Database Theory 236356 (Winter 2019/20)

The Logical Foundations of Relational Databases

FIRST MEETING: THURSDAY, 24.10.2019, 09:30

Course Home Page: `http://www.cs.technion.ac.il/ janos/COURSES/236356-19`

Lecturer: Prof. J.A. Makowsky (also Tirgul)

Language: Hebrew or English (if required), Reading assignments in English

Time: Thursday: 09:30-11:30 Lecture, 11:30-12:30 Tirgul

Place: Taub 4

Previously Given: Winter 94/95, Spring 1997, Winter 2000/01, Winter 2001/02, Winter 2010/11

Course prerequisites, I

We will use material from the following courses.

- 234129 Introduction to set theory and automata for CS

  The language of sets and inductive definitions

- 234292 Logic for CS

  Predicate Logic = Relational Calculus

  Quantifiers and their manipulation rules

  Writing and understanding formulas

Course prerequisites, II

- 236343 Computability Theory
  Turing machines, computable and computably enumerable sets, polynomial time computability, reductions, NP-completeness

- 236363 Database Management Systems
  The relational model and its basic query languages, dependencies, normal forms

The prerequisites are **NOT** enforced.

If required we will briefly repeat what is needed.

Sets and Logic (234293)

The material of Sets and Logic was recently redistributed into

- 234129 Introduction to set theory and automata for CS
- 234292 Logic for CS

The detailed lecture notes of the old course 234293 can be found at

https://www.cs.technion.ac.il/ janos/COURSES/LOGIC-2014/

I will refere to these slides when needed.

Literature, I

- P.C. Kannelakis,
  Elements of Relational Database Theory,
  in: Handbook of Theoretical Computer Science, volume 2, chapter 17.
  J. van Leeuwen, ed., Elsevier Science Publishers, 1990.

- P. Atzeni and V. de Antonellis,
  Relational Database Theory,
  The Benjamin/Cummings Publishing Company, 1993

- S. Abiteboul, R. Hull and V. Vianu,
  Foundations of Databases , Addison-Wesley 1995

- M. Levene and G. Loizou,
  A Guided Tour of Relational Databases and Beyond,
  Springer, 1999

- L. Libkin, Elements of Finite Model Theory, Springer, 2004

Literature, II

- H. Mannila and K.J. Raiha,
  The Design of Relational Databases,
  Addison-Wesley 1992

- B. Thalheim,
  Entity-Relationship Modeling. Foundations of Database Technology,
  Springer, 2000

- C.J. Date
  Database Design and Relational Theory:
  Normal Forms and All That Jazz ,
  O'Reilly Media, 2012

- Papers from the research literature to be posted on the course homepage

Good students read!

Books and classical research papers.

The material of this course was developed mostly between 1970 and 1990.

Slides, blackboard and lecture notes: Warnings

- I intend to prepare slides,
  but they only supplement the course.

- It is important that YOU make your own notes.

- For each lecture, I would like to have volunteer
  to take notes and discuss them with me.

Course requirements

Homework: I will indicate homework on the course slides.
Selected solutions are discussed in the Tirgul-sessions.

It is in your interest to do the homework.
It serves for your self-evaluation.

Final projects:

- Preparing handouts (slides) of parts of the lecture (up to 6 hours),
- Preparing handouts (slides) of additional material from articles related to the course,
- Projects are made in groups of maximal 3 students (preferably 2 or alone).
  Each member of the group must be able to
  - Explain all the material treated in the handout
  - Answer question covering all the material of the course

  This may be checked when the final project is handed in.

Final grades are given on the basis of:

Projects or a take home exam.

Outline of the course

- Expressive power of query languages
  Relational Algebra, Relational Calculus,
  Codd's Theorem,
  SQL, Datalog

- Computability and complexity of query languages
  Datacomplexity, expression complexity,
  polynomial time computable queries,
  polynomial space computable queries,
  fixed point logic

- Design theory
  Dependencies, Normal forms revisited

Lecture 1: Outline

Lecture:

- Relational calculus
- Predicate calculus
- Relational structures vs. databases
- Codd's Theorem

Tirgul:

- Tools from Logic and Computability:
  The Prenex Normal Form Theorem for FOL

Databases as structures

We are given:

Syntax:
- A finite set of attributes (attribute symbols) $\mathcal{A}$
- A set of $\mathbf{R}$ relation schemes $R_i(\bar{A}_i)$, $i = 1, \ldots, r_i$, where $\bar{A}_i(A_{i,1} \ldots, A_{i,r_i})$ is a sequence of attributes from $\mathcal{A}$.
- The set $\mathbf{R}$ is called a database scheme.

Semantics:
- For each $A \in \mathcal{A}$ a finite set $D(A)$, the domain of A.
- For each relation scheme $R(\bar{A}) = R(A_1, \ldots A_r)$ a subset (relation, table) $T_R \subseteq D(A_1) \times \ldots \times D(A_r)$
- A database is a collection of tables corresponding to the database scheme $\mathbf{R}$.

Relational Algebra, I

- The basic relation schemes are RA-terms.
- If T_1, T_2 are RA-terms over the same attribute sequence then $T_1 \cup T_2$ is an RA-term
- If T_1, T_2 are RA-terms over the same attribute sequence then $T_1 \setminus T_2$ is an RA-term.
- If T_1, T_2 are RA-terms then $T_1 \times T_2$ is an RA-term.
- If X is a set of attributes and T is an RA-term, then $\pi_X T$ is an RA-term.
- If T is an RA-term, then $\sigma_{x=y}T$, $\sigma_{x\neq y}T$, $\sigma_{x=a}T$, $\sigma_{x\neq a}T$ are RA-terms.

Relational Algebra, II

We use abbreviations:

For disjoint attribute sets X, Y, Z and an RA-term $\sigma_{X=Y}T(X, Y, Z)$ for

$$\sigma_{x_1=y_1}\left(\sigma_{x_2=y_2}\left(\ldots\sigma_{x_n=y_n}\left(T(X, Y, Z)\right)\right)\ldots\right)$$

For disjoint attribute sets X, Y, Z and RA-terms $T_1(X, Y)$ and $T_2(Y, Z)$ we write $T_1 \bowtie T_2$ for

$$\pi_{X,Y,Z}\sigma_{Y=Y'}\left(T_1(X, Y) \times T_2(Y', Z)\right)$$

Semantics of Relational Algebra

We define the meaning function for terms of Relational Algebra inductively:

- For $\cup, \cap, \setminus$ we use the corresponding set operations.
- For π we use projections.
- For $\times$ we use cartesian products.
- For σ we use the principle of selection.

Queries in RA

A query is a function which

- takes as input a databasis (set of tables)
- and gives as output a relation (a table)

For each RA expression T, the meaning function defines a query Q_T.

Computing Queries in RA

It seems clear (does it?):

Q_T is intuitively computable.

- How can we make this precise?
- Are all intuitively computable queries expressible using RA expressions?

Two fundamental papers by A. Chandra and D. Harel (1980 and 1982)

- **Ashok K. Chandra and David Harel**,
  Computable Queries for Relational Data Bases,
  Journal of Computer and System Sciences, vol. 21, (1980), pages 156-178.

- **Ashok K. Chandra and David Harel**,
  Structure and Complexity of Relational Queries,
  Journal of Computer and System Sciences, vol. 25, (1982), pages 99-128

What we have learned in Database Systems (236363)

- We have introduced other Query Languages: DRC, DATALOG, stratified DATALOG.
- There are intuitively computable queries not expressible in RA but expressible in DRC or DATALOG
- Every query expressible in RA is also expressible in DRC.
- Every query expressible in DRC is also expressible in stratified DATALOG.
- But are there intuitively computable queries not expressible in stratified DATALOG?

How can we prove this and answer similar questions?

Tools from Logic

In Predicate Logic (FOL), we also deal with infinite structures. We have

- Completeness Theorem for proof sequences.

  The consequence relation is semi-compuatble, but not computable.

- Compactness Theorem for satisfiability.

What happens when we restrict to finite structures?

Logic on finite structures

A database is always FINITE.

If we restrict ourselves to finite structures, the situation changes:

- The Completeness Theorem is NOT true!

  The consequence relation is NOT semi-compuatble, but satisfiability in finite structures is semi-computable.

- Compactness Theorem is NOT true anymore.

We have to use other tools.

Tools from Computability

- The basic data structures are strings or words.
- Computable functions are defined using Turing machines.
- Using Turing machines for other data structures requires coding of these data structures as strings.

We need a higher level notion of computability for

- databases
- natural numbers

We will introduce register machines.

FOL-formulas

Vocabularies are sets of relation symbols.

Atomic formulas are of the form $R(x_1, \ldots, x_n)$ (when R is n-ary) and $x_i = x_j$.

Formulas are built inductively using
parentheses (,), $\wedge, \vee, \rightarrow, \neg, \exists x_i, \forall x_i$.

All *occurences of variables* in atomic formulas are *free*.

If an occurrence of x_i in ϕ is free, then this particular occurence is *bound* in $\exists x_i \phi$ and $\forall x_i \phi$.

A *variable* x_i *is free* in ϕ if it has a free occurrence.

Domain Relational Calculus DRC

The **basic query** looks as follows:

$$\{(x_1, \ldots, x_n) \in D_1 \times \ldots \times D_n : \phi(x_1, \ldots, x_n)\}$$

where $\phi(x_1, \ldots, x_n)$ is a FOL-formula over the vocabulary appropriate for the relation scheme.

$$\{(x_1, \ldots, x_n, x_{n+1}) \in D_1 \times \ldots \times D_{n+1} : \phi(x_1, \ldots, x_n)\}$$

is an $(n+1)$-ary relation, even if ϕ has fewer free variables.

$$\{(x_1, \ldots, x_n) \in D_1 \times \ldots \times D_n : \phi(x_1, \ldots, x_n, y)\}$$

stands for

$$\{(x_1, \ldots, x_n) \in D_1 \times \ldots \times D_n : \exists y \phi(x_1, \ldots, x_n, y)\}$$

Equivalence of queries

Two query expressions Q_1 and Q_2 are equivalent

if in all databases they give the same answer.

$RA \subseteq DRC$

We prove by induction for a fixed database scheme:

Theorem: For every expression $T \in RA$ there exists a FOL-formula ϕ_T such that T and ϕ_T are equivalent.

Furthermore, ϕ_T can be computed in polynomial time (logarithmic space) in the size of T.

Proof of $RA \subseteq DRC$

Let ϕ_{T_1}, ϕ_{T_2} be the translations of T_1 and T_2.

(i) $R(X_1, \ldots, X_m)$ is translated as $R(x_1, \ldots, x_m)$

(ii) $T_1 \cup T_2$ is translated by $\phi_{T_1} \vee \phi_{T_2}$.

(iii) $T_1 \setminus T_2$ is translated by $\phi_{T_1} \wedge \neg\phi_{T_2}$.

(iv) $T_1 \times T_2(X_1, \ldots, X_m, X'_1, \ldots, X'_m)$
is translated by
$\phi_{T_1}(x_1, \ldots, x_m) \wedge \phi_{T_2}(x'_1, \ldots, x'_m)$.

(v) $\pi_X T_1(X, Y)$ is translated by $\exists y_1, \ldots, y_m \phi_{T_1}$.

(vi) $\sigma_\theta T_1(X)$ is translated by $(\phi_{T_1} \wedge \theta)$.

$DRC \subseteq_? RA$

We use only $\wedge, \neg, \exists$.

Let T_1, T_2 be the translations of ϕ_1 and ϕ_2.

(i) $R(x_1, \ldots, x_m)$ is translated as $R(X_1, \ldots, X_m)$

(ii) $\exists y_1, \ldots, y_m \phi_1(x, y)$. is translated by $\pi_X T_1(X, Y)$.

(iii) $\phi_1 \wedge \phi_2$ is translated by $T_1 \bowtie T_2$.

(iv) $\neg\phi_1(x_1, \ldots, x_m)$ is translated by $(D_1 \times \ldots \times D_m) \setminus T_1(X_1, \ldots, X_m)$.

But this does not work when we don't have terms in RA for the domains.

$DRC = RA^+$

We define RA^+ by augmenting RA with terms for the domain. We have shown:

Theorem: $RA^+ = DRC$

We shall see that RA is strictly weaker than DRC.

One way to see this is:

- Even if the domains are infinite, but all the basic relations are finite, every RA query represents a finite table.
- If the domain of one variable is infinite, and the basic relations are finite, the result of the query given by $\neg R(x_1, \ldots, x_m)$ is infinite.

Domain independence, I

Let $R(X_1, \ldots, X_m)$ a relation scheme and $D_1, \ldots, D_m$ the domains of the attributes.

In a relation instance $r \subseteq D_1 \times \ldots \times D_M$ the *actual domain* $AD_i(r)$ is defined by the query $\pi_{X_i} R$ evaluated over r.

Definition:
A query expression T (ϕ) is *domain invariant* if for every relation instance r and for every choice of domains U_i with

$$AD_i(r) \subseteq U_i \subseteq D_i$$

the evaluation of T (ϕ) over the instance $\langle U_1, \ldots, U_m, r \rangle$ gives the same result.

Domain independence, II

Theorem:

RA-expressions are domain independent.

Observation:

Not domain invariant:
$\neg R$, $(R(x_1, x_2) \vee R(x_1', x_2'))$, $\forall x_1 R(x_1, x_2)$.

Domain invariant:
$\forall x_1(\exists x_2 R(x_1, x_2) \rightarrow R(x_1, x_3))$

Homework:

Prove the theorem by induction.
Prove the observation.

SafeRC, I

We are looking for a subset $SafeRC \subseteq DRC$ wich captures the notion of domain invariance.

Formulas in $SafeRC$ should be

(i) domain invariant;

(ii) recognizable (computable) efficiently;

(iii) complete for domain invariance, i.e. every domain invariant formula of DRC is equivalent to some expression in $SafeRC$.

(iv) effectively complete for domain invariance, i.e. every domain invariant formula of DRC is equivalent to some effectively computable expression in $SafeRC$.

Applying Rice' Theorem

Theorem:(From the computability course)
Any nontrivial property about the language recognized by a Turing machine is undecidable.

Let $DI \subseteq DRC$ the set of DRC-formulas which are domain invariant.

Conclusion: DI is not computable (effectively recognizable).

How can we achieve (iv) ?

Codd's Theorem (1972)

Theorem: $RA = DI = SafeRC$.

(i) For every expression $Q_1 \in RA$ there is an expression $Q_2 \in DI$ which is equivalent to it.

(ii) For every expression $Q_1 \in DI$ there is an expression $Q_2 \in RA$ which is equivalent to it.

(iii) For every expression $Q_1 \in DI$ there is an expression $Q_2 \in SafeRC$ which is equivalent to it.

(iv) For every expression $Q_1 \in SafeRC$ there is an expression $Q_2 \in RA$ which is equivalent to it.

(v) It is the equivalence which is affected by Rice's Theorem!

Proof of Codd's Theorem

- This uses ideas going back to Löwenheim (1915) early work on Model Theory by A. Tarski.

- The proof will be given in the next lecture.

  See also Chapter 5 in the book by Abiteboul, Hull and Vianu.