DBDM

Relational calculus, Algebra

E.Coquery, R.Thion, A. Bonifati, M. Plantevit, M. Kaytoue, C. Robardet

emmanuel.coquery@liris.cnrs.fr

http://liris.cnrs.fr/~ecoquery/dbdm/

Outline

1. Relational Calculus
   - SQL & TRC
   - TRC Semantics
2. Relational Algebra
   - Operators

Relational calculus

- First-order Logic based theoretical language to represent queries
  - Declarative: Allow for expressing what to get, not how to do it
  - 2 flavors: domain or **tuple**
  - Example (tuple):

$$\{t \mid \exists u \; Employe(u) \\ \wedge t.Nom = u.Nom \wedge t.Fonction = u.Fonction \\ \wedge u.NumDept = 3\}$$

- No function symbols
- Predicate symbols can be
  - Comparison predicates with fixed interpretation
  - Relations from the database

Variables in Tuple Relational Calculus (TRC)

Variables represent tuples

- $t.A$ allow for accessing attribute A in tuple t
- "*named*" perspective:
  tuple = function: attributes → atomic value
  - As opposed to the usual *positional* perspective:
    $t = (v_1, \ldots, v_n)$
- When needed a tuple variable can be annotated with its attributes:
  - ex: $t^{Nom,Fonction}$: the domain of t (seen as a function) is the set of attributes $\{Nom, Fonction\}$

Predicates in TRC

Database relations

- Relations are unary
- Argument: a single tuple
- Meaning: given an instance of a relation R, $R(t)$ is true if t is in the instance of R.

Comparisons

- Can be (depending on the domain) $</2, =/2, \geq/2, \ldots$
- ex: $t.Salaire > 30000$
- ex: $t.Num = u.NumSup$
- Technically possible to extend to incorporate any expression built from constants and attribute values

TRC formulas

Build from:

- Atoms (relations applied to predicates, comparisons)
- Usual logical connectors: $\wedge, \vee, \neg$
- Quantifier: $\exists$

TRC query:

$$\{t \mid \phi\}$$

t is a tuple variable,
ϕ is a TRC formula with exactly t as free variable

Example SQL vs TRC

- ```
  SELECT Nom,Fonction
  FROM Employe
  ```

  $$\{t \mid \exists u \; Employe(u) \\ \wedge t.Nom = u.Nom \wedge t.Fonction = u.Fonction\}$$

- ```
  SELECT Nom
  FROM Employe
  WHERE Embauche < '1999-01-01'
  AND Salaire >= 30000;
  ```

  $$\{t \mid \exists u \; Employe(u) \\ \wedge t.Nom = u.Nom \\ \wedge u.Embauche < 1/1/1999 \\ \wedge u.Salaire \geq 30000\}$$

Example SQL vs TRC - renaming and variables

- ```
  SELECT u.Nom, u.Fonction
  FROM Employe u
  ```

  $$\{t \mid \exists u \; Employe(u) \\ \wedge t.Nom = u.Nom \wedge t.Fonction = u.Fonction\}$$

- ```
  SELECT u.Nom
  FROM Employe u
  WHERE u.Embauche < '1999-01-01'
  AND u.Salaire >= 30000
  ```

  $$\{t \mid \exists u \; Employe(u) \\ \wedge t.Nom = u.Nom \\ \wedge u.Embauche < 1/1/1999 \\ \wedge u.Salaire \geq 30000\}$$

Example SQL vs TRC - Join

```
SELECT d.Nomdept, b.Nombat
FROM Departement d, Batiment b
WHERE d.Numbat = b.Numbat
```

$$\{t \mid \exists d \ \exists b \ Departement(d) \wedge Batiment(b) \\ \wedge t.Nomdept = d.Nomdept \wedge t.Nombat = b.Nombat \\ \wedge d.Numbat = b.Numbat\}$$

Example SQL vs TRC - Negation & Subquery

```
SELECT d.Nomdept
FROM Departement d
WHERE NOT EXISTS (SELECT b.Numbat
                  FROM Batiment b
                  WHERE b.Numbat = d.Numbat)
```

$$\{t \mid \exists d \; Departement(d) \\ \wedge t.Nomdept = d.Nomdept \\ \wedge \neg(\exists b \; Batiment(b) \wedge b.Numbat = d.Numbat)\}$$

TRC semantics

Given a database d over $\mathbf{R}$ and a tuple assignment σ, the satisfaction of a TRC formula ϕ is inductively defined as follows:

- $\langle d, \sigma \rangle \models R(t)$ if $\sigma(t) \in d(R), R \in \mathbf{R}$
- $\langle d, \sigma \rangle \models t_1.A = t_2.B$ if $\sigma(t_1)(A) = \sigma(t_2)(B)$
- $\langle d, \sigma \rangle \models t.A = c$ if $\sigma(t)(A) = c$
- $\langle d, \sigma \rangle \models \neg\phi$ if $\langle d, \sigma \rangle \not\models \phi$
- $\langle d, \sigma \rangle \models \phi_1 \wedge \phi_2$ if $\langle d, \sigma \rangle \models \phi_1$ and $\langle d, \sigma \rangle \models \phi_2$
- $\langle d, \sigma \rangle \models \exists t\ \phi$ if there exists a tuple $\mathbf{u}$ such that $\langle d, \sigma_{t \mapsto \mathbf{u}} \rangle \models \phi$

σ : tuple variable $\rightarrow$ tuple (*i.e.* function attribute $\rightarrow$ atomic value)
d : relation name $\rightarrow$ relation (instance)
$\sigma_{t \mapsto \mathbf{u}}(t) = \mathbf{u}$ and if $t' \neq t$, $\sigma_{t \mapsto \mathbf{u}}(t') = \sigma(t')$

Answers to TRC Queries

Answers given by the tuple assignments satisfying the TRC formula:

$$ans(\{t \mid \phi\}, d) = \{\sigma(t) \mid \langle d, \sigma\rangle \models \phi\}$$

Not always computable:

$$\{t \mid \neg\exists u \; Employe(u) \wedge u.Nom = t.Nom\}$$

$\Rightarrow$ authorized TRC ensures possibility to compute answers

- principle: values should not be defined only in negations ([AVH, Chap.5])

Outline

1. Relational Calculus
   - SQL & TRC
   - TRC Semantics
2. Relational Algebra
   - Operators

Relational Algebra

Operations on relations (instances) to compute answers to queries

- Alternate query language
- Equivalent algebraic expressions may have different computing cost
  - $\Rightarrow$ Algebraic optimization
- Same expressive power as the authorized TRC [AVH, Chap.5]

Operators

- 5 base operators: selection, projection, union, difference and cartesian product.
- 1 syntactic operator, renaming, that only changes relation schema, not the values.
- Some other operators that can be obtained by composing the previous ones (syntactic sugar):
  - intersection, natural joins and θ joins.

Operators can be grouped in 2 categories

- set operators: union, intersection, difference, product
- database specific operators: selection, projection, joins, renaming

Semantics

Semantics can be given with an eval function $eval(e, d)$

- e is a relational algebra expression
- d a database instance (as for TRC semantics)

If e is a relation (name) R then $eval(R, d) = d(R)$.

Union $\cup$

- $R \cup S$: usual set union (relations seen as sets of tuples).
  - $eval(e_1 \cup e_2, d) = eval(e_1, d) \cup eval(e_2, d)$
- The 2 relations must have the same schema.

Example

∪

Students

FName	LName
Susan	Yao
Ramesh	Shah
Barbara	Jones
Amy	Ford
Jimmy	Wang

Teachers

FName	LName
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

Students ∪ Teachers

FName	LName
Susan	Yao
Ramesh	Shah
Barbara	Jones
Amy	Ford
Jimmy	Wang
John	Smith
Ricardo	Brown
Francis	Johnson

Intersection $\cap$

- $R \cup S$: usual set intersection (relations seen as sets of tuples).
  - $eval(e_1 \cap e_2, d) = eval(e_1, d) \cap eval(e_2, d)$
- The 2 relations must have the same schema.

Example ∩

Students

FName	LName
Susan	Yao
Ramesh	Shah
Barbara	Jones
Amy	Ford
Jimmy	Wang

Teachers

FName	LName
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

Students $\cap$ Teachers

FName	LName
Susan	Yao
Ramesh	Shah

Difference — or \

- $R \cup S$: usual set difference (relations seen as sets of tuples).
  - $eval(e_1 \setminus e_2, d) = eval(e_1, d) \setminus eval(e_2, d)$
- The 2 relations must have the same schema.

Example — or \

Students

FName	LName
Susan	Yao
Ramesh	Shah
Barbara	Jones
Amy	Ford
Jimmy	Wang

Teachers

FName	LName
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

Students — Teachers

FName	LName
Barbara	Jones
Amy	Ford
Jimmy	Wang

Cartesian product

- $R \times S$: new relation resulting from the combination of all tuples from R on one side and S on the other side
  - let $r_1 = eval(e_1, d)$ and $r_2 = eval(e_2, d)$
  - $eval(e_1 \times e_2, d) = \{t_1 \times t_2 \mid t_1 \in r_1, t_2 \in r_2\}$
    - $(t_1 \times t_2)(A) = t_1(A)$ if A is an attribute of t_1
    - $(t_1 \times t_2)(A) = t_2(A)$ if A is an attribute of t_2
- Schemas of R and S should be disjoint
- Not that useful as is, but used for representing joins

Exemple $\times$

Students

FName	LName
Susan	Yao
Ramesh	Shah

Teachers

FNameP	LNameP
John	Smith
Ricardo	Brown
Susan	Yao

Students $\times$ Teachers

FName	LName	FNameP	LNameP
Susan	Yao	John	Smith
Susan	Yao	Ricardo	Brown
Susan	Yao	Susan	Yao
Ramesh	Shah	John	Smith
Ramesh	Shah	Ricardo	Brown
Ramesh	Shah	Susan	Yao

Renaming ρ

- $\rho_{A_1/A'_1,\ldots,A_k/A'_k}(R)$
- Renaming of attributes in a relation R:
  A_1 becomes A'_1, ..., A_k becomes A'_k
  - $eval(\rho_{A_1/A'_1,\ldots,A_k/A'_k}(e), d) =$
    $\{\rho_{A_1/A'_1,\ldots,A_k/A'_k}(t) \mid t \in eval(e, d)\}$
    - $\rho_{A_1/A'_1,\ldots,A_k/A'_k}(t)(A'_i) = t(A_i)$
    - $\rho_{A_1/A'_1,\ldots,A_k/A'_k}(t)(B) = t(B)$ if $B \notin \{A_1, \ldots, A_k, A'_1, \ldots, A'_k\}$
- Useful to (dis)align schemas before using set operators

Example ρ

Employe

FName	LName	NoDept
John	Smith	5
Ricardo	Brown	3
Susan	Yao	5
Daniel	Johnson	2
Francis	Johnson	2
Ramesh	Shah	4
Ramesh	Shah	2

$\rho_{FName/First,LName/Last}(\text{Employe})$

First	Last	NoDept
John	Smith	5
Ricardo	Brown	3
Susan	Yao	5
Daniel	Johnson	2
Francis	Johnson	2
Ramesh	Shah	4
Ramesh	Shah	2

Selection σ

- $\sigma_C(R)$ selects tuples in R using condition C
- Conditions: combination of comparaisons $(=, <, >, \leq, \geq)$
  - between attributes
  - between an attribute and a constant
  - Example: $\sigma_{NoDept=5}(Employe)$
- Can be combined using $\wedge, \vee$
- $eval(\sigma_C(e), d) = \{t \mid t \in eval(e, d) and eval_{cond}(C, t) = true\}$
  - $eval_{cond}(A\Box B, t) = t(A)\Box t(B)$
  - $eval_{cond}(A\Box c, t) = t(A)\Box c$
  - $eval_{cond}(c_1 \wedge c_2, t) = eval_{cond}(c_1, t) \wedge eval_{cond}(c_2, t)$
  - $eval_{cond}(c_1 \vee c_2, t) = eval_{cond}(c_1, t) \vee eval_{cond}(c_2, t)$

Example σ

Employe

FName	LName	NoDept
John	Smith	5
Ricardo	Brown	3
Susan	Yao	5
Daniel	Johnson	2
Francis	Johnson	2
Ramesh	Shah	4
Ramesh	Shah	2

$\sigma_{NoDept=5}(\text{Employe})$

FName	LName	NoDept
John	Smith	5
Susan	Yao	5

Projection π

- $\pi_{A_1,\ldots,A_k}(R)$ keeps only attributes $A_1, \ldots, A_k$ in relation R.
  - $eval(\pi_{A_1,\ldots,A_k}(e), d) = \{t_{|A_1,\ldots,A_k} \mid t \in eval(e, d)\}$
- Does not delete lines
  - except when two lines match on $A_1, \ldots, A_k$

Exemple

π

Employe

FName	LName	NoDept
John	Smith	5
Ricardo	Brown	3
Susan	Yao	5
Daniel	Johnson	2
Francis	Johnson	2
Ramesh	Shah	4
Ramesh	Shah	2

$\pi_{LName,NoDept}(\text{Employe})$

LName	NoDept
Smith	5
Brown	3
Yao	5
Johnson	2
Shah	4
Shah	2

Join $\bowtie$

Natural join: $R \bowtie S$

- R and S have common attributes $A_1, \ldots, A_k$
- The result is the set of tuples obtained by combining tuples t_1 from R and t_2 from S having the same value of attributes $A_1, \ldots, A_k$.
  - Attributes $A_1, \ldots, A_n$ are not duplicated.

θ-join: $R \bowtie_C S$

- Equality condition of natural join is replaced by a custom condition C
- R and S must not share attributes

Exemple $\bowtie$

Employe

FName	LName	NoDept
John	Smith	5
Ricardo	Brown	3
Susan	Yao	5
Francis	Johnson	2
Ramesh	Shah	4

Emplacement

NoDept	Building
1	centre
3	sud
4	est
5	ouest

Employe $\bowtie$ Emplacement

FName	LName	NoDept	Building
John	Smith	5	ouest
Ricardo	Brown	3	sud
Susan	Yao	5	ouest
Ramesh	Shah	4	est

Join as a Composed Operator

Assume R and S have exactly $A_1, \ldots, A_n$ as shared attributes, $B_1, \ldots, B_m$ as attributes appearing only in R and $C_1, \ldots, C_P$ as attributes appearing only in S.

$$
\begin{aligned}
R \bowtie S \quad \equiv \quad & \pi_{A_1,\ldots,A_n,B_1,\ldots,B_m,C_1,\ldots,C_P}(\\
& \quad \sigma_{A_1=A'_1 \wedge \cdots \wedge A_n=A'_n}(\\
& \qquad R \times \rho_{A_1/A'_1,\ldots,A_n/A'_n}(S)))
\end{aligned}
$$

Expessive power of TRC and Relational Algebra

Theorem

Authorized TRC and Relational Algebra have the same expressive power:

- *let* $\{t \mid \phi\}$ *be an authorized TRC query and* d *a database instance. There exists a relational algebra expression* e *using relations in* d*, such that* $eval(e, d) = ans(\{t \mid \phi\})$
- *let* e *be a relational algebra expression* e *over a database* d*. There exists an authorized TRC query* $\{t \mid \phi\}$ *such that* $eval(e, d) = ans(\{t \mid \phi\})$

References

[AVH] Abiteboul, Hull, Vianu: *Foundations of Databases*, http://webdam.inria.fr/Alice/