14

Распределенные запросы, транзакции и базы данных

В организациях с распределенным владением данных эти данные могут размещаться в нескольких различных базах и в различных системах или узлах. Такие распределенные базы данных могут быть построены на основе программного обеспечения одного или нескольких поставщиков. В тех случаях, когда информацию необходимо получать из нескольких источников баз данных, используют распределенные запросы. Если требуется выполнять обновление данных в нескольких базах, можно реализовать распределенные транзакции.

В этой главе рассматриваются некоторые из основных технологий, используемых для связывания воедино распределенных данных и баз. Например, распределенные базы данных Oracle могут быть объединены за счет установки связей между отдельными базами. Применяя представления, процедуры и синонимы, программист может делать доступ к этим распределенным источникам данных прозрачным для конкретного местоположения. Мы рассмотрим дополнительные особенности, присущие распределенному обновлению нескольких баз данных Oracle в форме транзакций. Будут освещены также вопросы подключения к распределенным базам данных, отличных от Oracle, с помощью таких средств, как Heterogeneous Services (Гетерогенные службы), ODBC (Open Database Connectivity — Открытый интерфейс доступа к базам данных) и шлюзы.

Управление распределенными базами данных может осуществляться автономно — то есть управление каждой базой данных реализуется через ее собственный управляющий интерфейс. Фактически, начиная с версии Oracle Database 10*g*, установка каждой базы данных Oracle включает в себя установку на сервере интерфейса Enterprise Manager Database Control (Управление базой данных руководителя предприятия). В целях упрощения управления несколькими экземплярами Oracle управление распределенными базами данных Oracle может осуществляться с помощью единого интерфейса, получившего название Enterprise Manager Grid Control (Управление сетью руководителя предприятия). Поскольку данная книга рассчитана на программистов, в различных разделах этой главы интерфейс Enterprise Manager упоминается в кон-

тексте активизации распределенных баз данных, но для получения более подробной информации по управлению распределенными базами данных Oracle рекомендуем обратиться к документации по администрированию.

Многие организации вместо развертывания распределенных баз данных собирают данные в единой базе. Часто для обеспечения целостности данных и для создания информационного хранилища, представляющего единственную истинную версию данных, компании создают хранилища оперативных и производственных данных всей организации. Дополнительное преимущество такого подхода — упрощение настройки, поскольку эффективность оптимизатора может быть более предсказуемой в случае использования единой обобщенной базы данных. Естественно, управление транзакциями также в полной мере выполняется и единственной СУБД. Методы перемещения данных, используемые при загрузке таких хранилищ, освещены в главах 24 и 25 этой книги. В них описано высокоскоростное перемещение данных с помощью таких средств, как Data Pump, Transportable Tablespaces и Streams, а также более традиционных технологий загрузки и управления данными, которые удобны при перемещении данных в единую базу, включая использование SQL*Loader, External Tables и Change Data Capture.

Рассмотрение этих вопросов начнем с ознакомления с методами связывания распределенных баз данных Oracle и некоторыми особенностями применения этих связей с точки зрения программирования.

Связывание распределенных баз данных Oracle

Для выполнения запросов распределенные базы данных Oracle связываются за счет установки и поддержания связей баз данных. Чаще всего связь базы данных (database link) создается ее администратором, ответственным за управление доступом к распределенным источникам данных.

При создании связи базы данных используется имя базы данных, с которой необходимо установить связь, и имя сервера или домена, в котором она размещена. Эта комбинация имени базы данных и имени сервера представлена в глобальном имени базы данных. Глобальное имя обеспечивает большую уникальность, поскольку с высокой степенью вероятности во многих организациях одинаковые имена баз данных могут существовать в более чем одном узле (например, базы данных `sales` (продажи) могут с различными целями использоваться в двух различных подразделениях). Параметры именования могут быть инициализированы с помощью интерфейса Enterprise Manager Database Control и находятся в разделе Initialization Parameters (Параметры инициализации) вкладки Administration (Администрирование). Интерфейс SPFile с добавленным именем домена показан на рис. 14.1. Имена доменов должны подчиняться соглашению по стандартным именам Internet. В приведенном примере в качестве имени домена (`db_domain`) указано имя базы данных компании Stack.

После того как имя домена для глобальных имен определено, потребуется создать связь одной базы данных с другой. Как правило, связи создают приватными или общедоступными, в зависимости от присвоенных полномочий на создание таких связей и типа доступа, который должен быть предоставлен пользователям. По умолчанию пользователи связи и их пароли будут совпадать с параметрами исходной базы данных. Приватные связи ограничивают круг пользователей, в то время как работать с общедоступной связью может любой пользователь.

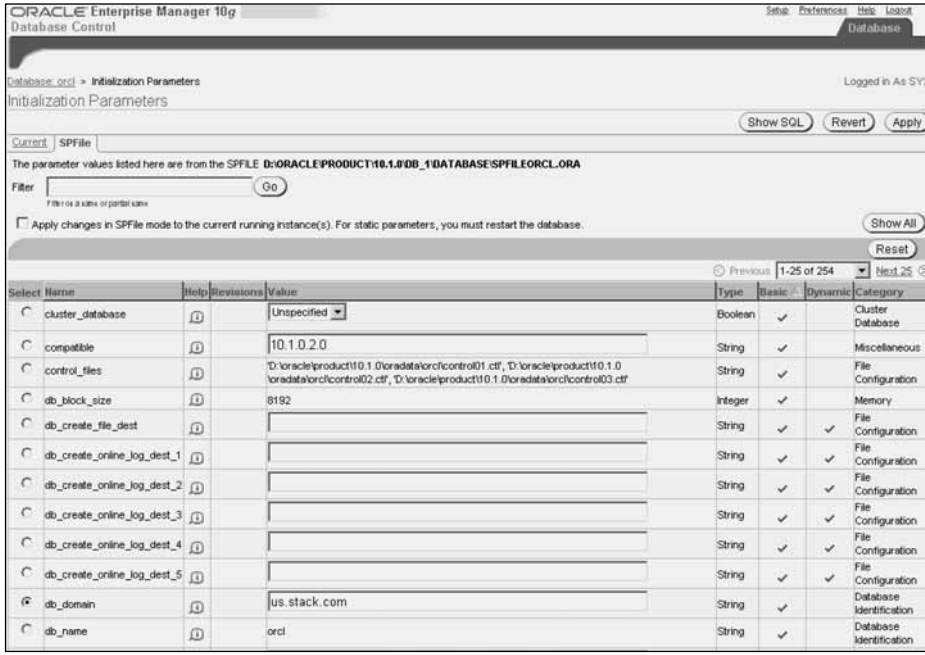

Рис. 14.1. Указание имени домена базы данных

Можно также создавать связи с указанными именами пользователей и паролями. Такие связи называют *аутентифицированными связями*, и их можно применять для установления как приватных, таки и общедоступных подключений.

Приватную связь для базы данных `orcl` в домене `us.stack.com` с использованием глобального имени базы данных можно создать с помощью следующего SQL-оператора:

```
CREATE DATABASE LINK orcl.us.stack.com;
```

Общедоступная связь для этой же базы данных создается с помощью аналогичного SQL-оператора, но с указанием ключевого слова `PUBLIC`:

```
CREATE PUBLIC DATABASE LINK orcl.us.stack.com;
```

Связь базы данных можно создать также с помощью интерфейса Oracle Enterprise Manager. Интерфейс (вкладка Administration диалогового окна Enterprise Manager Database Control), посредством которого администратор базы данных мог бы создать связь, указывая имя связи базы данных, имя службы Oracle*Net, тип соединения, имя пользователя и пароль, показан на рис. 14.2.

Как только связь создана, с ее помощью можно подключиться к базе данных, указывая ее глобальное имя. Например, запрос может иметь следующий вид:

```
SELECT * FROM sh.products@orcl.us.stack.com;
```

В программах распределенных баз данных удобно применять представления, синонимы и процедуры, поскольку они могут создавать видимость прозрачности местоположения базы данных, скрывая используемые глобальные имена. В следующем примере будет образовано представление `supply`, которое обеспечит прозрачность местоположения, указывая глобальное имя базы данных для выполнения операции

SELECT для выбора данных из удаленной таблицы товаров, одновременно позволяя выполнять эту же операцию для выбора данных из локальной таблицы.

```
CREATE VIEW supply AS
SELECT i.product_id, i.warehouse_id, i.quantity_on_hand, p.prod_name, p.prod_status
FROM oe.inventories i, sh.products@orcl.us.stack.com p
WHERE i.product_id = p.prod_id;
```

Синонимы — это ссылки на объекты, которые могут быть созданы для таблиц, типов, представлений, материализованных представлений, последовательностей, процедур, функций и пакетов. Синоним для использованной в предыдущем примере таблицы products базы данных orcl.us.stack можно создать следующим образом:

```
CREATE SYNONYM products FOR sh.products@orcl.us.stack.com;
```

Администратор базы данных может также создать синоним с помощью интерфейса Enterprise Manager Database Control (вкладка Administration), как показано на рис. 14.3.

Процедуры могут обеспечивать прозрачность, выполняя ссылки на связанные данные и внедряя глобальное имя в ссылку либо поддерживая синонимы. Чтобы оценить различные возможности организации доступа к распределенным данным в своих приложениях, программисту, скорее всего, придется сотрудничать с администраторами баз данных.

Распределенные приложения, в которых используются связи баз данных, требуют учета ряда обстоятельств. Поскольку по сети могут передаваться большие объемы данных, приложения следует проектировать так, чтобы они передавали только действительно необходимые данные.

В целях наиболее эффективной оптимизации потребуется независимо от используемой версии Oracle использовать оптимизатор, работающий на основе оценки затрат. (Помните, что в версии Oracle Database 10*g* и последующих оптимизатор на основе затрат является единственным, поддерживаемым Oracle.) Во всех случаях, когда это возможно, оптимизатор на основе затрат будет переписывать распределенные запросы, используя обобщенные встраиваемые представления.

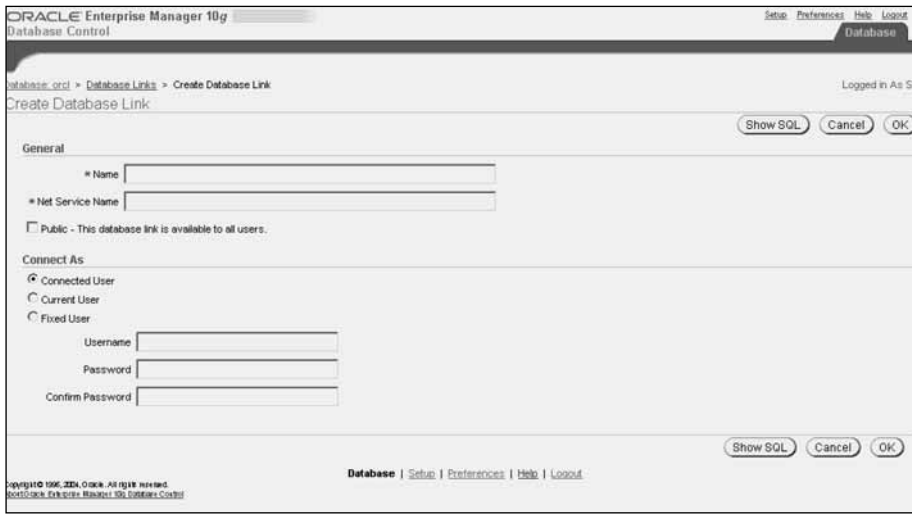

Рис. 14.2. Экран создания связи базы данных интерфейса Oracle Enterprise Manager

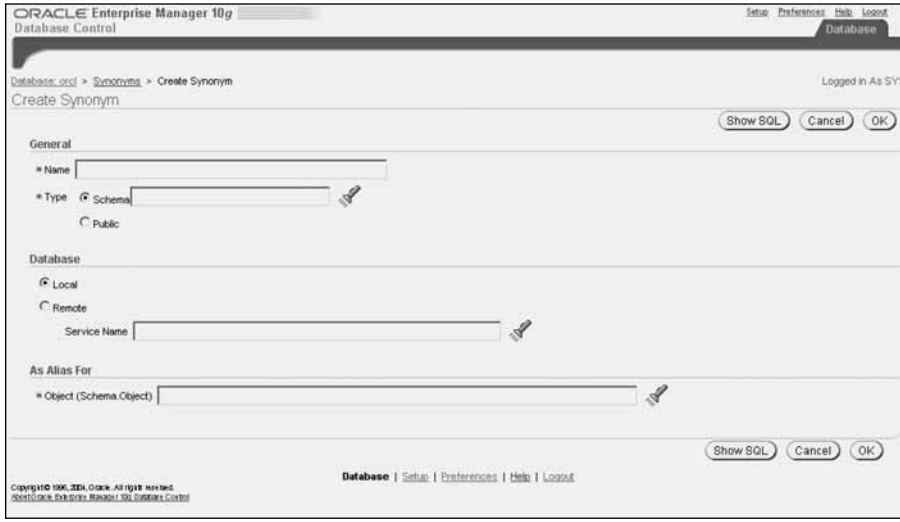

Рис. 14.3. Создание синонима в интерфейсе Oracle Enterprise Manager

Обобщенное встраиваемое представление позволяет выполнять операцию выбора данных из нескольких таблиц в одной локальной базе данных, тем самым уменьшая количество обращений к удаленной базе данных. Оптимизация включает в себя слияние всех допускающих это представлений, создание обобщенного встраиваемого представления, выполнение проверки блока обобщенного запроса и перезапись запроса с применением обобщенных встраиваемых представлений. Естественно, оптимизация полностью прозрачна для пользователя или приложения, представляющего запрос. В общем случае такая оптимизация не очень эффективна при наличии агрегирований, подзапросов или сложного SQL-кода.

Кроме поддержки оптимизатора иногда используют еще один метод повышения производительности и облегчения оптимизации — построение представлений. Например, можно создать представление для нескольких удаленных таблиц. Построение процедурного кода всегда возможно, хотя оно и не слишком часто требуется для реализации распределенных запросов.

В тех ситуациях, когда обновления выполняются в распределенных базах данных, составленных из нескольких отдельных баз данных, сбой какой-то части распределенного оператора может привести к нарушению требований целостности данных. В следующем разделе мы рассмотрим двухфазную фиксацию, поскольку приложения будут поддерживать эту возможность, и одновременно должны будут выполнять обработку нестандартных ситуаций, возникающих во время обновления транзакций.

Распределенные транзакции и двухфазная фиксация

Распределенные транзакции имеют место, когда один или более операторов обновляют данные в двух и более базах данных. Распределенные операции могут включать в себя транзакции языка манипулирования данными (data manipulation language — DML) или языка определения данных (data definition language — DDL) либо

операторы управления транзакциями. К операциям DML и DDL, поддерживаемым СУБД Oracle как распределенных транзакций, относятся CREATE TABLES AS SELECT, DELETE, INSERT, LOCK TABLE, SELECT и SELECT FOR UPDATE. В число поддерживаемых операторов управления транзакциями входят COMMIT, ROLLBACK и SAVEPOINT.

Целостность данных при выполнении изменений в распределенных базах данных обеспечивается Oracle с помощью двухфазной фиксации. Первая фаза двухфазной фиксации распределенных транзакций называется фазой предварительной подготовки. На этой фазе инициирующая система или узел (называемый глобальным координатором) уведомляет все сайты, участвующие в транзакции, и подготавливает их к выполнению фиксации или отката. Узлы, которые для завершения транзакции вынуждены ссылаться на данные в других узлах, иногда называют локальными координаторами. Подготовка включает в себя запись информации в сетевые журналы повтора для последующего выполнения фиксации или отката и размещение локальных блокировок на измененные таблицы в целях предотвращения считывания незафиксированных данных.

На второй фазе двухфазной фиксации выполняется либо фиксация транзакции, либо ее откат. Автоматический откат предпринимается, если фиксация транзакции невозможна в любом из узлов. Сайт пункта фиксации инициализирует фазу фиксации по инструкции глобального координатора. В качестве сайта пункта фиксации назначается наиболее надежный сайт, имеющий наибольшее значение параметра инициализации COMMIT_POINT_STRENGTH (обычно это значение устанавливает администратор базы данных). После фиксации транзакции каждый узел снимает свои блокировки, вносит запись о фиксации в свой локальный журнал повтора и уведомляет об этом глобального координатора.

В обычных ситуациях выполняемая в Oracle обработка распределенных транзакций вполне надежна. На случай непредвиденных ситуаций, таких как сбой сервера, сети или программного обеспечения в любой момент этого процесса, в СУБД Oracle имеется процесс RECO (процесс восстановления), призванный решать проблемы, связанные с транзакциями. Однако лучше все же проектировать приложения с учетом таких нештатных ситуаций, обеспечивая прозрачную автоматизацию процесса.

Рассмотрим, что происходит, если приложение не учитывает возможность сбоев, и создается сомнительная транзакция, которая требует разрешения вручную.

Когда такой сбой происходит, а приложение о нем не знает, данные блокируются для чтения и записи. Транзакции в этом состоянии называются сомнительными. Администратор базы данных может вмешаться для разрешения этой ситуации, поскольку такие сомнительные транзакции отображаются в представлении DBA_2PC_PENDING. Администратор базы данных может просмотреть значения полей LOCAL_TRAN_ID (локальный идентификатор транзакции), GLOBAL_TRAN_ID (глобальный идентификатор транзакции), STATE (состояние: сбор данных, подготовлена, зафиксирована, принудительная фиксация или принудительное прерывание/откат), MIXED (например, фиксация в одном узле и откат в другом), TRAN_COMMENT (описательное название транзакции), HOST (имя хост-компьютера) и COMMIT# (глобальный номер фиксации).

Администратор отслеживает сеанс с помощью представления DBA_2PC_NEIGHBORS. Это представление содержит поля LOCAL_TRAN_ID, IN_OUT (входящие или исходящие транзакции), DATABASE (инициатор клиентского подключения или связь базы данных), DBUSER_OWNER (локальный пользователь или владелец связи базы данных) и INTERFACE (состояние). Исходя из полученной из этого представления информации, администратор базы данных может предпринять соответствующие действия по изменению состояния транзакции.

Службы Heterogeneous Services

СУБД Oracle предоставляет службы Heterogeneous Services (Гетерогенные службы), используемые для получения доступа к системам баз данных, отличным от Oracle. Программисты могут подготовить SQL-операторы или процедуры для получения доступа к отличным от Oracle средам, а затем их применять для обеспечения подключений, поддерживаемых службами Heterogeneous Services. Службы Heterogeneous Services обеспечивают обобщенное подключение через интерфейс ODBC (Open Database Connectivity –открытый интерфейс доступа к базам данных) или OLE DB (используется для доступа к реляционным и нереляционным источникам данных). Heterogeneous Services могут также активизировать службу Transparent Gateways (Прозрачные шлюзы) Oracle, позволяющую получать доступ к определенным отличным от Oracle источникам баз данных с большей производительностью, чем через обобщенный интерфейс.

Для каждого подключения, использующего обобщенный интерфейс или интерфейс Transparent Gateways, устанавливается выделенный агент служб Heterogeneous Services. Агенты обобщенного подключения (ODBC или OLEDB) используют драйверы, установленные в СУБД Oracle. Агенты Transparent Gateways отличаются тем, что могут быть установлены в любой поддерживаемой системе. Агенты обеспечивают трансляцию SQL-кода и преобразование типов данных. Типичная топология показана на рис. 14.4.

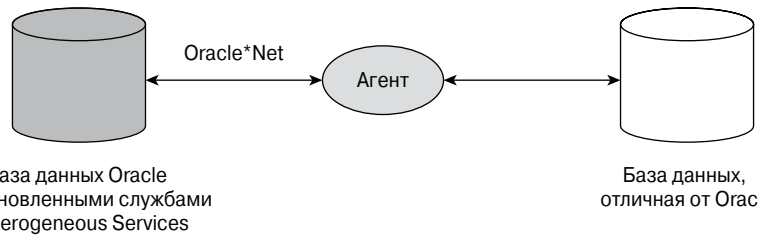

Рис. 14.4. Обмен данными между базой данных Oracle с установленной службой Heterogeneous Services и базой данных, отличной от Oracle

При использовании интерфейса Transparent Gateways и наличии большого количества подключений администратор базы данных может устанавливать многопоточные агенты. В целях минимизации использования системных ресурсов многопоточные агенты могут быть сконфигурированы для предоставления одного потока мониторинга и нескольких потоков диспетчеризации и выполнения задач. Применение интерфейсов Transparent Gateways позволяет также увеличить производительность, поскольку они предоставляют доступ отличным от Oracle системам через собственный интерфейс целевой базы данных, а не обобщенный интерфейс.

Чтобы читателям было проще понять процесс конфигурирования служб Heterogeneous Services для выполнения подключений, рассмотрим в качестве примера интерфейс ODBC.

ODBC

В целях обеспечения к ним доступа с помощью универсального SQL-кода многие СУБД поддерживают интерфейс ODBC. Интерфейс ODBC для Oracle предоставляет

программный интерфейс приложений (application programming interface — API) ODBC и обеспечивает поддержку всех основных функциональных возможностей, определенных в ODBC, а также поддержку синтаксиса, описанного спецификацией SQL-99.

Интерфейс Oracle поддерживает набор драйверов Oracle и других СУБД. Например, начиная с версии Oracle9*i* выпуска 2, с помощью поставщика ODBC.NET от компании Microsoft поддерживается технология .NET. Посредством ODBC Oracle обеспечивает поддержку и других технологий Microsoft, в том числе возможность использования сервера Microsoft Transaction Server (Сервер транзакций Microsoft) и EXEC-синтаксиса SQL Server. К числу дополнительных технологий, поддерживаемых Oracle, относятся поддержка Transparent Application Failover (Прозрачное устранение последствий сбоя приложения) службой Failsafe (Защита от сбоев) Oracle и поддержка RAC (Real Application Clusters — Реальные кластеры приложений).

Обобщенный интерфейс подключения служб Heterogeneous Services работает с инициализационными файлами, хранящимися в подкаталогах `hs` и `admin` каталога установки Oracle. Например, в типичной СУБД Oracle 10*g*, установленной под Windows, подкаталог `oracle\product\10.1.0\Db_1\hs\admin` в числе прочих будет содержать два файла: `inithsodbc.ora` и `initholedb.ora`. Первоначально эти файлы инсталлируются в качестве инициализационных файлов образца агента, соответственно содержащих параметры HS, необходимые для работы агентов ODBC и OLE DB. В этих файлах можно определять более 20 параметров, которые описаны в руководстве *Oracle Database Heterogeneous Connectivity Administrator's Guide* (Руководство администратора гетерогенных подключений баз данных Oracle). В этом же подкаталоге можно также найти образцы файлов `listener.ora` и `tnsnames.ora`.

Конфигурирование ODBC для доступа к источникам данных, отличных от Oracle — многоэтапный процесс, обычно выполняемый администратором базы данных. Вначале администратор добавляет запись `SID_DESC` в файл `listener.ora`, предназначенный для использования программой `hsodbc`. Затем в файл `tnsnames.ora` потребуется добавить новое описание SID, раздел `connect_data` которого должен содержать параметр `HS=OK`. На этом этапе файл инициализации SID (как правило, модифицированный на основе образца файла `inithsodbc.ora`) уже также установлен.

Затем нужно сконфигурировать клиентскую часть. Например, при установке на платформе Windows потребуется запустить утилиту Configuration and Migration Tools (Инструменты конфигурирования и миграции), инсталлированную в числе других программ Oracle, а затем выбрать команду меню Microsoft ODBC Administrator (Администратор Microsoft ODBC). В появившемся диалоговом окне ODBC Data Source Administrator (Администратор источников данных ODBC) необходимо перейти на вкладку System DSN (Системный DSN) и добавить драйвер Microsoft Access Driver (Драйвер Microsoft Access), как показано на рис. 14.5, после чего сконфигурировать его в качестве источника данных в соответствии с ранее созданным описанием ODBC. Затем потребуется создать связь базы данных и проверить ее путем обращения к удаленной базе данных.

В тех случаях, когда для подключения к источнику данных Oracle из другого компьютера используется ODBC, на этом компьютере должен быть установлен клиент Oracle OCI. При этом на сервере Oracle должен быть настроен соответствующий слушатель. Созданное приложение должно будет передать имя службы TNS (Transparent Network Substrate — Прозрачная сетевая среда) и имя пользователя и пароль Oracle.

При разработке приложений, использующих ODBC, для обеспечения максимальной производительности необходимо придерживаться нескольких правил здравого

смысла. Всегда, когда возможно, следует применять переменные связывания, что обеспечит возможность многократного использования операторов, и будет способствовать поддержке кэша Oracle. В оператор `SELECT` не следует включать столбцы, которые не будут использоваться. Если приложение будет выполнять частые подключения и отключения от базы данных, рекомендуется применять пул подключений.

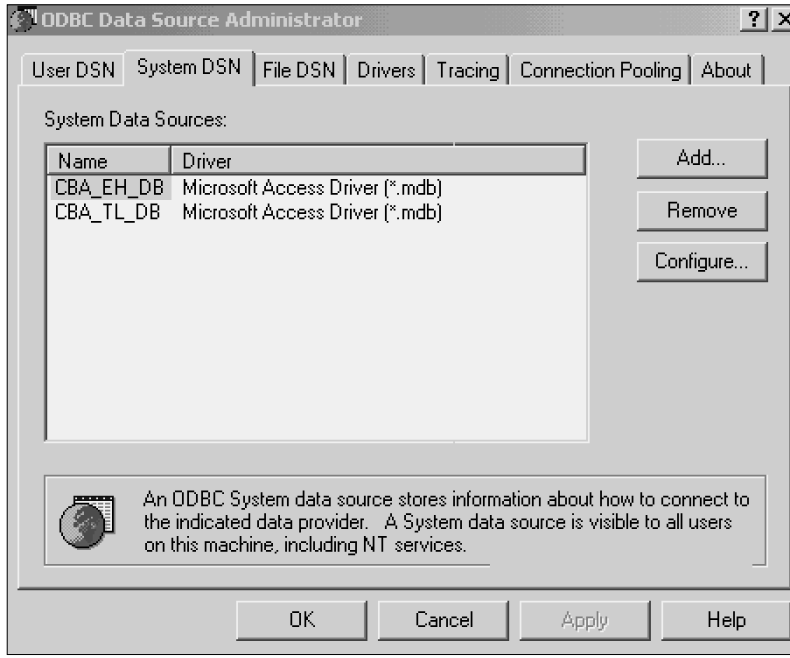

Рис. 14.5. Диалоговое окно ODBC Data Source Administrator в программе Configuration and Migration Tools в системе Windows

Transparent Gateways

Интерфейс Transparent Gateways позволяет получать доступ к базам данных, отличным от Oracle, так, как если бы они были базами данных Oracle. Это достигается с использованием того же SQL-синтаксиса Oracle. Как уже отмечалось, с помощью многопоточных агентов интерфейс Transparent Gateways может обеспечивать более высокий уровень производительности. Этот интерфейс удобен также в ситуациях, в которых требуются хранимые процедуры или распределенные транзакции, поскольку обобщенный интерфейс подключения к базам данных эти функциональные возможности не поддерживает.

Oracle предлагает ряд шлюзов для различных операционных систем. Для каждой конкретной аппаратной платформы следует тщательно проанализировать доступность и возможности целевых шлюзов, поскольку для определенных платформ доступны только определенные шлюзы.

Все доступные шлюзы можно разделить на следующие категории.

- **Шлюзы общедоступных систем**. К ним относятся шлюзы для источников данных Microsoft SQL Server, Sybase, Rdb, Ingres, Informix, Teradata и DMS.

- **Интеграционные шлюзы предприятия**. К ним относятся Access Manager для AS/400, Procedural Gateways для MQSeries и APPC и Transparent Gateways для DB2/400 и IBM DRDA.
- **Универсальные интеграционные шлюзы**. К ним относят шлюзы для DB2.
- **Шлюзы EDA/SQL**. К ним относят шлюзы для других универсальных источников данных, таких как VSAM, IMS, ISAM и другие.

С каждым шлюзом связано отдельное руководство администратора для тех платформ, для которых он доступен, содержащее описание поддерживаемого синтаксиса и функций.

Во время подключения интерфейс Transparent Gateway конкретного целевого источника данных, отличного от Oracle, сообщит СУБД Oracle о поддерживаемых функциях. Если приложение использует в операциях `INSERT`, `UPDATE` или `DELETE` специфичные функции Oracle, не поддерживаемые целевым источником данных, возникнет ошибка ORA-02070 со следующим сообщением: `database db_link_name does not support function in this context` (база данных имя_связи_БД не поддерживает функцию в этом контексте). В случае обращения к другим неподдерживаемым функциям (например, во время выполнения оператора `SELECT`) данные будут передаваться для обработки локальному экземпляру Oracle. Поэтому при проектировании и настройке приложений во избежание потенциальных ошибок и для увеличения производительности следует обращать внимание на то, какие функции поддерживаются в целевом источнике данных через шлюз.

Резюме

В целях увеличения производительности и упрощения программирования многие организации стараются, когда это возможно, объединять данные в единые базы. Для тех случаев, когда подобное объединение не возможно, Oracle предлагает ряд способов построения распределенных баз данных, состоящих как целиком из баз данных Oracle, так и смешанных (гетерогенных).

В основном распределенные базы данных определяются администраторами с помощью связей. В конфигурациях, состоящих только из баз данных Oracle, запрос распределенных баз данных сравнительно прост, а в смешанных конфигурациях требуется лишь знание любых ограничений, действующих в службах или шлюзах, используемых в этих конфигурациях. Обеспечивая возможность выполнения распределенных транзакций, Oracle поддерживает двухфазную фиксацию, которая также упрощает реализацию приложений, хотя программистам может требоваться учет возможности возникновения нештатных ситуаций в разрабатываемых ими приложениях.

Распределенные базы данных требуют учета ряда дополнительных соображений по поводу настройки производительности. Разработчики приложений и программисты должны всегда ставить перед собой цель снижения трафика между базами данных во всех случаях, когда это возможно.

Эта глава завершает собой часть книги, посвященную данным. Теперь, когда мы рассмотрели широкий круг основополагающих вопросов, вы должны быть готовы приступить к изучению материала, посвященного технологиям программирования на языках PL/SQL и Java.