

# Data Warehousing and Data Mining

CPS 116  
Introduction to Database Systems

---

---

---

---

---

---

---

---

## Announcements (November 25)

- ❖ Homework #3 graded
  - Pick them up from Ying during her office hours
- ❖ Homework #4 due today
  - Sample solution available next Tuesday
- ❖ Course project demo period: December 8-13
- ❖ Final exam next Saturday, Dec. 13, 7-10pm
  - Again, open book, open notes
  - Focus on the second half of the course
  - Sample final next Tuesday
  - Sample final solution available Thursday

---

---

---

---

---

---

---

---

## Data integration

- ❖ 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: ACID, transaction throughput

### OLAP

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

Implications on database design and optimization?

---

---

---

---

---

---

---

---

## 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
- 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

---

---

---

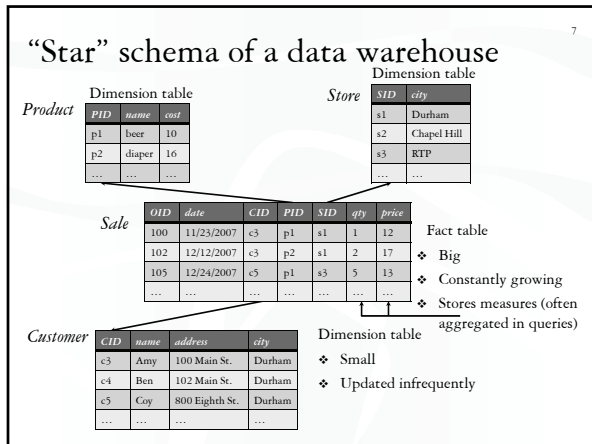
---

---

---

---

---




---

---

---

---

---

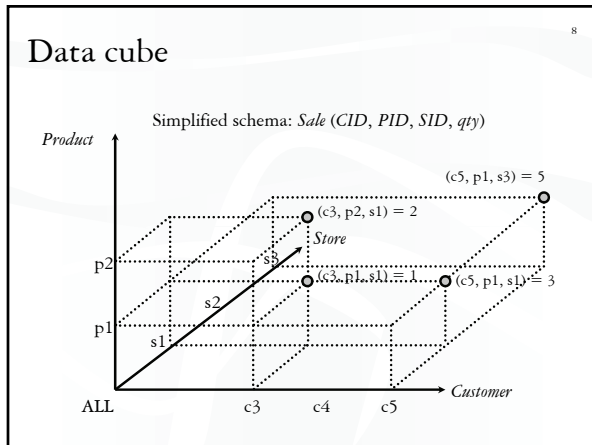
---

---

---

---

---




---

---

---

---

---

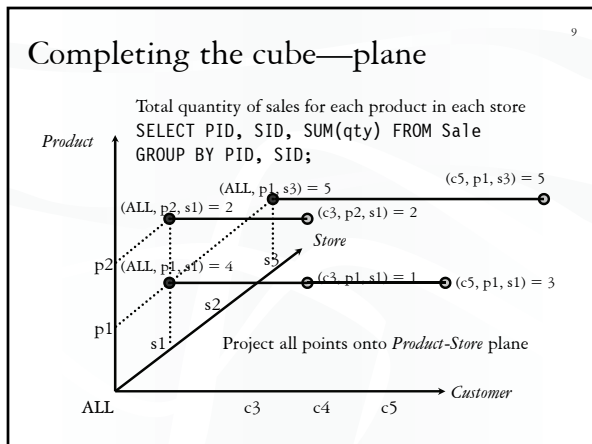
---

---

---

---

---




---

---

---

---

---

---

---

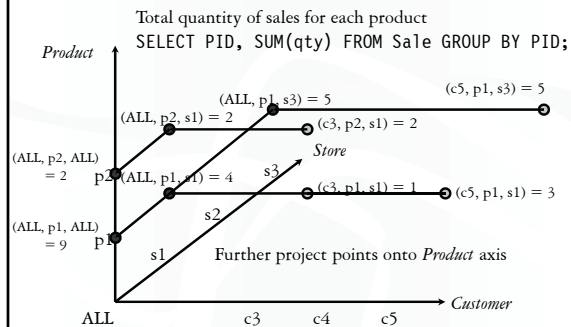
---

---

---

## Completing the cube—axis

10




---

---

---

---

---

---

---

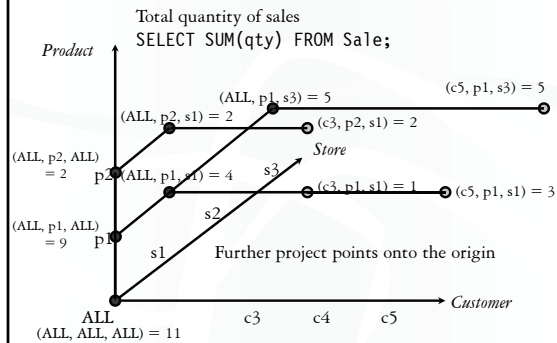
---

---

---

## 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

---

---

---

---

---

---

---

---

---

---

## Automatic summary tables

13

- ❖ Computing GROUP BY and CUBE aggregates is expensive
- ❖ OLAP queries perform these operations over and over again
- ☞ Idea: precompute and store the aggregates as automatic summary tables (a DB2 term)
  - Maintained automatically as base data changes
  - Same as materialized views

---

---

---

---

---

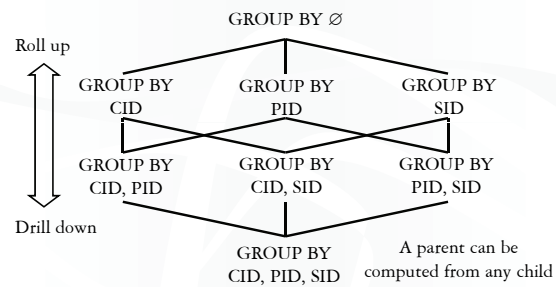
---

---

---

## Aggregation view lattice

14



---

---

---

---

---

---

---

---

## 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 ∅ 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

---

---

---

---

---

---

---

---

## Data mining

16

- ❖ Data → knowledge
- ❖ DBMS meets AI and statistics
- ❖ Clustering, prediction (classification and regression), association analysis, outlier analysis, evolution analysis, etc.
  - Usually complex statistical “queries” that are difficult to answer → often specialized algorithms outside DBMS
- ❖ We will focus on frequent itemset mining

---

---

---

---

---

---

---

---

## Mining frequent itemsets

17

- ❖ Given: a large database of transactions, each containing a set of items

- Example: market baskets

- ❖ Find all frequent itemsets

- A set of items  $X$  is frequent if no less than  $s_{\min}$  % of all transactions contain  $X$
- Examples: {diaper, beer}, {scanner, color printer}

TID	items
T001	diaper, milk, candy
T002	milk, egg
T003	milk, beer
T004	diaper, milk, egg
T005	diaper, beer
T006	milk, beer
T007	diaper, beer
T008	diaper, milk, beer, candy
T009	diaper, milk, beer
...	...

---

---

---

---

---

---

---

---

## First try

18

- ❖ 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:
- ❖ Think: How do we prune the search space?

---

---

---

---

---

---

---

---

## The Apriori property

19

- ❖ 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

20

Multiple passes over the transactions

- ❖ Pass  $k$  finds all frequent  $k$ -itemsets (itemset 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

21

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)

---

---

---

---

---

---

---

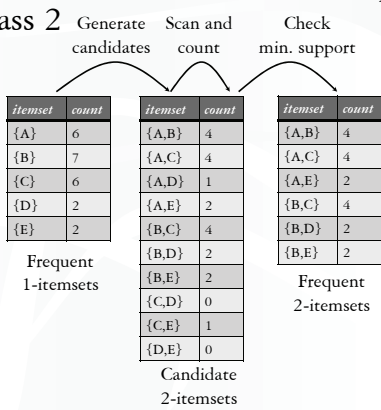
---

### Example: pass 2

22

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




---

---

---

---

---

---

---

---

---

---

---

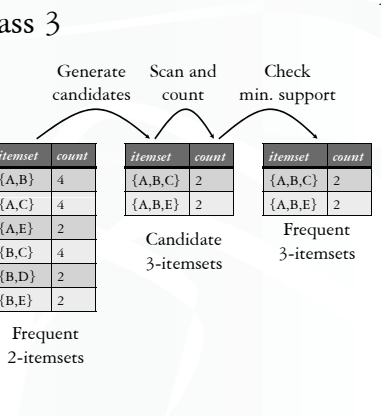
---

### Example: pass 3

23

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




---

---

---

---

---

---

---

---

---

---

---

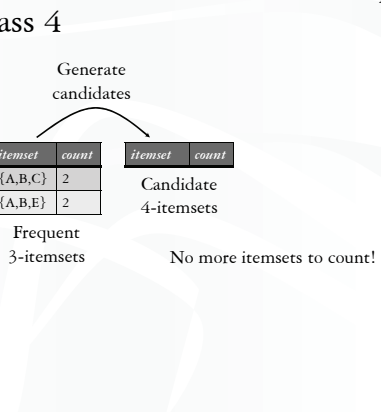
---

### Example: pass 4

24

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

25

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

26

- ❖ 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
- ❖ 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

---

---

---

---

---

---

---

---