Formal Relational Query Language

Relational Algebra

Relation Algebra-Introduction

- **Query Language**
  - user requests information from the database.
  - Two types: Procedural and non-procedural
- **Procedural language**
  - User instructs the system to perform a sequence of operations on the database to compute the desired result.
  - Relational Algebra
- **Non procedural language**
  - User describes the desired information without giving a specific procedure for obtaining that information.
  - Relational Calculus

Relational Operations

- Applied to either a single relation or a pair of relations
- Result is always a single relation
- Several operations can be combined
- Can be applied to results of queries

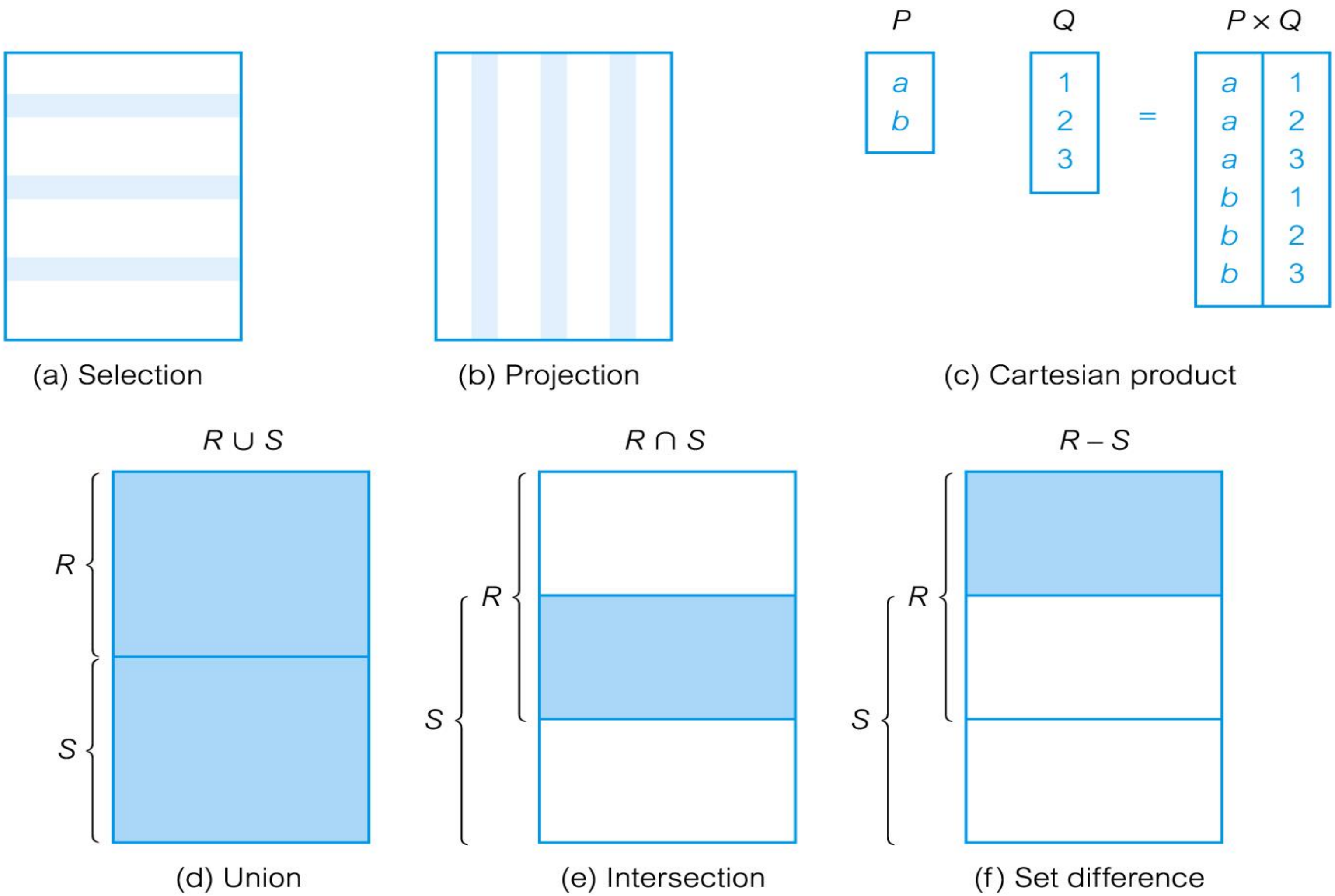
Relational Algebra

- Procedural query language.
- Six fundamental operations:
  - Select
  - Project
  - Rename

    Unary operations
  - Cartesian product
  - Union
  - Set Difference

    Binary operations
- Other operations:
  - Intersection
  - Natural join
  - Assignment

Relational Algebra Operations



Relational Algebra Operations

T

A	B
a	1
b	2

U

B	C
1	x
1	y
3	z

$T \bowtie U$

A	B	C
a	1	x
a	1	y

(g) Natural join

$T \triangleright_B U$

A	B
a	1

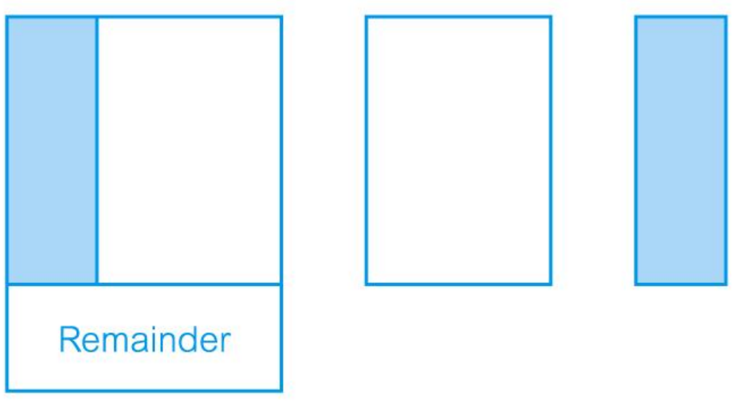
(h) Semijoin

$T ⟕_C U$

A	B	C
a	1	x
a	1	y
b	2	

(i) Left Outer join

R *S* $R \div S$



(j) Divis on (shaded area)

V

A	B
a	1
a	2
b	1
b	2
c	1

W

B
1
2

$V \div W$

A
a
b

Example of division

Relational Algebra operations - Example

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary}>=85000}(instructor)$ Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\,salary}(instructor)$ Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
$\bowtie$ (Natural join)	$instructor \bowtie department$ Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
$\times$ (Cartesian product)	$instructor \times department$ Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
$\cup$ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$ Output the union of tuples from the two input relations.

Select

- Selects tuples that satisfy a given predicate.
- Lowercase Greek letter sigma (σ)
- The predicate appears as a subscript to σ.
- In relational algebra, the term *select* corresponds to where clause in SQL.

Select - Example

select those tuples of the *instructor* relation where the instructor is in the “Physics” department

- $\sigma_{\textit{dept name} = \text{“Physics”}}$ (*instructor*)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Select - Exercise

- Find all instructors with salary greater than $90,000
- Find the instructors who works in Music department
- Find the instructor whose id is 76543
- Find the instructors in Physics with a salary greater than $90,000

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Project

- It is used to list selected attributes of a relation.
- Denoted by the uppercase Greek letter pi (π)

Project - Example

List the id, name and salary of the instructors

- $\pi_{ID,\ name,\ salary}\ (instructor\ )$

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Project - Exercise

- *List the id, name and salary of the instructors*
- *List the id and salary of the instructors.*
- Find the name of all instructors in the Physics department.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Project - Exercise

- To find the set of all courses taught in the Fall 2009 semester
- To find the set of all courses taught in the Spring 2010 semester

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Rename

- Result of relational algebra operation can be given a name
- Denoted with small Greek letter **rho** ρ.
  - ρ_{x} (E)
  - where the result of expression **E** is saved with name of **x**

Set operations

- Union
- Intersection
- Difference

Union

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates ∪ Managers

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38
9297	O'Malley	56

Intersection

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates $\cap$ Managers

Number	Surname	Age
7432	O'Malley	39
9824	Darkes	38

Difference

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates - Managers

Number	Surname	Age
7274	Robinson	37

Cartesian product

- Combine information from any two relation.
- IF r1 and r2 are two relations, then cartesian product is given by, r1 x r2
- If same attribute appears in both relation, a naming schema to differentiate both attributes must be adopted.
- *If* $n1$ tuples in r1 and $n2$ tuples in *r2, then the cartesian product result will have* $n1 * n2$ tuples.

Instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

(*instructor*.*ID*, *instructor*.*name*, *instructor*.*dept name*, *instructor*.*salary*
teaches.*ID*, *teaches*.*course id*, *teaches*.*sec id*, *teaches*.*semester*,
teaches.*year*)

Instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

(*instructor*.*ID*, *name*, *dept name*, *salary,*
teaches.*ID*, *course id*, *sec id*, *semester*, *year*)

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000
MCA	A-Block	650000

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Bloc	101	50

Result of (Department x Classroom)

dept_name	Department.Building	budget	Classroom.Building	Room_no	capacity
CSE	D-Block	95000	D-Block	101	80
CSE	D-Block	95000	D-Block	103	40
CSE	D-Block	95000	A-Block	201	60
CSE	D-Block	95000	A-Block	101	50
IT	D-Block	80000	D-Block	101	80
IT	D-Block	80000	D-Block	103	40
IT	D-Block	80000	A-Block	201	60
IT	D-Block	80000	A-Block	101	50
MCA	A-Block	650000	D-Block	101	80
MCA	A-Block	650000	D-Block	103	40
MCA	A-Block	650000	A-Block	201	60
MCA	A-Block	650000	A-Block	101	50

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000
MCA	A-Block	650000

Result of $\sigma_{dept\_name="CSE"}(Department\ x\ Classroom)$

dept_name	Department.Building	budget	Classroom.Building	Room_no	capacity
CSE	D-Block	95000	D-Block	101	80
CSE	D-Block	95000	D-Block	103	40
CSE	D-Block	95000	A-Block	201	60
CSE	D-Block	95000	A-Block	101	50

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Block	101	50

$$\sigma_{Department.Building=Classroom.Building}(\sigma_{dept_{name}=\text{CSE}}(Department\ x\ Classroom$$

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000
MCA	A-Block	650000

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Bloc	101	50

dept_name	Department.Building	budget	Classroom.Building	Room_no	capacity
CSE	D-Block	95000	D-Block	101	80
CSE	D-Block	95000	D-Block	103	40

$$\pi_{dept\_name,Room\_no}(\sigma_{Department.Building=Classroom.Building}(\sigma_{dept_{name}=\text{CSE}}(Department\ x\ Classroom)))$$

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000
MCA	A-Block	650000

dept_name	Room_no
CSE	101
CSE	103

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Bloc	101	50

Natural Join

- Allows us to combine certain selections and a Cartesian product into one operation.

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000
MCA	A-Block	650000

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Block	101	50

Result of (Department **JOIN** Classroom)

Building	dept_name	budget	Room_no	capacity
D-Block	CSE	95000	101	80
D-Block	CSE	95000	103	40
D-Block	IT	80000	101	80
D-Block	IT	80000	103	40
A-Block	MCA	650000	201	60
A-Block	MCA	650000	101	50

$$\pi_{dept\_name,room\_no}(Department\ \boldsymbol{JOIN}\ Classroom)$$

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000
MCA	A-Block	650000

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Block	101	50

dept_name	Room_no
CSE	101
CSE	103
IT	101
IT	103
MCA	201
MCA	101

Outer Join

- Left outer join
- Right outer join
- Full outer join

Left outer join

- Left outer join: $r ⟕ s$
  - If a tuple $t_r \in r$ doesn't match any tuple in s, result contains $\{ t_r, null, \ldots, null \}$
  - If a tuple $t_s \in s$ doesn't match any tuple in r, it's excluded

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Block	101	50

Result of (Department **Left outer join** Classroom)

Building	dept_name	budget	Room_no	capacity
D-Block	CSE	95000	101	80
D-Block	CSE	95000	103	40
D-Block	CSE	95000	null	null
D-Block	CSE	95000	null	null
D-Block	IT	80000	101	80
D-Block	IT	80000	103	40
D-Block	IT	80000	null	null
D-Block	IT	80000	null	null

Right outer join

- Right outer join: $r ⟖ s$
  - If a tuple $t_r \in r$ doesn't match any tuple in s, it's excluded
  - If a tuple $t_s \in s$ doesn't match any tuple in r, result contains $\{ null, \ldots, null, t_s \}$

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Block	101	50

Result of (Department **right outer join** Classroom)

Building	dept_name	budget	Room_no	capacity
D-Block	CSE	95000	101	80
D-Block	CSE	95000	103	40
A-Block	null	null	201	60
A-Block	null	null	101	50
D-Block	IT	80000	101	80
D-Block	IT	80000	103	40
A-Block	null	null	201	60
A-Block	null	null	101	50

Full outer join

Full outer join: $r ⟗ s$

- Includes tuples from *r* that don't match *s*, as well as tuples from *s* that don't match *r*

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Block	101	50

Result of (Department **full outer join** Classroom)

dept_name	Department.Building	budget	Classroom.Building	Room_no	capacity
CSE	D-Block	95000	D-Block	101	80
CSE	D-Block	95000	D-Block	103	40
null	null	null	A-Block	201	60
null	null	null	A-Block	101	50
IT	D-Block	80000	D-Block	101	80
IT	D-Block	80000	D-Block	103	40
null	null	null	A-Block	201	60
null	null	null	A-Block	101	50
CSE	D-Block	95000	null	null	null
CSE	D-Block	95000	null	null	null
IT	D-Block	80000	null	null	null
IT	D-Block	80000	null	null	null

Outer Joins - Summary

$r =$

attr1	attr2
a	r1
b	r2
c	r3

$s =$

attr1	attr3
b	s2
c	s3
d	s4

$r \bowtie s$

attr1	attr2	attr3
b	r2	s2
c	r3	s3

$r ⟕ s$

attr1	attr2	attr3
a	r1	*null*
b	r2	s2
c	r3	s3

$r ⟖ s$

attr1	attr2	attr3
b	r2	s2
c	r3	s3
d	*null*	s4

$r ⟗ s$

attr1	attr2	attr3
a	r1	*null*
b	r2	s2
c	r3	s3
d	*null*	s4

Assignment

- temp ← r1 x r2
- Relation will not be displayed; It will be assigned to a new variable.

Department

dept_name	Building	budget
CSE	D-Block	95000
IT	D-Block	80000
MCA	A-Block	650000

Classroom

Building	Room_no	capacity
D-Block	101	80
D-Block	103	40
A-Block	201	60
A-Block	101	50

Temp ← (Department x Classroom)

dept_name	Department.Building	budget	Classroom.Building	Room_no	capacity
CSE	D-Block	95000	D-Block	101	80
CSE	D-Block	95000	D-Block	103	40
CSE	D-Block	95000	A-Block	201	60
CSE	D-Block	95000	A-Block	101	50
IT	D-Block	80000	D-Block	101	80
IT	D-Block	80000	D-Block	103	40
IT	D-Block	80000	A-Block	201	60
IT	D-Block	80000	A-Block	101	50
MCA	A-Block	650000	D-Block	101	80
MCA	A-Block	650000	D-Block	103	40
MCA	A-Block	650000	A-Block	201	60
MCA	A-Block	650000	A-Block	101	50

Aggregate Functions

- Very useful to apply a function to a collection of values to generate a single result
- Most common aggregate functions:

sum	sums the values in the collection
avg	computes average of values in the collection
count	counts number of elements in the collection
min	returns minimum value in the collection
max	returns maximum value in the collection

- Aggregate functions work on <u>multisets</u>, not sets
  - A value can appear in the input multiple times

Aggregate functions - Example

"Find the total amount owed to the credit company."

$\mathcal{G}_{\textbf{sum}(balance)}(credit\_acct)$

4275

cred_id	limit	balance
C-273	2500	150
C-291	750	600
C-304	15000	3500
C-313	300	25

credit_acct

"Find the maximum available credit of any account."

$\mathcal{G}_{\textbf{max}(available\_credit)}(\Pi_{(limit\ -\ balance)\ \textbf{as}\ available\_credit}(credit\_acct))$

11500

Grouping and Aggregation

- Sometimes need to compute aggregates on a *per-item* basis
- Back to the puzzle database:

  puzzle_list(*puzzle_name*)

  completed(*person_name*, *puzzle_name*)

puzzle_name
altekruse
soma cube
puzzle box

puzzle_list

- Examples:
  - How many puzzles has *each person* completed?
  - How many people have completed each puzzle?

person_name	puzzle_name
Alex	altekruse
Alex	soma cube
Bob	puzzle box
Carl	altekruse
Bob	soma cube
Carl	puzzle box
Alex	puzzle box
Carl	soma cube

completed

Grouping and Aggregation

puzzle_name
altekruse
soma cube
puzzle box

puzzle_list

person_name	puzzle_name
Alex	altekruse
Alex	soma cube
Bob	puzzle box
Carl	altekruse
Bob	soma cube
Carl	puzzle box
Alex	puzzle box
Carl	soma cube

completed

"How many puzzles has each person completed?"

$$_{person\_name}\mathcal{G}_{\mathbf{count}(puzzle\_name)}(completed)$$

- First, input relation *completed* is grouped by unique values of *person_name*
- Then, **count**(*puzzle_name*) is applied separately to each group

Grouping and Aggregation

$$_{person\_name}\mathcal{G}_{\textbf{count}(puzzle\_name)}(completed)$$

Input relation is grouped by *person_name*

person_name	puzzle_name
Alex	altekruse
Alex	soma cube
Alex	puzzle box
Bob	puzzle box
Bob	soma cube
Carl	altekruse
Carl	puzzle box
Carl	soma cube

Aggregate function is applied to each group

person_name	
Alex	3
Bob	2
Carl	3

Distinct values

- Sometimes want to compute aggregates over sets of values, instead of multisets

Example:

  - Chage puzzle database to include a *completed_times* relation, which records multiple solutions of a puzzle

- How many puzzles has each person completed?
  - Using *completed_times* relation this time

person_name	puzzle_name	seconds
Alex	altekruse	350
Alex	soma cube	45
Bob	puzzle box	240
Carl	altekruse	285
Bob	puzzle box	215
Alex	altekruse	290

completed_times

Distinct values

"How many puzzles has each person completed?"

- Each puzzle appears multiple times now.

person_name	puzzle_name	seconds
Alex	altekruse	350
Alex	soma cube	45
Bob	puzzle box	240
Carl	altekruse	285
Bob	puzzle box	215
Alex	altekruse	290

completed_times

- Need to count <u>distinct</u> occurrences of each puzzle's name

$${}_{person\_name}\mathcal{G}_{\textbf{count-distinct}(puzzle\_name)}(completed\_times)$$