Testing considerations for chunk query functions

This working document outlines testing considerations for the chunk query functions. Its content will be extended based on the feedback and discussions and when finalized, will be included in the “Chunk query functionality in HDF5” RFC.

1. **Type of chunked datasets we need to test with**

Currently HDF5 has 6 different ways of indexing chunks. See Sections III.A.1 and Appendix C of the [File Format Specification](https://portal.hdfgroup.org/display/HDF5/File+Format+Specification). All query functions should work chunked datasets that exercise all 5 indexing schemas:

* 1. Chunked dataset that use B-tree v1 structures to index chunks. This structure was default structure for all chunked datasets (fixed or unlimited dimensions) in the versions prior to 1.10.0. Since we are dropping support for HDF5 1\_8 branch, the feature and the tests will be added only to 1\_10 and later versions. To create such datasets one must exercise the H5Pset\_libver\_bounds function with the “high” version set to H5F\_LIBVER\_V18 when opening a file in which the chunked dataset is created.
  2. Chunked datasets that use structures described in Appendix C should be created in a file with access property set by the H5Pset\_libver\_bounds function. The “low” and “high” parameters should be set to H5F\_LIBVER\_V110. Such settings will guarantee that the library will use chunking indexing schemas available in 1.10.\* releases only. If new major release introduces new chunking indexing, then one should add a new test that explicitly exercises new version H5\_LIBVER\_V\*. H5F\_LIBVER\_LATEST can be used in the tests of the develop branch, but will need to be changed to the setting that corresponds to the specific major release version when the version is known.

The list below outlines 5 major types of chunked datasets in 1.10 version of the library:

* + 1. Chunked dataset with one chunk.
    2. Chunked dataset of fixed size, without compression, and with space allocation is H5P\_ALLOC\_TIME\_EARLY
    3. Chunked dataset with fixed size dimensions
    4. Chunked dataset with only one unlimited dimension (doesn’t matter which one)
    5. Chunked dataset with 2 or more unlimited dimensions
  1. All tests should be run for the datasets outlined above in a-b. Data should be checked with using C read call and then applying datatype conversion and decoding is necessary. See [*H5Tconvert*](https://portal.hdfgroup.org/display/HDF5/H5T_CONVERT) function. Note: If we always create datasets in our tests and don’t use pre-created ones we can use H5T\_NATIVE\* when we create datasets and will not need to do datatype conversion.

To simplify tests we can also use *[H5Dread\_chunk](https://portal.hdfgroup.org/display/HDF5/H5D_READ_CHUNK)* and compare its data with data read by C read call. (🡺 check if we can open file with the system call due to the locks we introduced for non-SWMR mode; we probably will need to open/close HDF5 file and then open/close with C open/read/close. 🡺 check how this will work on Windows).

1. Modification, addition and deletion of chunks

Our tests should exercise new functions when a chunk is modified for a compressed dataset and can be written to the new location. (🡺 need to find out if chunk is written to the same location when its size is smaller than original). One can easily “shrink” or “blow up” chunk by using Gzip compression and data like all 0s, or random that will not compress well. As usual, this type of test should be skipped if compression is not available.

Chunk addition should be done for all types of datasets described above, while chunk deletion can be exercised only for datasets with unlimited dimension(s).

1. Compression consideration

We do not need to add anything special for compression except what was mentioned in the previous section.

1. Partially written chunks consideration

HDF5 library has a feature of [not applying compression to partially written chunks](https://portal.hdfgroup.org/display/HDF5/H5P_SET_CHUNK_OPTS). It would be good to tests the query functions with this feature.

1. Exercising hsize\_t sizes

We will need to write chunks beyond 2GB boundary and force coordinates and index go full range.