Introduction to HDF5

NCSA/University of Illinois at Urbana-Champaign

May 2000

http://hdf.ncsa.uiuc.edu

HDF

Topics

I. HDF Overview

II. HDF5 Objects and Structures

III. HDF5 Library and Other Software

IV. HDF5 Abstract Data Model

HDF Mission

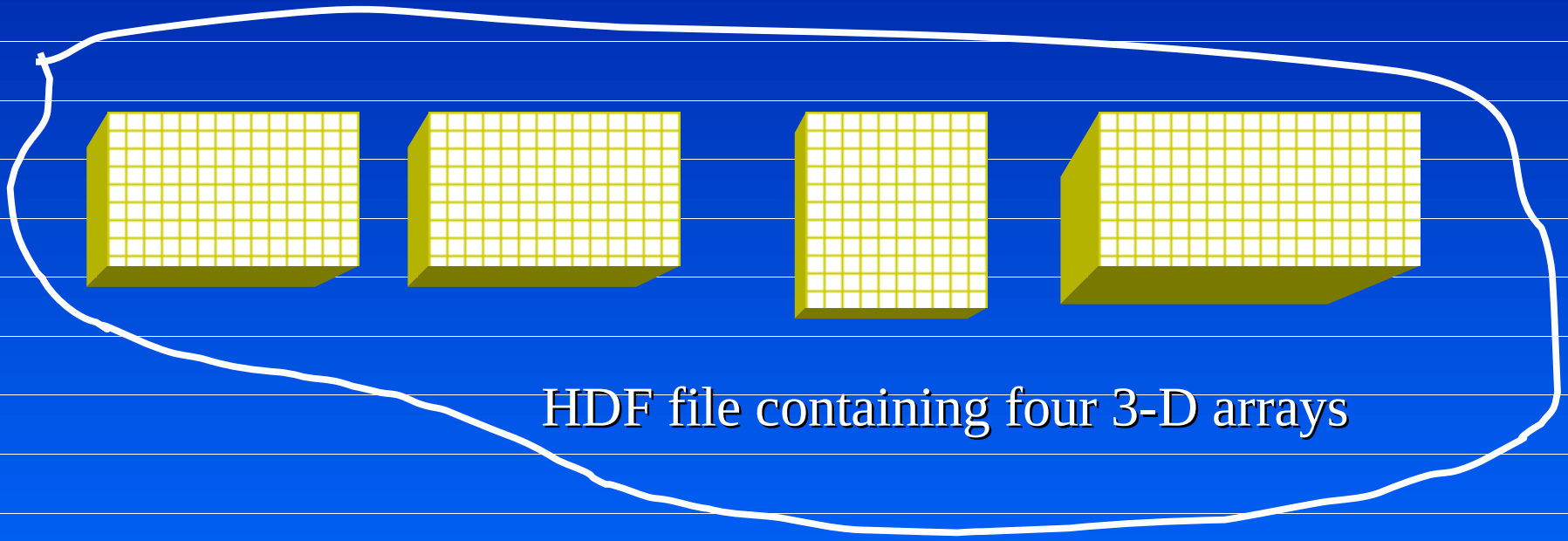
To develop, promote, deploy, and support open and free technologies that facilitate scientific data storage, exchange, access, analysis and discovery.

I. HDF Overview

What is HDF?

- Format and software for scientific data
- Stores images, multidimensional arrays, tables, etc.
- Emphasis on storage and I/O efficiency
- Free and commercial software support
- Emphasis on standards
- Users from many engineering and scientific fields

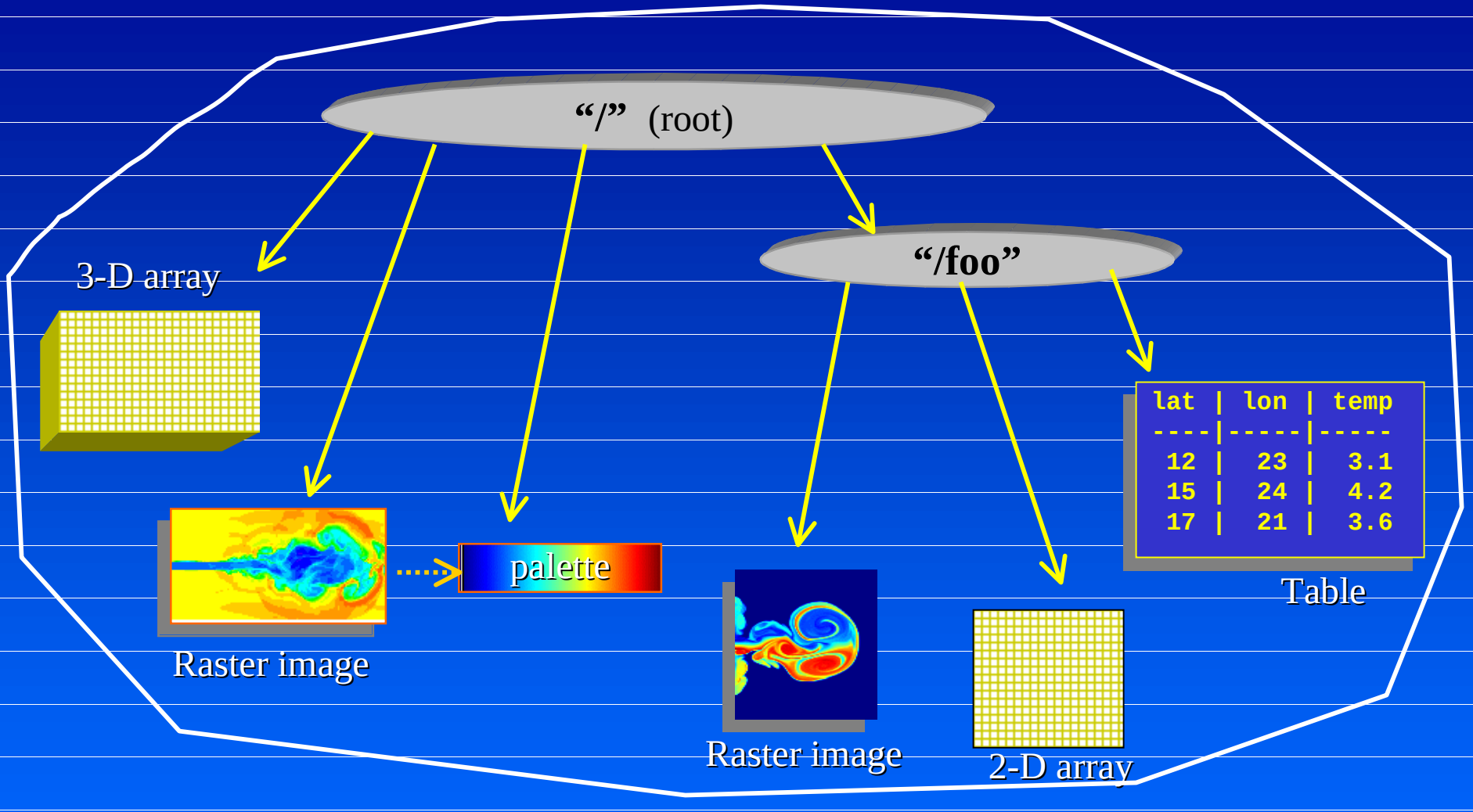
An HDF File: A Collection of Scientific Data Objects



HDF file containing four 3-D arrays

HDF

Example HDF5 file



HDF4 vs HDF5

- HDF4 - Based on original 1988 version of HDF
  - Backwardly compatible with all earlier versions
  - 6 basic objects
    - raster image, multidimensional array (SDS), palette, group (Vgroup), table (Vdata), annotation
- HDF5 - First released in 1998
  - New format(s) and library - not compatible with HDF4
  - 2 basic objects

HDF4 shortcomings

- Limits on object & file size (<2GB)
- Limited number of of objects (<20K)
- Rigid data models
- I/O performance

New Demands

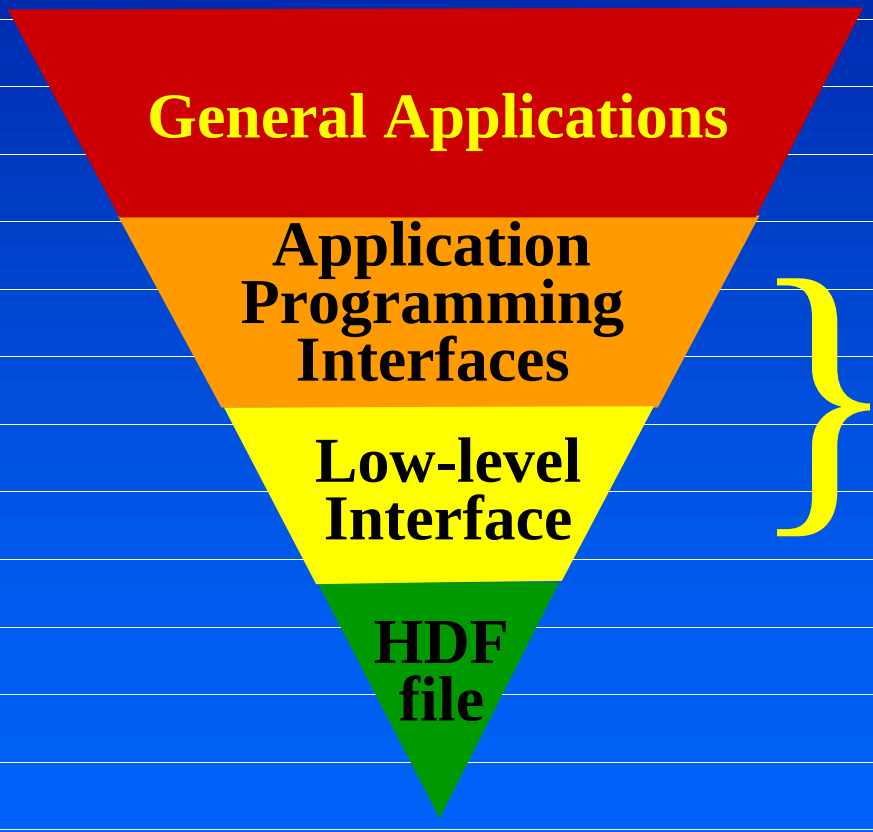
- Bigger, faster machines and storage systems
  - massive parallelism, teraflop speeds
  - parallel file systems, terabyte storage
- Greater complexity
  - complex data structures
  - complex subsetting
- Emphasis on remote & distributed access

These HDF shortcomings and new requirements motivated the development of HDF5

New HDF5 Features

- More scalable
  - Larger arrays and files
  - More objects
- Improved data model
  - New datatypes
  - Single comprehensive dataset object
- Improved software
  - More flexible, robust library
  - More flexible API
  - More I/O options

HDF Software



Utilities and applications for manipulating, viewing, and analyzing data.

HDF I/O library

– High-level, object-specific APIs.

– Low-level API for I/O to files, etc.

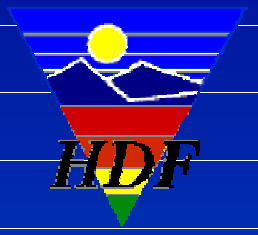
File or other data source

What platforms does HDF5 run on?

- AIX (IBM SP)
- Cray J90, T3E
- FreeBSD
- HP-UX
- IRIX 6.5, IRIX64
- Linux
- OSF1
- Solaris
- ASCI TFLOPS
- Windows NT4.0, 98

HDF supporters and users

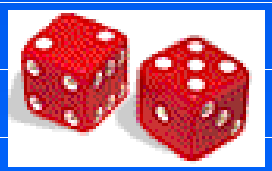
- NASA Earth Science Data & Info System
  - Commitment to migrate from HDF4 to HDF5
  - Prototype of HDF-EOS library on HDF5



- ASCI Data Models and Formats (DMF) Group
  - Open standard exchange format and I/O library
  - DOE tri-lab ASCI applications

- Globus (Argonne)
  - Infrastructure for distributed computation

- Army Research Lab DICE
  - Network Distributed Global Memory
  - HPC codes read/write into NDGM as if to HDF5



HDF supporters and users

- Alliance Problem-Solving Environments team
  - HDF5 access and visualization through VisAD
- TIKSL - Tele-immersion, collision of black holes
  - remote visualization and distributed file I/O
- Microsoft support for HDF5 on Win 2000
  - small grant + machines + training
- NCSA-affiliated Science teams
  - Visualization, data exch, fast I/O, ...
- Many others. See:
  - http://hdf.ncsa.uiuc.edu/users.html

HDF

Major User #1: EOSDIS

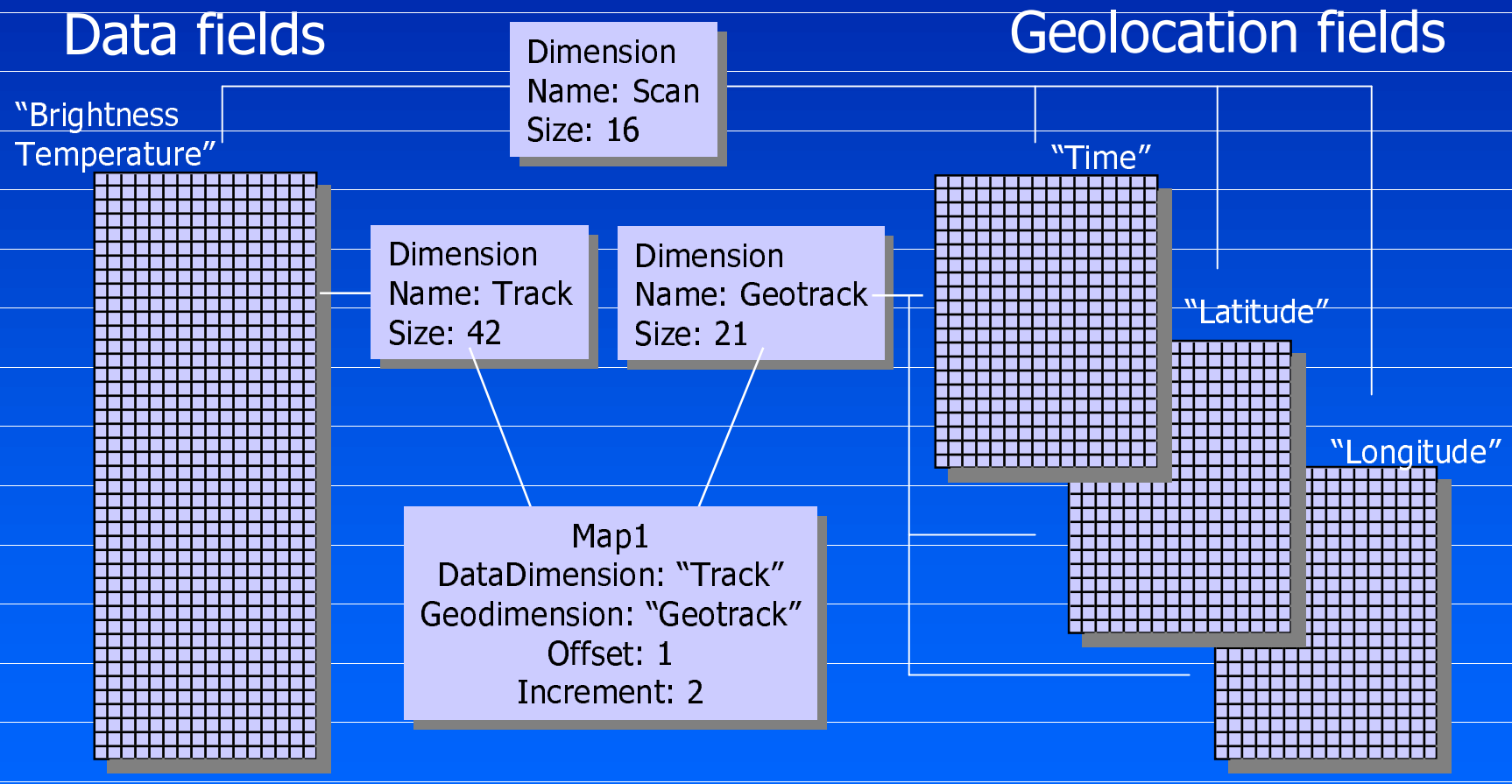
- ESDIS Project
  - open standard exchange format and I/O library for EOSDIS
  - EOS applications
- HDF requirements
  - Earth science data types (HDF-EOS, etc,)
  - User support for scientists, data producers, etc.
  - Library and file structure improvements
  - HDF tools, utilities, access software
  - Software maintenance and QA

EOS Constellation

HDF Standardization

- To share files, users must organize them similarly.
- HDF user groups create standard profiles
  - Ways to organize data in HDF files.
  - Metadata
  - API and library
- HDF-EOS example
  - Swath
  - Grid
  - Point

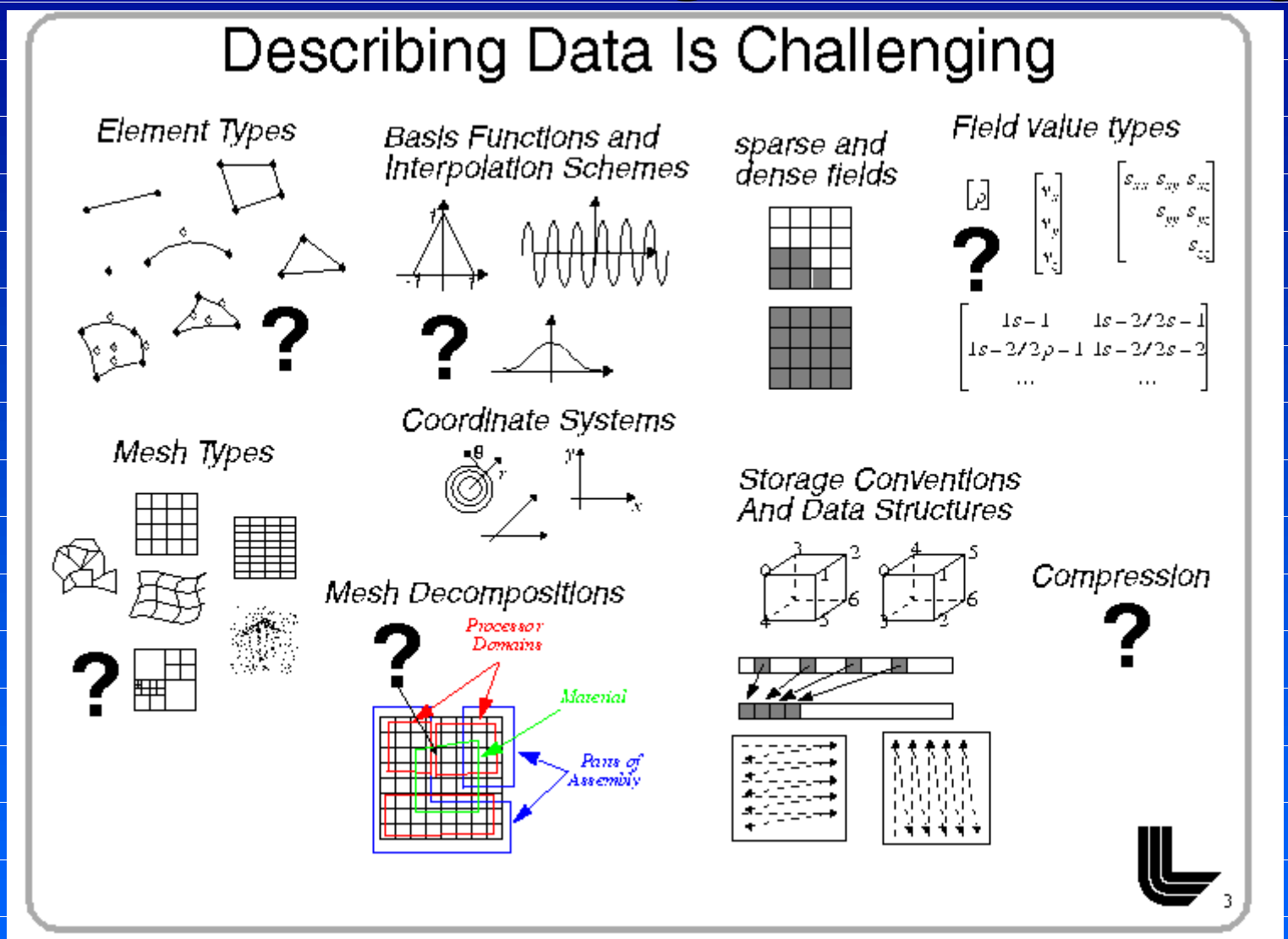
HDF-EOS "Swath" profile



Major User #2: ASCI

- ASCI Data Models and Formats (DMF) Group
  - open standard exchange format and I/O library
  - DOE tri-lab ASCI applications
- HDF requirements
  - large datasets (> a terabyte)
  - ASCI data types, especially meshes
  - good performance in massive parallel environments
  - primarily HDF 5

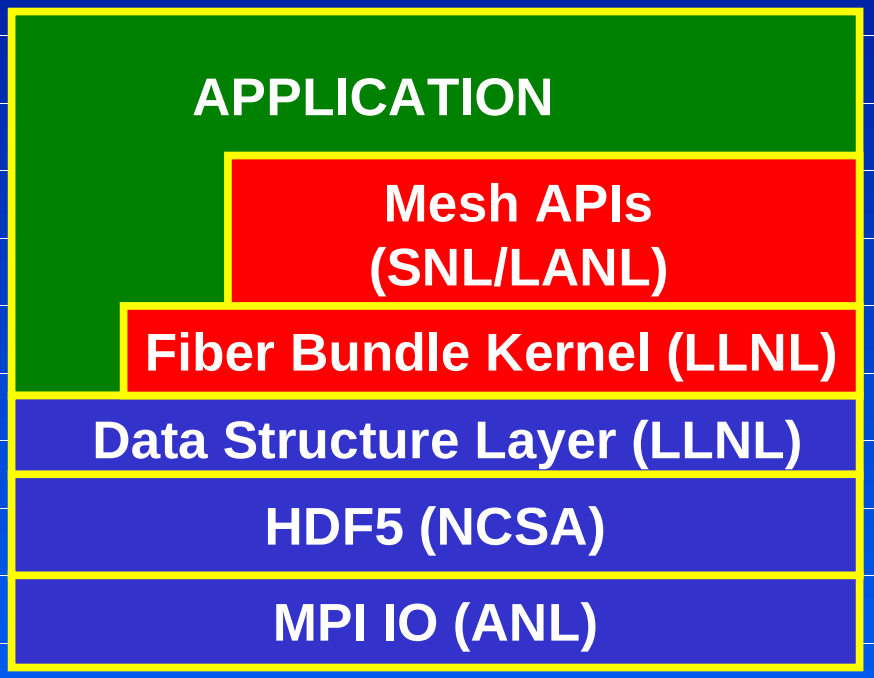
ASCI DMF: Describing Data is Challenging



HDF

ASCI DMF Data Abstraction

- Objectives
  - Sound data model with robust data abstractions
  - Computational mechanics data: meshes & fields
  - Based on mathematical field of fiber bundles
  - Common format allows common tools & sharing
  - Common API shield apps from model complexities



HDF

HDF5 objects and structures.

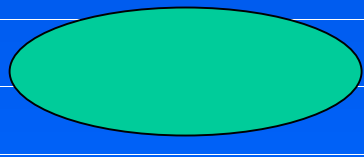
HDF5 File (conceptual view)

- Container for storing scientific data
  - Primary Objects:
    - Groups
    - Datasets
  - Secondary Objects:
    - Datatypes
    - Dataspaces
- Additional means to organize data
    - Attributes
    - Sharable objects
    - Storage and access properties

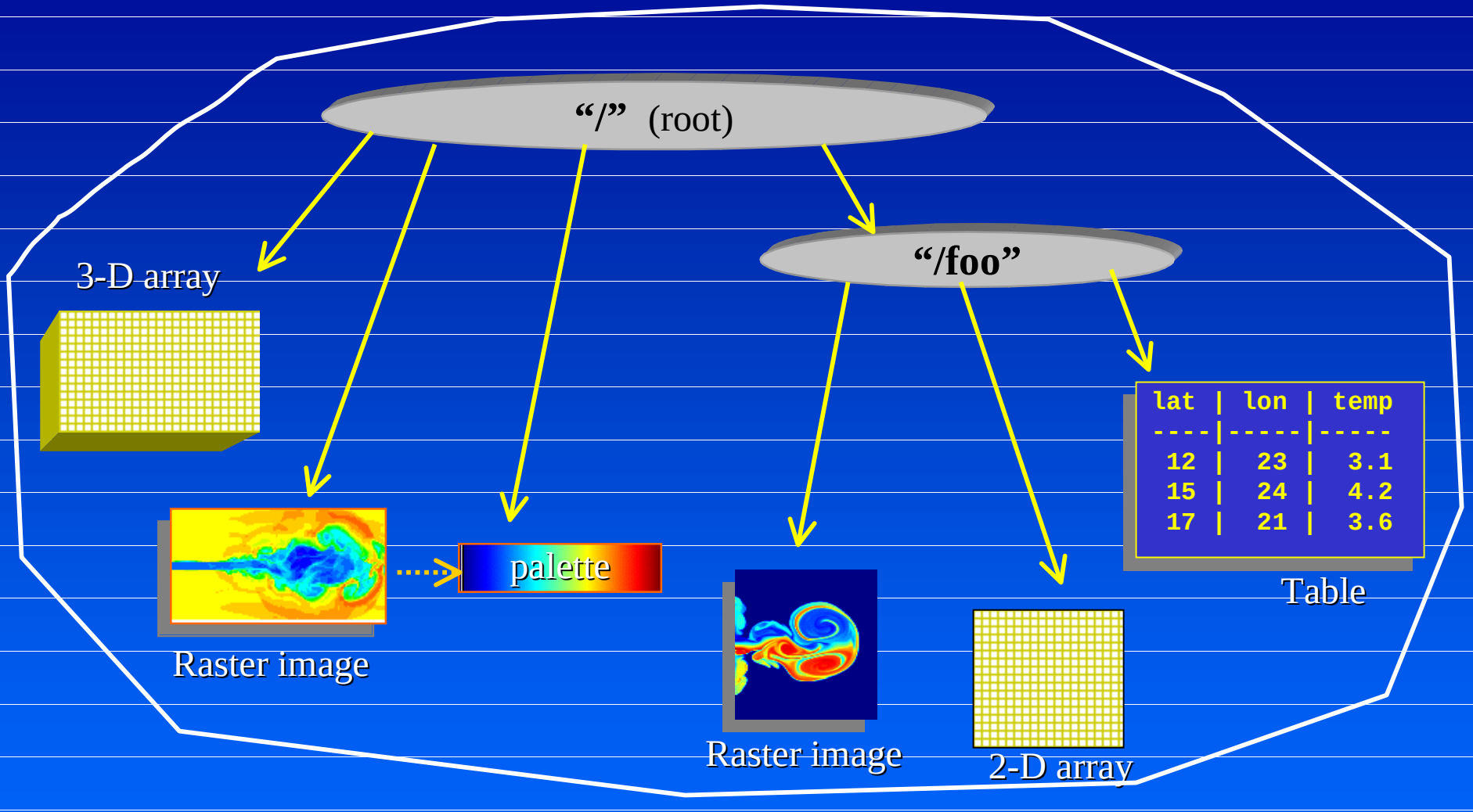
HDF5 data model

- Dataset
  - multidimensional array of elements, together with supporting metadata

- Group
  - directory-like structure containing datasets, groups, other objects



Example HDF5 file

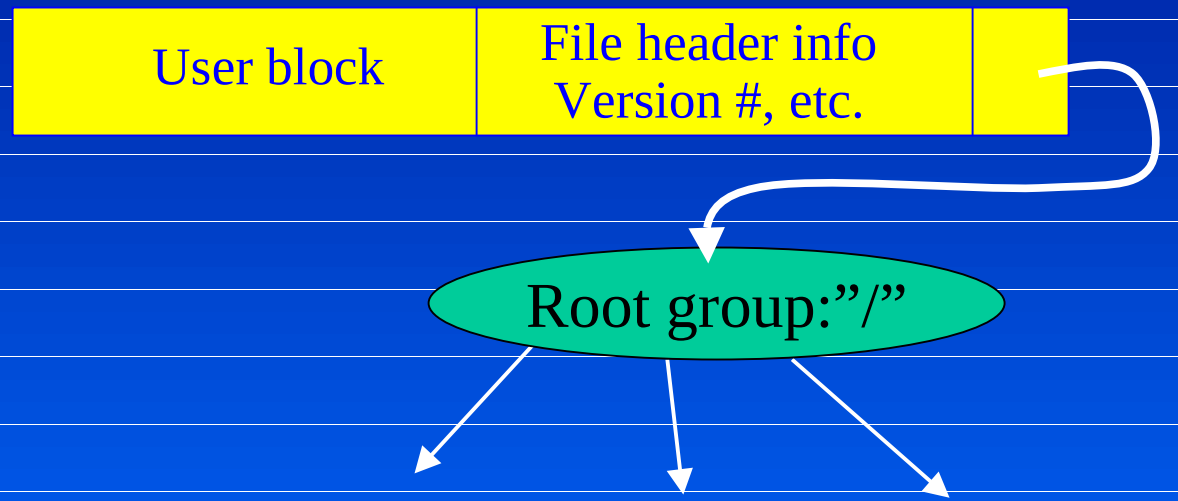


HDF

Components of a dataset

- Array
  - an ordered collection of identically typed data items distinguished by their indices (subscripts)
- Dataspace
  - information about the size and shape of a dataset array and selected parts of the array
- User-defined attribute list
- Special storage options
  - extendable, chunked, compressed, external

HDF5 file structure (many objects)



Other objects (datasets groups, etc.

HDF

Datatypes

- A datatype is
  - A classification specifying the interpretation of a data element
  - Specifies for a given data element
    - the set of possible values it can have
    - the operations that can be performed
    - how the values of that type are stored

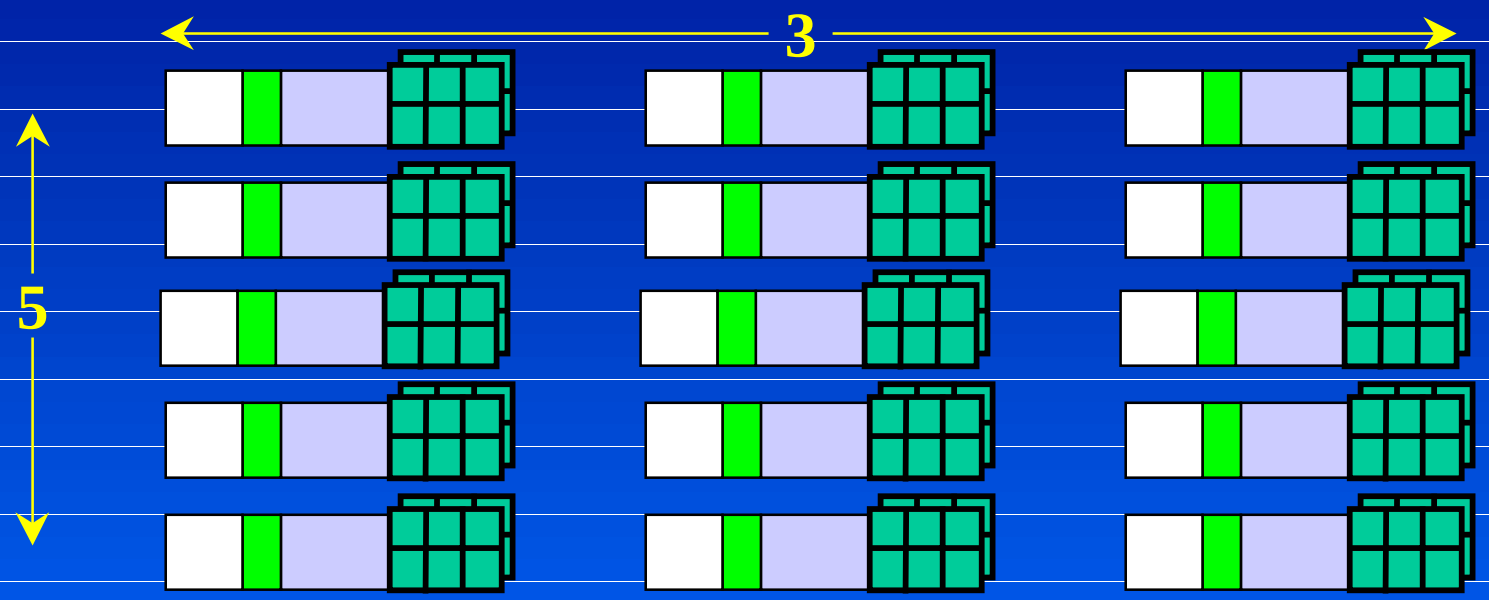
HDF5 datatypes

- Atomic types
  - standard integer & float
  - user-definable scalars (e.g. 13-bit integer)
  - variable length types (e.g. strings)
  - pointers - references to objects/dataset regions
  - enumeration - names mapped to integers

HDF5 Datatypes

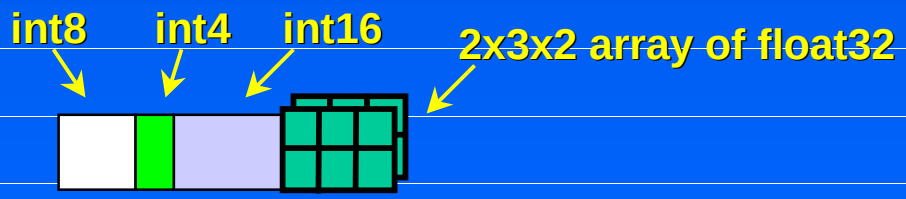
- compound types
  - Comparable to C structs
  - Members can be atomic or compound types
  - Members can be multidimensional

HDF5 dataset: array of records



Dimensionality: 5 x 3

Datatype:



Record

HDF

Dataspaces

- A dataspace contains information *about* a dataset
- Two components
  - how elements are organized to form a dataset
  - a subset of points, for partial I/O
- Applies to arrays in memory or in the file

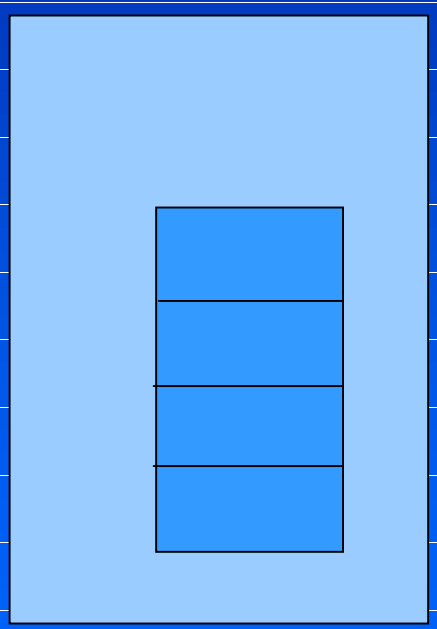
Data Spaces

- How elements are organized to form a dataset
  - rank
  - dimensions
- Subsetting during I/O operations
  - What subset of data is to be moved
- Named dataspaces will permit sharing

Dataspaces
Reading Dataset into Memory from File

File

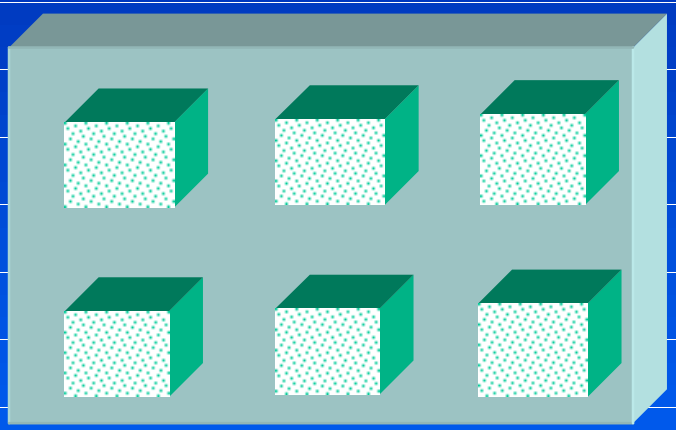
2D array of integers



Read

Memory

3D array of floats



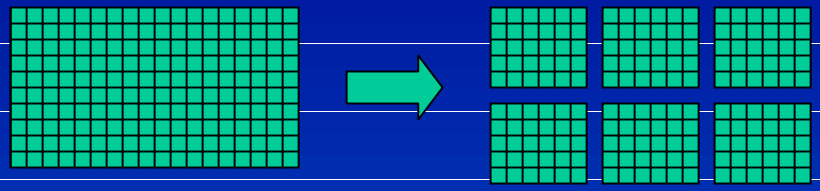
HDF

Attributes

- Are small pieces of data
- Attached to datasets or groups
- Operations are scaled-down versions of the dataset operations
  - Not extendible
  - No compression
  - No partial I/O

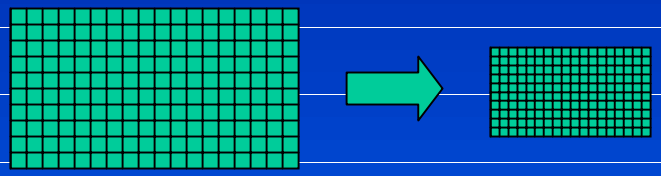
Special Storage Options

chunked



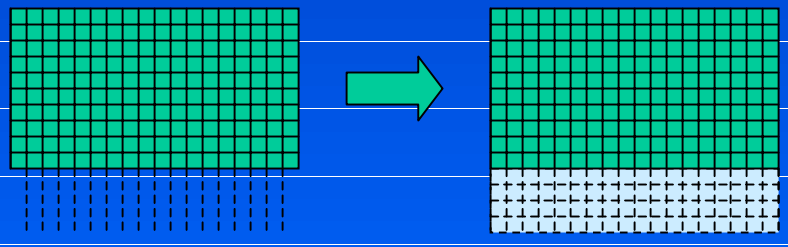
Better subsetting access time; extendable

compressed



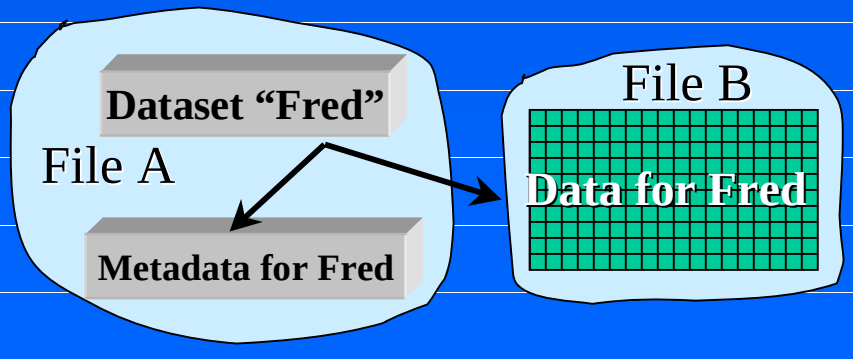
Improves storage efficiency, transmission speed

extendable



Arrays can be extended in any direction

Split file

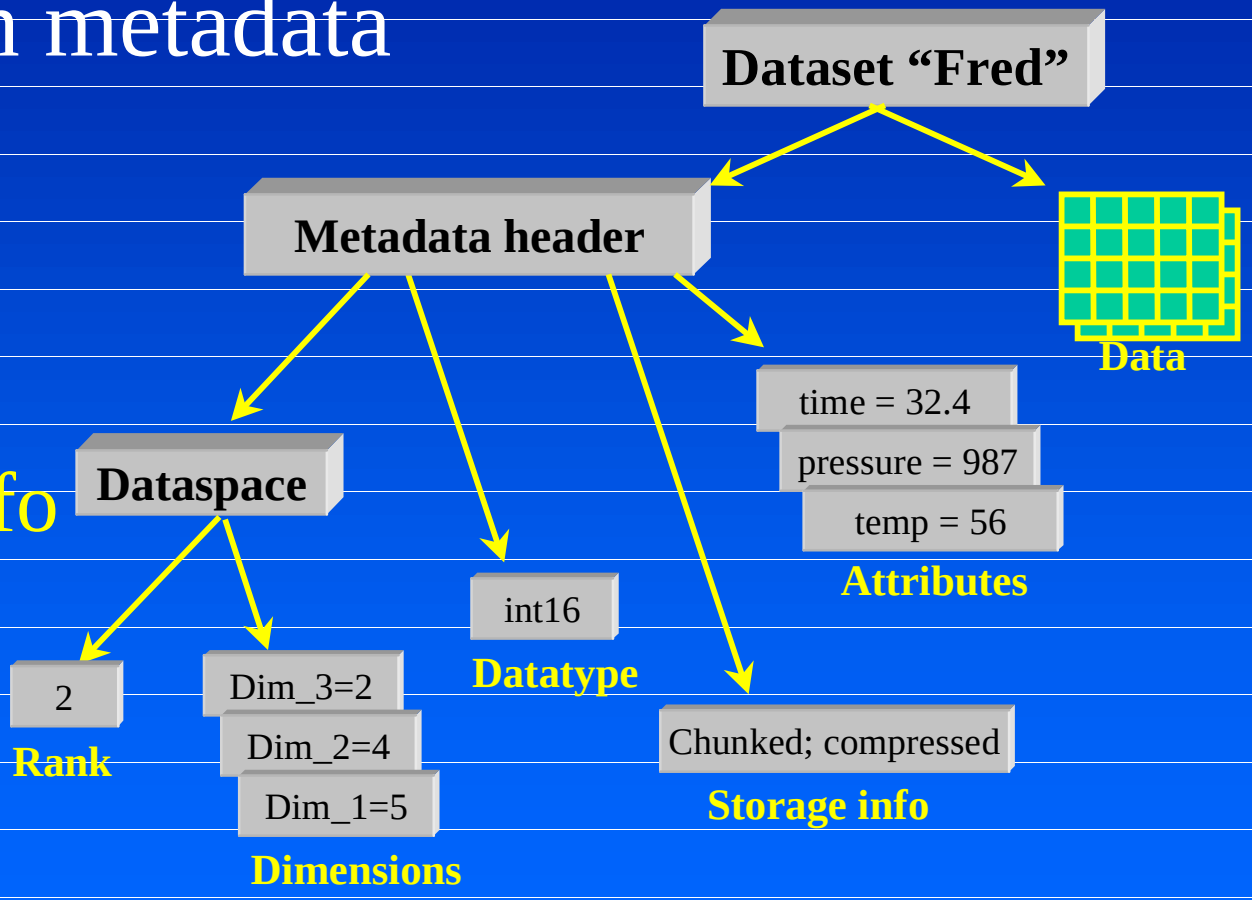


Metadata in one file, raw data in another.

HDF

Dataset components

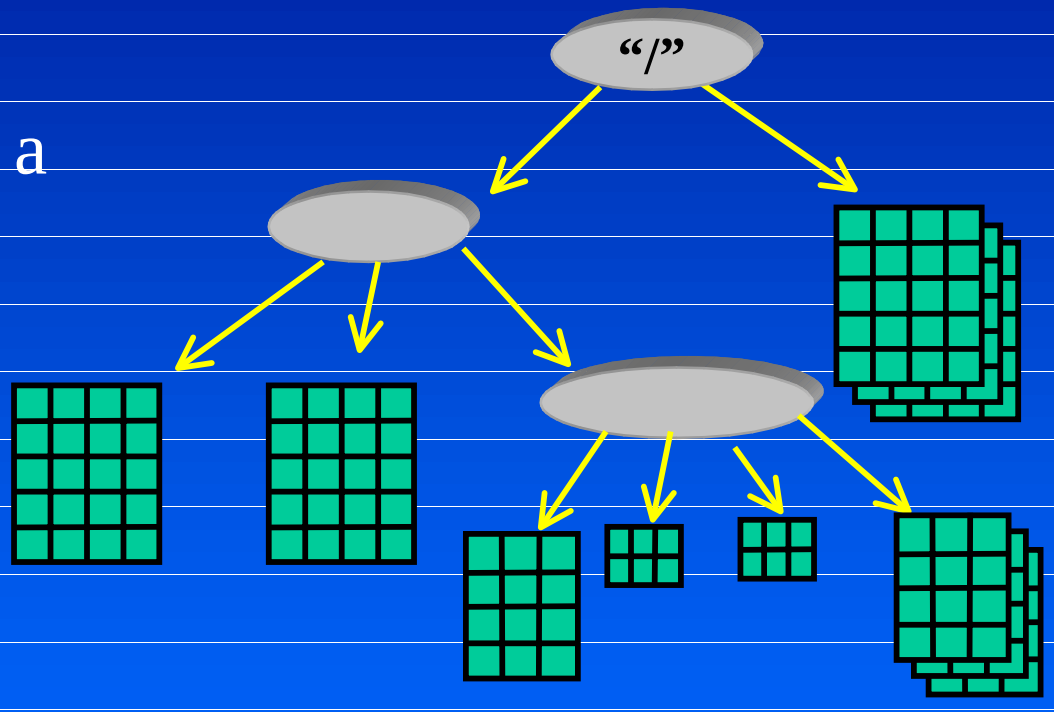
- a multidimensional array of data elements
- header with metadata
  - datatype
  - dataspace
  - attributes
  - storage info



HDF

Groups

- A mechanism for collections of related objects
- Every file starts with a root group
- Similar to UNIX directories
- Can have attributes



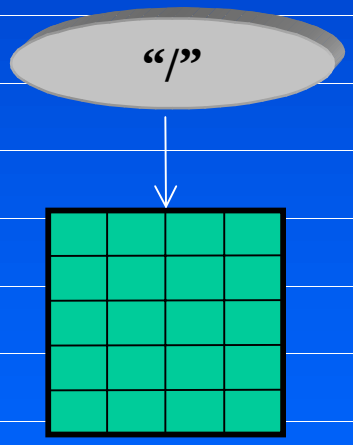
HDF

Groups

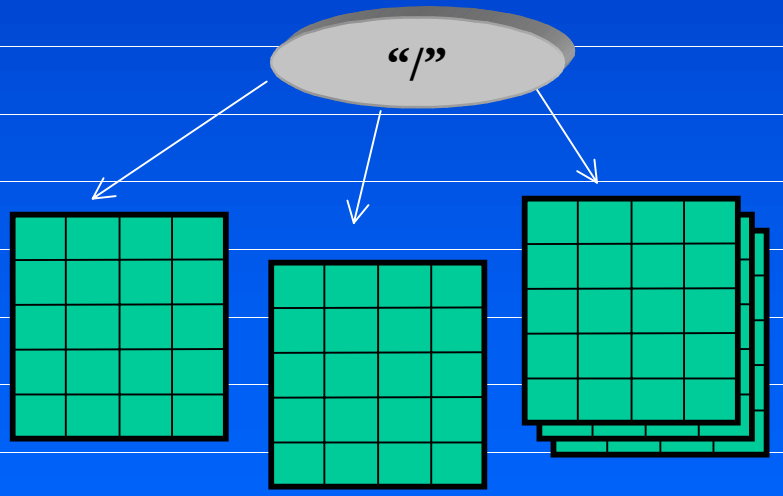
- Similar to Unix directories
  - Components separated by slashes
  - A root group called ``/''
  - Absolute and relative names
  - Hard and soft (symbolic) links
  - Not picky about repeated slashes
- Different from Unix directories
  - A graph rather than a tree
  - No automatic ``..'' entries

Groups

An HDF5 file with a root group and one 2D dataset



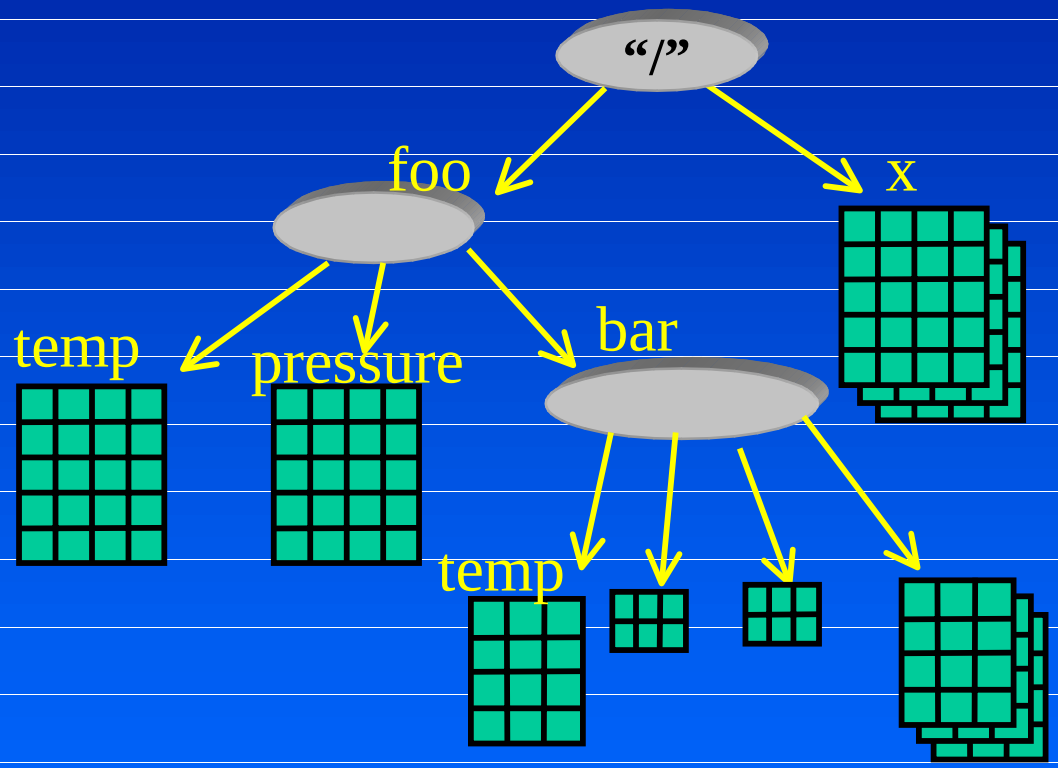
An HDF5 file with a root group, two 2D datasets and one 3D dataset.



HDF

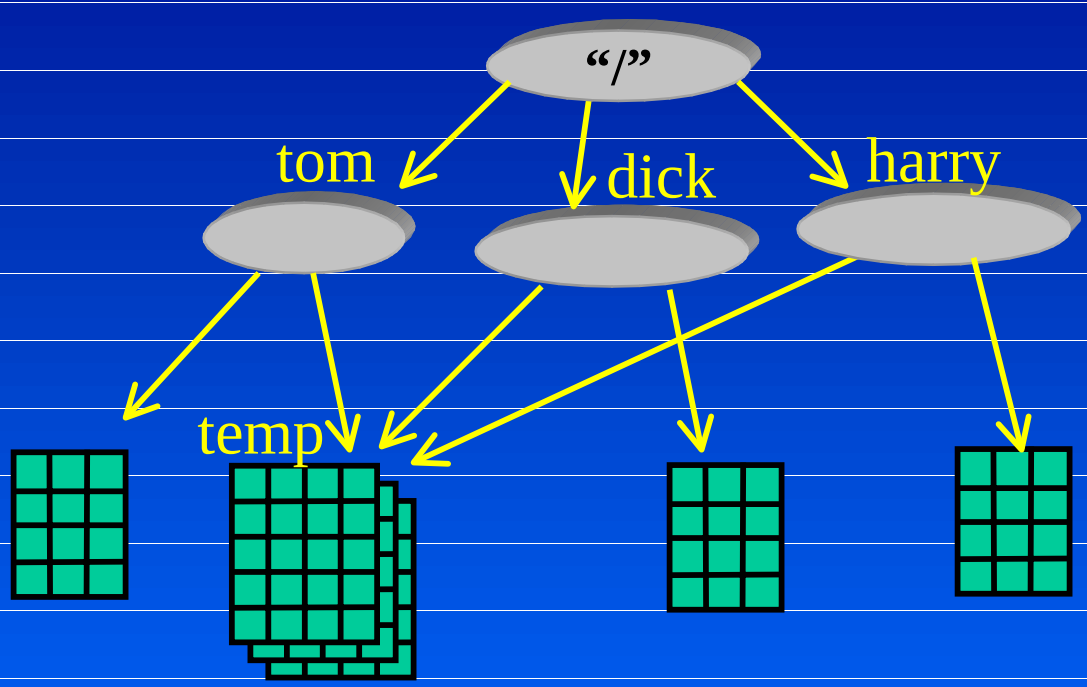
HDF5 objects are identified and located by their *pathnames*

/ (root)
/x
/foo
/foo/temp
/foo/pressure
/foo/bar/temp



HDF

Groups & members of groups can be shared



Note: Three pathnames identify the same object

/tom/temp
/dick/temp
/harry/temp

HDF

Other features

- User block
- Named datatypes
- Named dataspaces (future)

The HDF5 Library and other software

Goals

- Flexible API to support a wide range of operations on data
- High performance access in serial and parallel computing environments
- Compatibility with common data models and programming langages

Features

- Support for high performance applications
  - Ability to create complex data structures
  - Complex subsetting
  - Efficient storage
  - Flexible I/O (parallel, remote, etc.)
- Support for key language models
  - OO compatible
  - C & Fortran primarily
  - Also Java, C++

The General Programming Paradigm

- Objects are opened or created
- Then accessed
- Finally closed

- Example

```
CALL h5fopen_f (“myfile”, H5F_ACC_RDWR_F, file_id, err)
CALL h5dopen_f (file_id, “velocity”, dset_id, err)
CALL h5dread_f (dset_id, H5T_NATIVE_INTEGER, data, err)
CALL h5dclose_f (dset_id, error)
CALL h5fclose_f (file_id, error)
```

Anything that can be set from the API can also be queried

Creating an HDF5 Dataset

- Create an identifier for the dataset
- Independently define dataset characteristics
  - datatype, dataspace, property list
- Create the dataset
  - specify path, datatype, dataspace, etc.
- Close datatype, dataspace, dataset, etc.

Datasets and Dataspaces

- Opening a dataset occurs three ways
  1. Create a new dataspace with H5Screate_simple()
  2. Copy an existing dataspace with H5Scopy()
  3. Retrieve a dataspace from a dataset with H5Dget_space()

Atomic Data Types

- The library has predefined native atomic types:

H5T_NATIVE_CHAR	H5T_NATIVE_INT8
H5T_NATIVE_USHORT	H5T_NATIVE_UINT16
H5T_NATIVE_INT	H5T_NATIVE_INT32
H5T_NATIVE_LONG	H5T_NATIVE_LLONG
H5T_NATIVE_FLOAT	H5T_NATIVE_FLOAT32
H5T_NATIVE_DOUBLE	H5T_NATIVE_FLOAT64
H5T_NATIVE_STRING	H5T_NATIVE_TIME
H5T_NATIVE_DATE	H5T_NATIVE_BITFIELD
H5T_NATIVE_OPAQUE	H5T_NATIVE_INT64

- Some non-native types will be added for common architectures.
- New types can be derived from existing types

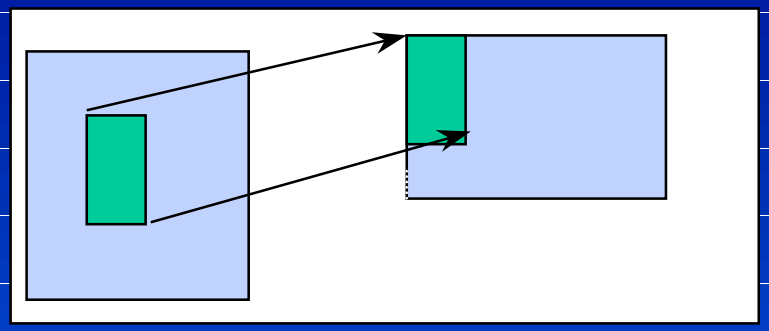
Dataset I/O

- Dataset I/O involves
  - reading or writing
  - all or part of a dataset
- During I/O operations data is translated between the source & destination
  - data types (e.g. 16-bit integer => 32-bit integer)
  - dataspace (e.g. 10x20 2d array => 200 1d array)
  - also compressed/uncompressed, etc.

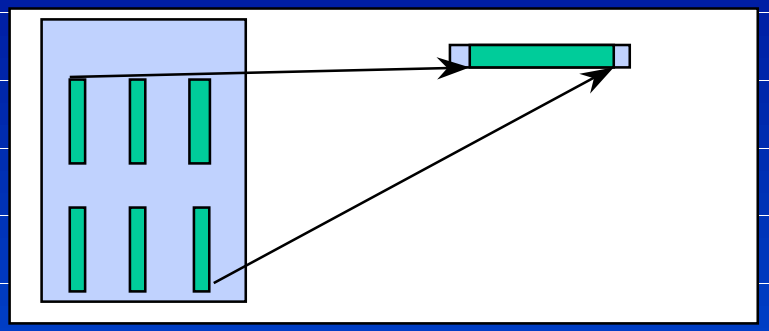
Partial I/O

- Selection regions define shapes of memory and file spaces
- Region in memory can be different shape from region in file
- Selection regions can be
  - hyperslabs
  - points
  - unions of hyperslabs
- Also supports parallel I/O via MPI-I/O

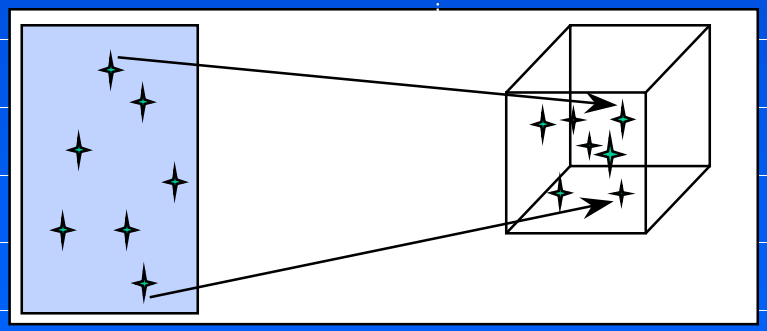
Mappings between file dataspaces/selections and memory dataspaces/selections.



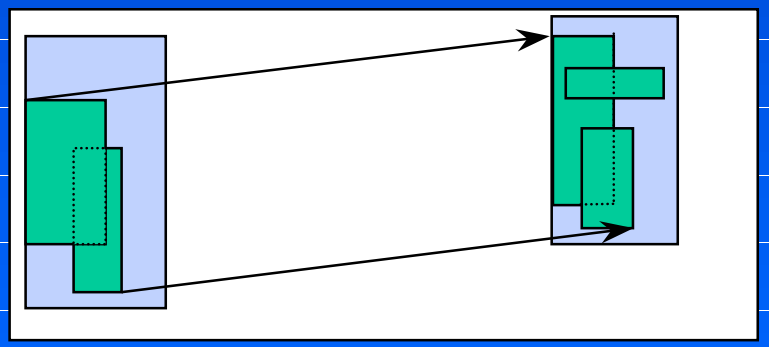
(a) A hyperslab from a 2D array to the corner of a smaller 2D array



(b) A regular series of blocks from a 2D array to a contiguous sequence at a certain offset in a 1D array



(c) A sequence of points from a 2D array to a sequence of points in a 3D array.



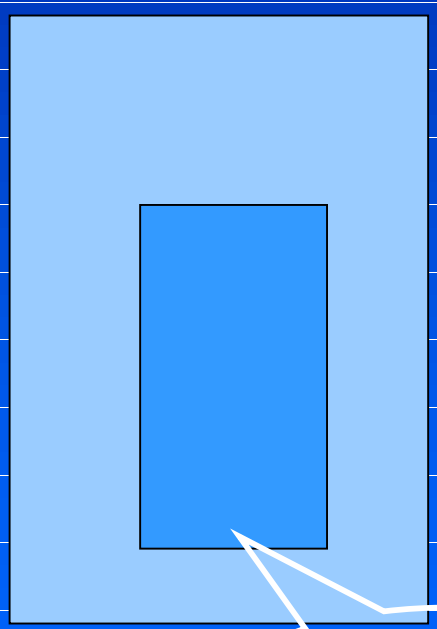
(d) Union of hyperslabs in file to union of hyperslabs in memory. Number of elements must be equal.

HDF

Reading Dataset into Memory from File

File

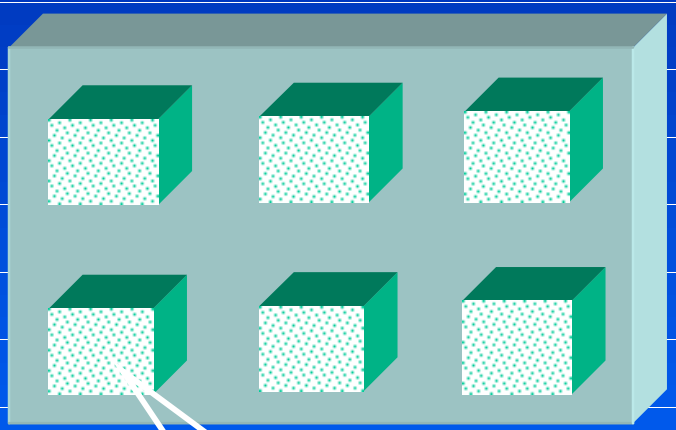
2D array of integers



Read

Memory

3D array of floats



2-d array

Regularly spaced series of cubes

HDF

Property Lists

- Reduce the number of arguments in the usual case with H5P_DEFAULT.
- Provide support for the unusual cases when
  - Creating files
  - Opening files
  - Creating datasets
  - Reading or writing data
- Allow functionality to be added without affecting applications.

Property Lists

- Dataset creation properties include
  - Compression
  - Extendibility
  - External storage

Property Lists

- Example: Creating a dataset with ``deflate'' compression

```
hid t dcpl = H5Pcreate(H5P DATASET CREATE);
H5Pset deflate(dcpl, 9);
hid t v = H5Dcreate(file, ”velocity”, ..., dcpl);
H5Pclose(dcpl);
```

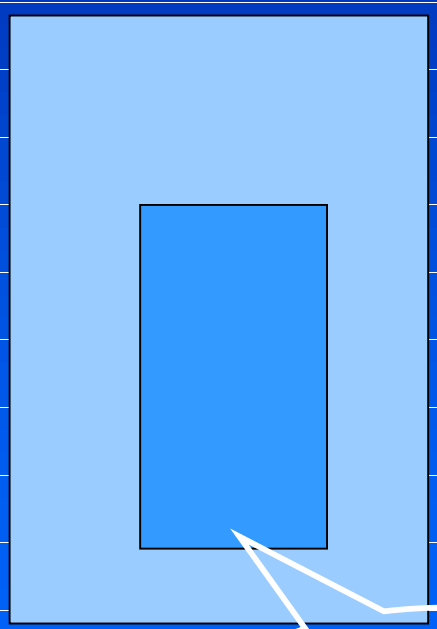
Order of operations

- The library imposes an order on the operations by argument dependencies.
  - Example: a file must be opened before a dataset because H5Dopen() takes a file handle as an argument.
  - Example: a data space must be created before a dataset because H5Dcreate() takes a data space handle as an argument.
- Objects can be closed in any order and reusing a closed object will result in an error.
- All objects are closed by normal program exit or H5close().

Reading Dataset into Memory from File

File

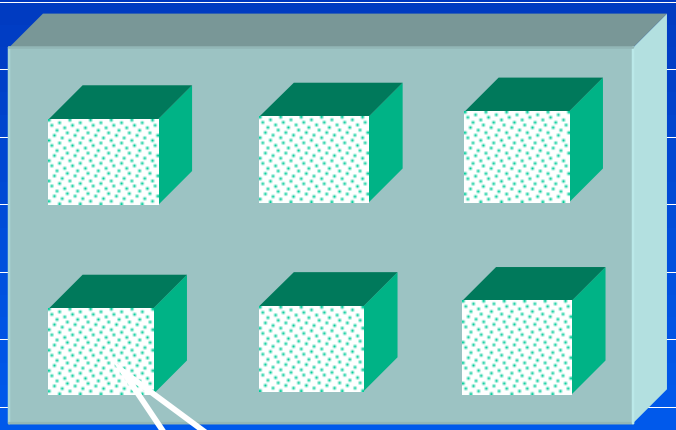
2D array of integers



Read

Memory

3D array of floats

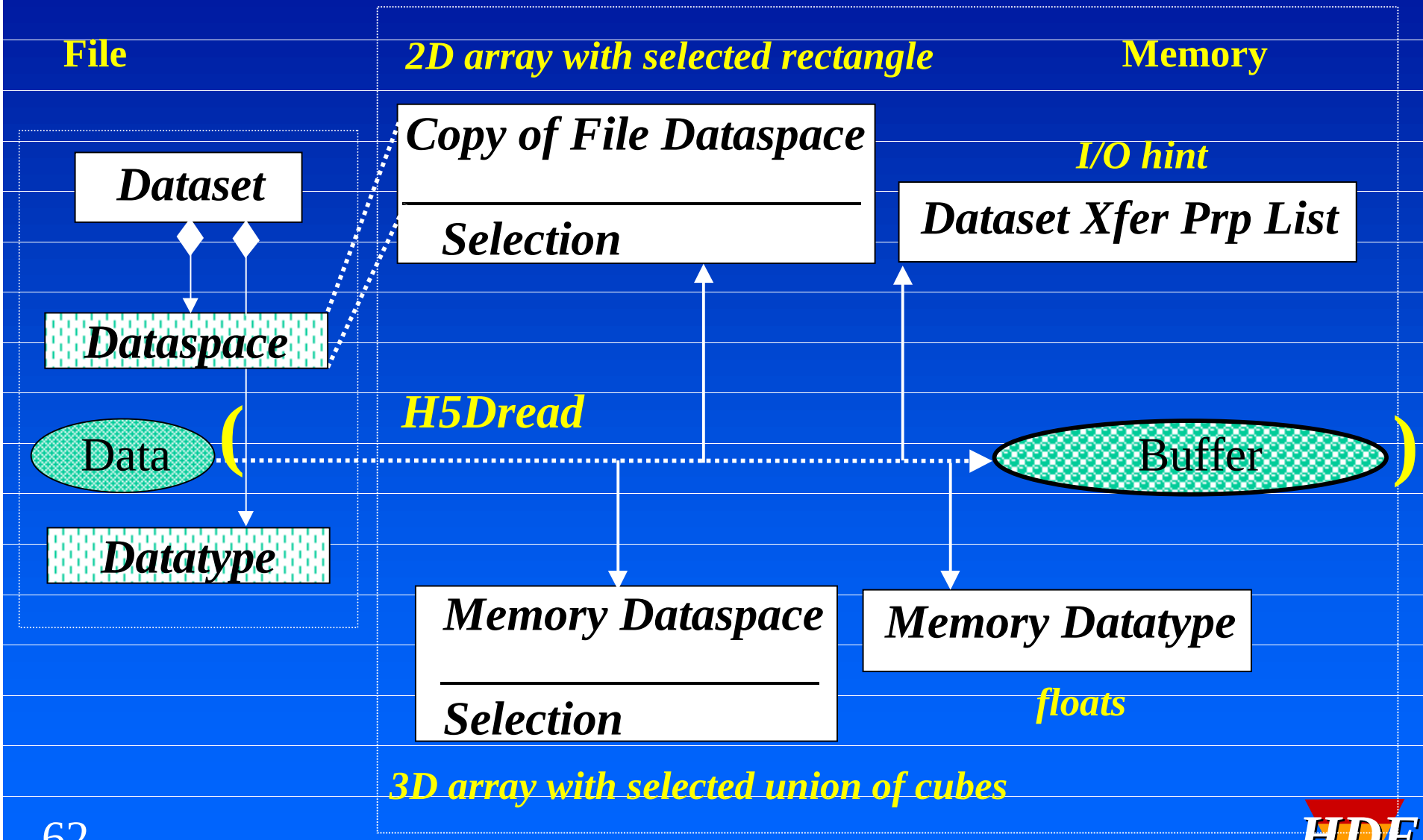


2-d array

Regularly spaced series of cubes

HDF

Reading Dataset into Memory from File



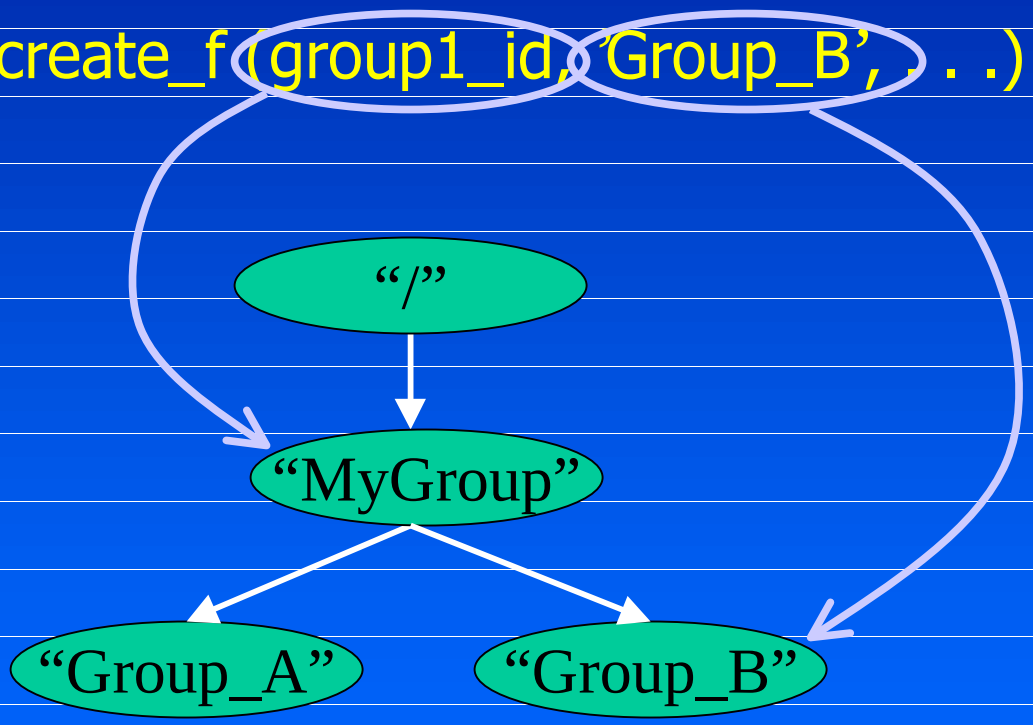
HDF

Steps to Create a Group

- Obtain the location identifier where the group is to be created
- Create the group
  - Absolute or relative name can be used
- Close the group

Example: Add “Group_B” to “MyGroup”

```
22  ! Add "Group_B" to "MyGroup" using relative name.
23  CALL h5gcreate_f (group1_id, ’Group_B’, . . .)
```



HDF

A few other features

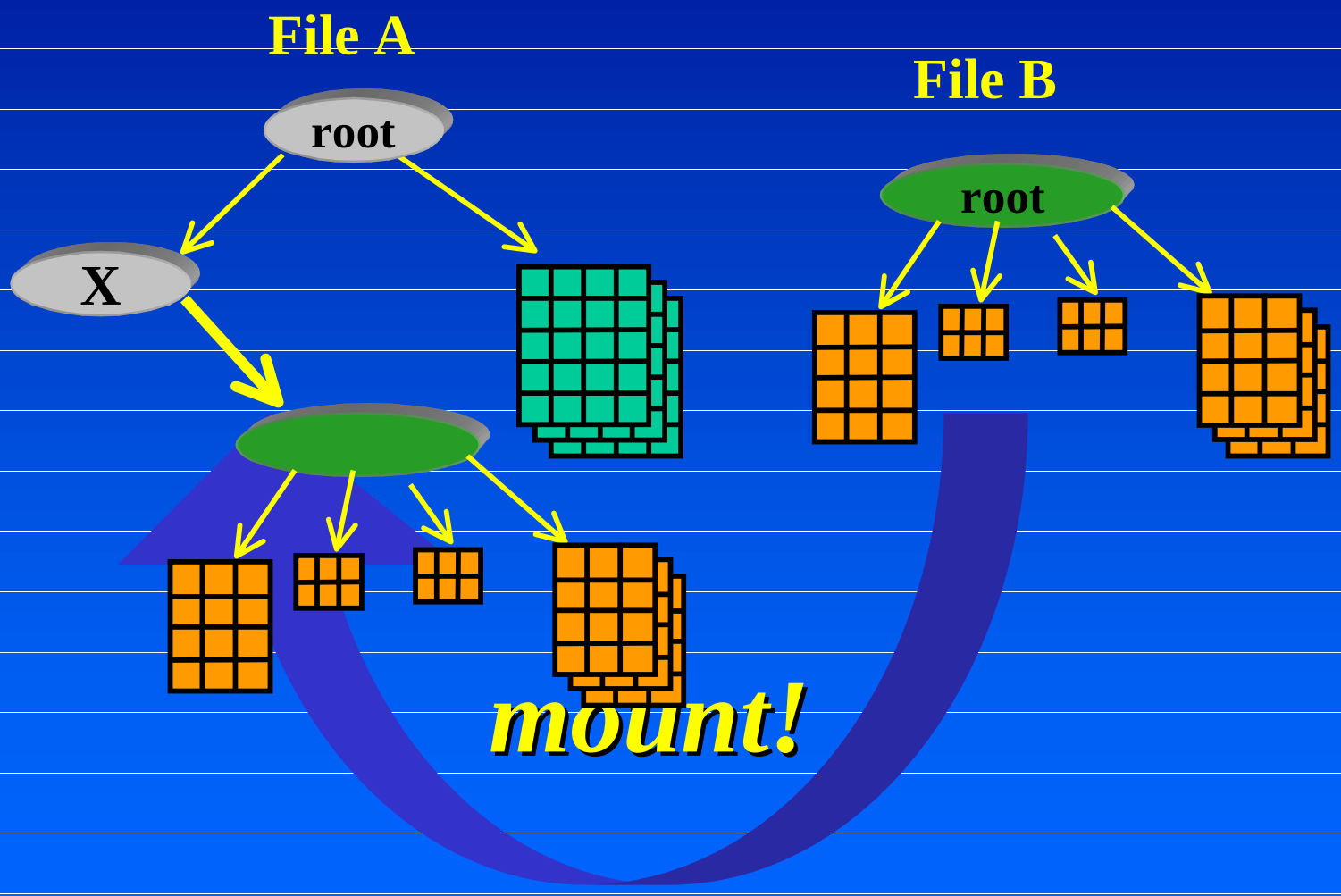
Mounting Files in HDF5

- Allows you to combine two or more HDF5 files in memory
- Similar to mounting files in UNIX.
- The group structure and metadata from one file appear as though they exist in another file.

Steps to Mount a File

- Open the HDF5 files
- Choose the "mount point" in the first file
- Call H5Fmount to mount the second file in the first one.

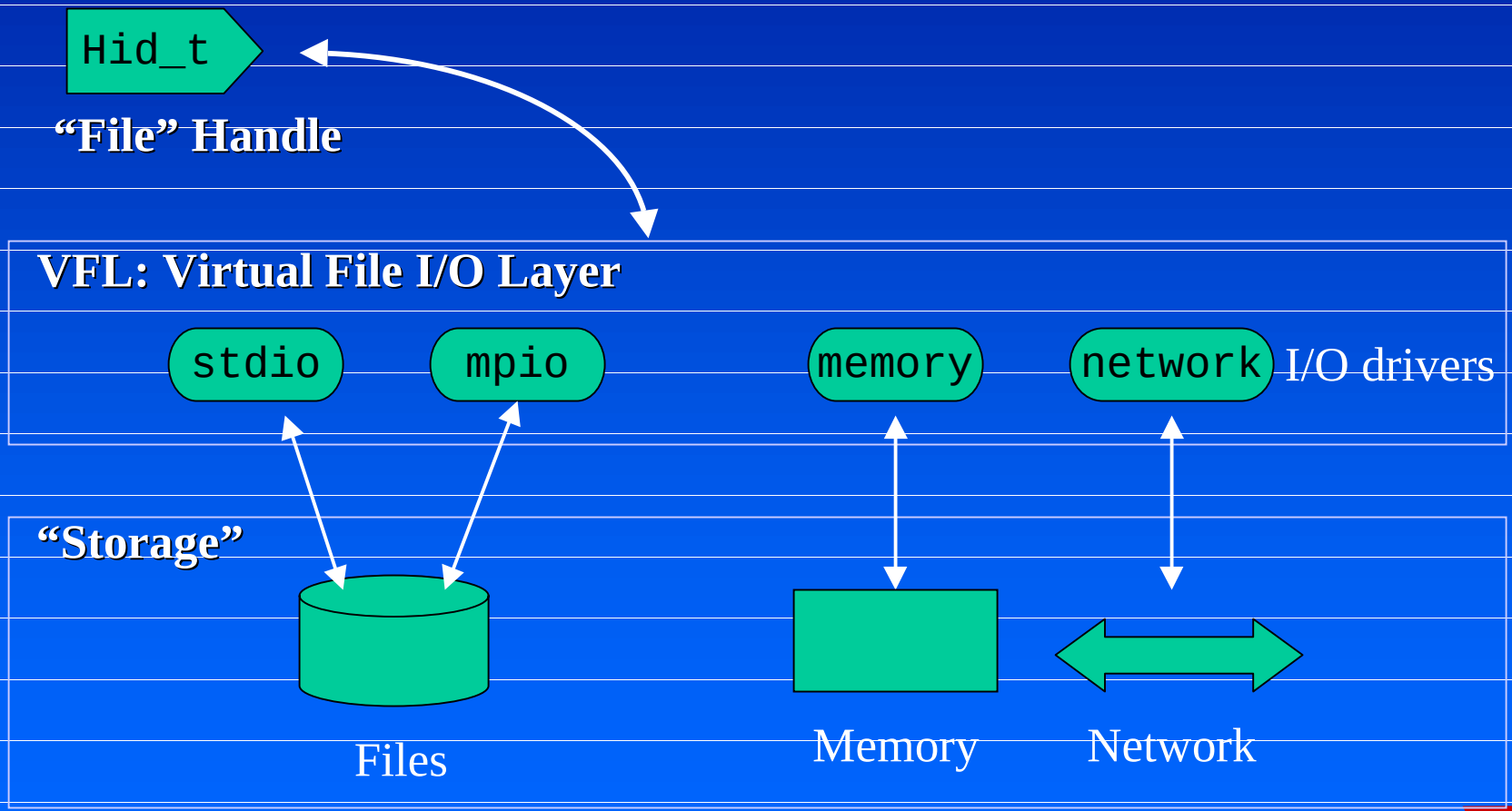
Mounting



HDF

Files needn't be files - Virtual File Layer

VFL: A public API for writing I/O drivers



HDF

File Families

- To access files larger than 2GB on file systems that don't support large files
- Any HDF5 file can be split into a family of files and vice versa
- A family member size must be a power of two

HDF5 tools

- Current
  - hdf5ls - lists contents of HDF5 file
  - h5dumper - higher level view
  - hdf5➔hdf4 converter
- Future
  - Convert HDF5 ⇔ ascii, binary, GIF, etc
  - Convert HDF4 ➔ HDF5
  - Java tools - VisAD, etc.
  - File/code generation from DDL description
  - Talking to vendors

Java applications

- HDF APIs
  - Basis for tools that access HDF
- HDF Viewers
  - HDF browser/visualizer
- HDF4 Data Server Prototype
  - Lessons learned about remote access to HDF data

Remote Data Access

- Java for remote access
- WP-ESIP: DODS project
- Computational Grids (Globus/GASS)
- The SDB: Web-based Server-side Data Browser

Other HDF5 activities

- Performance tuning
- Fortran and C++ API
- Thread-safe HDF5
- Object model

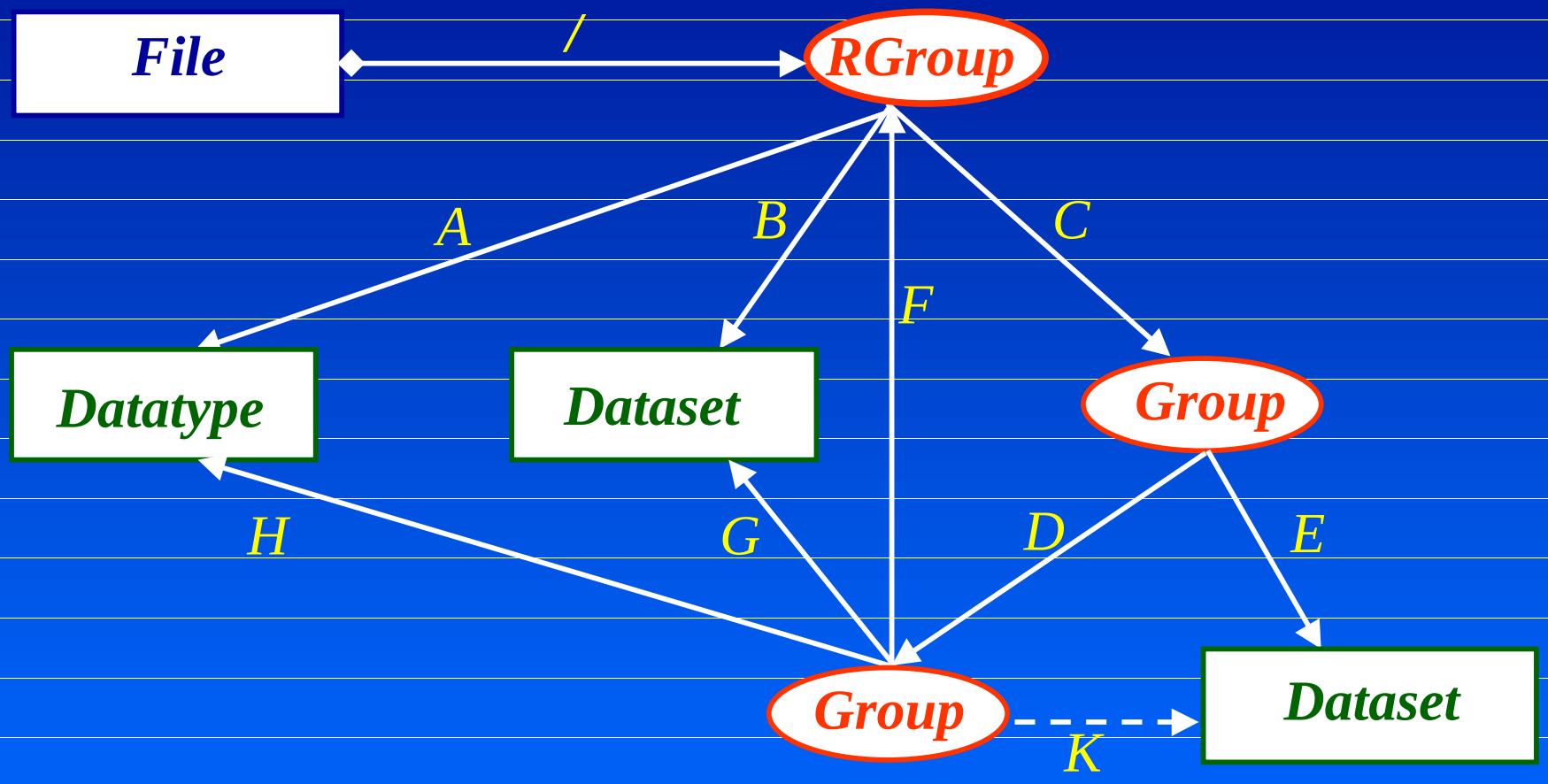
HDF5 Abstract Data Model

Classes, Objects and their relationships

HDF5 File (logical view)

- Directed graph with an entry point (root group)
- nodes are HDF5 objects :
  - Group, Dataset, Dataspace, Datatype
  - each object may have more than one path to it
- edges are inclusion directions (HDF5 links)
- graph may have:
  - loops
  - isolated nodes
  - “dangling” edges

Example of an HDF5 File



Dataset path : /B /C/D/G

HDF

UML Notation

Book:

UML Distilled

Applying the Standard Object Modeling Language

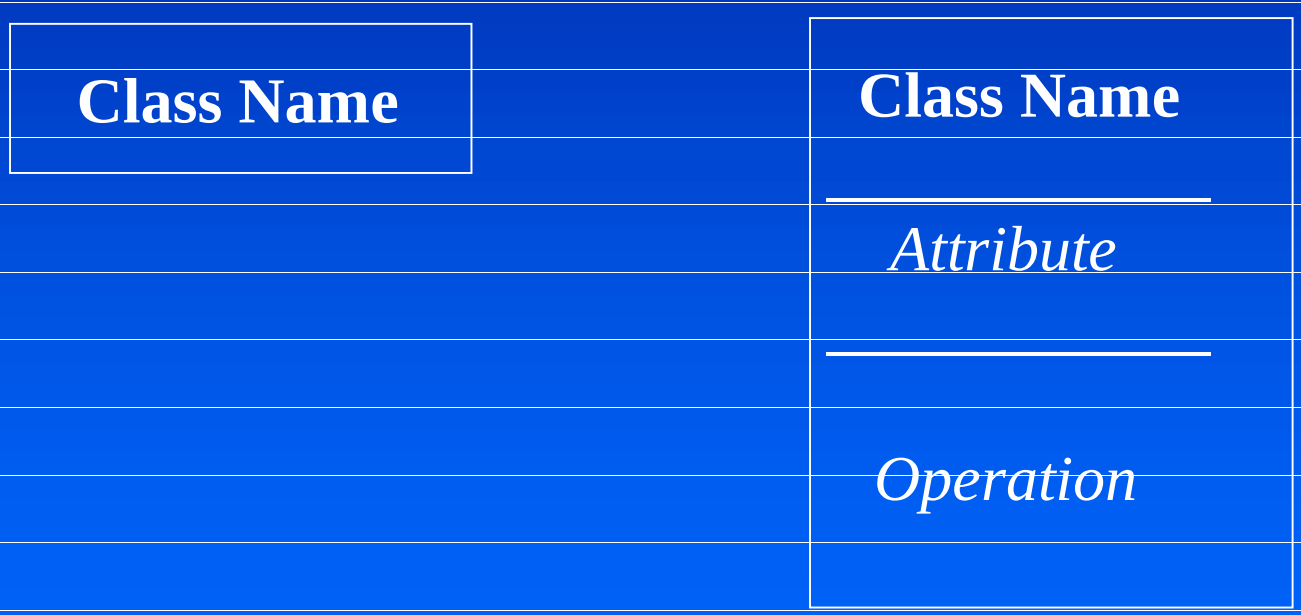
by Martin Fowler

Booch Jacobson Rumbaugh Object Technology Series

Addison-Wesley

UML Notation

Class



HDF

UML Notation

Association

Describes connection between object instances, should be a verb.

*Object A **has** object B associated with it.*

Multiplicity of Association

*Object A **has** zero or more objects B associated with it.*
*Object B **belongs** to at least one and up to N objects A*

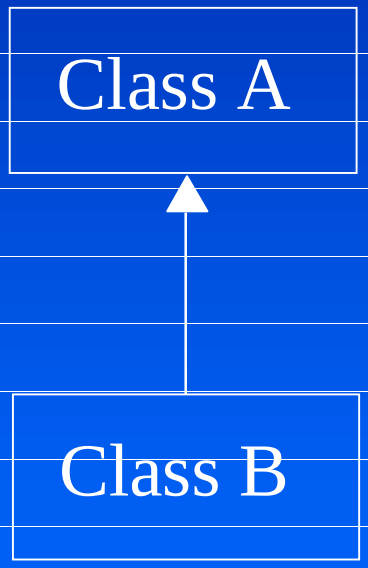
HDF

UML Notation

Generalization(Inheritance)

*"**is a**" relationship*

B is an A

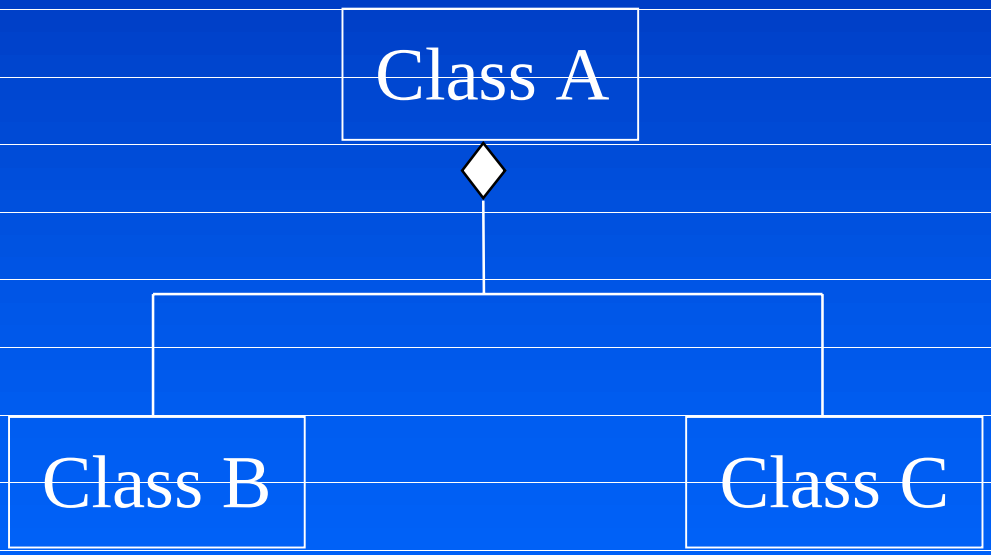


UML Notation

Aggregation

*"**a part of**" relationship*

*Objects B and C are **part of** object A*

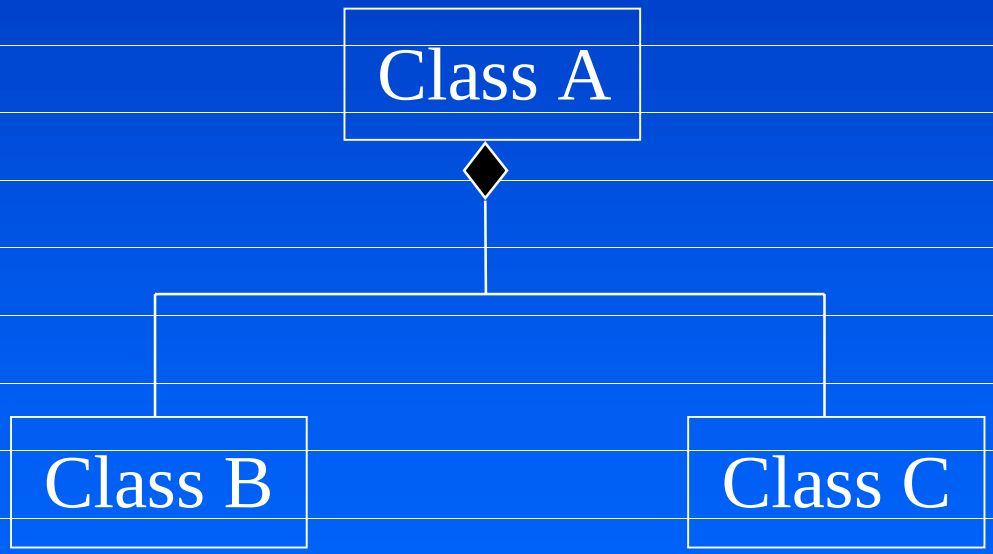


UML Notation

Composition

*"**a part of**" relationship*

Objects B and C "live and die" with A



HDF

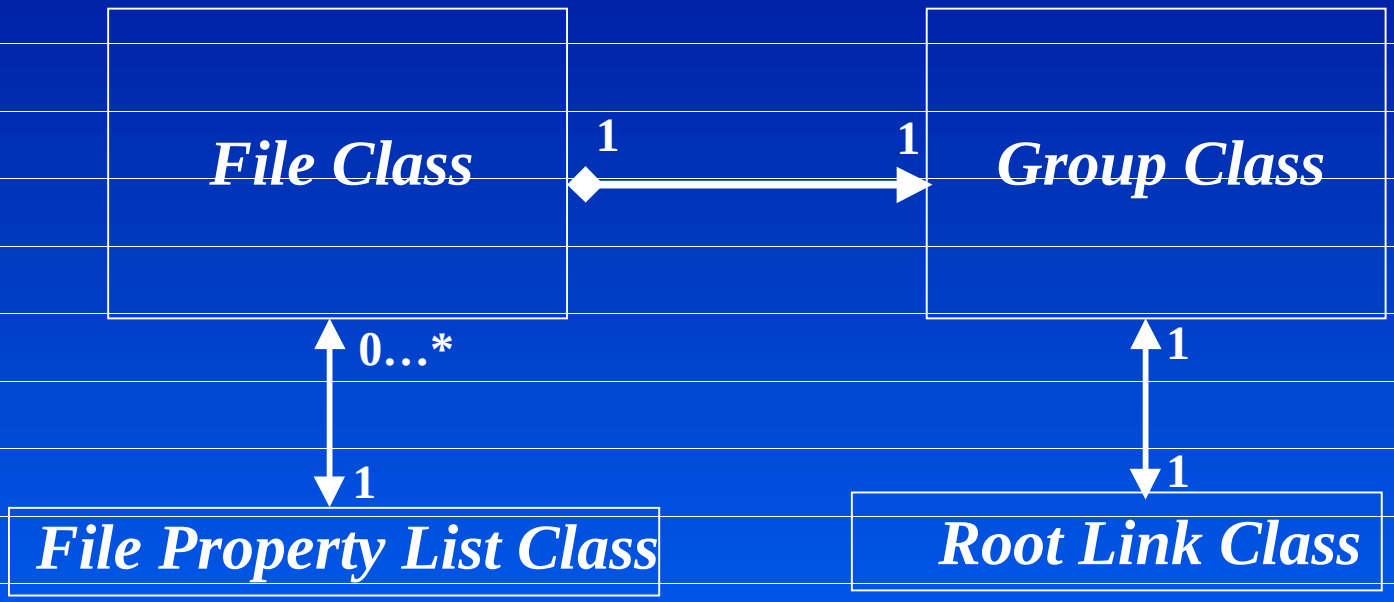
HDF5 Classes

- File Class
- Group Class
- Dataset Class
- Dataspace Class
- Datatype Class
- Named_Object Class
  - Group
  - Dataset
  - Named Dataspace
  - Named Datatype
- User_Defined_Attribute Class

HDF5 Classes

- Link Class
  - Root Link
  - Non-Root Link
    - Soft Link
    - Hard Link
- Property List Class
  - File Property List
    - Creation Property List
    - Access Property List
  - Dataset Property List
    - Transfer Property List
    - Storage property List

File Class File , Group Class and Root Link Class Association Diagram



File is a composition of a ***Group.***
Root Link is created when ***File*** is created.
Root Link points to the ***Group*** which is called ***Root Group***.
Root Group is automatically created/opened/closed when ***File*** is created/opened/closed.

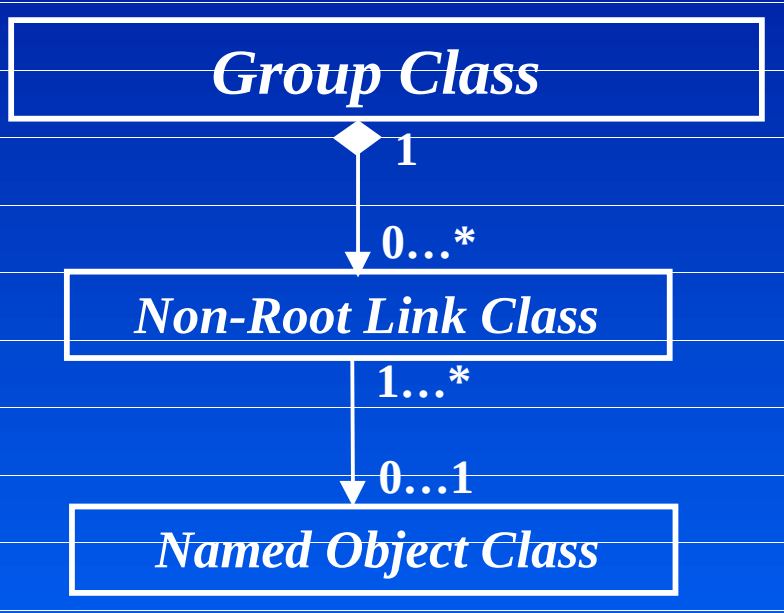
HDF

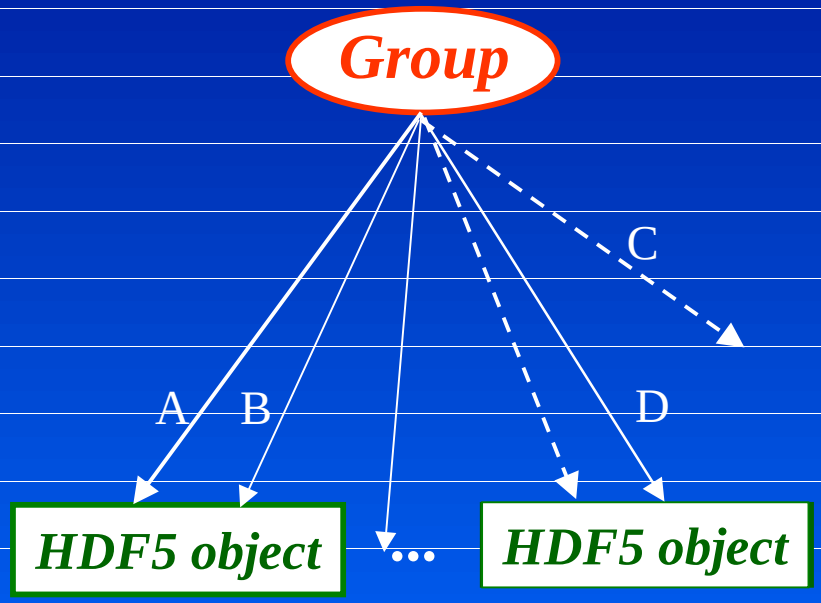
File and Group Classes

File Class
open/close
create
is_hdf5
get_create_prp
get_access_prp
mount/unmount
reopen

Group Class
open/close
iterate
get_object_info
get_link_value
get/set_comment

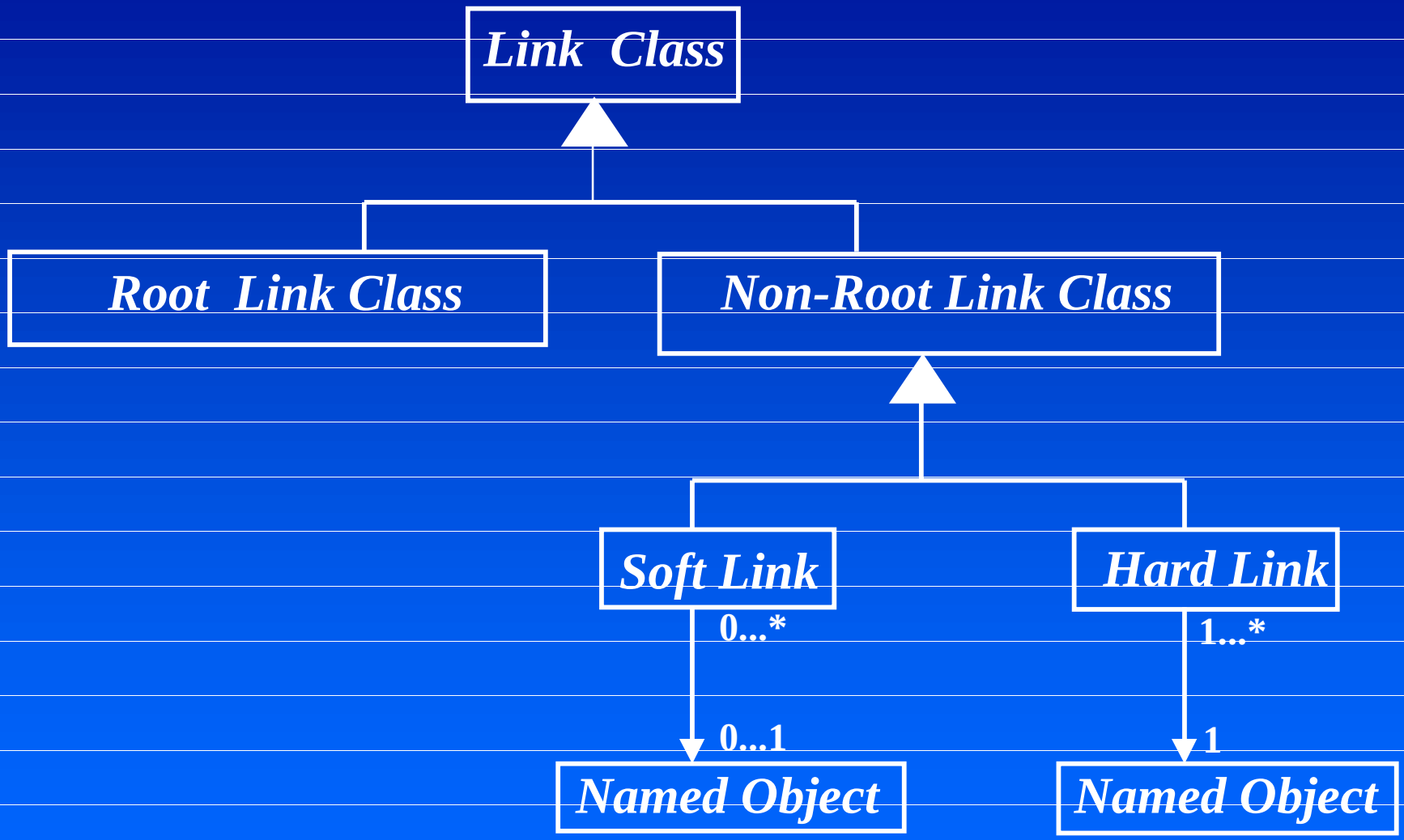
Group Class and Associated Classes





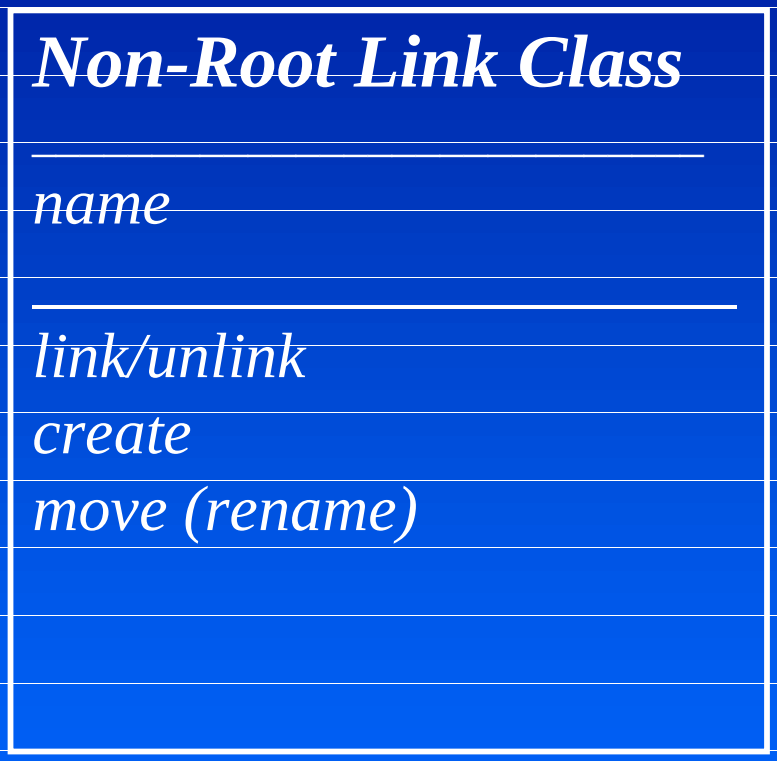
HDF

Link Class



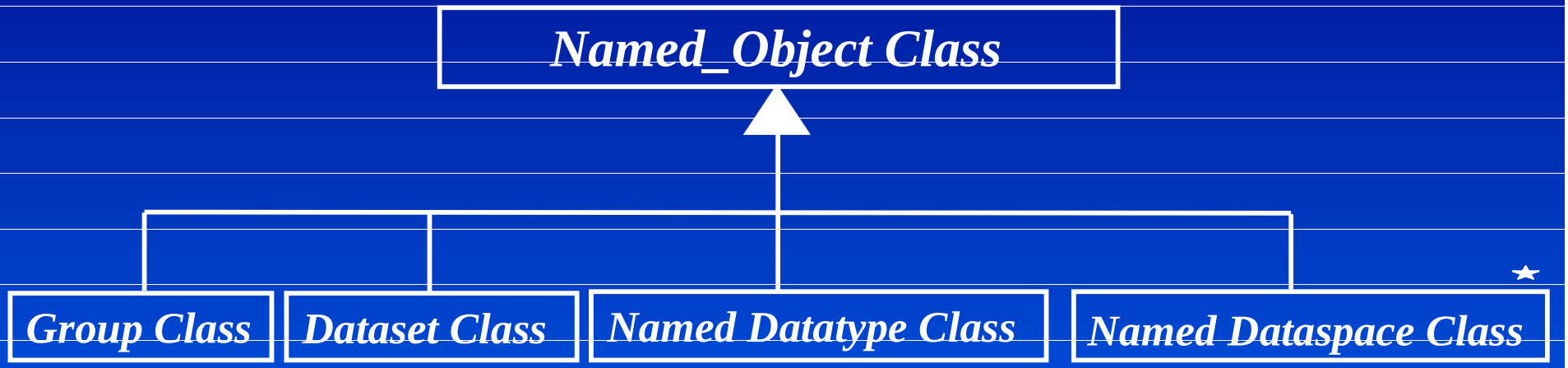
HDF

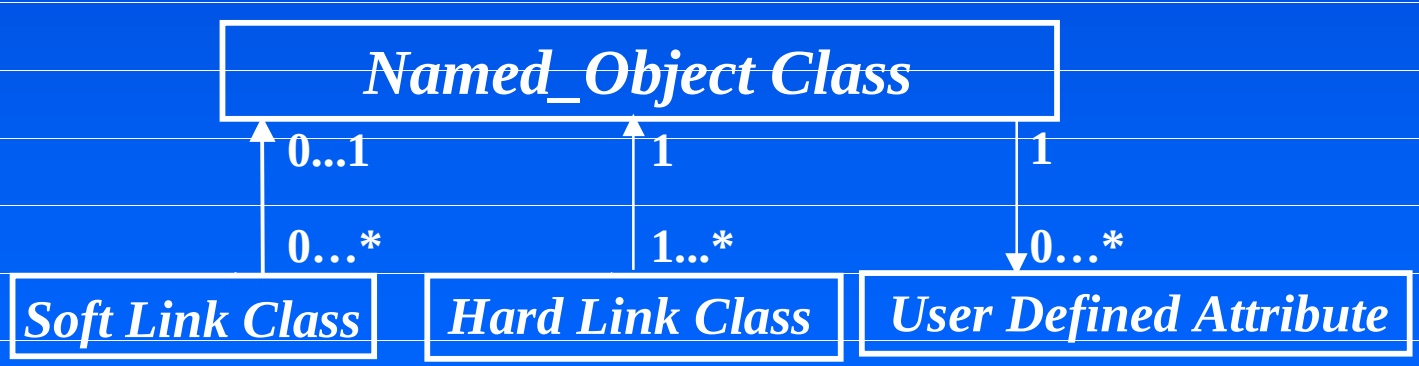
Non-Root Link Class



HDF

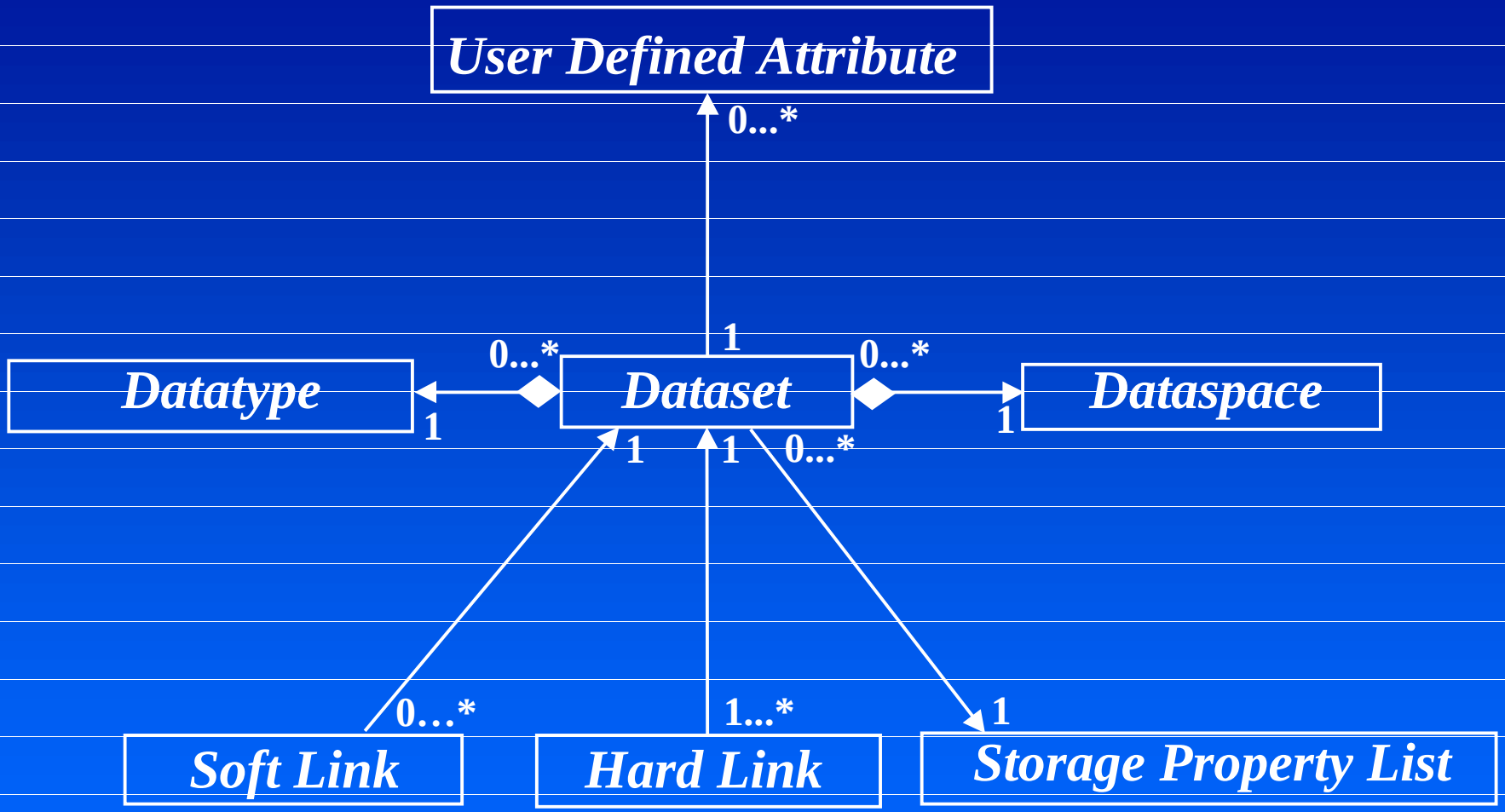
Named_Object Class





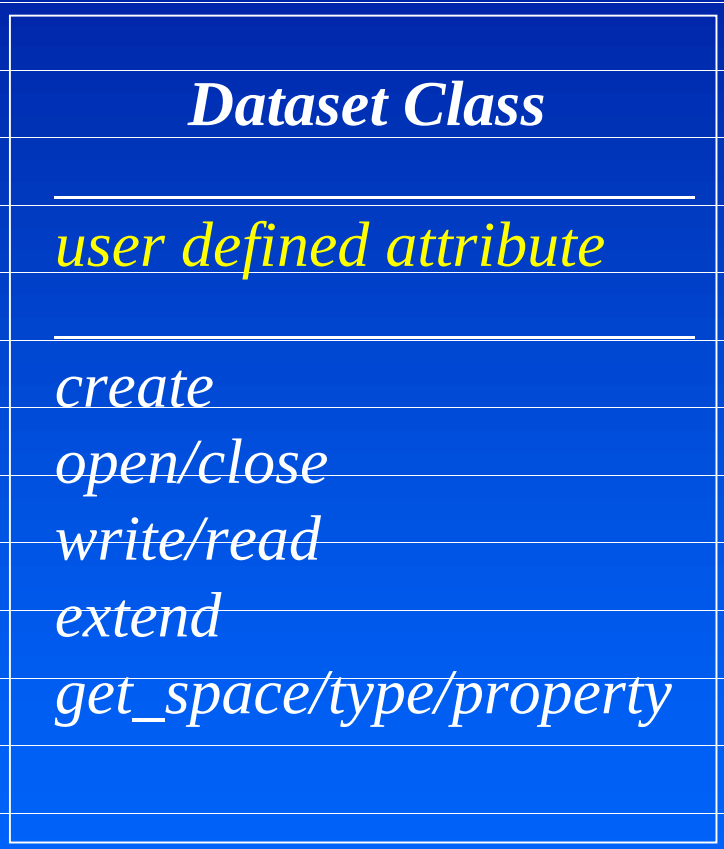
★*Not implemented yet*

Dataset Class and associated Classes



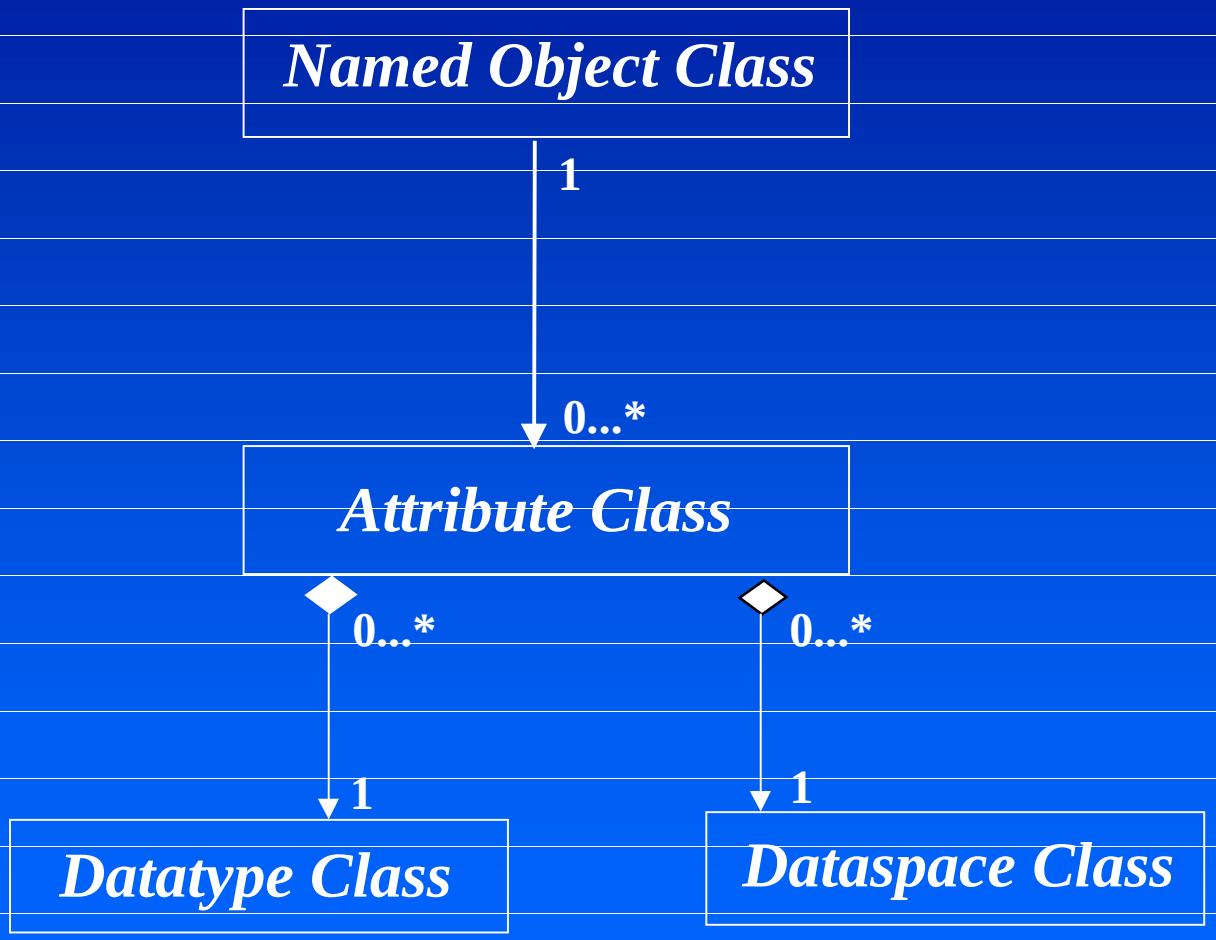
HDF

Dataset Class



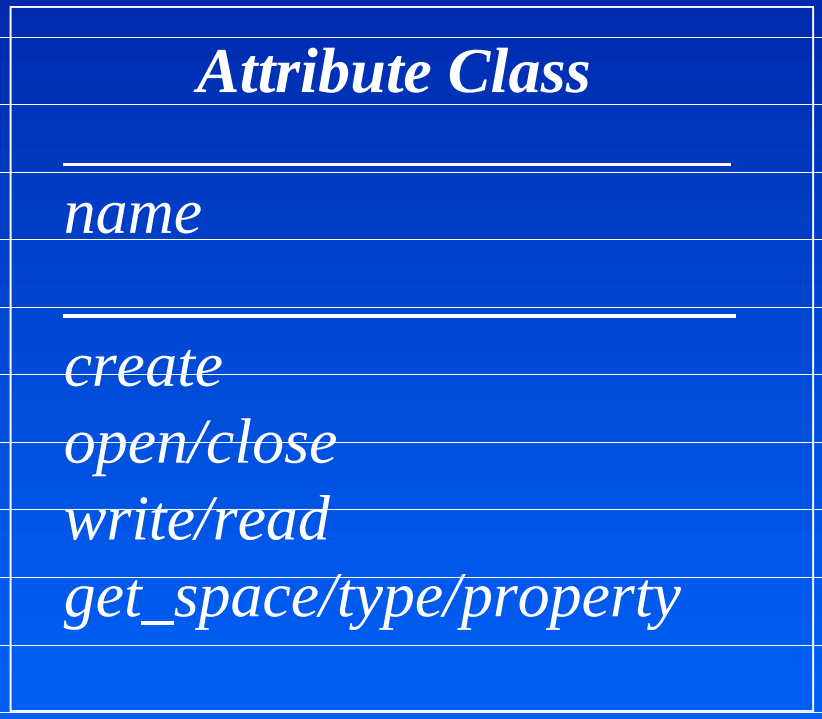
HDF

Attribute Class and associated Classes



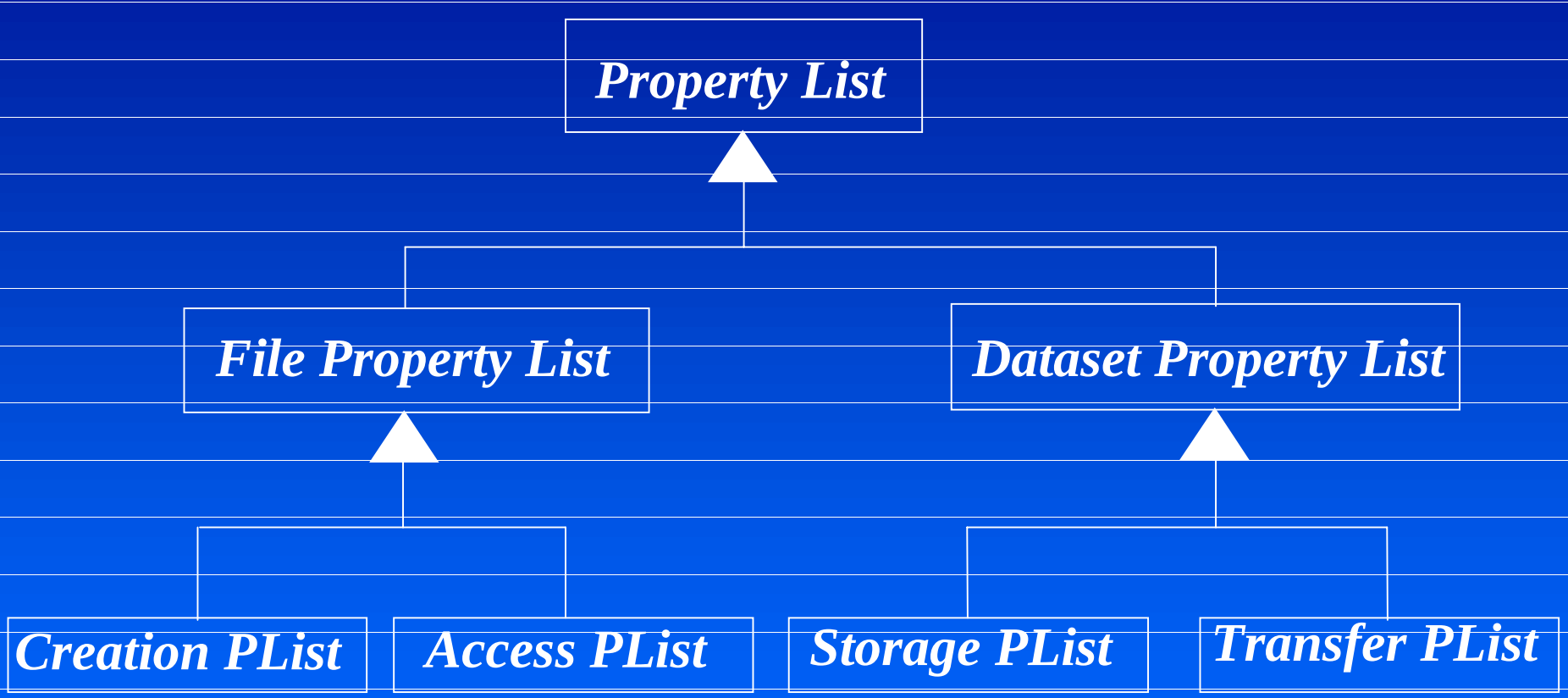
HDF

Attribute Class

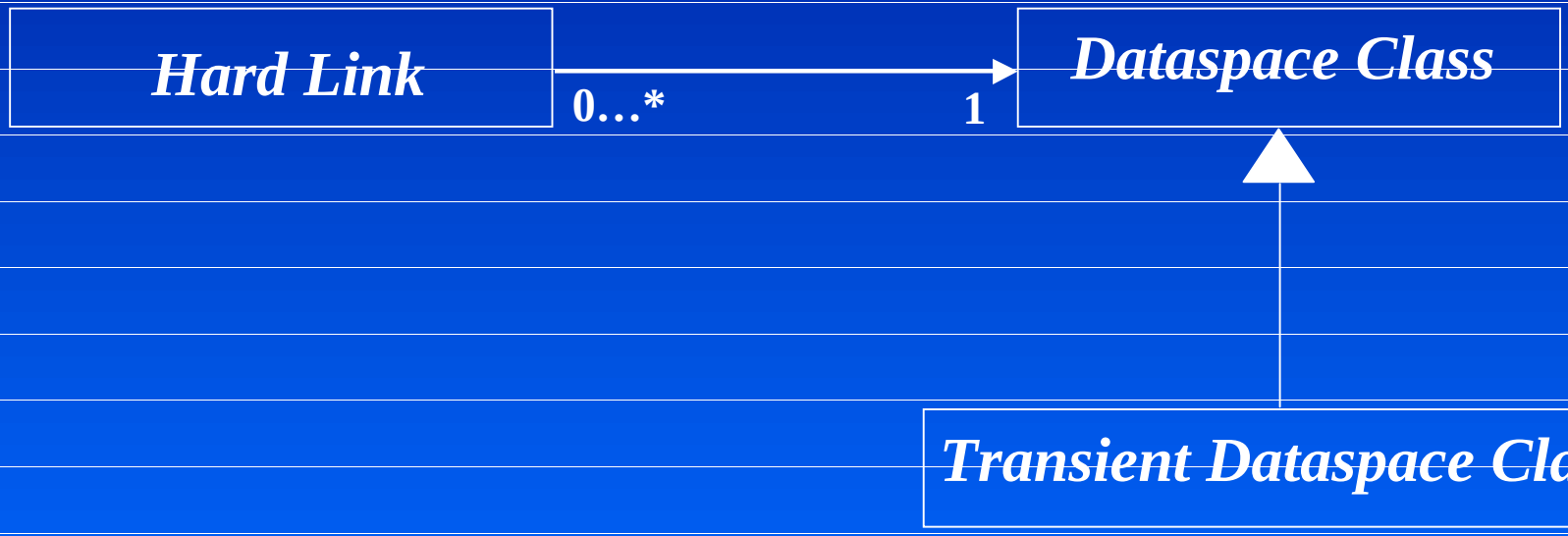


HDF

Property List Class and Subclasses

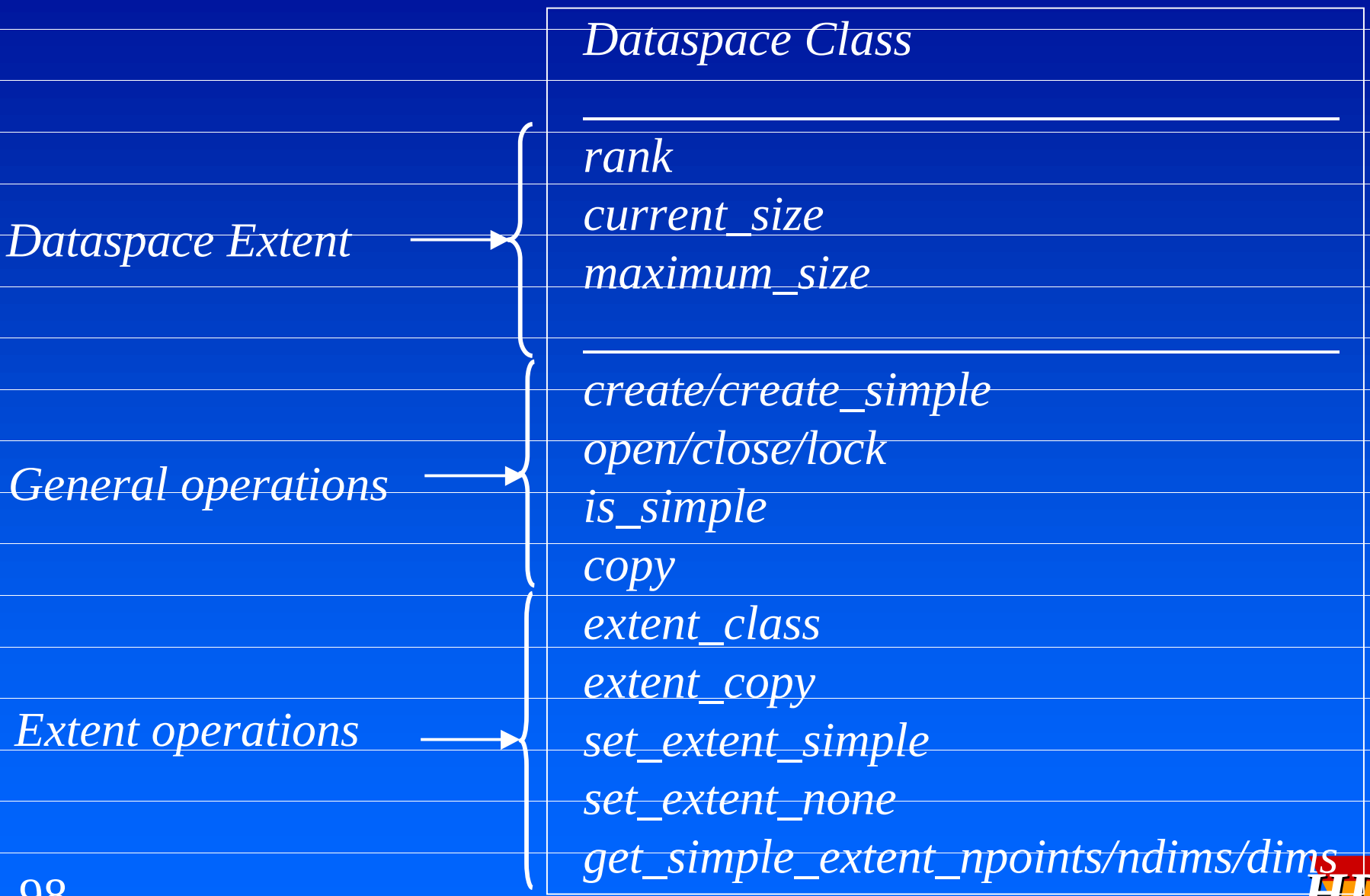


Dataspace Class



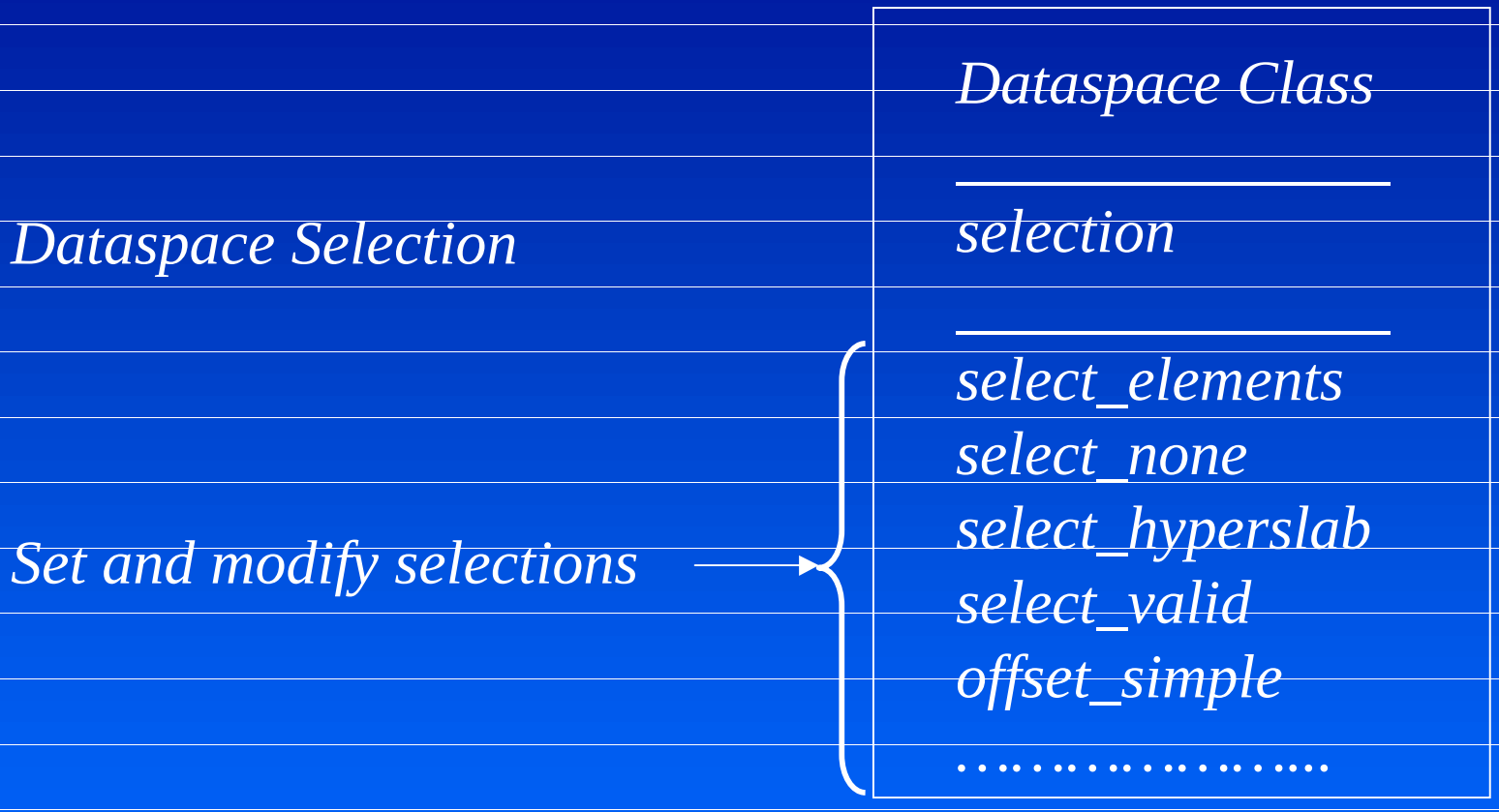
HDF

Dataspace Class



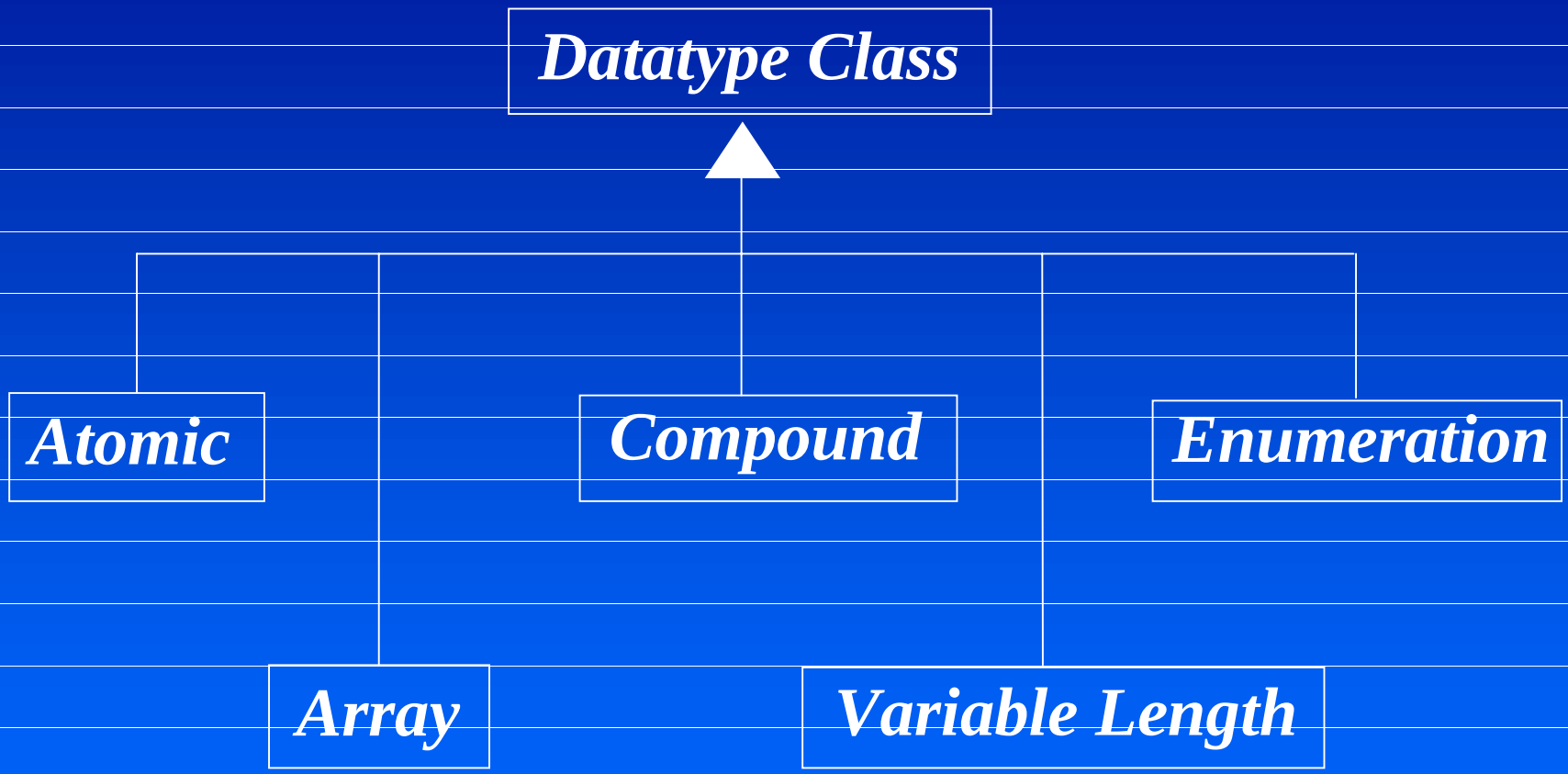
HDF

Transient Dataspace Class



HDF

Datatype Class and Subclasses



HDF

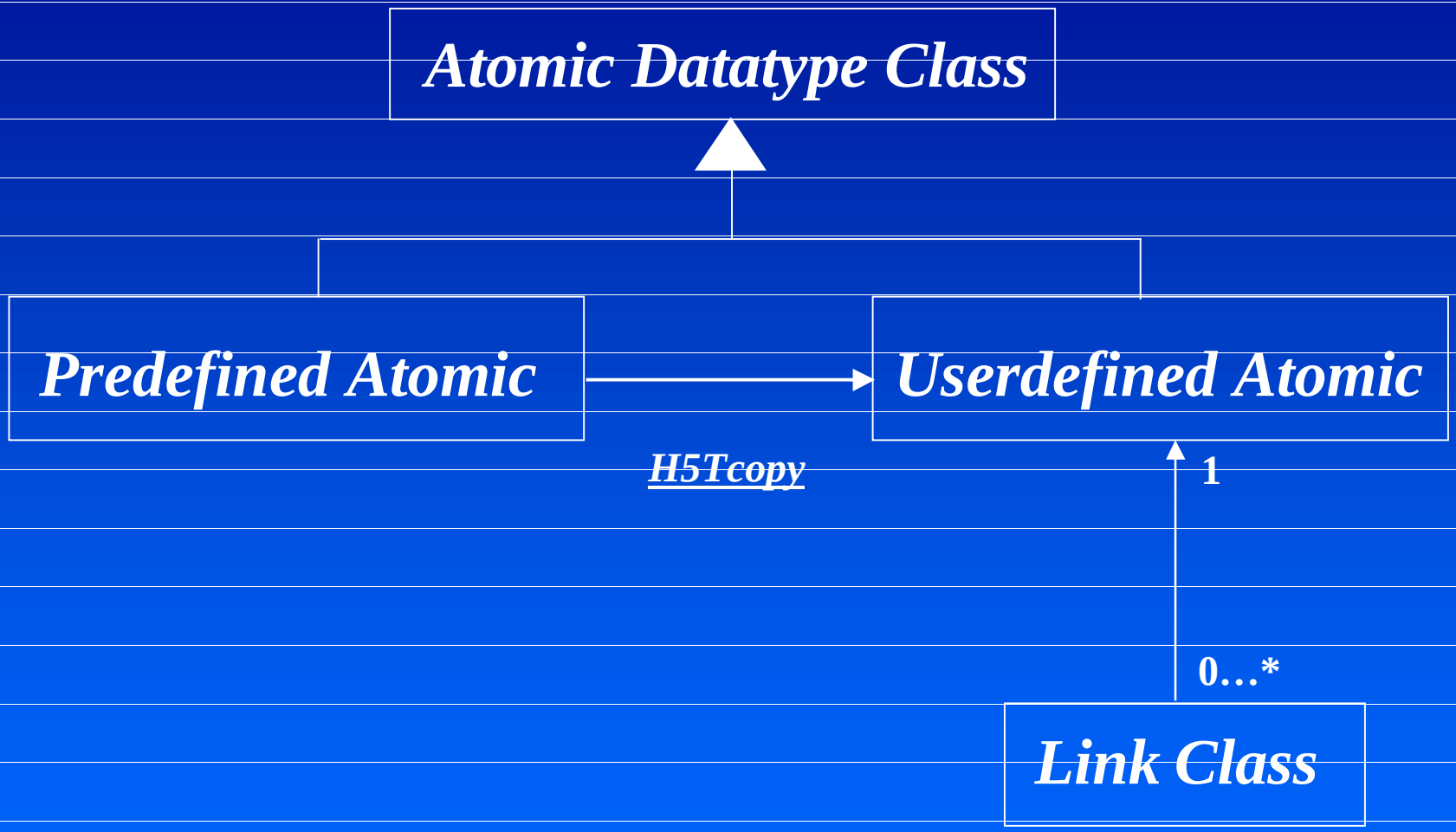
Datatype Class

Datatype Class

---

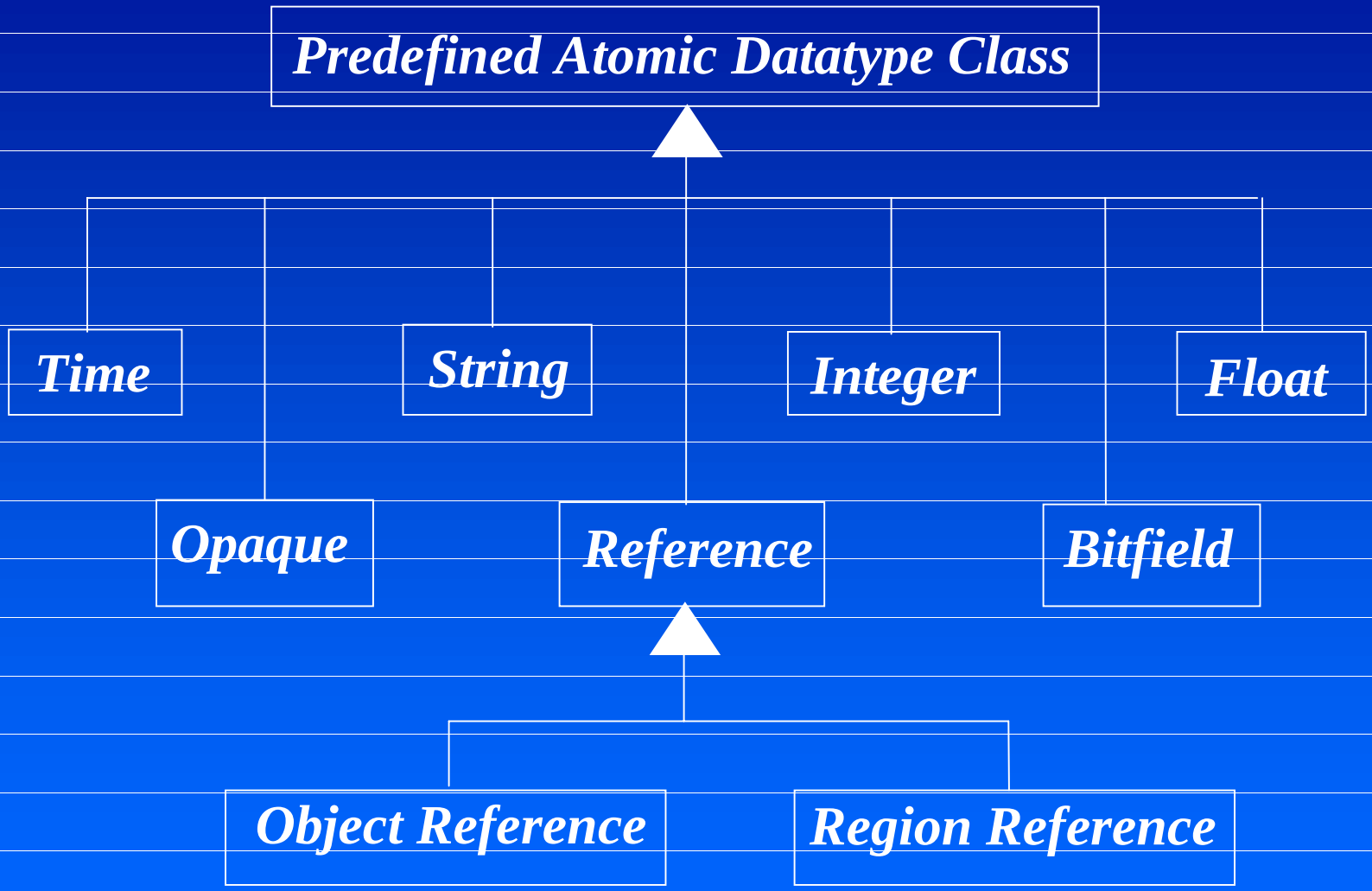
copy
equal
close
convert
get_class
get_size

Atomic Datatype Class and Subclasses



HDF

Predefined Atomic Datatype Class



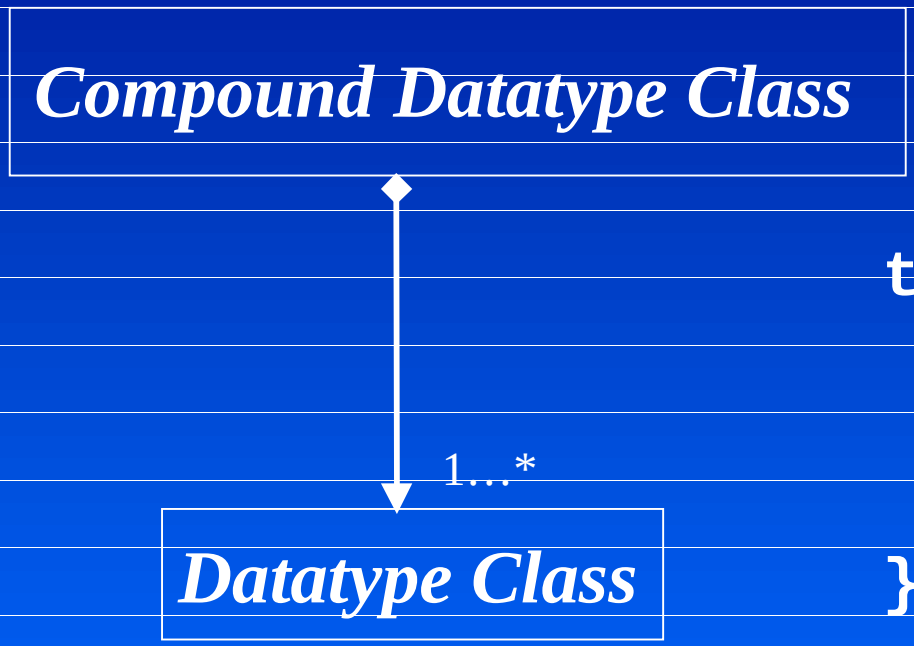
HDF

Examples of Predefined Atomic Datatype

Predefined Atomic Datatype
name = H5T_arch_base

H5T_IEEE_F64LE	Eight-byte little-endian, IEEE floating-point
H5T_STD_U16BE	Two-byte big-endian, unsigned integer
H5T_C_S1	One-byte null-terminated string of eight-bit characters
H5T_CRAY_F64	Eight-byte Cray floating point
H5T_STD_ROBJ	Reference to entire object in a file
H5T_NATIVE_LONG	**long**

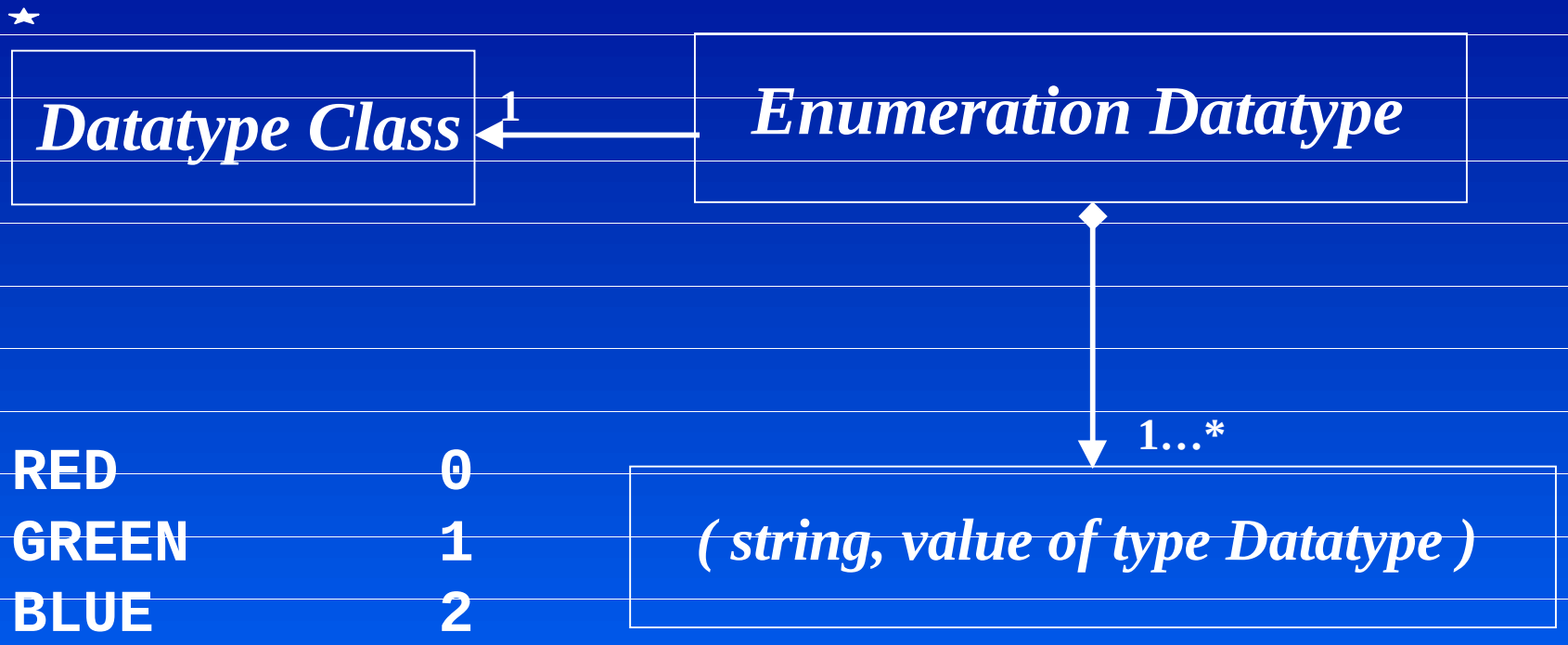
Compound Datatype Class




```
typedef struct {
        int a;
        float b;
        my_type c[10];
} compound_t
```


HDF

Enumeration Datatype Class

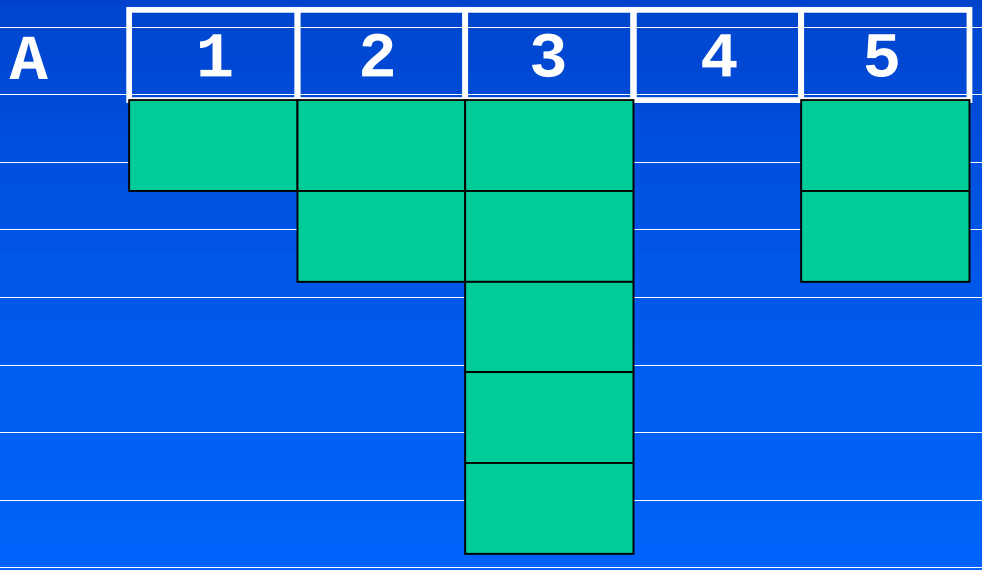



```
RED      0
GREEN    1
BLUE     2
```

★ *Only Integer datatype for now*

HDF

Variable Length Datatype Class



C

RED	GREEN	BLUE

HDF

HDF Information

- HDF Information Center
  - http://hdf.ncsa.uiuc.edu/
- HDF Help email address
  - hdfhelp@ncsa.uiuc.edu
- HDF users mailing list
  - hdfnews@ncsa.uiuc.edu