Extensions of the Relational Model

David Toman

School of Computer Science
University of Waterloo

Introduction to Databases CS348
Why?

1. To simplify *Data Modeling*
   - representing complex values conveniently
     \[\Rightarrow\text{records/sets/... as first-class citizens}\]
   - avoiding need for *(foreign) key invention*

2. To enhance *Expressive Power*
   - enabling *transitive closure*, etc.
Outline

More structure:

- OO models (essentially records)
- complex object model (sets as objects)

More expressive power:

- iteration and recursion
  ⇒ Datalog
  ⇒ Fixed-point Logics
- second-order quantification

More interpretation:

- constraint model
  ⇒ in the queries
  ⇒ as representation of databases
Idea

Allow non-atomic objects as values of attributes.

Records

- allow attribute values to be (other) records
- preferably w/o pointers

Sets

- allow attribute values to be sets of (other) objects
Classes of Objects (and language extensions)

Structure Data: \( C = \{ A_1 : C_1, \ldots, A_k : C_k \} \)

Example

\[
\begin{align*}
\text{Emp} & \quad = \{ \text{name: String, salary: Int, dept: Dept} \} \\
\text{Dept} & \quad = \{ \text{name: String, boss: Emp} \}
\end{align*}
\]

Query Extensions:

- variables range over "objects"
- attributes can form path expressions

Example

\[
\begin{align*}
\text{SELECT} & \quad e.\text{name}, \ e.\text{dept}.\text{boss}.\text{name} \\
\text{FROM} & \quad \text{Emp e, Dept d} \\
\text{WHERE} & \quad e.\text{dept} = d, \ e.\text{salary} > e.\text{dept}.\text{boss}.\text{salary}
\end{align*}
\]
Reification

How can we understand this?

Idea (Reification)

1. \textit{classes} = \textit{unary relations} (sets)
2. \textit{attributes} = \textit{binary relations} (functions)
3. \ldots \textit{and integrity constraints} (e.g., \textit{attributes are functions})

Example

- \texttt{Emp(x) and Dept(x)}
- \texttt{name(x, y), salary(x, y), dept(x, y), and boss(x, y)}
- \textit{integrity constraints (for dept)}:
  \[
  \forall x, y, z. \text{dept}(x, y) \land \text{dept}(x, z) \rightarrow y = z \\
  \forall x. \text{Emp}(x) \rightarrow \exists y. \text{dept}(x, y) \\
  \forall x, y. \text{dept}(x, y) \rightarrow \text{Dept}(y)
  \]
Records

Are Tuples Objects themselves?
  • if so, what is their “id”?

Problems with “oid” invention
  • when are two tuples equal?
  • structural equality may not work (cycles!)

... in general, we can do without.
Sets

how are sets created?

• as subsets of the universe/active domain
• as groups

what can we do with them in queries?

• variables range over objects of particular structure
• additional operations for/between such values:
  • \( x \in y \) for set membership
  • \( \{x|\varphi\} \) for set-value construction
• quantifiers can also range over sets
Example: Transitive Closure

Expressing non-first-order queries:

Example (Transitive Closure)

\[
\forall x. \left( \forall w, u. E(w, u) \rightarrow (w, u) \in x \\
\forall w, v, u. (w, v) \in x \land (v, u) \in x \rightarrow (w, u) \in x \right) \rightarrow (y, z) \in x
\]

Extremely Powerful:

Theorem

Every elementary query can be expressed in RC w/Powerset.

- can create powersets (exponential in size)
- can nest powerset construction
Nesting and Unnesting

What’s the problem with Powerset? \( \Rightarrow \) too many new objects!

Nesting/Unnesting

Idea

Only allow construction of sets for which we can create a name from values we already have in the database.

\( \Rightarrow \) just like the GROUP BY clause.

Theorem

Queries over flat databases are equivalent to flat queries.
Adding Recursion & Datalog

Can we have some middle ground (PTIME)?

Idea

Allow inductive definitions of relations of the form

- certain base tuples are in the relation
- if a tuple is in the relation then another is too
- nothing else is in the relation

Transitive closure revisited:

**Example**

\[
\text{tc}(x, y) \leftarrow E(x, y) \\
\text{tc}(x, y) \leftarrow E(x, z), \text{tc}(z, y)
\]
Models and Fixpoints

What does it mean?

Idea

*Consider tuples common to all databases (models) consistent with the inductive definition.*

How do we compute it?

\[
\text{inductive definition} = \text{monotone functional} \\
\text{fixpoint of the functional} = \text{minimal model [Knaster-Tarski]} \\
\text{iteration terminates (finitely many tuples)}
\]

Fixpoint definition:

\[
\mu t \cdot E(x, y) \lor \exists z. E(x, z) \land t \cdot c(z, y)
\]
What Happened to Negation?

Note

The inductive definition must provide a monotone functional

We cannot use general negation

- restrict to positive queries
- restrict to even number of negations
- more complex definition of minimal model
  - well-founded semantics
  - stable-model semantics
Caveat Emptor in SQL

Note

The framework only works for set semantics.

In SQL it breaks when:

- you use duplicate semantics
- you use arithmetic/string functions/. . .
- you use `EXCEPT`
- . . .
Constraint Extensions

How to capture large but regular sets?

Idea

Use characteristic formulas in tuples

Example

<table>
<thead>
<tr>
<th>Roads</th>
<th>Name</th>
<th>Geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Geometry</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$4 \leq x \leq 6 \land x + 2y = 10$</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>$3 \leq x \leq 4 \land 3x + y = 15$</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>$0 \leq x \leq 4 \land 4y - x = 8$</td>
<td></td>
</tr>
<tr>
<td>401</td>
<td>$4 \leq x \leq 6 \land x - y = 4$</td>
<td></td>
</tr>
<tr>
<td>401</td>
<td>$6 \leq x \leq 8 \land x - 2y = 2$</td>
<td></td>
</tr>
</tbody>
</table>
Constraint Queries

• for $R(x, y)$ a set of points; do all the points lie on a straight line?

$$\exists a, b. \forall x, y. R(x, y) \supset y = a \times x + b$$

⇒ is this “safe”? How can we find out $a$ and $b$?

• do relations $R(x, y)$ and $S(x, y)$ intersect in the plane?

$$\exists x, y. R(x, y) \land S(x, y)$$

⇒ imagine $R$ stores “objects” rather than points...
Summary

Extensions allow:

- (more) convenient modeling of data
- (more) expressive queries
- (more) compact representation

Note

Understanding why things work depends on understanding LOGICAL FOUNDATIONS

Also helps with understanding properties, . . .