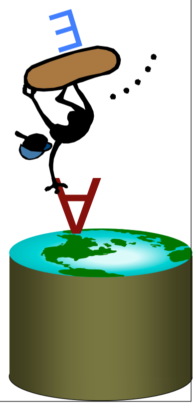
Relational Calculus

Guest lecturer:

Christopher Olston
Yahoo! Research



Relational Calculus

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

Formulae

- ***Atomic formulae:***

  T ∈ *Relation*

  T.a op T.b

  T.a op constant

  … *op* is one of $<,>,=,\le,\ge,\ne$
- **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

- **Quantifiers: ∃ 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* | *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**

  $\{S \mid S \in Sailors \wedge S.rating > 7\}$
- **Find names and ages of sailors with rating above 7.**

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

  - 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

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

Joins (continued)

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

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

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

Universal Quantification

Find sailors who've reserved all boats

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

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	b: 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., $\{S \mid \neg\{S \in Sailors\}\}$
  - Solution???? Don't do that!

Your turn …

- **Schema:**
  Movie(title, year, studioName)
  ActsIn(movieTitle, starName)
  Star(name, 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 any number of films**

* Try *two* degrees for starters ** Good luck with this one!

Answers …

1. Find all movies by Paramount studio

{M | M∈Movie ∧
M.studioName = ‘Paramount’}

Answers …

2. Movies whose stars are all women

{M | M∈Movie ∧
∀A∈ActsIn((A.movieTitle = M.title) ⇒
∃S∈Star(S.name = A.starName ∧
S.gender = ‘F’))}

Answers …

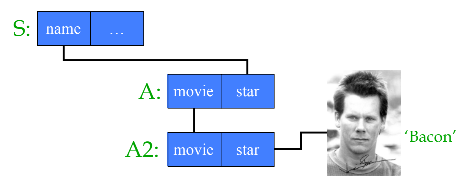
3. Movies starring Kevin Bacon

{M | M∈Movie ∧
∃A∈ActsIn(A.movieTitle = M.title ∧
A.starName = ‘Bacon’))}

Answers …

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

{S | S∈Star ∧
∃A∈ActsIn(A.starName = S.name ∧
∃A2∈ActsIn(A2.movieTitle = A.movieTitle ∧
A2.starName = ‘Bacon’))}

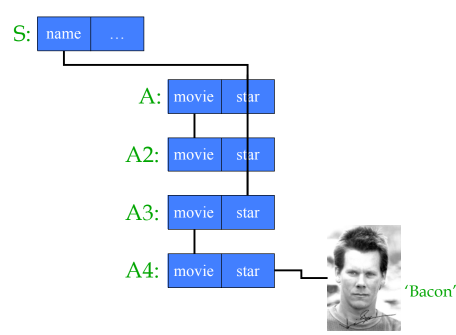


Answers …

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

{S | S∈Star ∧
∃A∈ActsIn(A.starName = S.name ∧
∃A2∈ActsIn(A2.movieTitle = A.movieTitle ∧
∃A3∈ActsIn(A3.starName = A2.starName ∧
∃A4∈ActsIn(A4.movieTitle = A3.movieTitle ∧
A4.starName = ‘Bacon’))}

Two degrees:



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 …

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.
- ***<u>Relational Completeness</u>*:**

  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…)

Question:

- **Can we express query #6 in relational algebra?**
- **A: If we could, then by Codd's theorem we could also express it in relational calculus. However, we know the latter is not possible, so the answer is <u>no</u>.**

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.