EOF

Title: EOF 1.1 Fetch Performance
Entry Number: 1917
Creation Date: September 29 1995
Last Updated: <<Date September 29 1995>>

Keywords: EOF, performance, fetch, memory usage

Question

Q: I heard that EOF 1.1 is much faster than EOF 1.0 and DBKit. Do you have any benchmarks or test results to back this up?

Answer

A: EOF 1.0 performed comparably to DBKit in terms of fetch speed. A major goal of EOF 1.1 was to both increase performance and reduce memory usage.

Here are the results of our test results comparing fetch performance in DBKit vs. EOF.

Fetching ~10,000 10 column wide rows from a Sybase server on a 60MHz Pentium

	DBKit	**EOF 1.0**	**EOF 1.1**	**v1.1 v.s. 1.0**	**v1.1 vs DBKit**
Adaptor	25.15*	19.20	8.89 sec	**2.2 X Faster**	2.8 X
DataSource**	32.93	51.32	19.51 sec	**2.6 X**	1.7 X

Fetching ~5,000 50 column wide rows from a Sybase server on a 60MHz Pentium

	DBKit	**EOF 1.0**	**EOF 1.1**	**v1.1 v.s. 1.0**	**v1.1 vs DBKit**
Adaptor	180.30	47.87	9.63 sec	**5.0 X**	18.7 X
DataSource	225.62	99.28	17.92 sec	**5.5 X**	12.6 X

** For DBKit "Adaptor" times represent fetches from the DBRecordList level, and "DataSource" represents fetches from the DBFetchGroup level (Of course the DBFetchGroup does not perform the uniquing, snapshotting, or creation of custom classes performed by EOF).*

*** Note that fetches from the DataSource level in EOF translate closely to times to fetch into a TableView connected up in InterfaceBuilder.*

Memory usage was also dramatically decreased between EOF 1.0 and 1.1.

EOF 1.0 vs. EOF 1.1:
- per object fetch memory usage reduced between 50-65%
- per row fetch memory usage reduced approximately 50%

Notes regarding these results:

- EOF 1.1 provides equal or better performance and memory usage than DBKit, and provides more power and flexibility with features such as object uniquing and snapshotting.
- The tests performed the same fetch against the same Sybase database on the same hardware. Steps were taken to minimize other performance impacts on the network and systems in use.

These tests perform simple fetches against a single table in the database. Your results may vary based on the configuration of your database server and your EOModel. If fetch performance is a significant issue in your application, keep the following factors in mind:

- The number of relationships in the Entity marked as class properties has a significant effect on fetch times. If you do not require object faulting for the relationship, you can unmark the class property setting and still use a master-peer controller setup to see master-detail type information in your interface.
- Flattened attributes produce SQL joins that may result in lengthy processing at the server.
- NSDate attributes are more expensive to create than strings (although this penalty was greatly reduced in EOF 1.1)
- If your code triggers many faults you may produce a large number of single object fetches to the server. You should be able to see this happening in the SQL output.
- Setting indexes in the database server on the primary key columns of tables involved in destination of to-one relationships should improve fetch times for faults and flattened attributes.

Valid for: EOF 1.1, NEXTSTEP 3.2 Developer, NEXTSTEP 3.3 Developer