**Epipog**

**Specification**

**Schema Class Family**

**Nov. 11, 2016**

# 1. Interface

The schema object is the interface definition for setting a schema for a collection. Derived objects are implemented from the interface, depending on their requirement in the database pipeline. This interface is defined in the file Schema.java. This interface defines the methods and interfaces for how a schema is specified and/or dynamically extended for a collection in an Epipog application and the operations that may be performed with the schema object.

## Types

The interface defines the following data types:

public enum BSONType {  
 DOUBLE ("double", (byte)1),  
 STRING ("string", (byte)2),  
 OBJECT ("object", (byte)3),  
 ARRAY ("array", (byte)4),  
 BINDATA ("bindata", (byte)5),  
 UNDEFINED ("undefined", (byte)6),  
 OBJECTID ("objectid", (byte)7),  
 BOOLEAN ("boolean", (byte)8),  
 DATE ("date", (byte)9),  
 NULL ("null", (byte)10),  
 REGEX ("regex", (byte)11),  
 JAVASCRIPT ("javascript", (byte)13),  
 INTEGER ("integer", (byte)16),  
 TIMESTAMP ("timestamp", (byte)17),  
 LONG ("long", (byte)18),  
 // extended (non-bson)  
 FLOAT ("float", (byte)51),  
 DECIMAL ("decimal", (byte)52),  
 SHORT ("short", (byte)53),|  
 TIME ("time", (byte)54),  
 URL ("url", (byte)55),  
 CHAR ("char", (byte)56);  
}

This enumerated type are the BSON specification for mapping string representation of data types into integer identifiers. Since the specification does not cover all the data types anticipated by the epipog application, it has been extended (values > 50).

## 1.2 Interface Methods

The interface contains the following non-implemented methods

***Accessors (Getter/Setter)***

public void Collection( String collectionName );

This setter method sets the name of the collection for which the schema is applied to.

public String CollectionName();

This getter method returns the name of the collection that the schema was applied to.

***Schema***

public void SetS( ArrayList<Pair<String,String>>) throws DataException;

This method sets a schema for a collection, where the data type is in represented as a string name of the data type. This method is generally used on table based collections that once set cannot be changed. If an error occurs, a DataException is thrown.

public void SetI( ArrayList<Pair<String,String>>) throws DataException;

This method sets a schema for a collection, where the data type is in represented as a BSON integer value of the data type. This method is generally used on table based collections that once set cannot be changed. If an error occurs, a DataException is thrown.

public void ExtendS( ArrayList<Pair<String,String>>) throws DataException;

This method extends a schema for a collection, where the data type is in represented as a string name of the data type. This method is generally used on non-table collections which can be continuously extended. If an error occurs, a DataException is thrown.

public void ExtendI( ArrayList<Pair<String,String>>) throws DataException;

This method extends a schema for a collection, where the data type is in represented as a BSON integer value of the data type. This method is generally used on non-table collections which can be continuously extended. If an error occurs, a DataException is thrown.

public boolean IsDefined( String key );

This method is used to check if the specified key is defined in the schema for the associated collection.

public boolean IsValid( String key, Integer type );

This method is used to check if the specified type is valid for the corresponding key in the schema for the associated collection.

public integer ColumnPos( String key );

This method is used to return the column position (ordinal ordering starting at one) of a key for a table based schema.

# 

# 2 SchemaTable Implementation

This implementation implements the “Schema” interface. They are implemented in the file SchemaTable.java. This implementation implements the methods and interfaces for a table based schema, which cannot be modified once set. This is typically used for collections that used a tabular data store (i.e., RDBMS).

## 2.1 Fields

The implementation defines the following fields, which are only accessible (private/protected) by this class or a derived class:

## private String collectionName; // collection name

## This field is the name of the collection that the schema is set to. protected ArrayList<Pair<String,Integer>> keys; // keys in schema

This field are the key names and associated data types, in BSON types, for the schema. The order of the keys also represents the ordinal ordering of the columns in the table-based schema.

## 2.2 Methods

The implementation contains the following methods:

***Constructor***

public SchemaTable( String collectionName);

This constructor instantiates the schema object and sets the name of the associated collection.

***Accessors (Getter/Setter)***

public void Collection( String collectionName );

This setter method sets the name of the collection for which the schema is applied to.

public String CollectionName();

This getter method returns the name of the collection that the schema was applied to.

***Schema***

public void SetS( ArrayList<Pair<String,String>>) throws DataException;

This method sets a schema for a collection, where the data type is in represented as a string name of the data type. This method is used on table based collections that once set cannot be changed. If an error occurs (e.g., duplicate key, invalid type), a DataException is thrown.

public void SetI( ArrayList<Pair<String,String>>) throws DataException;

This method sets a schema for a collection, where the data type is in represented as a BSON integer value of the data type. This method is used on table based collections that once set cannot be changed. If an error occurs (e.g., duplicate key, invalid type), a DataException is thrown.

public void ExtendS( ArrayList<Pair<String,String>>) throws DataException;

This method is not supported. If it is called, a DataException is thrown.

public void ExtendI( ArrayList<Pair<String,String>>) throws DataException;

This method is not supported. If it is called, a DataException is thrown.

public boolean IsDefined( String key );

This method is used to check if the specified key is defined in the schema for the associated collection.

public boolean IsValid( String key, Integer type );

This method is used to check if the specified type is valid for the corresponding key in the schema for the associated collection.

public integer ColumnPos( String key );

This method is used to return the column position (ordinal ordering starting at one) of the key for the table based schema.

# 3 SchemaDynamic Implementation

This implementation extends the “SchemaTable” implementation. They are implemented in the file SchemaDynamic.java. This implementation implements the methods and interfaces for a non-table based schema, which can be continuously extended. This is typically used for collections that used a dynamic schema store (e.g., CSV).

## 3.2 Methods

The implementation contains the following methods:

***Schema***

public void ExtendS( ArrayList<Pair<String,String>>) throws DataException;

This method extends a schema for a collection, where the data type is in represented as a string name of the data type. This method is used on non-table collections which can be continuously extended. If an error occurs (e.g., duplicate key, invalid type), a DataException is thrown.

public void ExtendI( ArrayList<Pair<String,String>>) throws DataException;

This method extends a schema for a collection, where the data type is in represented as a BSON integer value of the data type. This method is used on non-table collections which can be continuously extended. If an error occurs (e.g., duplicate key, invalid type), a DataException is thrown.