CompSci 590.6

Understanding Data:

Theory and Applications

Lecture 14

Inconsistent Databases

Instructor: Sudeepa Roy

Email: *sudeepa@cs.duke.edu*

Today's Reading

Summary/overview papers (available online):

1. Chomicki: Consistent Query Answering: Five Easy Pieces. ICDT, 2007
   – Acknowledgement: Slides available online of this invited talk
2. Bertossi: Consistent Query Answering in Databases. ACM SIGMOD Record, 2006
   – See this paper for references
3. Bertossi-Chomicki: Query Answering in Inconsistent Databases. In Logics for Emerging Applications of Databases, Springer-Verlag, 2003

Also see

Arenas-Bertossi-Chomicki

Consistent Query Answers in Inconsistent Databases

PODS'99

Why should we care about consistent query answering?

Application 1: Data Warehousing

- Data from several sources, may violate Integrity Constraints (IC)
- Standard: clean data before storing
    1. Need to determine which data is already clean so that unclean data can be removed
    2. Or, mark query answers as consistent or inconsistent
        - information loss can be avoided

Application 2: Data Integration

- Many databases are integrated together to provide a unified view to the user
- Databases may have different ICs
  - Can be satisfied locally but violated globally
  - e.g. different addresses for the same individual in the tax-payer database and voter-registration database
- May not be possible to clean the databases
  - each database is autonomous and independent
- Need to find out which query answers are consistent and which are not

Application 3: Unenforced IC

- Database systems allow IC
- But sometimes they cannot be enforced
  - May be a "legacy data source" that does not support IC
  - IC checking may be too costly
  - DBMS may support only a few IC and not the complex ones

Application 4: Active and Reactive Databases

- Active databases may violate ICs temporarily
  - e.g. inventory falls below a minimum level in a warehouse
  - it is ok if the replenishments have been ordered
  - but query answers should give an indication of the inconsistency with the ICs

Database D

Example - 1

Emp

Ename	Dept
Alice	Sales
Bob	Sales
Tom	HR

Manager

Ename	Mname
Sales	Harry
HR	Tom

Integrity Constraints IC

$Manager[mname] \subseteq Emp[Ename]$

Here Inclusion Dependency

First order structure: a set of tuples/facts/ground atoms

- A database instance D is consistent if D satisfies IC
  - $D \models IC$
  - Otherwise, D is inconsistent

Here $D \not\models IC$

Inconsistent Database

Example - 2

Database D

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat
Bob	Ph.D.	CS

Integrity Constraints IC

Student $\rightarrow$ Degree, Dept

Here Functional Dependency

First order structure: a set of tuples/facts/ground atoms

- A database instance D is consistent if D satisfies IC
  - $D \models IC$
  - Otherwise, D is inconsistent

Here $D \not\models IC$

Inconsistent Database

Problems with inconsistent database

Database D

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat
Bob	Ph.D.	CS

IC

Student → Degree, Dept

SELECT Student
From D
Where Dept = 'CS'

Student
Alice
Bob

Results not reliable

Solution: Database Repair

- “clean” an inconsistent database that satisfies ICs
- simulate manual cleaning automatically

Repairs: Example

Database D

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat
Bob	Ph.D.	CS

IC

Student → Degree, Dept

D1, D2 ⊨ IC

Repairs for D

Database D1

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat

Database D2

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	CS

Are these two only possible repairs?

Several intuitive ways to repair a database

Repairs: Definition

- Distance between two database instances
  - $\Delta(D, D') = (D - D') \cup (D' - D)$
- A repair D' of D is another instance over the same schema
  - that satisfies IC
  - and makes $\Delta(D, D')$ <u>minimal under set inclusion</u>

Repairs: Example

Database D

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat
Bob	Ph.D.	CS

IC

Student → Degree, Dept

D1, D2 ⊨ IC

Repairs for D

Database D1

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat

Database D2

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	CS

Are these two only possible repairs?

Yes (only these two are minimal)

The set of repairs is denoted by Rep(D, IC)

Consistent Query Answering

- How do we compare different minimal repairs
  - we do not need to
  - we can to figure out what answers will always appear irrespective of the repair

= Consistent query answering on inconsistent databases

Consistent Query Answering (CQA)

Database D

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat
Bob	Ph.D.	CS

IC

Student → Degree, Dept

CQA:
The answer should belong to Q(D') for ALL repairs D'

Database D1

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat

Database D2

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	CS

SELECT Student
From D
Where Dept = 'CS'

Student
~~Alice~~
~~Bob~~

Student
Alice

Recall sure tuples in incomplete db

CQA: Definition

- Given a query Q(x1, …, xn)
- A tuple **t** = (t1, …, tn) is a consistent answer
  - to Q in D w.r.t. IC
- If for every D' $\in$ Rep(D, IC)
- D' $\vDash$ Q(**t**)
  - i.e. Q becomes true in D when x1,… xn takes values t1,…,tn

- If n = 0, i.e. Q is Boolean, then "yes" is a consistent answer if D' $\vDash$ Q for all D' $\in$ Rep(D, IC)
  - Otherwise, "no" is the consistent answer

The Impact of Inconsistency

- Traditional view toward IC
  - Ignore ICs for query answering
  - Use ICs for query optimization
    - Semantic Query Optimization
- Newer approach
  - Inconsistency = Uncertainty
  - Throwing away inconsistent tuples may not be a good idea (difficult or undesirable)
  - query results may or may not depend on ICs
  - Eliminate or tolerate inconsistency

Chomicki ICDT'2007

How to find CQA?

- Naïve solution
  - Find all possible repairs Q
  - evaluate query Q[D]
  - take intersection
- There can be exponentially many repairs (why)

A	B
a1	b1
a1	c1
a2	b2
a2	c2
…	…
an	bn
an	cn

IC:
Functional dependency
A -> B

How to find CQA?

- Better solution

- Rewrite Q into a new query Q′
  - the usual answers to Q′ in D = the consistent answers to Q from D
- Compute query Q′ iteratively by appending to Q additional conditions called "residues"
  - obtained from Q and ICs

How to find CQA?

- Example of query rewriting

SELECT A from R

- R(A, B)
- $Q(x) \text{ :- } R(x, y)$
- $Q'(x) \text{ :- } R(x, y) \wedge \forall v\, [\neg R(x, v) \vee (y=v)]$
- $A \rightarrow B$

  $\Leftrightarrow R(x, y) \wedge R(x, v) \Rightarrow y = v$

- Recall that

  $m \Rightarrow n \quad \Leftrightarrow \quad \neg m \vee n$

R

A	B
a1	b1
a1	c1
a2	b2
a2	c2
…	…
an	bn
an	cn

IC:
Functional dependency
A -> B

Query Rewriting

Database D

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat
Bob	Ph.D.	CS

IC

Student → Degree, Dept

- Query Q(x, y, z) :- D(x, y, z)
- IC: $\forall$ x,y,z,y',z' Q(x, y, z) $\wedge$ Q(x, y', z') → z = z'
  = $\forall$ x,y,z,y',z' ¬Q(x, y, z) $\vee$ ¬Q(x, y', z') $\vee$ z = z'
- Rewritten query
  - Q(x, y, z) $\wedge$ $\forall$ y', z' ¬Q(x, y', z') $\vee$ z = z'

What else?

- Different classes/notions of
  - queries
  - integrity constraints
  - repairs
  - minimality
  - actions

Types of ICs

IC Type	General Form	Example
Universal Constraints	$\forall$ …… [¬A1 $\vee$ … $\vee$ ¬An $\vee$ B1 $\vee$ … $\vee$ Bn]	$\forall$x [Par(x) $\Rightarrow$ M(x) $\vee$ F(x)] = $\forall$x [¬Par(x) $\vee$ M(x) $\vee$ F(x)]
Denial Constraints	$\forall$ …… [¬A1 $\vee$ … $\vee$ ¬An]	$\forall$x [¬CS(x) $\vee$ ¬MATH(x)] = $\forall$x ¬[CS(x) $\wedge$ MATH(x)]
Functional Dependencies	X $\rightarrow$ Y A key dependency in F if X is a key	SSN $\rightarrow$ Name
Inclusion Dependencies	R[X] $\subseteq$ S[Y] A foreign key constraint if Y is a key of S	Manager[mname] $\subseteq$ Employee[ename]

Complexity of CQA

- If the query rewriting technique works (First order queries), the complexity of CQA is poly-time
- But it may be hard
  – e.g. scalar aggregate queries

Aggregate Queries

IC: Employee -> Salary

Employee	Salaray	year_of_ service
Alice	80k	10
Bob	100k	13
Bob	90k	13

Repairs

Employee	Salaray	year_of_ service
Alice	80k	10
Bob	100k	13

Employee	Salaray	year_of_ sevice
Alice	80k	10
Bob	90k	13

Query:

always 13

SELECT Max(year_of_service)
FROM E

SELECT SUM(Salary)
FROM E

180k or 170k

Aggregate Queries

IC: Employee → Salary

Employee	Salaray	year_of_service
Alice	80k	10
Bob	100k	13
Bob	90k	13

Repairs

Employee	Salaray	year_of_service
Alice	80k	10
Bob	100k	13

Employee	Salaray	year_of_sevice
Alice	80k	10
Bob	90k	13

Query:

SELECT Max(year_of_service)
FROM E

always 13

SELECT SUM(Salary)
FROM E

180k or 170k

- Range Semantics
  - Find an optimal interval (L, U) such that the answer in ALL repairs is always
  - >= L
  - <= U

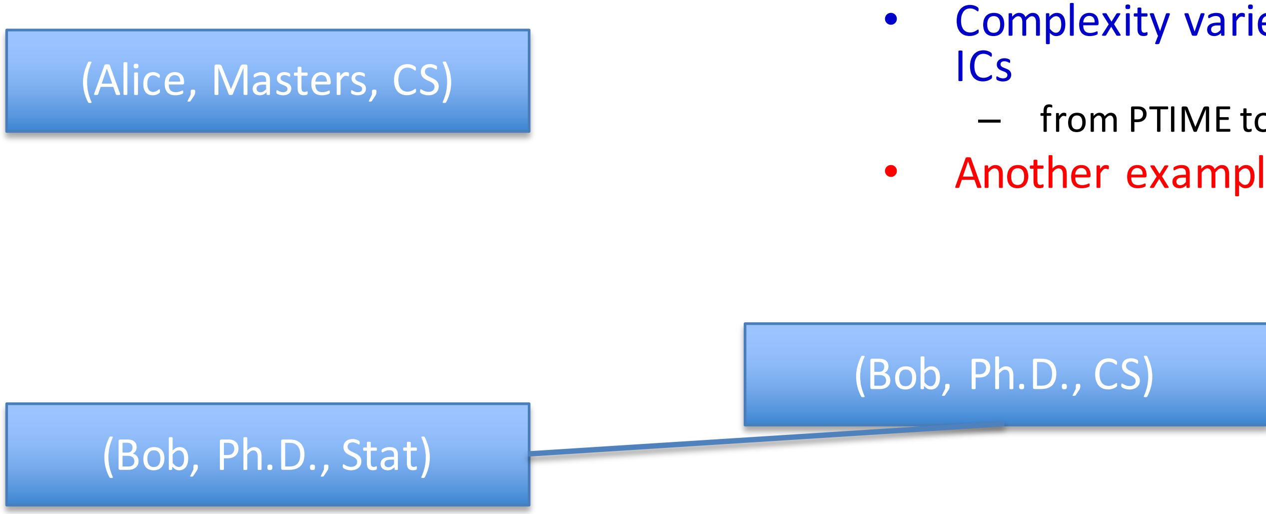
Graph-Theoretic Characterization of Repairs
Conflict Hypergraph

Database D

Student	Degree	Dept
Alice	Masters	CS
Bob	Ph.D.	Stat
Bob	Ph.D.	CS

IC: Student -> Degree, Dept

- What is a hypergraph?
- Vertices
  - Tuples in the database
- Edges
  - Minimal sets of tuples violating a constraint
- Repairs
  - Maximal Independent sets in the conflict graph
- What is an independent set?
  - Maximum vs. maximal
- Complexity varies with queries and ICs
  - from PTIME to Co-NP complete
- Another example on whiteboard



Tractable Queries & Join Graphs

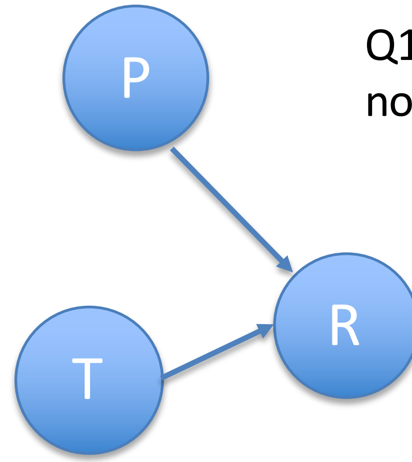
- Query Q
- Join graph G(Q)
- Vertices are database atoms
- edge (L, L')
  - if there is a variable that occurs in a non-key attribute in L and also occurs in L'
    - appears twice if L = L'

- Class C-Tree
  - Q does not have repeated symbol = self-join free
  - G(Q) is a forest (collection of rooted trees, no cycle)
  - every non-key to key join of Q is full (involves whole key)

- CQA is tractable for C-Tree
  - conjunctive queries and FDs

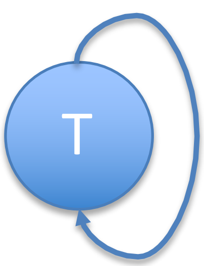
Example

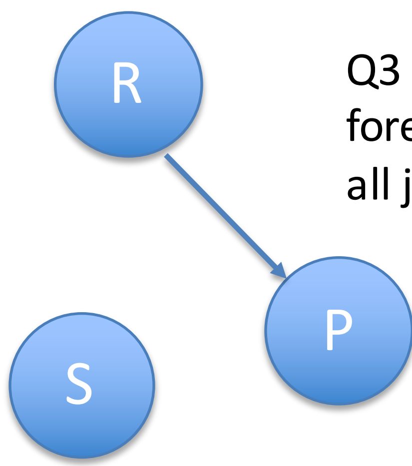
- Q1: P(x, y) ^ R(y, w) ^ T(u, v, y)
- Q2: T(x, y, y)
- Q3: R(x, y) ^ S(w, z) ^ P(y, u)



Q1
not a forest

Q2
not a forest





Q3
forest
all joins are full

Other Repair Semantics

- Attribute-based repairs
  - A-repairs
- Cardinality-based repairs
  - C-repairs

Attribute-based repairs

- Change some attribute values in the existing tuples
- Minimize an aggregate function over changes
  - (tuple, attr, newvalue)
- Examples:
  - for each change: count 1
  - minimize sum of square difference (numeric attribute)
- Decision Problems
  - If there is a repair with cost <= a given budget
  - Repair under range semantics (aggregate queries)

Cardinality-based repairs

- Recall: $\Delta(D, D') = (D - D') \cup (D' - D)$
- Minimize $| \Delta(D, D') |$

Conclusions

- Completes our current discussion on uncertain data
- We covered
  - Probabilistic databases
    - possible world
    - dichotomy
    - Safe vs. unsafe queries
    - #P-hardness
  - Incomplete databases
    - Codd-table, V-table, c-table
  - Inconsistent databases
    - repairs, CQA
- Possible other topics in TBD lectures
  - Probabilistic Relational Model
  - Data exchange and schema mappings
- Next lecture
  - on Thursday (fall break on Tuesday)
  - New topic: Causality (in AI, Stat, DB)