



An on-disk binary data container, query engine and computational kernel

Overview

- What PyTables is?
- Data structures in PyTables
- Compressing data
- Advanced capabilities in PyTables



What it is

- A binary data container for on-disk, structured data
- Can perform operations with data *on-disk*
- Based on the standard de-facto HDF5 format
- Free software (BSD license)

About HDF5

(Hierarchical Data File version 5)

- A **versatile data model** that can represent complex data objects as well as associated metadata
- A **portable** file format with **no limit** on the number or size of data objects in the collection
- Implements a high-level API with C, C++, Fortran 90, and Java interfaces
- Free software (BSD, MIT kind of license)

PyTables distinctive features

- Supports a good range of compressors: Zlib, bzip2, LZO and Blosc
- Powerful query capabilities for Table objects, including indexing
- Can perform out-of-core operations very efficiently

What it is not

- Not a relational database replacement
- Not a distributed database
- Not extremely secure or safe (it's more about speed!)
- Not a mere HDF5 wrapper

DATA STRUCTURES

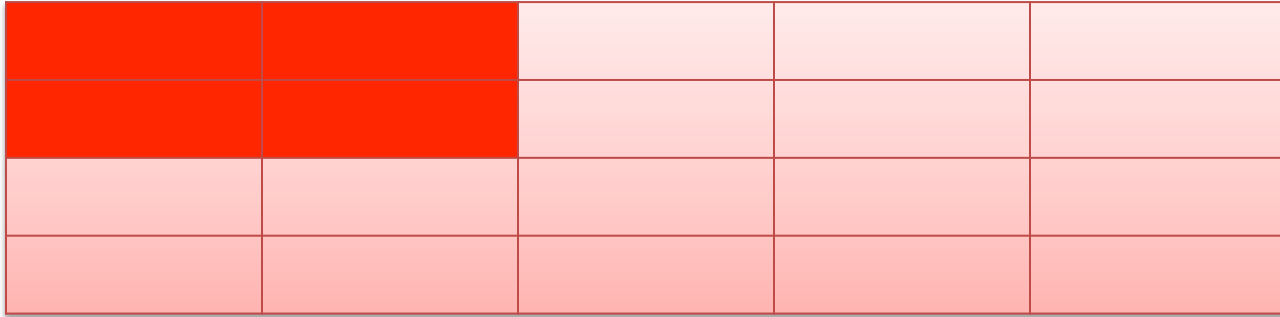
Data structures

- High level of flexibility for structuring your data:
 - Datatypes: scalars (numerical & strings), records, enumerated, time...
 - Tables support multidimensional cells and nested records
 - Mutidimensional arrays
 - Variable length arrays

The Array object

- Easy to create:
 - `file.createArray(mygroup, 'array', numpy_arr)`
- Shape cannot change
- Cannot be compressed

The CArray object



- Data is stored in chunks
- Each chunk can be compressed independently
- Shape cannot change

The EArray object



- Data is stored in chunks
- Can be compressed
- Shape can change (either enlarged or shrunk)
- Shape must be kept regular

The VLArray object

--	--	--

- Data is stored in variable length rows
- Can be enlarged or shrunk
- Data cannot be compressed

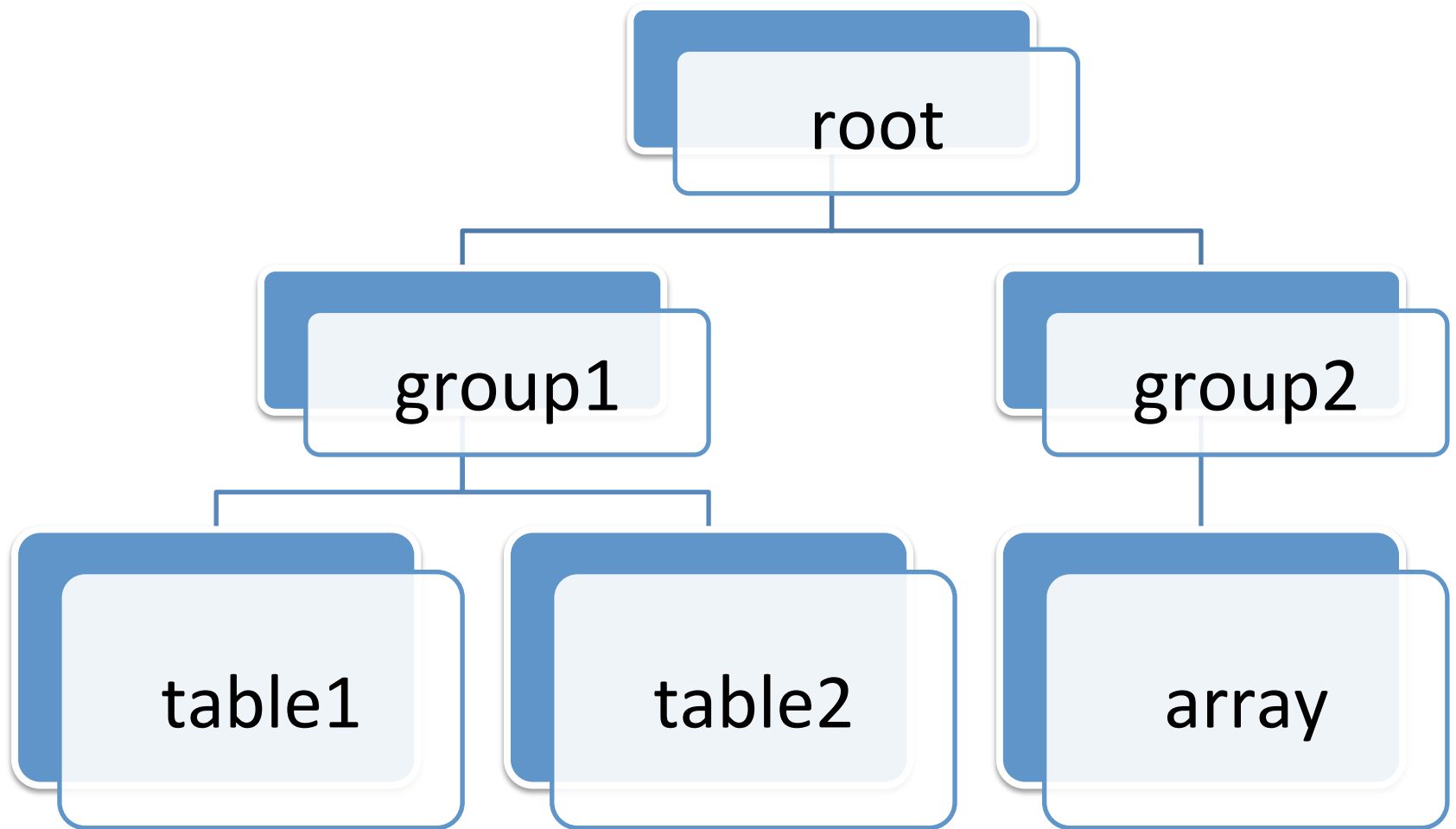
The Table object

Col1 (int32)	Col2 (string 10)	Col3 (bool)	Col4 (complex64)	Col5 (float32)

--	--	--	--	--

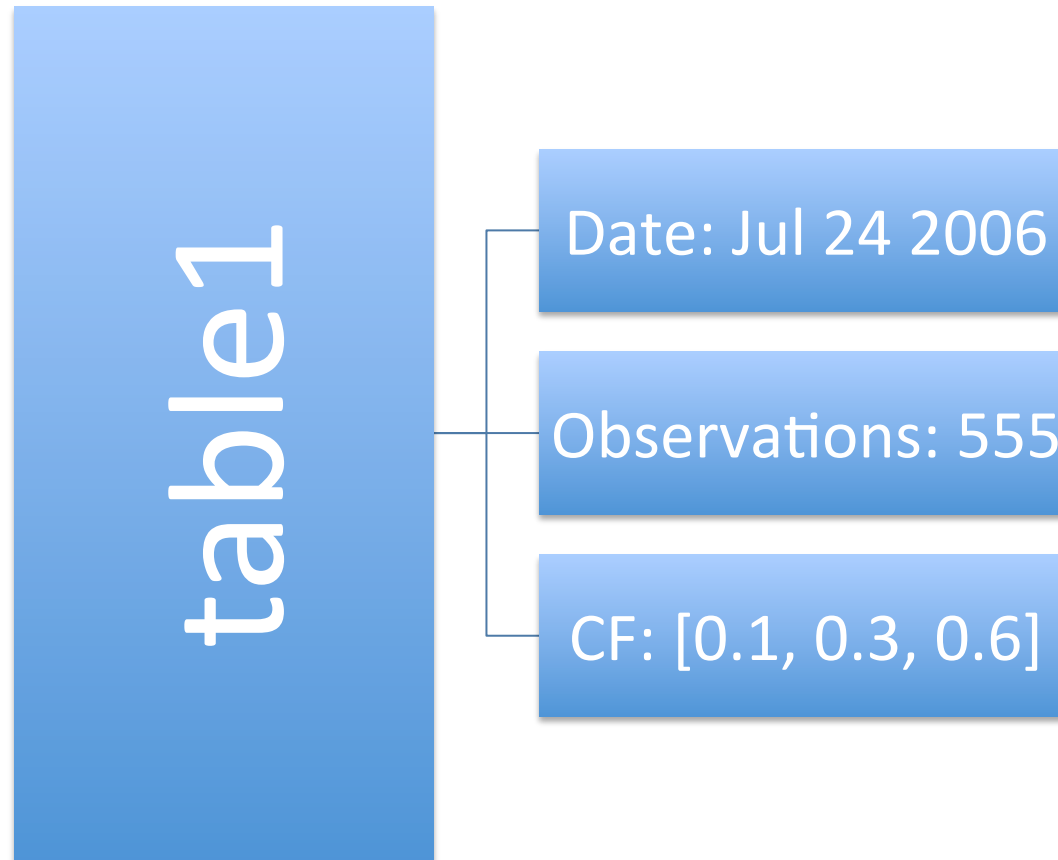
- Data is stored in chunks
- Can be compressed
- Can be enlarged or shrunk
- Fields cannot be of variable length

Dataset hierarchy



Attributes:

Metadata about data



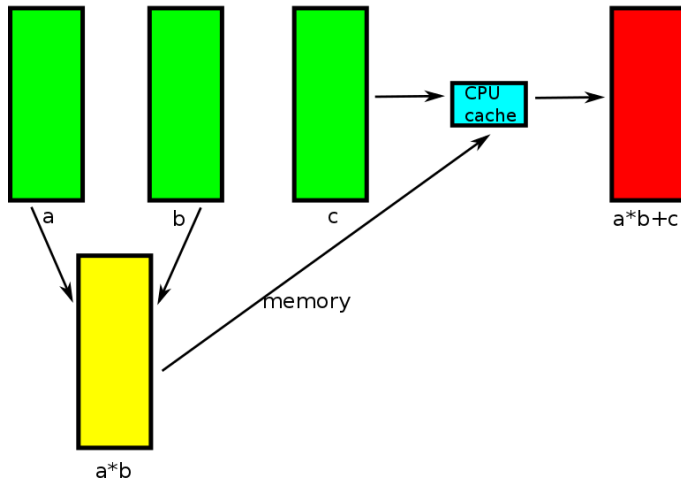
OUT-OF-CORE COMPUTATIONS

Operating with disk-based arrays

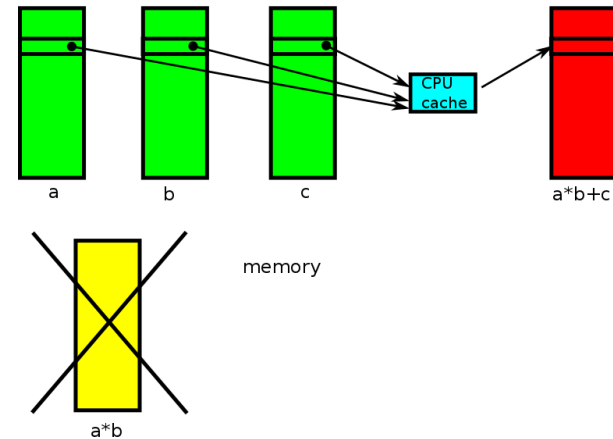
- `tables.Expr` is an optimized evaluator for expressions of disk-based arrays.
- It is a combination of the `Numexpr` advanced computing capabilities with the high I/O performance of `PyTables`.
- Similarly to `Numexpr`, disk-temporaries are avoided, and multi-threaded operation is preserved.

Avoiding temporaries with Numexpr

Computing "a*b+c" with NumPy. Temporaries goes to memory.

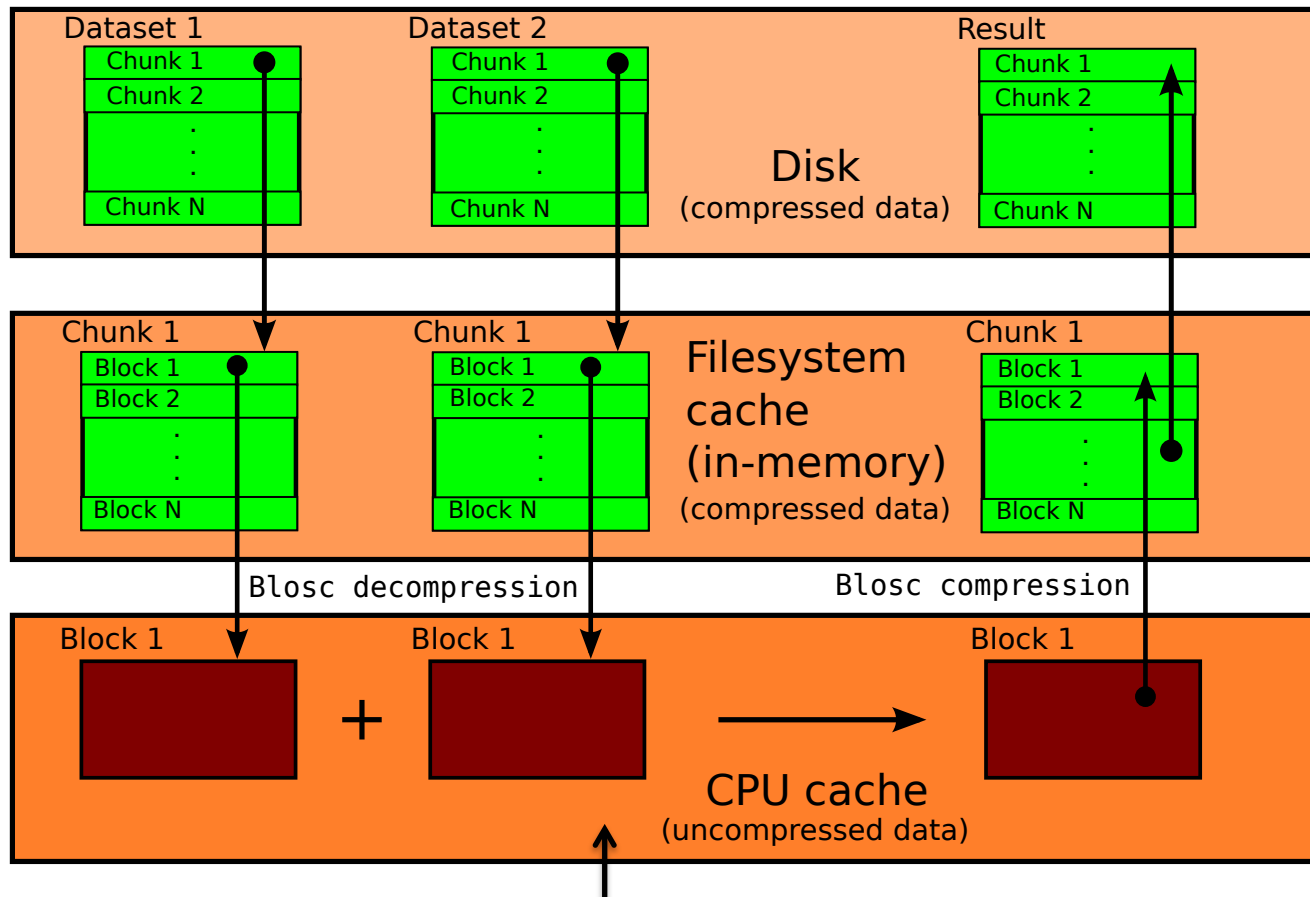


Computing "a*b+c" with Numexpr. Temporaries in memory are avoided.



Tables.Expr follows the same approach,
but with disk instead of memory

Performing out-of-core computations with PyTables



Virtual machine: numexpr

ADVANCED QUERY CAPABILITIES

Different query modes

Regular query:

- `[r['c1'] for r in table
if r['c2'] > 2.1 and r['c3'] == True))]`

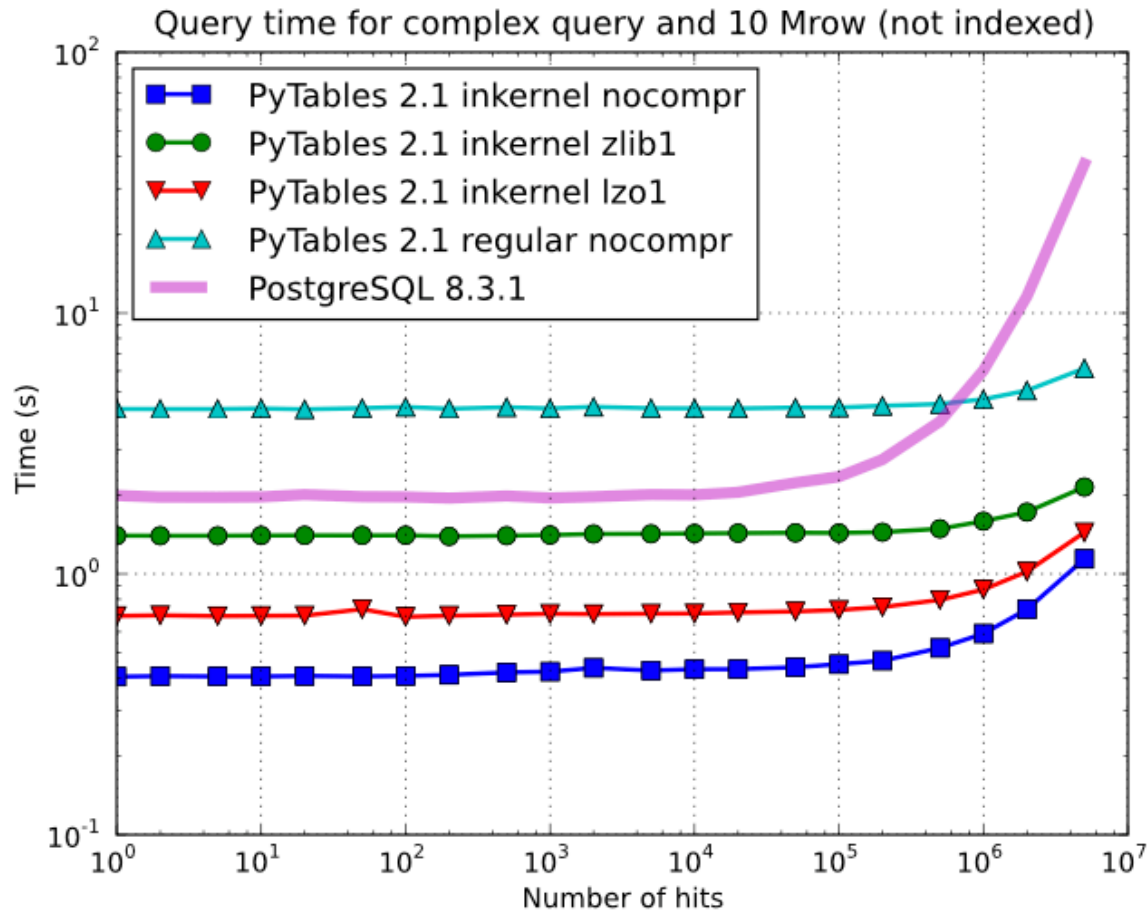
In-kernel query:

- `[r['c1'] for r in table.where('(c2>2.1)&(c3==True)')]`

Indexed query:

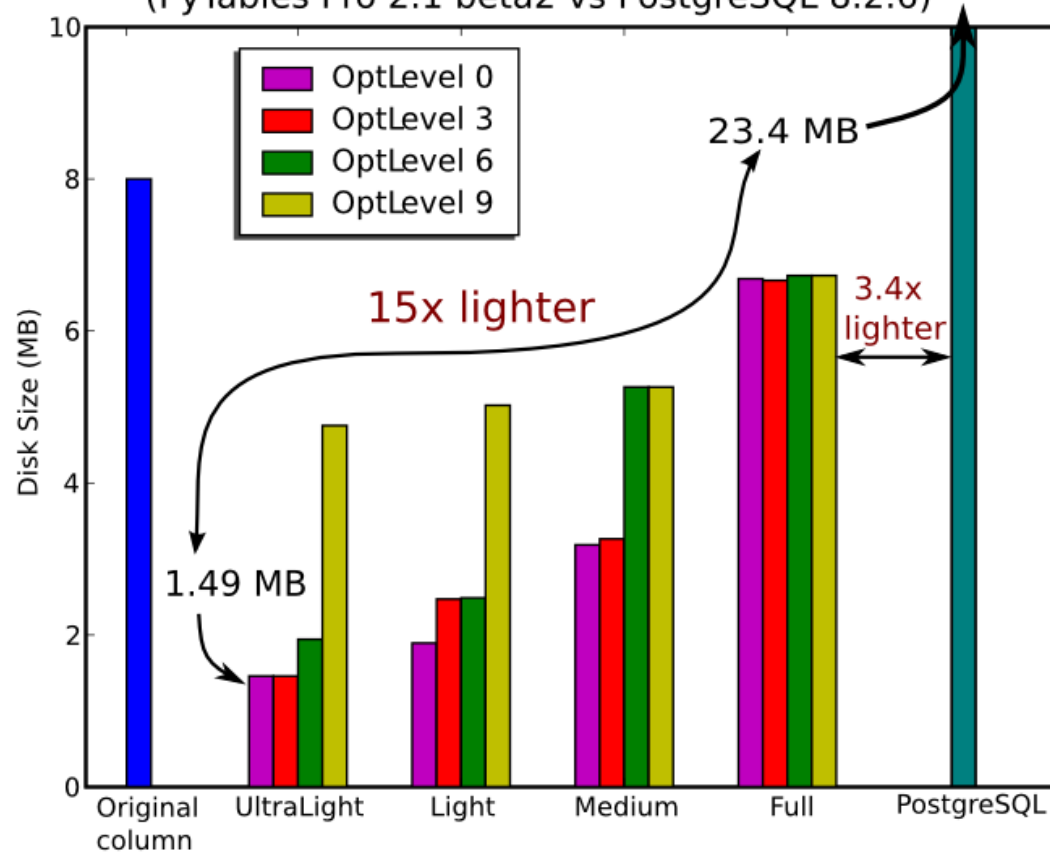
- `table.cols.c2.createIndex()`
- `table.cols.c3.createIndex()`
- `[r['c1'] for r in table.where('(c2>2.1)&(c3==True)')]`

Regular and in-kernel queries

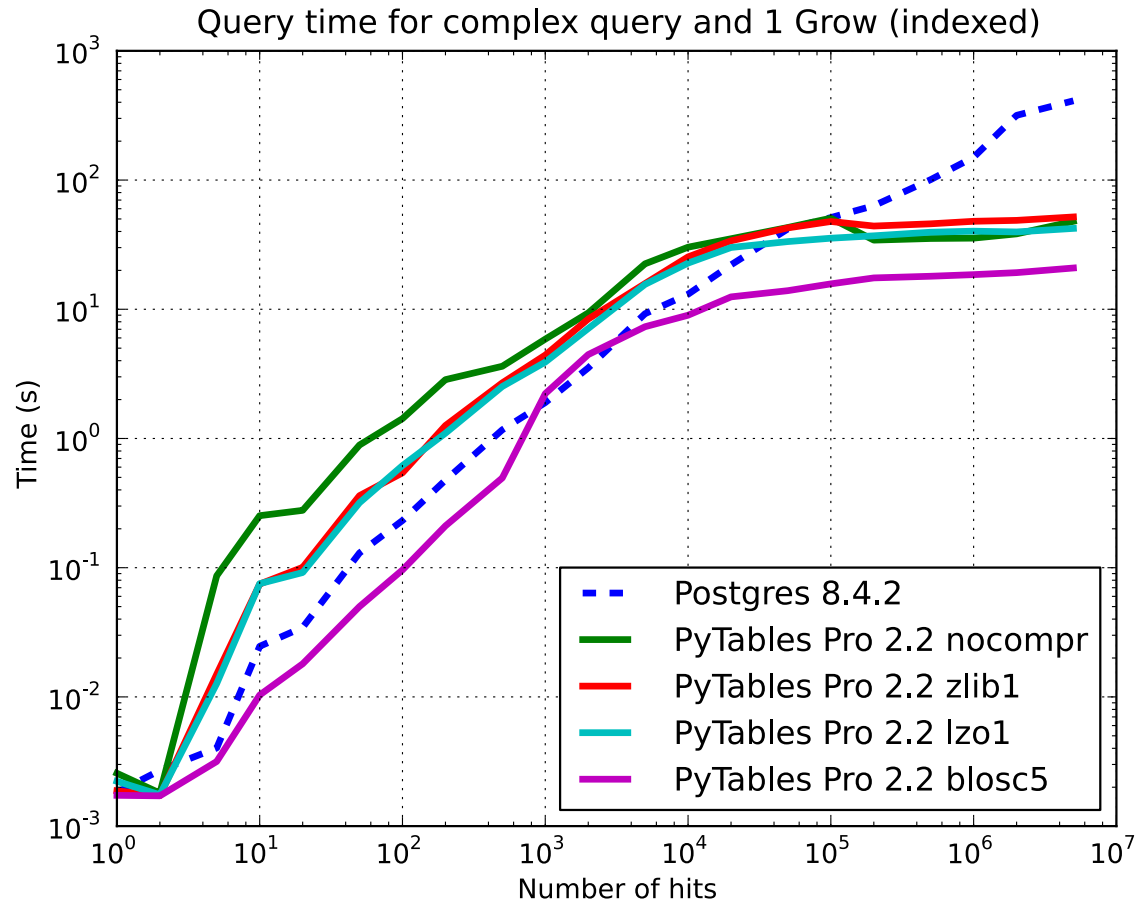


Customizable indexes

Sizes for index of a 1 Grow column with different optimizations
(PyTables Pro 2.1 beta2 vs PostgreSQL 8.2.6)



Indexed query performance



Concepts to take home

- PyTables is optimized to deal with data on disk
- Most of the operations use the iterator/generator machinery in Python: the goal is not to bloat memory with data
- Queries, indexes and out-of-core operations are good examples of the above