

Recommended Readings

- Chapter (complete book)
- Chapter 4
 - Section 4.3, 4.4
- Chapter 5
 - Section 5.3
- <http://philip.greenspun.com/sql/>
 - Simple queries

Winter 2003

2

Last Lecture

- What you have learnt
 - Simple SQL queries

```
SELECT [DISTINCT] c1, c2, ..., cm
FROM   R1, R2, ..., Rn
[WHERE condition]
```

For every combination of tuples in R_1, \dots, R_n ,
if that combination satisfies *condition* (i.e., *condition* is true for that combination),
then output c_1, \dots, c_m columns from that combination

Winter 2003

3

Last Lecture

- Make a cross product of relations in FROM clause, i.e., $R_1 \times R_2 \times \dots \times R_n$
- For every tuple t in $R_1 \times \dots \times R_n$,
If t satisfies *condition*,
Then output c_1, \dots, c_m columns of t

Winter 2003

4

Last Lecture

- Relational Algebra
 - Select
 - Project
 - Cross-Product
 - Union
 - Difference

Winter 2003

5

Today's Lecture

- Additional Operators
- Relational Algebra with Bag semantics
- Relational Calculus

Winter 2003

6

Additional Relational Operators

- The set of operators selection, projection, product, union, difference are the basic relational operators
- A query language is relationally complete if it can express any query expressible in relational algebra using these operators
- Additional relational operators include intersection, joins (conditional join (or θ -join), equijoin, natural join), division, and renaming. They are not essential but useful

Winter 2003

7

Intersection - $R \cap S$

- Returns tuples that exist in both R and S
- R and S must be union-compatible
- The output contains attributes identical to R or S
- $R \cap S = R - (R - S) = S - (S - R)$?

R

A	B	C
a ₁	b ₁	c ₁
a ₂	b ₂	c ₂

S

A	B	C
a ₁	b ₁	c ₁
a ₃	b ₃	c ₃

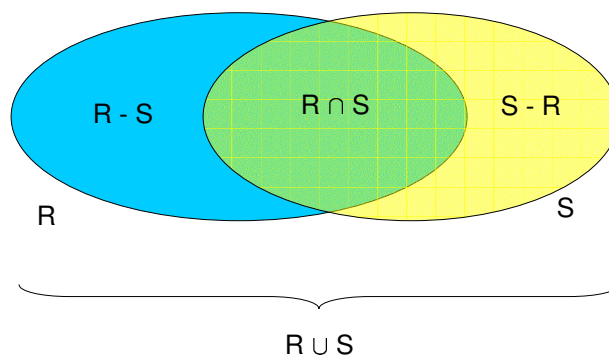
$R \cap S$

A	B	C
a ₁	b ₁	c ₁

Winter 2003

8

Venn diagram



Winter 2003

9

Conditional Join (or θ -join)

- $R \bowtie_{\theta} S$
- Meaning: take the cartesian product (or cross-product) of R and S and eliminate tuples that do not satisfy the condition θ
- Equivalent to writing it as $\sigma_{\theta}(R \times S)$
- θ is typically an equality condition or a conjunction of equality condition but it can also be others

R

A	B	C
a_1	b_1	89
a_2	b_2	78

S

D	E
100	e_1
78	e_2
39	e_3

$R \bowtie_{C < D} S$

A	B	C	D	E
a_1	b_1	89	100	e_1
a_2	b_2	78	100	e_1

Winter 2003

10

Conditional Join (or θ -join)

R

A	B	C
a_1	b_1	89
a_2	b_2	78

S

D	E
100	e_1
78	e_2
39	e_3

$R \bowtie_{C \leq D \text{ AND } C < 80} S$

A	B	C	D	E
a_2	b_2	78	78	e_2
a_2	b_2	78	100	e_1

Winter 2003

11

Equijoin

- Similar to conditional join where the condition θ contains only conjunctions of equality conditions.
i.e., θ_1 AND θ_2 AND ... AND θ_n where each θ_i is an equality condition of the form $R.name1 = S.name2$
- Resulting set of attributes similar to cross-product, but only one copy of fields for which equality is specified is retained ($R.name1$ is retained).

R		
A	B	C
a_1	b_1	89
a_2	b_2	78

S	
D	E
100	e_1
78	e_2
39	e_3

$R \bowtie_{C=D} S$

A	B	C	E
a_2	b_2	78	e_2

Winter 2003

12

Natural Join

- $R \bowtie S$ (no subscript)
- Equijoin on all common fields in R and S
- Example:
 - A and C are the common fields of $R(A, B, C, D)$ and $S(A, C, E, F)$
 - Therefore,
 - $R \bowtie S = R \bowtie_{R.A = S.A \text{ AND } R.C = S.C} S$
- The resulting relation has attributes A, B, C, D, E, F
- How do you express natural join with basic relational algebra operators?

Winter 2003

13

Example - Natural Join

- $R \bowtie_{\rho_{D \rightarrow C}}(S)$

R

A	B	C
a ₁	b ₁	89
a ₂	b ₂	78

S

D	E
100	e ₁
78	e ₂
39	e ₃

$R \bowtie S$

A	B	C	E
a ₂	b ₂	78	e ₂

Winter 2003

14

Division (Quotient) - R/S or $R \div S$

- In R/S , the set of attributes in S must be a subset of the attributes in R ; every attribute that occurs in S also occurs in R
- Let $R(A_1, \dots, A_m, B_1, \dots, B_n)$ and $S(B_1, \dots, B_n)$ be the schemas where $n > 0$, $m > 0$
- Meaning: Intuitively, if you have relations $R(A, B)$ and $S(B)$, the tuple $\langle a \rangle \in R/S$ if for every tuple $\langle b \rangle \in S$, $\langle a, b \rangle \in R$

Winter 2003

15

Example - Division

R

A	B	C
a ₁	b ₁	c ₁
a ₁	b ₂	c ₂
a ₂	b ₁	c ₁
a ₁	b ₃	c ₃
a ₄	b ₂	c ₂
a ₃	b ₂	c ₂
a ₄	b ₁	c ₁

S

B	C
b ₁	c ₁
b ₂	c ₂

R/S

A
a ₁
a ₄

Winter 2003

16

Division

- Another example
 - Enrollments(sid, cid, grade)
 - Courses(cid, cname)
 - Find all students who have taken all courses
 - $\text{Enrollments} / (\pi_{\text{cid}}(\text{Courses}))$

Winter 2003

17

Division

- Given $R(A_1, \dots, A_m, B_1, \dots, B_n)$ and $S(B_1, \dots, B_n)$, R/S consists of a set of tuples $\langle a_1, \dots, a_m \rangle$ where for every tuple $\langle b_1, \dots, b_n \rangle \in S$, the tuple $\langle a_1, \dots, a_m, b_1, \dots, b_n \rangle$ exists in R
- What happens if S is an empty relation?
- How can we express R/S with basic operators?
 - Show that

$$R/S = \pi_{A_1, \dots, A_m}(R) - \pi_{A_1, \dots, A_m}((\pi_{A_1, \dots, A_m}(R) \times S) - R)$$

Winter 2003

18

Examples

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

- Q1: Find the names of sailors who have reserved boat 103
 - $\pi_{\text{sname}}((\sigma_{\text{bid}=103} \text{Reserves}) \bowtie \text{Sailors})$
 - $\pi_{\text{sname}}(\sigma_{\text{bid}=103}(\text{Reserves} \bowtie \text{Sailors}))$
- Q2: Find colors of boats reserved by Lubber
 - $\pi_{\text{color}}(\sigma_{\text{sname}=\text{“Lubber”}}(\text{Sailors}) \bowtie \text{Reserves} \bowtie \text{Boats})$
 - Is this an equivalent query?
 - $\pi_{\text{color}}(\sigma_{\text{sname}=\text{“Lubber”}}(\text{Sailors} \bowtie \text{Reserves} \bowtie \text{Boats}))$

Winter 2003

19

Examples

- Q3: Find the names of sailors who have reserved at least one boat
 - $\pi_{\text{name}}(\text{Sailors} \bowtie \text{Reserves})$
- Q4: Find the names of sailors age over 20 who have not reserved any boats
 - $\pi_{\text{name}}(\sigma_{\text{age}>20} \text{Sailors}) - \text{Q3}$
- Q5: Find the names of sailors who have reserved a red or green boat
 - $\pi_{\text{name}}((\sigma_{\text{color}=\text{"red"}} \text{OR color}=\text{"green"}} (\text{Boat})) \bowtie \text{Sailors} \bowtie \text{Reserves})$
 - $\pi_{\text{name}}((\sigma_{\text{color}=\text{"red"}} (\text{Boat}) \cup \sigma_{\text{color}=\text{"green"}} (\text{Boat})) \bowtie \text{Sailors} \bowtie \text{Reserves})$

Winter 2003

20

Examples

- Q6: Find the names of sailors who have reserved a red and a green boat.
 - $\pi_{\text{name}}((\sigma_{\text{color}=\text{"red"}} \text{AND color}=\text{"green"}} (\text{Boat})) \bowtie \text{Sailors} \bowtie \text{Reserves})$
 - Wrong! Empty result always!
 - $Q_{\text{red}} = \pi_{\text{sid}} (\sigma_{\text{color}=\text{"red"}} (\text{Boats}) \bowtie \text{Reserves})$
 - $Q_{\text{green}} = \pi_{\text{sid}} (\sigma_{\text{color}=\text{"green"}} (\text{Boats}) \bowtie \text{Reserves})$
 - $\pi_{\text{name}} (Q_{\text{red}} \bowtie Q_{\text{green}} \bowtie \text{Sailors})$
 - Does the following query compute the same answer? Why?
 - $\pi_{\text{name}} ((Q_{\text{red}} \cap Q_{\text{green}}) \bowtie \text{Sailors})$

Winter 2003

21

Examples

- Q7: Find the names of sailors who have reserved at least 2 boats
 - $Q = \rho_{\text{sid.1} \rightarrow \text{sid1}, \text{sid.2} \rightarrow \text{sid2}, \text{bid.1} \rightarrow \text{bid1}, \text{bid.2} \rightarrow \text{bid2}} (\text{Reserves} \times \text{Reserves})$
 - $\pi_{\text{sname}} (\sigma_{\text{sid1} = \text{sid2} \text{ AND } \text{bid1} \neq \text{bid2}} (Q) \infty \text{Sailors})$
- Can you write a query that finds the names of sailors who have reserved exactly two boats?

Winter 2003

22

Things to ponder

- When is $R \infty S = R \cap S$?
- When is $R \infty S = R \times S$?
- When is $R \infty S \subseteq R$?
- Given that the queries are equivalent? Which is better in terms of efficiency?
 - $\sigma_{\text{age}>20} (\text{Sailors} \infty \text{Reserves})$
 - $\sigma_{\text{age}>20} (\text{Sailors}) \infty \text{Reserves}$
- How can a basic SQL query be translated into a relational algebra query and how can a relational algebra query consisting of select, project, and product operators be translated into SQL?
- Is every relational operator in $\sigma, \pi, \times, \cup, -$ necessary? Can one be expressed in terms of the rest?

Winter 2003

23

Summary

- Relational algebra is not as generic and powerful as general purpose programming languages
 - Can we write a recursive query in relational algebra?
- Not intended for calculations that are too complex
- However, the query language is good enough for writing most practical queries and nice properties exist for relational algebra
 - optimization rules
 - operators can be efficiently implemented
- SQL is often translated into possibly many different but equivalent relational algebra queries. The optimizer picks the most efficient one to execute
- A good balance between efficiency and expressiveness

Winter 2003

24

Relational Algebra with Bag Semantics

- Recall

```
SELECT age
FROM Students
```

returns the result { 21, 21, 20, 19 }

Students

Sid	Name	Age
1124	John	21
2187	Ann	20
8992	John	21
2346	Jane	19

- The evaluation engine often does not remove duplicates unless it is explicitly instructed to do so through the DISTINCT keyword
- Why do we use bag semantics?
 - Removing duplicates requires a sort followed by a scan through the sorted results
 - Sometimes it is necessary to retain the multiset result for certain calculations. E.g., find the total age of all students

Winter 2003

25

Relational Algebra with Bag Semantics

- Select, Project
 - Duplicates are not eliminated
 - A tuple that occurs n times in input occurs n times in the output
- Product: $R \times S$
 - Suppose $\langle x_1, \dots, x_p \rangle$ occurs m times in R and a tuple $\langle y_1, \dots, y_q \rangle$ occurs n times in S , then $\langle x_1, \dots, x_p, y_1, \dots, y_q \rangle$ occurs $m \times n$ times in $R \times S$
- Union: $R \cup S$
 - Duplicates are not eliminated
 - If the same tuple occurs m times in R and n times in S , then it occurs $m+n$ times in $R \cup S$
- Difference: $R - S$
 - Duplicates are not eliminated
 - Suppose a tuple t occurs m times in R and n times in S , then
 - T occurs $\max(0, m-n)$ times in $R - S$

Winter 2003

26

Relational Calculus

- Relational algebra – an operational formalism for expressing queries
- Relational calculus is another formalism for the relational model
 - Describes the desired result without specifying how to compute it
 - a declarative language
 - Logical formalism for expressing queries based on first-order formulas that describes the desired result
- Big influence on the design of languages such as SQL or QBE (Query by Example)
- Tuple Relational Calculus (TRC)
- Domain Relational Calculus (DRC)

Winter 2003

27

Propositional Logic (Quick review)

Recall Propositional Logic:

- An infinite set of propositional variables: x, y, z, \dots
- Propositional constants: true, false
- Connectives: $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$
- Variables are assigned with truth values through a truth assignment function ϕ
- Well-Formed Formulas: $x \wedge z, (x \vee z \wedge (\neg y))$
- Truth value defined on the structure of the formula

Winter 2003

28

Predicate Logic (Quick review)

- More general than propositional logic
- The syntax of predicate logic starts with variables, constants and predicates that can be built using a collection of boolean-valued operators (boolean expressions)
 - Constant symbols: E.g., Mary, 34, true, ...
 - Variables: $x, y, z \dots$
 - Predicate Symbols.
 - E.g., Person
 - maps tuples of elements to true or false
 - Person(12-33, John, 34)
 - Variables range over elements in a universe using quantifiers \exists and \forall

Winter 2003

29

Predicate Logic (Quick review)

- Atoms. (evaluates to true or false)
 - n-ary predicate of n terms E.g. Person(12-33, John, 34), Person(x, y, z)
 - If P and Q are atoms, so is $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$
- Sentence: atom or $(\exists x P)$ or $(\forall x P)$ where P is a sentence
- Well-formed formulas: a sentence that does not contain any “free” variable (i.e., all variables are “bound”)

• $\exists x P(x, y)$ vs $\exists x \forall y P(x, y)$

free

Winter 2003

30

Tuple Relational Calculus

- Assume the following binary predicates are available and denote *op* as one of the following predicates:
 - $<, =, >, \leq, \neq, \geq$
- Atomic formula:
 - $x \in Rel$ where x is a tuple variable, $x.a \text{ op } y.b$, $x.a \text{ op constant}$, or $constant \text{ op } x.a$
- Formula: (P and Q are formulas, P(x) is a formula)
 - An atomic formula
 - $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$
 - $\exists x(P(x))$ where x is a **tuple** variable
 - “there exists a tuple t which x binds to such that P(t) holds”
 - $\forall x(P(x))$ where x is a **tuple** variable
 - “for every tuple t which x binds to, P(t) holds”

Variable gets bound to a tuple

Winter 2003

31

Semantics

- A TRC query has the form $\{ x \mid P(x) \}$
 - This query is well-formed when x is the only free variable that occurs in the formula P
- Semantics:
 - A TRC query $\{ x \mid P(x) \}$ consists of all tuples t that satisfy the TRC formula $P(t)$

Winter 2003

32

Examples

Sailors (sid: integer, sname: string, rating: integer, age: real)

Boats (bid: integer, bname: string, color: string)

Reserves (sid: integer, bid: integer, day: date)

- $\{ y \mid \exists x \in \text{Sailors}, x.\text{age} > 30 (y.\text{name} = x.\text{name} \wedge y.\text{rating} = x.\text{rating}) \}$
 - The names and ratings of sailors whose age is greater than 30
 - y is the only free variable, x is a bound variable
 - For every tuple t in Sailors, bind x to t ,
 - if the age of x is greater than 30
 - return $y = \langle \text{name: } x.\text{name}, \text{rating: } x.\text{rating} \rangle$
 - Bind y to a tuple t (with name and rating columns)
 - if the statement $P(t)$ is true, return y

Winter 2003

33

Examples

- $\{ z \mid \exists x \in \text{Sailors } \exists y \in \text{Reserves } (x.\text{sid}=y.\text{sid} \wedge y.\text{bid}=103 \wedge z.\text{sname} = x.\text{sname})\}$
 - Find the names of sailors who have reserved boat 103
- $\{ w \mid \exists x \in \text{Sailors } \forall y \in \text{Boats } \exists z \in \text{Reserves } x.\text{sid}=z.\text{sid} \wedge z.\text{bid}=y.\text{bid} \wedge w.\text{name} = x.\text{name} \}$
 - What does the above query compute?
 - Find the names of sailors who have reserved all boats

Winter 2003

34

Examples

- Swap the order:
 - $\{ w \mid \exists x \in \text{Sailors } \exists z \in \text{Reserves } \forall y \in \text{Boats } x.\text{sid}=z.\text{sid} \wedge z.\text{bid}=y.\text{bid} \wedge w.\text{name} = x.\text{name} \}$
 - What does this query mean? If there is more than one boat in Boats relation, the result of the query is always empty
- $\{ w \mid \exists x \in \text{Sailors } \forall y \in \text{Boats } (y.\text{color}=\text{"red"} \rightarrow \exists z \in \text{Reserves } x.\text{sid}=z.\text{sid} \wedge z.\text{bid}=y.\text{bid} \wedge x.\text{name}=w.\text{name})\}$
 - Find the names of sailors who have reserved all red boats

Winter 2003

35

Domain Relational Calculus

- Instead of tuple variables, we have domain variables
- A domain variable ranges over the elements in the domain of some attribute.
- A DRC query has the form
 - $\{ \langle x_1, \dots, x_n \rangle \mid P(x_1, \dots, x_n) \}$
 - x_i is either a domain variable or a constant
 - $P(x_1, \dots, x_n)$ denotes a DRC formula
 - The only free variables are x_1, \dots, x_n
- Semantics: The set of all $\langle a_1, \dots, a_n \rangle$ tuples for $P(a_1, \dots, a_n)$ evaluates to true (a_i s are constants)

Winter 2003

36

DRC formula

- $\langle x_1, \dots, x_n \rangle \in Rel$ where Rel is a relation with n attributes and x_i is either a variable or a constant
- $x \text{ op } y$
- $x \text{ op constant}$, or $\text{constant op } y$
- A DRC formula is
 - An atomic formula
 - $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$
 - $\exists x(P(x))$ where x is a **domain** variable
 - “there exists a value c which x binds to such that $P(c)$ holds”
 - $\forall x(P(x))$ where x is a **domain** variable
 - “for every value c which x binds to, $P(c)$ holds”

Variable gets bound to a value of some domain

Winter 2003

37

Examples

- Find all sailors older than 30.
 - $\{ \langle s,n,r,a \rangle \mid \langle s,n,r,a \rangle \in \text{Sailors } a > 30 \}$
- Find the names of all sailors older than 30
 - $\{ \langle n \rangle \mid \exists s,r,a \langle s,n,r,a \rangle \in \text{Sailors } a > 30 \}$
- Find the names of sailors who have reserved boat 103
 - $\{ \langle n \rangle \mid \exists s,r,a \langle s,n,r,a \rangle \in \text{Sailors } \exists s',b,d \langle s',b,d \rangle \in \text{Reserves } s=s' \wedge b=103 \}$
 - $\{ \langle n \rangle \mid \exists s,r,a \langle s,n,r,a \rangle \in \text{Sailors } \exists d \langle s,103,d \rangle \in \text{Reserves} \}$
- Find the sailors who have reserved all red boats
 - $\{ n \mid \exists s,r,a \langle s,n,r,a \rangle \in \text{Sailors } \forall b,bn,c \langle b,bn,c \rangle \in \text{Boats} (c=\text{"red"} \rightarrow \exists si,bi,d \langle si,bi,d \rangle \in \text{Reserves } s=si \wedge bi=b) \}$

Winter 2003

38

Safety

$\{ \langle s,n,r,a \rangle \mid \neg (\langle s,n,r,a \rangle \in \text{Sailors}) \}$

- The set of all tuples that are **not** in the Sailors relation. **Infinite!** (in the context of infinite domains, e.g., Age)
- A query is **safe** if it produces a finite answer under all possible instances of the input relations
- A program that evaluates an unsafe query will not terminate
- Consider only safe queries.

Winter 2003

39