# CS294-248 Special Topics in Database Theory
# Unit 9: Review

Dan Suciu

University of Washington

# Announcement

- Office hours in the afternoon (only if you need): at Simons, Room 338
- Presentations: Thursday, 11/30, at 9:30am sharp, Simons, Room 116.

# What We Covered

- Logic and Queries
- Conjunctive Queries
- Incremental View Maintenance (Dan Olteanu)
- The AGM bound and Information Inequalities
- Worst Case Optimal Algorithmis (Hung Ngo)
- Database Constraints, Repairs
- Incomplete Databases
- Semirings, K-relations
- Tree Decomposition, FAQ (Hung Ngo)
- Datalog

# Logic and Queries

# Logic and Model Theory

Logic: $\wedge, \vee, \neg, \forall, \exists$ what are formulas? Sentences?

Models: $\boldsymbol{D} \models \varphi$

Validity, Satisfiability what are they?

$\exists x \exists y \exists z (\forall u (u = x) \vee (u = y) \vee (u = z))$

# Relational Databases

[Codd, 1970]

"A database is a finite structure, a query is an FO formula"

Relational Algebra $\sigma, \Pi, \bowtie, \cup, -$.

FO = RA why is this significant?

what is domain independence?

# Query Evaluation v.s. Static Analysis

Query Evaluation:

Compute $Q(\boldsymbol{D})$

[Vardi, 1982]:

- Data complexity
- Query complexity
- Combined Complexity

what are they?

Data complexity of FO?

Static analysis:

Decide something about $Q$

Trakhtenbrot's theorem:
finite satisfiability is undecidable

Consequence: basically any static analysis for FO is undecidable, e.g. $Q_1 = Q_2$; or $Q_1 \subseteq Q_2$, etc.

# Conjunctive Queries

# CQs, Databases Hypergraphs

$$Q = R(X, Y) \wedge S(Y, Z) \wedge R(Z, U) \wedge T(U, X)$$

what is the hypergraph?

canonical database?

# Query Containment

$Q_1 \subseteq Q_2$

The homomorphism criterion.

The canonical database.

$$R(X,Y) \wedge R(Y,Z) \qquad \subseteq \qquad R(X,Y) \wedge R(Z,Y) \wedge R(Z,V) \wedge R(V,W)$$

What is the complexity of checking $Q_1 \subseteq Q_2$?

Can check $Q_1 \equiv Q_2$.

# Acyclic Queries

What is it?

Discuss Yannakakis' algorithm.

# Incremental View Maintenance (Dan Olteanu)

# Incremetnal Update

$$Q(\boldsymbol{D} + \delta\boldsymbol{D}) = Q(\boldsymbol{D}) + \delta Q(\boldsymbol{D}, \delta\boldsymbol{D})$$

- If we are in a semiring, e.g. $\mathbb{B}$, then $\delta$ means only insertion.
- If we are in a ring, e.g. $\mathbb{Z}$, then $\delta$ may be insertion/deletion

Example: $Q(X, Y, Z) = R(X, Y) \bowtie S(Y, Z)$ what is $\delta Q$?

# Incremetnal Update

$$Q(\boldsymbol{D} + \delta \boldsymbol{D}) = Q(\boldsymbol{D}) + \delta Q(\boldsymbol{D}, \delta \boldsymbol{D})$$

- If we are in a semiring, e.g. $\mathbb{B}$, then $\delta$ means only insertion.
- If we are in a ring, e.g. $\mathbb{Z}$, then $\delta$ may be insertion/deletion

Example: $Q(X, Y, Z) = R(X, Y) \bowtie S(Y, Z)$ what is $\delta Q$?

$$\begin{aligned}\delta Q(X, Y, Z) =& \delta R(X, Y) \bowtie S(Y, Z) \\ &+ R(X, Y) \bowtie \delta S(Y, Z) \\ &+ \delta R(X, Y) \bowtie \delta S(Y, Z)\end{aligned}$$

# Incremental View Maintenance

If $Q$ is a complicated query, then we may want to keep several intermediate results to facilitate incremental computation.

# The AGM bound and Information Inequalities

# Edge Covers

$Q = A \wedge B \wedge C \wedge D \wedge \cdots$

If $A, C, F$ form an edge cover, then $|Q| \leq |A| \cdot |C| \cdot |F|$. why?

If $a, b, c, d, \ldots$ form a fractional edge cover then $|Q| \leq |A|^a \cdot |B|^b \cdots$

# Vertex Packing

We construct the worst-case database instance by using a fractional vertex packing.

$L_5$: $\boxed{A_1(x_1, x_2) \wedge A_2(x_2, x_3) \wedge A_3(x_3, x_4) \wedge A_4(x_4, x_5)}$

# Vertex Packing

We construct the worst-case database instance by using a fractional vertex packing.

$L_5$: $\boxed{A_1(x_1, x_2) \wedge A_2(x_2, x_3) \wedge A_3(x_3, x_4) \wedge A_4(x_4, x_5)}$
$\boldsymbol{w}^* = (1, 1, 0, 1)$, $\boldsymbol{v}^* = (1, 0, 1, 0, 1)$.
$AGM = N^3, \quad A_1, \ldots, A_4 = [N] \times [1], \quad [1] \times [N], \quad [N] \times [1], \quad [1] \times [N]$

# Vertex Packing

We construct the worst-case database instance by using a fractional vertex packing.

$L_5$: $\boxed{A_1(x_1, x_2) \wedge A_2(x_2, x_3) \wedge A_3(x_3, x_4) \wedge A_4(x_4, x_5)}$

$\boldsymbol{w}^* = (1, 1, 0, 1)$, $\boldsymbol{v}^* = (1, 0, 1, 0, 1)$.

$AGM = N^3, \quad A_1, \ldots, A_4 = [N] \times [1], \quad [1] \times [N], \quad [N] \times [1], \quad [1] \times [N]$

$C_5$: $\boxed{A_{12}(x_1, x_2) \wedge A_{23}(x_2, x_3) \wedge A_{34}(x_3, x_4) \wedge A_{45}(x_4, x_5) \wedge A_{51}(x_5, x_1)}$

# Vertex Packing

We construct the worst-case database instance by using a fractional vertex packing.

$L_5$: $\boxed{A_1(x_1, x_2) \wedge A_2(x_2, x_3) \wedge A_3(x_3, x_4) \wedge A_4(x_4, x_5)}$
$\boldsymbol{w}^* = (1, 1, 0, 1)$, $\boldsymbol{v}^* = (1, 0, 1, 0, 1)$.
$AGM = N^3, \quad A_1, \ldots, A_4 = [N] \times [1], \quad [1] \times [N], \quad [N] \times [1], \quad [1] \times [N]$

$C_5$: $\boxed{A_{12}(x_1, x_2) \wedge A_{23}(x_2, x_3) \wedge A_{34}(x_3, x_4) \wedge A_{45}(x_4, x_5) \wedge A_{51}(x_5, x_1)}$
$\boldsymbol{w}^* = (1/2, \ldots, 1/2)$, $\boldsymbol{v}^* = (1/2, \ldots, 1/2)$.
$AGM = N^{5/2}$; $A_{12} = A_{23} = \cdots = [N^{1/2}] \times [N^{1/2}]$

# Shannon Inequalities

Monotonicity: $h(\boldsymbol{UV}) \geq h(\boldsymbol{U})$
Submodularity $h(\boldsymbol{UV}) + h(\boldsymbol{UW}) \geq h(\boldsymbol{U}) + h(\boldsymbol{UVW})$

The triangle inequality:
$h(XY) + h(XZ) + h(YZ) \geq h(XYZ) + h(X) + h(YZ) \geq$
$h(XYZ) + h(XYZ) + h(\emptyset) = 2h(XYZ)$

Implies $|Q| \leq (|R| \cdot |S| \cdot |T|)^{1/2}$ where
$Q(X, Y, Z) = R(X, Y) \wedge S(Y, Z) \wedge T(X, Z)$.

# Worst Case Optimal Algorithmis (Hung Ngo)

# Queries and their Evaluation Strategies

A progression:

- SQL:
  `select...from...where...`
- Naive evaluation:
  for $t_1 \in R_1$ for $t_2 \in R_2 \ldots$.
- All database systems:
  $((R_3 \bowtie R_7) \bowtie R_2) \ldots$
- Worst-Case-Optimal-Join:
  for $x \in R_1.X \cap R_3.X \cap \cdots$ for $y \in \ldots$
  Runtime: $O(AGM)$.

# Database Constraints, Repairs

# Type of Constraints

- Functional Dependencies $\boldsymbol{U} \to \boldsymbol{V}$.
- Multivalued Dependencies: $\boldsymbol{U} \twoheadrightarrow (\boldsymbol{V}|\boldsymbol{W})$ means $R(\boldsymbol{UVW}) = R(\boldsymbol{UV}) \bowtie R(\boldsymbol{UW})$
- Inclusion dependencies ...
- Generalized dependencies: TGDs, EGDs.
- Chase/backchase: "apply" the GDs repeatedly forwards, then backwards. Optimize queries to use indices, materialized views, etc.

# Incomplete Databases

# Incomplete Databases

Pure theoretical notion: $\mathcal{I} = \{\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots\}$.

Representations:

- Codd tables: they have NULLs $\perp, \perp, \perp, \ldots$
- v-tables or "naive tables" have marked NULLs $\perp_1, \perp_2, \ldots$.
- c-tables have arbitrary Boolean formulas or expressions.

# Semirings, K-relations

# Useful Semirings

Booleans $\mathbb{B} = (\{0,1\}, \vee, \wedge, 0, 1)$: set semantics

Natural numbers $\mathbb{N}$: bag semantics

Tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$: shortest path.

The access control semiring: $\boxed{(\mathbb{A}, \min, \max, 0, P)}$
$\mathbb{A} = \{\text{Public} < \text{Confidential} < \text{Secret} < \text{Top-secret} < 0\}$ 0 "No Such Thing"

# Tree Decomposition, FAQ (Hung Ngo)

# Various Notions of Tree-Width

Tree decomposition: place multiple atoms (hyperedges) in a tree node (bag), but ensure the running intersection property holds.

How do we measure the "width"?

- Tree-width Number of variables minus 1 in each bag.
- (Generalized) Hypertree Width: number of atoms in each bag.
- Fractional hypertree width: the AGM bound of each bag.

# Datalog

# Recursion!

Things to know:

- Least fixpoint, and minimal model are the same.
- Some cool datalog programs: transitive closure (linear and non-linear), regular expressions, same generation, AGAP.
- Naive/semi-naive algorithm.
- Once we add negation, it gets a lot more complicated.

# Final Thoughts

- Database theory: it informs and explains, but does not necessarily prescribe.

- To understand or build a database system you still need to learn/know lots of systems-level aspects: but theory will help you understand better the context of what you are doing.

The End

Presentations: Thursday, 9:30am sharp, Simons Institute, Romm 116

Codd, E. F. (1970).
A relational model of data for large shared data banks.
*Commun. ACM*, 13(6):377–387.

Vardi, M. Y. (1982).
The complexity of relational query languages (extended abstract).
In Lewis, H. R., Simons, B. B., Burkhard, W. A., and Landweber, L. H., editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 137–146. ACM.