Mimir: ETL Made On-Demand

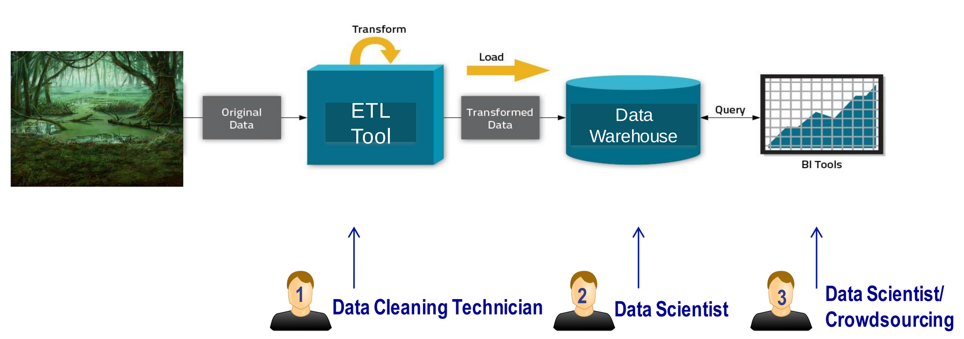
University at Buffalo
The State University of New York™

Poonam Kumari, William Spoth, Aaron Huber, Jon Logan,
Lisa Lu, Oliver Kennedy
Alumni: Arindam Nandi, Niccolo Meneghetti, Vinayak Karuppasamy, Jacob Varghese, Ying Yang

The ODIn Lab @ UB

Motivation

Efficient analytics depends on *accurate, reliable, high-quality* information. However, raw data is messy.

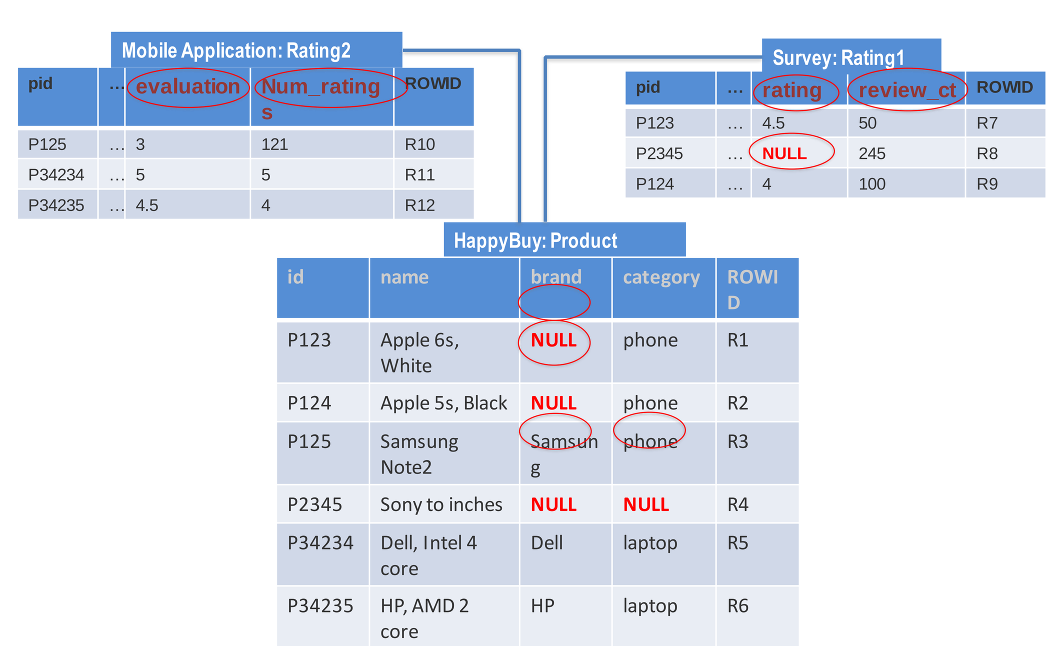


1. Upfront cleaning: clean all messy data before analysis. *Drawbacks*: Unnecessary processing of unused data.
2. Inline cleaning: clean all messy data when analyzing. *Drawbacks*: (1) Unnecessary processing of unused data. (2) Duplication of work.
3. On-demand cleaning: delay the cleaning process until needed and clean incrementally. *Advantages:* Time and cost efficient compared to 1 and 2.

We need a general on-demand cleaning framework.

Example

Alice is an analyst from HappyBuy. She wants to explore the ratings of HappyBuy products.



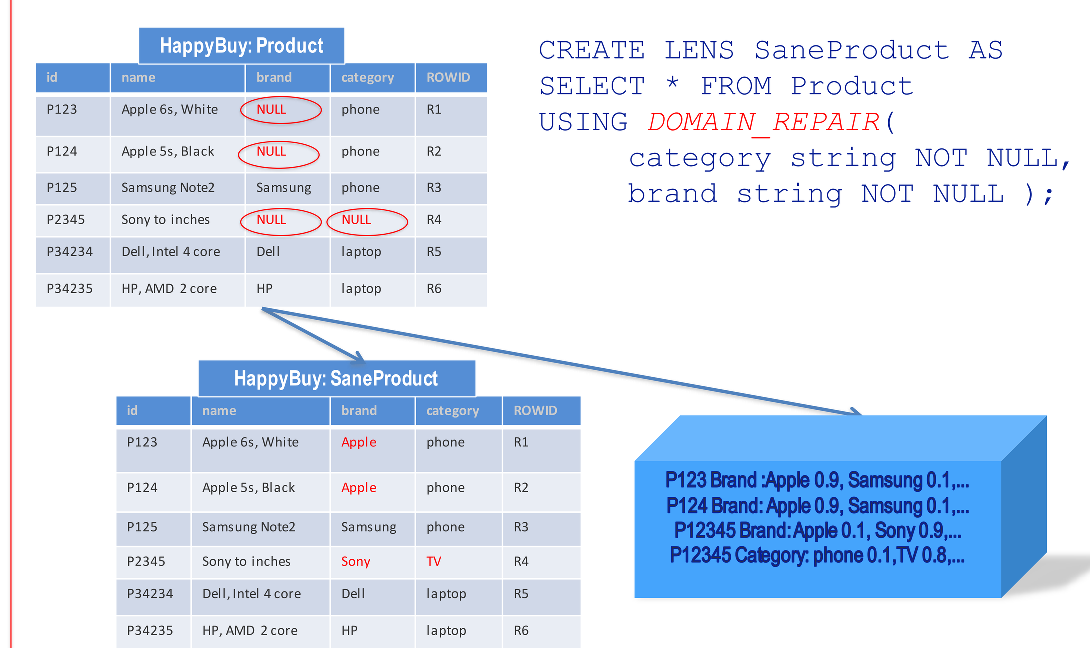
I am interested in phones and TVs and other product with good ratings.

```
SELECT
p.pid,p.category,r.rating,r.review_ct
FROM Product p, Rating r
WHERE p.category IN ('phone', 'TV')
    OR r.rating  > 4
```

Lenses

Domain repair lens

```
CREATE LENS SaneProduct AS
SELECT * FROM Product
USING DOMAIN_REPAIR(
      category string NOT NULL,
      brand string NOT NULL );
```



Lenses make best use of source data and make a **best-effort guess** using the learnt model.

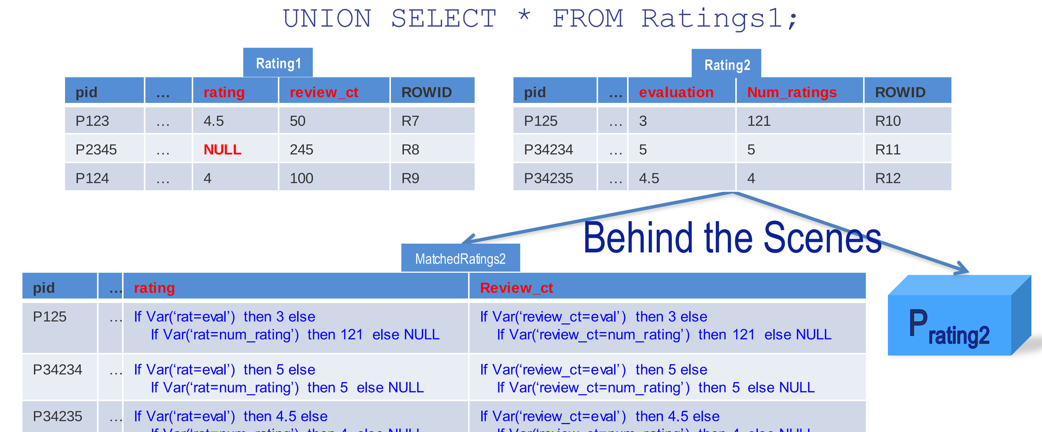
SaneProduct

id	name	brand	category	ROWID
P123	Apple 6s, White	VAR('X',R1)	phone	R1
P124	Apple 5s, Black	VAR('X',R2)	phone	R2
P125	Samsung Note2	Samsung	phone	R3
P2345	Sony to inches	VAR('X',R4)	VAR('Y',R4)	R4
P34234	Dell, Intel 4 core	Dell	laptop	R5
P34235	HP, AMD 2 core	HP	laptop	R6

$P_{product}$ Behind the Scenes

Schema matching lens

```
CREATE LENS MatchedRating2 AS SELECT * FROM Rating2
      USING SCHEMA_MATCHING( pid string, ...,
  rating float, review_ct float, NO LIMIT );
CREATE VIEW AllRatings AS SELECT * FROM MatchedRatings2
      UNION SELECT * FROM Ratings1;
```



Cells in a generalized C-Table can have arbitrary expressions.

JSON shredder lens

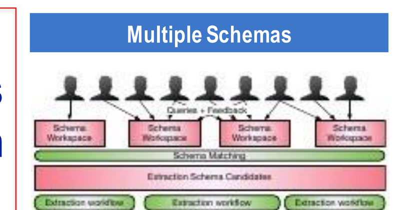
```
{"grad":{"students":[
{name:"Alice",deg:"PhD",credits:"10"},
{name:"Bob",deg:"MS"}, ...]},
"undergrad":{"students":[
{name:"Carol"},{name:"Dave",deg:"U"},
...]}}
```

Flatten JSON Data

gr_st_name	gr_st_deg	…	un_st_names	un_st_deg
Alice	PhD	…	Carol	Null
Bob	MS	…	Dave	U
…	…	…	…	…

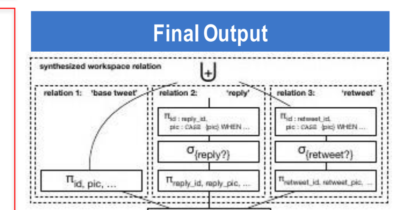
Build functional dependency between columns, this allows us to group columns into 'entities' (parent column) that contain attributes (children columns)

Multiple Schemas



Mapping entities to other entities allows us to preform schema matching on entity selection. This allows simple queries to analyze wide data sets

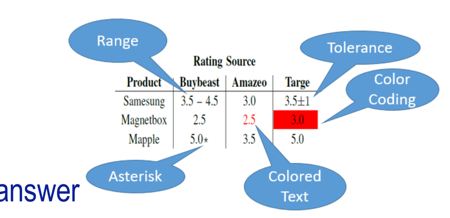
Final Output



User Interface

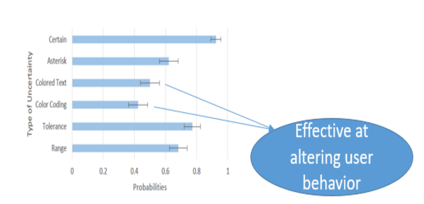
Aim is to design a user interface for presenting query results with attribute-level uncertainty, optimizing for three objectives.

- Familiarity
- Effectiveness
- Efficiency

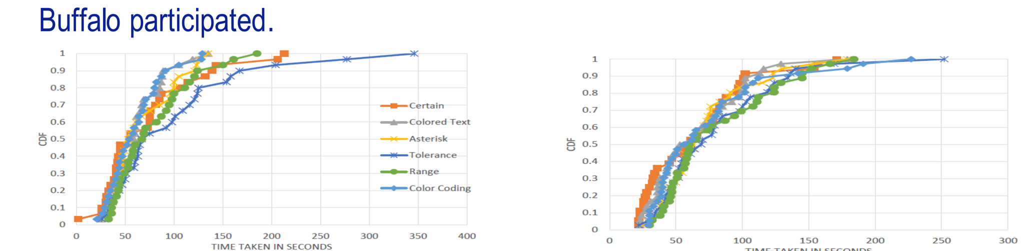


The two primary questions that we sought to answer for each of the representations of uncertainty were

- Is the representation effective at communicating uncertainty?
- What is the cognitive burden of interpreting representation?



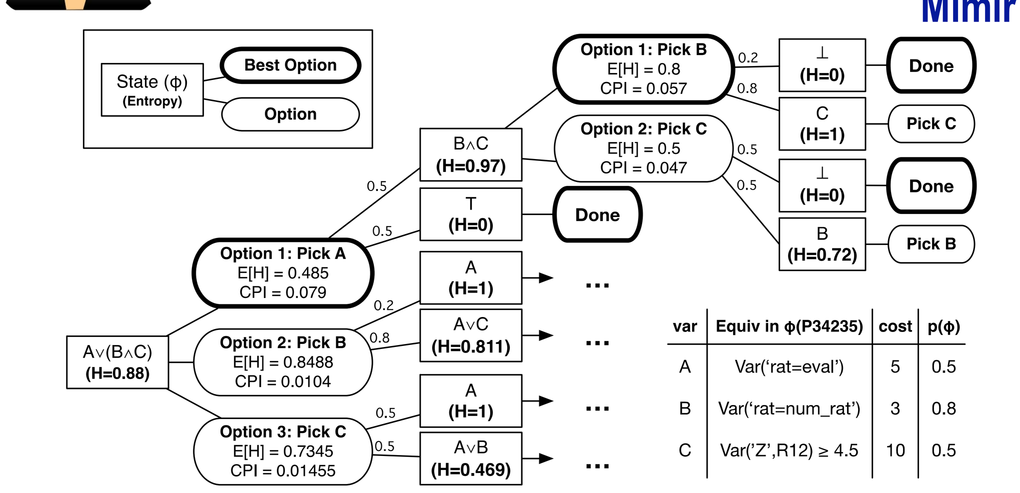
A total of 22 participants drawn from the entire student body of the University at Buffalo participated.



- Time taken to interpret uncertainty is consistent across all forms except Tolerance for CS students.
- Non-CS background participants displayed a quicker decision compared to CS participants in case of asterisk, colored Text and color coding representations. The comparison might suggest that being familiar with the representation (tolerance and ranges) reduces the cognitive burden of interpreting uncertainty.
- As a result of this study, we showed that users made rational decisions more quickly with low-bandwidth uncertainty representations like red text or red backgrounds.

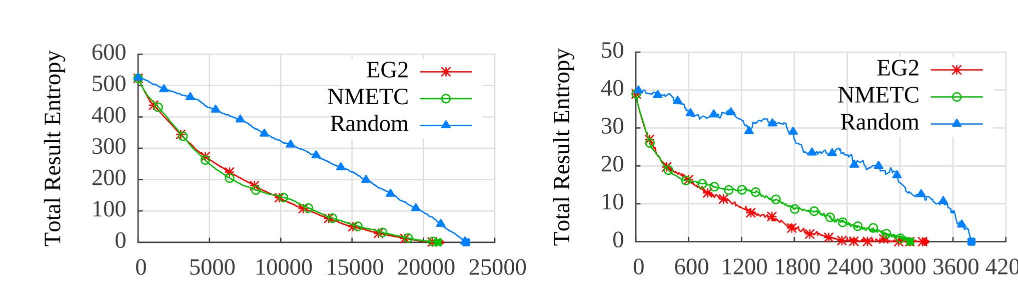
Feedback

We use *cost of perfect information* (CPI) to rank the uncertainties.



var	Equiv in ϕ(P34235)	cost	p(ϕ)
A	Var('rat=eval')	5	0.5
B	Var('rat=num_rat')	3	0.8
C	Var('Z',R12) ≥ 4.5	10	0.5

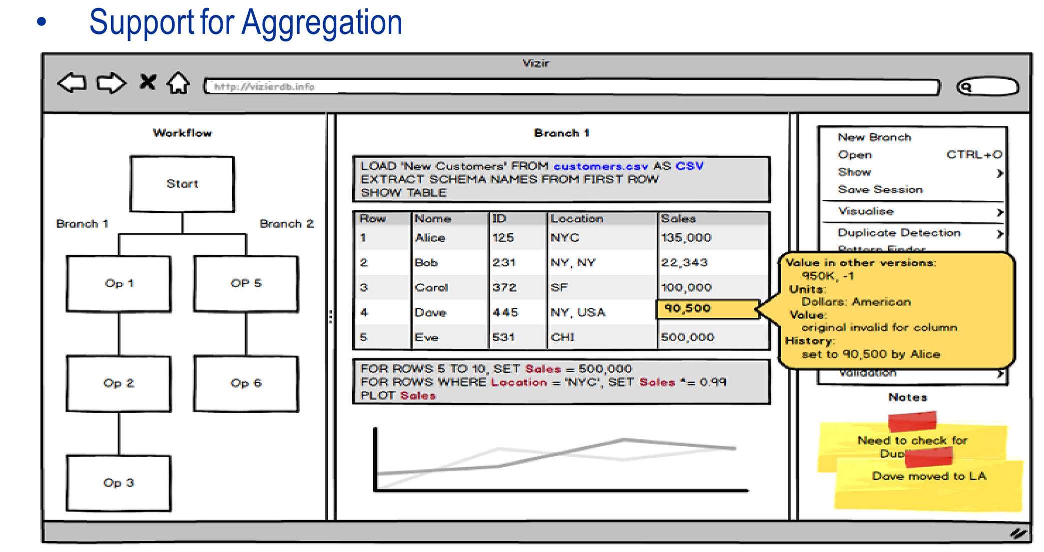
EG2-based CPI method is sufficiently close to NMETC in units of effort invested and has steep curve to produce high-quality results with minimal investment.



Integration with GProM & VisTrails

Integration with GProM provides Mimir rich provenance capabilities:

- GProM uses generic semiring structure to represent multiple forms of provenance:
- Support for Aggregation

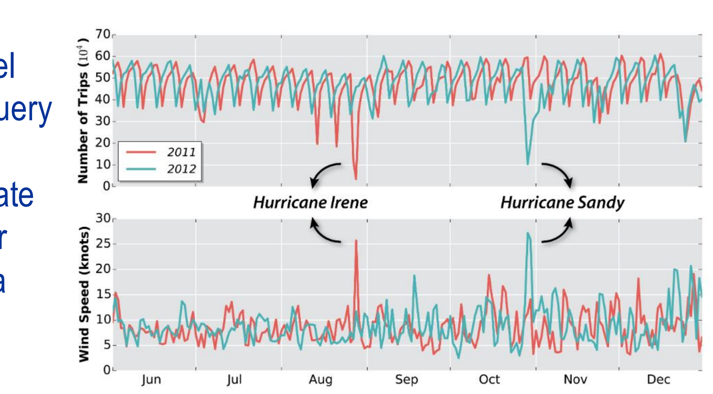


Integration with VisTrails with a spreadsheet UI

- Notebook workflow provenance for visualizations
- Spreadsheet provenance for reproducible ad-hoc data repair.
- Graceful transition from ad-hoc data cleaning to generalizable bulk data processing workflows.

Generic Schemes For Metadata Propogation

- Propagating deterministic metadata at the query level
- Avoids changing Mimir query annotation
- Allows analyst to propagate information through Mimir queries to determine data correlations



Probabilistic System Catalog

Schema-level Information Presentation Responsive To UI

- Clearly represents data schema level information to user
- Allows responsiveness to feedback generated by UI
- Tracks JSON data as it changes, including nested JSON data
- Represents changes as possible schemas and use cases that a user may wish to work on based on current task
- Possible probabilistic information is retained by Mimir to create best guess assumptions of data

Contributions

We propose ***Mimir*** to provide:

- Lens: a structure to represent different kinds of messy data in a uniform way.
- Analysis: presenting (uncertain) query results to user.
- Feedback: improving the data quality in a cost efficient way.

Mimir is supported by NSF Award #1640864, NPS Award #N00244-16-1-0022, and gifts from Oracle.

http://odin.cse.buffalo.edu/research/mimir/