ВОЗМОЖНЫЕ ЗАДАЧИ

### Вывод списка всех представлений словаря данных, доступных пользователю

SELECT \* FROM DICTIONARY;

### Вывод имен всех таблиц пользователя

SELECT object\_name FROM user\_objects WHERE object\_type ='TABLE’;

### Просмотр типов объектов, принадлежащих пользователю, с помощью ключевого слова DISTINCT

SELECT DISTINCT object\_type FROM user\_objects;

### Просмотр подключений к серверу

SELECT \* FROM v$session WHERE username is not null;

### Поиск таблиц словаря данных по заданным темам в столбце COMMENTS таблицы DICTIONARY

SELECT \* FROM dictionary WHERE LOWER(comments) LIKE '%grant%';

### Вывод списка объектов базы данных

SELECT object\_name, owner, status FROM dba\_objects;

### Вывод списка пользовательских констрейнов

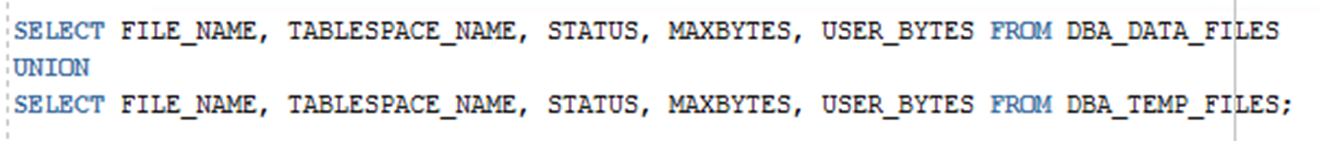
SELECT constraint\_name, constraint\_type FROM user\_constraints WHERE table\_name = ‘…’;

### Просмотр сведений о подключаемых БД

SELECT name, open\_mode, total\_size FROM v$pdbs;

### Запрос табличного пространства

* SELECT TABLESPACE\_NAME, STATUS, CONTENTS LOGGING FROM SYS.DBA\_TABLESPACES;
* select \* from dba\_tablespaces;
* select tablespace\_name, file\_name from dba\_data\_files;
* select \* from dba\_temp\_files;



### Получить перечень всех ролей

select \* from dba\_roles;

### Получить перечень привилегий для определенной роли

* select granted\_role, privilege from dba\_role\_privs pr join dba\_sys\_privs p on pr.grantee = p.grantee where granted\_role = 'RL\_SYMCORE';
* select \* from dba\_role\_privs;
* select \* from dba\_sys\_privs;
* SELECT \* FROM DBA\_ROLE WHERE ROLE LIKE ‘RL%’;
* SELECT \* FROM DBA\_SYS\_PRIVS WHERE GRANTEE = ‘ROLE’;

### Получить перечень всех пользователей

select \* from dba\_users;

### Получить перечень всех профилей безопасности

select distinct profile from dba\_profiles;

### Параметры профиля безопасности

* select \* from dba\_profiles where profile = 'PROFILE';
* select \* from dba\_profiles where profile = 'DEFAULT';

### Сегмент

SELECT DISTINCT SEGMENT\_TYPE FROM DBA\_SEGMENTS;



### Получите список сегментов табличного пространства.

select segment\_name from dba\_segments where tablespace\_name = 'SYSTEM';

### Экстент

select \* from user\_extents where tablespace\_name = ‘ts\_ea’

### Вычислите количество блоков, занятых таблицей.

select blocks from dba\_segments where segment\_name = 'T4';

### Подсчитайте размер данных в таблице.

select bytes from dba\_segments where segment\_name = 'T4';

### Получите список всех параметров экземпляра.

* select \* from v$parameter;
* show parameter instance;

### Просмотр пользователей в файле паролей

select \* from v$pwfile\_users;

### Файл сообщений

select \* from v$diag\_info;

### Получите перечень всех групп журналов повтора.

select \* from v$log;

### Определите текущую группу журналов повтора.

select \* from v$log where status = 'CURRENT';

### Получите перечень файлов всех журналов повтора.

select \* from v$logfile;

### Выведите, производится ли архивирование журналов повтора.

select log\_mode from v$database;

### Переключение журналов повтора

alter system switch logfile;

select group#, sequence#, bytes, members, status, first\_change# from v$log;

select \* from v$archived\_log;

### Режим работы экземпляра

select name, log\_name, loge\_mode from v$database;

select instance\_name, archiver, active\_state from v$instance;

### Режим соединения

select username, service\_name, server from v$session where username is not null;

### Структура оперативной памяти(sga) размер области и еще что-то там

select \*from v$sga;

select sum(value) from vsga;

### Представления словаря связанные с SGA

select current\_size from v$sga\_dynamic\_free\_memory(v$sga\_dynamic\_components

);

select \* from v$sgastat;

### Все что связано с LISTENER

select name, network\_name, pdb frm v$services;

### Выведите список сессий.

select \* from v$session;

select username, sid, serial4, server, paddr, status from v$session where username is not null;

### Параметры dispatcher

show parameter dispatcher

### Получите перечень всех процессов СУБД Oracle. Для серверных процессов укажите режим подключения. Для фоновых укажите работающие в настоящий момент.

* select \* from v$bgprocess where paddr != '00'; -- активные
* select \* from v$process; -- вообще все
* select name, network\_name, pdb from v$services; -- активные подключения
* select username, status, server from v$session where username is not null; --активные серверные подключения
* select username, status, server from v$session where server = ‘DEDICATED’ -- режим подключения - выделенный

### Добавление новой группы журнала повтора

alter database add logfile group 4 ‘c:\bd\redo4.log’

size 50m blocksize 512;

### Добавление файла в группу

alter database add logfile members ‘c:\bd\redo4.log’ to group 4;

**ТУТ ВСЕ ПОСЛЕДОВАТЕЛЬНО ЧТО нужно сделать ДО СОЗДАНИЯ ПОЛЬЗОВАТЕЛЯ(включительно)**

### Создание табличного пространства

CREATE TABLESPACE ts\_ea

DATAFILE ‘C:\bd\ts\_EA.dbf’

SIZE 10 m

AUTOEXTEND ON NEXT 500K

MAXSIZE 100M

EXTENT MANAGEMENT LOCAL;

### Временное табличное пространство

CREATE TEMPORARY TABLESPACE ts\_temp\_ea

TEMPFILE ‘C:\bd\ts\_temp\_es.dbf’

SIZE 10 m

AUTOEXTEND ON NEXT 500K

MAXSIZE 100M

EXTENT MANAGEMENT LOCAL;

### Создание роли

create role ROLE;

--drop role ROLE;

grant create session,

create table,

create procedure,

create view

to ROLE;

### Создание профиля безопасности

create profile PROFILE limit

password\_life\_time 180

sessions\_per\_user 3

failed\_login\_attempts 7

password\_lock\_time 1

password\_reuse\_time 10

password\_grace\_time default

connect\_time 180

idle\_time 30

### Создание пользователя

create user USER identified by "123456"

default tablespace ts\_ea quota unlimited on ts\_ea;

temporary tablespace ts\_temp\_ea

profile PROFILE

account unlock

password expire

GRANT ROLE TO USER;

### Сформируйте PFILE

* show parameter spfile;
* create pfile = 'test\_exam\_pfile.ora' from spfile = 'SPFILEORCL.ORA';

--or

* create pfile = 'test\_exam\_pfile\_from\_memory.ora' from memory;

#### сформировать spfile

CREATE SPFILE = ‘TEXT\_EXAM’ FROM PFILE=’test\_exam\_pfile.ora’

### Управляющий файл

select name from v$controlfile;

select type, record\_size. records\_total from v$controlfile\_record\_section;

Включение/отключение архивирования

shutdown immediate;

startup mount;

alter database archivelog;

alter database open;

—

startup mount;

alter database noarchivelog;

### Создайте таблицу из двух столбцов, один из которых первичный ключ. Получите перечень всех сегментов. Вставьте данные в таблицу. Определите, сколько в сегменте таблицы экстентов, их размер в блоках и байтах

create table T1(

n1 number(3) primary key,

ch1 nvarchar2(4)

);

create table T2(

n2 number(3),

ch2 nvarchar2(4),

constraint fk\_T1 foreign key (n2) references T1(n1)

);

insert into T1 values (1, cc) (‘n1’, ‘ch1’ );

insert into T2 values (1, cc) (‘n2’, ‘ch2’, );

select \* from t1;

select \* from t2;

**И САМЫЙ ВАЖНЫЙ ЗАПРОС**

select segment\_name, segment\_type, extents, blocks, bytes from user\_segments where segment\_name in ('T1', 'T2');

### Создайте последовательность S1, со следующими характеристиками: начальное значение 1000; приращение 10; минимальное значение 0; максимальное значение 10000; циклическую; кэширующую 30 значений в памяти; гарантирующую хронологию значений. Создайте таблицу T1 с тремя столбцами и введите (INSERT) 10 строк, со значениями из S1.

create sequence S1

increment by 10

start with 1000

maxvalue 10000

minvalue 0

cycle

cache 30

order;

create table T1 (C1 number(10), C2 number(10), C3 number(10))

declare

val1 int;

val2 int;

val3 int;

max\_i int := 10;

begin

for i in 0..max\_i

loop

select S1.nextval into val1 from dual;

select S1.nextval into val2 from dual;

select S1.nextval into val3 from dual;

insert into T1 values (val1, val2, val3);

end loop;

commit;

end;

select \* from T1;

### Создайте частный и публичный синоним для одной из таблиц и продемонстрируйте его область видимости. Найдите созданные синонимы в представлениях словаря Oracle.

create synonym LS for T3;

create public synonym PS for T3;

select \* from dba\_synonyms where table\_name = 'T3';

### Разработайте анонимный блок, демонстрирующий возникновение и обработку исключений WHEN TO\_MANY\_ROWS и NO\_DATA\_FOUND

select \* from orders;

declare

order\_row orders%rowtype;

begin

select \* into order\_row

from orders;

dbms\_output.put\_line(order\_row.order\_num || ' ' || order\_row.order\_date);

exception

when too\_many\_rows

then dbms\_output.put\_line('Error number: ' ||sqlcode);

dbms\_output.put\_line('Error text: ' ||sqlerrm);

end;

declare

order\_row orders%rowtype;

begin

select \* into order\_row

from orders

where order\_num = '000000000';

dbms\_output.put\_line(order\_row.order\_num || ' ' || order\_row.order\_date);

exception

when no\_data\_found

then dbms\_output.put\_line('Error number: ' ||sqlcode);

dbms\_output.put\_line('Error text: ' ||sqlerrm);

end;

### Создайте таблицу и вставьте в нее 100 записей. Найдите таблицу и ее свойства в представлениях словаря.

create table T4 (x number(3));

declare

i number(3) :=1;

max\_i number(3) := 100;

begin

while i <= max\_i

loop

insert into T4 values (dbms\_random.value(1,998));

i := i + 1;

end loop;

end;

select \* from T4;

**ВОТ ЭТО ОЧЕНЬ ВАЖНЫЕ ЗАПРОСЫ!!!**

select \* from user\_segments where segment\_name = 'T4';

select \* from dba\_segments where segment\_name = 'T4';

select \* from user\_objects where object\_name = 'T4';

select \* from user\_tables where table\_name = 'T4';

### Продемонстрируйте эскалацию исключения.

declare

order\_row orders%rowtype;

begin

select \* into order\_row

from orders;

begin

dbms\_output.put\_line(order\_row.order\_num || ' ' || order\_row.order\_date);

exception

when too\_many\_rows

then dbms\_output.put\_line('привет из внутреннего блока');

end;

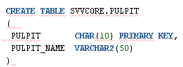
exception

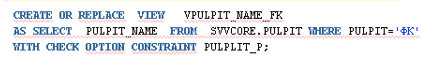
when too\_many\_rows

then dbms\_output.put\_line('привет из внешнего блока');

end;

### Создать представление





### Создать dblink

GRANT CREATE DATABASE LINK TO USERNAME

CREATE DATABASE LINK name

CONNECT TO USER2

IDENTIFIED BY PASSWORD

USING 'INST\_B'; 9сетевое имя создается в утилите net…)

SELECT name FROM table@name;

ВОПРОСЫ

1. Дистрибутивы СУБД Oracle. Установка СУБД Oracle 12с 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 базы данных: назначение, применение, основные представления.
6. Мультиарендная архитектура Oracle Multitenant.
7. Файлы экземпляра Oracle. Файл параметров, управляющие файлы, файлы паролей, файлы трассировки.
8. Файлы базы данных 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.
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. Материализованное представление и его параметры.
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. Связанные объявления переменных: 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. Курсоры. Курсорные переменные. Параметры инстанса, связанные с курсорами.
42. Курсоры. Курсорные подзапросы.
43. Курсоры. Использование конструкции CURRENT OF в курсорах.
44. Курсоры. 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 функции языка PL/SQL. Функции работы с датами, текстом и числами.
49. Встроенные функции языка PL/SQL. Функции регулярных выражений
50. Коллекции. Массивы переменной длины.
51. Коллекции. Вложенные таблицы.
52. Коллекции. Ассоциативные массивы.
53. Процедурные объекты. Хранимые процедуры. Вызов процедур. Входные и выходные параметры, позиционный и параметрический форматы передачи фактических параметров. Значения параметров по умолчанию.
54. Процедурные объекты. Хранимые функции. Параметры функции. Вызов функций. Понятие детерминированной функции. Понятие pipeline функции. Значения параметров по умолчанию.
55. Процедурные объекты. Пакеты. Спецификация и реализация пакета.
56. 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. Транзакции. Виды транзакций. Понятие автономной транзакции.
59. Обработка заданий. Системные пакеты обработки заданий в Oracle.
60. Системные пакеты Oracle.

ОТВЕТЫ

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

## Дистрибутивы

**Дистрибутив программы** – это набор файлов (пакет) самой программы, [архивированных](https://www.pc-school.ru/arxivaciya-fajlov-sredstvami-windows/) определенным образом, файлы с инструкциями для операционной системы (куда и что устанавливать) и файл инсталлятор, который является своего рода инструкцией для пользователя, так называемый мастер установки, который пошагово поможет провести весь процесс установки программы.

1977: первая версия

2005: Oracle 10g Release 2

2009: Oracle 11g Release 2 - поддерживается на платформах Windows (начиная с Windows Server 2008), Linux, Solaris, HP, AIX5L, IBM z/OS, Mac OS X Server.

2013: Oracle 12c.

#### Основные редакции

**Standard Edition One:** начальная редакция для дома и малого бизнеса;

**Standard Edition:** для малого и среднего бизнеса;

**Enterprise Edition:** решения промышленного уровня;

#### Лицензирование

**Named User Plus** – по пользователям, можно конкретно подсчитать точное количество пользователей системы

**Processor** – по процессорной мощности серверов на неограниченное количество пользователей

### Скачивание

на oracle.com выбрать Database 12c > Download > 2 файла для ОС

### Инсталляция

1. Запуск Oracle Database 12c Installer
2. Configure Security Updates - можно указать Email (необязательно)
3. Software Updates - пропускается
4. Installation Option - Create and configure a database
5. System Class - Server class
6. Grid Installation Options - Single instance database installation
7. Install Type - Typical install
8. Oracle Home User Selection - Create New Windows User (указываем User Name, Password, Confirm Password)
9. Typical Installation - указать Global database name, Administrative password, Confirm password
10. Prerequisite Checks
11. Summary
12. Install Product

## Global Database Name

**Глобальные имена баз данных** однозначно идентифицируют базу данных в системе. Основной задачей администрирования в распределенной системе является управление созданием и изменением имен глобальных баз данных.

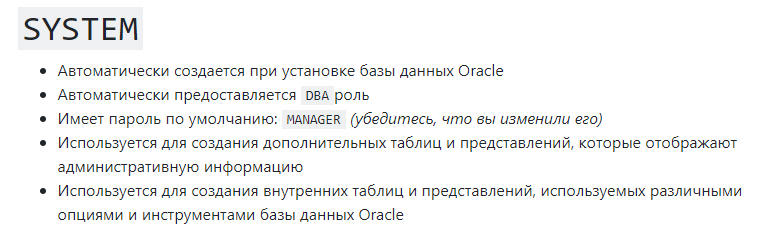
## SID

Каждый экземпляр базы данных идентифицируется с помощью **SID** **(System IDentifier – системный идентификатор)**. SID состоит из алфавитно‐цифровых символов, хранится в переменной среды ORACLE\_SID и используется утилитами и сетевыми компонентами для доступа к базе данных.

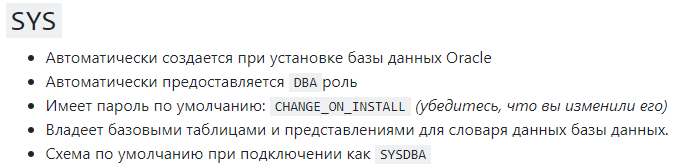
# 2. Основные системные пользователи. Основные специальные привилегии. Роль DBA.

## Основные системные пользователи

**SYSTEM** – MANAGER - предопределенный привилегированный пользователь, которому принадлежат ключевые ресурсы БД Oracle



**SYS** – CHANGE\_ON\_INSTALL - предопределенный привилегированный пользователь ранга администратора базы данных, который является владельцем ключевых ресурсов БД Oracle



**SCOTT** – TIGER

## Основные специальные привилегии

**SYSDBA и SYSOPER** - специальные привилегии администратора, которые позволяют выполнять базовые задачи администрирования: запуск или остановка экземпляра БД; создание, удаление, открытие или монтирования базы и др.

## Роль DBA

**DBA** – предопределенная роль, которая автоматически создается для каждой базы данных Oracle и содержит все системные привилегии, кроме SYSDBA и SYSOPER

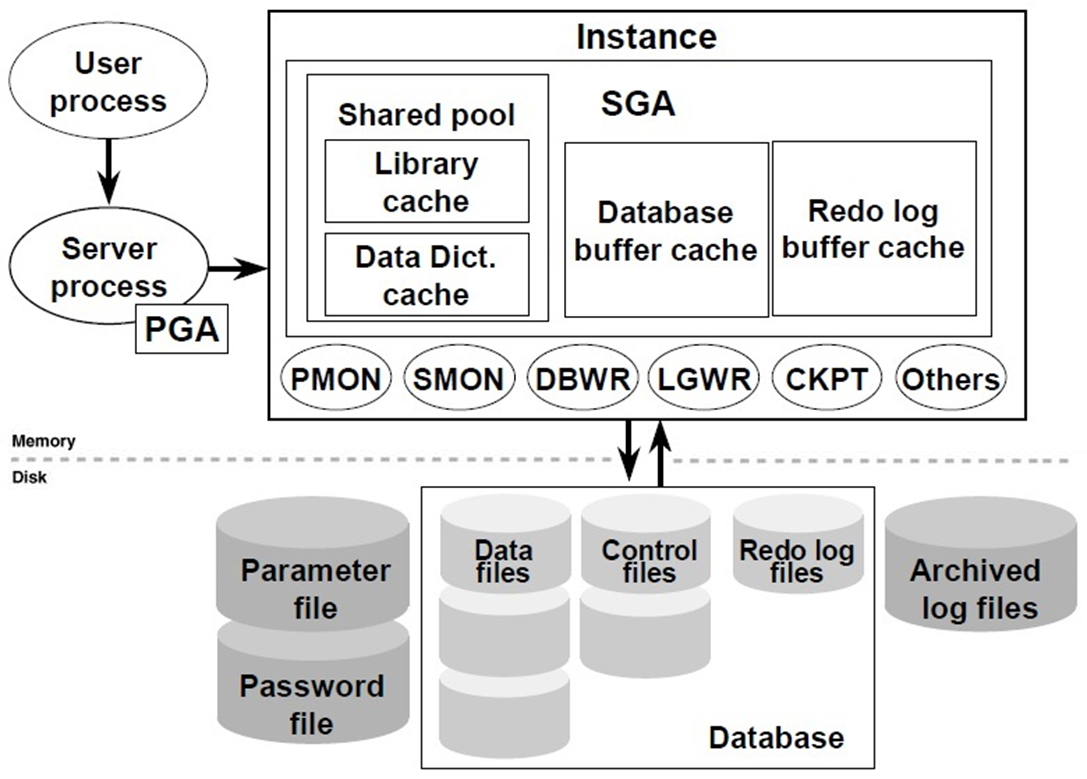
Роль DBA не включает SYSDBA и SYSOPER. Привилегии могут быть указаны при подключении (connect) пользователя к БД.

# 3. Понятия базы данных и экземпляра базы данных.

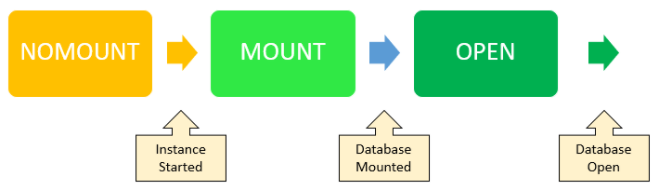
**База данных** — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД).

**Экземпляр** - набор процессов Oracle и область SGA

Системная глобальная область (англ. System global area, SGA) в Oracle Database — часть оперативной памяти, разделяемой всеми процессами одного экземпляра базы данных. SGA содержит всю необходимую информацию для операций экземпляра.



# 4. Запуск и останов экземпляра базы данных Oracle.

1. shutdown immediate - немедленное выключение
2. connect /as sysdba(SYSDBA – это системная привилегия)
3. startup - запустить
4. Когда Oracle Database запускает экземпляр, он проходит следующие этапы: NOMOUNT, MOUNT, и OPEN.
   1. **NOMOUN** - выделение системной глобальной области(SGA) на основе настроек параметров инициализации; запускаются фоновые процессы Oracle(SMON, PMON, LGWR); 
   2. **MOUNT** - связывает базу данных с экземпляром. Другими словами, экземпляр монтирует базу данных(когда бд смонтирована, она доступна только администраторам базы данных, а не всем пользователям)
   3. **OPEN** - открывает смонтированную базу данных, база данных становится доступной для обычных операций

# 5. Словарь базы данных: назначение, применение, основные представления.

Набор таблиц и связанных с ними представлений, который представляет возможность отследить внутреннюю структуру базы данных и деятельность СУБД Oracle

Создается при генерации базы данных

Обновляется и обслуживается сервером Oracle в фоновом режиме после выполнения операторов DDL

Позволяет запрашивать данные в виде представлений

Содержит следующую информацию:

- Имена пользователей сервера Oracle

- Уровни привилегий пользователей

- Имена объектов базы данных

- Табличные ограничения

- Учетные данные

## Описание словаря данных

Располагается в системном табличном пространстве SYSTEM

Владелец: пользователь SYS, некоторые представления - SYSTEM

Для доступа к словарю необходима специальная привилегия GRANT SELECT ANY DICTIONARY

## Запросы к словарю данных

* USER Объекты, принадлежащие пользователю
* ALL Объекты, к которым пользователь имеет доступ
* DBA Все объекты базы данных (для администратора БД)
* V$ Производительность сервера

Прочие представления (DICTIONARY и пр.)

## Основные представления

dba\_users, \_profiles, \_roles

dba\_ (role | sys | tab | col) \_privs,

role\_ (role | sys | tab | col) \_privs,

user\_ (role | sys | tab | col) \_privs

# 6. Мультиарендная архитектура Oracle Multitenant.

**Oracle Multitenant** - технология, позволяющая запустить несколько независимых баз данных в рамках одного экземпляра.

Каждая база данных имеет свой набор табличных пространств и набор схем, но при этом у них общая SGA и один набор серверных процессов.

Базы данных изолированы, друг о друге ничего не знают, не конфликтуют между собой.

Словарь разбивается на две части: общую часть и локальную.

**CDB** - **container DB** – контейнер базы данных

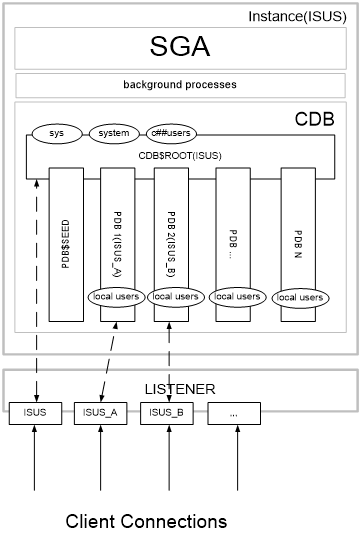
**PDB** - **pluggable DB** – подключаемые базы данных

Можно создавать несколько CDB – для разных версий программного обеспечения СУБД.

Одну и ту же PDB можно переносить между CDB.

В CDB создается главный контейнер Root. Root содержит метаданные CDB.

В одной CDB можно создать до 252 PDB.



# 7. Файлы экземпляра Oracle. Файл параметров, управляющие файлы, файлы паролей, файлы трассировки.

Файл параметров, Управляющие файлы, Файл паролей, Файл сообщений, Журналы повторного выполнения, Файлы трассировки

## Файл параметров

Файл параметров предназначен для хранения параметров экземпляра.

SELECT name, VALUE, DESCRIPTION FROM V$PARAMETER;

**SPFILE**  - файл параметров сервера в двоичном виде

Может изменяться командой ALTER SYSTEM … SCOPE=SPFILE

Изменение параметров сохраняется в SPFILE и будет применяться при следующем старте Oracle

Месторасположение файла параметров по умолчанию 

## Управляющие файлы

**Control files** – файлы, содержащие имена (местоположение) основных физических файлов базы данных и некоторых параметров

Используются для поиска других файлов операционной системы;

Местоположение управляющих файлов экземпляр получает из файла параметров.

По умолчанию для надежности создается 2 управляющих файла. Можно создать больше. Обычно их размещают на разных дисковых носителях (для надежности).

### Управляющие файлы – изменение

Остановить Oracle (shutdown transactional или immediate);

Скопировать один из управляющих файлов;

Изменить параметр CONTROL\_FILES в файле параметров;

Стартовать Oracle (startup open).

Если надо изменить управляющий файл, то следует создать сценарий, откорректировать его и выполнить

ALTER DATABASE BACKUP CONTROLFILE TO TRACE;

## Файл паролей

Предназначен для аутентификации администраторов базы данных



Можно создавать, пересоздавать и изменять

Windows PWDORACLE\_SID.ora ORACLE\_HOME\database

Просмотр пользователей в файле паролей

SELECT \* FROM V$PWFILE\_USERS;

## Файлы трассировки

**Трассировка** — процесс пошагового выполнения программы

Результаты трассировки можно сохранить в файле.

**Файл трассировки** — это файл, в который записаны результаты трассировки. Файл трассировки может находиться в локальном каталоге (например, C:\имя\_папки\имя\_файла.trc) или в сетевом (например, \\имя\_компьютера\сетевое\_имя\имя\_файла.trc).

воспроизведения трассировок;

Аудит SQL Server

выполнения анализа производительности;

сравнения событий трассировки со счетчиками производительности для улучшения распознавания неполадок;

выполнения анализа при помощи помощника по настройке ядра СУБД;

оптимизации запросов.

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

## Файлы данных

**Файлы данных** - это самый низкий уровень гранулярности взаимо­действий между Oracle и операционной системой. При размещении ба­зы данных на физических устройствах наименьшая сущность, кото­рую можно куда-то поместить, - это файл. Оптимизация подсистемы ввода/вывода для повышения производительности Oracle обычно включает перемещение файлов данных с одного набора дисков на дру­гой.

В файлах данных находятся собственно данные, хранящиеся в базе данных Oracle: [таблицы](https://oracle-patches.com/oracle/prof/3002-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%8B-oracle) и [индексы](https://oracle-patches.com/oracle/prof/3013-%D0%B8%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D1%8B-oracle), [словарь данных](https://oracle-patches.com/oracle/begin/2963-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-oracle-%D0%B8-%D0%B4%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5-%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8), в котором сохраняется информа­ция об этих структурах, и сегменты отката, необходимые для реализа­ции конкурентного доступа. Файл данных состоит из блоков базы данных, в свою очередь, состоя­щих из дисковых блоков операционной системы.

## Журналы повтора

**Журналы повторного выполнения** - дисковые ресурсы, в которых фиксируются изменения вносимых пользователями в базу данных;

журнал представляет собой файл операционной системы;

как минимум должно быть два файла;

журналы применяются при восстановлении базы данных.

файлы журналов используются циклически (сначала запись идет в 1й файл, после заполнения во 2й, 3й…, затем снова в 1й)

SELECT \* FROM V$LOGFILE;

**Мультиплексирование журналов повтора** – поддержка несколько копий каждого журнала

**SCN**  – **System change number** – системный номер изменений в базе данных

SELECT \* FROM V$LOG

### Параметра

Указывается в управляющих файлах:

MAXLOGFILES - максимальное количество групп журналов повтора

MAXLOGMEMBERS - максимальное количество файлов в группе

## Архивы журналов повтора

Режимы работы экземпляра:

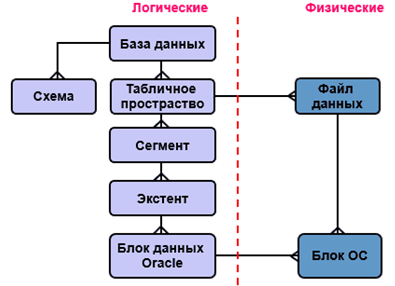
ARCHIEVELOG

NOARCHIEVELOG

SELECT NAME, LOG\_MODE FROM V$DATABASE; - покажет режим работы экземпляра

Архивные файлы журналов повтора жизненно важны при восстановлении. Если часть базы данных потеряна или повреждена, то для устранения повреждений обычно требуется несколько архивных журналов. Если один из архивных файлов журналов повтора пропущен, то остальные архивные файлы журналов не могут использоваться. Храните все свои архивные файлы журналов повтора с момента выполнения последней резервной копии. Файлы журналов постепенно накапливаются и разрастаются. Иногда необходимо их удалять. Все операции с данными файлами по применению их к базе выполняются исключительно средствами базы данных. А копировать и переносить их при желании можно как угодно.

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



В Oracle используется 2-х уровневая организация БД.

\*объекты верхнего ур-ня – логические структуры БД.

\*объекты нижнего ур-ня – физические структуры БД.

Логическая структура состоит из пользовательских компонентов: табличное пространство, сегмент, экстент и блок данных.

**Табличное пространство** - логическая структура хранения данных, контейнер сегментов.

**Сегмент** - область на диске, выделяемая под объекты

**Экстент** - непрерывный фрагмент дисковой памяти

**Блок** - минимальная единица объема памяти, применяемая при записи и чтении данных

### Блоки данных (Data Block)

**Блоки данных (Data Block)** - мельчайший строительный блок базы данных Oracle, состоящий из определенного количества байт на диске. Блок данных Oracle - логический компонент базы данных. Диски на которых располагаются блоки Oracle, сами делятся на блоки данных. Обычно блоки данных диска соответствуют блокам данных Oracle. Размер блока базы данных Oracle устанавливается параметром DB\_BLOCK\_SIZE в файле init.ora. Размер блока следует воспринимать, как минимальную единицу обновления, выбора или вставки данных. Общепринятый размер блока - 8 KByte. Если выбрать размер блока 64 KByte, то даже при извлечении имени длиной в четыре символа, придется прочесть весь блок размером 64 KByte, в котором содержатся интересующие четыре буквы.

Все блоки данных можно разделить на две основные части: часть строк данных и часть свободного пространства.

### Экстенты (extent)

Экстенты (extent) - это два или более последовательных блоков данных Oracle, представляющий собой единицу выделения места на диске. Когда комбинируется несколько непрерывных блоков данных, они называются экстентом. Когда создается объект базы данных вроде таблицы или индекса, выделяется им некоторый начальный объем пространства, называемый начальным экстентом, и, указывается размер следующего экстента.

### Сегмент:

Сегменты (segments) - набор экстентов, которые выделяются логической структуре, такой как таблица или индекс (или некоторый другой объект). Oracle называет сегментом все пространство, выделенное любому конкретному объекту базы данных. Есть также временные сегменты, которые база данных использует в транзакциях, включающих сортировку, а также сегменты отката, которые база использует для хранения информации отката. Когда все экстенты сегмента заполнены, Oracle автоматически выделяет дополнительные экстенты при необходимости и эти сегменты могут быть непрерывными.

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

**Физическая организация БД** - это совокупность методов и средств размещения данных во внешней памяти и созданная на их основе внутренняя (физическая) модель данных. В отличие от логических моделей, физическая модель данных связана со способами организации данных на носителях методами доступа к данным. Эта модель указывает каким образом записи размещаются в БД, как они упорядочиваются, как организуются связи, как локализовать записи и осуществить выборку. Внутренняя модель разрабатывается средствами СУБД.

Все компоненты физического уровеня БД могут быть разделены на 2 большие группы:

* системные объекты, используемые внутри системы и необходимы для выполнения функций СУБД
  + создаются и настраиваются админом БД и недоступны юзеру
  + файл параметров, трассировки, журнал повтора и архивы
* объекты юзера
  + файлы данных

# 11. Абстрактная модель Oracle. Структура SGA.

***При запуске экземпляра:***

* выделяется область разделяемой памяти, называемая SGA
* запускаются фоновые процессы

**Процесс** – это механизм в операционной система, который может выполнить последовательность шагов.

**SGA** – системная глобальная область;

**PGA** – глобальные программные области;

***SGA:***

группа областей разделяемой памяти

содержат данные и управляющую информацию для одного экземпляра Oracle

совместно используется всеми серверными и фоновыми

группа областей разделяемой памяти

содержат данные и управляющую информацию для одного экземпляра Oracle

совместно используется всеми серверными и фоновыми

### Структура SGA

**Разделяемый пул** - используется в базе данных Oracle для хранения: информации о последних выполненных командах SQL, информации из словаря данных, к которая недавно запрашивалась

#### Поддержка словаря данных

* Кэш словаря данных – содержит данные в виде строк
* Библиотечный кэш
* Все пользовательские процессы Oracle совместно используют эти два кэша для доступа к информации о словаре данных
* Каждый SQL-оператор распознается
* Для нового SQL-оператора выделяется память в разделяемом пуле
* Когда два пользователя выполняют одно и то же предложение SQL, оператор используется повторно из разделяемой области SQL
* Аналогично обрабатывается программные блоки

Большой пул - область памяти SGA, применяемая для хранения больших фрагментов памяти; в этой области не применяется вытеснение по алгоритму LRU; память становится свободной сразу после того, как перестает использоваться; аналог RECYCLE (разделяемый пул – KEEP); хранятся данные при резервном копировании (RMAN), специальные области UGA

**Фиксированная область SGA** - хранит переменные, указывающие на другие области памяти, значения параметров; представляет собой загрузочный бинарный код; размер области зависит от платформы, версии операционной системы; размером фиксированной области SGA управлять нельзя.

**Пул Java** - предназначен для работы Java-машины

**Буферный пул** - содержит образы блоков, считанные из файлов данных или созданные динамически, чтобы реализовать модель согласованного чтения.

Буферный пул (буферный кэш) хранит блоки данных табличных пространств

Блок – единица обмена информацией между оперативной памятью и диском

Поддерживается два списка блоков:

список грязных блоков (отличаются от своей копии на диске и должны быть записаны в табличное пространство)

список чистых блоков (не измененные блоки)

**Алгоритм LRU (least recently used)**– первыми вытесняются блоки с наименьшим значением счетчика

Запись грязных блоков на диск осуществляется в 4х случаях:

1) истечение тайм-аута (3 сек);

2) контрольная точка;

3) превышение длины грязных блоков заданного лимита;

4) процесс не может обнаружить свободный блок.

**Буфер журналов повтора** - предназначен для временного хранения циклического хранения данных журнала повтора

Позволяет ускорить работу сервера за счет буферизации

Содержимое сбрасывается на диск (в журнал повтора) в 4-х случаях:

1) каждые три секунды;

2) при фиксации транзакции;

3) при заполнении буфера на 1/3;

4) если в буфере более 1m данных журнала повтора.

### Основные пулы области SGA

java pool

large pool

shared pool

streams pool

“null” pool

Память различным пулам в SGA выделяется блоками – гранулами (наименьшая единица выделения памяти. Одна гранула (granule) - это область памяти размером 4, 8 или 16 Мбайт)

# 12. Абстрактная модель Oracle. Серверные процессы Oracle.

**Процесс (process)** – механизм ОС Windows, осуществляющий запуск и выполнение приложений:

* процесс создаётся, когда запускается приложение
* в общем случае выполняется в собственной области памяти

**Поток (thread)** – индивидуальная ветвь внутри процесса, выполняющая конкретные программные инструкции.

**Серверные процессы** – процессы, выполняющиеся на основании клиентского запроса

Клиентский процесс прямо взаимодействует с соответствующим серверным процессом по сети

Первичное установление соединения происходит через специальный серверный процесс Oracle Net Listener

Серверный процесс:

Получает и выполняет SQL-операторы

Читает файлы данных

Осуществляет поиск в кеше

Назначение — отвечать на получаемые SQL-запросы

Клиентом может быть любая программа, пытающаяся подключиться к СУБД

### Отправка SQL-запроса select \* from students;

Производит синтаксический разбор

Помещает в разделяемый пул (находит в пуле)

Создает план запроса и выполняет его

При необходимости, производит поиск данных в буферном кэше или запрашивает в кэш с диска

**Oracle Net Listener** – процесс на стороне сервера, прослушивающий входящие запросы клиента на соединение с экземпляром.

**Listener** – это программа-сервер, прослушивающая TCP-порт, принимающая запросы на соединение с Oracle экземпляром от программ-клиентов.

В результате успешной работы Listener устанавливается соединение между программой-клиентом и обработчиком запросов экземпляра.

По умолчанию TCP-порт 1521

Экземпляр может иметь несколько точек подключения

Точки подключения называются сервисами и имеют символические имена

#### При инсталляции автоматически создается два сервиса:

SYS$USERS (по умолчанию, указывается SID в параметрах соединения),

сервис с именем инстанса (указывается сервис)

При создании PDB для нее автоматически добавляется сервис с именем, совпадающим с PDB-именем

**lsnrctl** – утилита управления процессом Listener

# 13. Абстрактная модель Oracle. Фоновые процессы Oracle.

**Фоновые процессы** – запускаются вместе с базой данных и выполняют разнообразные задачи обслуживания.

**Фоновые (background) процессы** – специальная группа процессов для обеспечения производительности и поддержки работы большого числа пользователей

**LREG** - **Listener Registration Process** – периодическая регистрация сервисов в процессе Listener

**DBWn** - **Database Writer Process**: (n=0,…,9, a,…,z; BWm, m=36,…,99) – фоновый процесс записывающий по LRU(**least recently used**– первыми вытесняются блоки с наименьшим значением счетчика) измененные блоки (грязные блоки) в файлы базы данных.

* В большинстве случаев достаточно одного процесса
* Если процессор 1, то в более чем в 1 DBW нет смысла
* Проверяет с периодичностью не менее 3 сек. наличие измененных блоков и инициирует checkpoint и по команде фонового процесса CKPT сбрасывает некоторую их часть на диск
* Использует асинхронный ввод-вывод для записи блоков на диск

**CKPT** выполняет процесс checkpoint

* Выполняется при shutdown, alter system checkpoint, переключении REDO – журнала, периодическом сообщение от DWR, backup
* записывает информацию о контрольной точке в управляющие файлы
* дает команду DBW на сброс буферов
* дает команду LGWR на сброс буферов
* инициирует создание контрольной точки
* изменяет управляющие файлы (последний SCN)
* изменяет заголовки файлов
* сигнализирует DBWR о необходимости сбросить буферы

**LGWR** - **Log Writer Process** – только один управляет буфером журналов повтора

* записывает блоки буфера журналов повтора в группы журналов
* записывает изменения базы данных до их фиксации DBWR в базе данных
* переключает текущую группу:
  + Раз в 3 секунды
  + При фиксации транзакции
  + При заполнении буфера на 1/3 или записи в него 1Мб
* Если по какой-то причине LGWR не может записать данные в один из файлов группы, он продолжает работать (информация об этом событии в alert)
* Если файл в группе один или нет возможности записать ни в один файл группы, то экземпляр остановится
* Производительность LGWR является критической для экземпляра
* Инициирует создание контрольных точек
* Чем чаще контрольные точки, тем меньше времени потребуется при восстановлении базы данных при сбое

**ARCn** - **Archiver Process** – копирует файлы журнала повтора после переключения группы журналов (необязательный процесс)

**PMON** - **Process monitor** – отвечает за очистку после ненормального закрытия подключений;

* Инициирует откат незафиксированных транзакций, снятие блокировок, и освобождение ресурсов SGA
* следит за работой других фоновых процессов, отвечает за их перезапуск
* восстанавливает работу dispatcher или shadow процессов при системном сбое

**SMON** - **System Monitor Process** – системный монитор

* Восстановление экземпляра для узла
* Восстановление незавершенных транзакций
* Очистка временных сегментов данных
* Очистка временных табличных пространств
* Объединение свободного пространства
* Очистка таблицы OBJ
* Сжатие сегментов отката

**RECO** - **Recovery Process** – разрешение проблем связанных с распределенными транзакциями

* Одна из баз данных, к которой первоначально подключился клиент, становится координатором
* Сервер опрашивает остальные N -1 серверов, готовы ли они фиксировать транзакцию
* Каждый из N -1 серверов сообщает о своем состоянии готовности как да (YES) или нет (NO)
* Если любой из серверов вернул NO, вся транзакция откатывается
* Если все серверы вернули YES, координатор рассылает всем N - 1 серверам сообщение о постоянной фиксации
* Если серверы ответили YES и происходит сбой сети транзакция становится сомнительной (in-doubt) распределенной транзакцией
* Протокол обработки распределенных транзакций не может полностью предотвратить сомнительные транзакции
* Остальную обработку сомнительной транзакции выполняет процесс RECO
* Транзакция остается незафиксированной
* RECO может восстановить либо откатить транзакцию, связавшись с координатором
* Если процесс RECO не может связаться с координатором, то необходимо зафиксировать или откатить транзакции вручную

**FBDA** - **Flashback Data Archiever** – архивирование ретроспективных данных

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

**Oracle Net Listener** – процесс на стороне сервера, прослушивающий входящие запросы клиента на соединение с экземпляром.

**Listener** – это программа-сервер, прослушивающая TCP-порт, принимающая запросы на соединение с Oracle экземпляром от программ-клиентов.

* В результате успешной работы Listener устанавливается соединение между программой-клиентом и обработчиком запросов экземпляра.
* По умолчанию TCP-порт 1521
* Экземпляр может иметь несколько точек подключения
* Точки подключения называются сервисами и имеют символические имена
* При инсталляции автоматически создается два сервиса:
  + SYS$USERS (по умолчанию, указывается SID в параметрах соединения),
  + сервис с именем инстанса (указывается сервис)
* При создании PDB для нее автоматически добавляется сервис с именем, совпадающим с PDB-именем
* select name, network\_name, pdb from v$services;
* фоновый процесс LREG считывает имена и параметры зарегистрированных сервисов экземпляра
* регистрирует их в Listener
* если Listener не обнаружен, то попытки регистрации периодично повторяются
* Listener прослушивает запросы к сервисам

#### OracleNetListener – запрос на соединение

1. клиент выполняет запрос к Listener на соединение с сервисом экземпляра
2. Listener запрашивает соединение с сервером
3. сервер возвращает параметры соединения с обработчиком сервиса
4. Listener сообщает параметры соединения клиенту
5. клиент соединяется с обработчиком запросов сервиса для дальнейшей работы с сервисом в рамках соединения

* Конфигурационный файл программы Listener: ORACLE\_HOME\NETWORK\ADMIN\listener.ora (Файл считывается при старте Listener)
* ***lsnrctl*** – утилита управления процессом Listener

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

Oracle поддерживает 2 режима соединений: **dedicated(выделенный)** и **shared(разделяемый) server.**

**Shared** – обрабатывает программа dispatcher:

1. получает запрос от клиента,
2. помещает их во входную очередь к разделяемым серверам;
3. незанятый разделяемый сервер извлекает и обрабатывает запрос;
4. после обработки разделяемый сервер помещает результат обработки в выходную очередь;
5. из очереди результат извлекает диспетчер;
6. диспетчер пересылает результат клиенту

**Dedicated**(режим по умолчанию) – для каждого клиента выделяется отдельный выделенный серверный процесс (обработчик запросов, dedicated server process) который называется shadow process (теневой процесс).

За подключение отвечает 2 конфигурационных файла:

* listener.ora – для управления прослушивателем
* tnsnames.ora – подробное описание подключений к БД

**Oracle Net Services** – набор служб, которые устанавливают подключение между сервером БД и пользователями БД

**Oracle Net** – программный компонент, который инициализирует, устанавливает и поддерживает подключения между клиентом и сервером.

Должен быть установлен и на клиенте, и на сервере.

Состоит из двух компонентов:

***Oracle Network Foundation layer*** – отвечает за установку и поддержание подключений между клиентским приложением и сервером.

***Oracle Protocol Support*** – отвечает за отображение функциональности TNS (Transparent Network Substrate) на стандартные протоколы, используемые при подключении.

### Соединение клиента с сервером

Имена экземпляров (SID)

Имена служб – глобальные имена баз данных

Дескрипторы соединений - объединенная спецификация двух обязательных компонентов подключения к базе данных(Имени службы базы данных; Местоположения адреса базы данных)

Идентификаторы соединений

Строки соединений

### Виды подключений к Oracle

* Простое подключение – Basic - явно указываются все параметры соединения (CONNECT имя/пароль@[//]хост[:порт][/имя\_службы])
  + Должны быть установлены Oracle Net Services
  + Поддержка протокола TCP/IP – на сервере и клиенте
  + Нельзя использовать расширенные сетевые функциональные возможности Oracle
* Локальное именование – TNS
* LDAP-соединение - Метод именования с помощью службы каталогов(OID – Oracle Internet Directory; LDAP – Lightweight Directory Access Protocol) Требуется наличие специального LDAP-сервера
* Local/bequeath-соединение
  + Только на сервере
  + Можно соединяться с помощью sqlplus или sqldeveloper
  + без указания параметров соединения
  + только с выделенным сервером
  + Listener не задействован
  + Соединение со стандартным сервисом SYS$USERS
* Прочие

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

Базы данных Oracle логически делятся на одно или более табличных пространств.

**Табличное пространство Oracle** — это логическая сущность, содержащая физические базы данных. Табличные пространства хранят все доступные данные базы, и эти данные в табличных пространствах физически хранятся в одном или более файле данных(***файлы данных*** - это сформированные Oracle файлы операционной системы)

**Табличное пространство** – пограничный объект базы данных: с одной стороны Логическая структура сервера, с другой файл или файлы операционной системы. С одним табличным пространством связаны один или несколько файлов, с каждым файлом связано только одно табличное пространство.

Данные, временные данные, данные отката – организованы в виде табличных пространств.

## Предопределенные табличные пространства

**SYSTEM** используется для управления БД, содержит словарь базы данных, стандартные пакеты процедур.

**SYSAUX** – вспомогательное табличное пространство.

**TEMP** – временное табличное пространство по умолчанию.

**UNDOTBS1** – табличное пространство отката.

**USERS** – хранение пользовательских объектов и данных.

**EXAMPLE** – демонстрационные схемы.

**SELECT \* FROM DBA\_TABLESPACES**

## Виды табличных пространств

**PERMANENT** – предназначены для хранения постоянных объектов (таблиц, индексов, кластеров, ...)

**TEMPORARY** – для временных данных

**UNDO** – для отката

## Параметры табличного пространства

* **smallfile** – 1022 файлов, по 222 блоков
* **bigfile** – 1 файл, 128TB(блок 32К) или 32TB(блок 8К)
* **LOGGING / NOLOGGING / FORCE LOGGING**
* **online / offline** ALTER TABLESPACE *НАЗВАНИЕ\_ТАБЛИЧНОГО\_ПРОСТРАНСТВА* OFFLINE
* **Reuse**

### Создание таблиц

* Можно указать табличное пространство
* На это табличное пространство у пользователя создающего таблицу должна быть достаточная квота.
* Если tablespace не указан в create table, то таблица будет создана в табличном пространстве по умолчанию

### Табличное пространство для временных данных

Создается create temporary tablespace

Используются для хранения промежуточных результатов сортировки

Может быть одно общее для всех PDB (создается при инсталляции в CDB)

Может быть создано несколько

### Табличное пространство для отката UNDO

Создается create undo tablespace

Может быть несколько, но используется только одно

Указывается в файле параметров экземпляра

# 17. Роли и привилегии СУБД Oracle и их основные параметры.

**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**

| **системные** | **объектные** |
| --- | --- |
| **изменение системы: create table, create trigger** | **на изменение объекта: select** |
| **WITH ADMIN OPTION - дают право юзеру назначать/отбирать привилегии: alter, analyze, audit, backup, create, drop, select(стараться редко использовать)**  **any - для любого объекта, all - для всех объектов** | **WITH GRANT OPTION - дают право юзеру назначать/отбирать привилегии(но объектные): alter, delete, execute, insert, update, select, references**  **снимает правила тот кто их назначил** |
| **объекты грантов:**  **database, user, profile, tablespace, role, table, index, trigger, procedure, sequence, view** | **объекты грантов:**  **table, view, sequence, procedure** |

**Роль** – это именованный набор привилегий

**Роль** – это не объект схемы: у них нет владельца и поэтому нет префикса владельца. Но они используют ту же область хранения как и пользователи: поэтому невозможно создать роль с именем уже существующего пользователя и наоборот.

Для создания роли используется команды

CREATE ROLE rolename;

Затем права назначаются для роли используя обычные синтаксис команд включая WITH ADMIN или WITH GRANT OPTION если нобходимо

#### Предопределённые роли

Существует как минимум 50 предопределенных ролей в БД (может быть и больше в зависимости от установки). Роли которые должен знать каждый DBA это

**CONNECT** – эта роль существует только для обратной совместимости. Раньше эта роль содержала системные привилегии необходимые для хранения объектов. Теперь у этой роли только CREATE SESSION права.

**RESOURCES** – также используется для обратной совместимости. Эта роль может создавать объекты и PL/SQL объекты. Также эта роль включает UNLIMITED TABLESPACE права.

**DBA** – роль с наибольшим количеством системных привилегий, и некоторыми объектными правами и ролями. Любой пользователь с ролью DBA может управлять работой БД, за исключением команд STARTUP и SHUTDOWN.

**SELECT\_CATALOG\_ROLE** – роль имеет более двух тысяч объектных прав над словарем данных, но не имеет системных привилегий и привилегий над пользовательскими данными. Используется для младшего административного персонала к примеру выполнения задач мониторинга и отчетности.

**SCHEDULER\_ADMIN** – роль владеет системными правами необходимыми для управления выполнением задач по расписанию.

Также существует предопределенная роль **PUBLIC**, которая назначается по умолчанию всем аккаунтам БД. Отсюда и следует что если права назначены для PUBLIC, то эти права автоматически доступны всем пользователям. После выполнение команды

grant select on hr.regions to public;

все пользователи смогут выполнять select к таблице HR.REGIONS.

# 18. Пользователь СУБД Oracle и его основные параметры.

**Юзер** – физическое/юридическое лицо, которое имеет доступ к БД и пользуется услугами для получения информации

Оператор CREATE USER создает учетную запись базы данных, которая позволяет войти в базу данных Oracle.

create user ***user\_name***

identified by 12345

default tablespace .. quota unlimited on ..

temporary tablespace ..

profile ***profile\_name***

account ***unlock/lock***

***password expire***

***user\_name***

Имя учетной записи базы данных, которую вы хотите создать

***profile\_name(необязательный)***

Это имя профиля, который вы хотите назначить для учетной записи пользователя, чтобы ограничить объем ресурсов базы данных, присвоенных учетной записи пользователя. Если опустить этот параметр, профиль по умолчанию присваивается пользователю.

***PASSWORD EXPIRE(необязательный)***

Если этот параметр установлен, то пароль должен быть сброшен до того, как пользователь может войти в базу данных Oracle

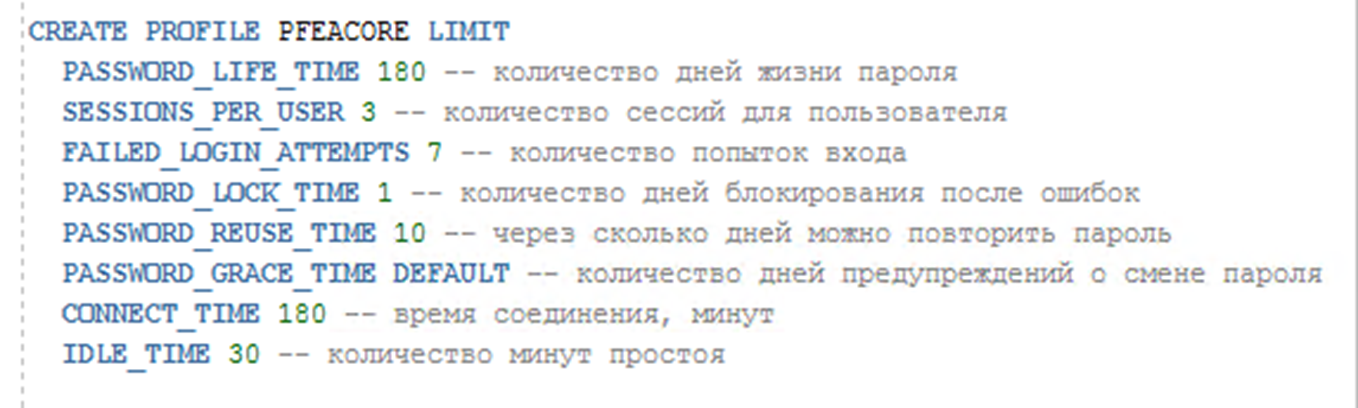
***LOCK(необязательный)***

Он отключает доступ к учетной записи пользователя

***UNLOCK(необязательный)***

Включает доступ к учетной записи пользователя

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

**Профиль** – коллекция атрибутов, связанных с использованием ресурсов и паролей, которые могут быть назначена юзеру

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

**SQL** *(structured query language — «язык структурированных запросов»)* — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

## Операторы языка делятся на группы:

**DDL** - к этому типу относятся различные команды, которые создают базу данных, таблицы, индексы, хранимые процедуры и т.д. В общем определяют данные.

* **CREATE**: создает объекты базы данных (саму базу данных, таблицы, индексы и т.д.)
* **ALTER**: изменяет объекты базы данных
* **DROP**: удаляет объекты базы данных
* **TRUNCATE**: удаляет все данные из таблиц

**DML** - к этому типу относят команды по выбору данных, их обновление, добавление, удаление - в общем все те команды, с помощью которыми мы можем управлять данными.

* **SELECT**: извлекает данные из БД
* **UPDATE**: обновляет данные
* **INSERT**: добавляет новые данные
* **DELETE**: удаляет данные

**DCL** - этому типу относят команды, которые управляют правами по доступу к данным.

* **GRANT**: предоставляет права для доступа к данным
* **REVOKE**: отзывает права на доступ к данным
* **DENY**: задаёт запрет, имеющий приоритет над разрешением.

**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.

* **BEGIN TRANSACTION** – служит для определения начала транзакции;
* **COMMIT TRANSACTION** – применяет транзакцию;
* **ROLLBACK TRANSACTION** – откатывает все изменения, сделанные в контексте текущей транзакции;
* **SAVE TRANSACTION** – устанавливает промежуточную точку сохранения внутри транзакции.

# 21. Таблица и ее основные параметры.

**Таблица** — это совокупность связанных данных, хранящихся в структурированном виде в [базе данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Она состоит из столбцов и строк.

Одно из важнейших достоинств реляционных баз данных состоит в том, что можно хранить логически сгруппированные данные в разных таблицах и задавать *связи* между ними, объединяя их в единую базу. Такая организация данных позволяет уменьшить избыточность хранимых данных, упрощает их ввод и организацию запросов и отчетов.

## Типы таблиц:

* Традиционные таблицы (heap organized table)
* Индекс-таблицы (index organized table)
* Кластеризованные индекс-таблицы (index clustered table)
* Кластеризованные хэш-таблицы (hash clustered table)
* Отсортированные кластеризованные хэш-таблицы (sorted hash clustered table)
* Вложенные таблицы (nested table)
* Временные таблицы (temporary table)
* Объектные таблицы
* Внешние таблицы

Может иметь до 1000 столбцов (<254)

Может иметь неограниченное число строк

Может иметь неограниченное число индексов

Нет ограничения на число таблиц

## Нормализация

Процесс нормализации состоит в приведении таблиц РБД к т.н. нормальным формам. Всего существует 6 нормальных форм

Оптимальная структура БД достигается при выполнении первых 3 правил нормализации

### 1-я нормальная форма

Все значения полей должны быть неделимыми и невычисляемыми, а все записи – уникальными

### 2-я нормальная форма

Необходимо, чтобы таблица находилась в 1-й нормальной форме и все неключевые поля полностью зависели от ключевого

### 3-я нормальная форма

Необходимо, чтобы все таблицы БД находилась во 2-й нормальной форме и все неключевые поля в таблицах зависели только от ключа таблицы и не зависели друг от друга.

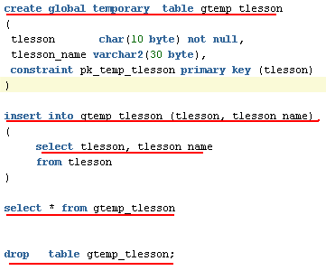
## Параметры PCTFREE и PCTUSED

* **Параметр PCTFREE** – процент памяти блока, резервируемой для возможных обновлений строк, уже содержащихся в блоке
* **Параметр PCTUSED** – процент занятой части памяти блока

# 22. Временные таблицы.

**Временные таблицы** – механизм хранения данных в БД

* Состоит из столбцов и строк, как и обычная таблица
* Временные таблицы – глобальны
* Привилегии для создания временной таблицы CREATE TABLE
* Можно разместить временную таблицу в заданном табличном пространстве.
* Временные таблицы – это шаблон, хранящийся в словаре базы данных, для нее выделяется временный сегмент в (по умолчанию) TEMPORARY-табличном пространстве и для каждого пользователя свой.
* Каждый пользователь видит только свои данные (свой сегмент данных).
* Статичны: временные таблицы создаются (CREATE) один раз и существуют, пока их не удалят (DROP).
* DROP не получится, если таблица в этот момент используется другим пользователем.
* Временные таблицы бывают:
  + ***ON COMMIT PRESERVE ROWS*** – на время сеанса, данные существуют только на время сеанса, возможны все DML-операторы, TCL-операторы
  + ***ON COMMIT DELETE ROWS*** – на время транзакции, данные существуют только на время транзакции, возможны все DML-операторы, после выполнения COMMIT или ROLLBACK таблица становится пустой
* В начале сеанса временная таблица всегда пуста
* Для временных таблиц можно создавать триггеры
* Для временных таблиц можно указать констрейны (ограничения)
* Для временных таблиц можно создавать индексы
* Не могут быть индексно-организованными, нельзя секционировать, размещать в кластере.
* Данные повторного выполнения генерируются, но их количество пренебрежительно мало.



# 23. Ограничения целостности в таблицах.

**Ограничения целостности** можно определить как специальные средства в базах данных, главное назначение которых - не дать попасть в базу недопустимым данным (например, предупредить ошибки пользователей при вводе данных).

**PRIMARY KEY** - столбец можно сделать первичным ключом

**FOREIGN KEY** - Устанавливает связь между таблицей со столбцом, имеющим свойство foreign key и таблицей, имеющей столбец со свойством primary key

**IDENTITY** - позволяет сделать столбец идентификатором. Этот атрибут может назначаться для столбцов числовых типов

**UNIQUE** - чтобы столбец имел только уникальные значения

**NULL и NOT NULL**- чтобы указать, может ли столбец принимать значение NULL

**DEFAULT** - определяет значение по умолчанию для столбца

**CHECK** - задает ограничение для диапазона значений, которые могут храниться в столбце

**DATA TYPE** - Предотвращает появление в столбце значений, не соответствующих типу данных

# 24. Типы данных базы данных.

CHAR / NCHAR

VARCHAR2 / NVARCHAR2

DATE

INTERVAL DAY TO SECOND / INTERVAL YEAR TO MONTH TIMESTAMP

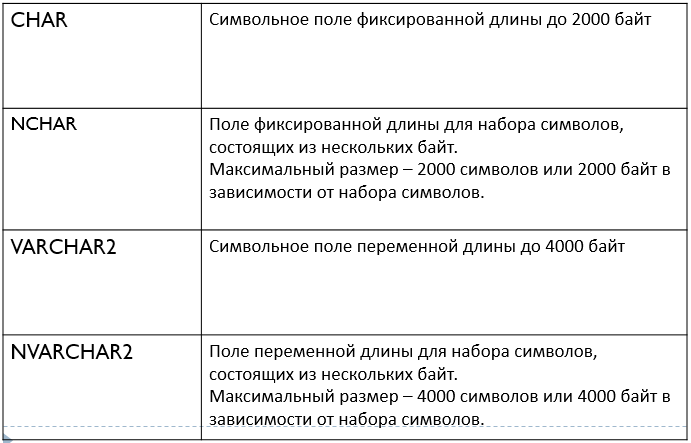
TIMESTAMP WITH TIME ZONE / TIMESTAMP WITH LOCAL TIME

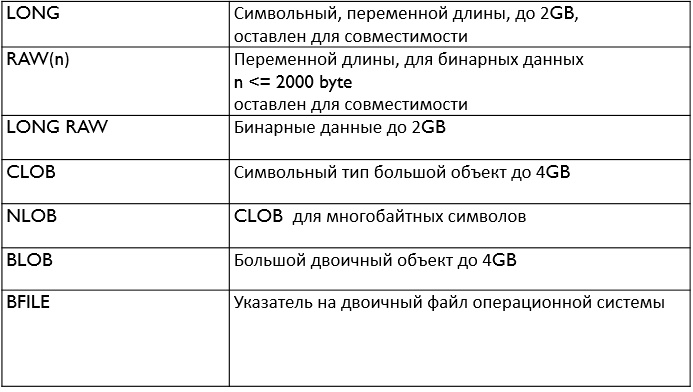
NUMBER (A,B)

LONG RAW/ LONG / RAW

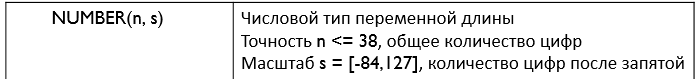
BLOB / CLOB / NCLOB

ROWID / UROWID









**ROWID** – 16-тиричный тип для уникального определения любой строки любой таблицы в БД

Длина – 18 символов, которая делится на 4 элемента

(6 + 3 + 6 + 3):

OOOOOO – уникальный номер объекта в БД, которому принадлежит строка;

FFF – уникальный номер файла данных БД, где хранится строка;

BBBBBB –номер блока данных, который хранит строку, уникален на уровне файла данных БД;

RRR – адрес строки в блоке

**UROWID** – специальный 16-тиричный тип данных для адресации строк в таблицах, организованных по индексу

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

**Индекс** – структура базы данных, используемая сервером для быстрого поиска строки в таблице

#### Типы индексов:

Табличный (B\*Tree) индекс

Битовый индекс

Функциональный индекс

Кластерный индекс

**Плотность запроса** – количество возвращаемых строк запроса

**Селективность запроса** – количество возвращаемых значений ключа (% разных ключей от общего количества)

***Хорошая селективность*** – если количество возвращаемых значений ключа меньше (<10%)

***Плохая селективность*** – если количество возвращаемых значений больше

**Табличный индекс (B\*Tree)** структурирован в виде сбалансированного дерева

Листовой блок содержит индексированные значения столбца и соответствующий ему идентификатор строки (RowId)

Предназначен для индексирования уникальных столбцов или столбцов с высокой селективностью

**Битовый индекс** создает битовые карты для каждого возможного значения столбца, где каждому биту соответствует строка, а значение бита 1 (0) означает, что соответствующая строка содержит (не содержит) индексируемое значение

Предназначен для индексирования столбцов с низкой селективностью

Не подходит для таблиц с частым обновлением

Хорошо подходят для хранилищ данных

**Функциональный индекс** – предварительно вычисляют значения функции по заданному столбцу и сохраняют результат в индексе

LOWER(NAME) / UPPER (NAME)

# 26. Последовательность СУБД Oracle и ее параметры.

**Последовательность** – объект базы данных, предназначенный для генерации числовой последовательности

**increment by** - увеличить на(интервал между порядковыми номерами)

**start with** - начать с(первый порядковый номер, который будет сгенерирован)

**nomaxvalue** - максимальное значение 10 27 для восходящей последовательности или -1 для нисходящей последовательности.

**maxvalue** - максимальное значение(если опустить параметр MAXVALUE, последовательность по умолчанию до:MAXVALUE 999999999999999999999999999)

**nominvalue** - минимальное значение 1 для восходящей последовательности или -10 26 для нисходящей последовательности

**minvalue** - минимальное значение(целочисленное значение может содержать 28 или меньше цифр)

**cycle** - последовательность продолжает генерировать значения после достижения максимального или минимального значения. После того, как восходящая последовательность достигает своего максимального значения, она генерирует свое минимальное значение. После того, как нисходящая последовательность достигает своего минимума, она генерирует свое максимальное значение.

**nocycle** - последовательность не может генерировать больше значений после достижения своего максимального или минимального значения. Это значение по умолчанию

**cache** - eказывается,, сколько значений последовательности предварительно выделяет база данных и хранит в памяти для более быстрого доступа. Это целочисленное значение может содержать 28 или меньше цифр. Минимальное значение этого параметра равно 2. Для циклически повторяющихся последовательностей это значение должно быть меньше числа значений в цикле. Вы не можете кэшировать больше значений, чем поместится в данном цикле порядковых номеров.

**nocache** - значения последовательности не назначаются заранее. Если опустить оба параметра CACHE и NOCACHE, база данных по умолчанию кэширует 20 порядковых номеров

**order** - гарантировать, что порядковые номера генерируются в порядке запроса. Это предложение полезно, если вы используете порядковые номера в качестве меток времени. Порядок гарантии обычно не важен для последовательностей, используемых для генерации первичных ключей. ORDER необходим только для гарантии упорядоченной генерации, если вы используете базу данных Oracle с реальными кластерами приложений. Если вы используете эксклюзивный режим, порядковые номера всегда генерируются по порядку.

**noorder** - порядковые номера генерировались в порядке запроса. Это значение по умолчанию

Привилегия **CREATE SEQUENCE**

# 27. Кластер и его параметры

**Кластер** - специализированный объект [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), используемый для физически совместного хранения одной или нескольких [таблиц](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0_(%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)), которые часто соединяются вместе в [SQL](https://ru.wikipedia.org/wiki/SQL)-запросах

**Кластер** – объект БД, который хранит значения общих столбцов нескольких таблиц

***Создание CREATE CLUSTER***

***Привилегия CREATE CLUSTER***

Таблицы, с которыми часто работают совместно, можно физически хранить совместно.

Для этого создается кластер, который будет их содержать

Строки из отдельных таблиц сохраняются в одних и тех же блоках, поэтому объединяющие запросы выполняются быстрее

Уменьшается количество операций ввода-вывода

Производительность операций вставки, обновления и удаления может быть ниже, чем для обычных таблиц

Связанные столбцы называются кластерным ключом

**Хэш-кластеры** используют функции хэширования кластерного ключа строки для определения физической локализации места, где строку следует хранить

Наибольшие преимущества – в запросах, использующих операции равенства:

select Name from STUDENT where Id = 999;

# 28. Представление и его параметры.

**Представление (view)**, или виртуальная таблица в базе данных Oracle Database — это специфический образ таблицы или набора таблиц, определенный оператором SELECT.

**Представление (VIEW)** - объект данных который не содержит никаких данных его владельца. Это - тип таблицы, чье содержание выбирается из других таблиц с помощью выполнения запроса.

**Представление Oracle** — результат хранимого запроса, поэтому в словаре данных сохраняется только определение представления.

**Представление** – хранимый запрос

* Можно обращаться, как к обычной таблице
* Данные хранятся в таблице
* Добавляют уровень защиты данных
* Скрывают сложность данных
* Скрывают имена столбцов таблиц

**Привилегия** – CREATE VIEW

**Создание** – CREATE (OR REPLACE) VIEW

***FORCE*** – создает представление, независимо от того, существуют ли таблицы и есть ли права

***NOFORCE*** – по умолчанию

***WITH CHECK OPTION*** – указывает, что будут вставлены или изменены строки, которые будут выбираться через это представление

***READ ONLY***

### Материализованные представления

**Привилегия** – CREATE MATERIALIZED VIEW

**Создание** – 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

# 29. Материализованное представление и его параметры.

**Материализованное представление** – хранят некоторые данные, которые обновляются в соответствии с указанным промежутком.

ОТЛИЧИЕ: хранят данные, в то время как обычные представления хранят запросы.

**Привилегия** – CREATE MATERIALIZED VIEW

**Создание** – 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

Далее – в разницу времени между START WITH и NEXT

# 30. Частные и публичные синонимы СУБД Oracle.

**Синоним** – способ обращаться к объекту базы данных без указания обязательной полной идентификации объекта (хост – экземпляр – владелец – объект).

***Частный синоним*** принадлежит пользователю, который его создал.

***Публичный синоним*** используется совместно всеми пользователями базы данных.

**Привилегия** – CREATE (PUBLIC) SYNONYM

**Создание** – CREATE (PUBLIC) SYNONYM

Допустимость синонима не проверяется сервером при создании!

Представление словаря dba.synonyms

***Может указывать на:***

* Таблицы,
* Процедуры,
* Функции,
* Последовательности,
* Представления
* Пакеты
* Объекты в локальной или удаленной базе данных

# 31. Основные характеристики языка PL/SQL.

Procedural Language extensions to SQL - [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), [процедурное](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) расширение языка [SQL](https://ru.wikipedia.org/wiki/SQL), разработанное корпорацией [Oracle](https://ru.wikipedia.org/wiki/Oracle_Corporation). Базируется на языке [Ад](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))а.

Основной язык для программирования хранимых процедур (stored procedures);

Интегрирован с базой данных Oracle;

Производительность серверных модулей;

Приложение может быть проще в реализации при написании бизнес-логики на основе хранимых процедур;

Отсутствие накладных расходов на приведение типов;

Может выполняться независимо от пользователя;

PL/SQL-функции можно вызывать из SELECT запросов

Взаимодействие с пользователем (user interaction);

Внутренний язык (proprietary for Oracle);

Cодержит элементы объектно-ориентированного программирования;

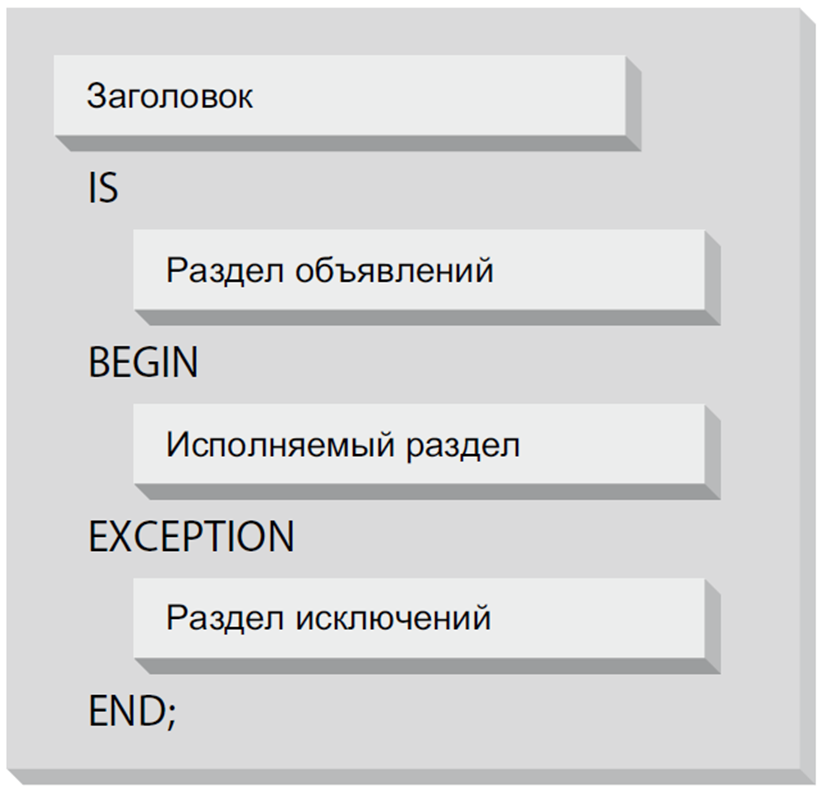
Позволяет использовать объектные типы;

Интерпретация (режим по умолчанию);

Компиляция (промежуточный код на C и конечный объектный код процессора);

Среда выполнения: SQL\*Plus, SQL Developer, TOAD.

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



Блок PL/SQL состоит из четырех разделов:

* раздел заголовка; (может не быть)
* раздел объявлений; (может не быть)
* исполняемый раздел;
* раздел обработки исключений. (может не быть)

Синтаксически блок PL/SQL выглядит:

раздел заголовка

**DECLARE**

раздел объявлений

**BEGIN**

исполняемый раздел

**EXCEPTION**

раздел обработки исключений

**END**;

В разделе заголовка указываются:

тип блока (процедура, функция);

имя блока (имя процедуры, функции);

имена параметров, их типы данных и режимы передачи значений.

## Анонимный блок

Не имеет секции заголовка

Используется как скрипт для выполнения PL/SQL выражений

Не может быть вызван из другого блока

Начинается с DECLARE или BEGIN

Варианты использования:

Триггер на стороне клиента (Oracle Development Tools)

Триггер базы данных (содержит АБ)

SQL-скрипт (описание процедур, функций и execute)

Откомпилированная программа (блок в execute команде, выполняющейся на сервере)

## Именованный блок

Именованные блоки - может быть процедурой, функцией или триггером

### Секция обработки исключений

Функция SQLERRM возвращает сообщение об ошибке, связанной с исключительной ситуацией

Функция SQLCODE возвращает номер ошибки, связанной с исключительной ситуацией

Могут быть использована только в разделе обработки исключений

Не имеют параметров или аргументов

Может содержать столько блоков WHEN, сколько выделяется обрабатываемых исключений

Остальные – в WHEN OTHERS

Можно определять свои исключения

## Вложенные блоки

Область действия (scope) – переменные, исключения, модули – локальны в рамках блока;

Область видимости – в текущем блоке;

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

## Символьные

**CHAR** - Символьное поле фиксированной длины до 2000 байт

**NCHAR** - Поле фиксированной длины для набора символов, состоящих из нескольких байт. Максимальный размер – 2000 символов или 2000 байт в зависимости от набора символов.

**VARCHAR2** - Символьное поле переменной длины до 4000 байт

**NVARCHAR2** - Поле переменной длины для набора символов, состоящих из нескольких байт. Максимальный размер – 4000 символов или 4000 байт в зависимости от набора символов.

**LONG** - Символьный, переменной длины, до 2GB, оставлен для совместимости

**RAW(n)** - Переменной длины, для бинарных данных n <= 2000 byte оставлен для совместимости

**LONG RAW** - Бинарные данные до 2GB

**CLOB** - Символьный тип большой объект до 4GB

**NLOB** - CLOB для многобайтных символов

**BLOB** - Большой двоичный объект до 4GB

**BFILE** - Указатель на двоичный файл операционной системы

## Дата/Время

**DATE** - 7 байтовое поле фиксированной длины, используемое для хранения даты и времени

**INTERVAL(DAY TO SECOND)** - 11 байтовое поле фиксированной длины для интервала времени: Дни, часы, минуты, секунды

**INTERVAL (YEAR TO MONTH TIMESTAMP)** - 5 байтовое поле фиксированной длины для интервала времени: Годы и месяцы

**TIMESTAMP (WITH TIME ZONE)** - 13 байтовое поле фиксированной длины Дата, время и настройки, связанные с часовым поясом.

**TIMESTAMP (WITH LOCAL TIME)** - 7-11 байтовое поле переменной длины Дата и время, приведенные к часовому поясу базы данных

## Числовые

**NUMBER(n, s)** - Числовой тип переменной длины. Точность n <= 38, общее количество цифр. Масштаб s = [-84,127], количество цифр после запятой.

## Типы данных

* Скалярные (Scalar)
  + символ/строка
  + число
  + булев
  + дата/время
* Ссылочные (Reference)
* Составные (Composite)
* Большие объекты (LOB)

## Константы

ключевое слово **constant**



# 34. Поддержка национальных языков в СУБД Oracle. Наборы символов. Байтовая и символьная семантика символов.

NLS - National Language Support, далее Globalization Support

Можно хранить данные множества национальных языков, используя Unicode или специальные кодировки – наборы символов (character set)

Символы хранятся как коды символов, зависящие от выбранного набора символов

В одной БД могут использоваться два набора символов: основной (database character set) и дополнительный (national character set)

Устанавливаются при создании БД

Изменяются alter database (national) character set

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.

## Переменная окружения NLS\_LANG:

**NLS\_LANG = language\_territory.charset**

**Язык (LANGUAGE)** – имена месяцев, имена дней, направление текста, сокращения для времени и дат. По умолчанию AMERICAN

**Территория (TERRITORY)** – настройки календаря, формат даты, формат денежной единицы. Если не указан, то будет взято значение, соответствующее языку (для RUSSIAN - CIS)

**Набор символов (CHARACTER SET)** – отображение символов, отображение и конвертация заглавных букв, порядок замещения символов при преобразовании. Каждому языку поставлен в соответствие набор символов по умолчанию

# 35. Связанные объявления переменных: инструкция %TYPE, инструкция %ROWTYPE.

Тип переменной основан на на известной структуре данных

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

**атрибут %TYPE** для переменной предоставляет тип данных столбца базы данных. Это особенно полезно при объявлении переменных, которые будут содержать значения столбцов таблиц базы данных.

#### Синтаксис объявления переменной с атрибутом %TYPE

v\_name table\_name.column\_name%TYPE

**v\_name** – имя переменной которой присваивается значение.

**table\_name** – имя таблицы базы данных.

**column\_name** – имя столбца таблицы table\_name.

* Для объявления переменной, вам не нужно знать фактический тип данных и такие атрибуты, как точность, размер или длина.
* Если точность в столбце изменяется, то тип данных переменной изменяется соответственно во время выполнения

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

атрибут **%ROWTYPE** предоставляет тип записи, представляющий строку в таблице (или представлении) базы данных Oracle. Запись может хранить целую строку данных, выбранных из таблицы, или извлекаться из курсора или строго типизированной переменной курсора.

#### Синтаксис объявления переменной с атрибутом %ROWTYPE

v\_rec table\_name%ROWTYPE

v\_rec - имя переменной которой присваивается значение записи

table\_name – имя таблицы базы данных

* Столбцы в строке и соответствующие поля в записи имеют одинаковые имена и типы данных.
* Поля в записи %ROWTYPE не наследуют ограничение столбца NOT NULL.

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

**Локальный (или вложенный) модуль** — это [процедура](https://oracle-patches.com/db/sql/3611-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D1%8B-pl-sql-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BD%D0%B0-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%D0%B0%D1%85) или [функция PL/SQL](https://oracle-patches.com/db/sql/3612-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8-pl-sql-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BD%D0%B0-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%D0%B0%D1%85), определяемая в разделе объявлений блока PL/SQL (анонимного или именованного). 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.

**Процедура** — это подпрограмма, которая выполняет специфическое действие (CREATE PROCEDURE).

**Функция** — это подпрограмма, которая вычисляет значение (CREATE FUNCTION).

1. Процедура называется блоком PL / SQL, который выполняет одну или несколько задач. где функция называется блоком PL / SQL, который выполняет определенное действие.
2. Процедура может возвращать или не возвращать значение, когда функция должна возвращать одно значение.

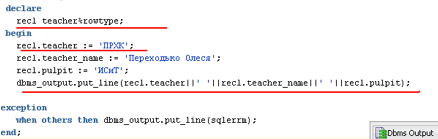
# 37. Использование записей в PL/SQL. Вложенные записи.

**Запись** – 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

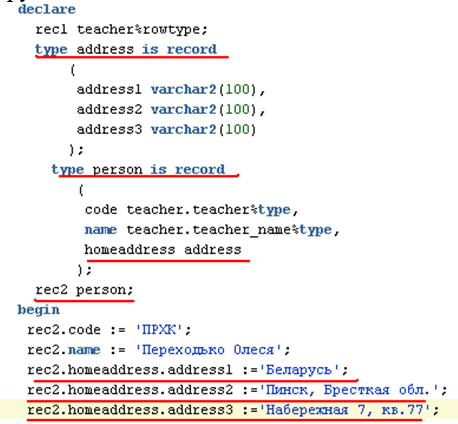


Сравнение производится по полям записи

Присвоение:

* Присвоение для отдельного поля
* SELECT INTO в запись в целом или в отдельные поля
* Присвоение значения одной записи другой записи – для одного и того же объявления TYPE

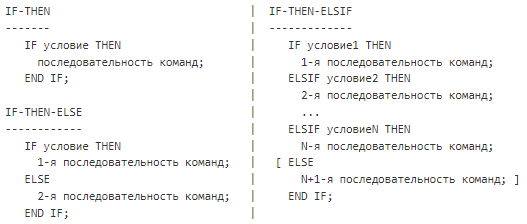
**Вложенные записи** – одно из полей внешней записи в действительности является полем другой записи.



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

## Операторы условного перехода (IF …)

Существует три модификации оператора условного перехода:



Во всех модификациях если «условие» или «условие1″ истинно (TRUE), то выполняется «последовательность команд» или «1-я последовательность команд» и управление передается на первый оператор после END IF. Если же оно ложно (FALSE), то:

* в модификации IF-THEN управление передается на первый оператор после END IF;
* в модификации IF-THEN-ELSE выполняется 2-я последовательность команд и управление передается на первый оператор после END IF;
* в модификации IF-THEN-ELSIF проверяется условие 2; если оно истинно, то выполняется 2-я последовательность команд и управление передается на первый оператор после END IF; если условия 1 и 2 ложны, а условие 3 истинно, то выполняется 3-я последовательность команд и управление передается на первый оператор после END IF; наконец, если условия 1, 2, …, N ложны, то выполняется N+1-я последовательность команд и управление передается на первый оператор после END IF.

Все это справедливо, если внутри последовательности команд нет операторов, осуществляющих переход за пределы этой последовательности.

## Операторы цикла (LOOP, WHILE…LOOP и FOR…LOOP)

Циклы служат для повторяемого выполнения последовательности команд. В PL/SQL используются три модификации операторов цикла: LOOP, WHILE…LOOP и FOR…LOOP.

Цикл LOOP имеет следующий синтаксис:



и приводит к бесконечному повторению последовательности команд, если внутри нее нет команд EXIT (выход из цикла), RAISE (вызов обработчика исключительных ситуаций) или GOTO (безусловный переход).

Цикл WHILE предназначен для повторения последовательности команд, пока условие остается истинным:



Наиболее распространен цикл FOR, имеющий следующий синтаксис:

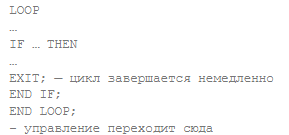


Здесь индекс (счетчик циклов) изменяется от нижней до верхней границы с шагом 1, а при использовании «REVERSE» – от верхней до нижней границы с шагом.

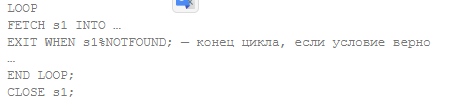
## Операторы EXIT, EXIT-WHEN и NULL

**EXIT** используется для завершения цикла, когда дальнейшая обработка нежелательна или невозможна. Внутри цикла можно помещать один или большее

количество операторов EXIT. Имеются две формы EXIT: EXIT и EXIT-WHEN



По оператору **EXIT-WHEN** цикл завершиться только в том случае, когда становится истинным условие в предложении WHEN. Оператор EXIT-WHEN позволяет завершать цикл преждевременно.



**NULL** – пустой оператор; он передает управление к следующему за ним оператору.

Однако, к нему может передаваться управление и его наличие часто улучшает читаемость программы.

оператор **CASE** имеет функциональность [IF-THEN-ELSE](https://oracleplsql.ru/if-then-else.html). Команда CASE позволяет выбрать для выполнения одну из нескольких последовательностей команд.

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

**Курсор** — объект БД, который позволяет приложениям работать с записями построчно

**Курсор Oracle** – указатель на область в PGA, в которой хранится:

1. строки запроса,
2. число строк,
3. указатель на разобранный запрос в общем пуле.

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

PL/SQL позволяет создавать 2 вида курсоров:

1. Статические курсоры, SQL выражение для которых определяется на этапе компиляции:

Используются для DML команд

Могут быть явно объявлены и именованы

1. Динамические курсоры, SQL выражение для которых определяется на этапе выполнения:

Могут использоваться для любых SQL выражений, включая DDL и DCL команды

Реализуются с помощью оператора EXECUTE IMMEDIATE или пакета dbms\_sql

#### Поддерживаются два типа курсоров:

**явный** — объявляется разработчиком

**неявный** — не требует объявления

Курсор может возвращать одну строку, несколько строк или ни одной строки

#### Операторы управления явным курсором

**DECLARE** — выполняет объявление явного курсора.

**OPEN** — открывает курсор, создавая новый результирующий набор на базе указанного запроса.

**FETCH** — выполняет последовательное извлечение строк из результирующего набора от начала до конца.

**CLOSE** — закрывает курсор и освобождает занимаемые им ресурсы.

#### Неявный курсор

* **Неявные курсор** – выполнение SQL выражения в секции исполнения или в секции исключений блока
* Операторы INSERT, UPDATE, DELETE, MERGE, SELECT INTO
* Не требуют объявления
* Не требуют OPEN, FETCH, CLOSE
* Когда неявный курсор не возвращает строк вообще, PL/SQL генерирует исключение NO\_DATA\_FOUND и передает управление в секцию исключений
* Когда SELECT возвращает более одной строки, PL/SQL генерирует исключение TOO\_MANY\_ROWS и также передает управление в секцию исключений
* SELECT INTO предназначен исключительно для того, чтобы возвращать ровно 1 строку – точную выборку

#### Ошибки неявного курсора:

- **no\_data\_found** – не возвращает строк вообще

- **too\_many\_rows** – вернуло более 1 строки

**select into** чтобы вернуть ровно 1 строку – точную выборку!

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

**Атрибут курсора** имеет форму %имя\_атрибута и добавляется к имени курсора или курсорной переменной. Это что-то вроде внутренней пере-менной Oracle, возвращающей информацию о состоянии курсора

**%ISOPEN** — возвращает значение TRUE, если курсор открыт

**%FOUND** — определяет, найдена ли строка, удовлетворяющая условию

**%NOTFOUND** — возвращает TRUE, если строка не найдена

**%ROWCOUNT** — возвращает номер текущей строки

Курсор, который принимает параметры

cursor cur (capacity int)

is select \* from aud where cap > capacity

begin

for aum in cur(80)

loop …output… end loop;

Параметризация курсоров помогает повысить степень их повторного использования.

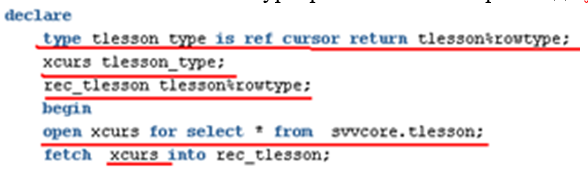
# 41. Курсоры. Курсорные переменные. Параметры инстанса, связанные с курсорами.

**Курсорные переменные** - это структуры данных, которые указывают на курсорный объект

**Используются для:**

Передачи курсора в качестве параметра,

Чтобы отложить связь курсора с SELECT-запросом до выполнения команды OPEN



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

## Параметры Oracle, связанные с курсорами

**cursor\_space\_for\_time = {TRUE|FALSE}** – больший объем памяти для курсоров и никогда не освобождается. Применяется для увеличения скорости работы курсоров при наличии памяти для разделяемого пула.

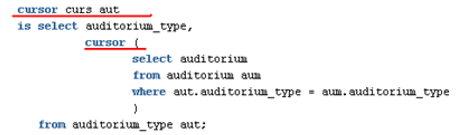
**cursor\_sharing = {EXACT|SIMILAR|FORCE}**

**open\_cursors** - максимальное количество открытых курсоров.

**session\_cached\_cursors** – максимальное количество кэшируемых курсоров для сессии.

# 42. Курсоры. Курсорные подзапросы.

**Курсорные подзапросы или вложенные курсорные выражения** – механизм встраивания курсоров в операторы SQL.



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

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

Если планируется обновить/удалить записи, на которые ссылается SELECT FOR UPDATE:

update таблица set … where current of имя курсора

delete from таблица where current of имя курсора

# 44. Курсоры. Динамические курсоры.

**Динамические курсоры**, SQL выражение для которых определяется на этапе выполнения:

Могут использоваться для любых SQL выражений, включая DDL и DCL команды

Реализуются с помощью оператора EXECUTE IMMEDIATE или пакета dbms\_sql

EXECUTE IMMEDIATE - однострочные запросы и DDL команды

OPEN FOR, FETCH и CLOSE - динамические многострочные запросы

#### Синтаксис EXECUTE IMMEDIATE:

EXECUTE IMMEDIATE sql\_statement

[INTO {variable [,variable ...] | record}] [USING [IN | OUT | IN OUT] bind\_argument

[, [IN | OUT | IN OUT] bind\_argument . ..] ] [{RETURNING | RETURN} INTO bind\_argument [,bind\_argument]...];

Для улучшения производительности выполнения SQL выражений можно использовать динамические курсоры со связанными переменными

Это позволяет 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;

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

**ROWID** — это псевдостолбец, который является уникальным идентификатором строки в таблице и фактически описывает точное физическое расположение данной конкретной строки.

**ROWID** — идентификатор строки (ROW IDentifier), а точнее, двоичное значение, однозначно идентифицирующее строку данных в таблице Oracle, даже если таблица не имеет уникального ключа.

**ROWID**-это физическое местоположение строки. Следовательно, это самый быстрый способ поиска строки, даже быстрее, чем поиск по первичному ключу.

**ROWNUM** - логический номер записи в запросе

– это псевдостолбец, значения которого можно увидеть, включив его в любой результирующий набор, например в список столбцов оператора SELECT. Значениям столбца ROWNUM присваиваются номера 1, 2, 3, 4, ... N, где N – число строк результирующего набора запроса.

select \* from my\_table where ROWNUM < 5;

отберет первые 4 записи из таблицы my\_table.

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

**Исключительная ситуаци**я — Совокупность определенных условий, возникновение которых приводит к нарушению предусмотренной последовательности выполнения в программе

**SQLERRM** возвращает сообщение об ошибке, связанной с исключительной ситуацией

**SQLCODE** возвращает номер ошибки, связанной с исключительной ситуацией

**WHEN OTHERS** перехват всех остальных исключительных ситуаций, которые не были обработаны встроенными исключительными ситуациями или исключительными ситуациями, определенные программистом

**WHEN TO\_MANY\_ROWS** запрос вернул более одной строки

**WHEN NO\_DATA\_FOUND** неявный курсор не возвращает строк вообще

***Обработка исключений*** - перехват ошибки в секции исключения

**Секция исключений** – 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 исключений от 1 блока другому, если исключение не было обработано

**RAISE (RAISE\_APPLICATION\_ERROR)** – команда, которая прерывает выполнение текущего блока

* Определена в пакете DBMS\_ STANDARD
* Можно присвоить сообщение об ошибке

***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 функции языка PL/SQL. Функции работы с датами, текстом и числами.

* **Числовые функции**
  + ***ABS*** возвращает абсолютное значение числа
  + ***AVG*** возвращает среднее значение выражения
  + ***EXP*** возвращает e, возведенное в n-ную степень, где е = 2,71828183.
  + ***FLOOR*** возвращает наибольшее целое значение, равное или меньшее, чем число
  + ***MOD*** возвращает остаток от деления m на n.
  + ***POWER*** возводит m в степень n.
  + ***SING*** возвращает значение, определяющее знак числа.
  + ***SUM*** возвращает суммарное значение выражения
* **Символьные функции**
  + ***ASCII*** возвращает числовое представление крайнего левого символа строки.
  + ***COMPOSE*** возвращает строку Unicode.
  + ***CONCAT*** позволяет соединить вместе две строки.
  + ***DUMP*** возвращает значение varchar2, который включает код типа данных, длину в байтах, и внутреннее представление выражения.
  + ***INSTR*** возвращает n-е вхождение подстроки в строке.
  + ***LENGTH*** возвращает длину указанной строки.
  + ***LOWER*** преобразует все символы в заданной строке в нижний регистр
  + ***LPAD*** добавляет с левой части строки определенный набор символов
  + ***REPLACE*** заменяет последовательность символов в строке другим набором символов.
  + ***RPAD*** дополняет с правой части строки определенный набор символов
  + ***RTRIM*** удаляет все указанные символы из правой части строки.
  + ***SUBSTR*** позволяет извлекать подстроку из строки
  + ***TO\_CHAR*** преобразует число или дату в строку.
  + ***TRANSLATE*** заменяет последовательность символов в строке другим набором символов
  + ***TRIM*** удаляет все указанные символы с начала или окончания строки.
  + ***UPPER*** преобразует все символы строки в верхний регистр.
* **Функции по работе с датами**
  + ***ADD\_MONTHS*** возвращает дату плюс n месяцев.
  + ***EXTRACT*** извлекает значение из даты или значения интервала.
  + ***LAST\_DAY*** возвращает последний день месяца на основе значения даты
  + ***NEXT\_DAY*** возвращает первый день недели, который больше date.
  + ***ROUND*** возвращает дату округленную до определенной единицы измерения.
  + ***ROUND*** возвращает дату округленную до определенной единицы измерения.
  + ***TO\_DATE*** преобразует строку в дату.
  + ***TRUNC*** возвращает дату, усеченную к определенной единице измерения.
* **Конвертирование**
  + ***CAST*** конвертирует (преобразует) один тип данных в другой.
  + ***CONVERT*** преобразует строку из одного набора символов в другой.
  + ***DECOMPOSE*** принимает строку и возвращает строку Unicode.
  + ***TO\_NUMBER*** преобразует строку в число.
  + ***TO\_TIMESTAMP*** преобразует строку в значение TIMESTAMP.
* **Функции обработки ошибок**

# 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\_LIKE (source\_char, pattern [, match\_parameter ] )

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

REGEXP\_INSTR(source\_char, pattern [, start\_position [, nth\_appearance [, return\_option [, match\_parameter [, sub\_expression ] ] ] ] ] )

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

REGEXP\_REPLACE(source\_char, pattern [, replacement\_string [, start\_position [, nth\_appearance [, match\_parameter ] ] ] ] )

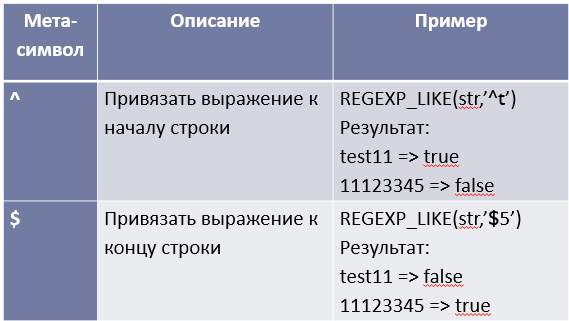
**REGEXP\_SUBSTR** выделяет из строки шаблон

REGEXP\_SUBSTR(source\_char, pattern [, start\_position [, nth\_appearance [, match\_parameter [, sub\_expression ] ] ] ] )

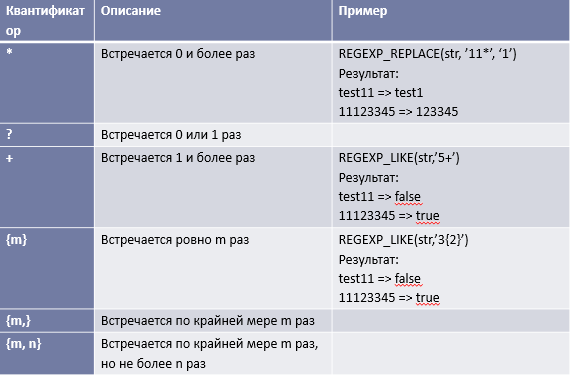
**REGEXP\_COUNT** определяет количество вхождений

REGEXP\_COUNT (source\_char, pattern [, position [, match\_param]])

#### Метасимволы привязки



#### Квантификаторы и операторы повтора



# 50. Коллекции. Массивы переменной длины.

**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

Одномерная, но можно создавать коллекции коллекций

* Массив переменной длины VARRAY
* Вложенная таблица (nested tables)
* Ассоциативный массив (associative array)

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

Коллекции типа VARRAY всегда ограничены

Вложенные таблицы и ассоциативные массивы неограничены

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

Массивы VARRAY всегда являются плотными

Вложенные таблицы первоначально всегда плотные, но по мере удаления некоторых элементов становятся разреженными

Ассоциативные массивы могут быть как разреженными, так и плотными в зависимости от способа их заполнения

#### Работа с коллекциями

Объявление коллекций

Инициализация коллекций

Явно с помощью конструктора

Неявно при выборке из базы данных

Прямым присвоением переменной с другой коллекции такого же типа

Добавление и удаление элементов

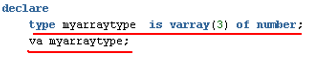
Вложенные таблицы и массивы переменной длины – сначала увеличить размер при помощи функции EXTEND, а затем присвоить значения новым элементам

Ассоциативный массив – присвоение значения новому элементу

**Массивы переменной длины** – одномерные, связанные коллекции однотипных элементов

Доступны в рамках PL/SQL и в БД

Являются плотными



# 51. Коллекции. Вложенные таблицы.

**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

Одномерная, но можно создавать коллекции коллекций

* Массив переменной длины VARRAY
* Вложенная таблица (nested tables)
* Ассоциативный массив (associative array)

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

Коллекции типа VARRAY всегда ограничены

Вложенные таблицы и ассоциативные массивы неограничены

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

Массивы VARRAY всегда являются плотными

Вложенные таблицы первоначально всегда плотные, но по мере удаления некоторых элементов становятся разреженными

Ассоциативные массивы могут быть как разреженными, так и плотными в зависимости от способа их заполнения

#### Работа с коллекциями

Объявление коллекций

Инициализация коллекций

Явно с помощью конструктора

Неявно при выборке из базы данных

Прямым присвоением переменной с другой коллекции такого же типа

Добавление и удаление элементов

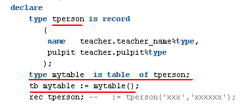
Вложенные таблицы и массивы переменной длины – сначала увеличить размер при помощи функции EXTEND, а затем присвоить значения новым элементам

Ассоциативный массив – присвоение значения новому элементу

**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



# 52. Коллекции. Ассоциативные массивы.

**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

Одномерная, но можно создавать коллекции коллекций

* Массив переменной длины VARRAY
* Вложенная таблица (nested tables)
* Ассоциативный массив (associative array)

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

Коллекции типа VARRAY всегда ограничены

Вложенные таблицы и ассоциативные массивы неограничены

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

Массивы VARRAY всегда являются плотными

Вложенные таблицы первоначально всегда плотные, но по мере удаления некоторых элементов становятся разреженными

Ассоциативные массивы могут быть как разреженными, так и плотными в зависимости от способа их заполнения

#### Работа с коллекциями

Объявление коллекций

Инициализация коллекций

Явно с помощью конструктора

Неявно при выборке из базы данных

Прямым присвоением переменной с другой коллекции такого же типа

Добавление и удаление элементов

Вложенные таблицы и массивы переменной длины – сначала увеличить размер при помощи функции EXTEND, а затем присвоить значения новым элементам

Ассоциативный массив – присвоение значения новому элементу

**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. Хранимые процедуры. Вызов процедур. Входные и выходные параметры, позиционный и параметрический форматы передачи фактических параметров. Значения параметров по умолчанию.

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

Для создания процедур необходима привилегия create procedure

#### Параметры

* Наименование
* Тип данных
* Режим передачи
* Начальное значение

#### Тип данных параметров

PL/SQL или программно-определенный

Не может быть ограничен по размеру

Размер определяется через вызывающую программу или через связанное объявление переменной

**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);

**Типы параметров:**

IN

OUT

IN OUT

**При выполнении:**

Значения OUT устанавливаются в NULL

Значения IN OUT остаются неизменными

При ошибке присвоения для параметров откатываются, кроме NOCOPY

**Значения по умолчанию**

IN, IN OUT

Можно не задавать при вызове

**Компиляция**

**OR REPLACE** – перестроение уже существующего модуля, привилегии на выполнение сохраняются

**AUTHID** – определяет, как будет выполняться модуль и разрешаться имена в БД:

**DEFINER** – (по умолчанию) от имени владельца модуля

**CURRENT\_USER** - от имени пользователя, выполняющего модуль

**Вызов процедуры:**

conn -> procedures -> выбрать -> Run

exec

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

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

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

**Функция без параметров**



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

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

**Значения по умолчанию**

IN, IN OUT

Конвейерная обработка отменяет надобность в создании огромных наборов, передавая строки по каналу из функции по мере их создания, сохраняя память и позволяя запустить последующую обработку еще до окончания генерации всех строк.

Конвейерные табличные функции включают фразу **PIPELINED** и используют вызов PIPE ROW, чтобы вытолкнуть строки из функции, как только они создадутся, вместо построения табличной коллекции. Заметим, что вызов RETURN пустой, поскольку нет никакой коллекции, возвращаемой из функции.

# 55. Процедурные объекты. Пакеты. Спецификация и реализация пакета.

**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

AUTHID {CURRENT\_USER|DEFINER}

Словарь: USER\_PROCEDURES, USER\_SOURCE

ALTER PACKAGE COMPILE PACKAGE

ALTER PACKAGE COMPILE BODY

DROP PACKAGE

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

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

**Триггер** – это хранимая процедура которая, автоматически выполняется при наступлении заданного события.

**Триггер** – часть транзакции, ошибка в триггере откатывает операцию, изменения таблиц в триггере становятся частью транзакции.

Если откатывается транзакция, изменения триггера тоже откатываются

### 

### Типы триггеров:

1. **По привязанному объекту:**

* На таблице
* На представлении - instead of trigger

1. **По событиям запуска:**

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

1. **По области действия:**

* Уровень оператора - statement level triggers
* Уровень записи - row level triggers
* Составные триггеры - compound triggers

1. **По времени срабатывания:**

* Перед выполнением операции – before
* После выполнения операции - after

### ***Уровни триггеров:***

1. **FOR EACH ROW** (для каждой строки) - срабатывает для каждой измененной строки
2. **ПО УМОЛЧАНИЮ** (операторный уровень) - срабатывает один раз на тригтерное событие

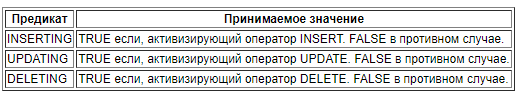
### Порядок выполнения:

* операторные BEFORE
* для каждой строки BEFORE
* выполнение оператор
* для каждой строки AFTER
* операторный AFTER

### 

### Предикаты

В триггерах(***INSERT, UPDATA, DELETE***) можно использовать три логические функции, определяющие тип выполняемой операции. Это логические ***функции (предикаты)*** INSERTING, UPDATING и DELETING.



### 

### Триггеры замещения

Триггеры замещения (ins­tead of) можно создавать только для представлений (либо объектных, либо реляционных). В отличие от триггеров DML, которые выполняют­ся в дополнение к операторам DML, триггеры замещения выполняются вместо операторов DML, вызывающих их срабатывание.

### 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

Триггеры выполняется под правами создателя триггера

Назначаются напрямую USERу, а не через роль

#### 

### Включение/отключение триггеров

**Включение и отключение триггеров:**

alter trigger { disable | enable }

**Всех для таблицы:**

ALTER TABLE table\_name { ENABLE | DISABLE } ALL TRIGGERS;

**Компиляция триггера:**

alter trigger TRIGGER\_NAME compile;

**Переименование триггера**

#### 

### Псевдозаписи new/old

При запуске триггера уровня строки заполняются две структуры данных, которые работают подобно записям. Это псевдозаписи NEW и OLD («псевдо» потому что они обладают не всеми свойствами настоящих записей PL/SQL).

OLD хранит начальные значения записи, обрабатываемой триггером,

NEW - новые значения.

Эти записи имеют такую же структуру, как запись, объявленная при помощи атрибута %ROWTYPE на основе таблицы, к которой относится триггер.

При работе с NEW и OLD есть правила:

Для триггеров **INSERT** - OLD не содержит никаких данных, “страрого” набора значений не существует

Для триггеров **UPDATE** -OLD и NEW заполняются. OLD - содержит исходные значение(до обновления), а NEW - значения после выполнения обновлений

Для триггеров **DELETE** - NEW не содержит никаких данных, ведь речь идет об удалении записи.

Изменение значений полей структуры OLD запрещено, попытка такого изменения приведет к возникновению ошибки ORA-04085.

Изменение значений полей структуры NEW допустимо.

Мутирование таблиц (ошибка ORA-04091) возникает, если в триггере уровня строки выполняется изменение или чтение данных из той же самой таблицы, для которой данный триггер должен был сработать

# 57. Секционирование таблиц. Виды секционирования.

**Секционирование (partitioning)** - это способ физического распределения таблиц и индексов среди двух или более табличных пространств в зависимости от значений ключевых колонок таблиц с целью повышения производительности операций ввода/вывода.

Метод, позволяющий хранить сегмент данных, такой как таблица, в виде нескольких сегментов, сохраняя логическую монолитную структуру.

* повышение производительности
* освобождение табличного пространства,
* секционирования является разбиение большой таблицы на оперативную и архивную части. Особенно это эффективно, если оперативная часть в виде секции интенсивно пополняется и модифицируется, а архивная часть (секции) менее подвержена изменениям, и существенно реже из нее извлекается информация.
* обеспечение устойчивости функционирования таблиц
* снижение конкуренции за строки и индексы таблицы, в том числе уменьшения вероятности блокировок.

**Ключ секционирования**( столбец или несколько столбцов) относительно значений которых будет делаться разнесение таблицы на секции

* **Range - секционирование по диапазону ключа** - по диапазону дат и по диапазону значений. Данные распределяются по секциям в зависимости от диапазона значения поля. Например, этот вид секционирования удобен для разбиения данных по месяцам или годам (таблица с данными о проданных товарах) на основании поля, содержащего дату продажи. ***Недостаток:*** при загрузке новых данных в таблицу нужно постоянно расщеплять секцию maxvalue

(

partition partitionname1 values less than (...),

partition partitionname2 values less than (...),

partition partitionname3 values less than (maxvalue)

);

* **List** **- секционирование по списку ключа** - есть возможность указать конкретный перечень дискретных значений столбца, по которому происходит разбиение. Отличается от Range partitioning тем, что для секции задается не диапазон, а список значений.
* **Hash - хеш-секционирование** - если не получается секционировать по диапазону RANGE или LIST, то применяется хешсекционирование, основанное на хеш-функции. В этом случае строки таблицы равномерно распределяются между секциями на основании внутренних алгоритмов хеширования Oracle. При создании такой таблицы достаточно указать только количество секций.
* составное секционирование.
* **интервальное секционирование** - создается единственная секция без maxvalue. Новые секции будут создаваться автоматически
* **ссылочное секционирование** - выполняется, если таблицы явным образом связаны ссылочными ограничениями целостности. Манипуляции с секциями главной таблицы автоматически отражаются на секциях подчиненной.

partition by reference (refcolumn)

* системное секционирование

**ALTER TABLE MERGE** - можно объединить содержимое двух или более разделов или подразделов в один новый раздел или подраздел

**ALTER TABLE SPLIT** - создается два новых раздела и перераспределяет содержимое старого раздела между ними

**ALTER TABLE EXCHANGE** - меняет местами существующую таблицу с разделом или подразделом

# 58. Транзакции. Виды транзакций. Понятие автономной транзакции.

**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. Фиксация (committing) транзакции закрепляет проведенные вами изменения, а откат (rollback) – конечно же, отменяет их. Как только вы зафиксировали транзакцию, все прочие транзакции других пользователей, которые начались после нее, смогут видеть изменения, проведенные вашими транзакциями.

Когда транзакция вообще не может выполниться (скажем, из-за отключения электропитания), то она вся целиком должна быть отменена. Oracle откатывает все изменения, проведенные предшествующими операторами SQL, возвращая данные в исходное состояние (которое они имели перед началом транзакции). Весь процесс построен так, чтобы поддерживать целостность данных – т.е. концепцию «все или ничего».

## Уровни изоляции транзакции:

**Уровень изоляции** задает степень защищенности данных в транзакции от возможности изменения другими транзакциями

* Read uncommitted (Чтение незафиксированных данных),
* Read committed (Чтение зафиксированных данных),
* Repeatable read (Повторяемое чтение)
* Serializable (Сериализуемость)

### 4 свойства транзакций

**Atomicity (атомарность)** — транзакция должна быть выполнена в целом или не выполнена вовсе.

**Consistency (согласованность)** — гарантирует, что по мере выполнения транзакций, данные переходят из одного согласованного состояния в другое, то есть транзакция не может разрушить взаимной согласованности данных.

**Isolation (изолированность)** — локализация пользовательских процессов означает, что конкурирующие за доступ к БД транзакции физически обрабатываются последовательно, изолированно друг от друга, но для пользователей это выглядит, как будто они выполняются параллельно.

**Durability (долговечность)** — устойчивость к ошибкам — если транзакция завершена успешно, то те изменения в данных, которые были ею произведены, не могут быть потеряны ни при каких обстоятельствах.

#### 

**READ UNCOMMITTED**

* Не изолирует операции чтения других транзакций
* Транзакция не задает и не признает блокировок
* Допускает проблемы:
* Грязное чтение
* Неповторяемое чтение
* Фантомное чтение

**READ COMMITTED**

* Самая плохая согласованность данных, но самая высокая скорость выполнения транзакций.
* Каждая транзакция видит незафиксированные изменения другой транзакции
* Является уровнем изоляции по умолчанию
* Проблемы:
* Неповторяемое чтение
* Фантомное чтение

**REPEATABLE READ**

* Устанавливает разделяемые блокировки на все считываемые данные и удерживает эти блокировки до тех пор, пока транзакция не будет подтверждена или отменена
* Не препятствует другим инструкциям вставлять новые строки
* Проблема:
* Фантомное чтение

**SERIALIZABLE**

* Устанавливает блокировку на всю область данных, считываемых соответствующей транзакцией
* Предотвращает вставку новых строк другой транзакцией до тех пор, пока первая транзакция не будет подтверждена или отменена
* Реализуется с использованием метода блокировки диапазона ключа
* Блокировка диапазона ключа блокирует элементы индексов

**Неповторяемое чтение** — транзакция повторно читает те же данные, что и раньше, и обнаруживает, что они были изменены другой транзакцией (которая завершилась после первого чтения).

**Фантомное чтение** — транзакция повторно выполняет запрос, возвращающий набор строк для некоторого условия, и обнаруживает, что набор строк, удовлетворяющих условию, изменился из-за транзакции, завершившейся за это время.

**Грязное чтение**  — происходит, когда транзакция A читает данные, которые были изменены другой транзакцией B, но еще не были зафиксированы. Если B откатывается, транзакция A считывает неверные данные. Это похоже на неповторяемое чтение, но вторая транзакция не должна выполнять фиксацию.

**Блокировка** - это средство предотвращения доступа других транзакций к указанным ресурсам.

**Автономная транзакция** – независимая транзакция, начатая другой транзакцией, которая является главной. В блоке автономной транзакции главная транзакция приостанавливается. Вы выполняете SQL-операции, затем производите их фиксацию или откат и возобновляете главную транзакцию.

Чтобы определить блок как автономную транзакцию, нужно включить в раздел объявлений директиву PRAGMA AUTONOMOUS\_TRANSACTION

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

### *Встроенные системные пакеты:*

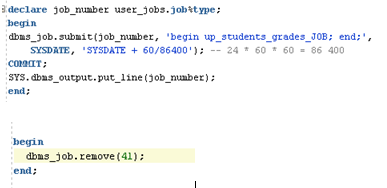
* Устанавливаются во время установки Oracle
* Используются для расширения основных функциональных возможностей Oracle
* Владелец пакетов - SYS (обычно, есть и другие)
* Написаны на C или на PL/SQL

***DBMS\_JOB*** Пакет был заменен ***DBMS\_SCHEDULER*** пакетом.

**DBMS\_JOB** управляет заданиями в очереди заданий.

**DBMS\_JOB** специальный механизм запланировать выполнение определенной программы, на заданное время, это может быть оператор SQL, программа на языке PL/SQL, либо даже внешняя программа.

* ***SUBMIT*** – создание задания, отправляет задание в очередь заданий
* ***ISUBMIT*** – создание задания с номером
* ***REMOVE*** – удаление задания из очереди, но это не останавливает выполнение задания
* ***RUN*** – немедленное выполнение задания в пользовательском сеансе
* ***BROKEN*** – процедура устанавливает неработающий флаг. Сломанное задания никогда не запускается.
* ***INSTANCE*** – назначить конкретный экземпляр для выполнения задани
* ***NEXT\_DATE*** – изменение времени выполнения
* ***INTERVAL*** – изменение интервала(частоты) выполнения
* ***CHANGE*** – изменить определяемые пользователем параметры, связанные с заданием
* ***WHAT*** – процедура изменяет то, что делает существующее задание



#### **Пакет DBMS\_SCHEDULER** содержит различные функции и процедуры, помогающие управлять планировщиком

***Oracle Scheduler*** — не только инструмент описания заданий (jobs), он также помогает контролировать использование ресурсов и устанавливать приоритеты заданий внутри базы данных.

Одним из ограничений пакета DBMS\_JOB является то, что он может выполнять только задания на на базе PL/SQL, и его нельзя использовать для планирования запуска системных сценариев либо исполняемых файлов.

***Oracle Scheduler*** позволяет использовать сценарии PL/SQL, сценарии оболочки операционной системы, программы Java, и родные двоичные исполняемые файлы для выполнения запланированных заданий.

# 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

* ***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.
* ***DBMS\_ADVANCED\_REWRITE*** – перехват и замена SQL-запросов
* ***DBMS\_ADVISOR*** – часть набора экспертной системы для решения проблем производительности, связанных с компонентами сервера базы данных
* ***DBMS\_SQLTUNE*** – сбор статистики, используется при анализе производительности SQL-операторов
* ***DBMS\_APPLICATION\_INFO*** – присвоение имени процессу для удобства мониторинга и отладки
* ***DBMS\_AQ*** – пакет для Advanced Queuing – система обмена очередями сообщений
* ***DBMS\_TRANSFORM*** – пакет для Oracle Advanced Queuing – преобразования данных
* ***DBMS\_DEFER*** – вызов удаленных процедур
* ***DBMS\_OFFLINE\_OG, DBMS\_REPCAT*** – поддержка репликации
* ***DBMS\_REFRESH*** – создание группы материализованных представлений для обновления как единого целого
* ***DBMS\_BACKUP\_RESTORE*** – пакет для создания резервных копий для управляющих файлов, журналов повтора, архивных файлов
* ***DBMS\_CRYPTO*** – пакет шифрования данных

Поддерживаются алгоритмы шифрования - DES, AES, RC4, 3DES, 3DES-2KEY

* ***DBMS\_CUBE*** - поддерживает развертывание кубических материализованных представлений из существующих реляционных материализованных представлений
* ***DBMS\_DIMENSION*** - отображение информации об измерениях – иерархических связях между данными
* ***Data Warehouse*** - предметно-ориентированная база данных, предназначенная для отчетов и/или OLAP
* ***DBMS\_DEBUG*** – пакет для отладки PL/SQL-кода на сервере
* ***DBMS\_DDL*** - эквиваленты команд DDL, которые нельзя использовать в PL/SQL непо­средственно: ALTER COMPILE, CREATE\_WRAPPED
* ***DBMS\_DESCRIBE*** – позволяет получить информацию о процедурах
* ***DBMS\_FGA*** – пакет настройки и использования политик аудита: добавление/удаление, включение/выключение. FGA - Fine-Grained-Audit
* ***DBMS\_FILE\_TRANSFER*** – пересылка файлов между файловыми системами и базами данных (COPY\_FILE, GET\_FILE, PUT\_FILE)
* ***DMS\_FLASHBACK*** – пакет поддержки технологии flashback: (DISABLE, ENABLE\_AT\_SYSTEM\_CHANGE\_NUMBER, ENABLE\_AT\_TIME, GET\_SYSTEM\_CHANGE\_NUMBER, TRANSACTION\_BACKOUT)
* ***DMS\_ HS\_PASSTHROUGH - HS – Heterogeneous Services*** - Пакет поддержки интерфейса с базами данных отличными от Oracle – ANSI SQL для поддерживаемых баз данных
* ***DBMS\_JAVA*** – пакет работы с хранимыми Java-процедурами
* ***DBMS\_LOCK*** – пакет управления блокировками(Задать блокировку, Задать уникальное имя блокировки, Изменить режим блокировки, Снять блокировку)
* ***DBMS\_OUTPUT*** – серверный вывод