2

Архитектура Oracle

Эта глава посвящена концепциям и структурам, относящимся к ядру СУБД Oracle. Разобравшись в архитектуре сервера Oracle, вы заложите фундамент для понимания остальных средств, описываемых в этой книге.

СУБД Oracle состоит из физических и логических компонентов. В первом разделе мы поговорим о различии понятий базы данных и экземпляра, а в последующих опишем физические компоненты, экземпляры и словарь данных.

Базы данных и экземпляры

Многие пользователи Oracle употребляют термины *экземпляр* и *база данных* как синонимы. На самом деле это разные (хотя и взаимосвязанные) вещи. Различие существенно, так как проливает свет на архитектуру Oracle.

В Oracle термином *база данных* описывается физическое хранилище информации, а термином *экземпляр* – программное обеспечение, работающее на сервере и предоставляющее доступ к информации в базе данных. Экземпляр исполняется на конкретном компьютере или сервере; база данных хранится на дисках, подключенных к этому серверу. Эта взаимосвязь изображена на рис. 2.1.

База данных – *физическая* сущность: она состоит из файлов, хранящихся на дисках. Экземпляр – сущность *логическая*: он состоит из структур в оперативной памяти и процессов, работающих на сервере. Например, Oracle использует область разделяемой памяти System Global Area (SGA, системная глобальная область) и области памяти в каждом процессе – Program Global Area (PGA, программная глобальная область). Экземпляр может быть частью одной и только одной базы данных. Напротив, с одной базой данных может быть ассоциировано не-

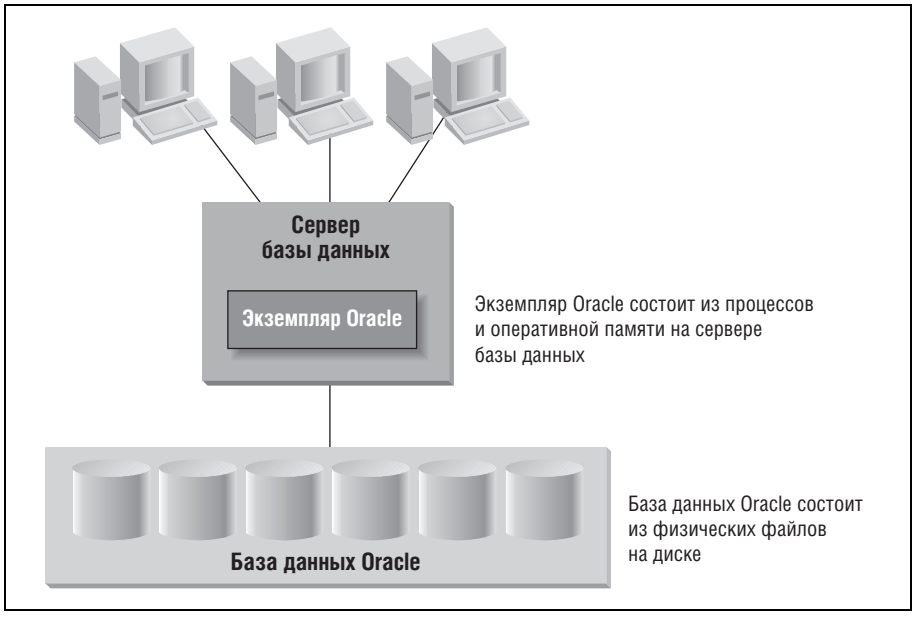

Рис. 2.1. *Экземпляр и база данных*

сколько экземпляров. Время жизни экземпляров ограничено, тогда как база данных при должном обслуживании может существовать вечно.

Пользователи не имеют прямого доступа к информации, хранящейся в базе данных Oracle; они должны запрашивать информацию у экземпляра Oracle.

В реальном мире есть хорошая аналогия экземплярам и базам данных. Можно считать экземпляр мостом к базе данных, а саму ее – островом. Транспорт попадает на остров и уходит с него по мосту. Если мост перекрыт, то остров на месте, но транспорту туда не попасть. В терминологии Oracle, если экземпляр запущен, то данные могут попадать в базу и уходить из нее. Физическое состояние базы данных при этом изменяется. Если же экземпляр остановлен, то пользователи не могут обращаться к базе данных, пусть даже физически она никуда не делась. База данных в этом случае статична, никаких изменений в ней не происходит. Экземпляр снова запущен – и данные тут как тут.

Структура базы данных Oracle

База данных состоит из табличных пространств, управляющих файлов, журналов, архивных журналов, файлов трассировки изменения блоков, ретроспективных журналов и файлов резервных копий (RMAN). В этом разделе мы познакомимся со многими из этих струк-

тур, а также с другими компонентами, составляющими в совокупности базу данных.

Табличные пространства

Любые данные, хранящиеся в базе Oracle, должны находиться в каком-то табличном пространстве. *Табличное пространство* (tablespace) – это логическая структура; нельзя попросить операционную систему показать вам табличное пространство. Каждое табличное пространство состоит из физических структур, называемых *файлами данных* (datafiles). В одном табличном пространстве может быть один или несколько файлов данных, тогда как каждый файл данных принадлежит ровно одному табличному пространству. При создании таблицы можно указать, в какое табличное пространство ее поместить. Тогда Oracle найдет для нее место в одном из файлов данных, составляющих указанное табличное пространство.

На рис. 2.2 показано соотношение между табличными пространствами и файлами данных.

Здесь мы видим два табличных пространства в базе данных Oracle. При создании новой таблицы ее можно поместить в табличное пространство DATA1 или DATA2. Физически таблица окажется в одном из файлов данных, составляющих указанное табличное пространство.

Начиная с версии Oracle Database 10*g* Release 2 для всех типов таблиц по умолчанию подразумеваются *локально управляемые табличные пространства*. В таком табличном пространстве можно создавать *большие файлы*, то есть при работе в 64-разрядных системах задействуется возможность создавать сверхбольшие файлы.

В Oracle9*i* появился механизм файлов, управляемых Oracle (Oracle Managed Files, OMF), позволяющий автоматически создавать, имено-

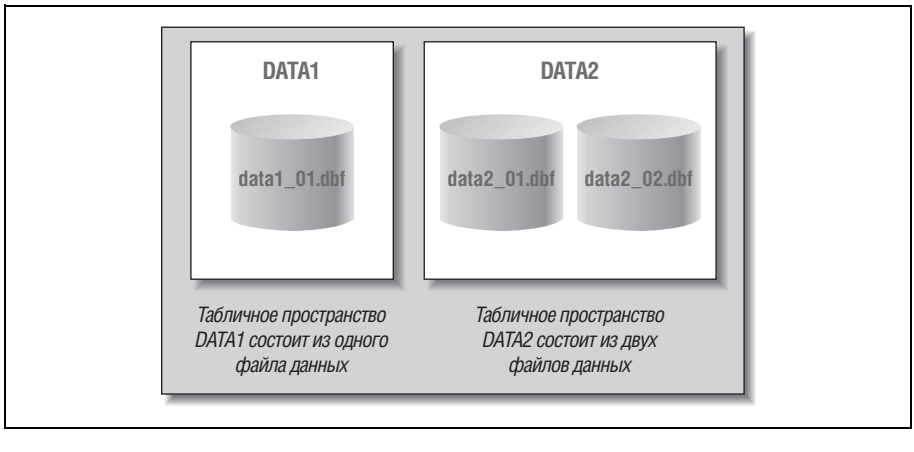

***Рис. 2.2.** Табличные пространства и файлы данных*

вать и, если понадобится, удалять все файлы, составляющие базу данных. OMF упрощает обслуживание базы данных, поскольку не нужно помнить имена всех составляющих ее файлов. К тому же не возникают проблемы из-за ошибок человека, ответственного за именование файлов. Начиная с версии Oracle Database 10*g* сочетание OMF и табличных пространств с большими файлами делает работу с файлами данных совершенно прозрачной.

Максимальное количество файлов данных в базе Oracle – 64 000. Поскольку табличное пространство с большими файлами может содержать файл, который в 1024 раза больше файла в табличном пространстве с малыми файлами, а размер блока в табличном пространстве с большими файлами для 64-разрядных операционных систем составляет 32 Кбайт, общий размер базы данных Oracle может достигать 8 экзабайт (1 экзабайт = 1 000 000 терабайт).[1] Табличные пространства с большими файлами предназначены для использования совместно с подсистемой автоматического управления хранением Automatic Storage Management (ASM), иными менеджерами логических томов, поддерживающими расслоение, и RAID-массивами.[2]

Файлы базы данных

База данных Oracle состоит из физических файлов трех основных типов:

- управляющие файлы (control files);
- файлы данных (datafiles);
- журнальные файлы, или журналы (redo log files).

На рис. 2.3 показаны эти три типа файлов и отношения между ними.

В управляющем файле хранится информация о местонахождении других физических файлов, составляющих базу данных, – файлов данных и журналов. Там же хранится важнейшая информация о содержимом и состоянии базы данных:

- имя базы данных;
- время создания базы данных;
- имена и местонахождение файлов данных и журнальных файлов;
- информация о табличных пространствах;
- информация о файлах данных в автономном режиме;
- история журналов и информация о порядковом номере текущего журнала;
- информация об архивных журналах;

---

[1] Максимальный размер большого файла зависит от ограничений, налагаемых операционной системой.

[2] RAID (Redundant Array of Inexpensive Disks) – массив недорогих дисковых накопителей с избыточностью. Эта тема обсуждается в главе 7.

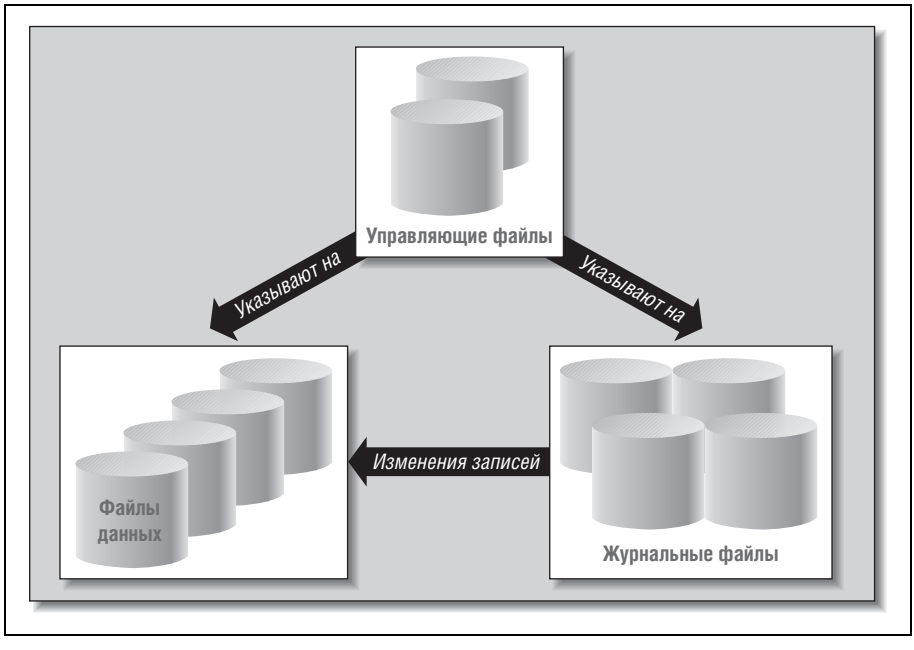

Рис. 2.3. *Файлы, составляющие базу данных*

- информация о наборах и фрагментах резервных копий, файлах данных и журналах;
- информация о копиях файлов данных;
- информация о контрольных точках.

Управляющие файлы не только содержат важную информацию, необходимую при запуске экземпляра, они полезны и при удалении базы данных. Начиная с версии Oracle Database 10*g* с помощью команды DROP DATABASE можно удалить все файлы, перечисленные в управляющем файле базы данных, а также сам управляющий файл.

Инициализация базы данных

При запуске экземпляра Oracle считываются параметры инициализации. Они определяют, как база данных должна использовать физическую инфраструктуру и иную конфигурационную информацию об экземпляре. Параметры инициализации хранятся в файле параметров инициализации экземпляра, который обычно называют просто *INIT.ORA*, или, начиная с версии Oracle9*i*, в репозитории, который называется файлом параметров сервера (или *SPFILE*). Количество обязательных параметров инициализации уменьшается с выходом каждой новой версии Oracle. В дистрибутиве Oracle есть пример файла инициализации, пригодный для запуска базы данных. Либо можно воспользо-

ваться программой Database Configuration Assistant (DCA), которая подскажет обязательные значения (например, имя базы данных).

Вот обязательные параметры инициализации для версии Oracle Database 11*g*:

CONTROL_FILES

Местонахождение управляющих файлов.

DB_NAME

Локальное имя базы данных.

DB_DOMAIN

Имя домена базы данных (например, us.companyname.com).

LOG_ARCHIVE_DEST

Местонахождение архивного журнала.

LOG_ARCHIVE_DEST_STATE

Параметр, включающий архивирование журналов.

DB_RECOVERY_FILE_DEST

Местонахождение области быстрого восстановления (flash recovery area) (каталог, файловая система или группа дисков ASM).

DB_RECOVERY_FILE_DEST_SIZE

Максимальный размер области быстрого восстановления базы данных в байтах.

DB_BLOCK_SIZE

Размер блока базы данных в байтах (например, для 4 Кбайт указывается значение 4096).

PROCESSES

Максимальное число процессов операционной системы, обслуживающих одновременный доступ к базе данных.

SESSIONS

Максимальное число сеансов работы с базой данных.

OPEN_CURSORS

Максимальное число открытых в базе данных курсоров.

SHARED_SERVERS

Минимальное число разделяемых серверов базы данных.

REMOTE_LISTENER

Имя удаленного прослушивателя.

COMPATIBLE

Версия базы данных, с которой должна поддерживаться совместимость, в тех случаях, когда то или иное средство затрагивает формат файла (например, 11.1.0, 10.0.0).

MEMORY_TARGET

Размер области памяти, автоматически выделяемой для SGA и PGA экземпляра.

DDL_LOCK_TIMEOUT

Для команд языка определения данных (DDL) – время (в секундах) ожидания возможности установить монопольную блокировку, прежде чем сообщить об ошибке.

NLS_LANGUAGE

Язык, определенный в подсистеме поддержки национальных языков (National Language Support, NLS) для базы данных.

NLS_TERRITORY

Территория, определенная в подсистеме поддержки национальных языков для базы данных.

В качестве признака взятого курса на автоматизацию отметим, что в версии Oracle Database 11*g* параметр UNDO_MANAGEMENT по умолчанию устанавливается в режим автоматического управления откатом (undo). Механизм отката применяется при откате транзакций, а также для восстановления базы данных, обеспечения согласованности по чтению и реализации ретроспекции. (Однако записи о повторном выполнении располагаются в физических журналах повтора, или наката, redo log; в них хранятся изменения, произведенные в сегментах данных и блоках сегментов отката, там же хранится таблица транзакций для сегментов отката.) Время хранения информации для отката Oracle теперь подбирает автоматически, исходя из того, как сконфигурировано табличное пространство отката.

Изучите поставляемую с вашей версией СУБД документацию в части дополнительных параметров инициализации, поскольку эта информация изменяется от версии к версии. Некоторые параметры описаны в следующих разделах.

Развертывание физических компонентов

Этот раздел не может заменить официальную документацию по установке Oracle. Наша цель – снабдить вас практическим руководством по планированию развертывания базы данных Oracle.

Управляющие файлы

Для любой базы данных следует хранить по меньшей мере два управляющих файла на разных физических дисках. Без актуальной копии управляющего файла вы рискуете потерять информацию о составляющих вашей базы данных. Утрата управляющих файлов не обязательно фатальна, их можно и воссоздать. Но процедура воссоздания довольно сложна и рискованна, а избежать ее несложно.

Местоположение управляющих файлов определяется, как уже было сказано, параметром инициализации CONTROL_FILES. Он позволяет задать несколько управляющих файлов, например:

```
control_files = (/u00/oradata/control.001.dbf,
            /u01/oradata/control.002.dbf,
            /u02/oradata/control.003.dbf)
```

Этот параметр сообщает экземпляру, где искать управляющие файлы. Oracle гарантирует, что все копии управляющего файла одинаковы, то есть любые изменения вносятся синхронно. Если параметр не задан, Oracle создаст управляющий файл с именем по умолчанию или прибегнет к услугам компонента Oracle Managed Files (если тот активирован).

Многие базы данных Oracle развертываются на том или ином варианте RAID-массива, например RAID-1 или RAID-5, чтобы избежать потери данных в случае выхода диска из строя. (Технология RAID более подробно рассматривается в главе 7.) Напрашивается вывод, что можно обойтись без нескольких копий, сохранив управляющий файл в защищенной дисковой памяти, и что утрата диска еще не означает утраты управляющего файла. Но этот вывод неправомерен по двум причинам:

1. Если в *массиве с расслоением* (striped array) или в *зеркальной паре* (mirror-pair) отказывает больше одного диска, то все данные, хранящиеся на этих дисках, теряются. Статистически это редкое событие, но все же такое случается, и тогда есть угроза повреждения или утраты управляющего файла. Поскольку вы и так будете по горло заняты восстановлением после множественных сбоев диска, вероятно, лучше при этом избежать хотя бы воссоздания управляющих файлов. Создание дополнительных копий, пусть даже хранящихся в избыточной дисковой памяти, – это дополнительный уровень защиты.
2. Избыточная дисковая память не поможет защититься от человеческих ошибок. Кто-то может случайно удалить или переименовать управляющий файл, затереть его другим или переместить в другое место. И зеркалированный диск честно отразит эти изменения. А при резервировании управляющих файлов хотя бы одна копия да останется.

Не стоит беспокоиться о том, что запись в несколько управляющих файлов понизит производительность. Обновление управляющих файлов – ничто по сравнению с другими операциями дискового ввода/вывода, производимыми Oracle.

Файлы данных

В файлах данных находятся собственно данные, хранящиеся в базе: таблицы и индексы, словарь данных, в котором сохраняется информация об этих структурах, и сегменты отката, необходимые для реализации конкурентного доступа.

Файл данных состоит из блоков базы данных, в свою очередь, состоящих из дисковых блоков операционной системы. Размер блока в Oracle варьируется от 2 до 32 Кбайт. До выхода версии Oracle9*i* база данных состояла из блоков одного размера. В более поздних версиях вы по-прежнему можете задать размер блока базы по умолчанию, но вообще-то в базе данных может использоваться до пяти разных размеров блоков (хотя в каждом табличном пространстве возможен только один размер блока). На рис. 2.4 показано соотношение блоков Oracle с блоками операционной системы.

Каждый файл данных принадлежит только одной базе данных и только одному табличному пространству в ней. Данные считываются с дисков в оперативную память блоками Oracle по мере необходимости, исходя из действий пользователей. Блоки данных переписываются из памяти в файлы данных на диске, когда это требуется для гарантии сохранности внесенных пользователями изменений.

Файлы данных – это самый низкий уровень гранулярности взаимодействий между Oracle и операционной системой. При размещении базы данных на физических устройствах наименьшая сущность, которую можно куда-то поместить, – это файл. Оптимизация подсистемы ввода/вывода для повышения производительности Oracle обычно включает перемещение файлов данных с одного набора дисков на другой. Подсистема Automatic Storage Management, входящая в версии начиная с Oracle Database 10*g*, обеспечивает автоматическое расслоение вместо ручной настройки этого аспекта работы.

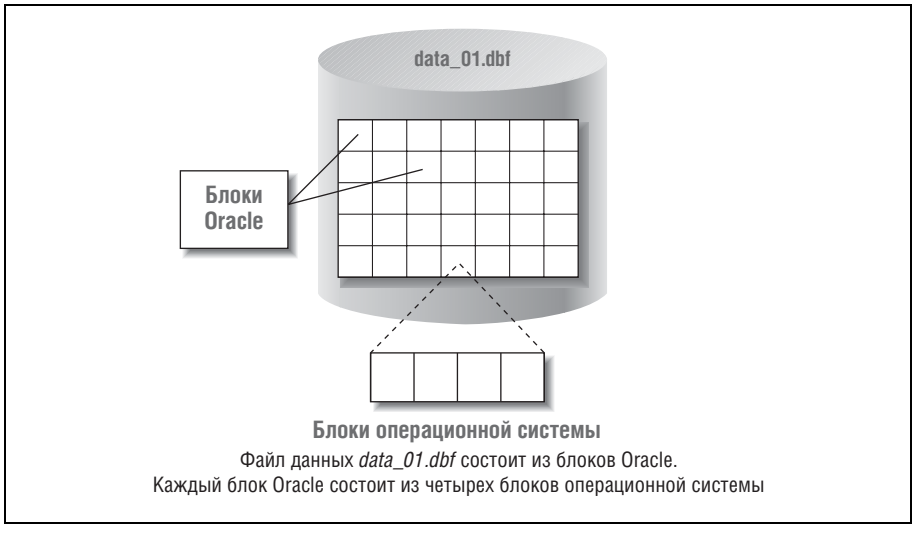

***Рис. 2.4.** Блоки Oracle и блоки операционной системы*

Задание размера блока базы данных

До выхода версии Oracle9*i* размер блока задавался в момент создания базы данных, а изменить его, не создавая базу заново, было невозможно. В Oracle9*i* появилась дополнительная гибкость – в одной и той же базе можно использовать блоки разного размера. Во всех версиях по умолчанию устанавливается размер блока, заданный в параметре инициализации экземпляра DB_BLOCK_SIZE.

Как выбрать правильный размер блока? По умолчанию размер блока Oracle совпадает с размером блока операционной системы. Однако зная, что именно зависит от размера блока, вы сможете подобрать размер, более подходящий для ожидаемой рабочей нагрузки.

Размер блока – это минимальный размер порции данных, считываемой или записываемой за один раз. В системах оперативной обработки транзакций (OLTP) типичная транзакция затрагивает относительно немного строк, например, строки размещения заказа от конкретного клиента на закупку ряда товаров. При этом доступ к строкам обычно производится с помощью индексов, а не путем сканирования всей таблицы. Поэтому вполне может хватить блоков небольшого размера (4 Кбайт). Тогда Oracle не будет зря расходовать системные ресурсы на передачу больших блоков, содержащих ненужные для транзакции данные.

В случае хранилищ данных запрос может потребовать считывания миллионов строк и сканирования всех данных в таблице. Для таких операций задание большого блока позволяет за один раз считать больше данных, необходимых пользователю. Поэтому для хранилищ данных обычно используются блоки размером 8 или 16 Кбайт. При этом каждая операция ввода/вывода занимает чуть больше времени, но это с лихвой окупается уменьшением общего числа операций.

Структура файла данных

Первый блок файла данных называется *заголовком файла данных*. В нем хранится важная информация, необходимая для поддержания целостности базы данных, в частности *структура контрольной точки*. Она представляет собой логическую временную метку, показывающую, когда в последний раз были записаны изменения в файл данных. Эта информация абсолютно необходима для процесса восстановления, поскольку определяет, какие журналы нужно накатить, чтобы привести файл данных к состоянию на текущий момент времени.

Экстенты и сегменты

С точки зрения физической организации, файл данных представляет собой набор блоков операционной системы. А с логической точки зрения, в файлах данных можно выделить три структурных уровня: блоки данных, экстенты и сегменты. *Экстентом* называется набор смежных блоков в файле данных. *Сегмент* – это объект в базе данных Oracle, например таблица или индекс, состоящий из одного или нескольких экстентов.

Выполняя операцию обновления, Oracle сначала пытается обновить данные, не выходя за пределы исходного блока. Если в этом блоке недостаточно места для новой информации, то Oracle записывает данные в новый блок, который может находиться в другом экстенте.

Дополнительные сведения о сегментах и экстентах и о том, как они влияют на производительность, см. в разделе «Фрагментация и реорганизация» главы 5. Этот материал особенно важен, если вы работаете со старыми версиями Oracle. В Oracle Database 10*g* добавлен консультант Segment Advisor, который сильно упрощает процедуру освобождения неиспользуемого пространства.

Журнальные файлы

Журнальный файл содержит протокол всех изменений, произведенных в базе данных в результате выполнения транзакций и внутренних операций Oracle. Обычно измененные блоки кэшируются в памяти, поэтому в случае сбоя экземпляра может оказаться, что какие-то блоки не записались в файлы данных. Тогда можно воспользоваться протоколом, хранящимся в журнальных файлах, чтобы воспроизвести изменения, оставшиеся не записанными в момент сбоя, и тем самым обеспечить согласованность транзакции.

Иногда эти файлы путают с буферами отката, необходимыми для поддержки конкуренции (они описаны в главе 8). Так вот, это не одно и то же!

Кроме того, журнальные файлы применяются для реализации операций отката (undo), инициируемых командой ROLLBACK. Незафиксированные изменения в базе данных откатываются, так что база остается в том состоянии, в котором находилась в момент последней фиксации.

Чтобы упростить работу в случая сбоя, рекомендуем после любой не зарегистрированной в журнале операции снимать резервную копию, если вы не можете позволить себе потерять созданный объект или по какой-то причине операцию невозможно повторить. Помимо использования слова NOLOGGING в отдельных командах, можно пометить таблицу или все табличное пространство атрибутом NOLOGGING. Тогда запись в журнал будет подавляться для всех операций с данной таблицей или с любой таблицей в помеченном табличном пространстве.

Подавление записи в журнал

По умолчанию Oracle записывает в журнал все изменения, произведенные в базе данных. Но ведение журналов сопряжено с накладными расходами. Для ускорения некоторых операций можно подавить запись в журнал, однако это означает, что вы не сможете восстановить результат этой операции в случае сбоя.

Чтобы подавить запись в журнал для какой-то операции, следует включить в соответствующую SQL-команду ключевое слово NOLOGGING. (Отметим, что в версиях до Oracle8*i* для этого применялось ключевое слово UNRECOVERABLE.) Если во время данной операции произойдет сбой, то ее нужно будет повторить. Например, индекс над таблицей можно строить без записи в журнал. Если произойдет сбой в базе данных и придется ее восстанавливать, то индекс не будет пересоздан, так как его создание не протоколировалось. Нужно будет просто еще раз выполнить сценарий, в котором создавался этот индекс.

Мультиплексирование журнальных файлов

В Oracle имеется специальная терминология для процедур управления журналами. Каждый экземпляр Oracle использует свою *цепочку журнальных файлов* (thread) для записи в журнал изменений, произведенных в базе. Цепочка журнальных файлов состоит из журнальных групп, в каждую из которых входит один или несколько элементов.

Логически журнальную группу можно представлять себе как единый журнальный файл. Однако Oracle позволяет определить несколько копий журнала, чтобы обеспечить его целостность, важность которой невозможно переоценить. Создавая несколько копий журнального файла, вы защищаете его от сбоев диска и других неприятностей.

На рис. 2.5 показана цепочка журнальных файлов с группами и элементами. В данном случае каждая группа состоит из двух элементов, причем все журналы зеркалируются.

Если в журнальной группе несколько элементов, то Oracle ведет несколько копий журнальных файлов. В пользу такого решения можно привести те же аргументы, что для мультиплексирования управляющих файлов. Однако если статическую часть утраченного управляющего файла можно воссоздать, то восстановить пропавший журнал невозможно. Поэтому обязательно поддерживайте несколько копий журнала. Одной лишь избыточности дискового массива для их защиты недостаточно, так как всегда есть риск, что кто-то испортит или удалит журнальный файл по ошибке.

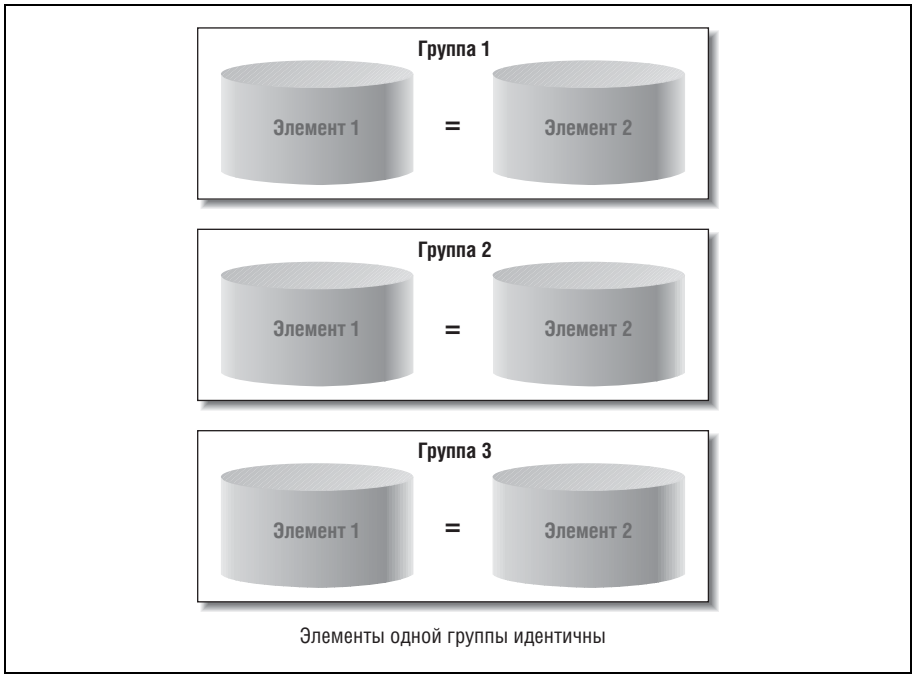

Рис. 2.5. *Цепочка журнальных файлов*

Oracle производит запись во все элементы журнальной группы *синхронно*. Запись считается завершенной лишь после того, как получено подтверждение, что все копии журнала успешно обновлены. Если одну копию поместить на быстрый или слабо загруженный диск, а другую – на медленный или активно используемый, то производительность будет ограничена скоростью более медленного диска. Oracle обязана убедиться, что все копии журнала успешно обновлены, дабы избежать потери данных.

Посмотрим, что может произойти, если запись в несколько журналов производится асинхронно, то есть сначала обновляется основной журнал, а потом – в фоновом режиме – его копии. Допустим, основной журнал поврежден при сбое системы, при этом не исключено, что обновление остальных журналов еще не завершилось. В этот момент часть зафиксированных транзакций пропала – основной журнал, в который изменения были записаны, недоступен, а в копии изменения еще не попали. Чтобы предотвратить такое развитие событие, Oracle и дожидается обновления всех копий журнала.

Как Oracle использует журнальные файлы

Заполнив один журнальный файл, Oracle автоматически переходит к следующему. Заполнив все имеющиеся журнальные файлы, Oracle

возвращается к первому и перезаписывает его. Для отслеживания журналов применяется порядковая нумерация. Порядковый номер записывается в текущий журнальный файл.

Чтобы лучше разобраться в именах журнальных файлов и их порядковых номерах, рассмотрим три таких файла с именами *redolog1.log*, *redolog2.log*, *redolog3.log*. Когда Oracle только начинает их использовать, журналам присвоены порядковые номера 1, 2 и 3 соответственно. После того как Oracle вернется к первому журналу – *redolog1.log* – и начнет перезаписывать его, журналу будет присвоен порядковый номер 4. Затем настанет очередь файла *redolog2.log*, который получит порядковый номер 5.

Помните, что операционная система идентифицирует журнальные файлы по имени, а Oracle определяет, в каком порядке заполнялись журналы, ориентируясь на порядковый номер. Поскольку журналы автоматически используются повторно, имя файла мало что говорит о его месте в последовательности.

На рис. 2.6 проиллюстрировано заполнение журналов и их циклический перебор.

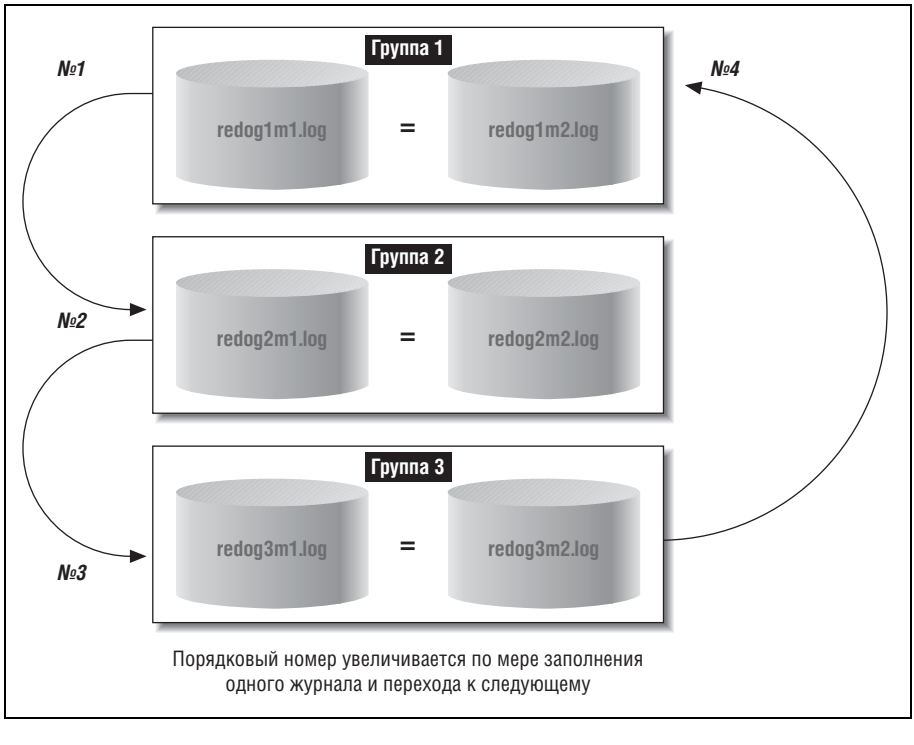

Рис. 2.6. *Циклический перебор журналов*

Соглашения об именовании журнальных файлов

Имена различных файлов, составляющих базу данных, очень важны – по крайней мере, для тех, кто порой вынужден искать файлы по именам. Если вы не пользуетесь компонентом Oracle Managed Files, то должны выработать схему именования, отражающую назначение и существенные характеристики файла. Одно из возможных соглашений о выборе имен журнальных файлов показано на рис. 2.6:

```
redog1m1.log, redog1m2.log, ...
```

Префикс `redo` и суффикс `.log` отражают тот факт, что файл содержит журнал. В строках `g1m1` и `g1m2` закодирована информация о номерах группы и элемента. Это всего лишь пример, но мы рекомендуем принять какое-то соглашение, которое кажется вам осмысленным, и строго придерживаться его.

Архивные журнальные файлы

Возможно, вас интересует, как избежать потери критически важной информации, коль скоро Oracle возвращается к ранее записанному журналу.

Есть два подхода к этой проблеме. Первый чрезвычайно прост: вы даже не пытаетесь избежать утраты информации в случае сбоя – со всеми вытекающими последствиями. При перезаписи журнала хранящаяся в нем история изменений теряется. Если в результате сбоя будут повреждены файлы данных, то можно будет восстановить всю базу данных в состоянии на момент последнего резервного копирования. Но воспроизвести изменения, внесенные позже последнего резервного копирования, без журнала не удастся. Таким путем идут очень немногие компании, работающие с Oracle, ведь невозможность восстановить ситуацию на момент сбоя недопустима – в результате теряются данные.

Второе (и более распространенное) решение проблемы заключается в архивировании заполненных журналов. Для того чтобы понять, что такое архивирование журналов, необходимо знать, что Oracle на самом деле поддерживает два типа журналов:

Оперативные журналы

Файлы операционной системы, в которые Oracle последовательно записывает изменения, произведенные в базе данных, циклически перебирая файлы.

Архивные журналы

Копии заполненных оперативных журналов, создаваемые во избежание потери данных при перезаписи оперативных журналов.

СУБД Oracle может работать в одном из двух режимов архивирования журналов:

NOARCHIVELOG

Как видно из названия, в этом режиме журналы не архивируются. По мере циклического перебора заполненные журналы повторно инициализируются и перезаписываются, при этом история изменений базы данных стирается. Именно о таком режиме шла речь выше: в этом случае сбой приводит к невосстановимой потере данных.

Тем, кто выбирает работу без архивирования журналов, будет доступно значительно меньшее количество вариантов и параметров резервного копирования базы данных (см. главу 11). Корпорация Oracle не рекомендует этот режим.

ARCHIVELOG

Переходя к новому журналу, Oracle архивирует предыдущий. Для того чтобы не допустить появления пропусков в истории изменений, повторное заполнение журнала начинается только после его успешного архивирования. Архивные и оперативные журналы содержат полную историю изменений, произведенных в базе данных. В совокупности они позволяют серверу восстановить все зафиксированные транзакции вплоть до момента сбоя. При работе в этом режиме возможно резервное копирование табличных пространств и отдельных файлов данных.

Использовать оперативные и архивные журналы на этапе восстановления базы данных серверу помогают вышеупомянутые внутренние порядковые номера.

Режим ARCHIVELOG и автоматическое архивирование

Начиная с версии Oracle Database 10*g* можно перевести БД в режим ARCHIVELOG командой:

```
ALTER DATABASE ARCHIVELOG
```

Если база данных работает в режиме ARCHIVELOG, то, заполнив очередной журнал, Oracle помечает его как подлежащий архивированию. Прежде чем заполненный журнал можно будет использовать повторно, его необходимо архивировать. Команда ALTER DATABASE ARCHIVELOG по умолчанию включает режим автоматического архивирования и запускает процессы-архиваторы.

До выхода версии Oracle Database 10*g* пометка файла журнала как готового к архивированию еще не означала, что он будет архивирован автоматически. Необходимо было также задать параметр в файле инициализации:

```
LOG_ARCHIVE_START = TRUE
```

Только в этом случае запускался процесс копирования заполненного оперативного журнала в архивный.

Местоположение и формат имен архивных журналов определяются еще двумя параметрами: LOG_ARCHIVE_DEST и LOG_ARCHIVE_FORMAT. Например, строка

```
LOG_ARCHIVE_DEST = C:\ORANT\DATABASE\ARCHIVE
```

определяет, в какой каталог Oracle будет записывать архивные журналы, а строка

```
LOG_ARCHIVE_FORMAT = ORCL%t_%s_%r.arc
```

описывает формат имен файлов архивных журналов. В данном случае имена начинаются с префикса ORCL и заканчиваются суффиксом .arc. В форматной строке распознаются следующие спецификаторы:

%t

Включать в имя файла номер цепочки журнальных файлов.

%s

Включать в имя файла порядковый номер журнала.

%r

Включать в имя файла идентификатор сброса порядкового номера журнала.

Если вы хотите, чтобы номер цепочки, порядковый номер журнала и идентификатор сброса номеров в именах архивных журнальных файлов дополнялись нулями, то следует записывать спецификаторы заглавными буквами, например:

```
LOG_ARCHIVE_FORMAT = "ORCL%T_%S_%R.arc"
```

Поскольку файл инициализации считывается в момент запуска экземпляра Oracle, любые изменения вступят в силу только после останова и перезапуска экземпляра. Однако не забывайте, что включение режима автоматического архивирования еще не переводит базу данных в режим ARCHIVELOG. И обратно, перевод базы данных в режим ARCHIVELOG не включает режим автоматического архивирования.

Следует также убедиться, что в файловой системе достаточно места для размещения архивных журналов. Если эта файловая система переполнится, то Oracle зависнет, не имея возможности архивировать журнальные файлы.

На рис. 2.7 показан порядок использования журналов, когда архивирование включено.

Архивные журналы исключительно важны для восстановления базы данных. Как и для оперативных, для архивных журналов можно задать несколько мест расположения. В этом случае Oracle будет копировать заполненные журналы в указанные места. Можно также указать, следует ли ожидать удачного завершения копирования всех журналов. Эта функциональность управляется следующими параметрами инициализации.

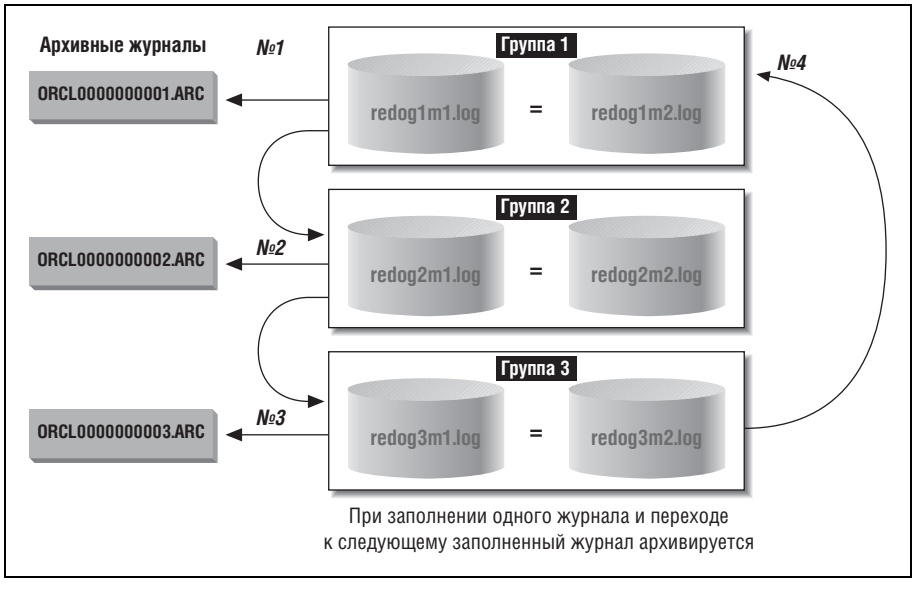

Рис. 2.7. *Циклический перебор журналов с архивированием*

LOG_ARCHIVE_DUPLEX_DEST

Задает дополнительные места для копий архивных файлов.

LOG_ARCHIVE_MIN_SUCCEED_DEST

Определяет, должно ли успешно завершиться копирование всех или хотя бы нескольких копий заполненного журнала. Допустимы значения от 1 до 10, если используется мультиплексирование, и 1 или 2 в случае дуплексного режима.

Дополнительные параметры и представления, управляющие описанной функциональностью, описаны в документации Oracle.

Память и процессы экземпляра

Экземпляр Oracle можно определить как область разделяемой памяти и набор фоновых процессов. Область разделяемой памяти экземпляра называется *системной глобальной областью* (System Global Area, SGA). Фактически SGA является не одной большой однородной областью памяти, а состоит из различных компонентов, описанных в следующем разделе. Все процессы экземпляра, как системные, так и пользовательские, совместно обращаются к SGA.

До версии Oracle9*i* размер SGA устанавливался при запуске экземпляра Oracle. Единственным способом изменения размера SGA или какой-то ее составляющей было изменение соответствующих параметров инициализации, остановка и перезапуск экземпляра. В Oracle9*i* мож-

но изменять размер SGA и ее компонентов, не останавливая экземпляр. В Oracle9*i* также введено понятие *гранулы*, то есть наименьшего объема памяти, который можно добавить или удалить из SGA.

В версии Oracle Database 10*g* появился механизм автоматического управления разделяемой памятью (Automatic Shared Memory Management, ASMM), а в Oracle Database 11*g* – механизм автоматического управления памятью (Automatic Memory Management, AMM) для компонентов SGA и PGA. Если задан параметр инициализации MEMORY_TARGET (появился в Oracle Database 11*g*) или SGA_TARGET, то база данных автоматически распределяет память между различными компонентами SGA, обеспечивая оптимальное управление памятью. К автоматически распределяемым компонентам относятся разделяемый пул (его размер вручную устанавливается с помощью параметра SHARED_POOL_SIZE), большой пул (LARGE_POOL_SIZE), пул Java (JAVA_POOL_SIZE), кэш буферов (DB_CACHE_SIZE) и пул Streams (STREAMS_POOL_SIZE). Параметры инициализации, относящиеся к автоматическому управлению памятью, можно задать в Oracle Enterprise Manager.

Фоновые процессы взаимодействуют с операционной системой и между собой, управляя структурами памяти экземпляра. Эти процессы также управляют собственно базой данных на диске и выполняют общие действия по обслуживанию экземпляра.

На рис. 2.8 показаны структуры памяти и фоновые процессы, рассматриваемые в следующем разделе.

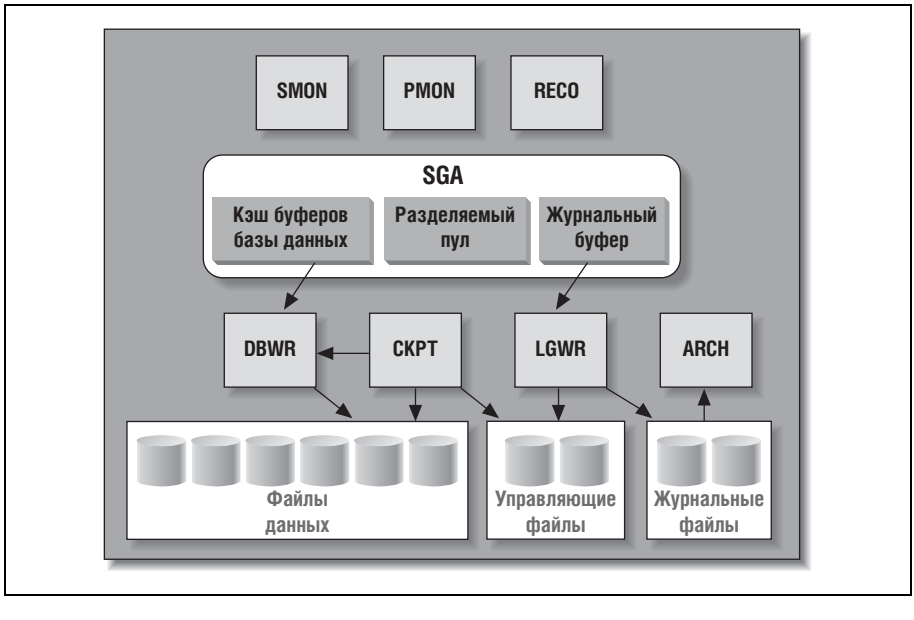

Рис. 2.8. *Экземпляр Oracle*

Если активированы еще какие-то компоненты СУБД, то могут работать и другие фоновые процессы, например: разделяемые серверы (до версии Oracle9*i* – Multi-Threaded Server, MTS), очереди заданий или репликация.

Структуры оперативной памяти экземпляра

Как видно из рис. 2.8, системная глобальная область (SGA) состоит из нескольких областей. На рисунке показаны кэш буферов базы данных, разделяемый пул и журнальный буфер. Еще могут присутствовать пул Java, большой пул и пул Streams. Все эти области описаны ниже. Более подробно вопрос о связи SGA с производительностью рассмотрен в разделе «Как Oracle использует системную глобальную область» главы 7.

Кэш буферов базы данных

В кэше буферов хранятся извлеченные из базы блоки данных. Такой буфер между пользовательскими запросами и реальными файлами данных повышает производительность СУБД Oracle. Если часть данных присутствует в кэше буферов (например, в результате недавно выполненного запроса), ее можно извлечь из оперативной памяти без затрат на обращение к диску. Oracle управляет кэшем на основе алгоритма LRU (Least Recently Used), когда выталкиваются элементы, не использовавшиеся дольше всех. Другими словами, если пользователь запрашивает данные, обращение к которым осуществлялось недавно, то с большой вероятностью они содержатся в кэше буферов и могут быть возвращены немедленно, без повторного считывания с диска.

Если же запрашивается блок, отсутствующий в кэше, то его необходимо считать и загрузить в кэш. Когда пользователь модифицирует данные в блоке, изменения сначала записываются в блок, находящийся в кэше, а спустя некоторое время сбрасываются в файл данных, которому принадлежит блок. Поэтому пользователю не нужно дожидаться завершения операции записи измененных блоков на диск.

Идея откладывания операций ввода/вывода до тех пор, пока это не станет абсолютно необходимо, красной нитью пронизывает всю СУБД Oracle. Диски – самый медленный компонент вычислительной системы, поэтому чем меньше объем ввода/вывода, тем быстрее работает система. Откладывая выполнение некритичных операций ввода/вывода, Oracle достигает более высокой производительности.

Начиная с версии Oracle8 кэш буферов базы данных может состоять из пулов буферов следующих типов:

DEFAULT

Стандартный кэш буферов базы данных Oracle. Если не оговорено иное, именно этот кэш используется всеми объектами.

KEEP

Для часто используемых объектов, которые хотелось бы кэшировать.

RECYCLE

Для объектов, к которым вы вряд ли обратитесь вновь.

Из пулов буферов KEEP и RECYCLE объекты выталкиваются на основе алгоритма LRU.

Можно указать, что некоторая таблица или индекс должны кэшироваться в определенном пуле буферов. Это обеспечивает хранение наиболее востребованных объектов и предотвращает борьбу всех объектов за место в одном центральном кэше. Разумеется, чтобы задействовать эту возможность, нужно хорошо знать характер доступа к различным объектам со стороны приложения.

В версии Oracle Database 10*g* управление размером кэша буферов упростилось за счет появления нового динамического параметра DB_CACHE_SIZE. Он позволяет задать объем памяти, отведенной под кэш, и заменяет прежний параметр DB_BLOCK_BUFFERS. Значение DB_CACHE_SIZE выбирается автоматически, если задан параметр MEMORY_TARGET или SGA_TARGET. Из прочих параметров упомянем DB_KEEP_CACHE_SIZE и DB_RECYCLE_CACHE_SIZE; если они используются, то их значения должны задаваться вручную.

Разделяемый пул

В разделяемом пуле кэшируются различные конструкции, которые могут использоваться совместно. Например, предъявляемые пользователями SQL-запросы и их фрагменты, а также результаты их выполнения кэшируются, чтобы их можно было повторно использовать в случае, если такой же запрос будет предъявлен еще раз. Написанные на PL/SQL функции также загружаются в разделяемый пул, а их результаты кэшируются, опять-таки по алгоритму LRU. Начиная с версии Oracle Database 11*g* PL/SQL-функцию можно пометить таким образом, что при ее вызове с прежними параметрами результат будет извлекаться из кэша, а не вычисляться заново. Разделяемый пул используется также для кэширования информации из словаря данных Oracle, то есть метаданных, описывающих структуры и содержимое самой базы.

Можно задать значение параметра инициализации SHARED_POOL_SIZE, в противном случае размер разделяемого пула будет установлен автоматически, если задан параметр MEMORY_TARGET или SGA_TARGET. Отметим, что до версии Oracle Database 10*g* в случае недостаточного размера разделяемого пула могла выдаваться ошибка «out of memory» (недостаточно памяти). В современных же версиях Oracle автоматически подстраивает объем разделяемой памяти.

Журнальный буфер

В журнальном буфере необходимая для повторного выполнения информация кэшируется до момента записи в физические журнальные файлы, расположенные на диске. Этот буфер также повышает производительность. Oracle кэширует журнальную информацию для того, чтобы записать ее на диск в наиболее удобное время, избегая таким образом накладных расходов, сопутствующих синхронной записи журналов на диск.

Другие пулы SGA

В SGA может находиться еще несколько пулов:

Большой пул

Предназначен для буферизации ввода/вывода различных серверных процессов, в том числе применяемых для резервного копирования и восстановления. Здесь же находятся области памяти сеансов в случае, если для обработки транзакций используются разделяемые серверы и подсистема Oracle XA.

Пул Java

Отсюда выделяется память, необходимая для Java-объектов и выполнения Java-программ, в том числе данные для виртуальной Java-машины, реализованной в СУБД.

Пул Streams

Здесь хранятся сообщения из очередей Oracle Streams, буферизованные из таблиц базы данных. Отсюда же выделяется память для захвата (capture) и применения (apply).

Для этих пулов предназначены динамические параметры инициализации LARGE_POOL_SIZE, JAVA_POOL_SIZE и STREAMS_POOL_SIZE. Они устанавливаются автоматически, если задан параметр MEMORY_TARGET или SGA_TARGET.

Автоматическое управление PGA

Oracle автоматически управляет выделением памяти для программной глобальной области (PGA). В PGA находятся память сеансов и приватная область SQL. Объемом памяти можно управлять с помощью параметра инициализации PGA_AGGREGATE_TARGET. С появлением в версии Oracle Database 10*g* механизма автоматического управления PGA администрирование рабочих областей SQL заметно упростилось, а необходимость задавать ряд старых параметров инициализации отпала вовсе. Начиная с Oracle Database 11*g* выделение памяти для PGA настраивается автоматически, как и для SGA, путем задания параметра MEMORY_TARGET.

Фоновые процессы экземпляра

Чаще всего используемые фоновые процессы показаны на рис. 2.8, хотя их состав меняется от версии к версии. Ниже описаны некоторые процессы.

Процесс записи в базу данных (Database Writer, DBWn)

Записывает блоки данных из кэша буферов SGA в файлы данных, расположенные на диске. Для одного экземпляра Oracle может существовать до 20 процессов DBW, обрабатывающих ввод/вывод в различные файлы данных, отсюда и обозначение DBW*n*. Большинство экземпляров обходится одним процессом DBW. DBW записывает блоки из кэша на диск по двум основным причинам:

- Для установки контрольной точки (то есть для обновления блоков файлов данных, с тем чтобы они «догнали» журналы). Oracle записывает в журнал информацию, необходимую для повторного выполнения транзакции после ее фиксации, а позже сохраняет сами блоки. Периодически Oracle устанавливает контрольную точку для того, чтобы привести содержимое файла данных в соответствие с информацией, записанной в журнал для зафиксированных транзакций.
- Когда необходимо прочитать запрошенные пользователем блоки, а в кэше буферов не осталось свободного места. Тогда на диск сбрасываются блоки, которые дольше всего не использовались. Запись блоков именно в таком порядке позволяет минимизировать влияние отсутствия блоков в кэше на производительность.

Процесс записи в журнал (Log Writer, LGWR)

Записывает информацию из журнального буфера в SGA во все копии текущего журнального файла на диске. Пока транзакция обрабатывается, необходимая для ее повторного выполнения информация хранится в журнальном буфере в SGA. После фиксации транзакции Oracle отправляет эту информацию на постоянное хранение, вызывая процесс LGWR для ее записи на диск.

Системный монитор (System Monitor, SMON)

Отвечает за общую исправность и безопасность экземпляра. SMON восстанавливает экземпляр при запуске после сбоя. Он же координирует доступ и реализует восстановление экземпляра, когда к базе данных обращаются сразу несколько экземпляров, как бывает при работе с Real Application Clusters. SMON также приводит в порядок смежные области свободного пространства в файлах данных, объединяя их, и избавляется от пространства, используемого для сортировки строк, когда в нем уже нет необходимости.

Монитор процессов (Process Monitor, PMON)

Следит за пользовательскими процессами, обращающимися к базе данных. В случае аварийного прекращения пользовательского про-

цесса PMON отвечает за очистку оставшихся занятыми ресурсов (таких как оперативная память) и за снятие всех блокировок, установленных сбойным процессом.

Архиватор (Archiver, ARCH)

Читает заполненные файлы журнала и копирует их в один или несколько архивных журналов. Поддерживается до 10 процессов архивации, которым присваиваются имена ARC0–ARC9. По мере необходимости LGWR запускает дополнительные архиваторы в зависимости от нагрузки. Максимально разрешенное количество процессов архивации задается параметром инициализации LOG_ARCHIVE_MAX_PROCESSES. По умолчанию этот параметр равен 2, необходимость изменять это значение возникает редко.

Контрольная точка (Checkpoint, CKPT)

Обновляет заголовки файлов данных после установки контрольной точки.

Процесс восстановления (Recover, RECO)

Автоматически очищает неудавшиеся и отложенные распределенные транзакции.

Диспетчер (Dispatcher)

Эти необязательные фоновые процессы запускаются, если развернут разделяемый сервер.

Служба глобального кэша (Global Cache Service, LMS)

Управляет ресурсами, необходимыми Real Application Clusters, а также координирует использование ресурсов несколькими экземплярами.

Очередь заданий (Job Queue)

Служба выполнения команд и процедур PL/SQL по заданному расписанию.

Монитор очередей (Queue Monitor, QMNn)

Осуществляет мониторинг очередей сообщений подсистемы Oracle Streams. Поддерживается до 10 таких мониторов.

Процессы подсистемы автоматического управления хранением (Automatic Storage Management, ASM)

Процесс RBAL координирует перебалансировку работ для групп дисков. ORB*n* выполняет собственно перебалансировку. Процесс ASMB обеспечивает коммуникацию между базой данных и экземпляром ASM.