Multipart Dataverse Names in AsterixDB

Dmitry Lychagin

Objectives

1. Allow dataverse names to contain multiple parts.
   a. Can be useful for product extensions
   b. Better aligns SQL++ with PartiQL that allows compound names in the database environment

2. Support multipart dataverse names in the following places:
   a. SQL++ grammar (DDL, DML)
   b. REST API
   c. Dataset functions: dataset_resources(), storage_components()
   d. Metadata datasets

3. The implementation must be backwards compatible

Example: DDL statements

1. **create dataverse** sales.east;
2. **create type** sales.east.orderType **as** { oid: bigint };
3. **create dataset** sales.east.orders(sales.east.orderType) **primary key** oid;
4. **create dataverse** sales.west;
5. **create type** sales.west.orderType **as** { oid: bigint };
6. **create dataset** sales.west.orders(sales.west.orderType) **primary key** oid;
7. **use** sales.east;
   insert into orders ({'oid' : 100, ... });

Example: Queries

1. **use** sales.east;
   select value o **from** orders **as** o

2. /* Backquote identifiers */
   use \`sales\`.\`east\`;
   select value o **from** orders **as** o

3. /* Fully qualified dataset name in FROM clause */
   select value o **from** sales.east.orders **as** o

4. /* Join */
   select *
   from sales.east.orders **as** e, sales.west.orders **as** w
   where e.oid = w.oid

The new Dataverse Name

The new Dataverse Name is an ordered list of strings (name parts).

Example: ["a", "b", "c"]

1. This form is a prefered user model.
   User should operate on ordered lists in
   a. SQL++ grammar,
   b. public REST API,
   c. public dataset functions
2. Metadata datasets are exempt due to backwards compatibility requirements.
   They store each dataverse name as a single encoded string (more on this later)
3. A dataverse name part can contain any character (including '.')

Scalar encoding of the new Dataverse Name (#1)

There's a scalar encoding of the new Dataverse Name that produces a single string.

It should be used in places where the ordered list model is:

- prohibited due to backwards compatibility requirements
  (Metadata datasets), or
- unnecessary because the dataverse name is not exposed to users in given context
  (private built-in functions dataset(), index-search(), etc).
  This also applies to function namespaces in Hyracks

Note:

- The scalar form is currently used as a directory name by the storage subsystem, but we need to change this because every file-system places restrictions on characters allowed in a file name.

Scalar encoding of the new Dataverse Name (#2)

This scalar encoding is called a "canonical form" in the API and is defined as follows:

1. escape_character = '@'
2. separator_character = '.'
3. prefix escape_character and separator_character in each name part
   with escape_character
4. concatenate these processed parts into a single string
   using separator_character as a separator

Examples:

1. ["Default"] is encoded as "Default"
2. ["a", "b", "c"] is encoded as "a.b.c"
3. ["a.b", "c.d"] is encoded as "a@.b.c@.d"
4. ["a@.b", "c@.d"] is encoded as "a@@@.b.c@@@.d"

Dataverse Name API (#1)

Class: org.apache.asterix.common.metadata.DataverseName

Static construction methods

- **create(List<String> parts)**
  general purpose method
- **createSinglePartName(String singlePart)**
  use when absolutely sure that the intent is to create a single part dataverse name.
  equivalent to *create(Collections.singletonList(singlePart))*, but slightly faster
- **createBuiltinDataverseName(String singlePart)**
  only used for built-in dataverse names ("Default", "Metadata", "Asterix", "Algebricks")
  equivalent to the above, but verifies that canonicalForm = singlePart
- **createFromCanonicalForm(String canonicalForm)**
  use when decoding the canonical form

Dataverse Name API (#2)

Class: org.apache.asterix.common.metadata.DataverseName

Accessors

- List<String> **getParts()**
  returns a new list of parts.
- void **getParts(Collection<? super String> outParts)**
  avoids list construction
- String **getCanonicalForm()**
- boolean **isMultiPart()**

Note: this class only stores the canonical form, therefore getCanonicalForm() is fast, while getParts() performs canonical form parsing and removes escape characters

Dataverse Name API (#3)

Class: org.apache.asterix.common.metadata.DataverseName

Other methods

- **equals()** and **hashCode()**
  operate on the canonical form
- **compareTo()** (implements Comparable)
  operates on the canonical form
- **toString()**
  returns the display form for error messages and logs.
  display form = part1 . part2 . … . partN (parts are as is, not escaped)

Dataverse Name API (#4)

Built-in dataverse name constants

["Default"]	MetadataBuiltinEntities.DEFAULT_DATAVERSE_NAME
["Metadata"]	MetadataConstants.METADATA_DATAVERSE_NAME
["asterix"]	FunctionConstants.ASTERIX_DV
["algebricks"]	FunctionConstants.ALGEBRICKS_DV

Metadata Datasets

Metadata datasets store canonical form of a dataverse name.

This is because "DataverseName" field is in the closed part and its type cannot be changed due to backwards compatibility.

There's no problem with '.' separator character there due to a bug in DatasetUtil.getDatasetInfo(). Currently '.' character is allowed in a dataverse name, but datasets in those dataverses are not queryable.

We provide a new built-in SQL++ function that takes a canonical form and returns an array of dataverse name parts: decode_dataverse_name(string): array(string)

TBD: whether we need an inverse function for encoding a dataverse name into its canonical form

Functions

1. Function namespaces
   a. FunctionSignature is in Asterix -> namespace is a Dataverse Name
   b. FunctionIdentifier is in Hyracks -> namespace is a canonical form of the Dataverse Name (single string).
2. Conversion methods
   a. FunctionSignature.createFunctionIdentifier(), and
   b. FunctionSignature.getDataverseName()
3. Function Libraries
   a. For now only support single part names
   b. Not yet migrated to avoid introducing more conflicts with pending UDF changes. E.g. UdfApiServlet

Other Subsystems

1. Metadata Lock Manager
   a. Previously was using a single string as a lock key
   (concatenation of entity kind, dataverse name, and entity name)
   b. Now uses MetadataLockKey class which keeps all these components separately

2. AQL grammar only supports single part names.
   Will not be migrated to handle multipart names

Guidelines for accepting dataverse names from users

1. Avoid the canonical form (single string) as much as possible.
   This form is currently only exposed in Metadata datasets because we cannot change metadata schemas.
2. In SQL++ grammar:
   accept multipart names using MultipartIdentifier and MultipartIdentifierWithHints productions in SQLPP.jj
3. In SQL++ built-in function arguments:
   accept both String (single part name) and Array values (single or multipart name).
   See FunctionRewriter.getDataverseName()
4. In JSON REST API requests:
   accept both String (single part name) and Array values (single or multipart name).
5. In HTTP request parameters:
   accept multi-valued HTTP parameters using ServletUtil.getDataverseName(). See also IServletRequest.getParameterValues()

Guidelines for returning dataverse names to users

1. Avoid canonical form as much as possible
   a. Permitted in Metadata datasets.
2. Returning as JSON from REST API (See JSONUtil.putArrayOrScalar())
   a. Single part name -> return as String (for backward compatibility)
   b. Multipart name -> return as Array of Strings
3. Returning as ADM from built-in SQL++ functions
   a. Single part name -> return as String (TBD: discuss this)
   b. Multipart name -> return as Array of Strings
4. Use display form in error/warning/log messages (DataverseName.toString())

General guidelines

1. Do not concatenate a canonical form of a dataverse name with a dataset name or another entity name. Dataset / entity names can have '.' characters too.

2. OK to do this for error / warning / log messages.
   See DatasetUtil.getFullyQualifiedDisplayName()

3. OK to use the canonical form for passing dataverse name inside Hyracks as long as this form is not user-accessible / long-term persisted.
   E.g.: AccessMethodJobGenParams

Code reviews for this change

1. AsterixDB
   https://asterix-gerrit.ics.uci.edu/c/asterixdb/+/4004

2. BAD
   https://asterix-gerrit.ics.uci.edu/c/asterixdb-bad/+/4064