CompSci 516
Data Intensive Computing Systems

Lecture 24
View Maintenance
Provenance
Probabilistic Databases
Crowd Sourcing

Instructor: Sudeepa Roy

Announcements

- Last lecture tomorrow 04/19 (Tuesday), D106, 3:05 pm

- HW5 due on 04/20 (Wednesday), 11:55 pm

- If you are/were unable to attend today's (04/18) lecture, and have questions on the slides, send the instructor an email

Today

Overview of some research areas in databases

- View Materialization and Maintenance
- Data Provenance
- Uncertain Data and Probabilistic Databases
- Crowd Sourcing
- (very briefly) Data Integration

- Understand the high-level ideas and basic techniques
  - The lecture slides will be sufficient for the exams, no additional reading material is needed

View Materialization and Maintenance

[RG] Chapters 25.8-25.10

(slides adapted from the instructor material by the authors)

Views

- Motivation (example)
  - Different groups of analysts within an organization are typically concerned with different aspects of a business
  - It is convenient to define "views" that give each group insight into the relevant business details
  - Other views can be defined or queries can be written using these views
  - Convenient and Efficient

View Example

View
(sales of products by category and state)

```
CREATE VIEW RegionalSales(category, sales, state)
    AS SELECT P.category, S.sales, L.state
       FROM Products P, Sales S, Locations L
       WHERE P.pid=S.pid AND S.locid=L.locid
```

Query
(total sales for each category by state)

```
SELECT R.category, R.state, SUM(R.sales)
FROM RegionalSales AS R
GROUP BY R.category, R.state
```

Query Modification
(SQL does not specify how to evaluate queries on views, but can consider it as a replacement)

```
SELECT R.category, R.state, SUM(R.sales)
FROM (SELECT P.category, S.sales, L.state
    FROM Products P, Sales S, Locations L
    WHERE P.pid=S.pid AND S.locid=L.locid) AS R
GROUP BY R.category, R.state
```

Views and OLAP/Warehousing

- OLAP queries are typically aggregate queries
  - Precomputation is essential for interactive response times
  - The CUBE is in fact a collection of aggregate queries, and precomputation is especially important
  - lots of work on what is best to precompute given a limited amount of space to store precomputed results.

- Warehouses can be thought of as a collection of asynchronously replicated tables and periodically maintained views
  - Factors: size, number of tables involved, many are from external independent databases
  - Has renewed interest in (asynchronous) view maintenance (more later)

View Materialization

- Query Modification may not be efficient
  - when the underlying view is complex
  - even with sophisticated optimization and evaluation
  - esp. when the underlying tables are in a remote database (connectivity and availability)

- Alternative: View Materialization
  - Precompute the view definition and store the result
  - Materialized views can be used as regular relations
  - Provides fast access, like a (very high-level) cache
  - Can create index on views too for further speedup
  - Drawback: to maintain the consistency of the materialized view when the underlying table(s) are updated (View Maintenance)
  - Ideally, we want Incremental View Maintenance algorithms (Lecture 21)

Index on Materialized Views: Examples

```
CREATE VIEW RegionalSales(category, sales, state)
    AS SELECT P.category, S.sales, L.state
       FROM Products P, Sales S, Locations L
       WHERE P.pid=S.pid AND S.locid=L.locid
```

```
SELECT R.category, R.state, SUM(R.sales)
FROM RegionalSales AS R
GROUP BY R.category, R.state
```

- Suppose we precompute RegionalSales and store it with a clustered B+ tree index on [category, state, sales].
  - Then, the query can be answered by an index-only scan.

```
SELECT R.state, SUM(R.sales)
FROM RegionalSales R
WHERE R.category="Laptop"
GROUP BY R.state
```

Index on precomputed view is great!

```
SELECT R.category, SUM(R.sales)
FROM RegionalSales R
WHERE R. state="Wisconsin"
GROUP BY R.category
```

Index is less useful (must scan entire leaf level)

(Research) Issues in View Materialization

1. What views should we materialize, and what indexes should we build on the precomputed results?

2. Given a query and a set of materialized views, can we use the materialized views to answer the query?
   – related to the first question (workload dependent)
   – Try to materialize a small, carefully chosen set of views that can be utilized to quickly answer most of the important queries

3. How frequently should we refresh materialized views to make them consistent with the underlying tables?
   – And how can we do this incrementally?

View Maintenance

- Two steps:
  - Propagate: Compute changes to view when data changes
  - Refresh: Apply changes to the materialized view table

- Maintenance policy: Controls when we do refresh
  - Immediate: As part of the transaction that modifies the underlying data tables
    - + Materialized view is always consistent
    - - updates are slowed
  - Deferred: Some time later, in a separate transaction
    - - View becomes inconsistent
    - + can scale to maintain many views without slowing updates

Types of Deferred Maintenance

Three flavors:

- Lazy:
  - Delay refresh until next query on view; then refresh before answering the query (slows down queries than updates)
- Periodic (Snapshot):
  - Refresh periodically (e.g. once in a day). Queries possibly answered using outdated version of view tuples. Widely used, especially for asynchronous replication in distributed databases, and for warehouse applications
- Event-based or Forced:
  - E.g., Refresh after a fixed number of updates to underlying data tables

- e.g. Snapshot in Oracle 7
  - periodically refreshed by entirely recomputing the view
  - Incremental ”fast refresh” or “simple snapshots” for simpler views (no aggregate, group by, join, distinct etc.)

Provenance

Selected/adapted slides from the keynote by
Prof. Val Tannen, EDBT 2010
(optional material: full slide deck is available on Val's webpage)

Data Provenance

provenance, n.

The fact of coming from some particular source or quarter; origin, derivation [Oxford English Dictionary]

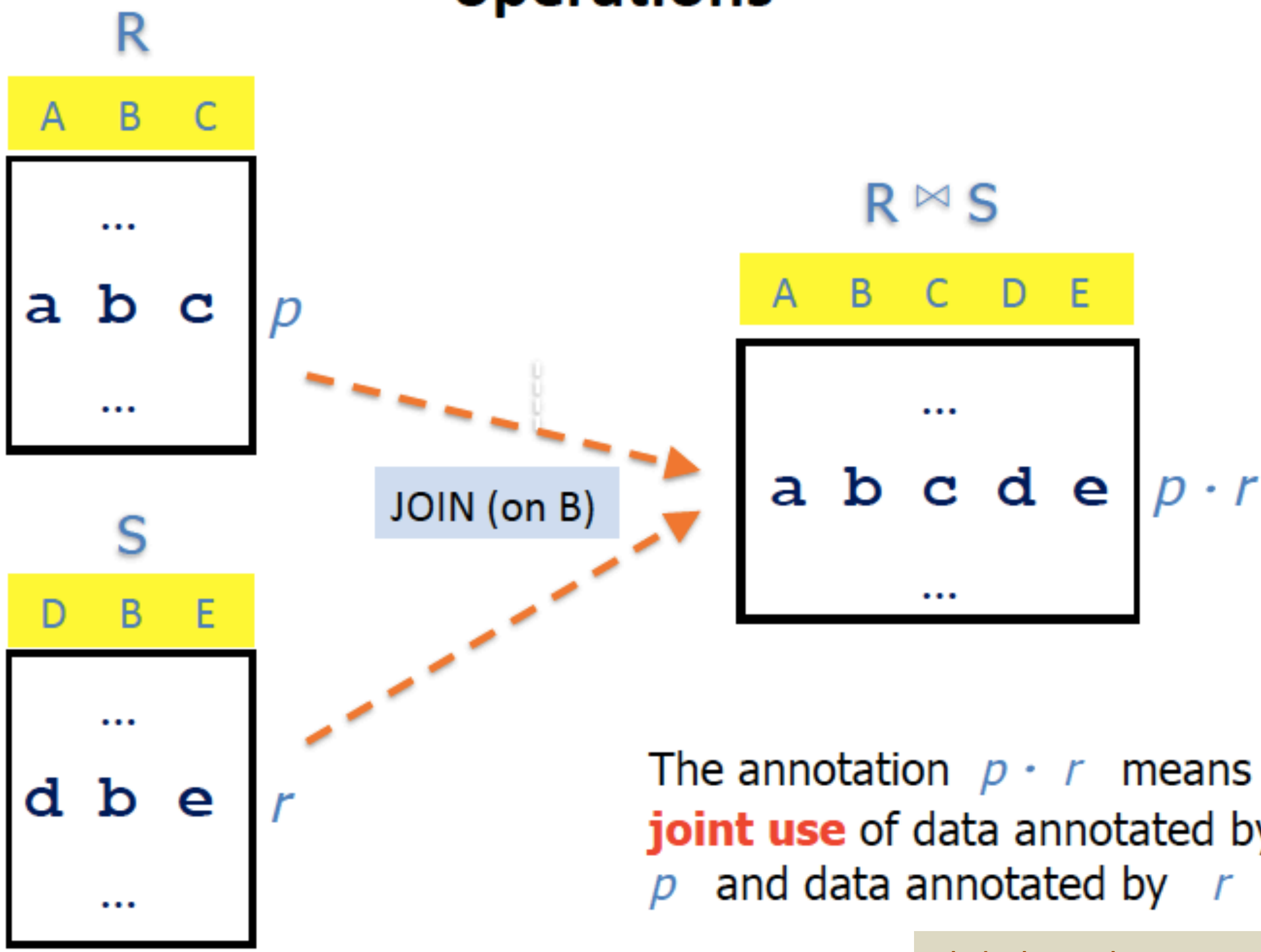
- **Data provenance** [BunemanKhannaTan 01]**:** aims to explain how a particular result (in an experiment, simulation, query, workflow, etc.) was derived.

- Most science today is **data-intensive**. Scientists, eg., biologists, astronomers, worry about data provenance all the time.

Slide by Val Tannen, EDBT 2010

03/24/10 EDBT Keynote, Lausanne 2

Propagating annotations through database operations

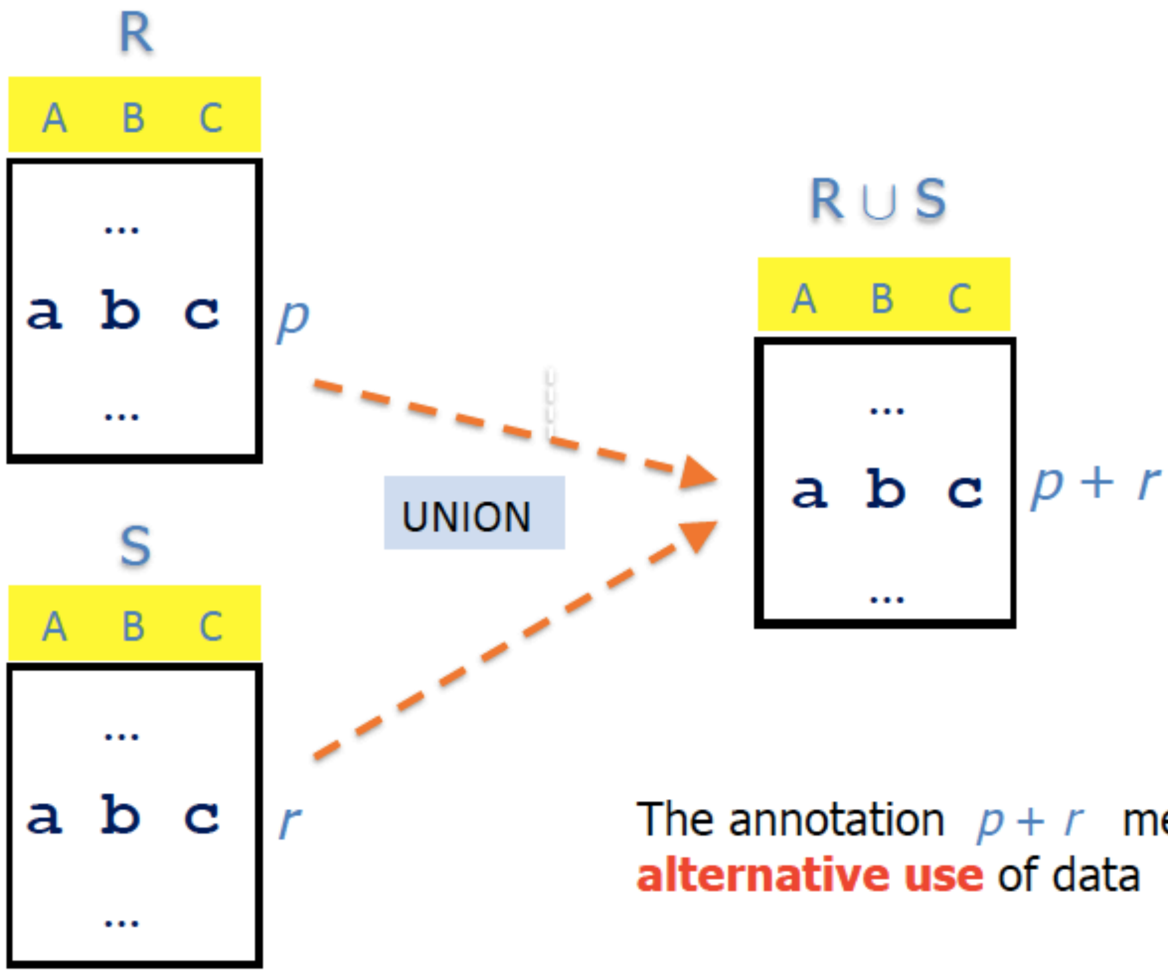


The annotation $p \cdot r$ means **joint use** of data annotated by p and data annotated by r

03/24/10 EDBT Keynote, Lausanne

Slide by Val Tannen, EDBT 2010

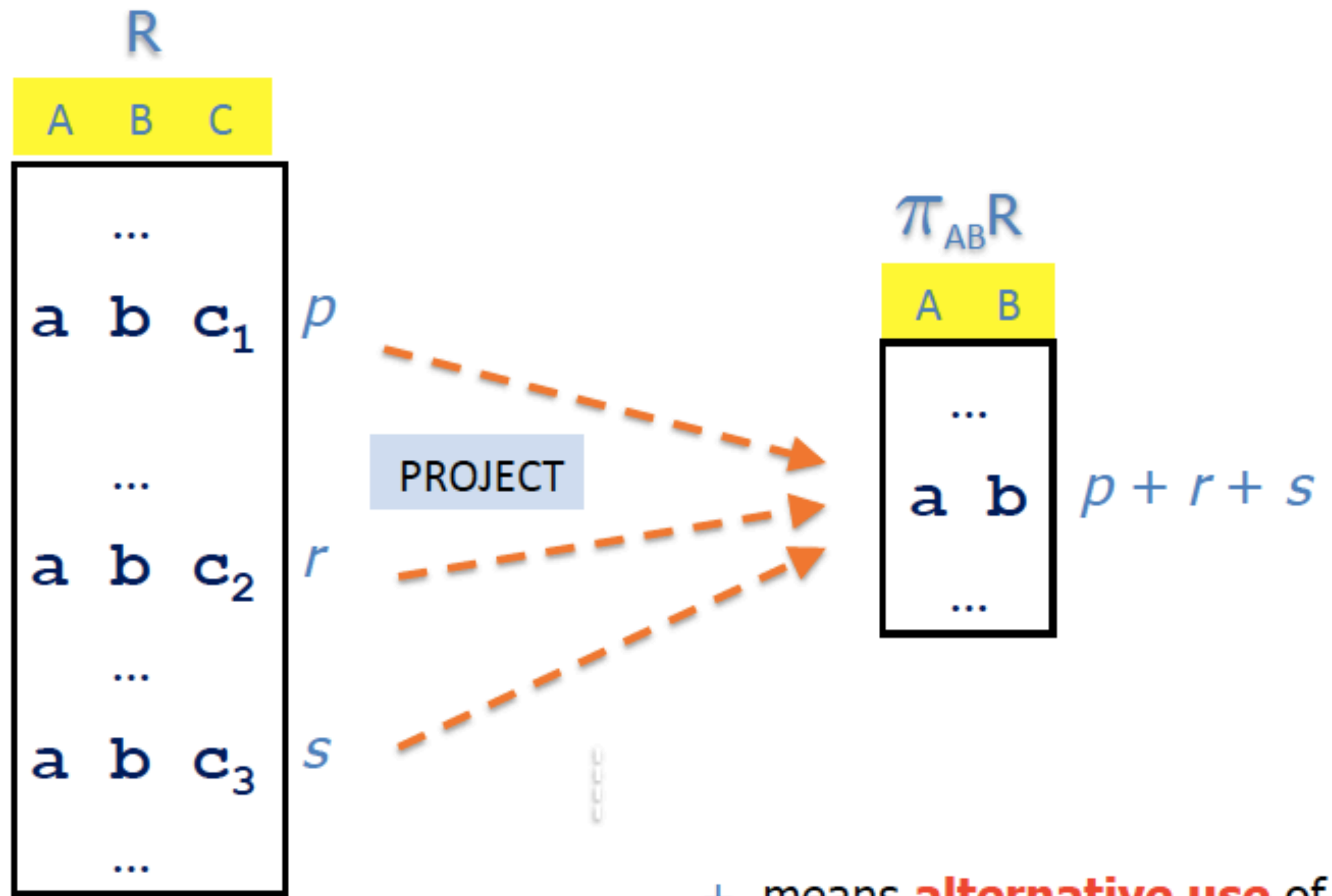
Another way to propagate annotations



The annotation $p + r$ means **alternative use** of data

Slide by Val Tannen, EDBT 2010

Another use of +



$+$ means **alternative use** of data

Slide by Val Tannen, EDBT 2010

An example in positive relational algebra (SPJU)

$$Q = \sigma_{C=e}\pi_{AC}(\pi_{AC}R \bowtie \pi_{BC}R \cup \pi_{AB}R \bowtie \pi_{BC}R)$$

R

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

A	C	
a	c	$(p \cdot p + p \cdot p) \cdot 0$
a	e	$p \cdot r \cdot 1$
d	c	$r \cdot p \cdot 0$
d	e	$(r \cdot r + r \cdot s + r \cdot r) \cdot 1$
f	e	$(s \cdot s + s \cdot r + s \cdot s) \cdot 1$

For selection we multiply
with two special annotations, 0 and 1

03/24/10 EDBT Keynote, Lausanne

Slide by Val Tannen, EDBT 2010

Summary so far

A space of annotations, K

K-**relations**: every tuple annotated with some element from K.

Binary operations on K: $\cdot$ corresponds to joint use (join), and $+$ corresponds to alternative use (union and projection).

We assume K contains special annotations 0 and 1.

"Absent" tuples are annotated with 0!

1 is a "neutral" annotation (no restrictions).

Algebra of annotations? What are the **laws** of $(K, +, \cdot, 0, 1)$?

03/24/10 EDBT Keynote, Lausanne

Slide by Val Tannen, EDBT 2010

Annotated relational algebra

- DBMS query optimizers assume certain equivalences:
  - union is associative, commutative
  - join is associative, commutative, distributes over union
  - projections and selections commute with each other and with union and join (when applicable)
  - Etc., but no $R \bowtie R = R \cup R = R$ (i.e., no idempotence, to allow for bag semantics)
- Equivalent queries should produce same annotations!

Proposition. Above identities hold for queries on K-relations iff $(K, +, \cdot, 0, 1)$ is a **commutative semiring**

03/24/10 EDBT Keynote, Lausanne

Slide by Val Tannen, EDBT 2010

Optional slide

What is a commutative semiring?

different meanings (examples later): + = plus_K, . = mult_K, $0 = 0_K$, $1 = 1_K$

An algebraic structure $(K, +, \cdot, 0, 1)$ where:

- K is the domain
- $+$ is associative, commutative, with 0 identity
- $\cdot$ is associative, with 1 identity
- $\cdot$ distributes over $+$
- $a \cdot 0 = 0 \cdot a = 0$

semiring

- $\cdot$ is also **commutative**

Unlike ring, no requirement for inverses to $+$

03/24/10 EDBT Keynote, Lausanne

Slide by Val Tannen, EDBT 2010

Back to the example

R

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

Q

A	C	
a	c	$(p \cdot p + p \cdot p) \cdot 0$
a	e	$p \cdot r \cdot 1$
d	c	$r \cdot p \cdot 0$
d	e	$(r \cdot r + r \cdot s + r \cdot r) \cdot 1$
f	e	$(s \cdot s + s \cdot r + s \cdot s) \cdot 1$

03/24/10 EDBT Keynote, Lausanne

Slide by Val Tannen, EDBT 2010

Using the laws: polynomials

R

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

Q

A	C	
a	e	pr
d	e	$2r^2 + rs$
f	e	$rs + 2s^2$

Polynomials with coefficients in $\mathbb{N}$ and annotation tokens as indeterminates p, r, s capture a very general form of **provenance**

Slide by Val Tannen, EDBT 2010

Provenance reading of the polynomials

R

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

Q

A	C	
a	e	pr
d	e	$2r^2 + rs$
f	e	$rs + 2s^2$

- three different ways to derive d e
- two of the ways use only r
- but they use it twice
- the third way uses r once and s once

03/24/10 EDBT Keynote, Lausanne

Slide by Val Tannen, EDBT 2010

Provenance Semiring is the Most General Semiring and Has Several Useful "Specialization"

= homomorphism

R

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

Q

A	C	
a	e	pr
d	e	$2r^2 + rs$
f	e	$rs + 2s^2$

No need to recompute that complex query

- Set semantic
  - Given input tuples exist or not, whether output tuples exist
  - $K = \{T, F\}$, $\text{mult}_K = \wedge$, $\text{plus}_K = \vee$, $1_K = T$, $0_K = F$
  - e.g. $p = r = T$, $s = F$. Then
  - annotation of (p, r): $T \wedge T = T$
  - annotation of (d, e): $r \vee (r \wedge s) = T$
  - annotation of (f, e): $(r \wedge s) \vee s = F$

(adapted from) slide by Val Tannen, EDBT 2010

Provenance Semiring is the Most General Semiring and Has Several Useful "Specialization"

= homomorphism

R

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

Q

A	C	
a	e	pr
d	e	$2r^2 + rs$
f	e	$rs + 2s^2$

- Bag semantic
  - Given multiplicity of input tuples, compute multiplicities of output tuples
  - K = N (natural nums), mult_K = *, plus_K = +, 1_K = 1, 0_K = 0
  - e.g. p = 2, r = 1, s = 3. Then
  - annotation of (p, r): 2*1 = 2
  - annotation of (d, e): $2*1^2 + 1*3 = 5$
  - annotation of (f, e): $1*3 + 2*3^2 = 21$

No need to recompute that complex query

(adapted from) slide by Val Tannen, EDBT 2010

Provenance Semiring is the Most General Semiring and Has Several Useful "Specialization"

(adapted from) slide by Val Tannen, EDBT 2010
other examples in the tutorial

R

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

Q

A	C	
a	e	pr
d	e	$2r^2 + rs$
f	e	$rs + 2s^2$

No need to recompute that complex query

Applications in probabilistic databases (see later)

- Positive Boolean Expression (PosBool) or Lineage
  - Given variables for input tuples, find Boolean expressions for the output tuples (condition of existence)
  - K = BoolExp(X) (set of input variables is X), $mult_K = \wedge$, $plus_K = \vee$, 1_K = T, 0_K = F
  - e.g. given p, r, s. Then
  - annotation of (p, r): pr
  - annotation of (d, e): $(r \wedge r) \vee (r \wedge s) = r$
  - annotation of (f, e): $(r \wedge s) \vee (s \wedge s) = s$

for (d, e) to exist in the output, it suffices as long as (d, b, e) exist in the output

Low-hanging fruit: deletion propagation

R

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

Q

A	C	
a	e	pr
d	e	$2r^2 + rs$
f	e	$rs + 2s^2$

Q

A	C	
a	e	0
d	e	0
f	e	$2s^2$

Q

A	C	
f	e	$2s^2$

Delete **d b e** from R ?

Set $r = 0$!

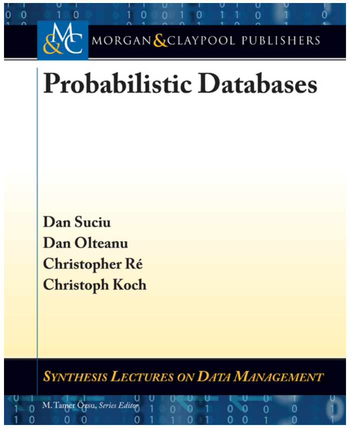
No need to recompute that complex query

Slide by Val Tannen, EDBT 2010

03/24/10 EDBT Keynote, Lausanne

Probabilistic Databases

Selected/adapted slides on the Probabilistic Database book by
Prof. Dan Suciu, 2014
(optional material: full slide decks are available on Dan's webpage)



Probabilistic Databases

- Data: standard relational data, plus probabilities that measure the degree of uncertainty
- Queries: standard SQL queries, whose answers are annotated with output probabilities

Slide by Dan Suciu

A Little History of Probabilistic DBs

Early days

- Wong'82
- Shoshani'82
- Cavallo&Pittarelli'87
- Barbara'92
- Lakshmanan'97,'01
- Fuhr&Roellke'97
- Zimanyi'97

Main challenge:
Query Evaluation
(=Probabilistic Inference)

Recent work

- Stanford (Trio)
- UW (MystiQ)
- Cornell (MayBMS)
- Oxford (MayBMS)
- U.of Maryland
- IBM Almaden (MCDB)
- Rice (MCDB)
- U. of Waterloo
- UBC
- U. of Florida
- Purdue University
- U. of Wisconsin

Unfortunately, no "practical/usable" prob. db. systems

Slide by Dan Suciu

Why?

Many applications need to manage uncertain data

- Information extraction
- Knowledge representation
- Fuzzy matching
- Business intelligence
- Data integration
- Scientific data management
- Data anonymization

Slide by Dan Suciu

What?

- Probabilistic Databases extend Relational Databases with probabilities
- Combine Formal Logic with Probabilistic Inference
- Requires a new thinking for both databases and probabilistic inference

[Gupta'2006]

Slide by Dan Suciu

Example 1: Information Extraction

```
52-A Goregaon West Mumbai 400 076
```

CRF

Standard DB: keep the most likely extraction

Id	House_no	Area	City	Pincode	Prob
1	52	Goregaon West	Mumbai	400 062	0.1
1	52-A	Goregaon	West Mumbai	400 062	0.2
1	52-A	Goregaon West	Mumbai	400 062	0.5
1	52	Goregaon	West Mumbai	400 062	0.2

Probabilistic DB: keep most/all extractions to increase recall

[Stoyanovich'2011]

Slide by Dan Suciu

Example 2: Modeling Missing Data

id	age	edu	inc	nw
t_1	20	HS	?	?
t_2	20	BS	50K	100K
t_3	20	?	50K	?
t_4	20	HS	100K	500K
t_5	20	?	?	?
t_6	20	HS	50K	100K
t_7	20	HS	50K	500K
t_8	?	HS	?	?
t_9	30	BS	100K	100K
t_{10}	30	?	100K	?
t_{11}	30	HS	?	?
t_{12}	30	MS	?	?
t_{13}	40	BS	100K	100K
t_{14}	40	HS	?	?
t_{15}	40	BS	50K	500K
t_{16}	40	HS	?	500K
t_{17}	40	HS	100K	500K

Standard DB: NULL

Probabilistic DB:
distribution on possible values

id	age	edu	inc	nw	prob
$t_{12}.1$	30	MS	50K	100K	0.30
$t_{12}.2$	30	MS	50K	500K	0.45
$t_{12}.3$	30	MS	100K	100K	0.10
$t_{12}.4$	30	MS	100K	500K	0.15

[Beskales'2009]

Slide by Dan Suciu

Example 3: Data Cleaning

This slide should give you some idea about the goals in data cleaning!
(e.g. maintain key constraints or FDs)

Unclean Data

ID	Name	ZIP	Birth Date
1	Green	51359	781310
2	Green	51358	781210
3	Peter	30128	870932
4	Peter	30128	870932
5	Gree	51359	19771210
6	Chuck	51359	19460924

One-shot Duplicate Detection

Single Repair

{1,2}
{3,4}
{5}
{6}

Standard DB
cleaning data means choosing one possible repair

Uncertain Duplicate Detection

Multiple Repairs

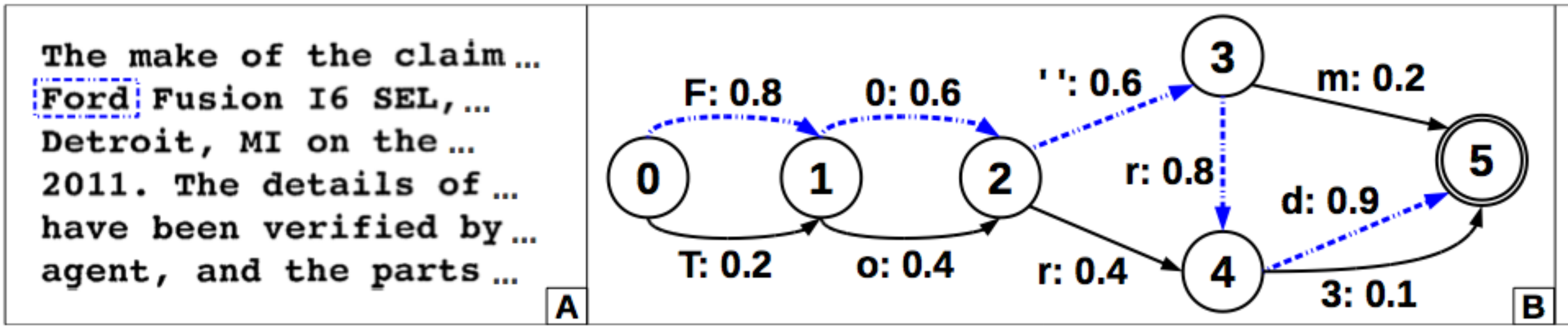
{1,2,5}	{1,2}	{1,5,6}
{3,4}	{3,4}	{2}
{6}	{5}	{3,4}
	{6}	

Probabilistic DB
keep many/all possible repairs

[Kumar’2011]

Slide by Dan Suciu

Example 4: OCR



They use OCRopus from Google Books: output is a stohastic automaton
Traditionally: retain only the Maximum Apriori Estimate (MAP)
With a probabilistic database: may retain several alternative recognitions: increase recall

```
SELECT DocId, Loss
FROM Claims
WHERE Year = 2010
    AND DocData LIKE '%Ford%’;
```

Slide by Dan Suciu

Summary of Applications

- Structured, but uncertain data
- Modeled as probabilistic data
- Answers to SQL queries annotated with probabilities

Probabilistic database:
- Combine data management with probabilistic inference

Slide by Dan Suciu

Review: Complexity of Query Evaluation

Query Q, database D

- Data complexity:
  fix Q, complexity = f(D)

- Query complexity:
  fix D, complexity = f(Q)

- Combined complexity: complexity = f(D,Q)

Moshe Vardi

Data complexity is unique to database research

All query languages that exist today in db systems have Poly-time data complexity (SQL, Datalog, Datalog+negation, Xquery for XML)

Slide by Dan Suciu

Incomplete Database

Definition An Incomplete Database is a finite set of database instances **W** = (W_1, W_2, …, W_n)

Each W_i is called a possible world

Slide by Dan Suciu

Incomplete Database

Definition An Incomplete Database is a finite set of database instances **W** = (W_1, W_2, …, W_n)

Each W_i is called a possible world

W_1	W_2	W_3	W_4

W_1:

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Slide by Dan Suciu

Incomplete Database

Definition An Incomplete Database is a finite set of database instances **W** = (W_1, W_2, …, W_n)

Each W_i is called a possible world

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3

W_4

Slide by Dan Suciu

Incomplete Database

Definition An Incomplete Database is a finite set of database instances **W** = (W_1, W_2, …, W_n)

Each W_i is called a possible world

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3

Owner

Name	Object
Jim	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

W_4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Slide by Dan Suciu

Incomplete Database: Query Semantics

Definition Given query Q, incomplete database **W**:
- An answer t is certain, if $\forall W_i$, $t \in Q(W_i)$
- An answer t is possible if $\exists W_i$, $t \in Q(W_i)$

Slide by Dan Suciu

Incomplete Database: Query Semantics

Definition Given query Q, incomplete database **W**:

- An answer t is certain, if $\forall W_i$, $t \in Q(W_i)$
- An answer t is possible if $\exists W_i$, $t \in Q(W_i)$

Q(z) = Owner(z,x), Location(x,t,'Office')

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

W_4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Slide by Dan Suciu

Incomplete Database: Query Semantics

Definition Given query Q, incomplete database **W**:
- An answer t is certain, if $\forall W_i$, $t \in Q(W_i)$
- An answer t is possible if $\exists W_i$, $t \in Q(W_i)$

Q(z) = Owner(z,x), Location(x,t,'Office')

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

Q=

Joe

W_3

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

Q=

Joe

W_4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

Slide by Dan Suciu

Incomplete Database: Query Semantics

Definition Given query Q, incomplete database **W**:
- An answer t is certain, if $\forall W_i$, $t \in Q(W_i)$
- An answer t is possible if $\exists W_i$, $t \in Q(W_i)$

Q(z) = Owner(z,x), Location(x,t,'Office')

Certain answers to Q: Joe
Possible answers to Q: Joe, Jim

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

Q=

Joe

W_3

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

Q=

Joe

W_4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

Slide by Dan Suciu

Probabilistic Database

Definition A Probabilistic Database is (**W**, P), where **W** is an incomplete database, and P: **W** → [0,1] a probability distribution: $\Sigma_{i=1,n} P(W_i) = 1$

Slide by Dan Suciu

Probabilistic Database

Definition A Probabilistic Database is (**W**, P), where **W** is an incomplete database, and P: **W** → [0,1] a probability distribution: $\Sigma_{i=1,n} P(W_i) = 1$

W_1

Owner 0.3

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2

Owner 0.4

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3

Owner 0.2

Name	Object
Jim	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

W_4

Owner 0.1

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Slide by Dan Suciu

Probabilistic Database: Query Semantics

Definition Given query Q, probabilistic database (**W**,P):

- The marginal probability of an answer t is:
  $P(t) = \Sigma \; \{ P(W_i) \mid W_i \in \mathbf{W}, \; t \in Q(W_i) \}$

Slide by Dan Suciu

Probabilistic Database: Query Semantics

Definition Given query Q, probabilistic database (**W**,P)

- The marginal probability of an answer t is:

$P(t) = \Sigma \ \{ P(W_i) \mid W_i \in \mathbf{W}, \ t \in Q(W_i) \}$

$Q(z) = \mathrm{Owner}(z,x), \mathrm{Location}(x,t,\text{'Office'})$

W_1 — 0.3

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2 — 0.4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3 — 0.2

Owner

Name	Object
Jim	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

W_4 — 0.1

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Slide by Dan Suciu

Probabilistic Database: Query Semantics

Definition Given query Q, probabilistic database (**W**,P)

- The marginal probability of an answer t is:
  $P(t) = \Sigma \{ P(W_i) \mid W_i \in \mathbf{W},\ t \in Q(W_i) \}$

Q(z) = Owner(z,x), Location(x,t,'Office')

W_1 — 0.3

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

W_2 — 0.4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

Q=

Joe

W_3 — 0.2

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

Q=

Joe

W_4 — 0.1

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

Slide by Dan Suciu

Probabilistic Database: Query Semantics

Definition Given query Q, probabilistic database (**W**,P)

- The marginal probability of an answer t is:
  $P(t) = \Sigma \{ P(W_i) \mid W_i \in \mathbf{W},\ t \in Q(W_i) \}$

Q(z) = Owner(z,x), Location(x,t,'Office')

P(Joe) = 1.0
P(Jim) = 0.4

W_1 — 0.3

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

W_2 — 0.4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

Q=

Joe

W_3 — 0.2

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

Q=

Joe

W_4 — 0.1

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

Slide by Dan Suciu

Discussion

- Intuition: a probabilistic database says that the database can be in one of possible states, each with a probability

- Possible query answers: a set of answers annotated with probabilities:

  $(t_1, p_1), (t_2, p_2), (t_3, p_3), \ldots$

  Usually: $p_1 \geq p_2 \geq p_3 \geq \ldots$

- Problem: the number of possible world in a probabilistic database is astronomically large. To represent it, we impose some restrictions
  - independence and/or disjointness of tuples

Slide by Dan Suciu

Independent, Disjoint Tuples

Definition Given a probabilistic database (**W**, P).
Two tuples t_1, t_2 are called:

- Independent, if: $P(t_1\ t_2) = P(t_1)\ P(t_2)$
- Disjoint (or exclusive), if: $P(t_1 t_2) = 0$

Slide by Dan Suciu

Independent, Disjoint Tuples

Definition Given a probabilistic database (**W**, P).
Two tuples t_1, t_2 are called:

- Independent, if: $P(t_1\, t_2) = P(t_1)\, P(t_2)$
- Disjoint (or exclusive), if: $P(t_1 t_2) = 0$

Definition A probabilistic database is called Block-Independent-Disjoint (BID), if its tuples are grouped into blocks such that:

- Tuples from the same block are disjoint
- Tuples from different blocks are independent

Slide by Dan Suciu

Example: BID Table

BID Table

Object	**Time**	Loc	P
Laptop77	9:07	Rm444	p_1
Laptop77	9:07	Hall	p_2
Book302	9:18	Office	p_3
Book302	9:18	Rm444	p_4
Book302	9:18	Lift	p_5

Disjoint

Disjoint

Inde-pen-dent

- At most two tuples in a world
- At most one tuple from the first block
- At most one tuple from the second block

Possible worlds

Slide by Dan Suciu

The Query Evaluation Problem

Given: a probabilistic database D, a query Q, and output tuple t

Compute: P(t)

Note: D has, say, 1000000 tuples,
while the number of possible worlds is $2^{1000000}$

Challenge: compute P(t) efficiently, in the size of D

Data complexity: the complexity of P depends dramatically on Q

Two approaches to query evaluation on tuple-independent probabilistic databases

1. Intensional query evaluation

2. Extensional query evaluation

(adapted from)Slide by Dan Suciu

Approach 1: Intensional Query Evaluation

Query Q + database D → lineage (provenance) expression F_Q

Compute P(F) using a general model counting system

Revisit the third (last) example
of provenance semiring on slide 27!

In general, computationally hard (weighted model counting)

But poly-time for some Q or Q, D

(adapted from) Slide by Dan Suciu

Example: Intensional Query Evaluation

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

$Q = R(x), S(x,y)$

R

x	
a1	X1
a2	X2
a3	X3

S

x	y	
a1	b1	Y1
a1	b2	Y2
a2	b3	Y3
a2	b4	Y4
a2	b5	Y5

$F_Q = X1\,Y1 \lor X1\,Y2 \lor X2\,Y3 \lor X2\,Y4 \lor X2\,Y5$

Now compute $Pr[F_Q]$, given Pr[X1]., Pr[X2], …. and assuming the variables are independent

For some provenance formulas, probability can be computed in poly-time

- Example 1: If the formula is "read once"
  - see next slide

- Example 2: If poly-size knowledge compilation forms (OBDD, FBDD) exist
  - similar idea like read-once
  - not covered in this class

Slide by Dan Suciu

Read-Once Boolean Formulas

A Boolean formula F is called read-once if it can be written such that every Boolean variable occurs only once

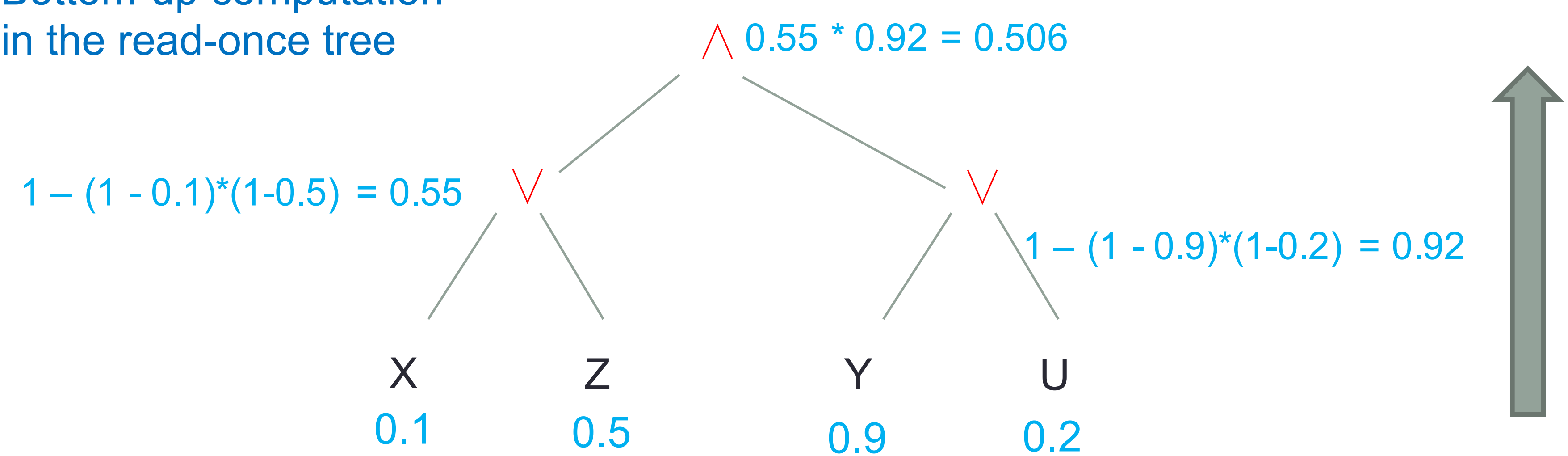
- P(F) can be computed in linear time (independence):

$$P(F_1 \wedge F_2) = P(F_1) \times P(F_2)$$
$$P(F_1 \vee F_2) = 1 - (1 - P(F_1)) \times (1 - P(F_2))$$

$(X \vee Z) \wedge (Y \vee U)$

$P_X = 0.1$, $P_Z = 0.5$,
$P_Y = 0.9$, $P_U = 0.2$

Bottom-up computation in the read-once tree



Slide by Dan Suciu

Read-Once Example

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q = R(x), S(x,y)

R

x	
a1	X1
a2	X2
a3	X3

S

x	y	
a1	b1	Y1
a1	b2	Y2
a2	b3	Y3
a2	b4	Y4
a2	b5	Y5

F_Q = X1 Y1 V X1 Y2 V X2 Y3 V X2 Y4 V X2 Y5

= X1 (Y1 V Y2) V X2 (Y3 V Y4 V Y5)

Read-once

Approach 2: Extensional Query Evaluation

- Main idea:
  - Modify each operator to compute output probabilities
  - Correct plans are "safe plans" (work for all databases)
  - Not always exist

Slide by Dan Suciu

An Example

Boolean query

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

P(Q) =

S

x	y	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

R

x	P
a1	p1
a2	p2
a3	p3

Slide by Dan Suciu

An Example

Boolean query

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

$$P(Q) = 1-(1-q1)*(1-q2)$$

R

x	P
a1	p1
a2	p2
a3	p3

S

x	y	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Slide by Dan Suciu

An Example

Boolean query

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

$Q() = R(x), S(x,y)$

$$P(Q) = p1*[\ 1-(1-q1)*(1-q2)\ ]$$

R

x	P
a1	p1
a2	p2
a3	p3

S

x	y	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Slide by Dan Suciu

An Example

Boolean query

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

P(Q) = p1*[1-(1-q1)*(1-q2)]

1-(1-q3)*(1-q4)*(1-q5)

R

x	P
a1	p1
a2	p2
a3	p3

S

x	y	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Slide by Dan Suciu

An Example

Boolean query

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

P(Q) = p1*[1-(1-q1)*(1-q2)]

p2*[1-(1-q3)*(1-q4)*(1-q5)]

R

x	P
a1	p1
a2	p2
a3	p3

S

x	y	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Slide by Dan Suciu

An Example

Boolean query

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

$$P(Q) = 1- \{1- p1*[\ 1-(1-q1)*(1-q2)\ ]\} *$$
$$\{1- p2*[\ 1-(1-q3)*(1-q4)*(1-q5)]\}$$

Condition for join on a1 or on a2

R

x	P
a1	p1
a2	p2
a3	p3

S

x	y	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Slide by Dan Suciu

An Example

Boolean query

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

$$P(Q) = 1- \{1- p1*[\ 1-(1-q1)*(1-q2)\ ]\} *$$

$$\{1-\ p2*[\ 1-(1-q3)*(1-q4)*(1-q5)]\}$$

One can compute P(Q) in PTIME in the size of the database D

R

x	P
a1	p1
a2	p2
a3	p3

S

x	y	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Slide by Dan Suciu

Extensional Operators

Independent join

A	B	P
a1	b1	p1*q1
a1	b2	p1*q2
a2	b3	p2*q3
a2	b4	p2*q4
a2	b5	p2*q5

$\bowtie^{i}$

R(A)

A	P
a1	p1
a2	p2
a3	p3

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Slide by Dan Suciu

Extensional Operators

Independent join

A	B	P
a1	b1	p1*q1
a1	b2	p1*q2
a2	b3	p2*q3
a2	b4	p2*q4
a2	b5	p2*q5

$$\bowtie^{i}$$

R(A)

A	P
a1	p1
a2	p2
a3	p3

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Independent project

A	P
a1	1 - (1-q1)*(1-q2)
a2	1 - (1-q3)*(1-q4)*(1-q5)

$$\Pi^{i}_{A}$$

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Slide by Dan Suciu

Extensional Operators

Independent join

A	B	P
a1	b1	p1*q1
a1	b2	p1*q2
a2	b3	p2*q3
a2	b4	p2*q4
a2	b5	p2*q5

$\bowtie^{i}$

R(A)

A	P
a1	p1
a2	p2
a3	p3

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Independent project

A	P
a1	1 - (1-q1)*(1-q2)
a2	1 - (1-q3)*(1-q4)*(1-q5)

Π^{i}_{A}

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Selection

A	B	P
a2	b2	q3
a2	b3	q4
a2	b2	q5

$\sigma_{A=a2}$

S(A,B)

A	B	P
a1	b1	q1
a1	b1	q2
a2	b2	q3
a2	b3	q4
a2	b2	q5

Slide by Dan Suciu

SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x

$Q() = R(x), S(x,y)$

$$P(Q) = 1 - [1-p_1*(1-(1-q_1)*(1-q_2))] *[1- p_2*(1-(1-q_3)*(1-q_4)*(1-q_5))]$$

Old Example

R

x	P
a_1	p_1
a_2	p_2
a_3	p_3

S

x	y	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

Slide by Dan Suciu

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

$Q() = R(x), S(x,y)$

$$P(Q) = 1 - [1-p_1*(1-(1-q_1)*(1-q_2))] *[1- p_2*(1-(1-q_3)*(1-q_4)*(1-q_5))]$$

$$1-(1-p_1q_1)(1-p_1q_2)(1-p_2q_3)(1-p_2q_4)(1-p_2q_5)$$

$$1-\{1-p_1[1-(1-q_1)(1-q_2)]\}^* \{1-p_2[1-(1-q_4)(1-q_5)(1-q_6)]\}$$

Wrong

$p_1 q_1$
$p_1 q_2$
$p_2 q_3$
$p_2 q_4$
$p_2 q_5$

Π_Φ

$\bowtie$

R(x) S(x,y)

No longer independent

Right

Π_Φ

Independent

$\bowtie$

$1-(1-q_1)(1-q_2)$
$1-(1-q_4)(1-q_5)(1-q_6)$

R(x)

Π_x

S(x,y)

x	P
a_1	p_1
a_2	p_2
a_3	p_3

x	y	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

optional slide (check yourself!)

Two approaches to query evaluation on tuple-independent probabilistic databases

	Intensional Approach	Extensional Approach
Idea	Find the "provenance/lineage expression" as a Boolean formula (PosBool semiring). Compute the probability of this Boolean formula assuming the variables are independent	Find a "safe query plan" if possible
Specific to	A database and a query	A query (works for all input databases)
Existence	Always exist for a RA query	May or may not exist
Computation	Computation of the formula is in poly-time (data complexity), but computation of the probability may be computationally hard (#P-hard)	If a safe plan exists, computation is in poly-time (data complexity)

Challenges

- No safe plan even for simple queries like Q() :- R(x), S(x, y), T(y)
  - No safe plan for SPJU (RA with union)=> query is computationally hard (seminal result by Dalvi-Suciu)

- Study models and query evaluation (exact and approximate inference) that work well in practice

- Uncertain rules/queries vs. uncertain data
  - e.g. Markov Logic Network (Domingos et al.) or PSL (Getoor et al.)

Crowd Sourcing

Selected/adapted slides from the tutorial by
Profs. Daniel Deutch and Tova Milo, SIGMOD 2011
(optional material: full slide deck is available on Tova's webpage)

CrowdSourcing

- Main idea: Harness the crowd to a "task"
  - Task: solve bugs
  - Task: find an appropriate treatment to an illness
  - Task: construct a database of facts

  …

- Why now?
  - Internet and smart phones …

    We are all connected, all of the time!!!

The classical example

WIKIPEDIA

English
The Free Encyclopedia
3 907 000+ articles

日本語
フリー百科事典
799 000+ 記事

Español
La enciclopedia libre
879 000+ artículos

Deutsch
Die freie Enzyklopädie
1 383 000+ Artikel

Русский
Свободная энциклопедия
838 000+ статей

Français
L'encyclopédie libre
1 230 000+ articles

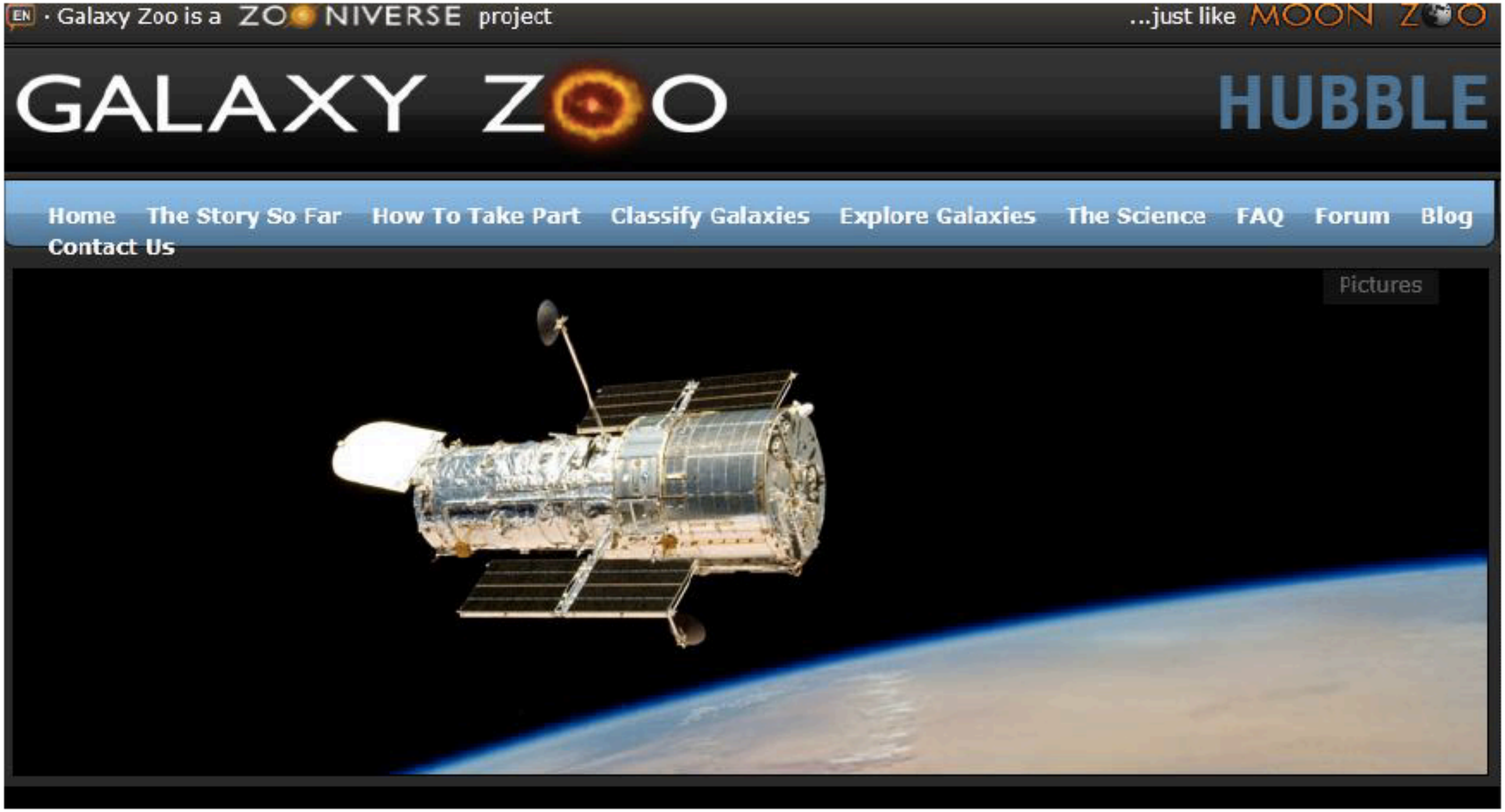
Italiano
L'enciclopedia libera
905 000+ voci

Polski
Wolna encyklopedia
887 000+ haseł

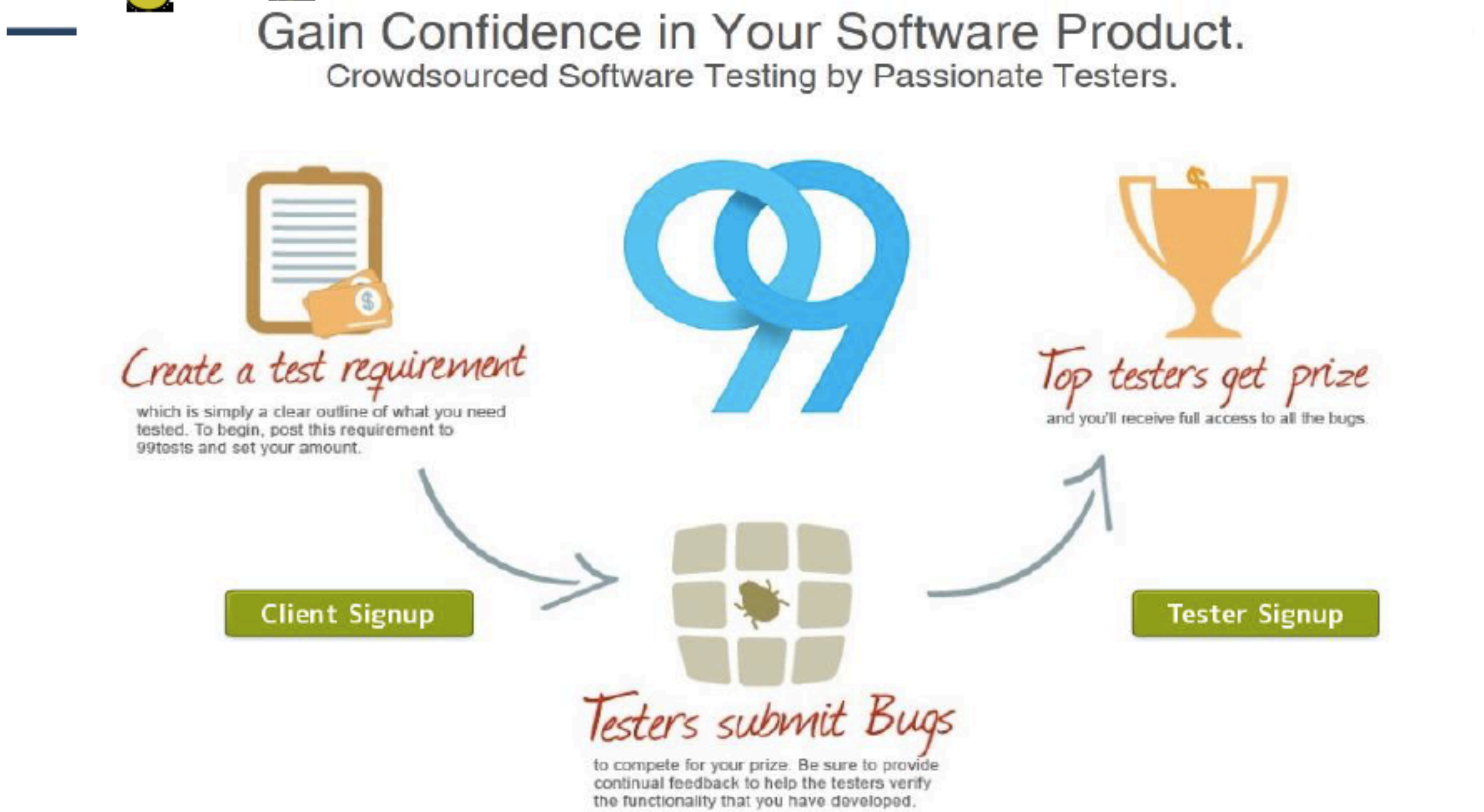
Português
A enciclopédia livre
718 000+ artigos

中文
自由的百科全書
429 000+ 條目

Galaxy Zoo


EN · Galaxy Zoo is a ZOONIVERSE project
...just like MOON ZOO
GALAXY ZOO
HUBBLE
Home The Story So Far How To Take Part Classify Galaxies Explore Galaxies The Science FAQ Forum Blog
Contact Us
Pictures

Collaborative Testing



Slide by Daniel Deutch & Tova Milo, SIGMOD 2011

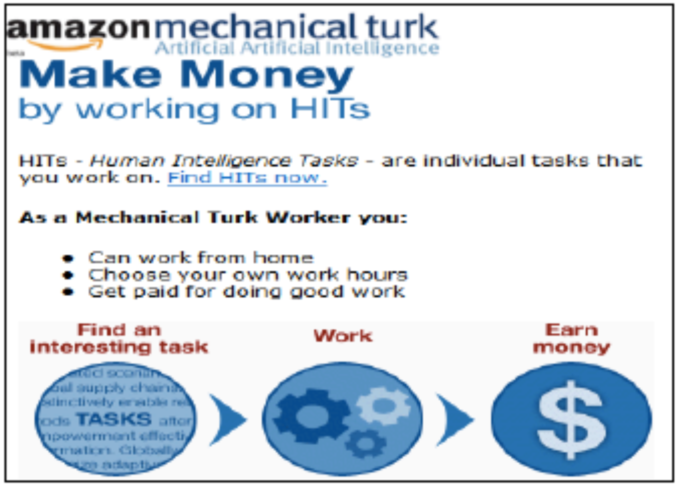
CrowdSourcing: Unifying Principles

- Main goal
  - "Outsourcing" a task to a crowd of users
- Kinds of tasks
  - Tasks that can be performed by a computer, but inefficiently
  - Tasks that can't be performed by a computer
- Challenges
  - How to motivate the crowd?
  - Get data, minimize errors, estimate quality
  - Direct users to contribute where is most needed \ they are experts

Motivating the Crowd

Altruism

Fun



Money

Crowd Data Sourcing

- The case where the task is collection of data
- Two main aspects **[DFKK'12]**:
  - Using the crowd to create better databases
  - Using database technologies to create better crowd datasourcing applications

[DFKK'12]: Crowdsourcing Applications and Platforms: A Data Management Perspective, A.Doan, M. J. Franklin, D. Kossmann, T. Kraska, VLDB 2011

Slide by Daniel Deutch & Tova Milo, SIGMOD 2011

Data-related Tasks (that can be) Performed by Crowds

- Data cleaning
  - E.g. repairing key violations by settling contradictions
- Data Integration
  - E.g. identify mappings
- Data Mining
  - E.g. entity resolution
- Information Extraction

[**Internet- Scale Collection of Human- Reviewed Data ,** Q. Su, D. Pavlov, J. Chow, W.C. Baker, WWW '07]
[**Matching Schemas in Online Communities: A Web 2.0 Approach,** R. McCann, W. Shen, A. Doan, ICDE '08]
[**Amplifying Community Content Creation with Mixed Initiative Information Extraction,** R. Hoffman, S. Amershi, K. Patel, F. Wu., J. Fogarty, D. Weld, CHI '09]

Slide by Daniel Deutch & Tova Milo, SIGMOD 2011

Main Tasks in Crowd Data Sourcing

- What questions to ask?
- How to define correctness of answers?
- How to clean the data?
- Who to ask? how many people?
- How to best use resources?

Declarative Framework!

Probabilistic Data!

Data Cleaning!

Optimizations and Incremental Computation

Platforms for Crowdsourcing

Qurk (MIT)

CrowdDB (Berkeley and ETH Zurich)

CrowdForge (CMU)

Deco (Stanford and UCSC)

MoDaS (Tel Aviv University)

...

[and many more, please forgive us if your project is not listed!]

Conclusions

- All classical issues:

  Data models, query languages, query processing, optimization, HCI
- Database techniques are very useful
  – "Classical" as well as new
- BUT
  - (Very) interactive computation
  - (Very) large scale data
  - (Very) little control on quality/reliability

Many (Research) Challenges

- Not only in databases, but in several other communities: ML, KD, Web, …

- Latency, quality, cost
  - Ask small #questions in small #rounds
  - Ask the right questions
- Efficiency
  - distributed processing
  - incremental processing
- Semantic
  - text/image processing
  - data mining with crowd (model how people think)

Data Integration

(Overview Only: [RG] Chapter 29.2)

Motivation

- As databases grow, users want to access data from more than one sources
  - e.g., compare travel packages from multiple agents/sites
  - e.g. large organizations have several databases created/maintained by different divisions – may have common info – need to determine the relationships between these databases
  - different forms of data – prices in USD/item, USD/dozen-of-items etc.
  - XML data – may not follow the same DTD – legacy databases – semantic mismatches

Approaches to Data Integration

- Semantic mismatches can be resolved and hidden from users by defining views over the two databases
  - Semantic aggregation
  - Challenges due to poor documentation – difficult to understand the meaning and define unifying views

- If the underlying databases are managed using different DBMSs,..
  - some kind of “middleware” must be used to evaluate queries over the integrated views to give the databases a uniform interface (ODBC)
  - alternatively, the integrating views can be materialized and stored in a data warehouse -- queries can be executed over the warehouse data without accessing the source DBMSs at run-time