CompSci 590.6

Understanding Data:

Theory and Applications

Lecture 13

Incomplete Databases

Instructor: Sudeepa Roy

Email: *sudeepa@cs.duke.edu*

Today's Reading

- "Alice Book": Foundations of Databases

Abiteboul-Hull-Vianu

Chapter 19

- Incomplete Information in Relational Databases

Imielinski-Lipski

JACM 1984

Incomplete Databases : Missing Information

Several scenarios:

- Information exists, but we do not have it
  - "John bought a car, but I don't know which one"
- Fuzzy values
  - "Heather lives in a large and cheap apartment"
- Irrelevant attributes for some tuples
  - Spouse's information if one is single
- Unreliable information

Semantic vs. Representation

- Semantic of incomplete databases
  - A set of possible worlds or instances (again)
- Representation of incomplete databases
  - What is actually stored

- Same problem as probabilistic databases, but now there is no probability

How to store missing values?

- Use NULL values
  - @ in Imielinski-Lipski paper
- Use variables
  - Different assignments = different possible worlds

- How do we choose?
- Choice depends on
  - our requirements of the representation system
  - the query language we want to support (SP, PJ, SPJ, SPJU etc)

Desired properties of a Representation System

- Intuitively, the representation system should be

1. Sound/Safe:
   - No incorrect conclusion is derived for the specified language
2. Complete
   - All valid conclusions are derivable

Representation Systems Overview (IL'84)

SUPPLIER	LOCATION	PRODUCT	QUANTITY
Smith	London	Nails	@
Brown	@	Bolts	@
Jones	@	Nuts	40000

- **Codd-table**
- @ = NULL
- Supports SP
- Cannot support both PJ

COURSE	TEACHER	WEEKDAY
Databases	x	Nails
Programming	y	Tuesday
Databases	x	Thursday
FORTRAN	Smith	z

- **V-table**
  - **Naïve table (Alice book)**
- Can use same x twice
- Supports positive RA

SUPPLIER	LOCATION	PRODUCT	con
x	London	Nails	x=Smith
Brown	New York	Nails	x ≠ Smith

- **Conditional-table or c-table**
- Nail is supplied by either Smith or Brown but not both

Switching to notations/terminology in the Alice book

Codd-table or Table

- Constants: “Alice”, “NY”, “50”
- Variables: x, y, z
- Attributes U
- A Table/Codd-table is a relation over U
  - A finite set of tuples over U
  - Each tuple contains constants and/or variables
  - No variable appears twice
- Extends to multiple tables

A	B	C
0	1	x
y	z	1
2	0	v

Codd-table

Valuation of a Codd-table

- Given a set of variables Var, a valuation v is a total function from Var to constants Consts
  - v: Var -> Const
- Semantic of a Codd-table T

  ≡ all possible valuations of variables in T

  ≡ rep(R) = {v(T): v is a valuation of variables in T}

A	B	C
0	1	x
y	z	1
2	0	v

Codd-table

A	B	C
0	1	2
2	0	1
2	0	0

A	B	C
0	1	2
3	0	1
2	0	5

Some Instances

A	B	C
0	1	1
2	0	1

Why two rows?

Closed vs. Open World Assumptions

A	B	C
0	1	x
y	z	1
2	0	v

c-table

A	B	C
0	1	2
2	0	1
2	0	0

A	B	C
0	1	2
3	0	1
2	0	5

Some Instances

A	B	C
0	1	1
2	0	1

Why two rows?

- Closed World Assumption (CWA)
  - Each tuple in any instance must be justified by the presence of a free tuple in the c-table
- Open World Assumption (OWA)
  - All instances "contain" an instance from the valuations rep(T)

So far…

- Semantic of incomplete databases
  - Possible worlds (CWA or OWA)
- Representation
  - Codd-tables

Next

- Query evaluation

Query Evaluation: Semantic

- Codd-table T
  - Represents possible worlds rep(T) – by CWA
- Query q
- For each instance I $\in$ rep(T), the answer is q(I)
- The set of possible answers q(rep(T))
- This gives the semantic.
- But what are the requirements of its representation?

Query Evaluation: Representation

- Codd-table T
  - A "system" to represent input incomplete database
- Query q from a language L (e.g. RA)
  - For each I $\in$ rep(T), the answer is q(I)
  - The set of possible answers q(rep(T))
- We would like to represent q(rep(T)) in the same system
- For each representation T and query q, there should exist a computable representation $\underline{q}(T)$ such that
  - rep($\underline{q}$(T)) = q(rep(T))

- Strong representation system:
  - The above property holds for a query language L (here RA)
  - i.e. the answer to a query represents ALL POSSIBLE answers
- Weak representation system:
  - Weaker requirement

Query $q = \sigma_{A=3}(T)$

Challenges: Query evaluation on Codd-table

A	B	C
0	1	x
y	z	1
2	0	v

Codd-table T

A	B	C
0	1	2
2	0	1
2	0	0

I_1

A	B	C
0	1	2
3	0	1
2	0	5

I_2

A	B	C
0	1	1
2	0	1

I_3

- No Codd-table representing the possible answers to q
  - I_1, I_3 : empty relation
  - I_2 : non-empty relation (single tuple)
- Suppose a table T' exists such that T' represents q(T)
- Either T' is empty
  - Then $q(I_2)$ is not in rep(T')
- Or T' is non-empty
  - Then empty relation is not in rep(T')
- Contradiction. T' does not exist

So far…

- The representation system of Codd-table is insufficient

- Alternatives

1. Be less demanding
   - weak representation system
   - Next
2. Consider richer representation system that lead to a complete representation system for all of RA
   - c-table
   - Later

Weak Representation Systems

- DOES NOT require that the answer to a query be a representation of the set of all possible answers

- ASKS which tuples are SURELY in the answer
  - i.e. belongs to all possible answers

Sure Tuples

- The set of "sure tuples"
  - sure(q, T) = ∩ {q(I) | I ∈ rep(T)}
  - t ∈ sure(q, T) ⇔ t is in q(I) for each possible world I
- Can be computed by dropping all variables
  - But there is a problem

$q = \sigma_{A=2}(T)$ $q' = \Pi_{AB}(R)$

Problem with dropping vars

A	B	C
0	1	x
y	z	1
2	0	v

Codd-table T

A	B	C
0	1	2
2	0	1
2	0	0

I_1

A	B	C
0	1	2
3	0	1
2	0	5

I_2

A	B	C
0	1	1
2	0	1

I_3

- sure(q, T) = ϕ
- q'(sure(q, T)) = ϕ

- <2, 0, *> $\in$ I for all I $\in$ q(rep(T))
- <2, 0> $\in$ I' for all I' $\in$ q'(q(rep(T)))
- sure(q'(q(rep(T))) = {<2, 0>} (why)

- Not compositional
- Need a notion for equivalence

Equivalence of Incomplete Databases

- L is a query language
- Two incomplete databases **I**, **J** are L-equivalent if they are indistinguishable w.r.t. sure tuples for all q in L
  - $\mathbf{I} \equiv_L \mathbf{J}$
  - $\cap \{q(I) \mid I \in \mathbf{I}\} = \cap \{q(J) \mid J \in \mathbf{J}\}$
  - For each $q \in L$

Weak Representation System (WRS)

- WRS for a query language L
  - For every q in L
  - For each representation T of an incomplete database
    - there can be several representation for the same db
  - There is a representation $\underline{q}(T)$ such that
  - $rep(\underline{q}(T)) \equiv_L q(rep(T))$
    - i.e. their "sure tuples" are identical

Weak Representation System (WRS)

- $rep(\underline{q}(T)) \equiv_L q(rep(T))$

- NOTE:
  - $\underline{q}(T)$ is not precisely sure(q, T)
  - But sure(q, T) can be obtained by eliminating tuples with vars at the end

- Codd-tables form a WRS
  - for Select-Project
  - NOT for Union-Join (proof in the Alice book)

WRS examples: SP

- Select σ:
  - return tuples s.t. the condition holds for all valuations of vars
- Project Π:
  - ordinary projection

$q = \sigma_{A=2}(T)$ $q' = \Pi_{AB}(R)$

A	B	C
0	1	x
y	z	1
2	0	v

$\underline{\sigma_{A=2}}(T) = \{<2, 0, v>\}$

$\underline{\Pi_{AB}}\ \underline{\sigma_{A=2}}(T) = \{<2, 0>\}$

The problem with Join/Union:

repeated vars are not allowed in Codd-tables

Next: Naïve Tables

- Naïve Tables
  - called V-tables in Imilienski-Lipski'84
  - allow repeated vars
- Work for SPJU = positive RA
- Basically vars are treated as distinct constants
- However
  - the representation system is still weak
- Next c-tables

A	B	C
0	1	x
x	z	1
2	0	v

Extend representation with conditions on vars

- Limitations of Codd-tables and naïve tables
  - for selection conditions, sometimes they hold sometimes do not
- Solution:
  - Extend representation with conditions on vars
- c-tables
  - Form a strong representation system for RA

Condition

- Conjunctions of
  - equality atoms
    - $x = y$
    - $x = c$ (const)
  - inequality atoms
    - $x \neq y$
    - $x \neq c$ (const)
- Two types of conditions associated with a table T
- Global:
  - ϕ_T associated with the entire table
- Local:
  - φ_t associated with a tuple t in T

conditional-table or c-table

- A triple (T, ϕ_T, φ)
  - T is a naive-table
  - ϕ_T is a global condition
  - φ is a mapping that associates a local condition φ_t to every tuple t of T
- Omit a condition
  = true (always holds) or x = x
- ϕ, φ can contain vars that DO NOT belong to table T or tuple t

A	B	
x ≠ 2, y ≠ 2		
0	1	z = z
1	x	y = 0
y	x	x ≠ y

means TRUE

c-table T'

Some instances

A	B
0	1
1	3
0	3

J_1

A	B
0	1
1	0

J_2

A	B
0	1

J_3

conditional-table or c-table

- A c-table (T, ϕ_T, φ) represents possible worlds as follows
  - rep(T) = {I | there is a valuation v satisfying ϕ_T such that the relation I consists exactly of those facts v(t) for which v satisfies φ_t}
- Sample assignments of (x, y, z)
  - J_1: (3, 0, 0)
  - J_2: (0, 1, 0)
  - J_3: (1, 1, 0)
  - ……

A	B	
x ≠ 2, y ≠ 2		
0	1	z = z
1	x	y = 0
y	x	x ≠ y

c-table T'

Some instances

A	B
0	1
1	3
0	3

J_1

A	B
0	1
1	0

J_2

A	B
0	1

J_3

Representation Power

- Can capture disjunction
- Example
  - Sally is taking MATH or CS (but NOT both) + another course
  - Alice takes BIOLOGY if Sally takes MATH
  - Alice takes MATH or PHYSICS (but not both) if Sally takes PHYSICS

Student	Course	
(x ≠ MATH) ^ (x ≠ CS)		
Sally	MATH	z = 0
Sally	CS	z ≠ 0
Sally	x	
Alice	BIOLOGY	z = 0
Alice	MATH	(x = PHYSICS) ^ (t = 0)
Alice	PHYSICS	(x = PHYSICS) ^ (t ≠ 0)

Observations

- There may be several c-tables for the same incomplete database
- Equivalence:
  - Two representations T, T’ are equivalence if rep(T) = rep(T’)
- Even checking membership in rep(T) is NP-complete
- Therefore, testing equivalence is not easy
  - but is decidable

c-tables form "Strong Representation Systems"

- Projection
  - simply project the columns
  - but preserve conditions
- Selection
  - add new conjuncts to local condition
- Union
  - make sure the tables use distinct vars
  - then choose appropriate local conditions
- Join
  - consider all possible pairs from two tables
- Set difference
  - not limited to positive RA

Example

T_1

B	C	
x	c	

T_2

B	C	
y	c	y = b
z	w	

T_3

A	B	
a	y	

Compute the c-tables for (on whiteboard)

- $\Pi_B (T_2)$
- $T_1 \bowtie T_3$ --- natural join
- $\sigma_{B=b} (T_1 \bowtie T_3)$
- $T_1 \cup T_2$
- $T_1 - T_2$