Unifying the CDK Data and HBC APIs

Tom White, September 2013, v1

The conceptual mapping between CDK Data and HBC

The public APIs for CDK Data and HBC (as it currently stands) have types which are in direct correspondence, as mapped by the following table:

Data Core Type	**HBC Type**
Dataset	Dao
DatasetReader	EntityScanner
DatasetWriter	EntityBatch
PartitionKey	PartialKey
DatasetDescriptor	KeySchema and EntitySchema
DatasetRepository	n/a
MetadataProvider	SchemaManager

In CDK Data the core type is the Dataset, which provides access to streaming readers and writers (DatasetReader and DatasetWriter) and partitioning (using PartitionKey objects). Dataset metadata is encapsulated by a DatasetDescriptor, which is stored in a MetadataProvider. A DatasetRepository is used to store and retrieve named datasets, using a MetadataProvider to handle metadata storage.

In HBC, the core type is Dao, which offers key-based random access and update (get/increment/put/delete) as well as streaming readers and batch writers (EntityScanner, EntityBatch). PartialKeys are used to partition the dataset so all the records with a given set of key fields can be read back. Dataset metadata is encapsulated by a SchemaManager. Rather than having a single schema per dataset, there are two: one for the key and one for the entity.

Consolidating CDK Data and HBC APIs and naming

The correspondence is more than a loose analogy; the two APIs could share types and follow the same naming conventions. The key insight here is that the existing CDK Data API deals with unordered collections of entities, while HBC is suitable for entities that have an associated key.

The following table summarizes the naming changes:

Existing name	**Proposed new name**
Dataset	BagDataset or SequenceDataset
DatasetReader	EntityReader or ValueReader
DatasetWriter	EntityWriter or ValueWriter
Dao	MapDataset
EntityScanner	KeyEntityReader or KeyValueReader
EntityBatch	KeyEntityWriter or KeyValueWriter

The two types of dataset are (tentatively) called BagDataset and MapDataset, which would share a common supertype, Dataset, with two methods:

```
String getName();
DatasetDescriptor getDescriptor();
```

The other methods that are on Dataset today (getReader(), getWriter(), methods for handling partitions) would be moved to BagDataset. MapDataset's methods would be the ones from Dao.

DatasetRepository would be parameterized by Dataset type. For example, FileSystemDatasetRepository would become FileSystemDatasetRepository<BagDataset>, and we would introduce a class called HBaseDatasetRepository<MapDataset>.

Currently, DatasetDescriptor holds a single schema, it would need changing to hold an optional schema for the key.

SchemaManager can be absorbed into MetadataProvider as described in the following table:

SchemaManager method	**MetadataProvider method**	**Notes**
createSchema	create	
migrateSchema	update	
getKeySchema and getEntitySchema	load	Schemas are wrapped in a DatasetDescriptor

hasManagedSchema	exists	
getEntitySchemas and getEntityVersion	n/a	Add loadHistory or similar to MetadataProvider
refreshManagedSchemaCache	n/a	Add to MetadataProvider