Data Warehousing Overview
CS245 Notes 11

Hector Garcia-Molina
Stanford University

Outline

- What is a data warehouse?
- Why a warehouse?
- Models & operations
- Implementing a warehouse

What is a Warehouse?

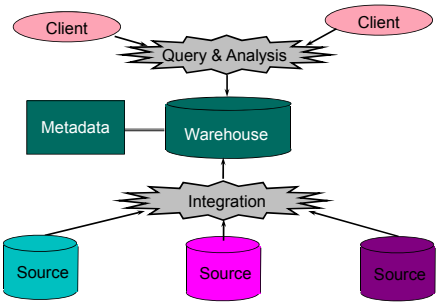
- Collection of diverse data
  - subject oriented
  - aimed at executive, decision maker
  - often a copy of operational data
  - with value-added data (e.g., summaries, history)
  - integrated
  - time-varying
  - non-volatile

more

What is a Warehouse?

- Collection of tools
  - gathering data
  - cleansing, integrating, ...
  - querying, reporting, analysis
  - data mining
  - monitoring, administering warehouse

Warehouse Architecture



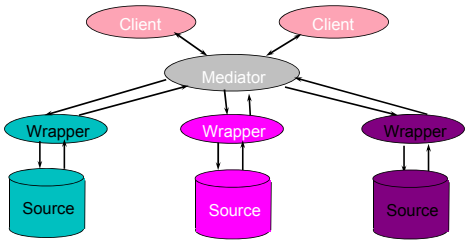
Motivating Examples

- Forecasting
- Comparing performance of units
- Monitoring, detecting fraud
- Visualization

Alternative to Warehousing

- Two Approaches:
  - Query-Driven (Lazy)
  - Warehouse (Eager)

Query-Driven Approach



Advantages of Warehousing

- High query performance
- Queries not visible outside warehouse
- Local processing at sources unaffected
- Can operate when sources unavailable
- Can query data not stored in a DBMS
- Extra information at warehouse
  - Modify, summarize (store aggregates)
  - Add historical information

Advantages of Query-Driven

- No need to copy data
  - less storage
  - no need to purchase data
- More up-to-date data
- Query needs can be unknown
- Only query interface needed at sources
- May be less draining on sources

Warehouse Models & Operators

- Data Models
  - relational
  - cubes
- Operators

Star

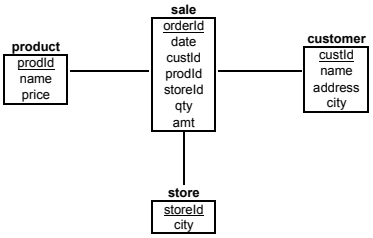
product	prodId	name	price
	p1	bolt	10
	p2	nut	5

store	storeId	city
	c1	nyc
	c2	sfo
	c3	la

sale	oderId	date	custId	prodId	storeId	qty	amt
	o100	1/7/97	53	p1	c1	1	12
	o102	2/7/97	53	p2	c1	2	11
	105	3/8/97	111	p1	c3	5	50

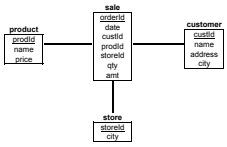
customer	custId	name	address	city
	53	joe	10 main	sfo
	81	fred	12 main	sfo
	111	sally	80 willow	la

Star Schema

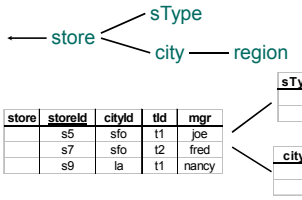


Terms

- Fact table
- Dimension tables
- Measures



Dimension Hierarchies



store	storeId	cityId	tId	mgr
	s5	sfo	t1	joe
	s7	sfo	t2	fred
	s9	la	t1	nancy

sType	tId	size	location
	t1	small	downtown
	t2	large	suburbs

city	cityId	pop	regId
	sfo	1M	north
	la	5M	south

region	regId	name
	north	cold region
	south	warm region

- ➔ snowflake schema
- ➔ constellations

Cube

Fact table view:

sale	prodId	storeId	amt
	p1	c1	12
	p2	c1	11
	p1	c3	50
	p2	c2	8

Multi-dimensional cube:

	c1	c2	c3
p1	12		50
p2	11	8	

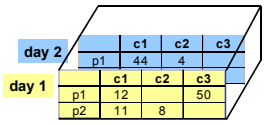
dimensions = 2

3-D Cube

Fact table view:

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

Multi-dimensional cube:



dimensions = 3

Operators

- Traditional
  - selection
  - aggregation
  - ...
- Analysis
  - clean data
  - find trends
  - ...

- Relational
- Cube

Aggregates

- Add up amounts for day 1
- In SQL: SELECT sum(amt) FROM SALE
  WHERE date = 1

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

→ 81

Aggregates

- Add up amounts by day
- In SQL: SELECT date, sum(amt) FROM SALE
  GROUP BY date

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

→

ans	date	sum
	1	81
	2	48

Another Example

- Add up amounts by day, product
- In SQL: SELECT date, sum(amt) FROM SALE
  GROUP BY date, prodId

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

→

sale	prodId	date	amt
	p1	1	62
	p2	1	19
	p1	2	48

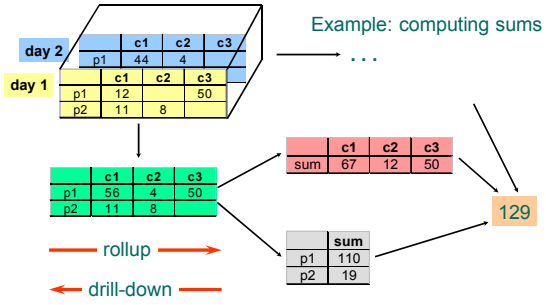
rollup →

← drill-down

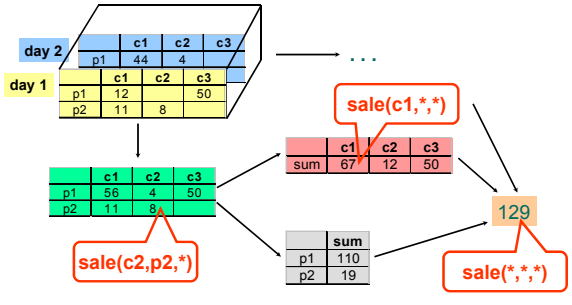
Aggregates

- Operators: sum, count, max, min, median, ave
- "Having" clause
- Using dimension hierarchy
  - average by region (within store)
  - maximum by month (within date)

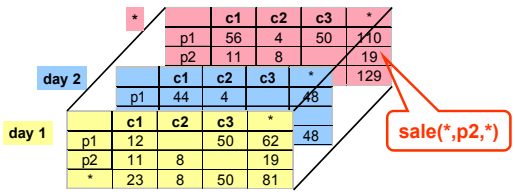
Cube Aggregation



Cube Operators



Extended Cube

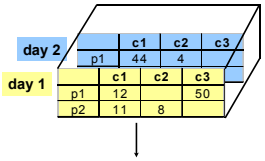


*	c1	c2	c3	*
p1	56	4	50	110
p2	11	8		19
*				129

	c1	c2	c3	*
p1	44	4		48

	c1	c2	c3	*
p1	12		50	62
p2	11	8		19
*	23	8	50	81

Aggregation Using Hierarchies



	c1	c2	c3
p1	44	4	

	c1	c2	c3
p1	12		50
p2	11	8	

customer
|
region
|
country

	region A	region B
p1	56	54
p2	11	8

(customer c1 in Region A;
customers c2, c3 in Region B)

Data Analysis

- Decision Trees
- Clustering
- Association Rules

Decision Trees

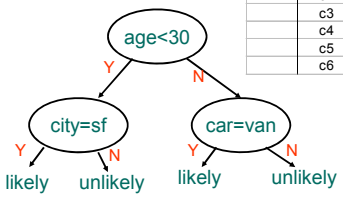
Example:
- Conducted survey to see what customers were interested in new model car
- Want to select customers for advertising campaign

sale	custId	car	age	city	newCar
	c1	taurus	27	sf	yes
	c2	van	35	la	yes
	c3	van	40	sf	yes
	c4	taurus	22	sf	yes
	c5	merc	50	la	no
	c6	taurus	25	la	no

training set

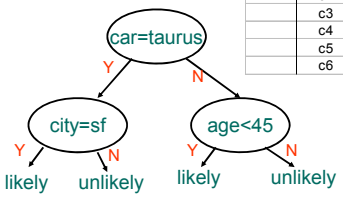
One Possibility

sale	custId	car	age	city	newCar
	c1	taurus	27	sf	yes
	c2	van	35	la	yes
	c3	van	40	sf	yes
	c4	taurus	22	sf	yes
	c5	merc	50	la	no
	c6	taurus	25	la	no



Another Possibility

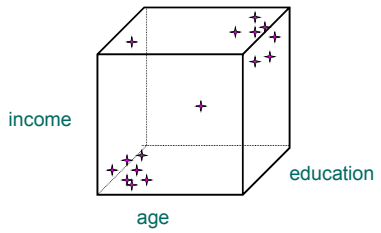
sale	custId	car	age	city	newCar
	c1	taurus	27	sf	yes
	c2	van	35	la	yes
	c3	van	40	sf	yes
	c4	taurus	22	sf	yes
	c5	merc	50	la	no
	c6	taurus	25	la	no



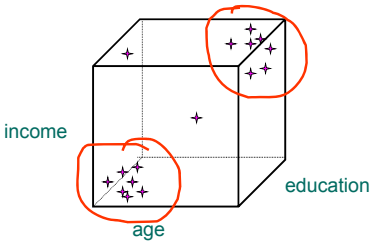
Issues

- Decision tree cannot be "too deep"
  - would not have statistically significant amounts of data for lower decisions
- Need to select tree that most reliably predicts outcomes

Clustering

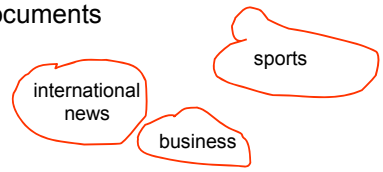


Clustering



Another Example: Text

- Each document is a vector
  - e.g., <100110...> contains words 1,4,5,...
- Clusters contain "similar" documents
- Useful for understanding, searching documents



Issues

- Given desired number of clusters?
- Finding "best" clusters
- Are clusters semantically meaningful?
  - e.g., "yuppies'' cluster?
- Using clusters for disk storage

Association Rule Mining

sales records:

transaction id	customer id	products bought
tran1	cust33	p2, p5, p8
tran2	cust45	p5, p8, p11
tran3	cust12	p1, p9
tran4	cust40	p5, p8, p11
tran5	cust12	p2, p9
tran6	cust12	p9

market-basket data

- Trend: Products p5, p8 often bough together
- Trend: Customer 12 likes product p9

Association Rule

- Rule: $\{p_1, p_3, p_8\}$
- Support: number of baskets where these products appear
- High-support set: support ≥ threshold s
- Problem: find all high support sets

Implementation Issues

- ETL (Extraction, transformation, loading)
  - Getting data to the warehouse
  - Entity Resolution
- What to materialize?
- Efficient Analysis
  - Association rule mining
  - ...

ETL: Monitoring Techniques

- Periodic snapshots
- Database triggers
- Log shipping
- Data shipping (replication service)
- Transaction shipping
- Polling (queries to source)
- Screen scraping
- Application level monitoring

↑ Advantages & Disadvantages!

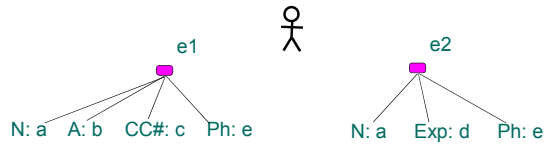
ETL: Data Cleaning

- Migration (e.g., yen ⇨ dollars)
- Scrubbing: use domain-specific knowledge (e.g., social security numbers)
- Fusion (e.g., mail list, customer merging)

billing DB → customer1(Joe) ↘
service DB → customer2(Joe) ↗ merged_customer(Joe)

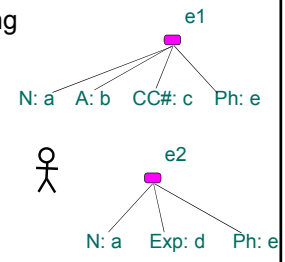
- Auditing: discover rules & relationships (like data mining)

More details: Entity Resolution



Applications

- comparison shopping
- mailing lists
- classified ads
- customer files
- counter-terrorism



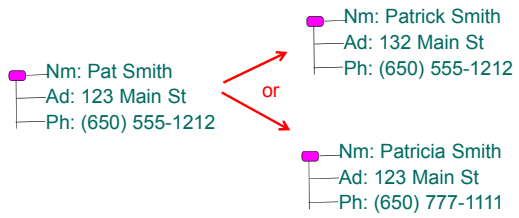
Why is ER Challenging?

- Huge data sets
- No unique identifiers
- Lots of uncertainty
- Many ways to skin the cat

43

Taxonomy: Pairwise vs Global

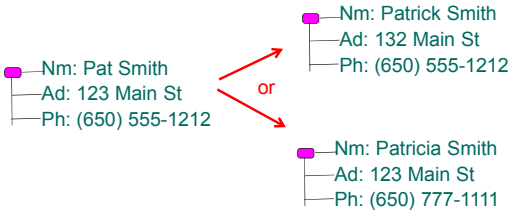
- Decide if *r, s* match only by looking at *r, s*?
- Or need to consider more (all) records?



44

Taxonomy: Pairwise vs Global

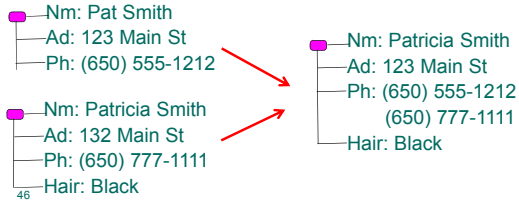
- Global matching complicates things a lot!
  - e.g., change decision as new records arrive



45

Taxonomy: Outcome

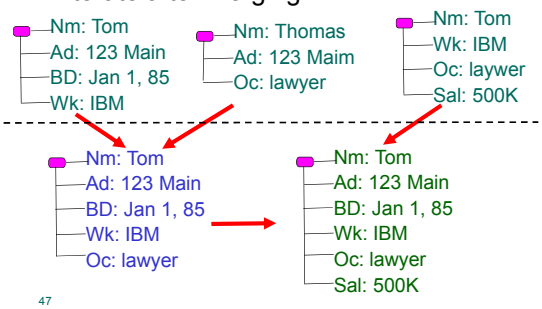
- Partition of records
  - e.g., comparison shopping
- Merged records



46

Taxonomy: Outcome

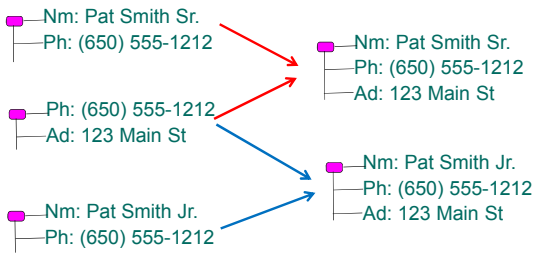
- Iterate after merging



47

Taxonomy: Record Reuse

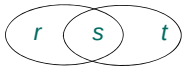
- One record related to multiple entities?



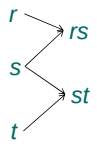
48

Taxonomy: Record Reuse

- Partitions



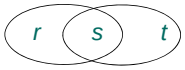
- Merges



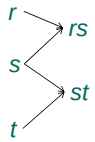
49

Taxonomy: Record Reuse

- Partitions



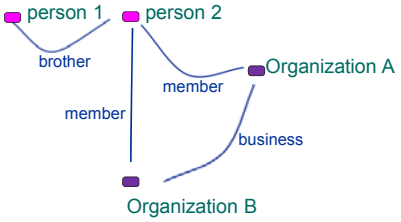
- Merges



- Record reuse ➡ complex and expensive!

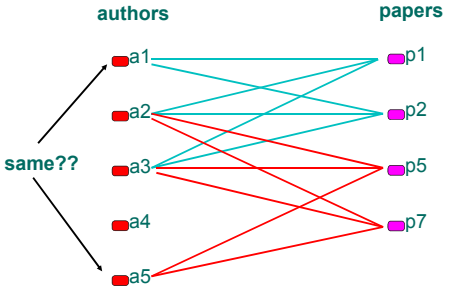
50

Taxonomy: Multiple Entity Types



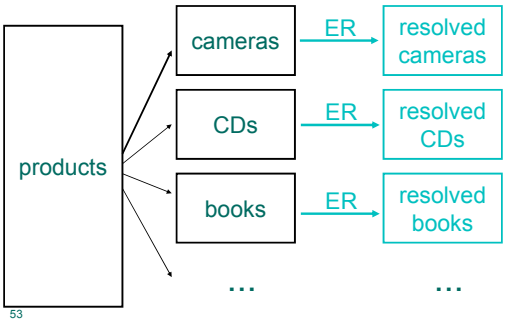
51

Taxonomy: Multiple Entity Types



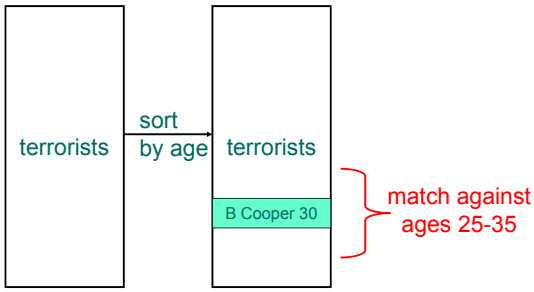
52

Taxonomy: Exact vs Approximate



53

Taxonomy: Exact vs Approximate



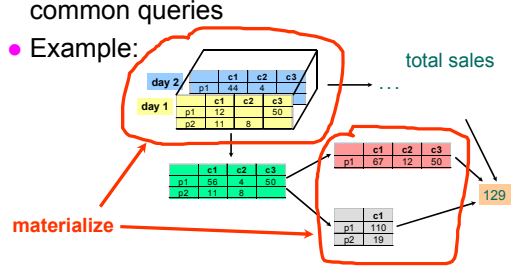
54

Implementation Issues

- ETL (Extraction, transformation, loading)
  - Getting data to the warehouse
  - Entity Resolution
- What to materialize?
- Efficient Analysis
  - Association rule mining
  - ...

What to Materialize?

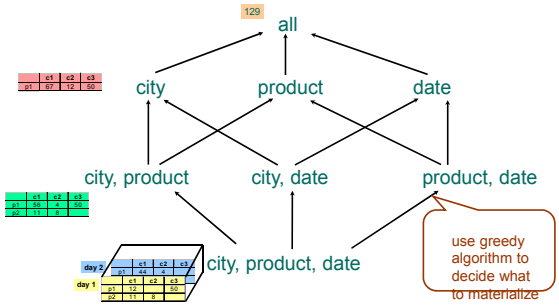
- Store in warehouse results useful for common queries
- Example:



Materialization Factors

- Type/frequency of queries
- Query response time
- Storage cost
- Update cost

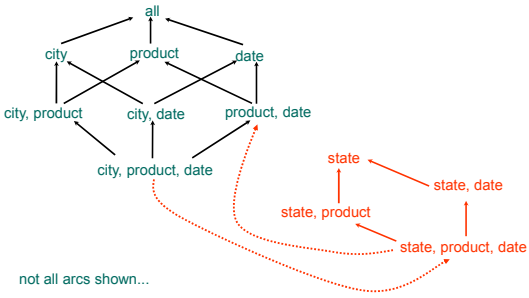
Cube Aggregates Lattice



Dimension Hierarchies

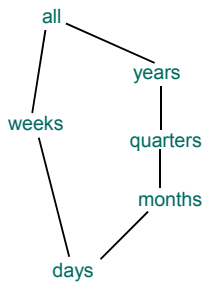
cities	city	state
	c1	CA
	c2	NY

Dimension Hierarchies



not all arcs shown...

Interesting Hierarchy



time	day	week	month	quarter	year
	1	1	1	1	2000
	2	1	1	1	2000
	3	1	1	1	2000
	4	1	1	1	2000
	5	1	1	1	2000
	6	1	1	1	2000
	7	1	1	1	2000
	8	2	1	1	2000

conceptual dimension table

Implementation Issues

- ETL (Extraction, transformation, loading)
  - Getting data to the warehouse
  - Entity Resolution
- What to materialize?
- Efficient Analysis
  - Association rule mining
  - ...

Finding High-Support Pairs

- Baskets(basket, item)
- SELECT I.item, J.item, COUNT(I.basket)
  FROM Baskets I, Baskets J
  WHERE I.basket = J.basket AND
  I.item < J.item
  GROUP BY I.item, J.item
  HAVING COUNT(I.basket) >= s;

Finding High-Support Pairs

- Baskets(basket, item)
- SELECT I.item, J.item, COUNT(I.basket)
  FROM Baskets I, Baskets J
  WHERE I.basket = J.basket AND
  I.item < J.item
  GROUP BY I.item, J.item
  HAVING COUNT(I.basket) >= s;

WHY?

Example

basket	item
t1	p2
t1	p5
t1	p8
t2	p5
t2	p8
t2	p11
...	...

basket	item1	item2
t1	p2	p5
t1	p2	p8
t1	p5	p8
t2	p5	p8
t2	p5	p11
t2	p8	p11
...	...	...

Example

basket	item
t1	p2
t1	p5
t1	p8
t2	p5
t2	p8
t2	p11
...	...

basket	item1	item2
t1	p2	p5
t1	p2	p8
t1	p5	p8
t2	p5	p8
t2	p5	p11
t2	p8	p11
...	...	...

check if count ≥ s

Issues

- Performance for size 2 rules

big

basket	item
t1	p2
t1	p5
t1	p8
t2	p5
t2	p8
t2	p11
...	...

basket	item1	item2
t1	p2	p5
t1	p2	p8
t1	p5	p8
t2	p5	p8
t2	p5	p11
t2	p8	p11
...	...	...

even bigger!

- Performance for size *k* rules

Association Rules

- How do we perform rule mining efficiently?

Association Rules

- How do we perform rule mining efficiently?
- Observation: If set *X* has support *t*, then each *X* subset must have at least support *t*

Association Rules

- How do we perform rule mining efficiently?
- Observation: If set *X* has support *t*, then each *X* subset must have at least support *t*
- For 2-sets:
  - if we need support *s* for {*i, j*}
  - then each *i, j* must appear in at least *s* baskets

Algorithm for 2-Sets

(1) Find OK products
- those appearing in *s* or more baskets

(2) Find high-support pairs
using only OK products

Algorithm for 2-Sets

- INSERT INTO okBaskets(basket, item)
  SELECT basket, item
  FROM Baskets
  GROUP BY item
  HAVING COUNT(basket) >= s;

Algorithm for 2-Sets

- INSERT INTO okBaskets(basket, item)
  SELECT basket, item
  FROM Baskets
  GROUP BY item
  HAVING COUNT(basket) >= s;
- Perform mining on okBaskets

```
SELECT I.item, J.item, COUNT(I.basket)
FROM okBaskets I, okBaskets J
WHERE I.basket = J.basket  AND
          I.item < J.item
GROUP BY I.item, J.item
HAVING COUNT(I.basket) >= s;
```


Counting Efficiently

- One way:

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

Counting Efficiently

- One way:

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

sort →

basket	I.item	J.item
t3	p2	p3
t3	p2	p8
t1	p5	p8
t2	p5	p8
t3	p5	p8
t2	p8	p11
...	...	...

Counting Efficiently

- One way:

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

sort →

basket	I.item	J.item
t3	p2	p3
t3	p2	p8
t1	p5	p8
t2	p5	p8
t3	p5	p8
t2	p8	p11
...	...	...

count & remove →

count	I.item	J.item
3	p5	p8
5	p12	p18
...	...	...

Counting Efficiently

- Another way:

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

Counting Efficiently

- Another way:

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

scan & count →

count	I.item	J.item
1	p2	p3
2	p2	p8
3	p5	p8
5	p12	p18
1	p21	p22
2	p21	p23
...	...	...

keep counter array in memory

Counting Efficiently

- Another way:

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

scan & count →

count	I.item	J.item
1	p2	p3
2	p2	p8
3	p5	p8
5	p12	p18
1	p21	p22
2	p21	p23
...	...	...

remove →

count	I.item	J.item
3	p5	p8
5	p12	p18
...	...	...

keep counter array in memory

Counting Efficiently

- Another way:

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

scan & count →

count	I.item	J.item
1	p2	p3
2	p2	p8
3	p5	p8
5	p12	p18
1	p21	p22
2	p21	p23
...	...	...

remove →

count	I.item	J.item
3	p5	p8
5	p12	p18
...	...	...

~~keep counter array in memory~~

Yet Another Way

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

(1) scan & hash & count →

count	bucket
1	A
5	B
2	C
1	D
8	E
1	F
...	...

in-memory hash table

Yet Another Way

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

(1) scan & hash & count →

count	bucket
1	A
5	B
2	C
1	D
8	E
1	F
...	...

in-memory hash table

(2) scan & remove →

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p5	p8
t5	p12	p18
t8	p12	p18
...	...	...

Yet Another Way

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

(1) scan & hash & count →

count	bucket
1	A
5	B
2	C
1	D
8	E
1	F
...	...

in-memory hash table

(2) scan & remove →

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p5	p8
t5	p12	p18
t8	p12	p18
...	...	...

false positive

Yet Another Way

threshold = 3

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

(1) scan & hash & count →

count	bucket
1	A
5	B
2	C
1	D
8	E
1	F
...	...

in-memory hash table

in-memory counters

(2) scan & remove →

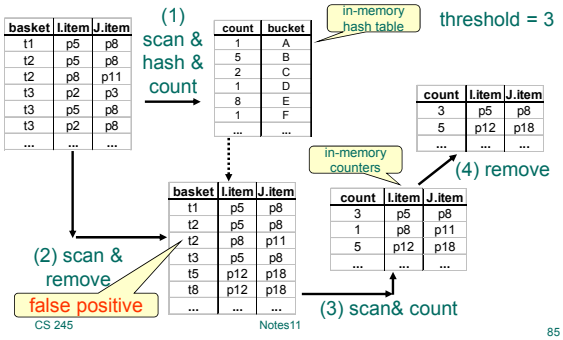
basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p5	p8
t5	p12	p18
t8	p12	p18
...	...	...

false positive

(3) scan& count →

count	I.item	J.item
3	p5	p8
1	p8	p11
5	p12	p18
...	...	...

Yet Another Way



basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p2	p3
t3	p5	p8
t3	p2	p8
...	...	...

count	bucket
1	A
5	B
2	C
1	D
8	E
1	F
...	...

basket	I.item	J.item
t1	p5	p8
t2	p5	p8
t2	p8	p11
t3	p5	p8
t5	p12	p18
t8	p12	p18
...	...	...

count	I.item	J.item
3	p5	p8
1	p8	p11
5	p12	p18
...	...	...

count	I.item	J.item
3	p5	p8
5	p12	p18
...	...	...

Discussion

- Hashing scheme: 2 (or 3) scans of data
- Sorting scheme: requires a sort!
- Hashing works well if few high-support pairs and many low-support ones

Discussion

- Hashing scheme: 2 (or 3) scans of data
- Sorting scheme: requires a sort!
- Hashing works well if few high-support pairs and many low-support ones

Implementation Issues

- ETL (Extraction, transformation, loading)
  - Getting data to the warehouse
  - Entity Resolution
- What to materialize?
- Efficient Analysis
  - Association rule mining
  - ...

Extra: Data Mining in the InfoLab

Recommendations in CourseRank

quarters

user	q1	q2	q3	q4
u1	a: 5	b: 5	d: 5	
u2	a: 1	e: 2	d: 4	f: 3
u3	g: 4	h: 2	e: 3	f: 3
u4	b: 2	g: 4	h: 4	e: 4
u	a: 5	g: 4	e: 4	

Extra: Data Mining in the InfoLab

Recommendations in CourseRank

quarters

user	q1	q2	q3	q4
u1	a: 5	b: 5	d: 5	
u2	a: 1	e: 2	d: 4	f: 3
u3	g: 4	h: 2	e: 3	f: 3
u4	b: 2	g: 4	h: 4	e: 4
u	a: 5	g: 4	e: 4	

u3 and u4 are similar to u

Recommend h

Extra: Data Mining in the InfoLab

Recommendations in CourseRank

quarters

user	q1	q2	q3	q4
u1	a: 5	b: 5	d: 5	
u2	a: 1	e: 2	d: 4	f: 3
u3	g: 4	h: 2	e: 3	f: 3
u4	b: 2	g: 4	h: 4	e: 4
u	a: 5	g: 4	e: 4	

Recommend d (and f, h)

Sequence Mining

- Given a set of transcripts, use *Pr[x|a]* to predict if *x* is a good recommendation given user has taken *a*.
- Two issues...

Pr[x|a] Not Quite Right

transcript	containing
1	-
2	a
3	x
4	a -> x
5	x -> a

target user's transcript:
[... a .... || unknown]

recommend x?

$Pr[x|a] = 2/3$

$Pr[x|a^{\sim x}] = 1/2$

User Has Taken >= 1 Course

- User has taken T= {a, b, c}
- Need $Pr[x|T^{\sim x}]$
- Approximate as $Pr[x|a^{\sim x} \wedge b^{\sim x} \wedge c^{\sim x}]$
- Expensive to compute, so...

CourseRank User Study