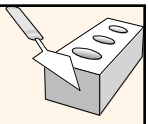
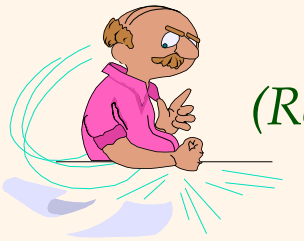
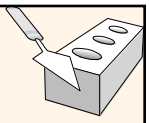


Introduction to Data Management



Lecture #12
(Relational Languages II)

Instructor: Mike Carey
mjcarey@ics.uci.edu



It's time again for....

Friday Nights with Databases

Brought to you by…

Announcements

- Homework notes
  - HW #3 was due today (tomorrow if one day late)
  - HW #4 will come out on Monday (after the exam)
- Exam notes (time flies!)
  - ***Midterm #1 is on Monday (in class)!***
  - We'll use assigned seating – come early!
  - Bring an 8.5"x11" (2-sided) cheat sheet if you want!
  - Don't bank on last-minute Piazza answers…
- Today's lecture plan:
  - Relational languages continued…
  - ***Note*:** Today's info won't be on Monday's exam! ☺

Ex: *Wisconsin Sailing Club Database*

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5

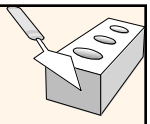
Reserves

sid	bid	date
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/93

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

REVIEW: Find names of sailors who've reserved a red boat



Sailors(sid, sname, rating, age) Reserves(sid, bid, day)
Boats(bid, bname, color)

- Information about boat color only available in Boats; so need to do two joins:

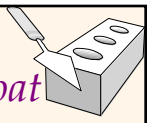
$$\pi_{sname}((\sigma_{color='red'} Boats) \bowtie Reserves \bowtie Sailors)$$

- A more "efficient" solution:

$$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color='red'} Boats) \bowtie Res) \bowtie Sailors)$$

A query optimizer will find the latter, given the 1st query!

Find sailors who've reserved a red or a green boat



Sailors(sid, sname, rating, age) Reserves(sid, bid, day)
Boats(bid, bname, color)

- Can identify all red or green boats, then find sailors who've reserved one of these boats:

$$\rho\ (Tempboats, (\sigma_{color='red' \vee color='green'} Boats))$$

$$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$$

- Can also define Tempboats using union! (Q: How?)
- What happens if $\vee$ is replaced by $\wedge$ in *this* query?

or直接换成and的话，是指这个color是红又是绿，没有这种颜色，因为是想说这个sailor有红色的船又有绿色的船

Find sailors who've reserved a red and a green boat

Sailors(sid, sname, rating, age) Reserves(sid, bid, day)
Boats(bid, bname, color)

- Previous approach won't work! Must identify sailors who'reserved red boats and sailors who've reserved green boats, then find their intersection (notice that *sid* is a key for Sailors!):

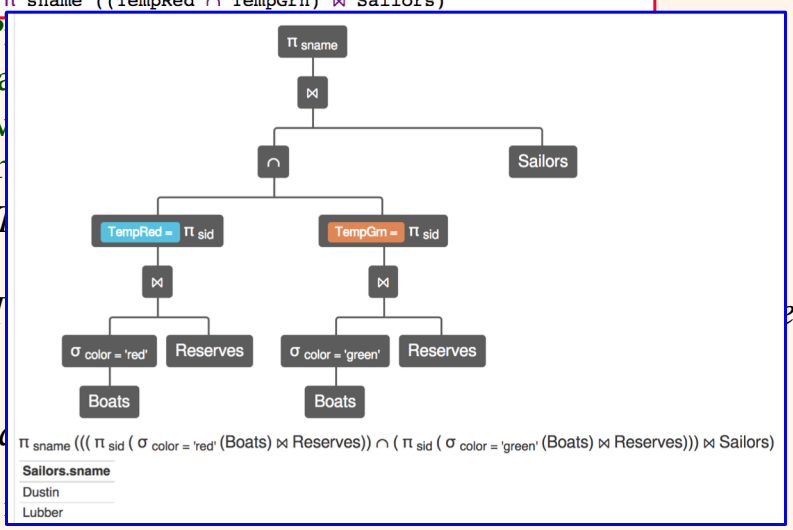
$$\rho\,(Tempred,\ \pi_{sid}((\sigma_{color='red'}\ Boats)\bowtie \mathrm{Re}serves))$$

$$\rho\,(Tempgreen,\ \pi_{sid}((\sigma_{color='green'}\ Boats)\bowtie \mathrm{Re}serves))$$

$$\pi_{sname}((Tempred \cap Tempgreen)\bowtie Sailors)$$

Find sailors who've reserved a red and a green boat

```
1 TempRed = π sid (σ color = 'red' (Boats) ⋈ Reserves)
2 TempGrn = π sid (σ color = 'green' (Boats) ⋈ Reserves)
3 π sname ((TempRed ∩ TempGrn) ⋈ Sailors)
```



π sname (((π sid (σ color = 'red' (Boats) ⋈ Reserves)) ∩ (π sid (σ color = 'green' (Boats) ⋈ Reserves))) ⋈ Sailors)

Sailors.sname
Dustin
Lubber

Find the names of sailors who've reserved all boats

Sailors(sid, sname, rating, age) Reserves(sid, bid, day)
Boats(bid, bname, color)

- Uses **division**; schemas of the input relations feeding the / operator must be *carefully chosen*:

$$\rho\,(Tempsids, (\pi_{sid,bid} Reserves) / (\pi_{bid} Boats))$$

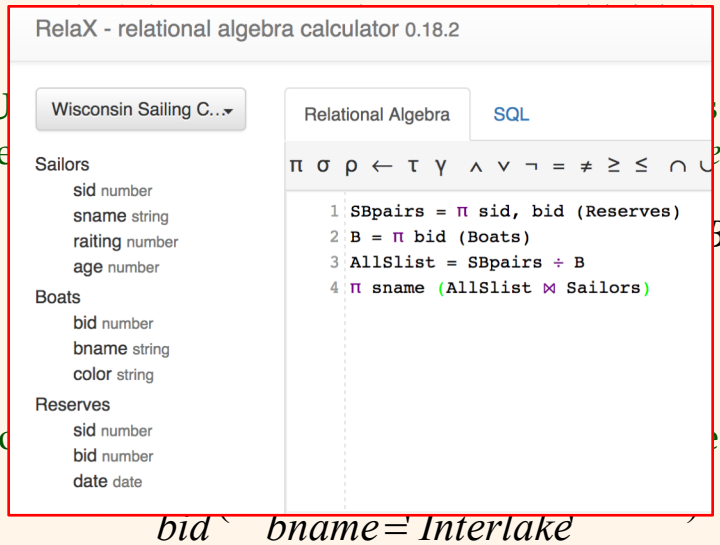
这两个should be the same type

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

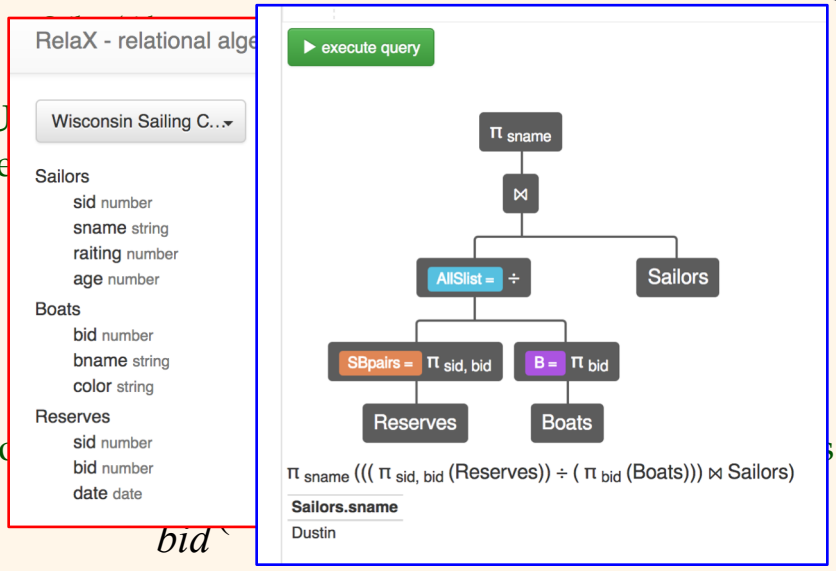
- To find sailors who've reserved all 'Interlake' boats:

$$..... / \pi_{bid}(\sigma_{bname='Interlake'} Boats)$$

Find the names of sailors who've reserved all boats



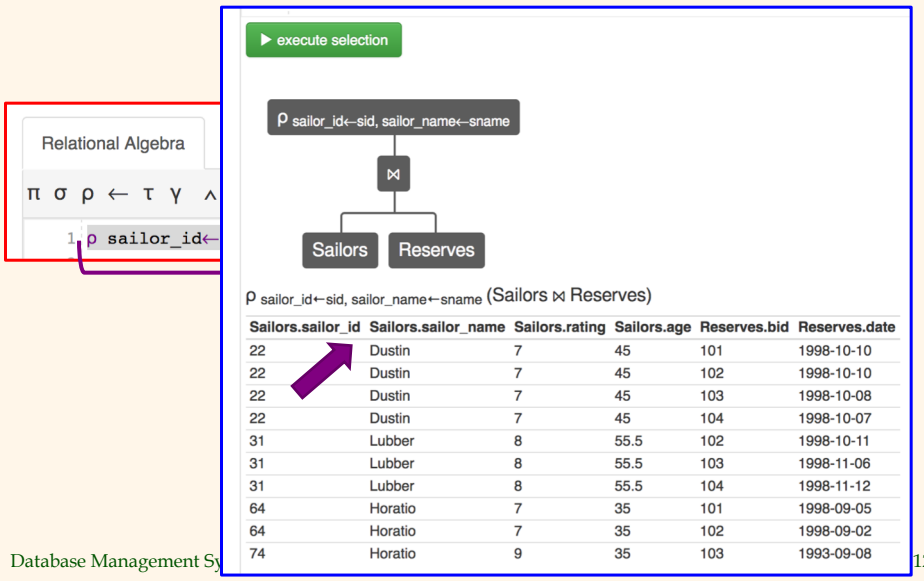
Find the names of sailors who've reserved all boats

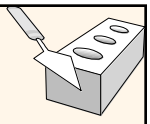


- U
- To

bid

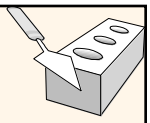
PS: RelaX Renaming Example...





Relational Algebra Summary

- The relational model has (several) rigorously defined query languages that are both simple and powerful in nature.
- Relational algebra is more "operational"; very useful as an internal representation for query evaluation plans. (SQL "EXPLAIN")
- Several ways of expressing a given query; a query optimizer should choose the most efficient version. (Take CS122C...! ☺)
- We'll add a few more operators later on…
- Next up for now: *Relational Calculus*



NEXT: Relational Calculus!

- Comes in two flavors: *Tuple relational calculus* (TRC) and *Domain relational calculus* (DRC).
- Calculus has *variables, constants, comparison ops, logical connectives* and *quantifiers*.
  - *TRC*: Variables range over (i.e., get bound to) *tuples*.
  - *DRC*: Variables range over *domain elements* (= field values).
  - Both TRC and DRC are simple subsets of first-order logic.
- Expressions in the calculus are called *formulas*. An answer tuple is essentially an assignment of constants to variables that make the formula evaluate to *true*.
- TRC is the basis for various query languages (Quel, SQL, OQL, XQuery, …), while DRC is the basis for example-based relational query UIs. We'll study ***TRC***!

2.查询理论

2.1 SQL之母：Relational Calculus

其实SQL并非起源与我们熟知的关系代数（Relational Algebra），实际上关系代数更多地应用于查询的分析和执行。SQL真正之母是很少听说过的关系演算（Relational Calculus），它的查询形式是基于谓词逻辑（Predicate logic）来定义的。具体来说，有元组关系演算（TRC）和域关系演算（DRC）两种：

2.2 Tuple Relational Calculus（TRC）　元组关系演算

TRC被Codd引入作为关系模型的一部分，用来为关系模型提供一种声明式的数据库查询语言。TRC查询的形式为：{T | Condition(T)}。其中T叫做目标（Target），是遍历所有元组的变量。Condition是查询的条件，用具体元组替换T来判断真值。例如，$\{S|Staff(S) \wedge S.salary > 10000\}$。

TRC与SQL有什么对应关系吗？例如上面的例子，S可以看做是SQL中的SELECT部分，但TRC只能返回整个Tuple。Staff(S)可以看做是FROM，而S.salary > 10000可以看做是WHERE。执行方式为：用具体的Tuple去替换S（不能凭空用任意值去替换S吧？没有SQL易理解），看是否满足S在Staff中和S.salary大于10000两个条件。

来看一个复杂一些的TRC，这里就要用到前面逻辑中的概念了。下面S是Free变量，T被量词限定所以是Bound变量。在TRC中，Bound变量用在条件中，用来判断关于Tuple的真假，决定是否保留。而Free变量则用在Target中，用来指定要返回的Tuple。条件中只能有一个Free变量，否则就没法判断真假了。

$$\{S|Student(S)AND(\exists T \in Transcript$$
$$(S.Id = T.StudIdANDT.CrsCode =' AAA'))\}$$

转换很简单，等价的SQL就是：

$$SELECTS$$
$$FROMStudentS, TranscriptT$$
$$WHERES.Id = T.StudIdANDT.CrsCode =' AAA'$$

上面这样一对比就能看出，**本质上，SQL不过是TRC的一层语法糖**：

- **量词哪去了？**：SQL制定了一些约定（Convention）简化了写法，例如SELECT中没有出现的关系都隐式地加上∃量词限定
- **运算丰富**：SQL还融入了聚合函数、关系代数以及集合的交并集运算等（只不过是更多的Sugar）
- **局限性**：但SQL中不允许使用∀量词及否定（Negation）。

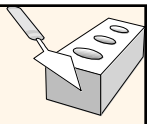
所以说，TRC更加通用，但SQL更加简单。

2.3 Domain Relational Calculus（DRC）

DRC的Target部分是由域变量（Domain Variable）而非TRC中的整个Tuple组成，即Tuple某个属性的值域中对应的变量。DRC查询的形式为：$\{X_1, X_2, \ldots, X_n | condition(X_1, X_2, \ldots, X_n)\}$。其中每个X都要么是域变量要么是常量。例如，TRC查询$\{S | Staff(S) \wedge S.salary > 10000\}$，在DRC中可以写作$\{S.id | Staff(S.id, S.salary) \wedge S.salary > 10000\}$

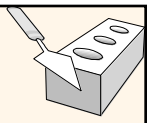
因为以属性而非整个元组为Target，所以查询条件中经常出现非Target属性的大量∃也是DRC的标志，但DRC能自动匹配属性值和同名属性关联。所以，某些查询用TRC表达简单，而有些用DRC能更简单些。

TRC和DRC可以相互转换，但转换成关系代数却不那么容易，有些查询甚至无法转换。



Tuple Relational Calculus

- *Query* in TRC has the form:
  { t(attrlist) | P(t) }
- *Answer* includes all tuples t with (optionally) specified schema (attrlist) that cause *formula* P(t) to be *true*.
- *Formula* is recursively defined, starting with simple *atomic formulas* (getting tuples from relations or making comparisons of values), and building up bigger and better Boolean formulas using *logical connectives*.

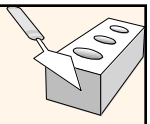


TRC Formulas

- *Atomic formula:*
  - r ∈ R, or r ∉ R , or r.a *op* s.b, or r.a *op* constant
  - *op* is one of <, >, ≤, ≥, ≠,=
- *Formula:*
  - an atomic formula, or
  - ¬ P, P∧Q, P∨Q, where P and Q are formulas, or
  - ∃r ∈ R (P(r)), where variable r is *free* in P(…), or
  - ∀r ∈ R (P(r)), where variable r is *free* in P(…), or
  - P ⇒ Q (pronounced "implies", equivalent to (¬ P) ∨ Q))

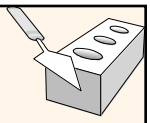
∃∄∀∈∉ ¬ ∧∨⇒=≠<>≤≥

Free and Bound Variables



- The use of a quantifier such as $\exists t \in T$ or $\forall t \in T$ in a formula is said to *bind* t.
  - A variable that is not bound is free.
- Now let us revisit the definition of a TRC query:
  - { t(a1, a2, …) | P(t) }
- There is an important restriction: the variable t that appears to the left of the | ("such that") symbol must be the *only* free variable in the formula P(...).
- Let's look at some examples!

Find sailors with a rating above 7



Sailors(sid, sname, rating, age) Reserves(sid, bid, day)
Boats(bid, bname, color)

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

- This is equivalent to the more general form:

$$\{ t(id, nm, rtg, age) \mid \exists s \in Sailors$$
$$(t.id = s.sid \wedge t.nm = s.sname$$
$$\wedge t.rtg = s.rating \wedge t.age = s.age$$
$$\wedge s.rating > 7) \}$$

*(**Q:** See how each one specifies the answer's schema and values...? Note that the second one's schema is different, as we've specified it.)*

Find ids of sailors who are older than 30.0 or who have a rating under 8 and are named "Horatio"

Sailors(sid, sname, rating, age) Reserves(sid, bid, day)
Boats(bid, bname, color)

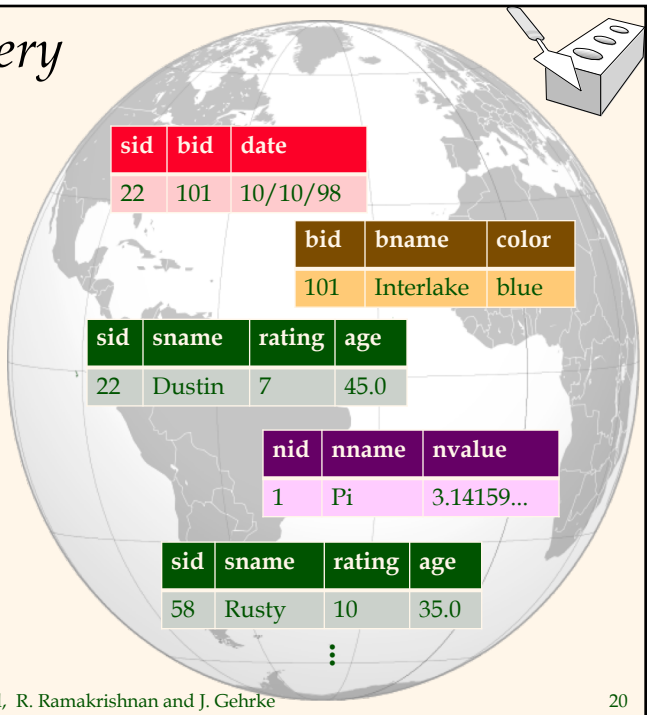
$$\{ t(sid) \mid \exists s \in Sailors ((s.age > 30.0 \vee (s.rating < 8 \wedge s.sname = \text{“Horatio”})) \wedge t.sid = s.sid) \}$$

- Things to notice:
  - Again, how result schema and values are specified
  - Use of Boolean formula to specify the query constraints
  - *Highly declarative nature of this form of query language!*

Ex: TRC Query Semantics

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5



sid	bid	date
22	101	10/10/98

bid	bname	color
101	Interlake	blue

sid	sname	rating	age
22	Dustin	7	45.0

nid	nname	nvalue
1	Pi	3.14159...

sid	sname	rating	age
58	Rusty	10	35.0

To Be Continued...