FlatFile Adaptor

The FlatFileAdaptor provides EOF with access to ASCII text files. Its purpose is not to take the place of a real SQL database, but to to elucidate the issues you might encounter when implementing your own custom adaptor. These include:

- Connecting to your database via a connection dictionary and a login panel.
- Mapping external data types like DATETIME and VARCHAR2 to object values such as NSCalendarDate and NSString.
- Supporting your database's transactional features (e.g. nesting and rollback).
- Generating a schema from a given EOModel and configuring an EOModel from an existing database.
- Generating and executing SQL statements for operations like insert, update, delete and select.

While this adaptor addresses these problems, it handles some of them differently than a normal SQL adaptor would. FlatFileAdaptor defines a database as a collection of files within a directory. Rather than connecting to a server, it reads these files and parses them according to the row and column separators specified in its connection dictionary. It only supports a few simple external data types: Number, String, Data and Date, whose default internal mappings are NSDecimalNumber, NSString, NSData, and NSCalendarDate, respectively. Within the scope of a transaction, all files are cached in memory as NSDatas. Modified files are saved when the transaction is committed ± if it's aborted, their cached editions are released and reread the next time they are needed. Nested transactions are not supported. Generating a Flat File schema consists of writing simple column headers to a set of files. These headers are later used to map EOAttributes to their corresponding columns.

Because there is no SQL server behind the scenes, FlatFileChannel has to perform basic database operations itself ± it evaluates EOQualifiers in memory via evaluateWithObject: and sorts its results using EOF's extensions to NSArray. A normal adaptor would simply pass SQL strings generated by its EOSQLExpression subclass to a server via evaluateExpression: and process the results. Its EOAdaptorChannel subclass's deleteRowsDescribedByQualifier: might look like:

```
- (void)deleteRowsDescribedByQualifier:(EOQualifier *)qualifier entity:(EOEntity *)entity
{
    EOSQLExpression *sqlExpression;
    NSDictionary *pkDict = nil;

    if (_delegateRespondsTo.willDeleteRows) {
        EODelegateResponse response;
        response = [_delegate adaptorChannel:self willDeleteRowsDescribedByQualifier:qualifier
entity:entity];
        if (response != EODelegateApproves)
            return;
```

```
    }

    sqlExpression = [[[_context adaptor] expressionClass] deleteStatementWithQualifier:qualifier
entity:entity];
    [self evaluateExpression:sqlExpression];

    if (_cda.rpc == 0) {    // check database error return code
        [NSException raise:EOGeneralAdaptorException format:@"%@ -- %@ 0x%x: Attempted to
delete a row that is not in the database", NSStringFromSelector(_cmd), NSStringFromClass([self class]),
self];
    }

    if (_delegateRespondsTo.didDeleteRows)
        [_delegate adaptorChannel:self didDeleteRowsDescribedByQualifier:qualifier entity:entity];
}
```

Despite such implementation differences, FlatFileAdaptor is structured the same way a SQL adaptor would be.

Classes

FlatFileAdaptor	Manages connection information and type mappings.
FlatFileContext	Maintains files' state within a transaction scope.
FlatFileChannel	Manipulates the NSData representations of the files.
FlatFileColumn	Facilitates conversion of ASCII strings to object values.
FlatFileSQLExpression	Provides support for creating and removing files via FlatFileChannel's evaluateExpression: which parses simple (non±SQL) statements. Don't be fooled by the class name.

Other Components

FlatFileDescription	Adds a category on FlatFileChannel to configure an EOModel to reflect the structure of an existing database.
FileScanning	Contains a simple function (FFNextTokenIn) for parsing ASCII text (read: char *) into a stream of tokens.
FlatFileLoginPanel	Provides a simple UI for connecting to a "database," which is really just a set of files.

Open Issues

nested transactions
string values with the row or column separator as a substring
flattened attributes
flattened relationships (many-to-many support)
file timestamps
locking