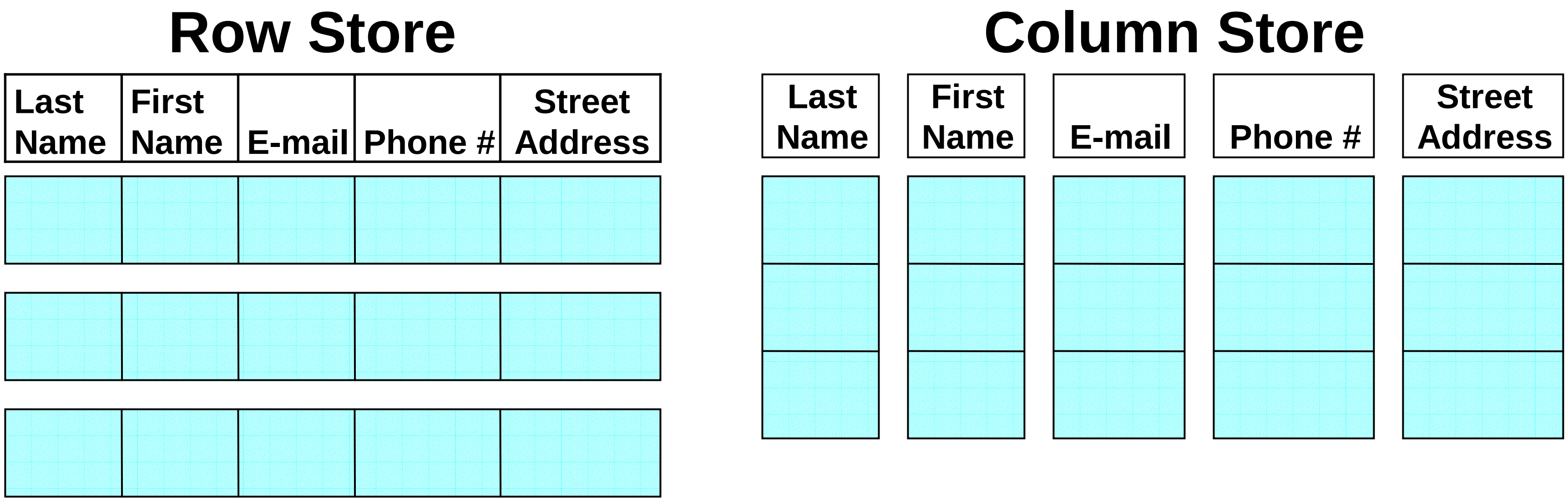
How to Create a New Column-Store DBMS Product In a Week

Daniel Abadi

February 4, 2008

Row vs. Column-Stores



- + **Easy to add a new record**
- – **Might read in unnecessary data**

- + **Only need to read in relevant data**
- + **Data compression**
- – **Tuple writes might require multiple seeks**

Column-Stores

- Really good for read-mostly data warehouses
  - Lot’s of column scans and aggregations
  - Writes tend to be in batch
  - [CK85], [SAB+05], [ZBN+05], [HLA+06], [SBC+07] all verify this
  - ParAccel zoomed to top TPC-H rankings
    - Factor of 5 faster on performance
    - Factor of 2 superior on price/performance

Column-Stores are the Answer

- Mike Stonebraker in a recent blog post:
  - “My prediction is that column stores will take over the warehouse market over time, completely displacing row stores. Since many warehouse users are in considerable pain (can't load in the available load window, can't support ad-hoc queries, can't get better performance without a "fork-lift" upgrade), I expect this transition to column stores will occur fairly quickly, as customers search for better ways to improve performance.”

Data Warehouse Software

- $4 billion industry (out of total $12-15 billion DBMS software industry)
- Growing 10% annually

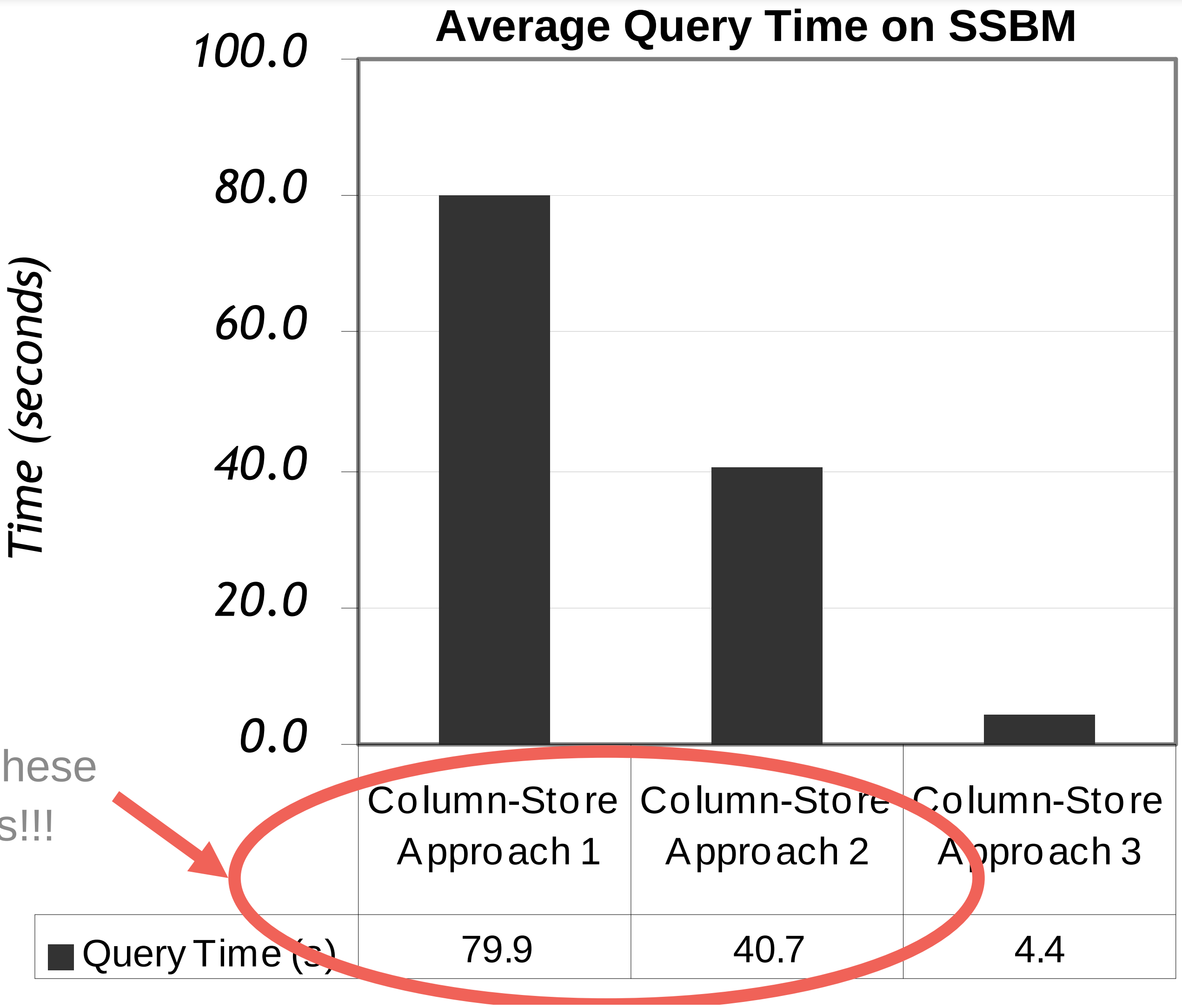
Momentum

- Right solution for growing market → $$$$
- ParAccel, Vertica, InfoBright, Calpont new entrants
- SybaseIQ, Sand/DNA older products

Want a piece of the action?

- Three options
  - Build on top of row-store (e.g., Postgres, Ingres)
  - Build a specialized storage manager
  - Build a full-fledged system

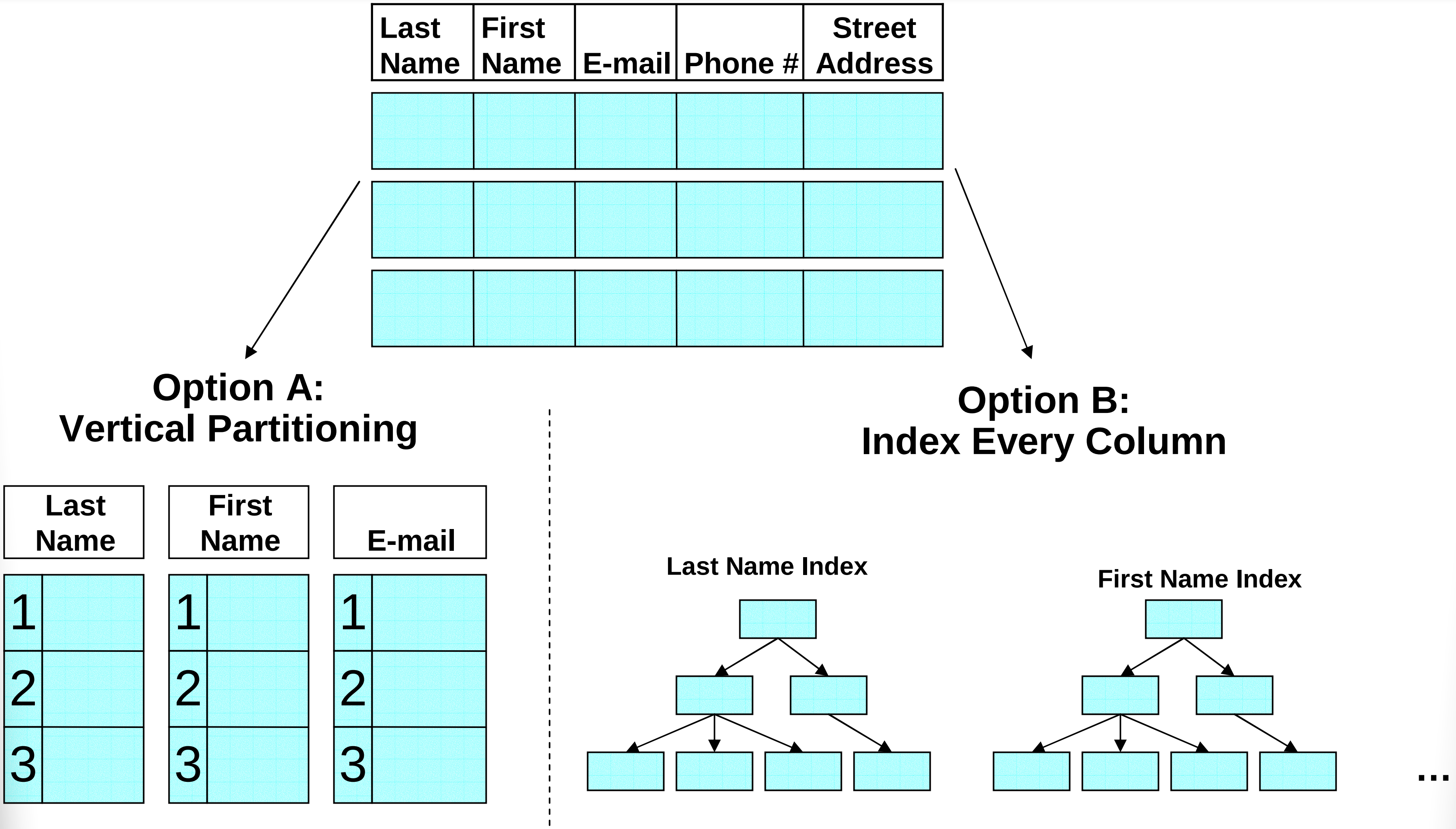
Why is the Distinction Important



	Column-Store Approach 1	Column-Store Approach 2	Column-Store Approach 3
■ Query Time (s)	79.9	40.7	4.4

Stop Calling These Column-Stores!!!

Column-Store Approach 1



Last Name	First Name	E-mail	Phone #	Street Address

Option A:
Vertical Partitioning

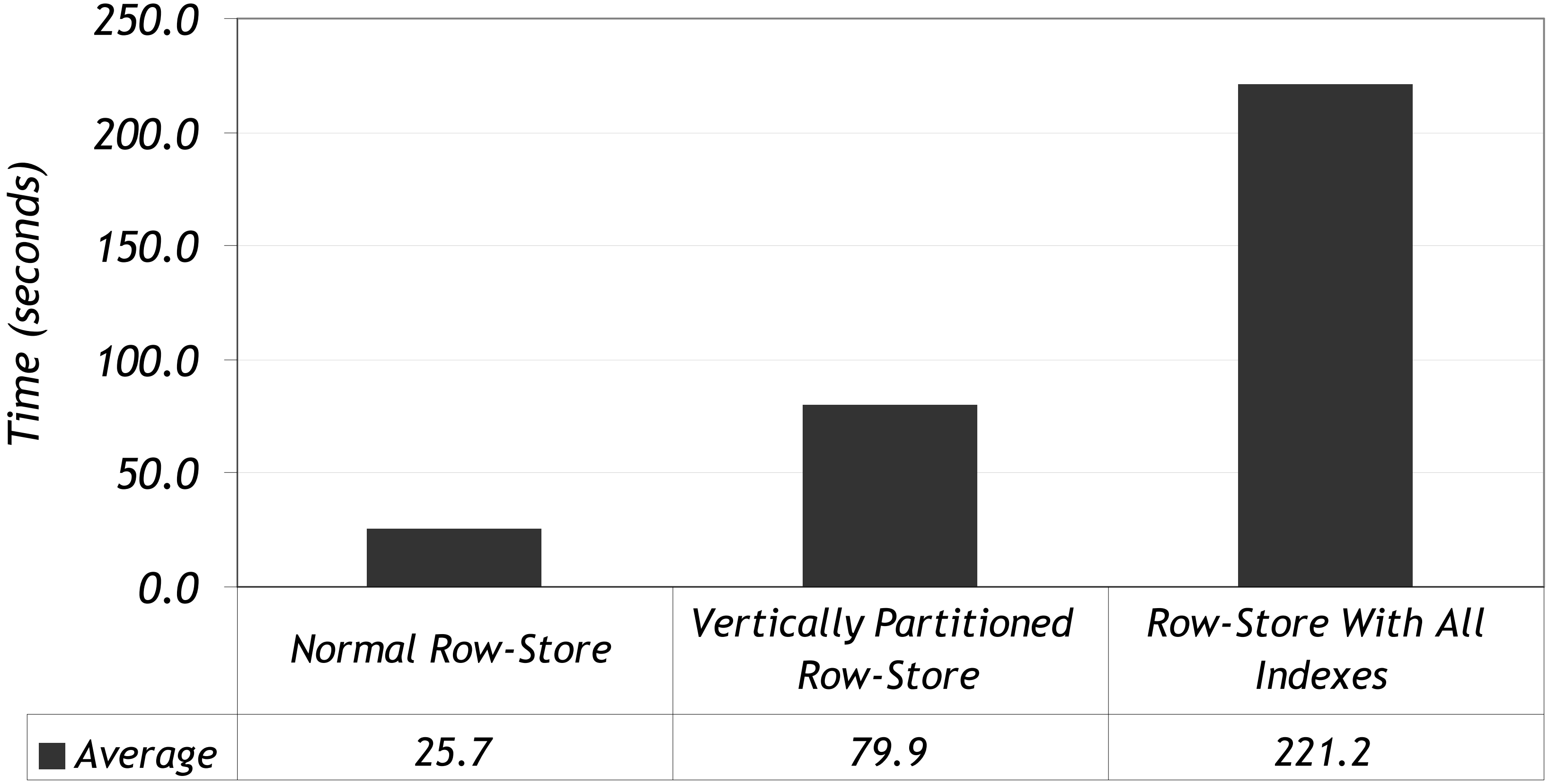
Last Name	
1	
2	
3	

First Name	
1	
2	
3	

E-mail	
1	
2	
3	

Option B:
Index Every Column

SSBM Averages



	Normal Row-Store	Vertically Partitioned Row-Store	Row-Store With All Indexes
■ Average	25.7	79.9	221.2

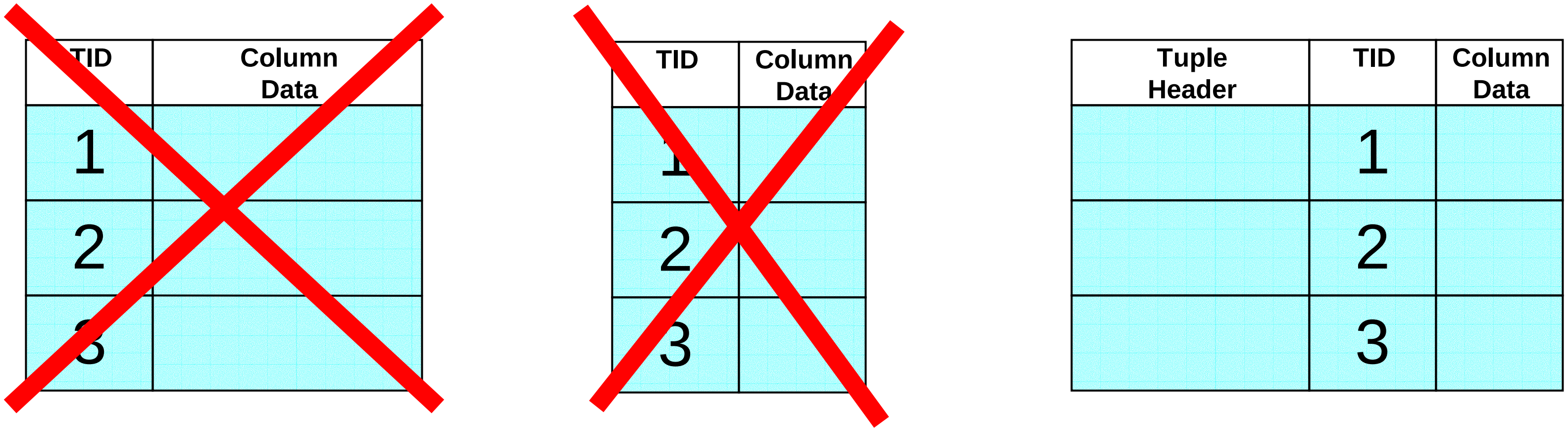
What’s Going On?

- Vertically Partitioned Case
  - Tuple Sizes
  - Horizontal Partitioning
- All Indexes Case
  - Tuple Reconstruction

Star Schema Benchmark

- Fact table contains 17 columns and 60,000,000 rows
- 4 dimension tables, biggest one has 80,000 rows
- Queries touch 3-4 foreign keys in fact table, 1-2 numeric columns

Tuple Size



~~TID~~	~~Column Data~~
~~1~~	
~~2~~	
~~3~~	

~~TID~~	~~Column Data~~
~~1~~	
~~2~~	
~~3~~	

Tuple Header	TID	Column Data
	1	
	2	
	3	

- Complete fact table takes up ~4 GB (compressed)
- Vertically partitioned tables take up 0.7-1.1 GB (compressed)

Horizontal Partitioning

- Fact table horizontally partitioned on year
  - Year is an element of the ‘Date’ dimension table
  - Most queries in SSBM have a predicate on year
  - Since vertically partitioned tables do not contain the ‘Date’ foreign key, row-store could not similarly partition them

What’s Going On?

- Vertically Partitioned Case
  - Tuple Sizes
  - Horizontal Partitioning
- All Indexes Case
  - Tuple Construction

Tuple Construction

- Pretty much all queries require a column to be extracted (in the SELECT clause) that has not yet been accessed, e.g.:

  ```
        SELECT store_name, SUM(revenue)
          FROM Facts, Stores
        WHERE fact.store_id = stores.store_id
              AND stores.area = “NEW ENGLAND”
    GROUP BY store_name
  ```

Tuple Construction

- Result of lower part of query plan is a set of TIDs that passed all predicates
- Need to extract SELECT attributes at these TIDs
  - BUT: index maps value to TID
  - You really want to map TID to value (i.e., a vertical partition)
  - → Tuple construction is SLOW

What does this all mean?

- All indexes approach is pretty obviously a poor way to simulate a column-store
- Problems with vertical partitioning are NOT fundamental
  - Store tuple header in a separate partition
  - Allow virtual TIDs
  - Allow HP using a foreign key on a different VP
  - So can row-stores simulate column-stores?

Come Join the Yale DB Group!