Chapter 6: Relational Algebra and Relational Calculus

Last Lecture:

- More Relational Algebra
- Division Operator
- Null Values and Bags in Relational Algebra
- Introduction to Relational Calculus

Today:

- Tuple Relational Calculus
- Domain Relational Calculus

Tuple Relational Calculus

- A nonprocedural query language, where each query is of the form

  $$\{t \mid P(t)\}$$

- It is the set of all tuples *t* such that predicate *P* is true for *t*
- *t* is a *tuple variable*, $t[A]$ denotes the value of tuple *t* on attribute *A*
- $t \in r$ denotes that tuple *t* is in relation *r*
- *P* is a *formula* similar to that of the predicate calculus

Predicate Calculus Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., $<$, $\leq$, $=$, $\neq$, $>$, $\geq$)
3. Set of connectives: and ($\wedge$), or (v), not ($\neg$)
4. Implication ($\Rightarrow$): x $\Rightarrow$ y, if x is true, then y is true

   $x \Rightarrow y \equiv \neg x \vee y$

5. Set of quantifiers:
   - **$\exists\, t \in r\, (Q\, (t\,)) \equiv$ "there exists" a tuple in *t* in relation *r* such that predicate *Q* (*t*) is true**
   - **$(\exists\, t)\ \ (t \in r \wedge Q\, (t\,)) \equiv$ "there exists" a tuple in *t* such that predicate *Q* (*t*) is true**
   - **$\forall t \in r\, (Q\, (t\,)) \equiv Q$ is true "for all" tuples *t* in relation *r***
   - **$(\forall t\,)\ (t \in r \Rightarrow Q\, (t\,)) \equiv Q$ is true "for all" tuples *t in relation r***

Don't use the notation in blue

Example Queries

Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000

$$\{t \mid t \in instructor \land t\,[salary\,] > 80000\}$$

As in the previous query, but output only the *ID* attribute value

$$\{t \mid (\exists\, s)\ (s \in \text{instructor} \land t\,[ID\,] = s\,[ID\,] \land s\,[salary\,] > 80000)\}$$

Notice that a relation on schema (*ID*) is implicitly defined by the query

Example Queries

Find the names of all instructors whose department is in the Watson building

$\{t \mid \exists s \in instructor\ (t[name] = s[name]$
$\wedge \exists u \in department\ (u[dept\_name] = s[dept\_name] \wedge u[building] =$ "Watson" $))\}$

Find the set of all courses taught in the Fall 2015 semester, **or** in the Spring 2016 semester, or both

$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id] \wedge$
$s[semester] = \text{"Fall"} \wedge s[\text{year}] = 2015$
$\vee \exists u \in section\ (t[course\_id] = u[course\_id] \wedge$
$u[semester] = \text{"Spring"} \wedge u[\text{year}] = 2016)\}$

Example Queries

- **Find the set of all courses taught in the Fall 2015 semester, and in the Spring 2016 semester**

$\{t \mid (\exists s)\ (s \in section \land t\ [course\_id\ ] = s\ [course\_id\ ] \land$
$s\ [semester] = \text{"Fall"} \land s\ [\text{year}] = 2015$
$\land\ (\exists u)\ (u \in section \land t\ \ [course\_id\ ] = u\ [course\_id\ ] \land$
$u\ [semester] = \text{"Spring"} \land u\ [\text{year}] =$
$2016)\}$

- **Find the set of all courses taught in the Fall 2015 semester, but not in the Spring 2016 semester**

$\{t \mid (\exists s)\ (s \in section \land t\ [course\_id\ ] = s\ [course\_id\ ] \land$
$s\ [semester] = \text{"Fall"} \land s\ [\text{year}] = 2015$
$\land \neg\ (\exists u)\ (u \in section \land t\ \ [course\_id\ ] = u\ [course\_id\ ] \land$
$u\ [semester] = \text{"Spring"} \land u\ [\text{year}] =$
$2016)\}$

Safety of Expressions

- It is possible to write tuple calculus expressions that generate infinite relations.
- For example, $\{ t \mid \neg\, t \in r \}$ results in an infinite relation
- To guard against the problem, we restrict the set of allowable expressions to safe expressions.
- An expression $\{t \mid P(t)\}$ in the tuple relational calculus is *safe* if every component of *t* appears in one of the relations, tuples, or constants that appear in *P*
  - NOTE: this is more than just a syntax condition.
  - E.g. $\{ t \mid t[A] = 5 \lor \textbf{true} \}$ is not safe --- it defines an infinite set with attribute values that do not appear in any relation or tuples or constants in *P*.

Universal Quantification

- Find all students who have taken all courses offered in the Biology department

$$\{t \mid (\exists\, r)\ (r \in student \land t[ID] = r[ID] \land$$
$$(\forall\, u)\ (u \in course \land u[dept\_name] = \text{“Biology”} \Rightarrow$$
$$(\exists\, s)(s \in takes \land t[ID] = s[ID] \land s[course\_id] = u[course\_id]))\}$$

- Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.

Producer (*id*: integer, *name*: string, *age*: integer)

Manage (*pid*: integer, *mid*: integer)

Movie (*mid*: integer, *budget*: real, *managerid*: integer)

a) Find the names of producers who manage at least one movie with budget larger than 1 million

b) Find the names of producers who manage all movies with budget larger than 1 million

c) Find the names of producers who manage only movies with budget larger than 1 million

Producer (<u>*id*: integer</u>, *name*: string, *age*: integer)

Manage (<u>*pid*: integer, *mid*: integer</u>)

Movie (<u>*mid*: integer</u>, *budget*: real, *managerid*: integer)

a) Find the names of producers who manage at least one movie with budget larger than 1 million

$$\{t \mid (\exists s)\ (s \in Producer \wedge t\,[name] = s\,[name]$$

$$\wedge$$

$$\wedge\ (\exists u)\ (u \in Movie \wedge s[id] = u\,[managerid] \wedge$$
$$u\,[budget] > 1000000)$$

Producer (<u>*id*: integer</u>, *name*: string, *age*: integer)

Manage (<u>*pid*: integer, *mid*: integer</u>)

Movie (<u>*mid*: integer</u>, *budget*: real, *managerid*: integer)

a) Find the names of producers who manage at least one movie with budget larger than 1 million

b) Find the names of producers who manage all movies with budget larger than 1 million

$$\{t \mid (\exists s)\ (s \in Producer \wedge t[name] = s[name] \wedge (\forall u)\ (u \in Movie \wedge u[\text{budget}] > 1000000 \Rightarrow s[\text{id}] = u[\text{managerid}])$$

Producer (*id*: integer, *name*: string, *age*: integer)

Manage (*pid*: integer, *mid*: integer)

Movie (*mid*: integer, *budget*: real, *managerid*: integer)

a) Find the names of producers who manage at least one movie with budget larger than 1 million

b) Find the names of producers who manage all movies with budget larger than 1 million

c) Find the names of producers who manage only movies with budget larger than 1 million

$$\{t \mid (\exists s)\ (s \in Producer \wedge t\,[name] = s\,[name]$$

$$\wedge\ (\forall\ u)\ (u \in Movie \wedge \mathrm{u[managerid] = s[id]}$$
$$\Rightarrow \mathrm{u[budget]} > 1000000)$$

Domain Relational Calculus

Domain Relational Calculus

- A nonprocedural query language equivalent in power to the tuple relational calculus
- Each query is an expression of the form:

$$\{ < x_1, x_2, \ldots, x_n > \mid P\,(x_1, x_2, \ldots, x_n)\}$$

  - $x_1, x_2, \ldots, x_n$ represent domain variables
  - P represents a formula similar to that of the predicate calculus

Example Queries

- Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000

  $\{< i, n, d, s> \mid < i, n, d, s> \in instructor \wedge s > 80000\}$

- As in the previous query, but output only the *ID* attribute value

  $\{< i> \mid < i, n, d, s> \in instructor \wedge s > 80000\}$

- Find the names of all instructors whose department is in the Olin building

  $\{< n > \mid (\exists\, i, d, s)(< i, n, d, s > \in instructor$
  $\wedge (\exists\, \text{b, a}) (< d, b, a> \in department \wedge b = \text{“Olin”}))\}$

Example Queries

- Find the set of all courses taught in the Fall 2015 semester, or in the Spring 2016 semester, or both

$$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, t> \in section \wedge s = \text{“Fall”} \wedge y = 2015\ )$$
$$\vee \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, t> \in section\ ] \wedge s = \text{“Spring”} \wedge y = 2016)\}$$

This case can also be written as

$$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, t> \in section \wedge$$
$$(\ (s = \text{“Fall”} \wedge y = 2015)\ \vee (s = \text{“Spring”} \wedge y = 2016))\}$$

- Find the set of all courses taught in the Fall 2015 semester, and in the Spring 2016 semester

$$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, t> \in section \wedge s = \text{“Fall”} \wedge y = 2015\ )$$
$$\wedge \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, t> \in section\ ] \wedge s = \text{“Spring”} \wedge y = 2016)\}$$

Safety of Expressions

The expression:

$\{ < x_1, x_2, \ldots, x_n > \mid P\,(x_1, x_2, \ldots, x_n\,)\}$

is **safe** if all of the following hold:

- All values that appear in tuples of the expression are values from *dom* (P) (that is, the values appear either in P or in a tuple of a relation mentioned in P).
- For every "there exists" subformula of the form $\exists\, x\ (P_1(x\ ))$, the subformula is true if and only if there is a value of x in *dom* (P_1)such that $P_1(x\ )$ is true.
- For every "for all" subformula of the form $\forall_x\ (P_1\ (x\ ))$, the subformula is true if and only if $P_1(x\ )$ is true for all values x from *dom* (P_1).

Universal Quantification

- Find all students who have taken all courses offered in the Biology department

$$\{< i > \mid \exists\, n, d, tc\ (< i, n, d, tc > \in student\ \wedge$$
$$(\forall\, ci, ti, dn, cr\ (< ci, ti, dn, cr > \in course \wedge dn = \text{“Biology”}$$
$$\Rightarrow\ \exists\, si, se, y, g\ (<i, ci, si, se, y, g> \in\ takes\ ))\}$$

Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.

* Above query fixes bug in page 246, last query

End of Chapter 6

Why Bags?

- SQL, the most important query language for relational databases, is actually a bag language.
- Some operations, like projection, are more efficient on bags than sets.

Operations on Bags

- Selection applies to each tuple, so its effect on bags is like its effect on sets.
- Projection also applies to each tuple, but as a bag operator, we do not eliminate duplicates.
- Products and joins are done on each pair of tuples, so duplicates in bags have no effect on how we operate.

Example: Bag Selection

R(A,	B)
1	2
5	6
1	2

$\sigma_{A+B<5}(R) =$

A	B
1	2
1	2

Example: Bag Projection

R(A,	B)
1	2
5	6
1	2

π_A (R) =

A
1
5
1

Example: Bag Product

R(A, B)

1	2
5	6
1	2

S(B, C)

3	4
7	8

R X S =

A	R.B	S.B	C
1	2	3	4
1	2	7	8
5	6	3	4
5	6	7	8
1	2	3	4
1	2	7	8

Example: Bag Theta-Join

R(A, B)

1	2
5	6
1	2

S(B, C)

3	4
7	8

$R \bowtie_{R.B<S.B} S =$

A	R.B	S.B	C
1	2	3	4
1	2	7	8
5	6	7	8
1	2	3	4
1	2	7	8

Bag Union

- An element appears in the union of two bags the **sum** of the number of times it appears in each bag.
- Example: {1,2,1} U {1,1,2,3,1} = {1,1,1,1,1,2,2,3}

Bag Intersection

- An element appears in the intersection of two bags the **minimum** of the number of times it appears in either.
- Example: $\{1,2,1,1\} \cap \{1,2,1,3\} = \{1,1,2\}$.

Bag Difference

- An element appears in the difference $A - B$ of bags as many times as it appears in A, **minus** the number of times it appears in B.
  - But never less than 0 times.
- Example: {1,2,1,1} – {1,2,3} = {1,1}.

Beware: Bag Laws != Set Laws

- Some, but *not all* algebraic laws that hold for sets also hold for bags.
- Example: the commutative law for union

  ($R \cup S = S \cup R$) *does* hold for bags.

  - Since addition is commutative, adding the number of times x appears in R and S doesn't depend on the order of R and S.

Example: A Law That Fails

- Set union is *idempotent*, meaning that $S \cup S = S$.

- However, for bags, if x appears n times in S, then it appears $2n$ times in $S \cup S$.

- Thus $S \cup S$!= S in general.
  - e.g., {1} ∪ {1} = {1,1} != {1}.

Bank Example

- Borrower (customer_name, loan_number, branch_name)
- Depositor(customer_name, account_number, branch_name)
- Loan (loan_number, amount, loan_type)
- Account (account_number, balance, account_type)
- Branch(branchname, branch_city, address, phone)

Example Queries

Find the names of all customers who have a loan and an account at bank.

$$\Pi_{customer\_name}\,(borrower) \cap \Pi_{customer\_name}\,(depositor)$$

Find the name of all customers who have a loan at the bank and the loan amount

$$\Pi_{customer\_name,\ loan\_number,\ amount}\,(borrower \bowtie loan)$$

Example Queries

- Find all customers who have an account from at least the “Downtown” and the Uptown” branches.

Query 1

$$\Pi_{customer\_name}\,(\sigma_{branch\_name\,=\,\text{“Downtown”}}\,(depositor \bowtie account)) \cap$$

$$\Pi_{customer\_name}\,(\sigma_{branch\_name\,=\,\text{“Uptown”}}\,(depositor \bowtie account))$$

Query 2

$$\Pi_{customer\_name,\ branch\_name}\,(depositor \bowtie account)$$

$$\div\ \rho_{temp(branch\_name)}\,(\{(\text{“Downtown”}),\ (\text{“Uptown”})\})$$

Note that Query 2 uses a constant relation.

Bank Example Queries

- Find all customers who have an account at all branches located in Brooklyn city.

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account)$$
$$\div\ \Pi_{branch\_name}(\sigma_{branch\_city\ =\ \text{“Brooklyn”}}(branch))$$

Relational Algebra on Bags

- A *bag* (or *multiset*) is like a set, but an element may appear more than once.
- Example: {1,2,1,3} is a bag.
- Example: {1,2,3} is also a bag that happens to be a set.