

1. Дистрибутивы СУБД Oracle. Установка СУБД Oracle 12c на Windows. Global Database Name и SID.
2. Основные системные пользователи. Основные специальные привилегии. Роль DBA.
3. Понятия базы данных и экземпляра базы данных.
4. Запуск и останов экземпляра базы данных Oracle.
5. Словарь БД: назначение, применение, основные представления.
6. Мультиарендная архитектура Oracle Multitenant.
7. Файлы Oracle. Файл параметров, управл файлы, паролей, трассировки.
8. Файлы Oracle. Файлы данных, журналы, архивы.
9. Абстрактная модель Oracle. Логическая структура внешней памяти.
10. Абстрактная модель Oracle. Физическая структура внешней памяти.
11. Абстрактная модель Oracle. Структура SGA.
12. Абстрактная модель Oracle. Серверные процессы Oracle.
13. Абстрактная модель Oracle. Фоновые процессы Oracle.
14. Процесс-слушатель Oracle и его основные параметры.
15. Сетевые настройки Oracle. Установление соединения по сети.
16. Табличные пространства СУБД Oracle и их основные параметры.
17. Роли и привилегии СУБД Oracle и их основные параметры. Роль-поименованный набор привилегий, создается привилегированным администратором. Применяется для управления привилегиями пользователей.
18. Пользователь СУБД Oracle и его основные параметры.
19. Профиль безопасности СУБД Oracle и его основные параметры.
20. Язык SQL. Основные операторы и их назначение.
21. Таблица и ее основные параметры.
22. Временные таблицы.
23. Ограничения целостности в таблицах.
24. Типы данных базы данных.
25. Индексы базы данных. Виды и особенности применения индексов.
26. Последовательность СУБД Oracle и ее параметры.
27. Кластер и его параметры
28. Представление и его параметры.
29. Материализованные представления СУБД Oracle.
30. Частные и публичные синонимы СУБД Oracle.
31. Основные характеристики языка PL/SQL.
32. Структура программы языка PL/SQL. Анонимные и именованные блоки.
33. Типы данных, основные операции, константы языка PL/SQL.
34. Поддержка национальных языков в СУБД Oracle. Наборы символов. Байтовая и символьная семантика символов.
35. Связ.объявления пер-ных: инструкция %TYPE, инструкция %ROWTYPE.
36. Локальные процедуры и функции языка PL/SQL.
37. Использование записей в PL/SQL. Вложенные записи.
38. Операторы управления, операторы цикла языка PL/SQL.
39. Курсоры. Виды курсоров. Схемы обработки курсора.

40. Курсоры. Атрибуты курсора. Курсоры с параметрами.
41. Курсорные переменные. Параметры инстанса, связанные с курсорами.
42. Курсорные подзапросы.
43. Использование конструкции CURRENT OF в курсорах.
44. Динамические курсоры.
45. Применение псевдостолбцов ROWID, ROWNUM в PL/SQL.
46. Обработка исключений в PL/SQL, стандартные исключения, генерация и обработка исключения.
47. Принцип распространения исключений в PL/SQL. Инструкция RAISE\_APPLICATION\_ERROR.
48. Встроенные ф-ции языка PL/SQL. Ф-ции работы с датами, текстом и числами.
49. Встроенные функции языка PL/SQL. Функции регулярных выражений
50. Коллекции. Массивы переменной длины в PL/SQL.
51. Коллекции. Вложенные таблицы в PL/SQL.
52. Коллекции. Ассоциативные массивы в PL/SQL.
53. Процедурные объекты.
54. Процедурные объекты. Хранимые процедуры. Вызов процедур. Входные и выходные параметры, позиционный и параметрический форматы передачи фактических параметров. Значения параметров по умолчанию.
55. Процедурные объекты. Хранимые функции. Параметры функции. Вызов функций. Понятие детерминированной функции. Значения параметров по умолчанию.
56. Процедурные объекты. Пакеты. Спецификация и реализация пакета. Процедурные объекты. Триггеры. Виды триггеров. Классификация, порядок выполнения и предикаты триггеров. Триггеры замещения. Привилегии. Включение/отключение триггеров. Псевдозаписи old и new.
57. Секционирование таблиц. Виды секционирования.
58. Обработка заданий. Системные пакеты обработки заданий в Oracle.
59. Транзакции. Виды транзакций.
60. Системные пакеты Oracle.

## 1. Дистрибутивы СУБД Oracle. Установка СУБД Oracle 12c на Windows. Global Database Name и SID.

**Oracle** — открытый дистрибутив, доступный и свободный для скачивания  
Дистрибутивы:

на [oracle.com](http://oracle.com) выбрать Database 12c > Download > 2 файла для вашей ОС

**SID (System ID)** – уник. имя, кот. однозначно идентиф. экземпляр/БД

Каждый экземпляр базы данных идентифицируется с помощью **SID (System Identifier)** – системный идентификатор). Он хранится в переменной среды **ORACLE\_SID** и используется утилитами и сетевыми компонентами для доступа к базе данных. Кроме понятия **SID** существует также и понятие **SERVICE NAME**, которые зачастую не различают.

Глобальное имя базы данных однозначно отличает базу данных от любой другой базы данных в той же сети. Вы указываете глобальное имя базы данных при создании базы данных во время установки или с помощью помощника по настройке базы данных Oracle.

**ПРИ ИНСТАЛЛЯЦИИ СОЗДАЮТСЯ 2 табличных пространства:**  
**SYSTEM, SYSAUX**

## 2. Основные системные пользователи. Основные специальные привилегии. Роль DBA.

Осн. системные пользователи:

. может выполнять все административные функции:

\* **SYS** – предопред. привилегир. юзер ранга админа БД, кот. явл. владельцем ключевых ресурсов БД Oracle. предоставляются привилегия SYSDBA и SYSOPER, которые позволяют выполнять административные задачи высокого уровня

\* **SYSTEM** – предопред. привилег. юзер, кот-му принадлежат ключевые ресурсы БД Oracle. кроме: резервное копирование и восстановление, обновление базы данных

Они оба создаются с паролем, который вы указали при установке, и им обоим автоматически назначается роль DBA.

Специальные сист. привилегии:

\* **SYSDBA, SYSOPER** – спец. привилегии админа, позвол. выполнять базовые задачи администрирования: запуск и остановка экземпляра БД; создание, удаление, открытие, монтирование бд...

**DBA** – предопред. роль, кот. авто- создается для каждой БД oracle и содержит все сист. привилегии, кроме SYSDBA и SYSOPER.

**Привилегия** - это право выполнять конкретный тип предложений SQL, или право доступа к объекту другого пользователя.

ORACLE имеет два вида привилегий: системные и объектные. Назначаются оператором GRANT. Отзываются оператором REVOKE.

ALTER	PROCEDURE
DELETE	PROFILE
EXECUTE	ROLE
INDEX	ROLLBACK SEGMENT
INSERT	SESSION
REFERENCE S	SEQUENCE
SELECT	SYSTEM
UPDATE	TABLE
	TABSPACE
	TRIGGER
	USER
	VIEW

Объектные пр:

Системные:

### 3. Понятия базы данных и экземпляра базы данных.

БД – набор физ. файлов в ОС (мб доступна в нескольких экземплярах)

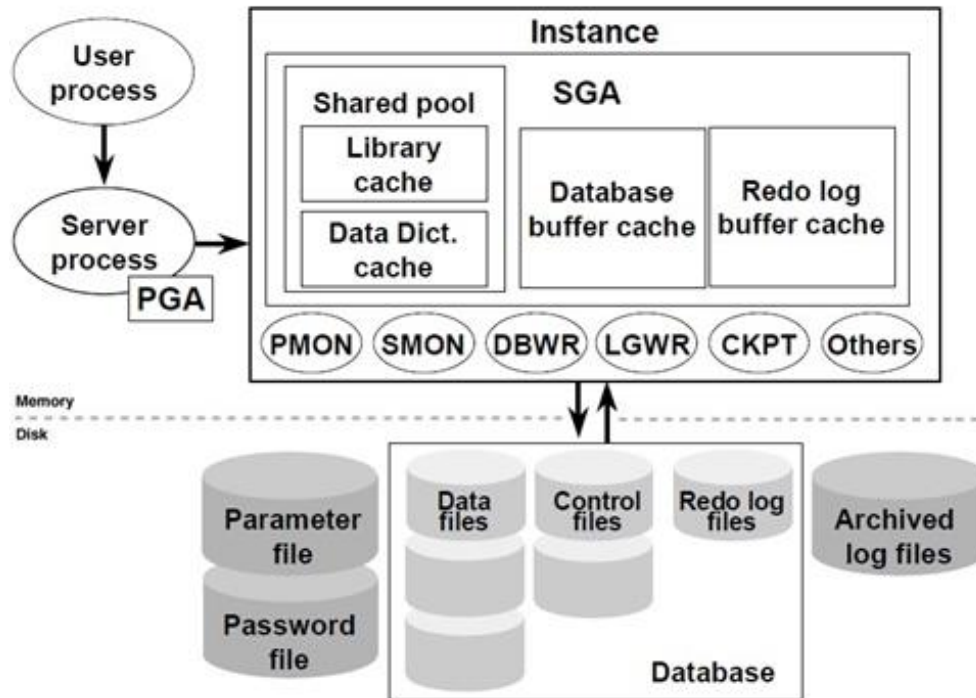
Экземпляр – набор процессов и область SGA (обесп доступ только к 1 БД)

Экземпляр (инстанс) составляет три комп:

- \* запущенный сервер (программа) СУБД

- \* общая (глоб) область памяти : SGA и другие системные области памяти ,которую разделяют все основные процессы запущенного экземпляра. Каждый запущенный экземпляр активно использует ресурсы процессора и памяти.

- \* фоновые процессы, предназн. для управления файлами БД



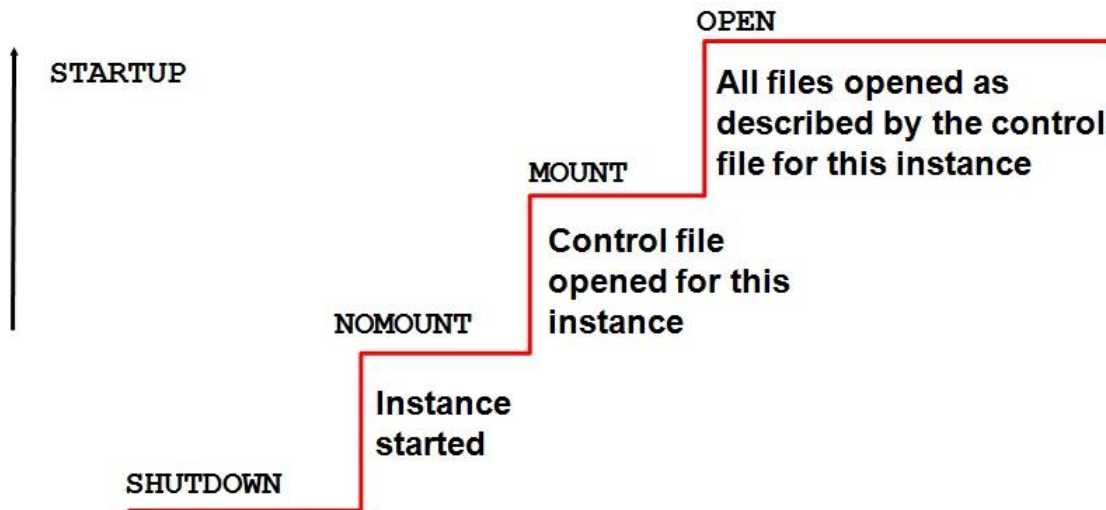
Компоненты архитектуры Oracle:

- Файлы, образующие базу данных и поддерживающие экземпляр - файлы параметров, сообщений, данных, временных данных и журналов повторного выполнения

- Структуры памяти - системная глобальная область (System Global Area — SGA) и входящие в SGA пулы

- Физические процессы или потоки - серверные процессы, фоновые процессы и подчиненные процессы

#### 4. Запуск и останов экземпляра базы данных Oracle.



**SHUTDOWN**(выключение) – новые подключения никогда не разрешаются!

**NOMOUNT**(номинант):

- запуск экземпляра
- выбор состояния, в которое перейдет экземпляр

**MOUNT**(монтировать):

- открыты контрольные файлы для экземпляра
- база данных ассоциируется с экземпляром
- читается файл параметров, инициализируются фоновые процессы
- база данных пока не запущена
- пользователи пока не подключаются
- определяются управляющие файлы

**OPEN**(открывать):

- все файлы открыты
- база данных запущена и открыта
- все связывается с дисковыми структурами
- доступно подключение пользователей

## 5. Словарь БД: назначение, применение, основные представления.

Словарь данных – набор таблиц. Пользователи просматривают содержимое словаря данных с помощью представлений.

Таблицы словаря данных описывают всю базу данных: ее логическую и физическую структуру, использование свободного пространства, ее объекты с их ограничениями, а также информацию о пользователях.

Словарь Oracle – набор таблиц и связ. с ними представлений, позвол. отследить внутр. структуру БД и деятельность СУБД oracle

- \* созд при генерации БД
- \* обнов сервером в фоновом режиме
- \* располож в системном таблпростр – SYSTEM
- \* владелец – юзер SYS, нек. представления – SYSTEM
- \* привилегия – grant select any dictionary

Запросы к словарю/обращение к его представлениям:

- USER – объекты юзера
- ALL – объекты, к кот. юзер имеет доступ
- DBA – все объекты БД (для админа)
- V\$ - производительность сервера

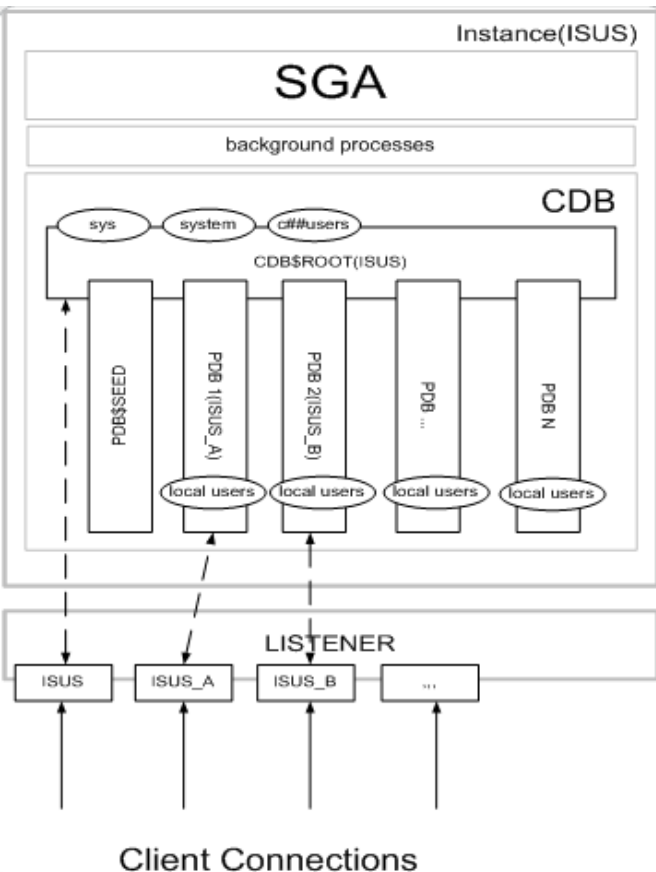
Основные прдставления:

```
select * from dictionary / v$session/ v$parameter/ v$instance;/ dba_tablespaces; /  
dba_users; / dba_profiles; / dba_roles;/ dba_(role | sys | tab | col)_privs;
```

## 6. Мультиарендная архитектура Oracle Multitenant.

Oracle Multitenant – технология, позвол. запустить неск. независимых БД в рамках одного экземпляра

- \* Каждая БД имеет свой набор табл простр и набор схем, но при этом у них общая SGA и один набор серверных процессов
- \* БД изолированы, друг о друге ничего не знают, не конфликтуют
- \* Словарь разбивается на 2 части: общую и локальную



- CDB – Container DB (контейнер бд)
- PDB – Pluggable DB (подключаемая бд)
- \* Можно создать неск. CDB – для разных версий ПО СУБД
- \* Одну PDB можно переносить между CDB
- \* В CDB создается главный контейнер Root (содержит метаданные CDB)
- \* В одной CDB м. создать до 252 PDB
- Локальный пользователь: создается только в PDB, администрирует только свою PDB

## 7. Файлы Oracle. Файл параметров, управл файлы, паролей, трассировки.

Файлы параметров(.ora) – хранит параметры инициализации экземпляра, например, количество сессий, размер SGA, количество процессов, курсоров.

\*select name, description from v\$parameter;

Упр. файлы(.ctl) – сод. имена осн. физ. файлов БД и нек. параметров – использ для поиска др. файлов ОС

\*для надежности созд. 2 упр.ф. (на разн диск носит)

\*show parameter control;--полная инфа

\*select name from v\$controlfile;--расположение

для измен: shutdown immediate -> скопир упр. файл -> измен парам CONTROL\_FILES в файле параметров -> startup open

Файл паролей – для аутентиф. админов БД. Можно созд, пересозд, измен.

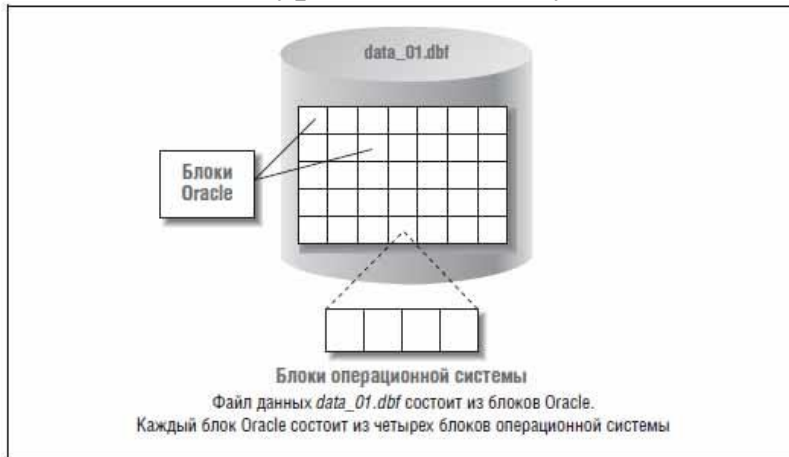
\*просмотр пользователей.\*select \* from v\$pwfile\_users;

Файл трассировки(сообщений, log.xml) – фиксирует все события, происх. на сервере БД, в т.ч. критических. По их содержанию м. выяснить причину сбоя

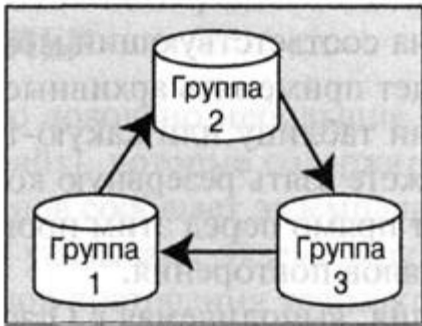


## 8. Файлы Oracle. Файлы данных, журналы, архивы.

Файлы данных (.dbf) – файлы, в которых хранятся все данные базы данных. В каждой базе данных такой файл хотя бы один. находятся собственно данные, хранящиеся в базе данных Oracle: таблицы и индексы, словарь данных, в котором сохраняется информация об этих структурах, и сегменты отката, необходимые для реализации конкурентного доступа.



Журналы повтора (redo, .log) – дисковые ресурсы, в которых фиксируются изменения, вносимые пользователем в базу данных. использ. циклически (сначала запись в 1й файл, 2, 3... 1);  
`select * from v$logfile;`



Мультиплексирование журналов повтора – поддержка неск. копий каждого журнала

`*select * from v$log;`

SCN (system change number) – сист. № измен.в БД (пред.селект (first\_change))

Параметры ЖП (указ.в CONTROLFILE):

`*maxlogfiles` – макс.кол-во групп ЖП

`*maxlogmembers` – макс.кол-во файлов в группе

`*maxinstances` – макс.кол-во инстансов

`*maxdatafiles` – макс.кол-во файлов д-х

Добавить новой группы:

`alter database add logfile group 5 'путь\redo5.log' size 50m blocksize 512;`

Добавление файла в группу:

`alter database add logfile member 'путь\redo5-1.log' to group 5`

Переключение журналов:

`alter system switch logfile;`

Архивы журналов повтора:

Режимы работы экземпляра:`archivelog; noarchivelog;`

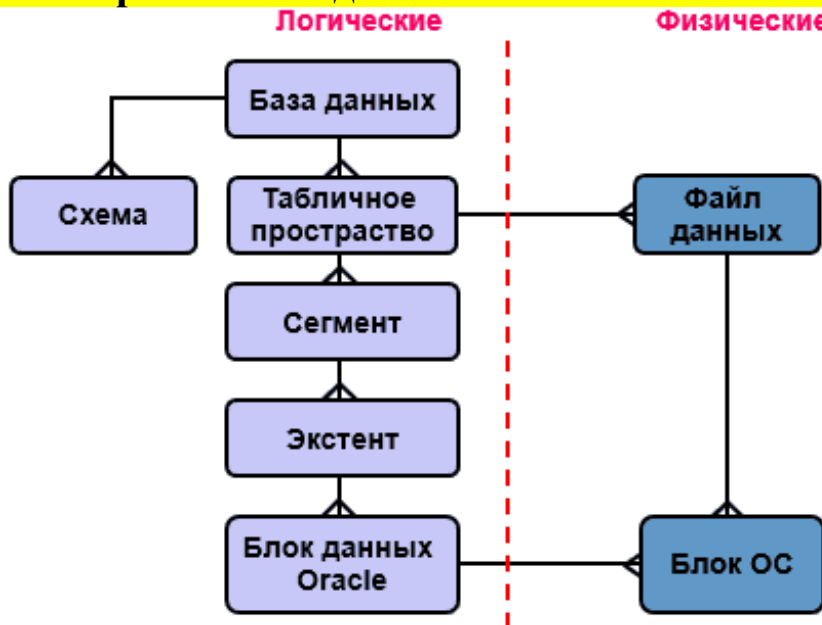
`*жизненно важны при восстановл.`

\*примен послед-но (если 1 пропущен – ост.не м.исп-ся)

Архивы:\*включить процесс архив-ния:alter database archivelog;

\*архивный журнал появл.после переключ.оперативн.журнала

## 9. Абстрактная модель Oracle. Логическая структура внешней памяти.



в Oracle исп. 2-уровневая организ. БД.

\* Логические структуры – это табличное пространство, сегмент, экстент, блок данных Oracle

\* Физические структуры – файл операционной системы, блок операционной системы

Лог. стр-ра сост. из польз. компонент: табл. пр-во, сегмент, экстент и блок д-х.

Табл. пр-во - логич. структура хранения д-х, контейнер сегментов. С ТАБЛИЧНЫМ ПРОСТРАНСТВОМ СВЯЗАНО МНОГО ФАЙЛОВ, С ОДНИМ ФАЙЛОМ – ТОЛЬКО ОДНО ТАБЛИЧНОЕ ПРОСТРАНСТВО

Сегмент – это набор экстенгов, крупная единица хранения. Сегменты данных и индексов – самые распространенные. Область на диске, выделяемая под объекты. В ОДНОМ ТАБЛИЧНОМ ПРОСТРАНСТВЕ МОЖЕТ БЫТЬ МНОГО СЕГМЕНТОВ! Сегмент хранит только данные, поэтому он создается только при добавлении данных (в примере строки в таблицу)

Экстент – непрерывный фрагмент дисковой памяти, 8 РЯДОМЛЕЖАЩИХ страниц, они не обязательно одного размера, ГЛАВНОЕ – РЯДОМ!

Блок – сервер пишет и читает блоками, в блоках всегда есть свободное место, скомпонован аккуратно. Наименьшая единица ввода-вывода

Сегмент сост. из экстенгов, экстент – из блоков. В одном ts мб неск сегментов. Сегмент если не секционирован, расп. в одном ts.

Сегмент:

хранит только д-е, поэтому он создается только при добавлении данных. При удалении строк из таблицы, сегмент не удал. При удалении таблицы измен. имя сегмента, и инф. об удалении запис. в словарь БД. Мб восстановлен с пом. механизма RECYCLEBIN.

\*сегмент д-х (хран. табл + их строки (1 табл – 1 сегмент))

\*сегмент индексов (хранят индексы (ключ. столбец + rowid))

\*сегмент отката (строятся с-мой и исп. при вып-нии транз.)

\*временный сегмент (врем. раб. обл. для промеж. стадий обраб. запроса)

Экстент:

длину выделяемого экстента вычисляет СУБД. Если при создании табл простр задана опция UNIFORM, то все экстенты имеют один. длину.

БД сост из ТС. ТС сост из файлов данных. ТС содержит сегменты.

- Сегменты сост. из экстентов. Сегмент привязан к ТС, но его данные могут находиться в разных файлах д-х, образующих это ТС.

- Экстент – набор расположенных рядом на диске блоков. Экстент целиком находится в одном ТС и в одном файле данных этого ТС.

- Блок – наим. единица управления пр-вом в БД. Блок – наим единица ввода-вывода, использ. сервером.

## 10. Абстрактная модель Oracle. Физическая структура внешней памяти.

Физически база данных организована как совокупность файлов, создаваемых обычными средствами операционной системы. Таким образом, основой физической структуры внешней памяти является файл операционной системы, состоящих из блоков операционной системы.

2 группы компонентов физического уровня:

- Системные объекты: файл параметров, управляющий файл, файл трассировки, журналы повтора, архивы
- Объекты пользователя – файлы данных

Физические структуры базы данных Oracle (Oracle Database) — это действительные файлы данных Oracle на уровне операционной системы. База данных Oracle состоит из следующих трех основных типов файлов.

- Файлы данных. Эти файлы хранят данные таблиц и индексов. Один или более файлов составляют логическую сущность – табличное пространство.
- Управляющие файлы. Эти файлы записывают изменения всех структур баз данных.
- Файлы журналов повторного выполнения. Эти онлайн-файлы содержат изменения, проведенные в табличных данных.

Когда экземпляр базы данных нуждается в чтении данных таблицы или индекса, он читает их из файлов данных на диске, если только нужные данные уже не кэшированы в памяти Oracle.

## 11. Абстрактная модель Oracle. Структура SGA.

SGA – System Global Area – область данных, которая нужна всем, это часть оперативной памяти, которую используют все процессы и которую разделяют все процессы одного экземпляра базы данных. SGA содержит всю необходимую информацию для операций 1 экземпляра. Совместно используется всеми СЕРВЕРНЫМИ И ФОНОВЫМИ процессами.

SGA (system global area):

- группа областей разделяемой памяти
- содержит данные и управляет инфой для одного экземпляра Oracle
- совместно всеми сервер. и фон. Процессами

SGA состоит из: 8 элементов

- Разделяемый пул SHARED POOL – кэш проверенных SQL-выражений, кэш словарей, данных, таблиц, представлений и триггеров, кэш словарей и библиотечный кэш. УПРАВЛЯЮЩИЕ СТРУКТУРЫ. Разделяемая SQL область и PL\SQL область, планы выполнения, операторы.

- Большой пул LARGE POOL – (для запросов) хранит большие объекты памяти. Здесь хранятся данные при резервном копировании. Не подчиняется алгоритму LRU

- JAVA POOL – позволяет работать Java-машине

- Фиксированная область SGA – НЕ ДОПУСКАЕТ ИЗМЕНЕНИЕ РАЗМЕРА! Хранит значения параметров, указатели на другие области памяти, ЗАГРУЗОЧНЫЙ БИНАРНЫЙ КОД. Размер зависит от ОС и платформы

- Буферный пул (кэш) – содержит образы блоков, считанных из файлов или созданных динамически, чтобы обеспечить согласованное чтение.

Буферные пулы - Кеер (постоянно хранит блоки данных), default (по умолчанию, если ничего не назначено), recycle (удаляет данные из кэша после использования). Служат для минимизации промахов при обращении к кэшу

Кэширование – это хранение поблизости копий любых объектов, которые возможно будут использоваться повторно, чтобы быстро получать к ним доступ

- Input/Output pool – пул ввода-вывода

- NULL POOL – свободное место, ничем не занято

- Буфер журналов повтора – redo log buffer, ускоряет работу сервера, работает циклически

Память различным пулам в SGA выдел. блоками – гранулами (наим. единица выделения памяти)

## 12. Абстрактная модель Oracle. Серверные процессы Oracle.

Процесс – механизм операционной системы Windows, осуществляющий запуск и выполнение приложений. Процесс создается, когда запускается приложение. В общем случае выполняется в собственной области памяти.

Процессы Oracle разделяются на серверные, фоновые и подчиненные.

Серверный процесс — это процесс, который обслуживает индивидуальный пользовательский процесс. Каждый пользователь, подключенный к базе данных, имеет свой отдельный серверный процесс, существующий на протяжении существования сеанса. Серверный процесс создается для обслуживания пользовательского процесса и используется этим пользовательским процессом для взаимодействия с сервером Oracle.

например, когда пользователь посылает запрос на выборку данных, серверный процесс, созданный для обслуживания пользовательского приложения, проверяет синтаксис кода и выполняет код SQL. Затем он читает данные из файлов в блоки памяти. (другой пользователь захочет прочесть те же данные, то пользовательский процесс прочтет их не с диска, а из памяти Oracle.) И, наконец, серверный процесс вернет запрошенные данные пользовательскому процессу.

Серверн.проц. м.:

- \* вып sql-операторы
- \*чит.файлы д-х
- \*возвр. результаты
- \*осуц.поиск в кэше

### 13. Абстрактная модель Oracle. Фоновые процессы Oracle.

Процесс – механизм операционной системы Windows, осуществляющий запуск и выполнение приложений. Процесс создается, когда запускается приложение. В общем случае выполняется в собственной области памяти.

Процессы Oracle разделяются на серверные, фоновые и подчиненные.

Фоновой процесс – позволяют большому числу пользователей параллельно и эффективно использовать информацию, хранимую в файлах данных.

Oracle создает эти процессы автоматически при запуске экземпляра

LREG – период.регистр.сервисов в проц.Listener;

DBWn (database writer) – запис. изменения в файлах данных, проверяет наличие грязных блоков

LGWR (log writer) – запис измен в журнал повтора (до фикс)

CKPT – выполняет проц.checkpoint- запись информации о контрольной точке в управляющий файл.кнтр точка- содержится в журнале, во время восстановления после неожиданного завершения работы или сбоя.

ARCn (archiver) – копир.файлы ЖП после переключ.группы журналов

PMON (process monitor) – отвеч.за очистку после ненорм.закрытия подклю.

RECO (recovery) – разреш.проблемн.связанных с распред.транзакциями.

### 14. Процесс-слушатель Oracle и его основные параметры.

Прослушиватель (Listener) – компонент, который позволяет устанавливать соединение между клиентом и базой данных. Один прослушивать может обслуживать неограниченное количество баз данных.

Принцип установки соединения с сервером Oracle по сети:

Клиент устанавливает начальное соединение со слушателем, слушатель принимает и проверяет запрос на подключение клиента и передает его обработчику служб базы данных.

1) Клиент выполняет запрос к Listener на соединение с сервисом экземпляра

2) Listener запрашивает соединение с сервером

3) Сервер возвращает параметры соединения обработчиком сервиса

4) Listener сообщает параметры соединения клиенту

5) Клиент соединяется с обработчиком запросов сервиса для дальнейшей работы в рамках соединения

Прослушиватель управляется файлом listener.ora.

Если неск.просл – каждый д.им.уник.имя.

Парам:

Протокол, порт, имя хоста.

Для упр-ния прослуш.исп.утилита lsnrctl; Для прослуш.доступны след.команды:Start – запуск проц.прослушивания;Stop – остановка;Status – тек.статус прослушивателя



## 15. Сетевые настройки Oracle. Установление соединения по сети.

Oracle поддержив. 2 режима соедин: dedicated(выделенный) и shared(разделяемый) server.

Режим выделенного сервера – (dedicated server) – каждому клиенту выделяется отдельный серверный процесс; работа клиента через диспетчера, или Dispatcher. Он получает запрос клиента, помещает в очередь к разделяемым серверам. Незанятый сервер обрабатывает запрос. Разделяемый сервер помещает результат обработки во входную очередь. Диспетчер извлекает результат и посылает клиенту.

Shared – в кач-ве обработчика выступает прога dispatcher, кот:

- 1) Получ запрос от К
- 2) Помещ.его во вх.очередь к раздел.С
- 3) Незанятый раздел.С извлек и обраб.запрос
- 4) После обраб.раздел.С помещ.рез-т обработки в вых.очередь
- 5) Из очереди рез-т извлек диспетчер
- 6) Диспетчер пересыл рез-т клиенту

За подключ отвеч 2 конфиг.файла:

Sqlnet.ora - обычный текстовый файл конфигурации, который содержит информацию о том, как и сервер Oracle, и клиент Oracle должны использовать возможности Oracle Net для сетевого доступа к базе данных.

Tnsnames.ora – файл конфигурации, определяющий адреса баз данных для установки соединения. Протоколы, порты, host

Виды подключений:

\*простое подключ (basic) – явно указ.все пар-ры соедин.

CONNECT имя/пароль@[//]хост[:порт][/имя\_службы]

\*локальное именованное (tns)

В tnsnames (host, port, service\_name) -> oracle net manager

\*LDAP-соед – с помощью службы каталогов.

\*local/bequeath- соединение со стандартным сервисом

## 16. Табличные пространства СУБД Oracle и их основные параметры.

Табличное пространство – единица хранения базы данных, эквивалент файловых групп MS SQL, логическая структура хранения и контейнер сегментов.

! м. указ для таблиц по умолч

! удаление табл.с опред.ТП:drop table xxx1 purge;

! вывести список ТП:select \* from dba\_tablespaces;

С одним табл простр связаны 1/неск файлов ОС, с каждым файлом ОС только 1 табл простр.

\* permanent– хранение постоян объектов бд,одно по умолч

\* temporary – хранение врем. д-х, приписано 1/неск юзерам, кот. могут там размещ. свои д-е

\* undo – хранение сегментов отката, исп. всегда один (указ. в SPFILE.ORA)

Параметры:

create [temp] tablespace ..

datafile (tempfile) путь...

size

auto\_extent on next

maxsize

-----

\* smallfile / bigfile (128 ТБ)

\* logging / nologging / force logging-(журналир измен)

\* online / offline-(раб/нераб сост)

\* reuse

## 17. Роли и привилегии СУБД Oracle и их основные параметры. Роль-поименованный набор привилегий, создается привилегированным администратором. Применяется для управления привилегиями пользователей.

Для создания роли используется команда `CREATE ROLE rolename [identified by 123];`

Привилегия – право выполнять конкретный тип предложений SQL или право доступ к объекту другого пользователя.

`GRANT privileges [ON object] TO role_name`

Системная привилегия – разрешение на выполнение определенных действий  
.Объектные – на изменение объекта

системные	объектные
на именование с-мы: create table / trigger	на изм объекта: select
WITH ADMIN OPTION – дают право юзеру также назнач/отбир привил: alter, analyze, audit, backup, create, drop, select (стараться редко исп)	WITH GRANT OPTION – дают право юзеру также назначать/отбир привил (но объектные): alter, delete, execute, insert, update, select, references  снимает привил тот, кто их назначал
объекты грантов: database, user, profile, tablespace, role, table, index, trigger, procedure, sequence, view	объекты грантов: table, view, sequence, procedure

PROCEDURE

PROFILE

ROLE

ROLLBACK SEGMENT

SESSION

SEQUENCE

SYSTEM

TABLE

TABLESPACE

TRIGGER

USER

VIEW

Системные:

Объектные:

ALTER

DELETE

EXECUTE

INDEX

INSERT

REFERENCES

SELECT

UPDATE

## 18. Пользователь СУБД Oracle и его основные параметры.

Юзер – физ./юр. лицо, кот. имеет доступ к БД и пользуется услугами для получ. инфы

```
create user .. identified by 12345
  default tablespace .. quota unlimited on ..
  temporary tablespace ..
  profile ..
  account unlock
  password expire
```

Чтобы пользователь мог подключиться к базе и начать обмен данными с ней, ему нужно предоставить системные полномочия CREATE SESSION, или назначить ему роль, для которой уже выданы соответствующие привилегии.

Если не желаете, чтобы пользователь вообще создавал какие-либо объекты в базе данных, не предоставляйте ему квоту ни в одном из табличных пространств

```
ALTER USER xxxx QUOTA 100M ON users;
```

Как только новому пользователю предоставлена квота в табличном пространстве, он сможет создавать объекты базы данных, такие как таблицы и индексы. По умолчанию любые объекты, созданные пользователем, будут помещаться в заданное по умолчанию постоянное табличное пространство пользователя. Однако пользователь может создавать объекты в любом табличном пространстве, если только обладает в нем квотой дискового пространства.

## 19. Профиль безопасности СУБД Oracle и его основные параметры.

Профиль – коллекция атрибутов, связанных с использованием ресурсов и паролей, которая может быть назначена пользователю. Это наложение индивидуальных ограничений пользователю на использование ресурсов Oracle. Позволяет ограничить потребление ресурсов и количество сессий

```
create profile .. limit
  password_life_time 180          -- дней жизни пароля
  sessions_per_user 3             -- кол. сессий для юзера
  failed_login_attempts 7         -- кол. попыток входа
  password_lock_time 1            -- дней блока
  password_reuse_time 10          -- через ск. дней повтор пароль
  password_grace_time default     -- кол-во дней предупр. о смене пароля
  connect_time 180                -- минут соедин
  idle_time 30                    -- минут простоя
```

## 20. Язык SQL. Основные операторы и их назначение.

SQL – язык структурированных запросов, с помощью него пишутся специальные запросы (SQL инструкции) к базе данных с целью получения этих данных из базы и для манипулирования этими данными. Декларативный язык программирования (ОПИСЫВАЕТСЯ ОЖИДАЕМЫЙ РЕЗУЛЬТАТ, А НЕ СПОСОБ ЕГО ПОЛУЧЕНИЯ).

Реляционная база данных хранит информацию в табличной форме со строками и столбцами, представляющими различные атрибуты данных и различные связи между значениями данных. Инструкции SQL можно использовать для хранения, обновления, удаления, поиска и извлечения информации из базы данных.

\* Data Definition Language (DDL)- это группа операторов определения данных. Другими словами, с помощью операторов, входящих в эту группы, мы определяем структуру базы данных и работаем с объектами этой базы, т.е. создаем, изменяем и удаляем их. CREATE, ALTER, DROP

Data Manipulation Language (DML) – это группа операторов для манипуляции данными. С помощью этих операторов мы можем добавлять, изменять, удалять и выгружать данные из базы, т.е. манипулировать ими. SELECT, INSERT, UPDATE, DELETE

Data Control Language (DCL) – группа операторов определения доступа к данным. Иными словами, это операторы для управления разрешениями, с помощью них мы можем разрешать или запрещать выполнение определенных операций над объектами базы данных. GRANT, REVOKE

Transaction Control Language (TCL) – группа операторов для управления транзакциями. Транзакция – это команда или блок команд (инструкций), которые успешно завершаются как единое целое, при этом в базе данных все внесенные изменения фиксируются на постоянной основе или отменяются, т.е. все изменения, внесенные любой командой, входящей в транзакцию, будут отменены. COMMIT, ROLLBACK, SAVEPOINT

## 21. Таблица и ее основные параметры.

Таблица – основная структура сохранения информации в БД

Типы таблиц:

- Традиционные таблицы -хранятся без определенного порядка по принципу КУЧИ, при добавлении данных – они идут в первое свободное место в сегменте
- Индекс-таблицы на основе B-tree сбалансированного дерева поиска упорядоченных ключей – индексная структура
- Кластеризованные индекс-таблицы (index clustered table)
- Кластеризованные хэш-таблицы (hash clustered table)
- Отсортированные кластеризованные хэш-таблицы (sorted hash clustered table)
- Вложенные таблицы (nested table) – генерируемые и поддерживаемые системой дочерние таблицы

Временные таблицы (temporary table) – временные данные на время транзакции или сеанса при необходимости выделяются экстенды временного табличного пространства. название временной таблицы начинается со знака решетки # или ##, # - локальная (на текущую сессию), # - глобальная (на все открытые сессии),

Объектные таблицы – создаются на основе объектного типа, строки такой таблицы могут иметь объектный тип

Внешние таблицы – используют наборы данных из внешних файлов операционной системы (external tables)

Секционированные таблицы - позволяет делить большие объемы данных на подтаблицы, именуемые разделами (partition), согласно различным критериям. Секционирование особенно полезно в среде хранилища данных

Параметры:

create table (...)

organized						index
nomonitoring	(сервер	смотрит	как	заполн		таблица)
logging	(все	изменения	таблицы	запис	в	log)
pctthreshold 20	(% заполненности в блоке ?)					

Параметры:

PCTFREE - % памяти блока, резерв. для возможных обновлений строк, уже содержащихся в блоке

PCTUSED - % занятой части памяти блока

- может иметь до 1000 столбцов (<254)
- может иметь неогр. число строк + индексов
- нет огранич на число таблиц

## 22. Временные таблицы.

табл., кот. хранят д-е на время сессии или текущей транзакции для нее выдел. временный сегмент в TEMPORARY-тс, и для каждого юзера свой.

Временные таблицы создаются и существуют, пока их не удалят к. юзер видит только свои д-е (свой набор), раб, как с обычной таблицей

Отличие от обычных: нет flashback

привил: create table

статичны: create единожды и сущ, пока их не drop !!, в начале сеанса всегда пуста

бывают:

- \* on commit preserve rows – данные сущ. на t сеанса, возм все DML, TCL

- \* on commit delete rows – данные сущ. на t транзакции, возм все DML, после вып. commit / rollback таблица стан. пустой

- м. создавать триггеры, констрейны (огранич), индексы

- не мб индексно-организованными, нельзя секционир, размещ в кластере

## 23. Ограничения целостности в таблицах.

1) primary key – первичный ключ, уникален и не null, однозначно идентифицирует запись в таблице

2) not null- запрет на неопределенные значения. Указывает, что столбец не может содержать неопределенных значений

3) check – обесп соотв столбца опр. парам : n number CHECK (n<1)

4) unique – гарант уникальность, мб на комбинации : CONSTRAINT n UNIQUE (n, name)

5) ссылочные (FK, foreign key) – внешний ключ, это ограничение указывает, что столбец или совокупность столбцов принимают значения, которые должны совпадать со значениями столбца (совокупности столбцов) из другого связанной таблицы.

6) default – значение по умолчанию для столбца

7) data type – предотвращение появления в столбце значений, не соответствующих заданному типу данных

Целостность базы данных (англ. database integrity) — соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам.



## 24. Типы данных базы данных.

- \* char / nchar [2]-количество символов фиксированной длины
- \* varchar2 / nvarchar2 [4] – кол строк переменной длины
- \* date-тип дата время
- \* interval day to second [11] / interval year to month timestamp [5] – показ точную разницу между двумя датами
- \* timestamp with time zone [13]/ timestamp with local time [7-11]– очень точная дата и время (до мкс), когда надо точная разница (ставки)
- \* number (A,B) – для числ значений
- \* long raw / long / raw – уже не исп
- \* blob / clob / nclob – b большие 2ич объ: изображ (хран в виде строки в отличии от bfile), c – биг тексты, n – большие д-е в юникоде
- \* rowid / urowid – id строки в таблице
- \* bfile – ук-ль на двоич файл ОС (биг файл, кот не надо хранить в БД)

## 25. Индексы базы данных. Виды и особенности применения индексов.

Индекс – структура базы данных, используемая сервером для быстрого поиска строки в таблице, сохраняя отсортированные значения указанных столбцов. Таким образом правильное использование индексов сокращает до минимума количество дорогостоящих операций ввода-вывода.

create index <index\_name> on <table\_name> ( <column1>, <column2>, ... );

Индексы могут относиться к нескольким типам, наиболее важные из которых перечислены ниже:

- Уникальные и неуникальные индексы. Они основаны на уникальном столбце – обычно вроде номера карточки социального страхования сотрудника. Когда накладывается ограничение уникальности на столбец таблицы, Oracle автоматически создает уникальные индексы по этим столбцам.

- Первичные и вторичные индексы. Первичные индексы – это уникальные индексы в таблице, которые всегда должны иметь какое-то значение и не могут быть равны null. Вторичные индексы – это прочие индексы таблицы, которые могут и не быть уникальными.

- Составные индексы – индексы, содержащие два или более столбца из одной и той же таблицы. Они также известны как сцепленные индексы (concatenated index).

- \* табличный (B\*Tree) --чаще всего (в виде сбаланс дерева)
- \* битовый -- созд бит. карта для к. возможного значения столбца, к. биту соотв. строка, а знач бита (1/0) означ. содержит строка индексп. значение или нет
- \* функциональный --индекс ч/функции (Blinova-blinova: LOWER(name) / UPPER(name))

- \* кластерный

Плотность запроса – кол-во возв-мых строк запроса

Селективность запроса – % ключей от общего кол-ва (идеал <10%)



## 26. Последовательность СУБД Oracle и ее параметры.

Последовательность – объект базы данных, предназначенный для генерации числовой последовательности. Этот объект Oracle, который используется для генерации последовательности чисел. Привилегия grant create sequence

Привилегия

create				sequence
increment		by	10-	приращение
start	with	100-первое	генерируемое	значение
nomaxvalue				
nominvalue				
nocycle / cycle	позволяет последовательности повторно использовать созданные значения по достижению maxvalue или minvalue			
cache 20 / nocache	сколько элементов последовательности Oracle распределяет заранее и поддерживает в памяти для быстрого доступа			
order / noorder	номера последовательности генерируются в порядке запросов			
select ИМЯ.CURRVAL/ NEXTVAL from dual;– получ след/тек значение				

## 27. Кластер и его параметры

Таблицы, с к-ми часто работают совместно, можно физически хранить совместно => созд кластер, кот. их содержит => запросы вып быстрее, т.к. строки из отд. таблиц сохран в одном блоке, меньше операций ввода-вывода, произв. оп-ций вставки, обновл, удаления ниже чем для обычных таблиц

связанные столбцы – кластерный ключ

Кластер – объект БД, кот. хранит значения общих столбцов неск. Таблиц. Он позволяет быстрее доставать данные из связанных таблиц – он знает, что таблицы связаны, и тогда он часть данных хранит в себе, эти данные извлекаются быстрее.

Пример:

create cluster xxxx ( id number ) hashkeys ____	create table T1 ( ..., id number) cluster xxxx (id)	create table T2 (id number) cluster xxxx(id)
---	---	--

к прим Если 2 таблицы имеют идентичный столбец, и нам часто приходится соединять таблицы по нему, то тогда выгодно хранить значения общих столбцов в одном и том же блоке данных –кластере.Хэш-кластеры: исп. функции хэширования кластерного ключа строки для определения физ. локализации места, где строку следует хранить. Можно создавать хэш-кластер и поместить в него таблицы. Строки извлекаются в соответствии с результатом некоторой хэш-функции. Параметр hashkeys показывает максимальное количество уникальных хэш-значений, которые могут быть сгенерированы хэш-функцией

## 28. Представление и его параметры.

Представление – именованный select-запрос. Обращение происходит как к обычной таблице. Добавляют уровень защиты. Скрывают сложность данных, имена столбцов таблиц.

```
create view xxx as  
select * from orders;
```

Параметры:

- \* FORCE – созд предст, неважно сущ. ли таблицы и есть ли права
- \* NOFORCE – по умолч
- \* WITH CHECK OPTION – нельзя update/insert, кот. не входят в это предст
- \* READ ONLY

## 29. Материализованные представления СУБД Oracle.

Обычное Представление не хранит данные, а только выполняет описанный в себе запрос. Это значит, что тратится некоторое время на получения ответа от самого Представления и его Запроса, так как данных в готовом виде нет.

Материализованное Представление - это готовая таблица с данными, которые были сформированы в момент создания самого Материализованного Представления. недостаток – не гарантируется, что мы видим самые свежие данные => НАДО ОБНОВЛЯТЬСЯ САМИМ

привил: create materialized view

► BUILD IMMEDIATE – создает представление в момент выполнения оператора

► START WITH – показывает, когда выполнится в первый раз (если не был построен сразу)

► NEXT – показывает, когда выполнится в следующий раз

►

## 30. Частные и публичные синонимы СУБД Oracle.

Синоним – способ обращаться к объекту БД без указ. обяз. полной идентификации объекта (хост – экземпляр – владелец – объект)

\* частный принадлежит юзеру, кот. его создал

\* публичный исп. совместно всеми юзерами БД

привил: create (public) synonym

словарь: dba.synonyms

м. указ. на: табл, проц, фун, послед, предст, пакеты, объекты в лок/уд БД

пример: create synonym T1 for svvcore.teacher;

Синонимы скрывают идентичность лежащих в их основе объектов и могут быть как приватными (private), так и общедоступными (public).

Общедоступные синонимы доступны всем пользователям базы данных, а приватные синонимы являются составной частью схемы отдельного пользователя, и другим пользователям базы следует выдавать права доступа для использования приватных синонимов.

### 31. Основные характеристики языка PL/SQL.

Procedure Language extensions to SQL-Основной язык для программирования хранимых процедур (stored procedure). Интегрирован с БД Oracle(быстрое переключ между движками PL/SQL и SQL)

Нету накладных расходов на приведение типов

Может вып независимо от юзера (вып от имени бизнес-юзера ? )

PL/SQL-функции можно вызвать из SELECT запросов

Это встроенный (внутренний) язык, работает только в конкретной управляющей среде – SQL Developer, SQL Plus. «Написать один раз и использовать везде»

Программа на PL/SQL состоит из блоков (анонимных или поименованных). Блок может содержать вложенные блоки, называемые иногда подблоками. Общая форма PL/SQL-блока:

DECLARE-- Описания блока, переменные, типы, курсоры и т. п. (опционально)

BEGIN-- Непосредственно код программы

EXCEPTION-- Обработка исключений (опционально)

END;-- Однострочный комментарий

Язык PL/SQL позволяет определять следующие типы именованных блоков:процедуры;функции;объекты;пакеты.

Все именованные блоки кода, кроме пакетов, не хранят внутреннее состояние от вызова к вызову.

Пакеты-позволяя сгруппировать наборы именованных блоков кода, в пакетах возможно хранение состояния на время жизни сессии базы данных, доступное для функций и процедур, входящих в пакет.

Пакеты в PL/SQL содержат спецификацию и тело.

Спецификация пакета может содержать определение констант, переменных, типов данных, объявление процедур и функций.

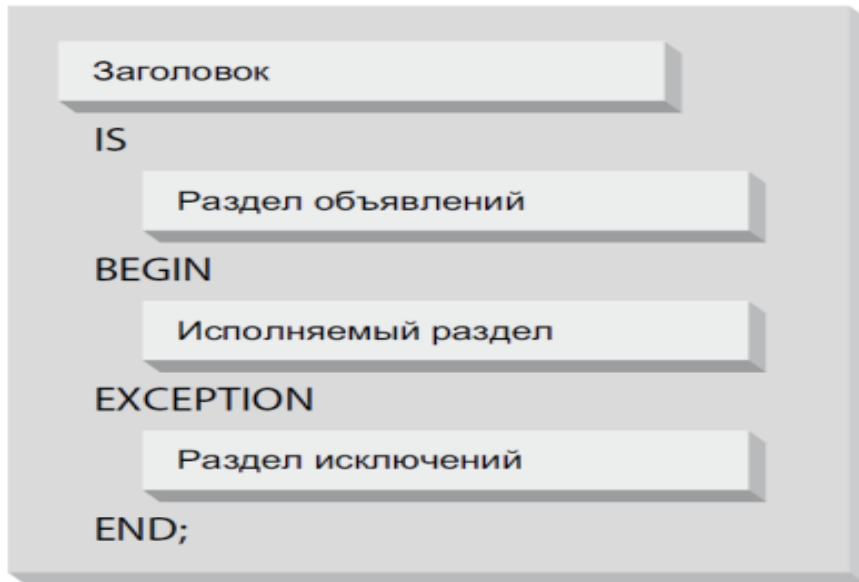
Тело пакета определяет объявленные в спецификации процедуры и функции, а также может содержать блок кода инициализации пакета, определения внутренних констант, переменных, типов данных, процедур и функций. Все компоненты пакета, объявленные в его спецификации, могут быть доступны для использования извне пакета, в то время как тело пакета инкапсулирует реализацию этих компонентов, и извне недоступно.



## 32. Структура программы языка PL/SQL. Анонимные и именованные блоки.

Программа на PL/SQL состоит из блоков (анонимных или поименованных). Блок может содержать вложенные блоки, называемые иногда подблоками. Общая форма PL/SQL-блока - анонимного блока:

DECLARE-- Описания блока, переменные, типы, курсоры и т. п. (опционально)  
BEGIN-- Непосредственно код программы  
EXCEPTION-- Обработка исключений (опционально)  
END;-- Однострочный комментарий



Анонимный блок – блок без имени. Другие блоки не могут его вызвать – нет идентификатора обращения:

- не имеет строки заголовка
  - используется как скрипт
  - не вызываем из другого блока
  - начинается обычно с declare/begin
  - варианты использования: триггеры, скрипты
- ```
begin
    null; --простейший
end;
```

Именованные блоки – процедуры и функции

Процедура – именованный модуль, который выполняет одно или несколько выражений и может принимать или возвращать значения через список параметров.

Функция – именованный модуль, который выполняет ноль или более выражений через фразу Return.

Исключения возникают, когда механизм PL/SQL встречает инструкцию, которую он не может выполнить из-за ошибки

when others – обработка любого исключения

sqlerrm – функция, кот. возвр сообщ об ошибке

sqlcode – функция, кот. возвр № ошибки (обычно 5-значный)

блоков when ск. угодно, самый последний – when others  
В PL/SQL каждый блок может быть вложен в другой.

```
begin  
    begin
```

```
        ...
```

```
    end;
```

```
end;
```

Переменные и исключения локальны в рамках блока. Переменные, объявленные в родительском блоке, видны во вложенном.

### 33. Типы данных, основные операции, константы языка PL/SQL.

Язык PL/SQL поддерживает следующие категории типов:

- встроенные типы данных, включая коллекции и записи;

- скалярные;
- составные;
- ссылочные;
- LOB-типы;

- объектные типы данных.

Character (символьные)

- CHAR фиксированный размер строки
- VARCHAR2 переменный размер строки
- VARCHAR переменный размер строки, аналог varchar2
- NCHAR фиксированный размер строки unicode
- NVARCHAR2 переменный размер строки Unicode

Numeric (числовые)

- Number (p, s)
- Binary\_double, binary\_float
- PLS\_INTEGER
- Natural

Логические

- boolean

Константы: с пом constant

## 34. Поддержка национальных языков в СУБД Oracle. Наборы символов.

### Байтовая и символьная семантика символов.

NLS – National Language Support

\* м. хранить д-е мн-ва нац. языков, используя Unicode или спец. кодировки (character set)

\* символы хранятся как коды символов, завис. от выбранного набора

\* в одной БД мб 2 набора: основной и доп.

\* устан при созд БД

\* измен: alter database (national) character set

\* кроме символов алфавита в набор вх. знаки преп, числ, \$...

Основной набор символов используется для:

-хранения символьных типов char, varchar2, clob и long

-описания имен объектов, переменных

-Ввода и хранения PL/SQL модулей

Дополнительный набор символов используется для:

-хранения символьных типов nchar, nvarchar2, nclob

Кроме символов алфавита в набор включаются знаки препинания, числа, символы денежных единиц и пр.

Поддержка нац яз:

пер-ная окружения NLS\_LANG = lang\_territory.charset

lang – имена месяцев, дней, направл текста : по умолч american

territory – настр календаря, формат даты, денег

charset – отобр символов, заглавных букв,

словари: nls\_ [session | instance | database] \_parameters

Семантика символов:

\* байтовая – рассм строки как посл-сть байтов (по умолч)

\* символьная – рассм строки как посл-сть символов

задается nls\_length\_semantics



### 35. Связ.объявления пер-ных: инструкция %TYPE, инструкция %ROWTYPE.

Тип переменной основан на известной структуре данных, если мы, например, не знаем заранее ее тип данных.

Скалярная ссылка %TYPE для определения переменной на основе другой переменной или поля в таблице

Ссылка на запись %ROWTYPE для определения структуры записи на основе таблицы или курсора

Атрибут %TYPE позволяет объявить константу, переменную, поле записи или параметр подпрограммы так, чтобы они имели тот же тип данных, что и ранее объявленная переменная или столбец (не зная, что это за тип)

### 36. Локальные процедуры и функции языка PL/SQL.

Лок.прогр. модуль – это процедура или функция PL/SQL, определяемая в разделе объявлений блока PL/SQL . Локальным такой модуль называется из-за того, что он определяется только внутри родительского блока PL/SQL и не может быть вызван из другого блока, определенного вне родительского.

Лок.процедуры и ф-ции:

\*их объявление дб в конце секции декларир.после всех типов, записей, курсоров, пер-ных и исключений

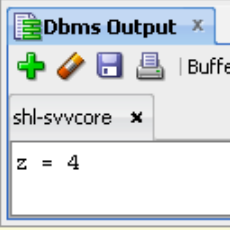
\*мб исп-ны только в рамках блока, в кот.они объявл

\*мб перегружены (разн.число пар, др.тип прогр.модуля, др.семейство пар-ров)

Лок.процедура:

```
declare
  x number(3) := 4;
  y number(3) := 5;
  z number(3);
  procedure summod5 (x1 number, x2 number, x3 out number)
  is
    z number(3) := 5;
    begin
      x3 := mod(x1+ x2,z);
    end summod5;
begin
  summod5(x,y,z);
  dbms_output.put_line('z = '||z);

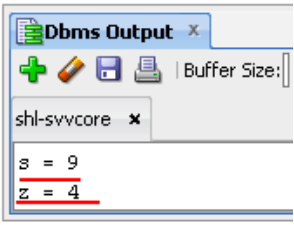
exception
  when others then dbms_output.put_line(sqlerrm);
end;
```



Лок.функция:

```
declare
  x number(3) := 4;
  y number(3) := 5;
  z number(3);
  s number(5);
  function summod5 (x1 number, x2 number, x3 out number)
  return number is
    z number(3) := 5;
    begin
      x3 := mod(x1+ x2,z);
      return (x1+x2);
    end summod5;
begin
  s := summod5(x,y,z);
  dbms_output.put_line('s = '||s);
  dbms_output.put_line('z = '||z);

exception
  when others then dbms_output.put_line(sqlerrm);
end;
```



### 37. Использование записей в PL/SQL. Вложенные записи.

Запись (record) – структура данных, составленная из нескольких частей информации, называемых полями

Для объявления записи вначале надо определить как тип, а потом объявить переменную типа «запись»

Типы записей:

- \*табличные
- \*курсорные
- \*программно-определенные

Объявление записей:

\*на осн.таблицы (%rowtype)

```
declare
    one_book books%rowtype;
```

\*на осн.курсора (cursor + %rowtype)

```
declat cursor myc is
    select * from books where author like '%Fadeev%';
one_curs mysc%rowtype;
```

\*запись опр-мая прогером:

```
declare
    rec1 teacher%rowtype;
    type person is record
    (
        code teacher.teacher%type,
        name teacher.teacher_name%type
    );
    rec2 person;
begin
    select * into rec1 from teacher where teacher = 'УРЕ';
    select teacher, teacher_name into rec2 from teacher where teacher = 'ДДК';
```

Вложенные записи – одно из полей внешней записи в действительности явл.полем другой записи.

```
declare
    rec1 teacher%rowtype;
    type address is record
    (
        address1 varchar2(100),
        address2 varchar2(100),
        address3 varchar2(100)
    );
    type person is record
    (
        code teacher.teacher%type,
        name teacher.teacher_name%type,
        homeaddress address
    );
    rec2 person;
begin
    rec2.code := 'ПРХК';
    rec2.name := 'Переходько Олеся';
    rec2.homeaddress.address1 := 'Беларусь';
    rec2.homeaddress.address2 := 'Пинск, Бресткая обл.';
    rec2.homeaddress.address3 := 'Набережная 7, кв.77';
```

### 38. Операторы управления, операторы цикла языка PL/SQL.

Оператор IF

- 1) if ... then ... end if;
- 2) if ... then ... else ... end if;
- 3) if ... then ... elsif ... then ... (elsif ... then ...) ... else ... end if;

Оператор CASE

```
case x
  when ... then ...;
  when ... then ...;
  when x between 13 and 20 then ....;
  else ....;
end case;
```

Отличие: в 1 выбир.знач и сравниваем его с чем-то. Во 2 – проверяем условия.

Циклы loop, for, while:

Цикл loop - Простой цикл удобно использовать, когда нужно гарантировать хотя бы однократное выполнение тела цикла. Так как цикл не имеет условия, которое бы определяло, должен ли он выполняться или нет, тело цикла всегда будет выполнено хотя бы один раз.

```
loop
  исполняемые_команды
end loop;
```

Цикл while - выполняется до тех пор, пока определенное в цикле условие остается равным true. А поскольку возможность выполнения цикла зависит от условия и не ограничивается фиксированным количеством повторений, он используется именно в тех случаях, когда количество повторений цикла заранее не известно.

WHILE условие

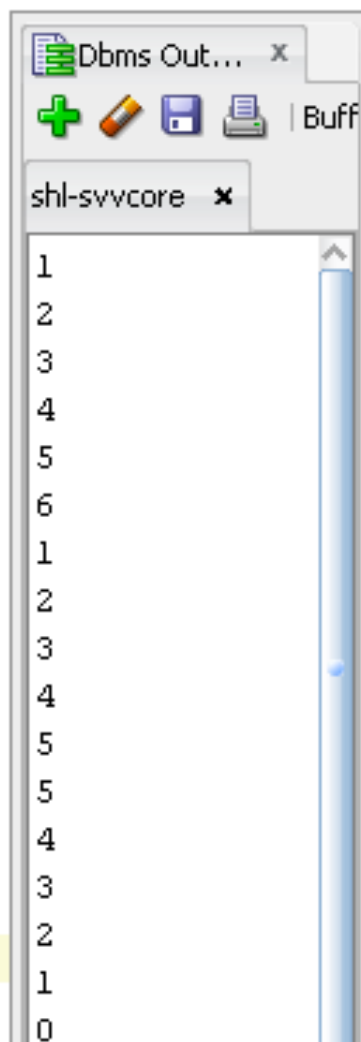
```
LOOP
  исполняемые_команды
END LOOP;
```

Цикл for – цикл со счетчиком. Количество итераций этого цикла известно еще до его начала. Использовать, если тело цикла должно быть выполнено определенное количество раз, а выполнение не должно прерываться преждевременно.

FOR счетчик IN [REVERSE] начальное\_значение .. конечное\_значение

```
LOOP
  исполняемые_команды
END LOOP;
```

```
declare
    x pls_integer := 0;
begin
    -----
    loop
        x := x + 1;
        dbms_output.put_line(x);
    exit when x > 5;
    end loop;
    -----
    for k in 1..5
    loop
        dbms_output.put_line(k);
    end loop;
    -----
    while (x > 0)
    loop
        x := x - 1;
        dbms_output.put_line(x);
    end loop;
    -----
end;
```



### 39. Курсоры. Виды курсоров. Схемы обработки курсора.

Курсор – объект БД, позв работать с записями построчно. средство извлечения данных из базы данных Oracle.

Курсор Oracle – указатель на область в PGA, в которой хранится: строки запроса, число строк, указатель на разобранный запрос в общем пуле.

Открытие курсора – создание контекстной области PGA – создается моментальный снимок (snapshot) данных запроса.

Виды:

► - явный (объяв разработчиком), неявный (не требует объявления. Операторы INSERT, UPDATE, DELETE, MERGE, SELECT INTO)

В неявн курсор можно будет открывать и извлекать данные в одной или нескольких программах, причем возможности контроля будут шире, чем при использовании неявных курсоров.

- статич (выражение опр при компил), динам (выраж опр при выполнении)

Операторы управл курсором:

- declare – объяв явного

- open – откр курсор, создавая новый рез набор на базе указ. запроса

- fetch – послед извлеч строк из рез набора от начала до конца

- close – закр курсор и освоб его ресурсы

Ошибки неявного курсора:

- no\_data\_found – не возвр строк вообще

- too\_many\_rows – более 1 строки

select into чтобы вернуть ровно 1 строку – точную выборку!

#### 40. Курсоры. Атрибуты курсора. Курсоры с параметрами.

**Курсор** — объект бд, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных oracle. позволяет проходить по результатам select-запроса запись за записью. Это указатель на область rga, в которой хранятся строки запроса, число строк, указатель на разобранный запрос в общем пуле

Атрибут курсора – хранит информацию о состоянии курсора и результате его работы

##### **Атрибуты явного курсора:**

Переменная\_курсора% isopen – true открыт, false закрыт

Переменная\_курсора% found – true, если строки были встав/уд/выбраны, false не выбрана

Переменная\_курсора% notfound – наоборот, т.е true, если строки не были выбраны/вставлены/удалены, false, если были

Переменная\_курсора% rowcount – количество строк, выбранных в курсоре

##### **Атрибуты НЕявного курсора:**

sql%found – null перед выполнением, true если были манипуляции  
sql%isopen ВСЕГДА FALSE ДЛЯ НЕЯВНЫХ – он сразу открывается и закрывается

sql%notfound – null перед выполнением, false если были манипуляции

sql%rowcount – количество строк, выбранных в курсоре

##### **Курсоры с параметрами:**

При объявлении курсора в скобках указывается имя параметра  
declare

cursor curs\_auditorium (capacity number) is select \* from auditorium where  
auditorium\_capacity > capacity (capacity – ПАРАМЕТР)

#### 41. Курсорные переменные. Параметры инстанса, связанные с курсорами.

**Курсор** — объект бд, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных oracle. позволяет проходить по результатам select-запроса запись за записью. Это указатель на область rga, в которой хранятся строки запроса, число строк, указатель на разобранный запрос в общем пуле.

**Курсорная переменная** – структура данных, которая указывает на курсорный объект. Используется:

- для передачи курсора в кач параметра
- чтобы отложить связь курсора с SELECT-запросом до выполнения OPEN

declare

**type имя\_курсорного\_типа is ref cursor;**

begin

**open имя\_курсорного\_типа for .... – и работаем как с обычным курсором**  
end;

Курсорная переменная ссылается на курсор. В отличие от явного курсора, имя которого в PL/SQL используется как идентификатор рабочей области результирующего набора строк, курсорная переменная содержит ссылку на эту рабочую область. С помощью же курсорной переменной можно выполнить любой запрос и даже несколько разных запросов в одной программе.

- ▶ **Курсорная переменная, объявленная с помощью REF CURSOR без указания RETURN может быть связана с любым запросом**
- ▶ **Курсорная переменная, объявленная с помощью REF CURSOR с указанием RETURN может быть связана только с запросом, который возвращает результат точно соответствующий числу и типам данных в записи после фразы RETURN во время выполнения**

Если курс. переменная объявлена с пом. REF CURSOR без **RETURN** – она может быть связана с любым запросом, иначе – только с запросом, возвращающий результат точно соответствующий числу и ТД в записи после фразы return

## 42. Курсорные подзапросы.

**Курсор** — объект бд, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных oracle. позволяет проходить по результатам select-запроса запись за записью. Это указатель на область pga, в которой хранятся строки запроса, число строк, указатель на разобранный запрос в общем пуле.

Курсорное выражение (подзапрос) – это выражение со специальным оператором cursor, используется в SQL-запросе и определяет вложенный курсор. ТАКИЕ НАБОРЫ ОБРАБАТЫВАЮТСЯ ВО ВЛОЖЕННЫХ ЦИКЛАХ! Главный цикл выбирает строки основного курсора, вложенный – строки вложенного курсора. Мы объявляем курсор в рамках другого курсора.

Declare

**cursor curs\_aut**

**is select auditorium\_type,**

**cursor**     (select     auditorium     from     auditorium     aum     where

aut.auditorium type=aum.auditorium type)

**from auditorium\_type aut;**

**curs\_aum sys\_refcursor;**

begin

open curs\_aut;

.....



### 43. Использование конструкции CURRENT OF в курсорах.

**Курсор** — объект бд, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных oracle. позволяет проходить по результатам select-запроса запись за записью. Это указатель на область rga, в которой хранятся строки запроса, число строк, указатель на разобранный запрос в общем пуле.

Оператор **WHERE CURRENT OF** позволяет обновить или удалить запись, которая была в курсоре последней.

Иногда, при выборке из курсора бывает ситуация, что какой-либо столбец или строки результирующего набора необходимо обновить. То есть, изменить их содержимое. Для того, чтобы это осуществить, непосредственно при объявлении курсора необходимо использовать конструкцию - FOR UPDATE (для обновления ..). А, так же конструкцию, **WHERE CURRENT OF** (где текущая строка ..) в операторах **UPDATE, DELETE**. Собственно конструкция FOR UPDATE, является частью оператора SELECT и объявляется последней

update **табл** set ... where current of **имя курсора**

delete from **табл** where current of **имя курсора**

## WHERE CURRENT OF

```
-- 12/50.sql
declare
  cursor curs_auditorium (capacity svvcore.auditorium.auditorium%type)
  is select auditorium, auditorium_capacity
  from auditorium
  where auditorium_capacity >= capacity for update;
  aum svvcore.auditorium.auditorium%type;
  cty svvcore.auditorium.auditorium_capacity%type;
begin
  open curs_auditorium(80);
  fetch curs_auditorium into aum, cty;
  while (curs_auditorium%found)
  loop
    if cty >= 90
    then
      cty := cty * 1.1;
      update auditorium
      set auditorium_capacity = cty
      where current of curs_auditorium;
    end if;
    dbms_output.put_line(' ||aum|| ' ||cty);
    fetch curs_auditorium into aum, cty;
  end loop;
  close curs_auditorium;
  rollback;
exception
  when others
  then dbms_output.put_line(sqlerrm);
end;
```

| shl-svvcore x |     |
|---------------|-----|
| 236-1         | 80  |
| 313-1         | 80  |
| 408-2         | 80  |
| 103-4         | 80  |
| 105-4         | 80  |
| 107-4         | 80  |
| 110-4         | 80  |
| 111-4         | 80  |
| 114-4         | 110 |
| 132-4         | 80  |
| 025-4         | 80  |
| 229-4         | 80  |
| 314-4         | 80  |
| 320-4         | 80  |
| 429-4         | 80  |
| ?             | 80  |
| 301-4         | 99  |

Если вы планируете обновлять или удалять записи, на которые ссылается оператор SELECT FOR UPDATE, вы можете использовать оператор WHERE CURRENT OF

Оператор **WHERE CURRENT OF** позволяет обновить или удалить запись, которая была последней выбранной курсором.

#### 44. Динамические курсоры.

**Курсор** — объект бд, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных oracle. позволяет проходить по результатам select-запроса запись за записью. Это указатель на область pga, в которой хранятся строки запроса, число строк, указатель на разобранный запрос в общем пуле.

Открытие курсора – создание контекстной области PGA – создается моментальный снимок (snapshot) данных запроса.

**execute immediate** – одностроч запросы и ddl-команды

**open for, fetch, close** – динам многострочные запросы

– для улучшения производительности выполнения sql-выражений можно использовать динам. курсоры со связанными переменными

– позволяют серверу Oracle повторно использовать разобранные SQL-выражения из разделяемого пула

```
EXECUTE IMMEDIATE 'INSERT INTO
```

```
dept (deptno, dname, loc) VALUES (:deptno,  
:dname, :loc)' USING deptno_in, dname_in, loc_in;
```

Термином «динамический SQL» обозначаются команды SQL, которые конструируются и вызываются непосредственно во время выполнения программы. Статическими называются жестко закодированные команды SQL, которые не изменяются с момента компиляции программы.

```
BEGIN
```

```
EXECUTE IMMEDIATE 'CREATE INDEX emp_u_1 ON employees (last_name)';  
END;
```

## 45. Применение псевдостолбцов ROWID, ROWNUM в PL/SQL.

Псевдостолбцами в Oracle принято называть столбцы, которые отсутствуют в таблицах в явном виде, но могут быть использованы в запросах.

**ROWID** — это псевдостолбец, который является уникальным идентификатором строки в таблице и фактически описывает точное физическое расположение данной конкретной строки. На основе этой информации Oracle впоследствии может найти данные, связанные со строкой таблицы. Идентификаторы **ROWID** состоят из четырех компонентов: номера объекта (32 бита), относительного номера файла (10 бит), номера блока (22 бита) и номера строки (16 бит). Эти идентификаторы отображаются как 18-символьные последовательности, указывающие местонахождение данных в БД, причем каждый символ представлен в формате base-64, состоящем из символов A-Z, a-z, 0-9, + и /. Первые шесть символов – это номер объекта данных, следующие три – относительный номер файла, следующие шесть – номер блока, последние три – номер строки.

**ROWNUM** возвращает номер текущей строки запроса. Логический номер записи в запросе. Возвращает число, представляющее порядок, в котором Oracle выбирает строку из таблицы. 1 строка ROWNUM = 1

Позволяет вводить ограничение на количество выводимых записей

select \* from students where rownum < 10;

## 46. Обработка исключений в PL/SQL, стандартные исключения, генерация и обработка исключения.

**Исключительная ситуация** – событие, возникающее в программе и требующее незамедлительной обработки.

- 1) Программно-определяемые
- 2) Предопределенные (стандартные)

**Sqlerrm** – функция возвращает сообщение об ошибке

**Sqlcode** – функция возвращает номер ошибки

**Обработка исключений** – перехват ошибки в секции exception

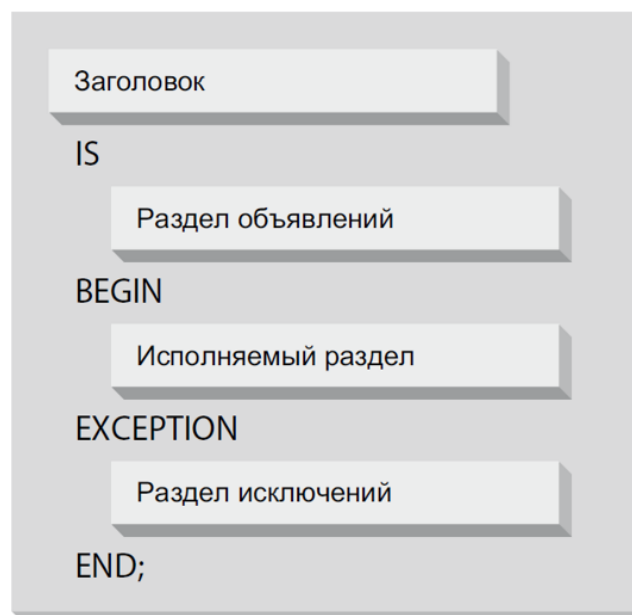
Секция исключительных ситуаций:

- может содержать столько блоков WHEN, сколько выделяется обрабатываемых исключений
- остальные в WHEN OTHERS
- можно определять свои исключения

```
EXCEPTION
    WHEN exception_name
    THEN
        операторы обработки ошибки;
    . . .
    [WHEN OTHERS
    THEN
        обработка исключения по умолчанию;]
```

### Обработка исключений

- ▶ Ошибка, сгенерированная сервером
- ▶ Ошибка в результате действий пользователя
- ▶ Ошибка, сгенерированная приложением пользователю



Секция исключений exception – необязательная секция в PL/SQL блоке, которая содержит один или несколько обработчиков исключений

## 47. Принцип распространения исключений в PL/SQL. Инструкция RAISE\_APPLICATION\_ERROR.

**Распространение исключений** – процесс передачи исключения от одного блока другому, если исключение не было обработано! С помощью raise

Существует два типа исключений:

- Системное исключение определяется в Oracle и обычно иницируется исполняемым ядром PL/SQL, обнаружившим ошибку. Одним системным исключениям присваиваются имена (например, NO\_DATA\_FOUND), другие ограничиваются номерами и описаниями.
- Исключение, определяемое программистом, актуально только для конкретного приложения.

Имя исключения можно связать с конкретной ошибкой Oracle с помощью директивы компилятора EXCEPTION\_INIT или же назначить ошибке номер и описание процедурой RAISE\_APPLICATION\_ERROR.

```
raise_application_error(  
    error_number,  
    message  
    [, {TRUE | FALSE}]  
);
```

Инициировать исключение с номером ошибки (в диапазоне от –20 999 до –20 000) может и разработчик приложения, воспользовавшись для этой цели процедурой RAISE\_APPLICATION\_ERROR.

При выполнении процедуры:

- выполнение блока прерывается
- любые изменения в аргументах IN OUT и OUT откатываются
- изменения в глобальных структурах(пакетные переменные, объекты базы данных) не откатываются – для отката надо явно выполнить ROLLBACK

## 48. Встроенные ф-ции языка PL/SQL. Ф-ции работы с датами, текстом и числами.

**TO\_DATE( string1, [ format\_mask ], [ nls\_language ] )** приводит к дате

```
SELECT TO_DATE('2019/07/22', 'yyyy/mm/dd') FROM DUAL;
```

--

Результат: 22.07.2019

**TO\_CHAR( value, [ format\_mask ], [ nls\_language ] )** приводит к символьной строке

```
SQL> SELECT TO_CHAR(1242.78, '9999.9') FROM DUAL;
```

--

Результат: 1242.8

### РАБОТА СО СТОРОКАМИ И СИМВОЛАМИ

select **concat**(Марина ', 'Шастовская') from dual; --Марина Шастовская

select **lower**('МАРИНА ШАСТОВСКАЯ') from dual; --марина шастовская

select **upper**('марина шастовская') from dual; --МАРИНА ШАСТОВСКАЯ

select **initcap**('марина') from dual; --Марина

select **trim**() вырезает пробелы с начала и с конца строки, **ltrim** () вырежет слева, **rtrim** () вырежет справа

select **length** – возвращает длину строки в байтах

Функция **TO\_NUMBER** – преобразует строковое значение в числовое

### Числовые

**Abs** – это взятие модуля

**Ceil** – округляет ВВЕРХ до целых((-10.493) => -10)

**Round** – округляет до опред числа знаков после запятой((-10.493) будет -10)

**Round** – округляет до опред числа зн после запятой((-10.493, 1) => -10.5)

**Floor** – округляет ВНИЗ до целых((-10.493) => 11)

**Mod**(-10, 3) деление по модулю

**Cos()**, **acos()**, **sin()**, **asin()**, **tan()**, **atan()**, **exp()**, **power**(число, степень), **sqrt**(число), **log**(число, основание)

### Работа с датами

**current\_date** – текущая дата

**sysdate** – ТЕКУЩИЕ ДАТА И ВРЕМЯ

**dbtimezone**, **sessiontimezone**

**localtimestamp**

**extract**(year from date '2003-08-22')

**extract**(month from date '2003-08-22')

**extract**(day from date '2003-08-22')

**month\_between** (startdate, enddate)

**trunc** (d, [формат]) – возвращает дату d, усеченную до указанной единицы измерения

### Обработка ошибок

**Sqlcode** – возвращает код(номер) ошибки,

**Sqlerrm** – возвращает текст(сообщение) ошибки

### Конвертирование

**to\_number**(строка, формат), **to\_char**(строка, формат), **to\_date**(строка, формат),

**convert**(строка, формат)



## 49. Встроенные функции языка PL/SQL. Функции регулярных выражений

Регулярные выражения - формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов

**REGEXP\_LIKE** выбирает все строки, соответствующие заданному шаблону

**REGEXP\_INSTR** определяет местоположение вхождения шаблона в строку

**REGEXP\_REPLACE** заменяет шаблон выражения на заданный

**REGEXP\_SUBSTR** выделяет из строки шаблон

**REGEXP\_COUNT** определяет количество вхождений

Шаблон регулярных проявлений в виде возникновения, возможной конструкции:

- Литеральные символы . обнаружение символов, которые следует искать (Например, шаблон xyz соответствует только вхождению «xyz»)
- Метасимволы . (Например, шаблон ^xyz соответствует только графику, начинающемуся с «xyz» — другие поступления не учитываются)

**^** Соответствует началу строки

**\$** Соответствует концу строки

**\*** Соответствует 0 или более вхождений.

**+** Соответствует 1 или более вхождений.

**?** Соответствует 0 или 1 вхождению.

**.** Соответствует любому символу, кроме NULL

**|** Используется как "OR", чтобы указать более одной альтернативы.

**[]** Используется для указания списка совпадений, где вы пытаетесь соответствовать любому из символов в списке.

**\w** Соответствует текстовому символу.

**\s** Соответствует символу пробел

В целом регулярное выражение предусматривает поиск в символьных строках определенных образцов. Символьная строка может относиться к типу CHAR, VARCHAR2, NCHAR, NVARCHAR2, а регулярное выражение — содержать одну из следующих функций: REGEXP\_LIKE, REGEXP\_REPLACE, REGEXP\_INSTR, REGEXP\_SUBSTR.

## 50. Коллекции. Массивы переменной длины в PL/SQL.

**Коллекция** – структура данных, содержащая элементы одного типа. **Элементом** коллекции может быть как скалярная величина, так и композитные данные. Элементы коллекций можно сравнивать между собой на эквивалентность. Можно передавать параметром

Коллекция состоит из **набора** элементов, причем каждый элемент находится в определенной позиции (имеется **индекс** элемента)

Необходимо объявить **тип коллекции** – командой TYPE

Необходимо объявить **коллекцию** – переменную этого типа для дальнейшего использования.

Коллекция называется **ограниченной**, если заранее определены границы возможных значений индексов ее элементов, иначе **неограниченной**

Коллекция называется **плотной**, если все ее элементы, определены и каждому из них присвоено некоторое значение (таковым может быть и NULL)

declare

```
type myarraytype is varray(3) of number;
```

```
va muarraytype := myarraytype(null, null, null)
```

**va.extend;** -добавляет один нулевой элемент

**va.extend(n);** -добавляет n нулевых элементов

**va.delete(idx);** - удаляет по индексу

**va.count;** вернет количество элементов

**Varray (массив с переменным размером)** - это массив, число элементов которого может варьироваться от нуля (пусто) до объявленного максимального размера.

Чтобы получить доступ к элементу переменной Varray, используйте синтаксис **variable\_name(index)**.

**Нижняя граница** index равна 1; **верхняя граница** - это текущее количество элементов.

**Верхняя граница** изменяется при добавлении или удалении элементов, но она не может превышать максимальный размер.

### Использование Varray

Varray целесообразно использовать, когда:

- Вы знаете максимальное количество элементов.
- Вы получаете доступ к элементам последовательно.
- Поскольку вы должны хранить или извлекать все элементы одновременно, Varray может оказаться непрактичным для большого количества элементов.



## 51. Коллекции. Вложенные таблицы в PL/SQL.

**Коллекция** – структура данных, содержащая элементы одного типа. **Элементом** коллекции может быть как скалярная величина, так и композитные данные. Элементы коллекций можно сравнивать между собой на эквивалентность. Можно передавать параметром

Коллекция состоит из **набора** элементов, причем каждый элемент находится в определенной позиции (имеется **индекс** элемента)

Необходимо объявить **тип коллекции** – командой TYPE

Необходимо объявить **коллекцию** – переменную этого типа для дальнейшего использования.

Коллекция называется **ограниченной**, если заранее определены границы возможных значений индексов ее элементов, иначе **неограниченной**

Коллекция называется **плотной**, если все ее элементы, определены и каждому из них присвоено некоторое значение (таковым может быть и NULL)

declare

```
type myarraytype is varray(3) of number;
```

```
va muarraytype := myarraytype(null, null, null)
```

**va.extend;** -добавляет один нулевой элемент

**va.extend(n);** -добавляет n нулевых элементов

**va.delete(idx);** - удаляет по индексу

**va.count;** вернет количество элементов

### Вложенные таблицы

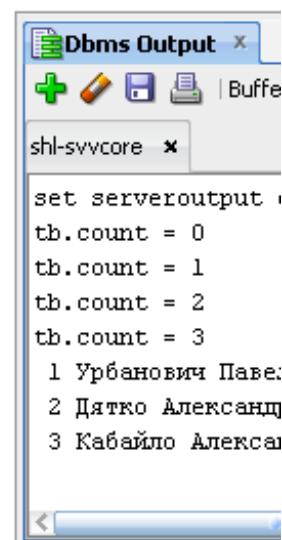
Вложенные таблицы – одномерные, несвязанные коллекции однотипных элементов

Доступны в рамках PL/SQL и как поля таблицы в БД

Изначально являются плотными, но могут впоследствии становиться разреженными

```
declare
    type tperson is record
    (
        name teacher.teacher_name%type,
        pulpit teacher.pulpit%type
    );
    type mytable is table of tperson;
    tb mytable := mytable();
    rec tperson; -- := tperson('xxx','xxxxxxx');
begin
    dbms_output.put_line('tb.count = '||tb.count);

    tb.extend;
    dbms_output.put_line('tb.count = '||tb.count);
    tb(1).name := 'Урбанович Павел Павлович';
    tb(1).pulpit := 'ИСИТ';
    tb.extend;
    dbms_output.put_line('tb.count = '||tb.count);
    tb(2).name := 'Дятко Александр Аркадьевич';
    tb(2).pulpit := 'ИСИТ';
    tb.extend;
```



## 52. Коллекции. Ассоциативные массивы в PL/SQL.

**Коллекция** – структура данных, содержащая элементы одного типа. **Элементом** коллекции может быть как скалярная величина, так и композитные данные. Элементы коллекций можно сравнивать между собой на эквивалентность. Можно передавать параметром

Коллекция состоит из **набора** элементов, причем каждый элемент находится в определенной позиции (имеется **индекс** элемента)

Необходимо объявить **тип коллекции** – командой TYPE

Необходимо объявить **коллекцию** – переменную этого типа для дальнейшего использования.

Коллекция называется **ограниченной**, если заранее определены границы возможных значений индексов ее элементов, иначе **неограниченной**

Коллекция называется **плотной**, если все ее элементы, определены и каждому из них присвоено некоторое значение (таковым может быть и NULL)

declare

```
type myarraytype is varray(3) of number;
```

```
va muarraytype := myarraytype(null, null, null)
```

**va.extend;** -добавляет один нулевой элемент

**va.extend(n);** -добавляет n нулевых элементов

**va.delete(idx);** - удаляет по индексу

**va.count;** вернет количество элементов

### Ассоциативные массивы

Ассоциативные массивы – одномерные, неограниченные (по максимальному количеству элементов при создании) коллекции элементов

Доступны только в рамках PL/SQL

Изначально являются разреженными, индекс может принимать непоследовательные значения

## 53. Процедурные объекты.

### Процедурные объекты Oracle

Для программирования алгоритмов обработки данных, реализации механизмов поддержки целостности базы данных Oracle использует такие объекты как процедура, функция, пакет и триггер. Для написания этих программных единиц используется встроенный в Oracle процедурный язык программирования PL/SQL (ProgramLanguage/SQL).

**Функция (FUNCTION)** – это поименованный, структурированный набор переменных и операторов SQL и PL/SQL, предназначенный для решения конкретной задачи и возвращающий значение (результат работы) в вызывающую программу (среду).

**Процедура (PROCEDURE)** – это поименованный, структурированный набор переменных и операторов SQL и PL/SQL, предназначенный для решения конкретной задачи. Процедура не возвращает значений в вызывающую программу (среду).

**Пакет (PACKAGE)** – это поименованный, структурированный набор переменных, процедур и функций, связанных единым функциональным замыслом. Например, Oracle предоставляет пакет DBMS\_OUTPUT, в котором собраны процедуры и функции, предназначенные для организации ввода-вывода.

**Триггер (TRIGGER)** – это хранимая процедура, которая автоматически запускается тогда, когда происходит связанное с триггером событие. Обычно события связаны с выполнением операторов INSERT, UPDATE или DELETE в некоторой таблице.

## 54. Процедурные объекты. Хранимые процедуры. Вызов процедур. Входные и выходные параметры, позиционный и параметрический форматы передачи фактических параметров. Значения параметров по умолчанию.

Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML). Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

Параметры:

- Наименование
- Тип данных
- Режим передачи
- Начальное значение

Три типа параметров:

- IN
- OUT
- INOUT.

Благодаря выходным параметрам процедура может косвенно возвращать результаты выполнения к точке вызова.

### Значения по умолчанию

#### ► IN, IN OUT

#### ► Можно не задавать при вызове

Необходимые привилегии:

```
GRANT CREATE PROCEDURE TO JEU_12;
```

Синтаксис:

```
CREATE OR REPLACE PROCEDURE XSUM(  
MIN_CY IN AUDITORIUM.AUDITORIUM_CAPACITY%TYPE,  
MAX_CY IN OUT AUDITORIUM.AUDITORIUM_CAPACITY%TYPE,  
N_AUD OUT NUMBER)  
IS  
M_MAX_CY AUDITORIUM.AUDITORIUM_CAPACITY%TYPE;
```

Передача параметров:

Позиционный — каждое значение в списке аргументов вызова ставится в соответствие формальному параметру по порядку. *Empid\_to\_name(23, name, surname);*

Именованный — явно связывает аргументы при вызове с параметрами по именам. *Empid\_to\_name(in\_id => 23, out\_name => name, out\_surname => surname);*

Можно комбинировать оба метода, пока позиционные аргументы стоят слева.

*Empid\_to\_name(23, name, out\_surname => surname);*

## 55. Процедурные объекты. Хранимые функции. Параметры функции. Вызов функций. Понятие детерминированной функции. Значения параметров по умолчанию.

Хранимые функции являются разновидностью хранимых процедур. функция рассматривается в качестве выражения, формирующего одно единственное значение. Хранимые функции применяются для расширения функциональных возможностей операторов **SELECT** и ряда других SQL-операторов.

С тем различием что функции явно возвращают значение в точку вызова они практически идентичны процедурам.

Синтаксис:

```
CREATE OR REPLACE FUNCTION SELCY(  
MIN_CY IN AUDITORIUM.AUDITORIUM_CAPACITY%TYPE,  
MAX_CY IN OUT AUDITORIUM.AUDITORIUM_CAPACITY%TYPE  
)
```

**DETERMINISTIC** – функция детерминирована, если она возвращает одно и то же значение при вызове с теми же параметрами

**AGGREGATE USING** – используется для агрегатных функций.

Может быть вызвана следующим образом:

- ▶ В присвоении начального значения переменной
- ▶ В выражении присвоения
- ▶ В булевом выражении
- ▶ В SQL запросе
- ▶ Как аргумент в списке параметров другой функции или процедуры



## 56. Процедурные объекты. Пакеты. Спецификация и реализация пакета. Процедурные объекты. Триггеры. Виды триггеров. Классификация, порядок выполнения и предикаты триггеров. Триггеры замещения. Привилегии. Включение/отключение триггеров. Псевдозаписи old и new.

Пакеты - коллекция PL/SQL объектов, сгруппированных вместе.

Преимущества:

- ▶ Скрытие информации
- ▶ Объектно-ориентированный дизайн
- ▶ Постоянство объектов в транзакциях
- ▶ Улучшенная производительность

Можно включать в пакет: процедуры, функции, константы, исключения, курсоры, переменные, TYPE выражения, записи, REF курсоры

Спецификация пакета (package) – обязательна, содержит список объектов для общего доступа из других модулей или приложения

Реализация пакета (package body) – содержит весь программный код для реализации процедур и функций и спецификации, приватные объекты и секцию инициализации

Вызов пакета:

Package\_name.package\_element;

Структуры данных, объявленные в пакете, называются пакетными данными

Пакетные переменные сохраняют свое состояние от одной транзакции к другой и являются глобальными данными

▶ **Триггер** – особый вид процедур, которые срабатывают по запускающему их событию

Привилегии:

- ▶ **CREATE TRIGGER** - создавать, удалять, изменять в своей подсхеме
- ▶ **CREATE ANY TRIGGER** - создать любой триггер в любой схеме, кроме SYS, не рекомендуется для словаря, не разрешает менять текст триггера
- ▶ **ALTER ANY TRIGGER** - разрешать, запрещать, изменять, компилировать, любые, кроме SYS-триггеров, триггеры
- ▶ **DROP ANY TRIGGER** - удалять любой триггер, кроме SYS-триггеров
- ▶ **ADMINISTER DATABASE TRIGGER** - создавать, изменять, удалять системные триггеры, должен иметь привилегию CREATE TRIGGER или CREATE ANY TRIGGER

### Классификация триггеров:

**\*По привязанному объекту:**

На таблице

- ▶ На представлении - instead of trigger

**\*По событиям запуска:**

- Вставка записей - insert
- Обновление записей - update
- Удаление записей - delete

**\*По области действия:**

- Уровень оператора - statement level triggers

-Уровень записи - row level triggers

► -оставные триггеры - compound triggers

**\*По времени срабатывания:**

-Перед выполнением операции – before

-После выполнения операции - after

**Псевдозаписи new, old**

| Операция<br>срабатывания<br>триггера | OLD.column      | NEW.column     |
|--------------------------------------|-----------------|----------------|
| Insert                               | Null            | Новое значение |
| Update                               | Старое значение | Новое значение |
| Delete                               | Старое значение | Null           |

Включение и отключение триггеров:

```
alter trigger { disable | enable }
```

Всех для таблицы:

```
ALTER TABLE table_name { ENABLE | DISABLE } ALL TRIGGERS;
```

Компиляция

```
alter trigger TRIGGER_NAME compile;
```

Переименование триггера

триггера:



## 57. Секционирование таблиц. Виды секционирования.

**Секционирование** – метод, кот. позвол. хранить сегмент д-х в виде неск. сегментов.

- \*повыш. произ-сть обраб. д-х
- \*упрощ. упр-ние крупными объектами д-х
- \*обесп. доп. надежность системы

**Диапазонное (range)** – секционир., при кот. для кажд. секции опред. диап. знач. ключа секционирования.

- \*ключ м. приним. знач даты и времени, числа, текста
- ! при загрузке новых д-х, необх. посто. расщеплять секцию maxvalue;
- \*для задания диап. исп. ключ. слово *less than*

Пример:

```
create table T_RANGE( docnum number) partition by range(docnum)
(partition n1 values less than(10) tablespace t1,
partition n2 values less than(20) tablespace t2,
partition n3 values less than(30) tablespace t3);
```

**Интервальное** – при загрузке новых д-х в табл. авто созд. новые секции для нового диап. ключей.

- \*новые секции будут созд. авто (по 1ой оп-ции insert, не попад. в диап. секций)

```
create table T_INTERVAL( t_date date) partition by range(t_date)
interval(NUMTOYMINTERVAL(3, 'MONTH'))
(partition d1 values less than (to_date('01.01.2016', 'DD/MM/YYYY')),
partition d2 values less than (to_date('01.03.2016', 'DD/MM/YYYY')));
```

**Хэш-секц** позвол. равномерн. распред. строки между секциями, т.е. разбросать строки по разным секциям и сделать эти секции равновеликими.

!не означ, что строки распред. по секциям случайным образом

```
create table T_HASH( doctype varchar2(5))
partition by hash(doctype)
(partition p1 tablespace th1,
partition p2 tablespace th2)
```

**Списочное** – позвол. разбить таблицу по списку конкр. знач.

- \*ключ спис. секц. мб только 1 столбцовым
- \*default – опис. все знач, не попавшие в другие списки
- \*null – мб ключ. знач.

```
create table T_LIST (l1 char(10)) partition by list(l1)
(partition ch1 values('a', 'b') tablespace tl1,
partition ch2 values('c') tablespace tl2);
```

**Композитное** - слишком большие таблицы – секционирование по 2 ур. (секции и подсекции)

## 58. Транзакции. Виды транзакций.

Транзакция – это логическая единица работы в базе данных Oracle, состоящая из одного или более операторов SQL. Транзакция начинается с первого исполняемого оператора SQL и завершается, когда вы фиксируете или отказываете транзакцию. Как только вы зафиксировали транзакцию, все прочие транзакции других пользователей, которые начались после нее, смогут видеть изменения, проведенные вашими транзакциями.

### Операторы:

- \* **COMMIT** – фиксирует все изменения для текущей транз
- \* **ROLLBACK** – откат незафикс. изменений: читает инфу из сегментов отката и восст. блоки д-х в сост, в кот. они находились до нач. транз, освобождает блокировку
- \* **SAVEPOINT** – в одной транз мб неск, созд возможность отмены только части работы, сделан. в транз. Освоб. блокировки, кот. были установлены отмененным оператором
- \* **SET TRANSACTION** – установ. атриб транз – уровень изолир, будет исполн. д/чтения или записи

**Автономные транзакции** – позв. создать *подтранзакции*, которыми можно сохранить/отменить изменения вне зависимости от родительской транзакции

## 59. Обработка заданий. Системные пакеты обработки заданий в Oracle.

DBMS\_JOB – поддержка управления заданиями. ПАКЕТ ПЛАНИРОВЩИКА ЗАДАНИЙ

Пакет DBMS\_JOB позволяет запланировать однократное или регулярное выполнение заданий в базе данных. Задание представляет собой хранимую процедуру, анонимный блок PL/SQL или внешнюю процедуру на языке C или Java. **Эти задания выполняются серверными процессами в фоновом режиме**

```
ALTER SYSTEM SET JOB_QUEUE_PROCESSES=9;
```

► SUBMIT – создание задания – АВТОНУМЕРАЦИЯ

dbms\_job.submit(jobnumber, - номер; action, - что; SYSDATE, - запустить сразу; 'SYSDATE + interval "1" minute' – следующий раз запуска );

► ISUBMIT – создание задания с номером

dbms\_job.submit(номер, действие, когда запустить первый раз, интервал);

► REMOVE – удаление задания из очереди

► RUN – немедленное выполнение задания в пользовательском сеансе

Run (job, force – true/false)

► BROKEN – разрушение задания (16 раз не смогло выполниться - разрушение)

dbms\_job.broken(jobnumber, true, null);

задача – принудительно – следующий запуск

► INSTANCE – указание экземпляра выполнения

► NEXT\_DATE – изменение времени выполнения – указать следующую дату

выполнения

dbms\_job.next\_date(jobnumber, SYSDATE + interval '5' minute);

► INTERVAL – изменение интервала выполнения

► CHANGE – изменение параметров задания

dbms\_job.change(jobnumber(КАКАЯ ЗАДАЧА), null(ЧТО ДЕЛАЕТ), null(NEXT\_DATE), 'SYSDATE + interval "2" minute' (ИНТЕРВАЛ), instance, force);

► WHAT – изменение задания

select \* from dba\_jobs;

select \* from dba\_jobs\_running;

**DBMS\_SCHEDULER – комбинирует расписание + программа + job задача**

**Job = SCHEDULER + PROGRAM**

Schedule – расписание

Program - программа

Job – плановая программа

**dbms\_scheduler.create\_schedule(**

schedule\_name => 'Sch01',

start\_date => SYSTIMESTAMP,

repeat\_interval => 'FREQ=MINUTELY;INTERVAL=1;',

comments => 'Sch01 Minutely');

**dbms\_scheduler.create\_program(**

program\_name => 'Pr01',

program\_type => 'STORED\_PROCEDURE',

```
program_action => 'SCHEDULPROC',  
number_of_arguments => 2,  
enabled => false,  
comments => 'SCHEDULPROC');
```

#### **dbms\_scheduler.create\_job**

```
job_name => jobname,  
program_name => 'Pr01',  
schedule_name => 'Sch01',  
enabled => true);
```

Изменение расписания:

```
dbms_scheduler.set_attribute(name => 'SCH01', attribute => 'repeat_interval', value  
=> 'FREQ=MINUTELY;INTERVAL=2;');
```

```
dbms_scheduler.disable(jobname);
```

```
dbms_scheduler.enable(jobname);
```

```
dbms_scheduler.stop_job(job_name => jobname);
```

```
begin
```

```
dbms_scheduler.drop_job(job_name => 'ARCH_WORKERS_JOB');
```

```
end;
```

```
begin
```

```
dbms_scheduler.drop_program(program_name => 'PR01');
```

```
end;
```

```
begin
```

```
dbms_scheduler.drop_schedule(schedule_name => 'SCH01');
```

```
end;
```

## 60. Системные пакеты Oracle.

Встроенные системные пакеты:

- Устанавливаются во время установки Oracle
- Используются для расширения основных функциональных возможностей Oracle
- Владелец пакетов - SYS (обычно, есть и другие)
- Написаны на C или на PL/SQL

### Пакеты APEX:

- ▶ APEX – Oracle Application Express – среда разработки веб-приложения
- ▶ APEX\_CUSTOM\_AUTH – проверка подлинности и управления сеансом
- ▶ APEX\_APPLICATION – использование глобальных переменных
- ▶ APEX\_ITEM – создание элементов форм на основе SQL-запроса
- ▶ APEX\_UTIL – различные утилиты состояния сеанса, файлов, авторизации и пр.

### Пакеты DBMS:

- dbms\_advanced\_rewrite – перехват и замена sql-запросов
- dbms\_advisor – часть набора экспертной системы для решения проблем производительности, связанных с компонентами сервера БД
- dbms\_sqltune – сбор статистики, используется при анализе производительности операторов
- dbms\_appl\_info – присвоение имени процессу для удобства мониторинга и отладки
- DBMS\_AQ – система обмена очередями сообщений
- DBMS\_DEFER – вызов удаленных процедур
- DBMS\_BACKUP\_RESTORE
- DBMS\_CRYPTO
- DBMS\_DEBUG
- DBMS\_DESCRIBE
- DBMS\_FILE\_TRANSFER – пересылка файлов между файловыми системами
- DBMS\_JAVA
- DBMS\_JOB
- DBMS\_SCHEDULER
- DBMS\_LOCK - блокировки
- DBMS\_LOB
- DBMS\_OUTPUT серверный вывод
- DBMS\_PIPE