

Data Warehousing and Data Mining

CompSci 316
Introduction to Database Systems

Announcements (Tue. Oct. 9) ²

- ❖ Midterm on Thursday, in class
 - Open-book, open-notes
 - No communication devices
 - Solution to sample midterm was emailed this weekend
 - Will cover all materials through today
 - But more focus will be on parts that you already “exercised”
- ❖ Project milestone #1 due next Thursday

Data integration ³

- ❖ Data resides in many distributed, heterogeneous OLTP (On-Line Transaction Processing) sources
 - Sales, inventory, customer, ...
 - NC branch, NY branch, CA branch, ...
- ❖ Need to support OLAP (On-Line Analytical Processing) over an integrated view of the data
- ❖ Possible approaches to integration
 - Eager: integrate in advance and store the integrated data at a central repository called the data warehouse
 - Lazy: integrate on demand; process queries over distributed sources—mediated or federated systems

OLTP versus OLAP

4

OLTP

- ❖ Mostly updates
- ❖ Short, simple transactions
- ❖ Clerical users
- ❖ Goal: transaction throughput

OLAP

- ❖ Mostly reads
- ❖ Long, complex queries
- ❖ Analysts, decision makers
- ❖ Goal: fast queries

Implications on database design and optimization?

OLAP databases do not care much about redundancy

- “Denormalize” tables
- Many, many indexes
- Precomputed query results

Eager versus lazy integration

5

Eager (warehousing)

- ❖ In advance: before queries
- ❖ Copy data from sources
- ☞ Answer could be stale
- ☞ Need to maintain consistency
- ☞ Query processing is local to the warehouse
 - Faster
 - Can operate when sources are unavailable

Lazy

- ❖ On demand: at query time
- ❖ Leave data at sources
- ☞ Answer is more up-to-date
- ☞ No need to maintain consistency
- ☞ Sources participate in query processing
 - Slower
 - Interferes with local processing

Maintaining a data warehouse

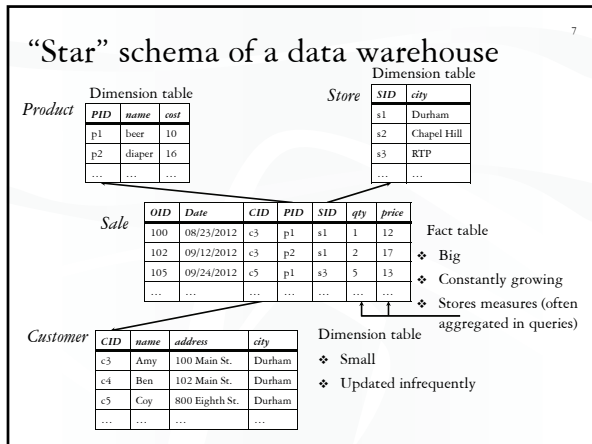
6

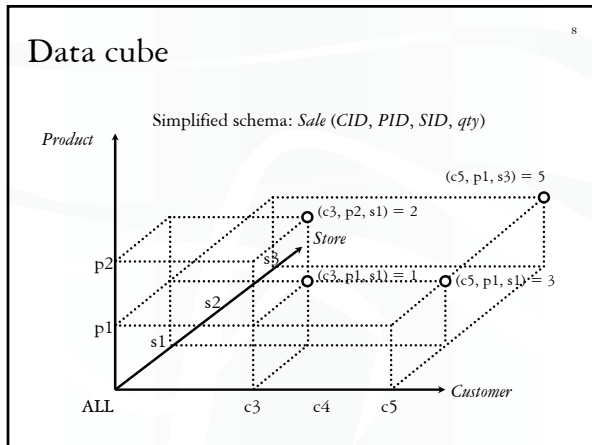
❖ The “ETL” process

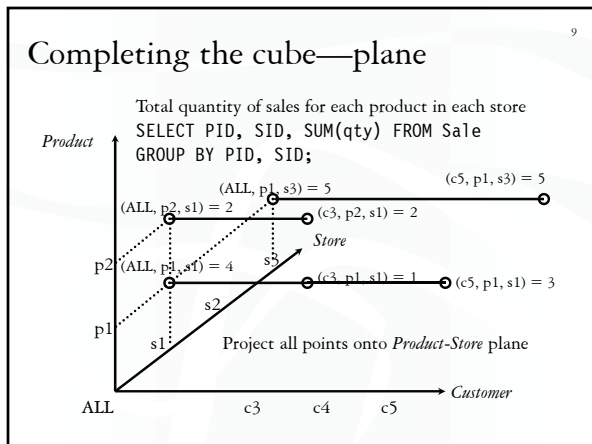
- Extraction: extract relevant data and/or changes from sources
- Transformation: transform data to match the warehouse schema
- Loading: integrate data/changes into the warehouse

❖ Approaches

- Recomputation
 - Easy to implement; just take periodic dumps of the sources, say, every night
 - What if there is no “night,” e.g., a global organization?
 - What if recomputation takes more than a day?
- Incremental maintenance
 - Compute and apply only incremental changes
 - Fast if changes are small
 - Not easy to do for complicated transformations
 - Need to detect incremental changes at the sources

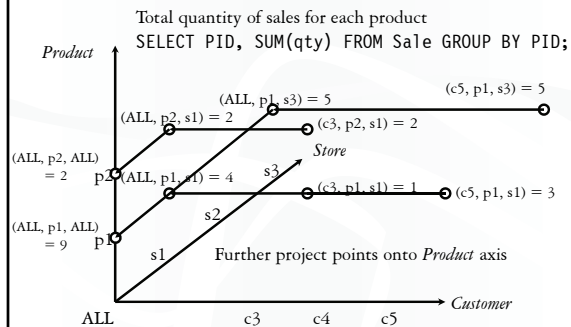






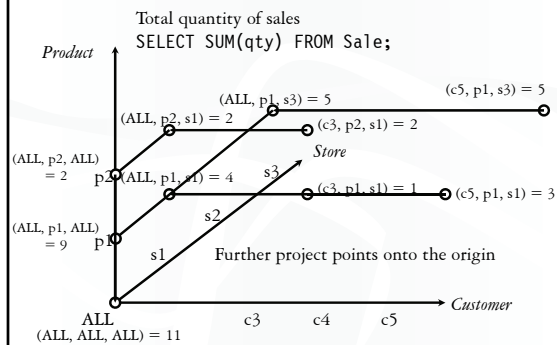
Completing the cube—axis

10



Completing the cube—origin

11



CUBE operator

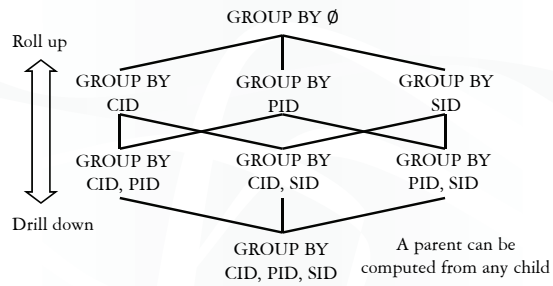
12

- ❖ *Sale* (*CID*, *PID*, *SID*, *qty*)
- ❖ Proposed SQL extension:
 SELECT SUM(*qty*) FROM *Sale*
 GROUP BY CUBE *CID*, *PID*, *SID*;
- ❖ Output contains:
 - Normal groups produced by GROUP BY
 - (*c1*, *p1*, *s1*, sum), (*c1*, *p2*, *s3*, sum), etc.
 - Groups with one or more ALL's
 - (ALL, *p1*, *s1*, sum), (*c2*, ALL, ALL, sum), (ALL, ALL, ALL, sum), etc.
- ❖ Can you write a CUBE query using only GROUP BY's?

Gray et al., "Data Cube: A Relational Aggregation Operator
 Generalizing Group-By, Cross-Tab, and Sub-Total." *ICDE* 1996

Aggregation lattice

13



Materialized views

14

- ❖ Computing GROUP BY and CUBE aggregates is expensive
- ❖ OLAP queries perform these operations over and over again
- ☞ Idea: precompute and store the aggregates as materialized views
 - Maintained automatically as base data changes
 - No. 1 user-requested feature in PostgreSQL!

Selecting views to materialize

15

- ❖ Factors in deciding what to materialize
 - What is its storage cost?
 - What is its update cost?
 - Which queries can benefit from it?
 - How much can a query benefit from it?
- ❖ Example
 - GROUP BY Ø is small, but not useful to most queries
 - GROUP BY CID, PID, SID is useful to any query, but too large to be beneficial

Harinarayan et al., "Implementing Data Cubes Efficiently." *SIGMOD* 1996

Other OLAP extensions

16

- ❖ Besides extended grouping capabilities (e.g., CUBE), window operations have also been added to SQL
- ❖ A “window” specifies an ordered list of rows related to the “current row”
- ❖ A window function computes a value from this list and the “current row”
 - Standard aggregates: COUNT, SUM, AVG, MIN, MAX
 - New functions: RANK, PERCENT_RANK, LAG, LEAD, ...

RANK window function example

17

sid	pid	cid	qty
Durham	beer	Alice	10
Durham	beer	Bob	2
Durham	chips	Bob	3
Durham	diaper	Alice	5
Raleigh	beer	Alice	2
Raleigh	diaper	Bob	100

↓ GROUP BY

sid	pid	cid	qty
Durham	beer	Alice	10
Durham	beer	Bob	2
Durham	chips	Bob	3
Durham	diaper	Alice	5
Raleigh	beer	Alice	2
Raleigh	diaper	Bob	100

```
SELECT SID, PID, SUM(qty),
       RANK() OVER w
FROM Sale GROUP BY SID, PID
WINDOW w AS
(PARTITION BY SID
 ORDER BY SUM(qty) DESC);
```

Apply WINDOW after processing FROM, WHERE, GROUP BY, HAVING

- ❖ PARTITION defines the related set and ORDER BY orders it

E.g., for the following “row,” the related list is:

Durham	beer	Alice	10
Durham	beer	Bob	2

Durham	beer	Alice	10
Durham	beer	Bob	2

Durham	diaper	Alice	5
Durham	chips	Bob	3

RANK window function example (cont'd)

18

sid	pid	cid	qty
Durham	beer	Alice	10
Durham	beer	Bob	2
Durham	chips	Bob	3
Durham	diaper	Alice	5
Raleigh	beer	Alice	2
Raleigh	diaper	Bob	100

```
SELECT SID, PID, SUM(qty),
       RANK() OVER w
FROM Sale GROUP BY SID, PID
WINDOW w AS
(PARTITION BY SID
 ORDER BY SUM(qty) DESC);
```

E.g., for the following “row,” the related list is:

Durham	beer	Alice	10
Durham	beer	Bob	2

Durham	beer	Alice	10
Durham	beer	Bob	2

Durham	diaper	Alice	5
Durham	chips	Bob	3

Then, for each “row” and its related list, evaluate SELECT and return:

sid	pid	sum	rank
Durham	beer	12	1
Durham	diaper	5	2
Durham	chips	3	3
Raleigh	diaper	100	1
Raleigh	beer	2	2

First try

22

- ❖ A naïve algorithm
 - Keep a running count for each possible itemset
 - For each transaction T , and for each itemset X , if T contains X then increment the count for X
 - Return itemsets with large enough counts
- ❖ Problem: The number of itemsets is huge!
 - 2^n , where n is the number of items
- ❖ Think: How do we prune the search space?

The Apriori property

23

- ❖ All subsets of a frequent itemset must also be frequent
 - Because any transaction that contains X must also contain subsets of X
- ☞ If we have already verified that X is infrequent, there is no need to count X 's supersets because they must be infrequent too

The Apriori algorithm

24

Multiple passes over the transactions

- ❖ Pass k finds all frequent k -itemsets (i.e., itemsets of size k)
- ❖ Use the set of frequent k -itemsets found in pass k to construct candidate $(k + 1)$ -itemsets to be counted in pass $(k + 1)$
 - A $(k + 1)$ -itemset is a candidate only if all its subsets of size k are frequent

Example: pass 1

25

TID	items
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

Transactions
 $S_{min}\% = 20\%$

itemset	count
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

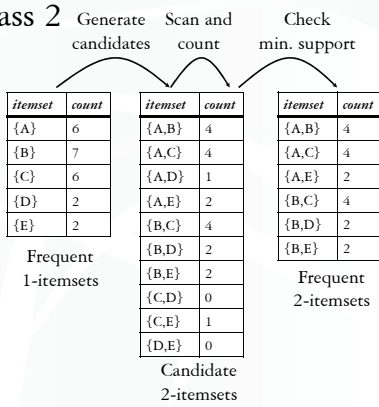
Frequent 1-itemsets
 (Itemset {F} is infrequent)

Example: pass 2

26

TID	items
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

Transactions
 $S_{min}\% = 20\%$

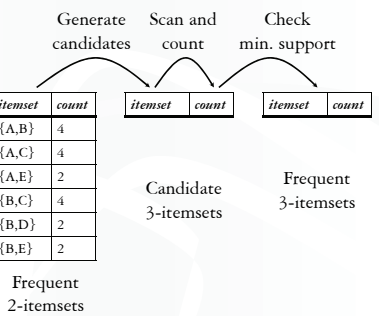


Example: pass 3

27

TID	items
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

Transactions
 $S_{min}\% = 20\%$

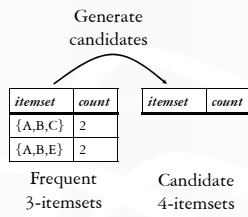


Example: pass 4

28

TID	items
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

Transactions
 $s_{min}\% = 20\%$



Example: final answer

29

itemset	count
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

Frequent 1-itemsets

itemset	count
{A,B}	4
{A,C}	4
{A,E}	2
{B,C}	4
{B,D}	2
{B,E}	2

Frequent 2-itemsets

itemset	count
{A,B,C}	2
{A,B,E}	2

Frequent 3-itemsets

Summary

30

- ❖ Data warehousing
 - Eagerly integrate data from operational sources and store a redundant copy to support OLAP
 - OLAP vs. OLTP: different workload → different degree of redundancy
 - SQL extensions: grouping and windowing
- ❖ Data mining
 - Only covered frequent itemset counting
 - Skipped many other techniques (clustering, classification, regression, etc.)
 - One key difference from statistics and machine learning: massive datasets and I/O-efficient algorithms
