Motivation

Examples
- 4 <-> 9
- Sensor Example
- NYC Taxi Cabs -> Hurricane Sandy vs $100 tip vs Dropoff in Brazil

Core problem: There is no longer one interpretation of the data

Current state of the art:
- Design a schema to account for uncertainty
  - Problem: Now users need to be explicitly aware of uncertainty
  - Problem: Slow, upfront work
- Settle on one interpretation that works for your use case
  - Problem: If the interpretation you pick is wrong, you get errors
  - Problem: The data could be wrong if used for a different use case
  - Problem: Slow, upfront work
- NULL values
  - Problem: Hides uncertain values
- Problem: Null value semantics are awful
  - Any arithmetic with a null value (e.g., NULL + 1) evaluates to NULL
  - Any comparison with null values (e.g., NULL >= 3) evaluates to UNKNOWN
  - 3-Valued Boolean Logic: TRUE, UNKNOWN, FALSE
  - SQL WHERE returns only TRUE values (UNKNOWN and FALSE are dropped)
    - Problem: It's possible for SELECT * FROM R WHERE (X > 3) AND (X <= 3) to return an empty result on a non-empty R

Improved Solution: API for Uncertain/Probabilistic Queries
- Query for 'certain' answers
  - Problem: Uncertain answers may still be useful
- Query for the best interpretation
  - Problem: How do you define "best"?
- Query for all possible interpretations
  - Problem: Hides correlations/anticorrelations
- Probabilistic queries as above, but also compute...
  - ... marginal probabilities of answers
  - ... expectations/variances/other statistical measures of answers
  - ... rank of each possible answer (when this makes sense)

Possible Worlds Semantics

- Each interpretation defines one world
An uncertain database is actually a set of databases, each representing one interpretation or "possible world"

- For now, all of these databases share the same schema.

How do we define query semantics for a set of possible worlds:

- Queries should return a set of "possible answers"
- Naive idea: Run the query independently in each possible world
  - Problem: Inefficient. Can be lots of possible worlds.
  - Problem: Could be impossible. Can be an infinite number of possible worlds
  - But... This still defines a self-consistent set of rules for evaluating queries on uncertain data

Representation Requirements

- Closed
  - There exists a Q’ such that Q’(Rep(D)) == Rep(Q(D))

- Meaningful
  - The representation has to be useful... although for what depends on the application
- ... or better still Bijective
  - Ideally, it would be nice to be able to reconstruct all possible worlds from the representation.

Factorization attempts

- Three types of uncorrelated uncertainty:
  - Row-level: A row is present precisely half of all possible worlds --- and other than the row, everything else is identical between the two halves
  - Attribute-level: There are N copies of all worlds where a row is present, differing only in a single attribute which takes N distinct values --- N may be infinity
  - Open-world: There are an infinite number of worlds with an unbounded number of rows in them, and we have rules for generating more rows

Adding correlations

- Create an integer "world-id"
- Define a function that maps the world-id to a concrete database (or relation) instance
  - ... so how do we define these functions?

V-Tables

Null Value Semantics on Steroids

- 'Label' each Null. i.e., Nulls become Variables
- A V-table is effectively a Function:
  - A possible world is defined by a mapping from labels to nulls
  - Externally provided ruleset defines what's allowed to be in a labeled null

Proving Closure for V-Tables

- Exercise for the reader
- Works for π, x, U, but not σ
• ... because there’s no way to represent a row that “might” be in the result set
• Works under both set and bag semantics
  • ... although the representation may have some duplicate rows that need to be removed

\section*{C-Tables}

\subsection*{V-Tables with an additional "Condition" column}

• Each table gets an added column containing a boolean expression that may reference label symbols
  • When evaluating the V-Table as a Function, plug label values into the boolean expression
  • Boolean expressions that evaluate to false are not present in that specific possible world.

\subsection*{Proving Closure for C-Tables}

• Also an exercise for the reader

\subsection*{Simplified C-Tables (U-Relations)}

• Remove Support for Labeled Nulls

\subsection*{Generalized C-Tables}

• Allow the creation of new variable symbols defined by formulas
  • e.g., \( \{ X + 2 \cdot Y \} \)

\subsection*{Weaker Models}

\subsection*{OR-SET encoding}

• Label tuples that are not in at least one possible world with a ? (this alone is generally called Tuple-independent)
• Use sets of allowable values instead of attributes
• Can not capture correlations

\subsection*{X-Tuples}

• Group tuples into sets of mutually exclusive possibilities (can be combined with OR-SET)

\subsection*{Queries on C-Tables}
Basic query types

- **Certain Answers**
  - Answers in "all" possible worlds

- **Possible Answers**
  - Answers in "any" possible world

Limitations

- Expensive to compute either of these
- Possible produces too much, while certain produces too little.

Tradeoff Points

- **Best Guess (Maximal Prior)** - Pick a (most likely) world and evaluate the query in it
- **Maximal Posterior** - Use probabilities (discussed next class) to pick result rows exceeding a given threshold probability.
- **Sampling** - Pick a set of possible worlds at random and evaluate the query in each of those (more discussed soon)