Module 2: The Relational Model

Spring 2022

Cheriton School of Computer Science

CS 348: Intro to Database Management

Reading Assignments and References

To be read during the Week of May 9–13:

- Chapter 2 of course textbook.[1] (Material in Sections 2.3 and 2.6 will be covered in later modules.)
- Section 27.2 of Chapter 27 of course textbook, available online at `db-book.com`.

References

1. Abiteboul, Hull and Vianu, *Foundations of Databases*. A book available online at `http://webdam.inria.fr/Alice/`.
2. *Backus-Naur Form*, wiki page.

[1] Silberschatz, Korth and Sudarshan, *Database Systems Concepts*, 7^{th} edition

Outline

A Basic Syntax for Asking Questions and for Answers

To begin with, assume ...

- Set comprehension syntax for queries:

$$\{\langle \textit{answer} \rangle \mid \langle \textit{condition} \rangle\}.$$

- Syntax for each $\langle \textit{answer} \rangle$ is a k-tuple of *variables*:

$$(x_1, \ldots, x_k).$$

- Answers to a query:

  all k-*tuples* $(c_1, \ldots, c_k)$ of *constants*
  denoting *values* for each variable x_i that satisfy $\langle \textit{condition} \rangle$.

Asking Questions about Natural Numbers

What are all pairs of natural numbers that add to 5?

Question: $\{(x, y) \mid x + y = 5\}$ or $\{(x, y) \mid \text{PLUS}(x, y, 5)\}^\dagger$

Answers: $\{(0, 5), (1, 4), (2, 3), (3, 2), (4, 1), (5, 0)\}$

Why? Because $(0, 5, 5)$, etc., appear in table PLUS!

What are all pairs of numbers that add to the same number they subtract to, where $x + y = x - y$?

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

Answers: $\{(0, 0), (1, 0), \ldots\}$ Is $(5, 5)$ also an answer?

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

What is the neutral element of addition?

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

Answers: $\{(0)\}$

† A *relational* form for basic conditions.

Table PLUS

A1	A2	R
0	0	0
0	1	1
0	2	2
⋮	⋮	⋮
1	0	1
1	1	2
⋮	⋮	⋮
2	0	2
2	1	3
⋮	⋮	⋮

Asking Questions about Employees

Who are all the employees and their departments who work for Bob?

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

Answers: $\{(Sue, \text{CS}), (Bob, \text{CO})\}$

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

Who are pairs of employees 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: $\{(Sue, Bob), (Fred, John), (Jim, Eve)\}$ $\leftarrow$ Is that all?

Who are the employees who are their own bosses?

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

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

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*

Relational Databases and the Relational Calculus

Based on *first order predicate logic* (FOL) and *Tarskian semantics*.

Recall example RM database using a common visualization:

AUTHOR

aid	name
1	Sue
2	John

WROTE

author	publication
1	1
1	4
2	2
1	2

PUBLICATION

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

Idea

All information is organized in a finite number of relations called *tables*.

Features:

- simple and clean data model accommodating data independence,
- declarative DML based on *well-formed formulas* in FOL, and
- *integrity constraints* also via well-formed formulas.

Relational Databases

Components:

Universe
- a set of values $\mathbf{D}$ (*domain*) with *equality* ($\approx$), *and with constants for each value*.

Relation (also called a table)
- intension: a relation name (*predicate name*) R, and *arity* k of R (the number of columns), written R/k, and
- extension: a set of k-tuples (*interpretation*) $\mathbf{R} \subseteq \mathbf{D}^k$.

Database
- signature (metadata): finite set ρ of predicate names R_i; and
- instance (data, *structure*): an extension $\mathbf{R_i}$ for each R_i.

Notation

Signature: $\rho = (R_1/k_1, \ldots, R_n/k_n)$ Instance: $\mathbf{DB} = (\mathbf{D}, \approx, \mathbf{R_1}, \ldots, \mathbf{R_n})$

Examples of Relational Databases

- The integers, with addition *and multiplication*:

  (signature) $\rho = (\texttt{PLUS/3}, \texttt{TIMES/3})$
  (data) $\mathbf{DB} = (\mathbb{Z}, \approx, \mathbf{PLUS}, \mathbf{TIMES})$

- The employee database:

  (signature) $\rho = (\texttt{EMP/3})$
  (data) $\mathbf{DB} = (\mathbb{STR}, \approx, \mathbf{EMP})$

- The simple bibliography database:

  (signature) $\rho = (\texttt{AUTHOR/2}, \texttt{WROTE/2}, \texttt{PUBLICATION/2})$
  (data) $\mathbf{DB} = (\mathbb{STR} \uplus \mathbb{Z}, \approx, \mathbf{AUTHOR}, \mathbf{WROTE}, \mathbf{PUBLICATION})$

Bibliography Relational Database, Version 2

(signature) $\rho = ($

```
AUTHOR (aid, name),
WROTE (author, publication),
PUBLICATION (pubid, title),
BOOK (pubid, publisher, year),
JOURNAL-OR-PROCEEDINGS (pubid),
JOURNAL (pubid, volume, no, year),
PROCEEDINGS (pubid, year),
ARTICLE (pubid, appears-in, startpage, endpage)
```

$)$

Arity is indicated by a sequence of *identifiers*, called attributes:

- Help with understanding semantics; and
- Used in some DMLs, such as some *relational algebras* and SQL.

Bibliography Relational Database, Version 2 (cont'd)

(data) **DB** $= (\mathbb{STR} \uplus \mathbb{Z}, \approx,$

$$
\begin{array}{rcll}
\texttt{AUTHOR} & = & \{ & (1, \text{Sue}), (2, \text{John}) \quad \}, \\
\texttt{WROTE} & = & \{ & (1,1), (1,4), (1,2), (2,2) \quad \}, \\
\texttt{PUBLICATION} & = & \{ & (1, \text{Mathematical Logic}), \\
 & & & (3, \text{Trans. on Databases}), \\
 & & & (2, \text{Principles of DB Systems}), \\
 & & & (4, \text{Query Languages}) \quad \}, \\
\texttt{BOOK} & = & \{ & (1, \text{AMS}, 1990) \quad \}, \\
\texttt{JOURNAL-OR-PROCEEDINGS} & = & \{ & (2), (3) \quad \}, \\
\texttt{JOURNAL} & = & \{ & (3, 35, 1, 1990) \quad \}, \\
\texttt{PROCEEDINGS} & = & \{ & (2, 1995) \quad \}, \\
\texttt{ARTICLE} & = & \{ & (4, 2, 30, 41) \quad \}
\end{array}
$$

$)$

Simple (Atomic) "Truth"

Idea

Relationships between values (tuples) that are *present* in an instance are *true*; relationships *absent* are *false*.

In the sample *bibliography* database instance:

- "John" is the name of an author with id "2" since: $(2, \text{John}) \in \texttt{AUTHOR}$;
- "Mathematical Logic" is the title of a publication since: $(1, \text{Mathematical Logic}) \in \texttt{PUBLICATION}$;
- Moreover, it is a book published by "AMS" in "1990" since: $(1, \text{AMS}, 1990) \in \texttt{BOOK}$;
- John wrote "Principles of DB Systems" since: $(2, 2) \in \texttt{WROTE}$;
- John has **NOT** written "Trans. on Databases" since: $(2, 3) \notin \texttt{WROTE}$;
- etc.

Query Conditions

Idea

Use variables and valuations to generalize conditions.

Example: AUTHOR(x, y) will be true of any *valuation* $\{x \mapsto v_1, y \mapsto v_2, \ldots\}$ exactly when the 2-tuple of values (v_1, v_2) occurs in **AUTHOR**.

Valuation

A valuation is a function θ that maps *variable names* to values in the universe:

$$\theta : \{x_1, x_2, \ldots\} \to \mathbf{D}.$$

To denote a modification to θ in which variable x is instead mapped to value v, one writes:

$$\theta[x \mapsto v].$$

Query Conditions (cont')

Idea

Allow more complex conditions to be built from simpler conditions with ...

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...): $\exists x.\texttt{author}(x, y)$

Examples:

- $\exists z.\texttt{PLUS}(x, y, z) \wedge \texttt{PLUS}(z, y, x)$, or
- $\exists y_1, y_2, z.\texttt{EMP}(x_1, y_1, z) \wedge \texttt{EMP}(x_2, y_2, z)$.

Summarizing, allow conditions to be *well-formed formulas* (wffs) in the language of FOL.

Relational Calculus

Conditions

Given a database signature $\rho = (R_1/k_1, \ldots, R_n/k_n)$, a set of variable names $\{x_1, x_2, \ldots\}$ and a set of constants $\{c_1, c_2, \ldots\}$, *conditions* are *formulas* defined by the grammar:

$$\varphi ::= \underbrace{\underbrace{\underbrace{R_i(x_{i,1}, \ldots, x_{i,k_i}) \mid x_i = x_j \mid x_i = c_j \mid \varphi_1 \wedge \varphi_2 \mid \exists x_i.\varphi_1}_{\text{conjunctive formulas}} \mid \varphi_1 \vee \varphi_2}_{\text{positive formulas}} \mid \neg\varphi_1}_{\text{first-order formulas}}$$

A condition is a *sentence* when it has no free variables.

The meta-language used to define the grammar is Backus-Naur form. (See wiki for a good overview.)

Relational Calculus (cont')

Free Variables

The *free variables* of a formula φ, written $\mathrm{Fv}(\varphi)$, are defined as follows:

$$
\begin{array}{lcl}
\mathrm{Fv}(R(x_{i_1}, \ldots, x_{i_k})) & = & \{x_{i_1}, \ldots, x_{i_k}\}; \\
\mathrm{Fv}(x_i = x_j) & = & \{x_i, x_j\}; \\
\mathrm{Fv}(x_i = c_j) & = & \{x_i\}; \\
\mathrm{Fv}(\varphi \wedge \psi) & = & \mathrm{Fv}(\varphi) \cup \mathrm{Fv}(\psi); \\
\mathrm{Fv}(\exists x_i.\varphi) & = & \mathrm{Fv}(\varphi) - \{x_i\}; \\
\mathrm{Fv}(\varphi \vee \psi) & = & \mathrm{Fv}(\varphi) \cup \mathrm{Fv}(\psi); \text{ and} \\
\mathrm{Fv}(\neg\varphi) & = & \mathrm{Fv}(\varphi).
\end{array}
$$

Semantics for Conditions

When a Condition is True (Tarski)

The *truth* of a formula φ over a signature $\rho = (R_1/k_1, \ldots R_n/k_n)$ is defined with respect to

1. a **database instance** $\mathbf{DB} = (\mathbf{D}, \approx, \mathbf{R}_1, \ldots, \mathbf{R}_n)$, and
2. a **valuation** $\theta : \{x_1, x_2, \ldots\} \to \mathbf{D}$

as follows:

$$
\begin{array}{ll}
\mathbf{DB}, \theta \models R_i(x_{i,1}, \ldots, x_{i,k_i}) & \text{if } (\theta(x_{i,1}), \ldots, \theta(x_{i,k_i})) \in \mathbf{R}_i; \\
\mathbf{DB}, \theta \models x_i = x_j & \text{if } \theta(x_i) \approx \theta(x_j); \\
\mathbf{DB}, \theta \models x_i = c_j & \text{if } \theta(x_i) \approx c_j; \\
\mathbf{DB}, \theta \models \varphi \wedge \psi & \text{if } \mathbf{DB}, \theta \models \varphi \text{ and } \mathbf{DB}, \theta \models \psi; \\
\mathbf{DB}, \theta \models \exists x_i.\varphi & \text{if } \mathbf{DB}, \theta[x_i \mapsto v] \models \varphi, \text{ for some } v \in \mathbf{D}; \\
\mathbf{DB}, \theta \models \varphi \vee \psi & \text{if } \mathbf{DB}, \theta \models \varphi \text{ or } \mathbf{DB}, \theta \models \psi; \text{ and} \\
\mathbf{DB}, \theta \models \neg\varphi & \text{if } \mathbf{DB}, \theta \not\models \varphi.
\end{array}
$$

Equivalences and Syntactic Sugar

Boolean Equivalences

- $\neg(\neg\varphi_1) \equiv \varphi_1$
- $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$
- $\varphi_1 \to \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$
- $\varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$
- $\ldots$

First-order Equivalences

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

Additional Syntactic Sugar

- $R(\ldots, c, \ldots) \equiv \exists x.(R(\ldots, x, \ldots) \wedge x = c)$, where x is *fresh*
- $\exists x_1, \cdots, x_n.\varphi \equiv \exists x_1.\cdots.\exists x_n.\varphi$
- $R(\ldots, -, \ldots) \equiv \exists x.R(\ldots, x, \ldots)$, where x is *fresh*

Relational Calculus (cont'd)

Relational Calculus (RC) Query

A *query* in the relational calculus is a set comprehension of the form

$$\{(x_1, \ldots, x_k) \mid \varphi\},$$

where $\{x_1, \ldots, x_k\} = \mathrm{Fv}(\varphi)$ (are the free variables of φ).

Also:

- a conjunctive query is where φ is a conjunctive formula, and
- a positive query is where φ is a positive formula.

Query Answers

The *answers* to a *query* $\{(x_1, \ldots, x_k) \mid \varphi\}$ over **DB** is the **relation**

$$\{(\theta(x_1), \ldots, \theta(x_k)) \mid \mathbf{DB}, \theta \models \varphi\}.$$

Answers to queries: valuations applied to tuples of variables that make the formula true with respect to a database.

Example Justification of an Answer to an RC Query

Who are pairs of employees 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: $\{(\textit{Jim}, \textit{Eve}), \ldots\}$

Because:

1. $\textbf{DB}, \theta_1(= \{x_1 \mapsto \textit{Jim}, y_1 \mapsto \text{CS}, z \mapsto \textit{Fred}, \ldots\}) \models \text{EMP}(x_1, y_1, z)$
2. $\textbf{DB}, \theta_2(= \{x_2 \mapsto \textit{Eve}, y_2 \mapsto \text{CS}, z \mapsto \textit{Fred}, \ldots\}) \models \text{EMP}(x_2, y_2, z)$
3. $\textbf{DB}, \theta_3(= \{x_1 \mapsto \textit{Jim}, y_1 \mapsto \text{CS}, x_2 \mapsto \textit{Eve}, y_2 \mapsto \text{CS}, z \mapsto \textit{Fred}, \ldots\})$
   $\models \text{EMP}(x_1, y_1, z) \wedge \text{EMP}(x_2, y_2, z)$
4. $\textbf{DB}, \theta_4(= \{x_1 \mapsto \textit{Jim}, x_2 \mapsto \textit{Eve}, \ldots\})$
   $\models \exists y_1, y_2, z.\text{EMP}(x_1, y_1, z) \wedge \text{EMP}(x_2, y_2, z)^\dagger$
5. $(\theta_4(x_1), \theta_4(x_2)) = (\textit{Jim}, \textit{Eve})$

where $\rho = (\mathtt{EMP}/3)$, and $\textbf{DB} = (\mathbb{STR}, \approx, \textbf{EMP})$.

† Check that $\{x_1, x_2\} = \text{Fv}\left(\exists y_1, y_2, z.\text{EMP}(x_1, y_1, z) \wedge \text{EMP}(x_2, y_2, z)\right)$.

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*

More Examples of RC Queries

Over signature $\rho =$ (EMP (name, dept, boss)):

1. *Who are the bosses that manage at least two employees?*

$$\{(b) \mid \exists e_1, e_2.(\exists d_1.\mathrm{EMP}(e_1, d_1, b)) \wedge (\exists d_2.\mathrm{EMP}(e_2, d_2, b)) \wedge \neg(e_1 = e_2)\}$$

   (or more simply, with the aid of some syntactic sugar)

$$\{(b) \mid \exists e_1, e_2.\mathrm{EMP}(e_1, -, b)) \wedge \mathrm{EMP}(e_2, -, b) \wedge \neg(e_1 = e_2)\}$$

2. *Who are the bosses that do not manage more than two employees?*

$$\begin{aligned}\{(b) \mid \exists e_1.\mathrm{EMP}(e_1, -, b) & \\ & \wedge \neg\exists e_2, e_3.\mathrm{EMP}(e_2, -, b) \wedge \mathrm{EMP}(e_3, -, b) \\ & \wedge \neg(e_1 = e_2 \vee e_1 = e_3 \vee e_2 = e_3))\}\end{aligned}$$

Choose variable names suggestive of what values or (indirectly) entities they refer to, e.g.:

- "e_1" refers indirectly to an employee, and
- "b" refers indirectly to a boss.

Exercises

1. Over the PLUS-TIMES relational database, with

   signature $\rho = (\texttt{PLUS/3}, \texttt{TIMES/3})$, and

   instance $\mathbf{DB} = (\mathbb{N}, \approx, \mathbf{PLUS}, \mathbf{TIMES})$:†

   1.1 *What are all composite numbers?*
   1.2 *What are all prime numbers?*

2. Over the bibliography relational database, 2nd version:

   2.1 *What are all publication titles?*
   2.2 *What are the publication titles that are journals or proceedings?*
   2.3 *What are the titles of all books?*
   2.4 *What are the publications without authors?*
   2.5 *What are all the ordered pairs of coauthor names?*
   2.6 *What are all publication titles written by a single author?*

† Much harder over the integers $\mathbb{Z}$.

Outline

Asking Questions about Natural Numbers (revisited)

What is the neutral element of addition?

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

Answers: $\{(0)\}$

But shouldn't the query really be

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

Observation

$(*)$ is the same as

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

because PLUS is commutative! And $(**)$ is the same as

$$\{(x) \mid \text{PLUS}(x, x, x)\}$$

because PLUS is monotone!

PLUS should satisfy integrity constraints that are the *laws of arithmetic* for natural numbers.

Table PLUS

A1	A2	R
0	0	0
0	1	1
0	2	2
⋮	⋮	⋮
1	0	1
1	1	2
⋮	⋮	⋮
2	0	2
2	1	3
⋮	⋮	⋮

Integrity Constraints for Addition

Sentences that should always be *true* for any extension of PLUS over the domain of natural numbers:

- *Addition is commutative:*

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

- *PLUS is a relational representation of a binary function:*

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

- *Addition is a total function:*

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

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

Integrity Constraints for Employees

Sentences that should always be *true* for any extension of table

EMP (name, dept, boss) :

- *Every boss is an employee:*

$$\forall e, d_1, b_1.\mathrm{EMP}(e, d_1, b_1) \rightarrow \exists d_2, b_2.\mathrm{EMP}(b_1, d_2, b_2))$$
$$\forall b.\mathrm{EMP}(-, -, b) \rightarrow \mathrm{EMP}(b, -, -)^{\dagger}$$

- *Every boss manages a unique department:*

$$\forall e_1, e_2, d_1, d_2, b.\mathrm{EMP}(e_1, d_1, b) \wedge \mathrm{EMP}(e_2, d_2, b) \rightarrow d_1 = d_2$$
$$\forall d_1, d_2.(\exists b.\mathrm{EMP}(-, d_1, b) \wedge \mathrm{EMP}(-, d_2, b)) \rightarrow d_1 = d_2$$

- *No boss has someone else as their boss:*

$$\forall e, b_1, b_2.\mathrm{EMP}(e, -, b_1) \wedge \mathrm{EMP}(b_1, -, b_2) \rightarrow b_1 = b_2$$
$$\forall b_1, b_2.(\mathrm{EMP}(-, -, b_1) \wedge \mathrm{EMP}(b_1, -, b_2)) \rightarrow b_1 = b_2$$

† Exercise: Show why this is equivalent.

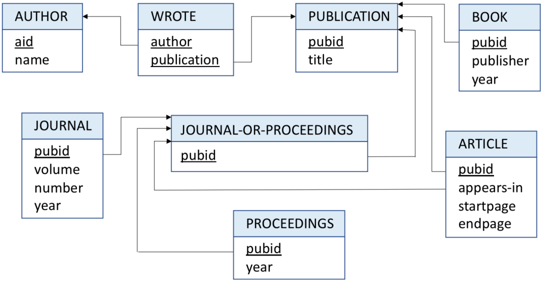
Integrity Constraints Generally

A relational *signature* captures only the structure of relations.

Valid database instances satisfy additional *integrity constraints* in the form of sentences over the signature.

- 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 or foreign keys).
- Values cannot appear simultaneously in certain relations (disjointness).
- Values in a relation must appear in at least one of another set of relations (coverage).
- etc.

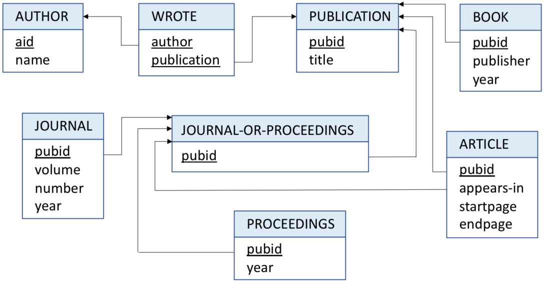
Bibliography Integrity Constraints



Typing Constraints / Domain Contraints

- *Author id's are integers.*
- *Author names are strings.*
- *Publication id's are integers.*
- *Publication titles are strings.*
- *etc.*

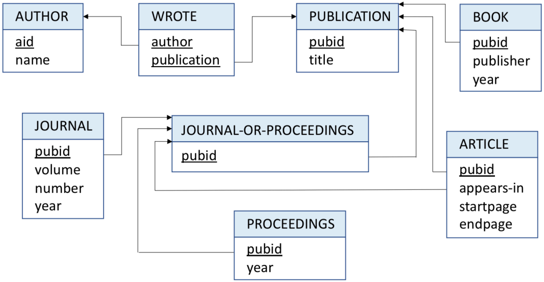
Bibliography Integrity Constraints (cont'd)



Uniqueness of Values / Identification (keys)

- *Author id's are unique and determine author names.*
- *Publication id's are unique as well.*
- *Articles can be identified by their publication id.*
- *Articles can also be identified by the publication id of the collection they have appeared in and their starting page number.*

Bibliography Integrity Constraints (cont'd)



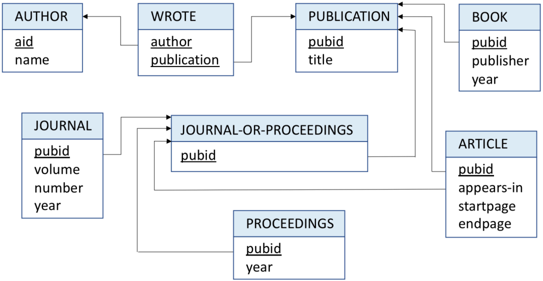
Referential Integrity / Foreign Keys

- *Books, journals, proceedings and articles are publications.*
- *The components of a WROTE tuple must be an author and a publication.*

Disjointness

- *Books are different from journals.*
- *Books are also different from proceedings.*

Bibliography Integrity Constraints (cont'd)



Coverage

- *Every publication is either a book, a journal, a proceedings, or an article.*
- *Every article appears in a journal or in a proceedings.*

Views and Integrity Constraints

The extension of a table can be determined by an integrity constraint.

The extension of table JOURNAL-OR-PROCEEDINGS *is the union of the publication id's occurring in table* JOURNAL *and in table* PROCEEDINGS*.*

$$\forall p.\text{JOURNAL-OR-PROCEEDINGS}(p) \leftrightarrow (\text{JOURNAL}(p) \vee \text{PROCEEDINGS}(p))$$

View

Given a signature ρ, a table R occurring in ρ is a *view* when the relational database schema contains exactly one integrity constraint of the form:

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

where $\{x_1, \ldots, x_k\} = \mathrm{Fv}(\varphi)$. Condition φ is called the *view definition* of R, and R is said to *depend on* any table mentioned in φ.

No table occurring in a schema is allowed to depend on itself, either directly or indirectly.

Relational Database Schemata and Consistency

Relational Database Schema

A *relational database schema* is a pair $\langle \rho, \Sigma \rangle$, where ρ is a signature, and where Σ is a finite set of integrity constraints that are sentences over ρ.

Relational Databases and Consistency

A *relational database* consists of a relational database schema $\langle \rho, \Sigma \rangle$ and an instance **DB** of its signature ρ.

The relational database is *consistent* if and only if, for any integrity constraint $\varphi \in \Sigma$ and any valuation θ:

$$\mathbf{DB}, \theta \models \varphi.$$

Outline

Story so far ...

databases	$\Leftrightarrow$	*relational structures*
queries	$\Leftrightarrow$	*set comprehensions with conditions as formulas in FOL*[†]
integrity constraints	$\Leftrightarrow$	*sentences in FOL*

So are there any remaining issues?

Yes!

Relational databases and RC[‡] queries should also have the following properties:

- The extension of any relation in a signature should be *finite*; and
- Queries should be safe: their answers should be *finite* when database instances are finite.

[†] *first order predicate logic*

[‡] *relational calculus*

Unsafe Queries

The set of answers to each of the following queries over the bibliography RDB is not finite:

Case 1 $\{(x, y) \mid x = y\}$

Case 2 $\{(pid, pub, year) \mid \mathrm{BOOK}(pid, pub, year) \vee \mathrm{PROCEEDINGS}(pid, year)\}$

Case 3 $\{(aname) \mid \neg\exists aid.\mathrm{AUTHOR}(aid, aname)\}$

Domain Independence

An RC query $\{(x_1, \ldots, x_k) \mid \varphi\}$ is *domain independent* when, for any pair of instances $\mathbf{DB}_1 = (\mathbf{D_1}, \approx, \mathbf{R_1}, \ldots, \mathbf{R_k})$ and $\mathbf{DB}_2 = (\mathbf{D_2}, \approx, \mathbf{R_1}, \ldots, \mathbf{R_k})$ and any θ, $\mathbf{DB}_1, \theta \models \varphi$ if and only if $\mathbf{DB}_2, \theta \models \varphi$.

Theorem

Let $(R_1, \ldots, R_k)$ be the signature of a relational database. Answers to domain independent queries contain only values *that occur in the extension* $\mathbf{R_i}$ *of any relation* R_i.

safety $\Leftrightarrow$ *domain independence* and *finite database instances*

Safety and Query Satisfiability

Theorem

Satisfiability of RC queries† is undecidable;

- co recursively enumerable in general, and
- recursively enumerable for finite databases.

† *Is there a database for which the answer is non-empty?*

Proof

Reduction from PCP (see Abiteboul *et. al.* book, p.122-126).

Theorem

Domain independence of RC queries is undecidable.

Proof

The query $\{(x, y) \mid (x = y) \land \varphi\}$ is satisfiable if and only if it is not domain independent.

Range Restricted RC

Range Restricted Conditions and Queries

Given a database signature $\rho = (R_1/k_1, \ldots, R_n/k_n)$, a set of variable names $\{x_1, x_2, \ldots\}$ and a set of constants $\{c_1, c_2, \ldots\}$, *range restricted conditions* are *formulas* defined by the grammar:

$$
\begin{array}{rcll}
\varphi & ::= & R_i(x_{i,1}, \ldots, x_{i,k_i}) & \\
& \mid & \varphi_1 \wedge (x_i = x_j) & \text{where } \{x_i, x_j\} \cap \mathrm{Fv}(\varphi_1) \neq \emptyset \text{ (case 1)} \\
& \mid & x_i = c_j & \\
& \mid & \varphi_1 \wedge \varphi_2 & \\
& \mid & \exists x_i.\varphi_1 & \\
& \mid & \varphi_1 \vee \varphi_2 & \text{where } \mathrm{Fv}(\varphi_1) = \mathrm{Fv}(\varphi_2) \text{ (case 2)} \\
& \mid & \varphi_1 \wedge \neg\varphi_2 & \text{where } \mathrm{Fv}(\varphi_1) = \mathrm{Fv}(\varphi_2) \text{ (case 3)}
\end{array}
$$

A *range restricted RC query* has the form $\{(x_1, \ldots, x_n) \mid \varphi\}$, where $\{x_1, \ldots, x_n\} = \mathrm{Fv}(\varphi)$ and where φ is a range restricted condition.

A query language for the relational model is *relationally complete* if the language is at least as expressive as the range restricted RC.

Range Restricted RC (cont'd)

Theorem

Every range restricted RC query is an RC query and is domain independent.

Proof Outline

Both claims follow by simple inductions on the form of a range restricted condition. Exercise: Details.

Do we lose expressiveness by requiring conditions in RC queries to be range restricted?

Theorem

Every domain independent RC query has an equivalent formulation as a range restricted RC query.

Proof Outline

1. Restrict every variable in φ to the *active domain*, and
2. express the active domain using a *unary query* over the database instance.

Exercise: Details.

Computational Properties of Query Answering

- There is an algorithm for computing the answers to any range restricted RC query.
  $\Rightarrow$ range restricted RC is not *Turing complete*.

- The data complexity, that is, complexity in the size of the database for a *fixed query* is
  $\Rightarrow$ in PTIME,
  $\Rightarrow$ in LOGSPACE, and
  $\Rightarrow$ AC_0 (i.e., constant time on polynomially many CPUs in parallel).

- The combined complexity, that is, complexity in size of the query and the database, is
  $\Rightarrow$ in PSPACE,
  (since queries can express NP-hard problems such as SAT).

Outline

Query Evaluation versus Query Satisfiability

Query Evaluation

Given an RC query $\{(x_1, \ldots, x_k) \mid \varphi\}$ and a finite database instance **DB**, find all answers to the query.

Query Satisfiability

Given an RC query $\{(x_1, \ldots, x_k) \mid \varphi\}$, determine whether there is a finite database instance **DB** for which the answer is non-empty.

- Much harder problem, in fact, undecidable.
- Same as the problem of query containment which is fundamental in *query compilation*.
- Can be solved for fragments of RC.

Relationship to Theorem Proving

Query Subsumption

A query $\{(x_1, \ldots, x_k) \mid \varphi_1\}$ *subsumes* a query $\{(x_1, \ldots, x_k) \mid \varphi_2\}$ *with respect to a relational database schema* $\langle \rho, \Sigma \rangle$ if, for every instance **DB** of the schema such that $\mathbf{DB}, \theta \models \psi$ for every $\psi \in \Sigma$:

$$\{(\theta(x_1), \ldots, \theta(x_k)) \mid \mathbf{DB}, \theta \models \varphi_2\} \subseteq \{(\theta(x_1), \ldots, \theta(x_k)) \mid \mathbf{DB}, \theta \models \varphi_1\}$$

- *Fundamental* in query compilation, e.g., query simplification.
- Equivalent to determining if the following is satisfiable:

$$\{(x_1, \ldots, x_k) \mid \varphi_2 \wedge \neg\varphi_1\}.$$

- Also equivalent to proving the following in FOL:

$$\left(\bigwedge_{\psi \in \Sigma} \psi\right) \to (\forall x_1, \ldots x_k . \varphi_2 \to \varphi_1).$$

- Again, undecidable in general, but decidable for fragments of RC.

What queries cannot be expressed in RC?

Recall that range restricted RC is not Turing complete
$\Rightarrow$ there are computable queries that cannot be expressed.

Built In Operations

- ordering, arithmetic, string operations, etc.

Counting and Aggregation

- Cardinality of Sets (*parity*)

Reachability, Connectivity, . . .

- *paths in a graph (binary relation)*

Data model extensions relating to incompleteness and inconsistency:

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

The Final Story

databases	$\Leftrightarrow$	*relational structures*
queries	$\Leftrightarrow$	*set comprehensions* *with conditions as formulas in FOL*†
integrity constraints	$\Leftrightarrow$	*sentences in FOL*
safety	$\Leftrightarrow$	*range restricted RC*‡ and *finite database instances*

† *first order predicate logic*

‡ *relational calculus*