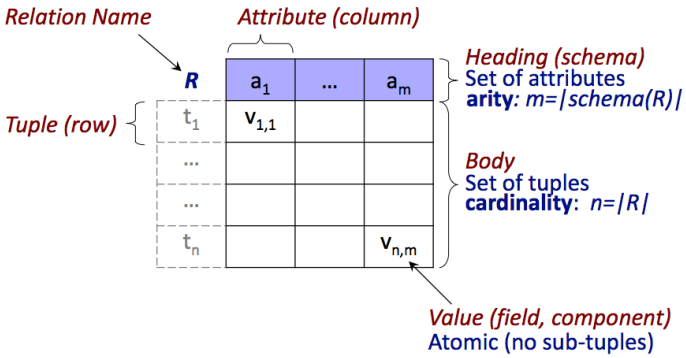
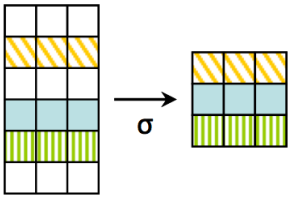
RELATIONAL ALGEBRA

- Relations and Tuples



- Relational algebra summary:
  - Values
    - Finite relations (cardinality & arity both bounded)
    - Attributes may/may not be types
  - Operators
    - Unary operators: σ, π, ρ
    - Set operators: ∪, ∩, −
    - Join operators: ×, ⋈
  - Expressions → Queries
- Unary operators: **Select** σ
  - $\sigma_P(R)$ → gives tuples of relation R which satisfy condition P
    - i.e. remove unwanted rows from relation
  - The schema of R remains the same



  - ex.

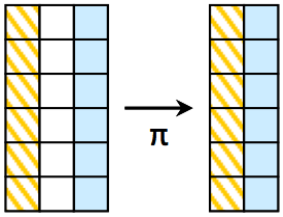
Employees

Surname	FirstName	Age	Salary
Smith	Mary	25	2000
Black	Lucy	40	3000
Verdi	Nico	36	4500
Smith	Mark	40	3900

$\sigma_{Age<30\ V\ Salary>4000}$ **(Employees)**

Surname	FirstName	Age	Salary
Smith	Mary	25	2000
Verdi	Nico	36	4500

- Unary operators: **Project** π
  - $\pi_Y(R)$ → gives subset Y of the set of attributes X of relation R
    - i.e. removes unwanted columns from relation



  - ex.

Employees

Surname	FirstName	Department	Head
Smith	Mary	Sales	De Rossi
Black	Lucy	Sales	De Rossi
Verdi	Mary	Personnel	Fox
Smith	Mark	Personnel	Fox

$\pi_{Surname,\ FirstName}$ **(Employees)**

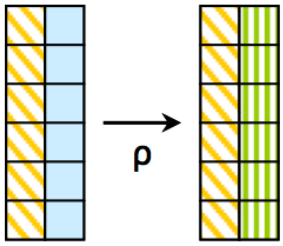
Surname	FirstName
Smith	Mary
Black	Lucy
Verdi	Mary
Smith	Mark

$\pi_{Department,\ Head}$ **(Employees)**

Department	Head
Sales	De Rossi
Personnel	Fox

Unary operators: Rename ρ

- Unary operators: Rename ρ
  - $\rho_S(R)$ → renames attributes of relation R to according to expression S
    - i.e. Modifies schema only, same values
    - ex. $\rho_{A=X,C=Y}(R)$ or $\rho_{A,C\to X,Y}(R)$ renames attributes A → X, C → Y



  - ex.

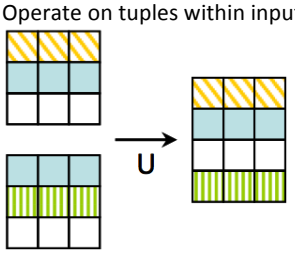
Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

$\rho_{Father\to Parent}$**(Paternity)**

Parent	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

- Set operators:
  - Standard set operators
  - Operate on tuples within input relations, but not on schema



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

  - **Union** ∪

Graduates ∪ Managers

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

  - **Intersect** ∩

Graduates ∩ Managers

Number	Surname	Age
7432	O'Malley	39
9824	Darkes	38

  - **Difference** −
    - If Graduates is a subset of Managers then the result = empty

Graduates - Managers

Number	Surname	Age
7274	Robinson	37

  - Union with renaming

Employees

Surname	Branch	Salary
Patterson	Rome	45
Trumble	London	53

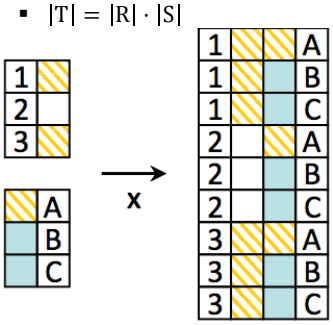
Staff

Surname	Factory	Wages
Cooke	Chicago	33
Bush	Monza	32

$\rho_{Branch,Salary\to Location,Pay}$**(Employees)** ∪ $\rho_{Factory,\ Wages\to Location,Pay}$**(Staff)**

Surname	Location	Pay
Patterson	Rome	45
Trumble	London	53
Cooke	Chicago	33
Bush	Monza	32

- Join
  - Mose used operator in relational algebra
  - Used to establish connections among data in different relations, taking advantage of "value-based" nature of relational model
  - Main versions of join: Natural join, Theta join
- Join operator: **Cartesian Product** ×
  - $T = R \times S \rightarrow$ `T` combines every tuple in `R` w/ every tuple in `S`
    - $\text{schema}_T = \text{schema}_R \cup \text{schema}_S$
    - $|T| = |R| \cdot |S|$



  - Note: the two schemas cannot have any overlap
    - Can't have a columns w/ the same name in both schemas
  - Ex.

Employees

Employee	Project
Smith	A
Black	A
Black	B

Projects

Code	Name
A	Venus
B	Mars

Employees × Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	A	Venus
Smith	A	B	Mars
Black	A	B	Mars
Black	B	B	Mars

- **Division** /
  - Let relation `R`(A_1, …, A_n, B_1, …, B_n) and `S`(B_1, …, B_n)
    - $T = R / S \rightarrow$ `T` = (A_1, …, A_n) i.e. attrbutes names in `R`, but not in `S`
      - For tuples t ∈ T such that, for every tuple s ∈ S, the tuple t ∪ s (the concatenation of t and s) is in relation `R`
      - `T` contains the largest possible set of tuples s.t. R × T ⊆ R
  - Ex.

R

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c1
a1	b2	c2
a2	b1	c2
a1	b2	c3
a1	b2	c4
a1	b1	c5

S1

C
c1

T1=R/S1

A	B
a1	b1
a2	b1
a1	b2

S2

C
c1
c2

T2=R/S2

A	B
a1	b2
a2	b1

S3

C
c1
c2
c3
c4

T3=R/S3

A	B
a1	b2

S4

B	C
b1	c1

T4=R/S4

A
a1
a2

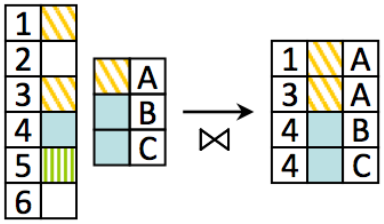
S5

B	C
b1	c1
b2	c1

T5=R/S5

A
a1

  - Division in RA: $A / B = \pi(A) - \pi(\sigma(\pi(A) \times B) - A)$
    - Relations:
      - `Take`(x, y) – "student x has taken course y"
      - `CS`(z) – "z is a CS course"
    - Query: all students who have taken all CS courses
      - 1. Relation of all student, CS course pairs:
        $\pi_x(\text{Take}) \times \text{CS}$
      - 2. Relation of all student and the CS courses they have not taken
        $(\pi_x(\text{Take}) \times \text{CS}) - \text{Take}$
      - 3. Relation of all students who have not taken all CS courses
        $\pi_x\big((\pi_x(\text{Take}) \times \text{CS}) - \text{Take}\big)$
      - 4. Relation of all students who have taken all CS courses
        $\pi_x(\text{Take}) - \pi_x\big((\pi_x(\text{Take}) \times \text{CS}) - \text{Take}\big) = \text{Take} / \text{CS}$
- Join operators: **Natural Join** ⋈
  - $T = R \bowtie S \rightarrow$ merges tuples from `R` and `S` having equal values where their schemas overlap (join attributes)
    - $R \bowtie S = \pi\big(\sigma(R \times \rho(S))\big)$
    - `T` Schema: Union of schemas
    - $|T| \leq |R| \cdot |S|$, usually $\max(|R|, |S|)$



  - Note: $\text{schema}_R \cap \text{schema}_S \neq \emptyset$
  - Ex.

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
sales	Brown

$r_1 \bowtie r_2$

Employee	Department	Head
Smith	sales	Brown
Black	production	Mori
White	production	Mori

  - Properties of Natural Join
    - Commutative: $R \bowtie S = S \bowtie R$
    - Associative: $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$
    - N-ary join w/o ambiguity: $R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$
  - N-ary Join Operation ex.

r_1

Employee	Department
Smith	sales
Black	production
Brown	marketing
White	production

r_2

Department	Division
production	A
marketing	B
purchasing	B

r_3

Division	Head
A	Mori
B	Brown

$r_1 \bowtie r_2 \bowtie r_3$

Employee	Department	Division	Head
Black	production	A	Mori
Brown	marketing	B	Brown
White	production	A	Mori

- Join operators: **Theta Join** $\bowtie_C$
  - $T = R \bowtie_C S \rightarrow$ Gives pairwise combinations of tuples which satisfy C
    - $R \bowtie_C S = \sigma_c(R \times \rho(S))$ –i.e. Cartesian Product, then Select
    - $|T| \leq |R| \cdot |S|$
  - Ex.

Car

Car	CarPrice
CarA	20000
CarB	30000
CarC	50000

Boat

Boat	BoatPrice
BoatA	10000
BoatB	40000
BoatC	60000

Car $\bowtie_{CarPrice>BoatPrice}$ Boat

Car	CarPrice	Boat	BoatPrice
CarA	20000	BoatA	10000
CarB	30000	BoatA	10000
CarC	50000	BoatA	10000
CarC	50000	BoatB	40000

- Join operators: **Equijoin**
  - Special case of theta join
  - $R \bowtie_{A=X,B=Y,\ldots} S$
    - Attribute names in `R` and `S` can differ
    - Still compare values for equality
  - Ex.

Employees

Employee	Project
Smith	A
Black	A
Black	B

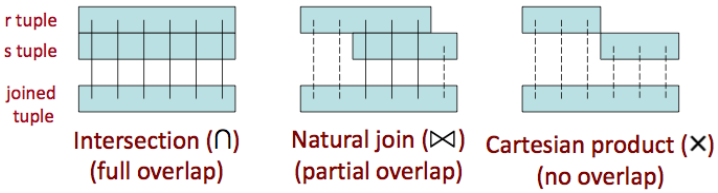
Projects

Code	Name
A	Venus
B	Mars

Employees $\bowtie_{Project=Code}$ Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	B	Mars

- Comparison: ∩ vs. ⋈ vs. ×
  - Same general operation
    - Test "overlapping" parts of tuples for equality
    - Combine "matching" pairs, ignore others
  - Differ in degree of schema overlap



  - Mathematical power vs. efficiency
    - Note: × expresses both ∩ and ⋈ → mathematically, intersection and joins are unnecessary
  - Performance
    - Efficient algorithms compute result directly
      - Ex. $|R| \cdot |S|$ rows vs. $\min(|R|, |S|)$
- Summary of Operators

Operation	Name	Symbol
Choose Rows	Select	σ
Choose Columns	Project	π
Rename Relation/Attribute	Rename	ρ
Combine Tables	Natural Join	⋈
	Theta Join	$\bowtie_{condition}$
	Cartesian Product	×
Sept operations	Intersection	∩
	Union	∪
	Subtraction	−
Assignment	Assignment	:=

- Expressing Integrity Constraints
  - Suppose R and S are expression in RA
    - $R = \emptyset$ or $R \subseteq \emptyset \rightarrow$ express the fact that R is an empty set
    - $R \subseteq S$ or $R - S = \emptyset \rightarrow$ express the R is a subset of S
  - Ex.
    - Given relations:
      - `Course`(<u>`Dept`</u>, <u>`CourseNum`</u>, `Title`, `Credits`)
      - `Section`(<u>`CRN`</u>, `Dept`, `CourseNum`, `Room`, `Time`, `InstructorID`)
    - Referential integrity constraints → `Dept`, `CourseNum` of `Course` is a foregin key in `Section`
      $\pi_{Dept,CourseNum}(\text{Section}) \subseteq \pi_{Dept,CourseNum}(\text{Course})$
    - Key Constraints → two tuples which agree on `CRN` must also agree on `Dept`, `CourseNum`, `Room`, `InstructorID`
      $\rho_{S1}(\text{Section}) \bowtie_{S1.CRN=S2.CRN \text{ and } S1.Dept!=S2.Dept} \rho_{S2}(\text{Section}) = \emptyset$
    - Domain Constraints -> CourseNum must be in range 100 – 999
      $\sigma_{CourseNum<100 \text{ or } CourseNum>999}(\text{Course}) \neq \emptyset$

RELATIONAL ALGEBRA EXAMPLES (Set-Semantics)

- Sample Data A
  - Database Schemas
    Employees(Number, Name, Age, Salary)
    Supervision(Head, Emp)
  - Integrity Constraints
    $\pi_{Head}(Supervision) \subseteq \pi_{Number}(Employees)$
    $\pi_{Emp}(Supervision) \subseteq \pi_{Number}(Employees)$

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Emp
210	101
210	103
210	104
231	105
301	210
301	231
375	252

  - **Query 1**: Find the number, and names and ages of employees earning more than 40K
    $\pi_{Number,Name,Age}(\sigma_{Salary>40}Employees)$

Number	Name	Age
104	Luigi Neri	38
210	Marco Celli	49
231	Siro Bisi	50
252	Nico Bini	44
301	Steve Smith	34
375	Mary Smith	50

  - **Query 2**: Find the registration numbers of the supervisors of the employees earning more than 40k
    $\pi_{Head}\left(Supervision \bowtie_{Emp=Number} (\sigma_{Salary>40}Employees)\right)$

Head
210
301
375

  - **Query 3**: Find the names and salaries of the supervisors of the employees earning more than 40k
    $SuperOfMoreThan40(Head) := \pi_{Head}\left(Supervision \bowtie_{Emp=Number} (\sigma_{Salary>40}Employees)\right)$
    $\pi_{Name,Salary}(Employees \bowtie_{Number=Head} SuperOfMoreThan40)$

NameH	SalaryH
Marco Celli	60
Steve Smith	70
Mary Smith	65

  - **Query 4**: Find the employees earning more than their respective supervisors; return registration numbers, names and salaries of the employees and their supervisors
    $Employees(Head, Emp, Number.Name, Age, Salary) := Supervision \bowtie_{Emp=Number} Employees$
    $Supers(HNum, HName, HSal, HAge) := \rho_{Number,Name,Salary,Age \rightarrow HNum,HName,HSal,HAge} Supervision$
    $EmployeeSuperPairs(HNum, HName, HSal, HAge, Head, Emp, Number.Name, Age, Salary) := Supers \bowtie_{HNum=Head} Employees$
    $\pi_{Number,Name,Salary,HNum,HName,Hsal}\left(\sigma_{Salary>HSal}(EmployeeSuperPairs)\right)$

Number	Name	Salary	NumH	NameH	SalH
104	Luigi Neri	61	210	Marco Celli	60
252	Nico Bini	70	375	Mary Smith	65

  - **Query 5**: Find registration numbers and names of supervisors, all of whose employees earn more than 40k
    $SuperLessThan40(Head) := \pi_{Head}\left(Supervision \bowtie_{Emp=Number} (\sigma_{Salary\leq 40}Employees)\right)$
    $SuperAllMoreThan40(Head) := \pi_{Head}(Supervision) - SuperLessThan40$
    $\pi_{Number,Name}(Employees \bowtie_{Number=Head} (SuperAllMoreThan40))$

Number	Name
301	Steve Smith
375	Mary Smith

  - **Query 6**: Find registration numbers of supervisors, who supervise all employees earning more than 40k
    $MoreThan40 := \pi_{Number}(\sigma_{Salary>40}Employees)$
    $\pi_{Number}(Supervision / MoreThan40)$
  - **Query 7**: Find the employees earning maximum salary
    $E1 := \sigma_{Number,Salary}(Employees)$
    $E2 := \sigma_{Number,Salary}(Employees)$
    $EmployeePairs(E1.Number, E1.Salary, E2.Number, E2.Salary) := E1 \bowtie_{E1.Number!=E2.Number} E2$
    $NotHighestEmployee(Number) := \rho_{E1.Number}\left(\sigma_{E1.Salary<E2.Salary}(EmployeePairs)\right)$
    $\rho_{Number}(Employees) - NotHighestEmployee$
  - **Query 8**: Find all locations that have at least two employees earning more than 40k
    - Suppose the schema is now:
      - Employees(Num, Name, Loc, Salary)
      - Supervision(Head, Emp)
    $MoreThan40(Num, Name, Loc, Salary) := \pi_{Num}(\sigma_{Salary>40}Employees)$
    $40Pairs(Num, Name, Loc, Salary, Num1, Name1, Loc1, Sal1) := MoreThan40 \times \rho_{Num,Name,Loc,Salary \rightarrow Num1,Name1,Loc1,Sal1}(MoreThan40)$
    $\pi_{Loc}\left(\sigma_{Loc=Loc1\ AND\ Num!=Num1}(40Pairs)\right)$

- Sample Data B
  - Database Schemas
    Films(Film#, Title, Director, Year, ProdCost)
    Artists(Actor#, Surname, FirstName, Sex, Birthday, Nationality)
    Roles(Film#, Actor#, Character)
  - Integrity Constraints
    $\pi_{\text{Film\#}}(\text{Roles}) \subseteq \pi_{\text{Film\#}}(\text{Films})$
    $\pi_{\text{Actor\#}}(\text{Roles}) \subseteq \pi_{\text{Actor\#}}(\text{Artists})$
  - **Query 1**: Find the titles of films starring Henry Fonda
    $$\text{FondaID}(\text{Actor\#}) \coloneqq \sigma_{\substack{\text{FirstName="Henry"} \\ \text{AND Surname="Fonda"}}}(\text{EmployeeSuperPairs})$$
    $$\text{FondaFilmID}(\text{Film\#}) \coloneqq \pi_{\text{Film\#}}(\text{FondaID} \bowtie \text{Roles})$$
    $$\pi_{\text{Title}}(\text{Films} \bowtie \text{FondaFilmID})$$
  - **Query 2**: Find the titles of all films in which the director is also an actor
    $$\pi_{\text{Title}}(\sigma_{\text{DIrector=Actor\#}}\text{Film} \bowtie \text{Role})$$
  - **Query 3**: Find the actors who have played two characters in the same film; show the title of each such film, first name and surname of the actor and the two characters
    $$\text{R1}(\text{Film\#}, \text{Actor\#}, \text{Character}) \coloneqq \text{Roles}$$
    $$\text{R2}(\text{Film\#}, \text{Actor\#}, \text{Character}) \coloneqq \text{Roles}$$
    $$\text{SameActorCharacters}(\text{Actor\#}, \text{Film\#}, \text{Character1}, \text{Character2})$$
    $$\coloneqq \pi_{\substack{\text{R1.Actor\#,R1.Film\#,} \\ \text{R1.Character,R2.Character}}} \left(\sigma_{\substack{\text{R1.Film\#=R2.Film\# AND} \\ \text{R1.Actor\#=R2.Actor\# AND} \\ \text{R1.Character!=R2.Character}}} (\text{R1} \times \text{R2}) \right)$$
    $$\pi_{\substack{\text{FirstName,Surname,Title,} \\ \text{Character1,Character2}}}(\text{SameActorCharacters} \bowtie \text{Film} \bowtie \text{Artists})$$
  - **Query 4**: Find the titles of the films which the actors are all of the same sex
    $$\text{A}(\text{Film\#}, \text{Actor\#}, \text{Sex}) \coloneqq \pi_{\text{Film\#,Actor\#,Sex}}(\text{Roles} \bowtie \text{Artists})$$
    $$\text{B}(\text{Film\#}, \text{Actor\#}, \text{Sex}) \coloneqq \pi_{\text{Film\#,Actor\#,Sex}}(\text{Roles} \bowtie \text{Artists})$$
    $$\text{DifferentSexFilms}(\text{Film\#})$$
    $$\coloneqq \pi_{\text{A.Film}} \left(\sigma_{\substack{\text{A.Film\#=B.Film\# AND} \\ \text{A.Actor\#!=B.Actor\# AND} \\ \text{A.Sex!=B.Character}}} (\text{A} \times \text{B}) \right)$$
    $$\text{SameSexFilms}(\text{Film\#}) \coloneqq \pi_{\text{Film\#}}(\text{Films}) - \text{DifferentSexFilms}$$
    $$\pi_{\text{Title}}(\text{SameSexFilms} \bowtie \text{Films})$$

- Tips & Tricks for Relational Algebra Queries
  - Evaluating RA Queries
    - RA is procedural → an RA query itself suggests a procedure for constructing the result
      - i.e. how to implement the query
    - RA suggests a query execution plan
      - Many expressions would yield the same result
      - Alt. expressions suggest different query execution plans
      - Which is best depends on the data in the database, what indices you have defined, join ordering, etc.
    - In real DBMSs query optimization takes places
      - Optimizer rewrites queries in a more efficient form
  - Determine which relations are needed, ignore the rest
  - After combining relations, confirm
    - 1. Attributes that should match will be made to match
    - 2. Attributes that will be made to match should match
  - Determine if there is an intermediate relation that would help you get the final answer
    - Break down the answer by defining intermediate relations using assignment
    - Use good names for the new relation
    - Name the attributes on the Left-Hand-Side each time
    - Add a comment that explains exactly what the relation contains
  - Tips for Specific RA Queries
    - To show "max"
      - 1. Pair tuples (self-join) and find those that are not the max
      - 2. Subtract from all to find the max(es)
    - To show "k or more"
      - Make all combinations of k different tuples that meet the required condition
    - To show "exactly k"
      - Show "k or more"
      - Then subtract "k + 1 or more"
    - To show "every"
      - Make all combinations that could have occurred
      - Subtract those that did occur to find those that didn't always; these are the failures
      - Subtract the failures from all to get the answer

RELATION OPERATIONS ON BAGS

- Limitations of relational algebra
  - Relational algebra is set-based
  - Real-life applications need more
    - Expensive (and often unnecessary) to eliminate duplicates
    - Important (and often expensive) to order output
    - Need a way to apply scalar expressions to values
    - What's "not" there often as important as what is
  - Solution: non-set extensions
- RA Extension: bag semantics
  - In practice, relations are bags (multisets)
    - Members are allowed to appear more than once
    - Sometimes people purposefully insert duplicates
  - Ex. {1, 2, 1, 1, 3} is a bag
    - Note: it is still unordered
  - Most operators still work
    - Select, Rename → unchanged
    - Project → no longer eliminates duplicates
    - Set operations need tweaks
    - Joins tend to multiply the number of duplicates
  - Some laws no longer apply
- Set Operations
  - **Union** – concatenation, except unordered
    - Ex. $\{1,1,2,3\} \cup \{2,2,3,4\} = \{1,1,2,3,2,2,3,4\}$
  - **Intersection** – take minimum count of each value
    - Ex. $\{1,1,2,3\} \cap \{2,2,3,4\} = \{2,3\}$
  - **Difference** – Each occurrence on right can cancel on occurrence on left
    - Ex. $\{1,1,2,3\} - \{2,2,3,4\} = \{1\}$
  - Union, Intersection no longer distributive
    - Ex. $\{1\} \cap (\{1\} \cup \{1\})$ vs. $(\{1\} \cap \{1\}) \cup (\{1\} \cap \{1\})$
    - Ex. $\{1\} \cap \{1,1\}$ vs. $\{1\} \cup \{1\}$
    - Ex. $\{1\}$ vs. $\{1,1\}$

- Consider a relation R modeling cars for sale

Make	Model	Color
Toyota	Prius	Gray
Toyota	Prius	Red
Honda	Accord	Green
Honda	Accord	Red
Honda	Accord	Red
Ford	Echo	Red
Ford	Echo	Gray
Ford	Echo	White

  - **Bag-projection** π
    - Bag-projection does not eliminate duplicate tuples (as in set-projection)
      - Duplicates important for summaries (i.e. how many)
    - $\pi_{\text{Make}}(R)$ returns the column under attribute Make as-is
  - **Duplicate elimination** δ
    - δ turns a **bag** into a **set**
    - $\delta(\pi_{\text{Make}}(R))$ returns a set

- Summarizing groups of tuples

Student	Year	Dept	Course	Grade
Xiao	2009	CS	A08	B-
Xiao	2009	CS	A48	B
Xiao	2009	CS	A65	B+
Xiao	2009	Math	A23	B
Xiao	2009	Math	A30	B+
Xiao	2009	Math	A37	A
Xiao	2010	CS	B07	B
Xiao	2010	CS	B09	B-
Xiao	2010	CS	B36	B-
Xiao	2010	CS	B58	B
Xiao	2010	Math	B24	A-
Xiao	2010	Math	B41	B
Xiao	2010	Stats	B52	B-
Xiao	2011	CS	C24	B+
Xiao	2011	CS	C43	A-
Xiao	2011	CS	C69	A

All courses Xiao has taken

All courses Xiao took in 2010

All math courses Xiao took in 2010

Student	Year	Dept	Course	Grade
Xiao	2009	CS	A08	B-
Xiao	2009	CS	A48	B
Xiao	2009	CS	A65	B+
Xiao	2009	Math	A23	B
Xiao	2009	Math	A30	B+
Xiao	2009	Math	A37	A
Xiao	2010	CS	B07	B
Xiao	2010	CS	B09	B-
Xiao	2010	CS	B36	B-
Xiao	2010	CS	B58	B
Xiao	2010	Math	B24	A-
Xiao	2010	Math	B41	B
Xiao	2010	Stats	B52	B-
Xiao	2011	CS	C24	B+
Xiao	2011	CS	C43	A-
Xiao	2011	CS	C69	A

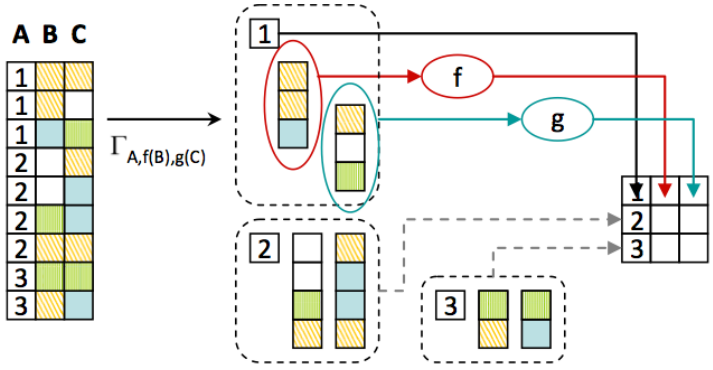
How to summarize this??

Student	Dept	Year	Course	Grade
Xiao	CS	2009	?	?
Xiao	Math	2009	?	?
Xiao	CS	2010	?	?
Xiao	Math	2010	?	?
Xiao	Stats	2010	B52	B-
Xiao	CS	2011	?	?

These columns are easy... equal for every tuple in a group

Show the best grade? Worst grade? Average?

  - Summarizing goals:
    - 1. Output a single tuple which summarizes a set of related tuples
    - 2. "Collapse" a set of tuples into a single "representative" tuple
  - **Grouping** Γ
    - Useful when computing statistics
    - **Grouping Key**: a subset of attributes to test for equality
      - To identify related tuples to collapse
    - **Aggregation function** (sum, count, avg, min, max, ...)
      - To collapse a column into a value
      - Aggregation function should be commutative: f(x,y) = f(y,x)
    - Syntax: $\Gamma_{A,B,C,f(x),g(y),h(z)}(R)$
      - Implicit projection → drops unreferenced attributes
      - A, B, C is the grouping key
      - x, y, z are attributes to aggregate
      - f, g, h are aggregating functions to apply
    - All tuples having the same key go to same group
      - One output tuple for each unique key
      - Output "group total" for each non-key attribute in group
    - Ex. given schema
      - A = Employee level → 1 = Employee; 2 = Manager, 3 = Executive
      - B = Age → f(B) gives average Age
      - C = Salary →g(C) gives average Salary
      - $\Gamma_{A,f(B),g(C)}(R)$



  - Duplicates and grouping
    - Consider a relation R modeling cars for sale
    - Ex. $\Gamma_{\text{Make},\text{count}(*)}(R)$ returns the number of cars of each Make

Make	Model	Color
Toyota	Prius	Gray
Toyota	Prius	Red
Honda	Accord	Green
Honda	Accord	Red
Honda	Accord	Red
Ford	Echo	Red
Ford	Echo	Gray
Ford	Echo	White

Make	Count
Toyota	2
Honda	3
Ford	3

- **Sorting** τ
  - $\tau_L(R)$ sorts tuples in R on list of attributes L
    - if L is A1, A2, …, An tuples sorted first by A1
      - Ties are broken based on A2, etc.
      - Ties that remain after An are broken arbitrarily
    - Default: ascending order
      - With "–" in front → descending order
  - Example: $\tau_{-Count,Make}(R)$
    - Descending count
    - Alphabetical order when count is equal

Make	Count
Toyota	2
Honda	3
Ford	3

τ

Make	Count
Ford	3
Honda	3
Toyota	2

  - Sorted relations not in the RA value domain
- Dangling tuple problem
  - Consider the following query: List employees and their total sales in descending order

$$\tau_{-Total}\left(\rho_{Name,Total}\left(\Gamma_{Name,sum(Value)}(Emp \bowtie Sales)\right)\right)$$

Emp

EID	Name
1	Mary
2	Xiao
3	Jaspreet

$\bowtie$

Sales

EID	Value	…
1	20	…
3	10	…
3	15	…

=

Name	Total
Jaspreet	25
Mary	20

Xiao?

  - What's not there can be very important
    - Join hides fact that Xiao has no sales
    - Want: rewrite query to include Xiao's zero
- **Outer Join**
  - All joins so far resemble intersection → tuples w/ no match discarded
    - Sometimes desirable to output dangling tuples
      - Ex. List all employees and their sales total (even if zero)
    - Problem: What to do use as missing half of tuple
  - $T = R \bowtie_{PAD} S$ computes the "outer" join of R (left) and S (right)
    - Like normal join, but all tuples from R and S appear in output
    - PAD = LEFT/RIGHT/FULL dangling tuples with special value $\perp$ (**Null**)
      - Pad dangling tuples as needed to match schema
      - Note: $\perp$ is technically outside RA value domain
      - **LEFT** – tuples in inner join padded w/ tuples in R that have no matching tuples in S
      - **RIGHT** – tuples in inner join padded w/ tuples in S that have no matching tuples in R
      - **FULL** – tuples in inner join padded w/ tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R
    - Natural Join, equijoin, Thetajoin variants apply
    - $|T| \geq \max(|R|, |S|)$
  - Ex1. given following relations

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
purchasing	Brown

    - $r_1 \bowtie_{LEFT} r_2$

Employee	Department	Head
Smith	sales	NULL
Black	production	Mori
White	production	Mori

    - $r_1 \bowtie_{RIGHT} r_2$

Employee	Department	Head
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

    - $r_1 \bowtie_{FULL} r_2$

Employee	Department	Head
Smith	Sales	NULL
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

  - Ex2. Consider the query: List the employees and their total sales in descending order

$$\tau_{-Total}\left(\rho_{Name,Total}\left(\Gamma_{Name,sum(Value)}(Emp \bowtie_{LEFT} Sales)\right)\right)$$

Emp

EID	Name
1	Mary
2	Xiao
3	Jaspreet

$\bowtie$

Sales

EID	Value	…
1	20	…
3	10	…
3	15	…

=

Name	Total
Jaspreet	25
Mary	20
Xiao	$\perp$

- Extended projection
  - $\pi_{x=E}(R)$ computes column x from expression E
    - Arithmetic (z = 3*x + y)
    - String manipulation (substring, capitalization)
    - Some conditional expressions
  - Example:

$$\tau_{-Total}\left(\pi_{Name,Total=T?T:0}\left(\rho_{Name,t}\left(\Gamma_{Name,sum(Value)}(Emp \bowtie_{LEFT} Sales)\right)\right)\right)$$

Emp

EID	Name
1	Mary
2	Xiao
3	Jaspreet

$\bowtie$

Sales

EID	Value	…
1	20	…
3	10	…
3	15	…

=

Name	Total
Jaspreet	25
Mary	20
Xiao	0