## 1. Понятие СУБД. Основные категории СУБД. Архитектура ANSI-SPARC.

**СУБД** — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

**Основные функции СУБД:**

* Управление данными на дисках.
* Управление данными в ОП (в т.ч., кэширование).
* Журналирование изменений, резервное копирование, восстановление после сбоев.
* Поддержка языков БД (DML + DDL).

**Состав СУБД:**

* Ядро отвечает за управление данными.
* Процессор языка БД транслирует запросы с высокоуровневой языка на низкоуровневый.
* Подсистема поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД.
* Сервисные программы, обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

**Классификация СУБД**

**По модели данных:**

* Иерархические— данные представляются в виде дерева. Пример — LDAP / AD, реестр Windows.
* Сетевые— используют сетевую модель данных. Сетевые базы данных подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию. Частный случай — графовые СУБД. Примеры — HypergraphDB, OrientDB.
* Объектно-ориентированные— используют ОО-модель данных. Эта система управления обрабатывает данные как абстрактные объекты, наделённые свойствами и использующие методы взаимодействия с другими объектами окружающего мира. Пример — InterSystems Cache.
* Реляционные и объектно-реляционные— используют реляционную модель данных (возможно, с частичной поддержкой ООП: объекты, классы и наследование реализованы в структуре бд и языке запросов). Примеры — Oracle, MySQL, PostgreSQL.

**По степени распределённости:**

* Локальные СУБД (все части локальной СУБД размещаются на одном компьютере).
* Распределённые СУБД (части СУБД могут размещаться на двух и более компьютерах).

**По способу доступа к БД:**

* Файл-серверные — данные находятся на файл-сервере (выделенный сервер, предназначенный для выполнения файловых операций ввода-вывода и хранящий файлы любого типа), СУБД — на каждом клиентском компьютере. Доступ СУБД к данным осуществляется через локальную сеть. Примеры — M$ Access, dBase, FoxPro.
* Клиент-серверные— СУБД находятся на сервере вместе с данными и осуществляет доступ к БД непосредственно. Примеры — Oracle, M$ SQL Server, Cache.
* Встраиваемые — СУБД встраивается в приложение, хранит только его данные и не требует отдельной установки. Примеры — SQLite, BerkeleyDB.

**Архитектура ANSI-SPARC**

Предложена в 1975 г. подкомитетом SPARC (Standards Planning And Requirements Commitee) ANSI.

Архитектура СУБД включает в себя 3 уровня:

* Внешний (пользовательский).
* Промежуточный (концептуальный).
* Внутренний (физический).

Почти все современные СУБД соответствуют принципам ANSI-SPARC.

В основе архитектуры ANSI-SPARC лежит концептуальный уровень. В современных СУБД он может быть реализован при помощи представления. Концептуальный уровень описывает данные и их взаимосвязи с наиболее общей точки зрения, — концепции архитекторов базы, используя реляционную или другую модель.

Внутренний уровень позволяет скрыть подробности физического хранения данных (носители, файлы, таблицы, триггеры …) от концептуального уровня. Отделение внутреннего уровня от концептуального обеспечивает так называемую физическую независимость данных.

На внешнем уровне описываются различные подмножества элементов концептуального уровня для представлений данных различным пользовательским программам. Каждый пользователь получает в свое распоряжение часть представлений о данных, но полная концепция скрыта. Отделение внешнего уровня от концептуального обеспечивает логическую независимость данных.

## 2. СУБД Oracle. Архитектура. Подключение, взаимодействие с БД.

**СУБД Oracle**

Исторически первая и наиболее распространённая коммерческая СУБД на основе языка SQL.

По классификации — объектно-реляционная распределённая клиент-сервернаяСУБД.

Очень показательный пример архитектуры Enterprise-level решения.

Первая версия (v2) была выпущена в 1979 г.

Актуальная версия — 12c («cloud» — «облако»).

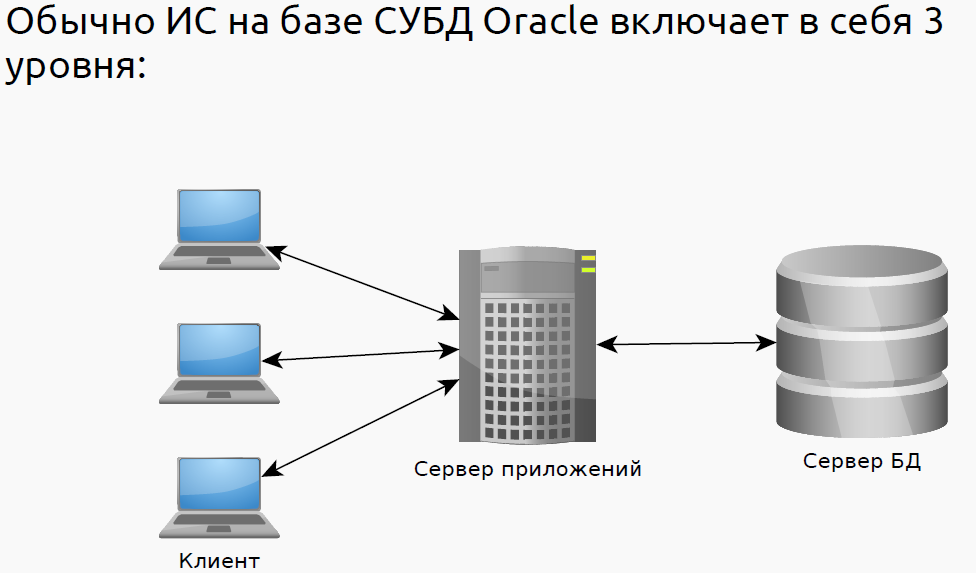
Начиная с версии 3 (1983 г.) реализована поддержка транзакций.

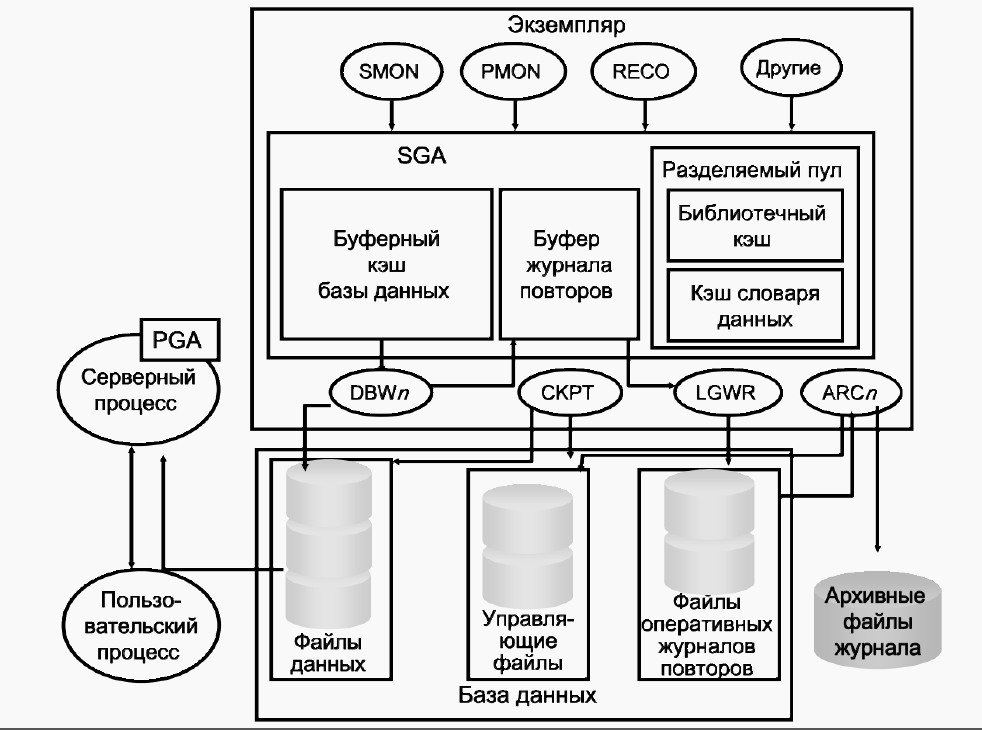
В версии 7 (1992 г.) появилась поддержка PL/SQL.

В версии 8 (1997 г.) реализована поддержка ООП.

В версии 9 (2001 г.) реализована технология RAC (Real Application Cluster) — появилась возможность реализации кластерных БД.

**Архитектура**





База данных Oracle состоит из экземпляра и связанных с ним базами данных. Экземпляр состоит из структур памяти и фоновых процессов. При каждом запуске экземпляра выделяется совместно используемая область памяти, которая называется глобальной системной областью (System Global Area, SGA), а также запускаются фоновые процессы.

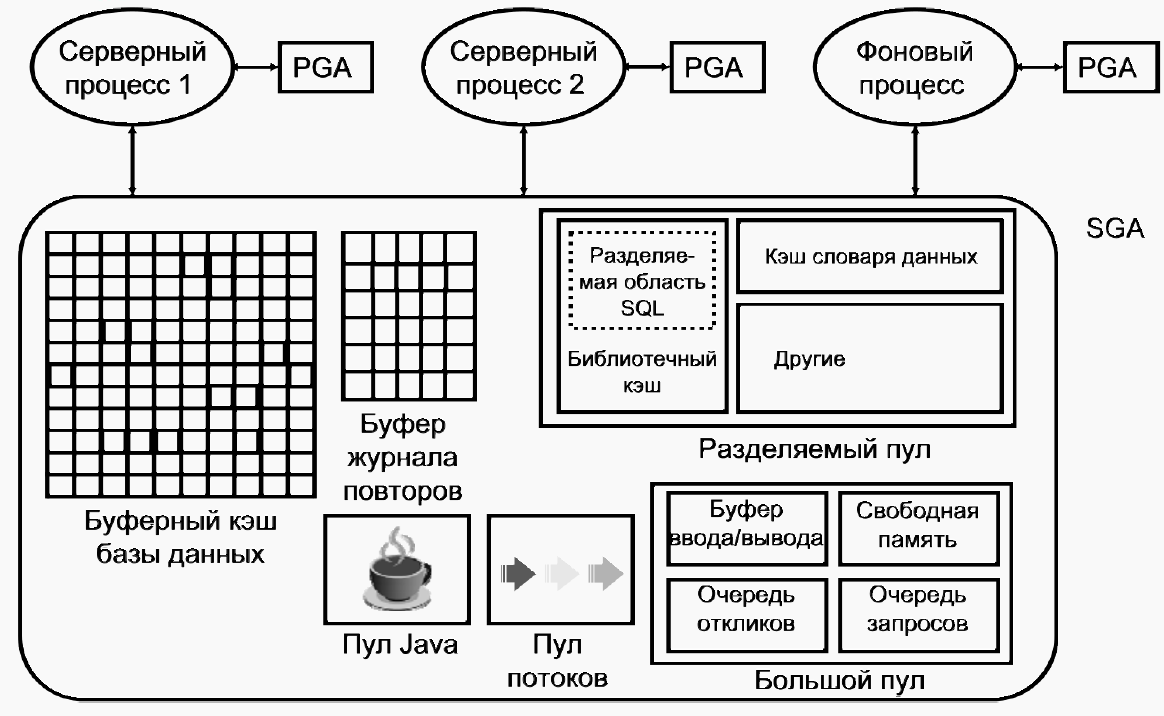
База данных состоит как из физических, так и из логических структур. Так как физические и логические структуры разделены, физическими хранилищами данных можно управлять, не влияя на логические структуры хранения.

Подключение – это канал связи между пользовательским процессом и экземпляром базы данных Oracle.

Сеанс представляет состояние подключения текущего пользователя к экземпляру базы данных. Сеанс длится с момента подключения пользователя и до момента отключения или выхода из приложения базы данных. В случае выделенного подключения сеанс обслуживается постоянным выделенным процессом. Сеанс обслуживается доступным серверным процессом, выбранным из пула – либо средним ярусом, либо разделяемой серверной архитектурой Oracle. Для одного пользователя базы данных Oracle может быть создано множество параллельных сеансов с одним и тем же именем пользователя

1. Экземпляр, запущенный на узле, на котором установлена база данных Oracle, часто называется хостом или сервером базы данных.
2. Пользователь запускает приложение, порождающее пользовательский процесс. Приложение пытается установить соединение с сервером. (Соединение может быть локальным, клиент- серверным или трехъярусным соединением из среднего яруса).
3. Сервер запускает прослушиватель с соответствующим обработчиком Oracle Net Services. Сервер обнаруживает запрос от приложения на соединение и создает выделенный серверный процесс по требованию пользовательского процесса.
4. Пользователь выполняет SQL-оператор DML-типа и подтверждает транзакцию. Например, пользователь изменяет в таблице адрес заказчика и подтверждает изменения.
5. Серверный процесс получает оператор и проверяет разделяемый пул (компонент SGA) на наличие любой разделяемой области SQL, которая содержит подобный SQL-оператор. Если разделяемая область SQL найдена, серверный процесс проверяет полномочия доступа пользователя к запрошенным данным, и оператор обрабатывается с помощью существующей разделяемой области SQL. Если же разделяемая область SQL не найдена, будет выделена новая разделяемая область SQL для синтаксического разбора и обработки оператора.
6. Серверный процесс извлекает все необходимые значения данных из файла (таблицы) фактических данных или из значений, хранящихся в SGA.
7. Серверный процесс изменяет данные в SGA. Поскольку транзакция была подтверждена, процесс LogWriter (LGWR) немедленно записывает ее в файл журнала повторов. Процесс Database Writer (DBWn) окончательно записывает измененные блоки на диск, когда это является целесообразным.
8. Если транзакция была обработана успешно, процесс сервера посылает по сети соответствующее сообщение приложению. В противном случае будет передано сообщение об ошибке.
9. На протяжении всей процедуры выполняются другие фоновые процессы, отслеживая условия, которые требуют вмешательства. Кроме того, сервер базы данных управляет транзакциями других пользователей и предотвращает конкуренцию между транзакциями за одни и те же данные.

## 3. Структура памяти БД Oracle.



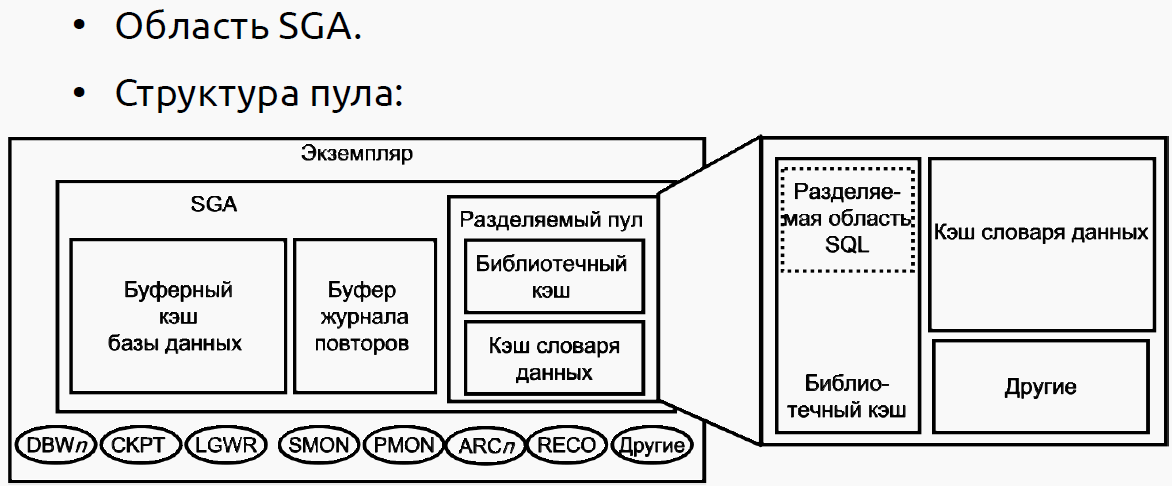
**Буферный кэш БД**

* Является частью SGA.
* Хранит копии блоков данных, считанных их файлов данных.
* Если нужного блока данных нет в кэше, он читается с диска и помещается в кэш.
* Совместно используется всеми параллельно работающими пользователями.
* Управляется сложным алгоритмом, основанным на LRU.

**Буфер журнала повторов**

* Циклический буфер в SGA.
* Хранит информацию об изменениях в БД.
* Содержит записи повторов, в которых хранится информация для повторного применения изменений, внесенных операциями DML и DDL.
* Записи повторов используются для восстановления базы данных в случае необходимости.
* Фоновый процесс LGWR производит запись буфера журнала повторов на диск.

**Разделяемый пул**



**Выделение памяти в разделяемом пуле**

* Данные вытесняются из пула по алгоритму LRU.
* Серверный процесс проверяет разделяемый пул на предмет наличия разделяемой области SQL для идентичного оператора.
* Серверный процесс выделяет частную область SQL по запросу сеанса.
* В некоторых случаях разделяемая область SQL сбрасывается целиком:

ALTER SYSTEM FLUSH SHARED\_POOL;

**Большой пул**

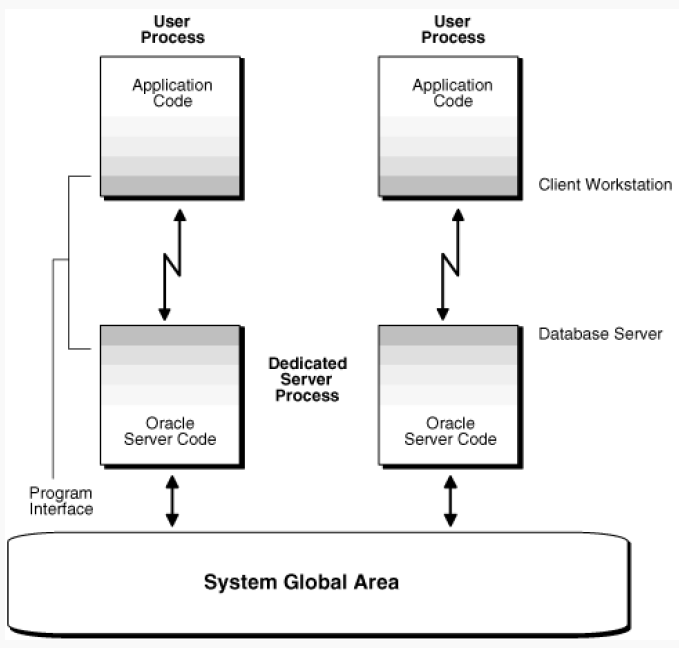
* Необязательная область SGA.
* Выделяется вручную администратором БД.
* В отличие от разделяемого пула, нет автоматического освобождения памяти по LRU.
* Может быть использован:
  + Для операций передачи большого объёма данных между разными БД.
  + Для операций резервного копирования / восстановления.
* Размер задаётся параметром инициализации LARGE\_POOL\_SIZE.

## 4. Архитектура процессов.

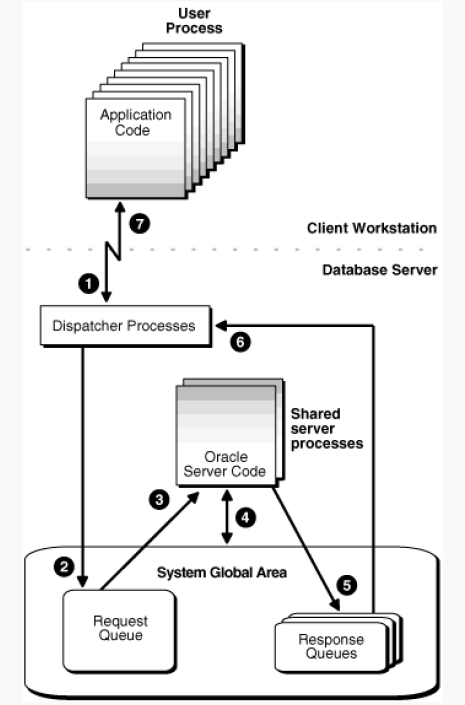
2 вида процессов:

* Пользовательские процессы. Запускаются в момент подключения пользователя к БД.
* Процессы базы данных:
  + Серверный процесс: подключается к экземпляру Oracle и запускается при установлении сеанса пользователем.
  + Фоновые процессы: запускаются при запуске экземпляра Oracle.

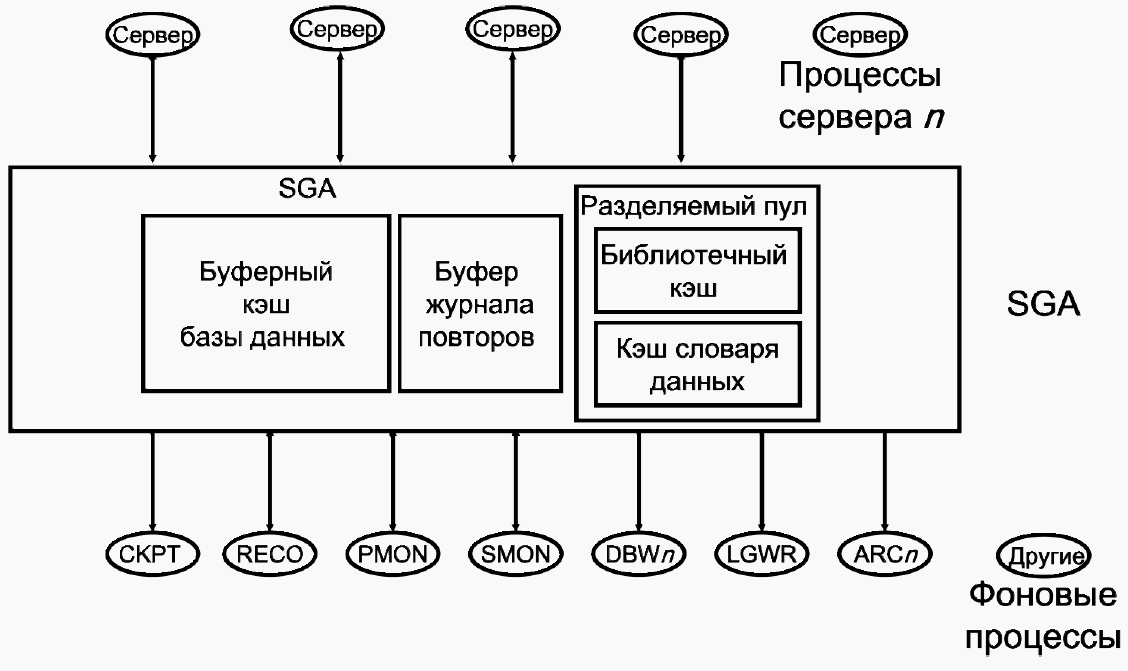
В режиме выделенного сервера каждому пользовательском у процессу создаётся свой «персональный» серверный.



В режиме разделяемого сервера каждому серверные процессы выделяются диспетчером из специального пула.



**Структуры процессов**



## 5. Запись данных в БД, логика работы процесса записи.

Процесс DBWn записывает измененные (заполненные) буферы из буферного кэша базы данных на диск:

* асинхронно во время выполнения другой обработки;
* периодически для перехода к следующей контрольной точке.

Администратор может сконфигурировать СУБД на использование до 20 параллельных процессов DBW с помощью параметра инициализации DB\_WRITER\_PROCESSES.

Изменения записываются в файлы в том порядке, в котором они были сделаны (согласно SCN — System Change Number).

Для этого используется LRUW (LRU-Write) — список заполненных буферов в кэше, отсортированный по SCN.

При записи данных в файл DBW одновременно «перемещает» указатель на контрольную точку, с которой будет начато восстановление в случае сбоя (инкрементальная установка контрольных точек).

Процесс записи в базу данных (DBWn) записывает содержимое буферов в файлы данных. Процессы DBWn осуществляют запись измененных (заполненных) буферов из буферного кэша базы данных на диск. Хотя одного процесса записи в базу данных (DBW0) достаточно для большинства систем, можно настроить дополнительные процессы (DBW1 – DBW9 и DBWa – DBWj), чтобы повысить производительность записи на случай интенсивного изменения данных в системе. Дополнительные процессы DBWn бесполезны в однопроцессорных системах.

Если буфер в буферном кэше базы данных изменяется, он помечается как заполненный и добавляется в список LRUW (LRU-запись) заполненных буферов, отсортированный по номеру SCN. Таким образом, порядок буферов совпадает с порядком операций повтора, записанных в журналы повторов для таких измененных буферов. Когда количество доступных буферов в буферном кэше становится меньше внутреннего порогового значения (при котором серверным процессам становится тяжело получать доступные буферы), DBWn записывает заполненные буферы в файлы данных в порядке их изменения, следуя порядку в списке LRUW.

SGA содержит структуру памяти, в которой находится журнальный байтовый адрес (RBA) позиции в потоке повторов, с которой следует начать восстановление в случае отказа экземпляра. Данная структура служит указателем на повтор и записывается в контрольный файл процессом CKPT один раз в три секунды. Поскольку процесс DBWn записывает заполненные буферы в порядке следования SCN, а журнал также формируется по порядку номеров SCN, то каждый раз, когда процесс DBWn записывает заполненные буферы из списка LRUW, он также перемещает указатель в структуре памяти SGA. Таким образом, при восстановлении экземпляра (если это необходимо) чтение начинается с приблизительно правильного размещения, что позволяет избежать лишних операций ввода-вывода. Это называется инкрементной установкой контрольных точек.

Примечание: процесс DBWn может выполнять запись и в других ситуациях (например, если табличные пространства становятся доступными только для чтения или переводятся в автономный режим). В таких случаях инкрементная установка контрольной точки не происходит, так как в базу данных записываются заполненные буферы, принадлежащие только соответствующим файлам данных, вне независимости от порядка SCN.

Алгоритм LRU оставляет в буферном кэше блоки, обращение к которым происходит наиболее часто. Тогда при записи буфера на диск он с минимальной вероятностью будет содержать данные, которые могут вскоре понадобиться.

Параметр инициализации DB\_WRITER\_PROCESSES задает количество процессов DBWn. Максимальное количество процессов DBWn равно 20. Если данный параметр не указан пользователем при запуске, Oracle Database определяет значение параметра DB\_WRITER\_PROCESSES в зависимости от количества центральных процессоров и групп процессоров.

Процесс DBWn записывает заполненные буферы на диск при таких условиях:

* Когда серверному процессу после сканирования порогового количества буферов не удается найти свободный многократно используемый буфер, он подает процессу DBWn сигнал начинать запись. Процесс DBWn записывает заполненные буферы на диск в асинхронном режиме во время выполнения других вычислений.
* Процесс DBWn периодически записывает буферы, чтобы передвинуть контрольную точку, которая является позицией в потоке (журнале) повторов и с которой начинается восстановление экземпляра. Эта позиция определяется наиболее старым заполненным буфером в буферном кэше.
* Во всех случаях процесс DBWn осуществляет пакетную (многоблоковую) запись, что повышает продуктивность. Количество блоков, записываемых при многоблоковой записи, зависит от используемой операционной системы.

## 6. Запись журнала повторов, логика работы процесса записи.

**Процесс LogWriter (LGWR)** записывает буфер журнала повторов в файл журнала повторов на диске.

Запись осуществляется:

* когда пользовательский процесс фиксирует транзакцию;
* когда буфер журнала повторов заполняется на одну треть;
* перед тем как процесс DBW*n* записывает измененные буферы на диск.

После того, как данные из буфера журнала повторов записаны на диск, серверные процессы могут записать на их место новые данные.

Данные в файл журнала повторов записываются сразу же после того, как пользователь вызвал оператор COMMIT. Т.е., данные в журнал повторов обычно записываются раньше, чем в файлы данных. Это называется механизмом быстрой фиксации транзакции.

Процесс LogWriter (LGWR) осуществляет управление буфером журнала повторов, сохраняя его записи в файл журнала повторов на диске. Процесс LGWR записывает все записи повторов, скопированные в буфер с момента последней записи на диск.

Буфер журнала повторов является циклическим. После того как процесс LGWR сохранит записи повторов из буфера в файл журнала повторов, серверные процессы смогут копировать новые записи на место записей в буфере журнала повторов, которые уже записаны на диск. Обычно LGWR выполняет запись достаточно быстро, чтобы обеспечить в буфере свободное место для новых записей даже при интенсивном обращении к журналу повторов. Процесс LGWR записывает на диск одну непрерывную часть буфера.

Процесс LGWR выполняет запись:

* когда пользовательский процесс pLogWriter фиксирует транзакцию
* когда буфер журнала повторов заполняется на одну треть
* перед тем как процесс DBWn записывает измененные буферы на диск (при необходимости)

Прежде чем процесс DBWn начнет запись измененного буфера, на диск должны быть сохранены все записи журнала повторов, связанные с изменениями буфера (протокол упреждающей записи). Если процесс DBWn обнаруживает, что некоторые записи журнала повторов не сохранены, он сигнализирует процессу LGWR, чтобы тот сохранил эти записи на диск, и ждет завершения записи буфера журнала повторов, прежде чем приступить к записи буферов данных. Процесс LGWR выполняет запись в текущую группу журналов. Если один из файлов в группе поврежден или недоступен, процесс LGWR продолжает запись в остальные файлы группы и записывает сообщение об ошибке в файл трассировки LGWR и системный журнал предупреждений. Если в группе повреждены все файлы или группа недоступна, так как не была заархивирована, процесс LGWR не сможет продолжить работу.

Когда пользователь выдает оператор COMMIT, процесс LGWR помещает запись фиксации транзакции в буфер журнала повторов. После этого LGWR немедленно записывает на диск буфер журнала повторов и записи повторов из транзакции. Внесение соответствующих изменений в блоки данных задерживаются до момента наиболее эффективной записи. Это называется механизмом быстрой фиксации транзакции. Простейшая операция записи повтора, которая содержит запись о фиксации транзакции, является единственным событием, определяющим, была ли зафиксирована транзакция. Oracle Database возвращает код успешной фиксации транзакции, хотя буферы данных еще не записаны на диск.

Если необходимо дополнительное пространство в буфере, процесс LGWR иногда сохраняет записи журнала повторов перед фиксацией транзакции. Эти записи становятся постоянными только при последующей фиксации транзакции. Когда пользователь фиксирует транзакцию, ей присваивается системный номер изменения (SCN), который Oracle Database записывает в журнал повторов вместе с записями повторов из транзакции. Номера SCN записываются в журнал повторов, чтобы операции восстановления в Real Application Clusters и распределенных базах данных были синхронизированы.

В моменты интенсивной нагрузки процесс LGWR может записывать файл журнала повторов при помощи групповой фиксации транзакций. Предположим, что пользователь фиксирует транзакцию. Процесс LGWR должен записать на диск записи повторов из этой транзакции. В это время другие какие-либо пользователи выдают оператор COMMIT. Однако процесс LGWR не может выполнить запись в файл журнала повторов, чтобы зафиксировать эти транзакции, пока не закончит предыдущую операцию записи. После сохранения в файл журнала повторов записей из первой транзакции весь список записей повторов из ожидающих транзакций (еще не зафиксированных) может быть записан на диск одной операцией. Это потребует меньшего количества операций ввода/вывода, чем отдельная обработка каждой записи транзакции. Следовательно, Oracle Database сводит к минимуму количество операций дискового ввода/вывода и максимально повышает производительность процесса LGWR. Если высокая интенсивность поступления запросов фиксации не снижается, каждая запись (выполняемая процессом LGWR) из буфера журнала повторов может содержать по несколько записей фиксации транзакций.

## 7. Создание контрольной точки, процесс архивирования журнала повторов.

**Контрольная точка**– это структура данных, определяющая SCN в потоке повторов базы данных.

Контрольные точки записываются в управляющий файл и в заголовок каждого из файлов данных. Эту операцию выполняет процессCKPT. Процесс CKPT не записывает блоки на диск, эту операцию выполняет процесс DBWn.

Запись номеров SCN в заголовки файлов гарантирует, что все изменения, внесенные в блоки базы данных до фиксации данного номера SCN уже записаны на диск.

Контрольная точка – это структура данных, определяющая системный номер изменения (SCN) в потоке повторов базы данных. Контрольные точки записываются в управляющий файл и в заголовок каждого из файлов данных. Они являются важнейшим элементом процесса восстановления.

При фиксации контрольной точки Oracle Database нужно обновить заголовки всех файлов данных, чтобы записать сведения о контрольной точке. Эту операцию выполняет процесс CKPT. Процесс CKPT не записывает блоки на диск, эту операцию выполняет процесс DBWn. Запись номеров SCN в заголовки файлов гарантирует, что все изменения, внесенные в блоки базы данных до фиксации данного номера SCN, уже записаны на диск.

Статические контрольные точки DBWR, которые отображает в Oracle Enterprise Manager монитор SYSTEM\_STATISTICS, показывают количество выполненных запросов создания контрольной точки.

**Процессы архиваторов (ARCn)**

Копируют файлы журнала повторов на указанное устройство хранения после заполнения журнала.

Могут собирать данные для восстановления транзакций.

Функционируют, только если БД работает в режиме ARCHIVELOG.

Можно изменить максимальное количество процессов архиваторов при помощи параметра инициализации LOG\_ARCHIVE\_MAX\_PROCESSES.

Процессы архиваторов (ARCn) копируют файлы журнала повторов на указанное устройство хранения после переключения журнала. Процессы ARCn функционируют, только если БД работает в режиме ARCHIVELOG и при этом включено автоматическое архивирование.

Если ожидается, что архивирование вызовет интенсивную загрузку (например, при пакетной загрузке данных), можно увеличить максимальное количество процессов архиваторов при помощи параметра инициализации LOG\_ARCHIVE\_MAX\_PROCESSES. Оператор ALTER SYSTEM позволяет изменять значение данного параметра динамически, увеличивая или уменьшая количество процессов ARCn.

## 8. Установка БД. Основные задачи администратора при установке в среде Unix-подобных систем.

**Переменные среды**, необходимые для установки Oracle.

* ORACLE\_BASE. Устанавливает путь к ≪корню≫ иерархии каталогов Oracle:

export ORACLE\_BASE=/u01/app/oracle

* ORACLE\_HOME. Устанавливает путь к ≪корневому≫ каталогу БД. Этот путь свой для каждого экземпляра БД:

export ORACLE\_HOME=$ORACLE\_BASE/product/11.1.0/db\_1

* ORACLE\_SID. Задаёт имя экземпляра Oracle. Значение по умолчанию - ORCL. Формат — строка, состоящая из цифр и букв и начинающаяся с буквы.
* NLS\_LANG. Устанавливает язык и кодировку БД. Формат - язык\_местность.набор символов:

export NLS\_LANG=RUSSIAN\_CIS.AL32UTF8

**Способы установки Oracle**

С помощью Oracle Universal Installer — в интерактивном режиме с помощью графической утилиты (написанной на Java):

./runInstaller

«Silent Mode» — с помощью файла конфигурации, (Response File) заданного в ходе одной из предыдущих установок:

./runInstaller –record -responseFile

./runInstaller -silent -responseFile

responsefilename

**Установка под \*nix**

После завершения работы OUI необходимо выполнить ряд скриптов из-под суперпользователя:

$ su

# password:

# cd /u01/app/oracle/oraInventory

# ./orainstRoot.sh

# cd /u01/app/oracle/product/11.1.0/db\_1

# ./root.sh

**Способы создания БД**

* «Вручную» — в SQL\*Plus, с помощью команды CREATE DATABASE.
* С помощью графической утилиты Database Configuration Assistant (DBCA)

**Создание БД**

1. Задаём Oracle System Identifier (SID):

ORACLE\_SID=mynewdb

export ORACLE\_SID

2. Ещё раз проверяем, задана ли переменная окружения ORACLE\_HOME:

echo $ORACLE\_HOME

/u01/app/oracle/product/11.1.0/db\_1

3. Выбираем метод аутентификации, который будет использоваться в БД:

* с помощью файла паролей;
* на уровне пользователей ОС.

4. Создаём текстовый файл с параметрами инициализации БД. Он должен содержать как минимум 3 параметра:

* DB\_NAME — имя БД (максимум 8 символов).
* CONTROL\_FILES — список управляющих файлов БД.
* MEMORY\_TARGET — общее количество памяти, которое будет выделено экземпляру БД.

5. Создаём экземпляр Oracle (только для Windows, в \*NIX экземпляр создаётся автоматически):

oradim -NEW -SID sid -STARTMODE MANUAL -PFILE file

6. Запускаем SQL\*Plus без подключения к БД:

$ sqlplus /nolog

7. Подключаемся к экземпляру с привилегиями SYSDBA:

SQL> CONNECT SYS AS SYSDBA

или

SQL> CONNECT / AS SYSDBA

8. Создаём бинарный файл параметров инициализации сервера на основании созданного ранее текстового:

SQL> CREATE SPFILE FROM PFILE;

9. Запускаем экземпляр Oracle без монтирования БД:

SQL> STARTUP NOMOUNT;

10. Вызываем команду CREATE DATABASE. Возможны 2 варианта вызова:

* Перечисляем все параметры конфигурации БД в аргументах;
* Все параметры конфигурации БД читаются из файла.

11. Создаём пользовательские и дополнительные табличные пространства:

CREATE TABLESPACE apps\_tbs LOGGING

DATAFILE

'/u01/app/oracle/oradata/mynewdb/apps01.dbf'

SIZE 500M REUSE AUTOEXTEND ON NEXT 1280K

MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL;

CREATE TABLESPACE indx\_tbs LOGGING

DATAFILE

'/u01/app/oracle/oradata/mynewdb/indx01.dbf'

SIZE 100M REUSE AUTOEXTEND ON NEXT 1280K

MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL;

12. Заполняем первичными данными представления словаря данных:

@?/rdbms/admin/catalog.sql

@?/rdbms/admin/catproc.sql

@?/sqlplus/admin/pupbld.sql

13 (…). Настраиваем резервное копирование БД, автозапуск БД при рестарте сервера, и т.д.

## 9. Архитектура хранения базы данных. Подход OFA.

Файлы, из которых состоит БД, делятся на следующие категории:

* **Управляющие файлы** содержат данные о самой БД (информация о физической структуре БД). Это критически важные файлы для БД, без них невозможно открыть файлы данных и получить доступ к БД.
* **Файлы данных** содержать данные пользователя или приложения БД, а также метаданные и словарь данных (набор таблиц и представлений, в которых хранится информация о базе). Файл данных может принадлежать только одному табличному пространству.
* **Оперативные файлы журналов повторов** используются для восстановления экземпляра БД. Если происходить сбой сервера БД, при котором не теряются файлы данных, экземпляр позволит восстановить БД с помощью информации, содержащейся в этих файлах.

Помимо перечисленных, экземпляр БД использует:

* **Файл параметров.** Используется для определения конфигурации экземпляра для запуска.
* **Файл паролей.** Позволяет удаленно подключаться к БД пользователям sysdba, sysoper и sysasm.
* **Резервные копии файлов.** Используются для восстановления БД.
* **Архивные файлы журнала повторов.** Содержат непрерывную историю изменений данных (повторных операций), которую создает экземпляр. С помощью этих файлов и резервной копии БД можно восстановить утраченные файлы данных.
* **Файлы трассировки.** Любой серверный или фоновый процесс может выполнять запись в определенный файл трассировки. При обнаружении процессом внутренней ошибки он записывает дамп информации об ошибке в свой файл трассировки.
* **Файлы журнала предупреждений.** Журнал предупреждений БД – это хронологический журнал сообщений и ошибок. Каждый экземпляр использует один файл журнала предупреждений.

**OFA** – метод конфигурирования БД Oracle или других БД. OFA использует преимущества ОС и дисковых подсистем и позволяет создать простую для администрирования конфигурацию, которая обеспечит максимальную гибкость растущих и высокопроизводительных баз данных.

OFA предназначена для:

* организации большого количества сложного ПО и данных на диске для избегания узких мест на устройствах и снижения производительности;
* упрощения повседневных задач администрирования (например, резервного копирования ПО и данных), которые нередко уязвимы к повреждению данных;
* упрощения переключения между несколькими БД Oracle;
* контроля и управления ростом базы данных;
* снижения фрагментации свободного пространства в словаре данных, изоляции прочих фрагментированных областей и сведения конкуренции за ресурсы к минимуму.

**Синтаксис точек монтирования:** в именах всех точек монтирования необходимо использовать синтаксис /*pm*, где *p* – строчная константа, а *m* – уникальный ключ фиксированной длины (обычно двухразрядное число), который используется для различения точек монтирования. Примерами точек монтирования, соответствующих OFA, являются точки /u01 и /u02.

**Синтаксис домашних каталогов:** в именах всех домашних каталогов необходимо использовать синтаксис /*pm*/*h*/*u*, где *pm -* имя точки монтирования, *h* – стандартное имя каталога, а *u* – имя владельца каталога. Примеры домашних каталогов в соответствии с OFA:

/u01/app/oracle

/u01/home/oracle

**Синтаксис каталогов программ:** каждую версию ПО Oracle следует хранить в каталоге с именем вида /*pm*/*h*/*u*/product/*v*. Здесь product – это литерал, а *v* – переменная для номера версии. Правильный синтаксис позволяет использовать возможности OFA для одновременного запуска нескольких версий программного обеспечения. Совместимая с OFA установка Oracle Database 11*g* версии 11.1.0 имеет следующий вид:

/u01/app/oracle/product/11.1.0/db\_1

**Синтаксис подкаталогов:** Чтобы упростить организацию данных администрирования, все файлы администрирования для базы данных следует хранить в подкаталогах следующего вида: /*h*/admin/*d*/*a*/. Здесь h – домашний каталог владельца ПО Oracle, admin – литерал, *d* – имя базы данных, *a* – подкаталог для каждого из файлов администрирования БД.

Ниже приведен список этих подкаталогов для файлов администрирования:

* adhoc: специальные SQL-сценарии для конкретной базы данных;
* arch: архивные файлы журнала повторов;
* adump: файлы аудита (присвойте параметру инициализации AUDIT\_FILE\_DEST значение каталога adump. Время от времени этот каталог следует очищать.)
* Create: программы, используемые для создания базы данных;
* Exp: файлы экспорта базы данных;
* Logbook: файлы с записями о состоянии и истории базы данных;
* Pfile: файлы параметров экземпляра.

**Синтаксис файлов БД.** Ниже приведены правила именования файлов базы данных,

позволяющие легко определять эти файлы:

* Управляющие файлы: /*pm*/*q*/*d*/control*n*.ctl
* Файлы журнала повторов: /*pm*/*q*/*d*/redo*n*.log
* Файлы данных: /*pm*/*q*/*d*/*tn*.dbf

В именах файлов использованы следующие переменные:

* pm: имя точки монтирования (описано выше);
* q: строка, отличающая данные Oracle от других файлов (обычно используется ORACLE
* или oradata);
* d: значение параметра инициализации DB\_NAME (имя БД);
* t: имя табличного пространства Oracle;
* n: двухразрядная строчная переменная.

Примечание: в каталоге /pm/q/d/ следует хранить только управляющие файлы, файлы

журнала повторов и файлы данных, связанные с базой данных d.

## 10. Параметры инициализации экземпляра БД. Файлы параметров.

**Файлы параметров инициализации БД** считываются экземпляром Oracle при его запуске.

Существует 2 типа файлов параметров:

* Файлы параметров сервера (SPFILE) — двоичный файл, чтение и запись в который осуществляет сервер БД. Самостоятельно изменять этот файл нельзя. Имя по умолчанию — spfile<SID>.ora.
* Текстовый файл параметров инициализации — может быть только считан сервером, но не записан. Настройки параметров инициализации необходимо задавать и изменять вручную (с помощью текстового редактора). Имя по умолчанию — init<SID>.ora. При наличии SPFILE этот файл игнорируется.

Существует два типа **параметров инициализации**:

* Статические параметры — могут быть изменены только путем редактирования файлов init.ora и SPFILE. Чтобы изменения вступили в силу, необходимо остановить и перезапустить БД.
* Динамические параметры — могут изменяться при работающей БД. Существует два типа динамических параметров:
  + Параметры уровня сеанса — оказывают влияние только на текущий сеанс. Пример — параметры поддержки национальных языков (NLS).
  + Параметры уровня системы — оказывают влияние на всю базу данных и все сеансы. Пример — параметры, отвечающие за сброс данных разделяемого пула или параметры расположения архивного журнала. Действие данных параметров зависит от настройки SCOPE.х.

Динамические параметры можно изменять с помощью команд ALTER SESSION

и ALTER SYSTEM:

SQL> ALTER SESSION SET NLS\_DATE\_FORMAT ='mon dd yyyy';

Некоторые дополнительные параметры инициализации БД:

* DB\_FILES — максимальное количество файлов БД.
* PROCESSES — максимальное количество параллельных пользовательских процессов.
* DB\_BLOCK\_SIZE — размер блока данных БД (в байтах; по умолчанию — 8 КБ).
* DB\_CACHE\_SIZE — размер блока буферного кэша БД (в байтах; по умолчанию — 48 МБ для однопроцессорной системы).
* SGA\_TARGET — общий размер SGA (в байтах).

**Пример файла параметров инициализации**

db\_name='ORCL'

memory\_target=1G

processes = 150

db\_block\_size=8192

db\_domain=cs.ifmo.ru

db\_recovery\_file\_dest='<ORACLE\_BASE>/flash\_recovery\_area'

db\_recovery\_file\_dest\_size=2G

diagnostic\_dest='<ORACLE\_BASE>'

dispatchers='(PROTOCOL=TCP) (SERVICE=ORCLXDB)'

open\_cursors=300

remote\_login\_passwordfile='EXCLUSIVE'

undo\_tablespace='UNDOTBS1'

# You may want to ensure that control files are created on

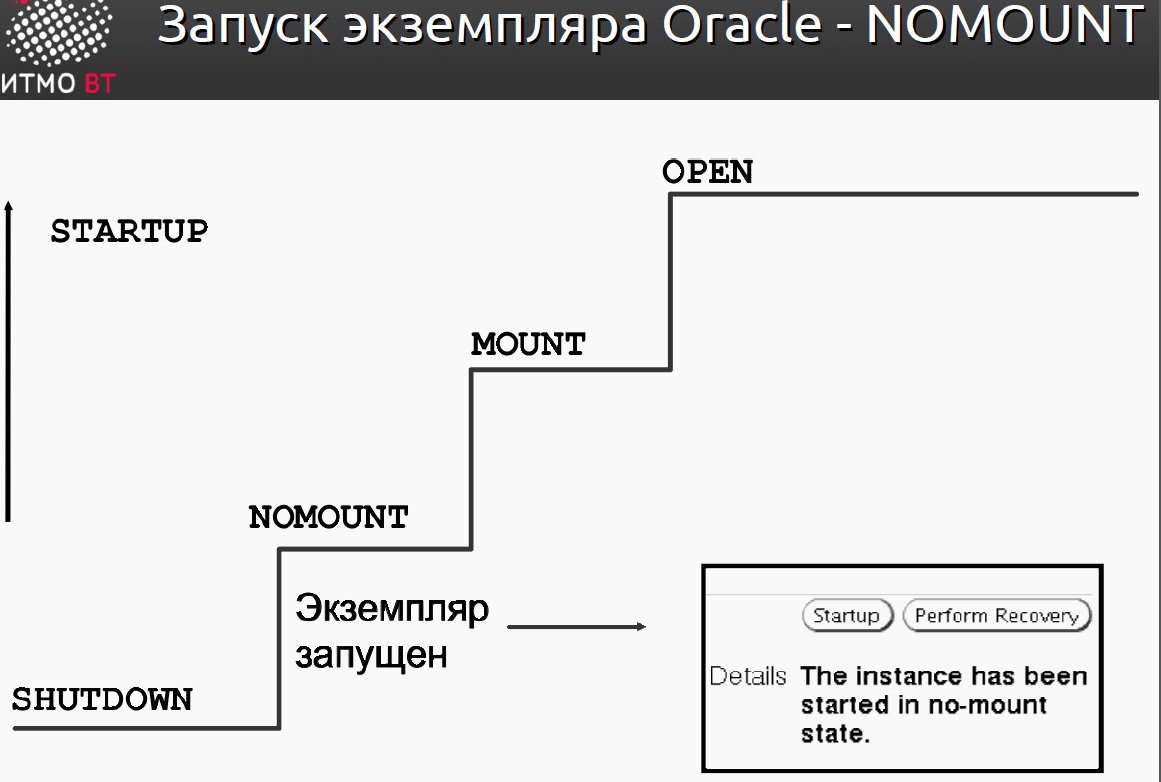
separate physical

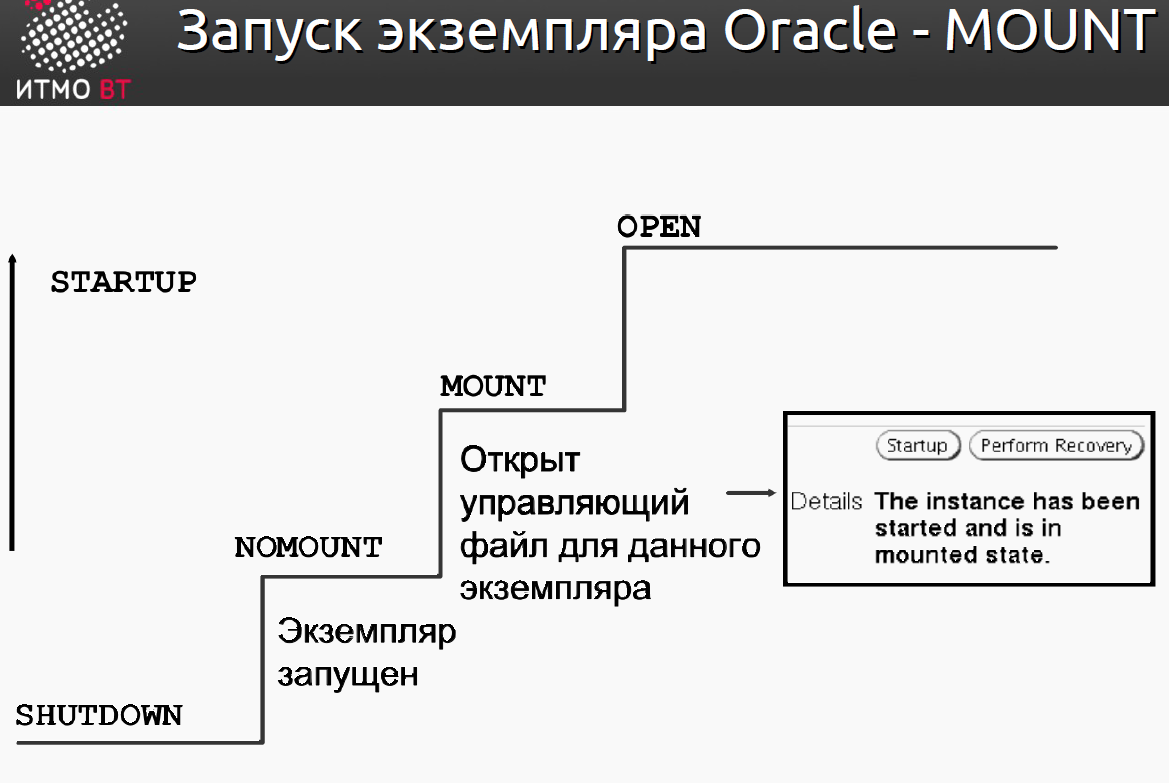
# devices

control\_files = (ora\_control1, ora\_control2)

compatible ='12.0.0'

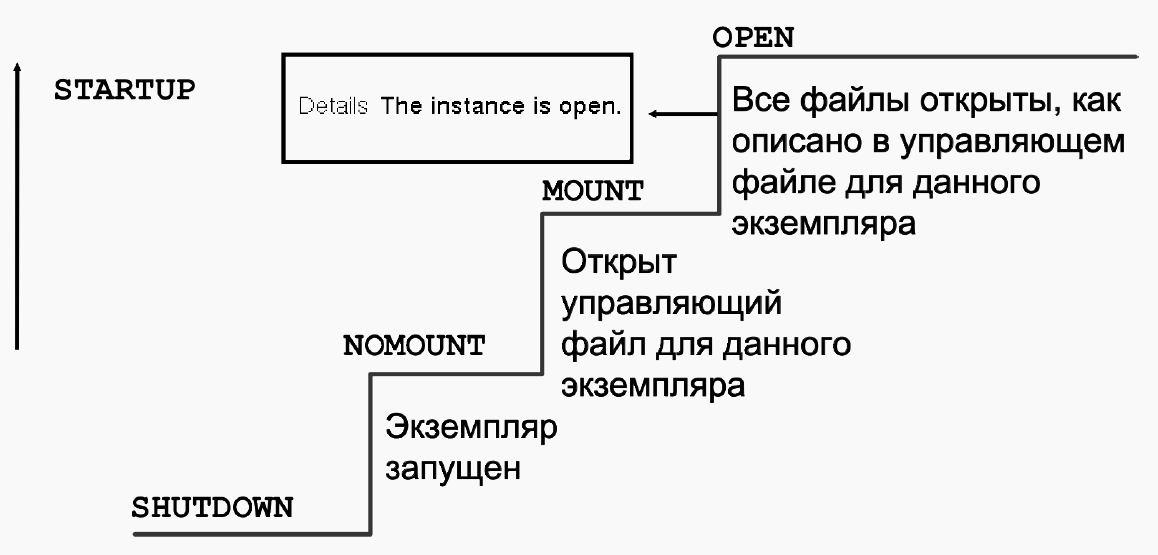
## 11. Запуск и остановка экземпляра БД. Режимы запуска и остановки.





При **монтировании БД** выполняются следующие операции:

* БД ассоциируется с ранее запущенным экземпляром.
* Выполняется поиск и открытие управляющих файлов, указанных в файле параметров.
* Производится чтение управляющих файлов для получения сведений об именах и состоянии файлов данных и интерактивных файлах журналов повтора.



**При открытии БД** выполняются следующие операции:

* Открытие файлов данных.
* Открытие оперативных файлов журнала повторов.

**Команда запуска экземпляра**

SQL> startup

Запускает экземпляр, ассоциирует с ним файлы БД, а затем монтирует и открывает БД.

SQL> startup nomount

Запускает экземпляр без монтирования БД.

SQL> alter database mount;

Монтирует и открывает БД, находящуюся в состоянии NOMOUNT.

**2 способа остановки экземпляра**:

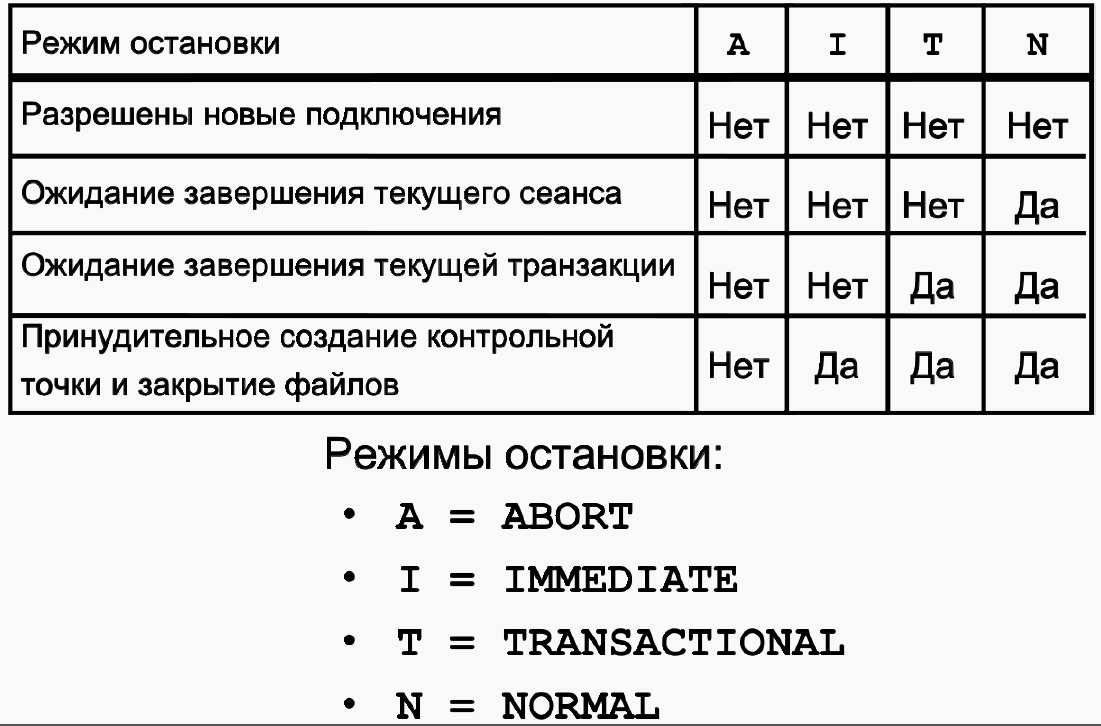
* Через веб-интерфейс Enterprise Manager.
* «Вручную» — с помощью SQL\*Plus:

SQL> shutdown

SQL> shutdown transactional

SQL> shutdown immediate

SQL> shutdown abort



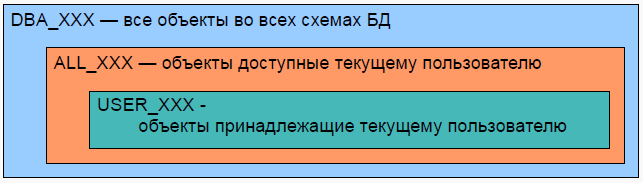
## 12. Словарь данных и динамические представления V$ и GV$.

**Словарь данных** – набор доступных только для чтения таблиц и представлений, которые содержат различную инфо о БД:

* Информацию обо всех объектах схемы БД (таблицах, представлениях, индексах, кластерах, синонимах, последовательностях, процедурах, функциях, пакетах, триггерах и т. д.).
* Информацию о том, сколько дискового пространства выделено объектам схемы, и какой процент этого пространства уже использован.
* Информацию о значениях полей таблиц по умолчанию.
* Информацию о существующих в БД ограничениях целостности.
* Сведения обо всех пользователях БД, их ролях и выданных привилегиях.
* Результаты текущего аудита — кто в данный момент имеет обращается и/или модифицирует объекты схемы.

Словарь данных состоит из двух уровней:

* Системные таблицы содержат инфу в логичном с точки зрения архитектуры системы виде.
* Представления содержат ту же инфу в более удобном для чтения и обработки формате.



Представление производительности:

* Отражают текущее состояние экземпляра БД.
* Информация динамически генерируется в зависимости от состояния.
* Префикс V$ - представления производительности экземпляра БД.
* Префикс GV$ - глобальные представления узлов кластера RAC.

Полезные представления:

* V$SYSTEM\_EVENT — содержит общесистемную информацию о ресурсах, которых ждет весь экземпляр.
* V$SESSION\_EVENT — список событий, которые приходилось ждать в каждом сеансе.
* V$SESSION\_WAIT — детальная посеансовая информация о ресурсах, которые сеанс ожидает в данный момент или ждал в последний раз.
* V$SESSION — информация о сеансе, в том числе о событии, которое ожидает сеанс в данный момент или ждал в последний раз

## 13. Методы разрешения имен, настройка псевдонимов.

Разрешение имен Oracle Net - процесс определения адреса, на котором запущен прослушиватель, порт, за которым следит, протокол и имя службы, с которой нужно установить соединение.

* Easy Connect - использует строку соединения TCP/IP. Информация передается прямо в ней: <username>/<password>@<hostname>:<port>/<service\_name>
  + используется по умолчанию
  + не требует настройки на стороне клиента
  + поддерживает только протокол TCP/IP
  + не поддерживает failover соединения, маршрутизацию от источника и балансировку нагрузки
* Локальное  - файл $TNS\_ADMIN/tnsnames.ora хранит список известных служб на стороне сервера и выбрать службу можно по псевдониму
  + требуется файл разрешения имен на стороне клиента
  + поддерживаются все протоколы Oracle Net
  + поддерживает failover соединения, маршрутизацию от источника и балансировку нагрузки
* На сервере каталогов - Админ добавляет службу в каталог LDAP - все клиенты могут к нему обратиться
  + Требуется протокол LDAP и загрузка информации
  + о разрешении имен Oracle Net:
    - сервер Oracle Internet Directory;
    - службы Microsoft Active Directory.
  + Поддерживаются все протоколы Oracle Net.
  + Поддерживаются дополнительные параметры соединения.
* Внешнее - концептуально как предыдущее, только со сторонней службой разрешения имен (Network Information Service, Distributing Computer Environment)
  + Используется сторонняя служба разрешения имен.
  + Она включает в себя:
    - службу внешнего разрешения имен Network Information Service (NIS);
    - Distributed Computing Environment (DCE) Cell Directory Services (CDS).

Псевдонимы можно настроить через графическое приложение Enterprise manager,  выбрав метод разрешения имен и указав адрес хоста, порт, протокол и имя службы

**Настройка псевдонимов служб**

Чтогбы создать локальный псевдоним службы Oracle Net, в раскрывающемся списке «Administer»

(Администрирование) выберите пункт «Local Naming» (Локальное разрешение имен) и нажмите

кнопку «Go» (Перейти). Затем нажмите кнопку «Create» (Создать).

Чтобы настроить псевдонимы служб для разрешения имен в сервере каталогов, выберите пункт

«Directory Naming» (Разрешение имен в сервере каталогов), а не «Local Naming» (Локальное

разрешение имен).

**Примечание.** Если разрешение имен на сервере каталогов не настроено, пункт «Directory Naming»

(Разрешение имен в сервере каталогов) выбрать невозможно. Разрешение имен на сервере каталогов

рассматривается в курсе *Управление идентификаторами с помощью Oracle Enterprise*, а также в

руководстве *Oracle Advanced Security Administration*.

На странице создания имени сетевой службы в поле «Net Service Name» (Имя службы) введите

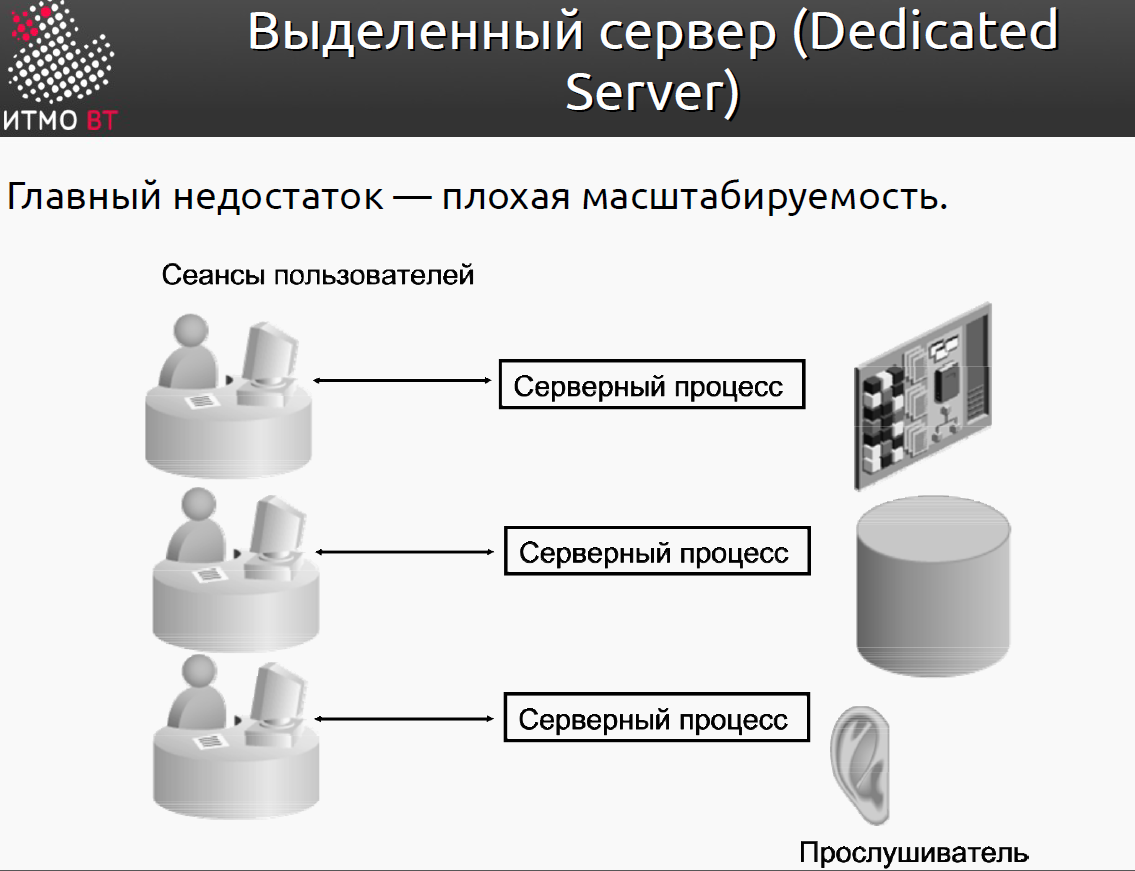
уникальное имя. (Это имя будут вводить пользователи.) Введите имя службы или системный

идентификатор (SID) базы данных, с которой устанавливается соединение, затем нажмите кнопку

«Add» (Добавить), чтобы добавить адрес для имени службы.

Для введенного адреса укажите протокол, порт и имя хоста, используемого прослушивателем.

## 14. Выделенный и разделяемый режим работы сервера. Преимущества и недостатки.





Для каждой службы в архитектуре разделяемого сервера используется как минимум один процесс диспетчера(обычно больше).

Слушатель хранит список доступных диспетчеров для каждого имени службы, а также информацию о загрузке соединения (количество одновременных соединений) для каждого диспетчера.

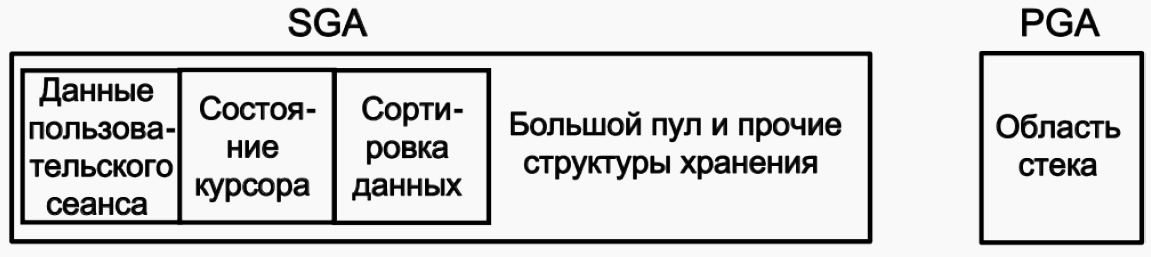
Запрос перенаправляется к наименее загруженному диспетчеру, обслуживающему службу с запрошенным именем.

Во время сеанса пользователь поддерживает соединение с одним и тем же диспетчером (но запросы могут обрабатывать разные серверные процессы).

Один диспетчер может обслуживать сотни сеансов пользователей.

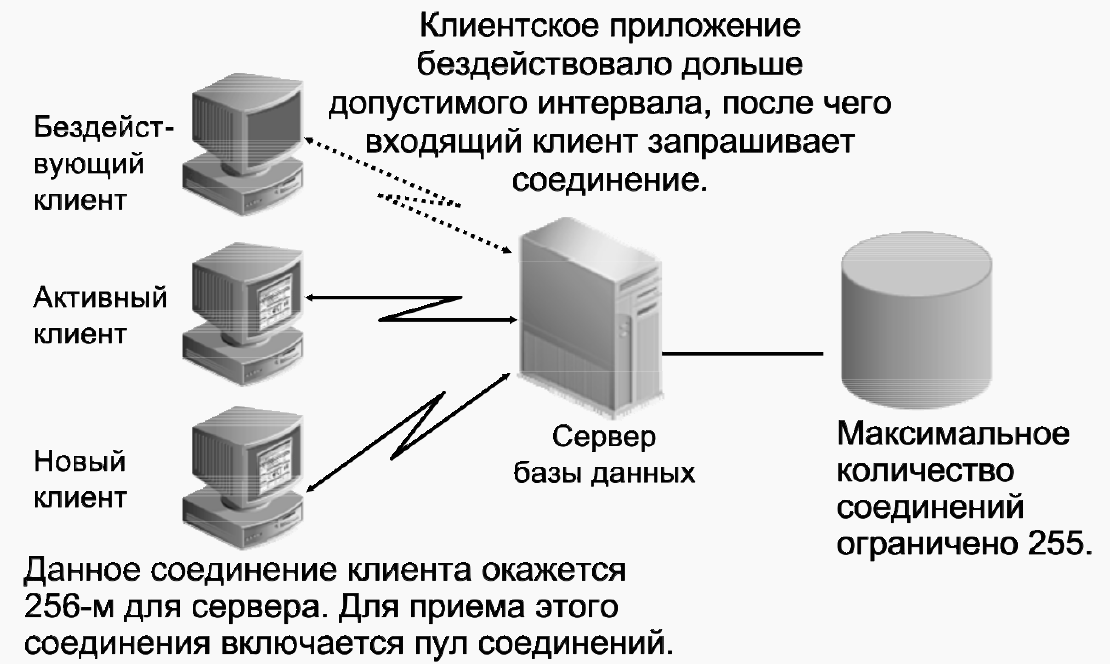
Диспетчеры направляют запросы пользователей в общую очередь, которая размещена в области SGA, выделенной для разделяемого пула.

Т.к. запросы одного пользовательского процесса могут обрабатывать *разные* серверные, большая часть данных из PGA переносится в SGA:



Это нужно учитывать при конфигурации размера SGA.

Пул соединений



Определённые операции с БД не стоит выполнять с помощью разделяемых серверов:

* Администрирование БД.
* Операции резервного копирования и восстановления.
* Пакетную обработку и операции с массовой загрузкой.
* Операции с хранилищами данных между режимами сервера

**Переключение между режимами сервера**

Текущее состояние можно проверить в DPV V$SESSION:

SQL> SELECT server FROM v$session;

Переключение из dedicated в shared:

SQL> ALTER SYSTEM SET SHARED\_SERVERS = 2;

Установка количества диспетчеров:

SQL> ALTER SYSTEM

SET DISPATCHERS =

'(PROTOCOL=TCP)(DISPATCHERS=5) (INDEX=0)',

'(PROTOCOL=TCPS)(DISPATCHERS=2) (INDEX=1)';

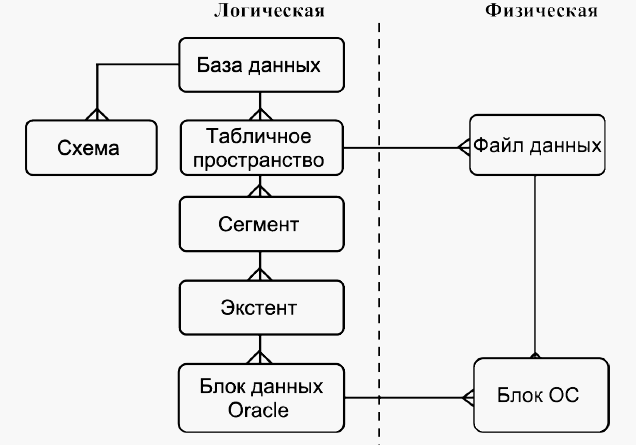
Переключение из shared в dedicated:

ALTER SYSTEM SET SHARED\_SERVERS = 0 scope = both;

или

ALTER SYSTEM SET MAX\_SHARED\_SERVERS = 0 scope = both;

## 15. Логические и физические структуры хранения. Представление табличных данных, блок базы данных.

Каждая БД логически состоит из одного или нескольких **табличных пространств**. Для каждого пространства явным образом создается один или несколько файлов данных. В них физически хранятся данные всех структур БД. Табличное пространство может находится в оперативном/доступном и автономном/недоступном режимах.

Любая БД Oracle должна содержать табличные пространства SYSTEM и SYSAUX. Они автоматически создаются вместе с БД и всегда должны находиться в оперативном режиме.

В **табличном пространстве SYSTEM** хранятся таблицы, обеспечивающие основные функции базы данных, например, таблицы словаря данных.

**Табличное пространство SYSAUX** является вспомогательным. В нем хранится множество

компонентов БД, например, репозиторий Enterprise Manager.

**Схема** – это набор объектов БД, принадлежащих пользователю БД. Объекты схемы представляют собой логические структуры, которые напрямую ссылаются на данные БД.

Объектами схемы являются:

* **Таблица** - это простейшая единица хранения данных в Oracle. Таблицы хранят все данные, к которым может обращаться пользователь. Каждая таблица имеет columns (колонки) и rows(ряды).
* **Представления** - это пользовательские представления данных одной или нескольких таблиц или других представлений. Еще представление можно рассматривать как хранимую выборку. Фактически представления не хранят данные, они извлекают данные из таблиц, на которых основываются, ссылаясь на них, как на базовые таблицы представлений.  
  С представлениями можно выполнять те же операции, что и с таблицами: выборка, обновление, вставка, удаление, причем с полным обеспечением целостности данных. Все операции, выполняемые над представлениями, воздействуют на данные базовых таблиц.  
  Представления обеспечивают дополнительный уровень безопасности, ограничивая доступ к таблице определенными записями и столбцами.
* **Индексы** - это дополнительные структуры, связанные с таблицей. Индексы могут быть созданы для ускорения выборки информации из таблицы. Во время выполнения запроса Oracle может использовать несколько или все возможные индексы, чтобы более эффективно найти нужные ряды данных. Индексы полезны при частых выборках диапазона рядов (например, выбрать всех служащих, чья зарплата выше 1000 долларов) или отдельной записи. Индексы создаются по одному или нескольким колонкам таблицы. Сразу после создания индекс автоматически используется и обслуживается Oracl'ом. Изменение данных в таблице (добавление записей, удаление, обновление) автоматически учитывается во всех соответствующих индексах абсолютно прозрачно для пользователей.
* **Кластеры** - это группы из одной или нескольких таблиц, физически хранимых вместе, т.к. они используют общие колонки и часто используются вместе. Совместное физическое хранение связанных записей улучшает время доступа к данным. Как и индексы, кластеры не влияют на разработку приложения. Включение таблицы в кластер прозрачно для пользователя и приложений. Доступ к данным в кластерных таблицах осуществляется такими же SQL средствами, как и к данным в не кластерных таблицах.
* **Последовательность** генерирует уникальные порядковые номера, которые могут использоваться как значения числовых столбцов таблиц БД.
* **Процедура** – это совокупность предложений SQL, сгруппированных вместе как выполнимая единица, исполняющая специфическую задачу. (функция возвращает единственное значение, а процедура не возвращает).

**Блоки данных, экстенты и сегменты**



* Объекты БД (таблицы и индексы) хранятся в табличных пространствах в виде **сегментов**. Каждый сегмент содержит один или более экстентов. **Экстент** состоит из непрерывного участка блоков данных, соответственно, один экстент может находиться только в одном файле данных. **Блок данных** – это наименьший элемент операции ввода-вывода в БД.
* Когда БД запрашивает у ОС набор блоков данных, ОС связывает его с текущей файловой системой или дисковыми блоками устройства хранения, поэтому нет необходимости знать физический адрес данных, хранящихся в БД. Размер блока данных можно задать при создании БД. Стандарт – 8 кб. Минимум – 2 кб.

## 16. Табличные пространства и файлы данных. Управление табличными пространствами.

БД состоит из одного или нескольких логических табличных пространств.

Каждое табличное пространство базы данных Oracle состоит из одного или нескольких файлов данных.

БД должна содержать минимум два табличных пространства — SYSTEM и SYSAUX, каждое из которых представлено минимум одним файлом данных.

Одна БД может иметь до 65534 файлов данных.

Если табличное пространство в течение всего жизненного цикла представлено одним (и только одним) файлом данных, оно называется табличным пространством типа BIGFILE.

**Временный файл** – это файл, который принадлежит временному табличному пространству:

Создается с параметром TEMPFILE:

alter tablespace temp

add tempfile 'c:\oracle\oradata\temp3\temp02.dbf'

size 50m

reuse

autoextend on

next 1m

maxsize 500m;

Временные табличные пространства используются для операций сортировки и не могут содержать постоянных объектов базы данных, таких как таблицы.

**Управление табличными пространствами**

Управляемое локально:

* управление свободными экстентами осуществляется в табличном пространстве;
* для записи свободных экстентов используется битовый образ;
* каждый бит соответствует блоку или группе блоков;
* значение бита описывает экстент: свободен или занят.

Управляемое словарём:

* свободными экстентами управляет Oracle через словарь данных;
* при выделении и освобождении экстентов обновляются соответствующие представления

**Табличные пространства, управляемые локально**

Экстенты могут выделяться одним из двух способов:

* Автоматически — размерами экстентов управляет система (он всегда кратен 64 КБ). Cпособ неприменим ко временным табличным пространствам.
* Унифицированно — табличные пространства используют унифицированный размер экстентов указываемый администратором (по умолчанию — 1 МБ). Режим нельзя использовать для табличных пространств отмены операций.

Управление пространством сегментов можно осуществлять:

* Автоматически — используются битовые образы. Битовый образ описывает состояние каждого блока данных в сегменте в соответствии с объемом пространства блока, которое доступно для вставки строк.
* Вручную — используются списки свободных сегментов (списки блоков данных, в которых имеется свободное пространство). Требуется вручную задавать и настраивать значения параметров хранения PCTUSED, FREELISTS и FREELIST GROUPS для объектов схемы.

**Состояния табличных пространств**

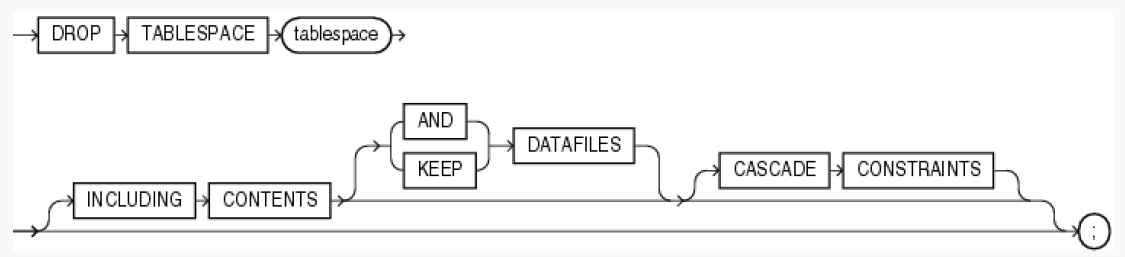
* **(Оперативное) Чтение-запись:** доступно как для чтения, так и для записи.
* **(Оперативное) Только чтение:** доступно только для чтения. После перевода в это состояние текущие транзакции будут завершены (с фиксацией или откатом), новые операции DML будут запрещены. Табличные пространства SYSTEM и SYSAUX нельзя перевести в этот режим.
* **Автономное:** эта часть БД становится временно недоступной для использования (остальная БД будет оставаться открытой и доступной). При переводе в автономный режим можно использовать следующие параметры:
  + Normal: режим можно использовать, если во всех файлах данных табличного пространства отсутствуют ошибки. СУБД установит контрольную точку для всех файлов данных перед переводом их в автономный режим.
  + Temporary: табличное пространство временно переводится в автономный режим даже при наличии ошибок в файлах данных. При «обратном» переводе в оперативный режим можетпотребоваться восстановление данных из журнала повторов.
  + Immediate: табличное пространство можно перевести в автономный режим немедленно, без установки контрольной точки для файлов данных БД Oracle. При «обратном» переводе в оперативный режим обязательнопотребуется восстановление данных из журнала повторов.

**Просмотр сведений о табличных пространствах**

* С помощью GUI Enterprise Manager.
* С помощью DPV:
  + Сведения о табличных пространствах:
    - DBA\_TABLESPACES
    - V$TABLESPACE
  + Сведения о файлах данных:
    - DBA\_DATA\_FILES
    - V$DATAFILE
  + Сведения о временных файлах:
    - DBA\_TEMP\_FILES
    - V$TEMPFILE

**Удаление табличных пространств**

Используется команда DROP TABLESPACE:



Нельзя удалить табличное пространство SYSTEM.

Табличное пространство SYSAUX может удалить только DBA и только если БД запущена в режиме MIGRATE.

Нельзя удалить табличное пространство отката если оно используется и в нём есть незавершённые транзакции.

## 17. Пользователи БД, учетные записи пользователей. Системные учетные записи.

**Основные понятия**

*Формуляр (account) пользователя БД* — это способ организации принадлежности и доступа к объектам БД.

*Пароль* необходим для аутентификации в БД Oracle.

*Полномочие (privilege)* — это право на выполнение определенного типа SQL-оператора или на доступ к объекту пользователя.

*Роль (role)* — это именованная группа связанных полномочий, которая предоставляется пользователям или другим ролям.

*Профили (profiles)* представляют собой именованные наборы ограничений на использование ресурсов БД и экземпляра.

*Квота (quote)* — это допустимый объем пространства в заданном табличном пространстве.

**Формуляр пользователя БД**

У каждого пользователя БД есть свой уникальный формуляр БД.

У каждого формуляра пользователя есть:

* Уникальное имя пользователя. Не может превышать 30 байт, не может содержать специальные символы, должно начинаться с буквы.
* Метод аутентификации. По умолчанию — пароль.
* Табличное пространство по умолчанию.
* Временное табличное пространство.
* Профиль пользователя. Набор ресурсов и ограничений с помощью паролей, присвоенных пользователю.
* Состояние формуляра. Пользователям доступны только «открытые» формуляры.

**Предопределённые формуляры SYS и SYSTEM**

Формуляр SYS:

* получает роль администратора БД;
* обладает всеми полномочиями с параметром ADMIN OPTION;
* необходим для запуска, остановки и выполнения некоторых служебных команд;
* является владельцем словаря данных;
* является владельцем репозитория автоматической рабочей нагрузки (AWR).

Формуляру SYSTEM предоставляется роль администратора БД.

Оба этих формуляра не должны использоваться для стандартных операций.

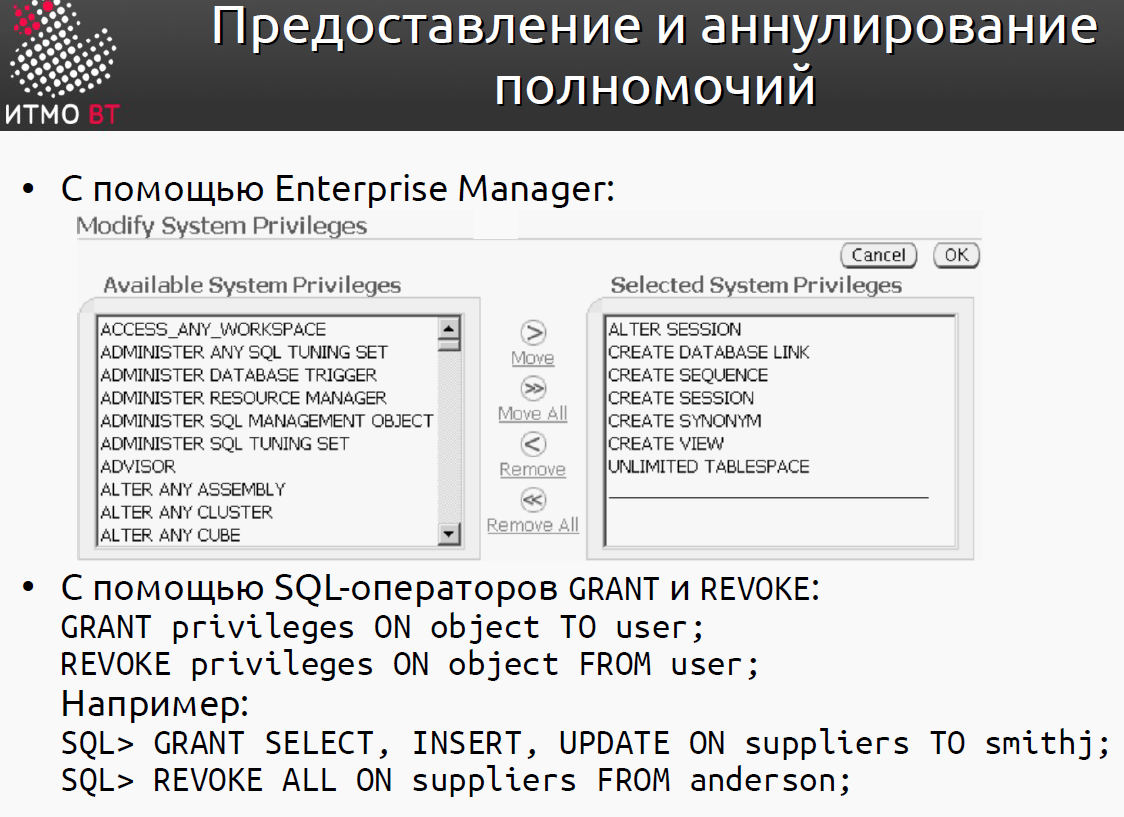
## 18. Системные и объектные полномочия. Назначение и удаление полномочий.

**Полномочия**

*Полномочие (privilege)* — это право на выполнение определенного типа SQL-оператора или на доступ к объекту пользователя.

Существует два типа полномочий пользователя:

* Системные: позволяют пользователям выполнять определенные действия с базой данных.
* Объектные: позволяют пользователям получать доступ к определенным объектам и манипулировать ими.



**Системные полномочия**

Предоставление полномочий, которые содержат фразу ANY, означает выход за пределы схемы. Например, обладая полномочиями CREATE TABLE, можно создать таблицу, однако только в собственной схеме. А полномочия SELECT ANY TABLE

позволяют выбирать среди таблиц других пользователей.

Пользователь SYS и пользователи, владеющие ролью DBA, обладают всеми полномочиями ANY.

SYSDBA и SYSOPER. Эти полномочия позволяют выполнять административные задачи в БД. SYSOPER позволяет пользователю выполнять оперативные задачи, но без возможности просмотра пользовательских данных. В него входят следующие системные полномочия:

* STARTUP и SHUTDOWN
* CREATE SPFILE
* ALTER DATABASE OPEN/MOUNT/BACKUP
* ALTER DATABASE ARCHIVELOG
* ALTER DATABASE RECOVER (только полное восстановление).
* RESTRICTED SESSION

Системное полномочие SYSDBA дополнительно дает разрешение на неполное восстановление и удаление базы данных.

* SYSASM. Позволяет запустить, остановить экземпляр ASM и управлять им.
* DROP ANY объект. Позволяет удалять объекты других пользователей схемы.
* CREATE, MANAGE, DROP и ALTER TABLESPACE. Позволяют управлять табличными пространствами.
* CREATE LIBRARY. В БД Oracle имеется возможность создавать и вызывать внешний код (например, библиотеку C) с помощью PL/SQL. Такой библиотеке должно быть присвоено имя объектом LIBRARY в базе данных.
* CREATE ANY DIRECTORY. В качестве меры безопасности каталог ОС, в котором хранится код, необходимо связать с виртуальным объектом каталога Oracle. Владелец полномочия CREATE ANY DIRECTORY потенциально может вызвать небезопасные объекты кода.
* GRANT ANY OBJECT PRIVILEGE. Позволяет предоставлять разрешения для объектов, которыми его обладатель не владеет.
* ALTER DATABASE и ALTER SYSTEM. Позволяют изменять базу данных и экземпляр Oracle (например, можно переименовать файл данных или очистить кэш буфера).

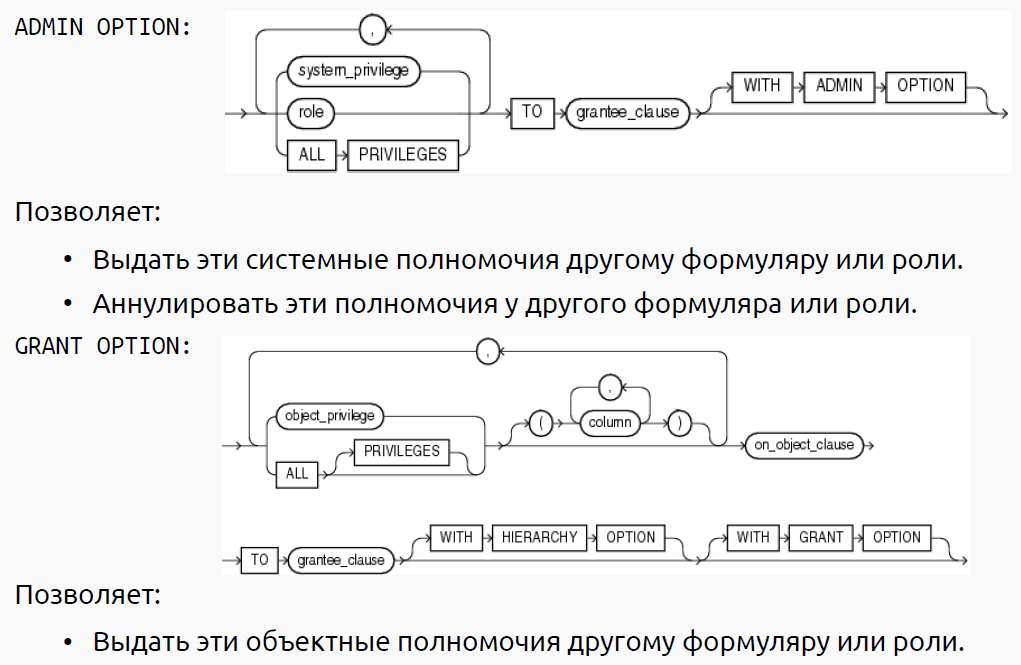
**Объектные полномочия**

GRANT privileges ON object TO user;

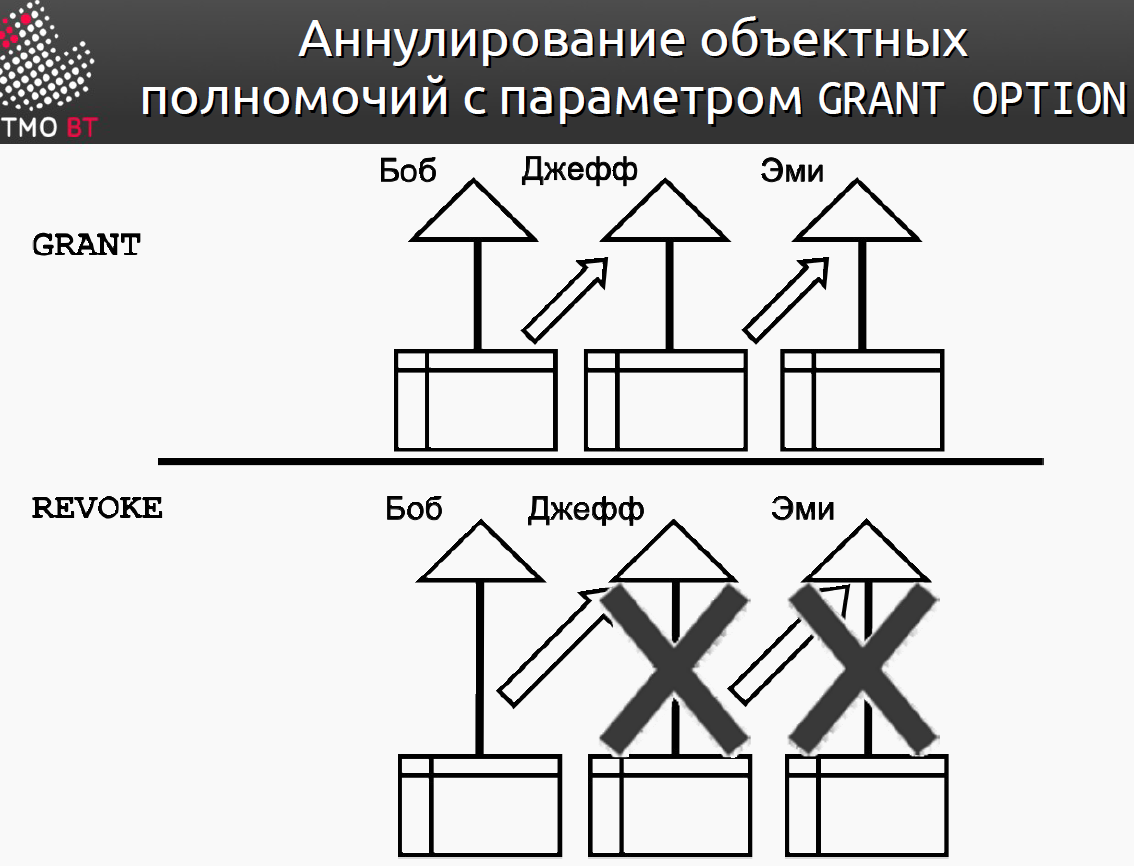
REVOKE privileges ON object FROM user;

Виды полномочий:

* SELECT — возможность выполнять выборку данных из таблицы.
* INSERT — возможность добавлять новые строки в таблицу.
* UPDATE — возможность изменять данные в таблице.
* DELETE — возможность удалять строки из таблицы.
* REFERENCES — возможность создавать ограничения целостности для выбранной таблицы.
* ALTER — возможность выполнять оператор ALTER TABLE применительно к выбранной таблице.
* INDEX — возможность создавать индексы на столбцы выбранной таблицы.
* ALL — все перечисленные выше полномочия.







## 19. Роли БД. Предопределенные роли. Профили пользователей.

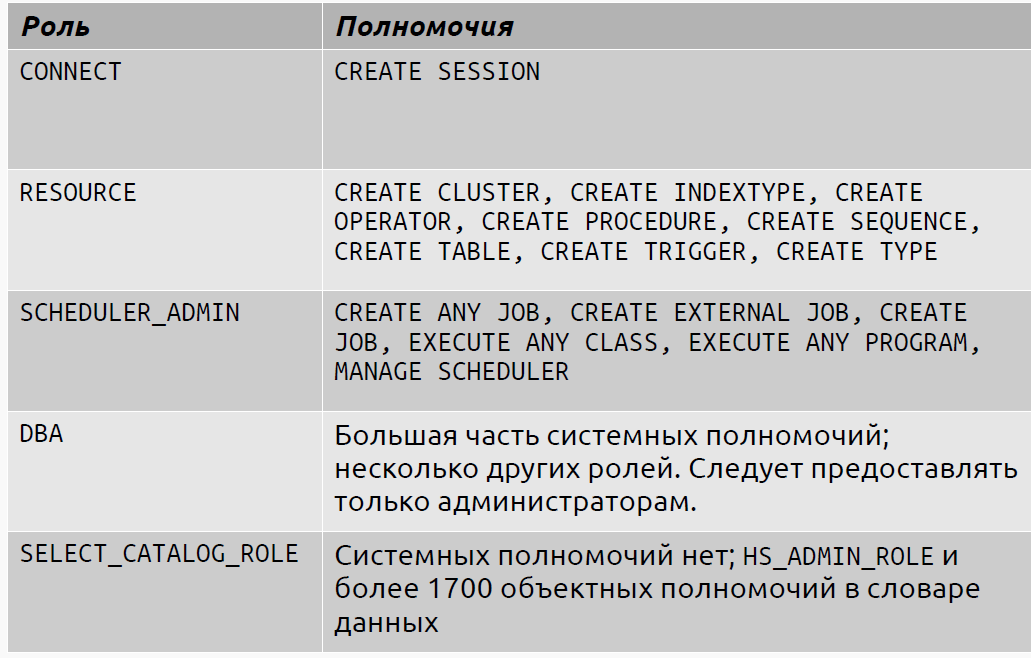
**Роль (role)**— это именованная группа связанных полномочий, которая предоставляется пользователям или другим ролям.

**Преимущества использования ролей**

* **Упрощенное управление полномочиями.** Роли используются для упрощенного управления полномочиями. Вместо того чтобы предоставлять нескольким пользователям один и тот же набор полномочий, можно предоставить эти полномочия роли, а затем предоставить эту роль пользователям.
* **Динамическое управление полномочиями.** Если полномочия, связанные с ролью, изменяются, все пользователи, которым предоставлена эта роль, немедленно автоматически получают обновленные полномочия.
* **Выбор доступных полномочий.** Роли можно включать и отключать, чтобы временно активировать или деактивировать полномочия. Таким образом можно контролировать полномочия пользователя в той или иной ситуации.



**Предопределенные роли**



**Назначение ролей пользователям**

Используются те же самые операторы GRANT и REVOKE:

SQL> CREATE ROLE cust\_serv\_mgr;

SQL> GRANT cust\_serv\_clerk TO cust\_serv\_mgr;

SQL> GRANT insert, update ON customers TO cust\_serv\_mgr;

SQL> GRANT delete ON issue\_track TO cust\_serv\_mgr;

SQL> GRANT cust\_serv\_mgr TO mary;

SQL> REVOKE insert ON customers FROM cust\_serv\_mgr;

SQL> REVOKE cust\_serv\_mgr FROM mary;

## 20. Целостность, ограничения, состояние ограничений.

Понимание целостности данных

Для недопущения введения в столбцы тех или иных значений можно использовать

следующие ограничения:

* **NOT NULL:** по умолчанию все столбцы таблицы могут принимать неопределенные значения. Слово *null* означает отсутствие значения. Ограничение NOT NULL указывает, что столбец таблицы не должен содержать неопределенных значений. Например, можно определить ограничение NOT NULL, чтобы указать, что ввод значений в столбец LAST\_NAME обязателен для всех строк таблицы EMPLOYEES.
* **Ключ UNIQUE:** ограничение целостности для ключа UNIQUE указывает, что все значения в столбце или наборе столбцов (ключе) должны быть уникальными, то есть ни одна пара строк не должна содержать одинаковых значений в этом столбце или наборе столбцов. Например, ограничение ключа UNIQUE определено для столбца DEPARTMENT\_NAME в таблице DEPARTMENTS, чтобы запретить строки с одинаковыми названиями отделов. За исключением отдельных случаев это задается с помощью уникального индекса.
* **PRIMARY KEY:** любая таблица в БД может иметь не более одного ограничения PRIMARY KEY. Значения из группы одного или нескольких столбцов, для которых определено данное ограничение, образуют уникальный идентификатор строки. Фактически, все строки именуются по значению первичного ключа

Реализация ограничения целостности PRIMARY KEY в сервере Oracle обеспечивает выполнение двух следующих условий:

* Ни одна пара строк таблицы не содержит одинаковых значений в заданном столбце или группе столбцов.
* Столбец первичных ключей не может содержать неопределенных значений. Таким образом, каждая строка такого столбца должна содержать значение. Обычно ограничения PRIMARY KEY накладываются с помощью индексов. Ограничение первичного ключа, созданное для столбца DEPARTMENT\_ID в таблице DEPARTMENTS, наложено посредством явного создания:
  + уникального индекса для данного столбца;
  + ограничения NOT NULL для данного столбца.
* **Ссылочные ограничения целостности:** разные таблицы в реляционной базе данных могут быть связаны общими столбцами, поэтому необходимо соблюдать правила отношения столбцов. Ссылочные правила целостности гарантируют сохранение этих отношений.

Ссылочные ограничения целостности накладывают следующее требование: для всех строк таблицы значение во внешнем ключе должно совпадать со значением в первичном ключе.

Например, внешний ключ определен для столбца DEPARTMENT\_ID таблицы EMPLOYEES. Это гарантирует, что все значения в столбце совпадают со значениями в первичном ключе таблицы DEPARTMENTS. Следовательно, в столбце DEPARTMENT\_ID таблицы EMPLOYEES отсутствуют неправильные номера отделов.

Еще один тип ссылочного ограничения целостности называется рефлексивным ограничением ссылочной целостности. При этом внешний ключ ссылается на родительский ключ в той же таблице.

* **Ограничения CHECK:** ограничение целостности CHECK для столбца или набора столбцов указывает, что для каждой строки таблицы должно выполняться или иметь неизвестное значение какое-либо заданное условие.

Если при выполнении DML-оператора проверки условия ограничения получен результат FALSE, выполняется откат оператора, если наложено ограничение IMMEDIATE. Если ограничение является отложенным (DEFERRED), откат производится при фиксации, а не выполнении DML.

**Нарушение ограничений**

Нарушение ограничения происходит, если переданный DML-оператор не совместим с ограничением. Нарушения ограничения могут обуславливаться разными условиями. Вот некоторые из них:

* **Неуникальность:** попытка хранения одинаковых значений в столбце с ограничением UNIQUE (например, если столбец является первичным ключом или используется уникальный индекс);
* **Целостность ссылочных данных:** нарушение правила, согласно которому у каждой дочерней строки должна быть родительская строка;
* **Проверка:** попытка сохранить в столбце значение, противоречащее правилам, определенным для столбца. Например, для столбца AGE задано ограничение CHECK, согласно которому все значения столбца должны быть положительными.

**Состояния ограничений**

Чтобы проще разрешать ситуации, когда данные временно могут нарушать ограничение, можно использовать различные состояния ограничений. Ограничение целостности можно включить (ENABLE) или выключить (DISABLE).

Если ограничение включено, проверка данных происходит при вводе данных или обновлении БД. Данные, противоречащие правилам ограничения, не вводятся. Если ограничение отключено, в базу данных могут вводиться противоречащие ограничению данные.

Ограничение целостности может находиться в одном из следующих состояний:

* DISABLE NOVALIDATE
* DISABLE VALIDATE
* ENABLE NOVALIDATE
* ENABLE VALIDATE

**DISABLE NOVALIDATE:** новые или уже существующие данные могут противоречить ограничению, если они не проверены. Этим часто пользуются при обработке данных из уже проверенного источника в режиме таблицы только для чтения, вследствие чего новые данные в таблицу не попадают. Состояние NOVALIDATE используется для хранилищ данных, в которых данные уже прошли проверку. Поэтому повторная проверка не нужна, что экономит время обработки.

**DISABLE VALIDATE:** при данном состоянии ограничения изменение столбцов, на которых оно наложено, запрещено, так как это нарушит целостность данных – уже существующие данные будут проверены, а новые будут вводиться без проверки. Это состояние часто используют, когда существующие данные нужно проверить, но не изменить, а индекс для производительности не нужен.

**ENABLE NOVALIDATE:** новые данные соответствуют ограничению, однако уже существующие данные находятся в неизвестном состоянии. Это состояние часто используется, когда точно известно, что таблица содержит обработанные и проверенные данные, не требующие проверки. При этом все новые вводимые данные не должны противоречить ограничению.

**ENABLE VALIDATE:** ограничению не должны противоречить ни новые, ни существующие данные. Это самое распространенное состояние ограничения, используемое по умолчанию.

**Проверка ограничений**

Ограничения проверяются при следующих событиях:

* исполнение оператора (для неотложенных ограничений);
* COMMIT(для отложенных ограничений).

**Неотложенные ограничения**, также называемые *немедленными ограничениями*, применяются после выполнения каждого DML-оператора. При нарушении ограничения выполняется откат оператора. Если ограничение приводит к такой операции, как каскадное удаление (delete cascade), эта операция выполняется как часть вызвавшего ее оператора. Если ограничение определено как неотложенное, его нельзя сделать отложенным При использовании неотложенных ограничений для ограничений первичного и уникального ключей требуются уникальные индексы. Если же для одного или более столбцов уже используется уникальный индекс, создать ограничение будет невозможно, так как эти индексы нельзя использовать для уникального или первичного ключа.

**Отложенные ограничения** проверяются только после фиксации транзакции. Если при этом нарушается ограничение, выполняется откат всей транзакции. Такие ограничения удобнее всего использовать, когда при отношении внешнего ключа родительские и дочерние строки вводятся одновременно, как в примере с системой регистрации заказов, где заказ и его позиции вводятся одновременно. При использовании отложенных ограничений для ограничений первичного и уникального ключей требуются неуникальные индексы. Если же для одного или более столбцов уже используется уникальный индекс, создать ограничение будет невозможно, так как эти индексы не являются отложенными.

## 21. Понятие блокировки, уровни блокировки и операторы их использующие.

**Блокировки**

* Не позволяют нескольким сеансам одновременно изменять одни и те же данные.
* Автоматически срабатывают на самом низком уровне, доступном для данного оператора.
* Не расширяются до более высоких уровней.

Перед тем как база данных позволит сеансу изменить данные, он должен заблокировать изменяемые данные. Блокировка дает сеансу монопольные права на работу с данными, благодаря чему другая транзакция не сможет изменить заблокированные данные до тех пор, пока блокировка не будет снята.

Транзакции могут блокировать отдельные строки, наборы строк или даже целые таблицы. База данных Oracle поддерживает как ручную, так и автоматическую блокировку. Автоматически созданные блокировки всегда действуют на самом нижнем уровне, чтобы минимизировать вероятность возникновения потенциальных конфликтов с другими транзакциями.

Перед изменением данных сеанс должен заблокировать изменяемые данные. Сеанс получит монопольное право на на работу с данными, а остальные транзакции будут ждать, пока блокировка не будет снята. Могут блокировать строки, наборы строк и таблицы. Автоматическая блокировка  работает только на самом нижнем уровне для минимизации потенциальных конфликтов.

Для ручной блокировки:

lock table kkok in exclusive mode nowait;

nowait - прервать текущую блокировку, вернуть контроль немедленно

Режимы блокировки:

* row share -  запрещает только блокировать всю таблицу
* share - запрет обновления таблицы (создание индекса)
* share row exclusive -  можно смотреть строки таблицы, нельзя блокировать и обновлять
* exclusive -  можно опрашивать только таблицу (для удаления таблицы)

DML транзакции используют EXCLUSIVE  на уровне строки  и ROW EXCLUSIVE  для таблицы

## 22. Конфликты блокировок, устранение конфликтов блокировок. Взаимные блокировки.

Наиболее распространенной причиной возникновения конфликта блокировок являются неподтвержденные изменения, однако существуют и другие причины:

* **Транзакции, которые выполняются длительное время:** множество приложений используют пакетную обработку для выполнения массовых обновлений. Такие операции обычно выполняются по расписанию в интервалы времени с низкой активностью пользователей или при ее отсутствии, но в некоторых случаях операция может не завершиться вовремя, или для ее завершения может потребоваться значительно больше времени, чем было рассчитано. Конфликты блокировок довольно часто возникают при одновременном выполнении операций пакетной обработки.
* **Неадекватно высокие уровни блокирования:** не все базы данных поддерживают низкоуровневое блокирование (Oracle получил поддержку этой функции в 6 версии, которая вышла в 1988 году). Некоторые базы данных все еще осуществляют блокировки на уровне страницы или таблицы. Разработчики приложения обычно используют при создании приложений множество различных баз данных с искусственно завышенными уровнями блокирования, из-за чего база данных Oracle вынуждена делать то же самое для обеспечения совместимости с менее гибкими системами баз данных. Разработчики, которые не имеют достаточного опыта работы с базами данных Oracle, также иногда используют без необходимости более высокие уровни блокирования, чем необходимо.

Возникают, когда незавершенная транзакция хочет получить блокировку над уже заблокированным объектом. Обычно решаются временем и механизмом обслуживания очереди. Возможные причины блокировок: неподтвержденные изменения, транзакции, выполняющиеся длительное время, неадекватно высокие уровни блокирования.

Для устранения сеанс, создавший блокировку должен подтвердить изменения. В крайнем случае его нужно завершить, использую команду kill session в sqlplus. Нужно знать его sid и serial# из представления v$session.

Взаимные блокировки возникают, когда несколько сеансов ожидают данные взаимно блокированные друг другом. Oracle выявляет взаимные блокировки и прерывает выполнение оператора сообщением об ошибке, реакцией должна быть отмена или подтверждение изменения данных, что освободит блокировку одного из сеансов.

Конфликты блокировок обычно устраняются временем и механизмом обслуживания очереди. В некоторых редких случаях для устранения конфликта блокировок может потребоваться вмешательство администратора. В примере, представленном на втором слайде, вторая транзакция создает блокировку для одной строки в 9:00:00 и при этом не подтверждает изменения, из-за чего блокировка не снимается. Первая транзакция пытается обновить всю таблицу в 9:00:05, для чего необходимо создать блокировки для всех строк. Первая транзакция заблокирована второй транзакцией, пока та не подтвердила изменения в 16:30:01. Пользователь, пытаясь осуществить первую транзакцию, почти наверняка свяжется с администратором, который должен выяснить причину конфликта и устранить ее.

**Устранение конфликтов блокировок**

Чтобы устранить конфликт блокировок, сеанс должен освободить созданную блокировку. Наилучший способ достичь этого – связаться с пользователем и попросить его завершить транзакцию.

В крайнем случае администратор может прервать сеанс, создавший блокировку, нажав кнопку «Kill Session» (Удалить сеанс). Помните, что при таком завершении сеанса все изменения, внесенные данной транзакцией, будут потеряны (будет выполнен откат). Пользователю завершенного сеанса нужно будет повторно подключиться к базе данных и повторить все операции, которые были выполнены после последней операции подтверждения. Пользователи, сеанс которых был прерван, получат следующее сообщение об ошибке при попытке выполнить новый SQL-оператор:

ORA-03135: connection lost contact

**Примечание.** Снайпер сеансов PMON может автоматически прерывать сеансы по истечении заданного времени простоя. Также это можно делать с помощью профилей или диспетчера ресурсов.

**Устранение конфликтов блокировок с помощью SQL**

Управление сеансами, как и большинство других заданий, можно осуществлять не только в Enterprise Manager, но и с помощью операторов SQL. Таблица v$session содержит подробные сведения обо всех подключенных сеансах. blocking\_session – идентификатор блокирующего сеанса. Если запросить SID и SERIAL# (где SID совпадает с идентификатором заблокированного сеанса), вы получите всю необходимую информацию для выполнения операции kill session (удаления сеанса).

**Примечание.** Для автоматического отключения сеансов, которые бездействуют и блокируют другие сеансы, можно использовать диспетчер ресурсов базы данных (Database Resource Manager).

**Взаимные блокировки**

Взаимная блокировка – это особый случай конфликта блокировок. Взаимные блокировки возникают при участии двух (или больше) сеансов, когда сеансы ожидают данные, взаимно заблокированные друг другом. Так как каждый из сеансов ожидает данные, заблокированные противоположной стороной, ни один из сеансов не может завершить транзакцию и устранить конфликт.

База данных Oracle автоматически выявляет взаимные блокировки и прерывает выполнение оператора с сообщением об ошибке. Реакцией на ошибку должна быть операция подтверждения или отмены изменений, которая освободит блокировку одного из сеансов, благодаря чему второй сеанс сможет завершить свою транзакцию.

В примере, приведенном в данном слайде, в ответ на ошибку о возникновении взаимной блокировки транзакция 1 должна выполнить операцию подтверждения или отмены изменений. В случае подтверждения потребуется еще раз выполнить второе обновление, чтобы успешно завершить транзакцию. В случае отката потребуется выполнить оба обновления, чтобы успешно завершить транзакцию.

23. Данные отмены операций (UNDO), изменение данных отмены операций и журнала повторов операций во время транзакции.

**Данные отмены операций:**

* являются копией оригинальных (неизмененных) данных;
* сохраняются для каждой транзакции, которая изменяет данные;
* сохраняются как минимум до завершения транзакции;
* используются для:
  + операций отката;
  + обеспечения целостности данных при чтении;
  + запросов, транзакций и таблиц моментального отката;
  + восстановления данных после сбойных транзакций.

База данных Oracle сохраняет исходные значения данных (данные отмены операций), когда процесс изменяет данные в базе данных. Данные сохраняются до внесения в них изменений. Сохранение данных отмены операций позволяет выполнить откат неподтвержденных данных. Данные отмены операций поддерживают целостность данных при чтении и запросы с моментальным откатом. Данные отмены операций могут использоваться для отмены (моментального отката) транзакций и таблиц.

Запросы с сохранением целостности данных при чтении возвращают результаты с сохранением целостности данных на момент обработки запроса. Для успешного выполнения запросов с сохранением целостности данных при чтении необходимо, чтобы исходные значения данных были сохранены в виде данных отмены операций. Если исходные данные недоступны, будет получена ошибка «Snapshot too old» (Снимок слишком старый). База данных Oracle может реконструировать данные для обеспечения целостности при чтении до тех пор, пока сохраняются данные отмены операций.

Опросы с моментальным откатом обращаются к версии данных, которая была актуальна некоторое время назад. До тех пор, пока данные отмены операций существуют для запрошенного отрезка времени, можно будет успешно выполнять запросы с моментальным откатом. Транзакции с моментальным откатом используют данные отмены операций для создания компенсирующих транзакций, с помощью которых можно будет откатить транзакцию и зависимые от нее транзакции. Таблица с моментальным откатом позволит восстановить состояние таблицы на заданный интервал времени.

Данные отмены операций также используются для восстановления данных после сбойных транзакций. Сбойные транзакции возникают в момент непланового завершения сеанса пользователя (например, из-за сетевых ошибок или сбоя на клиентском компьютере), из-за чего пользователю не удается подтвердить или откатить транзакцию. Сбойные транзакции также могут возникать при сбое экземпляра или в результате выполнения команды SHUTDOWN ABORT

В случае возникновения сбойной транзакции база данных Oracle отменяет все изменения, которые внес пользователь, и восстанавливает исходные значения данных. Данные отмены операций сохраняются для всех транзакций как минимум до завершения транзакции по одной из причин:

* пользователь отменил транзакцию (откат транзакции)
* пользователь завершил транзакцию (фиксация транзакции)
* пользователь выполнил DDL-оператор, такой как CREATE, DROP, RENAME или ALTER. Если текущая транзакция содержит DML-операторы, база данныхсначала подтвердит транзакцию, а затем выполнит и подтвердит операторы DDL в виде новой транзакции
* неплановое завершение сеанса пользователя (откат транзакции)
* запланированное завершение сеанса пользователя (фиксация транзакции).

Объем хранимых данных отмены операций и время хранения зависят от уровня активности базы данных и ее конфигурации.



Oracle сохраняет значения данных, когда процесс изменяет их. Можно выполнить откат неподтвержденных данных - отмены транзакций и таблиц. Данные отмены операций поддерживают целостность данных при чтении и запросы с моментальным откатом(версия данных некоторе время назад пока данные отмены еще хранятся). Если данные потеряны - Snapshot too old. Можно восстановиться после сбойных транзакций( ошибка сети) - пользователь не подтведил/отменил транзакци.

Данные отмены хранятся до завершения транзакции по одной из причин: пользователь отменил/подтвердил/DDL оператор (CREATE/DROP/RENAME), неплановое завершение сеанса пользователя.

Перед началом транзакции ей выделяется сегмент, и экстенты последовательно полностью заполняются. На протяжении жизненного цикла транзакции изменяемые значения копируются. Экстенты заполняются до окончания транзакции или до заполнения всего пространства, потом используется следующий экстент

Данные отмены операций нужны для возможности отмены изменений. защита от несогласованных операций чтения в многопользовательских процессах.

Данные повторов нужны, чтобы повторно применить изменения, защита от потери данных

## 24. Автоматическое и ручное управление основной и разделяемой памятью.

**Управление компонентами памяти**

* Автоматическое управление памятью (AMM):
  + позволяет указать общий объем памяти, выделенный экземпляру (включая как SGA, так и PGA).
* Автоматическое управление разделяемой памятью:
  + позволяет указать общий объем SGA-памяти с помощью одного параметра инициализации;
  + позволяет серверу Oracle управлять объемом памяти, выделенным для разделяемого пула, пула Java, буферного кэша, пула потоков и большого пула.
* Настройка управления разделяемой памятью вручную:
  + задание размера компонентов с помощью нескольких отдельных параметров инициализации;
  + использование помощника по распределению памяти для создания рекомендаций.

В Oracle Database 11*g* можно указать общий объем памяти, выделенный экземпляру. При этом память будет динамически перераспределяться между глобальной системной областью (SGA) и глобальной областью программ (PGA) по мере необходимости. Данный способ называется автоматическим управлением памятью (AMM) и доступен только на платформах, поддерживающих динамическое освобождение памяти. Он упрощает выполнение задач, связанных с управлением памятью.

Также доступны помощники по распределению памяти, которые помогают задавать параметры инициализации на различных уровнях. Доступность того или иного помощника зависит от уровня, на котором задаются параметры памяти. Если включена функция АММ, доступен только помощник по размеру памяти.

Функция автоматического управления разделяемой памятью (ASMM) позволяет управлять областью SGA в целом.

Эта область состоит из нескольких компонентов. Размер большинства этих компонентов настраивается динамически для обеспечения наилучшей производительности в пределах заданных параметров инициализации. При включении AMM функция ASMM включается автоматически. Если функция ASMM включена, а AMM нет, доступен только помощник по размеру SGA.

Размером отдельных компонентов можно управлять вручную, задавая параметр инициализации отдельно для каждого компонента. Если сервер Oracle уведомляет об ошибке производительности, связанной с размером компонента SGA или PGA, можно воспользоваться помощником по распределению памяти, чтобы определить новые параметры для этих компонентов. Этот помощник способен моделировать результат применения новых параметров.

**Помощник по автоматическому выделению разделяемой памяти**

Если функция ASMM включена, невозможно задать параметры инициализации для определенных компонентов разделяемой памяти, управляемых этой функцией. При включении ASMM становится доступен помощник по размеру SGA, с помощью которого можно подобрать наиболее подходящее значение общего размера SGA.

Если ASMM включена, не следует изначально задавать параметры инициализации для определенных компонентов, памятью которых эта функция управляет. Если результаты выделения памяти с помощью ASMM для определенных компонентов требуют регулировки, значения для этих компонентов можно указать вручную. Такие значения рассматриваются как минимальные объемы памяти для соответствующих компонентов. При этом ограничивается объем памяти, доступной для автоматической настройки, но если для используемой среды требуется указать особое значение размера, которое не может быть настроено с помощью функции ASMM, то такая возможность будет предоставлена.

Необходимые параметры инициализации:

* SHARED\_POOL\_SIZE
* LARGE\_POOL\_SIZE
* JAVA\_POOL\_SIZE
* DB\_CACHE\_SIZE
* STREAMS\_POOL\_SIZE

Чтобы настроить эти параметры при включенной функции ASMM, используйте команду ALTER SYSTEM.

**Настройка компонентов разделяемой памяти вручную**

Если автоматическое управление разделяемой памятью не используется, необходимо указать значения каждого из компонентов SGA при установке и создании базы данных.

Чтобы настроить параметры памяти, выполните следующие действия:

1. Перейдите на страницу помощников, щелкнув ссылку «Memory Advisors» (Помощники по распределению памяти) в области «Database Configuration» (Конфигурация базы данных).

2. Вызовите любой из помощников, нажав кнопку «Advice» (Совет) рядом с полем для ввода значения. Чтобы получить дополнительные сведения о работе помощника, нажмите кнопку «Help» (Справка) для вызова интерактивной справки.

3. В полях для указания значений компонентов введите новые значения, полученные в результате работы помощника или же из собственных наблюдений.

**Использование помощников по распределению памяти**

Используя помощники по распределению памяти на уровне компонентов, можно настраивать размеры структур памяти. Эти помощники можно использовать только в том случае, если отключены автоматическое управление памятью и автоматическое управление разделяемой памятью.

Помощник по распределению памяти включает в себя три помощника из состава Enterprise Manager, которые предоставляют рекомендации относительно следующих структур памяти:

* разделяемого пула в глобальной системной области (SGA);
* буферного кэша в SGA.

Для вызова помощников по распределению памяти выполните следующие действия:

1. На домашней странице базы данных в разделе «Related Links» (Связанные ссылки) щелкните ссылку «Advisor Central» (Центр рекомендаций).

2. На странице «Advisor Central» (Центр рекомендаций) щелкните «Memory Advisor» (Помощник по распределению памяти). На странице «Memory Advisors» (Помощники по распределению памяти) отображается разбивка использования памяти для области SGA.

**Примечание.** Для запуска помощников по отдельным компонентам необходимо отключить автоматическое управление разделяемой памятью.

3. Рядом со значением параметра «Shared Pool» (Разделяемый пул) или «Buffer Cache» (Буферный кэш) нажмите кнопку «Advice» (Совет), чтобы вызвать соответствующий помощник.

Доступ ко всем компонентам помощника по распределению памяти можно получить из SQL\*Plus, просмотрев соответствующие представления V$\*. Существует четыре представления для отдельных компонентов SGA, которые настраиваются автоматически.

В базе данных Oracle представлено несколько помощников по компонентам, с помощью которых можно задавать размер наиболее важных компонентов SGA. Это следующие помощники:

* **V$DB\_CACHE\_ADVICE:** содержит строки, прогнозирующие число физических операций чтения и время для размера кэша, соответствующего каждой строке;
* **V$SHARED\_POOL\_ADVICE:** отображает сведения о предполагаемом времени разбора в разделяемом пуле для различных размеров пула;
* **V$JAVA\_POOL\_ADVICE:** отображает сведения о предполагаемом времени загрузки класса в пул Java для различных размеров пула;
* **V$STREAMS\_POOL\_ADVICE:** отображает сведения о предполагаемом количестве разрозненных или неразрозненных сообщений и времени, потраченном на соответствующие операции с сообщениями для различных размеров пула потоков.

## 25. Ошибки пользователя БД. Область мгновенного восстановления.

Ошибка пользователя - сама операция выполнена успешно, но команда/транзакция неверна(удаление таблицы, ввод неверных данных).

Причины: случайное удаление или изменение данных. Если не изменения не зафиксированы, можно просто сделать откат rollback.

Для анализа и исправления используется технология Oracle Flashback - набор функций для просмотра прошлых состояний БД без восстановления из резервной копии. Для анализа используется запрос в режиме Flashback: select … **as of** time/SCN - получить данные в тот момент. Запрос версий - просмотр данных на интервале времени select … vesions between.  Запрос транзакций - просмотр всех изменений, сделанных на уровне транзакций.

Решения для восстановления: возврат транзакций: откат конкретной и связанных. таблица Flashback: восстановление одной таблицы на момент времени,не затрагивая остальные. Восстановление Flashback:  возврат удаленной таблицы из корзины в БД с зависимыми обектами (триггеры, индексы).

Область мгновенного восстановления - пространство для размещения архивных журналов, резервных копий, журналов Flashback, зеркальных управляющий и журналов повторов. Упрощает управление хранением бэкапов. **Размер**: в 2 раза больше, **Хранить**: на отдельном от файлов БД диске. **Методика сохранения** резервных копий определяет момент устаревания файлов, БД автоматически удаляет ненужные. **настройка области предполагает их.** расположение USE\_DB\_RECOVERY\_FILE\_DEST.

## 26. Ошибки экземпляра БД. Процедура и этапы восстановления экземпляра.

Сбой экземпляра происходит, если его закрыть до синхронизации файлов БД. Причины:  сбой питания, аппаратного обеспечения,критического фонового процесса, аварийного завершения работы(shutdown abort, startup force). При перезапуске автоматически происходит восстановление после сбоя(накат изменений в журналах повторов и откат всех незавершенных транзакций).

Процедура:

запуск экземпляра,

монтирование управляющих файлов,

попытка открыть файлы: обнаружение факта, что не синхронизированы - SCN в файлах!=управляющем,

накат: повторение из оперативного журнала повторов последовательно повторяются транзакции -> синхронизированы файлы данных и управляющий,

открытие базы,

откат незафиксированных транзакций (поскольку табличное пространство отмены тоже накатывается) .

файлы данных содержат только зафиксированные данные.

## 27. Резервное копирование. Полная и инкрементальная резервные копии.

**Резервная копия всей базы данных:** включает в себя все файлы данных и по крайней мере один управляющий файл. (все управляющие файлы в базе данных идентичны.)

**Частичная резервная копия базы данных:** может включать ноль или более табличных пространств и ноль или более файлов данных; управляющий файл может отсутствовать.

**Полное резервное копирование:** создает копию каждого блока данных, содержащего данные и принадлежащего к резервируемым файлам.

**Инкрементное резервное копирование:** создает копию всех блоков данных, изменившихся с момента предыдущего резервного копирования. База данных Oracle поддерживает два уровня инкрементного резервного копирования (0 и 1). Инкрементные резервные копии уровня 1 могут быть двух типов: *кумулятивные* или *дифференциальные*. Кумулятивная резервная копия включает все изменения, произошедшие с момента последнего резервного копирования уровня 0. Дифференциальная резервная копия включает все изменения, произошедшие с момента последнего инкрементного резервного копирования (уровня 0 или уровня 1).

**Автономные резервные копии** (также называются «холодными» или *согласованными*)**:** создаются в те периоды, когда база данных закрыта. Они являются согласованными, поскольку во время резервирования системный номер изменения (SCN) в заголовках файлах данных совпадает с номером SCN в управляющих файлах.

**Оперативные резервные копии** (также называются «горячими» или *несогласованными*)**:** создаются в те периоды, когда база данных открыта. Они являются несогласованными, поскольку при открытой базе данных синхронизация файлов данных с управляющими файлами не гарантируется. Чтобы несогласованные резервные копии можно было использовать, их потребуется восстановить.

## 28. Менеджер резервного копирования RMAN

RMAN представляет собой компонент базы данных Oracle, который используется для выполнения операций резервного копирования и восстановления. С его помощью можно создавать согласованные и несогласованные резервные копии, выполнять инкрементное или полное резервное копирование, а также резервировать всю базу данных или ее часть. Диспетчер восстановления RMAN использует собственный мощный язык для управления и написания сценариев, а также опубликованный API, который связывает RMAN со многими популярными программами резервного копирования.

RMAN может сохранять резервные копии на диск для обеспечения быстрого восстановления или помещать их на ленту для длительного хранения. Чтобы диспетчер RMAN мог сохранять резервные копии на ленту, необходимо либо использовать утилиту Oracle Secure Backup, либо настроить связь с ленточным накопителем, который называется библиотекой управления носителями (MML).

1. Во время сеанса работы с терминалом запустите RMAN и подключитесь к целевой базе данных.

2. Выполните команды конфигурирования:

- CONFIGURE DEFAULT DEVICE TYPE TO disk;

- CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COPY;

- CONFIGURE CONTROLFILE AUTOBACKUP ON;

3. Резервная копия базы данных представляет собой копию всех файлов данных и управляющего файла. В нее также можно включить файл параметров сервера (SPFILE) и архивные файлы журналов повторов. Для создания копии образа всех файлов базы данных с помощью RMAN нужно всего лишь смонтировать или открыть базу данных, запустить диспетчер RMAN и ввести команду BACKUP, представленную на слайде.

В качестве альтернативы можно указать параметр DELETE INPUT при резервном копировании архивных файлов журналов.

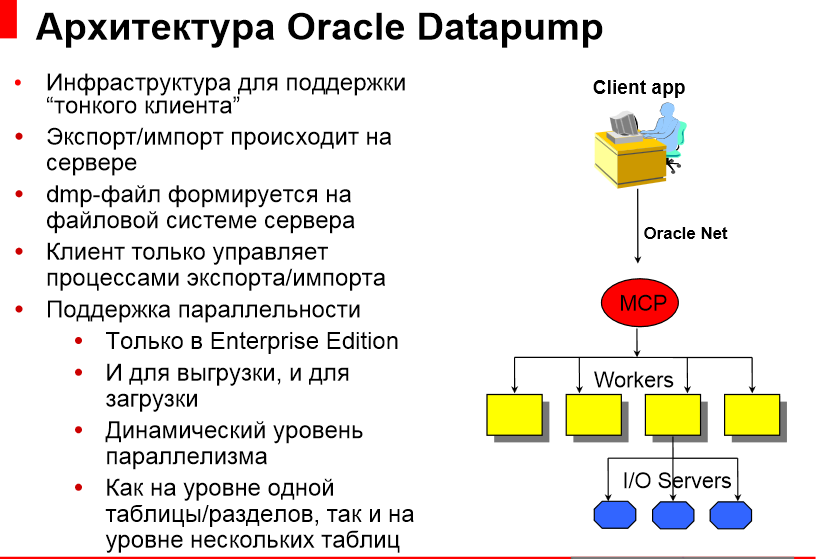
В результате диспетчер RMAN будет удалять архивные файлы журналов после создания их резервных копий. Это особенно полезно, если область мгновенного восстановления не используется, поскольку таким образом будет контролироваться пространство путем удаления файлов по мере заполнения пространства. В этом случае команда на слайде будет иметь следующий вид:

RMAN> BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;

Можно также создать резервную копию (набор резервирования или копии образов) предыдущих копий образов всех файлов данных и управляющих файлов базы данных с помощью следующий команды:

RMAN> BACKUP COPY OF DATABASE;

## 29. Data pump — архитектура. Утилиты export и import.

**Oracle Data Pump** - решение Oracle Database, предназначенное для быстрого переноса данных и метаданных из одной базы данных в другую. Разработано с целью замены традиционных утилит экспорта и импорта данных (exp и imp). Средства exp/imp сохранены в целях совместимости, но заявлены как устаревшие. Oracle Data Pump содержит новую серверную инфраструктуру, утилиты expdp и impdp, в наличии документированный API для разработчиков.

Особенности архитектуры Oracle Data Pump определяет то, что экспорт/импорт происходит на сервере, dmp-файл формируется на сервере, клиент только управляет процессами экспорта или импорта. Поддерживается параллелизм операций выгрузки и загрузки (только в редакции Oracle Database Enterprise Edition ), присутствует возможность предварительной оценки размера dump-файла. Одной из сильных сторон технологии является возможность импорта по сети из одной базы данных в другую, на "лету" без промежуточного файла и экспорт по сети из БД, если та находится в режиме "только чтение".  
Data Pump самостоятельно решает, какие методы доступа к данным использовать. Это может быть прямой маршрут или внешние таблицы. Технология использует загрузку или выгрузку по прямому маршруту, когда структура таблицы позволяет сделать это, и в случае если желательна максимальная производительность одиночного потока. Однако при наличие в базе данных кластеризируемых таблиц, ограничений ссылочной целостности, зашифрованных столбцов, или ряда других элементов, Data Pump применит использование внешних таблиц, а не прямой маршрут, чтобы переместить данные.  
Все действия Data Pump выполняются множественными задачами (jobs). Эти задачи координируются главным управляющим процессом, который использует расширенную очередь. Во время выполнения, создаётся и используется главным управляющим процессом таблица очереди, которая называется по имени задания. Таблица удаляется после успешного выполнения задания Data Pump. Задание и очередь могут быть названы абсолютно любым именем с использованием параметра "JOB\_NAME". Остановка клиентского процесса не останавливает связанное с ним задание. Передача клиенту комбинации клавиш "Ctrl+C" во время выполнения задания остановит вывод на стандартное устройство вывода и переведёт в командную строку. Ввод "status" в этой командной строке позволить проследить за состоянием текущего задания.  
**Импорт/Экспорт таблиц**

Параметр TABLES, используется для указания таблиц, которые будут экспортированы. Следующий пример демонстрирует синтаксис экспорта и импорта:

expdp scott/tiger@TESTDB tables=EMP,DEPT directory=TEST\_DIR dumpfile=EMP\_DEPT.dmp logfile=expdpEMP\_DEPT.log

impdp scott/tiger@TESTDB tables=EMP,DEPT directory=TEST\_DIR dumpfile=EMP\_DEPT.dmp logfile=impdpEMP\_DEPT.log

Параметр TABLE\_EXISTS\_ACTION=APPEND позволяет импортировать данные в существующие таблицы.

**Импорт/Экспорт схемы**

Параметр OWNER утилиты EXP был заменен параметром SCHEMAS, используемом для указания схемы для экспорта. Следующий пример демонстрирует синтаксис экспорта и импорта схемы:

expdp scott/tiger@TESTDB schemas=SCOTT directory=TEST\_DIR dumpfile=SCOTT.dmp logfile=expdpSCOTT.log

impdp scott/tiger@TESTDB schemas=SCOTT directory=TEST\_DIR dumpfile=SCOTT.dmp logfile=impdpSCOTT.log

**Импорт/экспорт базы данных**

Параметр FULL указывает что требуется полный экспорт базы данных. Следующий пример демонстрирует полный экспорт и импорт базы данных:

expdp system/password@db10g full=Y directory=TEST\_DIR dumpfile=TESTDB.dmp logfile=expdpTESTDB.log

impdp system/password@db10g full=Y directory=TEST\_DIR dumpfile=TESTDB.dmp logfile=impdpTESTDB.log

## 30. Автоматическое управление хранением (ASM). Назначение, основные возможности.

**Automatic Storage Management (ASM)** — технология в составе СУБД Oracle, реализующая автоматическое управление хранением данных на уровне менеджера томов (подсистема операционных систем Linux и OS/2, позволяющая использовать разные области одного жёсткого диска и/или области с разных жёстких дисков как один логический том).

**Особенности ASM:**

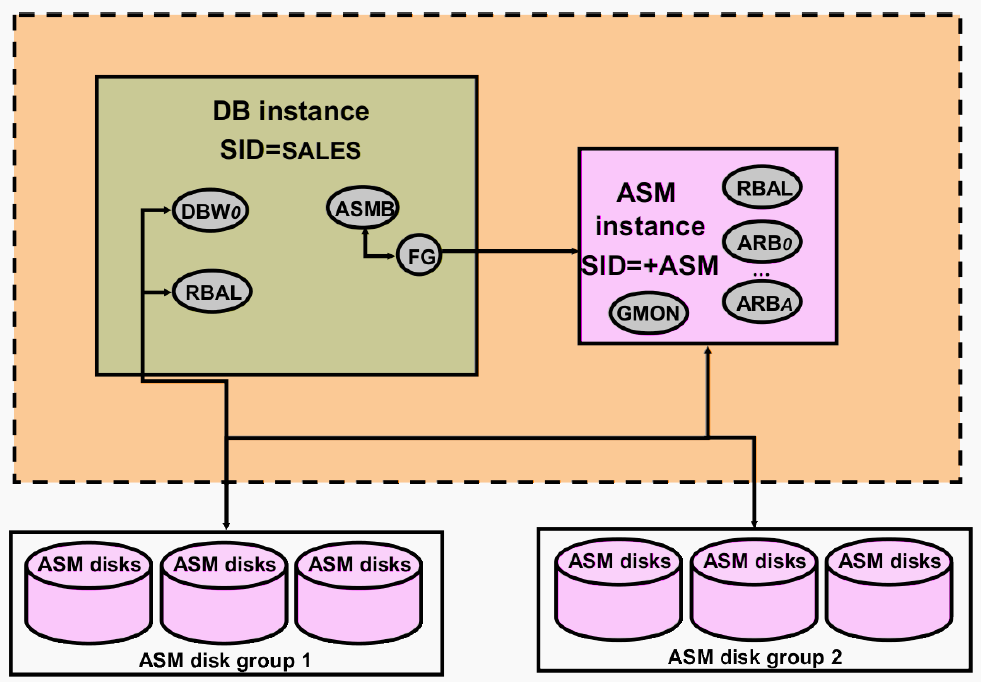
* Управление хранением осуществляется с помощью специальных экземпляров Oracle — ASM Instances.
* ASM может управлять хранением данных как на отдельной машине, так и на уровне кластера RAC (множество экземпляров Oracle, сост. единую БД) в целом.
* Один экземпляр ASM может управлять хранением данных сразу нескольких БД.

**Другие особенности ASM:**

* Управление операциями ввода/вывода осуществляется автоматически — «прозрачно» как для приложения, так и для администратора БД.
* Можно расширять хранилище, добавляя в него дополнительные накопители, без остановки БД.
* ASM может управлять созданием резервных копий данных самостоятельно, либо использовать для этого механизмы на уровне ОС и/или системы хранения данных.
* Файлы БД разделены на «блоки распределения» (Allocation Unit — AU). Для того, чтобы определить, где физически находятся файлы блока распределения, используются специальные индексы.
* При изменении ёмкости хранилища (например, при добавлении в него нового диска), файлы внутри AU автоматически перераспределяются пропорционально произошедшему изменению.
* ASM обеспечивает свой механизм зеркалирования/дублирование, независимый от используемого менеджера томов.
* ASM реализует распределённое хранилище для всех основных файлов в составе экземпляра Oracle — файлов данных, файлов журнала повторов (как оперативных, так и архивных), управляющих файлов и т. д.
* ASM обеспечивает полную поддержку RAC.

## 31. Экземпляр ASM. Конфигурация, взаимодействие с экземпляром Oracle. Дополнительные процессы в составе экземпляра ASM и экземпляра Oracle.

**Архитектура ASM**



Распределённым хранилищем управляет отдельный экземпляр Oracle (ASM Instance), независимый от «основного» экземпляра БД. Этот экземпляр должен быть запущен до запуска экземпляра БД.

Экземпляр БД получает у экземпляра ASM информацию о расположении необходимых ему файлов, после чего работает с ними напрямую, без участия экземпляра ASM.

**Создание экземпляра ASM**

* С помощью DBCA – DataBase Configuration Assistant
* Вручную

**Параметры инициализации экземпляра ASM**

При создании экземпляра ASM используется файл параметров — такой же, как и при создании «обычного» экземпляра Oracle. Тем не менее, он содержит ряд специфичных для ASM параметров:

* INSTANCE\_TYPE — должен быт задан как ASM.
* DB\_UNIQUE\_NAME — имя сервиса ASM.
* ASM\_POWER\_LIMIT — определяет количество ресурсов, которые может использовать ASM при «перебалансировке» БД. Чем выше это значение, тем быстрее ASM выполняет перебалансировку, но тем больше он при этом потребляет ресурсов. Принимает значения от 1 до 11.
* ASM\_DISKSTRING — параметр, определяющий набор дисков, которые «видит» ASM.
* ASM\_DISKGROUPS — список имён дисковых групп, которые «видит» ASM в момент запуска.

**Дополнительные процессы в составе экземпляра ASM и экземпляра Oracle в случае использования ASM.**

Экземпляр ASM содержит ряд дополнительных фоновых процессов, которых нет в «обычном» экземпляре БД:

* *RBAL* (Rebalance) — управляет перераспределением ресурсов при изменениях в дисковых группах.
* *ARB*n (Asm Rebalance Process) — пул процессов, непосредственно осуществляющих перемещение данных AU (блок распределения) между дисками.
* *GMON* (Group Monitor) — осуществляет мониторинг состояния дисков в группах.

В случае использования ASM, в экземпляре БД также появляются «дополнительные» процессы:

* *RBAL* — управляет доступом к дискам в группах.
* *ASMB* (ASM Background Process) — осуществляет взаимодействие с экземпляром ASM.

## 32. Дисковые группы. Назначение, особенности конфигурации. Добавление и удаление дисков. Allocation Units. Coarse- & Fine-Grained Striping.

**Дисковая группа** — логическое объединение нескольких дисков ASM.

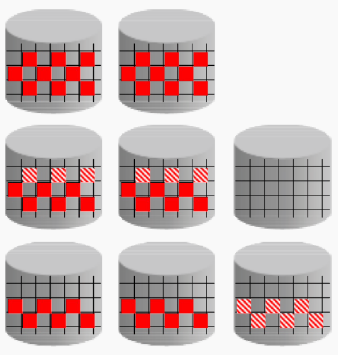
Может хранить данные нескольких БД.

Одна БД может хранить свои данные в нескольких дисковых группах.

Диск может принадлежать только одной дисковой группе.

Файл ASM может быть сохранён только на одной дисковой группе.

Файлы хранятся распределённо — сразу на всех дисках, входящих в соответствующую группу.

**Динамическая «перебалансировка» дисковых групп**

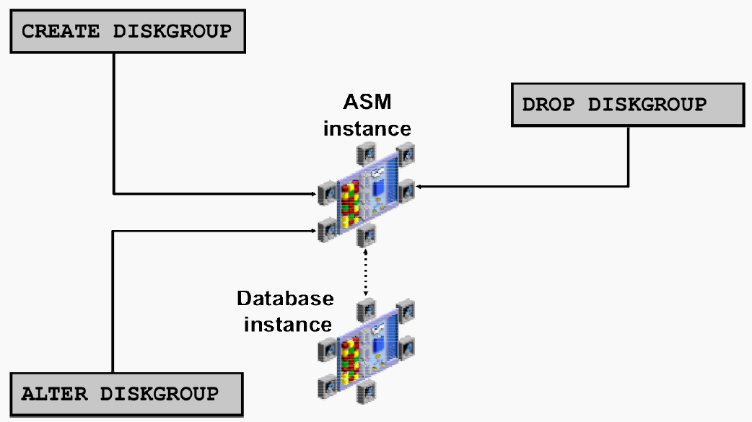
Механизм ASM позволяет изменять состав дисковых групп «на лету», без остановки доступа к данным на них расположенных. При добавлении или удалении дискового устройства из дисковой группы ASM производит автоматическую перебалансировку данных. Перебалансировка может осуществляться с разной степенью интенсивности, что позволяет избежать падения производительности производимых в этот момент операций ввода-вывода с базами данных.

Не требует перезапуска БД или какого-либо ограничения доступа к ней.

Переразмещает данные в соответствии с новой ёмкостью хранилища.

Осуществляется автоматически, никакой дополнительной конфигурации не требуется.

Нагрузкой на систему можно управлять с помощью параметра ASM\_POWER\_LIMIT.

**Управление дисковыми группами**

Для всех операций требуются полномочия SYSDBA или SYSASM.

**Создание и удаление дисковых групп**

≪A≫ и ≪B≫ - разные SCSI-контроллеры — создаём 2 fail groups:

CREATE DISKGROUP dgroupA NORMAL REDUNDANCY (избыточность)

FAILGROUP controller1 DISK

'/devices/A1' NAME diskA1 SIZE 120G FORCE,

'/devices/A2',

'/devices/A3'

FAILGROUP controller2 DISK

'/devices/B1',

'/devices/B2',

'/devices/B3';

DROP DISKGROUP dgroupA INCLUDING CONTENTS;

**Добавление дисков в группы**

ALTER DISKGROUP dgroupA ADD DISK

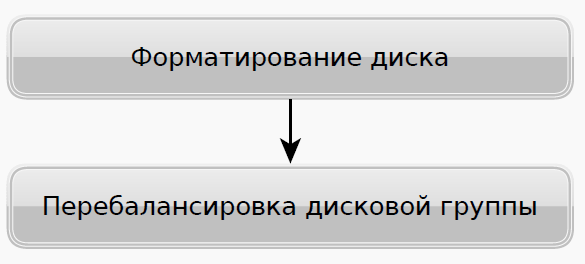
'/dev/rdsk/c0t4d0s2' NAME A5,

'/dev/rdsk/c0t5d0s2' NAME A6,

'/dev/rdsk/c0t6d0s2' NAME A7,

'/dev/rdsk/c0t7d0s2' NAME A8;

ALTER DISKGROUP dgroupA ADD DISK '/devices/A\*';



**Команда ALTER DISKGROUP**

Удаление диска из dgroupA:

ALTER DISKGROUP dgroupA DROP DISK A5;

Удаление и добавление дисков одной командой:

ALTER DISKGROUP dgroupA

DROP DISK A6

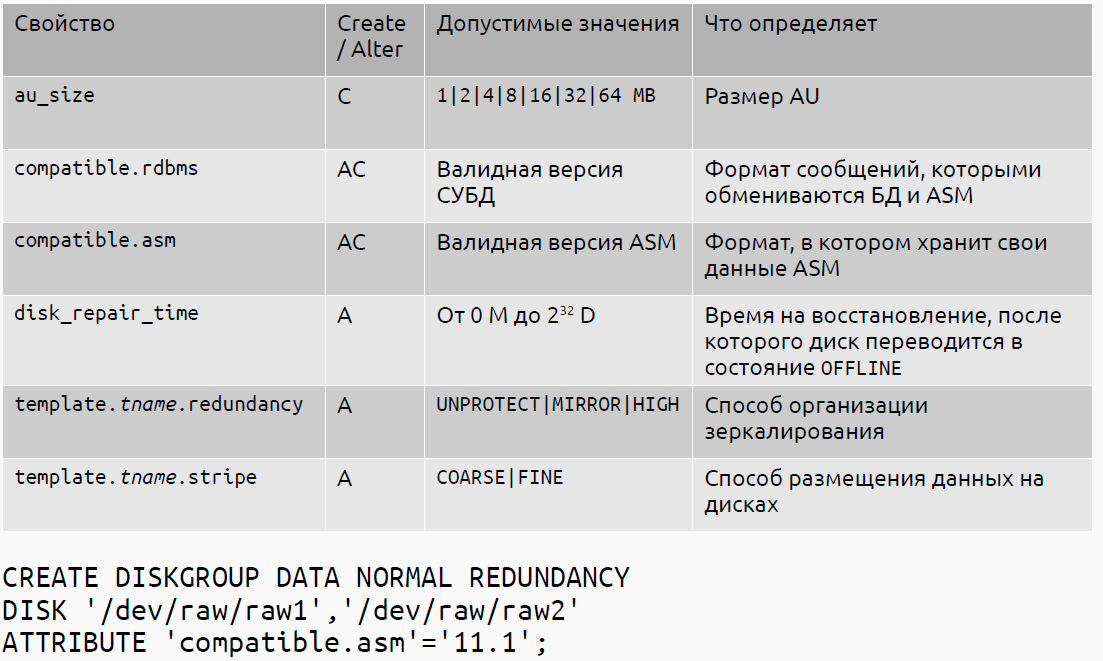
ADD FAILGROUP fred

DISK '/dev/rdsk/c0t8d0s2' NAME A9;

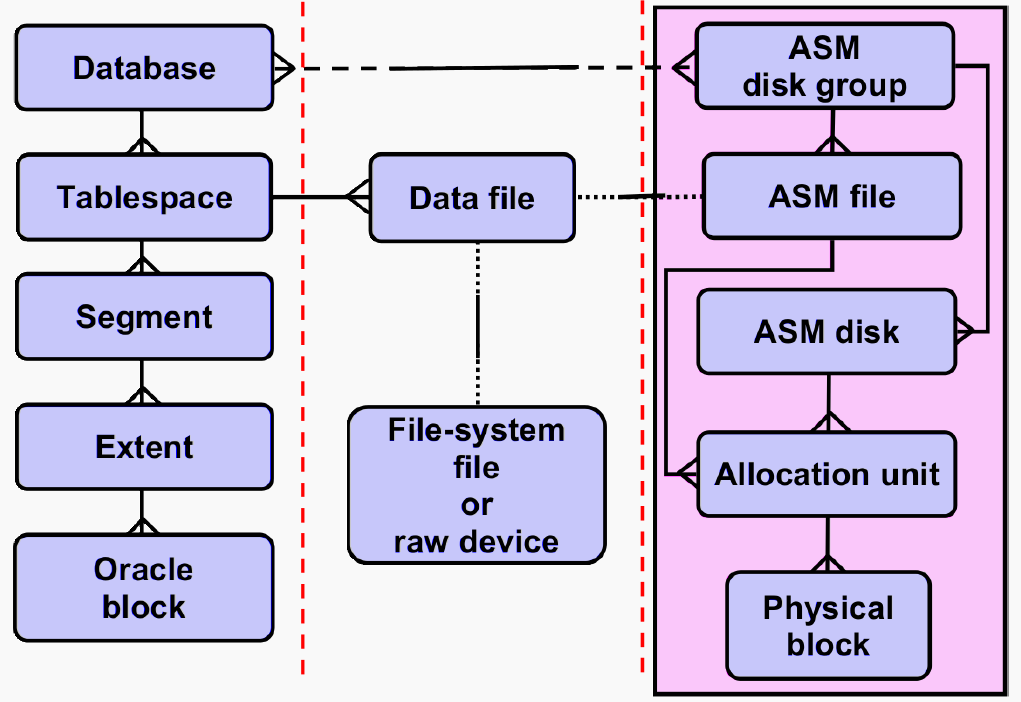
Отмена операции удаления диска:

ALTER DISKGROUP dgroupA UNDROP DISKS;

**Атрибуты дисковых групп**



**Хранилище данных ASM**



ASM никак не влияет на работу БД с ресурсами, существовавшими до конфигурации ASM.

Новые файлы могут создаваться как файлы ASM, а существующие — продолжать администрироваться «по-старому», или быть перенесены под управление ASM.

Верхний уровень иерархии — *дисковая группа ASM* (ASM disk group).

Диски ASM разбиты на *блоки распределения* (allocation units — Aus):

* Размер AU по умолчанию — 1 МБ.
* AU — минимальный объём дискового пространства, которым может оперировать ASM.
* Один блок данных может храниться только в одном конкретном AU.

Oracle ASM striping имеет две основные цели:

* сбалансировать нагрузки по всем дискам в дисковой группе;
* уменьшить задержку ввода / вывода.

**Coarse- & Fine-Grained Striping.**

Coarse-grained striping обеспечивает балансировку нагрузки для дисковых групп, fine-grained striping уменьшает задержки для определенных типов файлов, шире распространяя нагрузку.

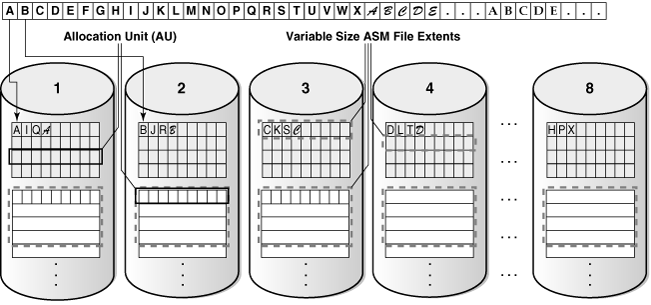
Oracle ASM разделяет файлы на stripes и распространяется данные равномерно по всем дискам в дисковых группах. Размер fine-grained stripe всегда равен 128 KB в любой кофигурации; это обеспечивает низкие задержки операций ввода/вывода. Размер coarse-grained stripe всегда равен размеру AU.

Рисунок 1-5 и Рисунок 1-6 иллюстрации к Oracle ASM file striping. Для обоих рисунков размер allocation unit установлен 1 M (AU\_SIZE = 1M) для дисковой группы, состоящей из 8 дисков.

The Oracle ASM instance is release 11.2 and the disk group compatibility attributes for ASM and RDBMS have been set to 11.2, so variable extents are shown in the graphic after the first 20,000 extents. For the first 20,000 extents, the extent size is 1 M and equals one allocation unit (AU). For the next 20,000 extents, the extent size is 4 M and equals 4 AUs.

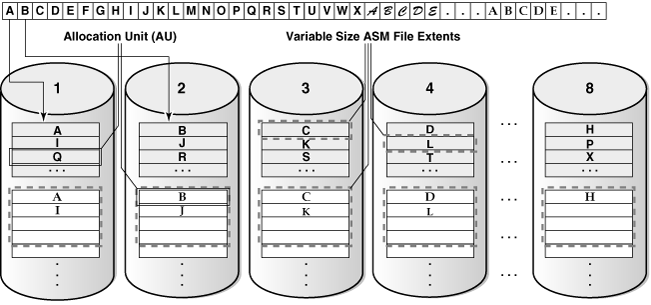
Для идентификации stripe файла они должны быть помечены A..X (24 буквы), используя разные шрифты для последовательного ряда A..X.

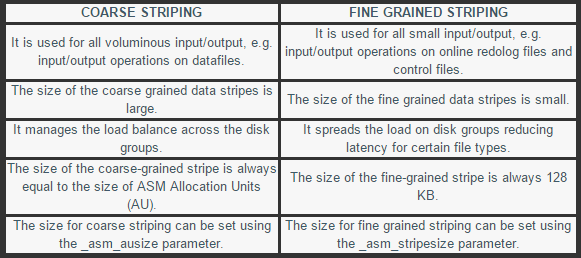
На рисунке 1-5 файл разбит на куски по 128 K (помеченные A..X) with each 128 K chunk stored in an extent, starting at the first extent in disk 1, then the first extent in disk 2, and then continuing in a round-robin pattern through all the disks until the entire file has been striped. As shown in this example, the striping chunks first fill up the first extent of each disk, then the second extent of each disk, and so on until the entire file has been striped. ***Figure 1-5 Oracle ASM Fine-Grained Striping***



In [Figure 1-6](https://docs.oracle.com/cd/E18283_01/server.112/e16102/asmcon.htm#BABDFCBI), the file is striped in 1 M chunks (labeled A..X) with each 1 M chunk stored uniquely in an extent, starting at the first extent in disk 1, then the first extent in disk 2, and then continuing in a round-robin pattern through all the disks until the entire file has been striped. For the first 20,000 extents where the AU equals the extent size (1 M), the stripe equals the extent size and allocation unit size.For the variable extents, where an extent is composed of multiple allocation units, the file stripe is located in an AU of the extent. The striping chunks are placed in the allocation units of the first extents of all the disks before the striping continues to the next extent.

***Figure 1-6 Oracle ASM Coarse-Grained Striping***



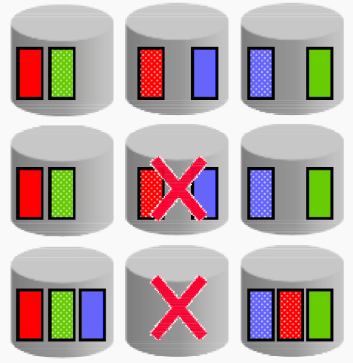


## 33. Избыточность в Oracle ASM. Виды избыточности, конфигурация зеркалирования. Failure-группы.

**Избыточность в Oracle ASM**

ASM поддерживает три режима избыточности данных:

* External/Внешняя - избыточность не поддерживается, используется аппаратная реализация зеркалирования. Рекомендуется использовать при применении RAID массивов (массив из нескольких дисков, управляемых контроллером, связанных между собой скоростными каналами передачи данных и воспринимаемых внешней системой как единое целое) осуществляющих избыточность данных на аппаратном уровне;
* Normal/Стандартная - 2-х кратная избыточность. Поддерживаются две копии одного экстента.
  + Двукратное зеркалирование.
  + Как минимум, 2 failure groups.
* High/Высокая - 3-х кратная избыточность. Поддерживаются три копии одного экстента.
  + Тройное зеркалирование.
  + Как минимум, 3 failure groups.

Для защиты от сбоев аппаратных устройств обеспечивающих работу сразу множество дисков, в дисковых группах можно определить failure группы, при этом избыточность данных будет поддерживаться между дисками находящимися в различных failure группах. Это позволяет обеспечить зеркалирование данных между дисками, находящимися под управлением разных контроллеров и даже между отдельными дисковыми массивами.

**Зеркалирование дисковых групп**

Реализовано на уровне AU (а не на уровне дисков).

На одном и том же диске могут храниться оригиналы и резервные копии различных AU.

Если «оригинал» AU хранится на одном диске, то его «зеркало» будет храниться на другом диске в рамках той же группы.

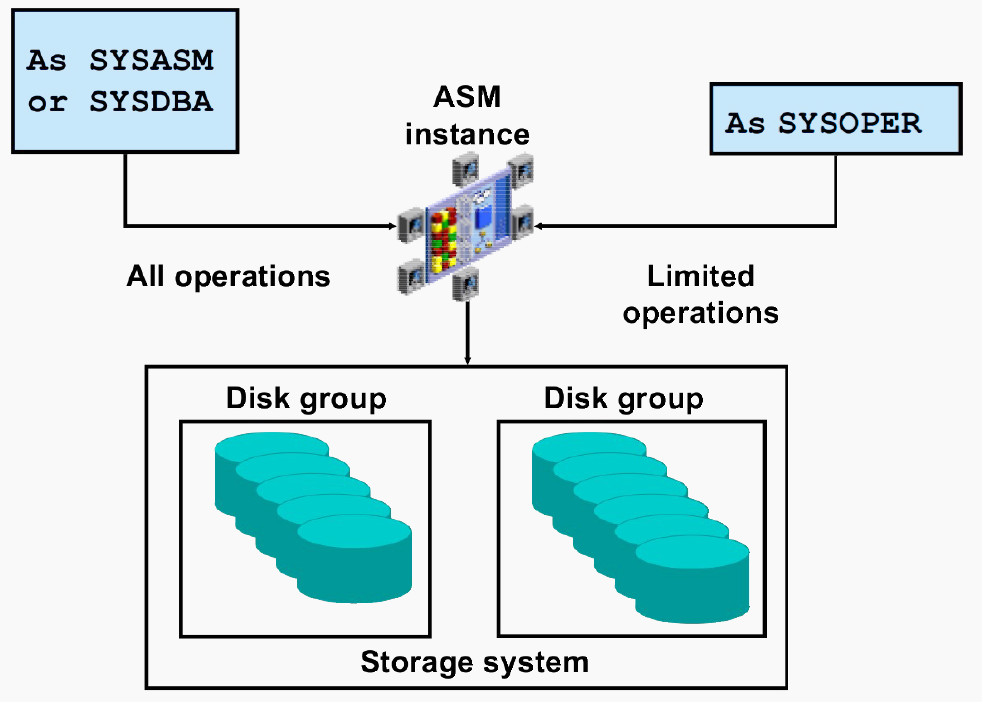
**ASM Failure Group**

Набор дисков внутри конкретной группы, использующий общий ресурс (например, контроллер), от отказа которого должна быть обеспечена защита.

По умолчанию ASM помещает каждый диск в свою собственную failure group. Если вариант «по умолчанию» не устраивает администратора, диски можно перегруппировать.

ASM автоматически располагает данные так, чтобы отказ разделяемого ресурса failure group не привёл к потере данных.

## 34. Роли ASM, отличия в уровне привилегий между ними.



У экземпляров ASM нет своего словаря данных, поэтому к ним можно подключаться только с использованием аутентификации на уровне ОС или файла паролей.

Существует 3 роли, под которыми пользователь может подключиться к ASM:

* SYSASM и SYSDBA — административный доступ без каких-либо ограничений.
* SYSOPER — набор доступных SQL-команд ограничен минимально необходимым для обслуживания уже сконфигурированной системы:
  + STARTUP/SHUTDOWN
  + ALTER DISKGROUP MOUNT/DISMOUNT
  + ALTER DISKGROUP ONLINE/OFFLINE DISK
  + ALTER DISKGROUP REBALANCE
  + ALTER DISKGROUP CHECK
  + SELECT all V$ASM\_\* views

Все остальные команды, в частности, CREATE DISKGROUP, ADD/DROP/RESIZE DISK и т. д., требуют наличия привилегий SYSASM или SYSDBA.

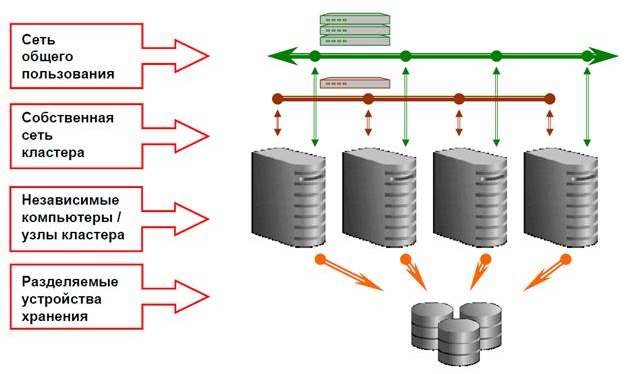
## 35. Архитектура быстрого восстановления с использованием сервера горячего резерва и кластерная архитектура RAC.

**Основные принципы горячего резервирования**

В общем, принципы организации горячего резерва в Oracle довольно просты:

(1) Организуются две активные БД: одна основная, и другая резервная.  
(2) Основная БД работает в режиме архивирования журналов. На возможность подсоединения к ней и на работу с данными никаких ограничений не накладывается.  
(3) Резервная БД работает в специальном режиме постоянного восстановления с архивированных журналов, получаемых от основной БД; она все время "догоняет" основную.   
(4) Из этого специального режима резервную БД можно перевести в другой, когда она приостановит процедуру восстановления и сможет предоставить свои данные, однако только для чтения. Из этого второго режима мы может вернуться в первый, для нее основной.  
(5) Кроме этого, из режима восстановления резервную БД можно перевести в режим обычной полнофункциональной БД, но теперь уже безвозвратно.

**Кластерная архитектура RAC**



Опция Real Application Clusters (RAC) позволяет настраивать отказоустойчивые и хорошо масштабируемые серверы баз данных на основе объединения нескольких вычислительных систем. В архитектуре RAC экземпляры СУБД Oracle одновременно выполняются на нескольких объединенных в кластер системах, производя совместное управление общей базой данных.

По существу, с точки зрения приложения – это единая СУБД. RAC используется для создания корпоративных сетей распределенной обработки данных. Такие сети строятся из большого количества стандартизованных недорогих компонентов: процессоров, серверов, сетевых устройств и устройств хранения данных.

**Масштабируемость**

RAC дает пользователям возможность добавлять в кластер новые узлы при возрастании требований к ресурсам, производить постепенное увеличение мощности системы при оптимизации затрат на оборудование.

**Высокая готовность**

Другое ключевое преимущество кластерной архитектуры на основе Oracle RAC – присущая ей устойчивость к отказам за счет наличия множества узлов. Поскольку физические узлы работают независимо друг от друга, отказ одного или нескольких узлов не оказывает влияния на работу остальных узлов кластера. Аварийное переключение сервиса может быть произведено на любой узел Grid. В самой крайней ситуации система на базе Real Application Clusters способна поддерживать работу базы данных даже при отказе всех узлов за исключением одного.

RAC имеет встроенные средства интеграции с сервером приложений Oracle Application Server для аварийного переключения пулов соединений. Благодаря этому приложение получает информацию об отказе немедленно, не тратя десятки минут ожидания до истечения тайм-аута TCP-соединения, и сразу предпринять необходимые меры по восстановлению. Средства балансировки нагрузки позволяют перераспределить нагрузку равномерно между ресурсами Grid.

**Баланс нагрузки**

RAC поддерживает новую абстракцию, получившую название сервиса. Сервисы представляют классы пользователей базы данных или приложений. Задание и применение бизнес-политик к сервисам позволяет разрешать такие проблемы, как выделение узлов на периоды пиковых вычислительных нагрузок или автоматическое устранение последствий отказа сервера. Это гарантирует предоставление системных ресурсов в требуемый период времени и там, где это необходимо для решения поставленных задач.

**Единый стек кластерного ПО**

Oracle Database Enterprise Edition и Опция Real Application Clusters составляют полный комплект ПО управления и функционирования кластера. Кластерное программное обеспечение Oracle (Clusterware) предоставляет все необходимые возможности, требуемые для работы кластера, включая учет узлов, службы сообщений и блокировки.

## 36. Понятие кластера. Аппаратная и программная реализация кластера.Масштабируемость и отказоустойчивость кластерных решений.

**Кластер** — группа компьютеров, объединённых высокоскоростными каналами связи, представляющая с точки зрения пользователя единый аппаратный ресурс.

**Кластер** - слабо связанная совокупность нескольких вычислительных систем, работающих совместно для выполнения общих приложений, и представляющихся пользователю единой системой.

Кластер состоит из двух или более взаимосвязанных машин — узлов кластера.

Узлы кластера «видны» снаружи как единое целое.

Внутренняя структура кластера «скрывается» кластерным ПО — все кластеры «выглядят»

снаружи как обычные серверы БД.

Все диски доступны для чтения и записи всем узлам кластера.

На всех узлах кластера используется одна и та же версия ОС.

Кластер мб либо отказоустойчивым, либо масштабируемым. Упор на что-то одно.

**Уровни масштабируемости**

Для того, чтобы использование кластера было эффективным, требуется обеспечить масштабируемость на всех уровнях:

* Аппаратный — скорость чтения / записи на диски.
* Взаимодействие между узлами — пропускная способность сети и время отклика.
* Операционная система — возможность работы на многопроцессорных машинах.
* СУБД — синхронизация при параллельном доступе к данным.
* Приложение — особенности архитектуры.

Два варианта масштабирования: горизонтальное и вертикальное.

Большинство компьютерных систем допускают несколько способов повышения их производительности: добавление памяти, увеличение числа процессоров в многопроцессорных системах или добавление новых адаптеров или дисков. Такое масштабирование называется **вертикальным** и позволяет временно улучшить производительность системы. Однако в системе будет установлено максимальное поддерживаемое количество памяти, процессоров или дисков, системные ресурсы будут исчерпаны. И пользователь столкнется с той же проблемой улучшения характеристик компьютерной системы, что и ранее.

**Горизонтальное** масштабирование предоставляет возможность добавлять в систему дополнительные компьютеры и распределять работу между ними. Таким образом, производительность новой системы в целом выходит за пределы предыдущей. Естественным ограничением такой системы будет программное обеспечение, которые вы решите на ней запускать. Самым простым примером использования такой системы является распределение различных приложений между разными компонентами системы. Например, вы можете переместить ваши офисные приложения на один кластерный узел приложения для Web на другой, корпоративные базы данных — на третий. Однако здесь возникает вопрос взаимодействия этих приложений между собой. И в этом случае масштабируемость обычно ограничивается данными, используемыми в приложениях. Различным приложениям, требующим доступ к одним и тем же данным, необходим способ, обеспечивающий доступ к данным с различных узлов такой системы. Решением в этом случае становятся технологии, которые, собственно, и делают кластер кластером, а не системой соединенных вместе машин. При этом, естественно, остается возможность вертикального масштабирования кластерной системы. Таким образом, за счет вертикального и горизонтального масштабирования кластерная модель обеспечивает серьезную защиту инвестиций потребителей.

В качестве варианта горизонтального масштабирования стоит также отметить использование группы компьютеров, соединенных через коммутатор, распределяющий нагрузку (технология Load Balancing).

**Отказоустойчивость**

Под отказоустойчивостью понимается доступность тех или иных функций в случае сбоя, другими словами, это резервирование функций и распределение нагрузки.

Приведем пример серверного кластера: каждый сервер в кластере остается относительно независимым, то есть его можно остановить и выключить (например, для проведения профилактических работ или установки дополнительного оборудования), не нарушая работоспособность кластера в целом. Тесное взаимодействие серверов, образующих кластер (узлов кластера), гарантирует максимальную производительность и минимальное время простоя приложений за счет того, что:

* в случае сбоя программного обеспечения на одном узле приложение продолжает функционировать (либо автоматически перезапускается) на других узлах кластера;
* сбой или отказ узла (или узлов) кластера по любой причине (включая ошибки персонала) не означает выхода из строя кластера в целом;
* профилактические и ремонтные работы, реконфигурацию и смену версий программного обеспечения в большинстве случаев можно осуществлять на узлах кластера поочередно, не прерывая работу приложений на других узлах кластера.

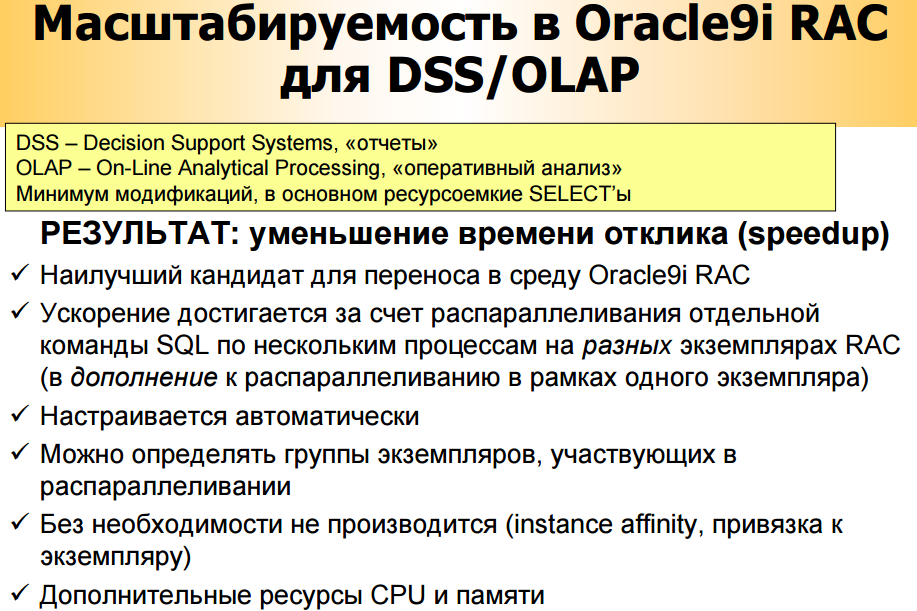
Возможные простои, которые не в состоянии предотвратить обычные системы, в кластере оборачиваются либо некоторым снижением производительности (если узлы выключаются из работы), либо существенным сокращением (приложения недоступны только на короткий промежуток времени, необходимый для переключения на другой узел), что позволяет обеспечить уровень готовности в 99,99%.

## 37. Основные принципы построения масштабируемых приложений. Speedup & Scaleup.

**ScaleUp** - способность сохранять те же самые уровни производительности (время отклика), когда и нагрузка (транзакции) и ресурсы (процессор, память) пропорционально возрастают.

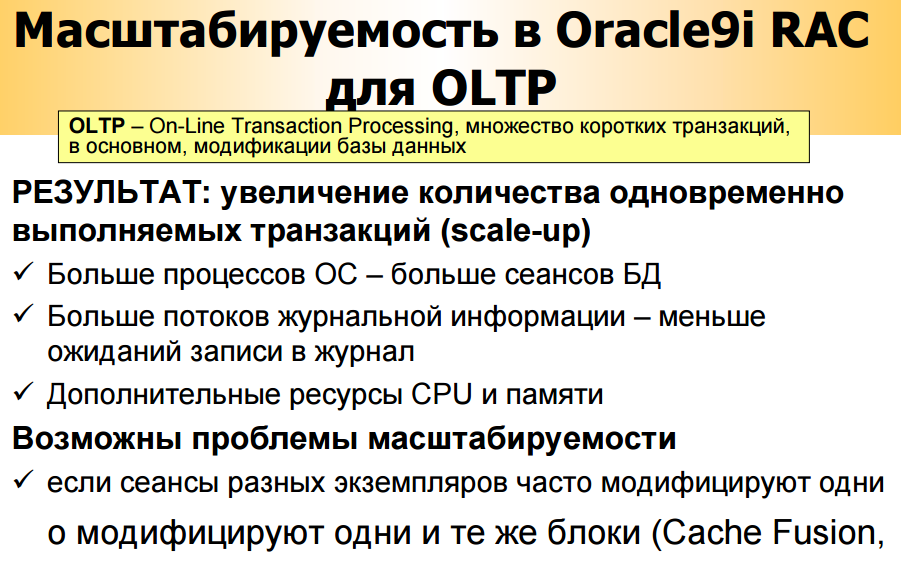
Например, если 50 пользователей потребляют почти 100 процентов ЦП, затем добавить больше пользователей, то система замедлится из-за конкуренции за ограниченные ресурсы ЦП. Однако, за счет добавления новых ЦП, мы можем добавить новых пользователей без снижения производительности.

ScaleUp увеличивает пропускную способность, т.е. количество задач, выполняемых в данный момент времени, увеличивается.

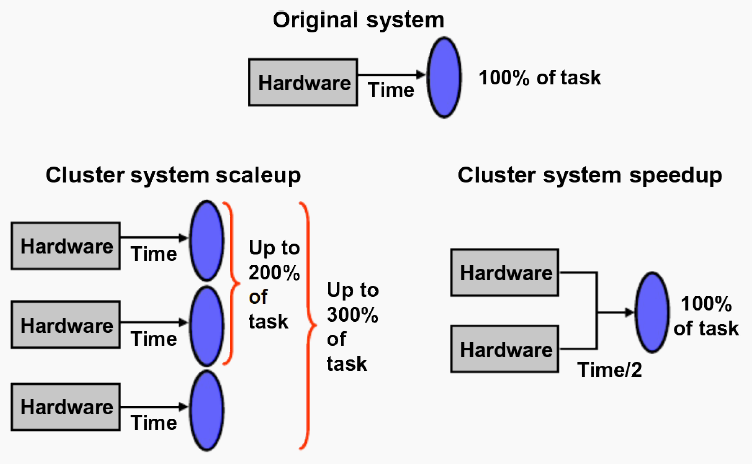


**Speedup** - эффект применения большого количества ресурсов для фиксированного объема работ для достижения пропорционального снижение времени выполнения:

Speedup достигается за счет наличия ресурсов для других задач. Например, если запросы обрабатываются, как правило, десять минут одним ЦП, то параллельная работа нескольких ЦП сокращает время.



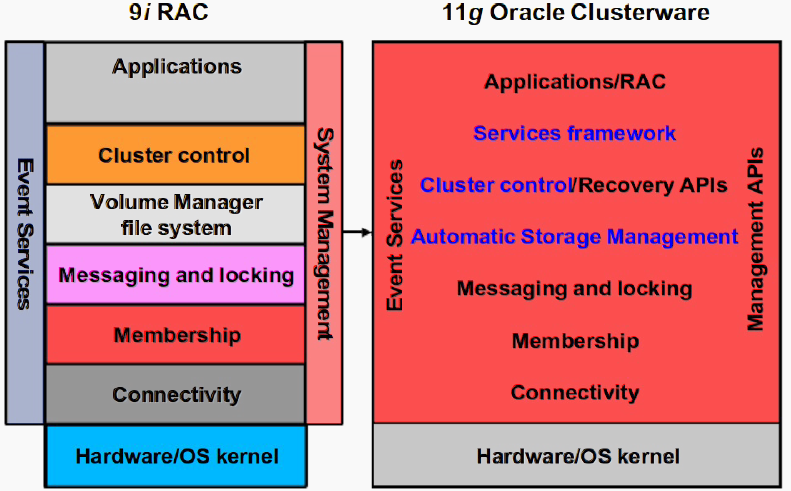
|  |  |
| --- | --- |
| **SPEED UP** | **SCALE UP** |
| Улучшает время отклика | Улучшает пропускную способность |
| As you multiply resources by a certain factor, the time taken to execute a transaction should be reduced by the same factor: | As you multiply resources the size of a task that can be executed in a given time should be increased by the same factor. |
| 10 seconds to scan a DB of 10,000 records using 1 CPU  1 second to scan a DB of 10,000 records using 10 CPUs | 1 second to scan a DB of 1,000 records using 1 CPU  1 second to scan a DB of 10,000 records using 10 CPUs |
| Концентрируется на времени, что значит улучшение временем отклика т.е. ускорение. | Концентрируется на данных, что значит улучшение масштабируемости. |
| [http://4.bp.blogspot.com/-09YjHA_0S24/TbEImTHZGPI/AAAAAAAAAME/aRvviceekaA/s320/Picture1.png](http://4.bp.blogspot.com/-09YjHA_0S24/TbEImTHZGPI/AAAAAAAAAME/aRvviceekaA/s1600/Picture1.png) | [http://3.bp.blogspot.com/-_Bj1zX9V2Ds/TbEH2Iwv94I/AAAAAAAAAMA/EZScxcb1-x0/s320/Picture2.png](http://3.bp.blogspot.com/-_Bj1zX9V2Ds/TbEH2Iwv94I/AAAAAAAAAMA/EZScxcb1-x0/s1600/Picture2.png) |



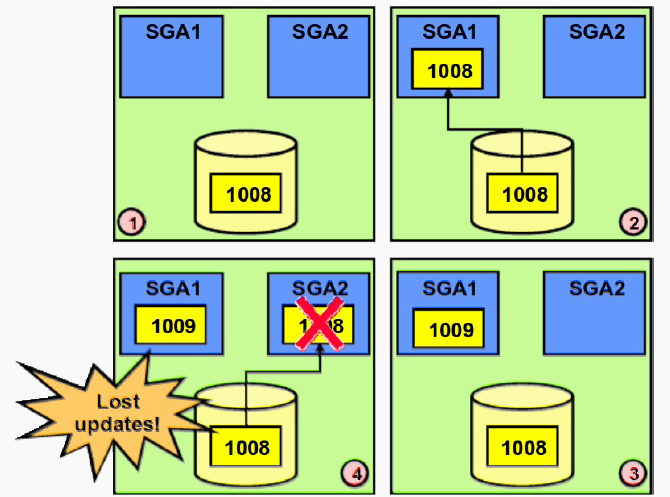


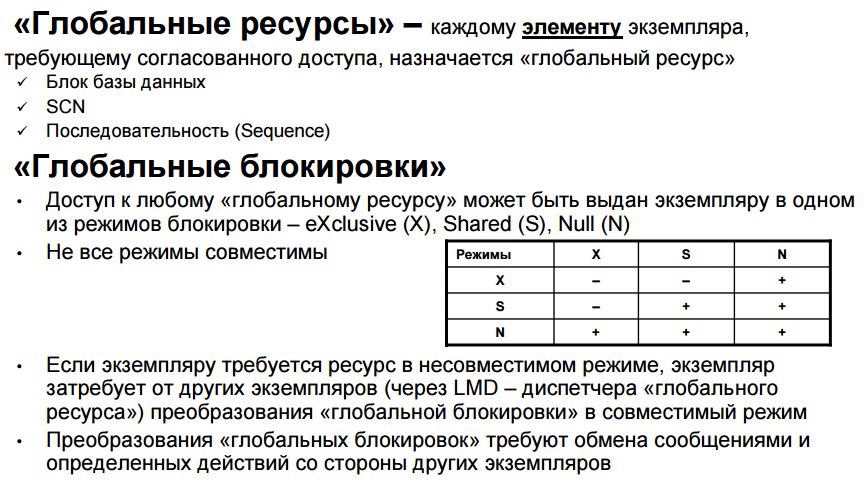
## 38. Глобальные ресурсы Oracle RAC, особенности управления ими. Глобальные DPV.

**Архитектура Oracle RAC**

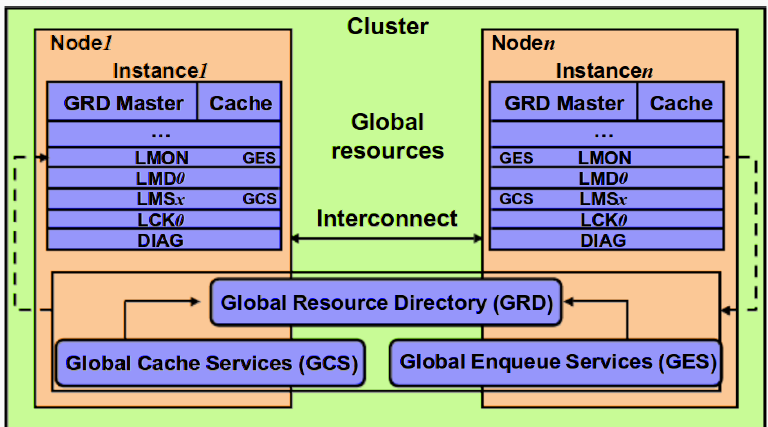


**Зачем нужны глобальные ресурсы?**



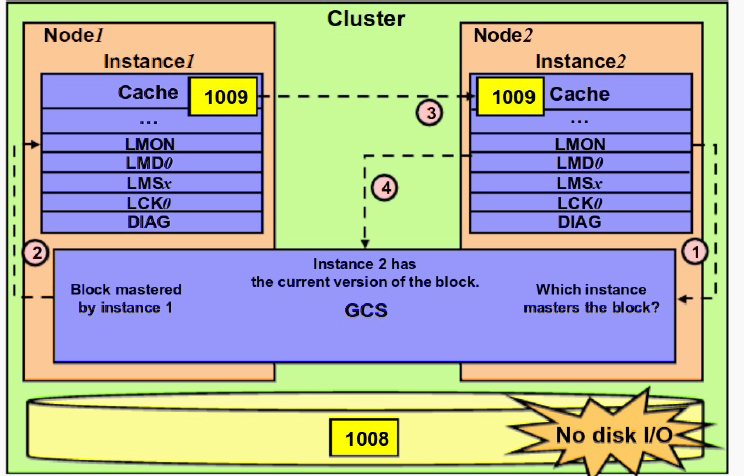


**Управление глобальными ресурсами**

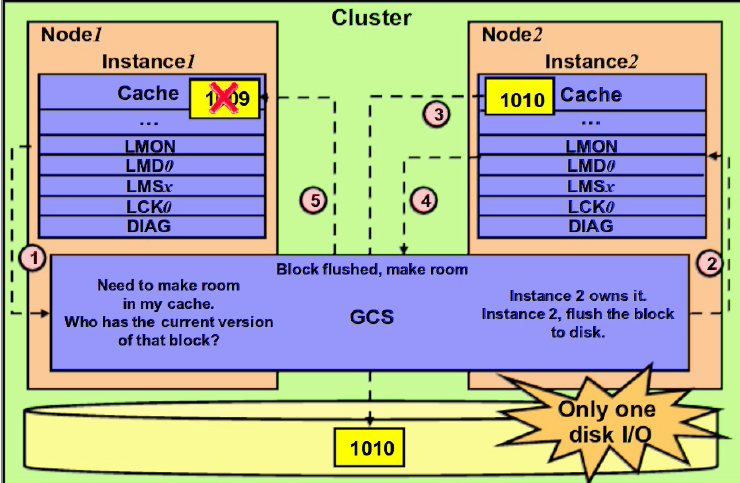




**Синхронизация глобального кэша**



**Координация записи на диск**

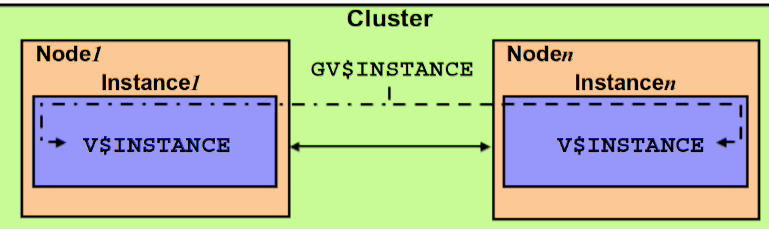


**Глобальные DPV**

Содержат информацию обо всех запущенных экземплярах в составе кластера. У каждого локального представления (V$) есть соответствующее ему глобальное представление (GV$).

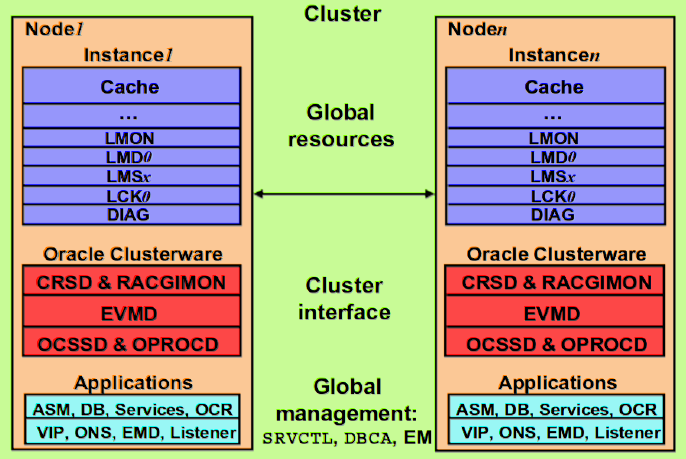
Исполняются параллельно на всех узлах кластера — «ведущий» запрос на узле, к которому осуществляется обращение и «ведомые» запросы к V$ на остальных узлах.

Параллелизмом управляет специальный сервис — координатор параллельного исполнения (Parallel Execution Coordinator, PEC).



## 39. Архитектура Oracle RAC: процессы, конфигурационные файлы, файлы БД.

**Процессы**

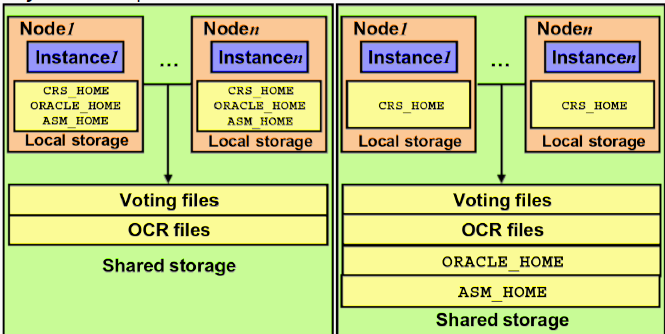


**Конфигурационные файлы**

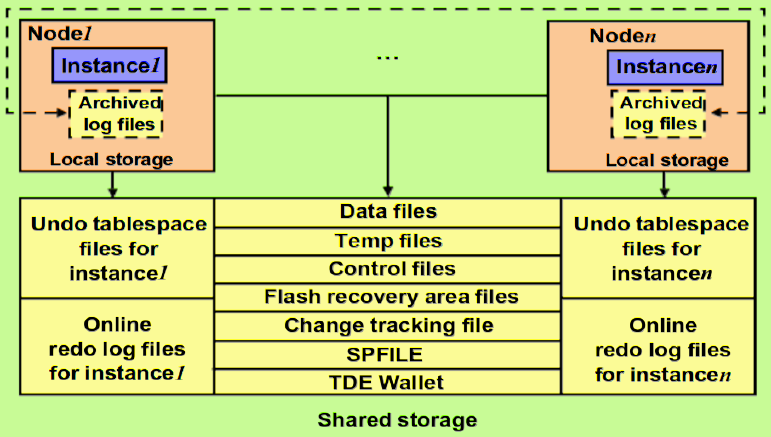
Появляются 2 новых категории файлов

* Файлы с результатами мониторинга состояния кластера.
* Файлы OCR — Oracle Cluster Registry.

Файлы OCR не могут храниться в ASM, т. к. они должны быть доступны ещё до запуска кластера.



**Файлы БД**



## 40. Варианты построения системы хранения в Oracle RAC, их преимущества и недостатки.

**Преимущества CFS:**

* Проще администрировать.
* Ставится вместе с Oracle, не нужна дополнительная конфигурация.
* Автоматически расширяется по мере возрастания объёма данных.
* Можно использовать для хранения файлов архива журнала повторов.

**Преимущества raw:**

* Потенциально быстрее.
* Ниже требования к инфраструктуре.
* Можно использовать ASM для расширения возможностей.

В разделяемом пространстве можно хранить файлы данных (database files), данные для восстановления (recovery files), исполняемые файлы (binaries), OCR(Oracle Custer Registry) и Voting Files.

* RAW devices (сырые устройства) - специальные символьные (character) устройства, предназначенные для организации доступа к блочным устройствам без кэширования.
* Гарантирована запись, исключена проблема двойной буферизации (в СУБД и ОС), ниже требования к инфраструктуре, потенциально быстрее всех, можно использовать ASM для расширения.

Для Oracle 1 raw устройство == 1 файлу => нужно много файлов, средствами ОС (df, ls -l) не оценить используемое дисковое пространство и не сделать бэкап/восстановление, можно хранить только файлы БД, не поддерживаются с Oracle 12g

* CFS (Clustered File System) -  эмулирует традиционную ФС с дополнительными возможностями по управлению разделяемыми данными и метаданными
* Просто администрировать, все скрипты работают так же как и локальной ФС, можно хранить не только файлы БД(архив журнала повторов), OCFS ставится вместе с Oracle, автоматически расширяется.

- Сложность начальной конфигурации, возможно уменьшается производительность, в стек БД добавляется новый продукт.

* ASM (Automatic Storage Management) рекомендация Oracle для всего, кроме бинарников.  ASM - урезанный экземпляр Oracle
* нет требования один диск == один файл, добавляет поддержку RAID, кроссплатформенность, благодаря зеркалированию не нужно полностью восстанавливать данные на диске при его падении

- не во всех версиях Oracle можно хранить любые файлы, не относящиеся к БД

* NFS (Network File System) + RAC - выглядит как CFS
* все компоненты БД можно хранить в NFS (+исполняемые),  NFS сервер имеет больше функциональных возможностей, чем raw devices
* Часто обходится дороже чем raw, некоторая нагрузка может не масштабироваться, в мире баз данных уже считается устаревшей, медленной и ненадежной технологией.

Все перечисленные технологии могут комбинироваться