ОГЛАВЛЕНИЕ

ОБ АВТОРЕ

Джонатан Льюис (Jonathan Lewis) – опытный преподаватель с математической подготовкой из Оксфордского университета. Хотя первый интерес к компьютерам появился у него еще в нежном возрасте, что-то около 12 лет, – во времена, когда под высокими технологиями понималось использование клавиатуры, а не устройства для перфорирования карт – он, все же, не получил развития, пока Джонатан не закончил университет и не начал заниматься вычислениями профессионально. Кроме первого года после обучения, когда он, как неопытный продавец отвечал на любой вопрос «Да», вместо «Да, но это займет несколько недель на программирование и настройку», Джонатан работал только на себя.

Первое знакомство Джонатана с Oracle состоялось с версией 5.1, которую он использовал на ПК, проектируя и создавая систему управления рисками для сырьевой торговой площадки в одной из крупных нефтяных компаний. (Тогда он написал на ПК первую версию системы с названием dBase III, в которой использовались таблицы, индексы и некоторые другие механизмы, характерные для систем управления реляционными базами данных.) Благодаря Oracle он узнал, что должна делать настоящая система управления реляционными базами данных, и эти знания немедленно были взяты им на вооружение.

С того времени Джонатан сосредоточился исключительно на использовании Oracle RDBMS. Спустя три или четыре года работы по контракту, в течении которых он обрел опыт строительства больших систем, Джонатан решил закончить работы по контракту и заняться оказанием консалтинговых услуг, и теперь полностью посвящает свое

время краткосрочным консалтинговым работам, проведению семинаров и «исследовательским» работам.

Джонатан хорошо известен в мире Oracle – в течение последних 10 лет он работал в 50 странах и в десятках штатов США. Он оказывает посильную поддержку группам пользователей; в частности, принимает активное участие в работе группы пользователей Oracle Великобритании «UK Oracle User Group» (www.ukoug.org). При каждой удобной возможности старается выкроить время для встреч с группами пользователей в других странах, за пределами Великобритании, которые иногда имеют формат коротких собраний после окончания рабочего дня. Он также ведет свой блог, посвященный Oracle (http://jonathanlewis.wordpress.com) и регулярно (когда не занят работой над очередной книгой) пишет статьи для разных журналов, форумов и семинаров.

Джонатан совсем недавно отпраздновал серебряную свадьбу со своей супругой Дианой (Diana) (в настоящее время работает учителем в начальной школе и заведующей отделением математики после многих лет работы бухгалтером). У них есть двое детей: Анна (Anna, в настоящее время учится на последнем курсе Оксфордского университета) и Симон (Simon, учится на первом курсе Йоркского университета). Несмотря на привязанность родителей к математике, ни один не пошел по их стопам.

О ТЕХНИЧЕСКИХ РЕЦЕНЗЕНТАХ

Танел Подер (Tanel Põder) – специалист международного уровня по вопросам производительности Oracle, помогающий решать сложные проблемы своим клиентам в более чем двадцати странах на пяти континентах. Специализируется на тонкой настройке производительности, полной диагностике и решении других сложных (и от того особенно интересных) задач, таких как миграция огромных баз данных за ограниченный интервал времени. Танел занимался оптимизацией производительности аппаратно-программных комплексов Exadata, начиная с версии Exadata V1, и планирует провести еще более обширные исследования производительности Exadata. Является соавтором книги «Expert Oracle Exadata» (Apress, 2011).

Танел одним из первых в мире получил сертификат «Oracle Certified Masters», имеет звание «Oracle ACE Director» и является полноправным членом OakTable Network. Регулярно выступает на конференциях по всему миру, публикует статьи, а также распространяет сценарии и инструменты в своем блоге http://blog.tanelpoder.com.

БЛАГОДАРНОСТИ

В первую очередь я хочу поблагодарить мою супругу Диану, за ее поддержку, пока я писал эту книгу. Без ее любви и терпения не было бы этой книги. Она приготовила мне несчетное число чашек чая (которые часто оставались на столе нетронутыми), терпела, когда я работал поздно вечером (или рано утром), и не жаловалась, когда я работал над книгой в ущерб другим делам. Эта книга требовала большой концентрации, и если вместо меня никто не делал бы повседневные дела, я не смог бы ее написать.

Любая техническая книга нуждается в рецензировании, поэтому я, конечно же, благодарен Танелу Подеру (http://tech.e2sn.com/ и http://blog.tanelpoder.com/), что стал моим техническим рецензентом. Его замечания, вопросы и исправления были весьма ценны и поучительны.

У меня были также два «неофициальных» рецензента: Мартин Видлейк (Martin Widlake, http://mwidlake.wordpress.com) и Тимур Ахмадеев (Timur Akhmadeev, http://timurakhmadeev.wordpress.com/), оказавшие мне немалую помощь в создании книги. Я просил их помочь мне выяснить, что еще добавить в книгу, что убрать, и какие ее части несут неоднозначную информацию; однако они здорово помогли мне также в вылавливании технических ошибок. Если вам понравятся диаграммы в этой книге, благодарите за это Мартина, потому что это он постоянно говорил мне о том, какие описания следует сопроводить иллюстрациями.

Андрей Николаев (Andrey Nikolaev, http://andreynikolaev.wordpress.com/) был специально приглашен для рецензирования главы 4. В Интернете имеется огромный объем информации с описанием работы внутренних механизмов Oracle – большая часть этого объема является простым повторением сведений, устаревающих с годами. Лучшей демонстрацией этого феномена являются сведения о «защелках» (latches); отчасти потому что это, безусловно, очень сложная тема. Андрей сделал намного больше, чем любой другой автор в Интернете, представив современное описание защелок (и мьютексов) –

поэтому я просил его составить свое мнение о главе 4. Я очень благодарен Андрею за его комментарии и исправления.

Даже при том, что над книгой работали такие превосходные рецензенты, вы все еще можете встретить в ней ошибки и недостатки – вина за все эти ошибки и недостатки лежит исключительно на мне. Например, иногда я преднамеренно допускал, возможно, излишнее упрощение, иногда я сам мог что-то неправильно понять. Значительная часть этой книги рассказывает о работе внутренних механизмов, но ни я, ни приглашенные мною рецензенты не имеют доступа к внутренней, закрытой документации с описанием таких механизмов; мы – обычные люди, применяющие научный подход ко всему, что можно исследовать и измерить. В течение долгого времени мы наблюдаем и обсуждаем работу того или иного механизма, вырабатывая единое мнение, которого придерживаемся, пока не появятся новые данные.

Я мог бы назвать еще много имен людей, так или иначе внесших свой вклад в эту книгу; людей, которые давали мне ответы, задавали вопросы или приглашали для решения интересных задач. Вы всегда сможете найти огромное число вопросов на форумах, таких как OTN (http://forums.oracle.com) и Oracle-L (www.freelists.org/archive/oracle-l), где описываются реальные проблемы с Oracle, и огромное число людей, обладающих знаниями и опытом, позволяющими решать эти проблемы. Без непрекращающегося потока вопросов так легко застрять в колее и думать, что вы знаете все. Я с полной уверенностью могу сказать, что значительная часть моих знаний об Oracle была получена в ходе исследований, которыми я занимался, пытаясь найти ответы, а не подсматривал готовые решения в Интернете.

ВВЕДЕНИЕ

Когда я написал книгу «Practical Oracle 8*i*», прошло всего три недели с момента публикации до первого электронного письма, где меня спрашивали, когда я напишу книгу для версии 9*i* – благодаря Ларри Эллисону (Larry Ellison)[1], выпустившему к тому моменту версию 9*i*. За последние 12 лет этот вопрос мне задавали много-много раз (менялся только номер версии). Эта книга показывает, насколько я был близок к решению начать работу над вторым изданием, правда в ней охватывается только первая (и немного вторая и третья) глава оригинала.

Два обстоятельства побудили меня вновь сесть за письменный стол. Во-первых, я очень часто наталкивался на вопросы вида: «Как Oracle делает то-то или то-то?». Во-вторых, было подмечено, что на такие вопросы почти невозможно найти ответы, которые были бы просты, понятны и содержательны. Обычно приходится перелопачивать горы руководств, чтобы найти ответы на часто задаваемые вопросы. Если заняться поисками ответа в Интернете, можно найти массу статей с описанием некоторых тонкостей в работе Oracle, но вы не найдете достаточно связного описания, которое поможет понять, как работает тот или иной механизм и почему он работает именно так, а не как-то иначе. В этой книге я попытался дать именно такие ответы. Мне хотелось рассказать, как работает Oracle, поведать вам связную историю, а не просто дать набор отрывочных сведений.

Цели

Руководства с описанием версии 11*g* занимают десятки тысяч страниц, поэтому кажется маловероятным, что в книге, содержащей всего пару сотен страниц, удалось описать «как все работает», поэтому я хочу обозначить главные цели. Книга описывает основные механизмы ядра базы данных, приводящие в движение все остальное; по сути в их число входят механизмы повтора, отмены, кэ-

[1] Лоренс Джозеф Эллисон (Lawrence Joseph Ellison) – основатель и исполнительный директор корпорации Oracle. Также известен как Ларри Эллисон. – *Прим. перев.*

ширования данных и разделяемого кода SQL. Но даже для такого ограниченного числа целей я вынужден был быть безжалостным к деталям и интересным специальным случаям, чтобы не увеличить объем книги и не превратить ее в толстый и неинтересный фолиант. Например, возьмем простой вопрос: «Как Oracle осуществляет логический ввод/вывод?», и посмотрим на структуру `x$kcbsw` со списком всех функций, которые Oracle может вызывать, чтобы посетить блок. В версии 11.2.0.2 таких функций 1164 – что бы вы предпочли получить, подробную справку по всем имеющимся вариантам или описание общих требований?

Проблема детализации повторяется на другом уровне – как много тонкостей вы хотите знать; и какие выгоды вы получили бы от книги, если бы я потратил все свое время на описание некоторых запутанных тонкостей. Как всегда, приходится искать золотую середину между полнотой, точностью и удобочитаемостью. Я старался следовать идее, которую выразил Эндрю Холдсворт (Andrew Holdsworth) из Oracle Real-World Performance Group на конференции Oracle OpenWorld в 2006 году. В презентации, посвященной оптимизатору и приемам сбора статистики, он говорил о методологии 90/9/1 следующее:

- в 90% случаев достаточно выборки по умолчанию;
- в 9% случаев необходима расширенная выборка;
- в 1% случаев размер выборки не имеет значения.

Это – расширенная формулировка известного правила Парето (Pareto) 80/20, и она, как мне кажется, вполне применима при изучении внутренних механизмов Oracle. Но для целей данной книги я хочу немного переупорядочить это правило: 90% читателей достаточно общей информации о работе Oracle; 1% читателей нужно знать все тонкости, чтобы понять, что пошло не так; и 9 процентам (для которых и написана данная книга) хотелось бы знать чуть больше о своих базах данных и потратить на изучение чуть меньше времени, чем необходимо на изучение всех тонкостей.

Куда дальше

Некоторое время тому назад Танел Подер (мой технический рецензент) в ответ на вопрос о том, когда он собирается начать писать книгу о внутреннем устройстве Oracle, сказал так:

> *«Никогда, если речь идет об обычной, печатной книге. Изменения в этой области следуют друг за другом слишком быстро. Чтобы книга получилась достаточно хорошей, нужен как ми-*

нимум год упорной работы, но ко времени публикации многие детали окажутся устаревшими»

Хороший ответ. Он еще больше утвердил меня во мнении уйти от подробного описания и сосредоточиться на описании общих требований с меньшей степенью детализации. Решение проблемы Танел нашел в создании «живой книги», доступной по адресу: http://tech.e2sn.com/oracle.

Цельная книга (даже электронная) лучше простой коллекции статей в Интернете (пусть самых лучших), потому что, как мне кажется, формат книги обеспечивает плавное течение мысли. Когда я работал над этим введением, в моем блоге насчитывалось более 650 статей (суммарный объем которых значительно превышает объем этой книги). Я мог бы объединить несколько статей в книгу, но в такой книге было бы мало ценного, даже если бы я потратил массу времени на абзацы, связывающие статьи между собой. Я глубоко убежден, что техническая книга тоже должна представлять собой связное повествование.

Чтобы решить проблему, свойственную печатным (не «живым») книгам, я разместил в своем блоге http://jonathanlewis.wordpress.com/oracle-core/ по одной странице из каждой главы. В течение долгого времени я мог получать от читателей сообщения об ошибках или короткие дополнения к опубликованной версии. Кроме того, как в любом другом блоге, читатели могли задавать свои вопросы и участвовать в обсуждении. На вопрос о том, когда выйдет второе издание моей той или иной книги, я отвечал, что никогда. Благодаря обратной связи, я обнаружил, что некоторые темы в этой книге выиграли бы от дополнительных пояснений, что я упустил из виду некоторые востребованные темы, и что есть целые новые области, заслуживающие отдельных глав или приложений.

Я сделал свой ход, попытавшись удовлетворить распространенные требования, теперь, уважаемый читатель, ход за вами.

ГЛАВА 1. Начало...

С чего начать

Моя цель в этой книге – рассказать достаточно об устройстве Oracle, чтобы вы могли сами находить и устранять причины возникающих проблем. Это означает, что я включил в книгу описание только необходимого объема деталей, достаточного для понимания сути. Это также означает, что я опустил упоминание разного рода особенностей, механизмов и прочей любопытной информации, не имеющих большого значения в контексте обсуждения, без объяснения причин.

Слова: «рассказать достаточно об устройстве», – мало помогают в выборе отправной точки. Может быть стоило нарисовать архитектуру процесса на первой странице, чтобы вы могли получить «общее представление»? (Нет, пожалуй, потому что в действительности процессы не являются основой.) Возможно, стоило начать с управления транзакциями. Но я не могу сделать это, не рассказав о заголовках сегментов отмены и списках заинтересованных транзакций (Interested Transaction Lists, ITL), а значит разговор нужно начать с операций отмены и повторения (undo и redo), но перед этим следует рассказать о буферах и механизмах записи... Нет, определенно нужно начинать с отмены и повторения, но мне будет сложно, если предварительно ничего не скажу о транзакциях.

Ядро Oracle само по себе очень мало и включает лишь несколько механизмов, понимание которых поможет находить причины проблем, – вам даже не потребуется знать все тонкости этих базовых механизмов. К сожалению, они настолько тесно связаны, что описать их по отдельности и – весьма непростая задача. Вообще, описание Oracle чем-то напоминает транзакцию: снаружи либо видны все изменения, либо не видно ни одно из них – никакое другое промежуточное состояние недопустимо.

Я не могу говорить о согласованном чтении, не рассказав о системном номере изменения (System Change Number, SCN) и записях от-

мены (undo records). Я не могу говорить о записях отмены, не рассказав о транзакциях. Я не могу говорить о транзакциях, не рассказав о слотах ITL и SCN. И так далее, по кругу. Из этого следует, что лучший метод преподавания Oracle (и этот метод используется в данной книге) – рассказать каждую тему несколько раз, увеличивая детальность описания: сначала я расскажу немного об А, чтобы затем рассказать немного о Б; рассказав о Б, я смогу поведать вам о В; затем, рассказав о В, я смогу чуть больше рассказать об А, что даст мне возможность сообщить вам дополнительные детали о Б. В конечном итоге вы узнаете все, что действительно необходимо знать.

Oracle в процессах

На рис. 1.1 изображена упрощенная схема процесса Oracle, которой более чем достаточно для понимания. На ней изображено только самое основное, о чем рассказывается в этой книге; все остальное – лишь глазурь на торте.

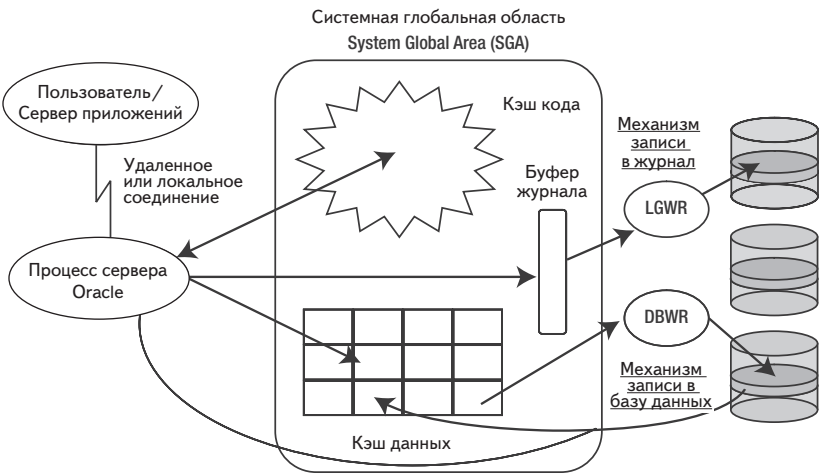

Рис. 1.1. Схема процесса Oracle, содержащая «только самое необходимое»

На рис. 1.1 показаны два типа файлов. *Файлы с данными*, хранящие «настоящие» данные, и файлы журнала повтора (redo log files, часто их называют просто файлами журнала), хранящие непрерывный поток всех изменений, производимых в файлах данных.

Файлы с данными поддерживают произвольный доступ, а для большей эффективности, каждому из них назначается размер единицы ввода/вывода – *размер блока* – который может быть равен 2 Кбайта, 4 Кбайта, 8 Кбайт (наиболее типичное значение по умолчанию), 16 Кбайт или (на некоторых платформах) 32 Кбайта. Файлы с данными могут объединяться в логические объекты, которые называют *табличными пространствами* (tablespace). Табличное пространство можно рассматривать, как естественную «крупномасштабную» единицу базы данных – простые объекты данных связываются с табличными пространствами, а не с файлами данных. Существует три основных типа табличных пространств, которые встретятся далее в книге: *табличные пространства отмены* (undo tablespaces), *временные табличные пространства* (temporary tablespaces) и «все остальное».

Временные табличные пространства появилось в версии Oracle 8, а табличные пространства отмены – в Oracle 9. До этого (начиная с версии 6, когда вообще появились табличные пространства) все табличные пространства были одинаковыми. Среди «всех остальных» имеется несколько табличных пространств, считающихся специальными (даже при том, что они интерпретируются так же, как другие табличные пространства): системное табличное пространство *system* и вспомогательное системное табличное пространство *sysaux*, которые не должны использоваться для хранения пользовательских данных. Табличное пространство sysaux появилось в версии Oracle 10*g* и служит для хранения наиболее динамических и потенциально объемных данных, сгенерированных внутренними механизмами управления и обслуживания. Табличное пространство system служит для хранения словаря данных (data dictionary) – метаинформации, описывающей базу данных.

Файлы журналов поддерживают последовательный ввод/вывод, и для них назначается минимальный размер блока, обычно 512 байт, для записи. Некоторые файлы журналов называются *оперативными* файлами журналов повтора (online redo log files) и находятся в постоянном использовании. Остальные называются *архивными* файлами журналов повтора (archived redo log files) и являются простыми копиями оперативных файлов журналов, которые создаются по мере их заполнения.

Примечание. *Разумеется, существуют и другие типы файлов, но они не будут рассматриваться в этой книге. Лишь в главе 6 будет сделано несколько замечаний, касающихся управляющих файлов (control file).*

Когда программное обеспечение выполняется под управлением ОС UNIX (и во многих других ОС), в памяти создается несколько копий одного и того же процесса, и эти копии совместно используют значительный сегмент памяти. В Windows создается единственный процесс с именем `oracle`, в рамках которого выполняется множество независимых *потоков*. В последнем случае немного проще представить потоки, совместно использующие сегмент памяти. Формально, файлы с данными называют *базой данных* (database), а комбинацию памяти и действующую программу (или программы) – *экземпляром* (instance). При использовании кластеризованной версии Real Application Clusters (RAC) можно настроить несколько компьютеров так, чтобы на каждом выполнялся отдельный экземпляр, но все они совместно использовали одну базу данных.

Сегмент совместно используемой памяти (формально: *системная глобальная область* (System Global Area), иногда ее называют *разделяемой глобальной областью* (Shared Global Area), но чаще просто используют аббревиатуру *SGA*) хранит массу разнообразной информации. Самыми важными хранимыми компонентами являются: *кэш данных* (окно в файлы с данными, хранящее копии некоторых блоков), *буфер журнала* (очень небольшой фрагмент памяти, используемый как циклический буфер для хранения информации, которая в скором времени будет записана в файлы журналов) и *кэш библиотек* (хранит информацию об инструкциях SQL и блоках PL/SQL, выполнявшихся последними). Формально, кэш библиотек является частью *разделяемого пула* (shared pool), но это слишком широкий термин и, к тому же, он часто применяется для обозначения любой памяти в SGA, используемой в текущий момент.

Примечание. *Существует еще несколько не менее важных компонентов, а именно: пул потоков данных (streams pool), Java-пул и большой пул (large pool). Но все они являются обычными областями памяти, изолированными от разделяемого пула и предназначенными для поддержки специализированных механизмов. Если вы сможете разобраться с разделяемым пулом, вы без труда разберетесь и с другими пулами.*

В SGA имеется сегмент, заслуживающий отдельного упоминания: «часы» (clock), используемые экземплярами для координации действий. Это простой счетчик, который называется *системным номером изменения* (System Change Number, SCN) иногда его называют (не совсем правильно) *системным номером подтверждения транзакции* (System Commit Number). Все процессы, имеющие доступ к SGA, могут читать и изменять SCN. Обычно процессы читают текущее

значение в начале каждого запроса или транзакции (с помощью подпрограммы *kcmgss* – Get Snapshot SCN), и каждый раз, когда процесс подтверждает транзакцию, он увеличивает значение SCN (с помощью подпрограммы *kcmgas* – Get and Advance SCN). Значение SCN увеличивается также в других случаях, именно поэтому название «системный номер изменения» (System Change Number) лучше соответствует его сути, чем название «системный номер подтверждения транзакции» (System Commit Number).

Теперь остаются всего три процесса (точнее, три типа процессов) и один важный факт, которые вы должны знать. Важный факт: программы конечного пользователя не взаимодействуют напрямую ни с файлами данных, ни даже с разделяемой памятью.

Существует отдельный процесс, копирующий информацию из буфера журнала в файлы. Его так и называют – процесс записи в журналы (log writer, известный также как *lgwr*). Каждый экземпляр имеет только один процесс *lgwr*. Аналогично существует отдельный процесс, копирующий информацию из кэша в файлы данных. Это – процесс записи в базу данных (database writer, известный также как *dbwr*). Часто экземпляры имеют только один такой процесс, но в очень больших и высоконагруженных системах возможно (а часто и необходимо) обеспечить запуск нескольких процессов записи в базу данных, которые получат имена *dbwN* (где диапазон возможных значений *N* отличается для разных версий Oracle).

Наконец, в каждом экземпляре существует несколько копий серверных процессов. Эти процессы выполняют операции с SGA и читают файлы с данными от имени конечного пользователя. Программы конечных пользователей передают инструкции и принимают результаты через конвейер *SQL*Net*. Администратор базы данных (то есть, вы!) может выбирать в настройках между двумя типами серверных процессов: *выделенные* (dedicated) серверные процессы и *разделяемые* (shared, прежде их называли *многопоточными* (multithreaded)). На практике чаще используются выделенные серверы, но в некоторых системах большинство легковесных задач решается с помощью разделяемых серверов, а более тяжеловесные задачи – с помощью выделенных серверов.

Oracle в действии

Что действительно нужно знать о работе Oracle? В конечном счете все сводится к следующему:

Конечный пользователь отправляет запросы в форме инструкций SQL (или PL/SQL) серверному процессу; каждая инструкция интерпретируется и выполняется; процесс выбирает нужные данные; процессу может потребоваться изменить данные, не нарушая их целостность; экземпляр предпринимает все меры по защите базы данных от повреждений.

Вся эта работа выполняется в контексте многопользовательской системы, где множество конечных пользователей пытаются одновременно манипулировать одними и теми же данными. Вследствие этого возникает несколько важных вопросов: «Как наиболее эффективно читать данные?», «Как наиболее эффективно записывать данные?», «Как защитить базу данных?», «Как минимизировать конфликты между пользователями?» и «Когда база данных развалится, можно ли будет собрать ее обратно?».

В заключение

В следующих главах мы постепенно выясним, как Oracle решает проблемы *эффективности* и *параллельного выполнения*. Мы начнем с простых изменений данных и механизмов, которые используются в Oracle для записи и применения изменений, а затем исследуем порядок объединения изменений в транзакции. По мере знакомства с этими механизмами, мы также узнаем, как они решают проблемы конкурентного доступа, и немного коснемся некоторых проблем, возникающих из-за отсутствия временных ограничений на выполнение операций в Oracle.

Затем последует предварительное обсуждение типичных для Oracle структур в памяти, и механизмов защиты разделяемой памяти от опасных параллельных изменений. Опираясь на эту информацию, мы перейдем к исследованию механизмов поиска данных в памяти и чтения данных из дисковых файлов в память.

После этого мы сможем обсудить другие механизмы передачи данных – записи из памяти в файлы – и параллельно узнаем, как Oracle отслеживает данные в памяти. Уделив значительную часть времени обработке данных, мы затем посмотрим, как Oracle обрабатывает код запросов (SQL) и узнаем, насколько механизмы обработки кода похожи на механизмы обработки данных, даже при том, что код запросов не имеет ничего общего с данными.

В заключение мы быстренько пройдемся по кластеризованной версии (RAC). Выясним, какие проблемы возникают из-за необходимости синхронизировать работу нескольких экземпляров на разных компьютерах.