schema of (only) input relation.
- Selects a subset of rows (horizontal) *Result* relation can be the *input* for another relational algebra operation! (*Operator composition.*)

Selection (σ)

Sailor (sid, sname, rating, age)

- **List the tuples where sailor rating is more than 8**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Selection (σ)

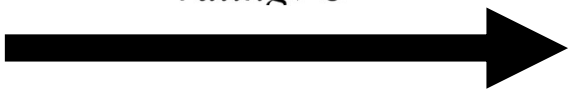
Sailor (sid, sname, rating, age)

- **List the tuples where sailor rating is more than 8**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Sailor

$\sigma_{rating>8}$ **(Sailor)**

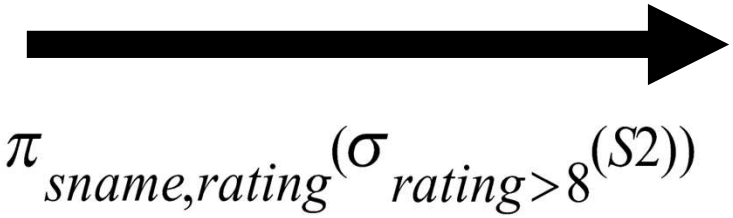


sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Composing Select and Project

- **Which sailors have a rating more than 8**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0



sname	rating
yuppy	9
rusty	10

What about:

$\sigma_{rating>8}(\pi_{sname}(S1))$

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

S1

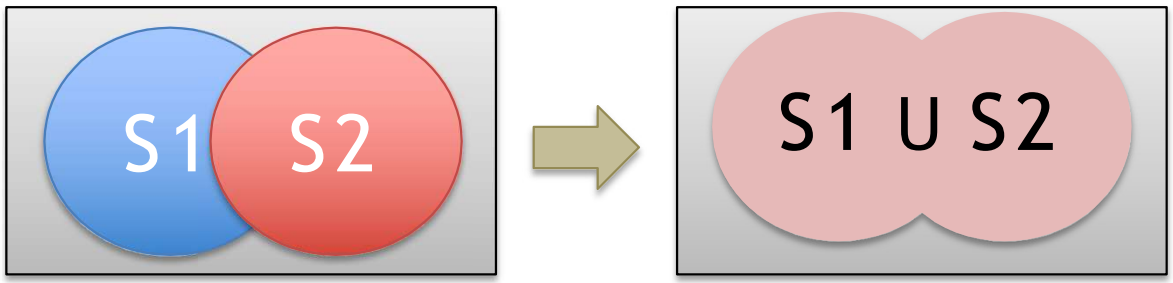
Relational Algebra Operators: Binary

- Binary Operators: on **pairs of relations**
- **Union** ($\cup$): Tuples in r1 or in r2.
- **Set-difference** (—): Tuples in r1, but not in r2.
- **Cross-product** ($\times$): Allows us to combine two relations.

Union (U)

All of these operations take two input relations, which must be *union-compatible*:

- Same number of fields.
- `Corresponding' fields have the same type.



Union (U)

Relational *Instance* **S1**

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Relational *Instance* **S2**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

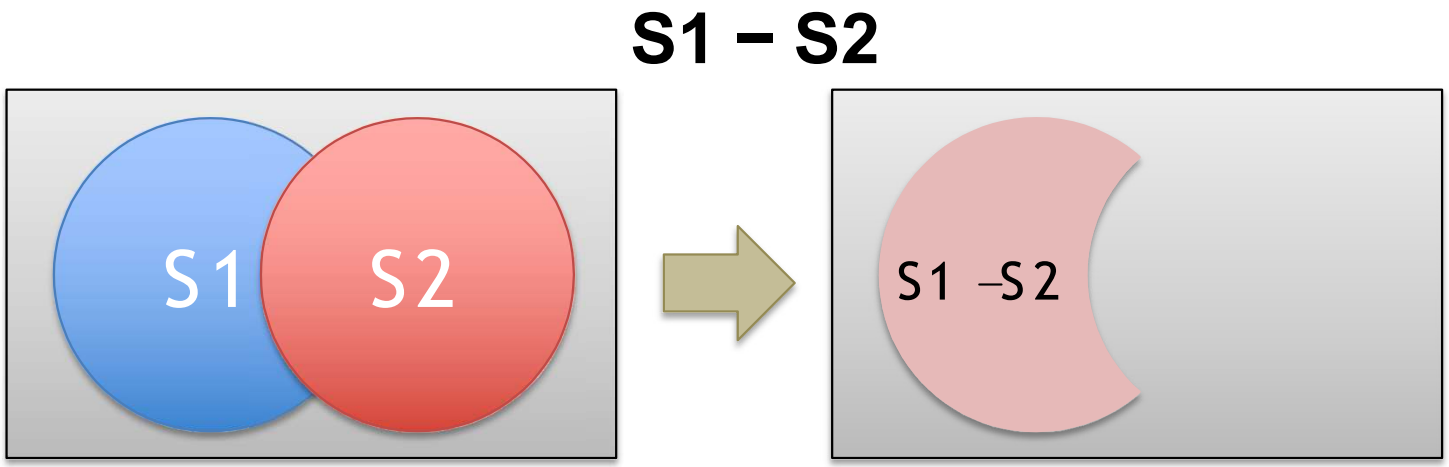
S1 ∪ **S2**

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
28	yuppy	9	35.0
44	guppy	5	35.0

Set Difference (−)

Same as with union, both input relations must be *compatible*.

SQL Expression: EXCEPT



Set Difference (−), cont.

Relational *Instance* **S1**

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Relational *Instance* **S2**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S1 − S2

sid	sname	rating	age
22	dustin	7	45

S2 − S1

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

Cross-Product (×)

- **R1 × S1:** Each row of **R1** paired with each row of **S1**
- Result schema has one field per field of S1 and R1, with field names `inherited' if possible.
- Conflict: Both S1 and R1 have a field called sid.

How many rows in result? |R1|*|S1|
Schema compatability? Not needed.
Duplicates? None generated.

Cross-Product (×)

R1:

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

×

S1:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
22	101	10/10/96	31	lubber	8	55.5
22	101	10/10/96	58	rusty	10	35.0

sid	bid	day	sid	sname	rating	age
58	103	11/12/96	22	dustin	7	45.0
58	103	11/12/96	31	lubber	8	55.5
58	103	11/12/96	58	rusty	10	35.0

=

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
22	101	10/10/96	31	lubber	8	55.5
22	101	10/10/96	58	rusty	10	35.0
58	103	11/12/96	22	dustin	7	45.0
58	103	11/12/96	31	lubber	8	55.5
58	103	11/12/96	58	rusty	10	35.0

Renaming (ρ = “rho”)

- *Renames relations and their attributes:*
- Note that relational algebra doesn’t require names.
  - We could just use positional arguments.

$$\rho(\text{Temp1}(1 \rightarrow \text{sid1}, 4 \rightarrow \text{sid2}), R1 \times S1)$$

Output Relation Name

Renaming List: position → New Name

Input Relation

R1 × S1

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
22	101	10/10/96	31	lubber	8	55.5
22	101	10/10/96	58	rusty	10	35.0
58	103	11/12/96	22	dustin	7	45.0
58	103	11/12/96	31	lubber	8	55.5
58	103	11/12/96	58	rusty	10	35.0

Temp1

sid1	bid	day	sid2	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
22	101	10/10/96	31	lubber	8	55.5
22	101	10/10/96	58	rusty	10	35.0
58	103	11/12/96	22	dustin	7	45.0
58	103	11/12/96	31	lubber	8	55.5
58	103	11/12/96	58	rusty	10	35.0

1. What are the names of employees who have a salary of more than 7000?

2. Find the employee ids whose first name is James, have a salary less than 10000 and are from department number 40

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	JOINING_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	Gerald	Cambrault	34675	AD_PRES	5500	10
101	Renske	Ladwig	34837	AD_VP	15000	20
102	Janette	King	35230	AD_VP	7000	20
103	Sarath	Sewall	35477	IT_PROG	12000	30
104	William	Gietz	35627	IT_PROG	5100	30
105	Jennifer	Whalen	35662	IT_PROG	4900	30
106	Britney	Everett	35733	IT_PROG	5800	30
107	Anthony	Cabrio	35788	IT_PROG	5600	30
108	Alexis	Bull	35861	FI_MGR	7500	40
109	Adam	Fripp	36033	FI_ACCOUNT	8000	40
110	James	Marlow	36066	FI_ACCOUNT	9000	50
111	James	Landry	36174	FI_ACCOUNT	8500	50
112	Payam	Kaufling	36260	FI_ACCOUNT	9500	50
113	Shelley	Higgins	36480	FI_ACCOUNT	8500	50
114	Shanta	Vollman	36501	PU_MAN	10500	50
115	Irene	Mikkilineni	36506	PU_CLERK	10000	50
116	Mozhe	Atkinson	36593	PU_CLERK	9500	50

Query Tree

- Represents the input relations of query as leaf nodes of the tree

- Represents the relational algebra operations as internal nodes

Find names of sailors who've reserved boat #103

Sailors

Sid	Sname	Rating	Age
28	Yuppy	9	35
31	Lubber	8	55
44	Guppy	5	35
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	10/10/23
58	103	11/12/23

Find names of sailors who've reserved boat #103

Sailors x Reserves

Sid	Sname	Rating	Age	Sid	Bid	Day
28	Yuppy	9	35	22	101	10/10/23
28	Yuppy	9	35	58	103	11/12/23
31	Lubber	8	55	22	101	10/10/23
31	Lubber	8	55	58	103	11/12/23
44	Guppy	5	35	22	101	10/10/23
44	Guppy	5	35	58	103	11/12/23
58	Rusty	10	35	22	101	10/10/23
58	Rusty	10	35	58	103	11/12/23

Find names of sailors who've reserved boat #103

$$\pi_{\text{sname}}(\sigma_{\text{Sailros.sid = Reserves.sid} \wedge \text{Bid=103}}(\text{Sailors x Reserves}))$$

Sid	Sname	Rating	Age	Sid	Bid	Day
28	Yuppy	9	35	22	101	10/10/23
28	Yuppy	9	35	58	103	11/12/23
31	Lubber	8	55	22	101	10/10/23
31	Lubber	8	55	58	103	11/12/23
44	Guppy	5	35	22	101	10/10/23
44	Guppy	5	35	58	103	11/12/23
58	Rusty	10	35	22	101	10/10/23
58	Rusty	10	35	58	103	11/12/23

Activity

1. List the names of all authors who are book editors
2. List the names of all authors who are not book editors
3. List the names of all authors who have at least one publication

$r(author)$

author_id	first_name	last_name
1	John	McCarthy
2	Dennis	Ritchie

$r(pub)$

pub_id	title	book_id
1	LISP	1
2	Unix	2

$r(author\_pub)$

author_id	pub_id	author_position
1	1	1
2	2	1

$r(book)$

book_id	book_title	month	year	editor
1	CACM	April	1960	1
2	CACM	July	1974	2

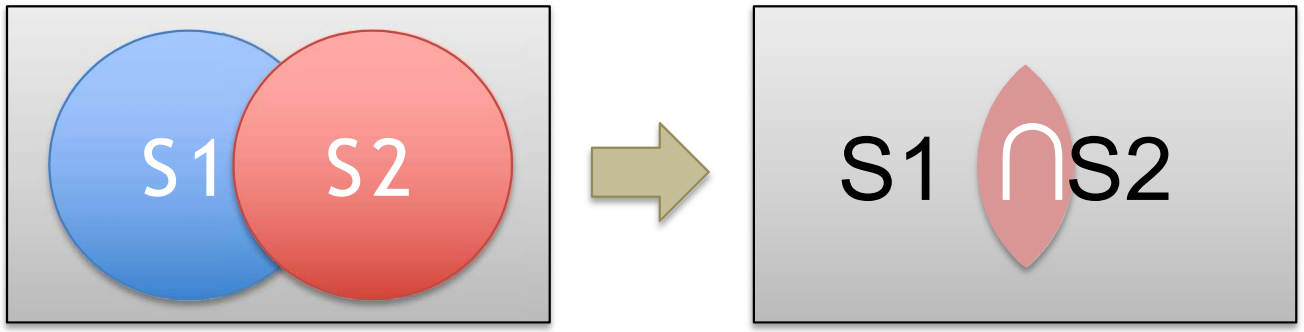
Relational Algebra Operators: Compound

- Compound Operators: common "*macros*" for the above
- **Intersection** (∩): Tuples in r1 and in r2.
- **Joins** ($\bowtie_{\$}$, $\bowtie$): Combine relations that satisfy predicates

Intersection

All of these operations take two input relations, which must be *union-compatible*:

- Same number of fields.
- `Corresponding' fields have the same type.
- Equivalent to: S1 — (S1 — S2)



Intersection (∩)

Relational *Instance* **S1**

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Relational *Instance* **S2**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Intersection (∩)

Relational *Instance* **S1**

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Relational *Instance* **S2**

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

S1 ∩S2

Join

- Joins are compound operators (like intersection):
  - Generally, $\sigma_{!}$(R × S)
- Two kinds:
  - Inner Join:
    - **Theta Join** ($\bowtie_{!}$): join on **logical expression** #
    - **Equi-Join:** theta join with theta being a conjunction of equalities
    - **Natural Join** ($\bowtie$): equi-join on all matching column names
  - Outer Join:
    - Left Outer Join
    - Right Outer Join
    - Full Outer Join

Theta Join ($\bowtie_!$) Example

- Find the sailors who reserved at least one boat
  - **1st step: R1 $\bowtie_{sid=sid}$ S1**
  - **2nd step: π_{sname}(R1 $\bowtie_{sid=sid}$ S1)**

R1:

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

$\bowtie_{sid=sid}$

S1:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

=

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
58	103	11/12/96	58	rusty	10	35.0

Another Theta Join ($\bowtie_!$) Example

- **Example:** *More senior sailors for each sailor.*
- **1: Rename the second S1: ρ(S2,S1)**
- 2: S1 $\bowtie_{s1.age < s2.age}$ S2

S1:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Another Theta Join ($\bowtie_!$) Example

- **Example:** *More senior sailors for each sailor.*
- **1: Rename the second S1: ρ(S2,S1)**
- 2: S1 $\bowtie_{s1.age < s2.age}$ S2

S1:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

$\bowtie$ **S1.age < S2.age**

S2:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1				S2			
sid	sname	rating	age	sid	sname	rating	age2
22	dustin	7	45.0	31	lubber	8	55.5
58	rusty	10	35.0	22	dustin	7	45.0
58	rusty	10	35.0	31	lubber	8	55.5

Another Theta Join ($\bowtie_{!}$) Example

S1:

f1	f2	f3	f4
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

x

S1:

f1	f2	f3	f4
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1				S1			
f1	f2	f3	f4	f5	f6	f7	f8
22	dustin	7	45.0	22	dustin	7	45.0
22	dustin	7	45.0	31	lubber	8	55.5
22	dustin	7	45.0	58	rusty	10	35.0
31	lubber	8	55.5	22	dustin	7	45.0
31	lubber	8	55.5	31	lubber	8	55.5
31	lubber	8	55.5	58	rusty	10	35.0
58	rusty	10	35.0	22	dustin	7	45.0
58	rusty	10	35.0	31	lubber	8	55.5
58	rusty	10	35.0	58	rusty	10	35.0

S1				S1			
sid	sname	rating	age	sid	sname	rating	age2
22	dustin	7	45.0	31	lubber	8	55.5
58	rusty	10	35.0	22	dustin	7	45.0
58	rusty	10	35.0	31	lubber	8	55.5

Equi-Join

Equi-Join: A special case of condition join where the condition *c* contains only ***equalities*.**

$\mathbf{R1} \bowtie_{sid} \mathbf{S1}$

Theta join with AND of = predicates

Result schema similar to cross-product, but only one copy of fields for which equality is specified.

Equi Join ($\bowtie_!$) Example

- Find the sailors who reserved at least one boat
  - **1st step: R1 $\bowtie_{sid=sid}$ S1**
  - **2nd step: π_{sname}(R1 $\bowtie_{sid=sid}$ S1)**

R1:

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

$\bowtie_{sid=sid}$

S1:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

=

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
58	103	11/12/96	58	rusty	10	35.0

Example: STUDENT $\bowtie_{\text{Student.subid} = \text{Subject.id}}$ SUBJECT

Student

sid	name	subid
101	Alex	10
102	Maria	11

$\bowtie_{\text{Student.subid} = \text{Subject.sid}}$

Subject

sid	name
10	Math
10	English
11	Music
11	Sports

sid	name	subid	sid	name
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

Natural Join (⋈)

- Special case of Equi-join in which equalities are specified for all matching fields and duplicate fields are projected away

$$\mathbf{R} \bowtie \mathbf{S} = \pi_{\text{unique fld.}}\, \sigma_{\text{eq. matching fld.}}(\mathbf{R} \times \mathbf{S})$$

- Compute R × S
- Select rows where fields appearing in both relations have **equal** values
- Project onto the set of all **unique** fields.

Natural Join (⋈) Example

- $R \bowtie S = \pi_{\text{unique fld.}} \sigma_{\text{eq. matching fld.}} (R \times S)$

R1:

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

$R1 \bowtie S1$

S1:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

sid	bid	day	sname	rating	age
22	101	10/10/96	dustin	7	45.0
58	103	11/12/96	rusty	10	35.0

Natural Join (⋈) Example

- $R \bowtie S = \pi_{\text{unique fld.}}\ \sigma_{\text{eq. matching fld.}}(R \times S)$

$R1 \bowtie S1$

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
22	101	10/10/96	31	lubber	8	55.5
22	101	10/10/96	58	rusty	10	35.0
58	103	11/12/96	22	dustin	7	45.0
58	103	11/12/96	31	lubber	8	55.5
58	103	11/12/96	58	rusty	10	35.0

sid	bid	day	sname	rating	age
22	101	10/10/96	dustin	7	45.0
58	103	11/12/96	rusty	10	35.0

R1:

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

S1:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Example: STUDENT $\bowtie_{\text{Student.subid} = \text{Subject.id}}$ SUBJECT

Courses

cid	course	dept
CS01	Database	CS
ME01	Mechanics	ME
EE01	Electronics	EE

Courses $\bowtie$ HoD

HoD

dept	head
CS	Alex
ME	Maya
EE	Mira

cid	course	dept	head
CS01	Database	CS	Alex
ME01	Mechanics	ME	Maya
EE01	Electronics	EE	Mira

Find names of sailors who've reserved boat #103

Sailors

Sid	Sname	Rating	Age
28	Yuppy	9	35
31	Lubber	8	55
44	Guppy	5	35
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	10/10/23
58	103	11/12/23

Find names of sailors who've reserved boat #103

Solution 1: $\pi_{\text{sname}}((\sigma_{\text{bid=103}}\text{Reserves}) \bowtie \text{Sailors})$

Solution 2: $\pi_{\text{sname}}(\sigma_{\text{bid=103}}(\text{Reserves} \bowtie \text{Sailors}))$

Find names of sailors who've reserved boat #103

Solution 3:

ρ(Temp1,($\sigma_{bid=103}$Reserves)

ρ(Temp2,(Temp1 $\bowtie$ Sailors)

π_{sname}(Temp2)

Find sailors who've reserved a red or a green boat

Sailors

Sid	Sname	Rating	Age
28	Yuppy	9	35
31	Lubber	8	55
44	Guppy	5	35
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	10/10/23
58	103	11/12/23

Find sailors who've reserved a red or a green boat

Can identify all red or green boats, then find sailors who've reserved one of these boats:

$\rho(\text{Tempboats}, (\sigma_{\text{color='red' V color='green'}}\text{Boats}))$

$\pi_{\text{sname}}(\text{Tempboats} \bowtie \text{Reserves} \bowtie \text{Sailors})$

- Can also define Tempboats using union!
- What happens if $\vee$ is replaced by $\wedge$ in this query?

Find sailors who've reserved a red and a green boat

Sailors

Sid	Sname	Rating	Age
28	Yuppy	9	35
31	Lubber	8	55
44	Guppy	5	35
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	10/10/23
58	103	11/12/23
58	104	11/13/23

Find sailors who've reserved a red and a green boat

Previous approach won't work! Must identify sailors who've reserved red boats, sailors who've reserved green boats, then find the intersection (note that *sid* is a key for Sailors):

$\rho(\text{Tempred}, \pi_{\text{sid}}((\sigma_{\text{color='red'}}\text{Boats}) \bowtie \text{Reserves}))$

$\rho(\text{Tempgreen}, \pi_{\text{sid}}((\sigma_{\text{color='green'}}\text{Boats} \bowtie \text{Reserves}))$

$\pi_{\text{sname}}((\text{Tempred} \cap \text{Tempgreen}) \bowtie \text{Sailors})$

An Example of a "Rewrite": Push-Down

- Want reservations for sailors whose age > 40

$\sigma_{age > 40}(R1 \bowtie S1)$

sid	bid	day	sname	rating	age
22	101	10/10/96	dustin	7	45.0
58	103	11/12/96	rusty	10	35.0

R1:

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

S1:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Q: Any other expressions?

Another equiv. exp: $R1 \bowtie \sigma_{age > 40} S1$

➔ This may be cheaper to compute!

An Example of a "Rewrite": Eliminating Nesting

- *Names of sailors who've **not** reserved boat #103:*

One approach:

$$\pi_{\text{sname}}(R) - \pi_{\text{sname}}((\sigma_{\text{bid}=103}(R)) \bowtie S))$$

R:

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

S:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Left Outer Join:

A

num	square
2	4
3	9
4	16

B

num	cube
2	8
3	27
5	75

num	square	cube
2	4	8
3	9	27
4	16	--

Right Outer Join:

A

num	square
2	4
3	9
4	16

B

num	cube
2	8
3	27
5	75

num	cube	square
2	8	4
3	27	9
5	75	--

Full Outer Join:

A

num	square
2	4
3	9
4	16

B

num	cube
2	8
3	27
5	75

num	square	cube
2	4	8
3	9	27
4	16	--
5	--	75

Activity

- **lives**(person-name, street, city)
- **works**(person-name, company-name, salary)
- **located-in**(company-name, city)
- **manages**(person-name, manager-name)

1. Find all tuples in **works** of all persons who work for the City Bank company

2. Find the name of persons working at City Bank who earn more than $50,000

3. Find the name and city of all persons who work for City Bank and earn more than 50,000.

4. Find names of all persons who do not work for City Bank

5. Find names of all persons who live in the same city as the company they work for.

6. Find names of all persons who live in the same city and on the same street as their manager