

Request for Comments

High Level Indexing in HDF5

Mike Folk, Quincey Koziol, James Laird, Rishi Sinha

May 2005

Version 0.3 by Mike

1. Abstract

2. Purpose

The purpose of this paper is to identify a model to support an indexing prototype in HDF5. This prototype derives from the requirements identified in the RFC “Requirements for Indexing Prototype in HDF5.”

3. Background

A companion document, “Request for Comments – Requirements for Indexing Prototype in HDF5” [1], identifies a range of possible uses of indexing in HDF5, and proposes a set of basic operations.

In the current study, the goal is to explore a small part of that range – the study will look for ways to allow simple, one-dimensional, indexing in HDF5. At the same time, understanding the potential for other types of indexing, a second goal of the study will be to provide a framework for future indexing efforts in HDF5.

4. Data model

4.1 Basic terms and conceptual structures

Index. An *index* is a data structure that allows us to quickly identify and locate *target objects* according to some criterion or criteria. An index may be created either for a *key field* in a table of records, for a combination of key fields, or for a *variable* represented by a dataset.

Query. An index *query* is an operation that describes and applies the search criteria used to locate an index record. A query is specified in terms of one or more keys, and returns table records.

Indexed file. An *indexed file* is one for which one or more data structures (indexes) exist for finding records or other objects of interest in a file. Indexes can be part of a file, or they can be in a separate file, part of a database, or elsewhere.

Target objects. Target objects include

- File objects – datasets, datatypes, and groups
- Dataset elements
- Selection regions (groupings of elements) within a dataset

{“dataset element” is a degenerate case of “selection region.” Is it ok to separate them?}

Index record. Each entry in an index will be called an *index record*. An index record may contain the values to which the criteria for identifying and locating objects are based (key fields), information used to locate the corresponding object (location fields), and possibly other information.

An index record may not require a key field if records in the index are located based only on their position in the table. Such an index will be referred to as an *implicit index*.

Key fields. Several fields in an index record may be specified as key fields. Keys may be designated as “primary” or “secondary.”

- A *primary key* uniquely defines a record – no two records may have the same value in the primary key field. There can be only one primary key field in an index
- A *secondary key* need not be unique – more than one record may share the same value in a secondary key field. There may be more than one secondary key fields in an index.

Index structures are organized so as to facilitate fast access based on key values. For instance, if an index stored as a table is sorted by a certain key field, a binary search may be used to look up keys in the table.

Location fields. *Location fields* can come in several forms. They can include the following, possibly in combination.

1. An *object identifier*, if the target is an independent object, such as an HDF5 dataset or group.
2. The *coordinates of a dataset element*, if the target is an element in a certain dataset.
3. An *HDF5 region reference*, if the target is a selection of elements within that dataset.
4. If there are more than one consecutive object at a particular location corresponding to a particular key value, there may also be a *count* field specifying the number of such objects.
{This is a little complicated, and probably shouldn't be done, at least not for whole HDF5 objects. I'm thinking of two things: (a) collections of objects, as in a group of objects, and (b) a set of consecutive records in a table. Case (a) is ambiguous, since the criteria for ordering thing ordering in a group is not unique. (b) is covered by #3 (region reference), but a region reference seems like overkill in this case, and also possibly harder to decipher by someone looking at the index.}

5. A general indexing model

5.1 Conceptual structures

An HDF5 index is a lookup table with zero or more key fields and one target object field. One or more key fields is sorted. One key field may be identified as a primary key. The datatype of a key field can be scalar numeric, enumerated, and string. The key fields do not have to have the same names as their counterparts, from which they may have been derived.

The target object field may include any number of target objects.

Indexes can be distinguished by type of target object:

- a. **File object index:** an index whose target objects are HDF5 objects.
- b. **Dataset element index:** index whose target objects are elements in a dataset.

For file object indexes, target objects can be HDF5 datasets, groups or datatypes.

For dataset element indexes, target objects can be a single element, or an HDF5 dataspace selection region. {In the latter case, do we distinguish between the case where all target objects for a given index are members of the same dataset and the case where they may be in different datasets?}

5.2 Operations

The following operations are included in the initial set of requirements.

1. Create_index
2. Query
3. Delete index
4. Cache index
5. Import index and export index

1. create_index

Index creation creates a table identified as an index in an HDF5 file.

For input, create_index requires a table consisting of fields corresponding to the sort keys, and a field containing target locations. In addition, input must include a list of the key values to be sorted, and the precedence for sorting each key, and the names to be assigned to the key fields in the index table.

Output is in the form of a lookup table, written to an HDF5 file and identified as an index.

2. Query

The select operation applies a query to one or more fields (keys) in a table, determines which records satisfy the query, and produces an array of record numbers for these records.

{There is debate about whether this function should apply only to indexes, or to all applicable structures. E.g. should they apply to tables that are unsorted? Different search algorithms would be used for the different structures. If it's just applied to tables, then it's pretty simple, but if applied more generally, at some level it almost certainly needs to be separate functions.}

For input, select requires a table and a query string. The query specifies a lexical or arithmetic range for each key, with each of these combined by Boolean operation. E.g. if the key are A and B, an example of a query string would be "(A>3 AND A<7) AND B=10". The full query syntax is yet to be determined. It has been proposed that limited

compound queries be permitted.

Output is in the form of an array listing the record numbers for records that satisfy the query.

When applied to an index, the select operation recognizes the sorted fields and takes advantage of these for improved efficiency.

3. delete_index

The delete_index operation removes an index from an HDF5 file and updates or removes any corresponding metadata from the file. If the index is cached, it removes the cached index from memory but does not touch any corresponding index in the file.

4. cache_index

The cache_index operation reads an index from an HDF5 file into memory for optimal speeding up queries and other operations. Cache_index supplies a special object id for the in-memory index. A cached index may be accessed using any of the other index operations. {Does this make sense?}

5. import_index and export_index

The import_index operation imports an index from a text-based format such as XML.

The export_index operation exports an index in a text-based format such as XML.

The precise format to be used is to be determined.

5.3 Data structure

{We're still investigating where to go with this. We're starting very simply, but if Rishi has time, he may make something more general.}

As discussed in [1], an index might be implemented in a number of ways, depending on the types of operations to be performed. Given the modest goals of the current model, a simple HDF5 table will suffice. A table can be described in terms of the HDF5 model as a one-dimensional dataset in which each element is an HDF5 compound datatype, representing an index record. When tables are sorted according to the appropriate search criterion, they can be easily searched to locate records.

Minimally, fields in an index table will include:

- A field for each key.
- A field for target objects.
- Alternate extra fields provided by applications.

The target object field will contain a variable-length datatype, allowing more than one target object per index key value. The datatype of the target object field will vary depending on the type of target object, and may include object references, selection region references, and integers.

Index attributes

Since this implementation of HDF indexes uses HDF5 tables, they can inherit the attributes of tables:

CLASS will have the value “TABLE”

FIELD_(n)_NAME will have the names of fields, including key fields and target object field.

Additional attributes to be considered:

TABLE_SUBCLASS, indicating this is an index. It might have the value “INDEX”.

INDEX_PRIMARY_KEY_FIELD (optional), identifying which field is the primary key.

INDEX_SECONDARY_KEY_(n) (optional), identifying secondary key fields.

INDEX_TARGET_FIELD, indicating which field contains target objects.

INDEX_TARGET_TYPE, indicating type of target object. There may be several properties for a given target type, making this attribute somewhat complex.

{What else?}

6. References

1. “Request for Comments – Requirements for Indexing Prototype in HDF5”.