УДК 004.4

СОВЕРШЕНСТВОВАНИЕ АЛГОРИТМА ОБХОДА ДЕРЕВА БИЗНЕС-ОБЪЕКТОВ

Н. В. Ерёмин; А. Ф. Тарасов, д-р техн. наук,
Донбасская государственная машиностроительная академия
nikyer@gmail.com

Существует ряд методов повышения производительности компонентов объектно-реляционного отображения (ORM), в частности, оптимизация запросов, кэширование объектов, метод отложенной загрузки (Lazy load).

В данной работе проанализирована типичная архитектура ORM в сервере приложений корпоративной информационной системы. Используемая структура данных представляет собой дерево бизнес-объектов различных типов, структура которого определяется отношениями классов. В типичной реализации алгоритма обхода дерева бизнес-объектов при выполнении исходного запроса из БД может быть выбрано множество объектов, но при дальнейшем обходе дерева для каждого вновь загружаемого объекта выполняется отдельный запрос.

Экспериментально установлено, что общее время выполнения запросов в сервере приложений существенно зависит от количества SQL-запросов к СУБД. Вместе с тем, время выполнения одного SQL-запроса слабо зависит от количества возвращаемых записей. Эксперимент проводился с использованием реляционной СУБД MSSQL Server. Такие же выводы сделаны в работе других авторов с использованием реляционной СУБД Interbase. Поэтому реального прироста производительности можно добиться за счёт уменьшения количества запросов к СУБД.

Уменьшить количество запросов к СУБД в данной модели можно, формируя запросы таким образом, чтобы они возвращали все необходимые объекты одного типа (из заданной таблицы). Объединить условия таким образом, чтобы выбрать полностью все объекты заданного типа в дереве невозможно, т.к. информация о том, какие объекты понадобятся на нижних уровнях, отсутствует на верхних уровнях дерева. При этом можно выбрать одним запросом все необходимые объекты данного типа на одном уровне иерархии дерева.

Рассмотрим предлагаемый алгоритм загрузки объектов при обходе дерева. Для каждого уровня выполняется следующая последовательность операций при наличии кэширования:

1. Выполнить запрос на выборку объектов текущего уровня по имеющемуся условию.
2. Если множество полученных объектов непустое, исключить из выборки объекты, уже содержащиеся в кэше.
3. Сформировать множество типов, которые ассоциированы с данным типом. Если какой-либо из типов является абстрактным, заменить его множеством наследников.
4. Если множество полученных типов непустое, для каждого найденного типа сформировать множество значений идентификаторов объектов, анализируя объекты из множества, полученного на шаге 2.
5. Для полученных непустых множеств идентификаторов сформировать условия выбора объектов для следующего уровня дерева (для всех типов).

Переходим на следующий уровень.

Например, обход дерева, содержащего объекты трех типов *A, B, C* (*B* ассоциирован с *A, C* ассоциирован с *B*) осуществляется следующим образом: выполняется первый запрос на

выборку всех объектов класса *A,* затем выполняется N_A запросов на выборку объектов класса *B*, на которые имеются ссылки из объектов класса *A*. После этого выполняется N_B запросов на выборку объектов класса *C*, ассоциированных с объектами класса *B.*

Разработанный алгоритм обхода дерева объектов позволяет сократить количество запросов к БД за счёт выборки всех объектов одного типа на каждом уровне дерева одним запросом, что может существенно снизить временные затраты на обход всего дерева объектов и повысить производительность сервера приложений корпоративной системы.