DBEntities

Adopted By: no NeXTSTEP classes

Incorporates: DBTypes

Declared In: dbkit/entities.h

Protocol Description

The DBEntities protocol lets an object represent a database *entity*. An entity comprises a list of data categories, or *properties*. As data is read from a database for a particular entity, an ªinstanceº of the entity (a *record*) is created and filled with data, one datum per property.

It's tempting to speak of an entity as a database table. They're similar. You can think of a table as the corporealization of an entity. Put another way, an entity describes how a table organizes its data into columns (properties). However, you should keep in mind that an entity doesn't contain data (nor do the properties within the entity). Furthermore, neither entities nor properties are ªplaceholdersº for data. Entities and properties neither store nor make room for data, they simply provide a description of the type and location of data so some other object (a record) can be created to adequately store this data.

Typically, an application doesn't create entity objects directly, but, instead, reads them from a database model file. This is performed by creating a DBDatabase object and connecting it to the file (through methods described in the DBDatabase class specification). You can retrieve, in a List, the entity objects that the DBDatabase read from the model file by sending the DBDatabase a **getEntities:** message. Alternatively, you can retrieve a single entity object by name through **entityNamed:**. Both of these methods return private DBEntities-conforming objects that are created and owned by the Database Kit.

Entity object are used as arguments in a handful of important methods. Most notable of these, you typically use an entity as the source in an invocation of DBRecordList's **setProperties:ofSource:**. In addition, an entity is required by the DBQualifier and DBExpression initialization methods.

The DBEntities protocol incorporates the DBTypes protocol. It does this for one reason: the type of Objective C data described by a property that represents a relationship is a DBEntities object. Thus, if the **isEntity** message returns YES when sent to the value returned by sending **propertyType** to a property, then that property is a relationship. This is demonstrated in the following example:

```
/* Get the properties from an entity.  Check for relationships. */
int counter;
List *propList = [[List alloc] init];
id prop;

[anEntity getProperties:propList];
for (counter = 0; counter < [aList count]; counter++)
{
   prop=[aList objectAt:counter];
   if ([[prop propertyType] isEntity])
      printf("Property named %s is a relationship.\n", [prop name]);
}
```

Warning: You should never send the DBTypes messages **objcClassName** or **databaseType** to the private entity objects that are returned by the aforementioned DBDatabase methods. The private entity class implements these DBTypes methods to raise DB_UNIMPLEMENTED_ERROR exceptions.

It isn't anticipated that you should need to create your own class that adopts the DBEntities protocol. The entity objects returned by **getEntities:** and **entityNamed:** should be adequate for most applications.

Method Types

Querying the object	- name - database - getProperties: - propertyNamed:
Comparing the object	- matchesEntity:

Instance Methods

database

- (DBDatabase *)**database**

Returns the DBDatabase object that created the entity.

getProperties:

- **getProperties:**(List *)*aList*

Returns, in *aList*, a list of the entity's properties. Each object in the list conforms to the DBProperties protocol.

matchesEntity:

- (BOOL)**matchesEntity:**(id <DBEntities>)*anEntity*

Returns YES or NO if the receiving entity and *anEntity* were created from the same model file entity.

name

- (const char *)**name**

Returns the entity's name. This is the same name as given to the entity in the model file from which it was read.

propertyNamed:

- **propertyNamed:**(const char *)*aName*

Returns the property named *aName*. If the entity has no such property, **nil** is returned.