|  |
| --- |
| **Лабораторная работа № 2. Администрирование СУБД Oracle.** |
| **Задание 1.**  SQL> set echo on  SQL> spool d:\spool.txt  При установке Oracle Database автоматически создаются следующие учетные записи администратора.  Они обе создаются с паролем, который вы предоставили при установке, и обе они автоматически  получают роль DBA.  О DBA роли Роль DBA – это предопределённая роль, которая автоматически создаётся для каждой  базы данных Oracle. Эта роль содержит все системные привилегии, кроме SYSDBA и SYSOPER.  Поэтому она очень мощная и должна назначаться только администраторам, которым требуется  полный доступ.  Выборка всех ролей с базы данных: SQL> select role from dba\_roles;  Выборка всех пользователей, которым назначена DBA роль:  SQL>select \* from dba\_role\_privs where granted\_role = ‘DBA’  Схема по умолчанию для SYSDBA: SYS Схема по умолчанию для SYSOPER: PUBLIC Выборка пользователей,  которые имеют SYSDBA или SYSOPER привилегии:   |  | | --- | | SQL>column sysdba format a10 SQL>column sysdba format a10 SQL>select \* from v$pwfile\_users; |   Таблицы в схеме SYS управляются только базой данных. Они никогда не должны модифицироваться ни администратором пользователя, ни администратором базы данных, и никто не должен создавать таблицы в схеме пользователя SYS. Пользователи базы данных не должны подключаться к базе данных Oracle с помощью учетной записи SYS.  Просмотрите привилегии/роли текущего пользователя из разных схем  (манипулируя различными соединениями)  SQL> connect system/masterkey…  SQL> conn sys as sysdba  SQL> conn HR/HR  SQL> conn user1/user1  SQL> show user  SQL> desc session\_privs ……………………………..  SQL> select \* from session\_privs;  SQL> select \* from session\_roles;  SQL> select \* from user\_sys\_privs;  SQL> select \* from dba\_role\_privs where grantee='SYSTEM';  SQL> select \* from dba\_role\_privs where grantee='SYS';  SQL> select \* from dba\_role\_privs where grantee='HR';  SQL> select \* from dba\_role\_privs where grantee='USER';  **Задание 2.**  В **базе данных Oracle схема (schema)** определяется как коллекция логических структур данных, или объектов схемы, схемы, можем перечислить:  SQL>SELECT USERNAME FROM ALL\_USERS ORDER BY USERNAME;  Существуют два основных способа создания схемы в базе данных Oracle. Более распространенный способ состоит в подключении от имени владельца схемы и создании всех таблиц, индексов и прочих объектов, которые планируется включить в схему. Поскольку все объекты создаются владельцем схемы, они автоматически становятся частью этой схемы. Второй способ создания схемы предусматривает ее явное создание с помощью оператора CREATE SCHEMA. Этот оператор позволяет создать множество таблиц и представлений, а также выдать пользователям привилегии доступа к ним, и все это в единственном операторе SQL.  **Задание 3. (просмотреть выборочно выполнить интересующие Вас запросы).**  **Что и где найти в Oracle**  Имена многих таблицы выглядят похоже. Например, dba\_users, all\_users, user\_users.  Префикс указывает на разный уровень доступности данных:  - dba - относится ко всей базе данных и доступна пользователю с правами DBA;  - all - относится к схеме пользователя и другим схемам, на которые у пользователя есть полномочия;  - user - относится только к схеме пользователя;   |  |  | | --- | --- | | пример запроса | примечание | | select \* from dict where table\_name like '%PUMP%' ; | список всех таблиц и представлений, в имени которых встречается "PUMP" | | select instance\_name from v$instance; | имя инстанса Сервер базы данных Oracle состоит из базы данных и инстанции Oracle. При каждом запуске базы данных запускаются фоновые процессы, и выделяется System Global Area (SGA) - Системная Глобальная Область. Совокупность фоновых процессов и буферов памяти называется инстанцией Oracle. | | select sid||','||serial# sess ,username ,program ,machine ,status from v$session; | информация об установленных сессиях  кто откуда подключен, какой статус подключения. sess - готовое значения для подстановки, например, в alter system disconnect session '' immediate; | | select \* from dba\_objects where status='INVALID'; | список объектов базы данных, которые по каким-либо причинам оказались невалидными. | | select \* from dba\_all\_tables where tablespace\_name not like 'sys%'; | выводит все таблицы, размещённые в табличных пространствах, имена которых не начинаются на sys | | select \* from dba\_data\_files; | выводит список всех файлов БД и информацию о них | | select distinct aat.owner , aat.table\_name , aat.tablespace\_name from all\_all\_tables aat , dba\_data\_files ddf where aat.tablespace\_name=ddf.tablespace\_name and ddf.file\_name like '%gbs\_ecap\_data%'; | выводит названия всех таблиц (с указанием их владельцев) и tablespace'ов входящих в data-files gbs\_ecap\_data | | select file\_id , file\_name , tablespace\_name , bytes/1024/1024 as mbytes , status from dba\_data\_files; | выводит названия всех data-файлов, связаных с ними tablespace'ов, размеров и статусов | | select name , value , display\_value , isses\_modifiable , issys\_modifiable from v$parameter where name like 'sga%'; | информация о параметрах БД  имя параметра, его значение, отображаемое значение и указание возможности/невозможности модификации значения для сессии и системы | | select banner from v$version; | информация о версии БД | | select s.sql\_fulltext , ses.username , ses.osuser from v$sql s , v$session ses where ses.sql\_address=s.address; | выводит sql-запросы и пользователей их запустивших (и oracle-пользователей и пользователей os) | | select \* from v$process; | информация о процессах БД и используемых ими ресурсах | | select \* from user$; | вывести всех пользователей | | select \* from dba\_users; | | select \* from dba\_tab\_privs | все пользователи и их полномочия, а так же кто выдал эти полномочия | | select \* from dba\_tab\_privs\_made where grantee='ctxsys'; | кто, на что и какие изменения полномочий проводил для пользователя ctxsys | | select \* from dba\_tab\_privs\_recd | все пользователи и их полномочия, а так же кто выдал эти полномочия | | select \* from dba\_sys\_privs where grantee='sys'; | вывод привилегий получателя привилегий sys | | select \* from dba\_role\_privs where grantee='sys'; | вывод ролей получателя привилегий sys | | select \* from dba\_roles; | список ролей | | select \* from dba\_ts\_quotas; | список tablespace'ов и пользователей, имеющих привилегии на эти tablespace'ы | | select \* from role\_role\_privs; | список ролей, предоставленных другим ролям | | select \* from role\_sys\_privs; | все назначения системных привелегий на таблицы (владелец, какие привилегии, кто выдал) | | select \* from role\_tab\_privs; | список таблиц, их владельцев и привилегий, данных определённым ролям | | select \* from session\_privs; | все привилегии текущего пользователя | | select \* from session\_roles; | все роли текущего пользователя | | connect user/password@sid; select \* from user\_sys\_privs; | список привилегий текущего пользователя | | connect user/password@sid; select \* from user\_role\_privs; | список ролей текущего пользователя | | select \* from v$nls\_parameters; | NLS-установки базы данных подключенного пользователя Поддержка национальных языков (NLS) - это технология, позволяющая Oracle приложения для взаимодействия с пользователями на их родном языке, используя их соглашения для отображения данных | | select \* from nls\_database\_parameters; | текущие NLS-установки базы данных | | select \* from nls\_instance\_parameters; | текущие NLS-установки инстанса | | select \* from nls\_session\_parameters; | текущие NLS-установки сессии | | select \* from v$nls\_valid\_values where parameter='CHARACTERSET' and VALUE like '%WIN%'; | отобрать все значения CHARACTERSET, содержащие в своём названии "WIN" | | select \* from dba\_errors; | просмотр ошибок | | select \* from dba\_scheduler\_jobs; | просмотр информации о задачах по расписанию | | select log\_date , job\_name , status , req\_start\_date , actual\_start\_date , run\_duration from dba\_scheduler\_job\_run\_details; | просмотр детализированной информации о выполненных задачах | | select log\_date , job\_name , status from dba\_scheduler\_job\_log; | просмотр истории о выполненных задачах по расписанию вывести дату, имя и статус выполненых задач | | select job\_name , session\_id , running\_instance , elapsed\_time , cpu\_used from dba\_scheduler\_running\_jobs; | просмотр информации о текущих задачах | | select \* from dba\_scheduler\_programs; | вывести все программы и их атрибуты | | select \* from dba\_scheduler\_program\_args; | вывести все программы и их аргументы | | select \* from dba\_jobs; | вывести все задания | | select \* from dba\_jobs\_running; | вывести все выполняемые задания | | select \* from v$sga; | вывести информацию о SGA | | select \* from v$sgastat; | вывести статистику SGA |   **Задание 4.**  Табличные пространства выполняют ряд ключевых функций в базе данных Oracle, табличные пространства облегчают выделение квот пространства различным пользователям базы данных. Табличные пространства позволяют вам выполнять частичное [резервирование и восстановление](https://oracle-patches.com/oracle/backup/2926-%D1%80%D0%B5%D0%B7%D0%B5%D1%80%D0%B2%D0%BD%D0%BE%D0%B5-%D0%BA%D0%BE%D0%BF%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%B8-%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B1%D0%B0%D0%B7%D1%8B-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-oracle-%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0%D0%BC%D0%B8-%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) на основе табличного пространства как единицы такой операции. Поскольку крупные объекты типа разбитой на разделы таблицы хранилища данных могут распространяться на несколько табличных пространств, вы можете повысить производительность, размещая табличные пространства на нескольких дисках и контроллерах.  SQL>SELECT TABLESPACE\_NAME FROM USER\_TABLESPACES;  **Задание 5.**  Просмотр пользователей включённых в файл паролей  В файле паролей на данный момент времени должны присутствовать три пользователя SYS, USER1 и USER2. Как убедиться, что они действительно туда попали? Конечно, можно просто посмотреть содержимое двоичного файла, но есть и более простой путь - это системное представление V$PWFILE\_USERS:  SQL> select \* from v$pwfile\_users  **Задание №6.**  **Блоки данных (Data Block)**Блоки данных (Data Block) - мельчайший строительный блок базы данных Oracle, состоящий из определенного количества байт на диске. 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.  **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.  **Сегменты (segments)**Сегменты (segments) - набор экстентов, которые вы выделяете логической структуре, такой как таблица или индекс (или некоторый другой объект). Набор экстентов формирует следующую более крупную единицу хранения, именуемую сегментом. Oracle называет сегментом все пространство, выделенное любому конкретному объекту базы данных.  SQL> desc DBA\_SEGMENTS  SQL> desc DBA\_EXTENTS  SQL> desc DBA\_FREE\_SPACE  **Displaying Segment Information**  SQL> SELECT SEGMENT\_NAME, TABLESPACE\_NAME, BYTES, BLOCKS, EXTENTS  FROM DBA\_SEGMENTS  WHERE SEGMENT\_TYPE = 'INDEX'  AND OWNER='HR'  ORDER BY SEGMENT\_NAME;  **Displaying Extent Information**  SQL> SELECT SEGMENT\_NAME, SEGMENT\_TYPE, TABLESPACE\_NAME, EXTENT\_ID, BYTES, BLOCKS  FROM DBA\_EXTENTS  WHERE SEGMENT\_TYPE = 'INDEX'  AND OWNER='HR'  ORDER BY SEGMENT\_NAME;  **Displaying the Free Space (Extents) in a Tablespace**  SQL> SELECT TABLESPACE\_NAME, FILE\_ID, BYTES, BLOCKS  FROM DBA\_FREE\_SPACE  WHERE TABLESPACE\_NAME='USERS';  SQL> SELECT segment\_name, file\_id, block\_id  FROM dba\_extents  WHERE owner = 'HR'  AND segment\_name LIKE 'DEPT%';  **Задание №7. Приложение поиска по метаданным Oracle**  Представьте себе, что стоит задача найти что-либо в исходном коде на Oracle, поискав информацию сразу на нескольких серверах. Речь идёт о поиске по объектам словаря данных Oracle. Рабочим местом для поиска является веб-интерфейс, куда пользователь-программист вводит искомую строку и выбирает галочками, на каких серверах Oracle осуществить этот поиск. Веб-поисковик умеет искать строку по серверным объектам Oracle одновременно в нескольких различных базах данных банка. Например, можно поискать:   * Где в коде Oracle зашита константа 61209, представляющая собой номер счёта второго порядка? * Где в коде и на каких серверах используется таблица accounts (в т.ч. через database link)? * С какого сервера из какой хранимой процедуры или триггера приходит сгенерированная программистом ошибка, например, ORA-20001 “Курс валюты не найден”? * Прописан ли планируемый к удалению индекс IX\_CLIENTID где-либо явным образом в хинтах оптимизатора в SQL-запросах? * Используются ли где-либо (в т.ч. через database link) планируемые к удалению таблица, поле, процедура, функция и т.д.? * Где в коде явно зашит чей-то е-мэйл или номер телефона? Такие вещи лучше выносить из серверных объектов в настроечные таблицы. * Где в коде на серверах используется зависящий от версии Oracle функционал? Например, функция wm\_concat выдаёт различный тип данных на выходе в зависимости от версии Oracle. Это может быть критично и требует внимания при миграции на более новую версию. * Где в коде используется какой-либо редкий приём, на который программисту хочется посмотреть, как на образец? Например, поискать в коде Oracle примеры использования функций sys\_connect\_by\_path, regexp\_instr или хинта push\_subq.   По результатам поиска пользователю выдаётся информация, на каком сервере в коде каких функций, процедур, пакетов, триггеров, вьюшек и т.п. найдены требуемые результаты. Опишем, как реализован такой поисковик.  Клиентская часть не сложная. Веб-интерфейс получает введённую пользователем поисковую строку, список серверов для поиска и логин пользователя. Веб-страница передаёт их в хранимую процедуру Oracle на сервере-обработчике. История обращений к поисковику, т.е. кто какой запрос выполнял, на всякий случай журналируется.  Получив поисковый запрос, серверная часть на поисковом сервере Oracle запускает в параллельных джобах несколько процедур, которые по database links на выбранных серверах Oracle сканируют следующие представления словаря данных в поисках искомой строки: dba\_col\_comments, dba\_jobs, dba\_mviews, dba\_objects, dba\_scheduler\_jobs, dba\_source, dba\_tab\_cols, dba\_tab\_comments, dba\_views. Каждая из процедур, если что-то обнаружила, записывает найденное в таблицу результатов поиска (с соответствующим ID поискового запроса).  Все выражения в SQL делятся на две основные категории:   * DDL - data definition language - язык определения данных; * DML - data manipulation language - язык манипулирования данными.   Язык DML необходим для изменения содержимого таблиц. Его операторы - это всем известные insert, update, delete. Для того, чтобы изменения стали видны всем пользователям базы данных, требуется выполнение оператора commit, а для возврата в предыдущее состояние (еще говорят для отката изменений) - оператора rollback.  DDL-выражения используются для создания и модификации объектов баз данных, таких как таблицы, процедуры, индексы. DDL-выражения, их еще называют метаданными, можно получить из уже существующей схемы. Они очень полезны при оптимизации и анализе баз данных. О том, как получить метаданные из БД Oracle и будет эта статья.  В более ранних версиях Oracle метаданные можно было получить с помощью SQL-запросов и разных утилит для экспорта. Эти способы имеют свои ограничения и недостатки. Ответом на все эти неудобства стал пакет **DBMS\_METADATA** с набором функций для работы с метаданными. Пакет DBMS\_METADATA появился в версии Oracle9i. Теперь определения данных доступны как в формате XML, так и в виде привычных нам DDL-выражений.  Основная функция, которая нам понадобится для извлечения метаданных - это get\_ddl. Вызывать ее можно как с двумя (тип объекта, имя объекта), там и с тремя (в дополнение еще и имя схемы) параметрами.  Основная функция, которая нам понадобится для извлечения метаданных - это get\_ddl. Вызывать ее можно как с двумя (тип объекта, имя объекта), там и с тремя (в дополнение еще и имя схемы) параметрами. Например, для получения метаданных о таблице ‘EMP’ можно использовать такое выражение:  **select** dbms\_metadata.get\_ddl(**'TABLE'**,  **'EMP'**) **from** dual;  conn HR/HR  **select** dbms\_metadata.get\_ddl(**'TABLE'**,  **'JOBS'**) **from** dual;    Для получения метаданных объекта конечно же необходимо знать его имя. Список объектов текущей схемы можно получить, например, так:  **select** \* **from** USER\_TABLES;  **select** \* **from** USER\_TAB\_COLUMNS;  **select** \* **from** USER\_INDEXES;  Это запросы для таблиц, колонок и индексов соответственно.  Данные по другим объектам базы данных находятся в таких таблицах:   * USER\_VIEWS - данные по представлениям (или как еще их ласково называют - вьюхи); * USER\_CONSTRAINTS - данные по первичным и вторичным ключам и проверкам типа not null; * USER\_SEQUENCES - данные по последовательностям-генераторам; * USER\_TRIGGERS - данные по триггерам; * USER\_SOURCE - данные по PL/SQL объектам, таким как процедуры, функции, пакеты.   **!!!!!!!! В СХЕМЕ “HR” cоздайте триггер,** чтобы заработная плата работника не уменьшалась.  create or replace trigger trg\_employees\_salary\_check  before update  on employees  for each row  begin  if :old.salary > :new.salary then  raise\_application\_error(-20111,'Sorry! Salary can not be decreased!');  end if;  end;  ПРОВЕРЬТЕ РЕЗУЛЬТАТ РАБОТЫ ТРИГГЕРА  SQL>update employees set salary = 10  where department\_id = 10;  НАЙДИТЕ ЭТОТ ТРИГГЕР  DESC USER\_TRIGGERS;  **select** trigger\_name **from** USER\_TRIGGERS;  **select** \* **from** USER\_TRIGGERS where trigger\_name=UPPER(’trg\_employees\_salary\_check’);  **Получение метаданных о триггере ’trg\_employees\_salary\_check’**  **select** dbms\_metadata.get\_ddl(**'TRIGGER'**,  UPPER(’trg\_employees\_salary\_check’)) **from** dual;   * + Как получить все имена столбцов, их тип данных и любые ограничения, определенные для любого столбца.   SET LINESIZE 32000;  SET PAGESIZE 40000;  SET LONG 50000;  select decode( t.table\_name  , lag(t.table\_name, 1) over(order by t.table\_name)  , null  , t.table\_name ) as table\_name -- <- just to eliminate  , t.column\_name -- repeated tab\_name  , t.data\_type  , cc.constraint\_name  , uc.constraint\_type  from user\_tab\_columns t  left join user\_cons\_columns cc  on (cc.table\_name = t.table\_name and  cc.column\_name = t.column\_name)  left join user\_constraints uc  on (t.table\_name = uc.table\_name and  uc.constraint\_name = cc.constraint\_name )  where t.table\_name in ('EMPLOYEES', 'DEPARTMENTS');  Также, чтобы получить полную спецификацию (при необходимости) таблицы, вы можете использовать dbms\_metadata пакет и get\_ddl функцию этого пакета:  select dbms\_metadata.get\_ddl('TABLE', 'EMPLOYEES') as table\_ddl  from dual;  **Задание №8.**  **Обязательные файлы:**  [Файлы данных (Data Files)](https://oracle-dba.ru/docs/architecture/files/#files1)  [Оперативные файлы журналов повтора (Online Redo Log Files)](https://oracle-dba.ru/docs/architecture/files/#files2)  [Управляющие файлы (Control Files)](https://oracle-dba.ru/docs/architecture/files/#files3)  [Файлы параметров pfile, spfie (Parameter Files)](https://oracle-dba.ru/docs/architecture/files/#files4)  **Необязательные файлы:**  [Архивные файлы журналов повтора (Archive Log Files)](https://oracle-dba.ru/docs/architecture/files/#files5)(в том смысле, что база может работать без них)  [Alert log и трассировочные файлы (trace file)](https://oracle-dba.ru/docs/architecture/files/#files6) (Alertlog - можно удалить, трассировочные по умолчанию не создаются)  [Файлы паролей (Password File)](https://oracle-dba.ru/docs/architecture/files/#files7)(Обычно не используются)  Следующий запрос, покажет, где находятся файлы данных.  SQL> set linesize 200;  SQL> set pagesize 0;  SQL> col name format a40;  SQL> select file#, name, status from v$datafile;  Каждая база данных должна иметь не менее двух оперативных файлов журналов повтора. Текущий файл постепенно заполняется, после его заполнения (или переключения некоторыми командами), база данных приступает к записи в следующий файл. Эта операция называется переключением журналов.  Поскольку файлы повтора необходимы для выполнения восстановления базы данных и являются критичными, их объединяют в группы. Запись происходит одновременно в файлы одной группы.  SQL> col member format a50;  SQL> select group#, member from v$logfile order by group#;  **Управляющие файлы (Control Files)**  Поскольку база данных Oracle является физическим набором связанных файлов данных, то для их синхронизации и контроля требуется особые методы. Для этих целей используются управляющие файлы. База данных Oracle может иметь один или несколько управляющих файлов. Если имеется несколько управляющих файлов, все они должны быть абсолютно идентичными. При каждом запуске базы данных Oracle читает информацию управляющего файла, а при каждом изменении размещения или добавления новых файлов данных и журналов базы данных обновляет управляющий файл.Управляющий файл критично важен для работы базы данных, и ее восстановление невозможно без доступа к свежему управляющему файлу. Oracle создает управляющий файл (и его копии) во время начального процесса создания базы данных.  SQL> col name format a100;  SQL> select name from v$controlfile;  При старте, Oracle считает файл spfileora112.ora. (файл серверных параметров). Преимущество spfile заключается в том, что при работе с базой данных, любые изменения в базе касающиеся изменения параметра системы, автоматически записываются в данный файл.  Если используется pfile, для сохранения изменений, необходимо либо “руками вносить эти изменения” в текстовый файл, либо в консоли выполнять команды для создания данных файлов Ораклом.  // создания pfile из памяти (в 11 версии Oracle)  SQL> create pfile from memory;  // создать pfile из spfile  SQL> Create pfile from spfile;  Как можно узнать, что база данных использует PFILE или SPFILE?:  Выполните следующий запрос, чтобы увидеть какой файл параметров был использован:  SQL> SELECT DECODE(value, NULL, 'PFILE', 'SPFILE') "Init File Type"  FROM sys.v\_$parameter WHERE name = 'spfile';  Архивные файлы журналов повтора жизненно важны при восстановлении. Если часть базы данных потеряна или повреждена, то для устранения повреждений обычно требуется несколько архивных журналов. Файлы журналов повтора должны применяться к базе данных последовательно. Если один из архивных файлов журналов повтора пропущен, то остальные архивные файлы журналов не могут использоваться. Храните все свои архивные файлы журналов повтора с момента выполнения последней резервной копии. Файлы журналов постепенно накапливаются и разрастаются. Иногда необходимо их удалять. Все операции с данными файлами по применению их к базе выполняются исключительно средствами базы данных. А копировать и переносить их при желании можно как угодно. Бездумно удалять их руками не рекомендуется.  SQL> col name format a100;  SQL> select name from v$archived\_log;  **Alert log и трассировочные файлы (trace file)**  При работе базы данных события и ошибки регистрируются в текстовых файлах на сервере базы данных. Файл журнала предупреждений (alert log) нужен администратору базы данных для отслеживания важнейших действий с базой данных - наподобие открытия и закрытия базы данных, установления параметров загрузки базы данных и переключения оперативных журналов повтора. Также в эти файлы записываются многие ошибки базы данных для последующего расследования их причин. Любые структурные изменения базы данных также регистрируются в файле журнала предупреждений.  // Следующий запрос покажет расположение трассировочных файлов.  SQL> show parameter dump\_dest  **Файлы паролей (Password File)**  Необязательный файл, используется для защиты информации о подключениях привилегированных пользователей. Если отсутствует, то вы можете выполнять администрирование своей базы данных, только локально. Кроме того, с его помощью контролируется количество привилегированных подключений для управления в одно и то же время. |

1. Структура базы данных Oracle. Блоки, экстенты, сегменты, табличные пространства.

**select** TABLESPACE\_NAME **from** dba\_tablespaces;

create tablespace myts1 datafile 'd:/myts1\_1.dbf' size 512k extent management local;

**select** TABLESPACE\_NAME **from** dba\_tablespaces;

create table myt1 (x int) storage(initial 256k next 256k) tablespace myts1;

select extent\_id, bytes, blocks from dba\_extents where segment\_name = 'MYT1';

alter table myt1 allocate extent;

select extents from dba\_segments where segment\_name = 'MYT1';

select blocks from dba\_extents where segment\_name = 'MYT1';

DROP TABLESPACE myts1 INCLUDING CONTENTS AND DATAFILES;

**select** TABLESPACE\_NAME **from** dba\_tablespaces;

**запрос sql, который будет извлекать всю информацию метаданных из схемы.**

CREATE PROCEDURE copy\_prc

AS

BEGIN

NULL;

END;

CREATE FUNCTION copy\_fun RETURN BOOLEAN

AS

BEGIN

RETURN TRUE;

END;

CREATE PACKAGE copy\_pkg AS

PROCEDURE pkg\_prc;

END copy\_pkg;

CREATE PACKAGE BODY copy\_pkg AS

PROCEDURE pkg\_prc

AS

BEGIN

NULL;

END;

CREATE TABLE source\_codes

(

s\_database VARCHAR2(30)

, s\_package\_name VARCHAR2(30)

, s\_source\_code CLOB

);

DECLARE

CURSOR objects\_cur IS

SELECT object\_name, DBMS\_METADATA.GET\_DDL(object\_type, object\_name) r\_source\_code

FROM dba\_objects

WHERE owner = USER

AND object\_type IN ('PROCEDURE', 'FUNCTION', 'PACKAGE')

AND object\_name IN ('COPY\_PRC', 'COPY\_FUN', 'COPY\_PKG');

TYPE objects\_ntt IS TABLE OF objects\_cur%ROWTYPE;

l\_objects objects\_ntt;

l\_database\_name VARCHAR2(100);

BEGIN

OPEN objects\_cur;

FETCH objects\_cur BULK COLLECT INTO l\_objects;

CLOSE objects\_cur;

l\_database\_name := sys\_context('USERENV', 'DB\_NAME');

FORALL indx IN 1..l\_objects.COUNT

INSERT INTO source\_codes

(

s\_database

, s\_package\_name

, s\_source\_code

)

VALUES

(

l\_database\_name

, l\_objects(indx).object\_name

, l\_objects(indx).r\_source\_code

);

COMMIT;

END;

SELECT s\_database, s\_package\_name, DBMS\_LOB.SUBSTR(s\_source\_code, 40)

FROM source\_codes;

/\*

my\_db\_name COPY\_FUN " CREATE OR REPLACE FUNCTION "SCHEMA\_NAME".""

my\_db\_name COPY\_PKG " CREATE OR REPLACE PACKAGE "SCHEMA\_NAME"."C"

my\_db\_name COPY\_PRC " CREATE OR REPLACE PROCEDURE "SCHEMA\_NAME"."

\*/

**Метаданные о схеме**

SQL> select owner,sum(bytes)/1024/1024 from dba\_segments where owner in ('EXAMPLE\_SCH') group by owner;

SQL> select tablespace\_name,sum(bytes)/1024/1024 from dba\_segments where owner in ('EXAMPLE\_SCH') group by tablespace\_name;

SQL> select default\_tablespace,username from dba\_users where username in ('EXAMPLE\_SCH');

SQL> select owner,status,object\_type,count(\*) from dba\_objects where owner in ('EXAMPLE\_SCH') group by object\_type,owner,status;

SQL> select object\_type,count(\*) from dba\_objects where owner in ('EXAMPLE\_SCH') group by object\_type;

SQL> select count(\*),status from dba\_objects where owner in ('EXAMPLE\_SCH') group by status;

SQL> select object\_type,count(\*),status from dba\_objects where owner in ('EXAMPLE\_SCH') group by status,object\_type;

SET это команда клиентского приложения, которая устанавливает его внутреннею системную переменную SERVEROUTPUT {ON|OFF}. Эту команду поддерживат: SQL\*Plus, SQL Developer и некоторые другие, но она не является частью синтаксиса PL/SQL или SQL. Её надо выполнить до запуска PL/SQL блока:

create or replace procedure proc is

begin

    dbms\_output.put\_line ('line 1');

    dbms\_output.put\_line ('line 2');

end;

/

set serveroutput on

exec proc

line 1

line 2

Что делает

SET SERVEROUTPUT ON

 "под капотом", поможет понять следующий пример. Отключим вывод, сбросив пременную

SERVEROUTPUT OFF

, и выполним всё, что делалось неявно с

SERVEROUTPUT ON

, теперь вручную. Сначала, заново включим вывод:

set serveroutput off

exec dbms\_output.enable ();

Запустим процедуру:

exec proc

PL/SQL procedure successfully completed.

Во время выполнения процедуры, каждый вызов

dbms\_output.put\_line('some text');

 будет добавлять одну строчку во внутренний массив строк, который сохраняется в области переменных сессии. Никакого вывода на консоль не будет, ни во время выполнения, ни после завершения. Это задача клиента - получить содержимое внутреннего массив строк и вывести его на консоль. Это выглядит как-то так:

var output varchar2

declare

    lines dbms\_output.CHARARR;

    nl int := 100;

begin

    dbms\_output.get\_lines (lines, nl);

    for i in 1..nl loop

        :output := :output||lines(i)||chr(10);

    end loop;

end;

/

PL/SQL procedure successfully completed.

print output

line 1

line 2