Advanced Data Management for Data Analysis

Stefan Manegold

Data Management @ LIACS

Group leader Database Architectures
Centrum Wiskunde & Informatica (CWI)
Amsterdam

s.manegold@liacs.leidenuniv.nl
http://www.cwi.nl/~manegold/

ADM: Agenda

- 07.09.2022: Lecture 1: **Introduction**
- 14.09.2022: Lecture 2: **SQL Recap**

  (plus Assignment 1 [in groups; 3 weeks]: TPC-H benchmark)
- 21.09.2022: Lecture 3: **Column-Oriented Database Systems (1/6) - Motivation & Basic Concepts**
- 28.09.2022: Lecture 4: **Column-Oriented Database Systems (2a/6) - Selected Execution Techniques (1/2)**
- 05.10.2022: Lecture 5: **Column-Oriented Database Systems (2b/6) - Selected Execution Techniques (2/2)**

  (plus Assignment 3 [in groups; 3 weeks]: Compression techniques)
- 12.10.2022: Lecture 6: **Column-Oriented Database Systems (3/6) - Cache Conscious Joins**
- 19.10.2022: Lecture 7: **Column-Oriented Database Systems (4/6) - "Vectorized Execution"**
- ~~26.10.2022~~: ***No lecture!***
- 02.11.2022: Lecture 8: **DuckDB: An embedded database for data science (1/2) (guest lecture & *hands-on*)**

  (plus Assignment 2 [individual; 2 weeks]: Analysing NYC Cab dataset with DuckDB)
- 09.11.2022: Lecture 9: **DuckDB: An embedded database for data science (2/2) (guest lecture & *hands-on*)**
- 16.11.2022: Lecture 10: **Branch Misprediction & Predication**

  (plus Assignment 4 [individual; 2 weeks]: Predication)
- 23.11.2022: Lecture 11: **Column-Oriented Database Systems (5/6) - Adaptive Indexing**
- 30.11.2022: Lecture 12: **Column-Oriented Database Systems (6/6) - Progressive Indexing**

ADM: Literature (1/2)

- **Column-Oriented Database Systems (1/6) - Motivation & Basic Concepts**
  - “An overview of cantor: a new system for data analysis“. Ilkka Karasalo, Per Svensson. SSDBM 1983.
  - “A decomposition storage model“. George P. Copeland, Setrag Khoshafian. SIGMOD Conference, 1985.
  - “Cache Conscious Algorithms for Relational Query Processing“. Ambuj Shatdal, Chander Kant, Jeffrey F. Naughton. VLDB 1994.
  - “MIL Primitives for Querying a Fragmented World“. Peter A. Boncz, Martin L. Kersten. VLDB J. 8(2): 101-119, 1999.
  - “Database Architecture Optimized for the New Bottleneck: Memory Access“. Peter A. Boncz, Stefan Manegold, Martin L. Kersten. VLDB 1999.
  - “DBMSs On A Modern Processor: Where Does Time Go?“. Anastassia Ailamaki, David J. DeWitt, Mark D. Hill, David A. Wood. VLDB 1999.
  - “Weaving Relations for Cache Performance ("PAX")“. Anastassia Ailamaki, David J. DeWitt, Mark D. Hill, Marios Skounakis. VLDB 2001.
  - “A Case for Fractured Mirrors“. Ravishankar Ramamurthy, David J. DeWitt, Qi Su. VLDB 2002.
  - “Data Morphing: An Adaptive, Cache-Conscious Storage Technique“. Richard A. Hankins, Jignesh M. Patel. VLDB 2003.
  - “Clotho: Decoupling Memory Page Layout from Storage Organization“. Minglong Shao, Jiri Schindler, Steven W. Schlosser, Anastassia Ailamaki, Gregory R. Ganger. VLDB 2004.
  - “MonetDB-X100 - A DBMS In The CPU Cache“. Marcin Zukowski, Peter A. Boncz, Niels Nes, Sándor Héman. IEEE Data Eng. Bull. 28(2): 17-22, 2005.
  - “"One size fits all": an idea whose time has come and gone“. Michael Stonebraker, Ugur Çetintemel. ICDE 2005.
  - “Performance Tradeoffs in Read-Optimized Databases“. Stavros Harizopoulos, Velen Liang, Daniel J. Abadi, Samuel Madden. VLDB 2006.
  - ...

ADM: Literature (2/2)

- **Column-Oriented Database Systems (1/6) - Motivation & Basic Concepts** *(cont.)*

  - ...
  - “One Size Fits All? - Part 2: Benchmarking Results“. Michael Stonebraker, Chuck Bear, Ugur Çetintemel, Mitch Cherniack, Tingjian Ge, Nabil Hachem, Stavros Harizopoulos, John Lifter, Jennie Rogers, Stanley B. Zdonik. CIDR 2007.
  - “C-Store: A Column-oriented DBMS“. Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel Madden, Elizabeth J. O'Neil, Patrick E. O'Neil, Alex Rasin, Nga Tran, Stanley B. Zdonik. VLDB 2005.
  - “Breaking the memory wall in MonetDB“. Peter A. Boncz, Martin L. Kersten, Stefan Manegold. Commun. ACM 51(12): 77-85, 2008.
  - “Column-Stores vs Row-Stores: How Different are They Really?“. Daniel J. Abadi, Samuel Madden, Nabil Hachem. SIGMOD Conference 2008.
  - “DSM vs. NSM: CPU performance tradeoffs in block-oriented query processing“. Marcin Zukowski, Niels Nes, Peter A. Boncz. DaMoN 2008.
  - “Fast Scans and Joins Using Flash Drives“. Mehul A. Shah, Stavros Harizopoulos, Janet L. Wiener, Goetz Graefe. DaMoN 2008.
  - “Read-Optimized Databases, In-Depth“. Allison L. Holloway, David J. DeWitt. Proc. VLDB Endow. 1(1): 502-513, 2008.
  - “Teaching an Old Elephant New Tricks“. Nicolas Bruno. CIDR 2009.
  - “Query Processing Techniques for Solid State Drives“. Dimitris Tsirogiannis, Stavros Harizopoulos, Mehul A. Shah, Janet L. Wiener, Goetz Graefe. SIGMOD Conference 2009.
  - “MonetDB: Two Decades of Research in Column-oriented Database Architectures“. Stratos Idreos, Fabian Groffen, Niels Nes, Stefan Manegold, K. Sjoerd Mullender, Martin L. Kersten. IEEE Data Eng. Bull. 35(1): 40-45, 2012.
  - “The Vertica Analytic Database: C-Store 7 Years Later“. Andrew Lamb, Matt Fuller, Ramakrishna Varadarajan, Nga Tran, Ben Vandiver, Lyric Doshi, Chuck Bear. Proc. VLDB Endow. 5(12): 1790-1801, 2012.

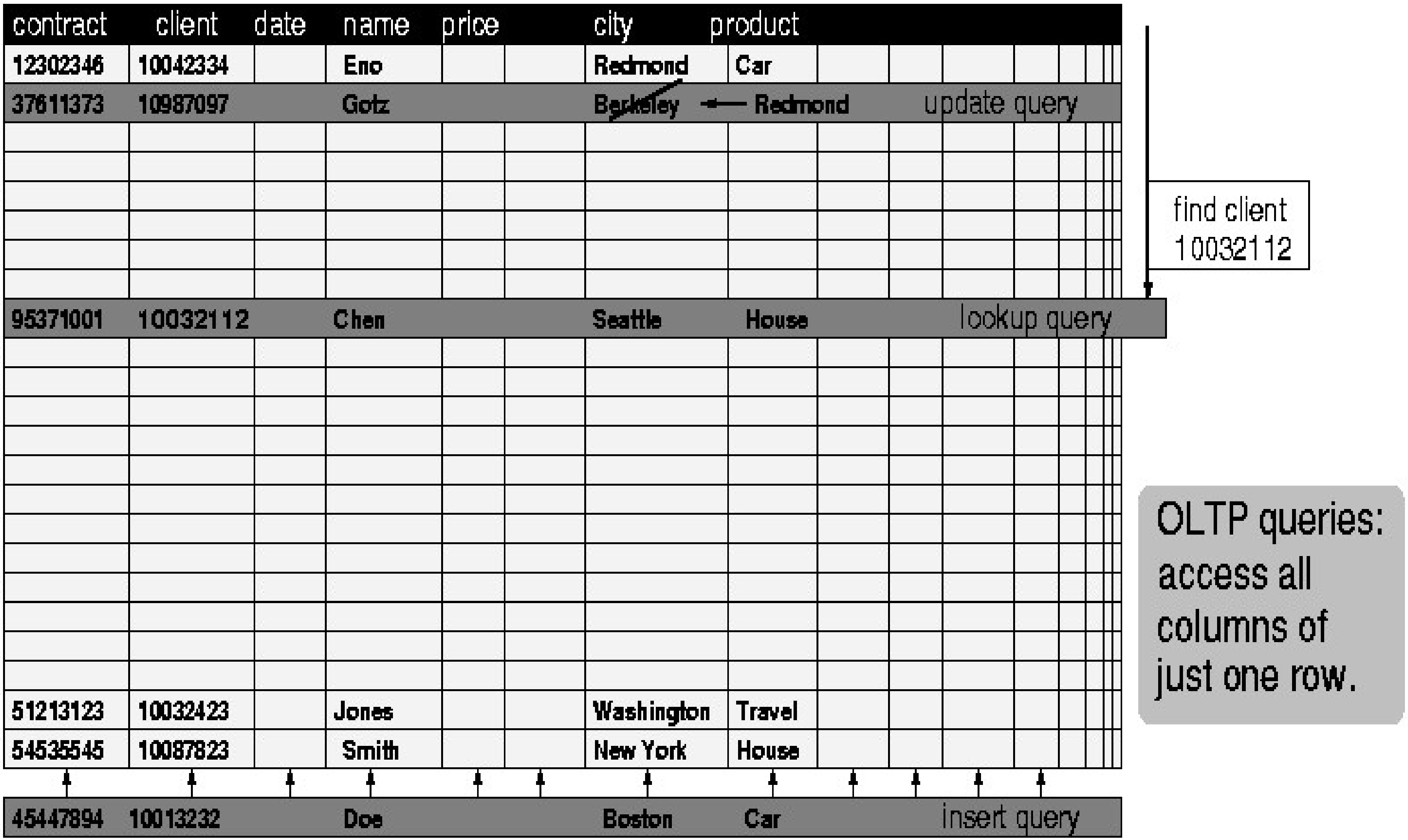
Why?

Motivation (early 1990s)

- Relational DBMSs dominate since the late 1970's / early 1980's
  - IBM DB2, MS SQL Server, Oracle, Ingres, ...
  - Transactional workloads (OLTP, row-wise access)
  - I/O based processing

Why?

Workload changes: **Transactions (OLTP) vs ...**



Why?

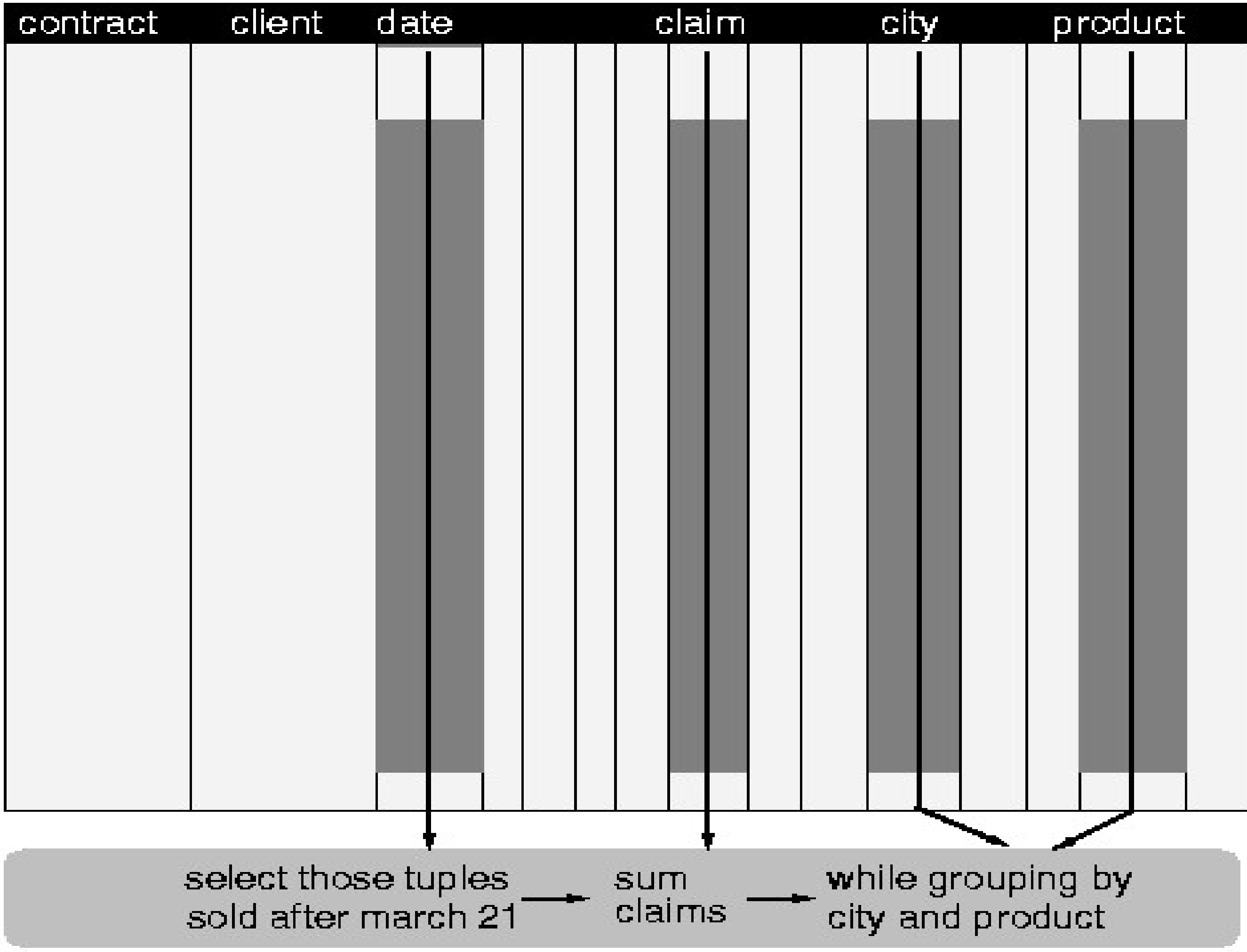
Motivation (early 1990s)

- Relational DBMSs dominate since the late 1970's / early 1980's
  - IBM DB2, MS SQL Server, Oracle, Ingres, ...
  - Transactional workloads (OLTP, row-wise access)
  - I/O based processing

- ***But:***
  - Workloads change (early 1990s)
  - Hardware changes (late 1990s)
  - Data "explodes" (early 2000s)

Why?

Workload changes: **... vs OLAP, BI, Data Mining, ...**

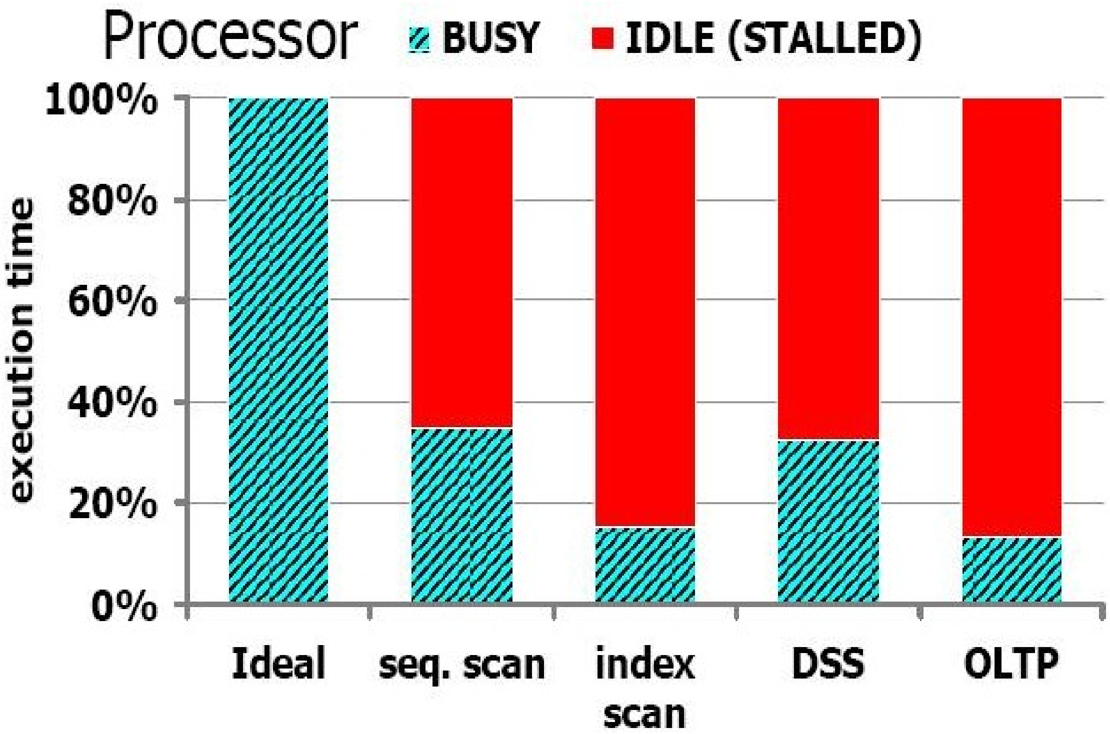


OLAP query: accesses only a few columns of almost all rows.

Why?

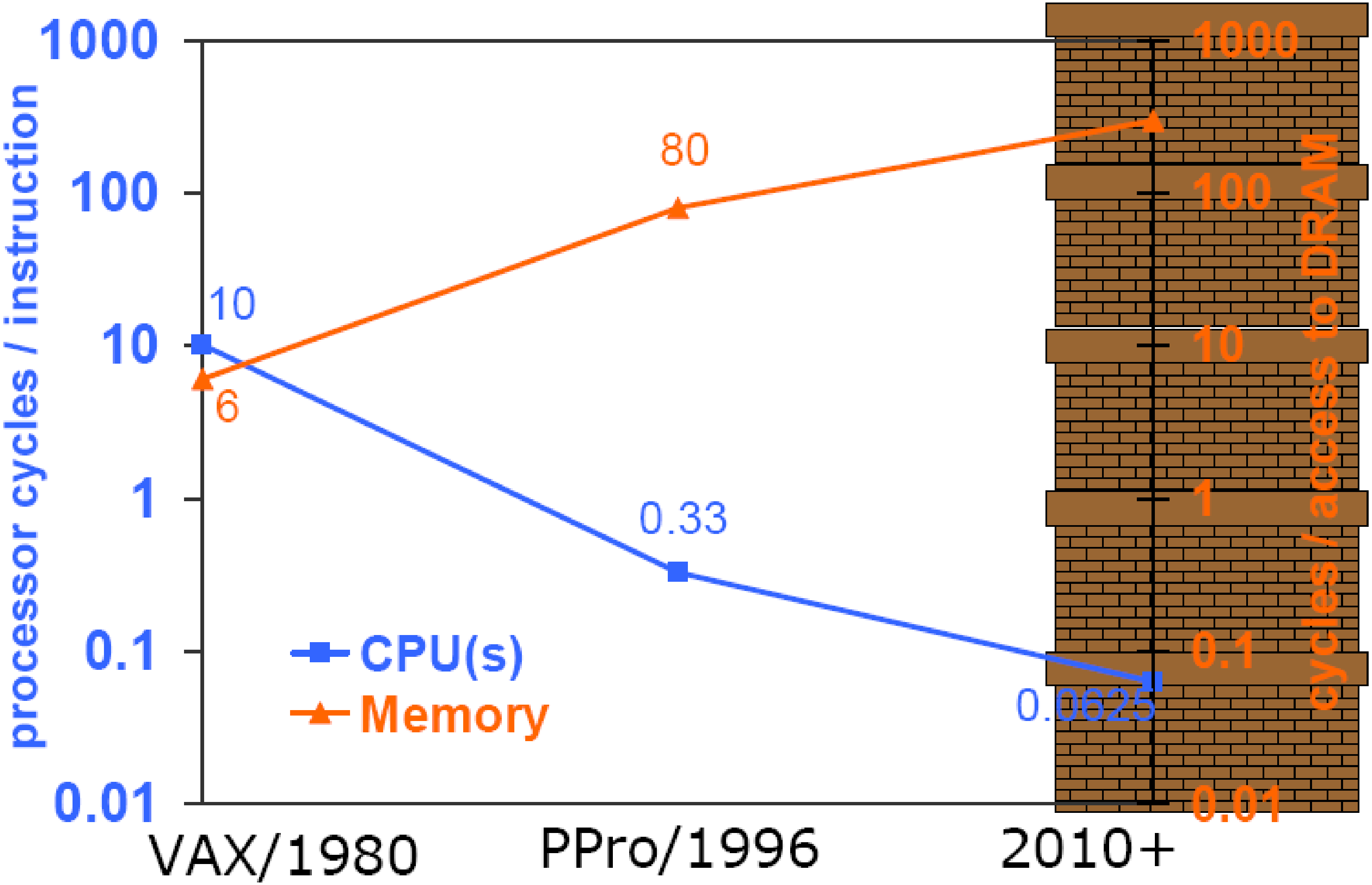
Databases hit The Memory Wall

- Detailed and exhaustive analysis for different workloads using 4 RDBMSs by Ailamaki, DeWitt, Hill, Wood in VLDB 1999: "*DBMSs On A Modern Processor: Where Does Time Go?*"
- CPU is 60%-90% idle, waiting for memory:
  - L1 data stalls
  - L1 instruction stalls
  - L2 data stalls
  - TLB stalls
  - Branch mispredictions
  - Resource stalls



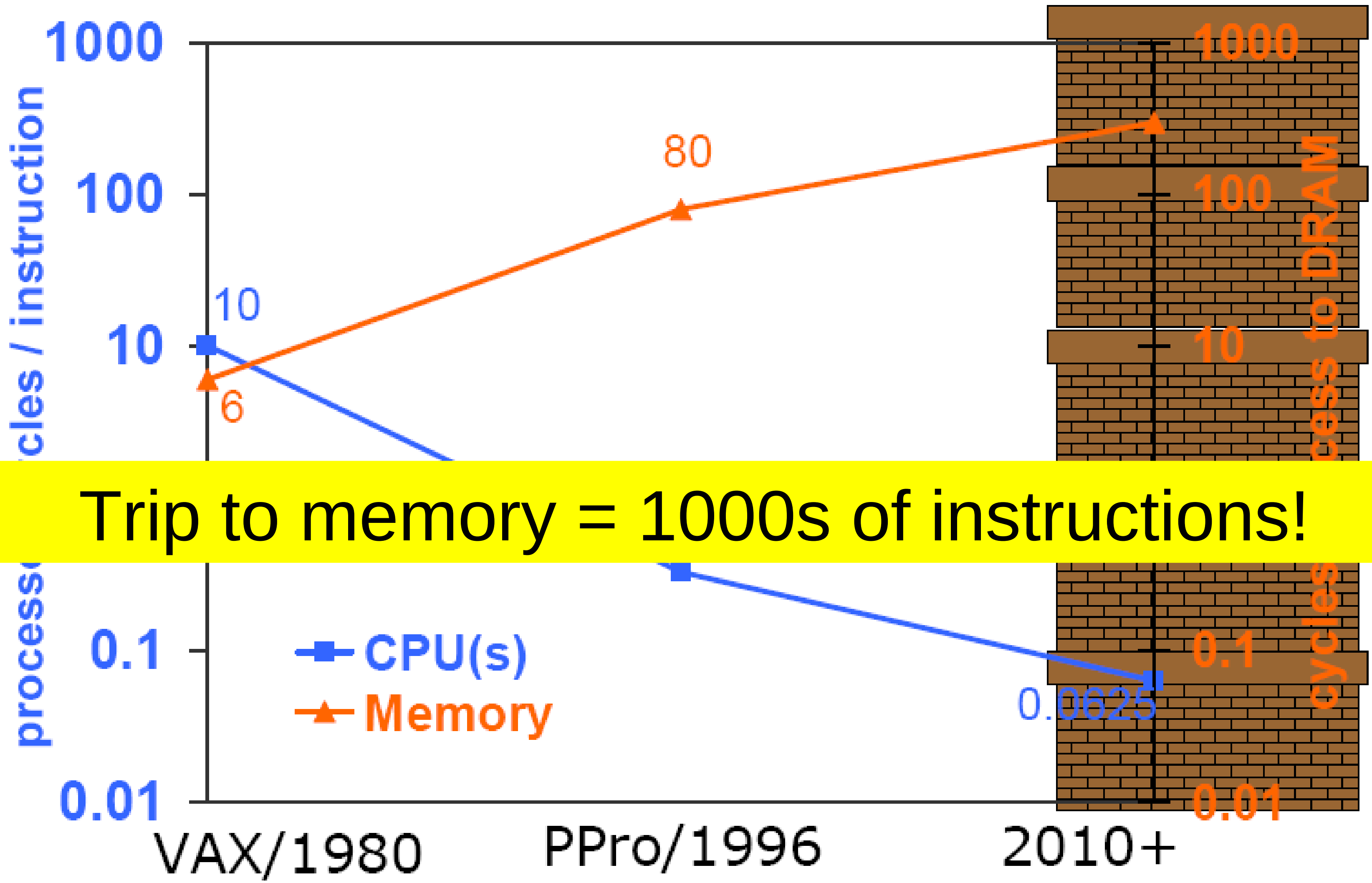
Why?

Hardware Changes: **The Memory Wall**



Why?

<u>Hardware Changes:</u> **The Memory Wall**

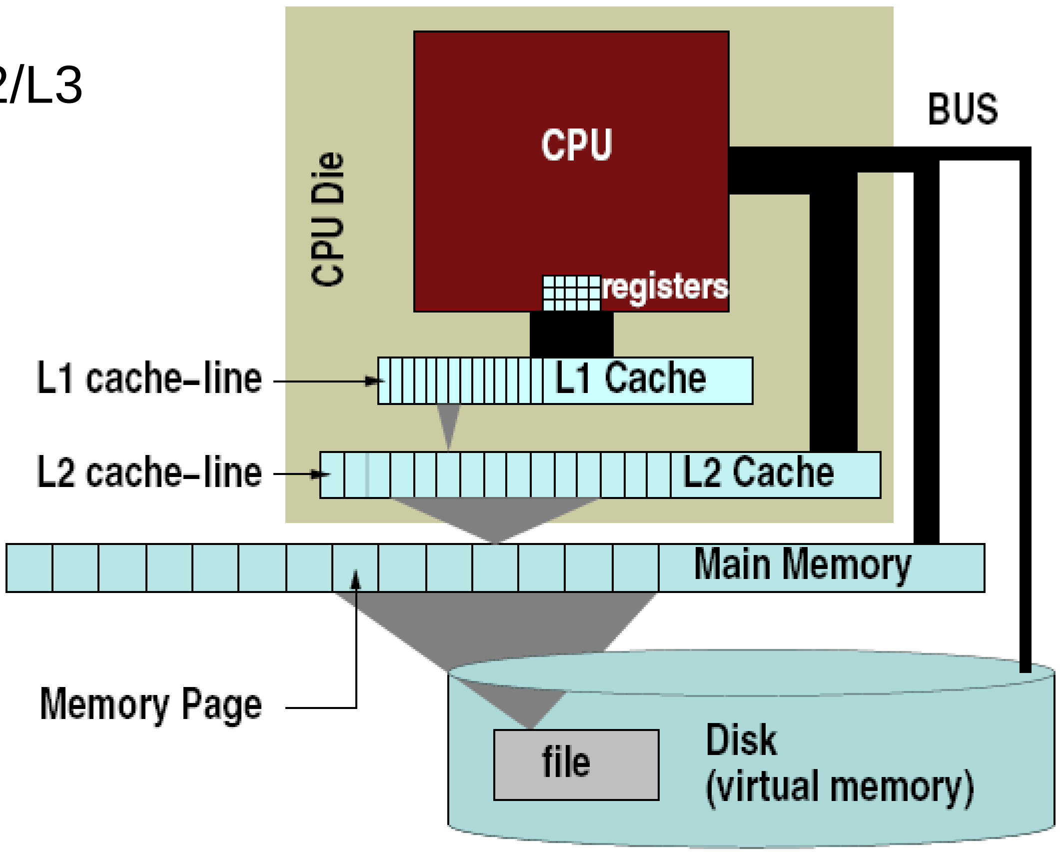


Trip to memory = 1000s of instructions!

CPU Architecture

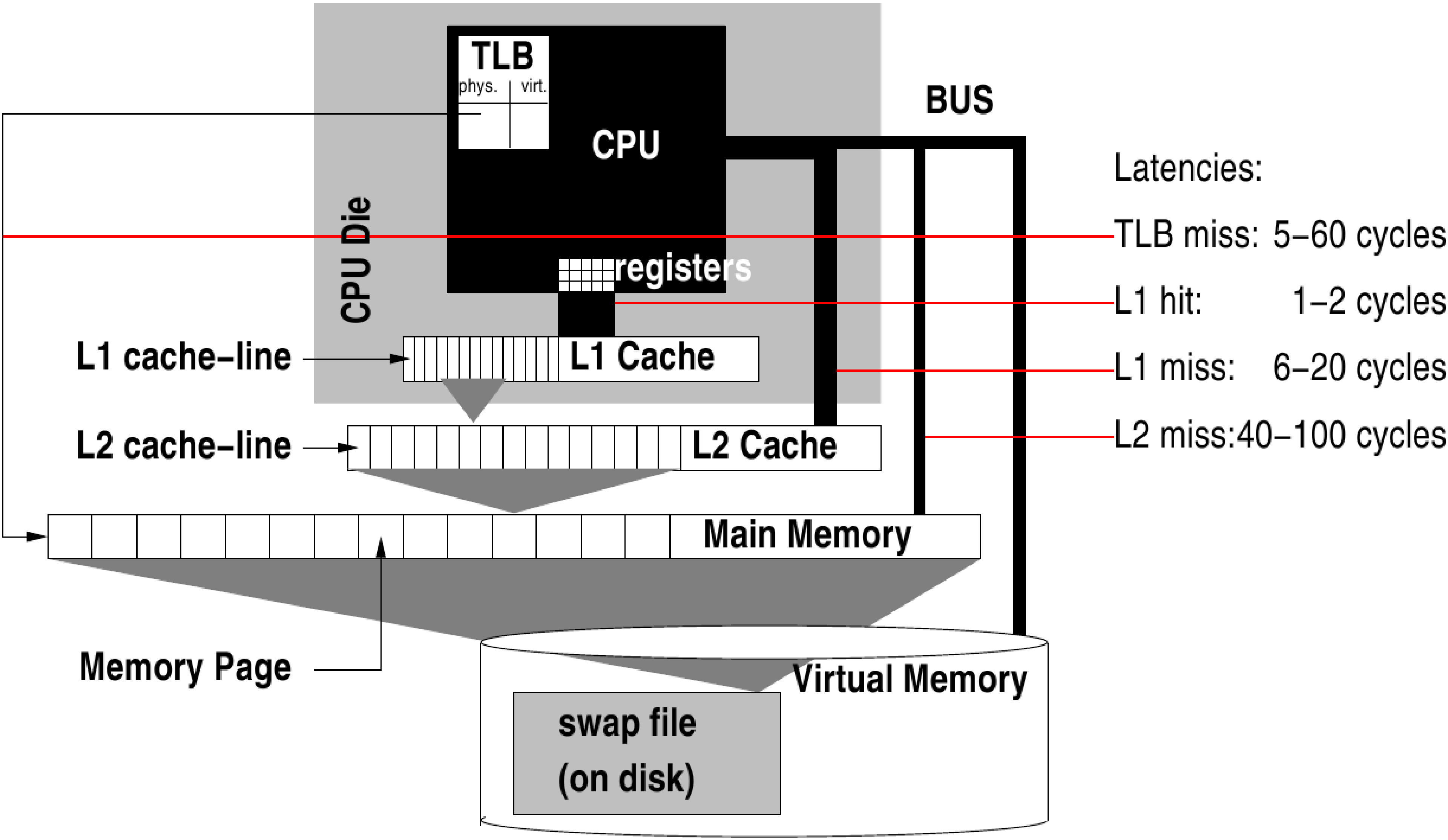
Elements:

- Storage
  - CPU caches L1/L2/L3
- Registers
- Execution Unit(s)
  - Pipelined
  - SIMD



Why?

Hardware Changes: Memory Hierarchies



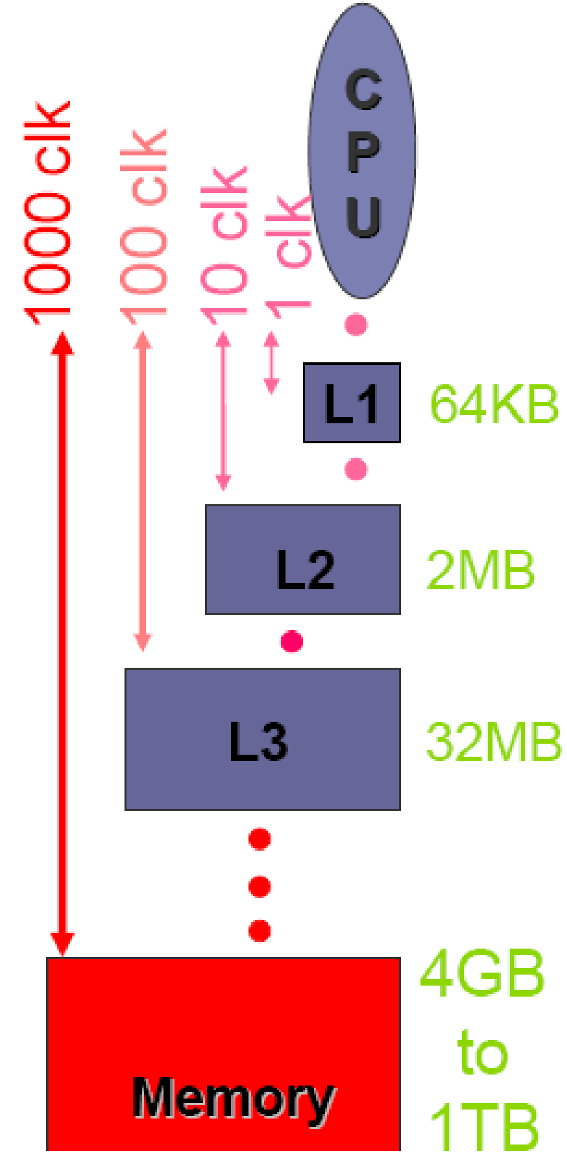
Why?

Hardware Changes: **Memory Hierarchies**

- Caches trade off capacity for speed
- Exploit instruction/data locality
- Demand fetch/wait for data

[ADH99]:

- Running top 4 database systems
- **At most 50% CPU utilization**



Why?

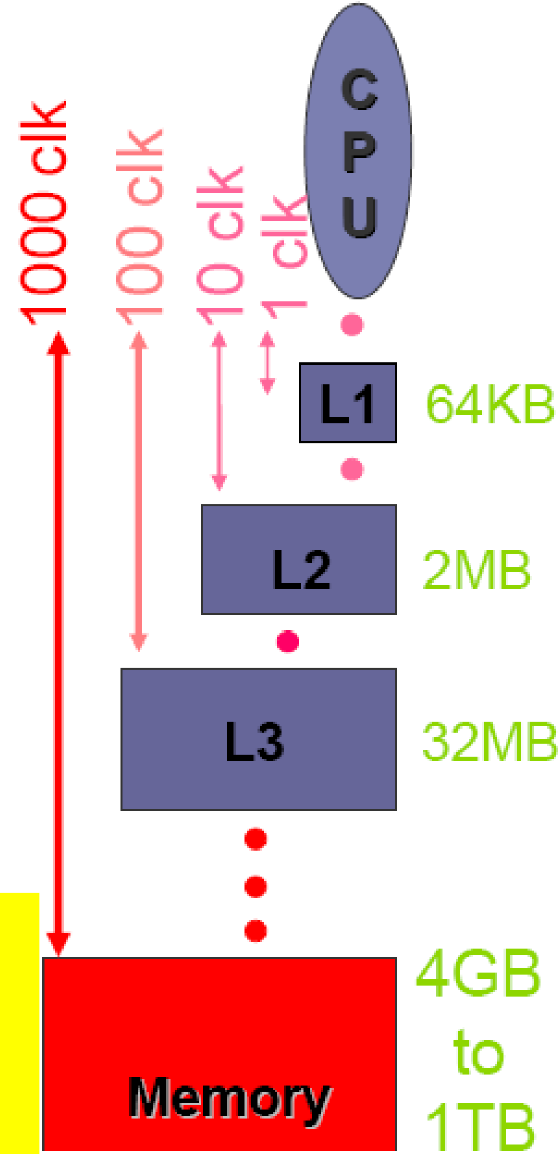
Hardware Changes: Memory Hierarchies

- Caches trade off capacity for speed
- Exploit instruction/data locality
- Demand fetch/wait for data

[ADH99]:

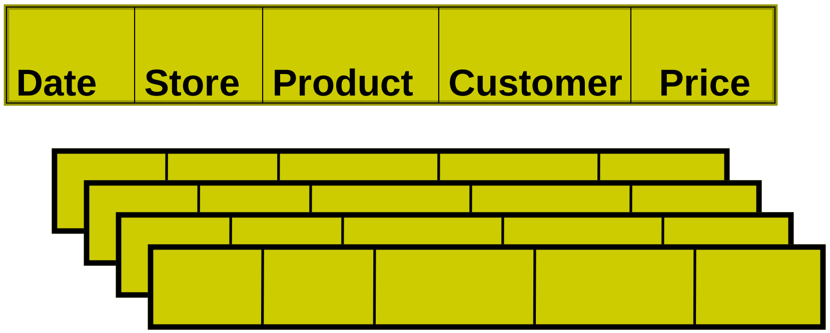
- Running top 4 database systems
- **At most 50% CPU utilization**

+Transition Lookaside Buffer (TLB)
Cache for VM address translation ➔
only 64 entries!



What is a column-store?

row-store

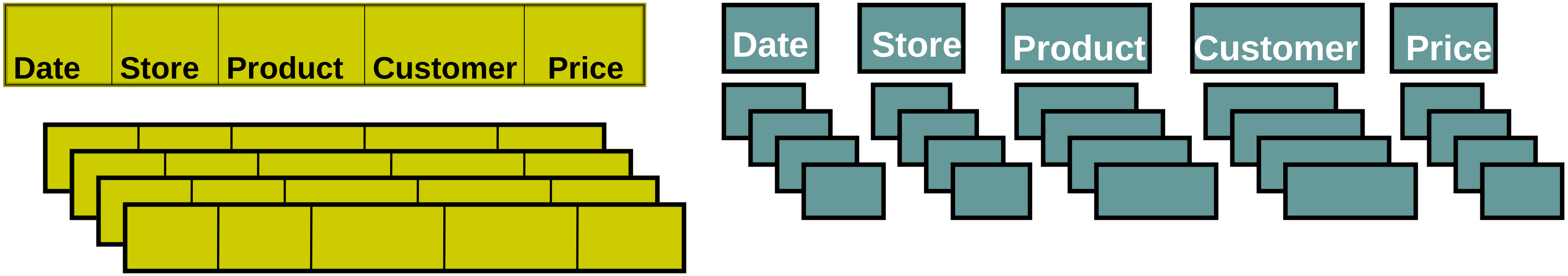


+ easy to add/modify a record

- might read in unnecessary data

What is a column-store?

row-store **column-store**



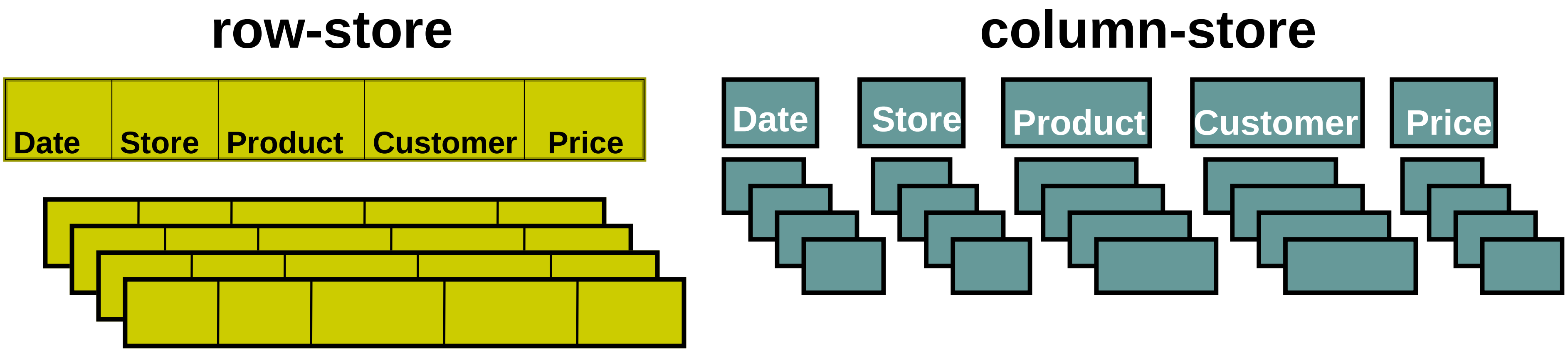
+ easy to add/modify a record

- might read in unnecessary data

+ only need to read in relevant data

- tuple writes require multiple accesses

What is a column-store?



+ easy to add/modify a record

- might read in unnecessary data

+ only need to read in relevant data

- tuple writes require multiple accesses

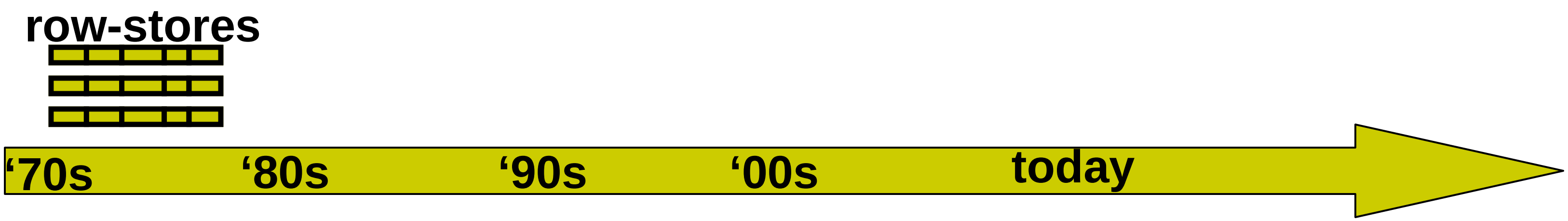
=> *suitable for read-mostly, read-intensive, large data repositories*

Are these two fundamentally different?

- The only fundamental difference is the storage layout
- However: we need to look at the big picture

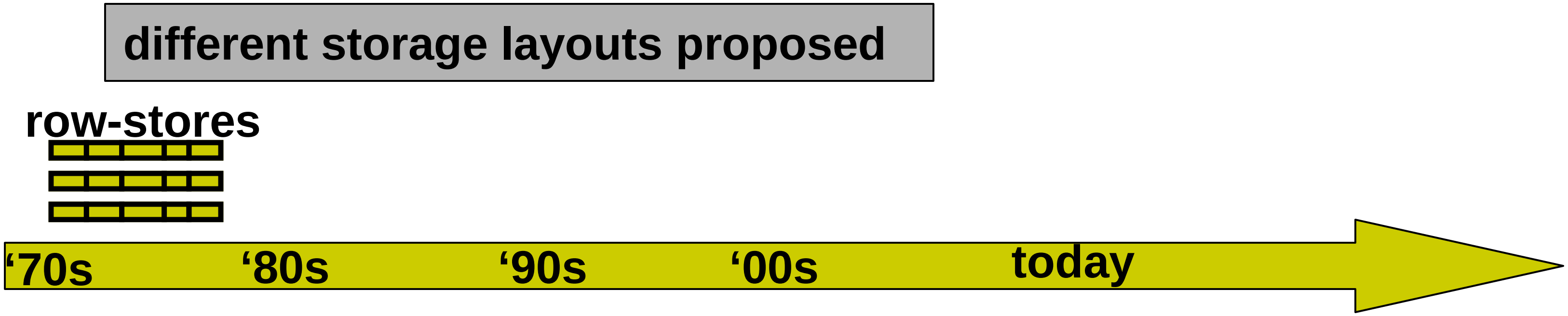
Are these two fundamentally different?

- The only fundamental difference is the storage layout
- However: we need to look at the big picture



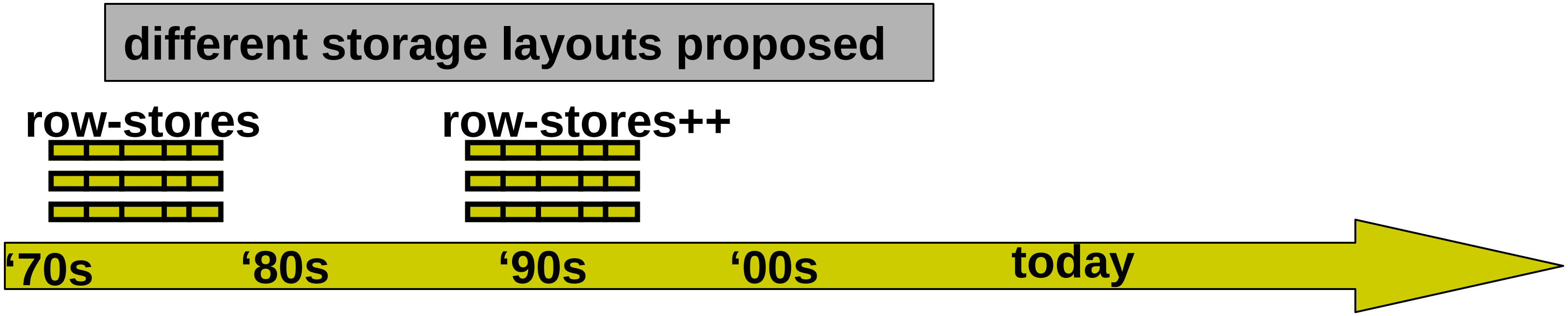
Are these two fundamentally different?

- The only fundamental difference is the storage layout
- However: we need to look at the big picture



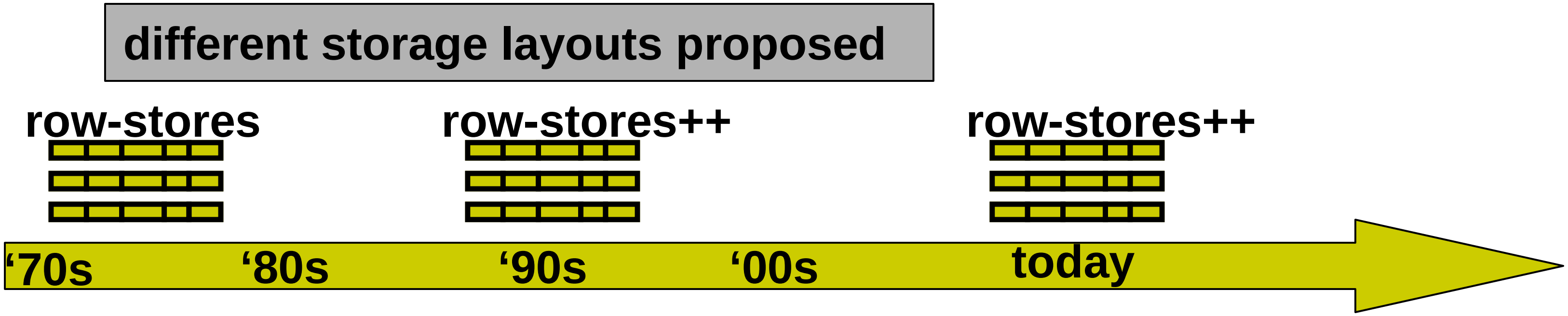
Are these two fundamentally different?

- The only fundamental difference is the storage layout
- However: we need to look at the big picture



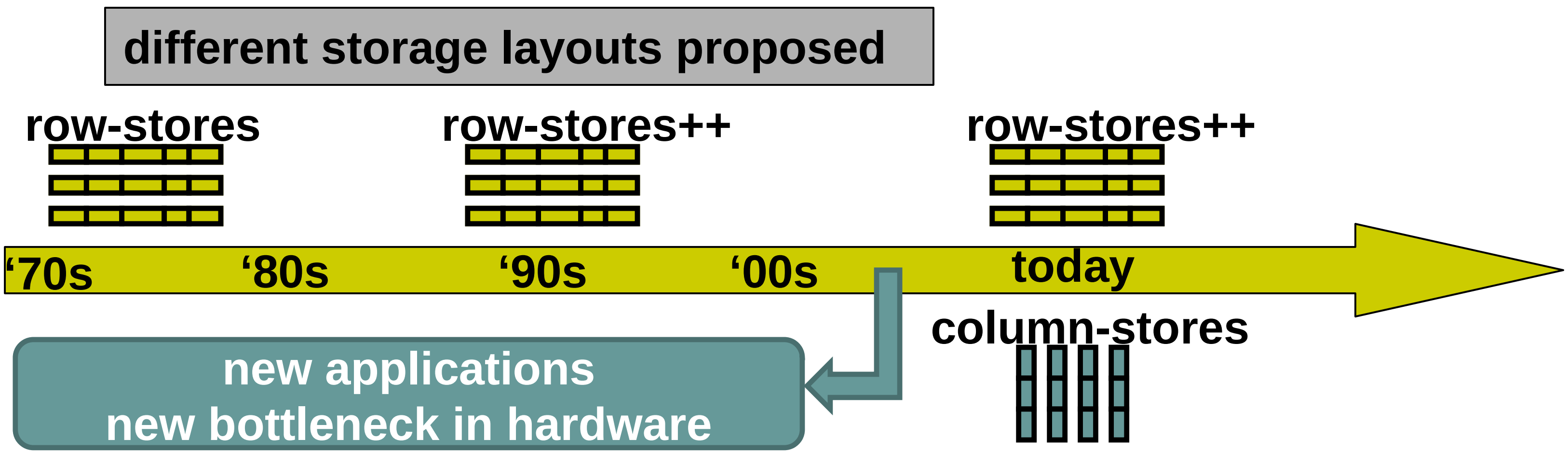
Are these two fundamentally different?

- The only fundamental difference is the storage layout
- However: we need to look at the big picture



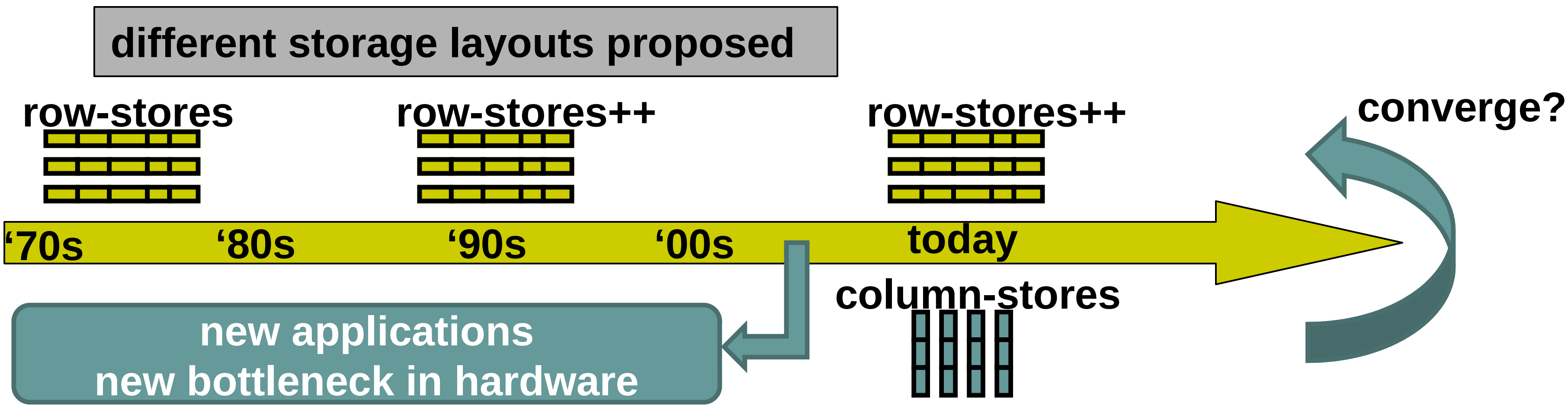
Are these two fundamentally different?

- The only fundamental difference is the storage layout
- However: we need to look at the big picture



Are these two fundamentally different?

- The only fundamental difference is the storage layout
- However: we need to look at the big picture

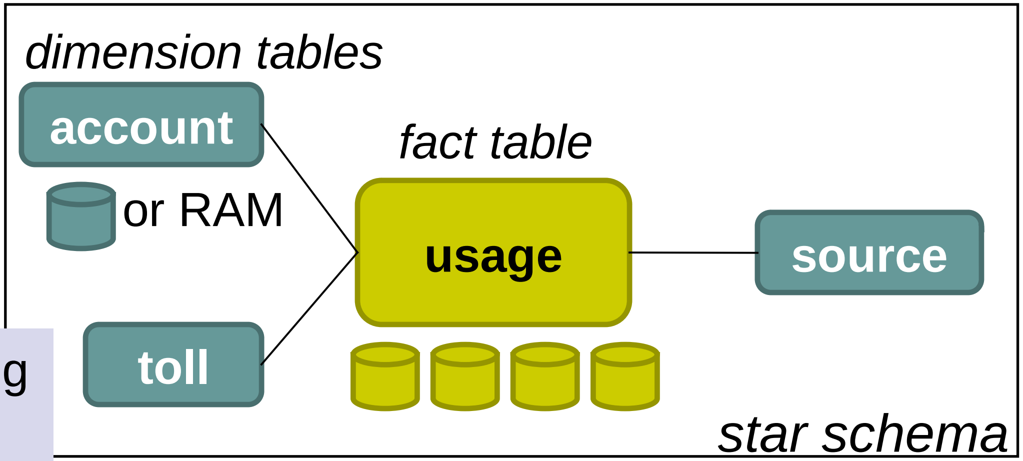


- How did we get here, and where we are heading?
- What are the column-specific optimizations?
- How do we improve CPU efficiency when operating on Cs?

Telco Data Warehousing example

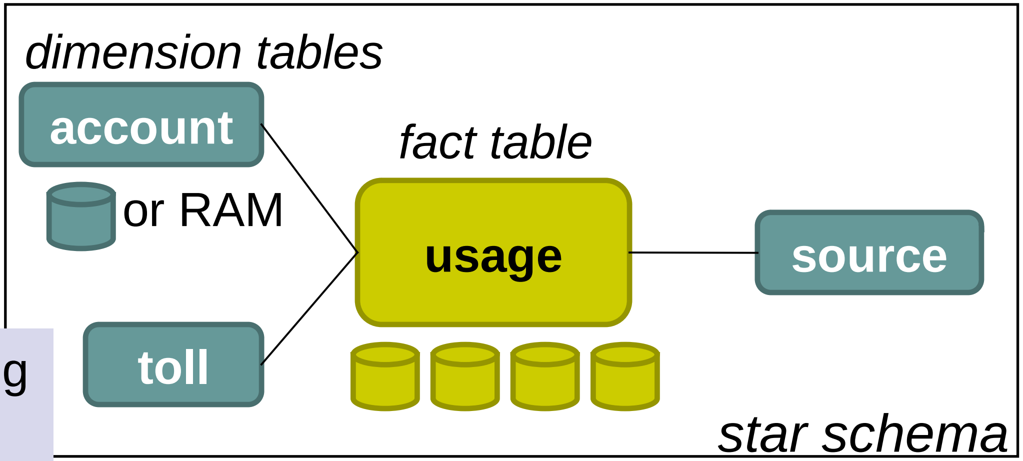
- Typical DW installation
- Real-world example

"One Size Fits All? - Part 2: Benchmarking Results" Stonebraker et al. CIDR 2007



Telco Data Warehousing example

- Typical DW installation
- Real-world example



"One Size Fits All? - Part 2: Benchmarking Results" Stonebraker et al. CIDR 2007

QUERY 2

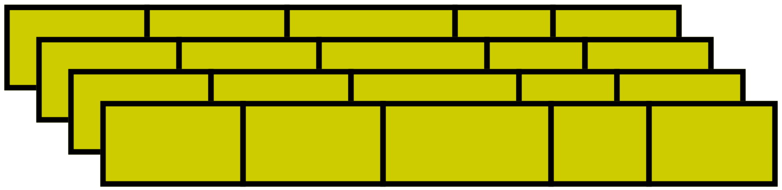
```
SELECT account.account_number,
sum (usage.toll_airtime),
sum (usage.toll_price)
FROM usage, toll, source, account
WHERE usage.toll_id = toll.toll_id
AND usage.source_id = source.source_id
AND usage.account_id = account.account_id
AND toll.type_ind in ('AE'. 'AA')
AND usage.toll_price > 0
AND source.type != 'CIBER'
AND toll.rating_method = 'IS'
AND usage.invoice_date = 20051013
GROUP BY account.account_number
```

	Column-store	*Row-store*
Query 1	*2.06*	*300*
Query 2	*2.20*	*300*
Query 3	*0.09*	*300*
Query 4	*5.24*	*300*
Query 5	*2.88*	*300*

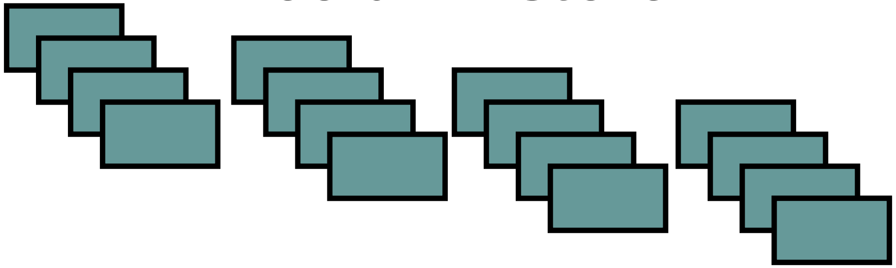
Why? Three main factors (next slides)

Telco example explained (1/3): *read efficiency*

row store



column store



read pages containing entire rows

one row = 212 columns!

is this typical? (it depends)

read only columns needed

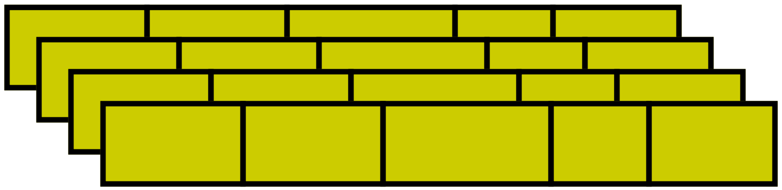
in this example: 7 columns

caveats:

- "select * " not any faster
- clever disk prefetching
- clever tuple reconstruction

Telco example explained (1/3): *read efficiency*

row store



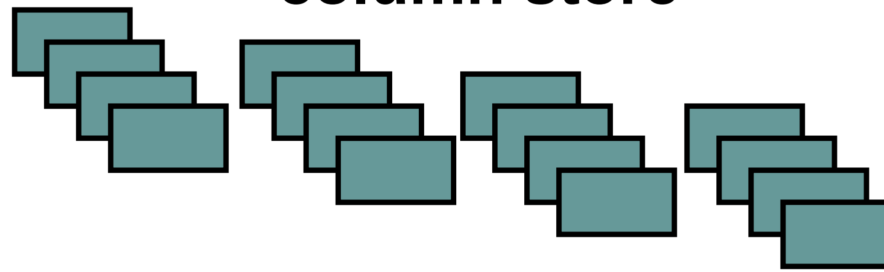
read pages containing entire rows

one row = 212 columns!

is this typical? (it depends)

What about vertical partitioning?
(it does not work with ad-hoc queries)

column store



read only columns needed

in this example: 7 columns

caveats:

- “select * ” not any faster
- clever disk prefetching
- clever tuple reconstruction

Telco example explained (2/3): *compression efficiency*

- Columns compress better than rows
  - Typical row-store compression ratio 1 : 3
  - Column-store 1 : 10
- Why?
  - Rows contain values from different domains
    => more entropy, difficult to dense-pack
  - Columns exhibit significantly less entropy
  - Examples:

Male, Female, Female, Female, Male
1998, 1998, 1999, 1999, 1999, 2000

  - Caveat: CPU cost (use lightweight compression)

Telco example explained (3/3): *sorting & indexing efficiency*

- Compression and dense-packing free up space
  - Use multiple overlapping column collections
  - Sorted columns compress better
  - Range queries are faster
  - Use sparse clustered indexes

What about heavily-indexed row-stores?
(works well for single column access,
cross-column joins become increasingly expensive)

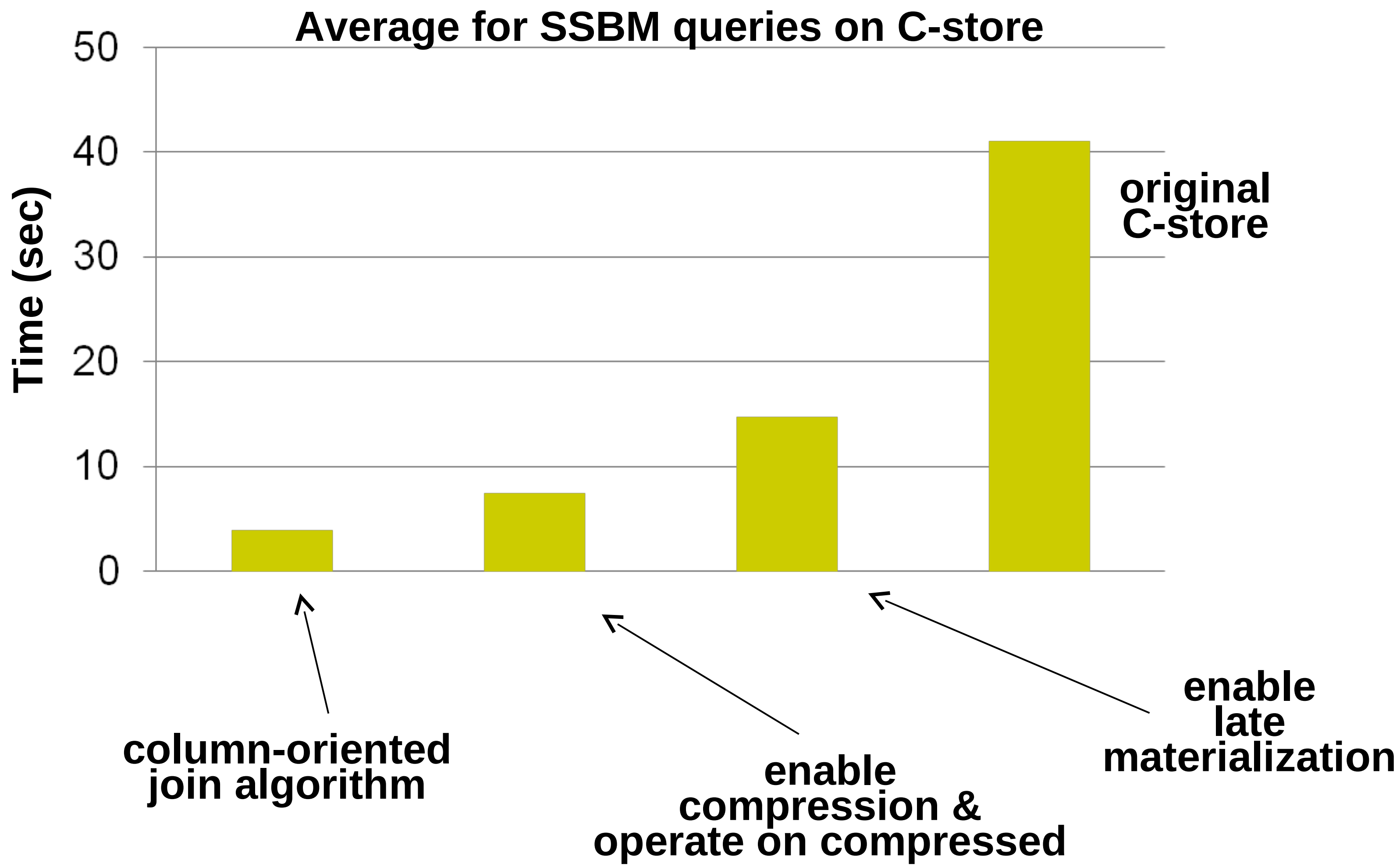
Additional opportunities for column-stores

- Block-tuple / vectorized processing
  - Easier to build block-tuple operators
    - Amortizes function-call cost, improves CPU cache performance
  - Easier to apply vectorized primitives
    - Software-based: bitwise operations
    - Hardware-based: SIMD
- Opportunities with compressed columns
  - *Avoid* decompression: operate directly on compressed
  - *Delay* decompression (and tuple reconstruction)
    - Also known as: *late materialization*
- Exploit columnar storage in other DBMS components
  - Physical design (both static and dynamic)

See: *Database Cracking*, from CWI

Effect on C-Store performance

"Column-Stores vs Row-Stores: How Different are They Really?" Abadi, Hachem, and Madden. SIGMOD 2008.



Summary of column-store key features

- Storage layout
  - columnar storage
  - header/ID elimination
  - compression
  - multiple sort orders
- Execution engine
  - column operators
  - avoid decompression
  - late materialization
  - vectorized operations
- Design tools, optimizer

From DSM to Column-stores

70s -1985:

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

1985: DSM paper

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

1990s: Commercialization through SybaseIQ

Late 90s – 2000s: Focus on main-memory performance

- DSM "on steroids" [1997 – now] — CWI: MonetDB
- Hybrid DSM/NSM [2001 – 2004] — Wisconsin: PAX, Fractured Mirrors

Michigan: Data Morphing

CMU: Clotho

2005 – : Re-birth of read-optimized DSM as "column-store"

MIT: C-Store

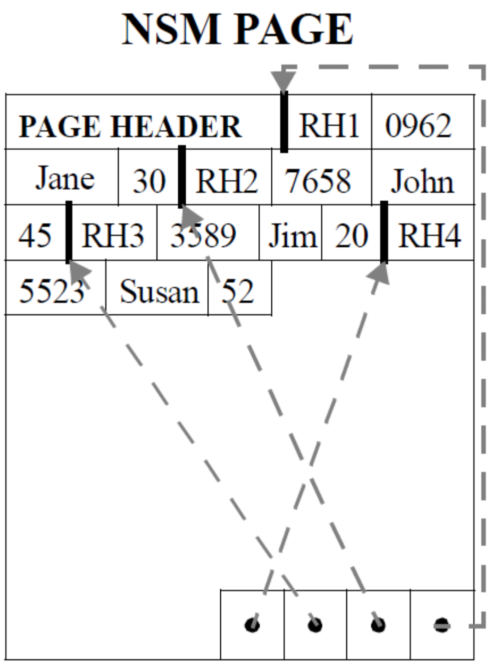
CWI: MonetDB/X100

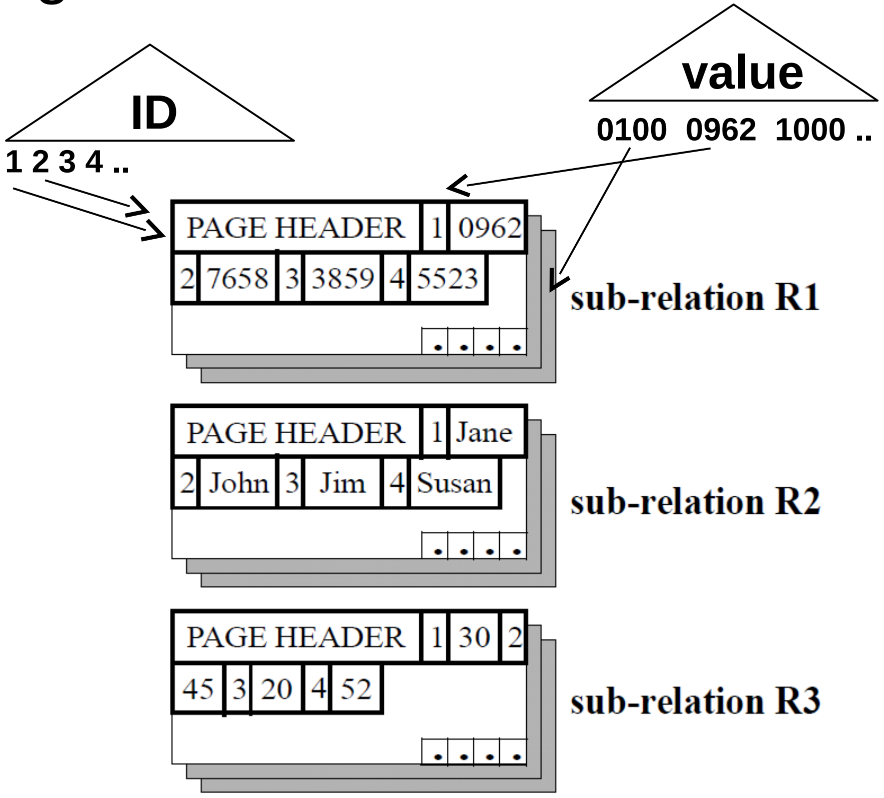
10+ startups

The original DSM paper

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

- Proposed as an alternative to NSM
- 2 indexes: clustered on ID, non-clustered on value
- Speeds up queries projecting few columns
- Requires more storage





Memory wall and PAX

- 90s: Cache-conscious research

from: "Cache Conscious Algorithms for Relational Query Processing." Shatdal, Kant, Naughton. VLDB 1994.

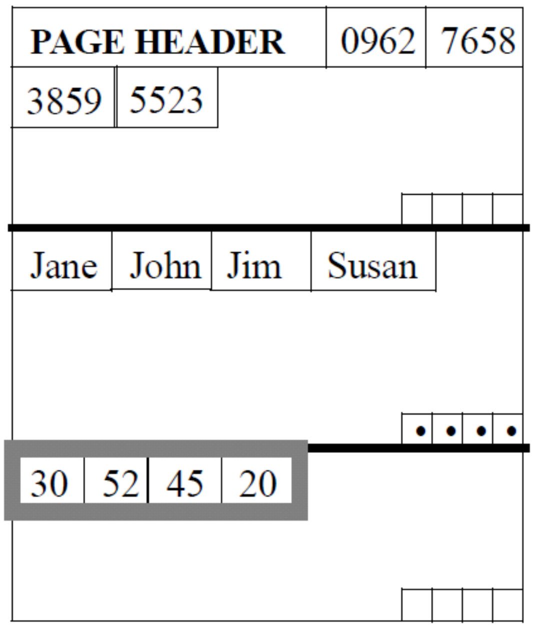
to: "Database Architecture Optimized for the New Bottleneck: Memory Access." Boncz, Manegold, Kersten. VLDB 1999.

and: "DBMSs on a modern processor: Where does time go?" Ailamaki, DeWitt, Hill, Wood. VLDB 1999.

- PAX: Partition Attributes Across
  - Retains NSM I/O pattern
  - Optimizes cache-to-RAM communication

"Weaving Relations for Cache Performance." Ailamaki, DeWitt, Hill, Skounakis, VLDB 2001.

PAX PAGE



More hybrid NSM/DSM schemes

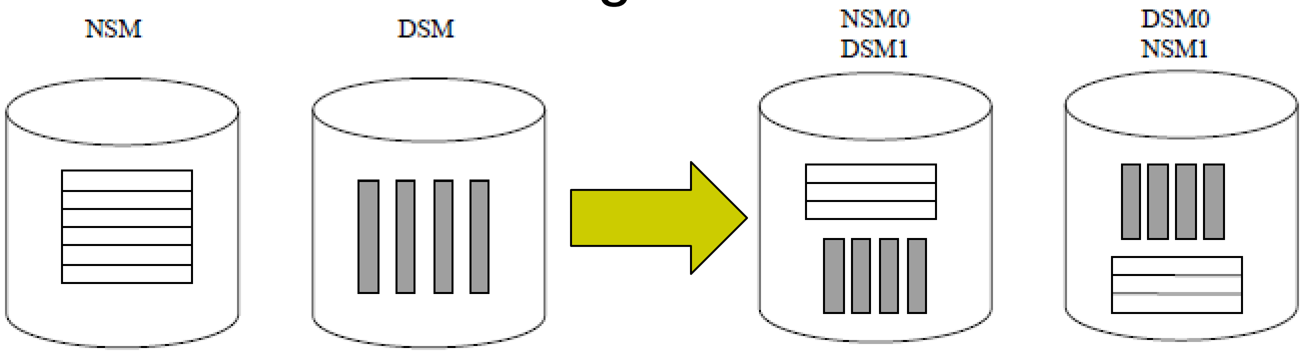
- Dynamic PAX: Data Morphing

  "Data morphing: an adaptive, cache-conscious storage technique." Hankins, Patel, VLDB 2003.

- Clotho: custom layout using scatter-gather I/O

  "Clotho: Decoupling Memory Page Layout from Storage Organization." Shao, Schindler, Schlosser, Ailamaki, and Ganger. VLDB 2004.

- Fractured mirrors
  - Smart mirroring with both NSM/DSM copies



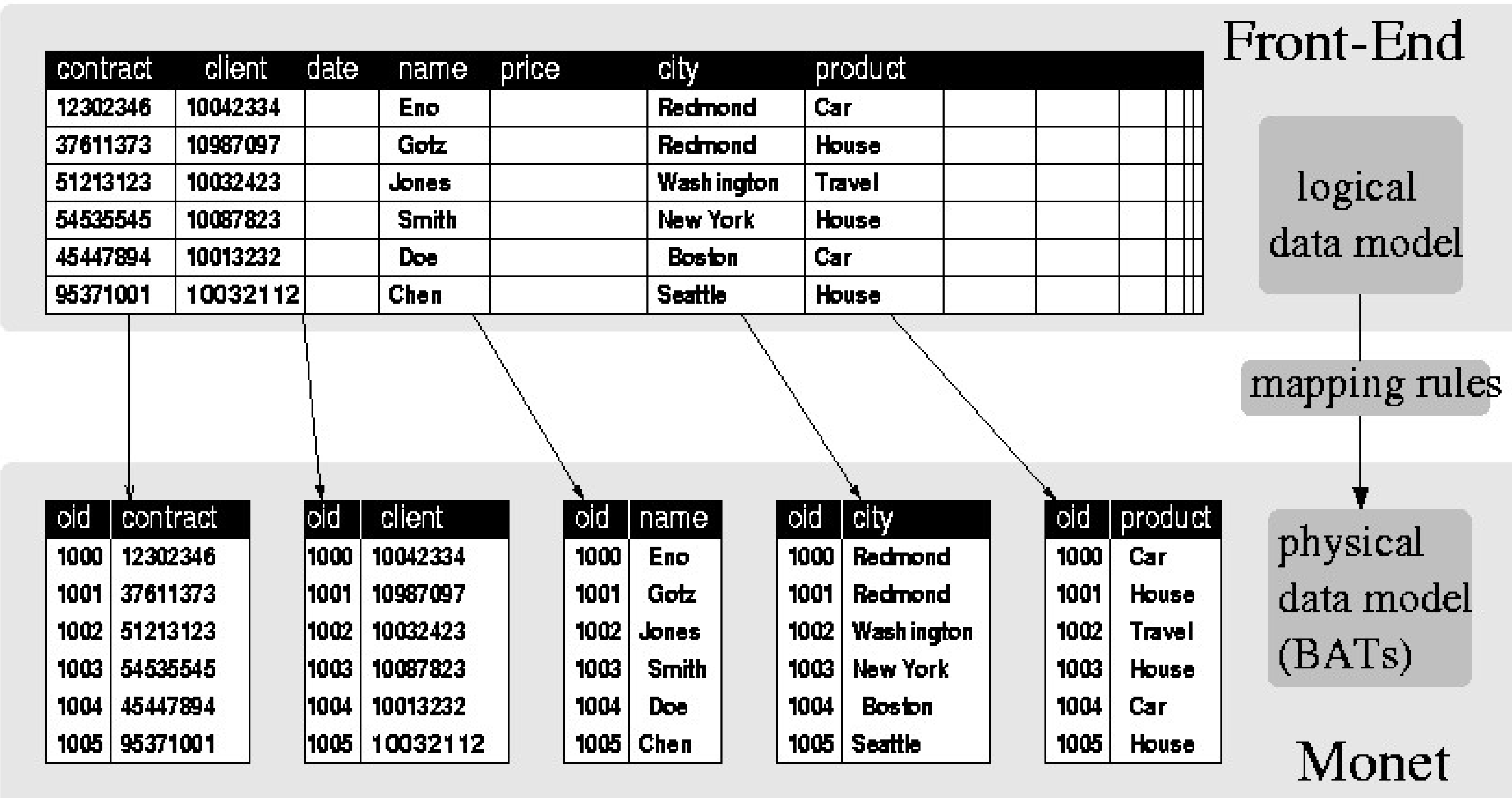
"A Case For Fractured Mirrors." Ramamurthy, DeWitt, Su, VLDB 2002.

MonetDB

- Late 1990s, CWI: Boncz, Manegold, and Kersten
- Motivation:
  - Main-memory
  - Improve computational efficiency by avoiding expression interpreter
  - DSM with virtual IDs natural choice
  - Developed new query execution algebra
- Initial contributions:
  - Pointed out memory-wall in DBMSs
  - Cache-conscious projections and joins
  - …

Storing Relations in MonetDB

DSM => Column-store



contract	client	date	name	price	city	product
12302346	10042334		Eno		Redmond	Car
37611373	10987097		Gotz		Redmond	House
51213123	10032423		Jones		Washington	Travel
54535545	10087823		Smith		New York	House
45447894	10013232		Doe		Boston	Car
95371001	10032112		Chen		Seattle	House

oid	contract
1000	12302346
1001	37611373
1002	51213123
1003	54535545
1004	45447894
1005	95371001

oid	client
1000	10042334
1001	10987097
1002	10032423
1003	10087823
1004	10013232
1005	10032112

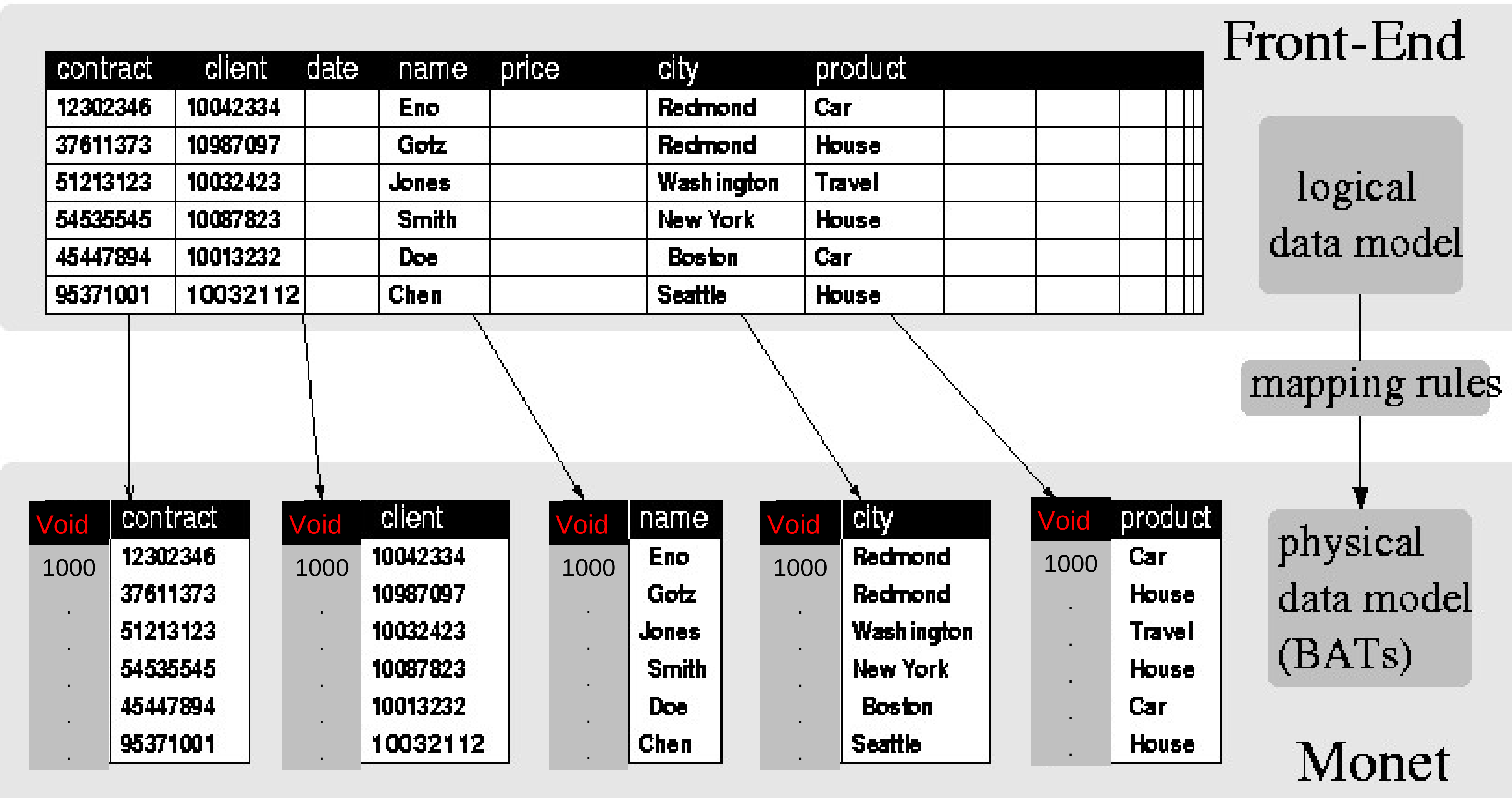
oid	name
1000	Eno
1001	Gotz
1002	Jones
1003	Smith
1004	Doe
1005	Chen

oid	city
1000	Redmond
1001	Redmond
1002	Washington
1003	New York
1004	Boston
1005	Seattle

oid	product
1000	Car
1001	House
1002	Travel
1003	House
1004	Car
1005	House

Storing Relations in MonetDB

DSM => Column-store



Virtual OID: seqbase=**1000** (increment=1)

2005: the (re)birth of column-stores

- New hardware and application realities
  - Faster CPUs, larger memories, disk bandwidth limit
  - Multi-terabyte Data Warehouses
- New approach: combine several techniques
  - Read-optimized, fast multi-column access, disk/CPU efficiency, light-weight compression
- C-store paper:
  - First comprehensive design description of a column-store
- MonetDB/X100
  - “proper” disk-based column store
- Explosion of new products

Column-Store History / Time-line

- 1985: ***DSM*** (Copeland et al.; SIGMOD 1985)
- 1992: First ideas and kernel for *MonetDB* (Kersten)
- 1993: *MonetDB* is born
- 1993: KDB (first commercial DSM system (?))
- 1995: Sybase IQ
- 2002: *MonetDB* goes open-source
- 2004[?]: Stonebraker et al. start “C-Store” project and coin DSM as “***Column-Store***”
- 2006[?]: Stonebraker et al. found “Vertica”; end of “C-Store” as research project
- 2008: Zukowski, Boncz, et al. (CWI) found VectorWise (based on *MonetDB*/X100)
- 2010: INGRES (now called Actian) acquires VectorWise
- 2011: HP acquires Vertica
- 201?: SAP HANA, IBM BLINK -> ISAO -> BLU, Oracle Database In-Memory
  Microsoft SQL Server Column-store indexes (“Apollo”), …
- 2018: Raasveldt & Mühleisen (CWI) start developing *DuckDB* as embeddable
  analytical DBMS (based on columnar storage & vectorized execution):
  “*SQLlite for analytics*”

Performance tradeoffs: columns vs. rows

DSM traditionally was not favored by technology trends
How has this changed?

- Optimized DSM in "Fractured Mirrors," 2002
- "Apples-to-apples" comparison

"Performance Tradeoffs in Read-Optimized Databases" Harizopoulos, Liang, Abadi, Madden, VLDB'06

- Follow-up study

"Read-Optimized Databases, In-Depth" Holloway, DeWitt, VLDB'08

- Main-memory DSM vs. NSM

"DSM vs. NSM: CPU performance tradeoffs in block-oriented query processing" Boncz, Zukowski, Nes, DaMoN'08

- Flash-disks: a come-back for PAX?

"Fast Scans and Joins Using Flash Drives" Shah, Harizopoulos, Wiener, Graefe. DaMoN'08

"Query Processing Techniques for Solid State Drives" Tsirogiannis, Harizopoulos, Shah, Wiener, Graefe, SIGMOD'09

Applications for column-stores

- Data Warehousing
  - High end (clustering)
  - Mid end/Mass Market
  - Personal Analytics
- Data Mining
  - E.g. Proximity
- Google BigTable
- RDF
  - Semantic web data management
- Information retrieval
  - Terabyte TREC
- Scientific datasets
  - SciDB initiative
  - SLOAN Digital Sky Survey on MonetDB

List of column-store systems

- Cantor (history)
- Sybase IQ
- SenSage (former Addamark Technologies)
- Kdb
- 1010data
- MonetDB
- C-Store/Vertica
- X100/VectorWise
- KickFire
- SAP Business Accelerator, SAP HANA
- Infobright
- ParAccel
- Exasol

Simulate a Column-Store inside a Row-Store

Date	Store	Product	Customer	Price
01/01	BOS	Table	Mesa	$20
01/01	NYC	Chair	Lutz	$13
01/01	BOS	Bed	Mudd	$79

Option A: Vertical Partitioning

Date

TID	Value
1	01/01
2	01/01
3	01/01

Store

TID	Value
1	BOS
2	NYC
3	BOS

Product

TID	Value
1	Table
2	Chair
3	Bed

Customer

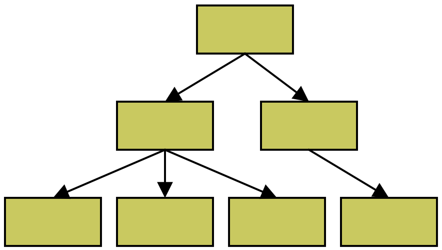
TID	Value
1	Mesa
2	Lutz
3	Mudd

Price

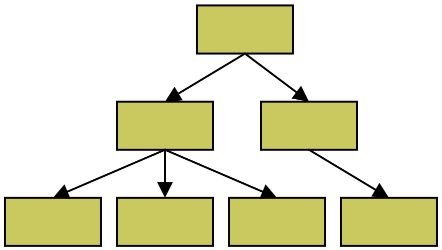
TID	Value
1	$20
2	$13
3	$79

Option B: Index Every Column

Date Index



Store Index



...

Simulate a Column-Store inside a Row-Store

Date	Store	Product	Customer	Price
01/01	BOS	Table	Mesa	$20
01/01	NYC	Chair	Lutz	$13
01/01	BOS	Bed	Mudd	$79

Option A: Vertical Partitioning

Date

Value	StartPos	Length
01/01	1	3

Store

TID	Value
1	BOS
2	NYC
3	BOS

Product

TID	Value
1	Table
2	Chair
3	Bed

Customer

TID	Value
1	Mesa
2	Lutz
3	Mudd

Price

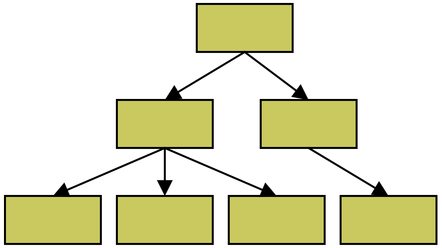
TID	Value
1	$20
2	$13
3	$79

Can explicitly run-length encode date

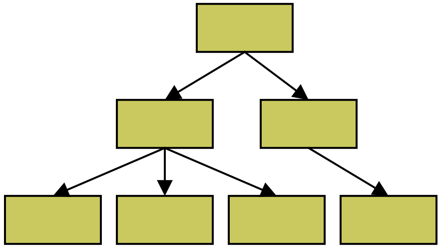
"Teaching an Old Elephant New Tricks." Bruno, CIDR 2009.

Option B: Index Every Column

Date Index



Store Index



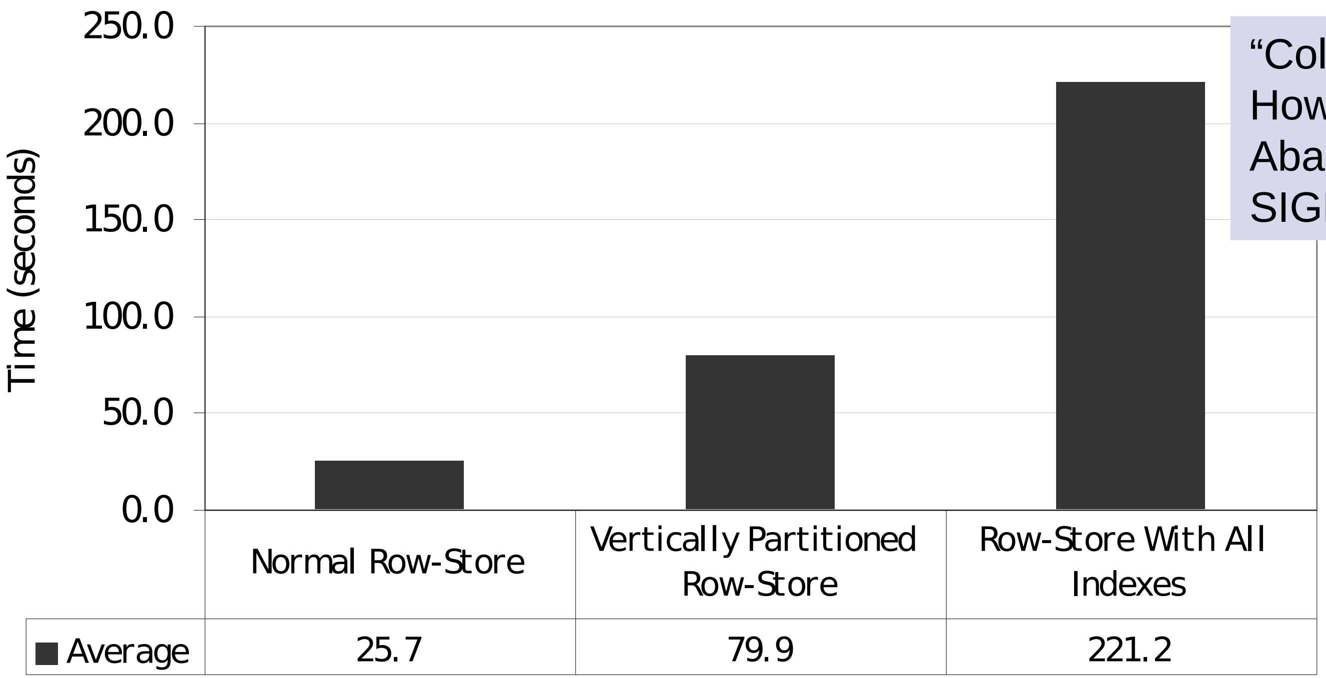
...

Experiments

- Star Schema Benchmark (SSBM)

Adjoined Dimension Column Index (ADC Index) to Improve Star Schema Query Performance”. O’Neil et. al. ICDE 2008.

- Implemented by professional DBA
- Original row-store plus 2 column-store simulations on same row-store product



“Column-Stores vs Row-Stores: How Different are They Really?” Abadi, Hachem, and Madden. SIGMOD 2008.

What's Going On? Vertical Partitions

- Vertical partitions in row-stores:
  - Work well when workload is known
  - ..and queries access disjoint sets of columns
  - See automated physical design

Tuple Header	TID	Column Data
	1	
	2	
	3	

- Do not work well as full-columns
  - TupleID overhead significant
  - Excessive joins

Queries touch 3-4 foreign keys in fact table, 1-2 numeric columns

Complete fact table takes up ~4 GB (compressed)

Vertically partitioned tables take up 0.7-1.1 GB (compressed)

"Column-Stores vs. Row-Stores: How Different Are They Really?" Abadi, Madden, and Hachem. SIGMOD 2008.

What's Going On? All Indexes Case

- Tuple construction
  - Common type of query:

```
SELECT store_name, SUM(revenue)
FROM Facts, Stores
WHERE fact.store_id = stores.store_id
    AND stores.country = "Canada"
GROUP BY store_name
```

  - 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

So….

- All indexes approach is 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
  - Combine clustered indexes, vertical partitioning
- So can row-stores simulate column-stores?

So….

- All indexes approach is 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
  - Combine clustered indexes, vertical partitioning
- So can row-stores simulate column-stores?
  - Might be possible, BUT:
    - Need better support for vertical partitioning at the storage layer
    - Need support for column-specific optimizations at the executer level
    - Full integration: buffer pool, transaction manager, ...

ADM: Literature (1/2)

- **Column-Oriented Database Systems (1/6) - Motivation & Basic Concepts**
  - “An overview of cantor: a new system for data analysis“. Ilkka Karasalo, Per Svensson. SSDBM 1983.
  - “A decomposition storage model“. George P. Copeland, Setrag Khoshafian. SIGMOD Conference, 1985.
  - “Cache Conscious Algorithms for Relational Query Processing“. Ambuj Shatdal, Chander Kant, Jeffrey F. Naughton. VLDB 1994.
  - “MIL Primitives for Querying a Fragmented World“. Peter A. Boncz, Martin L. Kersten. VLDB J. 8(2): 101-119, 1999.
  - “Database Architecture Optimized for the New Bottleneck: Memory Access“. Peter A. Boncz, Stefan Manegold, Martin L. Kersten. VLDB 1999.
  - “DBMSs On A Modern Processor: Where Does Time Go?“. Anastassia Ailamaki, David J. DeWitt, Mark D. Hill, David A. Wood. VLDB 1999.
  - “Weaving Relations for Cache Performance ("PAX")“. Anastassia Ailamaki, David J. DeWitt, Mark D. Hill, Marios Skounakis. VLDB 2001.
  - “A Case for Fractured Mirrors“. Ravishankar Ramamurthy, David J. DeWitt, Qi Su. VLDB 2002.
  - “Data Morphing: An Adaptive, Cache-Conscious Storage Technique“. Richard A. Hankins, Jignesh M. Patel. VLDB 2003.
  - “Clotho: Decoupling Memory Page Layout from Storage Organization“. Minglong Shao, Jiri Schindler, Steven W. Schlosser, Anastassia Ailamaki, Gregory R. Ganger. VLDB 2004.
  - “MonetDB-X100 - A DBMS In The CPU Cache“. Marcin Zukowski, Peter A. Boncz, Niels Nes, Sándor Héman. IEEE Data Eng. Bull. 28(2): 17-22, 2005.
  - “"One size fits all": an idea whose time has come and gone“. Michael Stonebraker, Ugur Çetintemel. ICDE 2005.
  - “Performance Tradeoffs in Read-Optimized Databases“. Stavros Harizopoulos, Velen Liang, Daniel J. Abadi, Samuel Madden. VLDB 2006.
  - ...

ADM: Literature (2/2)

- **Column-Oriented Database Systems (1/6) - Motivation & Basic Concepts** *(cont.)*
  - ...
  - “One Size Fits All? - Part 2: Benchmarking Results“. Michael Stonebraker, Chuck Bear, Ugur Çetintemel, Mitch Cherniack, Tingjian Ge, Nabil Hachem, Stavros Harizopoulos, John Lifter, Jennie Rogers, Stanley B. Zdonik. CIDR 2007.
  - “C-Store: A Column-oriented DBMS“. Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel Madden, Elizabeth J. O'Neil, Patrick E. O'Neil, Alex Rasin, Nga Tran, Stanley B. Zdonik. VLDB 2005.
  - “Breaking the memory wall in MonetDB“. Peter A. Boncz, Martin L. Kersten, Stefan Manegold. Commun. ACM 51(12): 77-85, 2008.
  - “Column-Stores vs Row-Stores: How Different are They Really?“. Daniel J. Abadi, Samuel Madden, Nabil Hachem. SIGMOD Conference 2008.
  - “DSM vs. NSM: CPU performance tradeoffs in block-oriented query processing“. Marcin Zukowski, Niels Nes, Peter A. Boncz. DaMoN 2008.
  - “Fast Scans and Joins Using Flash Drives“. Mehul A. Shah, Stavros Harizopoulos, Janet L. Wiener, Goetz Graefe. DaMoN 2008.
  - “Read-Optimized Databases, In-Depth“. Allison L. Holloway, David J. DeWitt. Proc. VLDB Endow. 1(1): 502-513, 2008.
  - “Teaching an Old Elephant New Tricks“. Nicolas Bruno. CIDR 2009.
  - “Query Processing Techniques for Solid State Drives“. Dimitris Tsirogiannis, Stavros Harizopoulos, Mehul A. Shah, Janet L. Wiener, Goetz Graefe. SIGMOD Conference 2009.
  - “MonetDB: Two Decades of Research in Column-oriented Database Architectures“. Stratos Idreos, Fabian Groffen, Niels Nes, Stefan Manegold, K. Sjoerd Mullender, Martin L. Kersten. IEEE Data Eng. Bull. 35(1): 40-45, 2012.
  - “The Vertica Analytic Database: C-Store 7 Years Later“. Andrew Lamb, Matt Fuller, Ramakrishna Varadarajan, Nga Tran, Ben Vandiver, Lyric Doshi, Chuck Bear. Proc. VLDB Endow. 5(12): 1790-1801, 2012.

ADM: Agenda

- 07.09.2022: Lecture 1: **Introduction**
- 14.09.2022: Lecture 2: **SQL Recap**

  (plus Assignment 1 [in groups; 3 weeks]: TPC-H benchmark)
- 21.09.2022: Lecture 3: **Column-Oriented Database Systems (1/6) - Motivation & Basic Concepts**
- 28.09.2022: Lecture 4: **Column-Oriented Database Systems (2a/6) - Selected Execution Techniques (1/2)**
- 05.10.2022: Lecture 5: **Column-Oriented Database Systems (2b/6) - Selected Execution Techniques (2/2)**

  (plus Assignment 3 [in groups; 3 weeks]: Compression techniques)
- 12.10.2022: Lecture 6: **Column-Oriented Database Systems (3/6) - Cache Conscious Joins**
- 19.10.2022: Lecture 7: **Column-Oriented Database Systems (4/6) - "Vectorized Execution"**
- ~~26.10.2022~~: ***No lecture!***
- 02.11.2022: Lecture 8: **DuckDB: An embedded database for data science (1/2) (guest lecture & *hands-on*)**

  (plus Assignment 2 [individual; 2 weeks]: Analysing NYC Cab dataset with DuckDB)
- 09.11.2022: Lecture 9: **DuckDB: An embedded database for data science (2/2) (guest lecture & *hands-on*)**
- 16.11.2022: Lecture 10: **Branch Misprediction & Predication**

  (plus Assignment 4 [individual; 2 weeks]: Predication)
- 23.11.2022: Lecture 11: **Column-Oriented Database Systems (5/6) - Adaptive Indexing**
- 30.11.2022: Lecture 12: **Column-Oriented Database Systems (6/6) - Progressive Indexing**

ADM: Literature

- **Column-Oriented Database Systems (2/6) - Selected Execution Techniques**
  - <u>Compression</u>
    - “Compressing Relations and Indexes”. Goldstein, Ramakrishnan, Shaft. ICDE’98.
    - “Query optimization in compressed database systems”. Chen, Gehrke, Korn. SIGMOD’01.
    - “Super-Scalar RAM-CPU Cache Compression”. Zukowski, Heman, Nes, Boncz. ICDE’06.
    - “Integrating Compression and Execution in Column-Oriented Database Systems”. Abadi, Madden, Ferreira. SIGMOD’06.
    - “Improved Word-Aligned Binary Compression for Text Indexing”. Ahn, Moffat. TKDE’06.
  - <u>Tuple Materialization</u>
    - “Materialization Strategies in a Column-Oriented DBMS”. Abadi, Myers, DeWitt, Madden. ICDE’07.
    - “Column-Stores vs Row-Stores: How Different are They Really?”. Abadi, Madden, Hachem. SIGMOD’08.
    - “Query Processing Techniques for Solid State Drives”. Tsirogiannis, Harizopoulos Shah, Wiener, Graefe. SIGMOD’09.
    - “Self-organizing tuple reconstruction in column-stores“. Idreos, Manegold, Kersten. SIGMOD’09.
  - <u>Join</u>
    - “Fast Joins using Join Indices”. Li and Ross. VLDBJ 8:1-24, 1999.