C-Store: Looking Back and Looking Forward

Daniel Abadi

Michael Stonebraker

(On behalf of all C-Store authors: Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran, Stan Zdonik)

DBg Database Group
MIT Computer Science and Artificial Intelligence Lab

Thank You

- 100s of Vertica employees (and consultants) who realized the C-Store vision
- Reviewers who accepted the paper in 2005
- Everyone who presented the C-Store paper in a seminar or database course
- Everyone who used ideas from C-Store paper and/or cited it

Column-stores have been around since the 70s

Pre-1985 Seminal Papers

TOD: Time Oriented Database – Wiederhold et al.
"A Modular, Self-Describing Clinical Databank System,"
Computers and Biomedical Research, 1975
More 1970s: Transposed files, Lorie, Batory, Svensson.

"An overview of cantor: a new system for data analysis" Karasalo, Svensson, SSDBM 1983

"A decomposition storage model" Copeland and Khoshafian. SIGMOD 1985.

Column-Stores that pre-date VLDB 2005

SybaseIQ

CWI: MonetDB, MonetDB/X100

Wisconsin: PAX, Fractured Mirrors

Michigan: Data Morphing

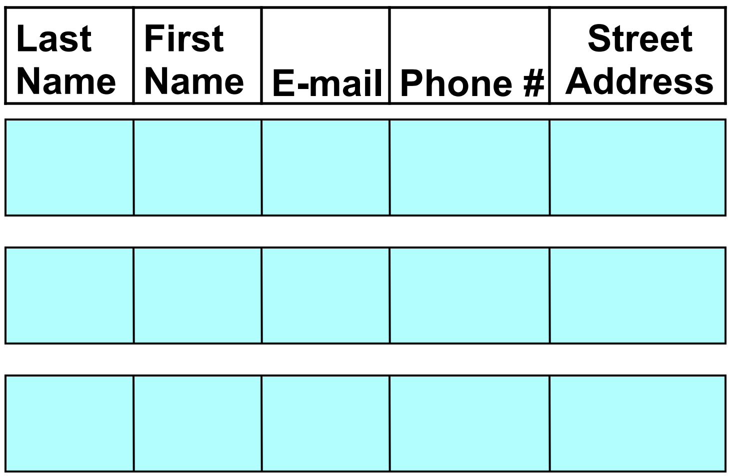
CMU: Clotho

Column-stores have been around since the 70s

C-Store's Practical System Design Expanded Applicability of Column-stores

Row vs. Column-Stores

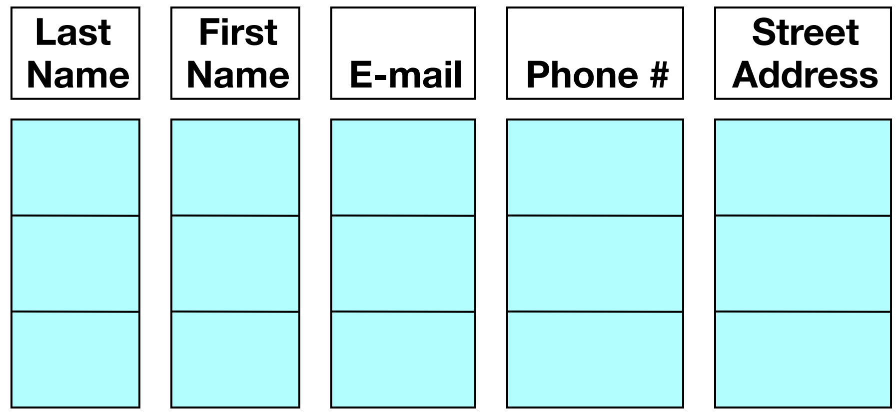
Row Store



+ Easy to add a new record

− Might read in unnecessary data

Column Store



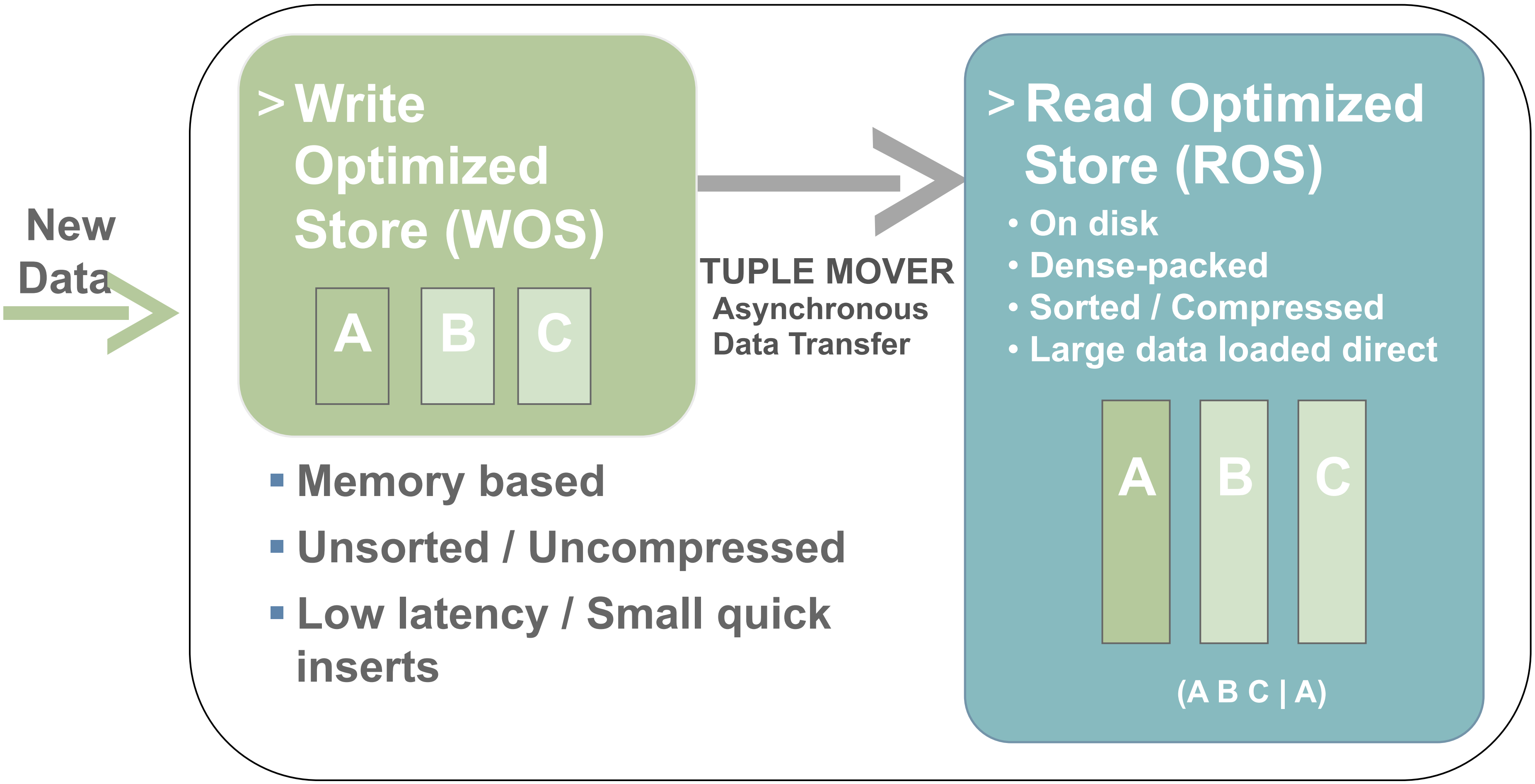
+ Only need to read in relevant columns

− SELECT X,Y,Z requires partial row reconstruction

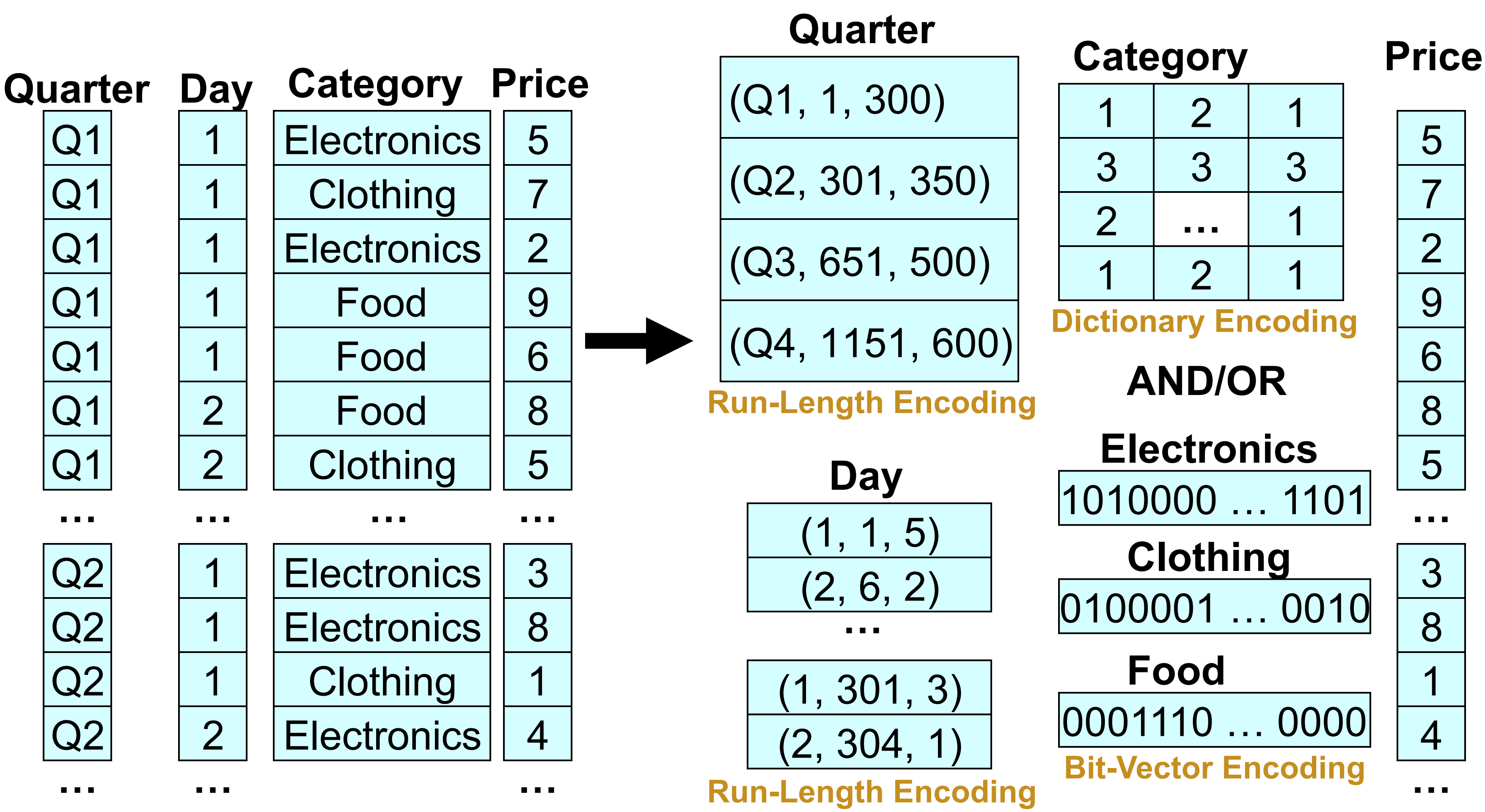
− Tuple writes might require multiple seeks

C-Store's Key Architectural Features

Hybrid Storage Architecture

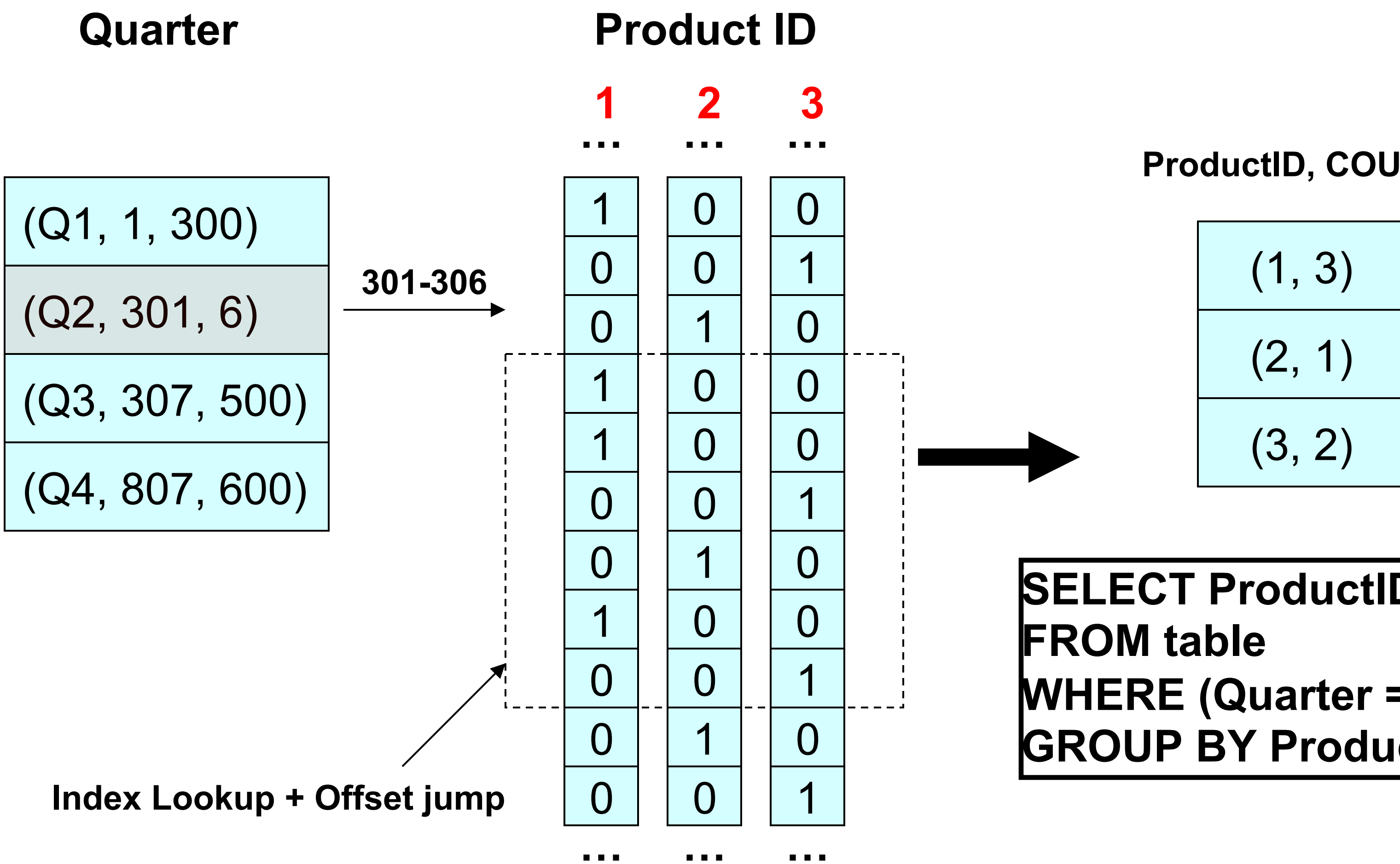


Key Features: Compression



Database Group
MIT Computer Science and Artificial Intelligence Lab

Key Features: Operating on Compressed Data

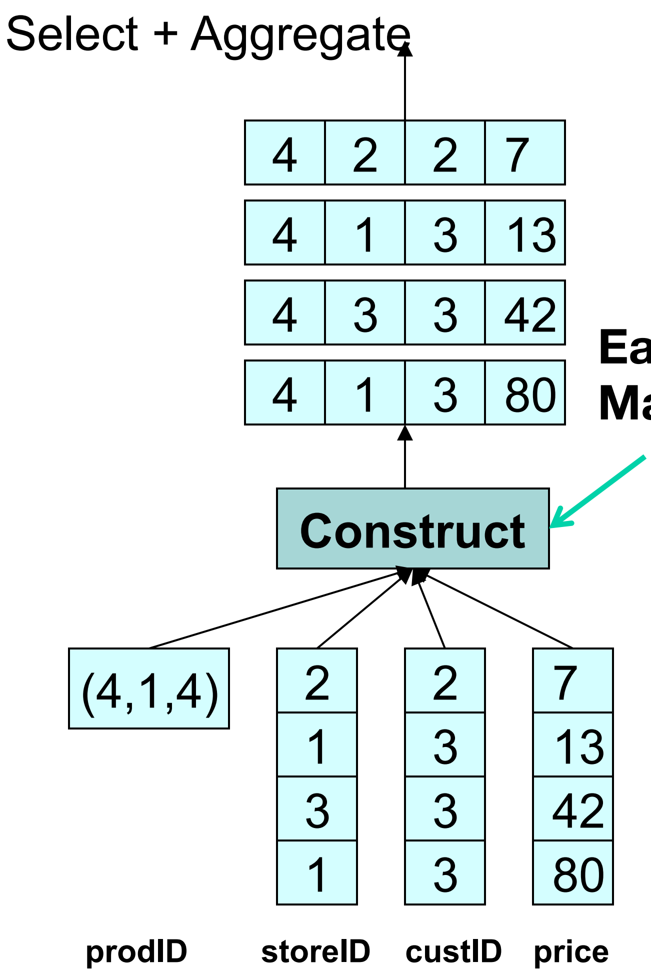



```
SELECT ProductID, Count(*)
FROM table
WHERE (Quarter = Q2)
GROUP BY ProductID
```

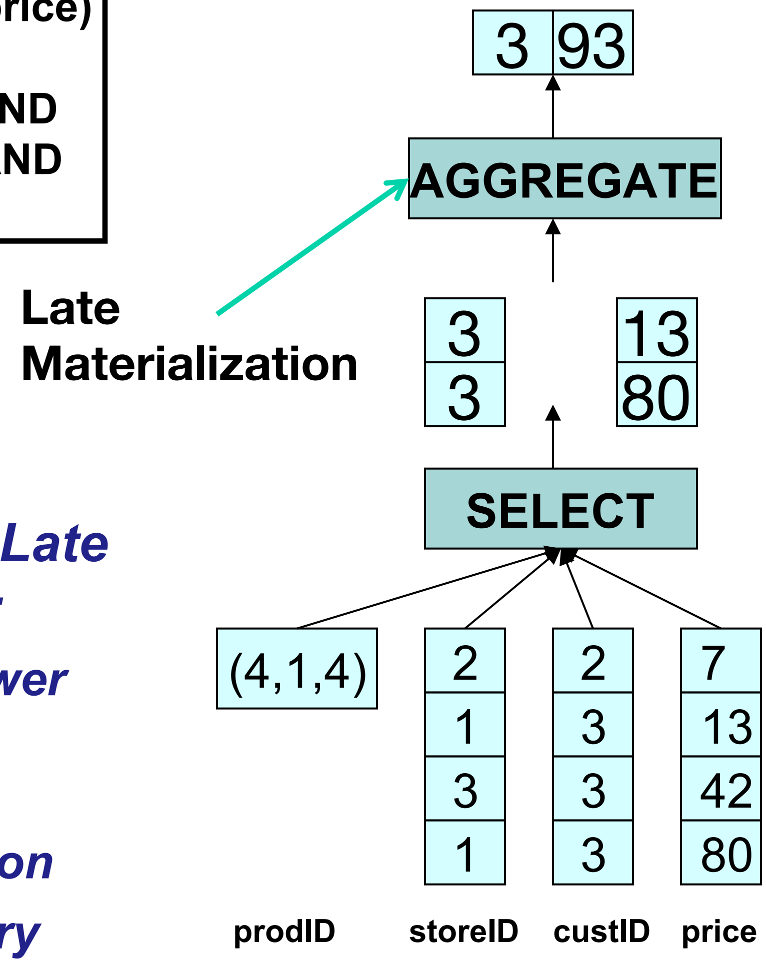
Key Features: Late Materialization

```
QUERY:

SELECT custID,SUM(price)
FROM table
WHERE (prodID = 4) AND
            (storeID = 1) AND
GROUP BY custID
```

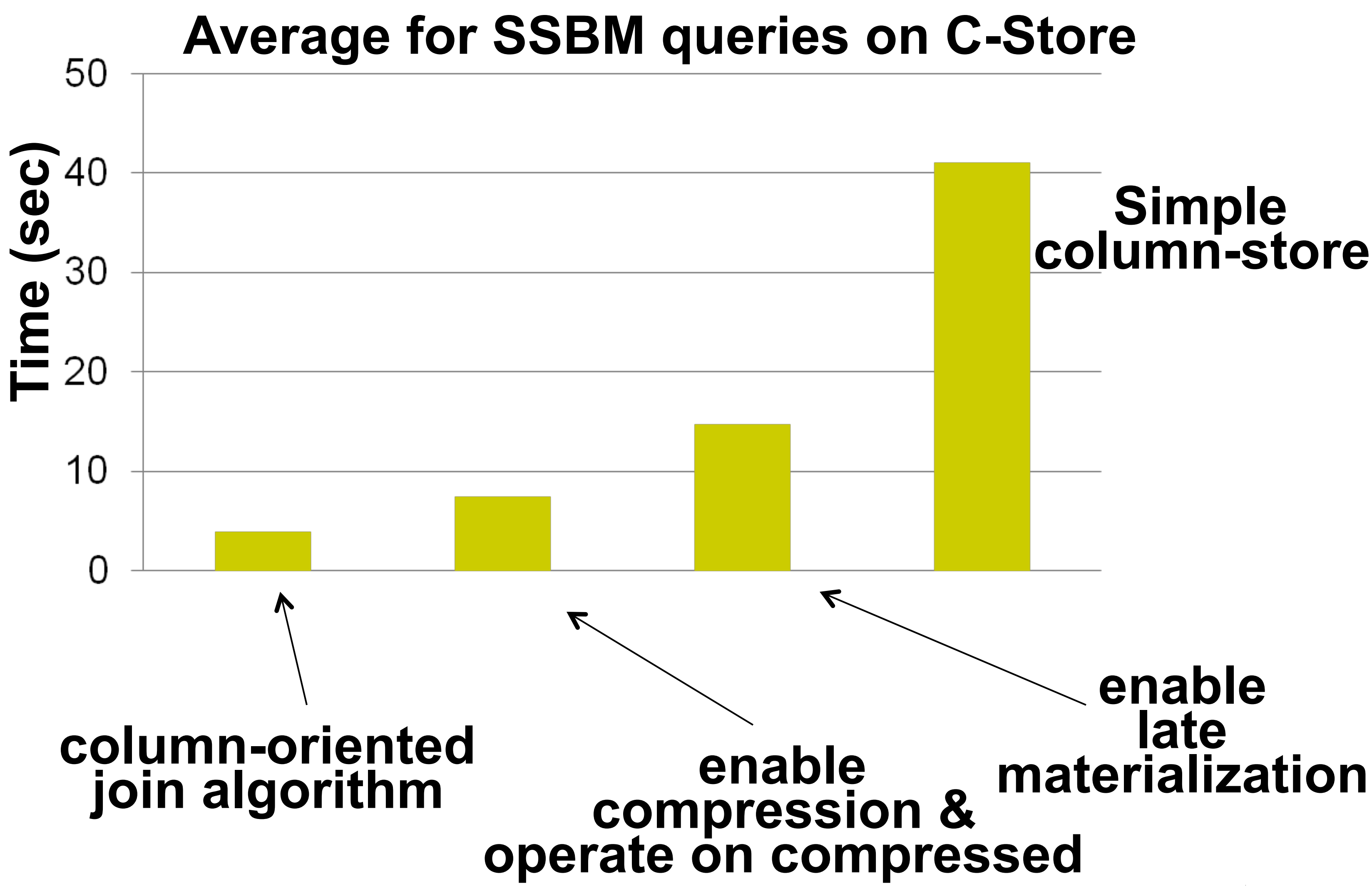


Early Materialization



- ***Advantages of Late Materialization:***
  - ***Construct fewer tuples***
  - ***Limits data decompression***
  - ***Better memory bandwidth utilization***

Effect on C-Store performance



Summary of key C-Store features

- Storage layout
  - columnar storage
  - header/ID elimination
  - compression
  - dense-packed values
- Execution engine
  - column operators
  - avoid decompression
  - late materialization

The Impact of Vertica

Warehouse vendor	2006	2015
IBM	Row store	Column store
Microsoft	Row store	Column store
Oracle	Row store	Rumors abound
Teradata	Row store	Row or column store
Greenplum	Row store	Row or column store
Cloudera		Column store
Amazon		Column store

Business Intelligence (Stupid Analytics)

- Is yesterday's news
- The coming world is "data science" (smart analytics)

Smart Analytics

- Complex math operations (machine learning, clustering, trend detection, ….)
  - The world of the “quants” and the “rocket scientists”
  - Mostly specified as linear algebra on array data

- A dozen or so common ‘inner loops’
  - Matrix multiply
  - QR decomposition
  - SVD decomposition
  - Linear regression

Smart Analytics
An Example

- Consider closing price on all trading days for the last 5 years for two stocks A and B

- What is the covariance between the two time-series?

  $(1/N) * \text{sum} (A_i - \text{mean}(A)) * (B_i - \text{mean}(B))$

Now Make It Interesting …

- Do this for all pairs of 4000 stocks
  - The data is the following 4000 x 1000 matrix

Stock	t_1	t_2	t_3	t_4	t_5	t_6	t_7	….	t_{1000}
S_1									
S_2									
…									
S_{4000}									

Array Answer

» Ignoring the (1/N) and subtracting off the means ….

$$\text{Stock} * \text{Stock}^T$$

What Does a Data Scientist Do? (after he curates his data)

» Until (tired) {

Data management

Analytics

}

How to Support This?

- Array DBMS (e.g. SciDB)
- Column store plus embedded stat (e.g. R, MatLab ..)
  - From Vertica, Cloudera, …

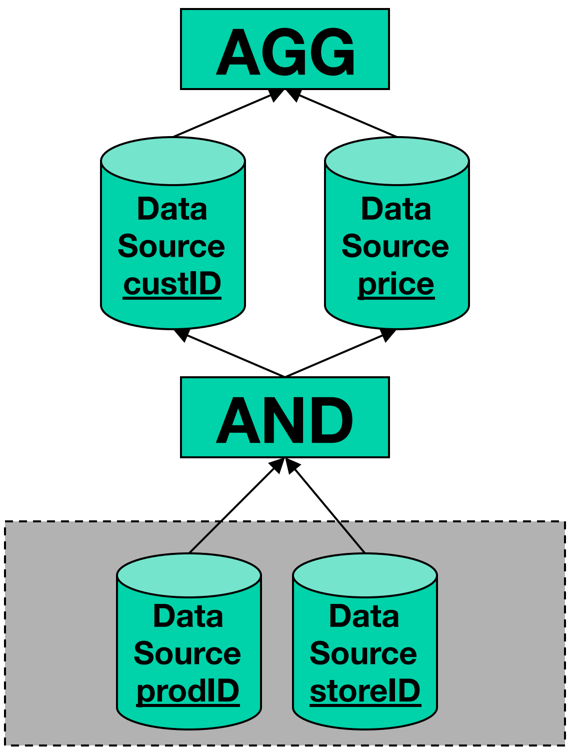
And….

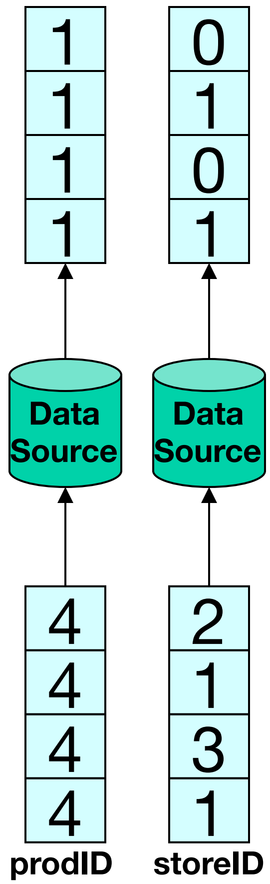
- Array stores do not need to cast tables to arrays (expensive N ** 2 operation)
  - So they have an unfair advantage
- But careful integration required in either case!!!
  - Block cyclic allocation
  - Same compression, same block layout, …

Linear Algebra

- Sparse and dense are both needed
  - And they are different
- On dense, there are 4 orders of magnitude difference between Python and C
- On dense, another order of magnitude from hardware-specific tweaking
- Focus on a few “inner loop” operations

Key Features: Late Materialization



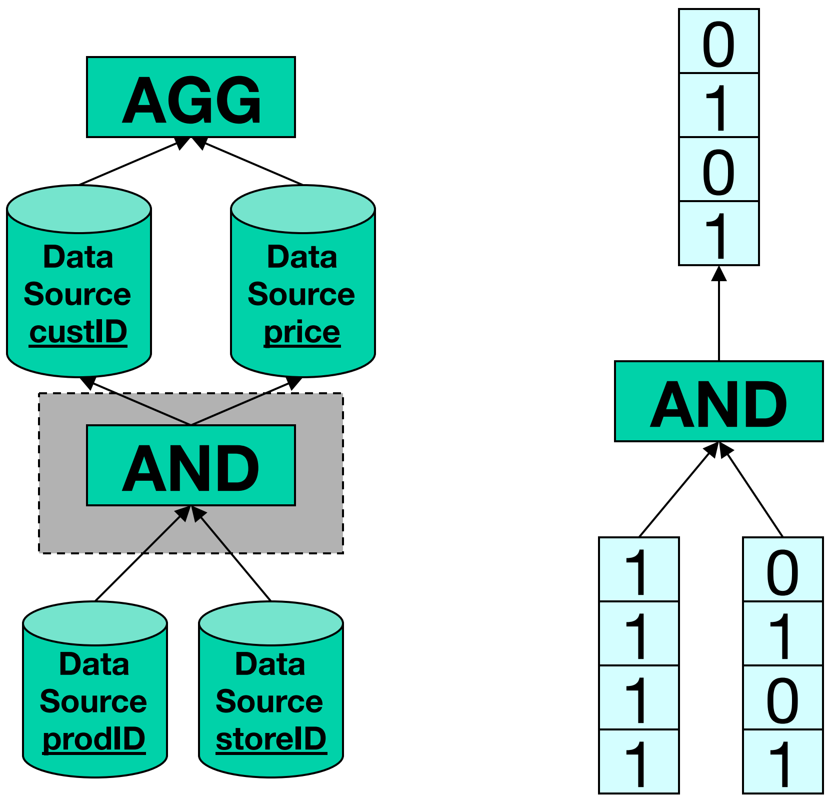


QUERY:

SELECT custID,SUM(price)
FROM table
WHERE (prodID = 4) AND
(storeID = 1) AND
GROUP BY custID

4	2	2	7
4	1	3	13
4	3	3	42
4	1	3	80
prodID	**storeID**	**custID**	**price**

Key Features: Late Materialization

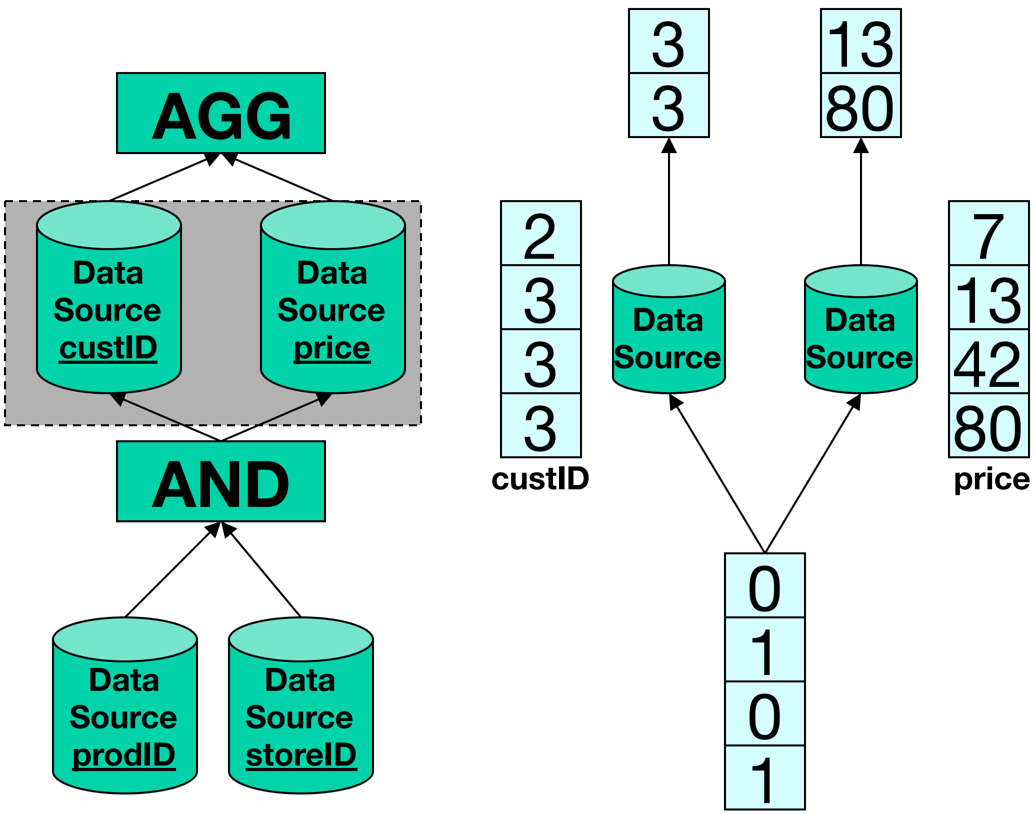



```
QUERY:

SELECT custID,SUM(price)
FROM table
WHERE (prodID = 4) AND
          (storeID = 1) AND
GROUP BY custID
```

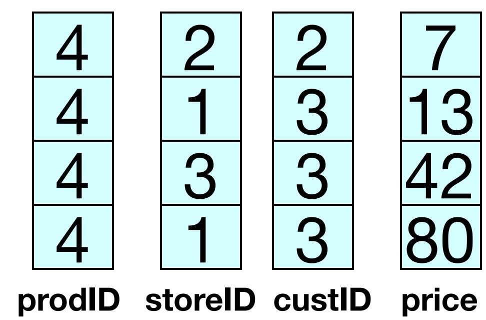
prodID	storeID	custID	price
4	2	2	7
4	1	3	13
4	3	3	42
4	1	3	80

Key Features: Late Materialization

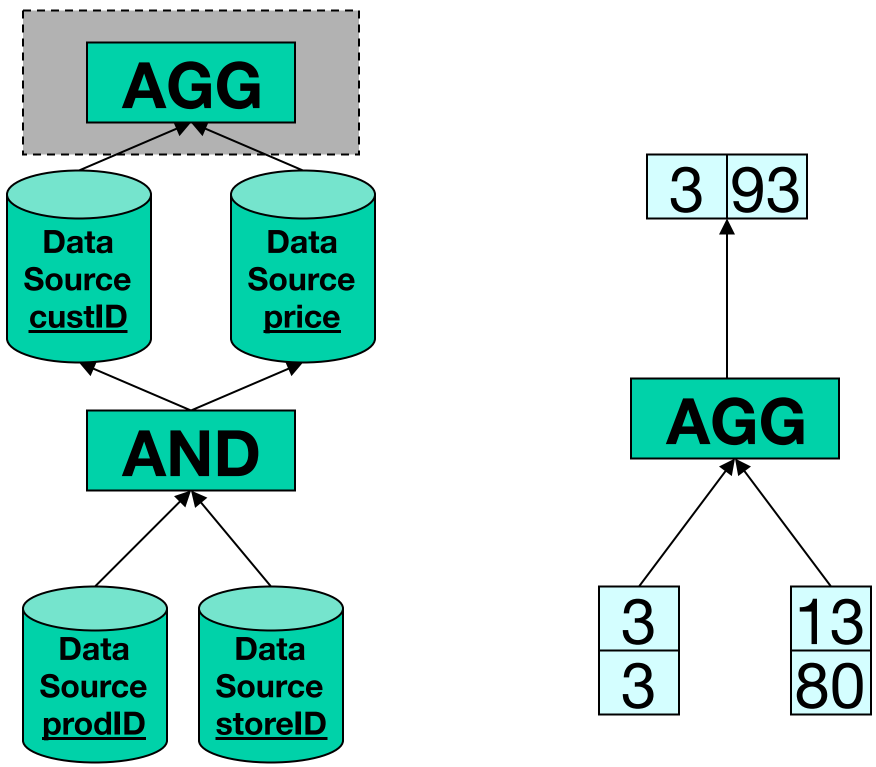


QUERY:

SELECT custID,SUM(price)
FROM table
WHERE (prodID = 4) AND
(storeID = 1) AND
GROUP BY custID



Key Features: Late Materialization



QUERY:

SELECT custID,SUM(price)
FROM table
WHERE (prodID = 4) AND
(storeID = 1) AND
GROUP BY custID

prodID	storeID	custID	price
4	2	2	7
4	1	3	13
4	3	3	42
4	1	3	80