Повышение эффективности запросов к базе данных в системе экологического мониторинга

Е.Л. Кулида, И.П. Крюкова, В.Г. Лебедев (ИПУ РАН)

Описываются методы повышения эффективности, расширения возможностей программирования, надежности и безопасности выполнения запросов к БД в системе экологического мониторинга за счет использования хранимых процедур (ХП) и совершенствования структуры БД.

Введение

На объекте по уничтожению химического оружия в п. Горный Саратовской области с 2002 г. действует система производственного экологического мониторинга [1]. Целью экологического мониторинга является постоянное получение оперативной информации, необходимой для контроля безопасности функционирования объекта по уничтожению химического оружия, для оперативного принятия решений по предотвращению аварий при нарушениях ТП, для оповещения рабочего персонала объекта и населения в случае аварийных выбросов. Комплекс технических средств системы мониторинга включает стационарные и мобильные технические средства, химико-аналитическую лабораторию, метеостанцию.

Информация, получаемая в процессе мониторинга, поступает в централизованную БД, расположенную на сервере информационно-аналитического центра [2], где она обрабатывается, анализируется и используется для оценки и прогноза воздействия объекта по уничтожению химического оружия на окружающую среду (рис. 1). БД функционирует под управлением СУБД Microsoft SQL Server 2000 [3].

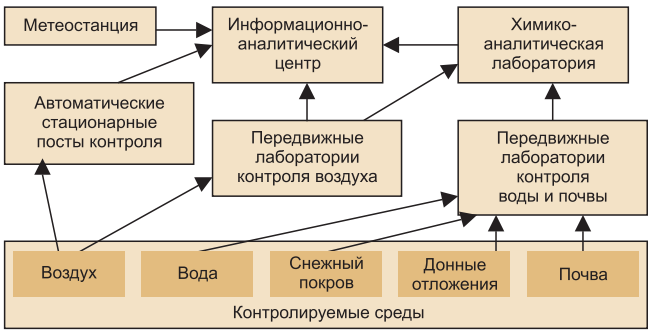

Рис. 1. Обобщенная структура системы экологического мониторинга

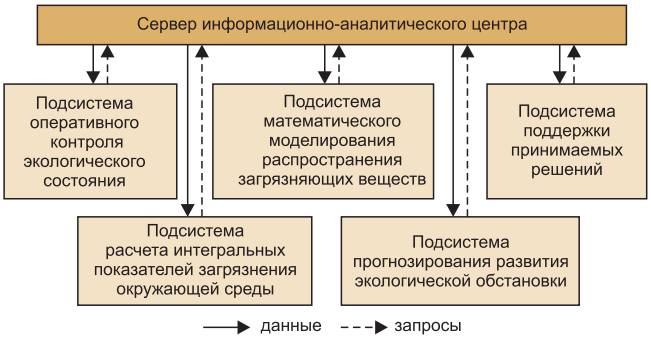

Рис. 2. Схема информационного обмена между сервером и клиентскими приложениями

Система функционирует на основе технологии клиент-сервер. Рабочие станции связаны с сервером сетью. Запросы на выполнение операций с данными от клиентских приложений, выполняемых на рабочих станциях, порождают на сервере поиск и извлечение данных, которые транспортируются по сети от сервера к клиентам. Схема информационного обмена между сервером и клиентскими приложениями представлена на рис. 2.

В основе успешного решения задач системы экологического мониторинга лежит эффективная работа с информацией, накапливаемой в БД. Рассмотрим методы повышения эффективности выполнения запросов к БД на основе использования ХП и оптимизации структуры БД.

Использование хранимых процедур

Один из путей оптимизации производительности БД – создание ХП для реализации сложных, часто используемых запросов. В этом случае производительность повышается за счет локального (по отношению к БД) хранения, прекомпиляции исходного текста и кеширования. Перед созданием процедуры ее команды проходят синтаксическую проверку. При первом запуске процедуры создается план выполнения, ХП компилируется и сохраняется в SQL Server.

Использование ХП обеспечивает модульность, повторное использование кода и облегчает сопровождение БД. При изменении правил функционирования достаточно внести изменения в ХП, после чего все клиентские приложения, использующие ее, будут работать по новым правилам без непосредственной модификации.

ХП могут принимать входные параметры, возвращать значения выходных параметров, поддерживать обратную связь с клиентским приложением посредством вывода кодов состояния и текстовых сообщений, вызывать другие процедуры, возвращать другой процедуре код состояния, в зависимости от которого последняя выполняет те или другие действия. Можно создавать сложные программы на таких языках, как

Гераклит утверждал, что в одну реку нельзя вступить дважды. Современные экологи утверждают, что есть реки, в которые нельзя вступить и раза.
Е. Кащеев

C++ и затем вызывать их из SQL Server как расширенные ХП.

ХП можно использовать в нескольких БД.

Одной из задач мониторинга является постоянное обеспечение клиентских приложений оперативной информацией о содержании вредных веществ, продуктов их деструкции и общепромышленных загрязнителей в контролируемых средах, а также о динамике изменения уровня их концентраций в местах контроля. Для повышения эффективности решения этой задачи реализованы ХП, которым можно передать параметры: контролируемая среда, место контроля, диапазон дат и времени. В результате вызова таких процедур клиентское приложение получает выборку из таблиц БД для заданных параметров.

Использование ХП повышает эффективность функционирования системы, поскольку позволяет избежать пересылки через сеть сотен команд; при вызове не требуется проверка синтаксиса, создание плана исполнения и компиляция текста процедуры.

Другое важное следствие использования ХП – повышение надежности функционирования системы. Клиентским приложениям можно предоставить право на использование ХП без непосредственного доступа к объектам БД, что позволяет изолировать структуру БД от клиентских приложений и тем самым обеспечить целостность данных и повысить надежность БД.

Использование триггеров

Триггеры представляют собой особый вид ХП, привязанных к таблицам и представлениям, вызываемых при модификации данных в таблицах в результате выполнения операторов вставки, удаления или обновления. Они позволяют реализовать сложные процедурные методы поддержания целостности данных и логику функционирования БД. Триггеры особенно полезны, поскольку позволяют описывать более сложную логику по сравнению с логикой, доступной методам декларативной целостности.

С помощью триггеров в системе экологического мониторинга реализованы:

- перераспределение оперативной информации по таблицам БД для эффективного дальнейшего ее использования, поскольку одним из факторов, существенно влияющих на эффективность записи/извлечения информации из БД, является структура таблиц;
- контроль и оперативная реакция на выход параметров мониторинга за границу диапазона значений (нормальное, предаварийное, аварийное);
- формирование сообщений операторам при возникновении событий в системе и т.д.

В задачу мониторинга входит визуализация последних поступивших значений концентрации вредных веществ, продуктов их деструкции и общепромышленных загрязнителей в контролируемых средах в различных местах контроля. Согласно регламенту мониторинга частота определения различных параметров в разных средах контроля может сильно различаться. Так, например, некоторые параметры воздушной среды контролируются каждые несколько минут автоматическими приборами, в то время как параметры снежного покрова или донных отложений контролируются два раза в год и результаты анализов вводятся в систему вручную. Для эффективной визуализации удобно использовать таблицу, содержащую последние поступившие значения концентраций и статус этих значений независимо от частоты измерений и источника этих значений. Обновление значений этой таблицы удобно реализовать при помощи триггеров для тех таблиц, в которые поступают оперативные данные.

На рис. 3 представлен фрагмент написанного на языке Transact-SQL триггера на вставку данных в таблицу, в которую поступают оперативные данные о состоянии окружающей среды от автоматических стационарных пунктов контроля, передвижных лабораторий и метеостанции.

С помощью этого триггера:

- информация перераспределяется по таблицам: метеопараметров и параметров по воздушной среде;
- обновляются значения в таблице, содержащей последние поступившие значения всех контролируемых параметров и их статус;
- выдаются сообщения оператору в случае превышения предельно допустимых концентраций контролируемых параметров.

Использование представлений

Еще один путь повышения эффективности работы с БД – создание представлений на сервере БД. Представления позволяют отбирать конкретные данные, упрощать работу с ними, настраивать их извлечение, выполнять импорт и экспорт, а также объединять распределенные данные для клиентских приложений системы экологического мониторинга.

Представление – это виртуальная таблица, содержимое которой определяется запросом. Как и таблица, представление состоит из столбцов и строк, содержащих данные. При ссылке на представление формирование строк и столбцов представления осуществляется динамически.

Представления позволяют выполнять функции:

- ограничения доступной пользователю области таблицы определенными строками и/или столбцами;
- объединения столбцов из нескольких таблиц, представляя их в виде единой таблицы;
- замены детальных сведений агрегированными.

Представление упрощает доступ к данным, поскольку не приходится каждый раз формировать за-

```
ALTER    TRIGGER ModeSignal ON dbo.RTData FOR INSERT AS
  DECLARE @Value FLOAT, @DT datetime, @Pdk FLOAT
  DECLARE @Color int, @Place int, @Subst int, @ID int , @Env int
  DECLARE @ColName nvarchar(30) , @SQLString nvarchar(200)
-Определение значений переменных
  SELECT  @Value = I.RTValue,@DT=I.Data FROM inserted I
  SELECT  @Pdk= dbo.Parameters.Pdk,@Place=dbo.Place.Place_id,
                  @Subst=dbo.Signal.Substance_id,@Env=dbo.Signal.Environ_id
     FROM  inserted I
INNER JOIN dbo.Signal ON I.Signal_id = dbo.Signal.Signal_id
INNER  JOIN dbo.Place ON dbo.Signal.Place_id = dbo.Place.Place_id
INNER JOIN dbo.EF ON dbo.Place.EF_id = dbo.EF.EF_id LEFT OUTER JOIN dbo.Parameters
ON dbo.Signal.Substance_id = dbo.Parameters.Substance_id
AND dbo.Signal.Environ_id = dbo.Parameters.Environ_id AND
                                                  dbo.EF.Zone_id = dbo.Parameters.Zones_id
  SELECT @ColName=Abr FROM Substances WHERE Substance_id=@Subst
  IF @Env=7 -метеопараметры
    BEGIN
---Запись в таблицу MeteoTabl
      IF  NOT EXISTS ( SELECT * FROM MeteoTabl WHERE DateTime=@DT AND Place_id=@Place)
      INSERT MeteoTabl (DateTime,Place_id) VALUES (@DT,@Place)
      SELECT @ID=Meteo_id FROM MeteoTabl WHERE DateTime=@DT AND Place_id=@Place
      SET  @SQLString= N'UPDATE MeteoTabl SET '+ @ColName+'= @v1 WHERE Meteo_id = @v2'
      EXEC sp_executesql @SQLString, N'@v1 float, @v2 int', @v1=@Value,@v2=@ID
---Запись в таблицу MeteoArch
      IF  NOT EXISTS ( SELECT * FROM MeteoArch WHERE DateTime=@DT AND Place_id=@Place)
      INSERT MeteoArch (DateTime,Place_id) VALUES (@DT,@Place)
      SELECT @ID=Meteo_id FROM MeteoArch WHERE DateTime=@DT AND Place_id=@Place
      SET  @SQLString= N'UPDATE MeteoArch SET '+ @ColName+'= @v1 WHERE Meteo_id = @v2'
      EXEC sp_executesql @SQLString, N'@v1 float, @v2 int', @v1=@Value,@v2=@ID
---Запись в таблицу Signal
      UPDATE S SET S.[RTValue]=@Value,  S.[Date]=@DT FROM Signal S
      INNER JOIN inserted I ON S.Signal_id=I.Signal_id
    END
IF @Env=1 -для воздуха
  BEGIN
--Запись в таблицу Air
    IF  NOT EXISTS ( SELECT * FROM Air WHERE DateTime=@DT AND Place_id=@Place)
        INSERT Air (DateTime,Place_id) VALUES (@DT,@Place)
    SELECT @ID=Air_id FROM Air WHERE DateTime=@DT AND Place_id=@Place
    SET  @SQLString= N'UPDATE Air SET '+ @ColName+'= @v1 WHERE Air_id = @v2'
    EXEC sp_executesql @SQLString, N'@v1 float, @v2 int', @v1=@Value,@v2=@ID
--Запись в таблицу ArchAir
    IF  NOT EXISTS ( SELECT * FROM ArchAir WHERE DateTime=@DT AND Place_id=@Place)
        INSERT ArchAir (DateTime,Place_id) VALUES (@DT,@Place)
    SELECT @ID=Air_id FROM ArchAir WHERE DateTime=@DT AND Place_id=@Place
    SET  @SQLString= N'UPDATE ArchAir SET '+ @ColName+'= @v1 WHERE Air_id = @v2'
EXEC sp_executesql @SQLString, N'@v1 float, @v2 int', @v1=@Value,@v2=@ID
--Запись в таблицу Signal
SET @Color=1
IF @Pdk IS NOT NULL
  BEGIN
          F( @Value<@Pdk) SET @Color=0
          ELSE SET @Color=2
  END
SET @Value=@Value/@Pdk
UPDATE S SET S.[RTValue]=@Value, S.[Pdk]=@Pdk, S.[Color]=@Color, S.[Date]=@DT FROM Signal S
          INNER JOIN inserted I ON S.Signal_id=I.Signal_id
END
```

Рис. 3

прос, определяющий представление, и отправлять его на сервер, и, вместе с тем, выполняет функцию безопасности, поскольку клиентским приложениям можно открыть доступ к данным через представление и запретить работать с таблицами напрямую.

Стратегия хранения информации

В процессе функционирования системы экологического мониторинга было установлено, что неограниченный рост таблиц, в которые поступает оперативная информация, приводит к замедлению работы клиентских приложений с этими таблицами. С другой стороны, по мере устаревания информации необходимость в ней возникает все реже. В связи с этим была разработана стратегия хранения информации, при которой по мере устаревания информация накапливается в таблицах за более продолжительный период времени.

Оперативная часто обновляемая информация (информация по воздуху, метеопараметры) поступает в соответствующие таблицы и хранится в них в течение 7 дней. Более старая информация за последний год накапливается в таблицах текущего года. Информация за прошедшие годы содержится в таблицах этих лет.

Постоянное обновление таких таблиц требует однообразных рутинных действий, которые можно автоматизировать при помощи службы SQL Server Agent, предусмотренной в Microsoft SQL Server. SQL Server Agent позволяет создавать задания, которые будут выполняться по расписанию, периодически или однократно в назначенное время, уведомлять операторов об ошибке или завершении выполнения различных автоматизированных задач, а также о наступлении определенных событий или условий производительности.

Для автоматизации указанной стратегии хранения информации были созданы ХП, которые вызываются при помощи заданий SQL Server Agent по заданному расписанию.

Список литературы

1. *Прангишвили И.В., Лебедев В.Г., Легович Ю.С., Толстых А.В., Воронин Б.Н., Назаров В.Д.* Комплексная система производственного экологического мониторинга промышленного предприятия // 4-й междун. форум "Высокие технологии XXI века". http://www.hitechno.ru/ecology-4.doc
2. *Кулида Е.Л., Лебедев В.Г., Крюкова И.П.* Создание БД системы производственного экологического мониторинга объекта уничтожения химического оружия // Приборы и системы. Управление, контроль, диагностика. 2003. №8.
3. Проектирование и реализация БД Microsoft SQL Server 2000. М.: Издательско-торговый дом "Русская Редакция". 2003.

Кулида Елена Львовна *– канд. техн. наук, ст. науч. сотрудник,*
Крюкова Ирина Павловна *– ведущий инженер-программист,*
Лебедев Валентин Григорьевич *– канд. техн. наук, зав. лабораторией Института проблем управления РАН.*
Контактные телефоны (095) 334-92-31, 334-92-49. E-mail: valya@ipu.rssi.ru