Introduction to Database Systems

Relational Calculus

Stéphane Bressan

National University of Singapore

Relational Query Languages

Two mathematical query languages form the basis for practical languages like SQL: Relational Calculus and Relational Algebra. Relational Calculus is declarative, i.e. the user says what she wants, rather than how to compute it. Relational Algebra is imperative, i.e. the user says how to compute the result. It is useful for representing execution plans. Query languages are NOT programming languages: they are not designed to be Turing complete.

There are further two relational calculi: Domain relational calculus (DRC) and Tuple relational calculus (TRC). Both are based on logic. They differ on what variables represent.

Semantics of Propositional Logic

The semantic of propositional logic is defined by truth tables.

A	B	$(A \vee B)$	$(A \wedge B)$	$(A \Rightarrow B)$	$\neg (A)$
T	T	T	T	T	F
F	T	T	F	T	T
T	F	T	F	F	F
F	F	F	F	T	T

$\neg A \vee \; B)$

$\neg(A \wedge \neg B)$

Syntax of First Order Logic

First order logic consists of formulae built from predicates (lower case) , constants (lower case), variables (upper case, quantified or free), quantifiers ($\forall$ and $\exists$) and logical operators ($\wedge$, $\vee$, $\neg$ and $\Rightarrow$).

For example:

$\forall$ X $\forall$ Y ((mother(X, Y) $\wedge$ greek(X)) $\Rightarrow$ greek(Y))

"Everyone (everything) whose mother is Greek is also Greek."

First Order Logic: Predicates, Constants, Variables and Quantifiers

greek(aristotle)

greek(X)

mother(olympias, alexander)

mother(X, Y)

$\exists$ X greek(X)

$\exists$ X mother(olympias, X)

$\exists$ X $\exists$ Y mother(Y, X)

$\exists$ Y $\exists$ X mother(Y, X)

$\forall$ X greek(X)

$\forall$ Y $\exists$ X mother(X, Y)

$\exists$ X $\forall$ Y mother(X, Y)

Remarks

To avoid confusion we agree that a variable is quantified once at most and that if a variable is quantified in a formula, it cannot appear outside of the scope of its quantifier.

$$\forall X \underbrace{(\exists X \overbrace{(\mathrm{mother}(X, Y))})}$$

$$\exists Y \overbrace{(\mathrm{mother}(X, Y))} \wedge \mathrm{father}\,(Z, Y)$$

Different scopes

Remarks

$\neg \forall X\ F(X)$ is equivalent to $\exists X \neg F(X)$.

$\neg \exists X\ F(X)$ is equivalent to $\forall X \neg F(X)$.

Here F(X) represents a formula that contains the variable X

Calculus

A Calculus defines formulae and their meaning. In Domain Relational Calculus (DRC) variables range over values. In Tuple Relational Calculus (TRC) variables range over tuples.

Calculus

How to represent the set of integers 2, 3, and 4?

In extension:

$$\{2, 3, 4\}$$

In Intention (set-builder notation, comprehension, abstraction):

$$\{ X \mid X \in N \wedge 1 < X \wedge X < 5\}$$

Calculus: Where is the Truth?

The truth is in the database

If a relation Mother in the database has a tuple

mother(olympias, alexander)

then Olympias is the mother of Alexander

Otherwise it is not (closed world assumption)

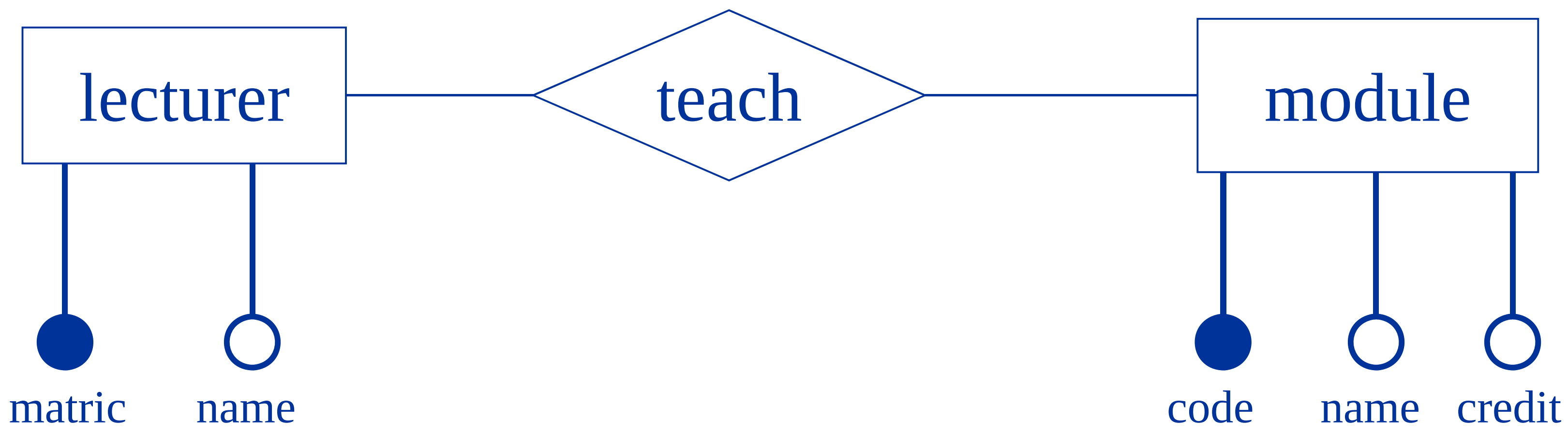
DOMAIN RELATIONAL CALCULUS

Example

lecturer(matric, name)

module(code, name, credit)

teach(matric, code)



Example

$\{\langle X \rangle \mid \exists Y\ lecturer(X, Y)\}$

```
SELECT l.matric
FROM lecturer l;
```

$\{\langle X \rangle \mid \exists Y\ lecturer(X, Y) \wedge Y = \text{“john”}\}$

```
SELECT l.matric
FROM lecturer l
WHERE l.name='John';
```

$\{\langle X \rangle \mid lecturer(X, \text{“john”})\}$

```
SELECT l.matric
FROM lecturer l
WHERE l.name='John';
```

$\{\langle X, Y \rangle \mid lecturer(X, Y)\}$

```
SELECT l.matric, l.name
FROM lecturer l;
```

Syntax of Domain Relational Calculus

{Head | Body}   {variableList | formula}

Head:

- Variable list: <X1, X2, …>
- Variables in the head are free

Body:

- Formula in first order logic
- Variables in the body that are not in the head are quantified

Semantics of Domain Relational Calculus

{variableList | formula}

Head:

Returns the set of tuples of values such that if we replace the variables in the variable list by the values in one such tuples,

Body:

Then the formula in the body is true.

Example

Find the lecturers teaching a module with less than 2 credits. Print the name of the lecturers and the name of the corresponding modules.

$$\{\langle LN, MN\rangle \mid \exists\, M1\ \exists\, M2\ \exists\, C1\ \exists\, C2\ \exists\, Cr$$
$$lecturer(M1, LN) \wedge module(C1, MN, Cr) \wedge teach(M2, C2)$$
$$\wedge\ C1 = C2 \wedge M1 = M2 \wedge Cr < 2\}$$

```
SELECT l.name,  m.name
FROM lecturer l, module m, teach t
WHERE l.matric=t.matric AND m.code = t.code AND
module.credit < 2;
```

Example

Find the lecturers teaching a module with less than 2 credits. Print the name of the lecturers and the name of the corresponding modules.

$$\{\langle LN, MN\rangle \mid \exists\, M\ \exists\, C\ \exists\, Cr$$
$$\quad lecturer(M, LN)$$
$$\quad \wedge\ module(C, MN, Cr)$$
$$\quad \wedge\ teach(M, C)$$
$$\quad \wedge\ Cr < 2\}$$

This shorthand notation may be confusing when translating to SQL.

Example

Find the lecturers with the same name. Print the name.

$\{\langle N \rangle \mid$

$\quad lecturer(M1, N) \land lecturer(M2, N)$

$\quad \land\ M1 <> M2\}$

$\{\langle N1 \rangle \mid \exists N2$

$\quad lecturer(M1, N1) \land lecturer(M2, N2)$

$\quad \land\ M1 <> M2 \land N1 = N2\}$

```
SELECT l1.name
FROM lecturer l1, lecturer l2
WHERE l1.matric <> l2.matric AND l1.name=l2.name;
```

Example

Find the name of the lecturers teaching all the modules.

$$\{\langle N\rangle \mid \exists M\ \forall C\ \forall MN\ \forall Cr\ \ lecturer(M, N) \land (module(C, MN, Cr) \Rightarrow teach(M, C))\}$$

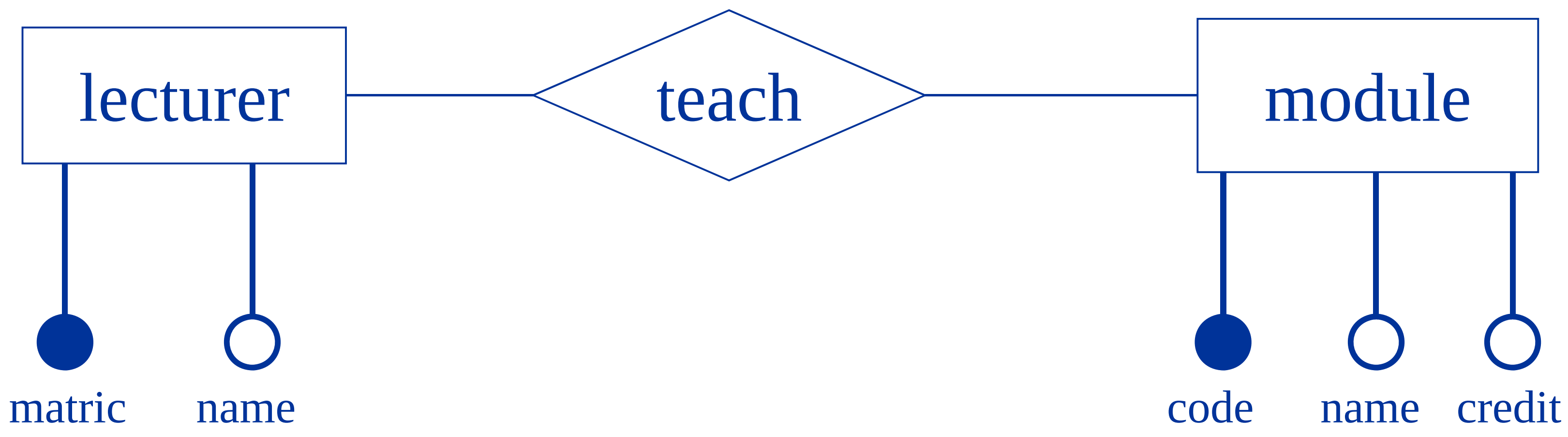
T-UPLE RELATIONAL CALCULUS

Example

lecturer(<u>matric</u>, name)

module(<u>code</u>, name, credit)

teach(<u>matric</u>, <u>code</u>)



Example

$\{T \mid$
$T \in lecturer\}$

```
SELECT *
FROM lecturer l;
```

$\{T \mid \exists L$
$(L \in lecturer$
$\wedge T = L)\}$

```
SELECT *
FROM lecturer l;
```

$\{T \mid \exists L\ ($
$L \in lecturer$
$\wedge\ T.matric = L.matric\ T.name = L.name)\}$

```
SELECT l.matric, l.name
FROM lecturer l;
```

$\{T \mid \exists L$
$(L \in lecturer$
$\wedge L.name = \text{“Smith”}$
$\wedge T.matric = L.matric)\}$

```
SELECT l.matric
FROM lecturer l
WHERE l.name='Smith';
```

Syntax of Tuple Relational Calculus

$\{T \mid formula\}$

Variables are tuples

$T \in rel$: T is a tuple in relation rel

T.a: The value of attribute a in T.

T1 = T2: T1 and T2 must have the same attributes with same values.

Parenthesis can be omitted if non ambiguous (not advised)

$\{T \mid \exists\, T1\ (T1 \in lecturer \land T.name = T1.name)\}$

$\{T \mid \exists\, T1\ T1 \in lecturer \land T.name = T1.name\}$

Semantics of Domain Relational Calculus

{T| formula}

Head:

Returns the set of tuples T such that if we replace the variables T with one such tuples,

Body:

Then the formula in the body is true.

Example

Find the lecturers teaching a module with less than 2 credits. Print the name of the lecturers and the name of the corresponding modules.

$$\{T \mid \exists L \; \exists M \; \exists T$$
$$(L \in lecturer \wedge M \in module \wedge T \in teach$$
$$\wedge\; L.matric = T.matric \;\wedge\; T2.code = T3.code \wedge T2.credit < 2$$
$$\wedge\; T.lec\_name = L.name \wedge T.mod\_name = M.name)\}$$

```
SELECT l.name AS lec_name,  m.name AS mod_name
FROM lecturer l, module m, teach t
WHERE l.matric=t.matric AND m.code = t.code AND
module.credit < 2;
```

Example

Find the name of the lecturers teaching all modules.

$$
\begin{aligned}
&\{T \mid \exists L \\
&\quad (L \in lecturer \\
&\qquad \wedge \forall M\ (M \in module \Rightarrow \\
&\qquad\qquad (\exists T\ (\\
&\qquad\qquad\ T \in teach \\
&\qquad\qquad \wedge L.matric = T.matric\ \wedge M.code = T.code))) \\
&\qquad \wedge T.name = L.name)\}
\end{aligned}
$$

Example

Find the name of the lecturers teaching all modules.

$$
\begin{aligned}
&\{T \mid \exists L\ \forall M\ \exists T \\
&\qquad (L \in lecturer \\
&\qquad\quad \wedge (M \in module \Rightarrow \\
&\qquad\qquad\quad (\\
&\qquad\qquad\quad T \in teach \\
&\qquad\qquad\quad \wedge L.matric = T.matric \wedge M.code = T.code)) \\
&\qquad\quad \wedge T.name = L.name)\}
\end{aligned}
$$

This is how we prefer to write it.

Example to SQL

Find the name of the lecturers teaching all modules.

$$
\begin{aligned}
&\{T \mid \exists L \\
&\qquad (L \in lecturer \\
&\qquad\qquad \wedge \forall M\ (M \in module \Rightarrow \\
&\qquad\qquad\qquad (\exists T\ (\\
&\qquad\qquad\qquad T \in teach \\
&\qquad\qquad\qquad \wedge L.matric = T.matric \ \wedge M.code = T.code))) \\
&\qquad\qquad \wedge T.name = L.name)\}
\end{aligned}
$$

Example to SQL

Find the name of the lecturers teaching all modules.

$\{T \mid \exists L$

$(L \in \text{lecturer}$

$\wedge \forall M \neg (M \in \text{module} \vee$

$(\exists T ($

$T \in \text{teach}$

$\wedge L.\text{matric} = T.\text{matric} \ \wedge M.\text{code} = T.\text{code})))$

$\wedge T.\text{name} = L.\text{name})\}$

$A \Rightarrow B$ is the same as $\neg A \vee B$

Example to SQL

Find the name of the lecturers teaching all modules.

$\{T \mid \exists L$

$(L \in lecturer$

$\wedge \neg \neg (\forall M \neg (M \in module \vee$

$(\exists T ($

$T \in teach$

$\wedge L.matric = T.matric \wedge M.code = T.code))))$

$\wedge T.name = L.name)\}$

Double Negation:

A is the same as $\neg \neg A$

Example to SQL

Find the name of the lecturers teaching all modules.

$\{T \mid \exists L$

$(L \in lecturer$

$\wedge \neg (\exists M (M \in module \wedge$

$\neg(\exists T ($

$T \in teach$

$\wedge L.matric = T.matric \wedge M.code = T.code)))$

$\wedge T.name = L.name)\}$

De Morgan:

$\neg \neg (\neg A \vee B)$ is the same as $\neg (A \wedge \neg B)$

De Morgan (for quantifiers):

$\neg \forall X (\neg A)$ is the same as $\neg (\exists X (A))$

Example to SQL

$\{T \mid \exists L$

$(L \in lecturer$

$\wedge \neg (\exists M (M \in module \wedge$

$\neg (\exists T$

$(T \in teach$

$\wedge L.matric = T.matric \wedge M.code = T.code)))$

$\wedge T.name = L.name)\}$

Example to SQL

```
SELECT l.name
FROM lecturer l
WHERE NOT EXISTS (
     SELECT *
     FROM module m
     WHERE NOT EXISTS (
          SELECT *
           FROM teach t
          WHERE l.matric = t.matric
          AND m.code = t.code));
```

Safety of Queries in Relational Calculus

$\{T \mid T \notin \text{lecturer}\}$

("mycat", 22, "red") is not a lecturer, any t-uple in the world maybe an answer if it is not already in the lecturer relation.

Safety of Queries in T-uple Relational Calculus

A query is safe if the set of t-uples in the answer is a subset of the set of t-uples that can be constructed from the constants explicitly referenced directly (they appear in the query) or indirectly (they appear in a relation mentioned in the query) in the query.

We consider only safe queries

Credits

The content of this lecture is based on the book "Introduction to database Systems"
By
S. Bressan and B. Catania, McGraw Hill publisher

Images and clips used in this presentation are licensed from Microsoft Office Online Clipart and Media

For questions about the content of this course and about copyrights, please contact Stéphane Bressan

steph@nus.edu.sg