Relational Query Languages

Section A

Week 3 Scribe Notes (Section A)

Members:

- Varun Chhangani
- Mohamed Moosa
- A N Vaishnavi
- Alivelu Sree Vidya
- Matla Sujala

Relational Query Languages

Relation Query Languages is majorly divided in 3 languages

- Relational Algebra (Procedural)
- Tuple Relational Calculus (Non Procedural)
- Domain Relational Calculus (Non Procedural)

By Codd's Theorem, all three of above are equivalent in computing power. All three are first order languages.
The above three languages are also called **pure languages**.

Relational Algebra

- **Relational Algerba** is a procedural language, That is, it has a set of procedures (or functions) that will operate on given relations.
- The operator here, takes one or two relations as input and produces a new relation as a result.
- There are six basic or fundamental operators in Relational Algebra:
  1. Select σ
  2. Project Π
  3. Rename ρ
  4. Union $\cup$
  5. Set Difference -
  6. Cartesian product x

Consider the following table instructor:

ID	name	salary	dept_name	building	budget
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
12121	Wu	90000	Finance	Painter	120000
15151	Mozart	40000	Music	Packard	80000
22222	Einstein	95000	Physics	Watson	70000
32343	El Said	60000	History	Painter	50000
33456	Gold	87000	Physics	Watson	70000
45565	Katz	75000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
76543	Singh	80000	Finance	Painter	120000
76766	Crick	72000	Biology	Watson	90000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000

(Example taken from Database System Concepts by Siberschatz, Sudershan and Korth)

Select Operation

- Notation $\sigma_p(r)$
- Where p is called the selection predicate, that if p is satisfied in a tuple existing in r, the tuple will be included in the output relation.
- Mathematically: $\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$. Here p is a formula in propositional calculus consisting of *terms* connected by : $\wedge$ (and), $\vee$ (or), $\neg$ (not). Here, each *term* is: *op* or where op is one of $=, \neq, \leq, <, \geq, >$
- Example: $\sigma_{dept\_name='Physics'}(instructor)$

Project Operation

- Notation $\Pi_{A_1,A_2,...,A_k}(r)$
- Where $A_1, A_2, ..., A_k$ are the attributes in r which will be present in the output relation. All the attributes that are not listed as A_i will not be shown in the output relation.
- As the relation is a set, duplicate values will be erased from output relation.
- Example, Treasurer may require the details of instructor and his salary only:
  - $\Pi_{ID,name,salary}(instructor)$

Rename Operation

- Notaion $\rho_x(E)$ or $\rho_{x(A_1,A_2,...,A_n)}(E)$
- It allows us to rename the given expression E and its attributes as x and $A_1, A_2, ...A_n$ respectively.
- Example: $\rho_{employee}(\Pi_{ID,name,salary}(instructor))$

Consider another relation section:

course_id	sec_id	semester	year	building	room	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Union Operation

- Notation: $r \cup s$
- Defined as: $r \cup s = \{t \mid t \cup r \text{ or } t \cup s\}$
- 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:
  - $\Pi_{course\_id}(\sigma_{semester='Fall' \wedge year=2009}(section)) \cap \Pi_{course\_id}(\sigma_{semester='Spring' \wedge year=2010}(section))$

Set Difference Operation

- Notation r – s
- Defined as: $r - s = \{t \mid t \in r \text{ and } t \notin s\}$
- 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
  - $\Pi_{course\_id}(\sigma_{semester='Fall' \wedge year=2009}(section)) - \Pi_{course\_id}(\sigma_{semester='Spring' \wedge year=2010}(section))$

Consider relation teaches:

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Cartesian Product Operation

- Notation r x s
- Defined as r x s $= \{$ t q $\mid$ t $\in$ r and q $\in$ s $\}$
- The output relation contains n(r) x n(s) tuples and the number of attributes are attributes in r + number attributes in s
- Example: $\Pi_{ID}(instructor)$ Œ $\Pi_{ID,year}(teaches)$ gives output:

instructor.ID	teaches.ID	year
10101	10101	2009
10101	10101	2010
10101	10101	2009
10101	12121	2010
10101	15151	2010
⋮	⋮	⋮

Set Intersection Operation

- Notation r $\cup$ s
- Defined as : r $\cap$ s = $\{$ t $\mid$ t $\in$ r and t $\in$ s$\}$
- For r $\cap$ 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)
- It is a derived operator. r $\cap$ s = r - (r - s)

Join

- Notation R $\underset{<join\ condition>}{\bowtie}$ S
- A join is a subset of a cartesian product satisfying a given condition.
- It is equivalent of $\sigma_{join\ condition}$(R x S)
- Suppose Q $\longleftarrow$ R($A_1, A_2, ..., A_n$) $\underset{A_i=B_j}{\bowtie}$ S($B_1, B_2, ..., B_m$), Q has n+m attributes as Q($A_1, A_2, ..., A_n, B_1, B_2, ...B_m$)

Theta Join

- Join with a boolean join condition is called a Theta Join.
- It may have join condition as logical and or logical or of multiple boolean conditions.

Equijoin

- Equijoin is a theta join with all the boolean conditions as a logical equality comparison.

Natural Join

- Notation R * S
- The join is applied implicitly as Equijoin on all the compatible attributes.
- Example: Q $\longleftarrow$ R(A,B,C,D) * S(C,D,E) outputs Q (A,B,C,D,E). Here, Q is R $\underset{R.C=S.C \wedge R.D=S.D}{\bowtie}$ S

Outer Join

- An extension of the join operation that avoids loss of information. It 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
  - All comparisons involving null are (roughly speaking) false by definition.
- Left Outer Join ($A \mathbin{⟕} B$)
  - If A has a tuple not satisfying the join condition, It will be considered in output relation with attribute values of attributes from right table set as NULL. Equivalent to: (r $\bowtie$ s) U (r – Π_R(r $\bowtie$ s) x {(null, . . . , null)}
- Right Outer Join ($A \mathbin{⟖} B$)
  - Equivalent of $B \mathbin{⟕} A$ (but with columns of B in right of attributes of A)
- Full Outer Join ($A \mathbin{⟗} B$)
  - $A \mathbin{⟕} B \cup A \mathbin{⟖} B$

Aggregate Functions

- These compute summary of information: for example, SUM, COUNT, AVG, MIN, MAX
- Notation ${}_{G_1,G_2,...,G_n}\mathcal{G}_{F_1(A_1),F_2(A_2),...,F_n(A_n)}(E)$
- $G_1, G_2, ..., G_n$ is a list of attributes on which to group (can be empty).
- E is an relational algebric expression
- $F_1, F_2, ..., F_n$ is an aggregate function and $A_1, A_2, ..., A_n$ are the attribute values of E.
- Example: ${}_{dept\_name}\mathcal{G}_{avg(salary)}(instructor)$
- Renaming is allowed directly on aggregate for simplicity. Example ${}_{dept\_name}\mathcal{G}_{avg(salary)\ as\ avg\_sal}(instructor)$

Division

- Notation r $\div$ s
- 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 x s $\subseteq$ r
- Example let r(ID,course_id) $= \Pi_{id,course\_id}(takes)$ and s(course_id) $= \Pi_{course\_id}(\sigma_{dept\_name='Physics'}(course))$, then r $\div$ gives us all the students who have taken *all* courses in Physics.

Outer Union

- Union for non or partially type compatible relations.
- Example if for relations R(X,Y) and S(X,Z), only X is the attribute that is type compatible, then output is T(X,Y,Z).

More on Relational Algebra

- Select, Project and Rename are Unary operators in Relational Algebra. Rest all are binary.
- Union, Intersection, Cartesian Product and Difference are Relation Algebra Operators from set theory
- Join and Division are Binary Relational Operations

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., $=$, $\neq$,$<$, $\leq$, $>$, $\geq$)
3. Set of connectives: and ($\wedge$), or ($\vee$) not ($\neg$)
4. Implication ($\rightarrow$): x $\rightarrow$ y, if x if true, then y is true
   - x $\rightarrow$ y $\equiv \neg$x $\vee$ y
5. Set of quantifiers:
   - Existential: $\exists$ t $\in$ r(Q(t)) $\equiv$ "there exists" a tuple in t in relation r such that predicate Q(t) is true
   - Universal: $\forall$ t $\in$ r(Q(t)) $\equiv$ all tuples in t in relation r, the predicate Q(t) is true

Domain Relational Calculus

- A nonprocedural query language equivalent in power to the tuple relational calculus.
- Query is in form of $\{ < x_1, x_2, \ldots, x_n > \mid \mathrm{P}\ (x_1, x_2, \ldots, x_n)\}$
  - Here x_i is a domain variable
  - P is a formula similar to predicate calculus.

Modifications of the Database

- The data in database can be modified using:
  - Deletion
  - Insertion
  - Updation

Deletion

- $r \longleftarrow r - \{ t \mid P(t) \}$

Insertion

- $r \longleftarrow r \cap \{$ ‘a’ ,b ,c ,d $\}$

Updation

- r $\longleftarrow$ r - { t | P(t) } $\cap$ { ‘a’ ,b ,c ,d }

Bibliography

- Database System Concepts by Silbershatz, Korth and Sudershan