Chapter 6: Formal Relational Query Languages

Database System Concepts, 6th Ed.

Chapter 6: Formal Relational Query Languages

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus

Relational Algebra

- Procedural language
- Six basic operators
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result.

Select Operation

- Notation: $\sigma_p(r)$
- p is called the **selection predicate**
- Defined as:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

- Example of selection:

$$\sigma_{dept\_name=\text{“}Physics\text{”}}(instructor)$$

Project Operation

- Notation: $\prod_{A_1, A_2, \ldots, A_k}(r)$

  where A_1, A_2 are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *dept_name* attribute of *instructor*

$$\prod_{ID,\ name,\ salary}(instructor)$$

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.
  1. *r, s* must have the *same* **arity** (same number of attributes)
  2. The attribute domains must be **compatible** (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)
- Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

$$\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \wedge year=2009}(section)) \cup \Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \wedge year=2010}(section))$$

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- Set differences must be taken between **compatible** relations.
  - r and s must have the same arity
  - attribute domains of r and s must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \wedge year=2009}(section)) -$$

$$\Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \wedge year=2010}(section))$$

Cartesian-Product Operation

- Notation $r \times s$
- Defined as:

$$r \times s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$$

- Relations r, s:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- $r \times s$:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Example Queries

- Find the IDs of all instructors in the Physics department, along with the *course_id* of all courses they have taught

instructor (ID, dept_name)

teaches (ID, course_id)

Example Queries

- Find the IDs of all instructors in the Physics department, along with the *course_id* of all courses they have taught

instructor (ID, dept_name)
teaches (ID, course_id)

```
select instructor.ID, course_id
from instructor, teaches
where dept_name = "Physics" and instructor.ID=teaches.ID
```

Example Queries

- Find the IDs of all instructors in the Physics department, along with the *course_id* of all courses they have taught

instructor (ID, dept_name)
teaches (ID, course_id)

```
select instructor.ID, course_id
from instructor, teaches
where dept_name = "Physics" and instructor.ID=teaches.ID
```

- Query 1

$$\Pi_{instructor.ID,course\_id}(\sigma_{dept\_name="Physics" \wedge instructor.ID=teaches.ID}(instructor \times teaches))$$

Example Queries

- Find the IDs of all instructors in the Physics department, along with the *course_id* of all courses they have taught

instructor (ID, dept_name)
teaches (ID, course_id)

```
select instructor.ID, course_id
from instructor, teaches
where dept_name = "Physics" and instructor.ID=teaches.ID
```

- Query 1

$$\Pi_{instructor.ID,course\_id}(\sigma_{dept\_name="Physics" \wedge instructor.ID=teaches.ID}(instructor \times teaches))$$

- Query 2

$$\Pi_{instructor.ID,course\_id}(\sigma_{instructor.ID=teaches.ID}(\sigma_{dept\_name="Physics"}(instructor) \times teaches))$$

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- $\rho_x(E)$

  returns the expression E under the name X
- $\rho_{x(A_1, A_2, \ldots, A_n)}(E)$

  returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

- find the instructors receiving salary higher than some other instructor.

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- $\rho_x(E)$

  returns the expression E under the name X
- $\rho_{x(A_1, A_2, \ldots, A_n)}(E)$

  returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

- find the instructors receiving salary higher than some other instructor.

name	*salary*
Crick	72000
Katz	75000
Srinivasan	65000
Brandt	92000
Kim	80000
Wu	90000
Singh	80000
El Said	60000
Califieri	62000
Mozart	40000
Gold	87000
Einstein	95000

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- $\rho_x(E)$

  returns the expression E under the name X
- $\rho_{x(A_1, A_2, \ldots, A_n)}(E)$

  returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

- find the instructors receiving salary higher than some other instructor.

name	*salary*
Crick	72000
Katz	75000
Srinivasan	65000
Brandt	92000
Kim	80000
Wu	90000
Singh	80000
El Said	60000
Califieri	62000
Mozart	40000
Gold	87000
Einstein	95000

name	*salary*
Crick	72000
Katz	75000
Srinivasan	65000
Brandt	92000
Kim	80000
Wu	90000
Singh	80000
El Said	60000
Califieri	62000
Mozart	40000
Gold	87000
Einstein	95000

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- $\rho_x(E)$

  returns the expression E under the name X
- $\rho_{x(A_1, A_2, \ldots, A_n)}(E)$

  returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

- find the instructors receiving salary higher than some other instructor.
  - $\Pi_{instructor.name}(\sigma_{instructor.salary > d.salary}(instructor \times \rho_d(instructor)))$

name	*salary*
Crick	72000
Katz	75000
Srinivasan	65000
Brandt	92000
Kim	80000
Wu	90000
Singh	80000
El Said	60000
Califieri	62000
Mozart	40000
Gold	87000
Einstein	95000

name	*salary*
Crick	72000
Katz	75000
Srinivasan	65000
Brandt	92000
Kim	80000
Wu	90000
Singh	80000
El Said	60000
Califieri	62000
Mozart	40000
Gold	87000
Einstein	95000

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- $\rho_x(E)$

  returns the expression E under the name X
- $\rho_{x(A_1, A_2, \ldots, A_n)}(E)$

  returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

- find the instructors receiving salary higher than some other instructor.
  - $\Pi_{instructor.name}(\sigma_{instructor.salary > d.salary}(instructor \times \rho_d(instructor)))$
- How can we find the highest salary?

name	*salary*
Crick	72000
Katz	75000
Srinivasan	65000
Brandt	92000
Kim	80000
Wu	90000
Singh	80000
El Said	60000
Califieri	62000
Mozart	40000
Gold	87000
Einstein	95000

name	*salary*
Crick	72000
Katz	75000
Srinivasan	65000
Brandt	92000
Kim	80000
Wu	90000
Singh	80000
El Said	60000
Califieri	62000
Mozart	40000
Gold	87000
Einstein	95000

Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Assignment
- Outer join

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

Natural-Join Operation

- Notation: $r \bowtie s$
- Let r and s be relations on schemas R and S respectively.
  Then, $r \bowtie s$ is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
    - t has the same value as t_r on r
    - t has the same value as t_s on s

Natural Join Example

- Relation *r* on relation schema *R*, relation *s* on relation schema *S*:

instructor_id	department_id
A	1
B	1
C	2

department_id	college
1	X
2	Y

- Natural Join
  - r ⋈ s

instructor_id	department_id	college
A	1	X
B	1	X
C	2	Y

- *r* ⋈ *s* is defined as:

$$\Pi_{r.instructor\_id,\ r.department\_id,\ s.college}\ (\sigma_{r.department\_id = s.department\_id}\ (r \times s))$$

Properties of Natural Join

- Natural join is associative
  - $(instructor \bowtie teaches) \bowtie course$ is equivalent to
    $instructor \bowtie (teaches \bowtie course)$
- Natural join is commutative
  - $instruct \bowtie teaches$ is equivalent to
    $teaches \bowtie instructor$

Theta Join

- The **theta join** operation $r \bowtie_\theta s$ is defined as
  - $r \bowtie_\theta s = \sigma_\theta (r \times s)$

r

A	B
□	1
▲	2

s

B	C
1	✻
2	○

- $r \bowtie_{r.B \le s.B} s$

A	r.B	s.B	C
□	1	1	✻
□	1	2	○
▲	2	2	○

Assignment Operation

- The assignment operation (←) provides a convenient way to express complex queries.
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  - Assignment must always be made to a temporary relation variable.
- Example: $temp1 \leftarrow \prod_{R\text{-}S} (r)$

  $temp2 \leftarrow \prod_{R\text{-}S} ((temp1 \times s) - \prod_{R\text{-}S} (r))$

  $result = temp1 - temp2$

Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist

Outer Join – Example

- Relation *instructor1*

ID	*name*	*dept_name*
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music

- Relation *teaches1*

ID	*course_id*
10101	CS-101
12121	FIN-201
76766	BIO-101

Outer Join – Example

- Join

  instructor ⋈ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201

- Left Outer Join

  instructor ⟕ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	*null*

Outer Join – Example

- Right Outer Join

  instructor ⟖ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
76766	null	null	BIO-101

- Full Outer Join

  instructor ⟗ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	*null*
76766	null	null	BIO-101

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

  $$t \times s \subseteq r$$

- E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

  $$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$$

  then $r \div s$ gives us students who have taken all courses in the Biology department

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

  $$t \times s \subseteq r$$

- E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

  $$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$$

  then $r \div s$ gives us students who have taken all courses in the Biology department

t

ID
1
2

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

  $$t \times s \subseteq r$$

- E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

  $$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$$

  then $r \div s$ gives us students who have taken all courses in the Biology department

t

ID
1
2

s

course_ID
A
B

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

  $$t \times s \subseteq r$$

- E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

  $$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$$

  then $r \div s$ gives us students who have taken all courses in the Biology department

t

ID
1
2

s

course_ID
A
B

r = t x s

ID	course_ID
1	A
1	B
2	A
2	B

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

  $$t \times s \subseteq r$$

- E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

  $$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$$

  then $r \div s$ gives us students who have taken all courses in the Biology department

t

ID
1
2

s

course_ID
A
B

r = t x s

ID	course_ID
1	A
1	B
2	A
2	B

$r \div s$?

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

  $$t \times s \subseteq r$$

- E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

  $$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$$

  then $r \div s$ gives us students who have taken all courses in the Biology department

t

ID
1
2

s

course_ID
A
B

r = t x s

ID	course_ID
1	A
1	B
2	A
2	B

r ÷ s?

ID
1
2

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

  $$t \times s \subseteq r$$

- E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

  $$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$$

  then $r \div s$ gives us students who have taken all courses in the Biology department

t

ID
1
2

s

course_ID
A
B

r = t x s

ID	course_ID
1	A
1	B
2	A
2	B

r

ID	course_ID
1	A
1	B
2	A

r ÷ s?

ID
1
2

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

  $t \times s \subseteq r$

- E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

  $s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$

  then $r \div s$ gives us students who have taken all courses in the Biology department

t

ID
1
2

s

course_ID
A
B

r = t x s

ID	course_ID
1	A
1	B
2	A
2	B

r

ID	course_ID
1	A
1	B
2	A

r ÷ s?

ID
1
2

r ÷ s?

Division Operator

- Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

  $t \times s \subseteq r$

- E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

  $s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$

  then $r \div s$ gives us students who have taken all courses in the Biology department

t

ID
1
2

s

course_ID
A
B

r = t x s

ID	course_ID
1	A
1	B
2	A
2	B

r ÷ s?

ID
1
2

r

ID	course_ID
1	A
1	B
2	A

r ÷ s?

ID
1
2

vs

ID
1

Extended Relational-Algebra-Operations

- Generalized Projection
- Aggregate Functions

Generalized Projection

- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\prod_{F_1, F_2, \ldots, F_n}(E)$$

- E is any relational-algebra expression
- Each of F_1, F_2, …, F_n are are arithmetic expressions involving constants and attributes in the schema of E.
- Given relation *instructor(ID, name, dept_name,* salary) where salary is annual salary, get the same information but with monthly salary

$$\prod_{ID,\ name,\ dept\_name,\ \mathbf{salary/12}}(instructor)$$

Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a single value as a result.

  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

- **Aggregate operation** in relational algebra

$$_{G_1,G_2,\ldots,G_n}\mathcal{G}_{F_1(A_1),F_2(A_2,\ldots,F_n(A_n)}(E)$$

  E is any relational-algebra expression

  - G_1, G_2 …, G_n is a list of attributes on which to group (can be empty)
  - Each F_i is an aggregate function
  - Each A_i is an attribute name

- Note: Some books/articles use γ instead of $\mathcal{G}$ (Calligraphic G)

Aggregate Operation – Example

- Relation *r*:

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

- $\mathcal{G}_{\mathbf{sum(c)}}(r)$

sum(*c*)
27

Aggregate Operation – Example

- Find the average salary in each department

  $dept\_name\ \mathcal{G}\ \mathbf{avg}(salary)\ (instructor)$

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Aggregate Functions (Cont.)

- Result of aggregation does not have a name
  - Can use rename operation to give it a name
  - For convenience, we permit renaming as part of aggregate operation

$$dept\_name\ \mathcal{G}\ \textbf{avg}(salary)\ \textbf{as}\ avg\_sal\ (instructor)$$

Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations can be expressed using the assignment operator
- Delete all account records in the Perryridge branch.
- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch.
- Make interest payments by increasing all balances by 5 percent.

Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations can be expressed using the assignment operator
- Delete all account records in the Perryridge branch.

$$account \leftarrow account - \sigma_{branch\_name = \text{“Perryridge”}}(account)$$

- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch.

- Make interest payments by increasing all balances by 5 percent.

Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations can be expressed using the assignment operator
- Delete all account records in the Perryridge branch.

$$account \leftarrow account - \sigma_{branch\_name = \text{“Perryridge”}}(account)$$

- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch.

$$account \leftarrow account \cup \{(\text{“A-973”}, \text{“Perryridge”}, 1200)\}$$

$$depositor \leftarrow depositor \cup \{(\text{“Smith”}, \text{“A-973”})\}$$

- Make interest payments by increasing all balances by 5 percent.

Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations can be expressed using the assignment operator
- Delete all account records in the Perryridge branch.

$$account \leftarrow account - \sigma_{branch\_name = \text{“Perryridge”}}(account)$$

- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch.

$$account \leftarrow account \cup \{(\text{“A-973”}, \text{“Perryridge”}, 1200)\}$$

$$depositor \leftarrow depositor \cup \{(\text{“Smith”}, \text{“A-973”})\}$$

- Make interest payments by increasing all balances by 5 percent.

$$account \leftarrow \prod_{account\_number,\ branch\_name,\ balance * 1.05}(account)$$

SQL and Relational Algebra

- **select** *A1, A2, .. An*
  from *r1, r2, …, rm*
  where P

  is equivalent to the following expression in multiset relational algebra

  $$\Pi_{A1, .., An} (\sigma_{P} (r1 \times r2 \times .. \times rm))$$

- **select** *A1, A2,* **sum***(A3)*
  from *r1, r2, …, rm*
  where P
  group by *A1, A2*

  is equivalent to the following expression in multiset relational algebra

  $$A1, A2\ \mathcal{G}_{\textbf{sum}(A3)} (\sigma_{P} (r1 \times r2 \times .. \times rm)))$$

Relational Algebra Summary

- Procedural language
- The operators take one or two relations as inputs and produce a new relation as a result.
- Six basic operators
  - select: $\sigma_p(r)$
  - project: $\prod_{A_1, A_2, \ldots, A_k}(r)$
  - union: $r \cup s$
  - set difference: $r - s$
  - Cartesian product: $r \times s$
  - rename: $\rho_{x(A_1, A_2, \ldots, A_n)}(E)$
- Additional Operators
  - intersection: $r \cap s$
  - natural join: $r \bowtie s$
  - theta join: $r \bowtie_\theta s = \sigma_\theta (r \times s)$
  - outer join: $r \mathbin{⟕} s$, $r \mathbin{⟖} s$, $r \mathbin{⟗} s$
  - division: $r \div s$
  - generalized projection: $\prod_{F_1, F_2, \ldots, F_n}(E)$
  - aggregation: $_{G_1, G_2, \ldots, G_n}\mathcal{G}_{F_1(A_1), F_2(A_2), \ldots, F_n(A_n)}(E)$

Tuple 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

Example Queries

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

Example Queries

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

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

Example Queries

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

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

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

Example Queries

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

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

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

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

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

instructor(name, dept_name) *department(dept_name, building)*

- Find the set of all courses taught in 2009 and 2010

section(course_id, year)

(C1, 2009)

(C2, 2009)

(C2, 2010)

Example Queries

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

instructor(name, dept_name) *department(dept_name, building)*

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

- Find the set of all courses taught in 2009 and 2010

section(course_id, year)

(C1, 2009)

(C2, 2009)

(C2, 2010)

Example Queries

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

instructor(name, dept_name)	*department(dept_name, building)*

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

- Find the set of all courses taught in 2009 and 2010

section(course_id, year)
(C1, 2009)
(C2, 2009)
(C2, 2010)

$$\{t \mid \exists s \in \textbf{section}\ (t[course\_id] = s[course\_id] \wedge s[\text{year}] = 2009 \wedge \exists u \in \textbf{section}\ (t[course\_id] = u[course\_id] \wedge u[\text{year}] = 2010)\}$$

Example Queries

- Find the set of all courses taught in 2009, but not in 2010

section(course_id, year)
(C1, 2009)
(C2, 2009)
(C2, 2010)

Example Queries

- Find the set of all courses taught in 2009, but not in 2010

section(course_id, year)
(C1, 2009)
(C2, 2009)
(C2, 2010)

$$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id] \wedge s[\text{year}] = 2009 \wedge \neg \exists u \in section\ (t[course\_id] = u[course\_id] \wedge u[\text{year}] = 2010)\}$$

Universal Quantification

- Find (the IDs of) all students who have taken all courses offered in the Biology department

student(ID, name, address)

course(course_id, dept_name)

takes(ID, course_id)

Universal Quantification

- Find (the IDs of) all students who have taken all courses offered in the Biology department

student(ID, name, address)

course(course_id, dept_name)

takes(ID, course_id)

```
select distinct S.ID
from student as S
where not exists ( (select course_id
                    from course
                    where dept_name = 'Biology')
                   except
                    (select T.course_id
                     from takes as T
                     where S.ID = T.ID));
```

Universal Quantification

- Find (the IDs of) all students who have taken all courses offered in the Biology department

student(ID, name, address)

course(course_id, dept_name)

takes(ID, course_id)

```
select distinct S.ID
from student as S
where not exists ( (select course_id
                    from course
                    where dept_name = 'Biology')
                   except
                    (select T.course_id
                     from takes as T
                     where S.ID = T.ID));
```

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

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**

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
  - $\{< i > \mid \exists\, n, d, s (< i, n, d, s > \in instructor \wedge s > 80000)\}$
- Find the names of all instructors whose department is in the Watson building

instructor(ID, name, dept_name, salary)

department(dept_name, building, budget)

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
  - $\{< i > \mid \exists\, n, d, s (< i, n, d, s > \in instructor \wedge s > 80000)\}$
- Find the names of all instructors whose department is in the Watson building

instructor(ID, name, dept_name, salary)

department(dept_name, building, budget)

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

Example Queries

- Find the set of all courses taught in 2009 and 2010

section(course_id, year)
(C1, 2009)
(C2, 2009)
(C2, 2010)

Example Queries

- Find the set of all courses taught in 2009 and 2010

section(course_id, year)
(C1, 2009)
(C2, 2009)
(C2, 2010)

$$\{\langle c \rangle \mid \exists\, y\ (\langle c, y \rangle \in section \wedge y = 2009) \wedge \exists\, y\ (\langle c, y \rangle \in section \wedge y = 2010)\}$$

Universal Quantification

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

student(ID, name)

course(course_id, dept_name)

takes(ID, course_id, year)

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

Universal Quantification

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

student(ID, name)

course(course_id, dept_name)

takes(ID, course_id, year)

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

- $\{< i > \mid \exists\, n\ (< i, n > \in student \wedge (\forall\, ci, d\ (< ci, d > \in course \wedge d = \text{“Biology”} \Rightarrow \exists\, y\ (< i, ci, y > \in takes\ ))\}$

Chapter 6: Formal Relational Query Languages

- Relational Algebra
  - Six basic operators
    - select: $\sigma_p(r)$
    - project: $\prod_{A_1, A_2, \ldots, A_k}(r)$
    - union: $r \cup s$
    - set difference: $r - s$
    - Cartesian product: $r \times s$
    - rename: $\rho_{x(A_1, A_2, \ldots, A_n)}(E)$
  - Additional Operators
    - intersection: $r \cap s$
    - natural join: $r \bowtie s$
    - theta join: $r \bowtie_\theta s = \sigma_\theta (r \times s)$
    - outer join: $r \mathbin{⟕} s$, $r \mathbin{⟖} s$, $r \mathbin{⟗} s$
    - division: $r \div s$
    - generalized projection: $\prod_{F_1, F_2, \ldots, F_n}(E)$
    - aggregation: $_{G_1, G_2, \ldots, G_n}\mathcal{G}_{F_1(A_1), F_2(A_2, \ldots, F_n(A_n))}(E)$
- Tuple Relational Calculus
- Domain Relational Calculus

End of Chapter 6