CPSC 304
Introduction to Database Systems

Formal Relational Languages

Textbook Reference
Database Management Systems: 4 - 4.2
(skip the calculii)

Learning Goals

- Identify the basic operators in Relational Algebra (RA).
- Use RA to create queries that include combining RA operators.
- Given an RA query and table schemas and instances, compute the result of the query.

Databases: the continuing saga

When last we left databases…

- We learned that they're excellent things
- We learned how to conceptually model them using ER diagrams
- We learned how to logically model them using relational schemas
- We knew how to normalize our database relations

We're almost ready to use SQL to query it, but first…

Balance, Daniel-san, is key

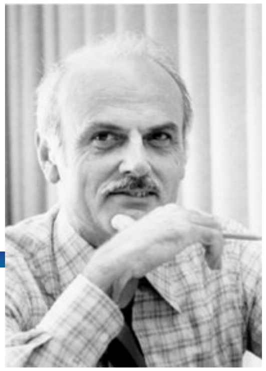
The mathematical foundations:

- Relational Algebra
  - Clear way of describing core concepts
  - *partially procedural*: describe what you want and how you want it, but the order of operations matters

Relational Query Languages

- Allow data manipulation and retrieval from a DB
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic
  - Allows for much optimization via *query optimizer*
- Query Languages **!=** Programming Languages
  - QLs not intended for complex calculations
  - QLs provide *easy access* to large datasets
  - Users *do not* need to know how to navigate through complicated data structures

Relational Algebra (RA)
All in one place



- Basic operations:
  - *Selection* (σ): Selects a subset of rows from relation.
  - *Projection* (π): Deletes unwanted columns from relation.
  - *Cross-product* (x): Allows us to combine two relations.
  - *Set-difference* (-): Tuples in relation 1, but not in relation 2.
  - *Union* ($\cup$): Tuples in relation 1 and in relation 2.
  - *Rename* (ρ): Assigns a (another) name to a relation
- Additional, inessential but useful operations:
  - *Intersection* ($\cap$), *join* ($\bowtie$), *division* (/), *assignment*($\leftarrow$)
- All operators take one or two relations as inputs and give a new relation as a result
- For the purposes of relational algebra, relations are sets
- Operations can be ***composed***. (Algebra is "closed")

Example Movies Database

Movie(<u>MovieID</u>, Title, Year)

StarsIn(<u>MovieID, StarID</u>, Character)

MovieStar(<u>StarID</u>, Name, Gender)

Example Instances

Movie:

MovieID	Title	Year
1	Star Wars	1977
2	Casablanca	1942
3	The Wizard of Oz	1939
4	Indiana Jones and the Raiders of the Lost Ark	1981

StarsIn:

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Ilsa Lund
3	3	Dorothy Gale

MovieStar:

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female
3	Judy Garland	Female

Selection (σ (sigma))

- Notation: $\sigma_p(r)$
- p is called the **selection predicate**

Set of tuples of r satisfying p

❖ Defined as:

$$\sigma_p(r) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of:

connectives : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

and

predicates:

<attribute> *op* <attribute> or

<attribute> *op* <constant>

where *op* is one of: $=, \neq, >, \geq, <, \leq$

Selection Example

Movie:

MovieID	Title	Year
1	Star Wars	1977
2	Casablanca	1942
3	The Wizard of Oz	1939
4	Indiana Jones and the Raiders of the Lost Ark	1981

$\sigma_{\text{year} > 1945}(\text{Movie})$

MovieID	Title	Year
1	Star Wars	1977
4	Indiana Jones and the Raiders of the Lost Ark	1981

Selection Example

Find all male stars from the MovieStar table.

StarID	Name	Gender
1	Harrison Ford	Male

$$\sigma_{\text{Gender} = \text{'Male'}} \text{MovieStar}$$

Projection (π (pi))

- Notation:

  $$\pi_{A1,\ A2,\ \ldots,\ Ak}\ (r)$$

  where *A1, …,Ak* are attributes (the projection list) and *r* is a relation.
- The result: a relation of the *k* attributes A1, A2, …, AK obtained from r by erasing the columns that are not listed
- Duplicate rows removed from result (relations are sets)

Projection Examples

Movie:

MovieID	Title	Year
1	Star Wars	1977
2	Casablanca	1942
3	The Wizard of Oz	1939
4	Indiana Jones and the Raiders of the Lost Ark	1981

$\pi_{\text{Title, Year}}$ (Movie)

Title	Year
Star Wars	1977
Casablanca	1942
The Wizard of Oz	1939
Indiana Jones and the Raiders of the Lost Ark	1981

π_{Year}(Movie)

Year
1977
1939
1942
1981

What is $\pi_{\text{Title,Year}}(\sigma_{\text{year} > 1945}(\text{Movie}))$?

Title	Year
Star Wars	1977
Indiana Jones and the Raiders of the Lost Ark	1981

Selection and Projection Example

Find the ids of movies made prior to 1950

Movie:

MovieID	Title	Year
1	Star Wars	1977
2	Casablanca	1942
3	The Wizard of Oz	1939
4	Indiana Jones and the Raiders of the Lost Ark	1981

MovieID
2
3

Union, Intersection, Set-Difference

- Notation: $\boldsymbol{r \cup s}$ $\boldsymbol{r \cap s}$ $\boldsymbol{r - s}$
- Defined as:

  $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

  $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$

  $r - s = \{t \mid t \in r \textbf{ and } \mathrm{t} \notin s\}$

- For these operations to be well-defined:
  1. *r, s* must have the *same arity* (same number of attributes)
  2. The attribute domains must be *compatible* (e.g., 2nd column of *r* has same domain of values as the 2nd column of *s*)
- What is the schema of the result?

Union, Intersection, and Set Difference Examples

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female
3	Judy Garland	Female

Singer

StarID	SName	Gender
3	Judy Garland	Female
4	Sam Smith	Non-binary

MovieStar ∪ Singer

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female
3	Judy Garland	Female
4	Sam Smith	Non-binary

MovieStar ∩ Singer

StarID	Name	Gender
3	Judy Garland	Female

MovieStar – Singer

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female

Set Operator Example

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female
3	Judy Garland	Female

Singer

StarID	Name	Gender
3	Judy Garland	Female
4	Sam Smith	Non-binary

Find the names of stars that are Singers but not MovieStars

Name
Sam Smith

Cartesian (or Cross)-Product

- Notation: ***r* x *s***
- Defined as:

$$r \times s = \{ t\ q \mid t \in r \textbf{ and } q \in s \}$$

- It is possible for r and s to have attributes with the same name, which creates a naming conflict.
  - In this case, the attributes are referred to solely by position.

Cartesian Product Example

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female
3	Judy Garland	Female

StarsIn

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Ilsa Lund
3	3	Dorothy Gale

MovieStar x StarsIn

1	Name	Gender	MovieID	5	Character
1	Harrison Ford	Male	1	1	Han Solo
2	Ingrid Bergman	Female	1	1	Han Solo
3	Judy Garland	Female	1	1	Han Solo
1	Harrison Ford	Male	4	1	Indiana Jones
2	Ingrid Bergman	Female	4	1	Indiana Jones
3	Judy Garland	Female	4	1	Indiana Jones
…	…	…	…	…	…

Rename (ρ (rho))

- Allows us to name results of relational-algebra expressions.
- Notation

$$\rho\ (X, E)$$

returns the expression *E* under the name *X*

- We can rename part of an expression, e.g.,
$\rho((StarID \rightarrow ID), \pi_{StarID,Name}(MovieStar))$

- We can also refer to positions of attributes, e.g.,
$\rho((1 \rightarrow ID)\ , \pi_{StarID,Name}(MovieStar))$
Is the same as above

Rename (ρ (rho))

- We can rename the resulting relation and the attributes in that relation
  $\rho(GenderlessStars(ID,Nom), \pi_{StarID,Name}(MovieStar))$

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female
3	Judy Garland	Female

$\pi_{StarID,Name}(MovieStar)$

StarID	Name
1	Harrison Ford
2	Ingrid Bergman
3	Judy Garland

GenderlessStars

ID	Nom
1	Harrison Ford
2	Ingrid Bergman
3	Judy Garland

ρ Example

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female
3	Judy Garland	Female

StarsIn

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Ilsa Lund
3	3	Dorothy Gale

ρ((1→StarID1, 5→StarID2), *MovieStar x StarsIn*)

StarID1	Name	Gender	MovieID	StarID2	Character
1	Harrison Ford	Male	1	1	Han Solo
2	Ingrid Bergman	Female	1	1	Han Solo
3	Judy Garland	Female	1	1	Han Solo
1	Harrison Ford	Male	4	1	Indiana Jones
2	Ingrid Bergman	Female	4	1	Indiana Jones
3	Judy Garland	Female	4	1	Indiana Jones
…	…	…	…	…	…

Additional Operations

- They can be defined in terms of the primitive operations
- They are added for convenience
- They are:
  - Join (Condition, Equi-, Natural) ($\bowtie$)
  - Division (/)
  - Assignment ($\leftarrow$)

Joins ($\bowtie$)

- *Condition Join*:

$$R \bowtie_c S = \sigma_c(R \times S)$$

- *Result schema* same as cross-product.
- Fewer tuples than cross-product
  - might be able to compute more efficiently
- Sometimes called a *theta-join*.
  - The reference to an attribute of a relation R can be by position (R.i) or by name (R.name)

Condition Join Example

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female
3	Judy Garland	Female

StarsIn

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Ilsa Lund
3	3	Dorothy Gale

$$\text{MovieStar} \bowtie_{\text{MovieStar.StarID} < \text{StarsIn.StarID}} \text{StarsIn}$$

1	Name	Gender	MovieID	5	Character
1	Harrison Ford	Male	2	2	Ilsa Lund
1	Harrison Ford	Male	3	3	Dorothy Gale
2	Ingrid Bergman	Female	3	3	Dorothy Gale

MovieStar $\bowtie_{\text{MovieStar.StarID} < \text{StarsIn.StarID}}$ StarsIn

MovieStar x StarsIn (first get the cross product)

1	Name	Gender	MovieID	5	Character
1	Harrison Ford	Male	1	1	Han Solo
2	Ingrid Bergman	Female	1	1	Han Solo
3	Judy Garland	Female	1	1	Han Solo
1	Harrison Ford	Male	4	1	Indiana Jones
2	Ingrid Bergman	Female	4	1	Indiana Jones
3	Judy Garland	Female	4	1	Indiana Jones
1	Harrison Ford	Male	2	2	Ilsa Lund
2	Ingrid Bergman	Female	2	2	Ilsa Lund
3	Judy Garland	Female	2	2	Ilsa Lund
1	Harrison Ford	Male	3	3	Dorothy Gale
2	Ingrid Bergman	Female	3	3	Dorothy Gale
3	Judy Garland	Female	3	3	Dorothy Gale

MovieStar $\bowtie_{\text{MovieStar.StarID} < \text{StarsIn.StarID}}$ StarsIn

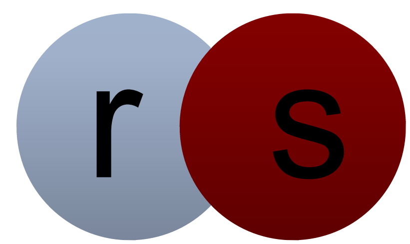
Now remove rows based on the condition stated above.

1	Name	Gender	MovieID	5	Character
~~1~~	~~Harrison Ford~~	~~Male~~	~~1~~	~~1~~	~~Han Solo~~
~~2~~	~~Ingrid Bergman~~	~~Female~~	~~1~~	~~1~~	~~Han Solo~~
~~3~~	~~Judy Garland~~	~~Female~~	~~1~~	~~1~~	~~Han Solo~~
~~1~~	~~Harrison Ford~~	~~Male~~	~~4~~	~~1~~	~~Indiana Jones~~
~~2~~	~~Ingrid Bergman~~	~~Female~~	~~4~~	~~1~~	~~Indiana Jones~~
~~3~~	~~Judy Garland~~	~~Female~~	~~4~~	~~1~~	~~Indiana Jones~~
1	Harrison Ford	Male	2	2	Ilsa Lund
~~2~~	~~Ingrid Bergman~~	~~Female~~	~~2~~	~~2~~	~~Ilsa Lund~~
~~3~~	~~Judy Garland~~	~~Female~~	~~2~~	~~2~~	~~Ilsa Lund~~
1	Harrison Ford	Male	3	3	Dorothy Gale
2	Ingrid Bergman	Female	3	3	Dorothy Gale
~~3~~	~~Judy Garland~~	~~Female~~	~~3~~	~~3~~	~~Dorothy Gale~~

One more thing you may find helpful: Assignment Operation

- Notation: $t \leftarrow E$
  assigns the result of expression E to a temporary relation t.
- Used to break complex queries to small steps.
- Assignment is always made to a temporary relation variable.
- Example: Write $r \cap s$ in terms of $\cup$ and/or $-$

$$temp1 \leftarrow r - s$$
$$result \leftarrow r - temp1$$



Equi-Join & Natural Join

- *Equi-Join*: A special case of condition join $R \bowtie_c S = \sigma_c(R \times S)$, where c contains only ***equalities***. Only the first of the repeated columns is retained.
- *Natural Join*: Equijoin on ***all*** common attributes
  - *Result schema:* similar to cross-product, but has only one copy of each common attribute
  - No need to show the condition
  - If the two attributes have no common attributes, this would be the same as cross product.
  - This is what we saw in BCNF & 3NF

Equi and Natural Join Examples

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Ingrid Bergman	Female
3	Judy Garland	Female

StarsIn

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Ilsa Lund
3	3	Dorothy Gale

MovieStar $\bowtie$ StarsIn

StarID	Name	Gender	MovieID	Character
1	Harrison Ford	Male	1	Han Solo
1	Harrison Ford	Male	4	Indiana Jones
3	Judy Garland	Female	3	Dorothy Gale
2	Ingrid Bergman	Female	2	Ilsa Lund

Join Example

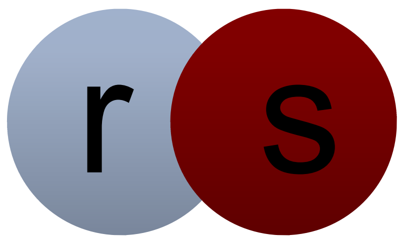
- Find the names of all Movie Stars who were in any Movie

Name
Harrison Ford
Ingrid Bergman
Judy Garland

Assignment Operation

- Notation: $t \leftarrow E$
  assigns the result of expression E to a temporary relation t.
- Used to break complex queries to small steps.
- Assignment is always made to a temporary relation variable.
- Example: Write $r \cap s$ in terms of $\cup$ and/or –

$$temp1 \leftarrow r - s$$
$$result \leftarrow r - temp1$$



Remember this one?

- Find the names of all Movie Stars who were in any Movie

$$\pi_{name}(\text{MovieStar} \bowtie \text{StarsIn})$$

Name
Harrison Ford
Ingrid Bergman
Judy Garland

- What if we wanted all Movies Stars who were in *all* movies?

Division

- Notation: ***r / s* or *r ÷ s***
- Useful for expressing queries that include a notion of "**for all**" or "**for every**", e.g., *Find movie stars who were in **all** movies*.
- Let *r* and *s* be relations on schemas R and S respectively where
  - r = $(A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - s = $(B_1, \ldots, B_n)$

  Then r / s is a relation on schema

  $r / s = (A_1, \ldots, A_m)$

  defined as

  $$r / s = \{\, t \mid t \in \Pi_{r\text{-}s}(r) \wedge \forall\, u \in s\,(\, tu \in r\,)\,\}$$

  - i.e., ***A/B*** **contains all *x* tuples (MovieStars) such that for *every* *y* tuple (movies) in *B*, there is an *x,y* tuple in *A.***

Examples of Division A/B

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

B2

pno
p2
p4

B3

pno
p1
p2
p4

A/B1

sno
s1
s2
s3
s4

A/B2

sno
s1
s4

A/B3

sno
s1

Find the name of actors who have been in ***all*** movies

Be careful in choosing the input relations!

$InAll \leftarrow \pi_{StarID,\ MovieID} StarsIn / \pi_{MovieID}(Movie)$

$\pi_{Name}(InAll \bowtie MovieStar)$

Case study of complex relational algebra: build up division of r/s from other operators

- Let X be attributes not in R and Y be attributes in S
- Idea: compute all values that are "disqualified" by some value in *s*.
  - value *x* is *disqualified* if by attaching *y* value from *s*, we obtain an *xy* tuple that is not in *r*.
- Take difference from all values

Expressing r/s Using Basic Operators

- Like a join, can be computed from basic operators
- *Idea*:
  - let X the set of attributes of r that are not in s
  - (1) compute the X-projection of r
  - (2) compute all *X*-projection values of r that are "disqualified" by some value in *s*.
    - value *x* is *disqualified* if by attaching *y* value from *s*, we obtain an *xy* tuple that is not in *r*.
  - result is (1)-(2)
- So,
  - Disqualified *x* values: $\pi_X((\pi_X(r) \times s) - r)$
  - r/s is $\pi_X(r) - \pi_X((\pi_X(r) \times s) - r)$

Example: building up division subtract off disqualified answers

$A = R$

Sno	Pno
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

$B2 = S$

Pno
P2
P4

$\pi_X(R)$

Sno
S1
S2
S3
S4

All possible values given R

$\pi_X(R) \times S$

Sno	Pno
S1	P2
S1	P4
S2	P2
S2	P4
S3	P2
S3	P4
S4	P2
S4	P4

Values needed for $\pi_X(R)$

$\pi_X(R) \times S - R$

Sno	Pno
S2	P4
S3	P4

These values aren't in R

$$\pi_X(R) - \pi_X(\pi_X(R) \times S - R) = A/B2$$

A/B2 =

Sno
S1
S4

Answers not disqualified

If you want to practice…

Try this site:

https://dbis-uibk.github.io/relax/

You can play with RA queries and see the results.

Learning Goals Revisited

- Identify the basic operators in RA.
- Use RA to create queries that include combining RA operators.
- Given an RA query and table schemas and instances, compute the result of the query.