How do we ask Questions about Employees?

**Find all employees who work for “Bob”!**

Question: \( \{(x, y) \mid \text{EMP}(x, y, Bob)\} \)

Answer: \( \{(Sue, CS), (Bob, CO)\} \)

why? because (Sue, CS, Bob), etc., appear in EMP!

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

Q: \( \{(x_1, x_2) \mid \exists y_1, y_2, z. \text{EMP}(x_1, y_1, z) \land \text{EMP}(x_2, y_2, z)\} \)

A: \( \{(Sue, Bob), (Fred, John), (Jim, Eve)\} \) ← is that all?

**Find employees who are their own bosses!**

Q: \( \{(x) \mid \exists y. \text{EMP}(x, y, x)\} \)

A: \( \{(Sue), (Bob)\} \)

---

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 \( D \) with equality \( (=) \)
- **Relation**
  - schema: name \( R \), arity \( k \) (the number of attributes)
  - instance: a relation \( R \subseteq D^k \)
- **Database**
  - schema: finite set of relation schemes
  - instance: a relation \( R_i \) for each \( R_i \)

**Notation**

*Signature:* \( \rho = (R_1, \ldots, R_n) \)

*Instance:* \( D = (D, =, R_1, \ldots, R_n) \)

Examples of Relational Structures a.k.a. Databases

- the integer numbers with addition and multiplication:
  \( \rho = (\text{plus, times}) \)
  \( D = (\mathbb{Z}, =, \text{plus, times}) \)
- a Bibliography Database
- ...

Example: Bibliography

Relations (signatures) used in examples:

- author(\text{aid, name})
- wrote(\text{author, publication})
- publication(\text{pubid, title})
- book(\text{pubid, publisher, year})
- journal(\text{pubid, volume, no, year})
- proceedings(\text{pubid, year})
- article(\text{pubid, crossref, startpage, endpage})

⇒ names of attributes will be important later (for SQL)

Example (sample instance)

\begin{align*}
\text{author} &= \{ (1, \text{John}), (2, \text{Sue}) \} \\
\text{wrote} &= \{ (1, 1), (1, 4), (2, 3) \} \\
\text{publication} &= \{ (1, \text{Mathematical Logic}), (3, \text{Trans. Databases}), (2, \text{Principles of DB Syst.}), (4, \text{Query Languages}) \} \\
\text{book} &= \{ (1, \text{AMS, 1990}) \} \\
\text{journal} &= \{ (3, 35, 1, 1990) \} \\
\text{proceedings} &= \{ (2, 1995) \} \\
\text{article} &= \{ (4, 2, 30, 41) \}
\end{align*}
Simple (Atomic) “Truth”

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

⇒ that’s why relations are often called “tables”.

---

**Queries**

**IDEA1:** use *variables* to collect answers

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

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

Logical connectives:
- Conjunction (and): \(\text{author}(x, y) \land \text{wrote}(x, z)\)
- Disjunction (or): \(\text{author}(x, y) \lor \text{publication}(x, y)\)
- Negation (not): \(\neg \text{author}(x, y)\)

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

------

**Relational Calculus: Summary**

**Definition (Syntax)**

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

\[ Q := R(x_1, \ldots, x_k) \mid x_i = x_j \mid Q_1 \land Q_2 \mid \exists x_i. Q \mid Q_1 \lor Q_2 \mid \neg Q \]

**Definition (Answer tuples and Answer(s))**

An answer tuple \(t\) for \(Q\) assigns values to \(Q\)'s (free) variables.

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

- \(t \in R(x_1, \ldots, x_k)\) if \((t(x_1), \ldots, t(x_k)) \in R^D\)
- \(t \in x_i = x_j\) if \(t(x_i) = t(x_j)\)
- \(t \in Q_1 \land Q_2\) if \(t \in Q_1\) and \(t \in Q_2\)
- \(t \in Q_1 \lor Q_2\) if \(t \in Q_1\) or \(t \in Q_2\)
- \(t \in \neg Q_1\) if it is not the case that \(t \in Q_1\)
- \(t \in \exists x_i. Q_i\) if \(t[x_i \mapsto v] \in Q_i\) for some value \(v\)

An answer to \(Q\) is \(\{t \mid t \in Q\}\) (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. EMP(x_1, y_1, z) \land EMP(x_2, y_2, z) \}\nA: \{ (Sue, Fred), \ldots \}

because:

1. \[ x_1 \mapsto Sue, y_1 \mapsto CS, z \mapsto Bob \in EMP(x_1, y_1, z) \]
2. \[ x_2 \mapsto Fred, y_2 \mapsto CO, z \mapsto Bob \in EMP(x_2, y_2, z) \]
3. \[ x_1 \mapsto Sue, y_1 \mapsto CS, x_2 \mapsto Fred, y_2 \mapsto CO, z \mapsto Bob \in EMP(x_1, y_1, z) \land EMP(x_2, y_2, z) \]
4. \[ x_1 \mapsto Sue, x_2 \mapsto Fred \in \exists y_1, y_2, z. EMP(x_1, y_1, z) \land EMP(x_2, y_2, z) \]

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
- \( \neg (\neg Q_1) \equiv Q_1 \)
- \( Q_1 \lor Q_2 \equiv (\neg Q_1 \land \neg Q_2) \)
- \( Q_1 \rightarrow Q_2 \equiv \neg Q_1 \lor Q_2 \)
- \( Q_1 \leftrightarrow Q_2 \equiv (Q_1 \rightarrow Q_2) \land (Q_2 \rightarrow Q_1) \)
- \( \ldots \)

First-order Equivalences
- \( \forall x. Q \equiv \neg \exists x. \neg Q \)

Find the neutral element (of addition)!
Question: \{ (x) \mid PLUS(x, x, x) \}\nAnswer: \{ (0) \}\n
but shouldn’t the query really be
\{ (x) \mid \forall y. PLUS(x, y, y) \land PLUS(y, x, y) \}\n
\( (*) \)

IDEA
(\( * \)) is the same as \{ (x) \mid \forall y. PLUS(x, y, y) \}\nbecause PLUS is commutative
is the same as \{ (x) \mid PLUS(x, x, x) \}\nbecause PLUS is monotone

⇒ Laws of Arithmetic for Natural Numbers
Laws a.k.a. Integrity Constraints

Idea

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

- Addition is commutative
  \[ \forall x, y, z. \text{PLUS}(x, y, z) \rightarrow \text{PLUS}(y, x, z) \]
  \[ (\neg \exists x, y, z. \text{PLUS}(x, y, z) \land \neg \text{PLUS}(y, x, z)) \]

- Addition is a (relational representation of a) binary function
  \[ \forall x, y, z_1, z_2. \text{PLUS}(x, y, z_1) \land \text{PLUS}(x, y, z_2) \rightarrow z_1 = z_2 \]
  \[ (\neg \exists x, y, z_1, z_2. \text{PLUS}(x, y, z_1) \land \text{PLUS}(x, y, z_2) \land (z_1 = z_2)) \]

- Addition is a total function
  \[ \forall x, y. \exists z. \text{PLUS}(x, y, z) \]

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

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 (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”.

Story so far...

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

... so is there anything new here?

⇒ YES: database instances must be finite

Views and Integrity Constraints

Idea

Answers to queries can be used to define derived relations (views) ⇒ 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 \( Q \).

Unsafe Queries

- \( \neg \exists x. \text{author}(x, y) \)
- \( \text{book}(x, y, z) \lor \text{proceedings}(x, y) \)
- \( x = y \)

⇒ we want only queries with finite answers (over finite instances).

Definition (Range restricted queries)

\[ Q ::= R(x_1, \ldots, x_k) \]
\[ Q_1 \land Q_2 \]
\[ Q \land x_i = x_j \]
\[ \exists x_i. Q \]
\[ Q_1 \lor Q_2 \]
\[ Q_1 \land \neg Q_2 \]

at least one of \( x_i, x_j \) is answer variable of \( Q \)
answer variables of \( Q_1 \) and \( Q_2 \) are the same

this explains SQL’s restrictions (e.g., \text{UNION} compatibility)!
Computational Properties

- Evaluation of every query terminates
  ⇒ relational calculus is not Turing complete

- **Data Complexity** in the size of the database, for a *fixed* query.
  ⇒ in PTIME
  ⇒ in LOGSPACE
  ⇒ \( \text{AC}_0 \) (constant time on polynomially many CPUs in parallel)

- **Combined complexity**
  ⇒ in PSPACE
  ⇒ can express NP-hard problems (encode SAT)

---

What queries cannot be expressed in RC?

**Note**

\( \text{RC is not Turing-complete} \)

⇒ 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)