3. Архитектура СУБД Oracle

3.1. Общие представления о СУБД Oracle

База данных

База данных - это набор данных, трактуемых как единое целое. Цель базы данных - хранить и получать соответствующую информацию.

Система управления реляционными базами данных **Oracle** (**RDBMS**) надежно управляет большим объемом данных в многопользовательской среде так, чтобы множество пользователей могли одновременно получать доступ к одним и тем же данным. Это выполняется с предоставлением высокой производительности. В то же время, предотвращается несанкционированный доступ и обеспечиваются эффективные решения для восстановления на случай сбоя.

Буква "**g**" в названии Базы данных **Oracle 11g** означает **Grid** или технологию **Грид**.

Корпорация **Oracle** создала программное обеспечение инфраструктуры **грид-вычислений**, которая балансирует все типы рабочих нагрузок, распределяя их по серверам, и позволяет всем этим серверам управляться как одна единая система. **Грид-вычисления** могут достигнуть того же высочайшего уровня надежности как вычисления мэйнфрейма, потому что все компоненты кластеризируются. Но в отличие от мэйнфреймов и больших **UNIX** серверов симметричной многопроцессорной обработки (**SMP**), **грид** может быть создан на технологиях открытых систем, таких как процессоры **Intel** и Операционные системы **Linux** - по очень низким ценам.

Технология **грид-вычислений Oracle** включает:

- Автоматическое управление Хранением (**ASM**)
- **Real Application Clusters** (**RAC**)
- Кластеры Серверов приложений
- Консоль Управления Гридом предприятия (**Enterprise Manager**)

Система управления базами данных (СУБД)

Система управления реляционными базами данных **Oracle** (**RDBMS**) обеспечивает открытый, всесторонний, комплексный подход для управления информацией. База данных **Oracle** является первой БД, которая спроектирована для промышленных **грид вычислений** (наиболее гибкого и эффективного в плане расходов способа управления информацией и приложениями).

Oracle проектировалась как максимально переносимая СУБД, — она доступна на всех распространенных платформах. Поэтому физическая архитектура **Oracle** различна в разных операционных системах.

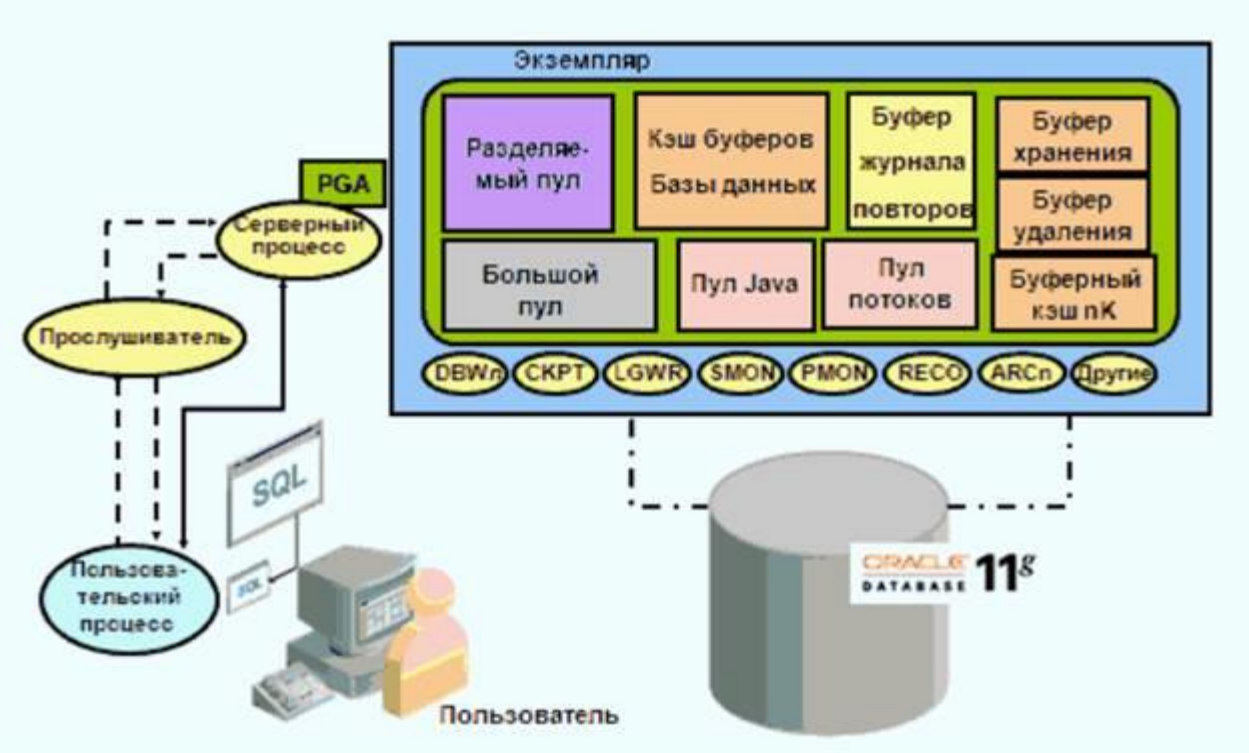
Например, в ОС **UNIX** СУБД **Oracle** реализована в виде нескольких отдельных процессов операционной системы — практически каждая существенная функция реализована отдельным процессом. Для **UNIX** такая реализация подходит, поскольку основой многозадачности в ней является процесс. Для **Windows**, однако, подобная реализация не подходит и работала бы не слишком хорошо (система получилась бы медленной и плохо масштабируемой). На этой платформе СУБД **Oracle** реализована как один многопоточный процесс, т.е. с использованием подходящих для этой платформы механизмов реализации.

На мэйнфреймах **IBM**, работающих под управлением **OS/390** и **zOS**, СУБД **Oracle** использует несколько адресных пространств **OS/390**, совместно образующих экземпляр **Oracle**. Для одного экземпляра базы данных можно сконфигурировать до 255 адресных пространств. Более того, СУБД **Oracle** взаимодействует с диспетчером загрузки **OS/390 WorkLoad Manager (WLM**) для установки приоритетности выполнения определенных компонентов **Oracle** по отношению друг к другу и к другим задачам, работающим в системе **OS/390**. В ОС **Netware** тоже используется многопоточная модель.

Хотя физические средства реализации СУБД **Oracle** на разных платформах могут отличаться, архитектура системы — достаточно общая, чтобы можно было понять, как СУБД **Oracle** работает на всех платформах.

Взаимодействие с базой данных Oracle

Следующий пример описывает операции базы данных Oracle на наиболее базовом уровне. Он иллюстрирует конфигурацию базы данных Oracle, в которой пользователь и соответствующий серверный процесс находятся на отдельных компьютерах, соединенных через сеть.



- Экземпляр запущен на узле, где установлена База данных **Oracle**, часто называемом хост-сервером или сервером базы данных.
- Пользователь запускает приложение, порождающее пользовательский процесс. Приложение пытается установить соединение с сервером. (Соединение может быть локальным, клиент-серверным или трехярусным соединением с среднего уровня.)
- На сервере работает прослушиватель, у которого есть подходящий обработчик Сетевых служб **Oracle**. Прослушиватель обнаруживает запрос на установку соединения от приложения и создает выделенный серверный процесс по запросу пользовательского процесса.
- Пользователь выполняет **SQL-оператор** типа **DML** и фиксирует транзакцию. Например, пользователь изменяет адрес заказчика в таблице и фиксирует изменение.
- Серверный процесс получает оператор и проверяет разделяемый пул (компонент **SGA**) на наличие любой разделяемой области **SQL**, которая содержит идентичный **SQL-оператор**. Если такая разделяемая область **SQL** находится, серверный процесс проверяет права доступа пользователя к запрошенным данным, и затем используется существующая разделяемая область **SQL**, чтобы обработать оператор. Если разделяемая область **SQL** не находится, для оператора выделяется новая разделяемая область **SQL** так, чтобы он мог быть разобран и обработан.
- Серверный процесс получает все необходимые значения данных, или непосредственно от файла данных (таблицы) или из значений, сохраненных в буферном кэше Базы данных.
- Серверный процесс модифицирует данные в **SGA**. Поскольку транзакция фиксируется, процесс записи в журнал (**LGWR**) сразу записывает транзакцию в файл журнала повторов. Процесс записи в Базу данных (**DBWn**) записывает модифицированные блоки на диск, когда это можно будет сделать эффективным образом.
- Если транзакция успешна, серверный процесс отправляет сообщение по сети приложению. В противном случае передается сообщение об ошибке.
- В каждый момент этой вышеописанной процедуры работают другие фоновые процессы, наблюдая за событиями, которые требуют вмешательства. Кроме того, сервер базы данных управляет транзакциями других пользователей и предотвращает разногласия между транзакциями, которые запрашивают те же самые данные.

Компоненты хранения ASM

ASM не уменьшает существующей функциональности базы данных. Существующие базы данных способны работать точно также. Новые файлы могут быть созданы как файлы **ASM**, тогда как существующие файлы могут администрироваться старым способом или могут быть перемещены на **ASM**.

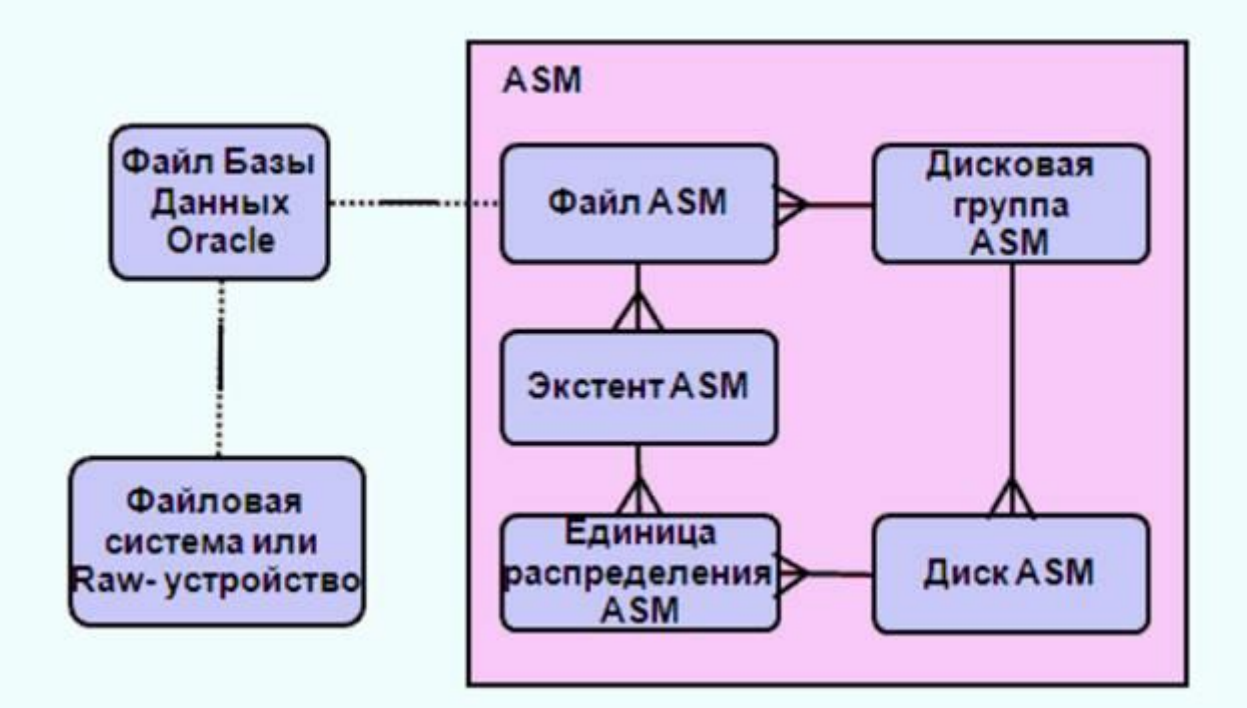


Схема поясняет отношения между файлом данных БД **Oracle** и **компонентами хранения ASM**. Обозначение в виде вороньей лапы представляет отношение "один ко многим". У файла данных БД **Oracle** есть взаимно-однозначное соответствие либо с файлом, хранящимся в файловой системе операционной системы или с файлом **ASM**.

Дисковая группа **Oracle ASM** - это набор одного или более дисков **Oracle ASM**, которые управляются как единый логический элемент.

Структуры данных в дисковой группе используют часть пространства для метаданных.

Диски **Oracle ASM** - устройства хранения, предоставленные дисковой группе **Oracle ASM**, они могут быть физическим дисками или разделами, номерами логических единиц (**LUN**) массива хранения, логическими томами (**LV**) или сетевыми прикрепленными файлами.

Каждый диск **ASM** делится на множество единиц выделения **ASM**, являющихся наименьшим непрерывным количеством дискового пространства, которое выделяет **ASM**. Когда Вы создаете дисковую группу **ASM**, можно установить размер единицы выделения **ASM** равным 1, 2, 4, 8, 16, 32 или 64 Мбайта в зависимости от уровня совместимости дисковой группы. Одна или более единиц выделения **ASM** формируют экстент **ASM**.

Экстент **Oracle ASM** - сырое хранилище, используемое для хранения содержимого файла **Oracle ASM**. Файл **Oracle ASM** состоит из одного или более файловых экстентов. Переменные размеры экстента в **1*AU**, **4*AU** и **16*AU** используются для того, чтобы поддерживать очень большие файлы **ASM**.

Автоматическое управление хранением

Автоматическое управление Хранением (**ASM**) обеспечивает вертикальную интеграцию файловой системы и менеджера томов для файлов базы данных **Oracle**. **ASM** может обеспечить управление для одиночных симметричных мультипроцессорных машин (**SMP**) или для множества узлов кластера, поддерживая **Oracle RAC**.

- Обеспечивает переносимую и высокоэффективную кластерную файловую систему
- Управляет файлами базы данных **Oracle**
- Управляет файлами приложения с помощью кластерной Файловой системы **ASM (ACFS)**
- Распространяет данные по дискам, чтобы сбалансировать загрузку
- Зеркалирует данные на случай отказов
- Решает задачи управления хранением

Кластерная файловая система **ASM Oracle** (**ACFS**) является многоплатформенной, масштабируемой файловой системой и технологией управления хранением, которая расширяет функциональность **ASM**, чтобы

поддерживать файлы приложения за пределами Базы данных **Oracle**, такие как исполняемые программы, отчеты, файлы типа **BFILE**, видео, аудио, текст, изображения и другие данные файлов приложений общего назначения.

ASM распределяет нагрузку ввода-вывода через все доступные ресурсы, чтобы оптимизировать производительность, устраняя потребность в ручной настройке ввода-вывода. **ASM** помогает администраторам БД управлять динамичной средой базы данных, позволяя им увеличить размер базы данных без необходимости выключения базы данных, чтобы скорректировать распределение хранения.

ASM может поддерживать избыточные копии данных, чтобы обеспечить отказоустойчивость, или оно может быть создано поверх предоставленных поставщиком механизмов хранения. Управление данными осуществляется на основе выбора требуемых характеристик надежности и показателей производительности для классов данных, вместо человеческого взаимодействия на пофайловой основе.

Возможности **ASM** экономят время **DBA**, автоматизируя ручное хранение и таким образом увеличивая возможность администратора управлять большим количеством более крупных баз данных с повышенной эффективностью.

Табличные пространства SYSTEM и SYSAUX

Каждая база данных **Oracle** должна содержать табличные пространства **SYSTEM** и **SYSAUX**.

- Табличные пространства **SYSTEM** и **SYSAUX** являются обязательными табличными пространствами, которые создаются во время создания базы данных. Они должны быть в режиме онлайн.
- Табличное пространство **SYSTEM** используется для базовой функциональности (например, для таблиц словаря данных).
- Вспомогательное табличное пространство **SYSAUX** используется для дополнительных компонентов базы данных (таких как Репозитарий **Enterprise Manager**).
- В табличных пространствах **SYSTEM** и **SYSAUX** не рекомендуется хранить данные приложений.

Табличные пространства **SYSTEM** и **SYSAUX** автоматически создаются при создании БД. В системе по умолчанию создается табличное пространство типа **SMALLFILE**. Можно также создать табличные пространства типа **BIGFILE**, которые позволяют базе данных **Oracle** управлять ультрабольшими файлами.

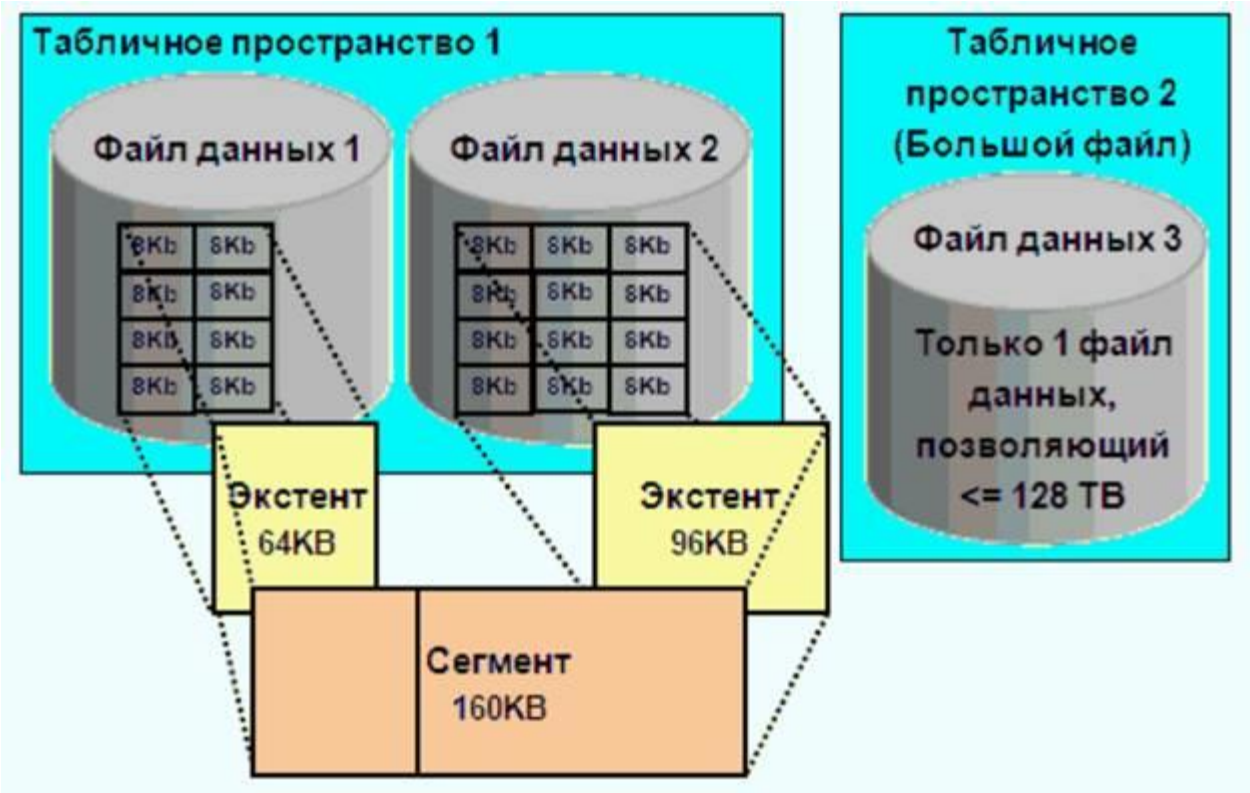
Табличное пространство может быть в режиме **онлайн** (доступно) или **офлайн** (не доступно). Табличное пространство **SYSTEM** всегда находится в режиме онлайн, когда база данных открыта. Она хранит таблицы, которые поддерживают базовую функциональность базы данных, такие как таблицы словаря данных.

Табличное пространство **SYSAUX** - вспомогательное табличное пространство по отношению к табличному пространству **SYSTEM**. Табличное пространство **SYSAUX** хранит много компонентов базы данных, и оно должно быть в режиме онлайн для корректного функционирования всех компонентов базы данных. Табличными пространствами **SYSTEM** и **SYSAUX** не рекомендуют пользоваться для хранения данных приложения. Дополнительные табличные пространства могут быть созданы для этой цели.

Примечание: Табличное пространство **SYSAUX** может быть переведено в режим **оффлайн**, чтобы сделать восстановление табличного пространства, тогда как это не возможно для табличного пространства **SYSTEM**. Ни одно из них не может быть сделано доступным только для чтения.

Табличные пространства и Файлы данных

База данных делится на табличные пространства, которые являются логическими единицами хранения данных и могут использоваться, чтобы сгруппировать связанные логические структуры.



Каждая база данных логически делится на два или более **табличных простанства**, в число которых входят табличные пространства **SYSTEM** и **SYSAUX**. Один или более **файлов данных** создаются явным образом для каждого табличного пространства, чтобы физически хранить данные всех логических структур в табличном пространстве.

Схема на рисунке иллюстрирует табличное пространство один, составленное из двух файлов данных. Сегмент размером 160 Кбайт распространяется на два файла данных, составленных из двух экстентов. Первый экстент размером 64 Кбайта находятся в первом файле данных и второй экстент размером 96 Кбайт находится во втором файле данных. Оба экстента формируются из непрерывных блоков **Oracle** по 8 Кбит.

Примечание: Можно также создавать табличные пространства типа **BIGFILE**, имеющие только один файл, который является обычно очень большим. Файл может быть любого размера вплоть до максимума, который позволяет архитектура **ID** строки. Максимальный размер равен размеру блока для табличного пространства, умноженному на 236, или 128 Тбайт для размера блока в 32 Кбайта. Традиционные табличные пространства типа **SMALLFILE** (которые являются значением по умолчанию) могут содержать несколько файлов данных, но файлы не могут быть столь большими. Для получения дополнительной информации о табличных пространствах типа **BIGFILE**, см. Руководство Администратора Базы данных **Oracle**.

Сегменты, Экстенты и Блоки

- **Сегменты** существуют в табличном пространстве.
- **Сегменты** - наборы экстентов.
- **Экстенты** - наборы блоков данных.
- **Блоки данных** отображаются на дисковые блоки.

Подмножество объектов базы данных, таких как таблицы и индексы, хранится как **сегменты** в табличных пространствах. Каждый сегмент содержит один или более **экстентов**. Экстент состоит из непрерывных блоков данных, что означает, что каждый экстент может существовать только в одном файле данных. **Блоки данных** - наименьшая единица ввода-вывода в базе данных.

Когда база данных запрашивает набор блоков данных от операционной системы (ОС), ОС отображает их на фактическую файловую систему или блок диска на устройстве хранения. Из-за этого Вы не должны знать физический адрес каких-либо данных в Вашей базе данных. Это также означает, что файл данных может чередоваться или зеркалироваться на нескольких дисках.

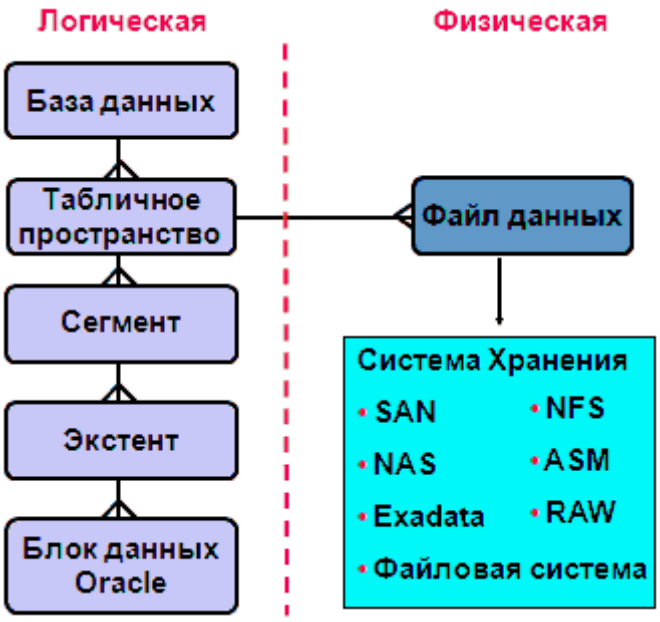
Размер блока данных может быть установлен во время создания базы данных. Размер по умолчанию 8 Кбайт подходит для большинства баз данных. Если Ваша база данных поддерживает приложение хранилища данных, у которого есть большие таблицы и индексы, больший размер блока может быть более подходящим.

Если Ваша база данных поддерживает транзакционное приложение, в котором чтения и записи происходят случайным образом, указание меньшего размера блока может быть более адекватным. Максимальный размер блока зависит от Вашей ОС. Минимальный размер блока **Oracle** составляет 2 Кбайта; маловероятно, что он будет где-либо уместен.

У Вас могут быть табличные пространства с нестандартным размером блока. Для получения дополнительной информации см. Руководство Администратора Базы данных **Oracle**.

Логические и Физические Структуры базы данных

У базы данных есть как логические, так и физические структуры.



Базы данных, Табличные пространства и Файлы данных

Взаимосвязь между базами данных, табличными пространствами и файлами данных проиллюстрирована на рисунке. Каждая база данных логически делится на два или более табличных пространства. Один или более файлов данных создаются явным образом для каждого табличного пространства, чтобы физически хранить данные всех сегментов в табличном пространстве. Если это ВРЕМЕННОЕ табличное пространство, у него имеется временный файл вместо файла данных. Файл данных табличного пространства может физически храниться с помощью любой поддерживаемой технологии хранения.

Табличные пространства

База данных делится на логические единицы хранения данных, называемые **табличными пространствами**, которые собирают в группу связанные логические структуры или файлы данных. Например, табличные пространства обычно группируют все сегменты приложения, чтобы упростить некоторые административные операции.

Блоки данных

На самом низшем уровне гранулярности данные базы данных **Oracle** хранятся в **блоках данных**. Один блок данных соответствует определенному числу байтов физического пространства на диске. Размер блока данных указывается для каждой табличного пространства, когда оно создается. База данных использует и выделяет свободное пространство базы данных в блоках данных **Oracle**.

Экстенты

Следующий уровень логического пространства базы данных - экстент. **Экстент** - определенное число непрерывных блоков данных **Oracle** (полученных за одно выделение), которые используются, чтобы хранить определенный тип информации. Блоки данных **Oracle** в экстенте логически непрерывны, но могут быть физически разбросаны на диске из-за чередования **RAID** и реализаций файловой системы.

Сегменты

Уровень логического хранения базы данных, стоящий над экстентом называют сегментом. **Сегмент** - набор экстентов, выделенных для определенной логической структуры. Например:

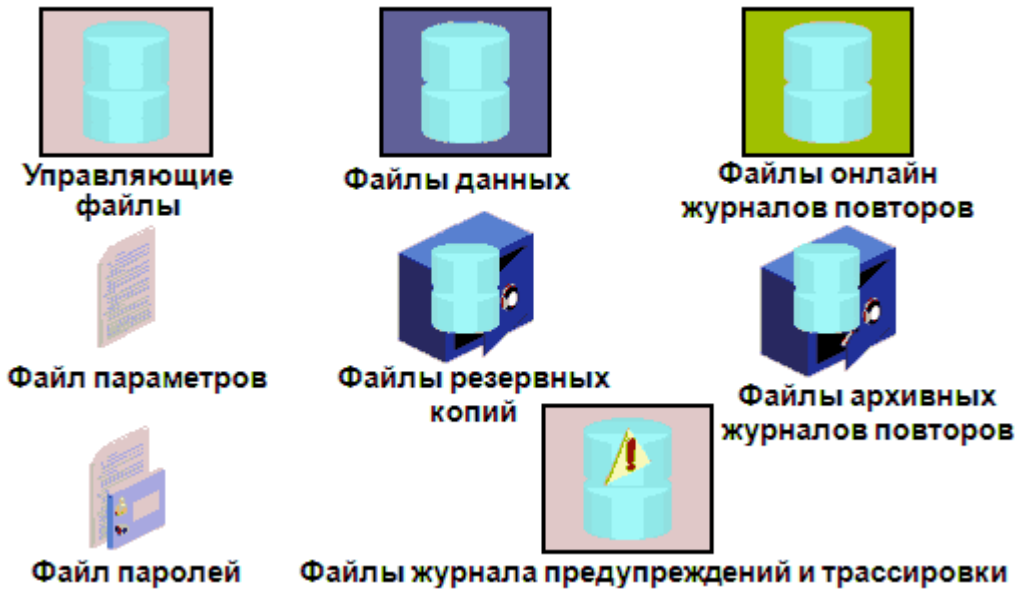
- Сегменты данных: Каждая не кластеризованная, не индексная таблица имеет сегмент данных, за исключением внешних таблиц, глобальных временных таблиц и секционированных таблиц, в которых у каждой таблицы есть один или более сегментов. Все данные таблицы хранятся в экстентах ее сегмента данных. Для секционированной таблицы у каждой секции есть сегмент данных. У каждого кластера есть сегмент данных. Данные каждой таблицы в кластере сохраняются в сегменте данных кластера.
- Индексные сегменты: У каждого индекса есть индексный сегмент, который сохраняет все его данные. Для секционированного индекса у каждой секции есть индексный сегмент.
- Сегменты отмены: Одно табличное пространство **ОТМЕНЫ** создается для каждого экземпляра базы данных. Это табличное пространство содержит многочисленные сегменты отмены, чтобы временно хранить информацию отмены. Информация в сегменте отмены используется, чтобы генерировать согласованную по чтению информацию базы данных, а также во время восстановления базы данных - для отката незафиксированных транзакций пользователей.
- Временные сегменты: Временные сегменты создаются базой данных **Oracle**, когда **SQL-оператор** нуждается во временной рабочей области, чтобы завершить выполнение. Когда оператор заканчивает выполнение, экстенты временного сегмента возвращаются экземпляру для будущего использования. Указывайте либо временное табличное пространство по умолчанию для каждого пользователя, либо временное табличное пространство по умолчанию, которое будет использоваться для всей базы данных.

Примечание: Есть другие типы сегментов, не упомянутые выше. Есть также объекты схемы, такие как представления, пакеты, триггеры, и т.д., которые не считают сегментами даже при том, что они являются объектами базы данных. Сегменту принадлежит соответствующее выделение дискового пространства. Другие объекты существуют как строки, хранимые в системном сегменте метаданных.

Сервер базы данных **Oracle** динамически выделяет пространство. Когда существующие экстенты сегмента заполняются, добавляются дополнительные экстенты. Поскольку экстенты выделяются по необходимости, экстенты сегмента могут быть, а могут и не быть непрерывными на диске, и они могут принадлежать различным файлам данных одного и того же табличного пространства.

Архитектура Хранения базы данных

Файлы, из которых состоит база данных Oracle, делятся на несколько типов: контрольные или управляющие файлы, файлы данных, файлы журналов транзакций и другие.



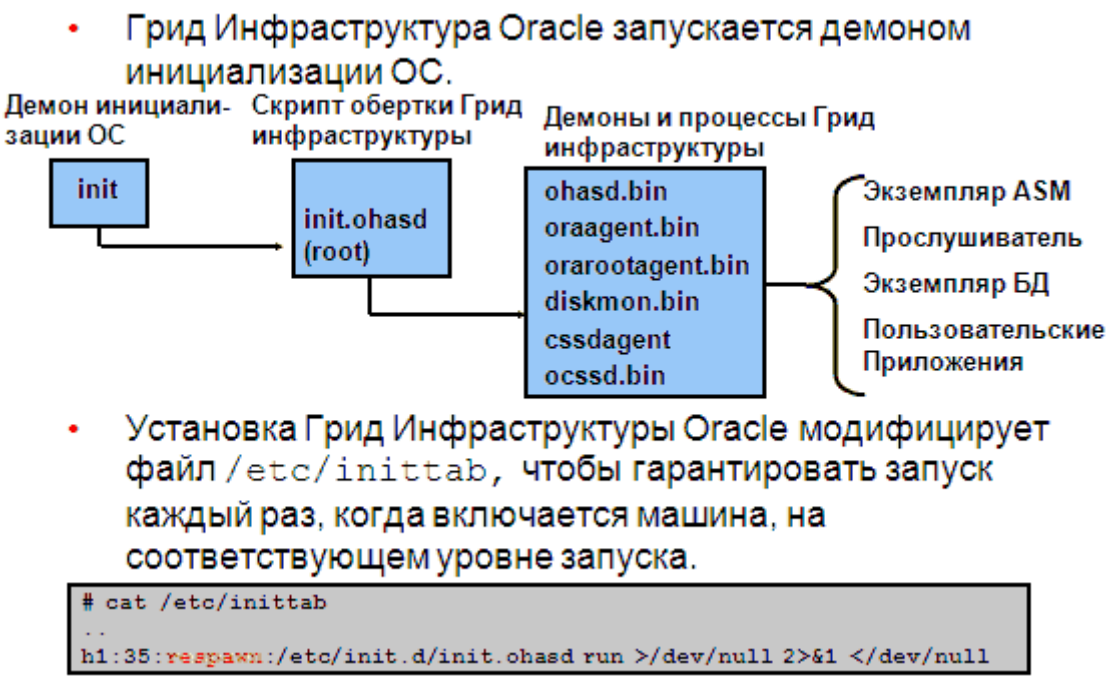
Файлы, которые составляют базу данных Oracle, организованы следующим образом:

- **Контрольные файлы:** Содержат данные о самой базе данных непосредственно (то есть, информацию о физической структуре базы данных). Эти файлы являются критическими по отношению к базе данных. Без них невозможно открыть файлы данных, чтобы получить доступ к данным в базе данных. Они могут также содержать метаданные, связанные с резервными копиями.
- **Файлы данных:** Содержат данные пользователя или приложения базы данных, а так же метаданные и словарь данных
- **Оперативные файлы журнала повторов:** Нужны для восстановления экземпляра базы данных. Если сервер базы данных терпит сбой, но не теряет файлов данных, экземпляр может восстановить базу данных с информацией в этих файлах.

Следующие дополнительные файлы важны для успешной работы базы данных:

- **Файл параметров:** используется, чтобы определить конфигурацию экземпляра при его запуске
- **Файл паролей:** Позволяет пользователям, использующим роли **sysdba**, **sysoper** и **sysasm** соединяться удаленно с экземпляром и выполнять задачи администрирования
- **Файлы резервных копий:** используются для восстановления базы данных. Обычно Вы восстанавливаете файл из резервной копии, когда отказ носителя или пользовательская ошибка повреждают или стирают исходный файл.
- **Заархивированные файлы журнала повторов:** Содержат продолжающуюся историю изменений данных (повторы), которые генерируются экземпляром. Используя эти файлы и резервное копирование базы данных, можно восстановить потерянный файл данных. Таким образом, архивные журналы позволяют осуществлять восстановление реставрированных файлов данных.
- **Файлы трассировки:** Каждый серверный и фоновый процесс могут делать записи в соответствующий трассировочный файл. Когда происходит внутренняя ошибка процесса, процесс сохраняет дамп с информацией об ошибке в свой трассировочный файл. Часть информации в файле трассировки предназначена для администратора базы данных, тогда как другая часть - для службы Поддержки **Oracle**.
- **Файл журнала предупреждений:** Это специальные трассировочные записи. Журнал предупреждений базы данных является хронологическим журналом сообщений и ошибок. **Oracle** рекомендует, чтобы Вы периодически просматривали журнал предупреждений.

Последовательность запуска процессов



Во время установки **Инфраструктуры Грида Oracle** в файл **/etc/inittab** операционной системы помещаются записи, чтобы запустить сценарий обертки. Сценарий обертки ответственен за установку переменных окружения и дальнейшего запуска демонов и процессов **Грид Инфраструктуры Oracle**.

Когда используется команда остановки **Инфраструктуры Грида Oracle**, демоны будут остановлены, но процесс сценария обертки останется работать.

Формат файла **/etc/inittab** ОС **UNIX** следующий:

идентификатор: уровни выполнения: действие: процесс с параметрами

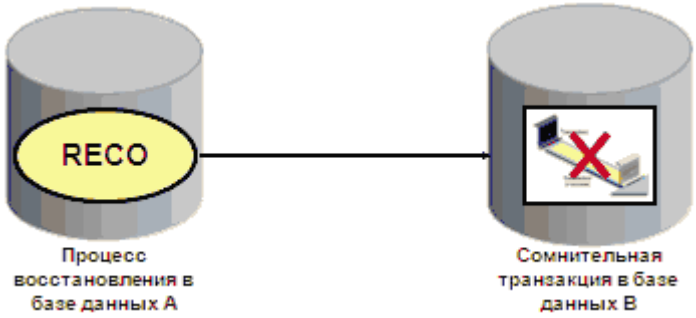
Сценарий обертки запускается с действием повторного порождения (**respawn**), таким образом, он будет перезапускаться всякий раз, когда он завершается.

Некоторые из демонов **Грид Инфраструктуры Oracle** будут выполняться из-под пользователя **root** с приоритетом реального времени, а другие будут выполняться из-под владельца **Грид Инфраструктуры** с приоритетами непривилегированного режима - после того, как они будут запущены. На платформе **Windows** используются службы операционной системы вместо сценариев инициализации обертки, и демонами являются исполняемые двоичные файлы.

Процесс восстановления

- Используется с конфигурацией распределенной базы данных

- Автоматически подключается к другим базам данных, вовлеченных в сомнительные распределенные транзакции
- Автоматически разрешает все сомнительные транзакции
- Удаляет любые строки, которые соответствуют сомнительным транзакциям



Процесс восстановления (**RECO**) является фоновым процессом, который используется в конфигурации распределенной базы данных, которая автоматически разрешает сбои, связанные с распределенными транзакциями. Процесс **RECO** экземпляра автоматически соединяется с другими базами данных, участвующими в распределенной транзакции, находящейся под сомнением. Когда процесс **RECO** восстанавливает соединение между участвующими серверами баз данных, он автоматически разрешает все сомнительные транзакции, удаляя из таблицы незавершенных транзакций каждой базы данных любые строки, которые соответствуют разрешенным сомнительным транзакциям.

Если процесс **RECO** не в состоянии соединиться с удаленным сервером, **RECO** автоматически пытается соединиться снова после некоторого интервала времени. При этом, **RECO** ожидает каждый раз все больше времени (интервал растет по экспоненте) прежде, чем он будет делать попытку нового соединения.

Структура памяти

Рассмотрим основные структуры памяти сервера **Oracle**:

- **SGA, System Global Area** — глобальная область системы. Это большой совместно используемый сегмент памяти, к которому обращаются все процессы **Oracle**.
- **PGA, Process Global Area** — глобальная область процесса. Это приватная область памяти процесса или потока, недоступная другим процессам/потокам.
- **UGA, User Global Area** — глобальная область пользователя. Это область памяти, связанная с сеансом. Глобальная область памяти может находиться в **SGA** либо в **PGA**. Если сервер работает в режиме **MTS**, она располагается в области **SGA**, если в режиме выделенного сервера, — в области **PGA**.

Рассмотрим кратко области **PGA** и **UGA**, затем перейдем к действительно большой структуре — области **SGA**.

PGA и UGA

Как уже было сказано, **PGA** — это область памяти процесса. Эта область памяти используется одним процессом или одним потоком. Она недоступна ни одному из остальных процессов/потоков в системе. Область **PGA** обычно выделяется с помощью библиотечного вызова **malloc()** языка **C** и со временем может расти (или уменьшаться). Область **PGA** никогда не входит в состав области **SGA** — она всегда локально выделяется процессом или потоком.

Область памяти **UGA** хранит состояние сеанса, поэтому всегда должна быть ему доступна. Местонахождение области **UGA** зависит исключительно от конфигурации сервера **Oracle**. Если сконфигурирован режим **MTS**, область **UGA** должна находиться в структуре памяти, доступной всем процессам, следовательно, в **SGA**. В этом случае сеанс сможет использовать любой разделяемый сервер, так как каждый из них сможет прочитать и записать данные сеанса. При подключении к выделенному серверу это требование общего доступа к информации о состоянии сеанса снимается, и область **UGA** становится почти синонимом **PGA**, — именно там информация о состоянии сеанса и будет располагаться. Просматривая статистическую информацию о системе, можно обнаружить, что при работе в режиме выделенного сервера область **UGA** входит в **PGA** (размер области **PGA** будет больше или равен размеру используемой памяти **UGA** — размер **UGA** будет учитываться при определении размера области **PGA**).

Размер области **PGA/UGA** определяют параметры уровня сеанса в файле **init.ora**: **SORT_AREA_SIZE** и **SORT_AREA_RETAINED_SIZE**. Эти два параметра управляют объемом пространства, используемым сервером **Oracle** для сортировки данных перед сбросом на диск, и определяют объем сегмента памяти, который не будет освобожден по завершении сортировки. **SORT_AREA_SIZE** обычно выделяется в области **PGA**, а **SORT_AREA_RETAINED_SIZE** — в **UGA**. Управлять размером областей **UGA/PGA** можно с помощью запроса к специальному представлению **V$** сервера **Oracle**. Эти представления называют также представлениями динамической производительности. С помощью представлений **V$** можно определить текущее использование памяти под области **PGA** и **UGA**.

Например, запущен небольшой тестовый пример, требующий сортировки большого объема данных. После просмотра нескольких первых строк данных, решено не извлекать остальное результирующее множество. После этого можно сравнить использование памяти "до" и "после":

```
tkyte@TKYTE816> select a.name, b.value
2  from v$statname a, v$mystat b
```

```
3  where a.statistic# = b.statistic#
4  and a.name like '%ga %'
5  /

NAME                           VALUE
------------------------------ ----------
session uga memory                  67532
session uga memory max              71972
session pga memory                 144688
session pga memory max             144688

4 rows selected.
```

Итак, перед началом сортировки в области **UGA** было около 70 Кбайт данных, а в **PGA** — порядка 140 Кбайт.

Первый вопрос: сколько памяти используется в области **PGA** помимо **UGA**? Вопрос нетривиальный и на него нельзя ответить, не зная, подключен ли сеанс к выделенному или к разделяемому серверу; но даже зная это нельзя ответить однозначно. В режиме выделенного сервера область **UGA** входит в состав **PGA**. В этом случае порядка 140 Кбайт выделено в области памяти процесса или потока. В режиме **MTS** область **UGA** выделяется из **SGA**, а область **PGA** относится к разделяемому серверу. Поэтому при работе в режиме **MTS** к моменту получения последней строки из представленного выше запроса разделяемый серверный процесс уже может использоваться другим сеансом. Соответственно, область **PGA** уже не принадлежит нам, так что мы используем всего 70 Кбайт памяти (если только не находимся в процессе выполнения запроса, когда областями **PGA** и **UGA** суммарно используется 210 Кбайт памяти).

Теперь разберемся, что происходит в областях **PGA/UGA** нашего сеанса:

```
tkyte@TKYTE816> show parameter sort_area

NAME                           TYPE        VALUE
------------------------------ ------- ------------------------------
sort_area_retained_size        integer                 65536
sort_area_size                 integer                 65536
```

```
tkyte@TKYTE816> set pagesize 10

tkyte@TKYTE816> set pause on

tkyte@TKYTE816> select * from all_objects order by 1, 2, 3, 4;

tkyte@TKYTE816> set pause off

tkyte@TKYTE816> select a.name, b.value
2  from v$statname a, v$mystat b
3  where a.statistic# = b.statistic#
4  and a.name like '%ga %'
5  /

NAME                           VALUE
------------------------------ ----------
session uga memory                  67524
session uga memory max             174968
session pga memory                 291336
session pga memory max             291336

4 rows selected.
```

Как видите, памяти использовано больше, поскольку данные сортировались. Область **UGA** временно увеличилась примерно на размер **SORT_AREA_RETAINED_SIZE**, а область **PGA** — немного больше. Для выполнения запроса и сортировки сервер **Oracle** выделил дополнительные структуры, которые оставлены в памяти сеанса для других запросов. Давайте выполним ту же операцию, изменив значение **SORT_AREA_SIZE**:

```
tkyte@TKYTE816> alter session set sort_area_size=1000000;

Session altered.

tkyte@TKYTE816> select a.name, b.value
```

```
  2  from v$statname a, v$mystat b
  3  where a.statistic# = b.statistic#
  4  and a.name like '%ga %'
  5  /

NAME                           VALUE
------------------------------ ----------
session uga memory                  63288
session uga memory max             174968
session pga memory                 291336
session pga memory max             291336

4 rows selected.

tkyte@TKYTE816> show parameter sort_area

NAME                           TYPE    VALUE
---------------------------------- ------- ---------------------
sort_area_retained_size              integer 65536
sort_area_size                       integer 1000000

tkyte@TKYTE816> select * from all_objects order by 1, 2, 3, 4;

tkyte@TKYTE816> set pause off

        tkyte@TKYTE816> select a.name, b.value
```

```
  2  from v$statname a, v$mystat b
  3  where a.statistic# = b.statistic#
  4  and a.name like '%ga %'
  5  /

NAME                           VALUE
------------------------------ ----------
session uga memory                 67528
session uga memory max            174968
session pga memory                1307580
session pga memory max            1307580

4 rows selected.
```

Как видите, в этот раз область **PGA** увеличилась существенно. Примерно на 1000000 байт, в соответствии с заданным значением **SORT_AREA_SIZE**. Интересно отметить, что в этот раз размер области **UGA** вообще не изменился. Для ее изменения надо задать другое значение **SORT_AREA_RETAINED_SIZE**, как показано ниже:

```
tkyte@TKYTE816> alter session set sort_area_retained_size=1000000;
Session altered.

tkyte@TKYTE816> select a.name, b.value
  2  from v$statname a, v$mystat b
  3  where a.statistic# = b.statistic#
  4  and a.name like '%ga %'
  5  /

NAME                           VALUE
------------------------------ ----------
session uga memory                 63288
```

```
session uga memory max            174968
session pga memory               1307580
session pga memory max           1307580

4 rows selected.

tkyte@TKYTE816> show parameter sort_area

NAME                                 TYPE    VALUE
------------------------------------ ------- ------------------------
sort_area_retained_size              integer 1000000
sort_area_size                       integer 1000000

tkyte@TKYTE816> select * from all_objects order by 1, 2, 3, 4;

tkyte@TKYTE816> select a.name, b.value
  2  from v$statname a, v$mystat b
  3  where a.statistic# = b.statistic#
  4  and a.name like '%ga %'
  5  /

NAME                                 VALUE
------------------------------ ----------
session uga memory                 66344
session uga memory max           1086120
session pga memory               1469192
```

```
session pga memory max          1469192

4 rows selected.
```

Теперь мы видим, что существенное увеличение размера области **UGA** связано с необходимостью дополнительно принять данные размером **SORT_AREA_RETAINED_SIZE**. В ходе обработки запроса 1 Мбайт сортируемых данных "кеширован в памяти". Остальные данные были на диске (где-то во временном сегменте). По завершении выполнения запроса это дисковое пространство возвращено для использования другими сеансами. Обратите внимание, что область **PGA** не уменьшилась до прежнего размера. Этого следовало ожидать, поскольку область **PGA** используется как "куча" и состоит из фрагментов, выделенных с помощью вызовов **malloc()**. Некоторые процессы в сервере **Oracle** явно освобождают память **PGA**, другие же оставляют выделенную память в куче (область для сортировки, например, остается в куче). Сжатие кучи при этом обычно ничего не дает (размер используемой процессами памяти только растет). Поскольку область **UGA** является своего рода "подкучей" (ее "родительской" кучей является область **PGA** либо **SGA**), она может сжиматься. При необходимости можно принудительно сжать область **PGA**:

```
tkyte@TKYTE816> exec dbms_session.free_unused_user_memory;

PL/SQL procedure successfully completed.

tkyte@TKYTE816> select a.name, b.value
  2  from v$statname a, v$mystat b
  3  where a.statistic# = b.statistic#
  4  and a.name like '%ga %'
  5  /

NAME                           VALUE
------------------------------ ----------
session uga memory                 73748
session uga memory max           1086120
session pga memory               183360
session pga memory max           1469192
```

Учтите, однако, что в большинстве систем это действие — пустая трата времени. Можно уменьшить размер кучи **PGA** в рамках экземпляра **Oracle**, но память при этом операционной системе не возвращается. В зависимости от принятого в ОС метода управления памятью, суммарное количество используемой памяти даже увеличится. Все зависит от того, как на данной платформе реализованы функции **malloc()**, **free()**, **realloc()**, **brk()** и **sbrk()** (стандартные функции управления памятью в языке **C**).

Итак, мы рассмотрели две структуры памяти, области **PGA** и **UGA**. Теперь понятно, что область **PGA** принадлежит процессу. Она представляет собой набор переменных, необходимых выделенному или разделяемому серверному процессу **Oracle** для поддержки сеанса. Область **PGA** — это "куча" памяти, в которой могут выделяться другие структуры. Область **UGA** также является кучей, в которой определяются связанные с сеансом структуры. Область **UGA** выделяется из **PGA** при подключении к выделенному серверу **Oracle** и — из области **SGA** при подключении в режиме **MTS**. Это означает, что при работе в режиме **MTS** необходимо задать такой размер области **SGA**, чтобы в ней хватило места под области **UGA** для предполагаемого максимального количества одновременно подключенных к базе данных пользователей. Поэтому область **SGA** в экземпляре, работающем в режиме **MTS**, обычно намного больше, чем область **SGA** аналогично сконфигурированного экземпляра, работающего в режиме выделенного сервера.

Область SGA

Каждый экземпляр **Oracle** имеет одну большую область памяти, которая называется **SGA**, **System Global Area** — глобальная область системы. Это большая разделяемая структура, к которой обращаются все процессы **Oracle**. Ее размер варьируется от нескольких мегабайт в небольших тестовых системах до сотен мегабайт в системах среднего размера и множества гигабайт в больших системах.

В ОС **UNIX** область **SGA** — это физический объект, которую можно "увидеть" с помощью утилит командной строки. Физически область **SGA** реализована как сегмент разделяемой памяти — отдельный фрагмент памяти, к которому могут подключаться процессы. При отсутствии процессов **Oracle** вполне допустимо иметь в системе область **SGA**; память существует отдельно от них. Однако наличие области **SGA** при отсутствии процессов **Oracle** означает, что произошел тот или иной сбой экземпляра. Эта ситуация — нештатная, но она бывает.

В ОС **Windows** увидеть область **SGA**, как в ОС **UNIX**, нельзя. Поскольку на этой платформе экземпляр **Oracle** работает как единый процесс с одним адресным пространством, область **SGA** выделяется как приватная память процесса **ORACLE.EXE**. С помощью диспетчера задач **Windows** (**Task Manager**) или другого средства контроля производительности можно узнать, сколько памяти выделено процессу **ORACLE.EXE**, но нельзя определить, какую часть по отношению к другим выделенным структурам памяти занимает область **SGA**.

В самой СУБД **Oracle** можно определить размер области **SGA** независимо от платформы.

Область **SGA** разбита на несколько пулов.

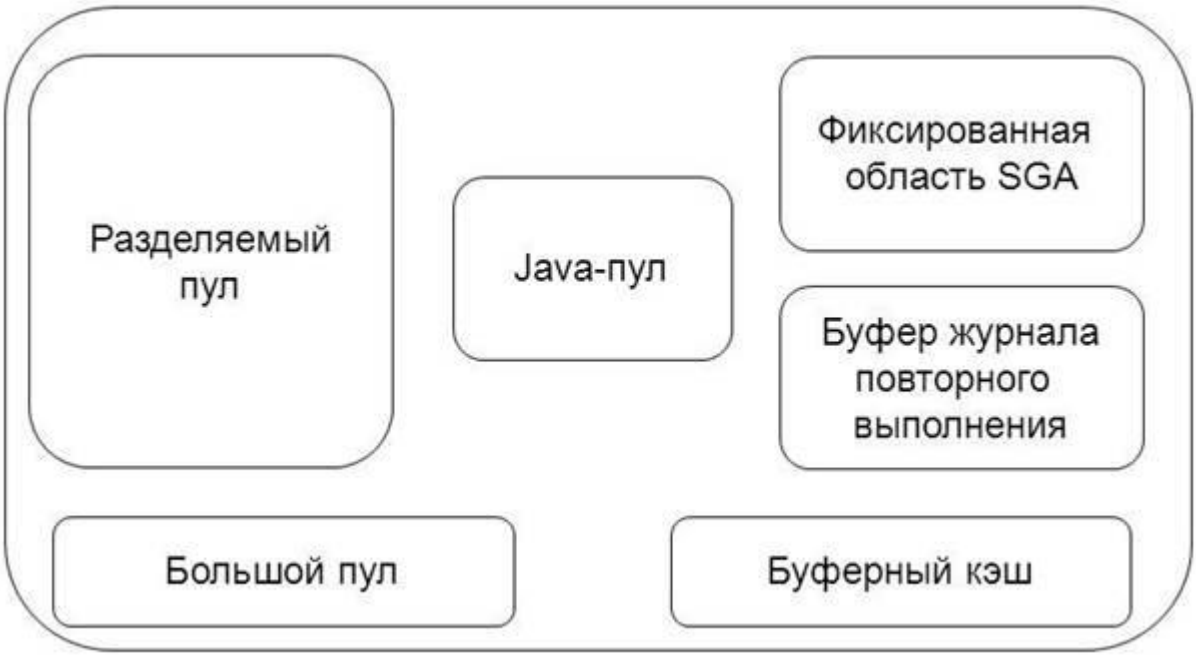
- **Java-пул**. **Java-пул** представляет собой фиксированный пул памяти, выделенный виртуальной машине **JVM**, которая работает в составе сервера.

- **Большой пул**. Большой пул (**large pool**) используется сервером в режиме **MTS** для размещения памяти сеанса, средствами распараллеливания **Parallel Execution** для буферов сообщений и при резервном копировании с помощью **RMAN** для буферов дискового ввода-вывода.

- **Разделяемый пул**. Разделяемый пул (**shared pool**) содержит разделяемые курсоры, хранимые процедуры, объекты состояния, кеш словаря данных и десятки других компонентов данных.

- **"Неопределенный" ("Null") пул**. Этот пул не имеет имени. Это память, выделенная под буферы блоков (кеш блоков базы данных, буферный кеш), буфер журнала повторного выполнения и "фиксированную область **SGA**".

Поэтому графически область **SGA** можно представить следующим образом:



На общий размер **SGA** наиболее существенно влияют следующие параметры **init.ora**.

- **JAVA_POOL_SIZE**. Управляет размером Java-пула.
- **SHARED_POOL_SIZE**. Управляет (до некоторой степени) размером разделяемого пула.
- **LARGE_POOL_SIZE**. Управляет размером большого пула.
- **DB_BLOCK_BUFFERS**. Управляет размером буферного кеша.
- **LOG_BUFFER**. Управляет (отчасти) размером буфера журнала повторного выполнения.

За исключением параметров **SHARED_POOL_SIZE** и **LOG_BUFFER**, имеется однозначное соответствие между значением параметров в файле **init.ora** и объемом памяти, выделяемой соответствующей структуре в области **SGA**. Например, если умножить **DB_BLOCK_BUFFERS** на размер буфера, получится значение, совпадающее с размером в строке **DB_BLOCK_BUFFERS** для пула **NULL** в представлении **V$SGASTAT** (добавляется определенный объем памяти на поддержку защелок). Суммарное количество байтов, вычисленное из строк представления **V$SGASTAT** для большого пула, совпадет со значением параметра инициализации **LARGE_POOL_SIZE**.

Фиксированная область SGA

Фиксированная область **SGA** — это часть области **SGA**, размер которой зависит от платформы и версии. Она "компилируется" в двоичный модуль сервера **Oracle** при установке (отсюда и название — "фиксированная"). Фиксированная область **SGA** содержит переменные, которые указывают на другие части **SGA**, а также переменные, содержащие значения различных параметров. Размером фиксированной области **SGA** (как правило, очень небольшой) управлять нельзя. Можно рассматривать эту область как "загрузочную" часть **SGA**, используемую сервером **Oracle** для поиска других компонентов **SGA**.

Буфер журнала повторного выполнения

Буфер журнала повторного выполнения используется для временного кеширования данных оперативного журнала повторного выполнения перед записью на диск. Поскольку перенос данных из памяти в память намного быстрее, чем из памяти — на диск, использование буфера журнала повторного выполнения позволяет существенно ускорить работу сервера. Данные не задерживаются в буфере журнала повторного выполнения надолго. Содержимое этого буфера сбрасывается на диск:

- раз в три секунды;

- при фиксации транзакции;

- при заполнении буфера на треть или когда в нем оказывается 1 Мбайт данных журнала повторного выполнения.

Поэтому создание буфера журнала повторного выполнения размером в десятки Мбайт — напрасное расходование памяти. Чтобы использовать буфер журнала повторного выполнения размером 6 Мбайт, например, надо выполнять продолжительные транзакции, генерирующие по 2 Мбайта информации повторного выполнения не более чем за три секунды. Если кто-либо в системе зафиксирует транзакцию в течение этих трех секунд, в буфере не будет использовано и 2 Мбайт, — содержимое буфера будет регулярно сбрасываться на диск. Лишь очень немногие приложения выиграют от использования буфера журнала повторного выполнения размером в несколько мегабайт.

Буферный кеш

До сих пор мы рассматривали небольшие компоненты области **SGA**. Теперь переходим к составляющей, которая достигает огромных размеров. В буферном кеше сервер **Oracle** хранит блоки базы данных перед их записью на диск, а также после считывания с диска. Это принципиально важный компонент **SGA**. Если сделать его слишком маленьким, запросы будут выполняться годами. Если же он будет чрезмерно большим, пострадают другие процессы (например, выделенному серверу не хватит пространства для создания области **PGA**, и он просто не запустится).

Буферный кеш в версиях до **Oracle 8.0** представлял собой один большой кеш. Все блоки кешировались одинаково, никаких средств деления пространства буферного кеша на части не существовало. В **Oracle 8.0** добавлена возможность создания буферных пулов. С ее помощью можно зарезервировать в буферном кеше место для сегментов (как вы помните, сегменты соответствуют таблицам, индексам и т.д.). Появилась возможность выделить место (буферный пул) достаточного размера для размещения целиком в памяти, например, таблиц-"справочников". При чтении сервером **Oracle** блоков из этих таблиц они кешируются в этом специальном пуле. Они будут конфликтовать за место в пуле только с другими помещаемыми в него сегментами. Остальные сегменты в системе будут "сражаться" за место в стандартном буферном пуле. При этом повышается вероятность их кеширования: они не выбрасываются из кеша как устаревшие при считывании других, не связанных с ними блоков. Буферный пул, обеспечивающий подобное кеширование, называется пулом **KEEP**. Блоками в пуле **KEEP** сервер управляет так же, как в обычном буферном кеше. Если блок используется часто, он остается в кеше; если к блоку некоторое время не обращались и в буферном пуле не осталось места, этот блок выбрасывается из пула как устаревший.

Можно выделить еще один буферный пул. Он называется пулом **RECYCLE**. В нем блоки выбрасываются иначе, чем в пуле **KEEP**. Пул **KEEP** предназначен для продолжительного кеширования "горячих" блоков. Из пула **RECYCLE** блок выбрасывается сразу после использования. Это эффективно в случае "больших" таблиц, которые читаются случайным образом. (Понятие "большая таблица" очень относительно; нет эталона для определения того, что считать "большим".) Если в течение разумного времени вероятность повторного считывания блока мала, нет смысла долго держать такой блок в кеше. Поэтому в пуле **RECYCLE** блоки регулярно перечитываются.

Разделяемый пул

Разделяемый пул — один из наиболее важных фрагментов памяти в области **SGA**, особенно для обеспечения производительности и масштабируемости. Слишком маленький разделяемый пул может снизить производительность настолько, что система будет казаться зависшей. Слишком большой разделяемый пул может привести к такому же результату. Неправильное использование разделяемого пула грозит катастрофой.

Итак, что же такое разделяемый пул? В разделяемом пуле сервер **Oracle** кеширует различные "программные" данные. Здесь кешируются результаты разбора запроса. Перед повторным разбором запроса сервер **Oracle** просматривает разделяемый пул в поисках готового результата. Выполняемый сеансом **PL/SQL-код** тоже кешируется здесь, так что при следующем выполнении не придется снова читать его с диска. **PL/SQL-код** в разделяемом пуле не просто кешируется, — появляется возможность его совместного использования сеансами. Если 1000 сеансов выполняют тот же код, загружается и совместно используется всеми сеансами лишь одна копия этого кода. Сервер **Oracle** хранит в разделяемом пуле параметры системы. Здесь же хранится кеш словаря данных, содержащий информацию об объектах базы данных. Короче, в разделяемом пуле хранится все, кроме продуктов питания.

Разделяемый пул состоит из множества маленьких (около 4 Кбайт) фрагментов памяти. Память в разделяемом пуле управляется по принципу давности использования (**LRU**). В этом отношении она похожа на буферный кеш: если фрагмент не используется, он теряется. Стандартный пакет **DBMS_SHARED_POOL** позволяет изменить это и принудительно закрепить объекты в разделяемом пуле. Это позволяет загрузить

часто используемые процедуры и пакеты при запуске сервера и сделать так, чтобы они не выбрасывались из пула как устаревшие. Обычно, если в течение определенного периода времени фрагмент памяти в разделяемом пуле не использовался, он выбрасывается как устаревший. Даже **PL/SQL-код**, который может иметь весьма большой размер, управляется механизмом постраничного устаревания, так что при выполнении кода очень большого пакета необходимый код загружается в разделяемый пул небольшими фрагментами. Если в течение продолжительного времени он не используется, то в случае переполнения выбрасывается из разделяемого пула, а пространство выделяется для других объектов.

Самый простой способ поломать механизм разделяемого пула **Oracle** — не использовать связываемые переменные. Отказавшись от использования связываемых переменных, можно "поставить на колени" любую систему, поскольку:

- система будет тратить много процессорного времени на разбор запросов;
- система будет тратить очень много ресурсов на управление объектами в разделяемом пуле, т.к. не предусмотрено повторное использование планов выполнения запросов.

Если каждый переданный серверу **Oracle** запрос специфичен, с жестко заданными константами, это вступает в противоречие с назначением разделяемого пула. Разделяемый пул создавался для того, чтобы хранящиеся в нем планы выполнения запросов использовались многократно. Если каждый запрос — абсолютно новый и никогда ранее не встречался, в результате кеширования только расходуются дополнительные ресурсы. Разделяемый пул начинает снижать производительность. Обычно эту проблему пытаются решить, увеличивая разделяемый пул, но в результате становится еще хуже. Разделяемый пул снова неизбежно заполняется, и его поддержка требует больших ресурсов, чем поддержка маленького разделяемого пула, поскольку при управлении большим заполненным пулом приходится выполнять больше действий, чем при управлении маленьким заполненным пулом.

Единственным решением этой проблемы является применение разделяемых операторов **SQL**, которые используются повторно. В главе 10 мы опишем параметр инициализации **CURSOR_SHARING**, который можно использовать для частичного решения подобных проблем, но наиболее эффективное решение — применять повторно используемые **SQL-операторы**. Даже самые большие из крупных систем требуют от 10000 до 20000 уникальных **SQL-операторов**. В большинстве систем используется лишь несколько сотен уникальных запросов.

Большой пул

Большой пул назван так не потому, что это "большая" структура (хотя его размер вполне может быть большим), а потому, что используется для выделения больших фрагментов памяти — больших, чем те, для управления которыми создавался разделяемый пул. До его появления в **Oracle 8.0**, выделение памяти выполнялось в рамках разделяемого пула. Это было неэффективно при использовании средств, выделяющих "большие" объемы памяти, например, при работе в режиме **MTS**. Проблема осложнялась еще и тем, что при обработке, требующей больших объемов памяти, эта память используется не так, как предполагает управление памятью в разделяемом пуле. Память в разделяемом пуле управляется на основе давности использования, что отлично подходит для кеширования и повторного использования данных. При выделении же больших объемов памяти фрагмент выделяется, используется и после этого он не нужен, т.е. нет смысла его кешировать.

Серверу **Oracle** требовался аналог буферных пулов **RECYCLE** и **KEEP** в буферном кеше. Именно в таком качестве сейчас и выступают большой пул и разделяемый пул. Большой пул — это область памяти, управляемая по принципу пула **RECYCLE**, а разделяемый пул скорее похож на буферный пул **KEEP**: если фрагмент в нем используется часто, он кешируется надолго.

Память в большом пуле организована по принципу "кучи" и управляется с помощью алгоритмов, аналогичных используемым функциями **malloc()** и **free()** в языке C. После освобождения фрагмента памяти он может использоваться другими процессами. В разделяемом пуле отсутствует понятие освобождения фрагмента памяти. Память выделяется, используется, а затем перестает использоваться. Через некоторое время, если эту память необходимо использовать повторно, сервер Oracle позволит изменить содержимое устаревшего фрагмента. Проблема при использовании только разделяемого пула состоит в том, что все потребности в памяти нельзя подогнать под одну мерку.

Большой пул, в частности, используется:

- сервером в режиме **MTS** для размещения области **UGA** в **SGA**;
- при распараллеливании выполнения операторов — для буферов сообщений, которыми обмениваются процессы для координации работы серверов;
- в ходе резервного копирования для буферизации дискового ввода-вывода утилиты **RMAN**.

Как видите, ни одну из описанных выше областей памяти нельзя помещать в буферный пул с вытеснением небольших фрагментов памяти на основе давности использования. Область **UGA**, например, не будет использоваться повторно по завершении сеанса, поэтому ее немедленно надо возвращать в пул. Кроме того, область **UGA** обычно — достаточно большая. Как было показано на примере, где изменялось значение параметра **SORT_AREA_RETAINED_SIZE**, область **UGA** может быть очень большой, и, конечно, больше, чем фрагмент в 4 Кбайта. При помещении области **UGA** в разделяемый пул она фрагментируется на части одинакового размера и, что хуже всего, выделение больших областей памяти, никогда не используемых повторно, приведет к выбрасыванию из пула фрагментов, которые могли бы повторно использоваться. В дальнейшем на перестройку этих фрагментов памяти расходуются ресурсы сервера.

То же самое справедливо и для буферов сообщений. После того как сообщение доставлено, в них уже нет необходимости. С буферами, создаваемыми в процессе резервного копирования, все еще сложнее: они большие и сразу после использования сервером **Oracle** должны "исчезать".

Использовать большой пул при работе в режиме **MTS** не обязательно, но желательно. Если сервер работает в режиме **MTS** в отсутствие большого пула, вся память выделяется из разделяемого пула, как это и было в версиях **Oracle** вплоть до 7.3. Из-за этого производительность со временем будет падать, поэтому такой конфигурации надо избегать. Большой пул стандартного размера будет создаваться при установке одного из следующих параметров инициализации: **DBWn_IO_SLAVES** или **PARALLEL_AUTOMATIC_TUNING**. Рекомендуется задавать размер большого пула явно. Однако стандартное значение не может использоваться во всех без исключения случаях.

Java-пул

Java-пул - это самый новый пул памяти в **Oracle 8i**. Он был добавлен в версии 8.1.5 для поддержки работы **Java-машины** в базе данных. Если поместить хранимую процедуру на языке **Java** или компонент **EJB** (**Enterprise JavaBean**) в базу данных, сервер **Oracle** будет использовать этот фрагмент памяти при обработке соответствующего кода. Одним из недостатков первоначальной реализации **Java-пула** в **Oracle** 8.1.5 было то, что он не отображался командой **SHOW SGA** и не был представлен строками в представлении **V$SGASTAT**. В то время это особенно сбивало с толку, поскольку параметр инициализации **JAVA_POOL_SIZE**, определяющий размер этой структуры, имел стандартное значение 20 Мбайт. Это заставляло людей гадать, почему область **SGA** занимает оперативной памяти на 20 Мбайт больше, чем следует.

Начиная с версии 8.1.6, однако, **Java-пул** виден в представлении **V$SGASTAT**, а также в результатах выполнения команды **SHOW SGA**. Параметр инициализации **JAVA_POOL_SIZE** используется для определения фиксированного объема памяти, отводящегося для Java-кода и данных сеансов. В **Oracle 8.1.5** этот параметр мог иметь значения от 1 Мбайта до 1 Гбайт. В **Oracle 8.1.6** и последующих версиях диапазон допустимых значений уже 32 Кбайта-1 Гбайт. Это противоречит документации, где по-прежнему указан устаревший минимум — 1 Мбайт.

Java-пул используется по-разному, в зависимости от режима работы сервера **Oracle**. В режиме выделенного сервера **Java-пул** включает разделяемую часть каждого **Java-класса**, использованного хоть в одном сеансе. Эти части только читаются (векторы выполнения, методы и т.д.) и имеют для типичных классов размер от 4 до 8 Кбайт.

Таким образом, в режиме выделенного сервера (который, как правило, и используется, если в приложениях применяются хранимые процедуры на языке Java) объем общей памяти для Java-пула имеет весьма невелик; его можно определить исходя из количества используемых Java-классов. Учтите, что информация о состоянии сеансов при работе в режиме разделяемого сервера в области SGA не сохраняется, поскольку эти данные находятся в области UGA, а она, если вы помните, в режиме разделяемого сервера является частью области PGA.

При работе в режиме **MTS Java-пул** включает:

- разделяемую часть каждого **Java-класса**.
- часть области **UGA** для каждого сеанса, используемую для хранения информации о состоянии сеансов.

Оставшаяся часть области **UGA** выделяется как обычно — из разделяемого пула или из большого пула, если он выделен.

Поскольку общий размер **Java-пула** фиксирован, разработчикам приложений необходимо оценить общий объем памяти для приложения и умножить на предполагаемое количество одновременно поддерживаемых сеансов. Полученное значение будет определять общий размер **Java-пула**. Каждая **Java-часть** области

UGA будет увеличиваться и уменьшаться при необходимости, но помните, что размер пула должен быть таким, чтобы части всех областей **UGA** могли поместиться в нем одновременно.