CS848 Fall 2025: Algorithmic Aspects of Query Processing

CS 348: Supplementary Materials on Relational Algebra

Relational data model

Modeling data as **relations** or **tables**, each storing logically related information together

User

uid	*name*	*age*	*pop*
142	Bart	10	0.9
123	Milhouse	10	0.2
857	Lisa	8	0.7
456	Ralph	8	0.3
…	…	…	…

Group

gid	*name*
abc	A Book Club
gov	Student Government
dps	Dead Putting Society
…	…

Member

uid	*gid*
142	dps
123	gov
857	abc
857	gov
456	abc
456	gov
…	…

relations (or tables)

Attributes

User

uid	*name*	*age*	*pop*
142	Bart	10	0.9
123	Milhouse	10	0.2
857	Lisa	8	0.7
456	Ralph	8	0.3
…	…	…	…

Group

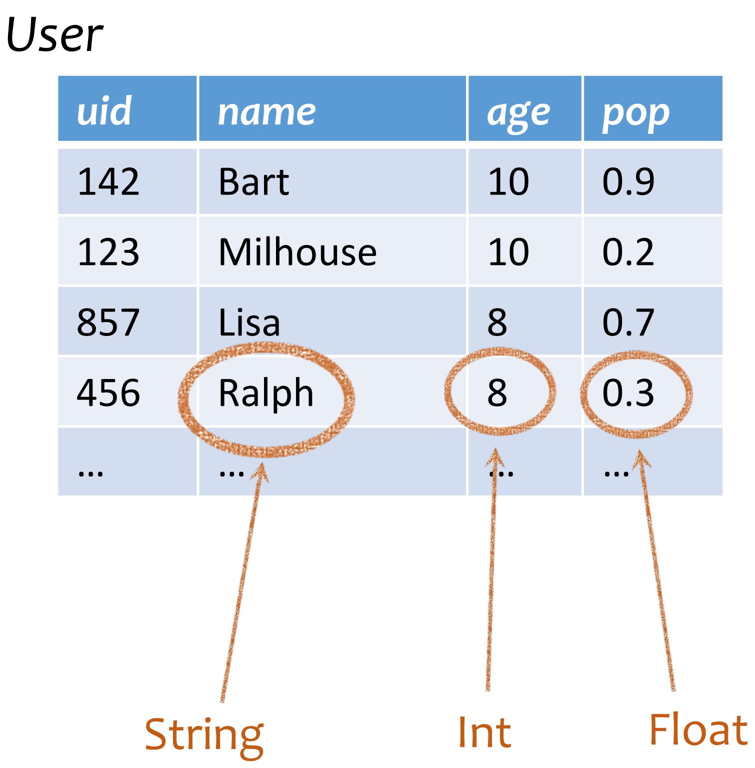
gid	*name*
abc	A Book Club
gov	Student Government
dps	Dead Putting Society
…	…

Member

uid	*gid*
142	dps
123	gov
857	abc
857	gov
456	abc
456	gov
…	…

attributes (or columns)

Domain



User

uid	*name*	*age*	*pop*
142	Bart	10	0.9
123	Milhouse	10	0.2
857	Lisa	8	0.7
456	Ralph	8	0.3
…	…	…	…

String

Int

Float

Group

gid	*name*
abc	A Book Club
gov	Student Government
dps	Dead Putting Society
…	…

Member

uid	*gid*
142	dps
123	gov
857	abc
857	gov
456	abc
456	gov
…	…

domain (or type)

Tuples

User

uid	*name*	*age*	*pop*
142	Bart	10	0.9
123	Milhouse	10	0.2
857	Lisa	8	0.7
456	Ralph	8	0.3
…	…	…	…

tuples (or rows)

Duplicates (all attr. have same val) are not allowed

Ordering of rows doesn't matter
(even though output can be ordered)

Group

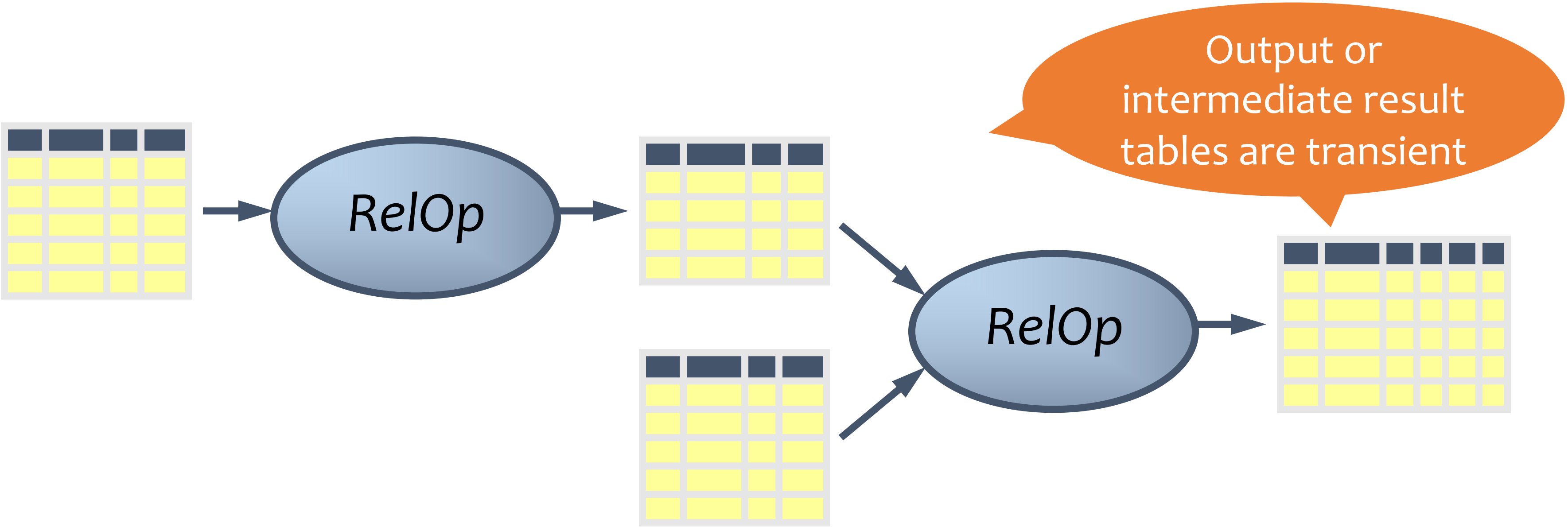
gid	*name*
abc	A Book Club
gov	Student Government
dps	Dead Putting Society
…	…

Member

uid	*gid*
142	dps
123	gov
857	abc
857	gov
456	abc
456	gov
…	…

Relational algebra

- A language for querying relational data based on "operators"
- Not used in commercial DBMSs (SQL)



Relational algebra

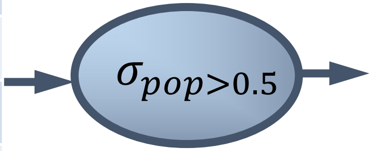
- Core operators:
  - Selection
  - Projection
  - Cross product
  - Union
  - Difference
- Additional, derived operators:
  - Join, Natural join, Intersection, etc.

Core operator 1: Selection σ

- Example query: Users with popularity higher than 0.5

$$\sigma_{pop>0.5}User$$

uid	*name*	*age*	*pop*
142	Bart	10	0.9
123	Milhouse	10	0.2
857	Lisa	8	0.7
456	Ralph	8	0.3
…	…	…	…



uid	*name*	*age*	*pop*
142	Bart	10	0.9
857	Lisa	8	0.7
…	…	…	…

Core operator 1: Selection

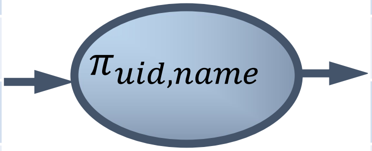
- Input: a table R
- Notation: $\sigma_p R$
    - p is called a selection condition (or predicate)
- Purpose: filter rows according to some criteria
- Output: same columns as R, but only rows of R that satisfy p
- Selection condition can include any column of R, constants, comparison ($=, \neq, <, \leq, >, \geq$ etc.) and Boolean connectives ($\wedge$: and, $\vee$: or, $\neg$: not)

Core operator 2: Projection π

- Example: IDs and names of all users

$$\pi_{uid,name}\ User$$

uid	*name*	*age*	*pop*
142	Bart	10	0.9
123	Milhouse	10	0.2
857	Lisa	8	0.7
456	Ralph	8	0.3
…	…	…	…



uid	*name*
142	Bart
123	Milhouse
857	Lisa
456	Ralph
…	…

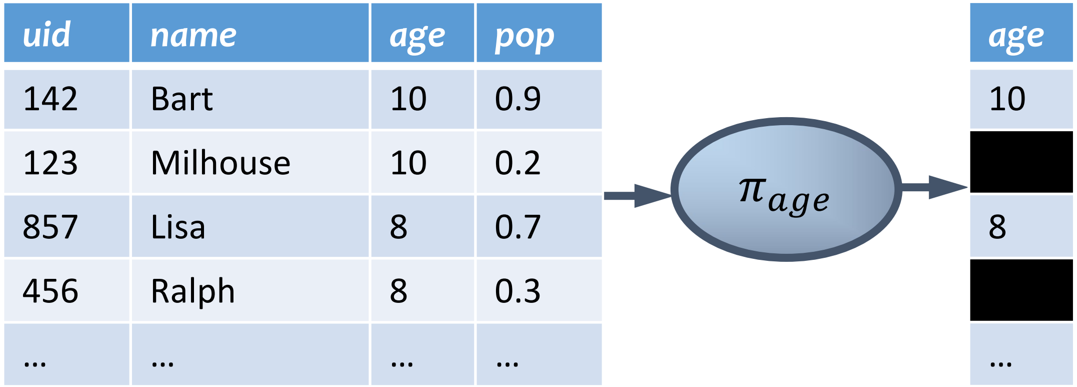
Core operator 2: Projection

- Input: a table R
- Notation: $\pi_L R$
  - L is a list of columns in R
- Purpose: output chosen columns
- Output: “same” rows, but only the columns in L

More on projection

- Duplicate output rows are removed (by definition)
  - Example: user ages

$$\pi_{age}\, User$$

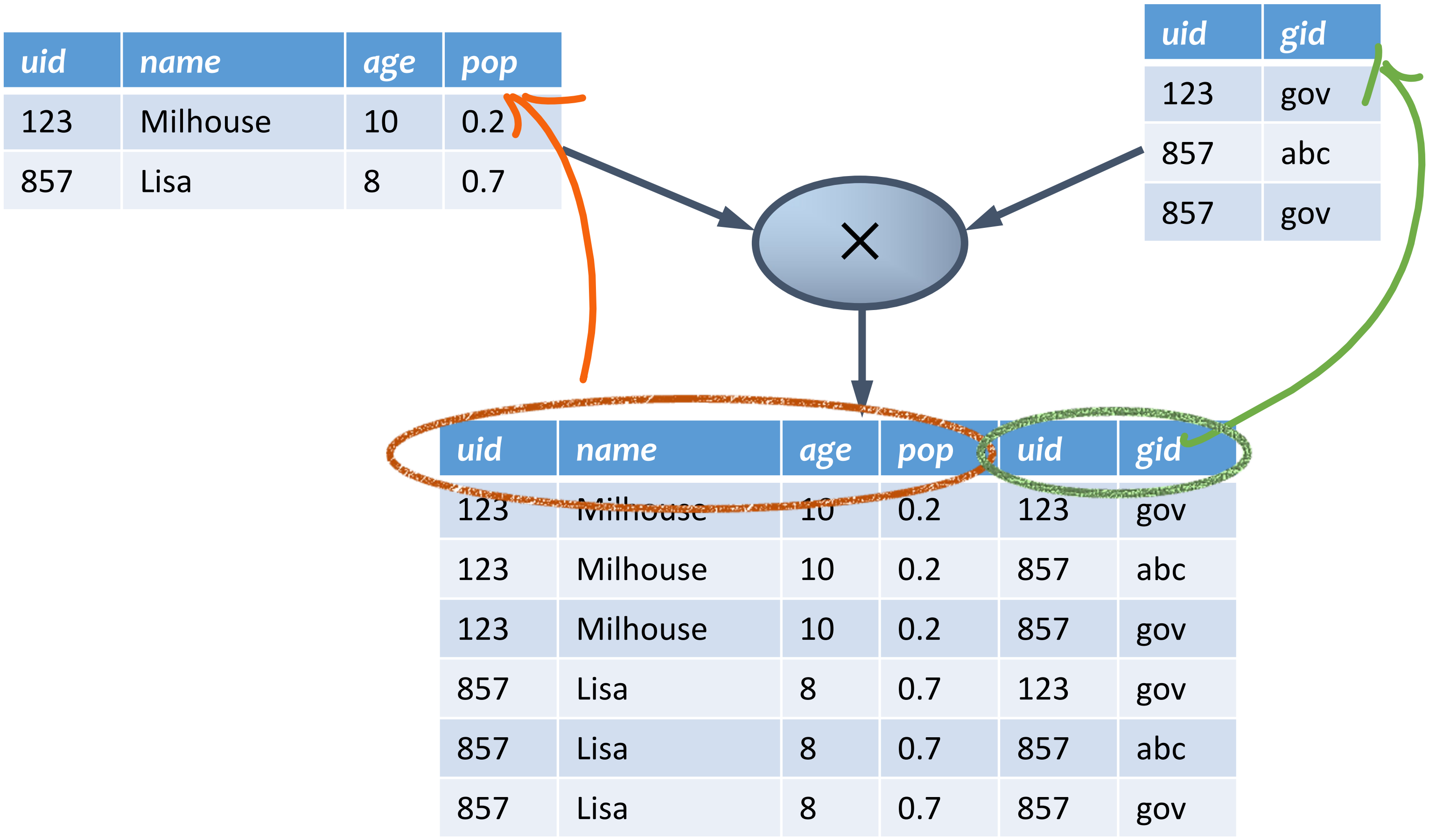


uid	*name*	*age*	*pop*
142	Bart	10	0.9
123	Milhouse	10	0.2
857	Lisa	8	0.7
456	Ralph	8	0.3
…	…	…	…

age
10
■
8
■
…

Core operator 3: Cross product ×

$$User \times Member$$



uid	name	age	pop
123	Milhouse	10	0.2
857	Lisa	8	0.7

uid	gid
123	gov
857	abc
857	gov

uid	name	age	pop	uid	gid
123	Milhouse	10	0.2	123	gov
123	Milhouse	10	0.2	857	abc
123	Milhouse	10	0.2	857	gov
857	Lisa	8	0.7	123	gov
857	Lisa	8	0.7	857	abc
857	Lisa	8	0.7	857	gov

Core operator 3: Cross product

- Input: two tables R and S

- Notation: $R \times S$

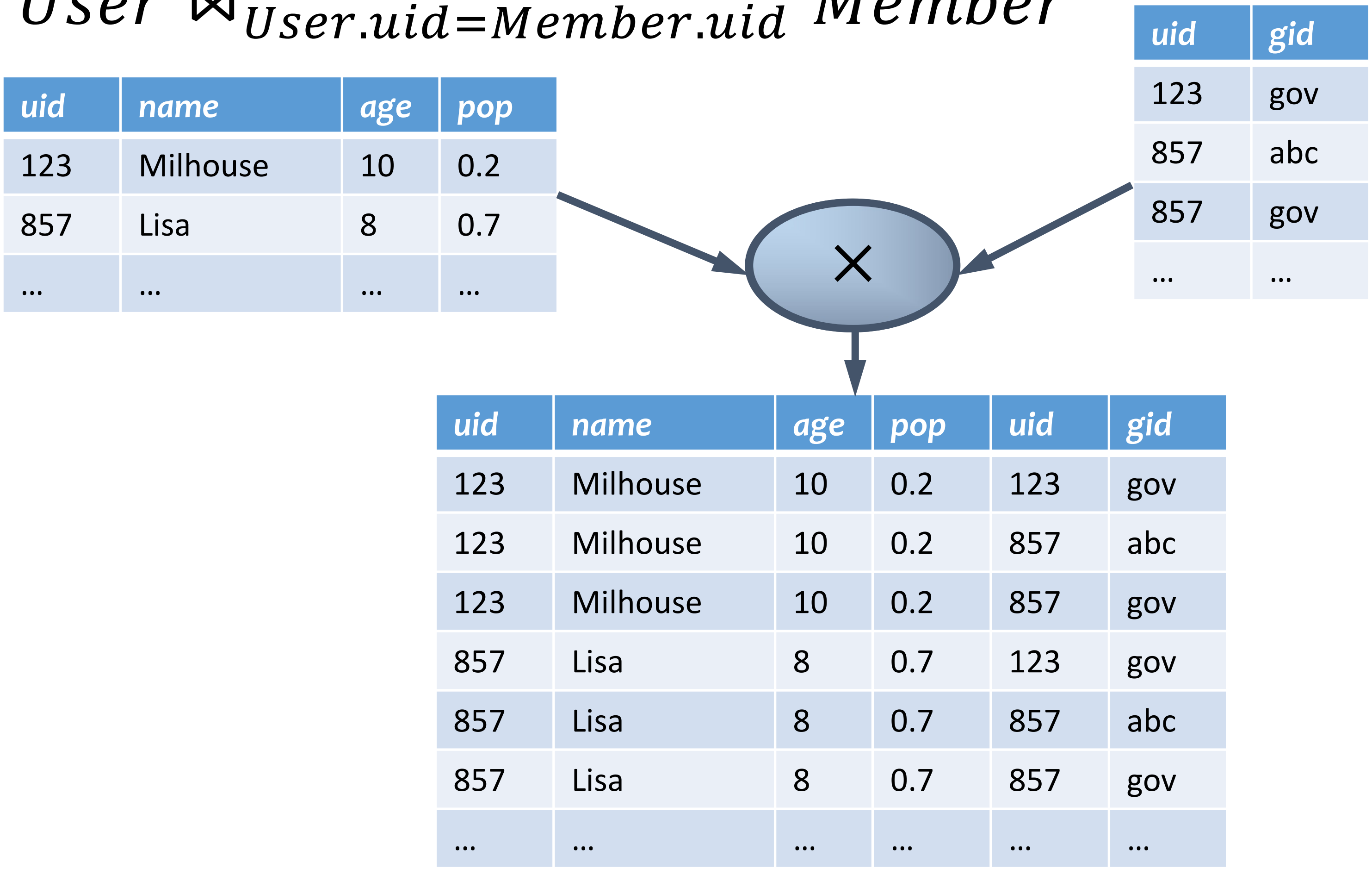
- Purpose: pairs rows from two tables

- Output: for each row r in R and each s in S, output a row rs (concatenation of r and s)

Derived operator 1: Join ⋈

- Info about users, plus IDs of their groups

$$User \bowtie_{User.uid=Member.uid} Member$$



uid	name	age	pop
123	Milhouse	10	0.2
857	Lisa	8	0.7
…	…	…	…

uid	gid
123	gov
857	abc
857	gov
…	…

uid	name	age	pop	uid	gid
123	Milhouse	10	0.2	123	gov
123	Milhouse	10	0.2	857	abc
123	Milhouse	10	0.2	857	gov
857	Lisa	8	0.7	123	gov
857	Lisa	8	0.7	857	abc
857	Lisa	8	0.7	857	gov
…	…	…	…	…	…

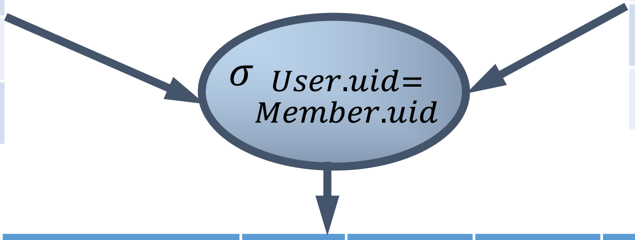
Derived operator 1: Join ⋈

- Info about users, plus IDs of their groups

$$User \bowtie_{User.uid=Member.uid} Member$$

uid	*name*	*age*	*pop*
123	Milhouse	10	0.2
857	Lisa	8	0.7
…	…	…	…

uid	*gid*
123	gov
857	abc
857	gov
…	…



$\sigma_{User.uid=Member.uid}$

uid	*name*	*age*	*pop*	*uid*	*gid*
123	Milhouse	10	0.2	123	gov
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
857	Lisa	8	0.7	857	abc
857	Lisa	8	0.7	857	gov
…	…	…	…	…	…

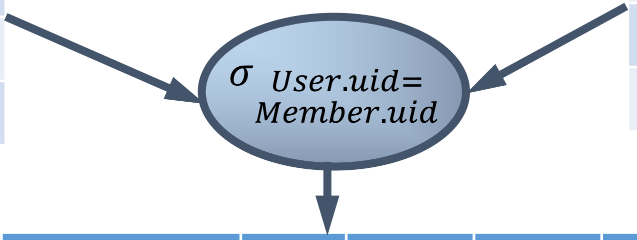
Derived operator 1: Join ⋈

- Info about users, plus IDs of their groups

$$User \bowtie_{User.uid=Member.uid} Member$$

uid	*name*	*age*	*pop*
123	Milhouse	10	0.2
857	Lisa	8	0.7
…	…	…	…

uid	*gid*
123	gov
857	abc
857	gov
…	…



$\sigma_{User.uid=Member.uid}$

Prefix a column reference with table name and "." to disambiguate identically named columns from different tables

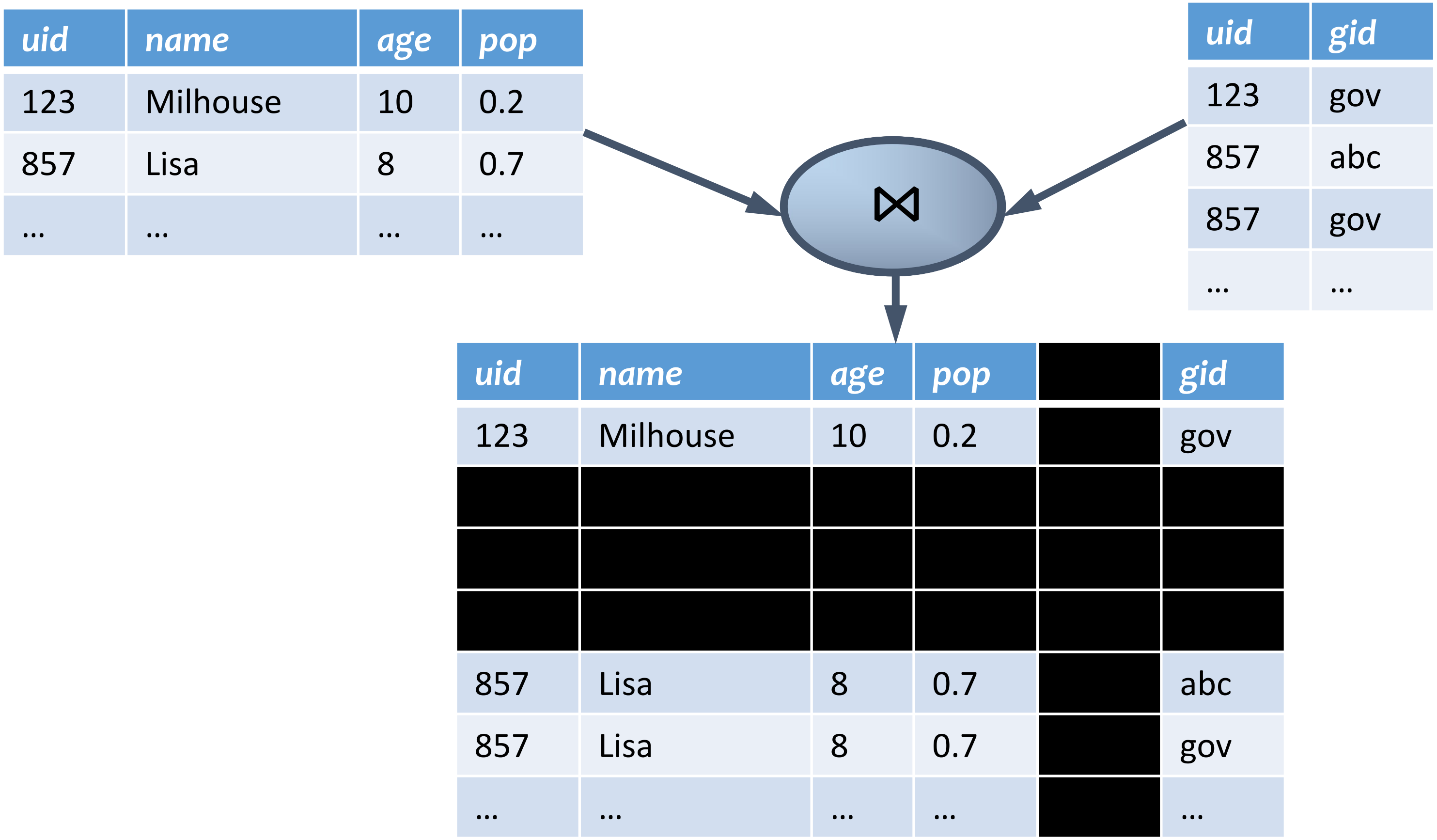
uid	*name*	*age*	*pop*	*uid*	*gid*
123	Milhouse	10	0.2	123	gov
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
857	Lisa	8	0.7	857	abc
857	Lisa	8	0.7	857	gov
…	…	…	…	…	…

Derived operator 1: Join

- Input: two tables R and S
- Notation: $R \bowtie_p S$
  - p is called a join condition (or predicate)
- Purpose: relate rows from two tables according to some criteria
- Output: for each row r in R and each row s in S, output a row rs if r and s satisfy p
- Shorthand for $\sigma_p(R \times S)$
- (A.k.a. “theta-join”)

Derived operator 2: Natural join

$$User \bowtie Member$$
$$= \pi_{uid,name,age,pop,gid}\left(User \bowtie_{\substack{User.uid= \\ Member.uid}} Member\right)$$



uid	name	age	pop
123	Milhouse	10	0.2
857	Lisa	8	0.7
…	…	…	…

uid	gid
123	gov
857	abc
857	gov
…	…

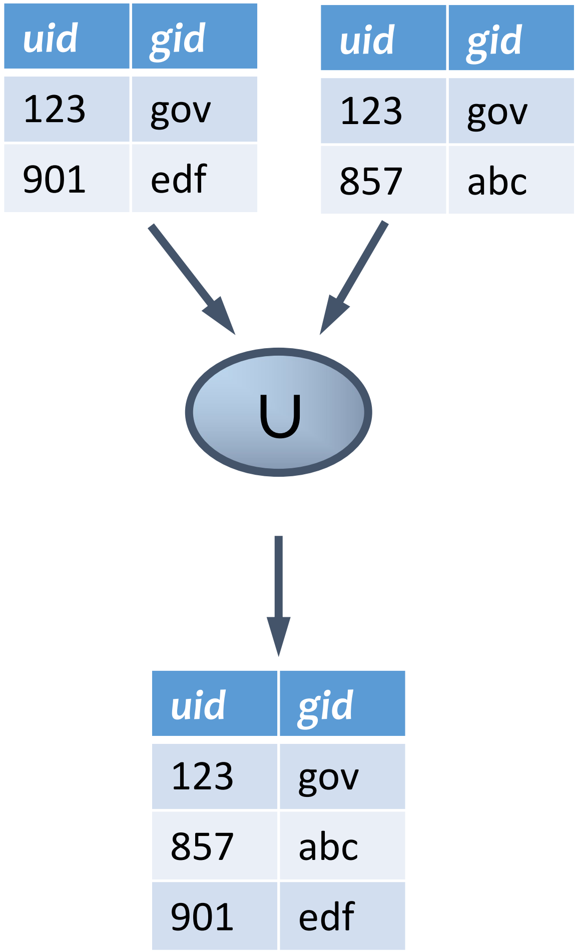
uid	name	age	pop	[redacted]	gid
123	Milhouse	10	0.2	[redacted]	gov
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
857	Lisa	8	0.7	[redacted]	abc
857	Lisa	8	0.7	[redacted]	gov
…	…	…	…	[redacted]	…

Derived operator 2: Natural join

- Input: two tables R and S
- Notation: $R \bowtie S$
- Purpose: relate rows from two tables, and
  - Enforce equality between identically named columns
  - Eliminate one copy of identically named columns
- Shorthand for $\pi_L\left(R \bowtie_p S\right)$, where
  - p equates each pair of columns common to R and S
  - L is the union of column names from R and S (with duplicate columns removed)

Core operator 4: Union

- Input: two tables R and S
- Notation: $R \cup S$
  - R and S must have identical schema
- Output:
  - Has the same schema as R and S
  - Contains all rows in R and all rows in S (with duplicate rows removed)



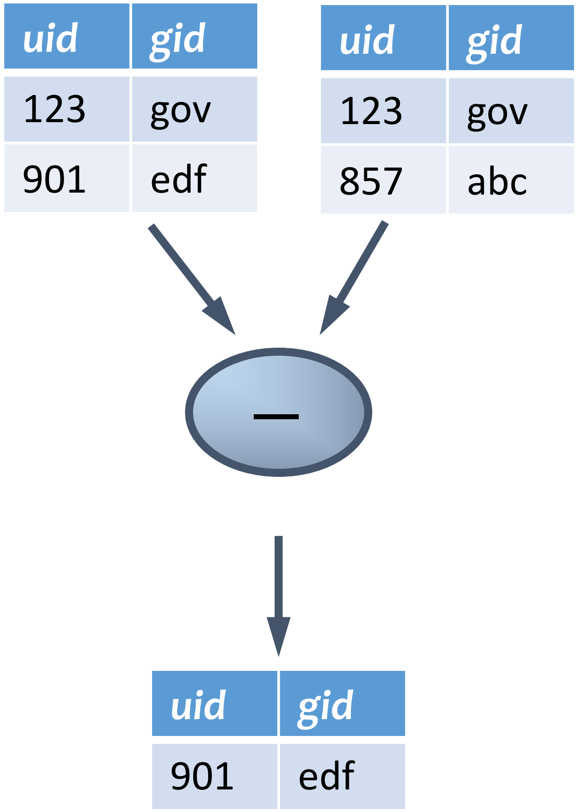
uid	*gid*
123	gov
901	edf

uid	*gid*
123	gov
857	abc

uid	*gid*
123	gov
857	abc
901	edf

Core operator 5: Difference

- Input: two tables R and S
- Notation: $R - S$
  - R and S must have identical schema
- Output:
  - Has the same schema as R and S
  - Contains all rows in R that are not in S



uid	*gid*
123	gov
901	edf

uid	*gid*
123	gov
857	abc

uid	*gid*
901	edf

Derived operator 3: Intersection

- Input: two tables R and S
- Notation: $R \cap S$
  - R and S must have identical schema
- Output:
  - Has the same schema as R and S
  - Contains all rows that are in both R and S
- Shorthand for $R - (R - S)$
- Also equivalent to $S - (S - R)$
- And equivalent to $R \bowtie S$ (why?)

Summary of operators

Core Operators

1. Selection: $\sigma_p R$
2. Projection: $\pi_L R$
3. Cross product: $R \times S$
4. Union: $R \cup S$
5. Difference: $R - S$

Derived Operators

1. Join: $R \bowtie_p S$
2. Natural join: $R \bowtie S$
3. Intersection: $R \cap S$