addRetrieveOrder:(DBRetrieveOrder)anOrder for:(id <DBProperties>)aProperty

Associates a retrieval order with the property aProperty. The permissible values of anOrder are:

binderDelegate

Returns the delegate used by the DBRecordStream's DBBinder objects.

setBinderDelegate:

cancelFetch

Terminates the current fetch operation this is generally only of use if the DBRecordStream is fetch
Returns self.

fetchUsingQualifier:

clear

Resets the DBRecordStream. The DBRecordStream's record data, list of properties, and list of key
emptied. Its database instance variable is set to nil, but its delegate remains unchanged. Its status
Returns self.

currentRetrieveStatus, free

(DBRecordRetrieveStatus)currentRetrieveStatus

Returns the DBRecordStream's status, which can be:

delegate

Returns the DBRecordStream's delegate or nil if no delegate has be set.

setDelegate:, recordStream:willFailForReason: (delegate method)

deleteRecord

Deletes the current record in the DBRecordStream and causes the DBRecordStream to access the
if any.

Returns nil if the deletion can't be accomplished otherwise, returns self. If the deletion fails, the D
attempt to notify its delegate of the reason, and the cursor remains unchanged (pointing to the reco
been deleted but wasn't).

recordStream:willFailForReason: (delegate method) cancelFetch, setProperties:ofSource:

free

Releases the storage for the DBRecordStream.

(List *)getKeyProperties:(List *)keyList

Fills keyList with objects that represent the key properties of the DBRecordStream. Each of these
DBProperties protocol. Returns the newly filled List object.

setKeyProperties:

(List *)getProperties:(List *)propertyList

Places the DBRecordStream's property list in propertyList and returns propertyList.

setProperties:ofSource:

getRecordKeyValue:(DBValue *)aValue

Places the value of the current record's key property (or properties) in aValue.

This method is especially useful when data must be exchanged between DBRecordStreams. For ex
DBRecordStream supplies employee information and another supplies department information to t
application. A user can change an employee's department by selecting from a list of department n
department name is selected, you can use getRecordKeyValue: to determine the corresponding rec
you can set the department identification in the employee's record.

Returns nil if the DBRecordStream has status DB_NotReady otherwise, returns aValue.

getValue:(DBValue *)aValue forProperty:aProperty

Places the value for aProperty into aValue. This method is the only means of retrieving record dat
DBRecordStream.

When aProperty is a relationship, the method sets aValue so that it includes the key value of the re
property and the entity that is the relationship's target. (In that case, sending aValue the DBValues
would get the response YES.) The fact that the value object identifies the target entity is exploited
setProperties:ofSource:.

If the status of the DBRecordStream is DB_NotReady, this method return nil. Otherwise, it return

setValueFor:from:, propertyNamed: (DBDatabase), isEntity (DBValues protocol), setProperties

init

database NO otherwise.

isNewRecord

(BOOL)isNewRecord

Returns YES if the current record is new that is, it the result of the DBRecordStream receiving a n

newRecord, isModified

(BOOL)isReadOnly

Returns YES if the records in the DBRecordStream can only be read, not modified. If a DBRecor
properties haven't been set, isReadOnly will return YES.

setKeyProperties:, getKeyProperties:

newRecord

Creates a new, empty record. Before this operation can take place, the DBRecordStream attempts
of the current record to the database. If these changes can't be saved, newRecord returns nil, no ne
and the cursor is not advanced. Otherwise, newRecord returns self, and the cursor is advanced to
current record.

saveModifications

(unsigned int)saveModifications

Saves the new or modified record to the database. If the database supports transactions and there's
progress, this save operation is nested within a new transaction.

If there is no transaction in progress, a new transaction is created for this operation. If the modific
the database, this transaction is committed. An error during this commit process raises a DB_TRA
exception.

Returns these values:

areTransactionsEnabled (DBDatabase), beginTransaction (DBDatabase)

setBinderDelegate:newDelegate

delegate, recordStream:willFailForReason: (delegate method)

(List *)setKeyProperties:(List *)propertyList

Sets the DBRecordStream's list of key properties to propertyList. Each of the objects in propertyL
DBProperties protocol. Typically, key properties are identified in the database model using DBM
invoke this method.

Returns nil if any property in propertyList is not a property of the DBRecordStream's source other
property list.

getKeyProperties:

setNext

Advances the DBRecordStream's internal cursor by 1, so that it points to the next record in the gro
available by a fetch operation.

Returns self if successful and nil if not. A nil return can mean that there are no further records to r
DBRecordStream was unable to save modifications to the current record.

saveModifications

(List *)setProperties:(List *)propertyList ofSource:aSource

Sets the properties that will be fetched or stored by a DBRecordStream, or its subclass, a DBRecor
transferred will be those contained in propertyList. The argument aSource specifies the entity that
this is typically a DBEntities object that's the ªrootº of all the properties in the property list. If aS
for the first property in propertyList is used.

The argument aSource can also be a DBValue object that's gotten by asking for the value for a rel
storage object that has already fetched data. The DBValue object encodes the relationship's attribu
that when the receiving DBRecordStream fetches, it qualifies the fetch to select the ªdetailº record
record from whence the DBValue was plucked.

The application should send a setProperties:ofSource: message before doing anything with a DBR
DBRecordList. Once the list of properties has been set, the application can send fetchUsingQualif
the list of properties that has been set. To a DBRecordList, the application can also send fetchUsin
can make multiple inserts or multiple deletes.   (After once calling setProperties:ofSource:, you sh
until you really need to establish a new property list, since each use discards any prior data withou

Returns nil if the properties in propertyList don't share the same entity or if some other error occu
self.

getProperties:, getValue:forProperty:, isEntity (DBValues protocol)

setValue:(DBValue *)aValue forProperty:aProperty

Sets the value for aProperty in the current record to that contained in aValue. Returns a nonzero va
otherwise, returns nil.

returning YES to this message acknowledges the failure and permits the operation to be aborted, t
local transaction of which it is part.

saveModifications, setDelegate:, delegate

(BOOL)recordStreamPrepareCurrentRecordForModification:aRecordStream

Notifies the delegate of a proposed modification to the current record, verifies that the record is un
modification to proceed only if the return is YES.

If implemented, this delegate method provides an alternative to the standard check that a DBRecor
before deleting or modifying a record. (The DBRecordStream or its subclass normally verifies that
and that it is unique. It invokes a ªconfirming selectº on the DBDatabase using the key value, and
properties to see that none has changed. The select is usually a locking select.) This delegate met
mechanism, making the delegate responsible for verification and locking. If the method returns Y
considered to be verified, and modification proceeds. If the method returns NO, the record is not m
cause the entire sequence containing saveModifications: to fail, depending on the transaction mode

This method should not call any of the methods implemented by DBRecordStream or DBRecordL
forProperty: