CompSci 516
Database Systems

Lecture 23
Data Cube
and
Data Mining

Instructor: Sudeepa Roy

Announcements

- HW3 due on Nov 30 (Fri), 5 pm
- Final report due on Dec 12, Wed
  - See feedback on piazza
- Please fill out the course evaluations!
  - We need to hear from each of you
  - We need your feedback/suggestions to improve this class

Announcements

- Class presentation next lecture (Thurs, Nov 29)
  - In the order of your group number
  - 4 min presentation (will be timed!)

- Presentation
  - 14 projects in 75 mins – 4 mins per project!
  - Not everyone has to present (up to you)
    - everyone in a group gets the same grade
  - You present the current status of the project
    - problem, example, your approach, what you plan
  - Best to show plots/ screenshots/ results/ demo!
  - Try to show the most interesting observation/findings in 4 mins!
  - Tell us what you want to do before you submit the final report (if anything)

Data Warehousing

Reading Material

- [RG]
  - Chapter 25

- Gray-Chaudhuri-Bosworth-Layman-Reichart-Venkatrao-Pellow-Pirahesh, ICDE 1996 *"Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals"*

- Harinarayan-Rajaraman-Ullman, SIGMOD 1996 *"Implementing data cubes efficiently"*

Acknowledgement:
- The following slides have been created adapting the instructor material of the [RG] book provided by the authors Dr. Ramakrishnan and Dr. Gehrke.
- Some slides have been prepared by Prof. Shivnath Babu

Warehousing

- Growing industry: $8 billion way back in 1998
- Data warehouse vendor like Teradata
    - big "Petabyte scale"customers
    - Apple, Walmart (2008-2.5PB), eBay (2013-primary DW 9.2 PB, other big data 40PB, single table with 1 trillion rows), Verizon, AT&T, Bank of America
    - supports data into and out of Hadoop
- Lots of buzzwords, hype
    – slice & dice, rollup, MOLAP, pivot, ...

https://gigaom.com/2013/03/27/why-apple-ebay-and-walmart-have-some-of-the-biggest-data-warehouses-youve-ever-seen/

Ack: Slide by Prof. Shivnath Babu

Introduction

- Organizations analyze current and historical data
  - to identify useful patterns
  - to support business strategies
- Emphasis is on complex, interactive, exploratory analysis of very large datasets
- Created by integrating data from across all parts of an enterprise
- Data is fairly static
- Relevant once again for the recent "Big Data analysis"
  - to figure out what we can reuse, what we cannot

OLTP	Data Warehousing/OLAP
Mostly updates	Mostly reads
Applications: Order entry, sales update, banking transactions	Applications: Decision support in industry/organization
Detailed, up-to-date data	Summarized, historical data (from multiple operational db, grows over time)
Structured, repetitive, short tasks	Query intensive, ad hoc, complex queries
Each transaction reads/updates only a few tuples (tens of)	Each query can accesses many records, and perform many joins, scans, aggregates
MB-GB data	GB-TB data
Typically clerical users	Decision makers, analysts as users
Important: Consistency, recoverability, Maximizing tr. throughput	Important: Query throughput Response times

Three Complementary Trends

- Data Warehousing (DW):
  – Consolidate data from many sources in one large repository
  – Loading, periodic synchronization of replicas
  – Semantic integration
- OLAP:
  – Complex SQL queries and views.
  – Queries based on spreadsheet-style operations and "multidimensional" view of data.
  – Interactive and "online" queries.
- Data Mining:
  – Exploratory search for interesting trends and anomalies

ROLAP and MOLAP

- Relational OLAP (ROLAP)
  - On top of standard relational DBMS
  - Data is stored in relational DBMS
  - Supports extensions to SQL to access multi-dimensional data

- Multidimensional OLAP (MOLAP)
  - Directly stores multidimensional data in special data structures (e.g. arrays)

ROLAP: Star Schema

- To reflect multi-dimensional views of data
- Single fact table
- Single table for every dimension
- Each tuple in the fact table consists of
  - pointers (foreign key) to each of the dimensions (multi-dimensional coordinates)
  - numeric value for those coordinates
- Each dimension table contains attributes of that dimension

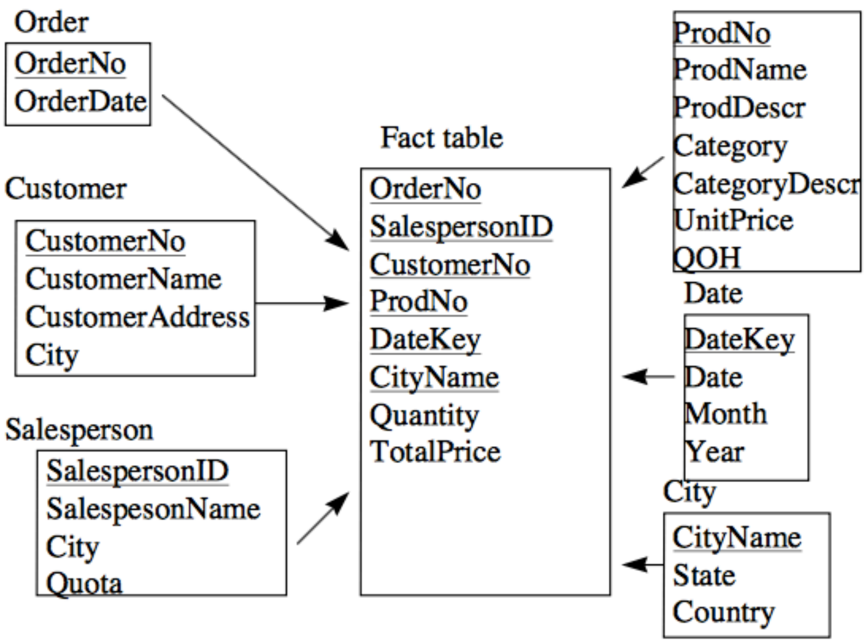


Figure 3. A Star Schema.

No support for attribute hierarchies

Dimension Hierarchies

- For each dimension, the set of values can be organized in a hierarchy:

ROLAP: Snowflake Schema

- Refines star-schema
- Dimensional hierarchy is explicitly represented

- (+) Dimension tables easier to maintain
  - suppose the "category description is being changed
- (-) Need additional joins
- Fact Constellations
  - Multiple fact tables share some dimensional tables
  - e.g. Projected and Actual Expenses may share many dimensions

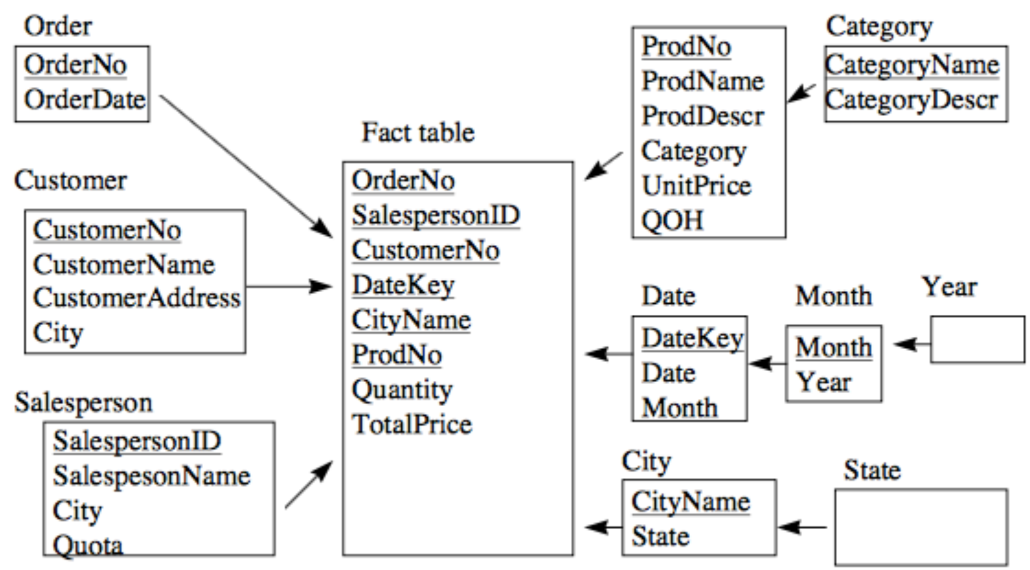


Figure 4. A Snowflake Schema.

OLAP
and
Data Cube

Sales (Model, Year, Color, Units)

Motivation: OLAP Queries

- Data analysts are interested in exploring trends and anomalies
  - Possibly by visualization (Excel) - 2D or 3D plots
  - “Dimensionality Reduction” by summarizing data and computing aggregates

- Find total unit sales for each
  1. Model
  2. Model, broken into years
  3. Year, broken into colors
  4. Year
  5. Model, broken into color, ….

Sales (Model, Year, Color, Units)

Roll-Ups

- Analysis reports start at a coarse level, go to finer levels
- Order of attribute matters
- Not relational data (empty cells no keys)

Roll-ups

Drill-downs

GROUP BY

Model	Year	Color	Model, Year, Color	Model, Year	Model
Chevy	1994	Black	50		
Chevy	1994	White	40		
				90	
Chevy	1995	Black	115		
Chevy	1995	White	85		
				200	
					290

- Roll-ups are asymmetric

Sales (Model, Year, Color, Units)

'ALL' Construct

Easier to visualize roll-up if allow ALL to fill in the super-aggregates

```
SELECT Model, Year, Color, SUM(Units)
     FROM Sales
     WHERE Model = 'Chevy'
         GROUP BY Model, Year, Color
UNION
SELECT Model, Year, 'ALL', SUM(Units)
     FROM Sales
     WHERE Model = 'Chevy'
     GROUP BY Model, Year
UNION
…
UNION
SELECT 'ALL', 'ALL', 'ALL', SUM(Units)
     FROM Sales
     WHERE Model = 'Chevy';
```

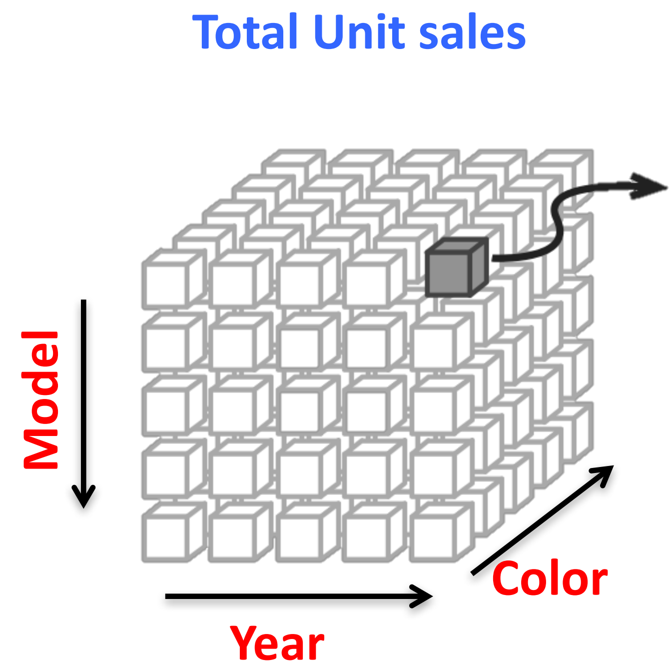
Model	Year	Color	Units
Chevy	1994	Black	50
Chevy	1994	White	40
Chevy	1994	'ALL'	90
Chevy	1995	Black	85
Chevy	1995	White	115
Chevy	1995	'ALL'	200
Chevy	'ALL'	'ALL'	290

Sales (Model, Year, Color, Units)

Data Cube: Intuition

More complex to do these with GROUP-BY

```
SELECT 'ALL', 'ALL', 'ALL', sum(units)
FROM Sales
UNION
SELECT 'ALL', 'ALL', Color, sum(units)
FROM Sales
GROUP BY Color
UNION
SELECT 'ALL', Year, 'ALL', sum(units)
FROM Sales
GROUP BY Year
UNION
SELECT Model, Year, 'ALL', sum(units)
FROM Sales
GROUP BY Model, Year
UNION
….
```



- How many sub-queries?
- How many sub-queries for 8 attributes?

Data Cube

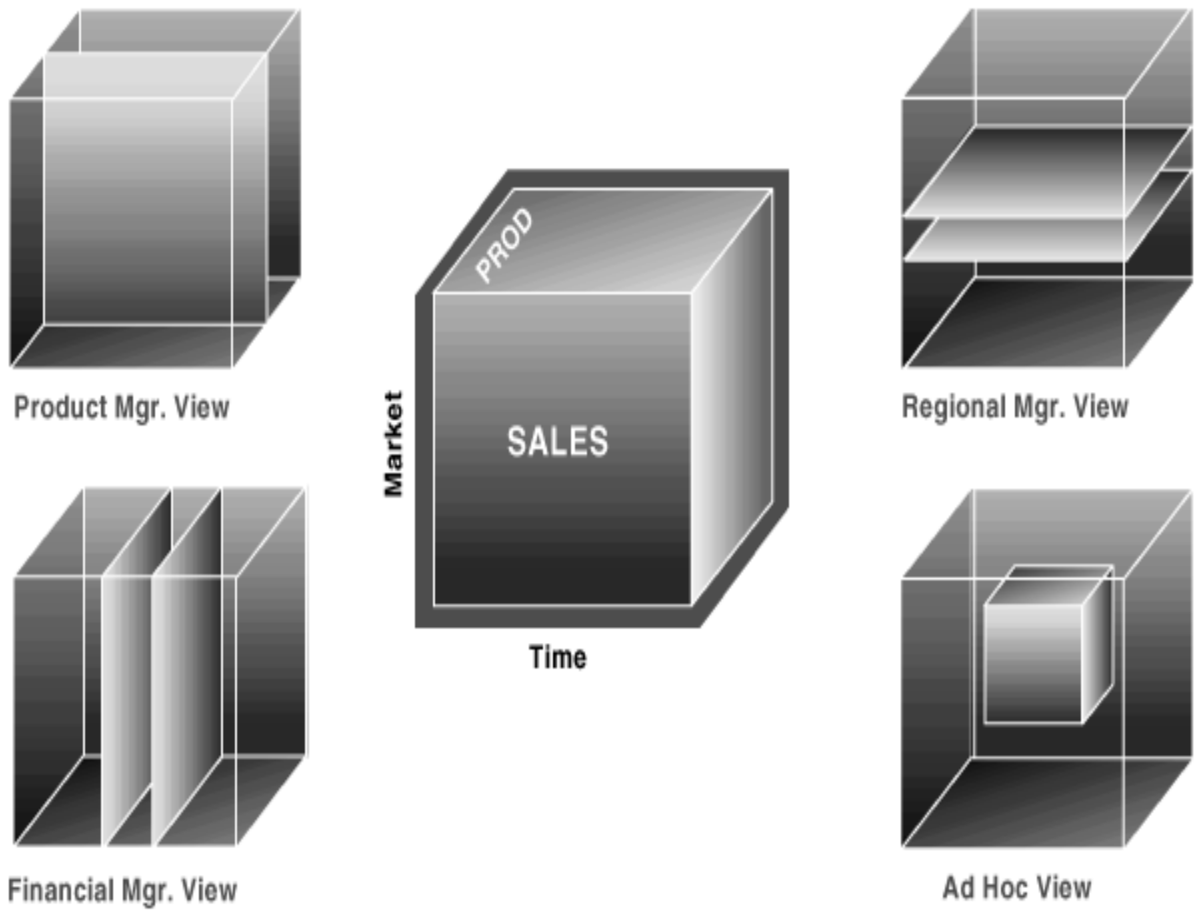
- Computes the aggregate on all possible combinations of group by columns.

- If there are N attributes, there are 2^N-1 super-aggregates.

- If the cardinality of the N attributes are C_1,..., C_N, then there are a total of $(C_1+1)...(C_N+1)$ values in the cube.

- ROLL-UP is similar but just looks at N aggregates

Data Cube



Ack: from slides by Laurel Orr and Jeremy Hyrkas, UW

Sales (Model, Year, Color, Units)

Data Cube Syntax

- SQL Server

```
SELECT Model, Year, Color, sum(units)
FROM Sales
GROUP BY Model, Year, Color
WITH CUBE
```

Implementing Data Cube

Basic Ideas

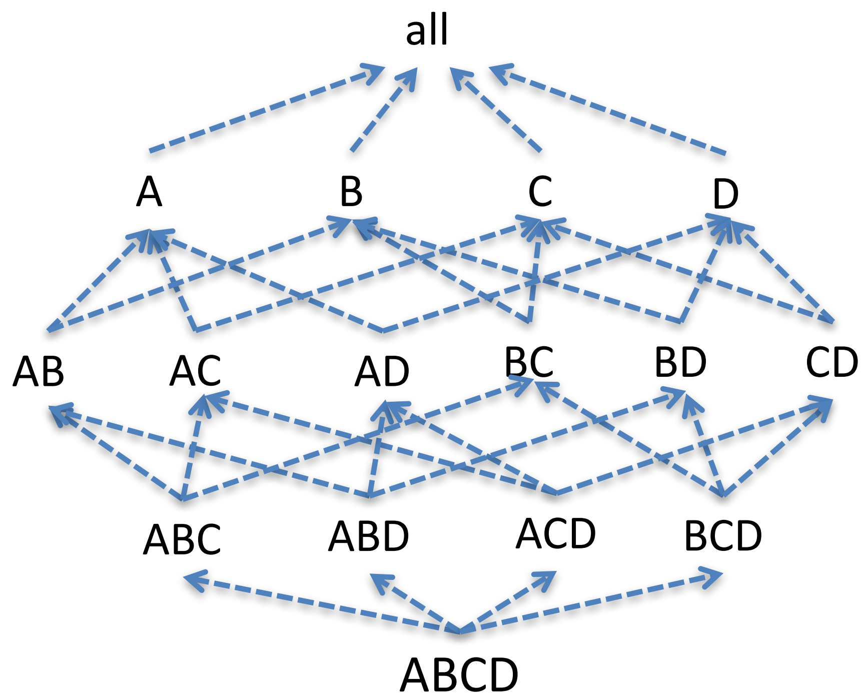
- Need to compute all group-by-s:
  - ABCD, ABC, ABD, BCD, AB, AC, AD, BC, BD, CD, A, B, C, D
- Compute GROUP-BYs from previously computed GROUP-BYs
  - e.g. first ABCD
  - then ABC or ACD
  - then AB or AC …
- Which order ABCD is sorted, matters for subsequent computations
  - if (ABCD) is the sorted order, ABC is cheap, ACD or BCD is expensive

Notations

- ABCD
  - group-by on attributes A, B, C, D
  - no guarantee on the order of tuples
- (ABCD)
  - sorted according to A -> B -> C -> D
- ABCD and (ABCD) and (BCDA)
  - all contain the same results
  - but in different sorted order

Optimization 1: Smallest Parent

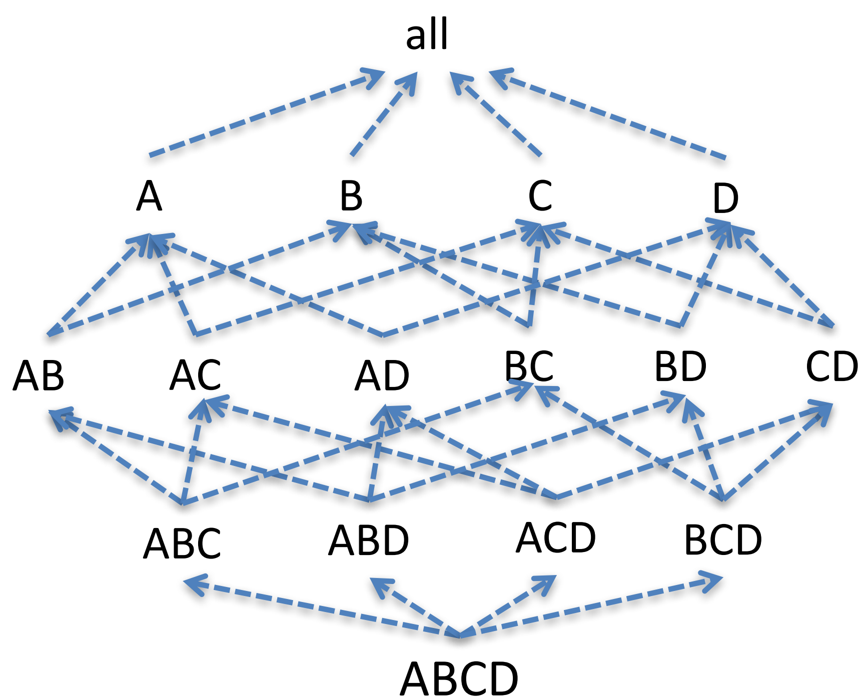
- Compute GROUP-BY from the smallest (size) previously computed GROUP-BY as a parent
  - AB can be computed from ABC, ABD, or ABCD
  - ABC or ABD better than ABCD
  - Even ABC or ABD may have different sizes, try to choose the smaller parent



LATTICE STRUCTURE of data cube

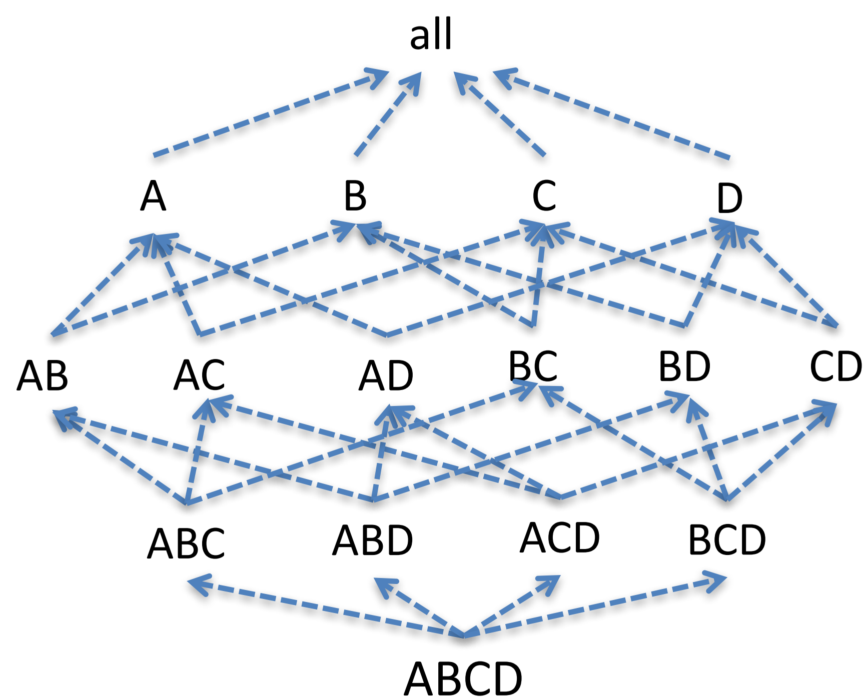
Optimization 2: Cache Results

- Cache result of one GROUP-BY in memory to reduce disk I/O
  - Compute AB from ABC while ABC is still in memory



Optimization 3: Amortize Disk Scans

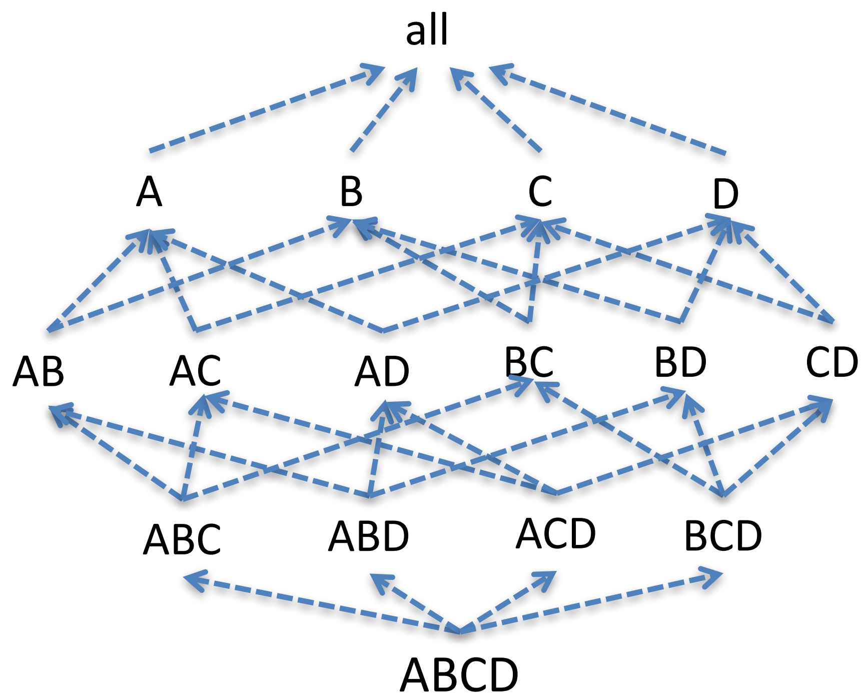
- Amortize disk reads for multiple GROUP-BYs
  - Suppose the result for ABCD is stored on disk
  - Compute all of ABC, ABD, ACD, BCD simultaneously in one scan of ABCD



Optimization 4: Shared partition

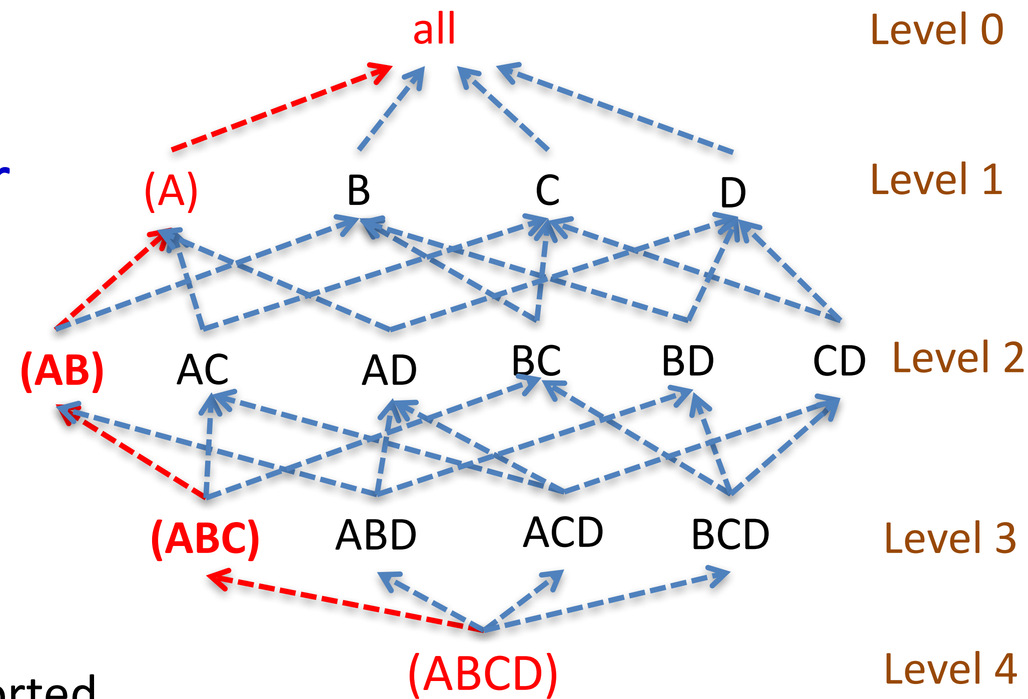
- for hash-based algorithms
  - Uses hash tables to compute smaller GROUP-Bys
  - If the hash tables for AB and AC fit in memory, compute both in one scan of ABC
  - Otherwise partition on A, and compute HTs of AB and AC in different partitions

- "pipe-hash" algorithm not covered in class



Optimization 5: Shared-sort

- From one sorted order of (ABCD)
  - Compute (ABC)
  - Compute (AB)
  - Compute (A)
  - Compute "all"



Sorted

A	B	C	sum
a1	b1	c1	5
a1	b1	c2	10
a1	b2	c3	8
a2	b2	c1	2
a2	b2	c3	11

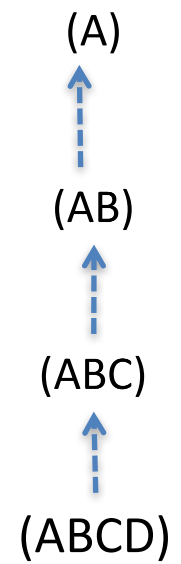
Not Sorted

A	B	C	sum
a2	b2	c3	11
a1	b1	c2	10
a2	b2	c1	2
a1	b1	c1	5
a1	b2	c3	8

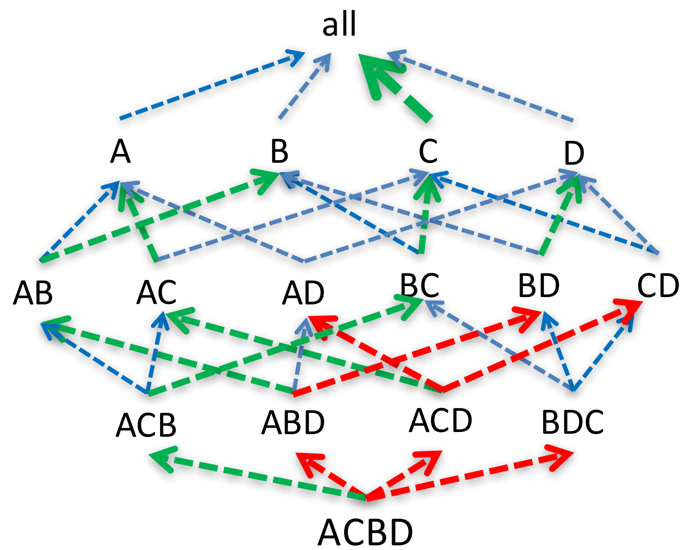
A	B	sum
a1	b1	15
a1	b2	8
a2	b2	13

PipeSort: Shared-sort optimization

- BUT, may have a conflict with "smallest-parent" optimization
  - (ABD) -> (AB) could be a better choice
  - Figure out the best parent choice by running a weighted-matching algorithm layer by layer (details not covered in class)



PipeSort Algorithm in picture

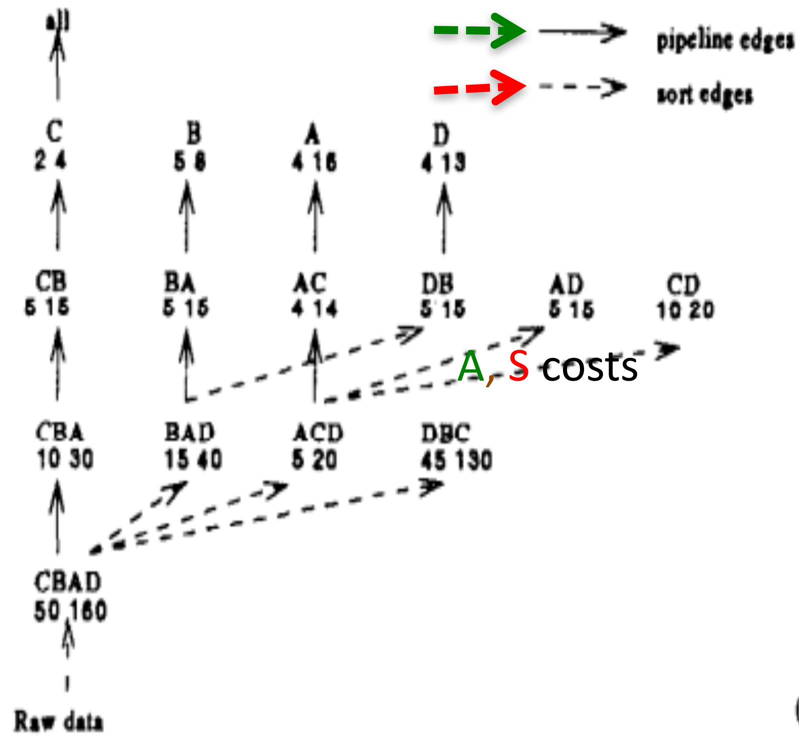


After computing the plan, execute all pipelines

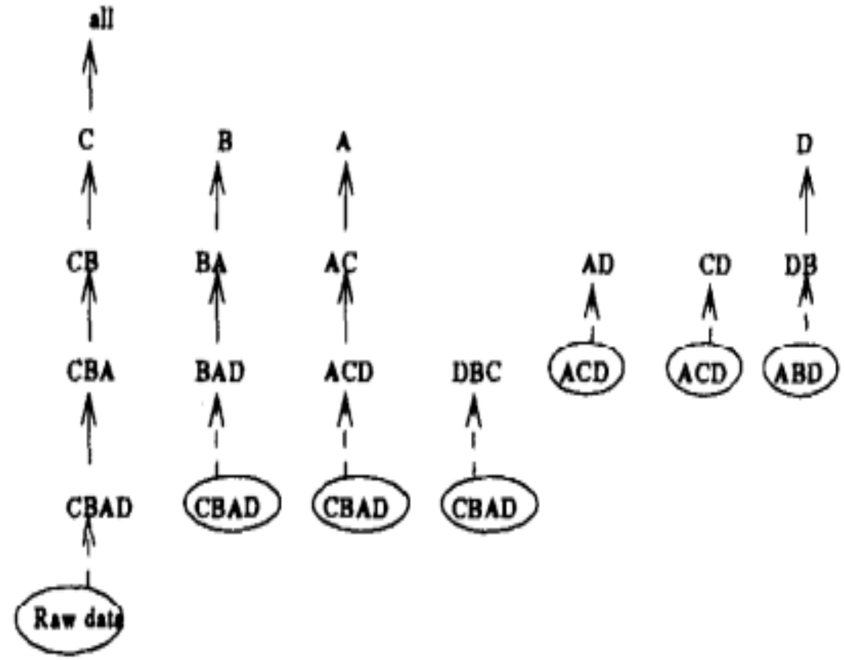
1. First pipeline is executed by one scan of the data

2. Sort (CBAD) -> (BADC), compute the second pipeline

3. .....



(a) The minimum cost sort plan



(b) The pipelines that are executed

Data Mining

Reading Material

Optional Reading:

1. [RG]: Chapter 26

2. "Fast Algorithms for Mining Association Rules"
Agrawal and Srikant, VLDB 1994

23,863 citations on Google Scholar in November 2018

- 23,038 in November 2017
- 20,610 in November 2016
- 19,496 in April 2016

One of the most cited papers in CS!

- Acknowledgement:

The following slides have been prepared adapting the slides provided by the authors of [RG] and using several presentations of this paper available on the internet (esp. by Ofer Pasternak and Brian Chase)

Four Main Steps in KD and DM (KDD)

Remember HW1!

- Data Selection
  - Identify target subset of data and attributes of interest
- Data Cleaning
  - Remove noise and outliers, unify units, create new fields, use denormalization if needed
- Data Mining
  - extract interesting patterns
- Evaluation
  - present the patterns to the end users in a suitable form, e.g. through visualization

Several DM/KD (Research) Problems

- Discovery of causal rules
- Learning of logical definitions
- Fitting of functions to data
- Clustering
- Classification
- Inferring functional dependencies from data
- Finding “usefulness” or “interestingness” of a rule
  - See the citations in the Agarwal-Srikant paper
  - Some discussed in [RG] Chapter 27

Mining Association Rules

- Retailers collect and store massive amounts of sales data
  - transaction date and list of items
- Association rules:
  - e.g. 98% customers who purchase "tires" and "auto accessories" also get "automotive services" done
  - Customers who buy mustard and ketchup also buy burgers
  - Goal: find these rules from just transactional data (transaction id + list of items)

Applications

- Can be used for
  - marketing program and strategies
  - cross-marketing (mass e-mail, webpages)
  - catalog design
  - add-on sales
  - store layout
  - customer segmentation

Notations

- Items $I = \{i_1, i_2, \ldots, i_m\}$
- D : a set of transactions
- Each transaction $T \subseteq I$
  - has an identifier TID
- Association Rule
  - $X \rightarrow Y$ (not Functional Dependency!)
  - $X, Y \subset I$
  - $X \cap Y = \emptyset$

Confidence and Support

- Association rule X→Y
- Confidence c = |Tr. with X and Y|/|Tr. with |X|
  - c% of transactions in D that contain X also contain Y
- Support s = |Tr. with X and Y| / |all Tr.|
  - s% of transactions in D contain X and Y.

Support Example

TID	Cereal	Beer	Bread	Bananas	Milk
1	X		X		X
2	X		X	X	X
3		X			X
4	X			X	
5			X		X
6	X				X
7		X		X	
8			X		

- Support(Cereal)
  - 4/8 = .5
- Support(Cereal → Milk)
  - 3/8 = .375

Confidence Example

TID	Cereal	Beer	Bread	Bananas	Milk
1	X		X		X
2	X		X	X	X
3		X			X
4	X			X	
5			X		X
6	X				X
7		X		X	
8			X		

- Confidence(Cereal → Milk)
  - 3/4 = .75
- Confidence(Bananas → Bread)
  - 1/3 = .33333…

Check yourself

X → Y is not a Functional Dependency

For functional dependencies

- F.D. = two tuples with the same value of of X must have the same value of Y
  - X → Y => XZ → Y (concatenation)
  - X → Y, Y → Z => X → Z (transitivity)

For association rules

- X → A does not mean XY→A
  - May not have the minimum support
  - Assume one transaction {AX}
- X → A and A → Z do not mean X → Z
  - May not have the minimum confidence
  - Assume two transactions {XA}, {AZ}

Problem Definition

- Input
  - a set of transactions D
    - Can be in any form – a file, relational table, etc.
  - min support (minsup)
  - min confidence (minconf)

- Goal: generate all association rules that have
  - support >= minsup and
  - confidence >= minconf

Decomposition into two subproblems

- 1. Apriori
  - for finding "large" itemsets with support >= minsup
  - all other itemsets are "small"
- 2. Then use another algorithm to find rules $X \rightarrow Y$ such that
  - Both itemsets $X \cup Y$ and X are large
  - $X \rightarrow Y$ has confidence >= minconf
- Paper focuses on subproblem 1
  - if support is low, confidence may not say much
  - subproblem 2 in full version of the paper

Basic Ideas - 1

- Q. Which itemset can possibly have larger support: ABCD or AB
  - i.e. when one is a subset of the other?
- Ans: AB
  - any subset of a large itemset must be large
  - So if AB is small, no need to investigate ABC, ABCD etc.

Basic Ideas - 2

- Start with individual (singleton) items {A}, {B}, …
- In subsequent passes, extend the "large itemsets" of the previous pass as "seed"
- Generate new potentially large itemsets (candidate itemsets)
- Then count their actual support from the data
- At the end of the pass, determine which of the candidate itemsets are actually large
  - becomes seed for the next pass
- Continue until no new large itemsets are found

Optional Slides on the Apriori Algorithm

Optional Slide

Notations

- Assume the database is of the form <TID, i1, i2, …> where items are stored in lexicographic order
- TID = identifier of the transaction
- Also works when the database is "normalized": each database record is <TID, item> pair

k-itemset	An itemset having k items.
L_k	Set of large k-itemsets (those with minimum support). Each member of this set has two fields: i) itemset and ii) support count.
C_k	Set of candidate k-itemsets (potentially large itemsets). Each member of this set has two fields: i) itemset and ii) support count.

ACTUAL

POTENTIAL

Used in both Apriori and AprioriTID

Optional Slide

Algorithm Apriori

$L_1 = \{large\ 1\text{-}itemsets\}$

For $(k = 2;\ L_{k-1} \neq \phi;\ k{+}{+})$ do begin

  $C_k = \boxed{\text{apriori-gen}(L_{k-1});}$

  forall transactions $t \in D$ do begin

    $C_t = \boxed{\text{subset}(C_k, t)}$

    forall candidates $c \in C_t$ do

      $c.count{+}{+};$

    end

  end

  $L_k = \{ c \in C_k \mid c.count \geq minsup \}$

end

$Answer = \bigcup_k L_k;$

Count individual item occurrences

Generate new k-itemsets candidates

count = 0

Find the support of all the candidates

C_t = candidates contained in t

increment count

Take only those with support >= minsup

Optional Slide

Apriori-Gen

- Takes as argument L_{k-1} (the set of all large k-1)-itemsets
- Returns a superset of the set of all large k-itemsets by augmenting L_{k-1}

Join step

$L_{k-1} \bowtie L_{k-1}$

insert into C_k

select $p.item_1, p.item_2, p.item_{k-1}, q.item_{k-1}$

from $L_{k-1} p, L_{k-1} q$

where $p.item_1 = q.item_1, \ldots, \quad p.item_{k-2} = q.item_{k-2}, \quad p.item_{k-1} < q.item_{k-1}$

p and q are two large

(k-1)-itemsets identical in all k-2 first items.

Join by adding the last item of q to p

Prune step

forall *itemsets c* $\in C_k$ do

forall *(k-1)-subsets s of c* do

if $(s \notin L_{k-1})$ then

delete *c from* C_k

Check all the subsets, remove all candidate with some "small" subset

Optional Slide

Apriori-Gen Example - 1

Step 1: Join (k = 4)

Assume numbers 1-5 correspond to individual items

L_3

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

C_4

- {1,2,3,4}

Optional Slide

Apriori-Gen Example - 2

Step 1: Join (k = 4)

Assume numbers 1-5 correspond to individual items

L_3

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

C_4

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

Optional Slide

Apriori-Gen Example - 3

Step 2: Prune (k = 4)

- Remove itemsets that can't have the required support because there is a subset in it which doesn't have the level of support i.e. not in the previous pass (k-1)

L_3

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

C_4

- {1,2,3,4}
- ~~{1,3,4,5}~~

No {1,4,5} exists in L_3
Rules out {1, 3, 4, 5}

Optional Slide

Correctness of Apriori

Check yourself

join

insert into C_k

select $p.item_1, p.item_2, p.item_{k-1}, q.item_{k-1}$

from $L_{k-1} p, L_{k-1} q$

where $p.item_1 = q.item_1, \ldots, \; p.item_{k-2} = q.item_{k-2}, \; p.item_{k-1} < q.item_{k-1}$

Show that $C_k \supseteq L_k$

- Any subset of large itemset must also be large
- for each p in L_k, it has a subset q in L_{k-1}
- We are extending those subsets q in Join with another subset q' of p, which must also be large
  - equivalent to extending L_{k-1} with all items and removing those

    whose (k-1) subsets are not in L_{k-1}
- Prune is not deleting anything from L_k

prune

forall *itemsets* $c \in C_k$ do

forall *(k-1)-subsets s of c* do

if $(s \notin L_{k-1})$ then

delete *c from* C_k

Conclusions

(of 516, Fall 2018)

Take-Aways

- DBMS Basics
- DBMS Internals
- Overview of Research Areas
- Hands-on Experience in DB systems

DB Systems

- Traditional DBMS
  - PostGres, SQL
- Large-scale Data Processing Systems
  - Spark/Scala, AWS
- New DBMS/NOSQL
  - MongoDB

- In addition
  - XML, JSON, JDBC, Python/Java

DB Basics

- SQL
- RA/Logical Plans
- RC
- Datalog
  - Why we needed each of these languages

- Normal Forms

DB Internals and Algorithms

- Storage
- Indexing
- Operator Algorithms
  - External Sort
  - Join Algorithms
- Cost-based Query Optimization
- Transactions
  - Concurrency Control
  - Recovery

Large-scale Processing and New Approaches

- Parallel DBMS
- Distributed DBMS
- Map Reduce
- NOSQL

Other Topics

- Data Warehouse/OLAP/Data Cube
- Association Rule Mining

- Hope some of you will further explore Database Systems/Data Management/Data Analysis/Big Data!