The Relational Model

Fall 2015

David Toman

School of Computer Science
University of Waterloo

Databases CS338

How do we ask Questions (and understand Answers)?

Find *all pairs of (natural) numbers that add to 5*!

Question: $\{(x, y) \mid x + y = 5\text{PLUS}(x, y, 5)\}$
Answer: $\{(0,5), (1,4), (2,3), (3,2), (4,1), (5,0)\}$

. . . but but but why? (explain this to a 6 year old!)
because $(0, 5, 5)$, etc., appear in PLUS!

Find pairs of numbers that add to the same number as they subtract to (i.e., $x + y = x - y$)!

Question: $\{(x, y) \mid \exists z.\text{PLUS}(x, y, z) \wedge \text{PLUS}(z, y, x)\}$
Answer: $\{(0,0), (5,5)?\}$

. . . answer depends on the content (instance) of PLUS!

Find the *neutral element* (of addition)!

Question: $\{(x) \mid \text{PLUS}(x, x, x)\}$
Answer: $\{(0)\}$

Addition Table

PLUS		
0	0	0
0	1	1
1	0	1
0	2	0
2	0	2
	...	
0	5	5
	...	
1	4	5
	...	
2	3	5
	...	
	...	

How do we ask Questions about Employees?

Find *all employees who work for "Bob"*!

Question: $\{(x, y) \mid \text{EMP}(x, y, \mathit{Bob})\}$
Answer: $\{(\mathit{Sue}, \text{CS}), (\mathit{Bob}, \text{CO})\}$

why? because $(\mathit{Sue}, \text{CS}, \mathit{Bob})$, etc., appear in EMP!

Find pairs of emp-s working for the same boss!

Q: $\{(x_1, x_2) \mid \exists y_1, y_2, z.\text{EMP}(x_1, y_1, z) \wedge \text{EMP}(x_2, y_2, z)\}$
A: $\{(\mathit{Sue}, \mathit{Bob}), (\mathit{Fred}, \mathit{John}), (\mathit{Jim}, \mathit{Eve})\}$ ← is that all?

Find employees who are their own bosses!

Q: $\{(x) \mid \exists y.\text{EMP}(x, y, x)\}$
A: $\{(\mathit{Sue}), (\mathit{Bob})\}$

Employee Table

EMP		
Name	Dept	Boss
Sue	CS	*Bob*
Bob	CO	*Bob*
Fred	PM	*Mark*
John	PM	*Mark*
Jim	CS	*Fred*
Eve	CS	*Fred*
Sue	PM	*Sue*

The Relational Model

Idea

All information is organized in (a finite number of) relations.

Features:

- simple and clean data model
- powerful and declarative query/update languages
- semantic integrity constraints
- data independence

Relational Structures/Databases

Components:

Universe
- a set of *values* $\mathbf{D}$ with equality ($=$)

Relation
- schema: name R, arity k (the number of attributes)
- instance: a relation $\mathbf{R} \subseteq \mathbf{D}^k$.

Database
- schema: finite set of relation schemes
- instance: a relation $\mathbf{R_i}$ for each R_i

Notation

Signature: $\rho = (R_1, \ldots, R_n)$ *Instance:* $\mathbf{D} = (\mathbf{D}, =, \mathbf{R_1}, \ldots, \mathbf{R_n})$

Examples of Relational Structures a.k.a. Databases

- the integer numbers with addition and multiplication:

  $\rho = (\texttt{plus}, \texttt{times})$ $\mathbf{D} = (\mathbf{Z}, =, \mathbf{plus}, \mathbf{times})$
- a Bibliography Database
- ...

Example: Bibliography

Relations (signatures) used in examples:

```
author(aid, name)
wrote(author, publication)
publication(pubid, title)
book(pubid, publisher, year)
journal(pubid, volume, no, year)
proceedings(pubid, year)
article(pubid, crossref, startpage, endpage)
```

$\Rightarrow$ names of attributes will be important later (for SQL)

Example (sample instance)

$$
\begin{aligned}
\textbf{author} &= \{ (1, \text{John}), (2, \text{Sue}) \} \\
\textbf{wrote} &= \{ (1,1), (1,4), (2,3) \} \\
\textbf{publication} &= \{ (1, \text{Mathematical Logic}), \\
&\quad (3, \text{Trans. Databases}), \\
&\quad (2, \text{Principles of DB Syst.}), \\
&\quad (4, \text{Query Languages}) \} \\
\textbf{book} &= \{ (1, \text{AMS}, 1990) \} \\
\textbf{journal} &= \{ (3, 35, 1, 1990) \} \\
\textbf{proceedings} &= \{ (2, 1995) \} \\
\textbf{article} &= \{ (4, 2, 30, 41) \}
\end{aligned}
$$

Example (tabular form)

author	
aid	name
1	John
2	Sue

wrote	
author	publication
1	1
1	4
2	3

publication	
pubid	title
1	Mathematical Logic
3	Trans. Databases
2	Principles of DB Syst.
4	Query Languages

$\Rightarrow$ that's why relations are often called "tables".

Simple (Atomic) "Truth"

Idea

Relationships between objects (tuples) that are present in an instance are true, relationships absent are false.

In the sample *Bibliography* database instance

- "John" is an *author* with id "1": $(1, \text{John}) \in \mathbf{author}$;
- "Mathematical Logic" is a publication: $(1, \text{Mathematical Logic}) \in \mathbf{publication}$;
  Moreover it is a book published by "AMS" in "1990": $(1, \text{AMS}, 1990) \in \mathbf{book}$;
- "John" wrote "Mathematical Logic": $(1, 1) \in \mathbf{wrote}$;
- "John" has **NOT** written "Trans. Databases": $(1, 3) \notin \mathbf{wrote}$;
- etc.

Queries

IDEA1: use *variables* to collect answers

$\texttt{author}(x, y)$ asks for all answer tuples $[x \mapsto a, y \mapsto b]$
such that the pair $(a, b) \in \mathbf{author}$

IDEA2: build more complex queries from simpler ones using...

Logical connectives:

- Conjunction (and): $\texttt{author}(x, y) \wedge \texttt{wrote}(x, z)$
- Disjunction (or): $\texttt{author}(x, y) \vee \texttt{publication}(x, y)$
- Negation (not): $\neg\texttt{author}(x, y)$

Quantifiers:

- Existential (there is...): $\exists x.\texttt{author}(x, y)$

Relational Calculus: Summary

Definition (Syntax)

Queries over a **database schema** $(R_1, \ldots, R_k)$ are

$$Q ::= R(x_{i_1}, \ldots, x_{i_k}) \mid x_i = x_j \mid Q_1 \wedge Q_2 \mid \exists x_i.Q \mid Q_1 \vee Q_2 \mid \neg Q$$

Definition (Answer tuples and Answer(s))

An answer tuple $\mathbf{t}$ for Q assigns *values* to Q's (free) variables.

The answer tuple $\mathbf{t}$ is an answer for formula Q (written $\mathbf{t} \in Q$) if:

$\mathbf{t} \in R(x_{i_1}, \ldots, x_{i_k})$	if	$(\mathbf{t}(x_{i_1}), \ldots, \mathbf{t}(x_{i_k})) \in \mathbf{R^D}$
$\mathbf{t} \in x_i = x_j$	if	$\mathbf{t}(x_i) = \mathbf{t}(x_j)$
$\mathbf{t} \in Q_1 \wedge Q_2$	if	$\mathbf{t} \in Q_1$ and $\mathbf{t} \in Q_2$
$\mathbf{t} \in Q_1 \vee Q_2$	if	$\mathbf{t} \in Q_1$ or $\mathbf{t} \in Q_2$
$\mathbf{t} \in \neg Q_1$	if	it is not the case that $\mathbf{t} \in Q_1$
$\mathbf{t} \in \exists x_i.Q_1$	if	$\mathbf{t}[x_i \mapsto v] \in Q_1$ for some value v

An answer to Q is $\{\mathbf{t} \mid \mathbf{t} \in Q\}$ (i.e., all answer tuples that make Q true).

Example

Find pairs of emp-s working for the same boss!

Q: $\{(x_1, x_2) \mid \exists y_1, y_2, z.\text{EMP}(x_1, y_1, z) \wedge \text{EMP}(x_2, y_2, z)\}$
A: $\{(\textit{Sue}, \textit{Fred}), \ldots\}$

because:

1. $[x_1 \mapsto \textit{Sue}, y_1 \mapsto \text{CS}, z \mapsto \textit{Bob}] \in \text{EMP}(x_1, y_1, z)$
2. $[x_2 \mapsto \textit{Fred}, y_2 \mapsto \text{CO}, z \mapsto \textit{Bob}] \in \text{EMP}(x_2, y_2, z)$
3. $[x_1 \mapsto \textit{Sue}, y_1 \mapsto \text{CS}, x_2 \mapsto \textit{Fred}, y_2 \mapsto \text{CO}, z \mapsto \textit{Bob}] \in \text{EMP}(x_1, y_1, z) \wedge \text{EMP}(x_2, y_2, z)$
4. $[x_1 \mapsto \textit{Sue}, x_2 \mapsto \textit{Fred}] \in \exists y_1, y_2, z.\text{EMP}(x_1, y_1, z) \wedge \text{EMP}(x_2, y_2, z)$

Emp Table

EMP

Name	Dept	Boss
Sue	CS	*Bob*
Fred	CO	*Bob*
Bob	PM	*Mark*
John	PM	*Mark*
Jim	CS	*Fred*
Eve	CS	*Fred*
Sue	PM	*Sue*

Sample Queries

over numbers (with addition and multiplication):

- list all composite numbers
- list all prime numbers

over the bibliography database:

- list all publications
- list titles of all publications
- list titles of all books
- list all publications without authors
- list (pairs of) coauthor names
- list titles of publications written by a single author

Equivalences and Syntactic Sugar

Boolean Equivalences

- $\neg(\neg Q_1) \equiv Q_1$
- $Q_1 \vee Q_2 \equiv \neg(\neg Q_1 \wedge \neg Q_2)$
- $Q_1 \rightarrow Q_2 \equiv \neg Q_1 \vee Q_2$
- $Q_1 \leftrightarrow Q_2 \equiv (Q_1 \rightarrow Q_2) \wedge (Q_2 \rightarrow Q_1)$
- $\ldots$

First-order Equivalences

- $\forall x.Q \equiv \neg\exists x.\neg Q$

How do we ask Questions (and understand Answers)?

Find the *neutral element* (of addition)!

Question: $\{(x) \mid \text{PLUS}(x, x, x)\}$
Answer: $\{(0)\}$

but shouldn't the query really be

$$\{(x) \mid \forall y.\text{PLUS}(x, y, y) \wedge \text{PLUS}(y, x, y)\} \qquad (*)$$

IDEA

$(*)$ is the same as $\{(x) \mid \forall y.\text{PLUS}(x, y, y)\}$
because PLUS is *commutative*

is the same as $\{(x) \mid \text{PLUS}(x, x, x)\}$
because PLUS is *monotone*

$\Rightarrow$ Laws of Arithmetic for Natural Numbers

Addition Table

PLUS

0	0	0
0	1	1
1	0	1
0	2	0
2	0	2
	...	
0	5	5
	...	
1	4	5
	...	
2	3	5
	...	
	...	

Laws a.k.a. Integrity Constraints

Idea

What must be always true for the natural numbers (i.e., for PLUS)?

- addition is commutative

$$\forall x, y, z.\text{PLUS}(x, y, z) \rightarrow \text{PLUS}(y, x, z)$$
$$(\neg\exists x, y, z.\text{PLUS}(x, y, z) \wedge \neg\text{PLUS}(y, x, z))$$

- addition is a (relational representation of a) binary function

$$\forall x, y, z_1, z_2.\text{PLUS}(x, y, z_1) \wedge \text{PLUS}(x, y, z_2) \rightarrow z_1 = z_2$$
$$(\neg\exists x, y, z_1, z_2.\text{PLUS}(x, y, z_1) \wedge \text{PLUS}(x, y, z_2) \wedge \neg(z_1 = z_2))$$

- addition is a total function

$$\forall x, y.\exists z.\text{PLUS}(x, y, z)$$

- addition is monotone in both arguments (harder), etc., etc.

Laws a.k.a. Integrity Constraints for Employees

Idea

Integrity constraints
$\Rightarrow$ *yes/no queries that must be true in every valid database instance.*

- Every Boss is an Employee

$$\forall x, y, z.\text{EMP}(x, y, z) \rightarrow \exists u, w.\text{EMP}(z, u, w)$$

- Every Boss manages a unique Department

$$\forall x_1, x_2, y_1, y_2, z.\text{EMP}(x_1, y_1, z) \wedge \text{EMP}(x_2, y_2, z) \rightarrow y_1 = y_2$$

- No Boss cannot have another Employee serving as their Boss

$$\forall x, y, z.\text{EMP}(x, y, z) \rightarrow \text{EMP}(z, y, z)$$

Integrity Constraints

Relational *signature* captures only the structure of relations.

Idea

Valid database instances satisfy additional integrity constraints.

- values of a particular attribute belong to a prescribed *data type*.
- values of attributes are unique among tuples in a relation (*keys*).
- values appearing in one relation must also appear in another relation (*referential integrity*).
- values cannot appear simultaneously in certain relations (*disjointness*).
- values in certain relation must appear in at least one of another set of relations (*coverage*).
- ...

Example Revisited (Bibliography)

Typing constraints

- Author id's are integers.
- Author names are strings.

Uniqueness of values/Keys

- Author id's are unique and determine author names.
- Publication id's are unique as well.
- Articles are identified by their id and the id of a collection they have appeared in.

Referential Integrity/Foreign Keys

- "books", "journals", "proceedings", and "articles" are "publications".
- The components of a "wrote" tuple must be an "author" and a "publication".

Example Revisited (cont.)

Disjointness

- "books" are different from "journals".
- "books" are different from "proceedings".

Coverage

- Every "publication" is a "book" or a "journal" or a "proceedings" or an "article".
- Every "article" appears in a "book" or in a "journal" or in "proceedings".

Views and Integrity Constraints

Idea

Answers to queries can be used to define derived relations (views)
$\Rightarrow$ *extension of a DB schema*

- subtraction, complement, ...
- *collection*-style publication, editor, ...

In general, a view is an integrity constraint of the form

$$\forall x_1, \ldots, x_k . R(x_1, \ldots, x_k) \leftrightarrow Q$$

for R a new relation name and $x_1, \ldots, x_k$ answer variables of Q.

Story so far...

1. databases $\Leftrightarrow$ relational structures
2. queries $\Leftrightarrow$ formulas in First-Order logic
3. integrity constraints $\Leftrightarrow$ closed formulas in FO logic

... so is there anything new here?

$\Rightarrow$YES: database instances must be finite

Unsafe Queries

- $\neg \exists x . \texttt{author}(x, y)$
- $\texttt{book}(x, y, z) \vee \texttt{proceedings}(x, y)$
- $x = y$

$\Rightarrow$ we want only queries with finite answers (over finite instances).

Definition (Range restricted queries)

$$
\begin{array}{lll}
Q ::= & R(x_{i_1}, \ldots, x_{i_k}) & \\
& Q_1 \wedge Q_2 & \\
& Q \wedge x_i = x_j & \text{at least one of } x_i, x_j \text{ is answer variable of } Q \\
& \exists x_i . Q & \\
& Q_1 \vee Q_2 & \left.\begin{array}{l}\\\end{array}\right\} \text{answer variables of } Q_1 \text{ and } Q_2 \text{ are the same} \\
& Q_1 \wedge \neg Q_2 &
\end{array}
$$

this *explains* SQL's restrictions (e.g., `UNION` compatibility)!

Computational Properties

- Evaluation of every query terminates
  $\Rightarrow$ relational calculus is not *Turing complete*
- **Data Complexity** in the size of the database, for a *fixed* query.
  $\Rightarrow$ in PTIME
  $\Rightarrow$ in LOGSPACE
  $\Rightarrow$ AC_0 (constant time on polynomially many CPUs in parallel)
- **Combined complexity**
  $\Rightarrow$ in PSPACE
  $\Rightarrow$ can express NP-hard problems (encode SAT)

What queries cannot be expressed in RC?

Note

RC is not Turing-complete
$\Rightarrow$ *there must be computable queries that cannot be written in RC.*

Built-in Operations

- ordering, arithmetic, string operations, etc.

Counting/Aggregation

- cardinality of sets (*parity*)

Reachability/Connectivity/...

- *paths in a graph (binary relation)*

Model extensions: Incompleteness/Inconsistency

- tuples with *unknown* (but existing) values
- incomplete relations and *open world assumption*
- conflicting information (e.g., from different data sources)