fetchUsingQualifier:empty:

recordLimit

swapRecordAt:withRecordAt:

Saving data

appendRecord

Adds an empty record at the end of the record list by invoking DBRecordList's insertRecordAt: m
value returned by insertRecordAt:.

insertRecordAt:, newRecord, deleteRecord, deleteRecordAt:

clear

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

empty (DBRecordStream)

(DBRecordListRetrieveMode)currentRetrieveMode

Returns the DBRecordList's retrieve mode, which can be DB_SynchronousStrategy, DB_Backgro
DB_BackgroundNoBlockingStrategy. See the class description above for more information.

setRetrieveMode:

deleteRecord

Deletes the current record. Returns nil if there's no current record otherwise, returns self.

deleteRecordAt:

deleteRecordAt:(unsigned)index

Deletes the record at position index. Returns nil if there's no record at index otherwise, returns sel

the department number from the employee record (see getRecordKeyValue:at:) and then using it as
fetchRecordForRecordKey: in a message to the DBRecordList containing department information.

Returns nil if no record has the supplied key value or if an error occurs otherwise, returns self.

fetchUsingQualifier:, fetchUsingQualifier:empty:

fetchUsingQualifier:(DBQualifier *)aQualifier

Invoking this method is equivalent to invoking

fetchUsingQualifier:(DBQualifier *)aQualifier empty:emptyFirst

Loads the DBRecordList with records from the database. Before invoking this method, use setPro
specify the source and properties of the data to be retrieved. The scope of the retrieved records is c
aQualifier. For example, assuming the data source is an SQL database, aQualifier could be an obje
expression ªwhere name = `Holbein'º. If aQualifier is nil, all records are retrieved.

If emptyFirst is YES, before loading new data, the method first empties the DBRecordList and its
Setting emptyFirst to NO leaves records already fetched in the DBRecordList, and append to them
retrieved by the current fetch. In that case, the effect of successive invocations with different quali
DBRecordList the union of the sets returned by the various qualifiers.

Each fetch can be done synchronously or asynchronously, depending on the fetch mode in effect at
begun (see the class description above for details). If you specify an invalid fetch mode, fetchUsin
raises a DB_UNIMPLEMENTED_ERROR exception.

A synchronous fetch is subject to a limit on the total number of records in the DBRecordList, set b
number of qualifying records would exceed that limit, the DBRecordList receives that number, and
recordStream:willFailForReason: message with the argument DB_RecordLimitReached.

Returns nil if the data can't be selected (for example, if the DBDatabase isn't connected to the data
and DBRecordList refer to different entities in the database otherwise, returns self. After fetchUsi
returns, the DBRecordList's current record is set to the first record in the list.

cancelFetch, fetchUsingQualifier:, setProperties:ofSource:

free

Releases the storage for the DBRecordList.

getRecordKeyValue:(DBValue *)aValue

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

Returns nil if the DBRecordList has status DB_NotReady or if there is no current record otherwise

getRecordKeyValue:at:

Returns nil if the DBRecordList has status DB_NotReady or if there is no record at index otherwis

getRecordKeyValue:

getValue:(DBValue *)aValue forProperty:aProperty

Places the value for the property aProperty of the current record into the DBValue object aValue a

setValue:forProperty:at:, setValue:forProperty, getValue:forProperty:at:

getValue:(DBValue *)aValue
forProperty:aProperty
at:(unsigned)index

Places the value for the property aProperty of the record at position index into aValue and returns a object that conforms to the DBProperties protocol. Such an object is returned by DBDatabase's pr The argument index identifies the record within the DBRecordList and has the range from 0 to the count method.

setValue:forProperty:at:, setValue:forProperty, getValue:forProperty:

init

Initializes a newly allocated DBRecordList. The DBRecordList's delegate instance variable is set mode is set to DB_SynchronousStrategy, and its cursor (its current record) is set to DB_NoIndex.

This method is the designated initializer for DBRecordList.

insertRecordAt:(unsigned)index

Adds a new, empty record to the record list at index. The newly inserted record becomes the curre

Returns nil if the DBRecordList has a DB_NotReady status or if an error prevents the insertion of returns self.

appendRecord, deleteRecord, deleteRecordAt:

(BOOL)isModified

Returns YES if any record in the DBrecordList has been modified, added, or deleted NO otherwise

isModifiedAt:, isModifiedForProperty:at:

(BOOL)isModifiedAt:(unsigned int)index

Returns YES if the record at index is new or has been modified NO otherwise.

(BOOL)isNewRecord

Returns YES if the current record is new that is, it the result of the DBRecordList receiving an app
insertRecordAt:, or newRecord message.

isNewRecordAt:, isModified

(BOOL)isNewRecordAt:(unsigned int)index

Returns YES if the record at index is new that is, if it was produced by the DBRecordList's receivi
insertRecordAt:, or newRecord message.

isNewRecord, isModified

moveRecordAt:(unsigned int)sourceIndex to:(unsigned int)destinationIndex

Moves the record at sourceIndex to destinationIndex. Returns nil if there is no record at sourceInd
prevents the insertion of the record are destinationIndex otherwise, returns self.

newRecord

Creates a new, empty record by invoking DBRecordList's insertRecordAt: method and passing the
row as the argument. Before this operation can take place, the DBRecordList attempts to save mo
current record to the database. If these changes can't be saved, newRecord returns nil, and no new
Otherwise, newRecord returns self, and the new record becomes the current record.

saveModifications

(unsigned int)positionForRecordKey:(DBValue *)aValue

Searches the records in the DBRecordList for the first record whose key value matches aValue. R
no such record is found otherwise, returns the index of the matching record.

(unsigned int)recordLimit

Returns the maximum number of records that a fetch can deliver to a DBRecordList (as set by setF
limit has been set, returns DB_NoIndex.

(unsigned int)saveModifications

areTransactionsEnabled (DBDatabase), beginTransaction (DBDatabase)

setRecordLimit:(unsigned int)count

Makes count the maximum number of records that can be retrieved during a fetch. If a fetch is atte
that would fetch more than this number of records, the method returns the maximum number perm
recordStream:willFailForReason: message to the delegate with the argument DB_RecordLimitRea

setRetrieveMode:(DBRecordListRetrieveMode)aMode

Sets the DBRecordList's retrieve mode, which can be DB_SynchronousStrategy, DB_Background
DB_BackgroundNoBlockingStrategy. See the class description above for more information.

currentRetrieveMode:

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.

getValue:forProperty:, setValue:forProperty:at:

setValue:(DBValue *)aValue
forProperty:aProperty
at:(unsigned int)index

Sets the value for aProperty in the record at index to that contained in aValue. Returns a nonzero v
otherwise, returns nil.

getValue:forProperty:, setValue:forProperty

swapRecordAt:(unsigned int)anIndex withRecordAt:(unsigned int)anotherIndex

Transposes the locations of two records. Both arguments must be valid positions in the DBRecord
records. Returns self, but if an argument is invalid, returns nil.