Jet Info

ИНФОРМАЦИОННЫЙ БЮЛЛЕТЕНЬ № 9-10(64-65)/1998

Шлюзы как средство интеграции баз данных. Практический подход

Объектные технологии в продуктах Oracle

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Новый стратегический продукт корпорации Oracle

10 ноября корпорация Oracle официально представила свой новый продукт — Oracle8*i* (см. http://www.oracle.com/database/oracle8*i*/). Уже из названия ясно, что речь идет о соединении двух технологий: СУБД и Интернет/Интранет.

Как сервер СУБД, Oracle8*i* является преемником Oracle8, объектно-ориентированные возможности которого рассмотрены в статье «Объектные технологии в продуктах Oracle», помещенной в этом же номере Jet Info. Однако не эти традиционные свойства составляют суть Oracle8*i*. Более важным, на наш взгляд, является включение в комплект поставки сервера приложений Oracle Application Server (также описанного в упомянутой выше статье), обеспечение поддержки Java на уровне сервера СУБД и реализация возможности видеть содержимое базы данных как дисковый том, что позволяет естественным образом состыковать Web-сервер и сервер баз данных. Кроме того, в состав Oracle8*i* входят картриджи для работы с нетрадиционными для реляционных СУБД видами данных, а также средства администрирования корпоративного класса.

Далее мы кратко рассмотрим перечисленные свойства и компоненты Oracle8*i*.

1. Совершенствование сервера СУБД

Как сервер СУБД, Oracle8*i* является очередной ступенью на пути совершенствования Oracle8. В Oracle8*i* улучшены средства управления ресурсами (процессорами, дисками) в многопользовательской среде. Это важно для смешанных конфигураций, где сочетаются оперативные транзакции и аналитические запросы. За счет назначения приоритетов можно добиться первоочередного выполнения оперативных действий.

Повышению эффективности работы сервера СУБД способствуют улучшенные возможности фрагментирования таблиц и индексов. Такое фрагментирование можно рассматривать как средство распараллеливания обработки запросов. Впрочем, вероятно, еще больший эффект можно ожидать от многочисленных усовершенствований, привнесенных корпорацией в Oracle Parallel Server.

Лет тридцать (если не более) назад в работах по искусственному интеллекту появилось понятие функции с памятью — функции, запоминающей свои результаты для некоторых наборов параметров и избегающей тем самым повторных вычислений. Теперь аналогичная возможность под названием «управление сводками» (Summary Management) появилась и в Oracle8*i*. Можно один раз вычислить сводную информацию о таблице и затем быстро выдавать ее в ответ на последующие запросы.

В Oracle8*i* открыт программный интерфейс к средствам загрузки данных. Теперь независимые производители программного обеспечения могут создавать продукты, не уступающие в эффективности «родному» SQL*Loader.

Повышена эффективность и при работе с хранилищами данных. Главный «козырь» здесь — расширенная поддержка операций с индексами и упрощение перемещения данных в хранилища или из них.

Современный корпоративный сервер СУБД невозможно представить без средств обеспечения надежности, готовности, обслуживаемости. В Oracle8*i* развиты возможности для поддержки резервных серверов, впервые появившиеся в версии Oracle 7.3. Журналы «накатки» теперь автоматически передаются на резервный сервер и выполняются там. Впрочем, по своему желанию системный администратор может временно остановить «автомат» и сделать резервную базу доступной на чтение, чтобы разгрузить тем самым основной сервер. После окончания срочных работ «накатка» может быть возобновлена.

В Oracle8*i* улучшены средства резервного копирования и восстановления данных. Менеджер восстановления (Recovery Manager) способен эффективно использовать параллелизм, присущий аппаратным платформам серверов СУБД.

Совершенствование механизмов репликации повышает как надежность, так и эффективность работы корпоративных информационных систем. В Oracle8*i* эти механизмы «погружены» в ядро СУБД, за счет чего достигается, кроме всего прочего, более высокая защищенность и прозрачность для остальных компонентов.

Важный аспект надежной работы сервера — постоянный мониторинг. С помощью «просмотрщика журналов» (LogMiner) администратор может оперативно следить за «жизнью»

СУБД, анализировать выполняемые SQL-запросы, видеть, кто, что и когда делает. В принципе администратор имеет достаточно информации, чтобы помочь в оптимизации работы пользователей.

2. Java в Oracle8*i*

Всемерная поддержка Java — один из краеугольных камней стратегии Oracle (см. раздел «Java-средства» в статье «Объектные технологии в продуктах Oracle»). Теперь Java-технология (точнее, виртуальная Java-машина) проникла в святая святых — сервер СУБД. В сочетании с поддержкой SQLJ — средства встраивания SQL-операторов в Java-программы — это дает разработчикам возможность писать хранимые процедуры и триггеры на Java. Отметим, что SQLJ по сути является открытым стандартом, в разработке которого, помимо Oracle, участвовали такие компании, как Sun и IBM.

В Oracle8*i* поддерживаются важнейшие объектные спецификации, в том числе Enterprise JavaBeans, CORBA, IIOP. Входящий в состав Oracle8*i* сервер приложений (Oracle Application Server) играет роль открытого, распределенного, стандартизованного каркаса, в который могут быть встроены объекты-компоненты, обладающие прикладной функциональностью.

Для корпорации Oracle всегда была характерна поддержка полного жизненного цикла информационных систем. Не стал исключением и Oracle8*i*. В его состав входят JDeveloper — компонентная среда разработки Java-приложений, а также WebDB — средство построения, распространения и мониторинга Web-ориентированных приложений для баз данных.

3. Файловая система iFS

Нетрудно догадаться, что iFS расшифровывается как internet File System. С технической точки зрения iFS представляет собой Java-приложение, выполняющееся на Java-машине Oracle8*i* и позволяющее 32-битным Windows-приложениям (таким, например, как навигатор, файловый или почтовый клиент) воспринимать содержимое базы данных как файловую систему. Реляционные таблицы и данные смешанного характера предстают в виде файлов, то есть в универсальной форме, допускающей простое и единообразное использование и администрирование.

Более точно, iFS обеспечивает взаимодействие со следующими клиентскими протоколами и платформами:

- SMB для Windows 95, 98 и NT;
- HTTP и FTP;
- SMTP, IMAP4, POP3.

Пользователи могут буксировать файлы iFS обычным для Windows образом, редактировать их и т.п. Таким образом, iFS существенно понижает порог сложности, присущий реляционным СУБД. Таблицы и их строки оказываются «на кончиках пальцев» без реляционной премудрости.

4. Картриджи данных interMedia

В статье «Объектные технологии в продуктах Oracle» подробно рассматривается механизм картриджей данных, а также картридж для интеллектуальной работы с текстами ConText (см. раздел «Картриджи данных»). Представляется вполне естественным, что корпорация Oracle не стала откладывать получение дивидендов со столь ценных механизмов и продуктов, включив в состав Oracle8*i* семейство картриджей interMedia.

В семейство interMedia, помимо картриджа ConText, входят средства для работы с аудио- и видеоданными, а также изображениями. Поддерживается потоковая доставка аудио и видео, например, с помощью Oracle Video Server или серверов RealAudio и RealVideo компании RealNetworks. Обеспечена интеграция interMedia со средствами создания Web-приложений и Web-документов (в число которых, напомним, входит WebDB).

Включена в interMedia и поддержка географических данных. Это значит, что в рамках привычного Web-интерфейса пользователи, опираясь на Oracle8*i*, могут отыскивать ближайшие объекты интересующего их характера.

5. Средства управления

В состав Oracle8*i* включен Oracle Enterprise Manager Version 2, отличающийся улучшенными показателями масштабируемости, надежности и простоты использования. Ключом к успеху стала трехуровневая архитектура, существенно облегчающая балансировку нагрузки, нейтрализацию отказов, разделение данных между административными консолями, выполненными, кстати сказать, в виде Java-приложения со всеми вытекающими отсюда плюсами в виде «администрирования из любой точки».

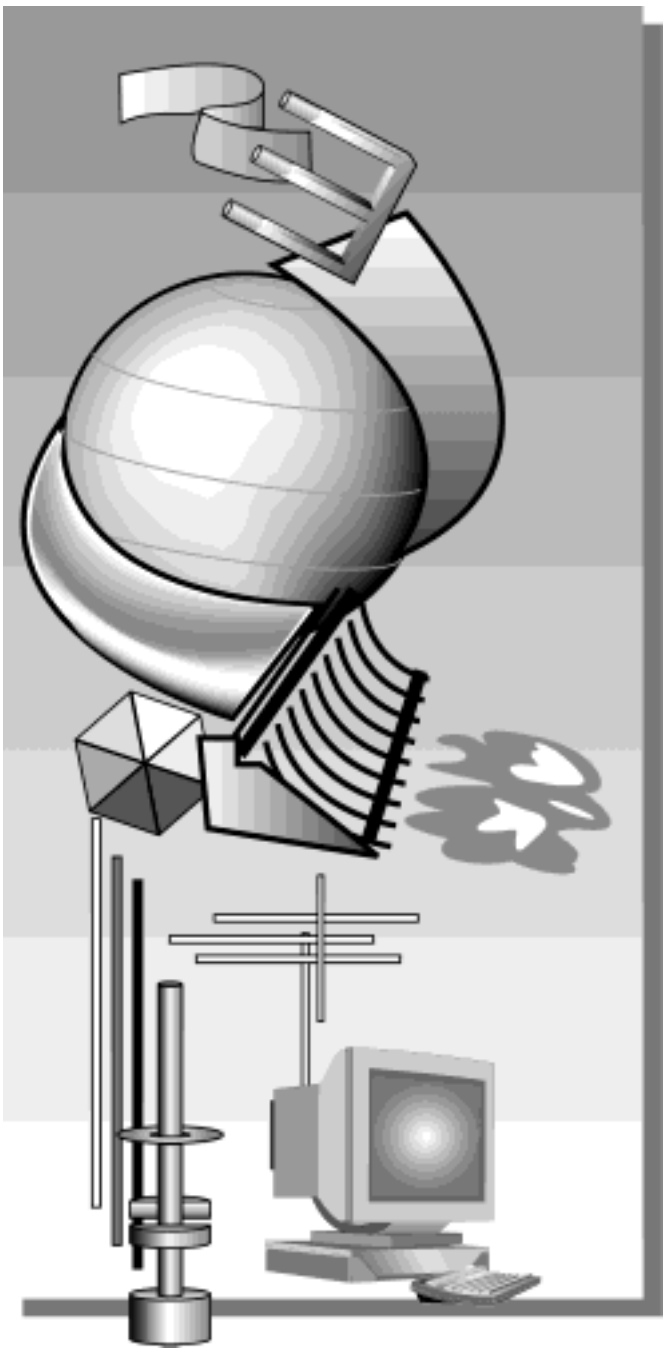
Упрощению администрирования способствует универсальный установщик (Oracle Universal Installer), также выполненный в виде Java-приложения. Установщик обладает интеллектом, достаточным, чтобы успешно справиться даже с кластерными конфигурациями, автоматически помещая программное обеспечение на все узлы кластера.

6. Вместо заключения

Мы видим, что Oracle8*i* является синтетическим продуктом, объединяющим лучшие качества систем управления базами данных и средств, ориентированных на Интернет/Интранет. Подобное слияние представляется необходимым и, одновременно, весьма перспективным, способствующим качественному решению многих трудных практических проблем.

Шлюзы
как средство интеграции баз данных.
Практический подход

Глеб Ладыженский



СОДЕРЖАНИЕ

1. Введение

Практика показывает, что сейчас в целом завершается этап создания оперативных баз данных организаций. В том или ином виде (в виде персональных или промышленных реляционных БД) во многих из них сформировались центры актуальных данных, необходимых для оперативной работы. Темой сегодняшнего дня становятся технологии и программные продукты, способные обеспечить безболезненную интеграцию баз данных, возможность концентрации информации с целью оперативного анализа, долгосрочного планирования и прогнозирования деятельности организации, создание систем поддержки принятия решений (СППР).

В статье представлена одна из базовых технологий интеграции баз данных — технология **шлюзов** (*gateways*). Статья иллюстрирована практическими примерами, взятыми из реальной жизни. Рассматривается группа продуктов Transparent Gateway корпорации Oracle, включающая средства с широким набором возможностей и представляющая, на взгляд автора, наиболее сильное решение на рынке **ПО промежуточного слоя** (*middleware*) в категории интеграционного программного обеспечения (Database-Connectivity Tools — см. классификацию [1]).

На самом деле задача статьи несколько шире. Хотелось бы рассказать, как принципы распределенных баз данных, сформулированные Э. Коддом, находят свое отражение в реальных проектах по интеграции разнородных баз данных и как на практике используются интеграционные средства Oracle, направленные на построение инфраструктуры СППР.

2. Проблемы интеграции баз данных

2.1. Разобщенные центры данных

Исторически в каждой конкретной организации сформировалось, как правило, несколько центров данных. В большинстве случаев это объясняется административными причинами. Важность информации как ресурса вполне осознавалась руководителями. Каждое подразделение стремилось создать свой собственный центр данных. В нем была сконцентрирована информация, находившаяся в ведении данного подразделения. Вовне она предоставлялась по официальным запросам, в электронном виде (на дискетах), но чаще просто на бумаге.

Однако период «феодальной раздробленности» заканчивается. Руководство любой уважающей себя организации не может допустить ситуации, когда информация о ее деятельности разобщена, скрыта внутри подразделений, ею распоряжающихся, и, следовательно, доступ к ней затруднен, а порой просто невозможен. В этой ситуации получение информации лицами, принимающими решения, обрастает массой бюрократических и технических сложностей, что практически не позволяет вести оперативную работу, поскольку выборка любой порции данных требует специальных административных усилий и отнимает массу времени.

В то же время каждый такой центр данных является точкой концентрации оперативной работы с данными. Часто, например, в рамках подобного центра функционирует система оперативной обработки транзакций (OLTP). Сложившаяся технология обработки данных, сделанные ранее значительные инвестиции в аппаратное и программное обеспечение не позволяют кардинально решить проблему их разобщенности путем механического переноса данных в центральную базу.

Тем не менее, новые задачи управления (СППР и системы хранилищ данных) требуют консолидации информации. Необходимы средства интеграции, которые обеспечивали бы не только унифицированный доступ к продолжающим функционировать центрам данных, но и позволяли создать инфраструктуру для доступа к данным, опирающуюся на единые стандарты и единые принципы сетевого взаимодействия.

Необходимо учитывать и процессы объединения (укрупнения посредством поглощения), которые происходят сегодня в России (особенно в банковской сфере). Несомненно, в число наиболее важных аспектов объединения информационных систем (которое неизбежно станет одним из этапов этого процесса) входит конструирование программной инфраструктуры новой организации с обязательным объединением разнородных баз данных.

Вообще говоря, построение осмысленной, архитектурно простой и эффективной программной инфраструктуры в типичной современной организации, где скопилось множество компьютеров различных моделей и эксплуатируются, как правило, СУБД от нескольких поставщиков, является ключевой задачей при создании информационной системы (ИС).

2.2. Унаследованные системы

Проблема унаследованных систем менее актуальна для России, нежели для Северной Америки и стран Западной Европы. Тем не менее в ряде российских компаний она весьма ощутима. Приобретенные ранее монолитные решения (в основном на базе мэйнфреймов) продолжают функционировать, например, как надежная платформа для систем OLTP, однако при решении новых задач (создания хранилищ данных и аналитических систем) акцент смещается на более популярные платформы (RISC/UNIX или Windows NT).

Новые задачи требуют доставки данных к месту их обработки. Необходим своего рода «программный канал» к унаследованным системам, который скрывал бы нижние уровни сетевого взаимодействия и обеспечивал свободный, «прозрачный» доступ к актуальным базам данных на мэйнфреймах или на иных частных (например, AS/400) вычислительных платформах.

Очевидно, что только задачами извлечения данных проблема не ограничивается. Возможна ситуация, когда потребуется синхронное обновление данных в нескольких БД, одна из которых принадлежит унаследованной системе. Так, например, процессинговый центр кредитных карточек может быть создан на основе RISC/UNIX-кластера с СУБД Oracle, в то время как автоматизированная система финансовой организации (Back Office) функционирует на мэйнфрейме. Владельцы карточек хотели бы иметь возможность перевода денег на основной счет в БД на мэйнфрейме и наоборот. Обновление в БД на обеих вычислительных установках возможно в рамках распределенной транзакции, что требует соответствующих программных средств.

Встречаются и задачи иного рода. Пусть, например, разрабатывается новое приложение (не имеет значения, с помощью каких средств), которое будет работать с БД Oracle. Предположим, что приложению потребуется, помимо доступа к БД Oracle, еще и доступ (возможно, и в режиме обновления) к БД на унаследованной системе. Адекватным решением будет использование шлюза, позволяющего работать с этой БД так, как будто это база данных Oracle.

2.3. Возможные решения

Спектр возможных решений по интеграции баз данных не ограничивается, разумеется, только технологией шлюзов. Причина столь пристального внимания к интеграционным средствам этого класса состоит в концептуальной и архитектурной простоте решения, четком ограничении функциональности[1], следствием чего является надежность систем на базе шлюзов. Кроме того, семейство продуктов Oracle Open Gateway вызывает доверие еще и тем, что это — достаточно старые системы, прошедшие длительный период эксплуатации в серьезных условиях (в частности, на мэйнфремах) и, следовательно, их программный код тщательно выверен.

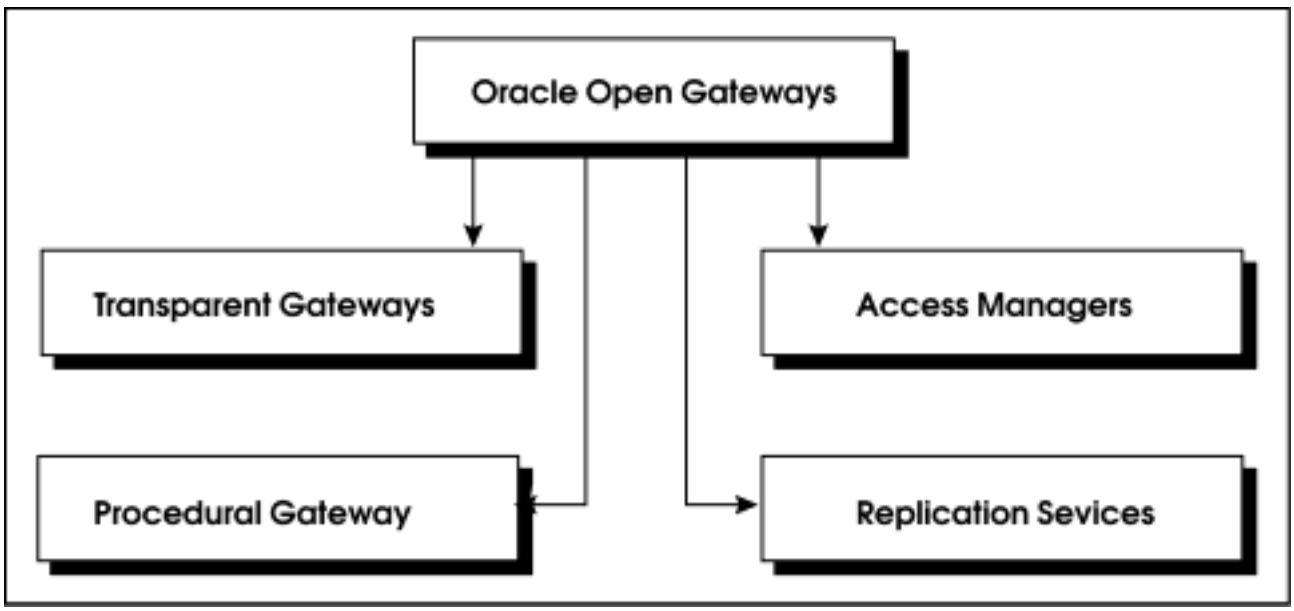


Рис. 1. Семейство продуктов Oracle Open Gateways.

[1] *Несколько дополнительных слов о простоте программных решений. Создание современных информационных систем представляет собой прежде всего конструирование программной инфраструктуры из готовых компонентов (например, на основе сервисов ПО промежуточного слоя, как это предложено в работе [2]). Функциональность компонентов должна быть ясной, обозримой и существенно ограниченной рамками конкретной задачи. Чем проще устроен компонент, тем более предсказуемо его поведение. Шлюзы в этом смысле очень показательны — их исключительная простота гарантирует надежность эксплуатации.*

3. Технология шлюзов Oracle

3.1. Характеристика продуктов

Корпорация Oracle разработала и уже в течение длительного времени предлагает пользователям семейство продуктов, называемое Oracle Open Gateway. Продукты Oracle Open Gateway предназначены для решения интеграционных задач и позволяют в совокупности с другими средствами Oracle (в частности, из семейства Oracle Universal Server) построить эффективную программную инфраструктуру современной ИС. Главная идея, положенная в основу технологии шлюзов Oracle, состоит в возможности простой интеграции СУБД и других изделий ведущих поставщиков ПО в программную среду на основе продуктов Oracle. Иными словами, технология шлюзов Oracle позволяет унифицировать доступ к данным (Oracle SQL) и хранимым процедурам, равно как и сетевое взаимодействие на прикладном уровне (Oracle SQL*Net) в вычислительной системе со сложной неоднородной архитектурой.

В семейство Oracle Open Gateways включены следующие продукты (рис.1).

Transparent Gateway (другое название — **SQL-based gateways**) — группа продуктов (шлюзов), обеспечивающих доступ (посредством использования языка запросов SQL) к данным, хранящимся в отличных от Oracle базах данных. В настоящий момент поддерживается доступ к следующим базам данных: MS SQL Server, Sybase, Rdb, Ingres, Informix, Teradata, RMS, DB2/400, Image/SQL, DB2, SESAM, IBM DRDA, EDA/SQL. Кроме того, в этот набор включен шлюз к базам данных, поддерживающим стандарт ODBC. То есть, если к некоторой СУБД существует драйвер ODBC, то данные из нее могут быть извлечены посредством языка запросов Oracle SQL с использованием продукта Oracle Transparent Gateway ODBC. Собственно, определение назначения продуктов группы Transparent Gateway содержится в самом названии группы — шлюзы обеспечивают прозрачный доступ к «чужим» данным, то есть позволяют работать с ними (с данными) так, как будто это данные в формате Oracle. Далее мы будем для простоты называть продукты этой группы **прозрачными** (*transparent*) шлюзами.

Procedural Gateways (*процедурные шлюзы*) — группа продуктов (шлюзов), обеспечивающих обработку вызовов удаленных процедур, причем удаленные процедуры определены и выполняются в отличной от Oracle программной системе.

Access Managers (*менеджеры доступа*) — группа продуктов (пока включает только один программный продукт под названием Access Manager for AS/400), основным назначением которых является поддержка доступа «чужих» приложений посредством языка запросов SQL к базам данных Oracle.

Replication Services (*сервисы репликации*) — группа продуктов, обеспечивающих репликацию данных из БД Oracle в «чужие» базы данных (равно как и репликацию из «чужих» баз данных в БД Oracle). Для репликации используются прозрачные шлюзы к соответствующим базам данных.

В свете сказанного выше об актуальности задач интеграции баз данных, из всего семейства шлюзов Oracle наибольший интерес представляет группа Transparent Gateways, поэтому настоящий текст целиком посвящен рассмотрению технологических аспектов прозрачных шлюзов. Процедурным шлюзам предполагается посвятить отдельную статью. Помимо прозрачных шлюзов, несколько слов будет сказано о сервисе репликации (в контексте решения задачи репликации данных из «чужих» баз данных в Oracle).

Технологию прозрачных шлюзов Oracle мы рассмотрим на примере одного из пилотных проектов, выполненных в 1998 году сотрудниками Technology Solution Group (технологического подразделения Oracle Russia) для крупной финансовой организации. Мы решили использовать прозрачные шлюзы в качестве каналов для доставки данных из двух источников в хранилище данных, макет которого предполагалось разработать в рамках пилотного проекта. То есть в данном проекте шлюзы самостоятельного интереса, казалось бы, не представляли по вполне понятным причинам — это был тематический DSS-ориентированный проект. Однако работа со шлюзами дала обширный материал для размышлений и обобщений, результатом которых и стала данная статья. По моему мнению, прозрачные шлюзы представляют несомненный практический интерес для проектов корпоративных ИС в качестве средства интеграции баз данных. Теоретически прозрачные шлюзы важны как яркие представители ПО промежуточного слоя.

Перед тем, как перейти к рассмотрению конкретных продуктов и технических решений, необходимо объяснить суть технологии сервер/сервер и ее особенности для разрешения проблем интеграции баз даных.

3.2. Технология сервер/сервер

Рассмотрим типичную для современной организации задачу. Программная инфраструктура включает несколько баз данных различных форматов. На компьютерах-клиентах выполняется приложение, в общем случае запрашивающее данные из нескольких БД.

Возможны следующие варианты решения. Во-первых, для организации доступа приложения к базам данных можно использовать частный интер-

фейс прикладного программирования (для каждой БД — свой собственный). Нетрудно видеть, что такое решение не годится. Детали доступа к БД определены в приложении, что крайне неудобно как при разработке (необходимо помнить особенности работы с каждой из СУБД), так и при эксплуатации.

Доступ приложения к данным должен быть безусловно унифицирован. Необходимо использовать обобщенный API, скрывающий особенности баз данных, к которым выполняется доступ. Таким образом, мы хотим добиться следующего. Во-первых, мы хотели бы иметь доступ к различным базам данных. Во-вторых, для реализации такого доступа мы хотели бы использовать унифицированный API, который скрывал бы особенности баз данных и детали сетевого взаимодействия клиент/сервер. Вопрос в том, где будет размещен компонент, обеспечивающий унификацию доступа к базам данных (далее для краткости будем называть его компонентом доступа к БД) и каким образом он это будет делать.

Данный компонент можно разместить либо на клиенте, либо на сервере. Наиболее очевиден первый вариант решения. На компьютере-клиенте, где выполняется приложение, размещается, помимо компонента доступа к БД, еще и сетевые компоненты (обобщенное название — X Net Client, где X — название конкретной СУБД), обеспечивающие доступ к каждой из баз данных. Так, если приложение работает с базами данных Oracle и Informix, то на компьютер-клиент должны быть установлены продукты Oracle SQL*Net Client и Informix-Net Client.

Недостатки подобного решения очевидны.

Во-первых, компьютер-клиент становится просто неприлично толстым. Мало того, что он должен поддерживать компонент представления и прикладной компонент (классическая RDA-модель). Помимо этого, на него дополнительно возлагается унификация доступа к базам данных (вся синтаксическая и, возможно, семантическая трансляция «обобщенный SQL — диалект SQL») для каждой из баз данных. Перегруженность клиента системной функциональностью никак не укладывается в современные представления об архитектуре клиент/сервер.

Во-вторых, наблюдается множественность сетевых протоколов прикладного уровня, обеспечивающих доступ к базам данных (для Oracle это будет SQL*Net, для Informix — Informix-Net и т.д.). То есть, мы добились унификации интерфейса прикладного программирования и теперь приложение обращается к любой базе данных на одном и том же языке — «обобщенном SQL». Однако сетевые протоколы не унифицированы, что порождает дополнительные проблемы. Так, всегда удобнее защищать (например, средствами шифрования) один протокол, нежели множество, то же можно сказать и об администрировании. Дело в том, что ныне компонент X Net у большинства поставщиков СУБД представляет собой развитую сетевую подсистему со специализированными сервисами (сервисом имен, например) и ее нормальное администрирование требует специальных знаний и навыков.

Примером интеграции баз данных на стороне клиента является прямая реализация (то есть клиент/сервер) подхода на основе стандарта ODBC[2]. Напомним, что ODBC позволяет программам, работающим в среде Windows[3], взаимодействовать (посредством операторов языка SQL) с различными СУБД, как с персональными, так и с многопользовательскими, функционирующими в различных операционных системах. Фактически, интерфейс ODBC универсальным образом отделяет чисто прикладную, содержательную сторону приложений (обработка электронных таблиц, статистический анализ, деловая графика) от собственно обработки и обмена данными с СУБД. Основная цель ODBC — сделать взаимодействие приложения и СУБД прозрачным, не зависящим от класса и особенностей используемой СУБД (мобильным с точки зрения используемой СУБД).

Интерфейс ODBC обеспечивает взаимную совместимость серверных и клиентских компонентов доступа к данным. Для реализации унифицированного доступа к различным СУБД было введено понятие драйвера ODBC (представляющего собой динамически загружаемую библиотеку в терминологии Windows).

ODBC-архитектура (рис.2) включает четыре компонента:

- приложение;
- менеджер драйверов (Microsoft Driver Manager);
- драйверы к каждой из СУБД;
- источники данных.

Приложение вызывает функции ODBC для выполнения операторов SQL, получает и интерпретирует результаты; менеджер драйверов динамически загружает ODBC-драйверы, когда этого требует приложение; ODBC-драйверы обрабатывают вызовы функций ODBC, передают операторы SQL СУБД и возвращают результат в приложение. Собственно, приложение черпает данные из источника данных — объекта, скрывающего от приложения детали СУБД и сетевого интерфейса, расположение и полное имя базы данных и т.д.

[2] Не стоит жестко связывать интеграцию баз данных на стороне клиента и доступ к БД на основе ODBC. Дело в том, что ODBC можно использовать и на стороне сервера с использованием продукта Oracle Transparent Gateway for ODBC (TGODBC), о котором также будет рассказано в статье.

[3] С ODBC можно работать также и в ОС UNIX.

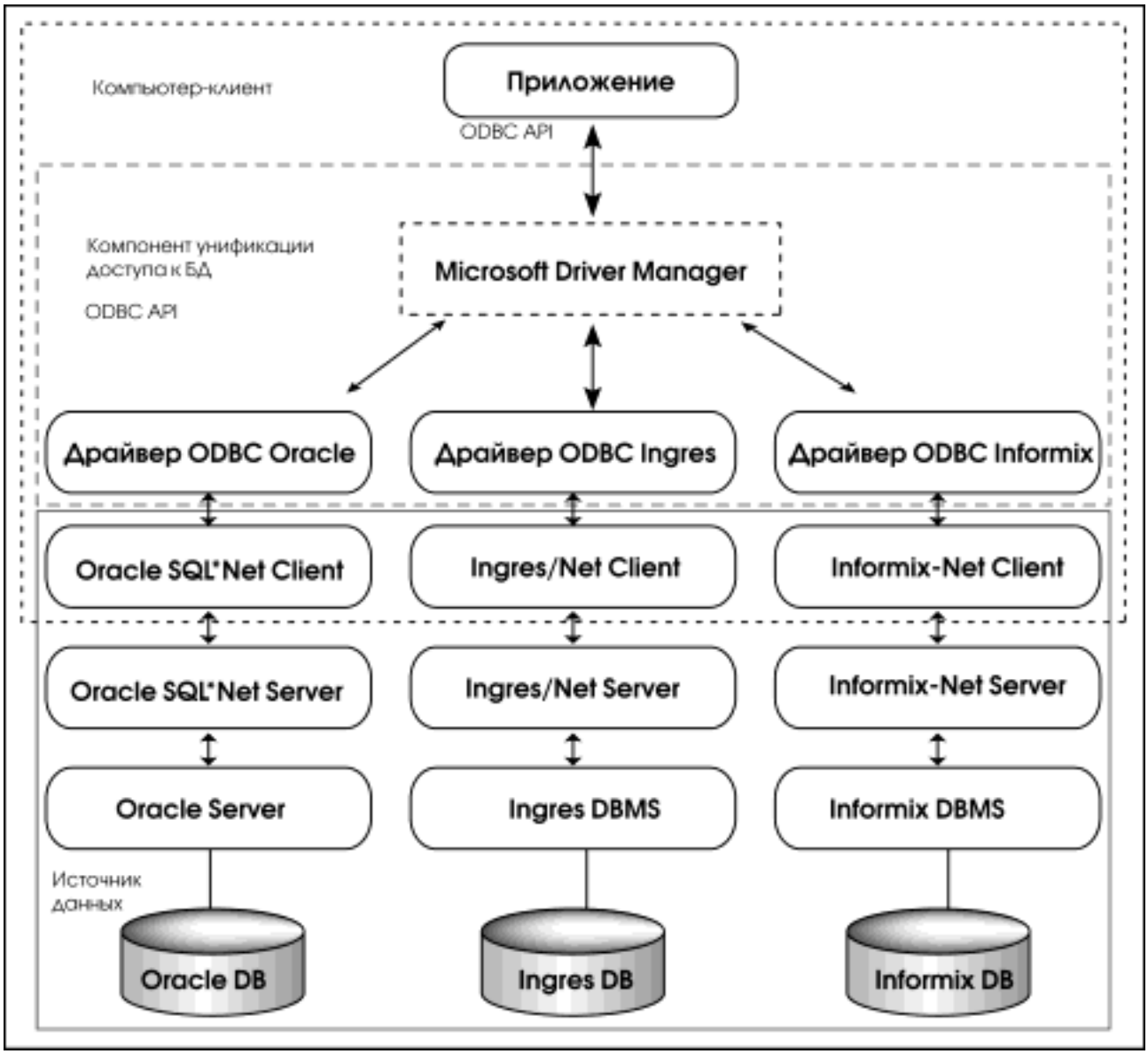


Рис. 2. Архитектура ODBC.

Другим возможным решением является размещение компонента доступа к БД на стороне сервера (рис. 3). Приложение на компьютере-клиенте взаимодействует с сервером посредством Oracle SQL. Для организации сетевого взаимодействия клиента и сервера используется SQL*Net (Net8). Клиент обращается к любым базам данных, доступным в рамках корпоративной сети, как к базам данных Oracle. Обращение к базам данных других форматов полностью прозрачно для клиента, то есть клиент ничего не знает о том, на каком из компьютеров-серверов расположена искомая база данных и в каком она формате. Однако обращение к базе данных происходит не напрямую (как в случае с ODBC), а опосредованно, через сервер Oracle (как это сделано технически, будет рассказано ниже). Именно этот факт дает основание говорить о том, что в данном случае имеет место технология сервер/сервер. То есть клиент обращается к Oracle Server, а тот взаимодействует с сервером другой базы данных.

Таким образом, интеграция баз данных на уровне сервера предполагает:

- Выделение некоторого компьютера, который выполнял бы роль интеграционного сервера баз данных[4];
- Единообразный способ доступа приложения на компьютере-клиенте СУБД к интеграционному серверу БД;
- Использование специализированных средств (шлюзов) для организации доступа сервера Oracle к базам данных других форматов.

[4] *Ввиду распространенности и популярности логично было бы установить на него СУБД Oracle.*

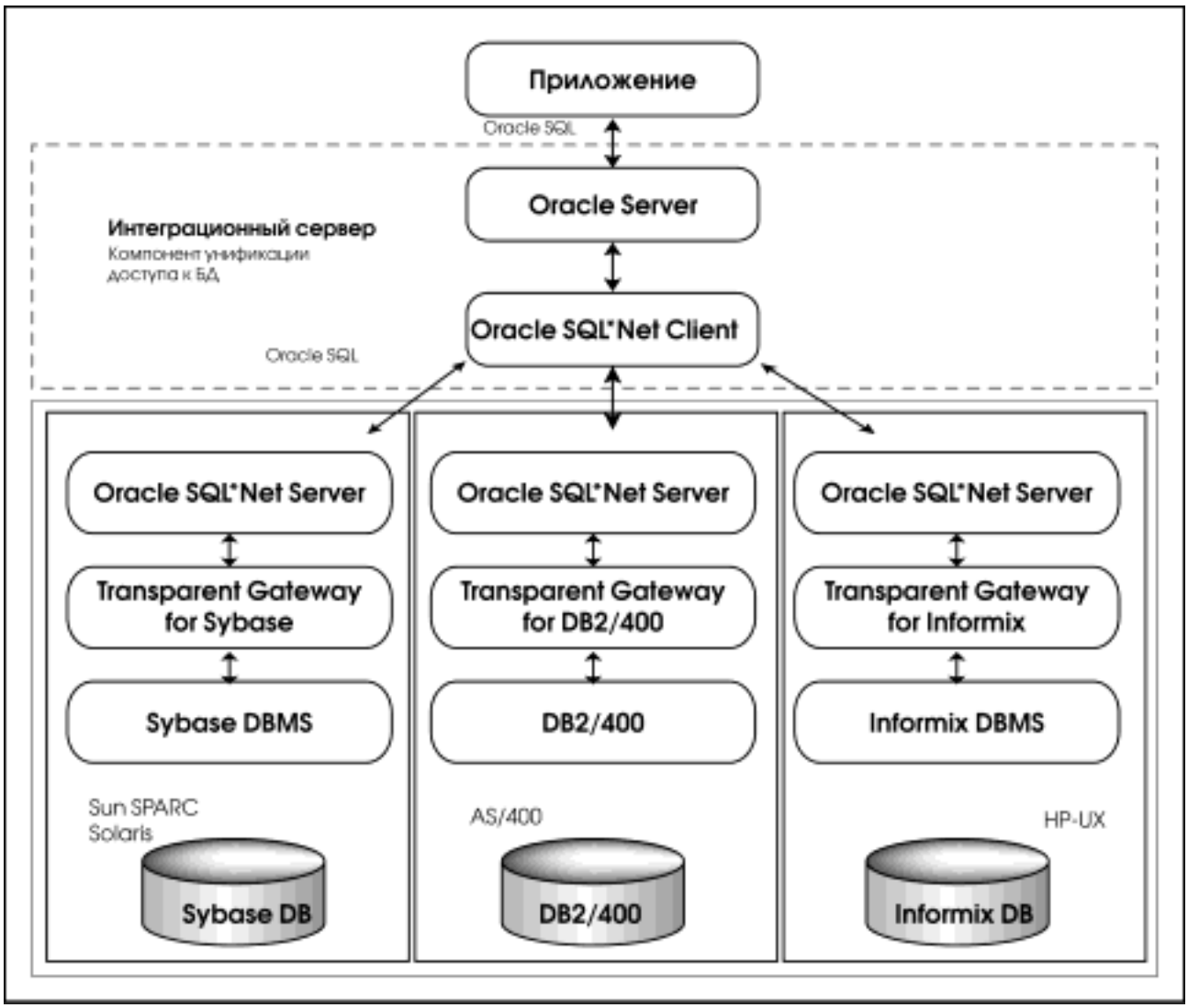


Рис. 3. Архитектура сервер/сервер.

Отметим, что два подхода к интеграции баз данных (на стороне клиента и на стороне сервера) выделены в статье исключительно для объяснения специфики работы продуктов группы Oracle Transparent Gateway. Вообще говоря, при подготовке статьи не ставилась задача детального анализа достоинств и недостатков решений по интеграции баз данных (ODBC, EDA/SQL, IBM DRDA).

3.3. Архитектура шлюза

Как было отмечено выше, архитектура шлюза в данной статье объясняется на примере пилотного проекта финансовой организации. В организации существуют и развиваются два центра данных. Один из них представляет собой систему OLTP на базе компьютера IBM AS/400, на котором функционирует СУБД DB2/400. Большая часть актуальных данных финансовой организации хранится в базах данных DB2/400 (версия СУБД DB2 производства IBM, реализованная на компьютерах AS/400). Другой центр обработки данных опирается на сервер на платформе Intel под управлением ОС Windows NT с СУБД MS SQL Server (версия 6.5). Собственно, задача пилотного проекта заключалась в отработке технологической схемы «источники данных (БД MS SQL Server, DB2/400) —> сервер очистки и согласования данных —> хранилище данных —> витрины данных —> приложения OLAP» на основе продуктов и технологий Oracle.

Далее мы будем рассматривать фрагмент информационной системы финансовой организации, в рамках которого выполнялись все работы. Системно-техническая инфраструктура была представлена следующими компьютерами:

- AS/400 под управлением OS/400 (СУБД DB2/400);
- Сервер Intel/Pentium под управлением Windows NT (СУБД MS SQL Server);
- IBM RS/6000 под управлением AIX (интеграционный сервер, СУБД Oracle8);
- Компьютер-клиент Intel/Pentium под управлением Windows NT.

Компьютеры были связаны в сеть Token Ring, поддерживался протокол TCP/IP.

Программная инфраструктура фрагмента ИС была образована следующими компонентами:

- Приложение (выполняется на компьютере-клиенте);
- Сервер Oracle (Oracle7 или Oracle8, функционирует на интеграционном сервере);
- Шлюзы Transparent Gateway for DB2/400 (TGDB2400), Transparent Gateway for Microsoft SQL Server (TGMSQL);
- SQL*Net или Net8.

Подобная программная инфраструктура является типовой для систем, обеспечивающих интеграцию баз данных посредством прозрачных шлюзов. Разумеется, сами шлюзы будут зависеть от того, к каким базам данных они обеспечивают доступ.

Функционально прозрачный шлюз представляет собой своего рода «интеллектуальный программный канал» для доступа к базам данных иных, нежели Oracle, форматов. Не следует думать, что шлюзу известен «формат» этих БД и обращение к данным идет «в обход» СУБД, то есть напрямую. Напротив, для извлечения данных из искомой базы шлюз обращается непосредственно к СУБД (например, посредством SQL, если речь идет о реляционной СУБД), которая и обеспечивает требуемый доступ. Назовем для краткости эту пару (искомая база данных и СУБД, обеспечивающая доступ к ней) **целевой системой** (*target system*).

Схема функционирования прозрачного шлюза представлена на рис. 4.

Последовательность действий при доступе к целевой системе такова:

1. Приложение направляет запрос серверу Oracle.
2. (Этот шаг не является обязательным.) Возможны ситуации, когда шлюз должен отыскать входные характеристики пользователя в словаре данных целевой системы и разрешить ему доступ к ней.
3. Сервер Oracle направляет запрос шлюзу.
4. Шлюз взаимодействует с целевой системой, которая разрешает доступ пользователя к объекту целевой системы.

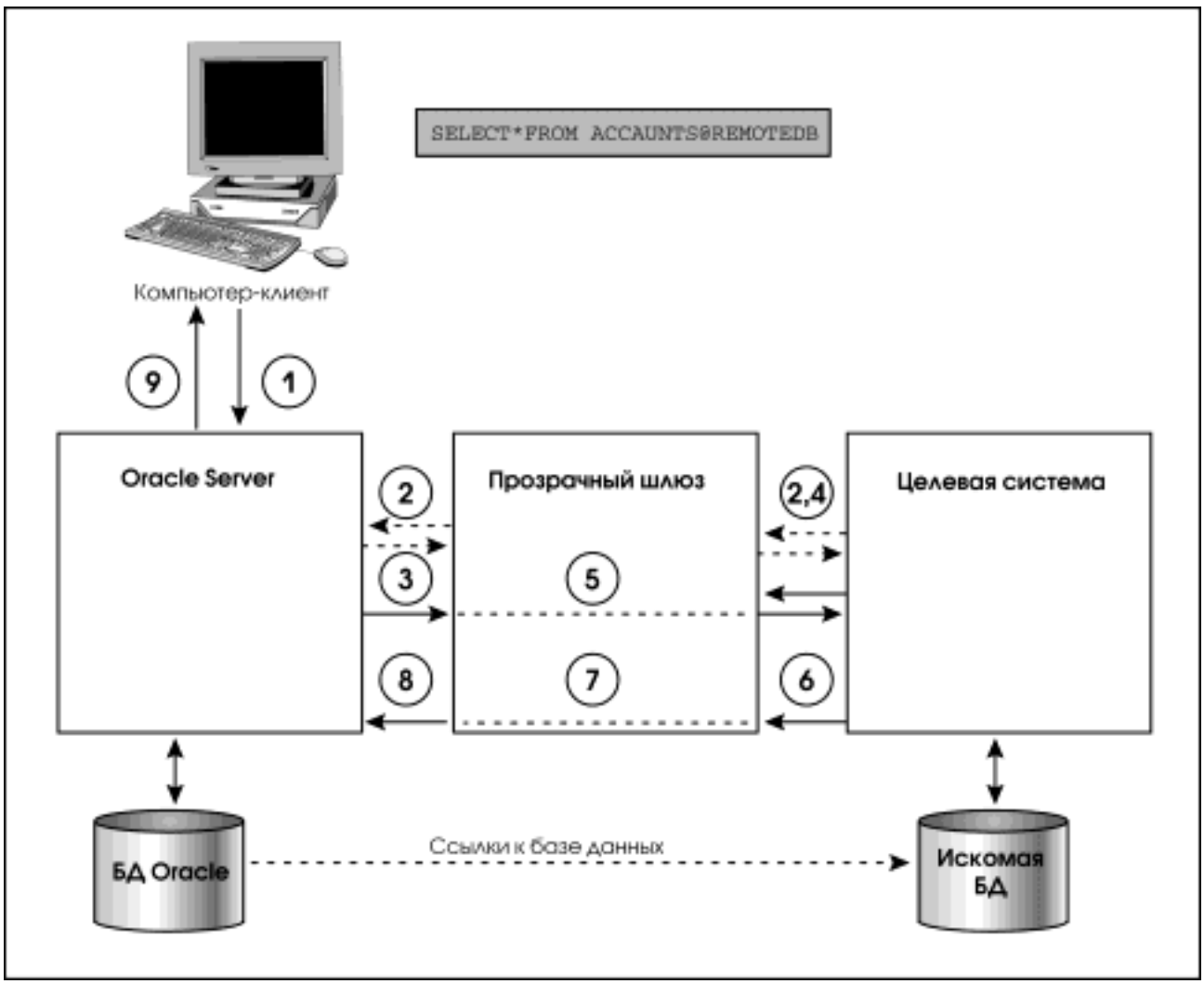


Рис. 4. Схема функционирования прозрачного шлюза.

5. Шлюз транслирует операторы Oracle SQL в SQL целевой системы.
6. Шлюз получает данные из целевой системы.
7. Шлюз конвертирует полученные данные в формат Oracle.
8. Шлюз возвращает результат запроса серверу Oracle.
9. Сервер Oracle возвращает результат запроса приложению.

С помощью шлюза можно не только работать с данными посредством запросов SQL, но и вызывать удаленные хранимые процедуры, принадлежащие целевой системе. Для этого используются прозрачные шлюзы с процедурными возможностями.

Необходимость использования таких шлюзов возникает в том случае, когда помимо работы с данными посредством SQL-запросов требуется вызывать удаленные хранимые процедуры целевой системы.

Пусть, например, мы используем СУБД DB/2 в целевой системе и для реализации некоторой прикладной функциональности был использован механизм хранимых процедур. Пусть, далее, приложению, работающему с БД Oracle, порой необходимы данные из DB/2. Это же приложение иногда должно обращаться к хранимым процедурам целевой системы.

Схема работы прозрачного шлюза с процедурными возможностями представлена на рис. 5.

Последовательность действий такова:

1. Приложение вызывает хранимую процедуру SALARY (считая, что она выполняется сервером Oracle, тогда как на деле это процедура целевой системы).
2. (Этот шаг не является обязательным.) Возможны ситуации, когда шлюз должен отыскать входные характеристики пользователя в словаре данных целевой системы и разрешить ему доступ к ней.

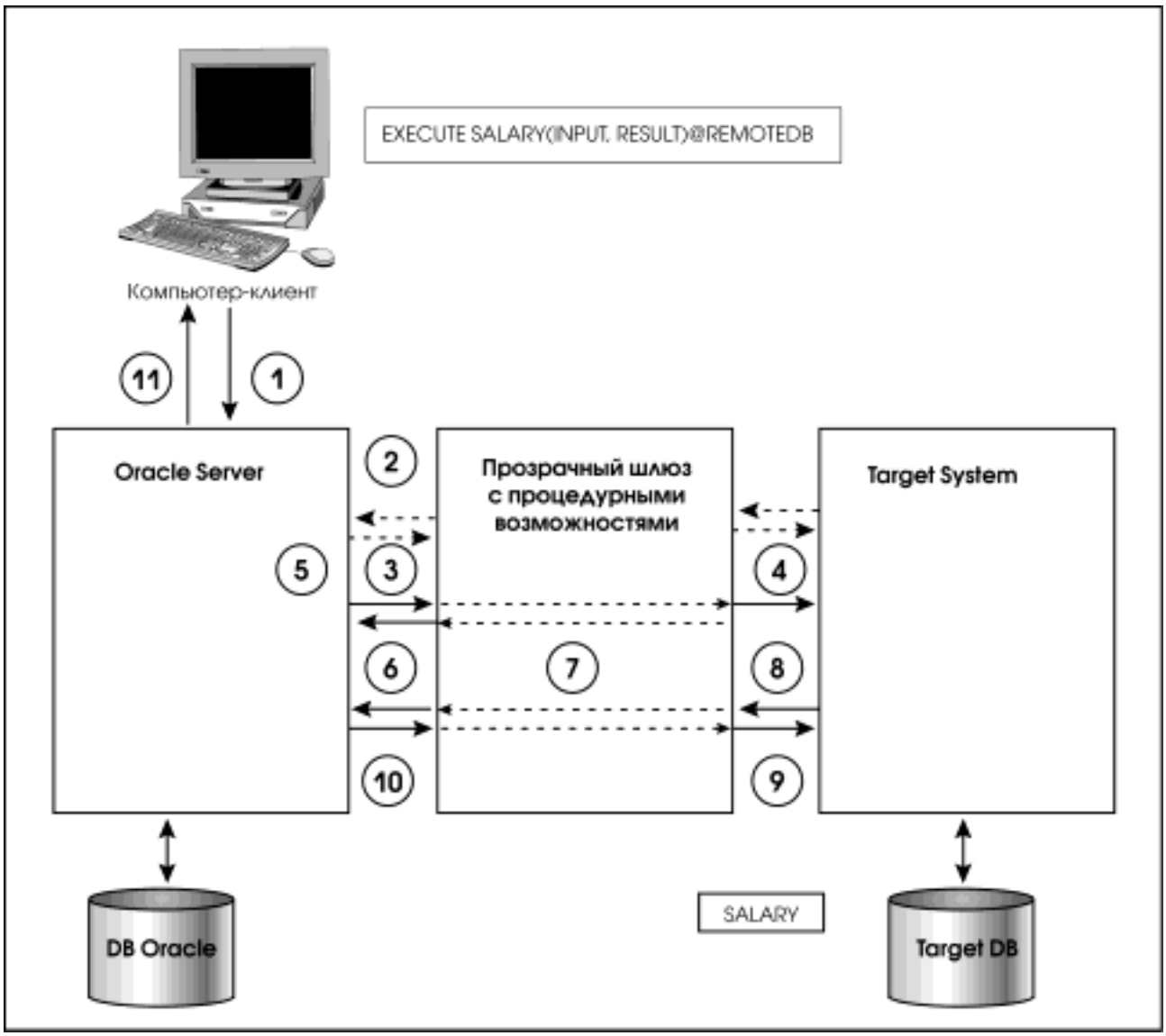


Рис. 5. Схема функционирования прозрачного шлюза с процедурными возможностями.

3. Сервер Oracle запрашивает описание хранимой процедуры SALARY.
4. Запрос через шлюз направляется целевой системе.
5. Целевая система возвращает через шлюз ответ (описание хранимой процедуры SALARY).
6. Сервер Oracle направляет запрос на выполнение хранимой процедуры SALARY.
7. Запрос через шлюз направляется целевой системе.
8. Выполняется хранимая процедура SALARY.
9. Шлюз получает код возврата процедуры и результат, конвертирует данные в формат Oracle.
10. Шлюз возвращает результат серверу Oracle.
11. Сервер Oracle возвращает результат процедуры приложению.

С точки зрения размещения на компьютерах-серверах, прозрачные шлюзы могут быть сконфигурированы как:

Стационарные (*host-based*)

Удаленные (*remote-based*)

В системе со стационарным шлюзом последний устанавливается на тот компьютер, где функционирует целевая система. Конфигурация с удаленным шлюзом предполагает размещение шлюза не на компьютере с целевой системой (например, на интеграционном сервере). Особо отметим, что речь идет именно о возможном варианте конфигурации шлюза (а вовсе не о его типе). Так, продукт TGMSQL может быть сконфигурирован и как стационарный, и как удаленный, а вот TGDB2400 можно установить только в стационарном варианте. В обоих вариантах конфигурации через шлюз можно получить доступ одновременно к нескольким источникам данных.

Архитектура прозрачного шлюза в стационарной конфигурации (реальная конфигурация ИС организации) представлена на рис. 6. На AS/400, наряду со шлюзом, функционирует также система оперативной обработки транзакций, к которой возможно одновременное обращение нескольких сотен пользователей (на рисунке она не показана). Стационарное размещение шлюза в целом не сказывается на производительности OLTP-системы (шлюз оформлен как несколько фоновых процессов), но, конечно, требует дополнительных системных ресурсов. Отметим, что сетевой протокол прикладного уровня для взаимодействия как «компьютер-клиент/интеграционный сервер», так и «интеграционный сервер/компьютер целевой системы» унифицирован и представляет собой Oracle SQL*Net[5].

Рассмотрим теперь средства Oracle, позволяющие обращаться к удаленным базам данных. Имеются в виду **ссылки к базам данных** (*database link*). Используя этот механизм, Oracle упрощает взаимодействие между базами данных в распределенной системе. Ссылка к БД представляет собой объект базы данных Oracle, который задает местонахождение удаленной БД и параметры доступа к ней.

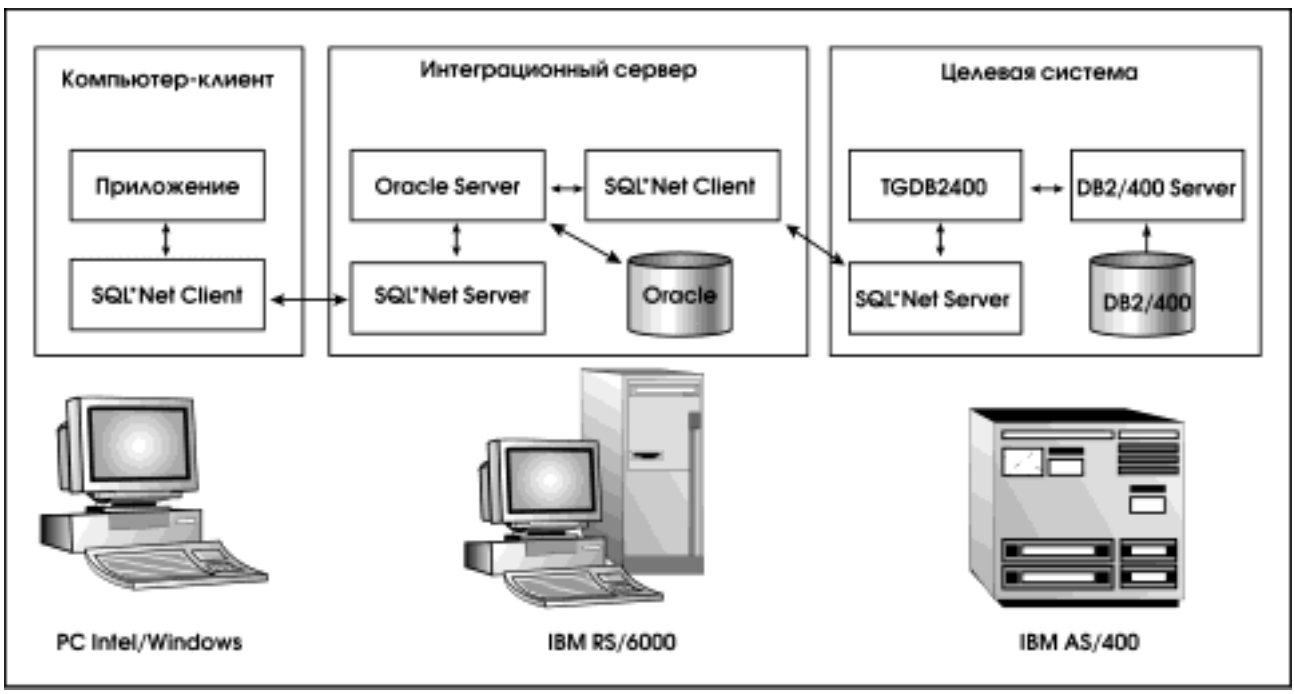


Рис. 6. Стационарная конфигурация прозрачного шлюза.

[5] *На наш взгляд, унифицированный (то есть распространенный по всей сети) сетевой протокол прикладного уровня является важнейшей характеристикой грамотно спроектированной информационной системы.*

Поясним детали на примере. Пусть в целевой системе (в конфигурации, представленной на рис. 6) существует база данных с именем **accounts**. Необходимо обеспечить доступ к ней (конкретно, к таблице **clients**) с компьютера-клиента. Для этого надо выполнить следующие шаги (предполагается, что все изображенные на рис. 6 программные компоненты уже установлены).

1. Обеспечить подключение интеграционного сервера к компьютеру целевой системы, используя протокол TCP/IP[6], и сконфигурировать SQL*Net для AS/400 (по завершении этой процедуры должен сработать **ping** к AS/400). Эта задача является сетевой, по сути не имеет отношения к рассматриваемой тематике, решается достаточно просто и по этой причине ее детали поясняться не будут.
2. Подключиться с интеграционного компьютера к AS/400 по SQL*Net (в конце этой процедуры должна успешно выполниться утилита **tnsping**, предназначенная для проверки связи по SQL*Net). Рассмотрим этот этап более подробно, так как мы будем ссылаться на него далее.

В файл TNSNAMES.ORA на интеграционном сервере необходимо добавить еще один дескриптор:

```
AS400 = (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)
        (PORT = 1521)
        (HOST = host_name)
      )
      (CONNECT_DATA = (SID = csi_name))
    )
```

где:

- **TCP** — протокол транспортного уровня для связи интеграционного сервера и AS/400;
- **1521** — номер порта, используемого процессом прослушивания (listener) на AS/400;
- **host_name** — символическое имя AS/400 в сети или, возможно, полный IP-адрес.
- **csi_name** – SID параметр, идентифицирующий шлюз TGDB2400 как сервер Oracle. Он используется на AS/400 при старте процесса прослушивания. Этот же параметр идентифицирует одну из четырех CL-программ (заданий), запускающих шлюз в том или ином варианте. В стандартных ситуациях в качестве значения **csi_name** следует использовать **ORAGATE**.

3. Правильно установить параметры инициализации TGDB2400. Обращаем на это особое внимание, так как в документации [5] описания этого этапа почему-то нет, тогда как несколько параметров инициализации являются крайне важными и должны быть установлены в определенные значения (в противном случае шлюз не заработает). Подробное описание параметров инициализации шлюза TGDB2400 можно найти в [6], здесь же будет сказано о наиболее важных.

Параметр **DB_DOMAIN** задает уникальный (в сети Oracle SQL*Net) адрес шлюза. По умолчанию принимает значение **WORLD**, что и было использовано в нашем варианте конфигурации.

Параметр **DB_NAME** является обязательным. Шлюз не может стартовать, если значение этого параметра не определено. Для всех продуктов из группы шлюзов (в том числе и для процедурных) значение этого параметра должно совпадать с SID, заданным в TNSNAMES.ORA на интеграционном компьютере. Так, если значение SID установлено **ORAGATE**, то и параметр **DB_NAME** должен также принять значение **ORAGATE**.

После того, как параметры установлены, с консоли AS/400 необходимо запустить процесс прослушивания SQL*Net.

4. Создать ссылку к удаленной базе данных на AS/400. Ввиду того, что ссылки к базам данных — это один из основных механизмов Oracle, обеспечивающий обработку распределенных данных, остановимся на нем подробнее.

3.4. Ссылки к базам данных

Ссылку к удаленной БД необходимо создать, используя следующий оператор SQL:

```
CREATE PUBLIC DATABASE LINK dblink
CONNECT TO userid IDENTIFIED BY password
USING 'tns_service_name'
```

где

- **dblink** — полное имя удаленной базы данных
- **userid**, **password** — параметры пользователя, от имени которого осуществляется доступ к удаленной базе данных (в нашем примере — к базе данных DB2/400 под названием accounts). Этот пользователь должен быть создан администратором баз данных DB2/400 и ему должны быть предоставлены соответствующие привилегии по доступу к данным.
- **tns_service_name** — должно соответствовать имени дескриптора в файле TNSNAMES.ORA.

[6] *В случае с AS/400 возможно подключение и посредством протокола APPC/LU6.2, однако в пилотном проекте был применен TCP/IP.*

Таким образом, если в базе данных **accounts** определен пользователь **gtuser** с паролем **dbtest**, то ссылка к ней может быть создана следующим оператором SQL[7]

```
CREATE PUBLIC DATABASE LINK remotedb
CONNECT TO gtuser IDENTIFIED BY dbtest
USING 'AS/400'
```

Единожды создав ссылку к удаленной базе данных, можно многократно пользоваться ею для доступа к БД. Пусть, например, необходимо выбрать все строки из таблицы **clients** базы данных **accounts** (она нам доступна теперь через ссылку **remotedb**):

```
SELECT * FROM clients.accounts@remotedb
```

Несколько слов о разграничении доступа к удаленной БД. Как видно из оператора CREATE DATABASE LINK, созданная ссылка явным образом идентифицирует пользователя, от имени которого через эту ссылку будет осуществляться доступ к удаленным базам данных целевой системы. Важно понимать, что этот пользователь не является пользователем Oracle — он задан и существует как специальный пользователь целевой системы, от имени которого к ее базам данных будет выполнен доступ из других систем. Поэтому DBA целевой системы должен сообщить его входные параметры DBA Oracle до создания ссылки. Если же в операторе CREATE DATABASE LINK параметры пользователя не указаны, то используется текущее имя пользователя Oracle и в этом случае необходимо, чтобы оно совпадало с именем пользователя целевой системы.

Определение ссылок к базам данных хранится в словаре Oracle. Представление (view) словаря с именем USER_DB_LINKS содержит определение ссылок к БД, созданных текущим пользователем, тогда как представление ALL_DB_LINKS позволяет увидеть характеристики всех ссылок (если на то администратором БД предоставлены соответствующие права).

Так как созданная нами ссылка является **общедоступной** (*public*), ее может использовать любой пользователь Oracle на интеграционном сервере для доступа к БД **accounts**. Можно создать и **частную** (*private*) ссылку к удаленной базе данных. С точки зрения безопасности правильно было бы использовать только частные ссылки к удаленной БД и выполнять сам доступ к ней только от имени специально выделенного пользователя.

Ниже будет рассмотрено несколько способов доступа к удаленной БД посредством ссылок. Отметим интересный способ именования удаленных баз данных, который обеспечивается в Oracle с помощью **синонимов** (*synonym*). Синоним — это новое упрощенное имя таблицы базы данных (в том числе удаленной). Как мы видим из приведенного выше запроса, полная прозрачность обращения к удаленной базе данных пока не достигнута, так как мы вынуждены указывать в явном виде ссылку к ней. Для достижения цели необходимо создать синоним **clients**:

```
CREATE PUBLIC SYNONYM clients
FOR clients.accounts@remotedb
```

Теперь приведенный выше запрос можно сформулировать проще:

```
SELECT * FROM clients
```

Отметим, что был создан общедоступный синоним. Это означает, что любой пользователь Oracle, зарегистрированный на интеграционном сервере, может использовать синоним **clients** для доступа к удаленной базе данных.

3.5. Кодировки

В неоднородной информационной среде неизбежно возникают проблемы с кодировками. Так, в данном случае мы имели дело с вычислительной установкой AS/400, для которой стандартной кодировкой является EBCDIC, в то время как на UNIX-компьютерах используется кодировка ASCII. Шлюз выполняет автоматическое преобразование кодировок EBCDIC-ASCII. Сложнее было с символами русского языка.

Известно, что Oracle поддерживает стандарт NLS. Группа продуктов Oracle Transparent Gateways не является исключением. Правильно установив параметры NLS на интеграционном сервере и параметры NLS шлюза, можно добиться корректной обработки символов русского языка. На интеграционном сервере мы использовали стандартную для Oracle на UNIX-компьютерах кодировку ISO8859-5. Параметры NLS шлюза TGDB2400 устанавливаются с консоли AS/400 командой CHGORATUN, параметры Oracle SQL*Net — командой CHGORANET. Синтаксис для спецификации языковых параметров в обоих случаях одинаков и выглядит следующим образом:

```
language [_territory.character_set]
```

Значение параметра character_set должно быть выбрано из таблицы Supported Character Set. Наиболее подходящим казался набор символов Western European (код WE8EBCDIC37). Одна-

[7] *При практической работе рекомендуется от имени системного администратора запустить на интеграционном сервере SQL*Plus и все дальнейшие операции выполнять с его помощью.*

ко с ним шлюз TGDB2400 выполнял преобразование кодов неверно. По рекомендации коллег мы установили значение CL8EBCDIC1025 — кодировка русского языка, используемая Oracle на мэйнфремах, несмотря на то, что этой кодировки в таблице указано не было. После этого шлюз заработал нормально. Было необходимо, чтобы данные в целевой системе имели кодировку CL8EBCDIC1025.

3.6. Доставка данных на интеграционный сервер

После того, как мы установили и наладили шлюзы к DB2/400 и MS SQL Server, необходимо заняться собственно передачей данных. Существует несколько вариантов решения.

Можно создать на интеграционном сервере временную таблицу, выбрав в нее все данные из исходной (в рамках одного оператора SQL), например:

```
CREATE TABLE oracle_table AS
SELECT * FROM
db2_table.db2_database@dblink
```

Для заполнения временной таблицы можно использовать оператор INSERT:

```
INSERT INTO oracle_table
SELECT * FROM
db2_table.db2_database@dblink
```

Однако мы хотели бы обеспечить периодическую «подкачку» данных на интеграционный сервер из целевой системы. Наиболее очевидным подходом является использование одного из простейших механизмов репликации Oracle — механизма **моментальных копий** или **снимков** (*snapshot*). Ниже мы будем говорить только о необновляемых снимках.

Следуя определению Oracle, снимок представляет собой предназначенную только для чтения копию **главной таблицы** (*master table*), расположенную на удаленном узле. К снимку можно обращаться с запросами на выборку, но нельзя его обновлять: обновляется главная таблица, а затем изменения автоматически переносятся на снимок.

Рассмотрим пример, где:

cusomers — имя снимка;

clients — имя главной таблицы;

accounts — имя исходной базы данных;

remotedb — ссылка к исходной базе данных.

```
CREATE SNAPSHOT customers
    PCTFREE 5 PCTUSED 60
    TABLESPACE users
    STORAGE (INITIAL 50K NEXT 50K)
    REFRESH COMPLETE NEXT SYSDATE + 1
AS
    SELECT * FROM clients.accounts@remotedb
```

Данный оператор создает снимок с таблицы удаленной базы данных. REFRESH специфицирует режим **обновления**[8] (*refresh*) снимка. Значение COMPLETE необходимо задавать, когда предполагается полное обновление снимка, то есть данные из исходной таблицы целиком переносятся в снимок. Значение FAST соответствует так называемому «быстрому» обновлению, когда в снимок помещаются не целиком все данные из исходной таблицы, а только изменения, имевшие место с момента последнего обновления. Возможен также режим FORCE, когда Oracle определяет по ситуации, возможно ли применить режим быстрого обновления и, если да, использует его; в противном случае действует режим COMPLETE.

NEXT задает интервал между автоматическими обновлениями. Значение **SYSDATE+1** соответствует обновлению с интервалом времени в сутки. Первое обновление производится в момент создания снимка.

Смысл созданного снимка таков. На интеграционном сервере в базе данных Oracle под именем **customers** создан «образ» таблицы **clients** в удаленной базе данных **accounts**. В момент создания в него помещены все данные из таблицы **clients**. После этого OLTP-система, работающая с базой данных DB2/400, может вносить в таблицу **clients** любые изменения. Пройдут сутки, и сервер Oracle скопирует через шлюз TGDB2400 целиком всю таблицу **clients** в **customers** и будет повторять эту процедуру и далее с интервалом в сутки. Разумеется, интервал времени между копированиями может быть установлен так, как это необходимо, непосредственно при создании снимка либо используя оператор ALTER SNAPSHOT.

Таким образом, мы добились того, что данные периодически передаются из исходной таблицы на интеграционный сервер. Приложение может читать данные из снимка, считая, что, во-первых, оно имеет дело с самой свежей версией данных, полученных на это время, и, во-вторых, что эти данные находятся в БД Oracle.

Однако этого недостаточно. Хотелось бы, чтобы из целевой системы на интеграционный сервер передавались не все данные, а только изменения в

[8] *Здесь речь идет об автоматическом обновлении снимка, в отличие от обновления (update), выполняемого приложением над таблицами. Как было сказано выше, эту операцию над необновляемыми снимками выполнить нельзя.*

данных, накопленные с момента последнего обновления снимка. Цель вполне понятна — уменьшить сетевой трафик. Передача изменений по сравнению с передачей всего массива данных позволяет существенно разгрузить сеть, так как в целом объем передаваемых по сети таблиц может быть очень велик.

Казалось бы, Oracle позволяет создать моментальный снимок с передачей только изменений (режим **FAST** REFRESH). Однако не все так просто. Дело в том, что этот режим можно использовать только в том случае, когда в целевой системе определен и используется некоторый механизм накопления изменений в главной таблице и их отображения в снимке.

Предположим первоначально, что исходная БД — это база данных Oracle. Тогда все выглядит просто. В целевой системе создается **журнал снимков**[9] (*snapshot log*). Это — таблица, ассоциированная с главной таблицей. Сервер Oracle (в составе целевой системы) сохраняет изменения, выполненные в главной таблице, в журнале снимков, и затем использует его для обновления снимков, ассоциированных с данной главной таблицей. То есть ответственность за накопление и передачу изменений берет на себя сервер Oracle. К сожалению, в данном пилотном проекте этот подход использовать мы не могли, так как целевая система представляла собой DB2/400.

3.7. Репликация данных из других систем

Однако и в этой ситуации было найдено решение. Выше мы уже говорили, что семейство Oracle Open Gateways включает группу продуктов Replication Service. Продукт Oracle Replication Service for Data Propogator (ORSDP) позволяет решить поставленную задачу, то есть реплицировать только изменения в исходных таблицах целевой системы (DB2/MVS и DB2/400), но не все содержимое таблиц целиком. Конфигурация соответствующих программных средств представлена на рис. 7.

Конфигурация включает следующие компоненты:

- Oracle Replication Services for DataPropogator;
- Oracle Server (на интеграционном сервере);
- Oracle Transparent Gateway for DB2/400 (на AS/400);
- IBM DataPropogator Relational Capture/400
- DB2/400.

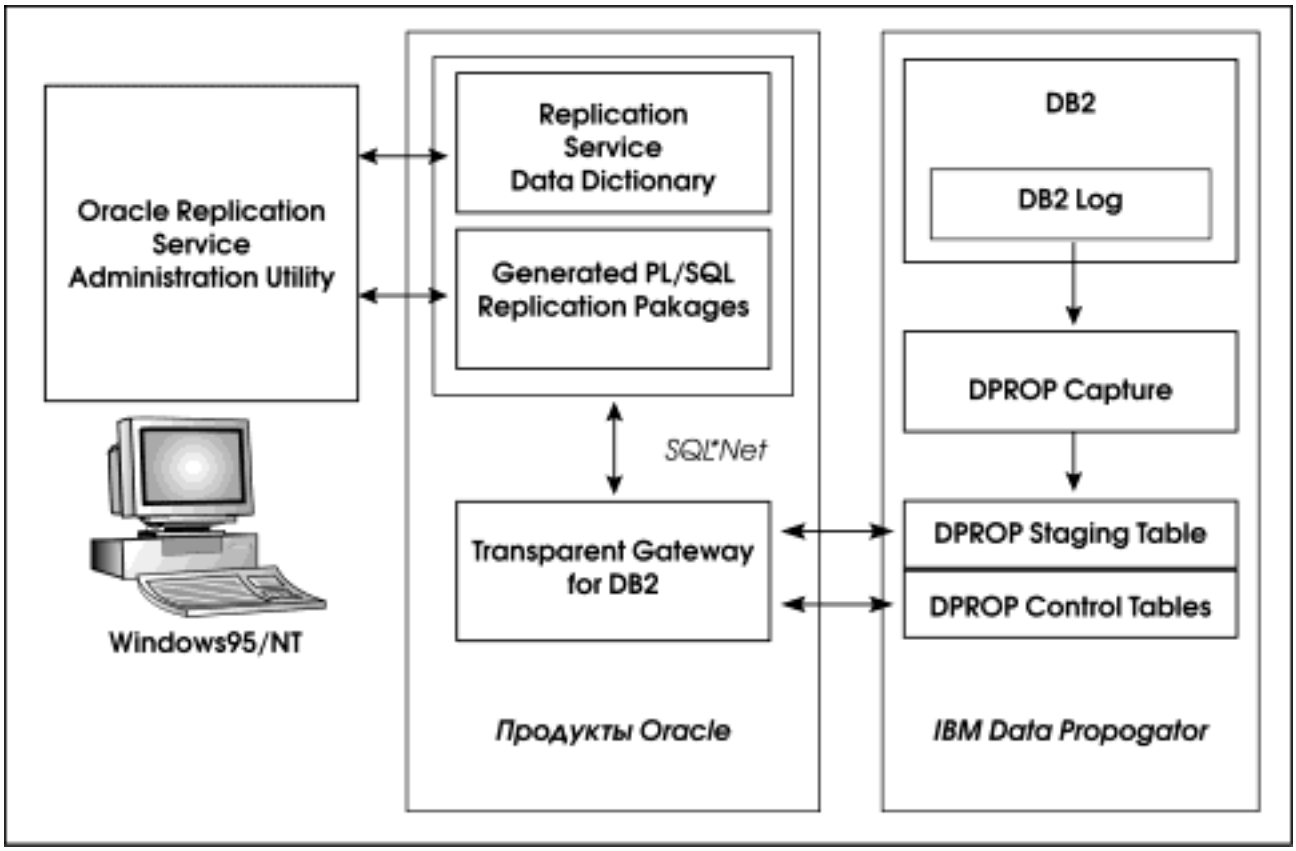


Рис. 7. Схема инкрементальной репликации данных из DB2 в Oracle.

[9] *Создается оператором CREATE SNAPSHOT LOG*

Какие действия необходимо предпринять, чтобы изменения, выполненные над таблицами DB2/400, отразить на таблицы Oracle? Во-первых, необходимо захватывать изменения в таблицах DB2 и где-то их накапливать. Во-вторых, необходимо накопленные изменения периодически применять к таблицам Oracle.

Первое выполняет продукт IBM DataPropogator Relational Capture (DPROP). Конкретно, DPROP сканирует журналы DB2 и помещает захваченные изменения в таблицы DB2/400. Вполне естественно, что эту часть работы осуществляет продукт IBM, понимающий формат журналов DB2/400.

Второе ложится на плечи Oracle Replication Services. Продукт поставляется для ОС Windows NT и Windows95 и устанавливается на отдельную рабочую станцию, которая берет на себя функции консоли для администрирования процесса репликации. Сам процесс переноса изменений в таблицы Oracle выполняется средствами PL/SQL (язык для разработки хранимых процедур Oracle), функционирующими на интеграционном сервере. Они извлекают (*pull*) изменения из таблиц DB2/400 и применяют их к результирующим таблицам Oracle. Oracle Replication Service также включает утилиты для администрирования окружения DPROP. Отметим, что с помощью Oracle Replication Services for DataPropogator можно реплицировать данные и в обратном направлении, то есть из Oracle в DB2.

Очевидно, почему Oracle Replication Services был разработан в первую очередь для СУБД DB2. Она очень распространена на мэйнфремах, а для AS/400 DB2 — это вообще стандарт (собственно, для AS/400 других баз данных, кроме DB2/400, нет). Видимо, в дальнейшем технология Oracle Replication Service будет распространена на другие системы.

3.8. Transparent Gateway for Microsoft SQL Server

В отличие от TGDB2400, который может быть установлен только в стационарной конфигурации, прозрачный шлюз к Microsoft SQL Server может работать и как стационарный, и как удаленный.

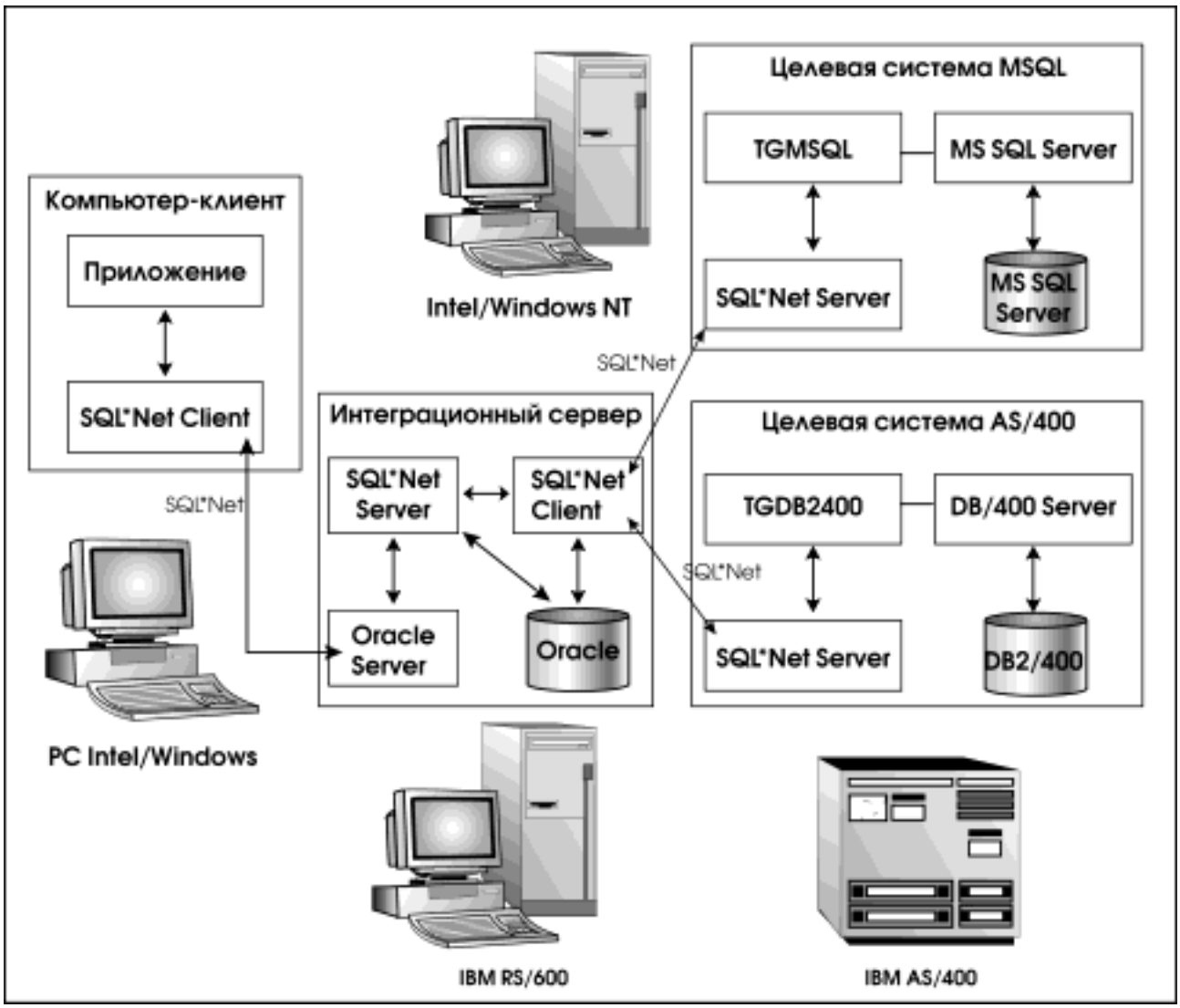


Рис. 8. Полная конфигурация пилотного проекта.

Однако предпочтение все-таки стоит отдать стационарному варианту конфигурации. Причина была указана выше — целесообразно унифицировать сетевой протокол прикладного уровня, выбрав SQL*Net, что достигается в стационарном варианте и было использовано в пилотном проекте (полная конфигурация пилотного проекта представлена на рис. 8).

Для организации доступа к Microsoft SQL Server и доставки данных на интеграционный сервер используются описанные выше механизмы (ссылки к базам данных и снимки). Однако в случае Microsoft SQL Server можно использовать только полную репликацию главной таблицы в снимок (режим **COMPLETE REFRESH**). Мы не можем накапливать изменения в исходных таблицах и передавать их в результирующие, так как для Microsoft SQL Server пока не существует продуктов, аналогичных ORSDP.

3.9. Transparent Gateway for ODBC

Допустим, что мы хотели бы разместить компонент унификации доступа на стороне сервера. Кроме того, мы хотели бы использовать для организации доступа к различным БД обобщенный (ODBC) API. В случае ODBC на сервере устанавливаются все компоненты ODBC, описанные выше (кроме, разумеется, приложения), а также Oracle Server и компонент доступа к БД (например, Oracle Transparent Gateway for ODBC — TGODBC), транслирующий запросы приложения к базе данных (на Oracle SQL) в запросы на SQL ODBC и направляющий их целевой базе данных. Очевидно, что здесь мы имеем дело с удаленной конфигурацией. Отметим, что пока TGODBC доступен только для Windows NT.

Заключение

По-видимому, технология шлюзов пока была мало известна российским специалистам. Возможно, это объясняется очень небольшим числом по-настоящему интеграционных проектов. Ранее очевидно прослеживалась тенденция образования изолированных «островков информации», вокруг которых и строились технологии обработки данных. Резкое возрастание масштабов проектов и стремление к связыванию островков информации неизбежно востребуют и технологию шлюзов. Арсенал программных продуктов Oracle готов к этому.

Литература

1. Middleware: the Key to Distributed Computing. OVUM White Paper, 1996.
2. Oracle Transparent Gateway for DB2/400 Installation and User's Guide. Release 4.0.1.1.0. October 1997.
3. Oracle Open Gateways. Guide for SQL-Based and Procedural Gateways. Version 4.0. May 1997.
4. Oracle Transparent Gateway for Microsoft SQL Server. Installation Guide. Version 4.0.0.2. August 1997.

Sun объявляет о поддержке Oracle8*i*

10 ноября, практически одновременно с официальным представлением Oracle8*i*, компания Sun Microsystems объявила о поддержке этого передового продукта (см. http://www.sun.com/smi/Press/sunflash/9811/sunflash.981110.4.html).

Oracle8*i* был спроектирован, разработан и протестирован на платформе Sun. Продукты Oracle и Sun отлично подходят друг другу в плане надежности, масштабируемости, управляемости и эффективности. В совокупности они образуют мощную платформу для корпоративных информационных систем. Тысячи пользователей могут одновременно работать с терабайтными базами данных.

Продукт Oracle8*i* примечателен еще и тем, что в нем Java-технология применяется не только на клиентской, но и на серверной стороне. Компонентная архитектура приложений, обладающих Java-мобильностью, способствует повышению их качества и технологичности.

Oracle8*i* называют первой СУБД для Интернет-вычислений. «Три кита» в виде SPARC-архитектуры, операционной среды Solaris и Java-технологии образуют надежный фундамент передовых Интернет-решений.

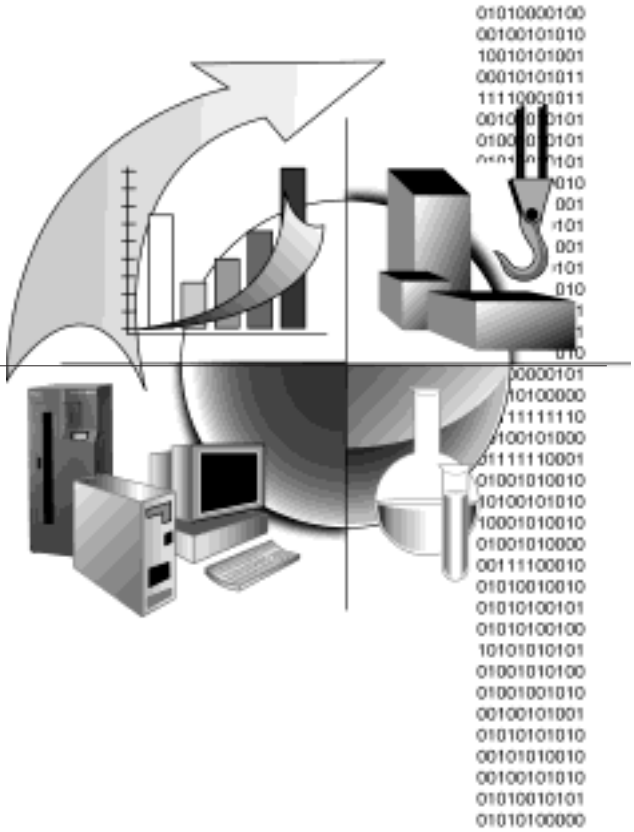
Очередной мировой рекорд Sun и Oracle

Объединение двух передовых продуктов — сервера СУБД Oracle8i и компьютера Sun Enterprise 10000 с ОС Solaris — позволило поставить очередной мировой рекорд производительности

продолжение на стр. 35

Объектные технологии в продуктах Oracle

Владимир Галатенко,
Глеб Ладыженский



Содержание

1. Введение

Спустя примерно 30 лет после зарождения (если за отправную точку брать язык программирования Симула-67), объектные технологии стали обязательным элементом коммерческих программных продуктов, в том числе систем управления базами данных. Выряжаясь официальным языком, сейчас у объектного подхода нет альтернатив. Причин тому несколько.

Во-первых, изменился характер данных, хранимых в БД. Существенно увеличился их объем, усложнились структуры данных и связи между ними. Без слов и словосочетаний «мультимедиа» и «информационно-насыщенные системы» не обходится ни один доклад. Традиционные реляционные таблицы с присущими им количественными ограничениями и недостаточно широким спектром базовых типов перестали соответствовать возросшим потребностям.

Во-вторых, изменилось место СУБД в общей архитектуре приложений. СУБД стали стандартным компонентом больших распределенных систем, компонентом, который больше не может диктовать правила игры, но должен подчиняться общим требованиям. От СУБД требуется совместимость с сетевыми сервисами (такими как HTTP), тонкое разграничение доступа, расширяемость и настраиваемость. На сегодняшний день только объектный подход позволяет успешно строить и эксплуатировать информационные системы большого масштаба, поэтому СУБД должны предоставлять соответствующие базовые средства и программные интерфейсы.

В-третьих, практически все компании — производители реляционных СУБД стремятся стать поставщиками готовых решений, а не отдельных компонентов, пусть и очень важных. Это заставляет обращать внимание на всю технологическую цепочку — от разработки до эксплуатации, от клиентов до серверов. И в каждом звене цепочки объектные технологии оказываются ключевым элементом.

Бюллетень Jet Info уже обращался к теме включения объектных средств в серверы реляционных СУБД (см. [1]). В настоящем материале нам хотелось бы представить подход корпорации Oracle к встраиванию объектных возможностей, а также к их архитектуре и реализации.

2. Трехуровневая архитектура современных информационных систем

В статье [2] подробно рассматривалась архитектура современных информационных систем. Было выделено три уровня:

- уровень представления (реализующий функции ввода и отображения данных);
- прикладной уровень (реализующий универсальные сервисы, а также функции, специфичные для определенной предметной области);
- уровень доступа к информационным ресурсам (реализующий фундаментальные функции хранения и управления информационно-вычислительными ресурсами).

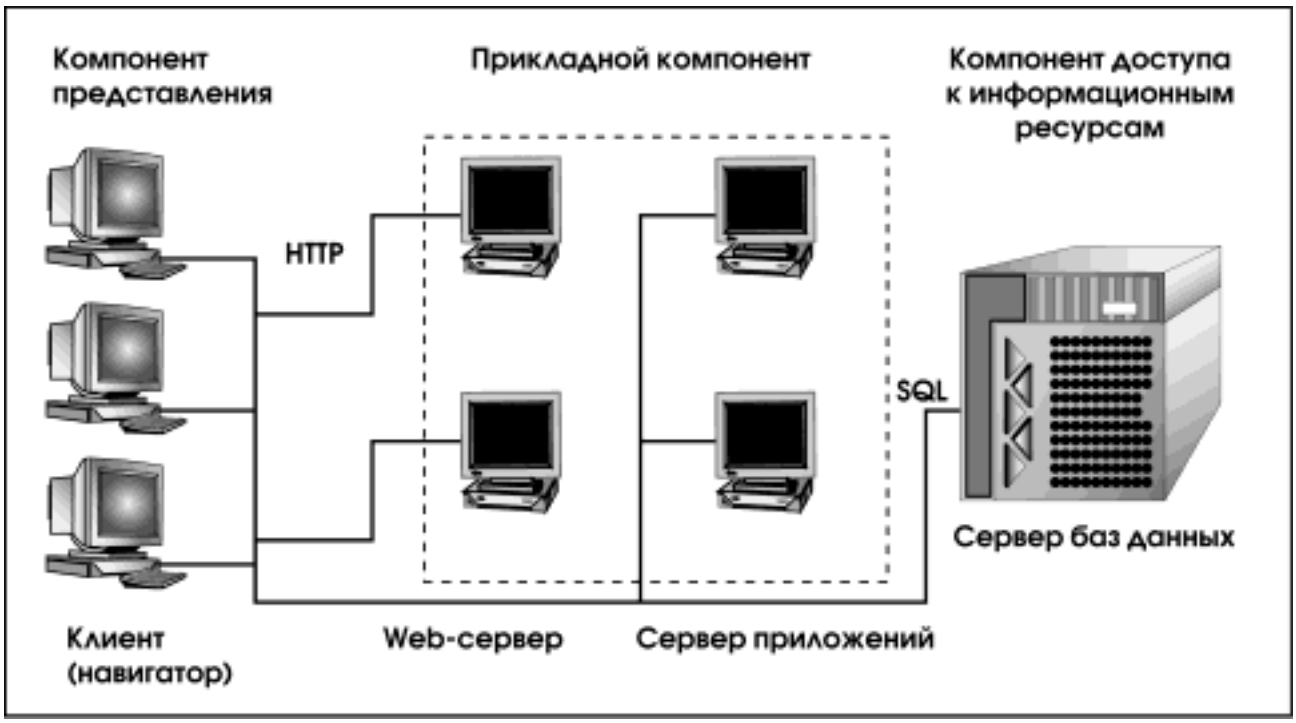


Рис. 1. Трактовка трехуровневой архитектуры в технологии Интранет.

Связь между уровнями обеспечивает менеджер транзакций и коммуникаций.

Технология Интранет наложила свой отпечаток на эту классическую схему, расположив на уровне представления универсальный клиент — Web-навигатор (возможно, пополненный прикладными аплетами) и возложив функции информационного концентратора (которые естественно объединить с функциями менеджера транзакций и коммуникаций) на Web-сервер. В результате получается схема, изображенная на рис. 1.

Опираясь на эту архитектуру, корпорация Oracle предлагает три ключевых элемента информационных систем:

- объектно-реляционный сервер СУБД Oracle8;
- универсальный сервер приложений Oracle Application Server 4.0, а также ряд специализированных прикладных серверов (Oracle Payment Server, Internet Commerce Server, Video Server и т.д.);
- набор драйверов в стандарте JDBC, специально оптимизированных для доступа из Java к СУБД Oracle8, а также SQLJ — SQL, встроенный в язык Java.

В интерпретации Oracle трехуровневая архитектура имеет еще одну особенность. Она изначально задумана как расширяемая (см. рис. 2). Основной единицей расширения является картридж — клиентский, прикладной или картридж данных. Предусмотрены средства для создания картриджей и интерфейсы для их состыковки с другими компонентами информационной системы.

Перечисленные элементы будут подробно рассмотрены в последующих разделах.

3. Объектно-реляционная СУБД Oracle8

Современный сервер СУБД можно сравнить с большим, изрядно нагруженным кораблем. Грузы эти разной степени свежести, но для заказчиков все они важны и все должны быть доставлены в порт назначения, поскольку за них заплачены деньги, и немалые.

Команда корабля способна улавливать ветер перемен, но не может быстро маневрировать, поскольку маршрут расписан на годы вперед, да и осадка такова, что волна вот-вот захлестнет палубу и смоет часть груза.

Для таких популярных СУБД, как Oracle, возможны только относительно неспешные, эволюционные изменения, не нарушающие устойчивость работы сервера и построенных на его основе многочисленных приложений. Даже такая ка-

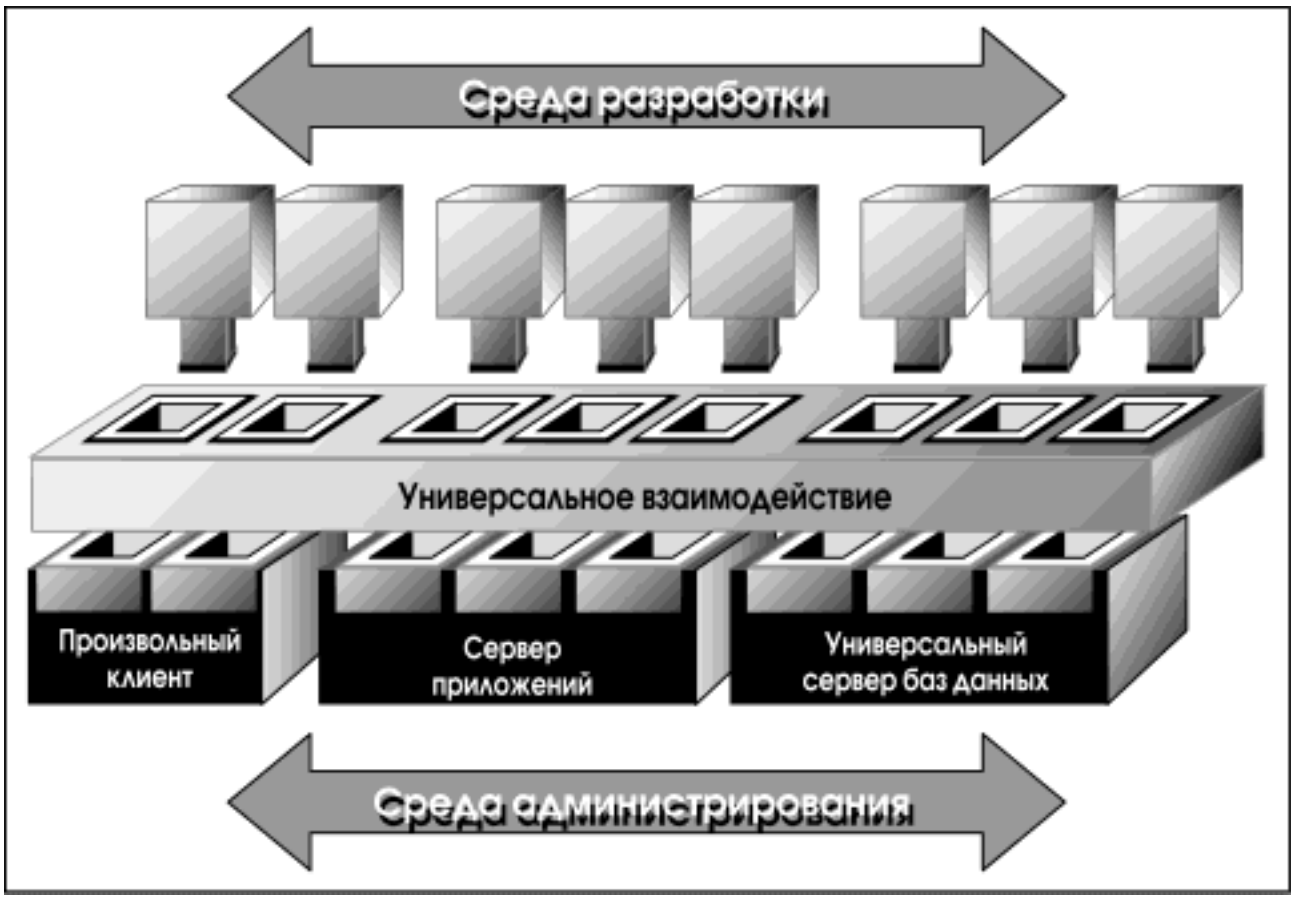


Рис. 2. Архитектура сетевых вычислений (Network Computing Architecture, NCA) корпорации Oracle.

залось бы ясная и апробированная вещь. как объектно-ориентированная технология, должна встраиваться постепенно, с продолжительными периодами тестирования и опытной эксплуатации каждой возможности. Подобные соображения следует иметь в виду, анализируя новые свойства сервера СУБД Oracle8.

3.1. Новые типы и структуры данных в Oracle8

Поддерживаемые типы и структуры данных являются основой любой СУБД, определяя в конечном счете стиль написания, эффективность и управляемость приложений. Опыт эксплуатации Oracle7 доказал необходимость развития типов и структур данных в трех направлениях:

- расширение набора встроенных типов данных;
- расширение спектра стандартных структур данных;
- предоставление пользователям возможности определять собственные типы и структуры данных.

В настоящем разделе мы рассмотрим первые два из перечисленных пунктов. Определяемые типы и структуры являются темой следующего раздела. Как правило, в примерах будет использоваться PL/SQL — язык хранимых процедур Oracle8, являющийся процедурным расширением SQL.

3.1.1. Поддержка национальных алфавитов

В Oracle8 представлены средства поддержки интернационализации /локализации (National Language Support, NLS). В частности, поддерживаются национальные алфавиты.

Типы данных NCHAR и NVARCHAR2 предназначены для хранения цепочек национальных символов, соответственно, фиксированной и переменной длины. Национальные символы могут быть многобайтными и даже иметь переменную длину.

На длину значений типов NCHAR и NVARCHAR2, хранящихся в столбцах реляционных таблиц, накладываются довольно жесткие ограничения — 2000 и 4000 байт соответственно. Если эти ограничения представляются обременительными, можно воспользоваться средствами для работы с большими объектами.

3.1.2. Средства для работы с большими объектами

«Каковы средства для работы с большими объектами?» — вот один из первых вопросов, который традиционно задают программисты, прежде чем приступить к реализации реальной системы. Имеются в виду прежде всего количественные ограничения, связанные с большими объектами.

В Oracle8 такие ограничения практически сняты. Размер большого объекта может достигать 4 Гб, а сами объекты полноценно участвуют в транзакциях.

С языковой точки зрения большие объекты представлены как бинарные (BLOB) и символьные (CLOB для текстов типа CHAR и NCLOB — для NCHAR, см. предыдущий пункт). Кроме того, большие объекты могут храниться внешним по отношению к СУБД образом, в файлах операционной системы. Этот вид хранения обслуживается типом данных BFILE.

С точки зрения реализации большие объекты представлены в столбцах реляционных таблиц так называемыми LOB-локаторами, содержащими, в частности, ссылку на реальное место хранения значений. Операции с большими объектами выполняются средствами программного интерфейса или пакета DBMS_LOB. При этом поддерживается случайный доступ внутри значений как на чтение, так и на запись (объекты типа BFILE доступны только на чтение).

3.1.3. Идентификаторы реляционных строк

Каждая строка каждой реляционной таблицы в СУБД Oracle8 имеет уникальный идентификатор (реализованный как физический адрес в БД). Этот идентификатор имеет тип ROWID. С каждой реляционной таблицей неявно ассоциируется столбец типа ROWID и с именем ROWID, хранящий адреса строк. Столбец ROWID, доступный только на чтение, может использоваться наравне с другими столбцами таблицы в операторе SELECT и конструкции WHERE.

По существу значения типа ROWID являются указателями на строки таблиц. В этом качестве они используются в индексах и объектных ссылках (см. далее описание типа REF). Ссылка остается корректной при модификациях строк, но «повисает» после удаления соответствующей строки.

Для работы со значениями типа ROWID служит пакет DBMS_ROWID. Предоставляемые этим пакетом функции позволяют разложить уникальный идентификатор на элементарные компоненты, а также выполнить некоторые другие вспомогательные действия.

3.1.4. Записи

Записи трактуются в языке PL/SQL Oracle8 вполне традиционным образом — как совокупность разнотипных (быть может, структурных) компонентов. Записи являются полноправным видом значений, их можно хранить в столбцах реляционных таблиц, передавать в качестве параметров и т.п.

На листинге 1 приведен пример описания типов записей и переменной-записи. Идея этого и следующего примеров заимствована нами из [3].

Обратим внимание, что второй тип (*AgendaItem*) содержит поле, представляющее собой запись первого типа (*TimeInterval*),

```
TYPE TimeInterval IS RECORD (
  minutes SMALLINT,
  hours   SMALLINT
);

TYPE AgendaItem IS RECORD (
  subject VARCHAR2 (100),
  duration TimeInterval
);

item_info AgendaItem;
```

Листинг 1. Пример описания типов записей и переменной-записи.

Для обращения к полям записей используется традиционная точечная нотация. На листинге 2 приведен пример обращения к полю записи, являющейся результатом функции.

```
FUNCTION longest_item (...)
    RETURN AgendaItem IS
  current_item AgendaItem;
  . . .
BEGIN
  . . .
  RETURN current_item;
END;

. . .
IF longest_item (...).duration.hours > 0 THEN
  . . .
```

Листинг 2. Пример обращения к полям записей.

В общем, записи введены в Oracle8 весьма последовательно, они практически свободны от реализационных ограничений. Несомненно, программисты, привыкшие к записям по универсальным языкам программирования, будут с удовольствием пользоваться ими и в PL/SQL.

3.1.5. Коллекции

Коллекции в Oracle8 представляют собой одномерные массивы с подвижными верхними границами. Коллекции подразделяются на два вида:

- вложенные таблицы (название подчеркивает тот факт, что подобные таблицы могут являться атрибутами реляционных таблиц);
- массивы переменного размера.

Вложенная таблица — это, в сущности, обычная реляционная таблица с одним столбцом. Число элементов во вложенной таблице практически не ограничено; существующие элементы могут удаляться, так что таблица не обязана являться непрерывным массивом (см. рис. 3).

Массивы переменного размера — более привычная сущность. При их описании задается максимальный размер; «дыр» в массиве быть не может.

На листинге 3 приведены примеры описаний типов — вложенной таблицы и массива переменного размера, элементами которых являются записи.

```
TYPE DictItem IS RECORD (
  word VARCHAR2 (30),
  translation VARCHAR2 (200)
);

TYPE MyDictionary IS TABLE OF DictItem;

TYPE HisDictionary IS VARRAY (1000)
    OF DictItem;
```

Листинг 3. Пример описания типов – вложенной таблицы и массива переменного размера.

Коллекции в Oracle8 живут как бы двойной жизнью — обычной для массивов и объектно-ориентированной. От массивов они унаследовали способ обращения к элементам — традиционную индексацию (см. листинг 4). Из объектного мира пришли конструкторы для создания значений-коллекций, а также методы для работы с ними.

Рис. 3. Вложенная таблица как одномерный массив.

```
DECLARE
  . . .
  dict MyDictionary;
  . . .
BEGIN
  . . .
  IF dict (i).word = 'Oracle' THEN . . .
  . . .
```

Листинг 4. Пример обращения к элементу коллекции.

Имя конструктора, как и положено, совпадает с именем типа коллекции. В качестве аргументов задаются значения инициализируемых элементов. Конструкторы могут использоваться как в выполняемых операторах, так и в декларациях.

К коллекциям применимы следующие методы:

- **EXISTS** (проверяет, существует ли элемент коллекции с заданным номером);
- **COUNT** (выдает текущее число элементов коллекции);
- **LIMIT** (выдает NULL для вложенных таблиц и максимальный размер для массивов);
- **FIRST/LAST** (выдают номер первого/последнего элемента коллекции);
- **PRIOR/NEXT** (выдают номер предыдущего/следующего элемента коллекции или NULL, если таковой отсутствует);
- **EXTEND** (добавляет к коллекции заданное число элементов);
- **TRIM** (удаляет из конца коллекции заданное число элементов);
- **DELETE** (удаляет заданные элементы вложенной таблицы).

Вызов этих методов оформлен по-объектному (см. листинг 5): за именем коллекции через точку следует имя метода, затем (при необходимости) — аргументы в скобках. Разработчики Oracle поступили весьма разумно, реализуя новые возможности в едином, современном стиле.

```
FOR i IN dict.FIRST () .. dict.LAST () LOOP
  IF dict.EXISTS (i) THEN dict.DELETE (i);
  END IF;
END LOOP;
```

Листинг 5. Пример использования методов коллекций.

3.2. Объектные типы данных

Объектный тип данных — это тип, определяемый пользователем, и задающий как структуру (атрибуты), так и поведение (методы) объектов. Объектный тип в Oracle8 является аналогом класса в других объектных языках программирования.

При описании объектных средств Oracle8 мы позволим себе немного пофантазировать, несколько расширив возможности языка PL/SQL. Дело в том, что в него пока не введены декларации вида

```
TYPE ... IS OBJECT ...
```

и, строго говоря, следует пользоваться операторами SQL*PLUS

```
CREATE TYPE ... AS OBJECT ...
```

но, конечно же, проблемы такого рода носят чисто технический характер и, несомненно, в следующих версиях будут устранены.

3.2.1. Описание объектных типов

Разделение интерфейса и реализации относится к числу общих мест объектного подхода. Такое разделение, позволяющее выбирать наиболее подходящую реализацию при сохранении интерфейса, проведено, например, в языке программирования ADA, оказавшем заметное влияние на PL/SQL.

Описание объектного типа состоит из двух частей (см. рис. 4). В интерфейсной декларируются атрибуты объектов и заголовки методов (процедур и функций). В теле типа приводится реализация методов.

На листинге 6 приведен пример описания объектного типа Person. В нем используется вспомогательный тип Address, также являющийся объектом (правда, без методов). Идея этого примера заимствована нами из [4].

По сравнению с традиционными языками программирования объектные средства Oracle8 имеют ряд особенностей. Первая из них состоит в том, что конструкторы объектов определяются неявно, их аргументами являются значения атрибутов. На листинге 7 приведен пример описания переменной типа Person и создания соответствующего значения. Подчеркнем, что конструкторы Person и Address мы не определяли.

Кроме обращения к конструкторам, на листинге 7 присутствует вызов метода age (). Видно, что он оформлен в традиционном стиле.

Вторая особенность объектных средств Oracle8 проистекает из того, что мы имеем дело не с языком программирования, а с СУБД, поэтому объекты, как и значения других типов, нужно индексировать, сравнивать между собой и т.п. Отношение порядка на объектах некоторого типа можно задать двумя способами:

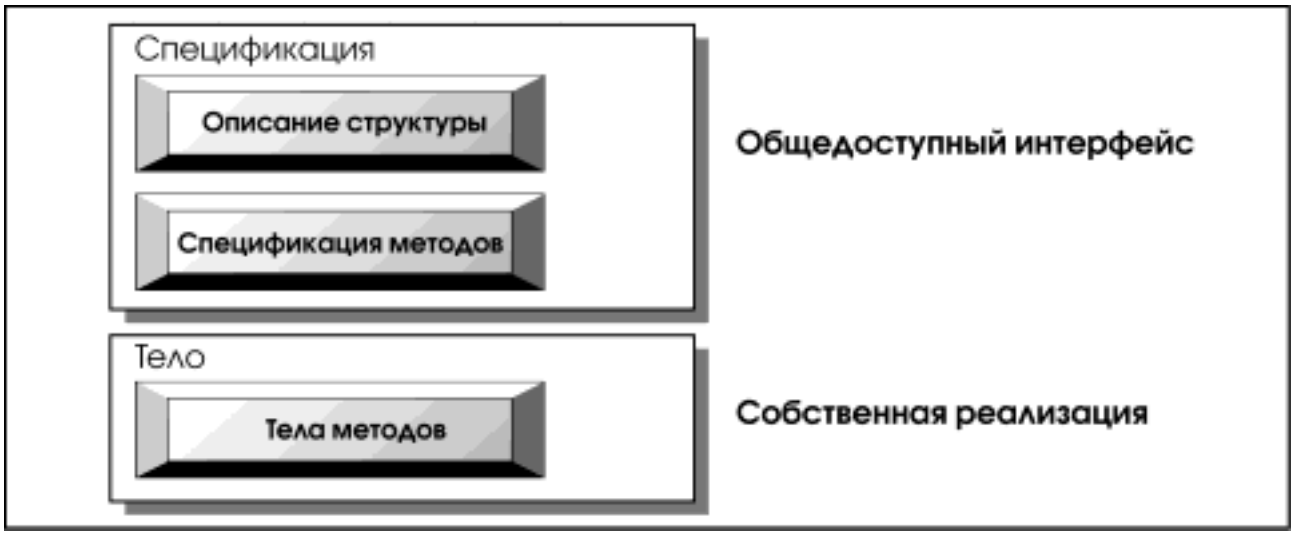


Рис. 4. Две части описания объектного типа данных.

```
TYPE Address IS OBJECT (
  country   VARCHAR2 (30),
  city      VARCHAR2 (30),
  street    VARCHAR2 (30),
  other     VARCHAR2 (50)
);

TYPE Person IS OBJECT (
  name      VARCHAR2 (30),
  addr      Address,
  birthday  DATE,
  MEMBER FUNCTION age ()
    RETURN NUMBER
);
. . .

TYPE BODY Person IS (
  MEMBER FUNCTION age () IS
    current_date DATE : =  SYSDATE ();
  BEGIN
    /* Вернем возраст, измеренный в днях */
    RETURN (current_date – birthday);
  END;

);
```

Листинг 6. Пример описания объектного типа данных.

```
DECLARE
  Smith Person : =  Person ('Smith',
                    Address ('Russia',
                          'Moscow',
                          'Savvinskaya nab.',
                          '15'),
                    '10-Nov-1960');
  SmithAge NUMBER : =  Smith.age ();
    . . .
```

Листинг 7. Пример декларации переменной объектного типа, использования конструктора и вызова метода объекта.

- определив функцию, отображающую объекты в упорядоченное множество значений (например, в числа);
- определив функцию, сравнивающую два значения-объекта.

Чтобы выделить такие «упорядочивающие» функции среди других методов, используются спецификаторы MAP и ORDER, соответственно. Например, функция age () могла быть описана следующим образом:

```
MAP MEMBER FUNCTION age ()
    RETURN NUMBER;
```

если, конечно, мы согласны сравнивать людей исключительно по возрасту.

Попутно отметим, что специфика СУБД сказывается еще и в том, что методам труднее, чем в обычном случае, хранить в объектах информацию о состоянии и извлекать ее. Точнее, в пределах транзакции проблем не возникает, а между транзакциями объект может измениться, и это следует принимать во внимание.

Наконец, третья особенность объектных средств Oracle8, носящая, очевидно, временный характер, состоит в том, что отсутствуют средства наследования, а все атрибуты и методы являются общедоступными. Пока не реализованы спецификаторы EXTENDS, PRIVATE, PROTECTED и т.п., присутствующие в традиционных объектно-ориентированных языках программирования.

3.2.2. Использование объектов

После того, как объектный тип описан (создан), его можно использовать наравне с другими типами данных. В частности, можно определять и создавать таблицы, в столбцах которых располагаются объекты, выполнять реляционные операции с подобными таблицами и т.п.

На листинге 8 приведен пример создания таблицы, описывающих работников. Один из столбцов этой таблицы имеет тип Person. К таблице применимы обычные реляционные операции, в которых могут участвовать компоненты объектов.

```
CREATE TABLE employees (
  empno          NUMBER (12) PRIMARY KEY,
  personal_data Person,
  salary          NUMBER (10, 2)
);

  . . .

SELECT empno, personal_data
FROM   employees
WHERE  personal_data.name = 'Smith';
```

Листинг 8. Пример таблицы со столбцом, имеющим объектный тип.

Если таблица содержит единственный столбец и этот столбец имеет объектный тип, то такая таблица называется в Oracle8 объектной. Объектные таблицы являются как бы мостом между двумя мирами — реляционным и объектным, поскольку на них можно смотреть с двух точек зрения. Можно «развернуть» объектный тип в совокупность атрибутов, и тогда мы получим обычную реляционную таблицу. Но можно оперировать с объектами-строками и как с единым целым, если воспользоваться оператором VALUE (см. листинг 9).

```
CREATE TABLE persons OF Person;

  . . .

   — Извлечем все атрибуты Person

SELECT *
FROM   persons p
WHERE  p.name LIKE '%Smith';

   — Извлечем некоторые атрибуты Person и результаты методов

SELECT p.name, p.age (), p.addr
FROM   persons p
WHERE  p.address.country = 'Russia';

   — Извлечем объекты-строки целиком

SELECT VALUE (p)
FROM   persons p
WHERE  p.name LIKE '%Smith';
```

Листинг 9. Пример операций с объектной таблицей.

Частным случаем объектных таблиц можно считать объектные представления. Вероятно, именно такие представления позволят постепенно преобразовать чисто реляционные данные в нечто более современное. Пример создания объектного представления приведен на листинге 10.

```
CREATE VIEW russian_persons OF Person
  WITH OBJECT ID (name)
  AS SELECT p.name, p.addr, p.birthday
    FROM persons p
    WHEN p.addr.country = 'Russia';
```

Листинг 10. Пример создания объектного представления.

3.2.3. Объектные ссылки

Для построения сложных структур данных нужны не только объекты, но и ссылки на них. В Oracle8 введен ссылочный тип данных REF, позволяющий ссылаться на объекты-строки (см. предыдущий пункт). На листинге 11 приведен пример декларации ссылки, ее порождения и использования.

```
DECLARE
  refToSmith REF Person;
BEGIN
  INSERT INTO persons p
  VALUES      (Person ('Smith', ...))
  RETURNING   REF (p) INTO refToSmith;

   . . .

  UPDATE persons p
  SET    p.birthday = '10-Nov-1958'
  WHERE  REF (p) = refToSmith;

   . . .

  UPDATE employees
  SET    personal_data = DEREF (refToSmith)
  WHERE  empno = 123;
END;
```

Листинг 11. Пример декларации ссылки на объект-строку, ее порождения и использования.

Для перехода от указателя к указуемому объекту служит функция DEREF, также использованная на листинге 11.

При изменении объектных таблиц ссылки на их строки могут «повиснуть». В Oracle8 введен SQL-предикат

```
IS DANGLING
```

выявляющий подобные ссылки. Пример использования этого предиката приведен на листинге 12.

Мы не определяли таблицу department, но смысл совершаемых действий достаточно очевиден — «обнулить» висячие ссылки.

```
UPDATE department
SET    manager = NULL
WHERE  manager IS DANGLING;
```

Листинг 12. Пример выявления и ликвидации висячих ссылок.

Разумеется, объектные ссылки могут являться атрибутами других объектов и храниться в столбцах реляционных таблиц, что позволяет более эффективно и, главное, более наглядно, по сравнению с внешними ключами, организовывать связи между элементами данных.

3.2.4. Некоторые реализационные соображения

Одно из главных достоинств реляционных СУБД состоит в развитом механизме оптимизации выполнения SQL-запросов. Вообще говоря, введение объектных средств вступает в конфликт с механизмами оптимизации, поскольку заранее неизвестен характер действий, выполняемых методами объектов, их относительная цена, устойчивость по отношению к распараллеливанию запросов и т.п.

В Oracle8 приняты определенные меры для решения проблемы оптимизации. При описании объектного типа посредством оператора

```
PRAGMA RESTRICT_REFERENCES
```

можно описать степень «чистоты» методов, то есть сообщить компилятору, каких побочных эффектов (состоящих в обращении к базам данных) при выполнении методов не будет.

В будущем предполагается предоставить открытые программные интерфейсы, помогающие строить индексы объектов и проводить глубокую оптимизацию SQL-запросов, в которых фигурируют вызовы методов.

Объекты могут не только мешать оптимизации, но и помогать ей. В Oracle8 на стороне клиента организуется специальный объектный кэш, куда за один раз передается с сервера как сам запрошенный объект, так и весь граф связанных с ним объектов. В результате экономится число сетевых взаимодействий, уменьшаются задержки в работе пользователей.

Для повышения возможности распараллеливания запросов Oracle8 производит блокировки на уровне отдельных объектов. Это важно,

Рис. 5. Основные составляющие технологической стороны картриджей данных.

если учитывать возможность разделения объектов (то есть наличия нескольких ссылок на один объект). Более «грубые» блокировки, несомненно, существенно сказались бы на работе пользователей.

3.3. Картриджи данных

Картридж данных — это совокупность определенных разработчиком типов данных (как правило, объектных) и подпрограмм (процедур и функций), привносящая в СУБД новую функциональность. Картриджи данных являются основным средством расширения для сервера Oracle8. На момент написания статьи существовали картриджи для работы с текстами (в том числе русскими), с изображениями, видеоинформацией, географическими данными и т.п.

У картриджа можно выделить две стороны:

- интерфейсную, видимую пользователю;
- технологическую, обращенную к серверу СУБД.

На рис. 5 показаны основные составляющие технологической стороны.

Строго говоря, в Oracle8 нет специальных средств для создания картриджей. Базой являются объектные типы данных и их методы, по большей части реализуемые в рамках внешних библиотек. Существенную помощь в группировке взаимосвязанных элементов оказывает аппарат пакетов языка PL/SQL, аналогичный имеющемуся в языке программирования ADA.

Как указывалось в предыдущем пункте, пока в Oracle8 не развиты технологические программные интерфейсы к подсистемам индексирования и обработки запросов. Трудно сказать, насколько это существенно, поскольку картриджи, хотя и интегрированы в СУБД, в значительной степени живут собственной жизнью, входящие в них методы потребляют значительные вычислительные ресурсы, так что оптимизация на уровне SQL может и не оказать заметного влияния на общий уровень эффективности.

Любое расширение сервера СУБД чревато снижением надежности его работы. Чтобы минимизировать риск, Oracle8 выполняет внешние подпрограммы в рамках отдельных процессов. Это несколько замедляет передачу параметров и возврат результатов, но зато защищает сервер от ошибок сторонних разработчиков и, что очень важно, обобщается на распределенный случай.

Для тех, кто хочет разрабатывать собственные картриджи, в качестве отправной точки можно порекомендовать статью [5]. Мы же рассмотрим в качестве примера картридж ConText, предназначенный для работы с текстами.

3.3.1. Картридж данных ConText

Реляционные СУБД являются самым продвинутым средством для хранения и обработки структурированных данных. С другой стороны, основная часть данных практически любой организации существует в виде неструктурированных текстов. Целесообразно объединить удобство SQL-запросов и технологичность реляционных СУБД со средствами текстовой обработки, в том числе интеллектуальными. Эту задачу и решает картридж данных ConText.

Отметим, что попытки организовать «самодельные» хранилища текстов, на наш взгляд, абсолютно бесперспективны. Поскольку, как мы уже отмечали, объем текстовых данных велик, проблемы эффективности, масштабируемости и управляемости встают для текстовых баз еще острее, чем для «чистых» реляционных СУБД. Без оптимизации и распараллеливания запросов, без поддержки многопроцессорных и кластерных конфигураций, без отработанных средств администрирования, короче говоря, без технологического багажа, отработанного за годы использования реляционных СУБД, системы хранения и обработки текстов не выдержат жестких требований промышленной эксплуатации.

С точки зрения хранения текстов картридж ConText не предлагает ничего особенного по той простой причине, что Oracle8 и так содержит достаточный набор возможностей (см. выше пункты «Поддержка национальных алфавитов» и «Средства для работы с большими объектами»). Единственной важной новинкой является хранение в столбцах таблиц универсальных локаторов ресурсов (URL) и прозрачная для пользователя выборка соответствующих документов.

Набор методов, реализуемых картриджем, можно подразделить на две группы:

- традиционная работа со словами;
- продвинутые лингвистические средства.

Первая группа методов базируется главным образом на словарных индексах, представляющих собой инвертированные списки, где для каждого слова указаны документы, его содержащие, и позиция слова в документе. Подобная структура позволяет эффективно осуществлять выборку по ключевым словам и фразам.

Разумеется, ConText обладает достаточным интеллектом для разбора различных форматов текстовых документов (обычный текст, HTML-документ, файлы MS-Word, PDF и т.п.). Кроме того, определенный интеллект проявляется и при построении индексов: можно задать (общеупотребительные) слова, которые в индекс не попадут. Для удобства работы введено понятие «текстовых политик», содержащих такие параметры, как подразумеваемый естественный язык, формат документов и т.д.

Разработчикам предлагается набор встроенных политик, пригодных для большинства практически важных случаев.

На уровне элементарных операций ConText предлагает обычные предикаты, операторы NEAR (встречается поблизости) SOUNDEX (звучит похожим образом) SYN (искать синонимы) и т.п. Искомым словам можно придавать различный вес, задавать «порог» для числа вхождений, подсчитывать итоговый «рейтинг» документа и т.д. Пожалуй, в рамках традиционной работы со словами ConText реализует максимум возможного.

На листинге 13 приведен пример таблицы с текстовыми атрибутами и запроса на языке PL/SQL, в котором задействованы методы картриджа ConText. Даже этот простой пример, заимствованный нами из статьи [6], показывает, насколько полезно сочетать текстовую специфику с обычными средствами языка SQL.

```
TEXTTABLE
  textkey    NUMBER    (unique primary key)
  candidate VARCHAR2 (100)
  source     VARCHAR2 (100)
  resume     NCLOB

  . . .

EXECUTE CTX_QUERY.CONTAINS
    ('TEXTTABLE_POLICY', 'Java', CTX_TEMP);

SELECT SCORE, candidate
FROM TEXTTABLE, CTX_TEMP
WHERE TEXTTABLE.textkey =
    CTX_TEMP.textkey AND
       TEXTTABLE.source = 'ACME Recruiting'
ORDER BY SCORE DESC;
```

Листинг 13. Пример использования возможностей картриджа ConText.

В примере обрабатываются резюме, поступившие в компанию из различных агентств по подбору персонала. Выбираются кандидаты, подавшие документы в агентство ACME Recruiting и являющиеся специалистами по Java. Аргумент 'TEXTTABLE_POLICY' метода CONTAINS обладает достаточной информацией о таблице TEXTTABLE; в частности, он «знает», в каком столбце располагается анализируемый текст.

Продвинутые лингвистические средства направлены на понимание смысла текстов, точнее, на определение тем, которые в тексте рассматриваются. На базе этих средств можно составить инвертированный список тем (где для каждой темы хранится перечень соответствующих ей документов), выдать «выжимку» документа (набор абзацев, наиболее полно освещающих затрагиваемые темы) и т.д. Понятно, что реализация возможностей такого рода требует масштабной настройки на естественный язык; замечательно, что для русского языка подобная настройка уже выполнена.

На примере картриджа ConText можно видеть, насколько органично на объектной основе сочетаются специфические методы, присущие определенной области знания, и реляционные механизмы. Перед нами тот случай, когда целое существенно больше арифметической суммы частей, поскольку составляющие поддерживают и усиливают друг друга.

4. Сервер приложений Oracle

4.1. Задачи, стоящие перед ПО прикладного уровня

Программное обеспечение прикладного уровня занимает в трехуровневой архитектуре (см. выше рис. 1) центральное место — и в прямом, и в переносном смыслах. На этом уровне решаются следующие важнейшие задачи:

- обеспечивается поддержка клиентских систем, работающих на различных аппаратно-программных платформах;
- обеспечивается связь с СУБД и другими сервисами третьего уровня;
- предоставляются инфраструктурные сервисы, такие как управление транзакциями, аутентификация, разграничение доступа, балансировка загрузки и т.п.;
- предоставляется прикладная функциональность с обязательной возможностью ее расширения.

ПО прикладного уровня, несмотря на разнообразие стоящих перед ним задач, должно оставаться технологичным, то есть:

- соответствовать требованиям к надежности работы ключевых элементов корпоративных информационных систем;
- быть управляемым (желательно максимально простым и однородным образом);
- быть масштабируемым (как в целом, так и по отношению к отдельным сервисам);
- поддерживать механизм транзакций (в том числе распределенных);
- предоставлять стандартные интерфейсы для расширения прикладной и инфраструктурной функциональности.

Наконец, должны существовать удобные инструментальные средства для разработки прикладных частей информационных систем.

Перечисленные требования выполнить отнюдь не просто. В принципе можно собрать набор программных продуктов (ПО промежуточного

слоя, Web-сервер, сервер аутентификации и т.д.), решающих поставленные задачи, но с большой вероятностью такой набор окажется нетехнологичным. Разнородными элементами трудно управлять, зачастую каждый из них расширяется по-своему, многие плохо масштабируются. В общем, если просто слить все имеющееся под рукой в пресловутый «один флакон», ничего хорошего не получится. Недостаточно и повторять заклинания про «технологию Интранет», поскольку ее внедрение в корпоративную среду решает далеко не все проблемы.

4.2. Идеи, положенные в основу сервера приложений Oracle

Основная идея, реализованная корпорацией Oracle в сервере приложений (Oracle Application Server, OAS), состояла в том, чтобы предложить единый программный продукт, решающий все перечисленные в предыдущем пункте задачи. Идея эта является нетривиальной хотя бы потому, что раньше подобных продуктов никто не делал. Решение, принятое корпорацией, требовало не только смелости, но и умения выявлять тенденции развития информационных технологий, поскольку реализация столь масштабного продукта и его «промышленная обкатка» — процесс, очевидно, многолетний, в котором неизбежны локальные неудачи и перестройки.

Стыковка различных элементов как внутри прикладного уровня, так и между уровнями, разумеется, должна выполняться на основе стандартов. Трудность состоит в том, чтобы из множества существующих спецификаций выбрать «минимально достаточное» подмножество, способное обслужить текущие и перспективне нужды. Двумя главными стандартами, взятыми на вооружение в OAS, стали HTTP и IIOP (Internet Inter-ORB Protocol, Интернет-протокол взаимодействия брокеров объектных запросов).

Протокол HTTP решает важнейшую задачу поддержки «универсальных тонких расширяемых клиентов» — Web-навигаторов. По этому поводу написано много, мы не станем повторяться.

Пожалуй, более существенной является ориентация на компонентную объектную архитектуру. Эта ориентация нашла свое выражение в поддержке архитектуры CORBA (см., например, [7]) и в следовании спецификациям Enterprise JavaBeans (EJB). CORBA и EJB позволяют решить несколько задач:

- обеспечить накопление прикладной функциональности в виде компонентов с практически неограниченными возможностями расширения;
- обеспечить поддержку распределенной объектной среды, что, помимо прочего, решает проблему масштабируемости;
- предоставить универсальный механизм общения с клиентскими системами — протокол IIOP.

Поддержка распределенной объектной среды, протоколов HTTP и IIOP дает возможность на тех же принципах построить систему управления прикладными сервисами, что очень важно с технологической точки зрения. Результирующая система получается концептуально экономной, технологичной, простой в освоении и использовании.

Организация взаимодействия с СУБД сейчас относится к числу рутинных задач, решаемых на основе спецификаций ODBC, JDBC, использования механизма хранимых процедур и мониторов управления распределенными транзакциями. Вполне понятно, что здесь у такой компании, как Oracle, проблем было меньше всего.

В заключение раздела — одно терминологической замечание. В материалах корпорации для обозначения прикладных компонентов служит термин «картридж», хотя по сути имеются в виду компоненты в смысле JavaBeans. В дальнейшем изложении мы будем следовать корпоративным традициям, чтобы подчеркнуть тот факт, что расширяемыми являются все уровни в сетевой архитектуре Oracle (правда, механизмы расширения оказываются разными).

4.3. Архитектура Oracle Application Server 4.0

Основные архитектурные элементы Oracle Application Server 4.0 и связи между ними показаны на рис. 6.

Как уже указывалось, для связи с клиентскими системами используются протоколы HTTP и IIOP.

HTTP-запросы принимает Web-сервер, который обслуживает их, если обращение производится к HTML-документам или CGI-процедурам. Прочие обращения передаются брокеру объектных запросов (ORB) или серверам картриджей.

IIOP-запросы сразу принимает ORB. Предполагается, что поддержка CORBA на клиентской стороне встроена в Web-навигатор или обеспечена аплетами, полученными по сети.

Агенты менеджера ресурсов обеспечивают маршрутизацию запросов от CORBA-клиентов к свободным экземплярам CORBA-объектов, возвращая объектную ссылку на такой экземпляр.

Картридж в контексте OAS — это компонент прикладного уровня, обеспечивающий некую содержательную функциональность (в следующем разделе мы рассмотрим основные картриджи, входящие в состав OAS 4.0). Сервер картриджей — это процесс операционной системы, в рамках которого функционируют экземпляры картриджей. Выбор картриджа для обслуживания

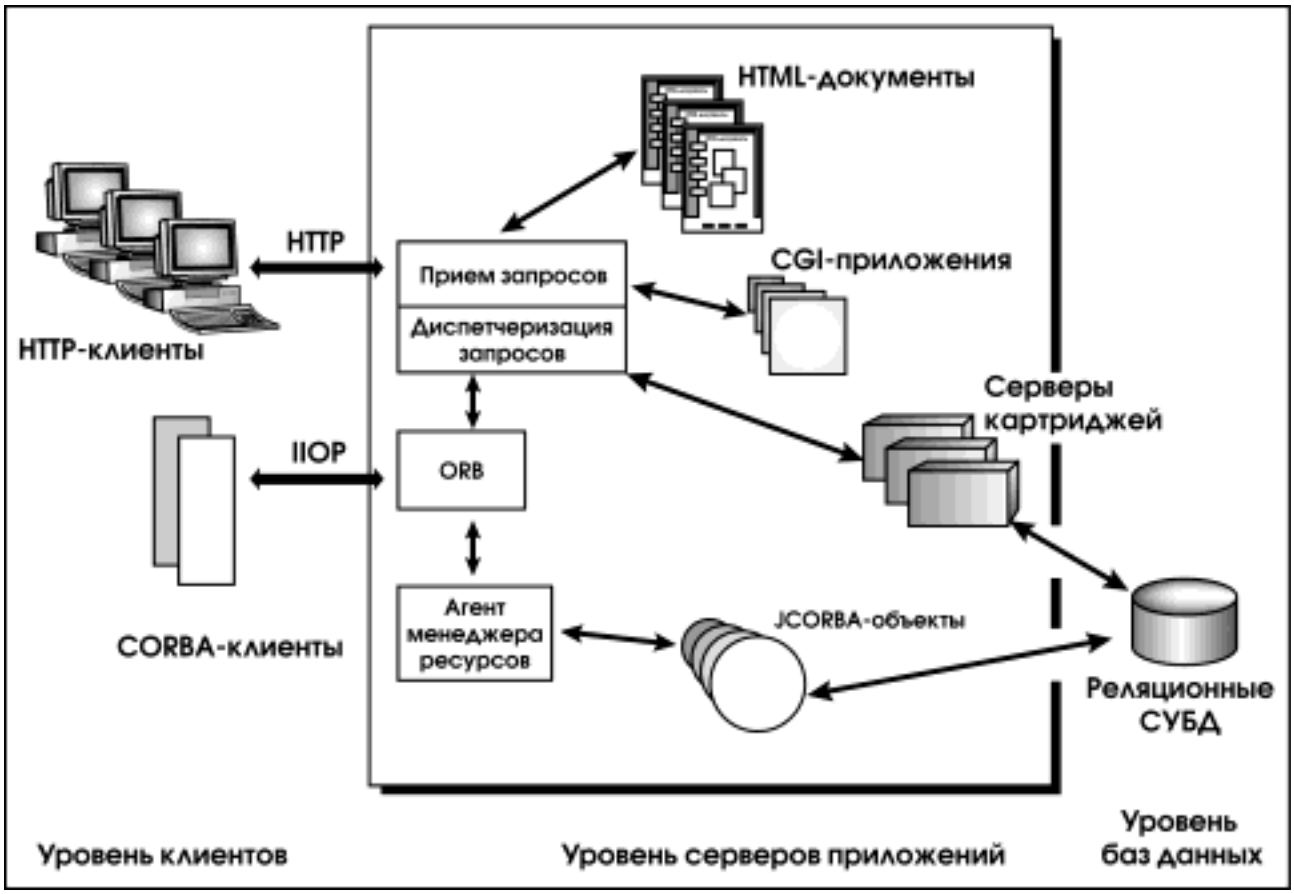


Рис. 6. Основные архитектурные элементы Oracle Application Server 4.0 и связи между ними.

пользовательского запроса осуществляется на основе анализа универсального локатора ресурсов (URL), заданного в запросе.

Помимо перечисленных элементов, непосредственно предназначенных для обслуживания пользовательских запросов, в состав OAS 4.0 входит ряд инфраструктурных сервисов:

- монитор транзакций (с поддержкой двухфазного механизма фиксации распределенных транзакций);
- сервис сообщений (современная разновидность программного обеспечения промежуточного слоя, служащая для поддержки различных схем интеграции приложений; получит дальнейшее развитие в последующих версиях);
- сервис аутентификации (с поддержкой нескольких схем проверки подлинности пользователей и приложений);
- доступ к реляционным базам данных (ODBC, JDBC, JSQL, X/A, а также ряд интерфейсов, специфичных для СУБД Oracle);
- доступ к службе каталогов по протоколу LDAP (Lightweight Directory Access Protocol, простой протокол доступа к каталогам), используемый при операциях с сертификатами X.509 версии 3 и списками разграничения доступа.

Еще один важный архитектурный элемент — взаимодействие между картриджами (Inter-Cartridge Exchange, ICX), организованное на основе протокола HTTP.

Более детальное представление об архитектуре Oracle Application Server 4.0 можно получить, прочитав статью [8].

4.4. Элементы, входящие в состав OAS 4.0

Фронтальным элементом сервера приложений Oracle является Web-сервер. В состав OAS 4.0 входит реализация Web-сервера, обеспечивающая поддержку HTML 3.2 и HTTP 1.1. При желании в OAS можно задействовать популярные продукты других компаний (разные варианты серверов Netscape, Microsoft IIS, Apache). Точнее, в OAS 4.0 имеются адаптеры для этих Web-серверов.

Брокер объектных запросов, входящий в OAS 4.0, удовлетворяет спецификациям CORBA 2.0. В комплект входят и реализации объектных сервисов CORBA (эти сервисы подробно описаны в статье [7]), и, в частности, сервис транзакций OTS. Особого упоминания заслуживает брокер Web-запросов (Web-Request Broker, WBR), являющийся объектной альтернативой CGI, альтернативой гораздо более мощной и эффективной.

WRB соединяет миры HTTP и CORBA и играет ключевую роль в OAS.

Вместе с OAS 4.0 поставляются следующие картриджи:

- Картридж PL/SQL. Этот картридж служит для выполнения в СУБД Oracle хранимых процедур, написанных на языке PL/SQL. Точнее говоря, картридж загружает исходные тексты процедур из файлов в базу данных и инициирует их выполнение. Вместе с этим картриджем поставляется PL/SQL Web Toolkit — инструментарий для организации вызовов хранимых процедур средствами HTTP и HTML.
- Картридж JWeb. Служит для выполнения Java-приложений. Содержит виртуальную Java-машину. Позволяет обращаться к базе данных посредством интерфейса JDBC или с помощью Java-классов, сгенерированных утилитой pl2java. Вместе с картриджем поставляется инструментальное средство JWeb Toolkit, позволяющее оформлять запросы к базе данных средствами HTTP и HTML.
- С-картридж. Предназначен для выполнения приложений, написанных на языке С. Обычно используется для программирования служебных функций, вызываемых сервером приложений.
- Картридж LiveHTML. Служит для интерпретации включаемых файлов на серверной стороне (Server Side Includes, SSI), позволяющих включать в HTML-документы динамические данные.
- Perl-картридж. Поддерживает выполнение Perl-процедур. Вместе с картриджем поставляется ряд Perl-модулей, таких как DBI/DBD, позволяющих обращаться из Perl-процедур к базам данных Oracle.
- ODBC-картридж. Позволяет клиентским системам посылать запросы к базам данных с помощью протокола HTTP.

В комплект поставки Oracle Application Server входит ряд продуктов третьих фирм:

- HTML-редактор Visual Page компании Symantec;
- средство мониторинга Net.Medic-Performance от компании VitalSigns;
- средство поддержки коллективной работы над Web-серверами Build-IT компании Wallop Software;
- программное обеспечение мониторинга Web-серверов компании WebTrends;
- инструментальный пакет Java Development Kit компании JavaSoft.

Oracle Application Server 4.0 поддерживает три модели приложений:

- модель запрос/ответ;
- сеансовая модель;
- транзакционная модель.

Модель запрос/ответ типична для протокола HTTP. Для ее реализации не нужно прилагать специальных усилий. В сеансовой модели вводится структура контекста, за счет чего сохраняется связь между клиентом и экземплярами картриджей. Для поддержки транзакционной модели в корпоративную редакцию OAS 4.0 введен Java Transaction Service (JTS), реализующий сервис OTS архитектуры CORBA.

Информационная безопасность не является предметом данной статьи. Тем не менее, следует отметить, что OAS 4.0 предоставляет весьма продвинутые средства безопасности:

- различные способы идентификации/аутентификации;
- управление доступом;
- средства протоколирования и аудита.

Таким образом, Oracle Application Server 4.0 есть нечто гораздо большее, чем Web-сервер с небольшими «нашлепками». OAS 4.0 предоставляет объектную инфраструктуру и полезные стандартные компоненты, позволяющие в короткие сроки создавать и развертывать серверы приложений корпоративного масштаба, обладающие всеми необходимыми функциональными и технологическими характеристиками.

5. Java-средства

Корпорация Oracle является одним из самых активных сторонников Java-технологии. Oracle поддерживает и внедряет разработки JavaSoft и других компаний, одновременно проводя собственные масштабные работы в области Java. Уже в статье [9] корпорация обнародовала Java-стратегию, которой продолжает следовать и поныне.

На рис. 7 сведены основные продукты Oracle, поддерживающие Java-технологию. Видно, что эти продукты распределены по всем уровням трехуровневой архитектуры, а также по всем фазам жизненного цикла информационных систем — от разработки до администрирования. Примечательно, что поддерживаются как «тонкие», так и «толстые» клиенты, то есть Oracle не пытается навязать своим заказчикам какую-то одну схему, он дает им возможность выбора (хотя и не скрывает собственных предпочтений).

Несколько продуктов, фигурирующих на рис. 7, на наш взгляд, нуждаются в пояснениях.

Корпорация предлагает собственную реализацию JDBC-драйверов, оптимизированную для взаимодействия с СУБД Oracle и поддерживающую то новое, что появилось в Oracle8. Разумеется, при этом сохраняется поддержка всех стандартных возможностей, а также распределенных конфигураций и (с помощью Oracle Open Gateways) «инородных» баз данных.

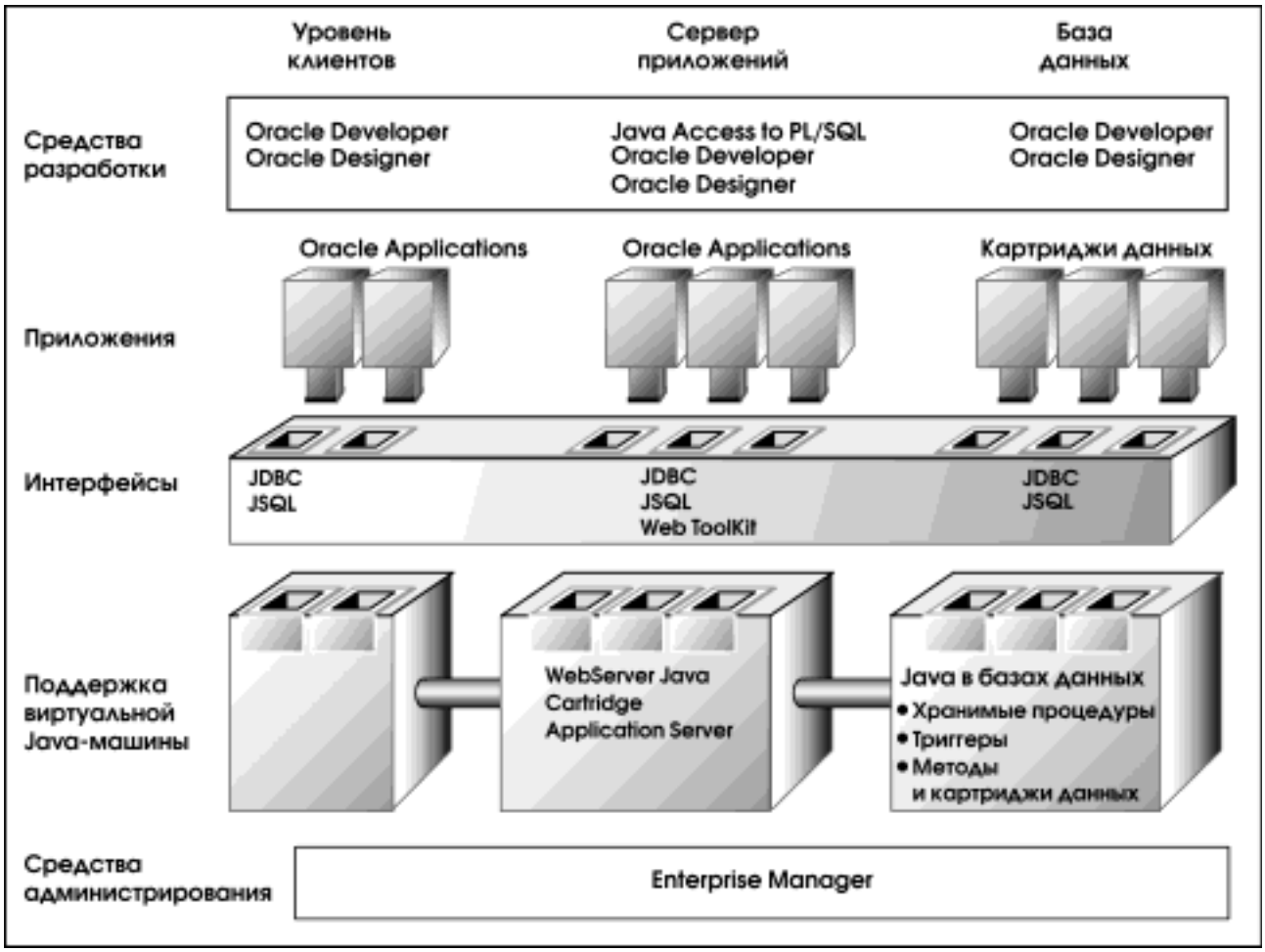


Рис. 7. Java в продуктах Oracle.

JSQL — это и спецификация, и препроцессор, обеспечивающий встраивание SQL-операторов в Java-программы. За прошедшие годы корпорация Oracle накопила огромный опыт по встраиванию SQL практически во все популярные языки программирования (C, C++, ADA, FORTRAN, COBOL, PASCAL), поэтому JSQL, несомненно, унаследует высокое качество предыдущих разработок. Отметим, что JSQL обладает важным преимуществом по сравнению с JDBC — возможностью обращаться к схеме базы данных во время компиляции. Это позволяет проводить проверки во время компиляции (с выдачей диагностики при обнаружении ошибок) и оптимизировать SQL-запросы.

6. Заключение

На наш взгляд, корпорация Oracle весьма последовательно и активно внедряет объектные технологии и на уровне серверов СУБД, и на прикладном и клиентском уровнях. Хотелось бы обратить внимание читателей на тот факт, что в данном случае объектный подход выполняет в первую очередь инфраструктурную, системообразующую функцию, делая программные продукты Oracle технологичными, способными работать в разнородном, распределенном окружении. В то же время, превращение Oracle8 в объектно-реляционную СУБД имеет и, если так можно выразиться, утилитарную ценность, помогая пользователям более естественным образом строить базы данных и постепенно мигрировать от чисто реляционной к более содержательной модели данных.

Перспективность пути, выбранного корпорацией, подтверждает сопоставление с деятельностью других передовых компаний, таких как Sun Microsystems. Развитие компонентной модели программных систем, использование Java на серверной стороне и в качестве программного обеспечения промежуточного слоя — все это общие достижения, способствующие прогрессу информационных технологий, прогрессу, выгодному всем.

7. Литература

1. Галатенко В., Гвоздев А. Типы и структуры данных в INFORMIX-Universal Server. — Jet Info, 1997, 12/13.
2. Ладыженский Г. Tuxedo System — ключевой компонент корпоративных информационных систем. — Jet Info, 1996, 14/15.

3. PL/SQL User's Guide and Reference. Release 8.0. — Oracle Corporation, 1997.
4. Objects and SQL in Oracle8. Technical White Paper. — Oracle Corporation, 1997.
5. Oracle8 Data Cartridge Development. Technical White Paper. — Oracle Corporation, 1997.
6. Managing Text with Oracle8 ConText Cartridge. Technical White Paper. — Oracle Corporation, 1997.
7. Пуха Ю. Объектные технологии построения распределенных информационных систем. — Jet Info, 1997, 16.
8. Oracle Application Server 4.0 White Paper: Product Overview. — Oracle Corporation, 1998.
9. Bringing Java to the Enterprise. Technical White Paper. — Oracle Corporation, 1997.

Sun объявляет о поддержке Oracle8i

Начало на стр. 19

на тестах TPC-D с терабайтным объемом данных (см. http://www.sun.com/smi/Press/sunflash/-9811/sunflash.981103.5.html).

Напомним, что TPC-D является синтетическим тестом, характеризующим производительность системы на задачах поддержки принятия решений и работы с хранилищами данных.

При установлении рекорда использовалась аппаратная конфигурация, включающая сервер Sun Enterprise 10000 (64 процессора UltraSPARC II с тактовой частотой 336 МГц, каждый с внешним кэшем в 4 МБ, 60 ГБ оперативной памяти), а также дисковые подсистемы Sun StorEdge A5000 Fibre Channel Array и Sun StorEdge A3000 Array суммарным объемом 9.6 ТБ. Дисковые подсистемы функционировали в режиме RAID-5, чтобы обеспечить надежную защиту данных от повреждений и потерь. Основу программной конфигурации составляли ОС Solaris 2.6 и сервер СУБД Oracle8*i* версии 8.1.5.

Читатели Jet Info имели возможность подробно ознакомиться со всеми перечисленными продуктами. Сервер Sun Enterprise 10000 описан в статье А. Шадского «Семейство компьютеров Ultra компании Sun Microsystems» (Jet Info, 1997, 23/24). Дисковые подсистемы — тема статьи П. Анни и Л. Черняка «Интеллектуальная сеть хранения данных» (Jet Info, 1998, 4). Операционной среде Sun Solaris посвящена одноименная статья в журнале Jet Info, 1997, 15. Наконец, данный номер открывается материалом об Oracle8*i*.

До сих пор только три компании смогли выполнить тесты TPC-D на терабайтном уровне. Это Sun, IBM и NCR. Следует отметить, что превосходство команды Sun-Oracle над конкурентами выглядит весьма впечатляющим: производительность оказалась выше, чем у «второго призера» (IBM RS/6000 SP с СУБД DB2) примерно на 40%, а цена — на 15% меньше. Любопытно, что новый рекордсмен оттеснил на третье место кластерную конфигурацию из четырех узлов Sun Enterprise 6000 с СУБД Informix (здесь превосходство в производительности более чем двукратное при цене, меньшей примерно на те же 15%).

У продуктов Sun и Oracle исключительно большой потенциал. На рынке корпоративных систем с ними очень сложно конкурировать. И Sun, и Oracle относятся к числу лидеров, которые никогда не останавливаются на достигнутом.

Solaris в новой фазе

В конце октября 1998 года компания Sun Microsystems представила новую версию своей операционной среды — Solaris 7 (см. http://-www.sun.com/solaris/). Уже изменение в нумерации версий показывает, что сделан важный, принципиальный шаг вперед. Мы попытаемся кратко рассмотреть новые возможности Solaris 7.

Операционная среда Solaris стала основой для трех продуктов (см. рис. 1):

- Solaris Easy Access Server (рассчитан на уровень рабочих групп и отделов);
- Solaris Enterprise Server (продукт корпоративного уровня);
- Solaris ISP Server (рассчитан на поставщиков Интернет-услуг).

Среда Solaris включает:

- операционную систему в традиционном понимании (ядро, библиотеки программного интерфейса, пользовательские и административные утилиты);
- сетевое программное обеспечение;

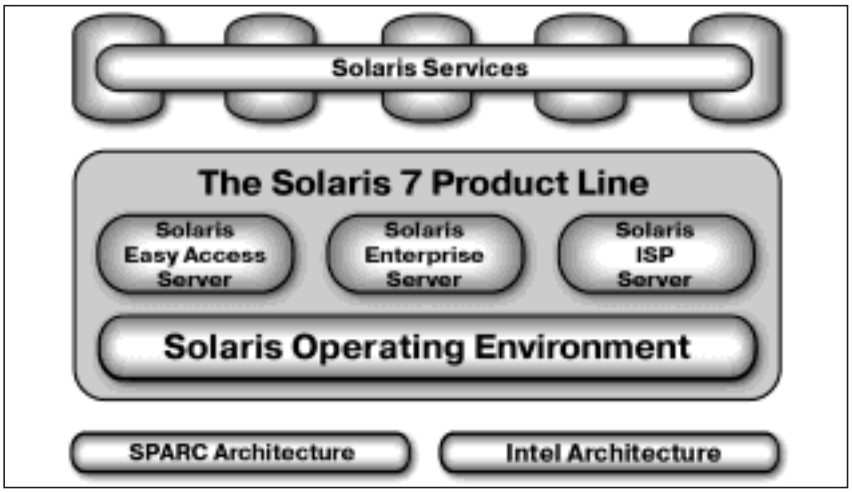


Рис. 1. Семейство продуктов Solaris 7.

- виртуальную Java-машину и среду разработки (JDK);
- средства поддержки надежности, готовности и обслуживаемости;
- графическую среду CDE (Common Desktop Environment, общее настольное окружение).
- поддержку платформ SPARC и Intel, как 32-битных, так и 64-битных.

Solaris Easy Access Server характеризуется прежде всего простотой установки и администрирования, которые обеспечиваются Web- и Java-средствами, а также тесной интеграцией с сетями персональных компьютеров, достигаемой применением семейства продуктов SunLink.

В Solaris Enterprise Server включена поддержка кластерных конфигураций, средства управления производительностью (системной и сетевой), продвинутые механизмы аутентификации, базирующиеся на Kerberos версии 5.

Solaris ISP Server характеризуется наличием средств администрирования Sun Internet Administrator и улучшенными реализациями Интернет-сервисов, интегрированными со службой каталогов.

Из новых свойств и возможностей среды Solaris 7 выделим следующие:

- 64-битность (при сохранении поддержки 32-битных аппаратных платформ и приложений);
- поддержка динамической реконфигурации плат и путей ввода/вывода.
- журнализация операций с файловыми системами UFS;
- поддержка выборочных подтверждений на TCP-уровне.

В Solaris 7 64-битность реализована полностью — в арифметике, в адресации, в операциях с файлами. Введено понятие модели данных. Привычные 32-битные возможности соответствуют модели ILP32 (под указатели и типы integer и long отводится 32 бита). Новой для Solaris 7 стала модель LP64, в соответствии с которой целые по-прежнему занимают 32 бита, а указатели и длинные целые — 64. Понятно, что изменение размеров базовых типов создает определенные проблемы; мы, однако, не будем обсуждать их здесь, поскольку предполагаем посвятить Solaris 7 отдельную статью.

Строго говоря, адресация в Solaris 7 стала не 64-, а 44-битной в соответствии с размером виртуального адреса в процессорах UltraSPARC. Впрочем, 16 ТБ — величина также немалая.

Работа с большими файлами была введена в Solaris ранее. Здесь мы отметим новый формат объектных файлов ELF64, поддерживающий 64-битные приложения.

Поддержка динамической реконфигурации плат и путей ввода/вывода распространяется в Solaris 7 на серверы Sun Enterprise 3x00 — 6x00 (для модели 10000 она существовала ранее). Возможность динамической реконфигурации процессоров и оперативной памяти будет добавлена в ближайших версиях.

Журнализация операций с файлами практически исключает повреждение данных даже при сбоях системы. Изменения файловых систем, смонтированных с опцией «-o logging», выполняются примерно в том же стиле, что и изменения таблиц в реляционных СУБД.

Поддержка выборочных подтверждений на TCP-уровне повышает эффективность длительных сетевых взаимодействий при большом размере окна. Для современных Интернет-сервисов это представляется весьма актуальным.

Выпуск Solaris 7 подтверждает не просто жизнеспособность, но безусловное лидерство операционной системы Unix как серверной платформы корпоративного уровня.

Jet Info
ИНФОРМАЦИОННЫЙ БЮЛЛЕТЕНЬ
Издается с 1995 года
Издатель: компания Джет Инфо Паблишер

Главный редактор: Галатенко В.А. *(galat@jet.msk.su)*
Технический редактор: Антонов А.Н. *(silver@jet.msk.su)*

Россия, 103006, Москва, Краснопролетарская, 6
тел. (095) 972 11 82, 972 13 32
факс (095) 972 07 91
e-mail: JetInfo@jet.msk.su

Подписной индекс по каталогу Роспечати **32555**