Relational Calculus

courtesy of Joe Hellerstein for some slides

Guifeng Zheng
SUN YAT-SEN UNIVERSITY

Relational Calculus

- Query has the form: $\{T \mid p(T)\}$
  - ***T*** is a tuple variable.
  - ***p(T)*** is a *formula* containing ***T*** .
- Answer = tuples T for which $p(T)$ = *true*.

Formulae

- *Atomic formulae:*

  T $\in$ *Relation*

  T.a op T.b

  T.a op constant

  … *op* is one of $<, >, =, \leq, \geq, \neq$
- A *formula* can be:
  - an atomic formula
  - $\neg p, p \wedge q, p \vee q, p \Rightarrow q$
  - $\exists R(p(R))$
  - $\forall R(p(R))$

Free and Bound Variables自由与约束变量

- Quantifiers量词: ∃ and ∀
- Use of $\exists X$ or $\forall X$ *binds* X.
  - A variable that is not bound is free.
- Recall our definition of a query:
  - $\{T \mid p(T)\}$

- **Important restriction:**
  – T must be the only free variable in p(T).
  – all other variables must be bound using a quantifier.

Simple Queries

- Find all sailors with rating above 7

$$\{S \mid S \in Sailors \ \wedge \ S.rating > 7\}$$

$$= RA: \ \sigma_{rating>7}(\text{Sailors})$$

- Find names and ages of sailors with rating above 7.

$$\{S \mid \exists S1 \in Sailors(S1.rating > 7 \ \wedge \ S.sname = S1.sname \ \wedge \ S.age = S1.age)\}$$

$$= RA: \pi_{\text{sname, age}}(\sigma_{rating>7}(\text{Sailors}))$$

  - Note: *S* is a variable of 2 fields (i.e. S is a projection of *Sailors*)

Joins

Find sailors rated > 7 who've reserved boat #103

$$\{ S \mid S \in Sailors \land S.rating > 7 \land \exists R(R \in Reserves \land R.sid = S.sid \land R.bid = 103) \}$$

=RA:

$$(\sigma_{rating>7}(Sailors)) \bowtie (\sigma_{bid=103}(Reserves))$$

Joins (continued)

Find sailors rated > 7 who've reserved a red boat

$$\{ S \mid S \in Sailors \wedge S.rating > 7 \wedge \exists R(R \in Reserves \wedge R.sid = S.sid \wedge \exists B(B \in Boats \wedge B.bid = R.bid \wedge B.color = \text{'red'})) \}$$

$$(\sigma_{rating>7}(Sailors)) \bowtie Reserves \bowtie (\sigma_{color=red}(Boats))$$

This may look cumbersome, but it's not so different from SQL!

Universal Quantification

Find sailors who've reserved <u>all</u> boats

$$\{ S \mid S \in \text{Sailors} \wedge \forall B \in \text{Boats} \; (\exists R \in \text{Reserves} \; (S.sid = R.sid \wedge B.bid = R.bid)) \}$$

RA: (hint: use ÷)

A trickier example…

Find sailors who've reserved all Red boats

$$\{ S \mid S \in Sailors \wedge \forall B \in Boats\ (B.color = \text{'red'} \Rightarrow \exists R (R \in Reserves \wedge S.sid = R.sid \wedge B.bid = R.bid)) \}$$

Alternatively…

$$\{ S \mid S \in Sailors \wedge \forall B \in Boats\ (B.color \neq \text{'red'} \vee \exists R (R \in Reserves \wedge S.sid = R.sid \wedge B.bid = R.bid)) \}$$

a $\Rightarrow$ **b** is the same as $\neg a \vee b$

		b	
		T	F
a	T	T	F
	F	T	T

A Remark: Unsafe Queries

- $\exists$ syntactically correct calculus queries that have an infinite number of answers! *Unsafe* queries.
  - e.g., $$\left\{ S \mid \neg \left(S \in Sailors \right) \right\}$$
  - Solution???? Don't do that!

Expressive Power

- Expressive Power (Theorem due to Codd):
  - Every query that can be expressed in relational algebra can be expressed as a safe query in relational calculus;   the converse is also true.

- *Relational Completeness*:

  Query language (e.g., SQL) can express every query that is expressible in relational algebra/calculus.

  (actually, SQL is more powerful, as we will see…)

Summary

- Formal query languages — simple and powerful.
  - *Relational algebra* is <u>operational</u>
    - used as internal representation for query evaluation plans.
  - *Relational calculus* is “declarative”
    - query = “what you want”, <u>not</u> “how to compute it”
  - *Same expressive power*

    --> *relational completeness.*

- Several ways of expressing a given query
  - a *query optimizer* should choose the most efficient version.

Your turn …

- Schema:

  Movie(<u>title</u>, year, studioName)

  ActsIn(<u>movieTitle</u>, <u>starName</u>)

  Star(<u>name</u>, gender, birthdate, salary)

- Queries to write in Relational Calculus:
  1. Find all movies by Paramount studio
  2. … movies whose stars are all women
  3. … movies starring Kevin Bacon
  4. Find stars who have been in a film w/Kevin Bacon
  5. Stars within six degrees of Kevin Bacon*
  6. Stars connected to K. Bacon via <u>any number</u> of films**

* Try *two* degrees for starters

** Good luck with this one!

Answers …

1. Find all movies by Paramount studio

$$\{M \mid M \in Movie \wedge M.studioName = \text{'Paramount'}\}$$

Answers …

2. Movies whose stars are all women

$$\{M \mid M \in Movie \wedge \forall A \in ActsIn((A.movieTitle = M.title) \Rightarrow \exists S \in Star(S.name = A.starName \wedge S.gender = \text{'F'}))\}$$

Answers …

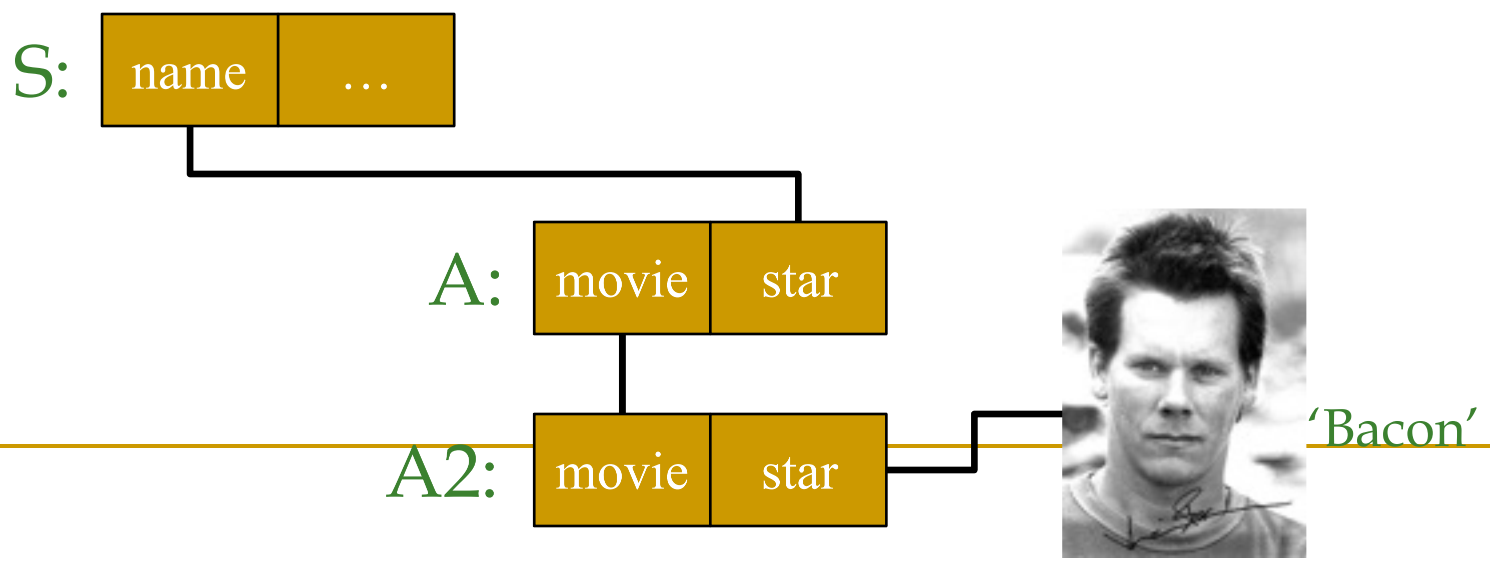
3. Movies starring Kevin Bacon

$$\{M \mid M \in Movie \wedge \exists A \in ActsIn(A.movieTitle = M.title \wedge A.starName = \text{‘Bacon’})\}$$

Answers …

4. Stars who have been in a film w/Kevin Bacon

$$\{S \mid S \in Star \wedge \exists A \in ActsIn(A.starName = S.name \wedge \exists A2 \in ActsIn(A2.movieTitle = A.movieTitle \wedge A2.starName = \text{'Bacon'}))\}$$

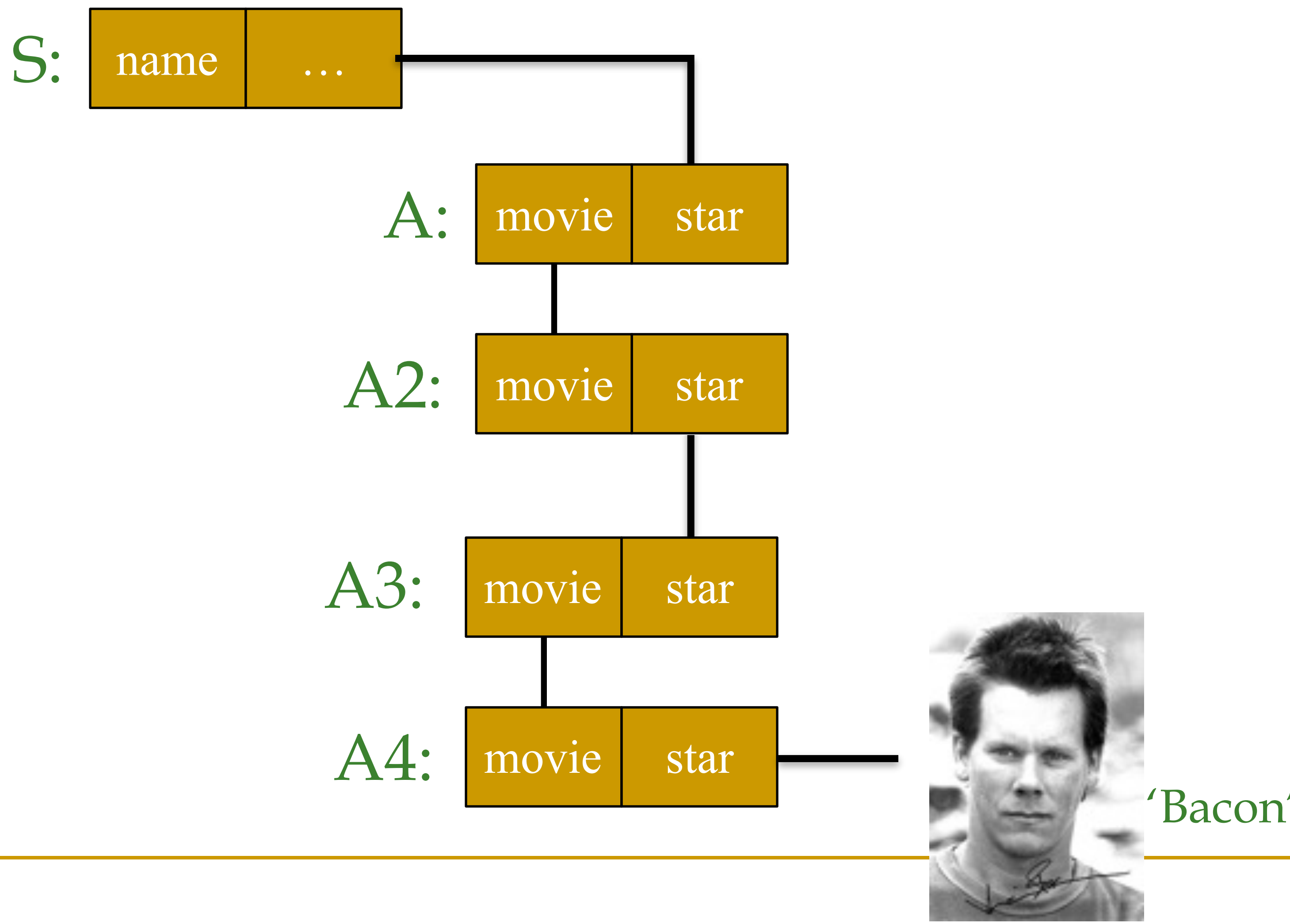


Answers …

5. Stars within ~~six~~ **two** degrees of Kevin Bacon

$\{S \mid S \in Star \wedge$
$\quad \exists A \in ActsIn(A.starName = S.name \wedge$
$\quad\quad \exists A2 \in ActsIn(A2.movieTitle = A.movieTitle \wedge$
$\quad\quad\quad \exists A3 \in ActsIn(A3.starName = A2.starName \wedge$
$\quad\quad\quad\quad \exists A4 \in ActsIn(A4.movieTitle = A3.movieTitle \wedge$
$\quad\quad\quad\quad\quad A4.starName = \text{'Bacon'}))\}$

Two degrees:


S:
name
…
A:
movie
star
A2:
movie
star
A3:
movie
star
A4:
movie
star
'Bacon'

Answers …

6. Stars connected to K. Bacon via <u>any number</u> of films

- **Sorry … that was a trick question**
  - <u>Not expressible</u> in relational calculus!!
- **What about in relational algebra?**
  - We will be able to answer this question shortly …