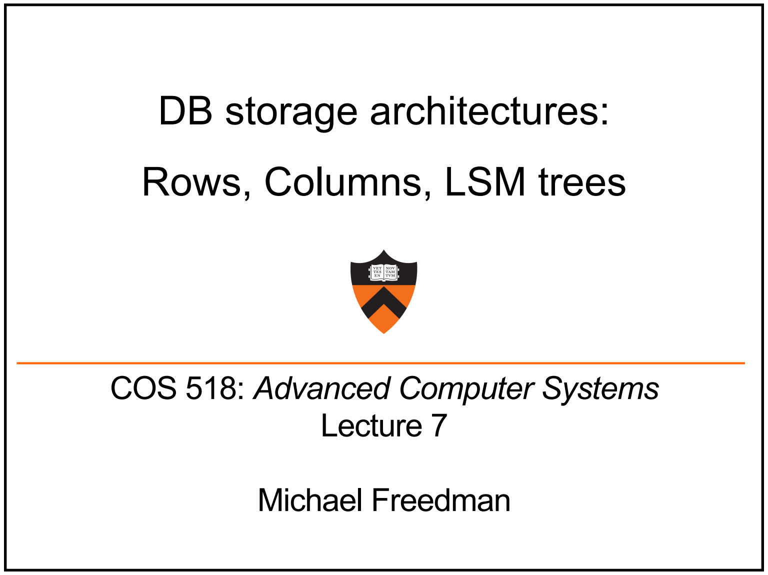
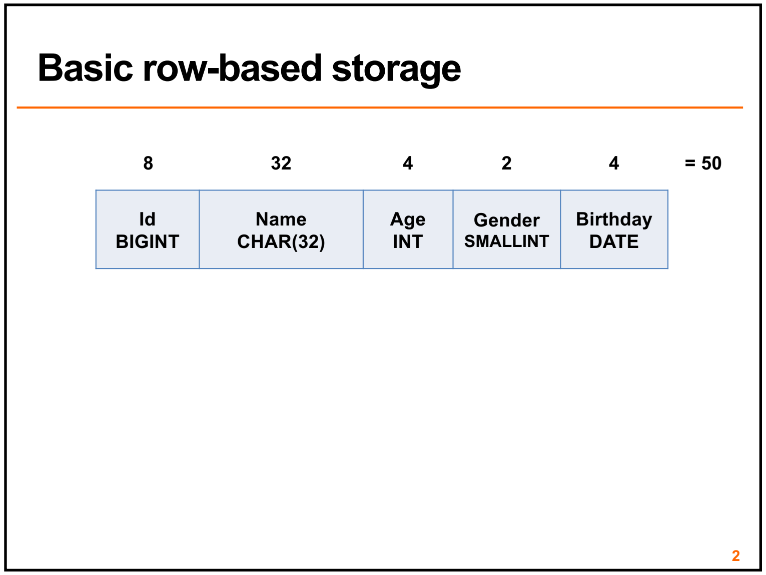


DB storage architectures:

Rows, Columns, LSM trees

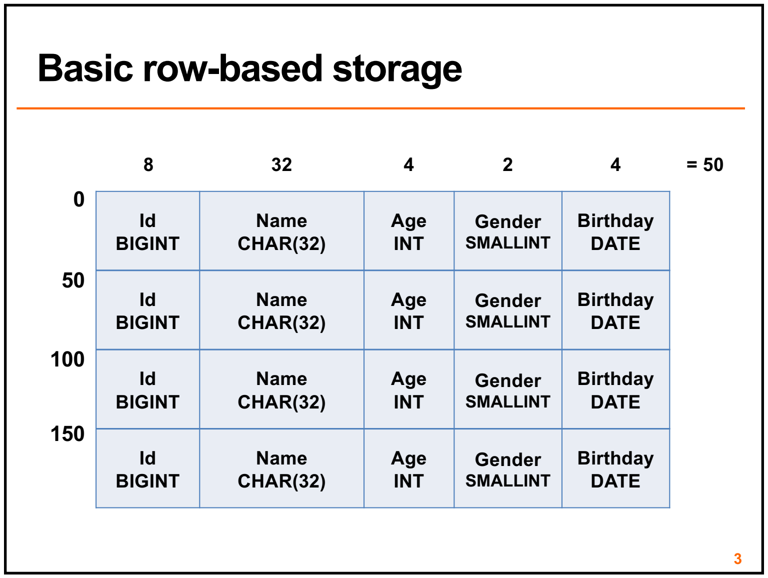
COS 518: *Advanced Computer Systems*
Lecture 7

Michael Freedman



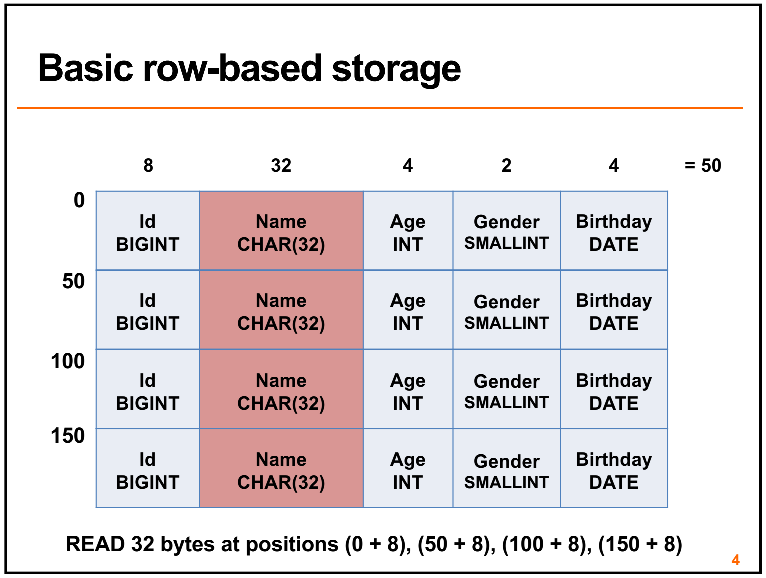
Basic row-based storage

8	32	4	2	4	= 50
Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	



Basic row-based storage

	8	32	4	2	4	= 50
0	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	
50	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	
100	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	
150	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	



Basic row-based storage

	8	32	4	2	4	= 50
0	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	
50	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	
100	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	
150	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	

READ 32 bytes at positions (0 + 8), (50 + 8), (100 + 8), (150 + 8)

Row-based storage: variable lengths

8	0 - 255	4	2	4	= 18 - 273
Id BIGINT	URL VARCHAR(255)	Size INT	Code SMALLINT	Fetched DATE	

How do you walk through all the URLs?
No longer at fixed offsets

Row-based storage: variable lengths

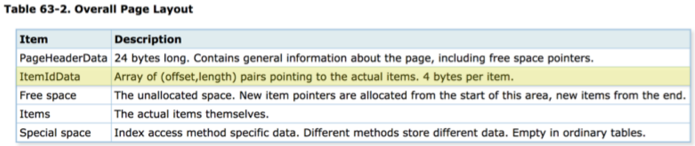


Table 63-2. Overall Page Layout

Item	Description
PageHeaderData	24 bytes long. Contains general information about the page, including free space pointers.
ItemIdData	Array of (offset,length) pairs pointing to the actual items. 4 bytes per item.
Free space	The unallocated space. New item pointers are allocated from the start of this area, new items from the end.
Items	The actual items themselves.
Special space	Index access method specific data. Different methods store different data. Empty in ordinary tables.

https://www.postgresql.org/docs/9.5/static/storage-page-layout.html

ItemIdData: [(0, 18), (18, 273), (291, 59)]

0	Id BIGINT	URL VARCHAR(255)	Size INT	Code SMALLINT	Fetched DATE
18	Id BIGINT	URL VARCHAR(255)	Size INT	Code SMALLINT	Fetched DATE
291	Id BIGINT	URL VARCHAR(255)	Size INT	Code SMALLINT	Fetched DATE

Row-based disk layout

- Data stored in fixed-sized pages on disk
  - E.g., typically 8K in PostgreSQL
  - Page includes metadata and actual data items
  - Items = indexes, data rows

Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	Id BIGINT	Na CHA

Row-based disk layout

- Data stored in fixed-sized pages on disk
  - E.g., typically 8K in PostgreSQL
  - Page includes metadata and actual data items
  - Items = indexes, data rows

Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	Id BIGINT	Name CHAR(32)	Age INT	Gender SMALLINT	Birthday DATE	Id BIGINT	Na CHA

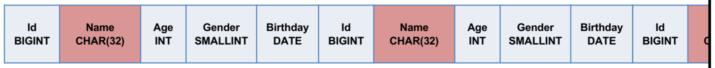
READ 32 bytes at positions (0 + 8), (50 + 8), (100 + 8), (150 + 8)

Types of database workloads

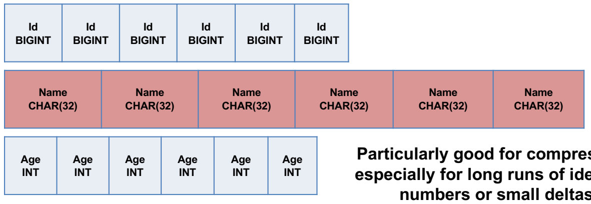
- OLTP = OnLine Transaction Processing
  - Write-heavy
  - Transactions
- OLAP = OnLine Analytical Processing
  - Read-heavy
  - Analytical scans or "rollups" along column
    - "SELECT AVG(latency) FROM system WHERE time > now() – interval("1h")

Comparison of disk layouts

- Row-oriented layout


Id BIGINT | Name CHAR(32) | Age INT | Gender SMALLINT | Birthday DATE | Id BIGINT | Name CHAR(32) | Age INT | Gender SMALLINT | Birthday DATE | Id BIGINT | C

- Column-oriented layout

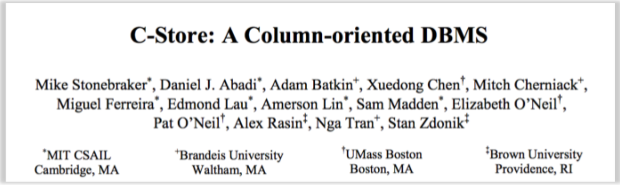

Id BIGINT | Id BIGINT | Id BIGINT | Id BIGINT | Id BIGINT | Id BIGINT

Name CHAR(32) | Name CHAR(32) | Name CHAR(32) | Name CHAR(32) | Name CHAR(32) | Name CHAR(32)

Age INT | Age INT | Age INT | Age INT | Age INT | Age INT

Particularly good for compression, especially for long runs of identical numbers or small deltas

Good discussion of benefits of columns…


C-Store: A Column-oriented DBMS

Mike Stonebraker*, Daniel J. Abadi*, Adam Batkin+, Xuedong Chen†, Mitch Cherniack+, Miguel Ferreira*, Edmond Lau*, Amerson Lin*, Sam Madden*, Elizabeth O'Neil†, Pat O'Neil†, Alex Rasin‡, Nga Tran+, Stan Zdonik‡

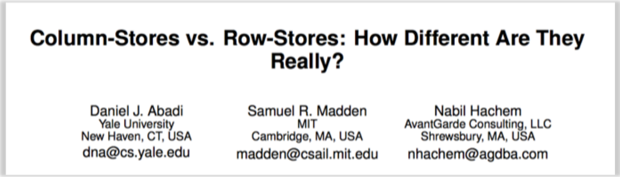
*MIT CSAIL
Cambridge, MA

+Brandeis University
Waltham, MA

†UMass Boston
Boston, MA

‡Brown University
Providence, RI

http://db.csail.mit.edu/projects/cstore/vldb.pdf


Column-Stores vs. Row-Stores: How Different Are They Really?

Daniel J. Abadi
Yale University
New Haven, CT, USA
dna@cs.yale.edu

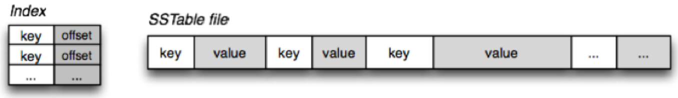
Samuel R. Madden
MIT
Cambridge, MA, USA
madden@csail.mit.edu

Nabil Hachem
AvantGarde Consulting, LLC
Shrewsbury, MA, USA
nhachem@agdba.com

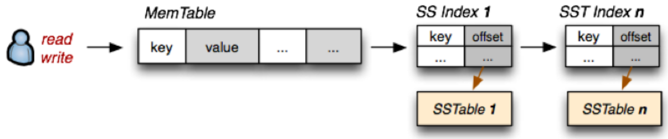
http://db.csail.mit.edu/projects/cstore/abadi-sigmod08.pdf

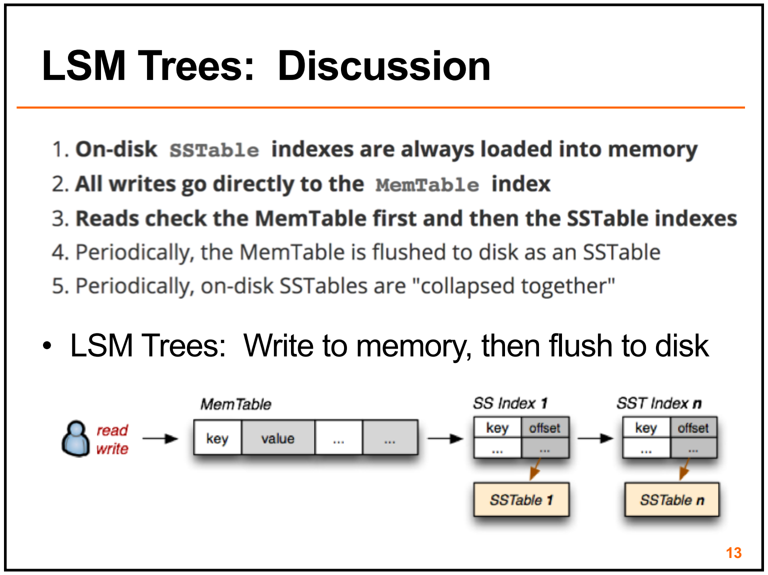
LSM Trees: Discussion

- SSTable: set of arbitrary, sorted key-value pairs



- LSM Trees: Write to memory, then flush to disk





LSM Trees: Discussion

1. **On-disk `SSTable` indexes are always loaded into memory**
2. **All writes go directly to the `MemTable` index**
3. **Reads check the MemTable first and then the SSTable indexes**
4. Periodically, the MemTable is flushed to disk as an SSTable
5. Periodically, on-disk SSTables are "collapsed together"

- LSM Trees: Write to memory, then flush to disk

13