Администрирование БД в Oracle

Материалы учебного курса

Владимир Викторович Пржиялковский
prz@yandex.ru
open-oracle.ru

Ноябрь 2019

Москва

... «мелкие предприятия недостойны римлян, а с другой стороны —совершить нечто великое без напряжения сил никому не дано и доступно разве одному божеству».
Тит у Иосифа Флавия в «Иудейской войне»

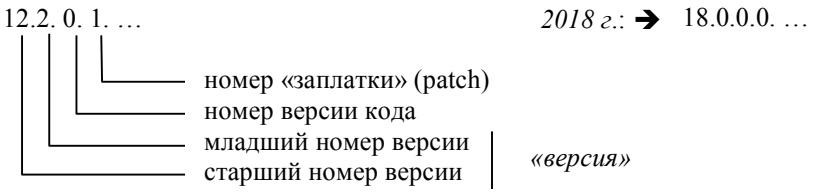
Оглавление

1. Общая информация о СУБД Oracle

1.1. Введение в Oracle

1.1.1. Версии и разновидности Oracle

Нумерация версий Oracle строится по принципу:



Версией 18.0 (18.3) была названа заплатка 12.2.0.2 (заплатка 2 к версии 12.2) в связи с решением фирмы перейти к ежегодному выпуску новых версий. Версией 19.3 названа 12.2.0.4, объявленная «последней в семействе 12.2» (с расширенной поддержкой до 2026 г.).

Разновидности ПО СУБД:

- Enterprise Edition (основной)
  - → Standard Edition (с ограниченной функциональностью и степенью распараллеливания вычислений)
    - → Standard Edition One (без возможности конфигурации RAC некоторыми дополнительными ограничениями; версиии 11-)
    - → Standard Edition Two (версиии 12+)
  - → Personal Edition (удешевленный вариант для разработчика)
  - → Express Edition (с версии 10.2, бесплатный; в версии 11 с ограничением 11 Гб файлов и 1 Гб ОЗУ)
- Инородные: Oracle Lite (легкий вариант для мобильных устройств), Berkley DB (с 2006 г.), TimesTen и mySQL (с 2005 г.), Rdb, No-SQL.

Сравнительный пример возможностей СУБД для типовых конфигураций Enterprise, Standard и Express версии 10 (получено из таблицы V$OPTION):

PARAMETER	ENTERPRISE	STANDARD	EXPRESS
Partitioning	TRUE	FALSE	FALSE
Objects	TRUE	TRUE	TRUE
Real Application Clusters	FALSE	FALSE	FALSE
Advanced replication	TRUE	FALSE	FALSE
Bit-mapped indexes	TRUE	FALSE	FALSE
Connection multiplexing	TRUE	TRUE	TRUE
Connection pooling	TRUE	TRUE	TRUE
Database queuing	TRUE	TRUE	TRUE
Incremental backup and recovery	TRUE	TRUE	TRUE
Instead-of triggers	TRUE	TRUE	TRUE
Parallel backup and recovery	TRUE	FALSE	FALSE
Parallel execution	TRUE	FALSE	FALSE
Parallel load	TRUE	TRUE	TRUE
Point-in-time tablespace recovery	TRUE	FALSE	FALSE
Fine-grained access control	TRUE	FALSE	FALSE
Proxy authentication/authorization	TRUE	TRUE	TRUE
Change Data Capture	TRUE	FALSE	FALSE
Plan Stability	TRUE	TRUE	TRUE
Online Index Build	TRUE	FALSE	FALSE

```
Coalesce Index                          TRUE       FALSE      FALSE
Managed Standby                         TRUE       FALSE      FALSE
Materialized view rewrite               TRUE       FALSE      FALSE
Materialized view warehouse refresh     TRUE       FALSE      FALSE
Database resource manager               TRUE       FALSE      FALSE
Spatial                                 TRUE       FALSE      FALSE
Visual Information Retrieval            TRUE       FALSE      FALSE
Export transportable tablespaces        TRUE       FALSE      FALSE
Transparent Application Failover        TRUE       TRUE       TRUE
Fast-Start Fault Recovery               TRUE       FALSE      FALSE
Sample Scan                             TRUE       TRUE       TRUE
Duplexed backups                        TRUE       FALSE      FALSE
Java                                    TRUE       TRUE       FALSE
OLAP Window Functions                   TRUE       TRUE       TRUE
Block Media Recovery                    TRUE       FALSE      FALSE
Fine-grained Auditing                   TRUE       FALSE      FALSE
Application Role                        TRUE       FALSE      FALSE
Enterprise User Security                TRUE       FALSE      FALSE
Oracle Data Guard                       TRUE       FALSE      FALSE
Oracle Label Security                   TRUE       FALSE      FALSE
OLAP                                    TRUE       FALSE      FALSE
Table compression                       TRUE       FALSE      FALSE
Join index                              TRUE       FALSE      FALSE
Trial Recovery                          TRUE       FALSE      FALSE
Data Mining                             TRUE       FALSE      FALSE
Online Redefinition                     TRUE       FALSE      FALSE
Streams Capture                         TRUE       FALSE      FALSE
File Mapping                            TRUE       FALSE      FALSE
Block Change Tracking                   TRUE       FALSE      FALSE
Flashback Table                         TRUE       FALSE      FALSE
Flashback Database                      TRUE       FALSE      FALSE
Data Mining Scoring Engine              TRUE       FALSE      FALSE
Transparent Data Encryption             TRUE       FALSE      FALSE
Backup Encryption                       TRUE       FALSE      FALSE
Unused Block Compression                TRUE       FALSE      FALSE
```

Список получен для установки без Real Application Clusters (RAC, бывший «параллельный сервер»; отсюда значение `Real Application Clusters = FALSE`). Для Standard Edition вариант RAC стал допускаться только с версии 10 и для количества процессоров узла кластера не больше 4.

В версии 11 из списка выпали строки:

`Visual Information Retrieval` *(заменено на поддержку* `DICOM`[(*)]*)*
`Table compression` *(заменено на* `Basic Compression`*)*
`Data Mining Scoring Engine` *(рассматривается как часть* `Data Mining`*)*

(*) Digital Imaging and COmmunications in Medicine, промышленный стандарт создания, хранения, передачи и визуализации медицинских изображений и документов обследованных пациентов.

Одновременно, в версии 11 список дополнен следующими позициями:

```
PARAMETER                               ENTERPRISE EXPRESS
--------------------------------------- ---------- ----------
Automatic Storage Management            FALSE      FALSE
Basic Compression                       TRUE       FALSE
Oracle Database Vault                   FALSE      FALSE
Result Cache                            TRUE       FALSE
SQL Plan Management                     TRUE       FALSE
SecureFiles Encryption                  TRUE       FALSE
Real Application Testing                TRUE       FALSE
Flashback Data Archive                  TRUE       FALSE
DICOM                                   TRUE       FALSE
```

```
Active Data Guard                          TRUE       FALSE
Server Flash Cache                         TRUE       FALSE
Advanced Compression                       TRUE       FALSE
XStream                                    TRUE       FALSE
Deferred Segment Creation                  TRUE       FALSE
```

Приведенный список получен в отсутствии установленного ПО для Automatic Storage Management (ASM) и Oracle Database Vault.

1.1.2. Расширения базовой поставки

Каждой версии СУБД присущ собственный набор расширений базовой поставки (server options). Например, в новых версиях прежние расширения могут оказаться включенными в число основных свойств.

1.1.2.1. Функциональные расширения

Лицензируются отдельно (версия 10):

- *Oracle Advanced Security* (до версии 9 — *Advanced Networking Option*, ANO): Дополнительные возможности по части защищенности сетевого обмена.
- *Oracle Label Security*: Возможность ограничения доступа к строкам таблицы методом «мандатного доступа».
- *«Real Application Clusters»* (RAC; в версии 9 — *параллельный сервер, Parallel Server Option*, OPS): Специальная конфигурация Oracle для работы на компьютерном кластере; когда несколько экземпляров СУБД на узлах кластера обслуживают доступ к одной БД с данными в общем дисковом пространстве.
- *Partitioning Option*: Средство разнесения строк одной таблицы по разным структурам хранения («разделам») по заданному признаку. Используется для ускорения доступа к данным особенно больших таблиц.
- *Oracle OLAP*: Возможность аналитической обработки данных в БД по технике OLAP.
- *Oracle Data Mining*: Возможность аналитической обработки данных в БД по принципам data mining.
- Несколько прочих: *Oracle Content Database Suite*, *Oracle Database Vault*, *Oracle In-Memory Database Cache*, *Oracle Real Application Testing*, *Oracle Records DB*.

В версии 11 к ним добавились:

- *Oracle Active Data Guard*, *Oracle Advanced Compression*, *Oracle Total Recall*.

В версиях 11- активировать и деактивировать функциональные расширения СУБД предполагалось установщиком OUI, с версии 11 предполагается отдельной программой *chopt* (например: *chopt enable partitioning*).

Для версии 12.2 список дается *здесь*.

Нередко какие-то расширения в очередной версии СУБД становятся штатными; например, *Procedural Option* (версия 6), *Object Option* (версия 8), *Oracle Context* (до версии 9).

1.1.2.2. Расширения типов данных

Относительно простой набор базовых типов данных в Oracle может быть расширен возможностью работы с более сложными типами. Для этого служит специальная техника построения «картриджей» (метафора «кассеты» вставляемой в «устройство» при необходимости; аналог DataBlades в Informix или Extenders в DB2). Пример такого готового расширения, шедшего в версиях 7 — 8 как дополнительная возможность, а с версии 9 включенного в основной состав ПО СУБД:

- *Oracle Text* = *Text Cartridge* — хранение и поиск текстовых документов.

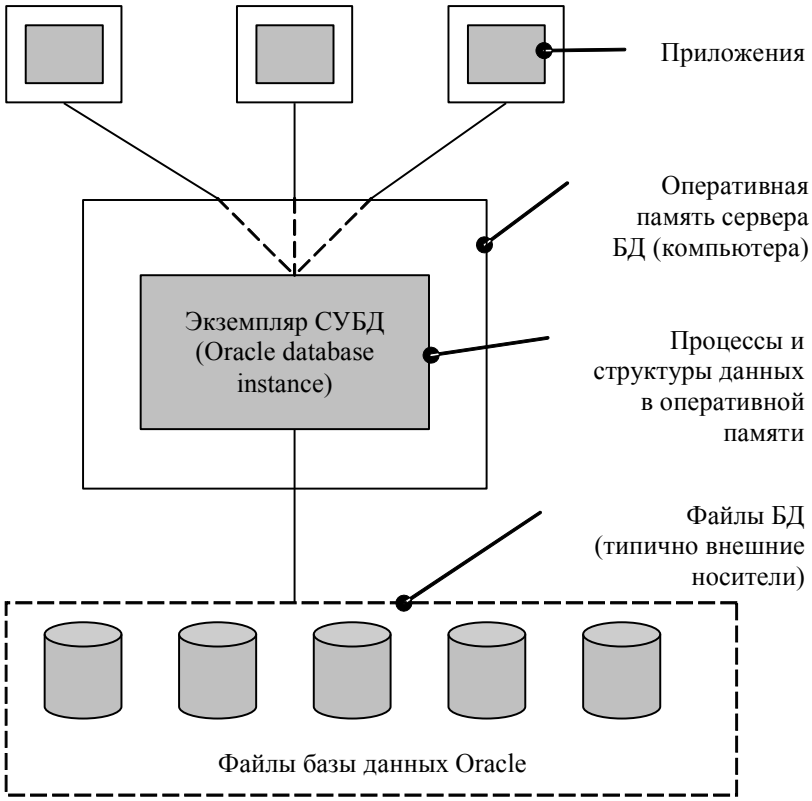
Помимо этого поставляется ряд готовых расширений типов для «сложных» данных, выполненых по «объектно-реляционной» технике и сгруппированых под «зонтичным» названием interMedia (до версии 10 включительно)/Multimedia (с версии 11; в соответствии с термином из стандарта на SQL); например:

- *тип ORDAUDIO* — хранение и использование аудиоинформации;
- *тип ORDIMAGE* — хранение и использование изображений;
- *тип ORDVIDEO* — хранение и использование видеоизображений;
- *тип ORDDOC* — хранение и использование разнородных «сложных» данных (аудио-, изо- и видео-) в одном столбце таблицы;
- *тип ORDSOURCE* — осуществление доступа к разнородным «сложно-устроенным» данным, расположенным в разных местах, в том числе вовне БД;
- *тип ORDDICOM* — хранение и использование медицинских данных в формате DICOM (с версии 11.2).

1.2. Общая архитектура Oracle

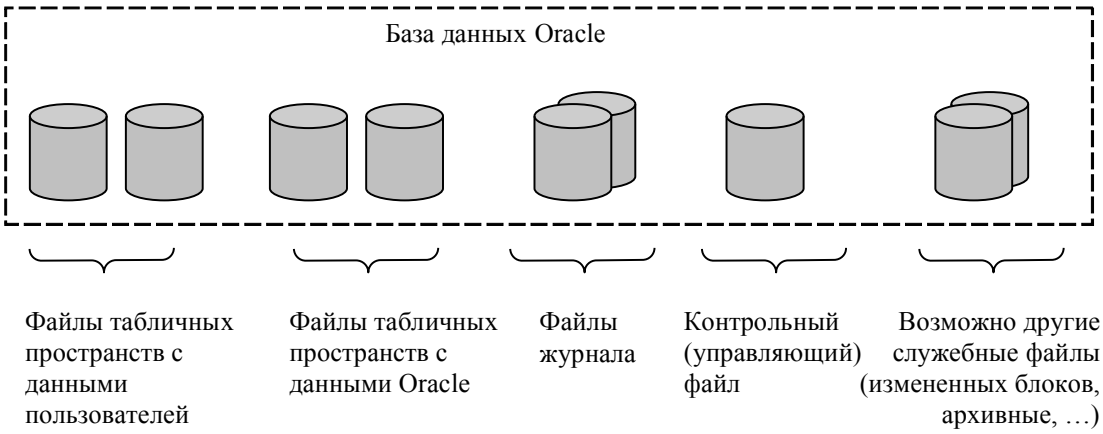
1.2.1. Основные элементы архитектуры

Общий вид соотношения между БД, экземпляром СУБД и сервером:



1.2.1.1. Файлы БД

В Oracle БД технически состоит всегда из комплекта нескольких файлов. Часть файлов используется для хранения самих данных базы, а часть носит чисто вспомогательный характер:

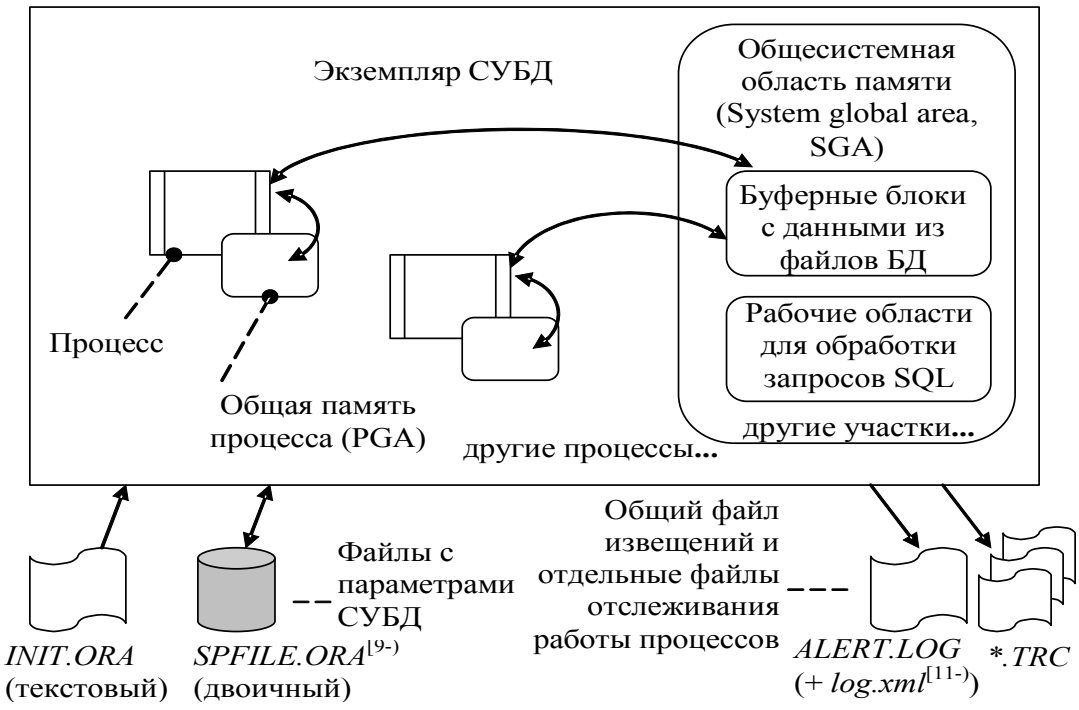


К ним можно еще добавить файлы, связанные с резервированием и восстановлением, если они используются.

Файлы БД чаще всего являются участниками файловой системы ОС, но с версии 10 постепенно приобретает популярность их размещение в собственной («автоматической») системе Oracle по управлению дисковым пространством (ASM); в этом случае работать с ними можно исключительно средствами Oracle, но не файловой системы ОС.

1.2.1.2. Общее устройство СУБД

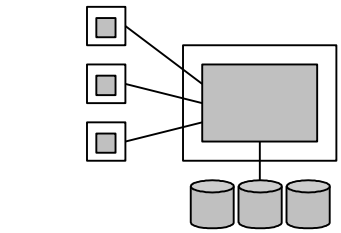
Экземпляр СУБД, то есть конкретная программа, управляющая доступом к конкретной БД в Oracle всегда состоит из набора процессов и набора внутренних структур памяти:



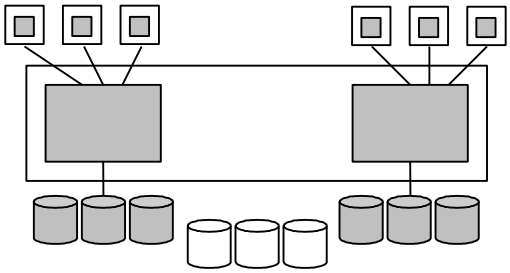
[9-) Начиная с версии 9.
[11-) Начиная с версии 11.

1.2.2. Разновидности рабочих конфигураций

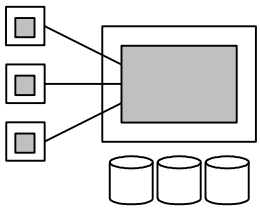
Ниже приведены разные сочетания БД, СУБД, компьютеров и прикладных программ при использовании Oracle:



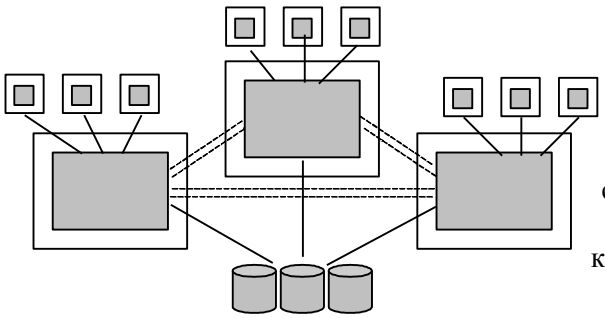
Наиболее распространенная конфигурация (single instance)



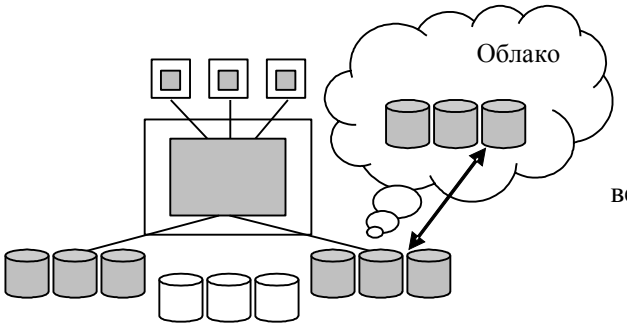
Возможная конфигурация (single instance)



СУБД работает без подключения к БД (возможный рабочий момент в использовании БД)



RAC (real application cluster, «истинный кластер для приложений»), кластерная организация БД



Контейнерная и подключаемые БД с версии 12 (container – pluggable)

1.2.3. Службы БД

Типично одна БД Oracle обслуживает разные задачи приложений. В то же время, в общем случае одна и та же задача может решать свои потребности по работе с данными через разные СУБД (например, при наличии кластерной конфигурации RAC, или в случае использования пары «основная — резервная БД»).

Oracle разрешает отношению «задача — СУБД» быть типа «многие ко многим». Для удобств организации приложения и администрирования фирма Oracle ввела абстракцию «служба БД».

Каждая «служба БД» объединяет собой множество сеансов, порожденных приложениями, что позволяет следить за этими сеансами в целом. Принадлежность сеанса службе БД клиент задает (сообщает) в момент открытия сеанса. Служба, на которую ссылается клиент, открывая сеанс, должна быть заранее создана и запущена, иначе сеанс связи не откроется. Запуск служб БД осуществляется администратором вручную или через использование средств автоматизации. Одна служба, с глобальным именем БД, существует изначально, а все остальные должны создаваться администратором по мере необходимости (наличие XML DB в составе БД порождает еще одну «изначальную» службу, но уже специализированную, а не общего характера).

1.3. Задачи администрирования БД

На задачи администрирования Oracle может иметься по крайней мере три точки зрения:

(1) точка зрения работодателей
(2) усредненная точка зрения практикующих администраторов
(3) точка зрения представителей фирмы Oracle

(1) См. объявления о найме на работу АБД.

(2) Мнения пользователей Oracle о том, что должен уметь АБД.

Пример. Опрос Educational Testing Service, США, *www.ets.org*, вторая половина 90-х:

Архитектура и варианты функционирования Oracle. Безопасность. Администрирование данных. Резервное копирование и восстановление. Поддержка функционирования ПО. Управление объектами БД. Настройка и отладка.

Близкие точки зрения:
http://en.wikipedia.org/wiki/Database_Administration,
https://www.orafaq.com/wiki/Roles_and_Responsibilities.

(3) Программы экзаменов на получение сертификатов фирмы Oracle на умение администрировать БД (здесь: 12.1; *http://www.oracle.com/*):

1Z0-062: Oracle Database 12c: Installation and Administration
Oracle Database Administration
Exploring the Oracle Database Architecture
Oracle Database Management Tools
Oracle Database Instance
Configuring the Oracle Network Environment
Managing Database Storage Structures
Administering User Security
Managing Space
Managing Undo Data
Managing Data Concurrency
Implementing Oracle Database Auditing
Backup and Recovery Concepts
Backup and Recovery Configuration
Performing Database Backups
Performing Database Recovery
Moving Data
Performing Database Maintenance
Managing Performance
Managing Performance: SQL Tuning

Managing Resources Using Database Resource Manager Automating Tasks by Using Oracle Scheduler **Installing, Upgrading and Patching the Oracle Database** Oracle Software Installation Basics Installing Oracle Grid Infrastructure for a Standalone Server Installing Oracle Database Software Creating an Oracle Database Using DBCA Using Oracle Restart Upgrading Oracle Database Software Preparing to Upgrade to Oracle Database 12c Upgrading to Oracle Database 12c Performing Post-Upgrade Tasks Migrating Data by Using Oracle Data Pump *подробнее ...*
1Z0-063: Oracle Database 12c: Advanced Administration
Backup and Recovery Oracle Data Protection Solutions Performing Basic Backup and Recovery Configuring for Recoverability Using the RMAN Recovery Catalog Implementing Backup Strategies Performing Backups Configuring RMAN Backup Options and Creating Backup of Non-Database Files Using RMAN-Encrypted Backups Diagnosing Failures Performing Restore and Recovery Operations Recovering Files Using RMAN Using Oracle Secure Backup Using Flashback Technologies Using Flashback Database Transporting Data Duplicating a Database Monitoring and Tuning of RMAN Operations **Managing Pluggable and Container Databases** Introduction Multitenant Container and Pluggable Database Architecture Creating Multitenant Container and Pluggable Databases Managing a CDB and PDBs Managing Storage in a CDB and PDBs Managing Security in a CDB and PDBs Managing Availability Managing Performance Moving Data, Performing Security Operations and Interacting with Other Oracle Products *подробнее ...*

Более обобщенно — в документации (здесь: по 12.1).

Три приведенные точки зрения близки, но не во всем совпадают.

1.4. Ресурсы знаний

- Документация.
  - документация на русском языке
- Печатные издания:

- журналы: Oracle Magazine, Select, Oracle Professional, др.
- книги: на русском и английском языке

- Сетевые места:
  - *www.oracle.com*, включая:
    - → *technet.oracle.com* (Oracle technology network, OTN; для всех желающих, но нужно зарегистрироваться)
    - → *support.oracle.com* (для лицензированных пользователей Oracle; ранее *metalink.oracle.com*)
  - *www.ioug.org* (IOUG, независимая группа пользователей Oracle)
  - блоги, частные и корпоративные страницы, и так далее.
- Группы новостей:
  - *comp.databases.oracle.server* (*groups.google.com/group/comp.databases.oracle.server/topics?lnk*), *comp.databases.oracle.tools* (*groups.google.com/group/comp.databases.oracle.tools/topics?lnk*)
  - *fido7.ru.rdbms.oracle* (*groups.google.com/group/fido7.ru.rdbms.oracle/topics*, с 1992 г.)
  - *relcom.comp.dbms.oracle* (*groups.google.com/group/relcom.comp.dbms.oracle/topics*)
  - др.
- Списки рассылки. Единственным активным списком по тематике СУБД сегодня является:
  - *Oracle-L* (*www.freelists.org/list/oracle-l*)

Упражнение. Открыть электронную документацию на CD, ознакомиться с ее устройством и возможностями поиска сведений по СУБД.

2. Установка Oracle

Здесь говорится об установке ПО СУБД Oracle на сервере (*не* о разворачивании БД в Облаке).

2.1. Местонахождение Oracle в операционной и файловой системе

Места Windows, затрагиваемые установкой Oracle:

- Каталоги ПО СУБД и БД (хранение файлов программных компонент, файлов БД и ПО СУБД, сценариев, вспомогательных и т.д.). О рекомендуемой организации см. ниже.
- Служебные каталоги (для работы установщика и пр.; в версиях 8.1+):
  - → описание установленного ПО СУБД: *\Program Files\Oracle\Inventory*
- Реестр; в том числе разделы с параметрами запуска и работы Oracle:
  - → *HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE*
  - → о службах (services) и режимах запуска программных компонент в разделе *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services*
- Переменные окружения командной оболочки ОС (*PATH*, *ORACLE_HOME*, др.).
- Меню Start в ОС.

Содержимое директории *Inventory* на деле отсылает к реестру (раздел *HKLM\Software\Oracle*: `>reg query HKLM\Software\Oracle`), однако формально отвечает за перечень установленного ПО именно она.

Места, затрагиваемые установкой Oracle в Unix/Linux:

- Каталоги ПО СУБД и БД. О рекомендуемой организации см. ниже.
- Служебные каталоги (для работы установщика и пр.; в версиях 8.1+):
  - → описание установленного ПО СУБД: */opt/oracle/oraInventory* (вариант: */u01/app/oraInventory*)
- Файл *oratab* (каталог */etc* в AIX, HP-UX, Linux или */var/opt/oracle* в Solaris), используемый для автозапуска СУБД и открытия БД.
- Файл */etc/init.d/oracle* (*/etc/init.d/oracleasm*) с командами запуска, останова и проверки состояния элементов ПО Oracle (в зависимости от конфигурации возможно и некоторые другие файлы в */etc/init.d* или же добавления в и без того существующие).
- Переменные окружения командной оболочки ОС (*PATH*, *ORACLE_HOME*, др.).

2.1.1. Рекомендуемая структура каталогов для Oracle

Некоторые файлы ПО СУБД и БД имеют фиксированное местонахождение, а часть может располагаться произвольно. Ниже приводится *рекомендуемая* схема организации каталогов для них. Такая схема обладает определенной гибкостью и удобством употребления. Однако следовать ей пунктуально не обязательно, и в жизни нередко так и происходит.

2.1.1.1. В версиях 10-

Optimal Flexible Architecture (OFA): разработана для Unix, но ее разумно придерживаться и на других платформах. Со временем она претерпевала изменения. Первоначальный ее вид может быть пояснен следующим примером:

/u00/oracle/product/7.3.4/…… */u00/oracle/product/8.1.5/……*	← каталоги файлов программного обеспечения СУБД
/u01/oracle/oradata/DBX/…… */u02/oracle/oradata/DBY/……* */u03/oracle/oradata/DBY/……*	← каталоги файлов баз данных
/u00/oracle/admin/DBXinst1/…… */u00/oracle/admin/DBXinst2/……*	← каталоги файлов СУБД (параметры, трассировка и пр.)

Здесь для примера учтены *четыре* дисковые устройств, *две* заведенные БД и *две* разные версии ПО Oracle — все в рамках одного компьютера.

Обозначения приведены для Unix. На Windows роль пронумерованых устройств играют буквы, например: ***f:****oracle**product*\….

Ключевые особенности OFA:

- Каждому диску сопоставляется корневой каталог, именованный общей составляющей и собственным последовательным номером.
- Каждый такой корневой каталог имеет подкаталог *oracle*, в котором хранятся файлы, связанные с Oracle. Каталог с ПО Oracle и каталогами БД/СУБД (в примере — */u00/oracle*) обозначается как ORACLE_BASE.
  - Один из подкаталогов ORACLE_BASE хранит ПО конкретной версии СУБД (полностью на одном диске), обозначается как ORACLE_HOME.
  - Один из подкаталогов ORACLE_BASE содержит файлы БД. Файлы данных одной БД могут находиться на разных дисках.
  - Один из подкаталогов ORACLE_BASE содержит файлы, используемые СУБД.

Пример для версии 10:

```
/u01/oracle/admin/ORACLE_SID/bdump    – ALERT.LOG и *.trc фоновых процессов СУБД
            |                 /cdump    – участки оперативной памяти при сбоях
            |                 /udump    – *.trc серверных процессов СУБД
            |                 ...
            /oradata/DB_NAME            – файлы БД
            /product/10.2.0/db_1        – файлы ПО СУБД
            ...
```

Здесь:
ORACLE_SID обозначает действительное имя СУБД,
DB_NAME — локальное имя БД,
ORACLE_HOME — */u01/oracle/product/10.2.0/db_1*.

Типичные отклонения от этой организации каталогов:

- В типовых конфигурациях OUI может размещать *admin* и *oradata* в */u01/oracle/product/10.2.0*).
- С версии 10 в конкретных случаях каталог, обозначенный выше как *oradata*, может отсутствовать при условии, что для хранения файлов БД используется ASM.

Полные имена каталогов *bdump*, *cdump* и *udump* содержатся в переменных СУБД
BACKGROUND_DUMP_DEST,
CORE_DUMP_DEST и
USER_DUMP_DEST
соответственно.

Значения для ORACLE_BASE и ORACLE_HOME могут (а часто должны) быть заранее установлены одноименными переменными командной оболочки ОС, а в Windows, сверх того, — одноименными ключами реестра ОС. В *%ORACLE_HOME%\bin* имеется файл *oracle.key*, куда OUI заносит точное название раздела в *HKEY_LOCAL_MACHINE* реестра Windows, где хранятся характеристики установленного ПО, включая точную расшифровку ORACLE_BASE.

2.1.1.2. В версиях 11 — 12

В версии 11 *рекомендации* по расположению файлов, используемых СУБД, подверглись изменениям с расчетом на более общую конфигурацию RAC (кластерную) и расположение этих файлов на общем для всех экземпляров СУБД (и для кластера) дисковом пространстве:

- Каталог ORACLE_BASE с ПО Oracle называется по схеме: *u*nn*/app/*LOGNAME.
  - Подкаталог ORACLE_BASE *diag/rdbms/*DB_UNIQUE_NAME*/*ORACLE_SID содержит файлы, используемые СУБД.
  - Подкаталог ORACLE_BASE *product/<номер_версии>/dbhome_*n соответствует ORACLE_HOME.

Здесь ORACLE_SID обозначает действительное имя СУБД, DB_UNIQUE_NAME — локальное имя БД, а LOGNAME — имя учетной записи (пользователя) ОС, пользуясь которой осуществлялась установка ПО. (В Windows имени переменной ОС LOGNAME соответствует USERNAME.)

Каталог *$ORACLE_BASE/diag* определяет точку отсчета для введенной в версии 11 новой схемы правил организации диагностических файлов, получившнй название Automatic Diagnostic Repository (ADR, автоматический репозитарий диагностических данных). На его полное название указывает новый для версии 11 параметр СУБД DIAGNOSTIC_DEST.

Пример для версии 11.2:

```
/u01/app/LOGNAME/diag/rdbms/DB_UNIQUE_NAME/ORACLE_SID/trace
                 |    |                      - ALERT.LOG и *.trc процессов
                 |    |
                 |    ...
                 /oradata/DB_UNIQUE_NAME     - файлы БД
                 /product/11.2.0/dbhome_1    - файлы ПО СУБД
                 ...
```

Здесь:
ORACLE_HOME — это */u01/app/*LOGNAME*/product/11.2.0/dbhome_1*.

При использовании для хранения файлов БД техники ASM, каталог файловой системы ОС, обозначенный выше как *oradata*, может отсутствовать.

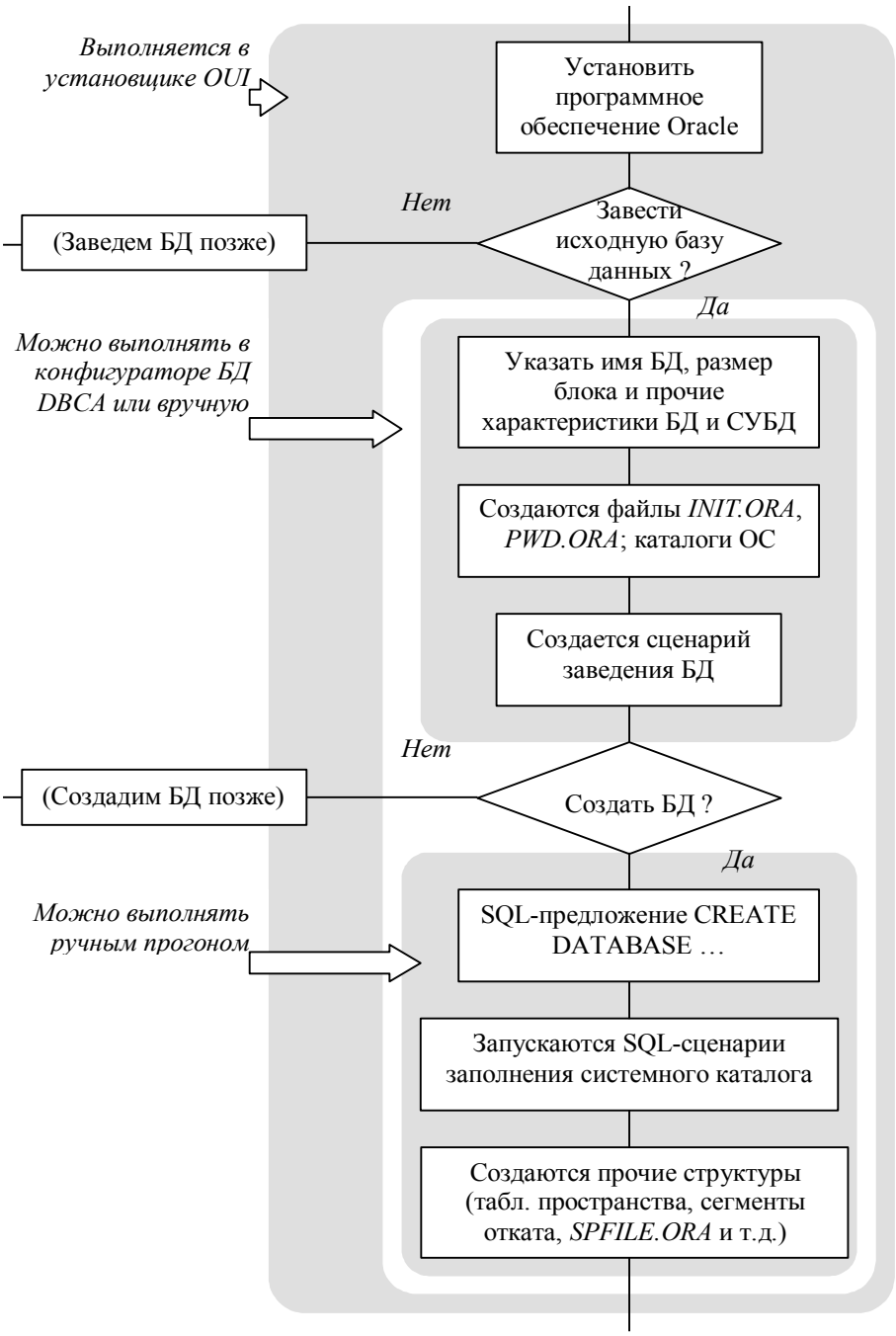
2.1.1.3. В версиях 18+

В версии 18 фирма фактически упразднила связь ORACLE_BASE и ORACLE_HOME, предложив администратору то и другое устанавливать по отдельности и произвольно. Структура директорий в ORACLE_BASE, при этом (за вычетом цепочки к ORACLE_HOME), сохранилась прежняя.

2.2. Общая схема установки Oracle

Для установки ПО Oracle и наложения заплат (patchsets) с версии 8.1 используется общая для всех платформ (и большей части продуктов фирмы Oracle) программа Oracle Universal Installer (OUI, установщик ПО Oracle), а для заведения БД — Database Configuration Assistant (DBCA, помощник конфигурирования БД). Внешнее оформление работы программ может изменяться от версии Oracle к версии, в то время как конкретная «траектория» установки зависит от ответов администратора.

Обобщенная схема действий установщика при установке ПО СУБД неизменна во всех версиях:



OUI способен как установить исключительно ПО СУБД, так и заодно запустить модули DBCA для создания (первой с этим ПО на сервере) БД.

2.3. Основные этапы установки

2.3.1. Установка ПО СУБД Oracle

Выполняется в OUI путем выбора (а) одного из предлагаемых типовых вариантов (с небольшим числом уточняющих вопросов) или (б) элементов из предлагаемого списка компонентов ПО (вариант Custom, или Advanced). Во втором случае с помощью OUI можно еще доустанавливать новые компоненты ПО (например, для расширений БД или администрирования, специфичного для Windows), а также убирать ранее установленные.

Установщик вызывается с установочного диска запуском программы *setup.exe* в Windows и *runInstaller* в Unix/Linux. При установленном ПО установщик можно вызвать запуском *\%ORACLE_HOME %\oui\bin\setup.exe* или из Start-меню в Windows и *$ORACLE_HOME/oui/bin/runInstaller* в Unix/Linux.

2.3.2. Предшествующая установка ПО Oracle

Установке ПО СУБД в кластерном варианте Oracle RAC *должна* предшествовать установка (на кластер) ПО Oracle Grid Infrastructure. В состав последнего входит ПО: Oracle Clusterware, listener, Oracle ASM. Oracle Clusterware обеспечивает управление кластерной архитектурой.

Установке ПО СУБД в варианте «одна БД — одна СУБД» (single instance database) *может* предшествовать (если администратор сочтет необходимым) установка (на сервер) ПО Oracle Grid Infrastructure. В состав последнего входит в этом случае: Oracle Restart, listener, Oracle ASM. Oracle Restart будет автоматически перезапускать компоненты ПО Oracle в случае их отказа.

Доустановить на сервер ПО Grid Infrastructure после установки ПО СУБД невозможно.

2.3.3. Формирование характеристик БД и СУБД

Этот этап открывает фазу процедуры заведения БД (осуществляемую фактически программными модулями DBCA). Поскольку для использования любой БД требуется обслуживающая ее СУБД, при заведении БД должны быть сформированы (автоматически или явными указаниями администратора) одновременно параметры не только самой БД, но и СУБД.

Если БД заводится отдельно от установки ПО и не вручную, программу-помощник для установки можно вызвать обращением к *\%ORACLE_HOME%\bin\dbca.bat* или из Start-меню в Windows и *$ORACLE_HOME/bin/dbca* в Unix/Linux.

Все указываемые на этом этапе администратором характеристики, вплоть до имени БД, можно будет впоследствии поменять (хотя и с разной степенью легкости) за исключением (а) размера блока для табличных пространств базы (с версии 9 — «основного» размера блока системных табличных пространств), и (б) за редким исключением — кодировки текстовых данных в БД.

С версии 9 установщик ПО OUI (и конфигуратор БД DBCA) может при желании запомнить введенные пользователем характеристики в т. н. «шаблоне» (template), то есть в виде файла в формате XML. Созданные «шаблоны» можно использовать для упрощения заведения однотипных баз на разных компьютерах, вплоть до пакетного способа.

Если заведение БД делается не вручную, а графическими средствами (OUI или DBCA), можно воспользоваться типовыми наборами характеристик, и по отдельности их не указывать.

2.3.4. Заведение инфраструктуры для размещения планируемой БД

Этап нужен для заведения в ОС каталогов для файлов БД и обслуживающей СУБД, а также для заведения некоторых файлов.

2.3.4.1. Файл с параметрами СУБД

Файл параметров, задающий конфигурацию конкретного экземпляра СУБД при формировании его в оперативной памяти компьютера. Без этого файла невозможен запуск СУБД. До версии 9 использовался один файл параметров с названием *INIT.ORA*, а начиная с версии 9 файлов параметров два: *INIT.ORA* и *SPFILE.ORA*. Оба названия носят жаргонный характер, так как в жизни их имена обычно включают имя СУБД (в отдельных случаях могут и не включать).

Версия СУБД	***Файл***	***Формат***	***Обычное расположение***

все версии	*INIT.ORA*	текстовый	*%ORACLE_HOME%**database*** (Windows)[(*)] *$ORACLE_HOME/**dbs*** (Unix)[(*)]
9+	*SPFILE.ORA*	двоичный	*%ORACLE_HOME%**database*** (Windows), *$ORACLE_HOME/**dbs*** (Unix), дисковые группы ASM[10-)]

[(*)] Текстовый файл *INIT.ORA* имеет право именоваться и располагаться произвольно, но в таких случаях при запуске СУБД на него потребуется ссылаться явно.
[10-)] С версии 10.

Часть параметров СУБД допускает изменение значений только путем перезапуска СУБД (*статические* параметры); другая часть допускает изменения по ходу работы СУБД командами ALTER SYSTEM … или даже ALTER SESSION … (*динамические* параметры, доступные для изменения на уровне СУБД или даже отдельного сеанса). Динамические изменения значений параметров, выполненные командой ALTER SYSTEM …, Oracle автоматически запоминает в *SPFILE.ORA*.

Параметры в *INIT.ORA* и их значения приводятся в любом порядке (за мелкими исключениями) и в любом регистре (за исключением закавыченных строк). Всего официальных параметров несколько сотен (неофициальных — много сотен), но обязательных для явного указания всего три из них (с версии 9):

DB_NAME
DB_BLOCK_SIZE
CONTROL_FILES (в момент выдачи команды CREATE DATABASE и этот не нужен)

Остальные либо имеют умолчательно подразумеваемые значения, либо выводят свои значения из прочих (в редких случаях еще из значений особых переменных ОС).

В типичном файле *INIT.ORA* присутствует несколько десятков параметров. С версии 10 они условно разбиты на две группы, по степени значимости: основные и вспомогательные.

Упражнение. Найти в файловой системе файл параметров СУБД.

2.3.4.2. Парольный файл СУБД

Используется для хранения списка имен и парольных сведений узкого круга пользователей с очень большими административными полномочиями (привилегиями SYSDBA, SYSOPER, SYSASM, …), разрешающими особо важные действия типа запуска СУБД и нескольких других.

Жаргонное название — *PWD.ORA*. Реальное имя *PWD%ORACLE_SID%.ora* (Windows) или *orapw$ORACLE_SID* (Unix). Заводится в *%ORACLE_HOME%\database* (Windows) или в *$ORACLE_HOME/dbs* (Unix) программой *orapwd* из состава ПО Oracle.

Упражнение. Найти в файловой системе парольный файл предустановленной СУБД. Обратиться к программе *orapwd* за подсказкой к употреблению.

2.3.5. Порождение сценария заведения БД

Этот этап выполняется всегда (явно или неявно), когда БД заводится в выборочном (Custom) режиме. При ручном заведении БД сценарные файлы требуется подготовить самостоятельно. Сформированная группа этих файлов предназначена для фактического создания БД. Обычно она состоит из сценариев:

- номинального создания БД SQL-предложением CREATE DATABASE (формально обязательно);
- заведения дополнительных объектов словаря-справочника (формально необязательно, но выполняется всегда);
- выполнения некоторых дополнительных действий по конфигурированию созданной БД.

2.3.5.1. Номинальное создание БД: предложение CREATE DATABASE

Выполнение SQL-предложения CREATE DATABASE является первым и формально единственным обязательным шагом создания любой БД в Oracle. Полный синтаксис приведен в документации. Он включает указание имени базы, кодировки, небольшого числа граничных переметров и основной конфигурации.

В результате выполнения предложения CREATE DATABASE заводятся файлы:

- Табличного пространства SYSTEM типа PERMANENT. Оно предназначено для хранения данных таблиц (и их индексов) словаря-справочника Oracle. АБД имеет право разрешить хранение в нем объектов обычных пользователей БД, однако врядли так поступит из соображений производительности и защищенности.
- Табличного пространства SYSAUX типа PERMANENT[10-). Предназначено для хранения административных данных, не относящихся к словарю-справочнику (например, данных OEM, Oracle Streams, Oracle Text, LogMiner и др.; перечень можно посмотреть в таблице SYS.V$SYSAUX_OCCUPANTS)
- Табличного пространства типа TEMPORARY (для сегментов временных данных).
- Табличного пространства типа UNDO (для сегментов отмены)[*9-).
- Журнальные (log files).
- Контрольный (control, иначе — управляющий).

[10-) С версии 10; обязательно.

[*9-) С версии 9; практически обязательно (необязательность поддерживается ради обратной совместимости).

Упражнение. В случае использования файловой системы ОС для файлов имеющейся БД, найти эти файлы.

Кроме того, результатом выполнения команды CREATE DATABASE является изначальное наличие в созданной БД следующих пользователей:

Имя	***Изначальный пароль***(-9.0]	***Пояснение***
SYS	CHANGE_ON_INSTALL	Владелец объектов словаря-справочника и «суперпользователь» системы.
SYSTEM	MANAGER	Администратор с суженным по отношению к SYS набором полномочий.

(-9.0] С версии 9.2 указать изначальный пароль на свое усмотрение разрешено непосредственно в тексте команды CREATE DATABASE.

2.3.5.2. Упрощения для версий 9+

Благодаря нескольким новым для версии 9 параметрам *минимальный* вид предложения CREATE DATABASE может выглядеть примерно так:

```
CREATE DATABASE mydb
DEFAULT TEMPORARY TABLESPACE tempts
UNDO TABLESPACE undots;
```

Для этого файл *INIT.ORA* обязан содержать особые параметры DB_CREATE_FILE_DEST и DB_CREATE_ONLINE_LOG_DEST_*n*, наподобие следующего:

```
db_name = mydb
db_block_size = 8192

# Эту строку добавить после CREATE DATABASE:
#control_files = (имена_контрольных_файлов_созданных_автоматически)

db_create_file_dest = c:\oracle\oradata\mydb
```

```
db_create_online_log_dest_1 = c:\oracle\oradata\mydb

undo_management = AUTO
undo_tablespace = UNDOTS
```

Тогда предложение CREATE DATABASE создаст в каталоге ОС *c:\oracle\oradata\mydb* авторастущие файлы для табличных пространств SYSTEM и TEMPTS по 100 Мб, для пространства UNDOTS (типа UNDO) размером 20 Мб, а так же журнальные файлы по 100 Мб и контрольный файл. Другие размеры возможны, но требуют явного задания. Файлы получат автоматические имена.

Такая техника, называемая Oracle managed files, OMF, была придумана для упрощения администрирования в случае большого количества файлов. Она не стала преобладающей, но нашла применение: например, в ASM.

2.3.5.3. Заведение дополнительных объектов словаря-справочника для БД

В узком понимании словарь-справочник есть множество таблиц пользователя SYS с описанием объектов, хранимых в БД. Oracle дополняет эти таблицы большим количеством представлений (view), пакетов, синонимов и прочего, без которых работа АБД и пользователей была бы малоудобна, а нередко невозможна. Создаваемые представления с некоторой вольностью можно также называть справочными таблицами.

Эти дополнительные объекты словаря-справочника создаются автоматическим прогоном большого числа сценариев из каталога ОС *%ORACLE_HOME%\rdbms\admin*. Чаще всего для минимального по объему варианта БД достаточно прогона только следующих сценариев:

%ORACLE_HOME%\rdbms\admin\catalog.sql (представления и синонимы словаря-справочника)
%ORACLE_HOME%\rdbms\admin\catproc.sql (встроенные пакеты)
%ORACLE_HOME%\rdbms\admin\catexp.sql

Эта фаза создания БД обычно самая долгая.

2.3.5.4. Создание дополнительной инфраструктуры в БД

В зависимости от указанных АБД характеристик при автоматическом (не ручном) заведении БД могут возникнуть некоторые другие пользователи помимо SYS и SYSTEM. Их назначение — создать ту или иную специализированную схему данных (о взаимоотношении «пользователь»/«схема» будет сказано далее):

Имя	***Первоначальный пароль***	***Пояснение***
SCOTT	TIGER	Схема («старого») демонстрационного примера
ORDSYS	ORDSYS	Учетная запись для администрирования interMedia
CTXSYS	CTXSYS	Схема для служебных объектов Oracle Text
DBSNMP	DBSNMP/указывается при создании БД[DBCA10-)	Учетная запись для наблюдения за работой СУБД и БД при использовании OEM
SYSMAN	указывается при создании БД[DBCA10-)	Схема для накопления сведений OEM
прочие	…	…

[DBCA10-) С версии 10 указать пароль требует программа DBCA.

Все указанные схемы (через учетные записи пользователей) создаются в результате прогона соответствующих сценариев в *$ORACLE_HOME/rdmbs/admin* и при желании могут быть удалены обычным образом. Другие сценарии из этого же каталога или других могут создавать дополнительные объекты в схеме SYS, такие как таблицы и прочее для Advanced Queuing, обслуживания аудита, программы Server Manager (устарела в версии 9) и других дополнений к базовой конфигурации БД.

На этом же этапе могут создаваться дополнительные табличные пространства (например, USERS), сегменты отката (в старых версиях) и прочее.

2.3.6. Указание свойств местности для БД и клиентских программ

Языковая поддержка является необходимым элементом определения как ПО СУБД, так и создаваемых БД. Историческое название средств языковой поддержки в Oracle — NLS (National Language Support). С версии 9 средства языковой поддержки рассматриваются в расширенном контексте, как средства указания свойств местности (locale), и называются Oracle Globalization, однако старое обозначение NLS продолжает жить во многих внутренних названиях.

2.3.6.1. Кодировка БД и приложения

БД в Oracle может создаваться:

1) с основной однобайтовой кодировкой хранимых текстов (US7ASCII, CL8MSWIN1251, …) плюс дополнительной Unicode (в форматах UTF8 или AL16UTF16),
2) с основной кодировкой Unicode (AL16UTF16).

(Многобайтовые кодировки не-Unicode возможны, но устарели. Oracle использует собственные обозначения для кодировок текстовых данных; их можно найти в документации или запросить у БД.)

Основная кодировка указывается фразой CHARACTER SET в команде CREATE DATABASE и распространяется на данные типов CHAR, VARCHAR2 и CLOB.

Дополнительная кодировка указывается фразой NATIONAL CHARACTER SET в команде CREATE DATABASE и распространяется на данные типов NCHAR, NVARCHAR2 и NCLOB.

Выбор Unicode в качестве основной кодировки БД делается нечасто: к примеру, при заведении БД для OID/LDAP или для нужд полнотекстового поиска средствами Oracle Text в разноязычных документах, и по другим поводам. Однако распространенность основной кодировки Unicode постоянно растет. Она используется в БД в Oracle Express Edition.

При выборе кодировки для создаваемой БД разумно ориентироваться на кодировку, в которой предположительно будет работать б*о*льшая часть клиентских программ, независимо от операционной среды сервера. Для России очень часто это CL8MSWIN1251.

2.3.6.2. Выбор языка сообщений, форматов выдачи и прочего

Помимо кодировки данных, второй главной составной частью свойств местности является язык/территория: NLS_LANGUAGE/NLS_TERRITORY, определяющие формат выдачи СУБД сообщений и ряд других свойств. Актуальные варианты: AMERICAN/AMERICA, RUSSIAN/CIS, RUSSIAN/RUSSIA. Последние две пары равноценны.

В целом, набор национальных параметров более широк:

Параметр	***Зависимость***
NLS_DATE_LANGUAGE NLS_SORT	Если не указать явно, выводится из NLS_LANGUAGE
NLS_CURRENCY NLS_DATE_FORMAT NLS_DUAL_CURRENCY NLS_ISO_CURRENCY NLS_NUMERIC_CHARACTERS NLS_TIMESTAMP_FORMAT NLS_TIMESTAMP_TZ_FORMAT	Если не указать явно, выводится из NLS_TERRITORY
NLS_CALENDAR NLS_COMP NLS_LENGTH_SEMANTICS	Если не указать явно, определяются номером версии СУБД

2.3.6.3. Где выполняются установки свойств местности, и где их можно наблюдать

Места простановки кодировки и других параметров местности:

- Сервер:
  - Предложение CREATE DATABASE (фразы CHARACTER SET и NATIONAL CHARACTER SET).
  - Словарь-справочник (таблица DATABASE_PROPERTIES).
  - Справка: таблица NLS_**DATABASE**_PARAMETERS (основанная на общем источнике данных с DATABASE_PROPERTIES); таблица NLS_**INSTANCE**_PARAMETERS.
- Клиентская программа:
  - Переменные среды окружения ОС (*NLS_LANG* и некоторые другие).
  - Реестр (в Windows; ключ *NLS_LANG*; *работать в реестре с осторожностью !*).
  - Справка: таблица NLS_**SESSION**_PARAMETERS.

В качестве клиентских программ выступают, в частности, SQL*Plus и RMAN.

Примеры установления параметров местности *до* запуска клиентской программы (здесь: на Windows; в Unix аналогично):

```
>set NLS_LANG=RUSSIAN_RUSSIA.RU8PC866
>set NLS_LANG=AMERICAN_AMERICA
>set NLS_LANG=.RU8PC866
```

(Если какая-то из трех компонент в сочетании *территория_язык.кодировка* отсутствует, берется значение по умолчанию.)

В Windows *NLS_LANG* как переменная командной оболочки (среды) имеет право отсутствовать, и в этом случае клиентская программа (конкретно Oracle Net) возьмет нужное значение из ключа *NLS_LANG* в реестре.

Другие, более специфичные, переменные ОС:

NLS_CURRENCY
NLS_DATE_FORMAT
NLS_DATE_LANGUAGE
NLS_DUAL_CURRENCY
NLS_ISO_CURRENCY
NLS_NUMERIC_CHARACTERS
NLS_TIMESTAMP_FORMAT
NLS_TIMESTAMP_TZ_FORMAT

Значительная часть параметров местности может быть изменена во время и в пределах работающего сеанса связи с СУБД — за исключением NLS_LANGUAGE и NLS_TERRITORY.

Примеры установок местности, выполненных на уровнях ОС и сеанса связи с СУБД:

```
>set NLS_LANG=RUSSIAN_.RU8PC866
>set NLS_DATE_FORMAT=Day hh24:mi:ss
>sqlplus scott/tiger
SQL> SELECT sysdate FROM dual;
SQL> ALTER SESSION SET NLS_DATE_FORMAT='Day hh12:mi:ss am';
SQL> SELECT sysdate FROM dual;
```

2.3.6.4. Замена и правка свойств существующих языковых установок БД и создание новых

Замены кодировки в БД лучше избегать, хотя в некоторых случаях, когда новая кодировка представляет из себя строгое подмножество прежней, формально она выполнима командой типа:

```
ALTER DATABASE orcl CHARACTER SET CL8MSWIN1251;
```

Даже если это удается, это никак не перекодирует текстовые данные, уже заведенные в БД. Перед сменой кодировки желательно устроить проверку возможности перекодирования имеющихся таких данных из старой кодировки в новую. Это делается специальной программой *csscan* (Character Set Scan). Правила ее использования можно получить, набрав в консольном окошке:

```
>csscan help=y
```

Собственно замена данных выполняется экспортом/импортом, или программой *csalter*:

```
>sqlplus / as sysdba @csalter.plb
```

Обе программы запускаются по определенной технологии, описанной в документации.

Кроме того имеется специальная программа для правки имеющихся в Oracle языковых настроек (кодировок БД и языка и формата выдачи) или создания новых (более предпочтительно). Обратиться к ней можно через Start-меню (Windows), или из командной строки:

```
>%ORACLE_HOME%\nls\lbuilder\lbuilder
```

2.4. Запуск и останов СУБД, БД и служб БД

2.4.1. Службы ОС в Windows

В ОС Windows запуск СУБД и открытие БД возможны только после запуска службы, которая обычно создается автоматически при заведении БД с помощью OUI или DBCA, но может быть при необходимости заведена и вручную.

Запустить или остановить СУБД + БД («database system») можно запуском или остановом соответствующей службы в *Control Panel* → *Services*. Служба имеет имя вида *OracleService*ORACLE_SID. Имя образа службы плюс СУБД — *oracle.exe*. Если служба установлена в режиме автозапуска, БД будет запускаться операционной системой при старте автоматически.

То же из командной строки:

```
>sc start OracleServiceимя_экземпляра_СУБД
```

Те же обращением к специальной программе ПО Oracle *oradim.exe*:

```
>oradim -startup -sid имя_экземпляра_СУБД
```

останов:

```
>oradim -shutdown -sid имя_экземпляра_СУБД
```

Вследствие ошибок в коде службы, она не всегда справляется с задачей запуска СУБД после собственного старта. Чаще всего в этом случае достаточно дополнительно вручную выдать в SQL*Plus от имени SYS команду STARTUP (см. ниже).

Диагноз проблеме можно поставить по объему кода *oracle.exe*. В версии 9 он примерно 20Мб, а с версии 10 он примерно 10Мб. Размер же СУБД исчисляется сотнями мегабайт. Проверить фактический размер *oracle.exe* можно командами ОС:

```
>tasklist | find "oracle.exe"
>tasklist /fi "imagename eq oracle.exe"
```

2.4.2. Автоматический запуск и останов в Linux

Автоматический запуск БД в Linux (для перезапуска после сбоя) в отсутствии установленного ПО Oracle Restart осуществляется прогоном файлов запуска элементов ПО Oracle с помощью демонов ОС.

2.4.3. Запуск и останов СУБД и БД вручную

В Unix запуск и останов СУБД делается не через службы ОС (за отсутствием таковых), а командой SQL*Plus. Пример выполнения команд приводятся ниже.

Останов БД и работы СУБД:

```
1) $sqlplus / AS SYSDBA
2) SQL> SHUTDOWN IMMEDIATE
3) SQL> EXIT
```

До версии 10 такой вход в SQL*Plus требует использования двойных кавычек, позже ставших необязательными:

```
$sqlplus "/ AS SYSDBA"
```

В отсутствии доверительного подключения пользователя SYS вместо '/' потребуется указать '*имя*/*пароль*' (пароль можно скрыть за диалогом).

Запуск СУБД и БД:

```
1) $sqlplus / AS SYSDBA
2) SQL> STARTUP
3) SQL> EXIT
```

В Windows выполнение STARTUP/SHUTDOWN не затрагивает активность службы ОС для СУБД (управляемой программой *oradim.exe*) и подразумевают непрерываемое запущенное состояние этой службы.

2.4.4. Запуск и останов СУБД и БД с помощью Oracle Restart

Установленное ПО Oracle Grid Infrastructure открывает возможность АБД задействовать средство поддержки высокой доступности Oracle Restart. Oracle Restart реализует «службы обеспечения высокой доступности», High Availability Services (HAS). АБД работает с этим инструментом через специальные программы *srvctl* и *crsctl*.

Oracle Restart позволяет запускать и останавливать не только СУБД, но и некоторые прочие компоненты общего ПО Oracle, например сетевое ПО Oracle Net Listener и ASM. Важно, что Oracle Restart:

- будет это делать в положенном порядке (например, запускать сначала экземпляр ASM, а потом уже СУБД), и
- будет автоматически перезапускать компоненты в случае сбоев.

Более того, при использовании Oracle Restart фирма Oracle настоятельно советует запускать и останавливать работу компонент ПО не обычным путем (программами *sqlplus*, *lsnrctl* и прочими), а программой *srvctl*. Примеры:

```
>srvctl stop database -d prima
>srvctl start database -d prima
```

В Windows эти действия приводят к останову и запуску СУБД, но не соответствующей службы ОС (в данном случае *OracleServicePRIMA*); служба Windows при этих действиях продолжает работать. Управление работой компонент ПО программой БД *srvctl* следует осуществлять, настроившись на ORACLE_HOME конкретной компоненты. Так, приведенные выше две последние команды следует выдавать, настроившись на ORACLE_HOME ПО СУБД, а следующие две команды следует выдавать, настроившись на ORACLE_HOME ПО Grid Infrastructure:

```
>srvctl stop listener -l listener
>srvctl start listener -l listener
```

Саму же службу Oracle Restart АБД останавливает и запускает, по мере необходимости, программой *crsctl*. Примеры останова и запуска:

```
>crsctl stop has
>crsctl start has
```

Здесь слово has обозначает сокращение HAS, раскрытое выше.

Обратите внимание, что ранее остановленные экземпляры СУБД, ASM, процессы listener запускаются этими действиями в необходимом порядке.

В Windows службе Oracle Restart соответствует служба ОС *OracleOHService*. Приведенные выше две команды остановят и запустят службу *OracleOHService*, однако это можно делать и средствами Windows:

```
>sc start OracleOHService
>sc stop OracleOHService
```

Перевод Restart в автоматический или ручной запуск осуществляется командами:

```
>crsctl enable has
>crsctl disable has
```

2.4.5. Запуск и останов служб БД

Служба БД с глобальным именем БД существует всегда и запускается и останавливается автоматически при запуске и останове БД. При желании иметь прочие службы для нужд конкретных приложений, их потребуется создать администратору отдельно.

Примеры создания и удаления *службы БД* с именем product и с *сетевым именем БД* productrsrv (в SQL*Plus):

```
CONNECT / AS SYSDBA
EXECUTE DBMS_SERVICE.CREATE_SERVICE ( 'product', 'productsrv' )
  ...
EXECUTE DBMS_SERVICE.DELETE_SERVICE ( 'product' )
```

Сетевое имя службы БД предназначено для использования клиентскими программами и может назначаться, в зависимости от желания, как совпадающим с именем БД, так и не совпадающим.

Примеры запуска (приведения в рабочее состояние, активации) и останова службы БД:

```
EXECUTE DBMS_SERVICE.START_SERVICE ( 'product' )
  ...
EXECUTE DBMS_SERVICE.STOP_SERVICE ( 'product' )
```

В целях автоматизации обращение к процедурам запуска нужных служб БД можно вставить в тело триггерной процедуры категории AFTER STARTUP ON DATABASE.

При использовании Oracle Restart службы создаются и управляются программой *srvctl*:

```
>srvctl add service -d prima -s product
>srvctl start service -d prima -s product
>sqlplus scott/tiger@localhost:1521/product
...
```

2.5. Действия по убиранию Oracle с компьютера

Складываются из убирания инфраструктуры и файлов ПО СУБД, а также инфраструктуры и файлов БД. На Windows можно дополнительно обратить внимание на чистку реестра. Убирание ПО Grid Infrastructure делается аналогично.

2.5.1. Убирание БД из компьютера

Может выполняться программой Database Configuration Assistant (DBCA) или же вручную.

Вручную БД можно убрать из компьютера (насовсем, или для последующего пересоздания), убрав файлы БД из соответствующих каталогов. Если в Windows БД убирается насовсем, следует дополнительно убрать службу запуска СУБД программой *oradim*.

2.5.2. Убирание программных компонент с помощью Oracle Universal Installer

Отдельные компоненты Oracle можно убрать с помощью установщика OUI. Простое убирание каталогов программной среды СУБД средствами ОС *не* вызовет удаления информации о компонентах из внутреннего справочника в OUI; при последующих установках OUI будет полагать, что компонента установлена, и не осуществит повторную установку (чтобы ее выполнить придется «убрать» компоненту с помощью OUI «повторно»).

2.5.3. «Чистое» убирание Oracle

Убирание отдельных компонент Oracle нужно делать с помощью OUI (убирание только файлов и каталогов может привести к нежелательным эффектам). Однако иногда желательно убрать ПО Oracle вместе с базами из ОС целиком; например, при необходимости перейти к версии с более низким номером. Это обеспечивает последовательность действий, приводимая ниже.

Следующие шаги обеспечивают убирание Oracle из Windows, позволяя осуществлять последующую «чистую» установку (ср. с разделом 2.1above) без переустановки ОС:

- Остановить запущенные службы Oracle через *Control Panel* → *Administrative Tools* → *Services.*
  Просмотр и останов из командной строки:
  ```
  >sc queryex type=service state=all | find /i "oracle"
  >sc stop имя_службы
  ```
- (*Рекомендуется*) Войти в программу OUI и убрать установленную ранее программную среду Oracle. Ценность этого шага в том, что OUI уберет с компьютера много «мелочей», о которых администратор может забыть.
- Войти в реестр:
  - Удалить раздел *HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE.*
  - В разделе *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services* удалить все разделы с именем Oracle в названии(OUI).
  - В разделе *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog\Application* удалить все разделы с именем Oracle в названии(OUI10+).
  - Выйти из реестра.

  Эта работа выполняется и из командной строки. Например, что касается служб:
  - Справка о службах Oracle:
    ```
    >reg query HKLM\System\CurrentControlSet\services | find /i "oracle"
    ```
  - Удаление по полученному списку (с подсказкой):
    ```
    >reg delete HKLM\System\CurrentControlSet\services\служба
    ```
- Войти в *Control Panel* → *System* → *Advanced* → *Environment Variables*:
  - Убрать из определения переменной PATH упоминания об Oracle (например, *C:\Oracle\Ora81\Bin* или *D:\Program Files\Oracle\jre\1.1.7\bin* в версии 8)(OUI).
- В файловой системе:
  - Убрать позиции Oracle из *\Winnt\Profiles\All Users\Start Menu\Programs* (— в Windows NT; в Windows 200x убрать напрямую из Start-меню)(OUI).
  - Убрать позиции Oracle из *\Program Files* (*Common Files\Odbc\Datasources* и *Oracle*)(OUI).
- Перезагрузить компьютер (строго говоря, не обязательно, но так проще).
- В файловой системе:
  - Удалить каталог *%ORACLE_BASE%* (он конкретизировался в *HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ORACLE_HOMES*).

(OUI) Если перед этим использовался OUI, это действие может оказаться ненужным.
(OUI10+) Если перед этим использовался OUI, в версиях 10+ это действие может оказаться ненужным.

Убирание Oracle (программного обеспечения плюс БД) из ОС Unix выполняется удалением, и отчасти корректировкой, соответствующих файлов. Работа может быть выполнена сценарием *deinstall* из каталога *$ORACLE_HOME/deinstall* в режиме диалога или пакетно.

Если помимо ПО СУБД и файлов БД на компьютере присутствовало другое ПО Oracle, приведенную схему удаления придется подкорректировать.

3. Использование SQL*Plus в администрировании

Программа SQL*Plus предназначена для общения пользователя с БД в диалоге посредством командной строки и для администрирования Oracle. Здесь она представляет интерес во втором качестве.

В отличие от специальных программных продуктов для администрирования (OEM), SQL*Plus значительно менее затратна по ресурсам ОС.

3.1. Вызов SQL*Plus

Варианты вызова SQL*Plus:

- Из командной строки ОС для работы в диалоге или в пакетном (ключ –s) режиме.
- Из графической оболочки (Windows)[-10.2].
- Из Oracle Enterprise Manager (SQL*Plus Worksheet); из браузера web (iSQL*Plus) [-10.2].

[-10.2] в версиях до 11.

Достоинством «консольного варианта» является его присутствие в поставках ПО (СУБД и клиентского) на всех платформах с одинаковой общей функциональностью.

3.2. Некоторые популярные установки параметров и режимов работы SQL*Plus

SQL*Plus имеет несколько десятков параметров и режимов работы. Режимы выставляются и узнаются командами SET и SHOW. Параметры и режимы описаны в документации по Oracle, но обычной работы все они не нужны. Ниже приводятся некоторые наиболее популярные:

- Режим LINESIZE (длина буфера строки при выдаче ответов на SELECT).
- COLUMN … FORMAT … (формат выдачи значений столбца на экран).
- Режим PAGESIZE (количество строк на странице при выдаче ответов на SELECT).
- SPOOL (протоколирование диалога на экране в файле ОС).
- Режим SERVEROUTPUT (для выдачи из DBMS_OUTPUT, DBMS_SHARED_POOL и некоторых других пакетов).
- Режим TIMING (автоматический замер времени исполнения всех запросов).

3.3. Наиболее популярные команды SQL*Plus

SQL*Plus имеет несколько десятков собственных команд. Они описаны в документации по Oracle, но обычной работы все они не нужны. Ниже приводятся некоторые наиболее популярные:

- CONNECT/DISCONNECT
  `CONNECT` *имя_пользователя*/*пароль*; `CONNECT / AS SYSDBA`
- EDIT
  Команды `SET editfile` *файл*, `DEFINE _EDITOR =` *редактор*
  Команда `EDIT` *файл*
  Сокращенный вариант: `ED`
  `EDIT %ORACLE_HOME%\rdbms\admin\scott` или
  `EDIT $ORACLE_HOME/rdbms/admin/scott`
- START
  `START %ORACLE_HOME%\rdbms\admin\scott` или
  `START $ORACLE_HOME/rdbms/admin/scott`
- SAVE
  `SAVE` *файл* [`REPLACE`]
- DESCRIBE
  Сокращенный вариант: DESC
  `DESC emp`

- EXECUTE
  Сокращенный вариант: EXEC
  `EXECUTE dbms_output.put_line ( 'user ' || USER || ' works' )`
  (полность эквивалентно `BEGIN` *то,_что_написано_после_EXECUTE*`; END;`)
- SHOW, примеры:
  `SHOW ERRORS`; сокращенный вариант: `SHOW ERR`
  `SHOW PARAMETERS`, `SHOW PARAMETER trace`
  `SHOW USER`
- TIMING, например:
  `TIMING START замер_времени`
  ... делаем что-нибудь ... `TIMING SHOW` ... делаем что-нибудь ...
  `TIMING STOP`
  можно и так: `TIMING START "замер времени"`
- HOST
  `HOST DEL` *мой_файл* (в Windows) или `HOST rm` *мой_файл*, а то и `!rm` *мой_файл* (в Unix)
- STARTUP/SHUTDOWN
- HELP

Условные сокращения в командах SQL*Plus:

- %ORACLE_HOME% или $ORACLE_HOME, ? (с версии 10 в Windows последнее в команде EDIT— только после выдачи SET SQLPLUSCOMPATIBILITY 9.2)
- @, %ORACLE_SID% или $ORACLE_SID
- @, @@ вместо START
- / (— команда SQL*Plus)
- / и \ (— разделители в полном имени файлов)

Обозначения параметров в коде для SQL*Plus:

- &1, &2, …
- *&имя*
- *&&имя*

Параметры обычно требуют подстановки значения в диалоге. В случае пакетных заданий воизбежание диалога значения для подстановки можно определить командой SQL*Plus DEFINE. Эта же команда позволяет задать значения нескольким встроенным «параметрам» работы, имеющимся в SQL*Plus. Например, определить на свое усмотрение редактор текста, вызываемый по команде EDIT, можно так:

```
SQL> DEFINE _EDITOR
DEFINE _EDITOR         = "Notepad" (CHAR)
SQL> DEFINE _EDITOR = "C:\Program Files\Windows NT\Accessories\wordpad.exe"
SQL> DEFINE _EDITOR
DEFINE _EDITOR         = "C:\Program Files\Windows NT\Accessories\wordpad.exe"
(CHAR)
```

3.4. Файлы *glogin.sql* и *login.sql*

Файл *glogin.sql* всегда автоматически запускается сразу после входа в SQL*Plus, а с версии 10 — и после каждой выдачи команды CONNECT. По умолчанию расположен в *%ORACLE_HOME%\sqlplus\admin*. Примеры полезных установок работы SQL*Plus в файле *glogin.sql*:

```
SET LINESIZE 100

SET EDITFILE /temp/sqlplusbuf.pls



SET PAGESIZE 50
```

← удобнее выдача результатов запросов SQL

← задает удобное расположение в ОС временного файла для редактирования буфера SQL*Plus для команд SQL

← удобнее выдача данных из

```
SET TRIMSPOOL ON
```

```
SET TRIMOUT ON

COLUMN "Enter time" FORMAT A15
SELECT
  USER "Enter name"
, TO_CHAR ( SYSDATE, 'HH24:MI:SS' )
"Enter time"
FROM DUAL;

SET SQLPROMPT "_user 'on' _date 'at'
 _connect_identifier >"
```

длинных таблиц
← не выдаются хвостовые пробелы при выдаче в файл через SPOOL
← не выдаются хвостовые пробелы при выдаче на экран
← на экран выдается имя вошедшего пользователя и время входа в SQL*Plus

← подсказка — с версии 10

Аналогично можно строить содержимое файла *login.sql*. SQL*Plus версий 11- будет пытаться запустить его из текущего каталога пользователя, вызывающего эту программу; таким образом этих файлов может иметься много в разных каталогах файловой системы. В версиях 12+ расположение файла задается переменной ОС ORACLE_PATH в Unix и SQLPATH в Windows. (В версии 12.2 *login.sql* не считается из-за внутренней ошибки.) Это сделано с целью усиления безопасности.

3.5. Использование SQL*Plus для форматированной выдачи

SQL*Plus обладает довольно развитыми средствами для подготовки и оформленной выдачи данных, что, учитывая его роль как средства администрирования, весьма полезно. Ниже приводится простой пример, иллюстрирующий использование средств SQL*Plus при подготовке форматированной выдачи:

```
SET SPACE    2
SET FEEDBACK OFF
SET LINESIZE 60
SET PAGESIZE 30
COLUMN empno  HEADING "Номер|сотрудника" FORMAT 9999
COLUMN ename  HEADING "Имя|сотрудника"   FORMAT a10
COLUMN sal    HEADING "Нынешний|оклад"   FORMAT $9999.99
COLUMN newsal HEADING "Новый|оклад"      FORMAT $9999.99
COLUMN comm   HEADING "Комиссионные"     FORMAT $9999.99
TTITLE LEFT  'Подставная фирма' -
       RIGHT 'Страница: '                FORMAT 99 SQL.PNO SKIP 2
BTITLE CENTER 'Закрытая информация'

SELECT empno, ename, sal, sal * 1.10 AS newsal, comm
FROM   scott.emp
;
```

К сожалению, при работе с Unix/Linux особенности консольного окна в этих ОС не всегда способствуют удобному отображению подобной выдачи.

3.6. Совместное использование команд SPOOL, SAVE и START

Совместное использование команд SPOOL, SAVE и START дает администратору богатые возможности для автоматизации своей работы. Ниже приводится пример последовательности действия в SQL*Plus, иллюстрирующей это. Подготовим файл *rebuild.sql*:

```
STORE SET /temp/sqlplussettings.sql REPLACE
SET VERIFY OFF  FEEDBACK OFF  PAGESIZE 0
SPOOL /temp/dorebuild.sql REPLACE
SELECT 'rem Rebuild all unusable indexes' FROM dual;
```

```
SELECT
  'ALTER INDEX ' || index_name || ' REBUILD;'
FROM
  user_indexes
WHERE
  status = 'UNUSABLE'
;
SPOOL OFF
@/temp/sqlplussettings.sql
@/temp/dorebuild
```

Запуск сценария *rebuild.sql* в конечном счете перестроит все индексы схемы, где он запускается, помеченные как «неисправные» (UNUSABLE), например, в результате пересоздания или дефрагментации сегмента данных таблицы. Файл *dorebuild.sql* в корневом каталоге *temp* сохранит последовательность выполнявшихся операций. Если последующий запуск сценария *rebuild.sql* приведет к пустой последовательности операций, перевод всех индексов в исправное состояние прошел успешно.

Команда STORE (ранее не упоминавшаяся) сохраняет текущие установки SQL*Plus на случай, если сценарий вызывается не из командной строки ОС (`>sqlplus @rebuild`), а как эпизод работы в этой программе, после которого желательно продолжать сеанс связи с СУБД.

При желании, работу подобным образом составленных сценариев можно параметризовывать.

4. Отслеживание работы Oracle

Старайся наблюдать различные приметы:
Пастух и земледел в младенческие леты,
Взглянув на небеса, на западную тень,
Умеют уж предречь и ветр, и ясный день ...
А. С. Пушкин, Приметы

Одна из основных обязанностей АБД состоит в отслеживании процессов, которые происходят с БД и в СУБД при работе приложений с данными, и в выявлении негативных из них, с целью предупреждения или устранения нежелательных последствий.

Несколько примеров вопросов, которые могут интересовать администратора при отслеживании работы Oracle:

- наличие, состояние и текущие характеристики процессов СУБД;
- размеры и степень загруженности основных частей SGA;
- характеристики внутренних ожиданий доступа процессов к структурам общего пользования в SGA;
- загруженность работой журнальных файлов;
- загруженность работой сегментов отмены, табличных пространств временных данных;
- загруженность работой табличных пространств с данными;
- прочие.

К средствам диагностики, позволяющим следить за работой Oracle, можно отнести следущие:

- исходные: таблицы словаря-справочника;
- упрощенные: готовые (PL/)SQL-сценарии;
- файлы, порождаемые компонентами ПО Oracle;
- специальные процедуры проверки «монитор здоровья»;
- специализированные системы слежения общего характера.

4.1. Использование таблиц и представлений данных словаря-справочника

Список таблиц словаря-справочника с пояснениями имеется в документации (книжка Reference). Его же можно наблюдать в таблице DICTIONARY схемы SYS (публичные синонимы — DICTIONARY и DICT). Пример поиска таблиц с шаблоном JOB в названии:

```
COLUMN comments FORMAT A40 WORD
SELECT * FROM dictionary WHERE table_name LIKE '%JOB%'
;
```

Родственная по духу таблица — DICT_COLUMNS. Пример, как с ее помощью найти таблицы словаря-справочника со столбцами под названием USERNAME:

```
SELECT table_name, comments FROM dict_columns WHERE column_name = 'USERNAME'
;
```

Все таблицы делятся на две категории:

- «статические»: хранятся в БД на общих основаниях для всех таблиц БД,
- «динамические»: по сути — табличный взгляд на состояние внутренних переменных СУБД как программы.

4.1.1. Статические таблицы

В основе статических «таблиц» словаря-справочника Oracle лежит относительно небольшое число *собственно* таблиц, например OBJ$, TAB$, COM$, IND$, AUD$, PROPS$ и пр. На их основе построено несколько сотен *виртуальных* таблиц («представлений»), доставляющих нужную администратору информацию намного удобнее, и, собственно, предназначенных разработчиками Oracle для потребителей (посредством одноименных публичных синонимов), например таблицы:

USER(ALL, DBA)_TABLES	Общие сведения о таблицах
USER(ALL, DBA)_TAB_COLUMNS	Общие сведения о столбцах в таблицах
USER(ALL, DBA)_INDEXES	Общие сведения об индексах
USER(ALL, DBA)_CONSTRAINTS	Общие сведения об ограничениях целостности
USER(DBA)_TABLESPACES	Общие сведения о табличных пространствах в БД
прочие.	

Префикс USER обозначает перечисление объектов пользователя; префикс ALL — объектов, доступных для работы пользователю, но возможно из чужой схемы; префикс DBA — всех объектов БД.

Примеры:

```
SELECT owner, COUNT ( table_name ) FROM all_tables GROUP BY owner;

SELECT tablespace_name, block_size, contents FROM dba_tablespaces;

COLUMN comp_name FORMAT A35
SELECT comp_id, comp_name, version, status FROM dba_registry;
```

Другие примеры см. по тексту.

4.1.2. Динамические таблицы

Ниже приводится перечень некоторых таблиц словаря-справочника, динамически заполняемых значениями при работе СУБД и позволяющих следить за работой системы. (Большинство этих «таблиц» являются воображаемыми, view, а не действительными; однако для выборки данных это не имеет значения).

Группа динамических таблиц, предназначенных для пользователей Oracle, носит префикс V$ (так называемые «V$-таблицы»). Как правило, все они построены на основе уже действительных «X$-таблиц», которые, в свою очередь, являются по сути функциями, возвращающими в табличной форме значения данных из структур памяти в SGA в СУБД (но в небольшом количестве случаев — из контрольного файла). Официальное их название — «представления производительности [СУБД]» (performance views).

Примеры запросов:

```
SELECT * FROM v$option;

SELECT * FROM v$version;

COLUMN name        FORMAT a35
COLUMN description FORMAT a45
COLUMN value       FORMAT a10
SELECT
      name
    , value
    , isdefault
    , isses_modifiable
    , issys_modifiable
    , description
FROM  v$parameter
```

```
WHERE name LIKE '%trace%'
;
```

Права на выборку из V$-таблиц обычным пользователям (не SYS) передаются необычно. Фактически V$-объекты являются публичными синонимами для V_$-таблиц, принадлежащим пользователю SYS. Права следует выдавать на обращения к V_$-таблицам, но обращаться придется к их V$-синонимам:

```
CONNECT / AS SYSDBA
GRANT SELECT ON v_$mystat TO scott;
CONNECT scott/tiger
SELECT COUNT ( * ) FROM v$mystat;
```

4.1.2.1. Таблицы статистических показателей работы СУБД

Серия таблиц со значениями т. н. «статистик», различных показателей степени загруженности СУБД. Часть таких показателей носит накопительный характер (то есть они могут только прибавлять свои значения), а часть — текущий (могут и уменьшаться). Все они по форме являются счетчиками, отмечающими возникновение определенных внутренних событий.

Основные таблицы показателей работы СУБД:

- V$SYSSTAT: содержит общую информацию о значении показателей для СУБД.
- V$SESSTAT: содержит информацию о значениях показателей индивидуально для *каждого* из существующих сеансов. Когда сеанс завершится, его данные будут приплюсованы к V$SYSSTAT, и из V$SESSTAT удалены.
- V$MYSTAT: содержит значения показателей для *текущего* сеанса.
- V$SERVICE_STATS[10-): показатели *основных* статистик для зарегистрированных в рамках СУБД служб БД.

[10-) — начиная с версии 10.2

(Статистика по времени ожидания собирается, если параметр СУБД TIMED_STATISTICS = TRUE).

Для общей диагностики СУБД употребляется таблица V$SYSSTAT. Пример «прикидочного» запроса:

```
COLUMN name         FORMAT a35
COLUMN value        FORMAT 9999999999
SELECT * FROM v$sysstat WHERE ROWNUM <= 20;
```

Столбец V$SYSSTAT.CLASS характеризует принадлежность показателя одной из групп: *1* (пользователь); *2* (журнализация); *4* (функционирование замков с очередями); *8* (буфера); *16* (ОС), *32* (параллельный сервер RAC); *64* (SQL); *128* (отладка). Эти коды групп образуют маску, так что, например, класс 18 = (2 + 16) означает принадлежность статистики одновременно и процессам журнализации, и взаимодействия с ОС.

Перечень всех видов статистик, доступных в используемой версии Oracle, для каждой из групп можно получить из таблицы V$STATNAME.

Вот пример нескольких показателей, характеризующих использование журнального буфера:

Показатель	***Описание***
redo log space requests	количество ожиданий серверным процессом места в журнальном буфере, которое освободится, когда процесс LGWR сбросит предыдущие записи в журнальный файл
redo entries	количество занесений серверным процессом записи в журнальный буфер
redo buffer allocation retries	число повторных попыток (фактически — ожиданий) при занесении записей в журнальный файл

Значение redo log space requests/redo entries > 1/5000 может служить основанием для увеличения LOG_BUFFER или свидетельствовать о медленном переключении на очередной журнальный файл. На это же может указывать более чем линейный рост со временем redo buffer allocation retries (в идеале этот показатель должен быть 0).

Упражнение. Следующий пример показывает, как таблица V$MYSTAT позволяет замерять количество байтов журнальной информации, порождаемой отдельными операциями с данными. Выдать в схеме SCOTT:

```
VARIABLE absolute NUMBER
VARIABLE delta    NUMBER

BEGIN :absolute:= 0; END;
/

BEGIN
   SELECT m.value, m.value - :absolute INTO :absolute, :delta
   FROM   v$mystat m, v$statname n
   WHERE  m.statistic# = n.statistic# AND n.name = 'redo size';
END;
/

SAVE delta REPLACE

HOST echo PRINT delta >> delta.sql

UPDATE emp SET sal = sal;

@delta

HOST del delta.sql  -- для Unix указать rm вместо del
```

4.1.2.2. Таблицы существующих сеансов связи с СУБД и процессов

- V$SESSION: таблица, выдающая сводную информацию об имеющихся в настоящее время сеансах работы с Oracle (имя пользователя, имя инициировавшей сеанс клиентской программы и машины, время начала сеанса и т. д.).
- V$PROCESS: таблица процессов СУБД, и фоновых, и серверных (по обслуживанию запросов пользователей).

Пример запроса о существующих сеансах связи с СУБД:

```
COLUMN username FORMAT a10
COLUMN machine  FORMAT a20
COLUMN program  FORMAT a20
SELECT sid, serial#, username, osuser, machine, program, state
FROM   v$session
;
```

Еще пример:

```
COLUMN pu    FORMAT A8     HEADING 'OS|Login|ID'
COLUMN su    FORMAT A8     HEADING 'Oracle|User ID'
COLUMN stat  FORMAT A8     HEADING 'Session|Status'
COLUMN ssid  FORMAT 999999 HEADING 'Oracle|Session|ID'
COLUMN sser  FORMAT 999999 HEADING 'Oracle|Serial|No'
COLUMN spid  FORMAT 999999 HEADING 'OS|Process|/Thread'
COLUMN txt   FORMAT A28    HEADING 'Current Statment'

SELECT
  p.username         pu
, s.username         su
, s.status           stat
, s.sid              ssid
, s.serial#          sser
, LPAD ( p.spid, 7 ) spid
, SUBSTR ( sa.sql_text, 1, 540 ) txt
FROM
```

```
  v$process p INNER JOIN v$session s   ON ( p.addr = s.paddr )
              LEFT OUTER JOIN v$sql sa USING ( sql_id )
WHERE
  s.username IS NOT NULL
ORDER BY 1, 2, 7
;
```

В версии 9 последний запрос приходилось формулировать сложнее:

```
SELECT
  p.username         pu
, s.username         su
, s.status           stat
, s.sid              ssid
, s.serial#          sser
, LPAD ( p.spid, 7 ) spid
, SUBSTR ( sa.sql_text, 1, 540 ) txt
FROM
  v$process p INNER JOIN v$session s ON ( p.addr = s.paddr )
  LEFT OUTER JOIN v$sql sa ON ( s.sql_address = sa.address
                           AND  s.sql_hash_value = sa.hash_value )
WHERE
      s.username IS NOT NULL
ORDER BY 1, 2, 7
;
```

4.1.2.3. Таблицы для выявления внутренних задержек в работе СУБД

Доступ процессов СУБД к ресурсам в SGA не обязательно предоставляется сразу по требованию, и подвержен регулированию. В настоящее время используется три механизма такого регулирования (защелки, очереди ожидания и мьютексы), но в любом случае процессы время от времени испытывают задержки доступа. Группа динамических таблиц дает возможность наблюдать внутренние задержки СУБД, выявлять наиболее серьезные из них и перенастроить систему с целью повышения скорости работы.

Событие ожидания (wait event) представляет из себя особый показатель статистики, прирастающий на 1 всякий раз, когда серверный процесс не может сразу приступить к работе с общественным ресурсом и вынужден (так или иначе) ждать. Вот неполный перечень таблиц, дающих сведения о возникающих в СУБД событиях ожидания:

- V$EVENT_NAME: список имен событий ожиданиями (в версии 10 дополнены столбцами 'WAIT_CLASS%' для указания принадлежности события *классу* ради удобства анализа).
- V$SYSTEM_EVENT: накапливает обобщенные данные ожиданий в СУБД в целом.
- V$SESSION_EVENT: накапливает данные ожиданий по отдельным существующим сеансам.
- V$SERVICE_EVENT: данные ожиданий по отдельным существующим службам БД.
- V$SESSION_WAIT: содержит данные об ожиданиях сеансов в данный момент времени (в версии 10 столбцы этой таблицы ради удобства запросов включены также в V$SESSION).
- V$SYSTEM_WAIT_CLASS[10-)
- V$SESSION_WAIT_CLASS[10-)
- V$SERVICE_WAIT_CLASS[10-)
- V$EVENT_HISTOGRAM[10-)
- V$SESSION_WAIT_HISTORY[10-)
- V$FILE_HISTOGRAM[10-)
- V$TEMP_HISTOGRAM[10-)
- другие.

[10-) — начиная с версии 10.

Информацию можно видеть представленною с разных точек зрения: СУБД (SYSTEM), сеансов пользователей (SESSION), имеющихся в БД служб (SERVICE), а с версии 10 — показанную через гистограммы распределения.

Пример общего запроса о внутренних ожиданиях в СУБД:

```
SELECT
     event, total_waits, total_timeouts, time_waited, average_wait
FROM v$system_event
;
```

Некоторые примеры классов ожидания (с версии 10): *application* — ожидания прикладных программ, вызванные блокировками строк; *administration* — ожидания, вызванные действиями АБД, например при перестройке индекса; *commit* — ожидания подтверждения о занесении журнальной записи; *concurrency* — ожидания вследствие состязания за защелки и замки при одновременных попытках разбора запросов или обращения к буферизованному блоку; *configuration* — ожидания из-за недостаточного размера журнального буфера, журнальных файлов, буфера блоков, shared pool и др.; *user i/o* — ожидания при чтении блоков с диска; *network communications* — ожидания посылки данных по сети; другие (полный список — в документации).

Информация о внутренних ожиданиях, возникающих при обращении СУБД к *объектам*:

- V$WAITSTAT: наличие задержек при обращении к разных категориям блоков данных

4.1.2.4. Таблица открытых транзакций

Таблица V$TRANSACTION содержит список открытых в настоящий момент транзакций. Примеры ее столбцов:

- SES_ADDR: адрес памяти сеанса; по нему можно устраивать соединение этой таблица со столбцом SADDR таблицы V$SESSION
- START_TIME: время начала транзакции
- USED_UBLK: число используемых в настоящий момент блоков в сегменте отката

Пример запроса сведений об открытых транзакциях:

```
SELECT
   s.username, s.sid, t.xidusn, t.ubafil, t.ubablk, t.used_ublk
FROM
   v$session s INNER JOIN v$transaction t
   ON ( s.saddr = t.ses_addr )
;
```

4.1.2.5. Другие полезные динамические таблицы

Приведены в документации по Oracle. Некоторые примеры использования встречаются далее.

4.1.2.6. Упражнения

Выдать список пользователей, выполняющих незавершенные транзакции, времена начала транзакций и текущее число используемых блоков в сегментах отката. Открыть новый сеанс от имени SCOTT, выполнить UPDATE emp SET sal = sal и проследить за изменениями в списке.

Выдать список сеансов и текущих ожиданий сеансов.

Определить величину внутренних простоев на заголовке сегментов отката (undo header) и в теле (undo block).

4.1.2.7. Таблицы затрат времени СУБД (версии 10+)

В версии 10 появилось понятие *модели времени* (time model) как набора статистических показателей, описывающих распределение времени СУБД между разными видами деятельности. Данные берутся из основных таблиц:

- V$SYS_TIME_MODEL: распределение времени по разным видам работ в СУБД целиком.
- V$SESS_TIME_MODEL: распределение времени по разным видам работ в пределах отдельных сеансов.

В этих таблицах показатели:

- **'DB time'** = время *не-фоновых* процессов, расходуемые СУБД на активную обработку запросов: **'DB CPU time' + 'DB wait time'** (где 'DB wait time' = время ввода/вывода + время внутренних ожиданий помимо ожиданий Idle)
- **'background elapsed time'** = **'background cpu time' + время внутренних ожиданий** для *фоновых* процессов СУБД
- **'parse time elapsed'** = время разбора запросов SQL, и hard parse (**'hard parse time elapsed'**), и soft parse
- и т. д.

'DB time' в V$SYS_TIME_MODEL, как и большинство других показателей, получается суммированием значений для сеансов из V$SESS_TIME_MODEL, а значит способно давать величину больше, чем время работы СУБД.

Пример. Общее время работы каждого процесса СУБД и расходованное процессами время процессора:

```
COLUMN program FORMAT A30
SELECT   sid
       , program
       , ( SELECT value
           FROM   v$sess_time_model st
           WHERE  stat_name = 'DB time' AND st.sid = s.sid
         ) "DB time"
       , ( SELECT value
           FROM   v$sess_time_model sc
           WHERE  stat_name = 'DB CPU' AND sc.sid = s.sid
         ) "DB CPU"
FROM     v$session s
;
```

Показатель 'DB time' — просто количественная оценка, смысл которой может быть неоднозначен. Большая величина может свидетельствовать как (1) о большой активности пользователей (много подключений, интенсивная работа), так и (2) о конкуренции за процессорное время, (3) о чрезмерных внутренних ожиданиях или (4) об избыточном вводе/выводе. Его ценность в том, что он позволяет администратору конкретно формулировать задачи настройки СУБД в конкретных ситуациях.

Типично целью настройки может выступать снижение 'DB time'.

'DB CPU time' следует сопоставлять с 'DB wait time'. Если 'DB CPU time' >> 'DB wait time', возможно требуется настройка запросов, если 'DB CPU time' << 'DB wait time', возможно требуется настройка СУБД. Если таких перекосов нет, система считается удачно масштабируемой.

Остальные показатели сообщают о расходовании времени различными внутренними действиями. Например, 'sequence load elapsed time' — общее время извлечения очередных значений из датчиков новых чисел (sequences), и так далее.

4.1.2.8. Вторичные показатели работы СУБД (версии 10+)

Методически более практично обращать внимание не на абсолютные значения статистик СУБД, а на скорость их изменения. С версии 10 это позволяют делать *метрики*. В составе процессов СУБД имеется процесс MMON (manageability monitor), опрашивающий некоторые основополагающие динамические

таблицы с частотой раз в минуту или 15 секунд, и изменения заносит в таблицы «метрик». Данными метрик можно пользоваться самостоятельно, но их же используют и программы анализа и автонастройки самой системы (ADDM, советники). Для возможности более точного анализа метрики дополняются *гистограммами распределения* ряда показателей статистик.

4.1.2.8.1. Метрики

Метрики существуют для СУБД, сеансов, файлов и событий ожидания. Таблицы для текущих значений метрик по длинному (60 секунд) и короткому (15 секунд) интервалам:

- V$METRICNAME
- V$SYSMETRIC
- V$SESSMETRIC
- V$SERVICEMETRIC
- V$EVENTMETRIC, V$WAITCLASSMETRIC
- V$FILEMETRIC

Часовая история метрик (60 минут) сохраняется в оперативной памяти и наблюдаема в таблицах:

- V$SYSMETRIC_HISTORY
- V$SYSMETRIC_SUMMARY (для метрик по длинному интервалу)
- V$SERVICEMETRIC_HISTORY
- V$WAITCLASSMETRIC_HISTORY
- V$SESSION_WAIT_HISTORY (последние 10 событий ожидания для каждого сеанса)
- V$FILEMETRIC_HISTORY

Более длительная история сбрасывается в статические таблицы на диск:

- DBA_HIST_SYSMETRIC_SUMMARY
- DBA_HIST_SYSMETRIC_HISTORY (только извещения, alerts)
- DBA_HIST_SESSMETRIC_HISTORY
- DBA_HIST_SYSTEM_EVENT (накопительные данные)
- DBA_HIST_WAITCLASSMETRIC_HISTORY (только извещения, alerts)
- DBA_HIST_FILEMETRIC_HISTORY (только извещения, alerts)
- DBA_HIST_FILESTATXS (накопительные данные)

4.1.2.8.2. Гистограммы

Дают распределение некоторых показателей статистики по интервалам < 1 ms, < 2 ms, <4 ms, <6 ms, … < 2**22 ms (— примерно 70 минут), >= 2**22 ms:

- V$EVENT_HISTOGRAM
- V$FILE_HISTOGRAM
- V$TEMP_HISTOGRAM

Например, если V$FILEMETRIC или другое средство сравнительного анализа покажет задержки обращения к файлу № 1 (табличное пространство SYSTEM), то подробности распределения величин задержек можно уточнить в V$FILE_HISTOGRAM.

4.2. Сценарии на SQL и PL/SQL, поставляемые Oracle

4.2.1. AWR (версии 10+)

«Автоматический репозитарий данных о загруженности СУБД разными видами работ» (automatic workload repository, AWR). Является развитием более раннего средства Statspack (версии 8.1.6 — 10), а тот развитием пары сценариев *utlbstat.sql* и *utlestat.sql* в *%ORACLE_HOME%\rdbms\admin* для снятия показателей загрузки между двумя моментами прогона и выдачи сравнительного отчета.

AWR представляет собой инфраструктуру в БД для сбора и обработки статистики загрузки СУБД работой, плюс средства СУБД для автоматического пополнения и чистки соответствующих данных. Предназначение AWR — служить основой для встроенных средств автонастройки системы и для анализа ее предшествующей работы администратором.

Можно использовать для выявления изменений производительности СУБД за периоды времени.

В состав инфраструктуры входят пакеты (в первую очередь DBMS_WORKLOAD_REPOSITORY), служебные таблицы и представления. Исходные таблицы принадлежат SYS, а их данные расположены в табличном пространстве SYSAUX (но считанное число — в SYSTEM). Таблицы с префиксом 'WRM_' содержат метаданные; 'WRH_' — исторические данные; 'WRI_' — данные советников (advisors). Таблицы с префиксом 'WRR_' существуют с версии 11 для данных захвата и воспроизведения нагрузки (workload replay).

Снятие показаний статистики осуществляется вызовом (владелец пакета — SYS):

```
SQL> EXECUTE DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT
```

При желании в качестве аргумента можно указать значение 'TYPICAL' или 'ALL', заявляющее обычный или расширенный объем сбора показателей.

Если эти же значения (= TYPICAL или = ALL, но не = BASIC) имеет параметр СУБД STATISTICS_LEVEL,
то снятие показаний статистики выполняется автоматически: по умолчанию ежечасно, и с длительностью хранения 7, либо 8 дней (в разных версиях). За автоматическое пополнение репозитария и за его чистку от устаревших данных отвечает специально введенный (версия 10) процесс СУБД MMON.

Подобно Statspack, AWR позволяет сформировать на основе интервала между двумя снимками, и запомнить в БД «линию отсчета» производительности СУБД, чтобы впоследствии уметь давать обоснованную оценку поведению системы («лучше»/ «хуже» «нормы»):

```
SQL> EXECUTE DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE -
  2  ( start_snap_id => 30, end_snap_id => 35         -
  3  , baseline_name => 'Baseline1' )
```

Сведения из репозитария AWR и о его установках можно получить в таблицах:

V$ACTIVE_SESSION_HISTORY
V$%METRIC%
DBA_HIST_%
DBA_HIST_WR_CONTROL
DBA_HIST_BASELINE

Например, список снимавшихся (не важно, вручную или автоматически) показаний:

```
SELECT snap_id, startup_time, snap_level FROM dba_hist_snapshot;
```

Установки накопления снимков:

```
SELECT * FROM dba_hist_wr_control;
```

Просмотр созданных ранее линий отсчета:

```
SELECT baseline_id, baseline_name, start_snap_id, end_snap_id
FROM   dba_hist_baseline
;
```

Хранение снимков с данными загруженности СУБД работой затратно для БД, и умолчательные установки могут потребовать вмешательства. График и свойства автоматического снятия показаний можно изменить процедурой MODIFY_SNAPSHOT_SETTINGS, например:

```
BEGIN
DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS (
  interval  => DBMS_WORKLOAD_REPOSITORY.MIN_INTERVAL * 3     -- 30 минут
, retention => DBMS_WORKLOAD_REPOSITORY.MIN_RETENTION * 4    -- 4 дня
, topnsql   => 100 -- в каждом снимке сведений не более чем о 100 «тяжелых» запросах SQL
);
END;
/
```

Сократить время хранения снимков не удастся, если новому значению будет препятствовать установленная длина «окошка для гибкого формирования линии отсчета» (moving window baseline). В нашем случае длину окошка достаточно предварительно сократить до 4-х дней (изначально она больше):

```
BEGIN
DBMS_WORKLOAD_REPOSITORY. MODIFY_BASELINE_WINDOW_SIZE (
  window_size => 4         -- 4 дня
);
END;
/
```

Простое удаление:

```
EXECUTE DBMS_WORKLOAD_REPOSITORY.DROP_SNAPSHOT_RANGE –
                                  ( low_snap_id => 1077, high_snap_id =>
1078 )
```

Получить сравнительный отчет по паре указанных имеющихся снимков AWR можно программно, но удобнее сценариями *awrrpt.sql* и *awrrpti.sql* (в форматах 'html', 'text', а с версии 12 и 'active-html'); например:

```
SQL> @?/rdbms/admin/awrrpt
```

4.2.2. ASH

С версии 10 Oracle отдельно собирает сведения об активности последних *сеансов* (не находящихся в состоянии ожидания класса Idle). Данные этого рода носят название active session history (ASH).

ASH можно использовать для выявления подробностей производительности и событий по сеансам, службам, классам ожидания, CLIENT_ID или SQL_ID.

Сведения об активности текущих и недавних (примерно последние 30 минут) сеансов, включая выдачу ими запросов SQL, ежесекундно добавляются во внутренний кольцевой буфер и доступны через V$ACTIVE_SESSION_HISTORY. Следующий запрос покажет идентификаторы наиболее активных запросов за последнюю минуту:

```
SELECT
  sql_id
, COUNT ( * )
, ROUND ( COUNT ( * ) / SUM ( COUNT ( * ) ) OVER ( ), 2 ) pctload
FROM     v$active_session_history
WHERE
         sample_time > SYSDATE - 1 / 24 / 60
     AND session_type <> 'BACKGROUND'
GROUP BY sql_id
ORDER BY COUNT ( * ) DESC
```

```
;
```

Эти данные переносятся процессом СУБД MMNL в таблицы AWR для более долговременного хранения и могут быть запрошены оттуда через таблицу DBA_HIST_ACTIVE_SESS_HISTORY.

Пример получения отчета ASH из SQL*Plus:

```
SQL> @?/rdbms/admin/ashrpt
```

4.2.3. Пакет DBMS_WORKLOAD_REPOSITORY

Пакет DBMS_WORKLOAD_REPOSITORY включает средства для построения разного рода отчетов о расходовании ресурсов СУБД, представляя их в разных форматах.

Предположим что БД имеет DBID = 2094368491 и 1 СУБД (не кластер). Следующие три запроса дадут ответ в формате HTML.

Отчет о наиболее активных сеансах за последний час:

```
SELECT *
FROM
  TABLE ( DBMS_WORKLOAD_REPOSITORY.ASH_REPORT_HTML (
            l_dbid     => 2094368491
          , l_inst_num => 1
          , l_btime    => SYSDATE - 1 / 24
          , l_etime    => SYSDATE
        ) )
/
```

Пусть 123 и 124 — номера двух снимков AWR. Отчет об общей загрузке СУБД (включая SQL) в промежутке между этими снимками:

```
SELECT *
FROM
  TABLE ( DBMS_WORKLOAD_REPOSITORY.AWR_REPORT_HTML (
            l_dbid     => 2094368491
          , l_inst_num => 1
          , l_ bid     => 123
          , l_ eid     => 124
        ) )
/
```

Пусть 5m6mu5pd9w028 — значение SQL_ID конкретого запроса. Отчет о загрузке СУБД вычислением этого запроса в промежутке между снимками:

```
SELECT *
FROM
  TABLE ( DBMS_WORKLOAD_REPOSITORY.AWR_SQL_REPORT_HTML (
            l_dbid     => 2094368491
          , l_inst_num => 1
          , l_ bid     => 123
          , l_ eid     => 124
          , l_sqlid    => '5m6mu5pd9w028'
        ) )
/
```

Ту же выдачу, но в формате простого текста, дадут функции ASH_REPORT_TEXT, AWR_REPORT_TEXT и AWR_SQL_REPORT_TEXT.

4.3. Активное отслеживание событий в системе (версии 10+)

Для существующих с версии 10 метрик (частота выдачи программами COMMIT; уровень заполненности табличного пространства, количество физических чтений за секунду и пр.) есть возможность установить более 100 «пороговых значений», *metric thresholds*. Процесс СУБД MMON (с участием MMNL), или же агент OEM, регулярно проверяет факты перехода через эти пороговые значения и помещает извещения о таких переходах, *alerts*, во внутреннюю очередь сообщений под названием ALERT_QUE (она принадлежит пользователю SYS и состоит из элементов типа SYS.ALERT_TYPE). Подписчиком очереди сообщений изначально являются AWR и консоль OEM (— показывает полученные соотщения на странице Home), но средствами пакетов DBMS_AQADM DBMS_AQELM и DBMS_AQ администратор в состоянии добавить в подписчики очереди любую свою программу.

Серверные тревожные сообщения порогового типа (threshold, или же stateful) «попадают» в таблицу DBA_OUTSTANDING_ALERTS, а после устранения причины переносятся в DBA_ALERT_HISTORY (страница Alert History в OEM).

Непороговые тревожные сообщения (nonthreshold, или же stateless: 'snapshot too old', 'recovery area low on free space' и пр.) попадают сразу в DBA_ALERT_HISTORY.

Пример. Породим пороговое тревожное сообщение TABLESPACE_PCT_FULL.

```
CREATE TABLESPACE tbs1 DATAFILE 'G:\TEMP\TBS1.DBF' SIZE 1M AUTOEXTEND OFF;
CREATE TABLE t1 ( a CHAR ( 2000 ) ) TABLESPACE tbs1;
DECLARE
PROCEDURE print_threshold IS
   l_warning_operator         BINARY_INTEGER;
   l_warning_value            VARCHAR2 ( 4000 );
   l_critical_operator        BINARY_INTEGER;
   l_critical_value           VARCHAR2 ( 4000 );
   l_observation_period       BINARY_INTEGER;
   l_consecutive_occurrences  BINARY_INTEGER;

BEGIN
DBMS_SERVER_ALERT.GET_THRESHOLD (
  metrics_id               => DBMS_SERVER_ALERT.TABLESPACE_PCT_FULL
, warning_operator         => l_warning_operator
, warning_value            => l_warning_value
, critical_operator        => l_critical_operator
, critical_value           => l_critical_value
, observation_period       => l_observation_period
, consecutive_occurrences  => l_consecutive_occurrences
, instance_name            => NULL
, object_type              => DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE
, object_name              => 'TBS1'
);

DBMS_OUTPUT.PUT_LINE ( l_warning_operator );
DBMS_OUTPUT.PUT_LINE ( l_warning_value );
DBMS_OUTPUT.PUT_LINE ( l_critical_operator );
DBMS_OUTPUT.PUT_LINE ( l_critical_value );
DBMS_OUTPUT.PUT_LINE ( l_observation_period );
DBMS_OUTPUT.PUT_LINE ( l_consecutive_occurrences );
-- иначе можно взять из таблицы DBA_THRESHOLDS

EXCEPTION WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE ( SQLERRM );
END;

BEGIN
print_threshold;      -- проверка до установки порогов

DBMS_SERVER_ALERT.SET_THRESHOLD (
  metrics_id               => DBMS_SERVER_ALERT.TABLESPACE_PCT_FULL
, warning_operator         => DBMS_SERVER_ALERT.OPERATOR_GE
```

```
, warning_value           => 80
, critical_operator       => DBMS_SERVER_ALERT.OPERATOR_GE
, critical_value          => 90
, observation_period      => 1
, consecutive_occurrences => 1
, instance_name           => NULL
, object_type             => DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE
, object_name             => 'TBS1'
);

print_threshold;      -- проверка после установки порогов
END;
/

INSERT INTO t1 VALUES ( 'a' );
BEGIN LOOP INSERT INTO t1 SELECT * FROM t1; END LOOP; END;
/
-- ошибка !
COMMIT;
```

Тревожное сообщение обнаружится с характерным запаздыванием:

```
COLUMN object_name      FORMAT A11
COLUMN object_type      FORMAT A11
COLUMN reason           FORMAT A30 WORD
COLUMN suggested_action FORMAT A30 WORD

SELECT object_name, object_type, message_level, reason, suggested_action
FROM   dba_outstanding_alerts
;
```

Ответ:

```
OBJECT_NAME OBJECT_TYPE REASON                         SUGGESTED_ACTION
----------- ----------- ------------------------------ ------------------------------
TBS1        TABLESPACE  Табличное пространство [TBS1]  Добавьте дисковое пространство
                        [100 percent] заполнено        к данному табличному
                                                       пространству
```

Далее:

```
SELECT COUNT ( * ) FROM dba_alert_history /* проверка */;
TRUNCATE TABLE t1;

COLUMN creation_time FORMAT A18

SELECT reason, creation_time, suggested_action, resolution
FROM   dba_alert_history
;
```

Ответ:

```
REASON                         CREATION_TIME      SUGGESTED_ACTION
RESOLUT
------------------------------ ------------------ ------------------------------
-------
Табличное пространство [TBS1]  18.09.15 18:04:02, Добавьте дисковое пространство
cleared
[12 percent] заполнено         414000 +04:00      к данному табличному
                                                  пространству
```

Еще пример для количества операций COMMIT в секунду:

```
BEGIN
DBMS_SERVER_ALERT.SET_THRESHOLD (
  metrics_id              => DBMS_SERVER_ALERT.USER_COMMITS_SEC
, warning_operator        => DBMS_SERVER_ALERT.OPERATOR_GE
```

```
, warning_value           => 10
, critical_operator       => DBMS_SERVER_ALERT.OPERATOR_GE
, critical_value          => 20
, observation_period      => 1
, consecutive_occurrences => 1
, instance_name           => NULL
, object_type             => DBMS_SERVER_ALERT.OBJECT_TYPE_SYSTEM
, object_name             => NULL
);

INSERT INTO t1 VALUES ( 'a' );

-- работа может занять около 5 минут
FOR i IN 1..300 LOOP
  FOR j IN 1..5 LOOP UPDATE t1 SET a = 'a'; COMMIT; END LOOP;
  DBMS_LOCK.SLEEP ( 1 ); -- спим 1 секунду
END LOOP;
END;
/
```

Результат из DBA_ALERT_HISTORY:

```
REASON                          CREATION_TIME      SUGGESTED_ACTION
RESOLUT
------------------------------- ------------------ ------------------------------
-------Обновлен порог метрики "User   18.09.15 19:39:13, Проверьте представление
cleared
Commits Per Sec" для            654000 +04:00      DBA_THRESHOLDS для проверки
экземпляра "database_wide"                         результата
```

Сброс порогов:

```
BEGIN
DBMS_SERVER_ALERT.SET_THRESHOLD (
  metrics_id              => DBMS_SERVER_ALERT.USER_COMMITS_SEC
, warning_operator        => NULL
, warning_value           => NULL
, critical_operator       => NULL
, critical_value          => NULL
, observation_period      => 1
, consecutive_occurrences => 1
, instance_name           => NULL
, object_type             => DBMS_SERVER_ALERT.OBJECT_TYPE_SYSTEM
, object_name             => NULL
);
END;
/
```

4.4. Диагностические файлы ПО Oracle

Диагностические файлы СУБД, процесса listener и прочих компонент ПО Oracle порождаются в текстовом формате, доступном для рассмотрения в простом текстовом редакторе.

В версии 11 схема хранения диагностических файлов ПО Oracle претерпела изменение и получила новое название ADR. Расположение этих файлов предполагается в каталоге *$ORACLE_BASE/diag*, полное название которого задается параметром СУБД DIAGNOSTIC_DEST. Если DIAGNOSTIC_DEST не установлена, опорным для диагностических файлов назначается каталог из переменной ОС ORACLE_BASE, а если и она не установлена, *$ORACLE_HOME/log*.

Отойти от этой техники размещения диагностических файлов и вернуться к старой позволяет скрытый параметр СУБД _DIAG_ADR_ENABLED со значением TRUE. Это будет сопровождаться заметным сокращением файлов диагностики, но поступать так на промышленной БД не рекомендуется. Последнее в полной мере относится также к значению TRUE другого скрытого параметра СУБД, _DISABLE_HEALTH_CHECK.

Общую информацию о местонахождении разного рода диагностических файлов можно получить запросом к БД:

```
SELECT * FROM v$diag_info;
```

Пример ответа:

```
   INST_ID NAME                 VALUE
---------- --------------------
------------------------------------------
         1 Diag Enabled         TRUE
         1 ADR Base             c:\app\oracle
         1 ADR Home             c:\app\oracle\diag\rdbms\prima\prima
         1 Diag Trace           c:\app\oracle\diag\rdbms\prima\prima\trace
         1 Diag Alert           c:\app\oracle\diag\rdbms\prima\prima\alert
         1 Diag Incident
c:\app\oracle\diag\rdbms\prima\prima\incident
         1 Diag Cdump           c:\app\oracle\diag\rdbms\prima\prima\cdump
         1 Health Monitor       c:\app\oracle\diag\rdbms\prima\prima\hm
         1 Default Trace File   c:\app\oracle\diag\rdbms\prima\prima\trace\
                                prima_s000_1840.trc
         1 Active Problem Count 0
         1 Active Incident Count 0
```

Содержимое файлов можно смотреть самостоятельно, однако для удобства извлечения из них нужных сведений в ПО Oracle 11 введена специальная программа *adrci* (ADR command interpreter), работающая по принципу командной строки. Она может запускаться как в пакетном режиме, так и в диалоге, например:

```
>adrci exec = "show problem ; show incident"

>adrci
...
adrci> SHOW PROBLEM
...
adrci> SHOW INCIDENT
...
```

4.5. Средства «монитора здоровья» БД (версии 11+)

Монитор здоровья (health monitor, HM; надзиратель здоровья) представляет из себя инфраструктуру, основанную на серии внутренних процедур выявления повреждений ряда составных частей БД. Помимо самих процедур проверки повреждений в инфраструктуру входят справочные таблицы с именами LIKE 'V$HM_%' и пакет DBMS_HM. Виды проверок способен вернуть следующий запрос (версия 11):

```
SQL> COLUMN description FORMAT A50
SQL> SELECT   name, internal_check, offline_capable, description
  2> FROM     v$hm_check
  3> ORDER BY 2;

NAME                          I O DESCRIPTION
----------------------------- - -
--------------------------------------------------
DB Structure Integrity Check  N Y Checks integrity of all database files
CF Block Integrity Check      N Y Checks integrity of a control file block
Data Block Integrity Check    N Y Checks integrity of a data file block
Redo Integrity Check          N Y Checks integrity of redo log content
Transaction Integrity Check   N N Checks a transaction for corruptions
```

```
Undo Segment Integrity Check N N Checks integrity of an undo segment
Dictionary Integrity Check   N N Checks dictionary integrity
ASM Allocation Check         N Y Diagnose allocation failure
HM Test Check                Y Y Check for health monitor functionality
Logical Block Check          Y N Checks logical content of a block
No Mount CF Check            Y Y Checks control file in NOMOUNT mode
Mount CF Check               Y Y Checks control file in mount mode
CF Member Check              Y Y Checks a multiplexed copy of the control
file
All Datafiles Check          Y Y Checks all datafiles in the database
Single Datafile Check        Y Y Checks a data file
Tablespace Check Check       Y Y Checks a tablespace
Log Group Check              Y Y Checks all members of a log group
Log Group Member Check       Y Y Checks a particular member of a log group
Archived Log Check           Y Y Checks an archived log
Redo Revalidation Check      Y Y Checks redo log content
IO Revalidation Check        Y Y Checks file accessibility
Block IO Revalidation Check  Y Y Checks file accessibility
Txn Revalidation Check       Y N Revalidate corrupted transaction
Failure Simulation Check     Y Y Creates dummy failures
ASM Mount Check              Y Y Diagnose mount failure
ASM Disk Visibility Check    Y Y Diagnose add disk failure
ASM File Busy Check          Y Y Diagnose file drop failure
```

Проверки, помеченные как OFFLINE_CAPABLE = 'Y', не требуют для выполнения открытой БД.
Проверки, помеченные как INTERNAL_CHECK = 'Y', выполняются самой СУБД, остальные администратор имеет право устраивать по своему усмотрению, например:

```
BEGIN
DBMS_HM.RUN_CHECK (
  check_name => 'Dictionary Integrity Check'
, run_name   => 'MY_DICTIONARY_CHECK'
);
END;
/
```

У некоторых видов проверок существуют параметры. Допускается их указание при запуске задания на проверку:

```
BEGIN
DBMS_HM.RUN_CHECK (
  check_name   => 'Dictionary Integrity Check'
, run_name     => 'MY_DICTIONARY_CHECK'
, timeout      => 0
, input_params => 'TABLE_NAME=tab$'
);
END;
/
```

(TAB$ — одна из статических таблиц словаря-справочника, с данными о хранимых таблицах.)

Список обязательных и не обязательных параметров сведен в таблице V$HM_CHECK_PARAM.

Результат проверки в виде простого текста, XML или HTML позволяет функция GET_RUN_REPORT:

```
SET LONG 10000
SELECT DBMS_HM.GET_RUN_REPORT ( 'MY_DICTIONARY_CHECK' ) text_report
FROM   dual
;
```

То же получается обращением к V$HM_FINDING. Однако этот результат помещается в файл в ADR, откуда он читается текстовым редактором или же средствами программы *adrci*:

```
>adrci SHOW REPORT HM_RUN MY_DICTIONARY_CHECK
```

Обращаться к монитору здоровья и наблюдать результаты проверок позволяет также OEM.

4.6. Oracle Enterprise Manager

На рынке имеется небольшое количество специализированных программных продуктов, обеспечивающих слежение за работой Oracle. Одни создаются методом открытых текстов (open source) и представлены в интернете бесплатно, другие распространяются на коммерческой основе.

Из штатных продуктов Oracle для Windows можно отметить «счетчики» (counters) для Performance Monitor (версии 12-). Штатным средством администрирования Oracle для всех платформ, и средством отслеживания работы, является Oracle Enterprise Manager (OEM).

В версии 7 OEM поставлялся самостоятельно; в версии 8.1 в состав основного комплекта поставки СУБД вошел частично (в полном объеме поставлялся самостоятельно); в версии 9 включен в состав ПО СУБД полностью, в виде клиентского приложения на Java, взаимодействующего с программой под названием Management Server.

В версии 10 OEM был переписан как приложение для web на основе контейнера сервлетов OC4J (хотя клиентская Java-консоль в этой версии еще поставлялась). Вынесенный вариант, предполагающий администрирование с отдельной машины множества баз данных в сети, получил название Oracle Grid Control; установка Grid Control выполняется самостоятельно. Заменой ему служит заведение OEM на одной машине с БД исключительно для обслуживания этой БД. Этот вариант OEM носит название EM DB Console, и установкой EM DB Console обычно занимается DBCA (если в свойствах проектируемой БД об этом попросить), но сделать это можно и вручную.

В случае собственной OEM консоли для конкретной БД, адресация БД осуществляется номером порта в URL обращения к OEM:

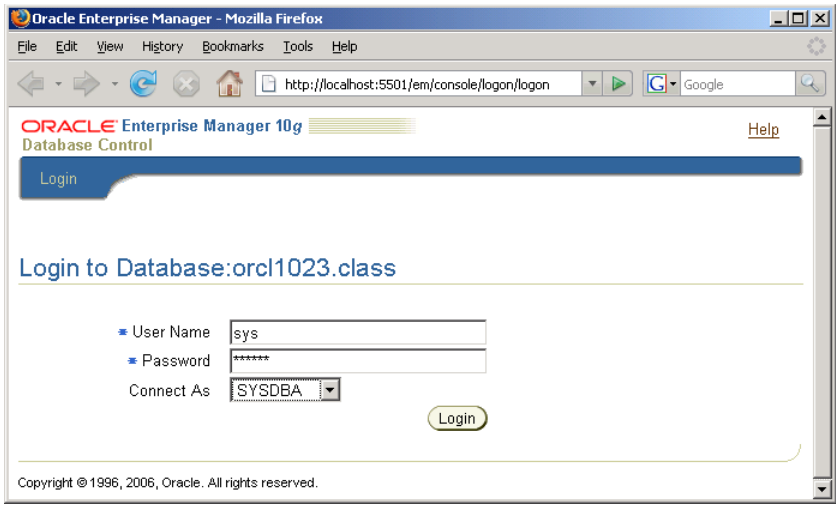
протокол://компьютер:порт/em

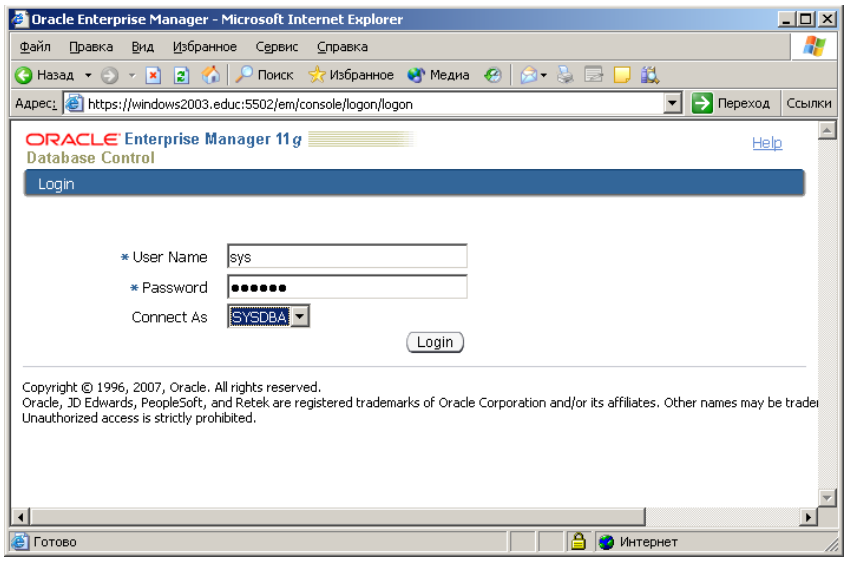
Здесь:

- *протокол* — HTTP или HTTPS.
- *компьютер* — имя компьютера в сети (возможен цифровой адрес IP, хотя из общих соображений это и не приветствуется).
- *порт* — номер порта, к которому привязана консоль OEM как приложение для web. Если не устанавливать специально, для первой заведенной на компьютере консоли OEM будет назначен порт 1158, для второй — 5500, для третьей 5501 и так далее (— для индивидуального обслуживания БД).
- *em* — название приложения для web (консоли OEM).

Номер порта и протокол переустанавливаются администратором по желанию.

Ниже приведены примеры страниц web для подключения к OEM в версиях 10 и 11 (простой вариант для индивидуального обслуживания БД). Доступ к этим страницам web выполнен по адресам URL ***http****://localhost:5501/em* и ***https****://windows2003.educ:5502/em*:





Упражнение. Ознакомиться в общем с работой OEM. Найти сведения о снятых AWR ранее снимках характеристик работы СУБД. Найти ссылки для выполнения проверок внутренних рассогласований в БД средствами «монитора здоровья» БД (checkers; для версий 11+).

Замечания.

1) С версии 10 OEM работать с ним в полном объеме можно только через браузер; прежней консоли OEM (на Java) оставлен круг функций DBA Studio из предыдущих версий (с версии 11 DBA Studio не поставляется). Запуск, останов, реконфигурирование и прочее выполняется программами *emctl* и *emca*. Примеры запуска, останова и просмотра состояния:

```
>emctl start dbconsole
>emctl stop dbconsole
>emctl status dbconsole
```

В Windows консоль OEM оформлена в виде службы *OracleDBConsole*<ORACLE_SID>, и запуск и останов OEM можно выполнять через запуск и останов службы. Заводится служба программой DBCA при создании БД, либо же вручную.

Примеры создания консоли OEM (и службы, если в Windows) — «простой» и «сложный»:

```
>emca -config dbcontrol db

>emca -config dbcontrol db -silent -DB_UNIQUE_NAME prima -PORT 1521 -EM_HOME
C:\oracle\product\10.2.0\db_1 -LISTENER LISTENER -SERVICE_NAME prima.class -SYS_PWD
```

```
пароль_SYS -SID prima -ORACLE_HOME C:\oracle\product\10.2.0\db_1 -DBSNMP_PWD
пароль_DBSNMP -HOST компьютер -LISTENER_OH C:\oracle\product\10.2.0\db_1 -LOG_FILE
C:\oracle\product\10.2.0\admin\prima\scripts\emConfig.log -SYSMAN_PWD пароль_SYSMAN
```

Пример внесения изменения в конфигурацию OEM:

```
>emca -reconfig ports -DBCONTROL_HTTP_PORT 1820 -AGENT_PORT 1821 -RMI_PORT 5520
```

Справки:

```
>emca --help
>set ORACLE_SID=prima
>emctl --help
```

2) Работать с OEM может пользователь, обладающий привилегией SELECT ANY DICTIONARY. Например, это пользователи Oracle SYSMAN (это имя используется также для собственного пользователя Grid/Cloud Control), SYS и SYSTEM, но врядли пользователь SCOTT.

3) Для работы «заданий» (jobs) и «событий» (events) в OEM, а также некоторых задач, запускаемых в OEM на основе аппарата «заданий» (например, экспорта/импорта или резервного копирования и восстановления данных) пользователю Windows, от имени которого ведется работа, требуется обеспечить привилегию ОС Log on as a batch job (*Control Panel* → *Administrative Tools* → *Local Security Policy*).

В версии 12 OEM был переписан под контейнер сервлетов (сервер приложений) WebLogic. С этого времени полноценный OEM существует только в вынесенном варианте (обслуживание с отдельной машины множества БД в сети), и под названием Cloud Control. В урезанном же объеме (примерно соответствующем старой Java-консоли DBA Control в версиях Oracle 7 и 8) OEM встроен в каждую СУБД, называется EM Database Express и устанавливается с помощью DBCA по желанию администратора.

Примеры страниц входа в EM Database Express и в EM Cloud Control приводятся ниже.

Умолчательный отсчет портов для EM Database Express начинается с 5500.

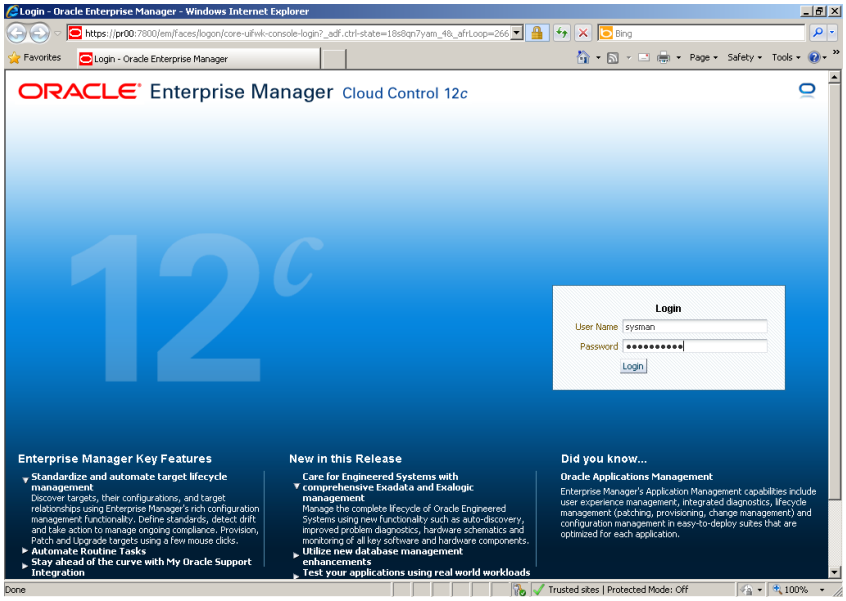
Может понадобиться доступ открыть (пользователем SYS):

```
SQL> EXEC DBMS_XDB_CONFIG.SETGLOBALPORTENABLED ( TRUE )
```

В случае схемы контейнерная БД — подключаемые БД для работы с последними понадобится самостоятельно открыть порт, например:

```
SQL> EXEC DBMS_XDB_CONFIG.SETHTTPSPORT ( 5501 )
```

EM Cloud Control:

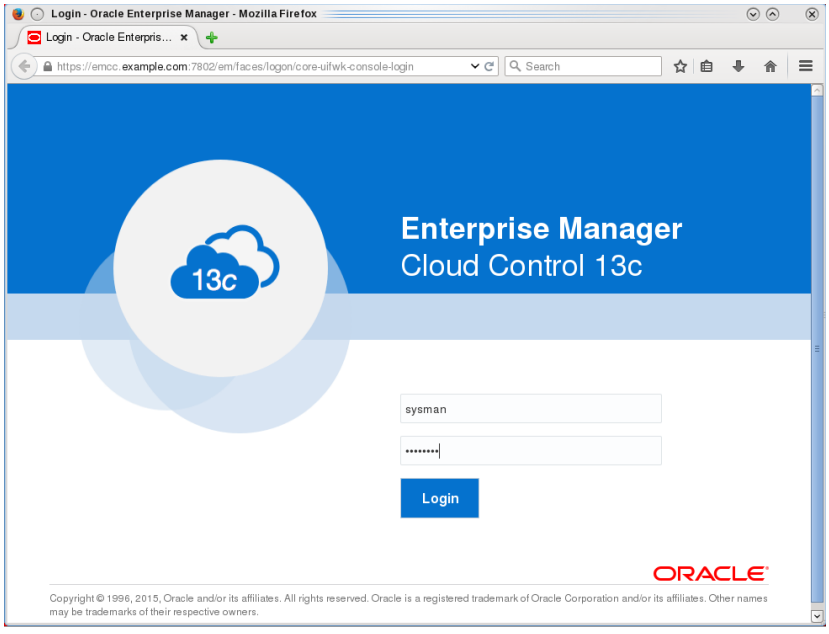


Запуск, останов и проверка состояния Cloud Control из командной строки (OMS = Oracle Management Server):

```
$ emctl start oms
$ emctl stop oms
$ emctl status oms
```

На стороне наблюдаемой БД аналогично запускается программа-агент (`$ emctl start agent` и т. д.).

Пример страницы входа в версии 13 Cloud Control:



OEM предоставляет доступ к справочным таблицам СУБД (V$- и статическим, словаря-справочника) и к AWR, доводя до администратора их содержимое в виде HTML и графиков. Дополнительно, OEM способен осуществлять некоторые вторичные (по сравнению с V$-таблицами) наблюдения и позволяет планировать потребную реакцию на события, включая пересылку сообщения по электронной почте.

... ранний селянин,
Готовясь уж косить высокой злак долин,
Услыша бури шум, не выйдет на работу
И погрузится вновь в ленивую дремоту.
А. С. Пушкин, Приметы

5. Конфигурирование, настройка и поддержка

Деятельность администратора по извлечению из СУБД дополнительной производительности:

1. Конфигурирование
2. Настройка

Формального разграничения между этими понятиями нет. Возможное интуитивное понимание следующее:

Конфигурирование: выбор набора и свойств элементов, составляющих БД, СУБД или операционную среду.

Настройка: изменение параметров элементов, составляющих БД, СУБД или операционную среду.

Под поддержкой можно понимать деятельность АБД по поддержанию БД в рабочем состоянии.

5.1. Конфигурирование составных частей БД и СУБД Oracle

5.1.1. Конфигурирование контрольного файла

Контрольный файл — двоичный файл, с которого начинается открытие БД. Создается при отработке команды CREATE DATABASE и содержит общую информацию об устройстве и состоянии БД:

- внутренний номер («DBID») и имя БД; время создания БД; номер последнего изменения в БД (SCN); SCN последней контрольной точки, некоторые общие параметры
- полные имена файлов данных; номера SCN
- полные имена файлов оперативного журнала; последовательный номер текущего журнала; историю переключений журнальных файлов
- имена табличных пространств и информация об их состоянии
- данные о резервных копиях БД для программы RMAN

Содержание контрольного файла изменяется командой ALTER DATABASE.

Внутри состоит из нескольких разделов («секций», заполняемых «записями» — по секции на каждую категорию информации, перечисленную выше), часть из которых допускает повторное использование собственной памяти, а часть нет. Пример раздела первого вида — секция LOG HISTORY, накапливающий информацию о переключениях журнала; пример раздела второго вида — секция DATAFILE, не являющаяся по характеру накопительной.

Объекты конфигурирования:

- Размер некоторых разделов и, как следствие, общий размер. Определяется один раз при создании значениями параметров MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, MAXINSTANCES (то есть изменить размеры можно только путем пересоздания).
- Число дней, в течение которых записи из *повторно используемых секций* контрольного файла гарантированно не затираются более поздними (параметр СУБД CONTROL_FILE_RECORD_KEEP_TIME).
- Количество зеркальных копий (параметр СУБД CONTROL_FILES).

Пересоздать контрольный файл с новыми характеристиками, или после потери, можно командой CREATE CONTROLFILE лишь при наличии и исправности прочих файлов БД.

Справочную информацию о контрольном файле, его копиях и содержимом можно почерпнуть из таблиц:

V$DATABASE
V$CONTROLFILE_RECORD_SECTION
V$CONTROLFILE

V$TABLESPACE
V$DATAFILE
V$TEMPFILE
V$LOGFILE
V$LOG
V$LOG_HISTORY
V$BACKUP
V$ARCHIVE
V$ARCHIVED_LOG

Пример. Запомним в файле запрос к текущему номеру SCN и номеру контрольной точке, заносимым постоянно в контрольный файл:

```
SELECT current_scn "Current SCN", checkpoint_change# "Current checkpoint
SCN"
FROM   v$database
.
SAVE show_current_and_checkpoint_scn REPLACE
```

Запросим несколько раз указанные номера:

```
@show_current_and_checkpoint_scn
/
/
/
```

Рост текущего номера SCN БД характерен для версий 10+, в которых СУБД активно исполняет внутренние задания поддержки БД. Для обычных пользователей, не имеющих прав на чтение V$DATABASE, номер можно узнать иначе:

```
SELECT DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER FROM dual;
/
/
@show_current_and_checkpoint_scn
/
/
```

5.1.2. Конфигурирование табличных пространств

Табличные пространства в Oracle — понятие, используемое для обозначения места хранения данных внутри БД; «контейнер» данных. (При этом, с точки зрения подсистемы хранения, единицами заполнения табличного пространства считаются «сегменты» данных.) Может выступать в качестве единицы группового размещения данных, переноса, резервного сохранения и восстановления, а также временного частичного (на запись) или полного (и на запись, и на чтение) закрытия доступа.

Для постоянного хранения объектов используются пространства типа PERMANENT, для хранения временных структур — пространства типа TEMPORARY, для хранения сегментов отмены (начиная с версии 9) — пространства типа UNDO.

Важнейшими элементами конфигурирования табличных пространств для постоянно хранимых данных (и отчасти двух других типов) являются следующие:

- размер блока (физическая единица обмена данными с СУБД)[9-)]
- число и расположение файлов ОС, в которых организовано конкретное табличное пространство; максимальные размеры[10-)]: SMALLFILE — до 1022-х «обычных» файлов, BIGFILE — один файл размером до 2^{32} блоков от 8 до 32К
- механизм заполнения и освобождения свободного места для «сегментов» в табличном пространстве:
  - локальный[8.1.5-)] (LOCALLY MANAGED TABLESPACE; «новый», подробнее далее), и в этом случае механизм заполнения данными блоков в «сегментах»:

- автоматический[9-), «ASSM» (SEGMENT SPACE MANAGEMENT AUTO; «новый»)
- ручной, управляемый «списком свободных блоков» (SEGMENT SPACE MANAGEMENT MANUAL; «старый»)

- централизованный (TABLESPACE MANAGED BY DICTIONARY; «старый»)

[8.1.5-) — начиная с версии 8.1.5.
[9-) — начиная с версии 9.
(-11] — после версии 11 упразднена.

Размер только кажется характеристикой табличного пространства, на самом же деле он есть характеристика файлов (или неформатированного раздела диска). Добавление файлов к табличному пространству возможно всегда, а изъятие файла (при условии, что в нем нет данных) — только с версии 10.

Свойство BIGFILE создаваемого табличного пространства возможно только при условии локального управления сегментами LOCALLY MANAGED и автоматического управления местом внутри сегментов ASSM (и то, и другое давно действует по умолчанию). Oracle рекомендует его для ASM или же RAID, где применяется чересполосное размещение файлов (striping).

К средствам поддержки работоспособности можно отнести следующее:

- разрешение/запрет записи данных (READ WRITE/ONLY)
- рабочее или отключенное состояние (ONLINE/OFFLINE) всех файлов пространства
- умолчательное (default) табличное пространство постоянных данных для вновь создаваемых пользователей(*).
- переименование табличного пространства[10-).
- указание СУБД шифровать данные в табличном пространстве перед помещением на диск и симметричной расшифровки после чтения[11-).

(*) — распространяется на пространства типа TEMPORARY и, с версии 10, типа PERMANENT, но не на UNDO
[10-) — начиная с версии 10
[11-) — начиная с версии 11

Любопытно, что состояние READ ONLY табличного пространства не препятствует удалению из него таблиц и индексов, так как эти действия отрабатываются на уровне словаря-справочника. Перевод пространства в это состояние провоцирует создание очередной контрольной точки в соответствии с измененными в нем данными.

Состояние OFFLINE невозможно для пространств SYSTEM, текущего типа UNDO и умолчательного типа TEMPORARY.

Ниже приводится общий вид устройства команды SQL для создания табличного пространства типа PERMANENT, а также команд для изменения свойств и удаления пространств всех типов:

CREATE TABLESPACE *имя_ пространства* **DATAFILE** …
{ALTER | DROP} TABLESPACE *имя_ пространства* …

Примеры:

```
CREATE TABLESPACE users2
  DATAFILE 'C:\APP\ORACLE\ORADATA\ORCL\USERS201.DBF' SIZE 200M
;
ALTER DATABASE DEFAULT TABLESPACE users2
;
ALTER DATABASE DEFAULT TABLESPACE users
--  иначе следующим шагом USERS2 не удалим
;
DROP TABLESPACE users2 INCLUDING CONTENTS AND DATAFILES
--  если не написать INCLUDING…, файл в файловой системе останется
;
ALTER TABLESPACE users
  ADD DATAFILE 'C:\APP\ORACLE\ORADATA\ORCL\USERS02.DBF' SIZE 200M
;
```

```
ALTER TABLESPACE users DROP DATAFILE 5
--  удаление возможно только с версии 10, и если файл пустой
--  здесь сослались на номер из FILE_ID в DBA_DATA_FILES вместо имени
--  файл удалился из файловой системы автоматически
;
```

Техника OMF позволяет не заботиться о названиях файлов, например:

```
ALTER SYSTEM SET db_create_file_dest = 'g:\app\oracle\oradata'
--  настроили СУБД на технику Oracle managed files
;
CREATE TABLESPACE users3 DATAFILE SIZE 1M
;
HOST dir g:\app\oracle\oradata\ORCL\DATAFILE
--  проверка; в Unix/Linux по аналогии; здесь ORCL – имя БД
DROP TABLESPACE users3
--  пустой файл пропадает сам
;
```

Одновременно, OMF не запрещает называть создаваемые файлы на свое усмотрение.

Справочную информацию о табличных пространствах можно найти в таблицах:

V$TABLESPACE
DBA/USER_TABLESPACES

5.1.3. Конфигурирование файлов табличного пространства

Важнейшими параметрами конфигурирования отдельных файлов табличного пространства являются следующие:

- способ отображения данных файлов на диск:
  - в файлы ОС (указанные явно, или назначаемые по указанным правилам автоматически при использовании Oracle Managed Files, (OMF)
  - в пространство указанных явно неформатированных разделов дисков[(-11]
  - автоматическое, в файлы «дисковых групп», заведенных предварительно на неформатированных участках дискового пространства (Automatic Storage Management, ASM)[10-)]
- при использовании файла ОС, местонахождение и имя файла
- размер: начальный, максимальный или же измененный по необходимости в сторону увеличения или уменьшения (командой ALTER DATABASE DATAFILE … RESIZE)
- возможность или невозможность автоматического увеличения размера файла при его заполнении данными и величина автоматического прироста, когда такая возможность действует
- рабочее состояние либо отключенное (ONLINE/OFFLINE).

[10-)] — начиная с версии 10.

Первый файл табличного пространства создается командой CREATE TABLESPACE, последующие — командой ALTER TABLESPACE … ADD DATAFILE/TEMPFILE. Изменение свойств файла достигается командами ALTER DATABASE DATAFILE/TEMPFILE. Примеры:

```
ALTER DATABASE DATAFILE 'C:\APP\ORACLE\ORADATA\ORCL\USERS02.DBF' RESIZE 10M
;
ALTER DATABASE DATAFILE 4 RESIZE 10M
--  то же самое, если внутренний номер файла USERS02.DBF  равен 4
;
ALTER DATABASE DATAFILE 4 AUTOEXTEND ON
;
ALTER DATABASE DATAFILE 4 OFFLINE
--  индивидуальное отключение файла сработает только при включенном архивировании БД
;
```

Перевести разом все файлы табличного пространства в рабочее, или же отключенное состояние можно одной командой **ALTER TABLESPACE** … ONLINE/OFFLINE. Режима архивирования журнальных файлов БД в этом случае уже не требуется:

```
ALTER TABLESPACE users OFFLINE
--   все файлы табличного пространства USERS отключены разом
;
```

5.1.3.1. Файлы в ASM

В случае использования ASM отдельную проблему составляет прямая работа с файлами, расположенными на дисковых группах ASM. Для работы с такими файлами (на манер Unix, и по технике командной строки) дается программа *asmcmd*:

```
>asmcmd
ASMCMD> help
.........
ASMCMD> ls
DATA/
ASMCMD> ls DATA/ORCL
CONTROLFILE/
DATAFILE/
ONLINELOG/
PARAMETERFILE/
TEMPFILE/
spfileorcl.ora
ASMCMD> cd DATA/ORCL
ASMCMD> cp spfileorcl.ora /TEMP
copying +DATA/ORCL/spfileorcl.ora -> /TEMP/spfileorcl.ora
ASMCMD> .........
```

(Программа находится в ORACLE_HOME Grid Infrastructure и исполняется для ORACLE_SID=+ASM.)

Именуются (и размещаются в директориях) файлы ASM по технике OMF, поэтому для создания файла достаточно указать только имя дисковой группы:

```
ALTER TABLESPACE users ADD DATAFILE '+DATA' SIZE 200M
;
```

Ссылка на существующий файл делается, как обычно, либо по полному имени, либо по номеру.

5.1.3.2. Перенос/переименование файлов

В процессе эксплуатации БД иногда возникает необходимость переноса файла табличного пространства с данными пользователей в другой каталог (например, на другой диск) или же задача переименования файла. Эта операция выполняется согласовано на двух уровнях, ОС и СУБД Oracle, следующими шагами:

1) [SQL*Plus или OEM] `ALTER TABLESPACE имя_пространства OFFLINE;`
2) [ОС] Переносим файл с данными *старое_имя_файла* в *новое_имя_файла*
3) [SQL*Plus или OEM] `ALTER DATABASE RENAME FILE 'старое_имя_файла' TO 'новое_имя_файла';`
4) [SQL*Plus или OEM] `ALTER TABLESPACE имя_пространства ONLINE;`

Упражнение. Переименовать файл с данными табличного пространства USERS.

Файлы табличных пространств SYSTEM и типа UNDO не могут быть отключены (это было бы равносильно лишению БД работоспособности). По это причине переименование (или же перенос) таких файлов выполняется с заменой шагов 1 и 4 на (1') перевод БД в состояние MOUNT (будет рассмотрено далее), и на (4') ее открытие.

С версии 12 перенос (и переименование) файла делается без прерывания доступа к данным в нем путем выдачи всего одной команды, например:

```
ALTER DATABASE MOVE DATAFILE 1 TO '/tmp/system01.dbf'
;
```

(Дополнительно у этой команды есть вариации записи и исполнения.)

Перенос файла *умолчательного* пространства типа TEMPORARY потребует переназначения умолчательности на другое пространство того же типа, а значит наличия (пусть на время этой процедуры) по меньшей мере двух пространств TEMPORARY.

Замечание. Перенос прочих файлов БД осуществляется иначе. Перенос контрольного файла: согласованное (состоятельное) копирование файла (или файлов) на уровне ОС и правка параметра СУБД CONTROL_FILES. «Перенос» журнальных файлов: добавление новых с удалением старых.

Справочную информацию о свойствах и показателях использования файлов данных можно найти в таблицах

V$DATAFILE
V$DATAFILE_HEADER
V$FILESTAT
V$TEMPFILE
V$TEMPSTAT
DBA_DATA_FILES
DBA_TEMP_FILES

5.1.4. Конфигурирование табличных пространств для временных данных

Пространство, указанное при создании как TEMPORARY TABLESPACE (начиная с версии 8.1; до этого были «сегменты для временных данных»), используется, в частности, для:

- временного хранения промежуточных данных при отработке SQL-запросов (с сортировкой данных, с множественными операциями UNION/INTERSECT/MINUS, с обращением к данным удаленной БД);
- данных таблиц временного хранения (таблиц GLOBAL TEMPORARY);
- временного хранилища при создании и слиянии поразрядных индексов;
- хранения данных временных объектов LOB в программе.

Порождаемые сеансом объекты в пространстве временных данных существуют там не дольше окончания сеанса.

Таких пространств в БД можно завести несколько; одно из них всегда обязано быть назначено умолчательным (default).

Создание, изменение свойств и удаление табличных пространств временных данных выполняется командами SQL:

CREATE **TEMPORARY** TABLESPACE *имя_ пространства* **TEMPFILE** …
{ALTER | DROP} TABLESPACE *имя_ пространства* …

Действия командами {ALTER | DROP} TABLESPACE для таких пространств в основном те же, что для «обычных». Специфичные возможности конфигурирования:

- удаление файла (команда ALTER DATABASE **TEMPFILE** ... DROP)[-10]
- назначение пространства умолчательным (команда ALTER DATABASE DEFAULT TEMPORARY TABLESPACE)
- группировка путем включения в «группу табличных пространств» [10-)

[-10] — с версии 10 это уже не специфика, так как удаление стало возможно и для пространств типа PERMANENT, однако возможно удаление *единственного* файла, что недопустимо для пространств PERMANENT.
[10-) — начиная с версии 10.

Примеры:

```
CREATE TEMPORARY TABLESPACE temp2
  TEMPFILE 'C:\APP\ORACLE\ORADATA\ORCL\TEMP201.DBF' SIZE 200M
;
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp2
;
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp
--  иначе следующим шагом TEMP2 не удалим
;
DROP TABLESPACE temp2 INCLUDING CONTENTS AND DATAFILES
--  если не написать INCLUDING..., файл в файловой системе останется
;
ALTER TABLESPACE temp
  ADD TEMPFILE 'C:\APP\ORACLE\ORADATA\ORCL\TEMP02.DBF' SIZE 200M
;
ALTER TABLESPACE temp DROP TEMPFILE 2
--  здесь сослались на номер из FILE_ID в DBA_TEMP_FILES вместо имени
--  файл удалился из файловой системы автоматически
;
```

Группы пространств временных данных задуманы ради увеличения места для рабочих областей временных данных на диске и ради распараллеливания доступа к таким областям; в конечном счете для ускорения обращения к ним. Создание группы выполняется неявно, путем «включения» первого пространства в до сих пор отсутствовавшую группу:

```
ALTER TABLESPACE temp TABLESPACE GROUP tempgroup1;
```

Дальнейшие аналогичные команду будут пополнять табличными пространствами уже существующую группу TEMPGROUP1. Имя группы можно использовать на равных правах с именем пространства временных данных. Удаление пространства из группы делается припиской его в группу '' (пустая строка), а удаление группы происходит автоматически при исключении из нее последнего пространства:

```
ALTER TABLESPACE temp TABLESPACE GROUP '';
```

Справочную информацию о табличных пространствах временных данных можно найти в таблицах:

DBA_TEMP_FILES
DBA_TABLESPACE_GROUPS
DBA_HIST_TEMPFILE
DBA_HIST_TEMPSTATXS
DBA_TEMP_FREE_SPACE[11-)
V$TEMPSTAT
V$TEMPFILE
V$TEMPSEG_USAGE
V$TEMP_EXTENT_MAP
V$TEMP_SPACE_HEADER

[11-) — начиная с версии 11.

Oracle советует заводить табличные пространства временных данных с локальной организацией (в последних версиях по умолчанию) с единым размером экстента. Для БД, обслуживающей приложения типа DSS (OLAP) или интенсивно использующих временные объекты LOB (то есть в программах)

рекомендуется выбор экстента от 2 до 10 Мб. В БД для приложений типа OLTP или использующих небольшие по размеру таблицы временного хранения данных (GLOBAL TEMPORARY) рекомендуется размер кратный 64 Кб до 1 Мб величиной. Контролировать использование таких пространств удобно через таблицу V$TEMPSEG_USAGE. Пример запроса:

```
SELECT
    t.username
  , t.session_num
  , t.blocks
  , s.sql_fulltext
FROM
  v$tempseg_usage t INNER JOIN v$sql s
  USING ( sql_id )
;
```

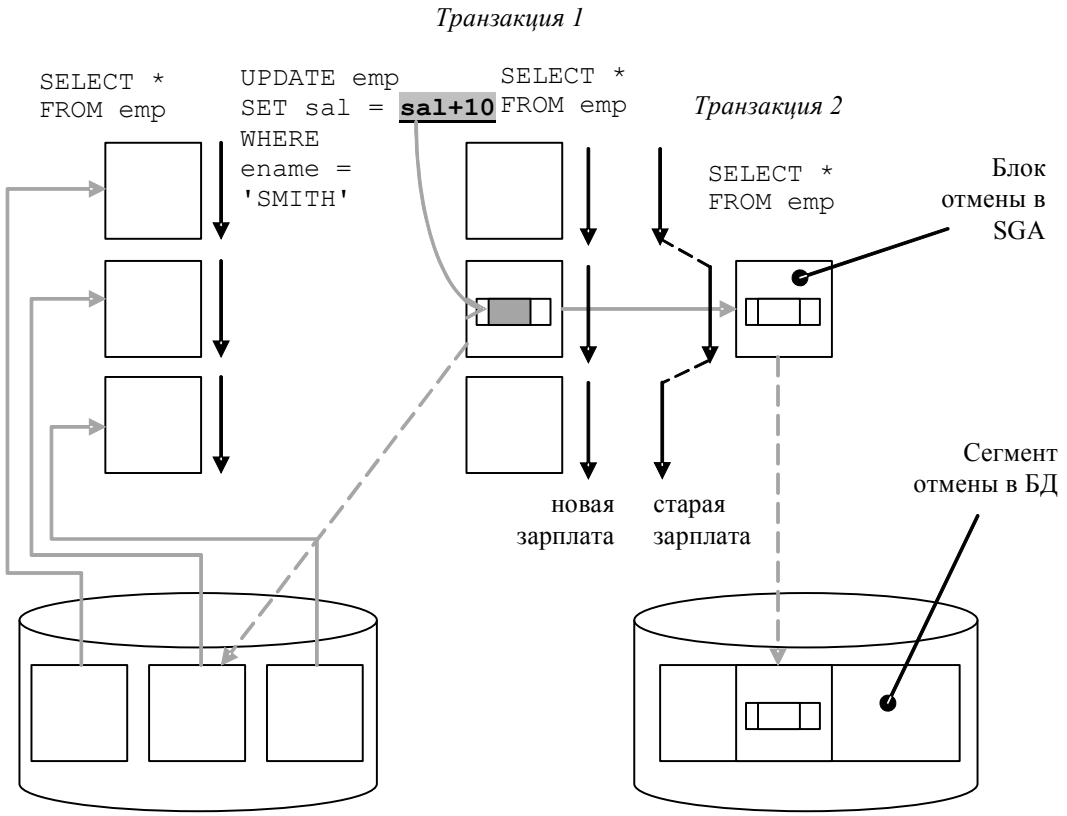
5.1.5. Конфигурирование данных отмены (сегментов отката)

Сегменты отмены/отката есть служебные структуры в Oracle, предназначенные для запоминания исходных представлений блоков данных в SGA перед внесением в блок изменений в соответствии с очередной командой DML. Такие исходные представления необходимы:

a) для поддержки целостного вида на выбираемые из таблиц данные при исполнении SELECT;
b) для обеспечения другим транзакциям правильного вида данных таблицы, подвергающейся изменению;
c) для выполнения команды ROLLBACK *отката* транзакции (*отмены* изменений);
d) для обращения к старым данным таблиц (с версии 9) — для удобства разработчиков.

До версии 9 использовалось название «сегменты отката» (*rollback* segments; при том, что тип таких сегментов в табличном пространстве был ROLLBACK); с версии 9 используются название «сегменты отмены» (*undo* segments; тип сегментов — TYPE2 UNDO) и обобщающий термин «данные отмены» (undo).

Основное использование данных отмены(/отката) поясняется следующим рисунком:



Если *Транзакцией 1* завершится фиксацией (COMMIT), правленый в SGA блок с данными будет сброшен на свое исходное место в файле БД, в противном случае нет. «Ветка» прочтения данных *Транзакцией 2* обращается к блоку отмены (созданному перед изменением данных *Транзакцией 1*) и носит название «согласованное чтение» (буквально — «состоятельное» чтение; consistent read, CR). Большое количество блоков отмены, порожденное пока незавершенной транзакцией, увеличивает размер сегмента отмены, к которому прикреплена транзакция, и создает нагрузку на БД.

Технически, блок отмены не есть точная копия блока с данными до его правки, но содержит информацию, позволяющую его в точности восстановить.

5.1.5.1. Конфигурирование сегментов отката в версии 8 и предшествовавших

Объектами конфигурирования для аппарата работы сегментов отката могут быть:

- число и размер сегментов отката
- месторасположение сегментов отката
- параметр OPTIMAL
- INIT-параметры:

ROLLBACK_SEGMENTS	Список сегментов отката для использования экземпляром СУБД
TRANSACTIONS_PER_ ROLLBACK_SEGMENT	Планируемое администратором число одновременных транзакций для каждого используемого сегмента отката. Если в *INIT.ORA* указан параметр TRANSACTION = 101 (число одновременных транзакций в системе), а параметр TRANSACTIONS_PER_ ROLLBACK_SEGMENT = 10, то число сегментов отката для успешного старта Oracle должно быть не менее CEIL(101/10) = 11.
MAX_ROLLBACK_SEGMENTS	Максимальный размер буфера сегментов отката в SGA, то есть максимальное число сегментов отката одновременно в рабочем (online) состоянии. Значение по умолчанию: MAX (30, TRANSACTIONS / TRANSACTIONS_PER_ROLLBACK_SEGMENT).

Создание, изменение и удаление сегментов отката выполняется SQL-командами:

{CREATE | ALTER | DROP} **ROLLBACK SEGMENT** *имя_сегмента* …;

Некоторые рекомендации по конфигурированию сегментов отката:

- Выделять для сегментов отката отдельное табличное пространство и помещать его на отдельный дисковод.
- Не создавать сегменты отката в пространстве SYSTEM и (в версиях до 8.1) не забыть удалить публичный сегмент отката, создаваемый автоматически на период создания БД.
- Следить, чтобы в табличном пространстве сегментов отката было достаточно места для роста сегментов (особенно интенсивного при пакетных загрузках).
- В случае сегментов отката размеры INITIAL и NEXT должны быть строго одинаковыми (рекомендуется это указать в качестве DEFAULT STORAGE при создании пространства для сегментов).
- Не следует забывать при расчетах, что каждый сегмент отката имеет не менее двух экстентов, т.е. минимальный размер — INITIAL + NEXT.
- Указывать в определении сегмента значение OPTIMAL, чтобы не приходилось время от времени уменьшать размер сегмента вручную после большого роста (однако при этом не задавать OPTIMAL слишком маленьким, чтобы сегмент «часто не рос», затрачивая на это системное время, то есть вел себя более статично).

5.1.5.2. Конфигурирование сегментов отмены в версиях 9+

В версии 9 появился «автоматический», механизм поддержки хранения исходных видов модифицируемых данных (новый тип UNDO табличного пространства). Его назначение — снять с администратора прежнюю заботу о конфигурировании сегментов отката. Кроме этого в пространстве типа UNDO обеспечивается ранее отсутствовавшая возможность хранить прежние значения измененных в какой-то момент данных в течение некоторого времени, с тем, чтобы позднее к ним при желании обратиться. С помощью специальных параметров СУБД версии Oracle 9+ можно запускать либо в старом режиме поддержки откатов, либо в более позднем, поддержки данных отмены.

При использовании в БД *сегментов отмены* и табличного пространства типа UNDO, один *сегмент отката*, SYSTEM, принадлежащий пользователю SYS, и расположенный в табличном пространстве SYSTEM, все же остается. СУБД использует его для выполнения изменений в таблицах словаря-справочника. Кроме того в пространстве SYSTEM Oracle будет самостоятельно заводить «отложенные» сегменты отмены (deferred undo, или save undo) при восстановлении табличных пространств в состоянии OFFLINE; при переводе восстановленного пространства в ONLINE СУБД перенесет изменения в данных (буде они были) из отложенных сегментов в это пространство, и, за дальнейшей ненадобностью, их удалит. Отложенные сегменты не требуют конфигурирования со стороны АБД, но способны на время увеличивать дисковые потребности файлов пространства SYSTEM.

Объектами конфигурирования табличного пространства типа UNDO являются общие характеристики табличных пространств всех типов, например:

- размер файлов,
- месторасположение файлов на диске.

Суммарный размер файлов (типично — одного файла) выбирается с учетом целей, в известной мере противоречивых:

(а) обеспечить целостность читаемых из таблицы в долгих запросах данных, и
(б) обеспечить желаемый срок хранения старых данных для возможности к ним обращаться.

Корень противоречия в том, что для достижения обеих целей используется одно общее пространство данных UNDO (долго длящийся SELECT может оставить недостаточно блоков в пространстве UNDO для хранения на текущий момент достаточно старых данных, и наоборот). Подобрать для установки необходимый размер помогает советник UNDO Advisor, работать с которым можно средствами пакета DBMS_ADVISOR. OEM в версиях 10+ дает графический интерфейс для работы с UNDO Advisor и отображает свои советы (построенные на основе анализа предшествующей нагруженности БД работой) в виде графика.

Собственными элементами конфигурирования пространства UNDO являются:

- параметры СУБД:

UNDO_MANAGEMENT	Значение AUTO (по умолчанию[11-]) заставляет СУБД

	работать с сегментами *отмены* и пространством UNDO.
UNDO_TABLESPACE	Имя «рабочего» в БД табличного пространства типа UNDO (когда UNDO_MANAGEMENT = AUTO).
UNDO_RETENTION	Количество секунд, в течение которых СУБД будет *пытаться* хранить (не затирать) исходные образы блоков перед внесением в них правок в SGA. По умолчанию 900.
TEMP_UNDO_ENABLED[12-)	Если = TRUE, СУБД создает сегменты отмены таблиц GLOBAL TEMPORARY не в пространстве UNDO, а в пространстве TEMPORARY (и работать с такими таблицами можно в состоянии БД READONLY). Для БД горячего резерва *standby* это автоматическое поведение.
UNDO_SUPPRESS_ERRORS[9]	Если = TRUE, СУБД не сообщает об ошибках при выполнении команд непосредственной работы с сегментами отката (например, ALTER ROLLBACK SEGMENT …) старыми приложениями, написанными из расчета на сегменты откатов.

[12-) — с версии 12.
[9] — в версии 9.

- свойство RETENTION GUARANTEE/NOGUARANTEE табличного пространства UNDO: RETENTION GUARANTEE *гарантирует* сохранение старых образов блоков в течение срока, заданного параметром СУБД UNDO_RETENTION, после их правки в SGA и *гарантирует* выполнение быстрых (flashback) запросов к старым данным в течение этого срока[10-).

[11-) — начиная с версии 11.
[10-) — начиная с версии 10.

1. При фиксированном размере табличного пространства UNDO время удержания, задаваемое UNDO_RETENTION, может не выдерживаться, если текущие транзакции активно порождают данные отмены.
2. Если установлено свойство RETENTION GUARANTEE, удержание гарантируется (в том числе выполнение долгих запросов), однако если место в пространстве UNDO заканчивается, выполнение команд SQL получает шанс завершаться ошибкой.
3. При авторастущем пространстве UNDO СУБД будет подстраиваться под время удержания, ориентируясь на наиболее долгие исполняемые запросы.
4. Малое значение UNDO_RETENTION, нехватка места в пространстве UNDO и изменения данных другими транзакциями могут порождать риск ошибки ORA-01555: snapshot too old при долгом исполнении SELECT.

Создание, изменение свойств и удаление табличных пространств типа UNDO выполняется командами SQL:

CREATE **UNDO** TABLESPACE *имя_пространства_UNDO* **DATAFILE** …
{ALTER | DROP} TABLESPACE *имя_пространства_UNDO* …

Примеры:

```
CREATE UNDO TABLESPACE undotbs2
  DATAFILE 'C:\APP\ORACLE\ORADATA\ORCL\UNDOTBS201.DBF' SIZE 2M
;
ALTER TABLESPACE undotbs2 RETENTION GUARANTEE
;
ALTER SYSTEM SET undo_tablespace = undotbs2 /* на время: */ SCOPE = MEMORY
;
--  ... поработали с UNDOTBS2; при этом ранее открытые сегменты отмены
--  ... сохраняются в старом пространстве UNDO, пока в них остается нужда
--  ...
ALTER SYSTEM SET undo_tablespace = undotbs1
;
DROP TABLESPACE undotbs2 INCLUDING CONTENTS AND DATAFILES
```

```
--  если не написать INCLUDING…, файл в файловой системе останется
;
```

Иногда подобное временное переключение на искусственное «маленькое» пространство позволяет быстро освободить место в основном пространстве UNDO, исчерпанное в результате разрастания в нем сегментов отмены после выполнения чрезмерно длинных транзакций, или по другим причинам.

Пример обращения к старым данным (на основе TIMESTAMP, но можно по DATE, SCN):

```
SELECT * FROM scott.emp AS OF TIMESTAMP ( SYSTIMESTAMP - INTERVAL '10' MINUTE )
;
```

5.1.5.3. Справочная информация

Справочную информацию о наличии, свойствах и характеристиках использования *конкретно* сегментов отмены/отката можно найти в таблицах:

V$ROLLSTAT
V$ROLLNAME
DBA_ROLLBACK_SEGS
V$UNDOSTAT
DBA_UNDO_EXTENTS
V$TEMPUNDOSTAT[12-)
[12-) — с версии 12.

Например, для удаления нерабочего пространства типа UNDO требуется, чтобы в нем не оставалось сегментов отмены с востребованной информацией о транзакциях (например, незакрытых). Эти сведения даст таблица DBA_ROLLBACK_SEGS или же V$ROLLSTAT (текущие рабочие сегменты и их состояния) в комбинации с DBA_ROLLBACK_SEGS или DBA_SEGMENTS.

V$UNDOSTAT дает статистику использования блоков в сегментах отмены за последнее время (до 4-х дней) с 10-минутной разбивкой, характеризуя загрузку пространства UNDO работой. Например:

```
SELECT
  TO_CHAR ( begin_time, 'HH24:MI' ) begintime
, TO_CHAR ( end_time,   'HH24:MI' ) endtime
, undoblks                          blocks
, txncount                          transactions
, tuned_undoretention
FROM     v$undostat
ORDER BY begin_time
;
```

V$TEMPUNDOSTAT дает статистику использования пространства TEMPORARY для нужд данных (блоков) отмены таблиц GLOBAL TEMPORARY в случае TEMP_UNDO_ENABLED = TRUE. (При этом сам параметр сессионный.)

5.1.5.4. Отдельное накопление данных отмены для нужных объектов

Хранение данных отмены в пространстве UNDO подчинено общим правилам для всех объектов в БД. Иногда хочется каким-то отдельным таблицам обеспечить более длительное хранение. В версиях 11+ это достигается созданием «архива быстрых старых данных» (архива FLASHBACK, Flashback data archive, FDA). Он создается на основе одного или нескольких (по желанию) табличных пространств, а в свойствах таблицы мы свяжем ее с этим архивом; тогда данные UNDO, порождаемые при работе с этой таблицей, будут поступать в табличные пространства архива, а не UNDO. Время удержания для архива можно указать значительно дольше, чем параметром UNDO_RETENTION.

Пример:

```
CREATE TABLESPACE fda_ts1
  DATAFILE 'C:\APP\ORACLE\ORADATA\ORCL\FDA_TS1F01.DBF' SIZE 20M
;
CREATE FLASHBACK ARCHIVE DEFAULT db_archive1 TABLESPACE fda_ts1 RETENTION 1
DAY
;
ALTER FLASHBACK ARCHIVE db_archive1 MODIFY RETENTION 2 MONTH
--   передумали ...
;
CREATE TABLE scott.emp_copy AS SELECT * FROM scott.emp
;
ALTER TABLE scott.emp_copy FLASHBACK ARCHIVE
--   с другими таблицами так же
;
```

Обращение к старым данным делается, как обычно.

```
ALTER TABLE scott.emp_copy NO FLASHBACK ARCHIVE
--   развязались ...
;
DROP FLASHBACK ARCHIVE db_archive1
;
DROP TABLESPACE fda_ts1 INCLUDING CONTENTS AND DATAFILES
;
```

Справочные таблицы:

DBA_FLASHBACK_ARCHIVE
DBA_FLASHBACK_ARCHIVE_TABLES
DBA_FLASHBACK_ARCHIVE_TS

Эта техника получила развитие в версии 12.

5.1.6. Конфигурирование журнальных файлов

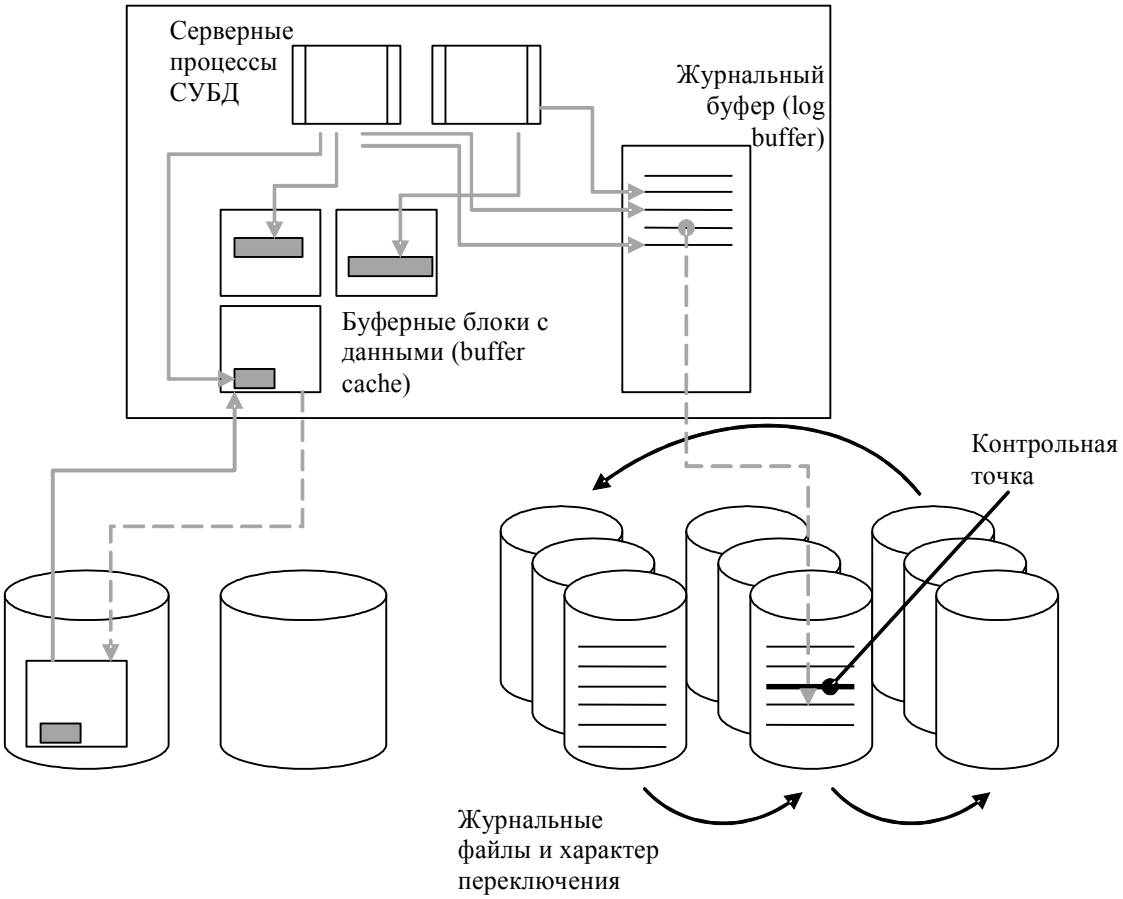
Журнализация изменений в БД (redo log, журнализация ради *повторения* внесения изменений) используется в Oracle с основною целью восстановления данных, которые не успели закрепиться в файлах БД по причине *аварийного* останова работы СУБД.

С этой целью каждое изменение в данных на блоках в SGA СУБД сопровождает помещением в журнальный файл БД *журнальной записи*, полностью описывающей действие соответствующей команды SQL. Число журнальных файлов ограничено, и заполняются они журнальными записями в закольцованной очередности. Обнаружив при открытии БД, что предыдущий останов был аварийным, СУБД самостоятельно начнет процесс такого восстановления. Для этого она:

1. найдет в журнальной «ленте» в журнальных файлах место, выше которого расположены только записи о командах DML, результат действия которых успел попасть в файлы БД (найдет *контрольную точку*);
2. повторит по очереди все операции DML, начиная с контрольной точки, двигаясь по журнальной ленте вперед («подкатит БД»);
3. откроет БД и (параллельно с обслуживанием сеансов пользователей) откатит все незакрытые к моменту сбоя транзакции.

Когда журнальные файлы архивируются (после перевода БД в специальный *режим архивирования*), та же идея восстановления данных БД используется для восстановления с резервных копий файлов БД, снятых достаточно давно.

Механизм журнализации внесения изменений в БД поясняется следующим рисунком:



Объектами конфигурирования процесса журнализации в Oracle могут быть:

- число и размер [групп] журнальных файлов;
- число файлов в группах (то есть наборов зеркальных копий файлов, создаваемых для повышения сохранности журнальной информации);
- места расположения журнальных файлов;
- параметры СУБД:

LOG_BUFFER	Размер буфера журнальных записей в SGA.
LOG_ARCHIVE_MAX_PROCESSES	Максимальное число процессов ARC*n* архивирования журналов (по умолчанию 1).
LOG_CHECKPOINTS_TO_ALERT	Если = TRUE, формирование контрольной точки протоколируется в *ALERT.LOG*.
FAST_START_MTTR_TARGET	Регулирует интенсивность формирования инкрементальной контрольной точки.
LOG_CHECKPOINT_TIMEOUT	Устанавливает предел времени, в течение которого любой измененный блок имеет право быть не сброшенным на диск.
LOG_ARCHIVE_BUFFERS[-8]	Число буферов для процесса архивирования журналов.
LOG_ARCHIVE_BUFFER_SIZE[-8]	Размеры буферов архивирования журналов.

[-8] в версии 9 переведены в разряд недокументированных (скрытых).

Некоторые рекомендации по конфигурированию журнальных файлов:

- Завести более одного журнального файла в группе (Oracle обеспечивает идентичность содержимого файлов одной группы самостоятельно). (→ надежность).

- Теоретический минимум количества файлов (групп) 2 следует на практике заменить на 3. (→ надежность).
- Завести количество журнальных файлов (групп), достаточное, чтобы при работе СУБД не возникало ожиданий.
- Малый размер журнальных файлов чреват частым переключением, а большой — вероятностью потери недавних изменений в БД при порче файла вместе с отказом СУБД.

Примеры действий:

```
ALTER SYSTEM SWITCH LOGFILE
--   перенос записей из журнального буфера в журнальный файл и переключение на следующий;
--   до версии 8.1 вдобавок к этому формирование полной контрольной точки (команда ниже)
;
ALTER SYSTEM CHECKPOINT
--   сброс измененных блоков из SGA в файлы с данными и простановка новой контрольной точки
--   в заголовки файлов с данными и в контрольный файл
--   (столбец CHECKPOINT_CHANGE# в V$DATAFILE и в V$DATABASE соответственно)
;
```

Первая из приведенных команд оставит прежнюю текущую журнальную группу на некоторое время в состоянии ACTIVE, а вторая вручную переведет в INACTIVE.

Действие последних двух команд помогает проследить таблица V$DATABASE:

```
@show_current_and_checkpoint_scn
/
/
ALTER SYSTEM SWITCH LOGFILE;
@show_current_and_checkpoint_scn
/
/
ALTER SYSTEM CHECKPOINT;
@show_current_and_checkpoint_scn
/
/
```

Если включен режим архивирования журнальных файлов, а архивная копия фактически не создается (например, вследствие нехватки места на диске), то по достижении незаархивированного журнального файла СУБД прекратит принимать изменения (но будет продолжать обрабатывать запросы).

С точки зрения производительности, в обычных случаях фирма Oracle рекомендует для приложений типа OLTP выбирать размер журнальных файлов так, чтобы *обычно* переключение происходило не чаще раза в 20 минут. В системах горячего резерва, с применением потоков данных (Streams) или при особо жестких требованиях к сохранности вносимых изменений допустимыми могут считаться и более частые переключения. Для подбора достаточного размера журнальных файлов можно ориентироваться на следующие значения из таблицы V$INSTANCE_RECOVERY: OPTIMAL_LOGFILE_SIZE, а затем WRITES_LOGFILE_SIZE:

```
SELECT optimal_logfile_size FROM v$instance_recovery
;
```

Справочную информацию о текущем состоянии журнализации можно найти в таблицах:

V$LOG
V$LOGFILE
V$LOG_HISTORY
V$THREAD
V$INSTANCE_LOG_GROUP
V$INSTANCE_RECOVERY

Пример:

```
SELECT
    group#
  , thread#
  , sequence#
  , bytes
  , status
  , first_change#
  , next_change#
FROM
  v$log
;
SELECT * FROM v$log_history
;
```

Журнальные записи формируются в двоичном виде, и прочитать их из файлов текстовым редактором нет возможности. При необходимости это сделать следует (а) либо воспользоваться средством Log Miner (программно — через обращение к встроенным пакетам, графически — средствами OEM), (б) либо выдать команду сброса содержимого журнального файла (в том числе архивированного) в трассировочный файл текущего процесса Oracle, например:

```
ALTER SYSTEM DUMP LOGFILE 'C:\APP\ORACLE\ORADATA\ORCL\REDO01.LOG';
```

У этой команды допускаются уточнения для выдачи журнальных записей, соответствующих нужному диапазону времени, номеров состояний БД SCN и пр.

Сообщения о переключениях журнальных файлов автоматически заносятся в файл *ALERT.LOG*.

5.2. Конфигурирование хранения и использования данных некоторых объектов БД

5.2.1. Общее конфигурирование сегментов данных таблиц

В дополнение к «обычной» организации таблиц в Oracle существует некоторое разнообразие специальных форм, например таблицы с раздельным хранением строк и индексно-организованные таблицы (версия 8; см. ниже), таблицы с внешним хранением (версия 9) и т. д. Специальные формы организации и хранения таблиц далее рассматриваются отдельно.

Ниже рассматриваются общие свойства хранения и употребления таблиц с обыкновенным размещением строк по принципу «первого подходящего» блока. Такое устройство имеют не только (а) обычные *таблицы*, но и (б) таблицы с данными *mview*, а так же (в) *разделы* таблиц с раздельным хранением строк (см. далее). Бо́льшая часть сказанного ниже для обычных таблиц (а) распространяется и на (б) и (в).

5.2.1.1. Пересоздание или корректировка сегмента данных таблицы

Таблицы, где часто изменяются старые строки, могут со временем ухудшить свои рабочие характеристики из-за накопления чрезмерной фрагментированности (а значит и сверх необходимого разрастания) собственного сегмента с данными или из-за чрезмерного удлиннения «списков свободных блоков» (последнее — если для табличного пространства не задано использование поразрядной карты заполненности блоков сегмента, ASSM, как альтернатива списку свободных блоков).

В таких случаях одна из рекомендаций состоит в пересоздании сегмента данных таблицы заново командой ALTER TABLE … MOVE:

```
ALTER TABLE emp MOVE;
```

По этой команде СУБД заведет новый сегмент, перенесет в него строка за строкой все данные из старого, назначит таблице новый сегмент, а старый удалит. Новый сегмент окажется максимально плотный и компактный, как после загрузки данных.

Команда ALTER TABLE … MOVE распространяется на разделы таблиц, но не не распространяется на таблицы с данными mview.

Следующее предложение пересоздаст сегмент в другом табличном пространстве. Это дает дополнительный ресурс увеличения скорости доступа благодаря возможности иного расположения данных на внешнем носителе или, с версии 9, иному размеру блока:

```
ALTER TABLE emp MOVE TABLESPACE sysaux;
```

Еще пример изменения характеристик хранения (тут более полное заполнение блоков):

```
ALTER TABLE emp MOVE PCTFREE 0;
```

С версии 12:

```
ALTER TABLE emp MOVE ONLINE;
ALTER TABLE emp MOVE ONLINE TABLESPACE sysaux;
```

и так далее. На такое пересоздание сегмента имеются небольшие специфические ограничения.

С версии 10 в табличных пространствах с автоматическим управлением памятью в сегменте (ASSM) команда ALTER TABLE позволяет дефрагментировать существующий сегмент, а не создавать новый. При указании ALTER TABLE … SHRINK SPACE СУБД дефрагментирует сегмент, перенесет заполненные блоки к его началу, сдвинет наивысшую точку заполнения сегмента (high watermark, HWM) к началу и вернет незанятые блоки табличному пространству. При указании SHRINK SPACE COMPACT СУБД только дефрагментирует сегмент и перенесет заполненные блоки к его началу. Пример:

```
ALTER TABLE emp ENABLE ROW MOVEMENT;
ALTER TABLE emp SHRINK SPACE;
ALTER TABLE emp SHRINK SPACE COMPACT /* -- вариант; HWM остается на месте */;
```

Выдача команды ALTER TABLE … MOVE влечет изменение физических адресов строк таблицы, в силу чего связанные с таблицей индексы потребуется откорректировать. Вариант ALTER TABLE … SHRINK SPACE в общем случае приводит к изменению физических адресов строк таблицы в рамках старого сегмента, а это требует специального предварительного разрешения (ENABLE ROW MOVEMENT).

К тому же приведет в конечном счете пересоздание *таблицы*:

(а) программами *exp* и *imp* или *expdp* и *impdp*
(б) SQL-командами CREATE TABLE y AS x, DROP TABLE x и RENAME y TO x
(в) с помощью систем программирования третьих фирм (фактически будут выполняться действия, описанные выше)

Однако пересоздание хуже использования MOVE или SHRINK тем, что приводит к частичной потере ограничений целостности для таблицы и индексов.

5.2.1.2. Умолчательные свойства некоторых выполняемых с таблицами операций

Для большинства таблиц есть возможность установить несколько свойств, способных повлиять на технику выполнения над ними ряда операций DML. Это:

CACHE/NOCACHE,
PARALLEL/NOPARALLEL,
COMPRESS/NOCOMPRESS,
LOGGING/NOLOGGING.

Фактически перечисленные «свойства» соответствуют свойствам конкретных выполняемых операций SQL и могут указываться индивидуально для операций (подсказками либо ключевыми словами); в применении же к таблицам они играют роль умолчательных установок.

Свойство BUFFER_POOL является собственно свойством таблицы.

5.2.1.2.1. Кеширование блоков при сканировании таблиц

В последних версиях Oracle с помощью указания CACHE/NOCACHE в предложениях CREATE/ALTER TABLE … можно явно задать тот участок LRU-списка для буфера блоков в SGA («свежий» или «старый», «горячий» или «холодный»), в котором будут размещаться указатели на прочитанные блоки таблицы в процессе полного ее сканирования. Пример:

```
CREATE TABLE test ( col NUMBER )
TABLESPACE users
CACHE;
```

(Стандартное поведение СУБД приводит к размещению указателей на прочитанные в результате полного сканирования больших таблиц блоков, вопреки обыкновению, сразу в «холодный» участок, что приводит к ускоренному их устареванию).

Умолчательный режим кеширования блоков для каждой таблицы можно сменить, например:

```
ALTER TABLE test NOCACHE;
```

Того же результата можно достичь с помощью подсказки CACHE в конкретном предложении SQL:

```
SELECT /*+ CACHE ( test ) */ col FROM test;
```

В любом случае общий порядок хранения в буфере блоков конкретной таблицы изменится, так что последующие обращения к ее строкам ускорятся.

5.2.1.2.2. Распараллеливание обработки таблиц

Таблицу можно создать в режиме параллельной обработки или перевести в него, например:

```
ALTER TABLE emp PARALLEL;
ALTER TABLE emp PARALLEL 4;
ALTER TABLE emp NOPARALLEL;
```

То же можно сделать с помощью подсказки PARALLEL в конкретном предложении SQL:

```
SELECT /*+ PARALLEL ( e 2 ) */ ename FROM emp e;
```

Хотя этот режим формально включается во всех вариантах СУБД Enterprise Edition, реальную отдачу от него можно получить только при наличии у платформы нескольких процессоров.

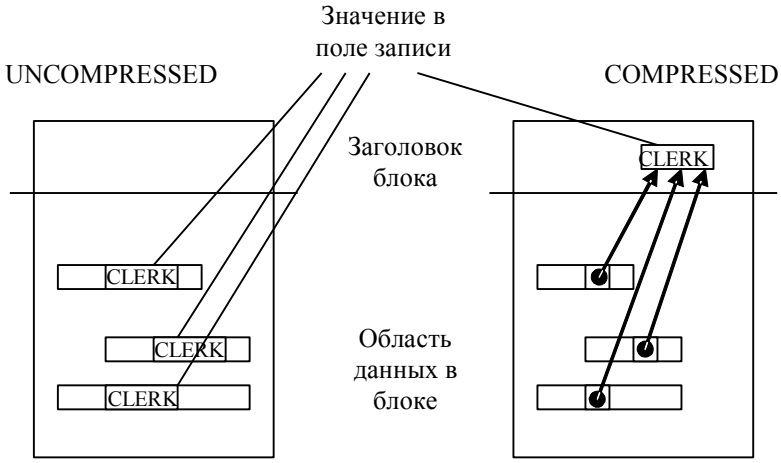
5.2.1.2.3. Уплотненное хранение данных в блоке

Начиная с версии 9.2 для таблицы можно указать особый режим уплотненного хранения (сжатого, ужатого, compressed). Пример:

```
ALTER TABLE emp COMPRESS;
```

Компактность обеспечивается СУБД на уровне блока, за счет выноса одних и тех же повторяющихся в разных записях значений (например, одинаковая «должность» у разных сотрудников) из области данных в справочную часть блока, а в строку вместо самого значения ставится ссылка на него. Фактический

выигрыш компактности зависит от количества одинаковых значений в полях записей, попавших в общий блок:



Уплотненное хранение приводит к ускорению доступа за счет уменьшения общего количества блоков с данными, а в случае некоторых запросов и за счет более быстрого извлечения строки из блока.

Реально уплотненное хранение будет выполняться только при количестве столбцов не более 255 (при том, что по завершении процедуры «уплотнения» соотвествующий признак таблицы в словаре-справочнике все равно окажется установлен).

В версиях 9.2 и 10 уплотненное хранение для таблиц с указанием COMPRESS возникает только при прямоточной вставке (direct path INSERT; в частности обеспечивается загрузчиком SQL*Loader в виде «прямоточной загрузки», direct load, а так же программой импорта *impdp*, но не программой *imp*) и не сохраняется при прочих операциях со строками, например при вставках отдельных строк. Обычные команды INSERT и UPDATE будут обрабатываться обычным образом, так что при активном изменении данных уплотненность может пропасть.

В версии 11.1 такое поведение сохранилось, но помимо COMPRESS с равным успехом стало возможным указывать по-новому: COMPRESS **FOR DIRECT_LOAD** [OPERATIONS]. В противовес этому появилось указание COMPRESS **FOR ALL** [OPERATIONS], при котором уплотненние данных в блоках будет поддерживаться также при обычных операциях INSERT и UPDATE (Oracle будет накапливать изменения в блоке обычным образом, а когда сочтет своевременным, автоматически доуплотнит данные, как было указано раньше). При этом уплотнение данных окажется в целом несколько менее эффективным, чем в первом случае.

С версии 11.2 предлагается вместо COMPRESS или COMPRESS FOR DIRECT_LOAD писать COMPRESS **BASIC** (уплотнения по типу Basic Compression), а вместо COMPRESS FOR ALL писать COMPRESS **FOR OLTP** (уплотнения по типу Advanced Compression). Примеры:

```
ALTER TABLE emp COMPRESS FOR OLTP
--   таблицы с частым изменением строк
;
ALTER TABLE emp COMPRESS BASIC
--   для таблицы с нечастым изменением строк
;
```

5.2.1.2.4. Отключение журнализации

Свойство NOLOGGING позволяет практически отменить журнализацию при выполнении прямоточного добавления строк в таблицу, что способно дать значительный выигрыш в скорости. Пример:

```
ALTER TABLE emp NOLOGGING;
```

5.2.1.2.5. Прикрепление к указанной секции буфера блоков

Свойство BUFFER_POOL, указываемое во фразе STORAGE, позволяет сообщить СУБД, хотим ли мы помещать буферизуемые блоки таблицы в секции буферной области DEFAULT, KEEP или RECYCLE (при необходимости последние две из них должны быть явно заведены администратором). Пример:

```
ALTER TABLE emp STORAGE ( BUFFER_POOL KEEP );
```

Выполненное действие говорит о том, что мы намерены обращаться к относительно небольшой таблице EMP часто и без лишних задержек (если таблица велика, привязывание ее блоков к области KEEP наразумно).

5.2.2. Общее конфигурирование сегментов данных индексов

Индексы в Oracle допускают довольно большое разнообразие выбора внутренней организации (см. ниже: древовидные/поразрядные индексы; уникальный/неуникальный индекс; простой/составной индекс; индекс с инвертированным ключом; индекс, основанный на функциональном преобразовании значения ключа; глобальная/локальная организация индексов для таблиц с раздельным хранением строк). Выбор разработчиком БД конкретной организации есть первый шаг конфигурирования конкретного индекса.

Ниже рассматриваются некоторые возможности конфигурирования индексов с наиболее популярноой организацией в виде B*-дерева.

5.2.2.1. Пересоздание или корректировка сегмента индекса

Индексы часто изменяемых таблиц могут со временем ухудшить свои эксплуатационные характеристики из-за накопления чрезмерной фрагментированности (а значит увеличения сверх необходимого размера) сегмента, а так же из-за неудачно сложившейся структуры внутренней организации или даже нарушений внутренней структуры. В последнем случае индекс может оказаться в состоянии UNUSABLE, и будет игнорироваться в планах запросов.

В таких случаях сегмент индекса рекомендуется:

- пересоздать (подправить) командой ALTER INDEX … **REBUILD**, или же
- дефрагментировать командами:
  - (все версии) ALTER INDEX … **COALESCE**, или
  - (с версии 10) ALTER INDEX … **SHRINK SPACE COMPACT**
    ALTER INDEX … **SHRINK SPACE**.

Более поздний вариант SHRINK SPACE COMPACT равносилен более раннему по созданию варианту с COALESCE (дублирующий синтаксис прежнего действия).

Действие REBUILD более затратно, но дает и лучший эффект, выполняя при необходимости и перестройку древовидной структуры. При выполнении REBUILD из листьевых блоков удаляются старые вхождения элементов индекса (пар <*индексированное значение*, *ROWID*>), которые не удаляются при обычных операциях UPDATE, и могут накапливаться, например, при «скользящем изменении» строк в таблицах. Кроме того REBUILD способно перевести индекс из состояния UNUSABLE в состояние USABLE. При этом REBUILD выполняется быстрее пересоздания заново, поскольку не выполняет внутреннюю сортировку элементов индекса, пользуясь имеющимся в старой структуре порядком (как при указании NOSORT во время создания индекса).

Выполнение REBUILD не препятствует запросам к таблице, но препятствует ее изменениям другими транзакциями. Блокировки таблицы, препятствующей операциям DML (не особенно интенсивным, однако), можно избежать, уточнив команду, например:

```
ALTER INDEX pk_dept REBUILD ONLINE;
```

Расплачиваться за REBUILD ONLINE приходится тем, что во время такой перестройки перестает действовать распараллеливание выполнения запросов к таблице.

Действия COALESCE/SHRINK SPACE менее затратны для СУБД, но только дефрагментируют и только листьевую часть индекса.

Перенос сегмента индекса в другое табличное пространство, возможно с другим размером блока или местом хранения (то есть аналог команды ALTER TABLE … MOVE TABLESPACE …), выполняется командой ALTER INDEX … REBUILD:

```
ALTER INDEX pk_dept REBUILD TABLESPACE sysaux;
```

5.2.2.2. Умолчательные свойства некоторых выполняемых с индексами операций

Для большинства индексов, как и для таблиц, есть возможность установить несколько свойств, способных повлиять на технку выполнения над ними ряда операций:

PARALLEL/NOPARALLEL,
COMPRESS/NOCOMPRESS,
LOGGING/NOLOGGING.

Как и для таблиц, они фактически являются свойствами операций, а будучи указанными как «свойство» индекса определяют умолчательное поведение.

Свойство BUFFER_POOL является собственно свойством индекса.

5.2.2.2.1. Распараллеливание обработки индексов

Индекс, аналогично таблице, можно создать или перевести в режим параллельной обработки, и достичь более высокой скорости работы с ним:

```
ALTER INDEX pk_dept PARALLEL 4;
ALTER INDEX pk_dept NOPARALLEL;
```

Замечание. Реальное распараллеливание, тем не менее, имеет место *только* для индексов с *раздельным хранением* частей (*partitioned indexed*) и при некоторых дополнительных условиях. (Тем самым, все примеры в этом разделе чисто формальны!)

Для отдельной команды того же достигается с помощью подсказки PARALLEL_INDEX в предложении SQL (противоположная подсказка — NO_PARALLEL_INDEX):

```
ALTER TABLE dept    PARALLEL;
ALTER INDEX pk_dept PARALLEL;
SELECT /*+ PARALLEL_INDEX ( dept pk_dept ) */ deptno FROM dept ORDER BY
deptno;
```

Такую подсказку, для лучшего действия, часто употребляют совместно с INDEX_FFS, например:

```
SELECT   /*+ INDEX_FFS ( dept pk_dept ) PARALLEL_INDEX ( dept pk_dept ) */
         deptno
FROM     dept
ORDER BY deptno
;
```

5.2.2.2.2. Уплотненное хранение данных индекса

Начиная с версии 8 для древовидного индекса, подобно таблице (см. выше), можно указать особый режим уплотненного хранения. Пример:

```
ALTER INDEX pk_dept REBUILD COMPRESS;
```

Компактность обеспечивается СУБД на уровне блока за счет выноса одних и тех же повторяющихся значений (в том числе ROWID в неуникальном индексе) из области данных в справочную часть блока. Фактический выигрыш компактности зависит от количества одинаковых значений ROWID и ключей индексации, попавших в общий блок.

Для индексов указание COMPRESS может сопровождаться уточнением в виде целого числа, задающего количество ведущих столбцов в списке индексируемых.

Особенности:

- индекс должен быть изначально либо уплотненным, либо нет (то есть для уплотнения существующего индекса его придется перестроить);
- уплотнению подвержены только листьевые блоки;
- уплотненное хранение уникального одностолбцового индекса невозможно.

Последнее обстоятельство значительно снижает ценность уплотненного хранения данных индекса, так как оно становится неприменимым к индексам одностолбцовых первичных ключей — самой популярной их категории.

5.2.2.2.3. Отключение журнализации

Свойство NOLOGGING позволяет практически отменить журнализацию при построении индекса, что в способно заметно ускорить эту операцию. Примеры:

```
ALTER INDEX pk_dept NOLOGGING;
```

```
CREATE INDEX i_dept_loc ON dept ( loc ) NOLOGGING;
```

5.2.2.2.4. Прикрепление к указанной секции буфера блоков

Свойство BUFFER_POOL, указываемое во фразе STORAGE, позволяет сообщить СУБД, хотим ли мы помещать буферизуемые блоки индекса в секции буферной области DEFAULT, KEEP или RECYCLE (при желании их иметь последние две должны быть явно заведены). Пример:

```
ALTER INDEX pk_emp STORAGE ( BUFFER_POOL RECYCLE );
```

Выполеннное действие говорит о том, что мы намерены обращаться к индексу PK_EMP редко.

5.2.3. Некоторые специальные случаи конфигурирования хранения и использования таблиц и индексов

В Oracle имеется чрезвычайно много специальных случаев конфигурирования таблиц, индексов, а также запросов, позволяющих добиваться хорошего эффекта в одних ситуациях, однако часто за счет ухудшения характеристик использования в других. Полный набор возможностей см. в описании CREATE/ALTER TABLE … в документации. Описание некоторых возможностей приводится ниже.

5.2.3.1. Таблицы с раздельным хранением строк

Partitioned tables были введены в версии 8 в Enterprise Edition и позволяют хранить строки одной и той же таблицы в разных обособленых «разделах», в зависимости от значений в определенных полях (например, значений в поле «Год» строки). При выполнении запроса строки будут выбираться только из разделов, содержащих значения данных, требуемых в запросе, и таким образом количество обращений к БД может оказаться существенно сокращено. Основная область применения подобной организации хранения — таблицы с особенно большим количеством строк.

Технологически разделы со строками такой таблицы обладают значительной степенью самостоятельности: например, их можно по отдельности резервировать и восстанавливать, создавать, удалять, сливать, разбивать, располагать в разных табличных пространствах и так далее. Это добавляет потребительской выгоды к такой организации, сверх увеличения скорости доступа.

5.2.3.2. Индексно организованные таблицы

В версии 8 Oracle появилась возможность заведения таблиц, данные которых целиком размещаются в структуре индекса, вместе с ключом индексирования (фраза ORGANIZATION INDEX в CREATE TABLE).

Ограничения:

- Всегда должен иметься первичный ключ
- Отсутствует ROWID[-9]
- Индексы помимо «основного» должны отсутствовать[-9]
- На «широких» строках не дают выигрыша в поиске по индексу
- Некоторые ограничения по употреблению (например, в реплицировании данных)

[-9] в версии 9 это ограничение снято, однако «основной» индекс все равно будет более эффективнее и экономичнее.

5.2.3.3. Поразрядные индексы

Создаются указанием CREATE **BITMAP** INDEX .... В этом случае поисковая часть индекса формируется так же в виде дерева, как для B*-древовидного индекса, но только технически. Логически это дерево воплощает поразрядную карту признаков наличия в столбцах конкретных значений. Способен дать выигрыш в таблицах, где число разных значений индексированного столбца невелико («пол», «семейное положение» и т. п.).

5.2.3.4. Невосстанавливаемые операции

Начиная с версии 7.2 можно создавать индекс без использования записи в журнальный файл (в версии 8 — указание NOLOGGING, в версии 7 — UNRECOVERABLE в предложении CREATE INDEX).

Значительно ускоряет процедуру заведения данных.

Другой пример — операция TRUNCATE TABLE.

5.3. Настройка SQL-запросов

Более детально см. соответствующий раздел ниже. Здесь приводятся общие положения.

Два метода обработки запросов в Oracle:

- Метод простейшего доступа (rule-based). Основан на использовании фиксированной внутренней таблицы приоритетов метода доступа к данным.
- Метод минимизации общих затрат на обработку (cost-based). Основан на использовании статистики, имеющейся для объектов, фигурирующих в запросе.

Выбор одного или другого метода для обработки конкретного запроса может существенно изменить время выполнения («время работы Oracle», DB time).

Основные пути влияния на обработку запроса со стороны Oracle:

- Добавить индексы к таблицам
- Переписать запрос по-другому
- Указать оптимизатору запроса тот или иной псевдокомментарий
- Изменить некоторые параметры СУБД.

6. Администрирование доступа в Oracle

Выше обсуждались вопросы установки системы и «приведения в рабочее состояние». Здесь будут рассмотрены вопросы администрирования доступа, связанные с каждодневной эксплуатацией. Они касаются доступа

- к объектам, хранимым в БД
- к ресурсам СУБД

6.1. Политика безопасности

Безопасность данных в БД являет собой частный случай безопасности ИС, а последняя —безопасности вообще, регламентируемой законодательными основами межправительственного уровня, отдельных государств, ведомств, единиц самоуправления и отдельных организаций. Нормативно-методическими документами по обеспечению безопасности служат стандарты, рекомендации и указания разных уровней.

Ниже рассматриваются некоторые средства Oracle для обеспечения безопасного доступа к данным: в первую очередь конфиденциальности и отчасти целостности и доступности. Они затрагивают доступ к объектам, хранимым в БД и к ресурсам СУБД. Технически это средства:

- аутентификации,
- авторизации,
- контроля доступа,
- аудита (отдельная глава),
- шифрования.

Для умелого и действенного их употребления необходимы определенные организационно-методические усилия.

6.2. Основные средства администрирования доступа

Администрирование доступа к БД и к ресурсам СУБД обеспечивается в Oracle основополагающими понятиями «пользователь», «привилегия» и «роль».

6.2.1. Пользователи и схемы

Заведение пользователей обычно осуществляется «администратором», однако это может сделать любой пользователь с привилегией CREATE USER (см. ниже).

Создание, изменение свойств и удаление осуществляется командами:

{CREATE | ALTER | DROP} USER …

Пример:

```
CREATE USER yard IDENTIFIED BY pass;
```

По этой команде СУБД создаст в БД *пользователя* и одновременно автоматически — *схему* с тем же именем, к которой будет относить все объекты «этого пользователя», и только их. Схема будет удалена из состава БД также автоматически при удалении пользователя. Подобная связь этих двух понятий характерна для Oracle и может быть иной в других типах СУБД. Она значительно менее развита, чем в давно разработанной архитектуре ANSI/SPARC (см., например, *http://de.wikipedia.org/wiki/ANSI-SPARC-Architektur*).

Большинство объектов БД, например таблицы, индексы, процедуры, вида SYNONYM, обязаны находиться в одной и ровно одной схеме («принадлежать одному пользователю»). Однако небольшая часть объектов,

например вида PUBLIC SYNONYM, или же DIRECTORY, или роль, хранятся в БД вне всякой схемы и «не принадлежат» никакому конкретному «пользователю».

Пользователь, соединившийся из программы с СУБД, автоматически получает возможность работы с объектами «своей» схемы и с имеющимися внесхемными объектами.

До версии 11 Oracle автоматически приводила регистр указываемого пароля к заглавным буквам; в версии 11 значение TRUE параметра СУБД SEC_CASE_SENSITIVE_LOGON (умолчательное) обрабатывает указываемый пароль *без* предварительного преобразования.

Пример изменения свойств пользователя:

```
ALTER USER yard
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp
QUOTA 3M ON sysaux
QUOTA UNLIMITED ON users
;
```

Заметьте, что квота распространяется на «базовые» объекты хранения, имеющие самостоятельную структуру (сегмент) в табличном пространстве, таких как таблица, или же индекс. Создание, например, процедур, или же представлений не требует наличия у пользователя каких-либо квот. Значение QOUTA = 0 несет особый смысл: пользовательским объектам запрещается занимать новое место в табличном пространстве, фиксируя сложившееся status quo.

Сведения о пользователях в БД имеются в таблицах:

USER/ALL/DBA_USERS
USER/DBA_TS_QUOTAS

6.2.2. Привилегии

Привилегии *разрешают* пользователю выполнять действия с СУБД или в БД (то есть носят разрешающий характер). (Слово «привилегия» наврядли является удачным; более правильно в данном случае было бы говорить о «полномочиях»). Каждому пользователю можно выдать произвольное число привилегий (полномочий).

- *Системные привилегии* дают возможность осуществлять пользователю действия с СУБД или в БД общего характера. Примеры: CREATE SESSION, SELECT ANY TABLE, CREATE USER и другие. Выдаются предложением:

  GRANT *привилегия* TO {*пользователь* [WITH ADMIN OPTION] | *программная_единица*}

  Вариант выдачи привилегии программной единице (хранимые процедуры и функции, пакеты, типы) действует с версии 12.

- *Привилегии на работу с конкретными объектами в БД* дают возможность обращаться к конкретным объектам в БД (просматривать, изменять, вызывать). Выдаются предложением:

  GRANT *привилегия* ON *объект* TO *пользователь* [WITH GRANT OPTION]

Примеры выдачи системных и объектной привилегий:

```
CONNECT / AS SYSDBA
GRANT CREATE SESSION, SELECT ANY TABLE TO yard;
CONNECT scott/tiger
GRANT SELECT ON emp TO yard;
```

Команды GRANT в применении к некоторым категориям объектов сопровождаются дополнительными ключевыми словами, например:

```
GRANT READ ON DIRECTORY data_pump_dir TO yard;
```

Отбираются привилегии обоего рода предложением:

```
REVOKE привилегия [ON объект] FROM пользователь
```

Например:

```
REVOKE SELECT ANY TABLE FROM yard;
```

Более специфичные формы команды REVOKE приведены в документации по Oracle.

Некоторые основные виды привилегий на пользование объектами:

Вид привилегии	***Вид объекта***
SELECT	Таблица, представление данных, материализованное представление данных, датчик новых чисел
DELETE	Таблица, представление данных, материализованное представление данных
INSERT(*)	Таблица, представление данных, материализованное представление данных
UPDATE(*)	Таблица, представление данных, материализованное представление данных
ALTER	Таблица, представление данных
INDEX	Таблица, материализованное представление данных
REFERENCES(*)	Таблица
EXECUTE	Пакет, процедура, функция, библиотека, тип, оператор, тип индекса
READ (с версии 8)	Каталог ОС
WRITE (с версии 9)	Каталог ОС
...	...

(*) Привилегию можно формулировать с точностью до отдельных столбцов, а не только строк.

Некоторые общие правила:

- Хозяин существующего объекта *по определению* имеет все возможные привилегии на действия с объектом (даже если сам объект создавался другим пользователем).
- По условному соглашению именования обладатели системной привилегии с припиской ANY (например, CREATE ANY TABLE) получают возможность совершать соответствующие действия с объектами в любой схеме БД (в версиях до 9 — за исключением схемы SYS). Однако с версии 8 возможно исключение из этого правила: для дополнительной защиты введен параметр СУБД O7_DICTIONARY_ACCESSIBILITY, значение которого FALSE запретит обладателю привилегии SELECT ANY TABLE, отличному от SYS, видеть базовые таблицы словаря-справочника.
- Системная привилегия GRANT ANY OBJECT PRIVILEGE (появилась в версии 9) дает возможность ее обладателю выдать привилегию на использование объекта из чужой схемы; таким образом, например, пользователь SYS может выдать:

  ```
  GRANT SELECT ON scott.emp TO yard;
  ```

Список всех привилегий, имеющихся в конкретной версии Oracle, можно посмотреть, выдав:

```
SELECT name "Системные привилегии" FROM system_privilege_map;

SELECT name "Виды объектных привилегий" FROM table_privilege_map;
```

Список *действующих* во время текущего сеанса связи с СУБД системных привилегий (как переданных пользователю напрямую, так и через действующие роли) можно посмотреть в системной таблице SESSION_PRIVS, а посмотреть список привилегий, *выданных* пользователям непосредственно, можно в USER/DBA_SYS_PRIVS:

```
SELECT privilege FROM session_privs
```

```
;
SELECT privilege FROM dba_sys_privs WHERE grantee = 'SCOTT'
;
```

За счет ролей, о которых ниже, результат первого запроса может оказаться шире, чем второго.

Перечень имеющихся у пользователя объектных привилегий можно узнать из таблиц USER/ALL/DBA_TAB_PRIVS, например:

```
SELECT * FROM dba_tab_privs WHERE grantee = 'SCOTT'
;
```

6.2.2.1. Автоматическое распространение действия объектных привилегий

Предложение

GRANT SELECT ON *таблица* TO *пользователь*;

дает право на выборку из конкретной *таблицы* конкретному *пользователю*. Однако команда GRANT позволяет также дать право на выборку из любой таблицы, и дать право выборки из таблицы всем.
Предложение

GRANT **SELECT ANY TABLE** TO *пользователь*;

даст право выборки из любой *таблицы* в БД конкретному *пользователю* [-обладателю этой привилегии].
Предложение

GRANT SELECT ON *таблица* **TO PUBLIC**;

даст право на выборку из конкретной *таблицы* любому *пользователю*.

Аналогично расширительное действие устанавливается и для INSERT, UPDATE, DELETE, EXECUTE и ряда прочих привилегий.

6.2.2.2. Привилегии SYSDBA, SYSOPER и SYSxxx

Привилегии SYSDBA и SYSOPER имеют свою специфику. Их назначение — дать обладателю полномочия, затрагивающие взаимодействие с ОС. Для SYSDBA это:

- запуск и останов СУБД и БД (STARTUP, SHUTDOWN);
- выдача команд CREATE/ALTER DATABASE, CREATE SPFILE;
- выполнения действий резервного копирования и восстановления.

Для SYSOPER круг административных действий сужен; сама эта привилегия маловостребована, и Oracle в последних версиях оставляет ее для обратной совместимости.

Привилегии соответствуют ролям ОС: dba и oper в Unix и ORA_DBA и ORA_OPER в Windows. В Unix соответствующие роли требуется заводить при установке ПО Oracle, в Windows они создаются установщиком OUI автоматически.

При выдаче этих привилегий сведения о пользователе заносятся в файл *PWD.ORA*, что (а не информация в словаре-справочнике) и является признаком обладания. При пересоздании файла *PWD.ORA* программой *orapwd* эти сведения пропадут. Содержимое файла покажет V$PWFILE_USERS:

```
SELECT * FROM  v$pwfile_users;
```

Со временем появились дополнительные привилегии для специальных случаев администрирования:

- SYSASM (11+) для ASM
- SYSBACKUP (12+) для резервного копирования и восстановления
- SYSDG (12+) для резервного копирования и восстановления в пределах DataGuard
- SYSKM (12+) для TDE (Transparent Data Encryption) и Data Vault

Они были введены в целях рассредоточения полномочий; при желании к ним можно не прибегать, работая как SYSDBA. Их полномочия согласованы с группами ОС, создаваемыми аналогично группам для SYSDBA и SYSOPER.

До версии 9 обязательным обладателем привилегии SYSDBA был псевдопользователь INTERNAL, а начиная с версии 9 — пользователь SYS. Начиная с версии 9 SYS может подсоединяться к СУБД только с уточнением AS SYSDBA, если только не заменять умолчательное значение параметра СУБД O7_DICTIONARY_ACCESSIBILITY = FALSE (в версии 8, где он появился, было = TRUE). Другие обладатели привилегии SYSDBA могут не указывать AS SYSDBA при соединении с СУБД, но тогда в границах соединения эта привилегия действовать не будет.

Привилегии SYSDBA, SYSOPER и прочие SYSxxx во многом похожи на роль, или даже пользователя, и в связи с этим в документации по Oracle иногда наблюдается путаница определений.

6.2.3. Упражнения

Завести пользователя Oracle и снабдить его полномочиями подключения к СУБД, создания таблиц. Выдать пользователю квоту на использование табличного пространства USERS. Заблокировать подключение его к СУБД. Разблокировать. Навязать пользователю смену пароля.

6.2.4. Роли

Механизм групповой передачи привилегий пользователям. (Как видно из дальнейшего, смысл «роли» гораздо точнее определяется термином «группа», наподобие групп пользователей в ОС). Передача роли подразумевает следующее:

1) Роль была уже создана кем-то, имеющим привилегию CREATE ROLE
2) Созданной роли были приписаны необходимые привилегии пользования объектами БД
3) Роль приписывается пользователю другим пользователем, обладающим привилегией GRANT ANY ROLE или обладателем этой роли, получившим ее с указанием WITH ADMIN OPTION

Наполнение ролей привилегиями выполняется опять-таки командами GRANT и REVOKE:

GRANT/REVOKE ***привилегия***_*или_роль* [ON *объект*] TO/FROM ***имя_роли***,

а выдача или изъятие — командами:

GRANT/REVOKE ***имя_роли*** TO/FROM {***имя_пользователя*** | ***программная_единица***}

Наделение ролью программной единицы (хранимой процедуры и функции, пакета, типа) действует с версии 12.

Иногда Oracle способен выдавать какие-нибудь разрешения не только пользователям, но и ролям: например, разрешения обращений ко внешним ресурсам в системе безопасности Java VM, встроенной в СУБД, или же обращений к файлам и директориям репозитария в XML DB. В таких случаях бывают осмысленными «пустые» роли, *без* наполнения их привилегиями.

Таблицы с информацией о ролях:

DBA_ROLES	Все роли в базе данных
DBA_ROLE_PRIVS	Роли, приписанные пользователям и ролям
USER_ROLE_PRIVS	Роли, приписанные пользователю-автору запроса

ROLE_ROLE_PRIVS	Роли, приписанные ролям
ROLE_SYS_PRIVS	Системные привилегии, приписанные ролям
ROLE_TAB_PRIVS	Привилегии на пользование объектами, приписанные ролям
SESSION_ROLES	Действующие («включенные») роли для текущего сеанса
SYSAUTH$	Лес деревьев всех присвоенных ролей в системе

Просмотреть полный набор ролей, приписанных текущему пользователю, с учетом ролей, в составе других ролей, можно, обратившись к следующей виртуальной таблице (создается от имени SYS):

```
CREATE VIEW user_role_hierarchy
AS
SELECT su.name granted_role
FROM
  (
  SELECT * FROM sys.sysauth$
  CONNECT BY PRIOR privilege# = grantee#
  START WITH grantee# = UID
          OR grantee# = 1
  ) sa
, sys.user$ su
WHERE su.user# = sa.privilege#
UNION ALL SELECT 'PUBLIC' FROM DUAL
UNION ALL SELECT USER FROM DUAL
;
```

Чтобы она была видна всем пользователям, достаточно выдать:

```
GRANT SELECT ON user_role_hierarchy TO PUBLIC;

CREATE PUBLIC SYNONYM FOR user_role_hierarchy;
```

Теперь с ее помощью можно перечислить все объектные привилегии, приписанные пользователю:

```
SELECT DISTINCT privilege, owner, table_name
FROM   dba_tab_privs
WHERE  grantee IN ( SELECT * FROM user_role_hierarchy );
```

Аналогично все системные привилегии, приписанные текущему пользователю, можно получить, выдав:

```
SELECT DISTINCT privilege
FROM   dba_sys_privs
WHERE  grantee IN ( SELECT * FROM user_role_hierarchy );
```

Иногда в Oracle имеют смысл роли без назначенных им привилегий. Например, во встроенной в СУБД виртуальной машине Java получателем разрешения от системы безопасности Java может выступать как пользователь, так и роль. В этом случае практично выдать разрешение (системной процедурой) специально созданной «пустой» роли, а правами пользователей управлять командами SQL GRANT и REVOKE, приписывая нужных пользователей этой роли, или отписывая их. Это еще раз напоминает, что «роль» в Oracle по сути скорее является «группой пользователей».

6.2.4.1. Предопределенные роли

После создания БД в любой конфигурации всегда возникает некоторое число предопределенных ролей. В их число всегда входят роли CONNECT, RESOURCE, DBA, EXP_FULL_DATABASE, IMP_FULL_DATABASE. По своему поведению предопределенные роли ничем не отличаются от обычных, так как создаются предложениями CREATE ROLE, включенными в стандартный сценарий заведения БД.

Тем не менее, роли RESOURCE и DBA имеют особенность: при приписывании их пользователю последний автоматически получает вдобавок привилегию UNLIMITED TABLESPACE, а при отзыве этих ролей привилегия

UNLIMITED TABLESPACE так же автоматически изымается. Это единственное исключение из правила действия ролей в Oracle.

Методически пользоваться в промышленной БД предопределенными ролями для «собственных» пользователей не рекомендуется; лучше полагаться на собственноручно созданные. В тренировочной БД этим правилом можно пренебречь.

Помимо упомянутого побочного эффекта роли RESOURCE, могут возникать и другие. Например, в версии 10 содержание роли CONNECT было сведено только к привилегии CREATE SESSION, отчего разработчик, активно пользующийся этой ролью для создаваемых пользователей, может попасть в неприятность.

6.2.4.2. Роли и сеансы

Между ролями и привилегиями есть важная разница: первые, в отличие от вторых, составляют свойства не пользователя, а сеанса. Отличие проявляется в том, что «приписанные» пользователю роли можно во время каждого *конкретного* сеанса связи этого пользователя с СУБД активировать («включать») и отключать. Для этого используются команда SET ROLE, например:

```
SQL> CONNECT scott/tiger
Connected.
SQL> SET ROLE NONE;

Role set.

SQL> SET ROLE resource;

Role set.
```

Это же можно сделать с помощью двух подпрограмм пакета DBMS_SESSION, одна из которых позволяет, к тому же, узнать текущее состояние роли в сеансе:

```
SQL> SET SERVEROUTPUT ON
SQL> BEGIN
  2  IF dbms_session.is_role_enabled ( 'CONNECT' )
  3  THEN dbms_output.put_line ( 'CONNECT' ); END IF;
  4  END;
  5  /

PL/SQL procedure successfully completed.

SQL> EXECUTE dbms_session.set_role ( 'CONNECT' )

PL/SQL procedure successfully completed.

SQL> /
CONNECT

PL/SQL procedure successfully completed.
```

6.2.4.3. Динамическое переопределение ролей

Динамическое изменение наполнения роли сразу отражается на полномочиях ее обладателей. В этом легко убедиться на следующем примере.

Пусть пользователь YARD обладает единственной привилегией CREATE SESSION. Выполним в SQL*Plus следующие действия:

```
CONNECT / AS SYSDBA
```

```
CREATE ROLE tablecreator;

GRANT tablecreator TO yard;

CONNECT yard/pass

SELECT * FROM session_privs;

HOST sqlplus "/ AS SYSDBA"

GRANT CREATE TABLE TO tablecreator;

EXIT

/
```

(Мы переключились на *другой сеанс*, от имени SYS приписали роли TABLECREATOR привилегию CREATE TABLE, вернулись в *прежний* сеанс и «старым» запросом снова посмотрели его полномочия).

6.2.4.4. Защищенные роли как средство дополнительной защиты данных

В качестве одной из своих функций роли в Oracle позволяют организовать второй эшелон парольной защиты от несанкционированного доступа. Это возможно в виду того, что (а) роли во время работы сеанса связи с СУБД можно «включать» и «отключать», и (б) роли можно защитить паролем.

Продолжим пример выше:

```
CONNECT / AS SYSDBA

ALTER ROLE tablecreator IDENTIFIED BY say123;
```

Проверка:

```
SQL> CONNECT yard/pass
Connected.
SQL> SELECT * FROM session_roles;

ROLE
------------------------------
TABLECREATOR
```

Переведем роль (только ее) в изначально отключенное состояние:

```
CONNECT / AS SYSDBA

ALTER USER yard DEFAULT ROLE ALL EXCEPT tablecreator;
```

(Конструкция DEFAULT ROLE допускает указание также просто ALL, NONE или же явного перечисления ролей, которые мы хотим сделать изначально активными для всех сеансов пользователя).

Снова проверка:

```
SQL> CONNECT yard/pass
Connected.
SQL> SELECT * FROM session_roles;

no rows selected

SQL> SET ROLE tablecreator IDENTIFIED BY say123;
```

```
Role set.

SQL> SELECT * FROM session_roles;

ROLE
------------------------------
TABLECREATOR
```

Теперь для возможности реально создавать таблицы пользователь YARD обязан не только указать собственный пароль при подключении к СУБД, но указать еще и пароль для активации роли.

Фирма Oracle рекомендует создавать для отдельных видов приложений отдельные роли (возможно, составные) и распоряжаться ими способом, указанным выше. Если приложение будет активизировать роль самостоятельно, санкционирование реальной работы приложения можно распределить на двух человек вместо одного: один будет осведомлен только о *пароле пользователя Oracle*, а второй — только о *пароле роли*.

6.2.4.5. Роли, контролируемые извне СУБД Oracle

Oracle позволяет переключить СУБД на внешнее управление ролями, когда активность роли в сеансе будет задаваться не командами СУБД SET ROLE, а извне. Эта возможность реализуется следующими конструкциями создания роли:

- CREATE/ALTER ROLE *имя_роли* **IDENTIFIED EXTERNALLY**
- CREATE/ALTER *имя_роли* **IDENTIFIED GLOBALLY**

В первом случае наличие роли у пользователя Oracle будет определяться средствами операционной системы, а во втором — службой каталогов.

Эти возможности перекливаются с особыми способами идентификации пользователей, о чем речь пойдет ниже.

6.3. Опосредованный доступ к данным в таблицах

Базовый механизм привилегий (полномочий) позволяет осуществлять более сложные схемы доступа к объектам БД, нежели чем прямое разрешение конкретных операций DML с чужими объектами. Так, пользователю можно не давать доступ к основным таблицам чужой схемы, но дать ему доступ к представлениям (views), или же к подпрограммам, обращающимся к данным чужих таблиц.

6.4. Ограничение доступа к отдельным частям таблицы

Ограничить разным пользователям доступ (видимость или возможность изменений) к данным конкретных таблиц в Oracle можно с помощью аппарата view или процедурной логикой. Однако иногда более функциональными или эффективными могут оказаться два других специальных способа:

- организация «виртуальных частных баз данных» (virtual private databases)
- использование меток доступа (Label Security) к отдельным строкам таблицы

6.4.1. Организация «виртуальной частной БД»

Достигается использования системного пакета DBMS_RLS (маркетинговое название Oracle — «виртуальные частные базы данных», virtual private databases, а техническое название — «детальный контроль доступа», fine grained access).

С помощью подпрограмм пакета DBMS_RLS можно определить специальную функцию P, играющую роль предиката отбора, и связать ее с конкретной таблицей, конкретным пользователем и конкретным действием (SELECT, UPDATE, INSERT или DELETE). Если, к примеру, речь идет о таблице EMP, то всякое обращение этого пользователя к EMP будет автоматически заменено системой на SELECT * FROM emp WHERE P, подобно как если упомянутая в основном запросе таблица EMP была бы виртуальной.

В версии 10 пакет DBMS_RLS расширен возможностью аналогично ограничивать видимость значений в определенных *столбцах* определенных *строк* таблицы.

6.4.2. Использование техники «меток доступа»

Эта возможность появилась в версии 9 под названием Label Security. Она реализует подход, известный в ИТ под названием «мандатного регулирования доступа» (mandatory access control в контексте выполнения требованиям международного стандарта сертификации компьютерной безопасности Common Criteria), на уровне отдельных строк, делая доступной или невозможной работу с ними разным категориям пользователей.

Каждая строка нужной таблицы помечается специальной меткой, значение которой можно менять. Для пользователей создаются группы доступа, задающие семантику меток, наподобие грифам «ДСП», «секретно» и пр. на документах и категориям лиц, для которых разрешен доступ к различным документам.

6.5. Защита сведений в БД внешними средствами

Собственные («внутренние») средства разграничения доступа в Oracle не всегда могут удовлетворить потребностям приложения. В отдельных случаях разработчики вынуждены прибегать к внешним средствам.

6.5.1. Шифрование данных

Несмотря на проработанные механизмы разграничения доступа к данным в Oracle, пользователь SYS обладает достаточными полномочиями, чтобы суметь обратиться к любым данным любого пользователя в БД («суперпользователь»). Для того, чтобы обезопаситься от этого, а также случайного доступа со стороны других пользователей, особо важные данные можно перед помещением в базу шифровать.

Начиная с версии 8.1.6 с Oracle поставляется пакет для шифрования и расшифровки методом DES под названием DBMS_OBFUSCATION_TOOLKIT. Процедурой DESENCRYPT этого пакета можно с помощью ключа зашифровать текстовую строку, а процедурой DESDECRYPT расшифровать.

В версии 10 в состав системных пакетов включен (в перспективе — на замену DBMS_OBFUSCATION_TOOLKIT) более функциональный пакет DBMS_CRYPTO, позволяющий шифровать данные других типов (не RAW и VARCHAR2, а RAW, CLOB и BLOB) и другими алгоритмами (не только DES, 3DES, но еще и AES, и RC4, и 3DES_2KEY плюс алгоритмы хеширования плюс параметризация этих алгоритмов).

6.5.2. «Шифрование» исходных текстов программных элементов в БД

Тексты подпрограмм, создаваемых предложениями CREATE PROCEDURE, CREATE FUNCTION и CREATE PACKAGE BODY, запоминаются в словаре-справочнике и также доступны для просмотра пользователем SYS. Однако есть возможность передварительно их преобразовать (фактически — частично оттранслировать), сделав недоступными для осмысленного чтения из БД. Для этого используется программа *wrap*, входящая в состав ПО Oracle.

Пусть файл *myproc.sql* содержит текст создания процедуры A (CREATE PROCEDURE a …). Следующая команда создаст файл *myproc.plb* с «закодированным» (частично оттранслированным) текстом заведения процедуры:

```
SQL> HOST wrap iname=myproc.sql
```

После этого завести процедуру в БД через SQL*Plus можно будет как обычно:

```
SQL> @myproc.plb
```

Этот способ, тем не менее, оставляет видимыми константы программы, так что при необходимости их придется разбивать на части в тексте подпрограммы и собирать в переменные при использовании. Кроме того, он не годится для триггерных процедур.

6.6. Подключение к СУБД

Стандартная схема подключения к СУБД подразумевает идентификацию (отождествление личности) именем пользователя с указанием пароля. Современные версии Oracle такую систему идентификации называют локальной (local), и поддерживают кроме нее другие: внешнюю (external) и глобальную (global). Их можно выбирать в момент создания пользователя, либо задавать позже:

CREATE/ALTER USER … **IDENTIFIED EXTERNALLY** ...

Для подключения к СУБД такой пользователь не указывает ни локального имени в БД, ни пароля, так как считается достаточным, что он указал имя и пароль, входя в ОС. Такой способ подключения позволяет опереться на средства внешней защиты (типа Kerberos, и т.д.) со всеми их достоинствами. Но если средства внешней защиты не используются, считается, что в ОС Windows этой возможности сопутствует повышенный риск несанкционированного доступа.

CREATE/ALTER USER … **IDENTIFIED GLOBALLY** …

Создаваемая схема подразумевает подключение через сервер каталогов LDAP.

Пользователь SYS имеет возможность «доверительного подключения» к СУБД, не указывая ни своего имени, ни пароля. Для того, чтобы это было действовало, требуется:

а) выполнять подключение к СУБД, работая в ОС от имени пользователя, входящего в группу ORA_DBA (Windows; группа создается автоматически установщиком OUI) или dba (Unix; группа заводится вручную в процессе установки ПО Oracle)

б) дополнительно *в Windows*: иметь одновременно на клиенте и на сервере в файле *%ORACLE_HOME%\network\admin\sqlnet.ora* (или же в *%TNS_ADMIN%\sqlnet.ora*) значение параметра SQLNET.AUTHENTICATION_SERVICES = (NTS)

Реализация этой возможности выполнена в Oracle не совсем аккуратно, что может проявиться, например, при использовании программы *oradim.exe* (имеющей собственные погрешности).

6.6.1. Пример организации внешней (EXTERNAL) идентификации в ОС Windows

Для того, чтобы пример ниже проработал в версиях 7.3.4 — 8.0, в реестре в разделе *HKEY_LOCAL_MACHINE\software\oracle\HOMEn* должно стоять значение ключа OSAUTH_PREFIX_DOMAIN = TRUE (начиная с версии 8.1 это значение принято умолчательным).

1. Проверить наличие доверительной аутентификации (или выставить ее): параметр SQLNET.AUTHENTICATION_SERVICES = (NTS) в *sqlnet.ora* (см. выше)
2. Рекомендуется установить в параметр СУБД OS_AUTHENT_PREFIX = '' (пусто) и перезапустить СУБД по измененному файлу *INIT.ORA* или же *SPFILE.ORA* (так удобнее; умолчательное значение OS_AUTHENT_PREFIX = 'OPS$' существует для обратной совместимости).
   Если требуется использовать этот способ для подключения с удаленной машины, нужно вдобавок выставить параметр СУБД REMOTE_OS_AUTHENT = TRUE. Однако этим параметром нужно пользоваться с осторожностью и, например, ограничивать допустимые адреса IP клиентов.
3. Узнать свое доменное имя в ОС, например из столбца OSUSER таблицы V$SESSION. Например, это WINSERV\student.
4. От имени SYS или SYSTEM завести пользователя:

```
CREATE USER "WINSERV\STUDENT" IDENTIFIED EXTERNALLY;
```

```
GRANT CREATE SESSION TO "WINSERV\STUDENT";
```
(В двойных кавычках нужно указать именно *заглавные* буквы).

Проверка:

```
SQL> CONNECT /
Connected.
SQL> SHOW USER
USER is "WINSERV\STUDENT"
CONNECT /@teacher
SQL> SHOW USER
USER is "WINSERV\STUDENT"
```

Примечание: Если имя пользователя ОС локальное, а не доменное (в нашем случае могло бы быть просто "STUDENT"), в реестре нужно указать OSAUTH_PREFIX_DOMAIN = FALSE.

6.7. Профили пользователей

Профили появились в версии 7 Oracle в качестве ограничителей использования системных ресурсов отдельными пользователями. При создании БД автоматически заводится профиль DEFAULT, изначально ничего не ограничивающий и назначаемый пользователям по умолчанию. Создание, изменение и удаление профилей выполняется командами {CREATE | ALTER | DROP} PROFILE …. Назначение профиля пользователю делается командами {CREATE | ALTER} USER ….

В рамках конкретного профиля указываются значения для параметров двух различных по характеру групп.

6.7.1. Ограничения расходования ресурсов СУБД

Можно задать 8 параметров, регулирующих расходование индивидуальных ресурсов СУБД, и один суммарный параметр. Можно указывать целое значение, UNLIMITED или DEFAULT (технически последние два значения хранятся в БД в виде особых целых значений).

Параметры позволяют регулировать использование в СУБД:

- Времени работы процессора в пределах сеанса
- Времени работы процессора в пределах обработки одного запроса
- Общего времени соединения
- Максимальный интервал простоя между выдачей двух последовательных обращений к БД
- Максимальное число соединений от имени пользователя
- Максимально допустимое за сеанс число логических чтений диска
- Максимально допустимое за одно обращение число логических чтений диска
- Общий размер личной памяти в SGA за сеанс работы
- Количество дней подряд, когда пользователь не соединялся с СУБД; после этого вход автоматически блокируется[12.2-)]
- Взвешенную сумму лимитирующих параметров использования ресурсов

[12.2-)] С версии 12.2.

6.7.2. Контроль за использованием паролей

Параметры позволяют регулировать использование пользователями паролей:

- Срок действия пароля
  - Число дней, в течение которых паролем все-таки еще можно будет пользоваться, хотя срок его уже истек (будет выдаваться сообщение)
- Минимальное число других паролей, которые следует употребить до повторения прежнего
  - Срок, который должен пройти до возможности использования прежнего пароля

- Максимально допустимое число попыток подряд указать неправильный пароль, после чего вход в систему пользователю будет заблокирован
  - Срок, на который вход в систему будет заблокирован после превышения допустимого числа неудачных попыток
- Имя функции на PL/SQL, которая будет запускаться при установке пароля.

Простой пример создания собственной функции проверки пароля:

```
CONNECT / AS SYSDBA

CREATE OR REPLACE FUNCTION password_verify (
     username     VARCHAR2
   , password     VARCHAR2
   , old_password VARCHAR2
)
RETURN BOOLEAN
IS
BEGIN
IF LENGTH ( password ) < 12 -- рекомендуется использовать более 12 символов
THEN
   RAISE_APPLICATION_ERROR ( -20010, 'Password less than 12 characters' );
END IF;

RETURN ( TRUE );
END;
/
```

Другой пример можно найти в *%ORACLE_HOME%\rdbms\admin\utlpwdmg.sql.*

6.7.3. Включение контроля ресурсов

Ограничения расходования ресурсов СУБД, задаваемые профилями, начинают действовать после установки параметра СУБД RESOURCE_LIMIT в TRUE. Параметр динамический, так что установить его значение можно без перезапуска СУБД, выдав `ALTER SYSTEM SET RESOURCE_LIMIT = TRUE`.

Контроль за использованием паролей, будучи установлен в профиле, действует безотносительно к значению параметра RESOURCE_LIMIT.

Просмотреть перечень профилей в БД можно в таблице DBA_PROFILES. Приписанные пользователям профили представлены в столбце DBA_USERS.PROFILE. Список действующих *ресурсных ограничений* сеанса пользователя можно посмотреть в USER_RESOURCE_LIMITS. Список действующих *парольных параметров* для сеанса можно посмотреть в таблице USER_PASSWORD_LIMITS. При включенных парольных параметрах история смены паролей отображается (в виде сверток, аналогично столбцу DBA_USERS.PASSWORD) в SYS.USER_HISTORY$.

6.7.4. Упражнение

Создать профиль, ограничивающий длительность выполнения SQL-запроса 5-ю минутами. Присвоить профиль пользователю. Задать в профиле срок действия пароля. Проверить наличие профильных ограничений (таблицы USER_RESOURCE_LIMITS и USER_PASSWORD_LIMITS) и понаблюдать их действие.

6.8. Распределение ресурсов СУБД и БД между сеансам

В версии 8.1 Enterprise Edition появиласть возможность сформулировать правила выделения разным сеансам ресурсов Oracle, основанные на приоритетности, учета фактической нагрузки и прочих факторах («диспетчер» *resource manager*). Важнейшим примером такого ресурса является гарантированная доля процессорного времени СУБД. Технология формулирования такого распределения основывается на

использовании системных пакетов DBMS_RESOURCE_MANAGER_PRIVS и DBMS_RESOURCE_MANAGER.

В версии 9 появилась возможность автоматического перевода сеанса в группу с иным приоритетом, если тот выполняет чрезмерно ресурсозатратные операции. Другие возможности раздатчика ресурсов расширялись и совершенствовались в версиях 10+.

6.9. Некоторые типичные просчеты в осуществлении ограничения доступа и рекомендации

- *Администратор забыл сменить пароли по умолчанию для SYS, SYSTEM* и ряда других пользователей. Эти пароли общеизвестны.
- *Пользователь использует один и тот же пароль во всех базах данных.*
- *Администратор забыл о существовании SCOTT/TIGER*. Все права, созданные как PUBLIC, действуют для SCOTT, умолчательныый пароль которого известен всем.
- *Администратор указал переменной СУБД UTL_FILE_DIR излишне пространный список каталогов*. Параметр СУБД UTL_FILE_DIR задает перечень каталогов ОС, куда может писать и откуда может читать файлы встроенный пакет UTL_FILE. Следовательно туда может обратиться *любая* подпрограмма на PL/SQL. Начиная с версии 9 рекомендуется при работе с пакетом UTL_FILE регулировать доступ к файлам ОС с помощью «каталогов» (DIRECTORY), что более избирательно и менее опасно.
- *Средствами ОС следует защитить от несанкционированного доступа файлы PWD.ORA и sqlnet.ora*. Используя эти файлы, можно подключиться к БД под именем SYS, не владея действующим паролем.

Общее правило гласит: всякая привилегия, выданная пользователю, не снижает риска несанкционированного доступа. Однако ряд привилегий требует повышенного внимания, например привилегии со словом ANY, ALTER USER, CREATE SESSION, EXEMPT ACCESS POLICY и другие.

Некоторые советы по ограничиванию доступа в имеющейся БД выводит OEM на странице Home.

7. Аудит

Понятие «аудит» пришло в тематику БД из счетоводства («бухгалтерского учета»), где оно обозначает проверку соответствия действительного состояния дел в организации отмеченному в бухгалтерской отчетности (от латинского auditus, оглашение).

Средства аудита в БД имеют целью слежение за действиями пользователей, представляющими интерес. Набор таких средств в Oracle многообразен и позволяет причислить к нему следующее:

- «системный аудит», основной для Oracle
- «подробный аудит», дополнительно к системному, реализуемый через API
- триггерные процедуры
- анализ журнальных файлов

Далее эти средства рассматриваются подробнее.

Средства аудита служат *задачам* аудита, аналогичным принятым в счетоводстве. Для БД их можно обозначить как выявление нарушений употребления прав доступа к данным и СУБД. Таким образом, аудит требует от администратора системного подхода и следования определенной политике, не ограниченной рамками только БД.

7.1. Виды действий для отслеживания системным аудитом Oracle

«Системный аудит» осуществляется кодом, специально встроенным в ядро СУБД Oracle. Он позволяет следить за следующими видами действий пользователей:

- *Действия с конкретными объектами схемы*. Oracle позволяет протоколировать обращения пользователей к конкретным объектам, существующим в БД.
- *Выдача определенных запросов на SQL*. Oracle позволяет следить за выдачей определенных предложений SQL.
- *Действия, регламентируемые системными привилегиями*. Oracle позволяет протоколировать осуществление действий из числа регулируемых системными привилегиями.

Объем собираемых сведений может уточняться конкретными заданиями на аудит следующим образом:

- BY *имя_пользователя* — при аудите SQL-предложений и системных привилегий можно указать, действия какого пользователя нужно подвергнуть аудит-проверке. Если специального указания не сделать, аудит будет выполняться для всех пользователей.
- WHENEVER SUCCESSFUL/WHENEVER NOT SUCCESSFUL — для всех трех видов аудита можно уточнить характер завершения операции, за которой требуется следить (успешное осуществление либо безуспешная попытка; последнее обычно возникает, когда пользователь не имеет достаточно полномочий).
- BY SESSION/BY ACCESS — для всех трех видов аудита можно указывать степень обобщенности регистрируемой информации (либо одна регистрирующая запись за сеанс, независимо от того, сколько раз за это время осуществлялось действие, либо отдельная запись на каждое действие).
- IN SESSION CURRENT[11-) — будут отслеживаться только действия в границах текущего сеанса.

[11-) С версии 11.

7.2. Общее разрешение на сбор СУБД информации о действиях пользователей

Изначально режим аудита отключен. Чтобы его активизировать, нужно установить в параметр СУБД AUDIT_TRAIL задающий направление для сбора аудит-информации:

a) AUDIT_TRAIL = DB — записи осуществляемого аудита заносятся в БД Oracle (таблица AUD$ в схеме SYS или же SYSTEM)
b) AUDIT_TRAIL = DB_EXTENDED[10-) — тоже, что при AUDIT_TRAIL = DB, но сохраняются также дополнительные сведения (например, текст запроса на SQL)

c) AUDIT_TRAIL = OS — записи аудита заносятся в журнал ОС или в специальный файл
d) AUDIT_TRAIL = XML[10-) — записи заносятся в специальный файл в формате XML
e) AUDIT_TRAIL = XML_EXTENDED[10-) — записи заносятся в специальный файл в формате XML с дополнительными сведениями
f) AUDIT_TRAIL = NONE — аудит выключен
g) AUDIT_TRAIL = TRUE или FALSE[<<].

[10-) Значение параметра появилось в версии 10.

[<<] Значения продолжают поддерживаться для совместимости с прежними версиями; то же, что DB и NONE.

С версии 9 параметр СУБД AUDIT_SYS_OPERATIONS позволяет включить слежение за действиями пользователя SYS.

С версии 10 параметр AUDIT_FILE_DEST указывает на размещение файла с данными аудита, когда они помещаются вовне БД, или если это данные об отслеживаемых действиях пользователя SYS.

После установки параметра AUDIT_TRAIL выдаваемые конкретные предложения AUDIT начинают иметь силу.

7.3. Примеры выдачи конкретных заданий на аудит

Аудит действий с объектом схемы БД (выборки данных из таблицы):

```
AUDIT SELECT ON scott.emp WHENEVER SUCCESSFUL;
```

Аудит выдачи команд SQL (создания таблиц):

```
AUDIT CREATE TABLE BY scott;
```

Аудит употребления системной привилегии (соединения с СУБД):

```
AUDIT CONNECT, DBA BY ACCESS WHENEVER SUCCESSFUL;
```

Пример завершения аудита:

```
NOAUDIT CONNECT WHENEVER SUCCESSFUL;
```

7.4. Примеры выполнения аудита и просмотра результатов

Единственной базовой таблицей, собирающей информацию о зафиксированных в результате системного аудита событиях, является AUD$. В более удобном виде данные из AUD$ можно посмотреть в (виртуальных) таблицах с шаблонами имен DBA_AUDIT_% и USER_AUDIT_%; наиболее общая из таблиц — DBA_AUDIT_TRAIL.

Упражнение. Выполнить следующие действия.

(1) Выдать команду аудита подключений к системе:

```
AUDIT CREATE SESSION;
```

(2) Сделать несколько удачных и неудачных попыток подключения к Oracle (в том числе под несуществующими именами. Снова подключиться как SYS.

(3) Выдать:

```
COLUMN username    FORMAT A15
COLUMN terminal    FORMAT A6
COLUMN timestamp   FORMAT A15
COLUMN logoff_time FORMAT A15
COLUMN action_name FORMAT A8
COLUMN returncode  FORMAT 9999
```

```
SELECT
  username
, terminal
, action_name
, TO_CHAR ( timestamp,'YYYYMMDD:HH24MISS' ) timestamp
, TO_CHAR ( logoff_time,'YYYYMMDD:HH24MISS' ) logoff_time
, returncode
FROM dba_audit_session
;
```

Обратите внимание на зафиксированные попытки войти в систему под несуществовавшими в тот момент именами. (Для анализа подобных сведений учтите, что таблица DBA_USERS выдает сведения о пользователях в данный момент времени. Тем не менее средства flashback быстрого чтения прежних данных начиная с версимй 9 позволяют построить адекватный запрос).

(4) Построить следующие запросы:
- выдать сведения о попытках подключения к СУБД с несуществующими в БД именами;
- выдать сведения о подключениях к СУБД в заданные интервалы времени;
- выдать сведения о пользователях, подключавшихся с разных терминалов (т.е. более чем с одного) и о терминалах, с которых подключалось более одного пользователя.

Упражнение. Выполнить следующие действия.

(1) Выдать команду аудита команды CREATE TABLE.

(2) Создать несколько таблиц от имени разных пользователей.

(3) Выдать:

```
COLUMN username   FORMAT A8
COLUMN priv_used  FORMAT A16
COLUMN obj_name   FORMAT A22
COLUMN timestamp  FORMAT A17
COLUMN returncode FORMAT 9999
SELECT
  username
, priv_used
, obj_name
, TO_CHAR ( timestamp, 'DD-MON-YYYY HH24:MI' ) timestamp
, returncode
FROM dba_audit_trail
WHERE priv_used IS NOT NULL
  AND priv_used <> 'CREATE SESSION'
;
```

Для получения сведений о включенном аудите при заведении БД автоматически создается ряд представлений данных (см. сценарий *%ORACLE_HOME%\rdbms\admin\cataudit.sql*); например следующие:

AUDIT_ACTIONS	Перечень доступных в системе действий для аудита
DBA_**OBJ**_AUDIT_OPTS	Признаки включения аудита для таблиц и представлений
DBA_**PRIV**_AUDIT_OPTS	Признаки включения аудита для системных привилегий (для всех пользователей)
DBA_**STMT**_AUDIT_OPTS	Признаки включения аудита для SQL-предложений (для всех пользователей)

Упражнение. Узнать из таблиц DBA_PRIV_AUDIT_OPTS и DBA_STMT_AUDIT_OPTS о включенном аудите системных действий и об аудите выдачи команд SQL.

7.5. Пример рекомендаций по осуществлению политики аудита

Собирать сведения обо *всех* осуществляемых действиях в БД очень накладно. Ниже приводятся некоторые рекомендации по выбору конкретной политики аудита.

- Использовать AUDIT CONNECT для сбора информации о пользователях, как часто они соединяются с БД и насколько много выполняют операции ввода/вывода.
- Использовать AUDIT DBA для регистрации попыток использования привилегий DBA.
- Периодически чистить таблицу аудита, сохраняя при необходимости обобщенную информацию в других таблицах.

7.6. Удаление записей аудита

Одного только включения слежения за действиями пользователей на деле недостаточно. Приходится позаботиться о технике вычищения старых записей, причем придумать, отладить и поставить на автоматическое выполнение соответствующий процесс следует с самого начала. Ниже приводятся некоторые элементы такого процесса в том виде, как они могли бы выглядеть при сборе записей аудита в БД. Более простое решение, состоящее в регулярной чистке строк таблицы AUD$ по прошествии положенного времени после их создания, очевидно, и не требует пояснений.

Создание таблицы для сбора обобщенной ежедневной статистики о сеансах связи с СУБД:

```
CREATE TABLE system.dba_audit_session_daily (
  os_username   VARCHAR2 ( 255 )
, username      VARCHAR2 ( 30 )
, userhost      VARCHAR2 ( 255 )
, terminal      VARCHAR2 ( 255 )
, timestamp     DATE
, sessions      NUMBER
, elapse_time   NUMBER
, logoff_lread  NUMBER
, logoff_pread  NUMBER
, logoff_lwrite NUMBER
)
   TABLESPACE sysaux
   STORAGE ( PCTINCREASE 0 )
;
CREATE INDEX system.dba_audit_session_daily_idx
ON system.dba_audit_session_daily ( timestamp )
   TABLESPACE sysaux
   STORAGE ( PCTINCREASE 0 )
;
```

Сбор обобщенной ежедневной статистики о сеансах связи:

```
INSERT INTO system.dba_audit_session_daily (
       os_username
     , username
     , userhost
     , terminal
     , timestamp
     , sessions
     , elapse_time
     , logoff_lread
     , logoff_pread
     , logoff_lwrite )
SELECT os_username
     , username
     , userhost
```

```
     , terminal
     , TRUNC ( timestamp )
     , COUNT ( * )
     , SUM ( logoff_time - timestamp )
     , SUM ( logoff_lread )
     , SUM ( logoff_pread )
     , SUM ( logoff_lwrite )
FROM
     dba_audit_session
WHERE
     action_name IN ( 'LOGOFF', 'LOGOFF BY CLEANUP' )
 AND logoff_time < TRUNC ( SYSDATE )
GROUP BY os_username
       , username
       , userhost
       , terminal
       , TRUNC ( timestamp )
-- включается только информация о завершенных сеансах работы
;
```

Чистка данных о сеансах связи из таблицы аудита:

```
DELETE
FROM  aud$ a
WHERE logoff$time < TRUNC ( SYSDATE )
  AND action BETWEEN 101 AND 102
  AND EXISTS
      ( SELECT 1
        FROM   system.dba_audit_session_daily d
        WHERE  TRUNC ( a.timestamp ) = d.timestamp
      )
;
```

Номера действий 101 и 102 (из AUD$) соответствуют 'LOGOFF' и 'LOGOFF BY CLEANUP' из DBA_AUDIT_SESSION в предыдущем запросе.

С версии 11 подобную работу можно выполнять средствами пакета DBMS_AUDIT_MGMT.

7.7. Выборочный аудит доступа к таблицам

Технологически осуществляется с помощью подпрограмм пакета DBMS_FGA. Для отслеживания обращений к *отдельным строкам* таблиц (для осуществления *детального*, или *подробного аудита*, fine grain auditing, FGA) требуется создать *политику аудита*, связав ее с нужной таблицей нужной схемы. С этой же политикой можно дополнительно связать условия для занесения учетной записи и процедуру-обработчик, которая будет автоматически срабатывать по завершению пользовательской SQL-операции.

7.7.1. Простой пример

Построим простой пример. От имени SYS выдадим:

```
BEGIN
DBMS_FGA.ADD_POLICY
(
     policy_name     => 'SALARY_OF_DEPT10_AUDIT'
   , object_schema   => 'SCOTT'
   , object_name     => 'EMP'
   , audit_condition => 'DEPTNO = 10'    -- необязательный параметр
   , audit_column    => 'SAL'            -- необязательный параметр
```

```
                                -- возможны другие необязательные параметры
);
END;
/
```

Последние два параметра необязательны. Но раз они указаны, запись аудита будет создаваться, если SQL-предложение обращается к столбцу SAL и к строкам о сотрудниках 10-го отдела из таблицы SCOTT.EMP.

Сигнальные записи помещаются в таблицу SYS.FGA_LOG$, но бывает удобнее обращаться к DBA_FGA_AUDIT_TRAIL, например:

```
COLUMN sql_text    FORMAT A30 WORD
COLUMN object_name FORMAT A30
COLUMN os_user     FORMAT A15
COLUMN db_user     FORMAT A10

SELECT os_user
     , db_user
     , TO_CHAR ( timestamp, 'HH24:MI:SS' )
     , policy_name
     , object_schema
     , object_name
     , sql_text
FROM dba_fga_audit_trail
;
```

Упражнение. Создать политику аудита и выдать от имени SCOTT:

```
SELECT ename       FROM emp WHERE empno = 7499;
SELECT sal         FROM emp WHERE empno = 7499;
SELECT ename       FROM emp WHERE empno = 7934;
SELECT sal         FROM emp WHERE empno = 7934;
SELECT *           FROM emp;
SELECT COUNT ( * ) FROM emp;
```

Посмотреть образовавшиеся сигнальные записи и объяснить результат.

Наличие и свойства имеющихся в БД политик подробного аудита можно наблюдать в таблицах:

{DBA | ALL | USER}_AUDIT_POLICIES
{DBA | ALL | USER}_AUDIT_POLICY_COLUMNS[10-)
[10-) Начиная с версии 10.

Упражнение. Убедиться в наличии в БД политики SALARY_OF_DEPT10_AUDIT.

7.7.2. Более сложный пример

Другие возможности пакета DBMS_FGA позволяют связать с политикой выборочного аудита процедуру, которая будет запускаться автоматически при регистрации события, или же создавать разные политики для всех четырех операций DML и QL с данными.

В этом примере задействована процедура-обработчик события. По организационным причинам припишем ее отдельной схеме (формально этого делать не обязательно).

```
CREATE USER fga_admin IDENTIFIED BY VALUES 'NoNeedToConnect';

GRANT  AUDIT ANY
     , AUDIT SYSTEM
     , CREATE PROCEDURE
     , EXECUTE ANY PROCEDURE
```

```
     , CREATE SESSION
     , EXECUTE_CATALOG_ROLE
TO fga_admin
;

ALTER USER fga_admin DEFAULT ROLE EXECUTE_CATALOG_ROLE;
```

Процедура должна быть оформлена по следующему правилу:

```
CREATE OR REPLACE PROCEDURE fga_admin.fga_alert_action (
   object_schema VARCHAR2
 , object_name   VARCHAR2
 , policy_name   VARCHAR2
) AS
BEGIN
NULL; -- используем имена схемы, таблицы и политики для написания кода выполняемого действия
END;
/
```

Пример построения политики аудита с автоматическим выполнением запрограммированного действия по факту возникновения события:

```
BEGIN
DBMS_FGA.DROP_POLICY (
     policy_name   => 'SALARY_OF_DEPT10_AUDIT'
   , object_schema => 'SCOTT'
   , object_name   => 'EMP'
);

DBMS_FGA.ADD_POLICY (
     policy_name     => 'SALARY_OF_DEPT10_AUDIT'
   , object_schema   => 'SCOTT'
   , object_name     => 'EMP'
   , audit_condition => 'DEPTNO = 10'
   , audit_column    => 'SAL'
   , handler_schema  => 'FGA_ADMIN'
   , handler_module  => 'FGA_ALERT_ACTION'
   , statement_types => 'SELECT, UPDATE'    -- если требуется, что-то вроде этого
);
END;
/
```

Обратите внимание, что такая техника фактически дает в руки инструмент для написания триггерных процедур на выборку данных (SELECT).

7.8. Общее управление записями аудита разных видов

Для единообразия Oracle предлагает таблицу (view) DBA_COMMON_AUDIT_TRAIL, в которую сведены данные как «стандартного» аудита (из DBA_AUDIT_TRAIL на основе AUD$), так и «подробного» (из DBA_FGA_AUDIT_TRAIL на основе FGA_LOG$).

С версии 11 общее управление записями аудита (стандартного и подробного) можно выполнять средствами пакета DBMS_AUDIT_MGMT. Пакет позволяет переносить сегменты данных таблиц AUD$ и FGA_LOG$ в отдельное табличное пространство, регулировать размеры файлов стандартного аудита, чистить записи из таблиц и файлов аудита.

7.9. Аудит с помощью триггерных процедур

Имеющийся в СУБД Oracle аппарат триггерных процедур дает еще одну возможность осуществлять слежение за действиями пользователей.

7.9.1. Отслеживание изменений отдельных строк таблиц

Ниже приводится пример триггерной процедуры, регистрирующей изменения строк таблицы пользователями (сценарий приведен для SQL*Plus):

```
CREATE TABLE row_modifications_audit (
  action     VARCHAR2 (  1 )
, table_key  VARCHAR2 ( 80 )
, timestamp  DATE
, table_name VARCHAR2 ( 30 )
, username   VARCHAR2 ( 30 )
);

CREATE OR REPLACE TRIGGER log_actions_on_&&table_name
AFTER INSERT OR UPDATE OR DELETE
ON &&table_name
FOR EACH ROW
DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;
   action    CHAR ( 1 );
   table_key VARCHAR2 ( 80 );
BEGIN
--
-- Занесем в рабочие переменные необходимую информацию
   IF DELETING THEN
        action := 'D';
        table_key := TO_CHAR ( :old.&&primary_key_name );
   ELSIF UPDATING THEN
        action := 'U';
        table_key := TO_CHAR ( :old.&&primary_key_name );
   ELSIF INSERTING THEN
        action := 'I';
        table_key := TO_CHAR ( :new.&&primary_key_name );
   END IF;
--
-- Занесем информацию из внутренних переменных в нашу таблицу аудита
   INSERT INTO row_modifications_audit (
     timestamp
   , table_name
   , table_key
   , action
   , username
   )
   VALUES (
     SYSDATE
   , '&&table_name'
   , table_key
   , action
   , USER
   );
   COMMIT;
END;
/
```

Упражнение. Создать таблицу ROW_MODIFICATIONS_AUDIT для сбора информации об изменениях строк. Создать триггерную процедуру для таблицы EMP и проверить ее работу.

Указание PRAGMA AUTONOMOUS_TRANSACTION и предложение COMMIT в теле триггернной процедуры использованны для гарантированной фиксации записи об изменении невзирая на возможную выдачу ROLLBACK в основном тексте программы. Если этого не требуется, и откат в основной

программе (означающий всего лишь *попытку* изменить запись в таблице) не должен сопровождаться записью аудита, и то, и другое из тела триггернной процедуры нужно будет убрать.

7.9.2. Отслеживание изменений строк с точностью до столбцов

Фунцкция-предикат UPDATING, которую можно использовать в теле DML-триггерной процедуры, допускает указание в качестве параметра имени столбца:

IF UPDATING (sal) THEN …

Это позволяет использовать триггерный механизм для слежения за изменениями значений в конкретных столбцах.

7.9.3. Отслеживание прочих действий

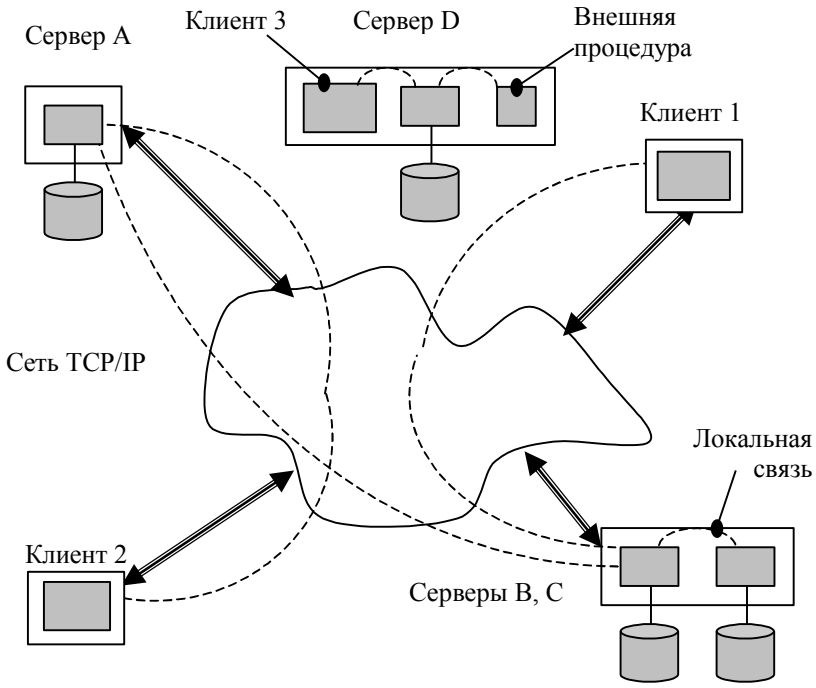
Триггерные процедуры в Oracle можно создавать также для событий класса «DDL» (выдача операторов DDL) и «database». Это позволяет регистрировать такие события, как выдачу команд CREATE, DROP, TRUNCATE, CONNECT, STARTUP и другие.

7.10. Отслеживание истории изменений в БД по журналу

Изменения, вносимые в БД, СУБД обычно протоколирует в журнале. Журнальные записи для компактности делаются в двоичном виде, и поэтому они недоступны для непосредственного прочтения администратором. Однако в версии 8.1 в состав ПО Oracle было введено средство LogMiner, позволяющее представить данные журнала в читабельном виде. LogMiner способен читать данные оперативных (online) журнальных файлов и архивных копий журнала, в том числе принадлежащих более старой версии Oracle, 8.0.

8. Администрирование работы в сети

8.1. Общая архитектура сетевой поддержки в Oracle



Основные элементы конфигурирования:

- Специальное сетевое ПО Oracle род названием Oracle Net (отвечает за способ адресации, способ соединения и вспомогательные операции по обмену данными между клиентом и сервером).
- СУБД (отвечает за схему соединений клиентов с серверами).

8.2. Устройство, конфигурирование и использование Oracle Net

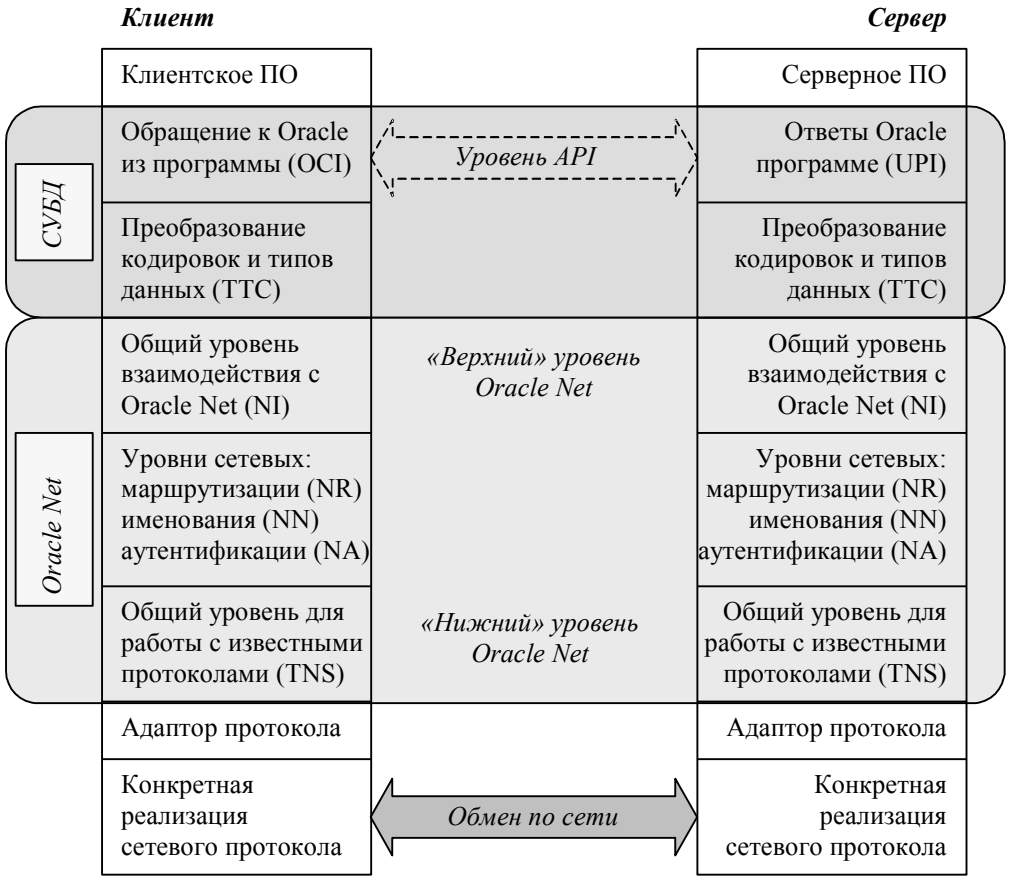
Названия ПО Oracle Net со временем претерпевали изменения: *SQL*Net версии 1* → *SQL*Net версии 2* → *Net8* → *Oracle Net*. Встречающиеся в сообщениях Oracle, именах объектов СУБД и элементов БД названия SQL*Net и Net8 в последних версиях следует полагать синонимами Oracle Net.

8.2.1. Общее устройство Oracle Net

Соответствие Oracle Net, TNS и архитектуры ISO/OSI:

Стек клиентской части	***Уровень OSI***	***Стек серверной части***
Клиентское приложение	7 (приложение)	Сервер Oracle
Oracle Net	6 (представление)	Oracle Net
TNS	5 (сеанс)	TNS
Адаптор протокола (Oracle)	4 (транспорт)	Адаптор протокола (Oracle)
	3 (сеть)	
	2 (соединение данных)	
	1 (физический)	

Стопка (стек, магазин) уровней взаимодействия клиентского ПО с серверным в наиболее типичном случае выглядит примерно так:



(Для JDBC — общение с БД программ на Java — стопка протоколов со стороны клиента выглядит иначе).

Расшифровка сокращений на рисунке (может понадобиться при разборе трассировочных сообщений):

OCI — Oracle Call Interface
UPI — User Program Interface
TTC — Two-Task Common
NI — Network Interface
NR — Network Routing
NN — Network Naming
NA — Network Authentication
TNS — Transparent Network Substrate

Адапторы протокола, поставляемые Oracle, позволяют работать с:

- TCP/IP (до версии 9 — с SPX/IPX, LU6.2 и др.), в том числе TCP/IP с SSL (TCPS);
- Named Pipes (Windows);
- IPC (связь между процессами);
- внешними по отношению к Oracle процедурами (благодаря тому, что связь СУБД с вызываемыми внешними процедурами осуществляется через IPC).

Взаимодействие на уровне приложения осуществляется с помощью API (OCI или OPI), вызовы которого выполняют следующие функции:

- Соединение с сервером и отсоединение
- Трансляцию SQL-предложений
- Открытие курсоров
- Привязку переменных приложения к участкам памяти на сервере
- Описание столбцов собственно таблиц и представлений данных

- Запуск на выполнение SQL-предложений
- Извлечение строк-результатов
- Закрытие курсоров
- Обработку исключительных ситуаций.

Пример.

Пусть в архитектуре клиент/сервер через Oracle Net осуществляется запрос:

```
SELECT empno, ename
FROM emp
WHERE deptno = :dept_needed
```

Выполнение этого запроса вызовет обмен сообщений по сети примерно со следующими действиями:

1. Будет открыт курсор
2. Выдана команда на трансляцию SQL-предложения
3. Осуществлена привязка переменной dept_needed
4. Описаны столбцы таблицы EMP
5. Выдана команда на выполнение запроса
6. Отработана процедура по извлечению результата (количество строк, извлекаемых за один раз, определяется значением параметра ARRAYSIZE)
7. Закрытие курсора

8.2.2. Расширения и дополнения к Oracle Net

8.2.2.1. Advanced Security

До версии 9 средство именовалось Advanced Networking Option.

Advanced Security обеспечивает «повышенную» безопасность сетевого обмена (а на самом деле простую защищенность):

- защищенность передаваемых данных путем:
  - шифрования информации
  - защиты от подмены (вычисления и передачи контрольной свертки)
- использование промышленных способов аутентификации:
  - Kerberos
  - RADIUS (Remote Authentication Dial-In User Service)
  - DCE (Distributed Computing Environment, набор служб для распределенной среды, предложенный консорциумом OSF)
  - SSL (Secure Sockets Layer, с поддержкой цифровых сертификатов)
  - Entrust/PKI (система открытых ключей в исполнении фирмы Enrust)

8.2.2.2. Справочник имен/каталогов

Использование справочника имен или каталогов (более сложный вариант) позволяет, в частности, осуществлять динамические преобразование имен соединений с серверами Oracle в сети в физические адреса (начиная с версии 8.1 фирма предпочитает говорить не о «соединениях с БД», а о «сетевых службах БД», services). Имеет смысл при наличии большого числа компьютеров, когда установка и переписывание файла *tnsnames.ora* на каждой клиентской машине затруднительна организационно, и при большом числе пользователей.

Устаревший ныне справочник имен Oracle Names является собственной разработкой Oracle, не совместимой с другими системам и протоколами. В версии 9 поддерживался последний раз для совместимости.

Современный Oracle Internet Directory (OID) — справочник каталогов, разработанный Oracle на основе стандарта LDAP (Lightweight Directory Access Protocol), и по этой причине совместимый с другими системами на основе LDAP. Позволяет организовать в Oracle:

- систему глобальных имен пользователей для работы с разными БД
- систему глобальных ролей пользователей БД
- систему имен БД.

В соответствие с правилами LDAP имена в справочнике OID моделируются иерархиями («иерархическая модель данных», с точки зрения БД). Сами иерархии OID хранит в одной или нескольких «обычных» БД Oracle, назначенных для этой цели. Если прикладная программа использует OID, она сообщит справочнику OID символическое имя, а он уже найдет для того соответствующий физический эквивалент, необходимый для работы с конкретной БД.

Кроме этого Oracle позволяет работать со справочниками каталогов Active Directory (Microsoft) и NDS (Novell; в версии 9 поддержка прекращена), которые также используют протокол LDAP и поэтому являются альтернативой для OID.

8.2.2.3. Connection Manager

Доверительный (proxy) сервер соединений между клиентами и серверами Способен решать следующие задачи:

- мультиплексирование сеансов соединений с сервером;
- контроль доступа (пользуясь встроенным механизмом описания правил доступа).

Может употребляться для подсоединения с сервером, принадлежащим внутренней сети, отгороженной межсетевым экраном (firewall) от интернета.

8.3. Конфигурирование Oracle Net

8.3.1. Конфигурируемые компоненты Oracle Net

В своем основном варианте ПО Oracle Net состоит из трех компонент:

- *Клиентская компонента*. ПО, инициирующее связь с сервером Oracle с клиентского компьютера, например со стороны web-страницы, программы или другого сервера.
- *Серверная компонента*. ПО, с которым соединяется клиент для связи с сервером Oracle или внешней процедурой.
- *Слушающий процесс, слухач (listener),* а по смыслу — *соединитель*. (TNS-слухач, Net-слухач). Процесс, фактически организующий соединение клиентской программы с СУБД или с внешними процедурами. Адреса точек соединения могут заранее заноситься в файл *tnsnames.ora*, запоминаться в справочнике Oracle Names (если он используется; его адрес в этом случае находится в файле *names.ora*), в OID или в каком-нибудь другом сервере каталогов.

Конфигурирование достигается правкой содержимого специальных файлов:

- *listener.ora* — задает конфигурацию слушающего процесса (серверная часть).
- *sqlnet.ora* — файл с параметрами соединения (клиентская и серверная части).
- *tnsnames.ora* — возможный файл с определением точек входа в БД (клиентская часть).
- *cman.ora*, *ldap.ora* — могут использоваться для конфигурирования дополнительных компонент Oracle Net (здесь: connection manager и справочника LDAP).

Расположение файлов по умолчанию — в *%ORACLE_HOME%\network\admin*; но их можно разместить в любом угодном каталоге, название которого прописать в специальной переменной среды окружения TNS_ADMIN в ОС. При обращении прикладной программы к СУБД, Oracle Net всегда (1) проверит сначала установку TNS_ADMIN, и только (2) если она не сделана, воспользуется умолчательным каталогом. Переменной TNS_ADMIN удобно пользоваться, когда на одном компьютере существует несколько разных каталогов

ORACLE_HOME — например при установленых на компьютере одновременно ПО СУБД Oracle нескольких разных версий, а может серверного и клиентского ПО, и так далее.

8.3.2. Программы для работы с Oracle Net

Основные программы для работы с Oracle Net следующие:

- Net Manager (*netmgr*) и Net Configuration Assistant (*netca*)
- *lsnrctl*
- *tnsping*

Примерами дополнительных программ могут служить *cmctl* (управление распорядителем соединений Connection Manager) и *srvctl* (управление БД, СУБД, процессами listener и др. в составе RAC).

8.3.2.1. Программы Net Manager и Net Configuration Assistant

Программа Net Manager (*netmgr*) может рассматриваться как своего рода специализированный редактор текстовых файлов *sqlnet.ora*, *tnsnames.ora* и *listener.ora*. Она достаточна для большинства потребностей конфигурирования Oracle Net. Но все же имеются указания, которые в файлы конфигурации сетевого ПО можно вносить только вручную с помощью текстового редактора. Кроме того знакомство с текстами конфигурируемых файлов дает понимание лежащих в основе механизмов.

Несмотря на простоту решаемых задач, Net Manager — программа, использовать которую настоятельно рекомендуется начинающим. В силу встроенных проверок, она предохраняет администратора от ошибок при правке конфигурационных файлов при том, что никакой техники отладки для этих файлов в Oracle Net не предусмотрено.

Программа Net Configuration Assistant (*netca*) может использоваться для создания и удаления компонент Oracle Net. В отличие от *netmgr*, эта программа регистрирует создаваемые ею компоненты в Oracle Restart, когда тот сконфигурирован.

8.3.2.2. Программа *tnsping*

Используется для проверки наличия в сети соединяющего процесса listener для подвода клиентов к нужной БД в сети. Общий вид обращения:

```
>tnsping <сетевое имя службы БД> [<счетчик повторений>]
```

Примеры:

```
>tnsping prz61:5521/orcl112.oraserv.class
>tnsping orcl112.oraserv.class 10
```

8.3.2.3. Программа *lsnrctl*

Программа *lsnrctl* служит для управления работой процесса listener и наблюдения за ним. Она вызывается из командной строки ОС и способна работать в режимах диалога и пакетном. Для диалога достаточно набрать:

lsnrctl [<*имя процесса-соединителя*>]

Пример пакетного запуска единственного на компьютере процесса listener:

```
>lsnrctl start
```

Пример пакетного останова работы процесса listener с именем LISTENER2:

```
>lsnrctl stop LISTENER2
```

Вопреки опасениям начинающих администраторов, останов процесса listener *не* разрывает существующих соединений с СУБД, а только прекращает установление новых.

В ОС Windows выдача последних двух команд равносильна запуску и останову службы *Oracle<...>TNSListener*. На каждый процесс listener Oracle заводит в Windows собственную службу.

8.3.3. Способы адресации клиентом нужной службы БД

В файлах конфигурации Oracle Net активно используется адресация службы БД. ПО Oracle Net разрешает клиентской программе по-разному указывать БД для соединения:

- «гостевая» адресация по сетевому имени компьютера с БД (HOSTNAME);
- «простая» адресация с указанием трех компонент — имени компьютера, порта и имени БД (EZCONNECT[10-]);
- полная адресация, указываемая клиентской программой напрямую или через посредство логического имени (TNSNAMES);
- адресация посредством службы каталогов (ONAMES[-9], LDAP);
- прочие способы (NIS, DCE — Cell Directory Service OSF, NDS — Novell Netware Directory Service).

[10-] С версии 10.
[-9] До версии 10.

В качестве адресата выступает служба БД, имя которой очень часто совпадает с глобальным именем БД (но не обязано), или (в конечном счете) СУБД, обслуживающая доступ к нужной БД. Последнее сохранено ради обратной совместимости с SQL*Net.

8.3.3.1. Адресация указанием имени компьютера

Возможна если полное имя БД совпадает с именем компьютера, где она расположена. Пример:

```
SQL> CONNECT scott/tiger@prima.class
```

Если на компьютере установлено несколько БД, ему можно присвоить несколько сетевых имен. Если сетевое имя не совпадает с именем БД, можно попытаться промоделировать этот способ адресации, подправив файл *drivers/etc/hosts* на клиентской машине.

Это самый простой способ адресации, не требующий использования конфигурационных файлов Oracle Net на клиенте, но он же и самый ограниченый в своих возможностях.

8.3.3.2. Адресация путем явного указания трех компонент адреса

«Простая» (easy) адресация заключается в явном указании клиентом сетевого имени компьютера, номера порта с программой *listener* и имени службы БД. Пример:

```
SQL> CONNECT scott/tiger@//localhost:1521/prima.class
```

Вольности написания: две черты подряд '//' можно опускать.

Не требует конфигурационных файлов Oracle Net на клиенте и не допускает дальнейших уточнений желаемых свойств соединения.

8.3.3.3. Полная адресация сервера клиентом

Полная адресация — самый продвинутый способ явного указания клиентом нужной службы БД в сети и желаемых свойств соединения. Пример явной полной адресации в явно указанном (для Oracle Net) описателе соединения:

```
>sqlplus scott/tiger@"(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST= oraserv)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=prima.class)))"
```

(все это набирается в ОС, а при необходимости в тексте программы, одной строкой).

Замечание. До версии 8.1 действовало правило соединения с ***СУБД***: CONNECT_DATA = (SID = ***имя_СУБД***); начиная с версии 8.1 фирма Oracle перешла на соединение со ***службой БД***: CONNECT_DATA = (SERVICE_NAME = ***имя_БД***), но соединение с СУБД по-прежнему допустимо.

Явно выписанный таким образом описатель («строка») соединения не нуждается в конфигурационном файле *tnsnames.ora* на клиентской машине. В жизни же почти всегда употребляется логическая адресация, при которой полный описатель заносится в файл *tnsnames.ora* под каким-нибудь логическим именем, а программа в команде установки соединения ссылается на это имя. Если адресная строка, приведенная выше, обозначена в *tnsnames.ora* под именем SOURCE (см. ниже), запрос программы на соединение сможет быть указан заметно короче:

```
>sqlplus scott/tiger@source
```

8.3.4. Файлы конфигурирации Oracle Net

8.3.4.1. Файл *listener.ora*

Задает параметры конфигурации слушающего процесса. Используется на компьютере с программой *listener*, то есть чаще всего там, где работает СУБД. Строго говоря, процесс listener в состоянии работать и без файла *listener.ora*, и тогда его параметры полагаются по умолчанию, но это не лучшая практика.

Пример содержимого *listener.ora* после типовой установки БД в версии 11:

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = CLRExtProc)
      (ORACLE_HOME = C:\app\oracle\product\11.2.0\dbhome_1)
      (PROGRAM = extproc)
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = oraserv)(PORT = 1521))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
  )

ADR_BASE_LISTENER = C:\app\oracle
```

В предшествующих версиях (и в других ОС) содержимое будет выглядеть аналогично.

Здесь определен один процесс listener под названием LISTENER. Он принимает заявки на сеансы соединения с СУБД по TCP/IP и запросы со стороны процесса extproc от внешних процедур. Заметьте, что описание службы БД, с которой сможет соединять клиентов этот процесс, не приведено. Это возможно с версии Oracle 8, где появилась автоматическая регистрация процессом listener наличных БД после его

запуска. Но ничто не мешает явно («статично») вписать нужные определения в *listener.ora*, и иногда в этом есть смысл.

Рассмотрим пример. Пусть для повышения производительности и надежности требуется иметь дополнительный процесс listener для приема заявок на соединение по TCP/IP (но тогда уже через другой порт) и для защищенности отдельный процесс listener для связи с внешними процедурами. Первый назовем BACKLISTENER, второй EXPPROCLISTENER. Тогда файл *listener.ora* можно исправить следующим образом:

```
SID_LIST_EXTPROCLISTENER =
  (SID_LIST =
    (SID_DESC =
      (PROGRAM = extproc)
      (SID_NAME = CLRExtProc)
      (ORACLE_HOME = C:\app\oracle\product\11.2.0\dbhome_1)
    )
  )

ADR_BASE_EXTPROCLISTENER = C:\app\oracle\product\11.2.0\dbhome_1\log

LISTENER =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = oraserv)(PORT = 1521))
  )

ADR_BASE_LISTENER = C:\app\oracle

BACKLISTENER =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = oraserv)(PORT = 1522))
  )

ADR_BASE_BACKLISTENER = C:\app\oracle

EXTPROCLISTENER =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
  )
```

Для того, чтобы новая конфигурация вступила в силу, потребуется перезапустить LISTENER и запустить BACKLISTENER и EXTPROCLISTENER:

```
>lsnrctl stop LISTENER
>lsnrctl start LISTENER
>lsnrctl start BACKLISTENER
>lsnrctl start EXTPROCLISTENER
```

При реконфигурациях, приводящих к удалению процесса listener в среде Windows не помешает удалить службу ОС, например:

```
>sc delete OracleOraDb11g_home1TNSListenerEXTPROCLISTENER
```

8.3.4.2. Файл *tnsnames.ora*

Задает параметры для осуществления соединения клиента с серверами. Устанавливается на клиентском компьютере.

Для приводившейся выше строки соединения, соединение под именем SOURCE может быть обозначено в *tnsnames.ora* так:

```
SOURCE =
```

```
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = oraserv)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = prima.class)
    )
  )
```

Отступы и пробелы служат исключительно ради удобства чтения.

Полная адресация позволяет запросить у Oracle Net соединение гораздо изощреннее, чем в «гостевом» и «простом» случаях. Вот пример, когда клиентскому ПО предлагается перебирать адреса в порядке их указания до установления успешной связи через *listener* на каком-нибудь из двух компьютеров:

```
SOURCE =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = winserv)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = winserv)(PORT = 1522))
      (ADDRESS = (PROTOCOL = TCP)(HOST = oelinux)(PORT = 1521))
      (SOURCE_ROUTE = yes)
      (FAILOVER = false)
    )
    (CONNECT_DATA =
      (SERVICE_NAME = prima.class)
    )
  )
```

Подобную возможность можно употребить для балансирования загрузки, для работы с БД, имеющей горячий резерв или для соединения через межсетевой экран при использовании программы Connection Manager.

Возможный вариант описания отказоустойчивого соединения клиента с БД Oracle в конфигурации RAC:

```
OLTP =
  (DESCRIPTION =
    (LOAD_BALANCE = ON)
    (FAILOVER = ON)
    (ADDRESS = (PROTOCOL = TCP)(HOST = server01)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = server02)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = server03)(PORT = 1521))
    (CONNECT_DATA =
      (SERVICE_NAME = OLTP_SERVICE)
      (FAILOVER_MODE =
        (TYPE = SELECT)
        (METHOD = BASIC)
        (RETRIES = 20)
        (DELAY = 1)
      )
    )
  )
```

Допустимы и другие уточнения свойств запрашиваемых соединений.

8.3.4.3. Файл *sqlnet.ora*

Файл, задающий (возможно дополнительные) параметры соединений клиентских программ с СУБД. Ниже перечисляются только *некоторые* свойства соединений, задаваемые с помощью этого файла.

- ***Последовательность перебора способов адресации клиентом сервера***. Например:

  NAMES.DIRECTORY_PATH = (HOSTNAME, TNSNAMES)

  Допускаются способы: TNSNAMES, HOSTNAME, EZCONNECT, ONAMES, LDAP, NIS, CDS, NDS

- ***Обнаружение разорванных пользовательских соединений с СУБД***. Задается интервал времени, через который сервер будет автоматически проверять активность клиента. Если клиент «пропал» (например, клиентскую машину перезагрузили), все отведенные для работы с ним ресурсы на сервере освободятся. Задается параметром

  SQLNET.EXPIRE_TIME = <*время_в_минутах*>

- ***Вид аутентификации при соединении по сети***. Указывается параметром

  SQLNET.AUTHENTICATION_SERVICES = (*вид_аутентификации*)

  Видами_аутентификации могут быть: NONE (при соединении нужно будет обязательно явно указывать имя/пароль), ALL и NTS (Windows NT native authentication, при которой имя/пароль, указанные при входе в ОС, обеспечат автоматическое подключение к СУБД). (Возможны и прочие *виды*, но при установленной возможности Advanced Security).

- ***Включение диагностики (трассировки) соединений до версии 11***. Две схожих группы параметров позволяют включить отладочную трассировку соединений с СУБД соответственно на клиентской и на серверной сторонах:

TRACE_LEVEL_CLIENT	уровень трассировки или ее отсутствие
TRACE_DIRECTORY_CLIENT	каталог ОС для помещения файлов трассировки
TRACE_FILE_CLIENT	начало имени для файла с данными трассировки
TRACE_UNIQUE_CLIENT	включение автоматического именования файлов
TRACE_TIMESTAMP_CLIENT	включение в записи о событиях указание времени
TRACE_FILELEN_CLIENT	ограничитель размера трассировочного файла
TRACE_FILENO_CLIENT	максимальное количество трассировочных файлов

TRACE_LEVEL_SERVER	уровень трассировки или ее отсутствие
TRACE_DIRECTORY_SERVER	каталог ОС для помещения файл трассировки
TRACE_FILE_SERVER	начало имени для файла с данными трассировки
TRACE_TIMESTAMP_SERVER	включение в записи о событиях указание времени
TRACE_FILELEN_SERVER	ограничитель размера трассировочного файла
TRACE_FILENO_SERVER	максимальное количество трассировочных файлов

  Если для обращения к Oracle Net используется переменная ОС TNS_NAMES, для реального выполнения трассировки требуется включить указание на каталог файлов трассировки в переменную PATH.

- ***Включение диагностики (трассировки) соединений с версии 11***. В версии 11 новые правила диагностики (ADR) используют для включения трассировки другую группу параметров:

ADR_BASE	каталог ОС для накопления файлов диагностики
DIAG_ADR_ENABLED	если ON, включены правила ADR
TRACE_LEVEL_CLIENT	то же, что без ADR
TRACE_LEVEL_SERVER	то же, что без ADR
TRACE_TIMESTAMP_CLIENT	то же, что без ADR
TRACE_TIMESTAMP_SERVER	то же, что без ADR

- ***Включение протоколирования соединений***. Две группы параметров позволяют включить отладочное протоколирование (журнализацию) соединений с СУБД соответственно на клиентской и на серверной машинах:

LOG_DIRECTORY_CLIENT	каталог ОС для помещения файла журнала
LOG_FILE_CLIENT	имя файла журнала
LOG_DIRECTORY_SERVER	каталог ОС для помещения файла журнала
LOG_FILE_SERVER	имя файла журнала

  Если для обращения к Oracle Net используется переменная ОС TNS_NAMES, для реального выполнения протоколирования требуется включить указание на каталог файла журнала в переменную PATH.

- ***Шифрование передаваемых данных, передача контрольной свертки и внешние способы аутентификации***. При установленной возможности Advanced Security ряд параметров *sqlnet.ora* позволяет включить автоматическую шифровку передаваемых по Oracle Net данных и пересылку вместе с сообщением его свертки (вычисляемой строки), препятствующей подмене в процессе передачи. Другие параметры устанавливают взаимодействие с промышленными средствами аутентификации.

- Прочие свойства соединений.

Пример указания параметров в *sqlnet.ora*:

```
AUTOMATIC_IPC = OFF
TRACE_LEVEL_CLIENT = USER
NAMES.DEFAULT_DOMAIN = class
SQLNET.AUTHENTICATION_SERVICES = ( NTS )
```

С версии Oracle 9 содержимое файла *sqlnet.ora* можно править при помощи программы Net Manager.

8.4. Настройка соединений по Oracle Net

Ниже приводятся два примера настройки производительности работы Oracle Net. В целом настройка Oracle Net специфична и часто неоднозначна, но умолчательных установок обычно вполне достаточно. Более подробную информацию о настройке можно найти в документации по Oracle.

8.4.1. Изменение параметра SDU

Oracle Net возвращает клиенту данные через буфер session data units (SDU). Размер SDU по умолчанию — 2048 байтов; допустимые значения — от 512 байтов до 32 Кб (Net8 в версии 8.1.7). Если требуется послать 4097 байтов данных, Oracle Net фактически шлет 2048 + 2048 + 2048 = 3 пакета.

Изменение параметра SDU всегда конкретно и общего правила нет. Следует помнить, что при обмене между клиентом и сервером Oracle Net выбирает наименьший SDU из заявленых клиентом и сервером.

Пример вставки указания SDU в файл *tnsnames.ora*:

```
PRIMA.CLASS =
     ( DESCRIPTION =
          ( SDU = 4096 )
          ( ADDRESS = ……
```

Пример вставки указания SDU в файл *listener.ora* для назначенных серверных процессов:

```
SID_LIST_LISTENER =
     ( SID_LIST =
```

```
          ( SID_DESC =
               ( SDU = 4096 )
               ( SID_NAME = PRIMA )
……….
```

Пример указания SDU в параметре СУБД для определения совместных серверных процессов:

```
DISPATCHERS="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp))(SDU=2920))"
```

Пример указания SDU в файл *sqlnet.ora*:

```
DEFAULT_SDU_SIZE = 32767
```

В целях снижения накладных расходов рекомендуется выбирать SDU кратным значению максимального размера сегмента (maximum segment size, MSS) в TCP с добавлением 1 в случае неиспользования шифрования в Advanced Security Opton. (Для TCP/IP version 4 в Ethernet значение MSS по умолчанию равно 1460; то есть без использования шифрования будем иметь 1460 * n +1). Другие соображения приведены в документации по Oracle.

8.4.2. Изменение параметра ARRAYSIZE

Строго говоря, это не параметр Oracle Net, а параметр управления передачей данных по Oracle Net. Задает число строк результата запроса, передаваемое в рамках отдельной акции сетевой передачи (то есть результат возвращается клиенту порциями по ARRAYSIZE строк). Регулируется по-разному в разных программных компонентах:

- Developer (SQL*Forms) (обработка массивами)
- Прекомпиляторы
- Драйвер JDBC фирмы Oracle для Java (свойство defaultRowPrefetch)
- Программный интерфейс с Oracle (OCI) (параметр OFEN операции извлечения данных OEXN)
- SQL*Plus
  Выставляется командой SET ARRAYSIZE. Значение по умолчанию — 15 (версия 8.1). Увеличение параметра при копировании больших таблиц с помощью соединения с удаленным сервером дает большой выигрыш.

Большая величина ARRAYSIZE ускоряет передачу результатов запроса в программу, но одновременно требует большого объема памяти под буфер возвращаемых данных.

8.5. Конфигурирование способа обслуживания СУБД соединений клиентов

Техника обслуживания СУБД соединений клиентских программ складывалась в Oracle постепенно, предлагая по мере возникающих задач и проблем новые варианты решений. Они рассматриваются ниже.

Все варианты установления соединений конфигурируются в СУБД (хотя и по-разному), однако клиент, запрашивая соединение, имеет право запросить желаемую технику, через локальный файл *tnsnames.ora* или явным указанием.

8.5.1. Техника назначенного (dedicated) серверного процесса

До версии 7 в Oracle действовала единственная техника установления соединения клиентской программы с СУБД. Каждому возникающему сеансу назначался специально создаваемый процесс СУБД (dedicated server process), который обслуживал текущие нужды сеанса и удалялся по его завершению.

Запрос на соединение в этом случае принимается процессом listener (он обязан при этом располагаться на одном компьютере с СУБД), который санкционирует порождение серверного процесса (SHAD) в СУБД и выдаст клиенту для общения с ним автоматически подобранный порт.

Пример из *tnsnames.ora* локального описания соединения с явным запросом клиента на работу с назначенным серверным процессом:

```
SOURCE =
```

```
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = oraserv)(PORT = 1521))
  )
  (CONNECT_DATA =
    (SERVICE_NAME = prima.class)
    (SERVER = DEDICATED)
  )
)
```

Пример запроса на работу с назначенным серверным процессом в случае простой адресации:

```
SQL> CONNECT scott/tiger@//localhost:1521/prima.class:dedicated
```

Поскольку когда-то эта техника была единственной, явного указания DEDICATED не требуется, однако для ясности приложения это лучше все-таки делать.

Некоторые операции по администрированию, в том числе по резервированию и восстановлению данных, могут выполняться только с использованием назначенного процесса-сервера. Для других задач эта техника может оказаться не единственной, но наиболее приемлемой.

8.5.2. Техника использования совместных (shared) серверных процессов

Когда одновременных сеансов довольно много, или когда они часто возникают на короткое время, техника соединения с назначенным серверным процессом становится чрезмерно затратной.

В версии 7 появилась *дополнительная* схема соединения. По ней в СУБД изначально заводятся запас (pool) совместных процессов-серверов (shared server processes) и организующие их употребление процессы диспетчеров-распределителей (dispatchers). Порядок работы такой схемы соединений следующий:

(1) Заявку на соединение процесс listener передает свободному диспетчеру, который далее будет обслуживать соединение. Он принимает от программы запросы и ставит их в *общую* для всех диспетчеров *очередь*.
(2) При первой свободной возможности *какой-нибудь* из совместных серверных процессов подхватывает первую по порядку заявку из общей очереди на обслуживание.
(3) После обработки запроса совместный серверный процесс помещает результат в *очередь ответов* того *диспетчера*, который обслуживает данное соединение.
(4) При первой возможности диспетчер выбирает из своей очереди первый по порядку результат и возвращает его в программу.

Таким образом в разное время жизни сеанса он может обслуживаться разными серверными процессами (хотя и через одного диспетчера); они не будут постоянно порождаться и исчезать и их число вполне реально иметь меньше количества сеансов. В версии 7 такая схема соединения носила название MTS-server (многониточный сервер), а с версии 8 — shared server, хотя следы первоначального названия кое-где сохранились.

В отличие от соединений с назначенным серверным процессом, соединениях с использованием совместных серверных процессов могут устанавливаться процессом listener, работающим на другом компьютере, чем СУБД. Это дает шанс снять часть нагрузки с основного сервера.

Архитектура совместных серверных процессов воплощена в СУБД, и поэтому конфигурируется параметрами СУБД. Объектами конфигурирования являются следующие:

Объект конфигурирования	***Параметр СУБД***
Количество и свойства процессов-диспетчеров	DISPATCHERS
Начальное, и в дальнейшем минимальное количество совместных серверных процессов (Snnn)	SHARED_SERVERS
Максимальное количество совместных серверных процессов	MAX_SHARED_SERVERS

Максимальное количество сеансов для обслуживания совместными серверными процессами	SHARED_SERVER_SESSIONS
Максимальное количество работающих цепочек соединений сеансов через диспетчеров и совместные процессы (расходуют память в SGA)	CIRCUITS

Для большинства значений параметров имеются умолчания, которые выводятся из значения DISPATCHERS.

Пример установки параметра DISPATCHERS:

```
DISPATCHERS = '(PROTOCOL=TCP) (DISPATCHERS=3) (PROTOCOL=IPC)
(DISPATCHERS=1)'
```

После запуска СУБД в ее составе будет присутствовать три процесса «диспетчер» для приема клиентских соединений по TCP/IP (например, D000, D001, D002) и один для соединений по IPC (например D003). Диспетчер будет принимать заявки на соединения со службой БД, указанной параметром СУБД SERVICE_NAMES.

Другой пример:

```
DISPATCHERS = '(PROTOCOL=TCP) (DISPATCHERS=2) (SERVICE=primaXDB,
prima.class)'
```

Два диспетчера будут принимать заявки на соединения по TCP/IP со службами БД primaXDB и prima.class.

В файле *INIT.ORA* параметр DISPATCHERS можно указывать несколько раз, и если значения относятся к разным протоколам, они добавляются друг к другу. Для файла SPFILE.ORA тот же эффект получится повторением команды ALTER SYSTEM; при этом сброс значения достигается сообщением ему пустой строки ''. Например, первую из установок, приведенных выше, можно выполнить следующей последовательностью:

```
ALTER SYSTEM SET DISPATCHERS = '';
ALTER SYSTEM SET DISPATCHERS = '(PROTOCOL=TCP) (DISPATCHERS=3)';
ALTER SYSTEM SET DISPATCHERS = '(PROTOCOL=IPC) (DISPATCHERS=1)';
```

В полном объеме установка параметра DISPATCHERS описана в документации по Oracle.

Пример из *tnsnames.ora* локального описания соединения с явным запросом клиента на работу с совместными серверными процессами:

```
SOURCE =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = oraserv)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = prima.class)
      (SERVER = SHARED)
    )
  )
```

Пример запроса на работу с совместными серверными процессами в случае простой адресации:

```
SQL> CONNECT scott/tiger@//localhost:1521/prima.class:shared
```

Справочные таблицы для наблюдения за работой механизма совместных серверных процессов:

V$CIRCUIT
V$SHARED_SERVER

V$SHARED_SERVER_MONITOR
V$QUEUE
V$DISPATCHER
V$DISPATCHER_RATE
V$DISPATCHER_CONFIG[10-)]
V$SESSION
[10-) С версии 10.

Использование совместных серверных процессов дает отдачу (экономию времени подключения и расходования памяти по отношению к назначенным серверным процессам) при наличии в сеансах большого количества коротких транзакций с небольшими объемами результата запросов. Например, оно неэффективно в задачах пакетной обработки данных, анализа данных или массовой загрузки.

8.5.3. Техника использования общего запаса назначенных серверных процессов (connection pool)

Для задач совместного использования БД с сервером приложений характерно обслуживание СУБД большого количества сеансов, выполняющих небольшой объем работ. Открывать на короткое время сеансы связи с СУБД с большой частотой оказывается затратно, и с версии 11 в СУБД встроили возможность держать открытым сравнительно небольшой *запас* (pool) назначенных серверных процессов, по которым распределять запросы, приходящие от сеансов (например, со стороны сервера приложений). Техника получила название Database Resident Connection Pooling (DRCP; «общего запаса сеансов»). Слова database resident напоминают о том, что подобную технику нередко имеет в своем арсенале инструментарий сервера приложений, внешний по отношению к СУБД; например, это касается JDBC. Oracle DRCP предназначен для тех случаев, когда приложение не обладает собственными возможностями connection pool (например, PHP).

Поступивший от клиента запрос при этом принимается *брокером сеансов* (connection broker), который выбирает из запаса свободный серверный процесс (Lnnn) и соединяет сеанс напрямую с этим процессом для обработки запроса. Когда серверный процесс обработал запрос, он освобождается и возвращается в общий запас.

В версии 11 может использоваться только один запас процессов SYS_DEFAULT_CONNECTION_POOL. В будущем предполагается разрешить заводить их больше.

Управление DRCP производится посредством пакета DBMS_CONNECTION_POOL.

Пример запуска DRCP для умолчательного встроенного запаса:

```
SQL> EXECUTE DBMS_CONNECTION_POOL.START_POOL ( )
```

Пример останова:

```
SQL> EXECUTE DBMS_CONNECTION_POOL.STOP_POOL ( )
```

Для конфигурирования предразначена процедура CONFIGURE_POOL пакета. Вот некоторые ее параметры, характеризующие возможности техники DRCP:

Параметр CONFIGURE_POOL	***Роль***
MINSIZE	Минимальное количество серверных процессов в запасе.
MAXSIZE	Максимальное количество серверных процессов в запасе.
INCRSIZE	Количество добавляемых процессов при поступлении заявки на соединение и при отсутствии свободных.
SESSION_CACHED_CURSORS	Параметр SESSION_CACHED_CURSORS для всех сеансов, работающих через процессы из запаса.
INACTIVITY_TIMEOUT	Время простоя сеанса, после которого он будет убит.
MAX_LIFETIME_SESSION	Максимально допускаемая продолжительность сеанса при работе через запас серверных процессов.

Как видно, техника расчитана на активную работу сеансов, так что бездействие сеанса приведет к его насильственному прекращению.

Пример из *tnsnames.ora* локального описания соединения с явным запросом клиента на использование резидентного в СУБД общего запаса назначенных серверных процессов:

```
SOURCE =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = oraserv)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = prima.class)
      (SERVER = POOLED)
    )
  )
```

Пример запроса на соединение по типу DRCP в случае простой адресации:

```
SQL> CONNECT scott/tiger@//localhost:1521/prima.class:pooled
```

Справочные таблицы для наблюдения за работой механизма запаса серверных процессов:

DB_CPOOL_INFO
V$CPOOL_STAT
V$CPOOL_CC_STATS

Использование общего запаса назначенных серверных процессов дает отдачу (экономию времени подключения по отношению к обычным назначенным серверным процессам и памяти по отношению к совместным серверным процессам) при наличии большого количества сеансов сервера приложений, выполняющих работу от имени небольшого количества пользователей Oracle и при том, что сеансы сервера приложений не требуют однозначного сопоставления конкретным сеансам Oracle.

9. Экземпляр СУБД Oracle

Экземпляром СУБД (instance) в Oracle называется совокупность процессов и структур оперативной памяти, реализующих работу с данными, расположенными в файлах, то есть собственно с БД.

9.1. Составные части экземпляра СУБД

Программно работу СУБД Oracle обеспечивают следующие процессы:

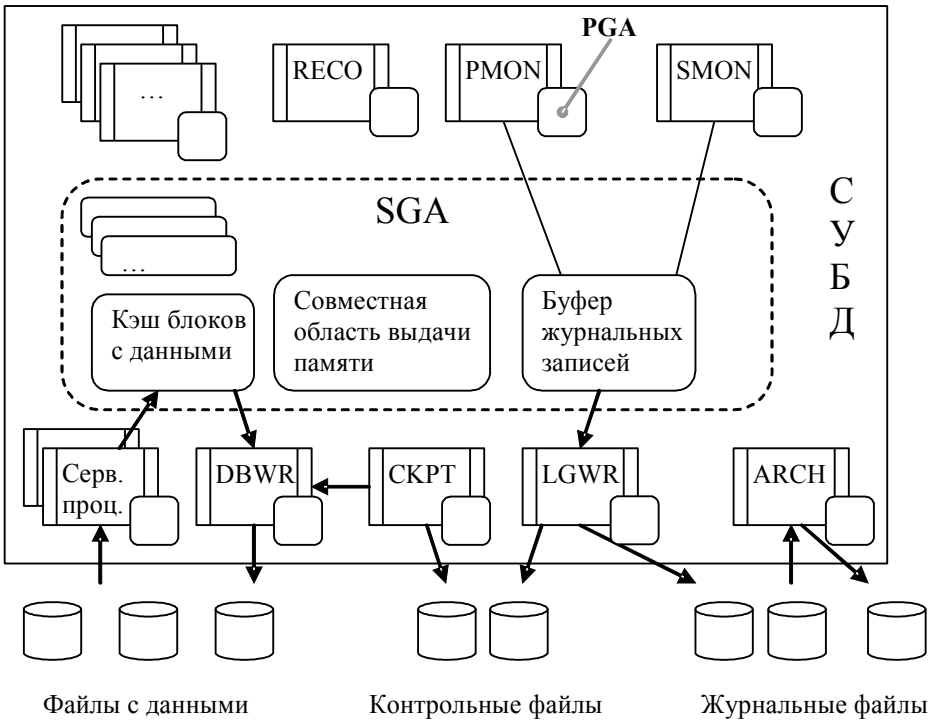
- *Фоновые процессы*. Процессы общего назначения, присутствуют постоянно и осуществляют необходимые действия по вводу/выводу, контролю работы системы и т.д.
- *Серверные (= «теневые») процессы*. Создаются СУБД специально для обслуживания сеансов связи прикладных программ с СУБД. Промежуточное звено между СУБД и процессами пользователей, возникающее по схеме «один к одному» (в случае назначенных серверных процессов, dedicated) или «один ко многим» (в случае совместных серверных процессов, shared).

В Windows процессы Oracle воплощены нитями (threads) вычислений (а процессом ОС является *oracle.exe*); в Unix (а) процессами ОС, однако (б) с версии 12 по выбору администратора также нитями.

Области данных в ОЗУ обеспечивают нужды СУБД в памяти самого разного характера. Имеется три основные категории областей данных:

- «Глобальная область системы» (SGA), рабочая область для использования всеми процессами СУБД.
- «Глобальные области процессов» (PGA), рабочие области памяти для каждого отдельного процесса СУБД.
- «Глобальные области сеансов» (UGA), рабочие области памяти для каждого отдельного сеанса. UGA, в отличие от SGA и PGA, — логическое понятие, не существующее самостоятельно физически.

Ниже показана взаимосвязь основных процессов и структур памяти Oracle.



9.2. Процессы СУБД

9.2.1. Обязательные фоновые процессы

Фактический набор процессов Oracle зависит от значений в файле параметров СУБД и от текущего режима работы. Ниже перечисляются фоновые процессы, присутствующие в каждом экземпляре СУБД, от персонального Oracle до крупной Unix-системы. Отказ любого из этих процессов приводит к останову остальных и для восстановления работоспособности СУБД нужен новый старт.

- DBWR (DBW*n*; database writer) — выполняет сброс модифицированных буферизованных блоков данных на диск. Должно быть не менее одного такого процесса (несколько их может быть только в Unix), причем наличие двух и более (указывается параметром СУБД DB_WRITER_PROCESSES с обязательной корректировкой DB_BLOCK_LRU_LATCHES) дает возможность асинхронной записи. Запись осуществляется в разных случаях, например:
  - когда поступил сигнал о формировании контрольной точки
  - по истечении положенного времени
  - когда число модифицированных блоков превысило определенный предел (до версии 8 мог явно устанавливаться параметром DB_BLOCK_MAX_DIRTY_TARGET; в версии 9 параметр стал недокументированным; в версии 10 пропал)
  - серверный процесс долго не может найти свободный блок в буфере
  - табличное пространство переводится в OFFLINE, READ ONLY или BEGIN BACKUP; удаляется (DROP) объект из БД и в некоторых других случаях
- LGWR (log writer) — осуществляет запись информации из буфера журналов в одном из следующих случаев:
  - поступила команда COMMIT
  - LGWR простоял без работы 3 секунды
  - буфер заполнился на одну треть (по умолчанию; пропорцию можно изменить недокументированным параметром) или же на 1 Мб
  - DBWR производит сброс модифицированных блоков на диск.
- SMON (system monitor) — выполняет ряд служебных функций, например:
  - восстановление (подкат) данных БД при открытии, последовавшем за аварийным остановом СУБД
  - чистку неиспользуемых сегментов временных данных (пространство TEMPORARY)
  - сращивание свободных сегментов в централизованно управляемых табличных пространствах
- PMON (process monitor) — выполняет ряд служебных функций, например:
  - откат транзакций
  - снятие блокировок и освобождение ресурсов СУБД при нормальном или нештатном завершении сеансов
  - перезапуск отказавших процессов диспетчера или совместных серверных процессов
  - регистрирует запущенную СУБД у процесса listener (версии 11-)
- CKPT (checkpoint) — отвечает за формирование контрольных точек, гарантирующих фиксацию в БД всех изменений данных, сделанных в блоках в SGA. Саму запись изменений CKPT поручает процессам DBW*n* и LGWR, а самостоятельно проставляет контрольную точку в журнальном файле, отталкиваясь от которой СУБД начнет восстановление данных (подкат) при открытии БД после аварийного останова. Номер SCN, соответствующий новой контрольной точке, CKPT заносит в контрольный файл и файлы с данными.
- RECO (recoverer) — восстанавливает данные после отказов распределенных транзакций.
- MMON (manageability monitor) и MMNL (manageability monitor lite) — выполняют действия для AWR (вычисление метрик и определение переходов через пороговые значения, снятия сведений о загрузке СУБД и проч.) и заносят по мере необходимости данные ASH на диск соответственно.
- LREG[12-) (listener registration) — при запуске СУБД передает процессу *listener* данные о СУБД и процессах-диспетчерах.

[12-) Начиная с версии 12.

9.2.2. Дополнительные фоновые процессы

Добавляются в СУБД вследствие определенных значений параметров СУБД или же возникают по мере надобности и пропадают автоматически. Некоторые примеры:

- ARCH (ARC*n*) — переписывает содержимое журнальных файлов в архив. Возникает при LOG_ARCHIVE_START = TRUE или же при переведении БД в режим ARCHIVELOG[10-).
- LCK*n* — выполняют взаимные блокировки между экземплярами в кластерной конфигурации RAC.
- P*nnn* — подчиненные (slave) процессы при параллельной обработке запроса.
- QMN*n* — процессы управления очередями («дополнительная» поддержка очередей в Oracle8).
- SNP*n*[-8] — процессы для планового запуска заданий, возникающий при JOB_QUEUE_PROCESSES > 0.
- CJQ0[9-) — координирующий процесс для планового запуска заданий, возникающий при JOB_QUEUE_PROCESSES > 0.
  - J*nnn*[9-) — процессы, фактически запускающие плановые задания; создаются и удаляются по необходимости (динамически) процессом CJQ0.
- C*nnn*[9-) и A*nnn*[9-) — динамические процессы, осуществляющие захват изменений и применение в потоках Oracle Streams.
- RBAL[10-) — координирующий процесс для динамической балансировки загрузки дисков при использовании автоматического управления дисковым пространством (ASM).
  - ARB*n*[10-) — процессы, фактически выполняющие балансировку.
- MMAN[10-) — процесс для автоматического (не ручного) управления общей памятью SGA в СУБД, возникающие при SGA_TARGET > 0 и STATISTICS_LEVEL <> BASIC.
- CTWR[10-) — заносит в поразрядный файл измененных блоков сведения об измененных блоках, если БД переведена в режим работы BLOCK CHANGE TRACKING (используется при резервировании изменений в БД с помощью программы *RMAN*).
- другие процессы.

[-8] Существовал до версии 8 включительно.
[9-) Начиная с версии 9.
[10-) Начиная с версии 10.

9.2.3. Серверные процессы

Выполняют в СУБД действия от имени сеансов пользователей: обработку предложений SQL, подчитывание блоков данных и прочее. В буквальном преводе — «процессы-служители», «процессы-исполнители» (запросов к СУБД).

Два вида серверных процессов включают в себя:

- Назначенные (dedicated) серверные процессы. Их области PGA содержат данные сеанса и состояние курсора запроса SQL. Со стороны клиентской программы непосредственно с ними взаимодействуют *процессы пользователя* (user processes).
- Совместные (shared) серверные процессы. Их области PGA свободны от данных сеанса и курсора запроса SQL, размещаемых в этом случае в SGA. Процесс пользователя непосредственно взаимодействует с процессом-диспетчером, а тот поручает обработку очередного запроса SQL свободному совместному серверному процессу.

Используются следующие обозначения:

- S*nnn* — совместные серверные процессы для обработки поступающих от клиентов запросов.
- D*nnn* — диспетчерские процессы, выбирающие конкретный совместный серверный процесс для обработки конкретного запроса, и отсылающие клиенту ответ.

9.2.4. Просмотр имеющихся в составе СУБД процессов

9.2.4.1. Просмотр в Oracle

Перечень имеющихся процессов:

```
SELECT * FROM v$process;
```

Перечень имеющихся фоновых процессов:

```
SELECT * FROM v$bgprocess WHERE LENGTH ( paddr ) >= 8;
```

(Если убрать фильтрацию строк фразой WHERE — перечень *всех возможных* для данной версии СУБД фоновых процессов).

Более удобный вид сведений дает комбинация этих двух таблиц, например:

```
COLUMN name     FORMAT A10
COLUMN program  FORMAT A20
COLUMN terminal FORMAT A15
COLUMN pid      FORMAT 99999

SELECT p.pid, bg.name, p.terminal, p.program, bg.description
FROM   v$process p LEFT OUTER JOIN v$bgprocess bg
ON     ( p.addr = bg.paddr )
;
```

Пояснение. Таблица V$PROCESS отличается от таблицы V$SESSION тем, что содержит сведения о процессах, физически реализующих сеансы; в общем случае соответствие между сеансами и процессами не является соотношением один-к-одному. Например, следующий запрос даст список процессов, вообще-то обслуживающих в сеансы пользователей, но простаивающих в данный момент:

```
SELECT
    pid
  , program
  , pga_used_mem
  , pga_alloc_mem
  , pga_freeable_mem
  , pga_max_mem
FROM
    v$process
WHERE
    addr NOT IN ( SELECT paddr FROM v$session )
AND spid IS NOT NULL
;
```

9.2.4.2. Просмотр и использование в ОС

В OELinux посмотреть перечень имеющихся процессов Oracle можно, выдав:

```
$ps –ef | grep "ora_" | grep -v grep
```

Обращение в ОС к обязательному (стандартному) фоновому процессу позволяет узнать о запущеных экземплярах СУБД и ASM:

```
$prep –lf pmon
```

В Windows для просмотра нитей вычислений (потоков, threads) ОС, соответствующих процессам Oracle, можно использовать Task Manager и Performance Monitor. В командной строке информацию о СУБД в целом (без данных о потоках) можно узнать с помощью программы *tasklist.exe*, например:

```
>tasklist /fi "imagename eq oracle.exe"
```

Столбец V$PROCESS.SPID содержит для Unix номер процесса, а для Windows номер нити вычислений ОС — технически реализующих конкретный процесс Oracle. Знание этого номера помогает удалить серверный процесс, даже когда сеанс пользователя формально уничтожен командой ALTER SYSTEM KILL SESSION ..., но долго не исчезает из списка в V$SESSION будучи помеченым как KILLED. Для этого в Windows можно воспользоваться программой ПО Oracle *orakill*:

```
>orakill имя_СУБД SPID_то_есть_номер_нити_Windows
```

В Unix/Linux аналогичное действие выполняется командой ОС *kill*:

```
$kill -9 SPID_то_есть_имя_процесса_Unix
```

Упражнение. Подготовить SQL-запрос, выдающий для сеансов, помеченных в столбце V$SESSION.STATUS как KILLED, команды удаления процессов для этих сеансов с помощью программы *orakill* или комадны *kill*.

9.3. Структуры данных в составе экземпляра СУБД

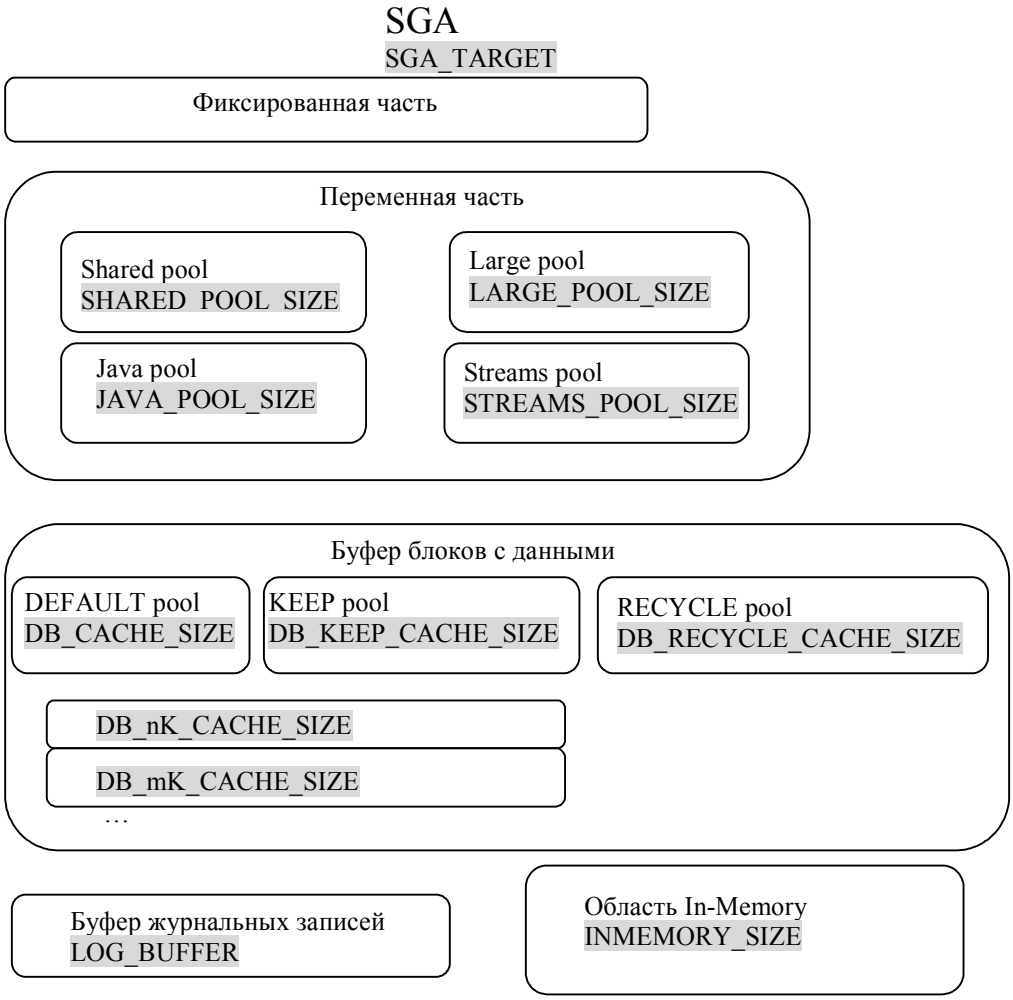
9.3.1. Область SGA

Область общих данных СУБД, доступных для всех процессов СУБД. Часть компонент SGA администратором не регулируется, а часть регулируется (настраивается) с помощью параметров СУБД. Компоненты SGA:

- Фиксированная часть [не регулируется пользователем]
- «Переменная часть», источники для динамического выделения памяти по мере надобности процессов СУБД (обработка запросов, потоки данных и пр.) по типу «кучи»; общий запас памяти под возникающие текущие нужды:
  - Совместный запас (источник) памяти для разных нужд всех процессов СУБД (shared pool)[7-):
    - Постоянная область[*10-)
    - Область динамического выделения участков памяти (собственно куча)
  - Запас памяти для больших участков в некоторых случаях (large pool)[8.0-):
  - Запас участков памяти под нужды встроенной виртуальной машины Java, JVM (java pool)[8.1-)
  - Запас участков памяти для организации потоков данных (streams pool)[10-)*
- Буферная область блоков данных в БД (buffer cache)
- Буферная область журнала (log buffer)
- Область данных in-memory[12.2-)

[7-) Начиная с версии 7.
[8.0-) Начиная с версии 8.0.
[8.1-) Начиная с версии 8.1.
[10-) Начиная с версии 10.
[*10-) Начиная с версии 10 учитывается величиной параметра СУБД SHARED_POOL_SIZE, а до этого не учитывалась.
[12.2-) Начиная с версии 12.2.

Их изображание в виде рисунка с указанием в скобках соответствующих регулирующих параметров СУБД:



Область shared pool переменной части присутствует в SGA всегда. Остальные области переменной части нужны в меру необходимости и носят специализированный характер.

Фактическое распределение памяти в SGA можно наблюдать в таблицах VSGA, VSGASTAT и V$SGAINFO (малозатратная «таблица» = представление; с версии 10). Примеры:

```
SELECT * FROM v$sga;
--  общая структура SGA
SELECT pool, bytes FROM v$sgastat WHERE name = 'free memory';
--  насколько в данный момент недоиспользуются участки памяти в переменной части SGA
```

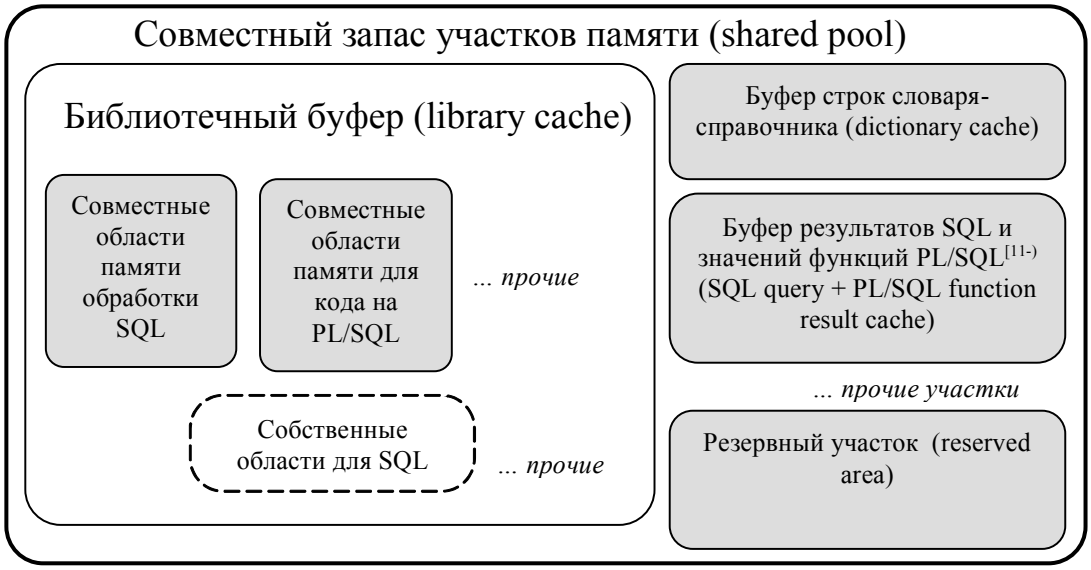
Области shared pool и large pool в целом управляются механизмом LRU, так что малоиспользуемые участки памяти, выделяемые в них, со временем могут этим механизмом передаваться под новые потребности («устаревать»; recreatable chunks), в результате чего повторные операции вынуждены будут заводить их заново и выполняться медленнее ожидаемого.

В целях отладки, проверочных и сравнительных испытаний администратор может «почистить» область shared pool, а также сбросить все измененные блоки с данными в базу, соответственно командами:

```
ALTER SYSTEM FLUSH SHARED_POOL;
ALTER SYSTEM FLUSH BUFFER_CACHE;
```

9.3.1.1. Область shared pool

Некоторые детали устройства «общего запаса» памяти для процессов СУБД:



[11-) С версии 11.
(*) Личные области для обработки запросов SQL, принадлежащие сеансом, попадают в shared pool в случаях, когда сеансы соединены с СУБД техникой совместных (shared) серверных процессов.

Примеры общих структур в shared pool:

- заголовки блоков данных
- массивы параметров процессов, сеансов, транзакций
- массивы очередей и захватов
- области для сбора статистики хранения объектов в БД
- другие

9.3.1.1.1. Библиотечный буфер

Основные динамические подобласти: область общих служебных участков памяти для обработки запросов SQL; область общих служебных участков памяти для выполнения программных единиц PL/SQL (неименованные блоки, хранимые процедуры и функции, пакеты, тела триггеров); несколько других.

Показатели статистики использования элементов (объектов) в библиотечном буфере отражают следующие процессы:

- *Get:* запрос данных о конкретном объекте. Если данных не оказалось, они будут заведены: будет выделена память для «заголовка» и для «тела» объекта.
- *Pin*: чтение «тела» размещенного в буфере объекта. Для SQL или PL/SQL соответствует исполнению запроса или программной единицы. Может потребовать повторного выделения памяти (RELOAD).
- *Reload*: повторное выделение памяти под «тело» объекта, для SQL, сопровождающееся повторным разбором. Происходит, когда память «тела» объекта оказалась передана под нужды другого объекта, или же она оказалась помечена как INVALIDATE.
- *Invalidation*: пометка участков памяти объекта как недействительных из-за совершения изменений в зависимых объектах (перетрансляция PL/SQL, пересчет статистики или изменение структуры таблицы и др.).

Эти характеристики для разных категорий объектов библиотечного буфера (SQL AREA, TABLE/PROCEDURE, BODY, TRIGGER и проч.) имеются в таблице V$LIBRARYCACHE.

Упражнение. Посмотреть характеристики использования библиотечного буфера в СУБД учебной БД. Категорию объекта см. в столбце NAMESPACE таблицы V$LIBRARYCACHE. Понаблюдать изменение характеристик при повторных выдачах составленного запроса.

Прочие таблицы с информацией об использовании библиотечного буфера:

V$LIBRARY_CACHE_MEMORY
V$DB_OBJECT_CACHE
V$SQL%

9.3.1.1.2. Буфер строк словаря-справочника

Содержит строки словаря-справочника БД с описанием объектов БД. «Дополнительный буфер» (для ускорения доступа) между описаниями объектов в библиотечном буфере и словарем-справочником в БД.

Основные понятия:

- *Get*: акт обращения процессом СУБД за информацией об объекте.
- *Getmiss*: акт обращения за информацией об объекте, при котором обнаружилось, что ее в буфере строк нет и ее потребовалось завести.

Статистика значений этих характеристик для разных категорий объектов буфера строк имеется в таблице V$ROWCACHE.

Упражнение. Посмотреть указанные характеристики использования буфера строк в СУБД учебной БД. Категорию объекта см. в столбце PARAMETER таблицы V$ROWCACHE.

9.3.1.1.3. Резервный участок

Иногда от процесса приходит запрос на выделение крупного куска памяти — например, для трансляции программной единицы на PL/SQL, размещения параметров сеанса или загрузки объектов Java. Резервный участок (reserved area) внутри shared pool создан специально для возможности выделять крупные куски памяти единым целым, а не серией мелких фрагментов. Роль этого участка памяти значительно снижается при установленном достаточном размере области large pool.

По умолчанию он занимает в разных версиях 10% или 5% от размера shared pool.

Статистика использования резервного участка собирается в V$SHARED_POOL_RESERVED.

9.3.1.1.4. Буфер результатов запросов SQL и функций на PL/SQL

Введен в версии 11 для ускорения выдачи СУБД ответов на запросы, когда это возможно. Максимальный размер буфера назначается параметром СУБД RESULT_CACHE_MAX_SIZE. Таблицы со сведениями о содержании и статистикой использования буфера:

V$RESULT_CACHE_MEMORY
V$RESULT_CACHE_OBJECTS
V$RESULT_CACHE_STATISTICS
V$RESULT_CACHE_DEPENDENCY

9.3.1.2. Область large pool

Запасом участков памяти больших размеров, large pool, администратор может управлять через параметр СУБД LARGE_POOL_SIZE. В случае достаточного значения LARGE_POOL_SIZE, СУБД будет в large pool, а не в shared pool размещать большие участки памяти под рабочие структуры, возникающие в следующих случаях:

1) при соединении клиентов с СУБД методом совместных серверных процессов (shared server mode) и при использовании монитора транзакций — для данных сеансов клиентов;
2) в операциях асинхронного ввода/вывода при резервировании и восстановлении данных с помощью RMAN, если используются параметры СУБД DBWR_IO_SLAVES и BACKUP_TAPE_IO_SLAVES (иначе расходуется память процессов, выполняющих операции для RMAN);
3) при параллельной обработке запросов.

Это позволяет снизить динамическую нагрузку на shared pool, повысив производительность этой области. О нехватке места в large pool могут свидетельствовать записи в *ALERT.LOG* о недостаточности памяти для запуска подчиненных процессов ввода/вывода.

9.3.1.3. Буферная область для блоков БД

Буферная область для доступа СУБД к блокам данных из БД (buffer cache, «наличка буферных блоков»). Содержит только сами буферизованные («обналиченные») блоки. Управляющая информация о блоках (например, заголовки) помещается в переменную часть SGA. Схемы буферизации и использования блоков в целом основываются на применении LRU-списка.

До версии 8 буфер блоков логически был однородным пространством размером DB_BLOCK_BUFFERS блоков одинаковой длины DB_BLOCK_SIZE байтов (параметры СУБД).

С версии 8 может иметь три раздела: DEFAULT (обязательный), KEEP и RECYCLE (оба дополнительные). Схема буферизации блоков во всех трех разделах одинаковая, но предполагается, что программист или АБД будет приписывать к разделу KEEP буферизацию блоков объектов с повышенными требованиями к скорости обращения, к RECYCLE — буферизацию объектов с редкими обращениями к ним, и к DEFAULT — буферизацию блоков объектов, характер доступа приложений к которым неочевиден или неважен.

С появлением в версии 9 понятия «основного размера блока» (размер блока табличных пространств SYSTEM и типов UNDO и TEMPORARY, указываемый при заведении БД), возможность заведения разделов KEEP и RECYCLE не распространилась на пространства с нестандартным в БД размером блока. В свою очередь создание пространства с нестандартным для БД размером блока *n* Кб требует предварительно обеспеченного наличия дополнительного участка («слоя») буферизации, размер которого сообщается параметром СУБД DB_*n*K_CACHE_SIZE.

Некоторые возможные характеристики блока в буфере:

- *free/unused*: на этом месте в буфере ничего нет (например, после запуска СУБД)
- *pinned*: блок используется процессом СУБД в данный момент
- *clean*: блок сейчас не используется и является кандидатом на удаление, так как либо он синхронизирован с блоком в БД, либо он служил копией для отката, сделанной перед изменением записей в БД
- *dirty*: блок сейчас не используется, но был изменен в буфере, поэтому является кандидатом на запись в БД процессом DBWR.

Основной таблицей для просмотра содержимого буфера блоков является V$BH. Пример:

```
COLUMN owner       FORMAT A30
COLUMN object_type FORMAT A30
BREAK ON owner

SELECT
  owner
, object_type
, dirty
, COUNT ( * ) AS blocks
FROM     v$bh, dba_objects
WHERE    v$bh.objd = dba_objects.data_object_id
GROUP BY owner, object_type, dirty
ORDER BY 1, 2, 3
;
```

(В названии таблицы BH означает block headers, то есть фактическим ее содержимым является список *заголовков* блоков, буферизованных в buffer cache.)

Прочие таблицы диагностики работы буфера блоков:

V$BUFFER_POOL_STATISTICS

V$BUFFER_POOL
V$SESSTAT
V$SYSTEM_EVENT
V$SESSION_WAIT

9.3.2. Область PGA

«Глобальная» область *процесса* — область памяти, «глобальная» для конкретного процесса, а не всех процессов СУБД. Возникает вместе с процессом и освобождается по завершению процесса. Содержит следующее:

- Собственные данные процесса, как-то: магазин («стек») вызовов подпрограмм.
- Память курсоров, открытых прикладной программой средствами OCI на время обработки запроса.
- Рабочие области (work area) обработки запросов для нужд сортировок, хеширования, выполнения обобщающих запросов, поразрядного индекса (создание и слияние), операций массовой загрузки данных.
- Данные сеанса, если это назначенный серверный процесс (dedicated).

Постоянная часть PGA имеет размер, столь незначительно контролируемый некоторыми из параметров СУБД, что практически его можно принимать зависимым только от версии СУБД.

Динамическая часть PGA может изменяться за счет выделения по мере надобности и освобождения памяти под рабочие области для обрабатываемых запросов. Операции на рабочих областях будут выполняться в один проход (*one-pass*), если размер PGA достаточен для размещения всех данных, используемых в операции, и в несколько проходов (*multi-pass*) с использованием дисковой подкачки, если размер PGA недостаточен.

Сведения о размерах можно получить из таблицы V$PROCESS, а в связи с сеансами пользователей — из V$SESSTAT, например:

```
COLUMN name FORMAT A25
BREAK ON sid

SELECT
  s.sid
, n.name
, MAX ( s.value ) AS maxmem
FROM
  v$sesstat  s
, v$statname n
WHERE
         n.statistic# = s.statistic#
  AND    n.name IN ( 'session pga memory', 'session pga memory max' )
GROUP BY s.sid, n.name
ORDER BY 1, 2
;
```

Таблицы диагностики качества использования областей PGA:

V$PGASTAT
V$SQL_WORKAREA
V$SQL_WORKAREA_ACTIVE
V$SQL_WORKAREA_HISTOGRAM
V$TEMPSEG_USAGE

9.3.3. Область UGA

Содержит:

- Собственные данные *сеанса*, как то: данные о пакетах и данные пакетов, состояние сеанса Java, сведения о состоянии приписанных пользователю ролей, параметры глобализации (NLS), открытые связи с посторонними БД (database links), метки доступа для Label Security и пр.

Содержит также две части: постоянную и динамическую (кучу). Постоянная часть UGA имеет размер, столь незначительно контролируемый некоторыми из параметров СУБД, что практически его можно принимать зависимым только от версии СУБД.

Будучи *логическим* понятием, UGA *физически* может располагать свои данные в PGA (при обслуживании пользовательского сеанса назначенным серверным процессом) или в SGA (при обслуживании пользовательского сеанса совместными серверными процессами). В SGA местом размещения может служить shared pool, либо large pool.

Сведения о размерах UGA сеансов связи с Oracle можно получить из таблицы V$SESSTAT, например:

```
COLUMN name FORMAT A25
BREAK ON sid

SELECT
  s.sid
, n.name
, MAX( s.value ) AS maxmem
FROM
  v$sesstat  s
, v$statname n
WHERE
         n.statistic# = s.statistic#
  AND    n.name IN ( 'session uga memory', 'session uga memory max' )
GROUP BY s.sid, n.name
ORDER BY 1, 2
;
```

Упражнение. Создать новое соединение с СУБД (новый сеанс) и посмотреть вызванное им изменение списка имеющихся областей PGA и UGA.

9.4. Схемы выполнения некоторых внутренних процедур

9.4.1. Журнализация изменений в БД

Журнальный буфер в SGA «закольцован», и заполняется «по кругу», но при этом СУБД не допускает затирания «хвоста», так что при чрезмерно интенсивном потоке изменений процесс LGWR может не успевать сбрасывать буфер на диск и возможны его простои, то есть задержки в работе.

Журнальная информация формируется серверными процессами индивидуально для своих сеансов и сбрасывается время от времени в общий буфер. Это происходит в два этапа:

(1) серверные процессы запрашивают место (redo allocation), а затем
(2) копируют туда свои данные (redo writing).

Оба этапа в состоянии порождать внутренние ожидания доступа.

Если буфер блоков данных велик, а буфер журнала мал, DBWR может не успевать сбрасывать в БД измененные блоки, и LGWR вынужден будет пережидать, пока сумеет занести новые журнальные записи в файл (ожидания log buffer space; общее время — показатель статистики СУБД redo log space wait time).

Показатель статистики redo log space wait time можно наблюдать в таблицах V$SESSTAT, V$SYSSTAT. Показатель ожидания log buffer space можно наблюдать в таблицах V$SESSION_EVENT, V$SESSION_WAIT,

V$SYSTEM_EVENT. Статистику защелок redo allocation и redo writing можно наблюдать в V$LATCH. Все перечисленные характеристики приводятся в отчетах OEM.

Если включен режим архивирования журнальных файлов и архивирование происходит медленно, LGWR автоматически добавит процессов ARC*n* до числа, не более LOG_ARCHIVE_MAX_PROCESSES (параметр СУБД).

9.4.2. Формирование (продвижение) контрольной точки

Понятие «контрольная точка» используется в двух смыслах:

- для обозначения в ленте журнальных записей позиции (фактически SCN), ранее которой встречаются только записи об изменениях, результаты которых попали из SGA в файлы с данными;
- для обозначения процедуры занесения данных из SGA в файлы БД, определения нового места такой позиции и занесения SCN в контрольный файл и файлы с данными.

Контрольные точки формируются процессом CKPT при разных обстоятельствах и бывают разных типов:

1. Полные (*full*). Из SGA на диск сбрасываются *все* измененные блоки. Формируются при выдаче ALTER SYSTEM CHECKPOINT, ALTER DATABASE BEGIN BACKUP, ALTER DATABASE CLOSE, SHUTDOWN.
2. Нитевые (потоковые, *thread*). Из SGA на диск сбрасываются только блоки измененные конкретным экземпляром СУБД (в конфигурации RAC). Формируются при выдаче ALTER SYSTEM CHECKPOINT LOCAL.
3. Инкрементальные (*incremental*). Каждые 3 секунды СУБД пытается сбрасывать на диск *некоторые* измененные блоки, количество которых определяется из расчета текущих значений параметров СУБД, управляющих журнализацией (FAST_START_MTTR_TARGET и др.).
4. Файловые (*file*). На диск сбрасываются только измененные блоки, относящиеся к файлам конкретного табличного пространства. Формируются при выдаче ALTER TABLESPACE … OFFLINE, ALTER TABLESPACE … BEGIN BACKUP, ALTER TABLESPACE … READ ONLY.
5. Параллельных запросов (*parallel query*). На диск сбрасываются только измененные блоки объектов — источников данных для запроса с параллельным планом обработки.
6. Объекта (*object*). На диск сбрасываются только измененные блоки с данными конкретного объекта. Формируются при выдаче DROP/TRUNCATE TABLE, DROP INDEX. Нужны для операций восстановления данных.
7. Переключения журналов (*log switch*). На диск сбрасываются только некоторык измененные блоки.

В случаях 1, 2, 4 SCN новой контрольной точки прописывается в контрольный файл и файлы с данными (столбец CHECKPOINT_CHANGE# в V$DATABASE и V$DATAFILE). Во всех случаях значение новой контрольной точки в форме RBA (redo byte address) заносится в SGA.

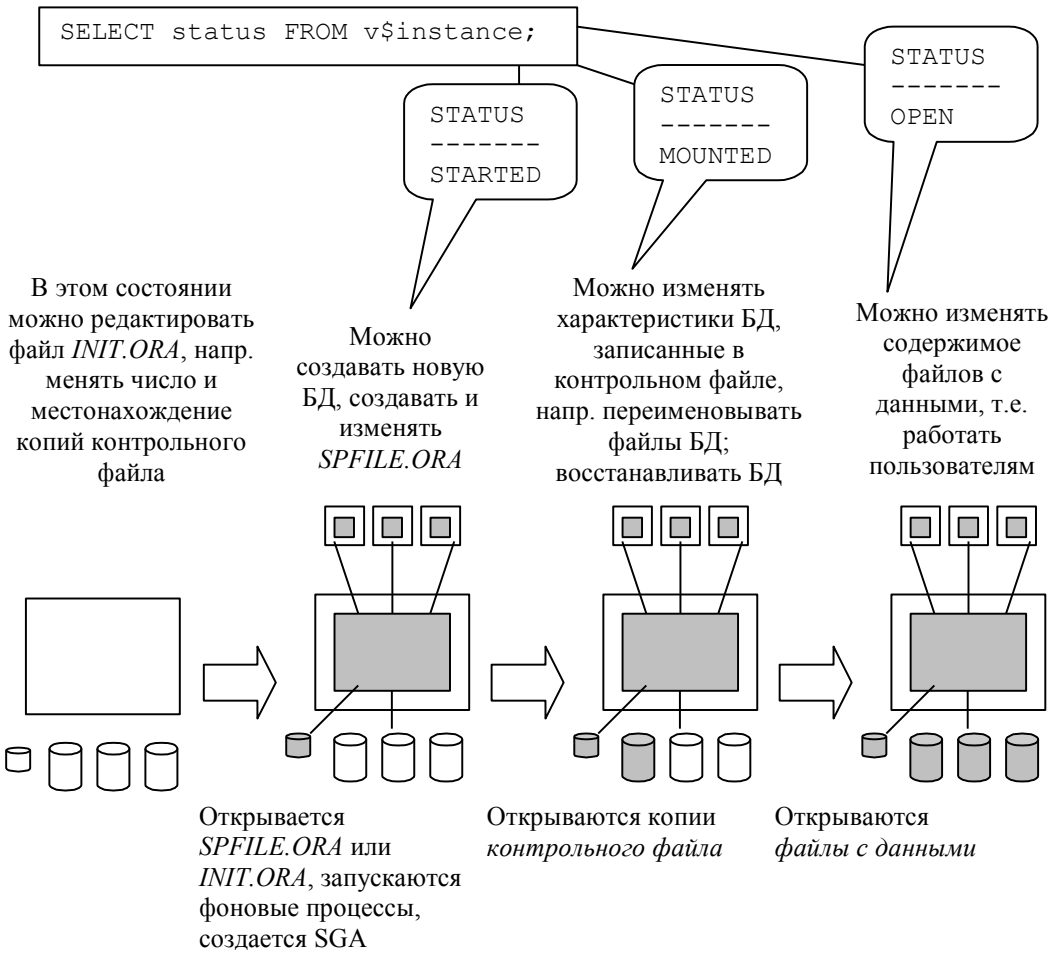
Значение параметра СУБД FAST_START_MTTR_TARGET — главный фактор влияния на интенсивность сброса образов блоков из SGA в файлы БД при формировании инкрементальных (пошаговых) контрольных точек. Сам же этот тип контрольных точек был введен в версии 8 для задания максимально допустимого времени автоматического восстановления данных, потерянных в результате аварийного останова СУБД.

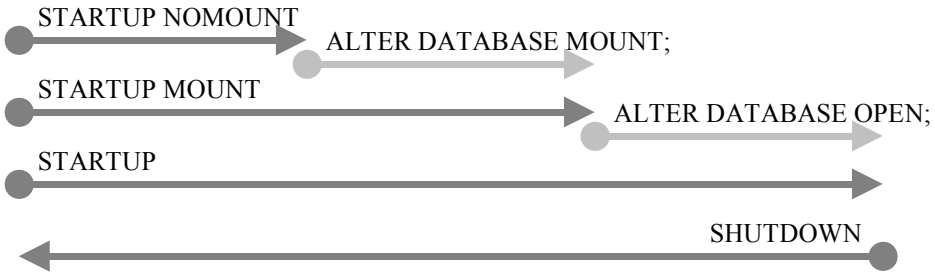
Статистику выполнения контрольных точек и оценку времени восстановления можно получить из V$INSTANCE_RECOVERY.

9.5. Состояния базы данных в Oracle

База данных Oracle может находиться в одном из трех (кроме выключенного) состояний, которым соответствует один из вариантов выдачи команды STARTUP. (Команды STARTUP и SHUTDOWN выдаются в SQL*Plus и не являются командами SQL).

Состояния БД поясняются следующим рисунком:





Варианты выдачи STARTUP:

STARTUP … **FORCE** …	SHUTDOWN ABORT, затем автоматический STARTUP.
STARTUP … **RESTRICT** …	Запуск с возможностью работать только администраторам.
STARTUP … **PFILE** = *INIT_файл*	Запуск СУБД по явно указанному файлу *INIT.ORA*.

Ввиду того, что с версии 9 имеются два разные файла параметров для запуска СУБД, появились и две соответствующие команды:

```
CREATE PFILE [= 'INIT_файл'] FROM {SPFILE [= 'SPFILE_файл'] | MEMORY[11-]};

CREATE SPFILE [= 'SPFILE_файл'] FROM {PFILE [= 'INIT_файл'] | MEMORY[11-]};
```

[11-] С версии 11.

При запуске СУБД в отсутствии варианта STARTUP PFILE = … поиск файла с параметрами осуществляется в следующем порядке до первого обнаружения:

1. в ASM, если БД создавалась с указанием ASM как основного места хранения файлов;
2. в *%ORACLE_HOME%\database* или *$ORACLE_HOME/dbs* ищется *spfile*ORACLE_SID.*ora*;
3. там же ищется *spfile.ora*;
4. там же ищется *init*ORACLE_SID.*ora*.

Пункты 2 и 3 различаются ради кластерной БД (типа RAC).

Использования текстового файла *INIT.ORA* для решения насущных задач можно почти избежать. Например, если СУБД была запущена по *SPFILE.ORA*, то для правки статического параметра СУБД можно выполнить две команды:

```
SQL> ALTER SYSTEM SET имя_параметра = новое_значение SCOPE = SPFILE;
SQL> STARTUP FORCE
```

Однако совсем от *INIT.ORA* лучше не отказываться: например, перед выдачей (как выше) команды ALTER SYSTEM SET … SCOPE = SPFILE имеет смысл подстраховаться командой `CREATE PFILE [...] FROM SPFILE;`. Переключиться временно на *INIT.ORA* удобно также при необходимости установить сразу несколько параметров запуска СУБД.

Переход из состояния STARTED (NOMOUNT) в состояние MOUNT выполняется командой SQL:

```
ALTER DATABASE MOUNT;
```

Переход из состояния MOUNT в состояние OPEN выполняется командой SQL:

```
ALTER DATABASE OPEN [READ ONLY];
```

Команда SHUTDOWN отрабатывает обратные к STARTUP действия в обратном порядке. Варианты выдачи:

SHUTDOWN [**NORMAL**]	Ждать пока последний пользователь не закроет сеанс. NORMAL можно не указывать.
SHUTDOWN **IMMEDIATE**	Всем работающим — ROLLBACK и разрыв сеанса.
SHUTDOWN **TRANSACTIONAL [LOCAL]**	Ждать, когда пользователь закончит транзакцию, и после разрыв сеанса. Если указано LOCAL, ждать конца только локальных транзакций.
SHUTDOWN **ABORT**	Бросить файлы в их текущем (= рассогласованом) состоянии и СУБД покинуть оперативную память.

Замечание. В Windows выдача SHUTDOWN ABORT не освобождает память процессов, поэтому если во время такого останова имелось много сеансов пользователей, то для предотвращения неконтролируемых затрат виртуальной памяти желательно перезапустить службу ОС для СУБД (*oradim.exe* или же окошко служб Windows).

Для запущенной СУБД возможны два особых состояния: RESTRICTED и QUIESCE. Переход в состояние RESTRICTED SESSION прекратит открытие новых сеансов пользователей, не обладающих одноименной привилегией RESTRICTED SESSION, но не закроет уже открытые. Примеры перевода в это состояние и обратно:

```
ALTER SYSTEM ENABLE RESTRICTED SESSION;
ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

Состояние QUIESCE прекратит открытие новых сеансов для пользователей, отличных от SYS и SYSTEM, и переведет уже открытые в «подвешеное» состояние (но не закроет их). Примеры перевода в это состояние и обратно:

```
ALTER SYSTEM QUIESCE RESTRICTED;
ALTER SYSTEM UNQUIESCE;
```

10. Настройка экземпляра СУБД Oracle

Настройка экземпляра СУБД в оперативной памяти — процесс итеративный. Администратор следит за эффективностью работы СУБД, при необходимости вносит изменения в конфигурацию СУБД, снова наблюдает, снова вносит изменения и т. д.

Основой для наблюдения преимущественно являются данные из системных таблиц (главным образом V$-таблиц). Обращение к системным таблицам может выполняться АБД прямыми запросами SQL, либо через посредство Statspack, либо через AWR: напрямую или же посредством OEM.

Наиболее действенным объектом настройки СУБД является (пере)распределение основных участков памяти. Многочисленные другие объекты влияния (защелки, повторяемость запросов и т. д.) следует принимать во внимание только если перераспределение участков памяти исчерпало себя и не дает выигрыша.

Начиная с версии 9 фирма Oracle постоянно привносит в технологию настройки средства автоматизации, конечной целью которых является полная самонастройка СУБД.

10.1. Ручная настройка (для всех версий)

Выполняется вручную, по результатам наблюдения за работой СУБД. Ручным образом можно настраивать работу СУБД во всех версиях, однако на практике этот способ в полной мере применяется преимущественно для версий 8 и предыдущих.

10.1.1. Методики настройки

Имеется две разные методики настройки, соответствующие выбору двух разных категорий объектов для наблюдения:

- «Метод коэффициентов», когда администратор следит за набором «индикаторов»: определенных показателей, которые в эффективно работающей СУБД должны быть (по мнению документации Oracle) «достаточно хорошими». Пример такого показателя — отношение числа физических чтений блока данных к общему числу обращений к блокам процессов.
- «Метод анализа статистики ожиданий», когда объектом внимания администратора становятся внутренние задержки, то есть простои процессов СУБД, замедляющие работу последней. Выявляются наиболее крупные задержки, изучаются причины их возникновения и предпринимаются действия по устранению.

Первая методика более проста в использовании, зато вторая — более точна. На практике разумно сочетать обе методики.

Далее в тексте приводится ряд «индикаторов» эффективного (по мнению разработчиков Oracle) функционирования СУБД.

10.1.2. Настройка SGA

10.1.2.1. Первоначальное распределение памяти

В виду итерационной природы настройки СУБД за начальные значения параметров SGA можно взять любые работоспособные. При этом можно отталкиваться от собственного опыта, или довериться предложениям DBCA — когда БД создавалась через эту программную компоненту ПО Oracle.

10.1.2.2. Закрепление и формирование в оперативной памяти

Область SGA Oracle можно прикрепить к ОЗУ, чтобы исключить ее из межстраничного обмена(swap) и гарантировать отсутствие задержек при обращении к ней. Для этого нужно выставить параметры СУБД:

- LOCK_SGA = TRUE (HP-UX и некоторые другие платформы с версии 8.0 или более поздней; например, в Windows с версии 10). Действует не во всех ОС. Если устанавливается через *SPFILE.ORA*, обязательно нужно подстраховаться и создать копию файла параметров.

- LOCK_SGA_AREAS действовал до версии 8.0 включительно.
- USE_ISM = TRUE (Solaris; с версии 8.1 скрытый параметр).

Следующий параметр формирует все страницы памяти SGA во время запуска СУБД, а не по мере работы:

- PRE_PAGE_SGA

Его использование замедляет запуск СУБД, но зато она сразу переходит в рабочий режим использования страниц памяти. В Windows этот параметр может сочетаться с установками реестра в *HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE* или в *HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE*[*HOME_KEY*]:

- *ORA_WORKINGSETMIN* или *ORA_%SID%_WORKINGSETMIN* — минимальный объем «набора памяти» для процесса *oracle.exe* в мегабайтах.
- *ORA_WORKINGSETMAX* или *ORA_%SID%_WORKINGSETMAX* — максимальный объем «набора памяти» для процесса *oracle.exe* в мегабайтах.

10.1.2.3. Настройка области shared pool

С помощью параметра СУБД SHARED_POOL_SIZE можно вручную указать примерный размер общего пространства SGA, выделяемого в совместной области для динамического размещения данных (а не общий размер совместной области). Размер постоянной части совместной области определяется в момент старта системы на основе ряда разных параметров СУБД (например, числа PROCESSES).

Некоторые общие задачи по настройке shared pool:

- снижение динамический нагрузки по выделению и освобождению памяти в этой области
- снижение фрагментированности shared pool
- снижение конкуренции за использование shared pool разными процессами

Некоторые общие пути решения:

- повышение степени повторного использования объектов в shared pool
- прикрепление больших объектов к библиотечному буферу
- увеличение размера shared pool

При отсутствии конкретных сведений за отправную величину SHARED_POOL_SIZE можно взять 40% SGA. Увеличение размера shared pool бессмыслено, если там есть свободное место:

```
SELECT bytes
FROM   v$sgastat
WHERE  name = 'free memory' AND pool = 'shared pool'
;
```

10.1.2.3.1. Библиотечный буфер

Заключение о степени эффективности использования СУБД библиотечного буфера может быть принято на основе слежения за такими показателями, как «степень буферизованности» объектов в буфере и доля повторных выделений оперативной памяти под объекты (после автоматического изъятия этой памяти под другие нужды):

```
SELECT
  SUM ( pins )                    "Executions"
, SUM ( reloads )                 "Misses"
, SUM ( reloads ) / SUM ( pins ) "Reload Ratio"
, SUM ( pins - reloads ) / SUM ( pins ) * 100
                                  "Library Cache Hit Ratio %"
FROM v$librarycache
;
```

Если значение Library Cache Hit Ratio меньше 99% или Reload Ratio больше 1%, то по общей рекомендации:

- в shared pool не хватает места, рабочие участки памяти быстро устаревают и замещаются новыми; нужно увеличить SHARED_POOL_SIZE;
- память для повторяющихся запросов быстро устаревает; нужно предпринять организационные меры, например:

- уменьшить долю полных разборов запросов (hard parse), например, в приложении увеличить долю параметризованных запросов SQL,
- прикрепить накоторые объекты (подпрограммы, курсоры и т.д.) к библиотечному буферу с помощью процедуры KEEP системного пакета DBMS_SHARED_POOL,
- ради некоторых «неприятных» запросов завести индексы.

10.1.2.3.2. Буфер строк словаря-справочника

Заключение о степени эффективности использования СУБД буфера строк может быть принято на основе слежения за показателем «степени буферизованности» объектов в буфере:

```
SELECT
  ( SUM ( gets - getmisses ) ) / SUM ( gets ) * 100
  AS "Dictionary Cache Hit Ratio %"
FROM v$rowcache
;
```

Если значение Dictionary Cache Hit Ratio меньше 85%, то по общей рекомендации

- следует увеличить размер shared pool или же
- предпринять другие действия для повышения эффективности. Например
  - отойти от практики систематического использования публичных синонимов, увеличивающих нагрузку на буфер строк.

10.1.2.3.3. Резервный участок

Параметр СУБД SHARED_POOL_RESERVED_SIZE позволяет изменить умолчательный размер резервного участка внутри shared pool. Поводом могут служить значения в столбцах таблицы V$SHARED_POOL_RESERVED.

Если REQUEST_MISSES = 0, участок может быть чрезмерно велик. Если REQUEST_MISSES растет, а REQUEST_FAILURES нет, участок может быть чересчур мал и СУБД часто упорядочивает списки занятых кусков в shared pool. Если растут оба значения, упорядочение памяти не помогает размещению нужного участка.

В идеале значение Hit % в запросе ниже должно быть близко к 100% (чтобы не усложнять запрос, считается, что СУБД в рабочем режиме и REQUESTS > 0):

```
SELECT
  requests
, TRUNC ( 100 - ( 100 * ( request_misses / requests ) ), 0 ) "Hit %"
, request_failures
, last_miss_size
, last_failure_size
FROM
 V$SHARED_POOL_RESERVED
;
```

Замечание. Ввиду ошибки Oracle 3669074 (см. *metalink.oracle.com*) этот запрос может не дать правильный ответ до версии 10.2, и вместо него придется сделать аналогичный, но очень громоздкий запрос в таблицам X$KSMSPR и X$KGHLU.

10.1.2.4. Настройка буфера блоков данных

10.1.2.4.1. Внутренние ожидания

Запросы к V$SYSTEM_EVENT, V$SESSION_EVENT, V$SESSION_WAIT и V$WAITSTAT позволяют обнаружить среди прочих внутренних ожиданий СУБД ожидания, вызванные использованием буфера блоков. Пример запроса:

```
COLUMN event FORMAT A30

SELECT
  event
, total_waits
, total_timeouts
```

```
, time_waited
, average_wait
FROM v$system_event
;
```

Некоторые виды ожиданий, характерные для буферизации блоков:

- *buffer busy waits*: распространяются на блоки данных, заголовков сегментов отката и данных отката. Свидетельствуют о повышенной конкуренции за доступ к определенным блокам в буфере
- *free buffer waits*: серверный процесс слишком долго ищет свободное место в буфере и не выдержав, дает сигнал процессу DBWR сбросить модифицированные блоки на диск. Примеры причин: медленно осуществляется ввод/вывод (можно перераспределить файлы по дискам, увеличить число процессов DBW*n* или использовать более быстрый диск); малый размер буфера блоков.

10.1.2.4.2. Эффективность буферизации блоков

Заключение о степени эффективности использования СУБД буфера блоков может быть принято на основе слежения за параметром «коэффициент попадания в буфер», то есть доля случаев обнаружения блоков в оперативной памяти по отношению к общему числу обращений ко блокам. Его можно получить примерно так:

```
SELECT
  1 - ( phy.value - lob.value - dir.value ) / ses.value
  AS "Cache Hit Ratio"
FROM
  v$sysstat ses
, v$sysstat lob
, v$sysstat dir
, v$sysstat phy
WHERE
     ses.name = 'session logical reads'
 AND dir.name = 'physical reads direct'
 AND lob.name = 'physical reads direct (lob)'
 AND phy.name = 'physical reads'
;
```

Грубое правило может рекомендовать увеличение размера буфера блоков, если этот коэффициент меньше 80 — 90%. В то же время ценность этого фактора может оказаться вторичной на фоне внутренних простоев СУБД.

Для отдельных разделов буфера обобщенные показатели буферизованности можно получить из V$BUFFER_POOL_STATISTICS:

```
SELECT
  name
, 1 - ( physical_reads / ( db_block_gets + consistent_gets ) )
  AS "Hit Ratio"
FROM
  v$buffer_pool_statistics
WHERE
  db_block_gets + consistent_gets > 0
;
```

10.1.2.4.3. Изменение структуры буфера блоков

В версии 8 и более поздних размеры секций DEFAULT, KEEP и RECYCLE буфера блоков выставляются по-разному. Пример установок для версии 8:

```
DB_BLOCK_BUFFERS = 1000
DB_BLOCK_LRU_LATCHES = 3
BUFFER_POOL_KEEP = (buffers:200,lru_latches:1)
BUFFER_POOL_RECYCLE = (buffers:100,lru_latches:1)
```

10.1.2.4.4. Ускорение обмена с диском

Распараллеливание сброса блоков из ОЗУ на диск может достигаться значениями параметров СУБД:

- DB_WRITER_PROCESSES > 1. Заведет несколько процессов DBW*n*. Имеет смысл, если имеется достаточно большое количество процессоров.
- DISK_ASYNCH_IO = TRUE. Процессы DBW*n* будут сбрасывать блоки на диск асинхронно, если это позволяет ОС.
- DBWR_IO_SLAVES > 0. Количество подчиненных процессов в/в, которые разделят поровну «пачку» блоков, сбрасываемую процессом DBW0 на диск. Имеет смысл, когда DB_WRITER_PROCESSES = 1 и DISK_ASYNCH_IO = FALSE.

Упреждающее чтение блоков возможно для операций «полного» сканирования (таблицы и индекса). Количество блоков устанавливается параметром СУБД DB_FILE_MULTIBLOCK_READ_COUNT. С версии 10, если = 0, то фактический размер выбирается СУБД самостоятельно в зависимости от платформы.

10.1.3. Настройка областей PGA

Общая задача настройки — сделать размеры PGA достаточно большими, чтобы доля операций multi-pass была невысока, однако не чересчур, так как это повлечет риск исчерпания оперативной памяти в определенные моменты времени.

Начальные размеры PGA серверных процессов практически не регулируются, однако есть возможность ограничивать максимальный размер участков памяти, временно включаемых в состав PGA. Для этого используются параметры СУБД

BITMAP_MERGE_AREA_SIZE
CREATE_BITMAP_AREA_SIZE
HASH_AREA_SIZE
SORT_AREA_SIZE
SORT_AREA_RETAINED_SIZE
OPEN_CURSORS

(Кроме этого PGA может расти от создания больших структур данных в программах на PL/SQL).

Параметры *_AREA_SIZE регулируют *рабочие области* (workarea) обрабатываемых серверным процессом запросов SQL. С версии 8 они динамические уровня сеанса.

На Windows при создании процесса (Oracle) = нити вычислений (Windows) резервируется память для стека вызовов, умолчательного размера 1M. Хотя по факту она не будет сразу выдана целиком, зарезервированная одним процессом память уже не может быть выдана другому процессу. При наличии большого числа соединений это может породить избыточно большие запросы к памяти. Если заранее известно, что сеансы связи с СУБД не будут предъявлять больших претензий к оперативной памяти, резервируемый объем можно сократить. Делается это с помощью правки исполняемого модуля Oracle специальной программой *orastack.exe*, но согласовано для всех используемых компонент. Пример сокращения резервируемой памяти до примерно 500K:

```
>orastack oracle.exe 500000
>orastack tnslsnr.exe 500000
>orastack sqlplus.exe 500000
```

(Фирма Oracle отдельно оговаривает, что чрезмерное сокращение резервируемой для процессов памяти, например, до 300K и ниже, провоцирует возникновение ошибки ORA-3113 аварийного прекращения процесса).

10.2. Настройка в версии 9

Версия 9 позволяет использовать все основные возможности ручной настройки версии 8, но дополнила их новшествами по части динамики, автоматики и экспертных оценок. Некоторые такие дополнения раскрываются ниже.

10.2.1. Настройка SGA

10.2.1.1. Динамическое перераспределение памяти

В версии 9 размеры буфера блоков данных, области shared pool и области large pool можно изменять во время работы СУБД. Для этого параметры DB_CACHE_SIZE, DB_*n*K_CACHE_SIZE, SHARED_POOL_SIZE и LARGE_POOL_SIZE сделаны динамическими. Изменения размеров допускаются в пределах параметра SGA_MAX_SIZE (статического), который методически рекомендуется выставить побольше (но в границах физической памяти).

Технически память выделяется непрерывными кусками виртуальной памяти, называемых «гранулами». Размер гранулы может быть разный:

4 Mb для SGA_MAX_SIZE <= 128 Mb,
8 Mb для SGA_MAX_SIZE > 128 Mb на 32-разрядной Windows,
16 Mb для SGA_MAX_SIZE > 128 Mb на большинстве других ОС.

Примеры:

```
ALTER SYSTEM SET DB_4K_CACHE_SIZE = 4m;

ALTER SYSTEM SET LARGE_POOL_SIZE = 10m;
```

Выделения памяти не произойдет, если нет свободной гранулы (заявлено чрезмерное увеличение области или же в текущий момент не завершено освобождение гранулы после предыдущей команды).

10.2.1.2. Нагрузка на блоки данных в буфере

Сегменты с «горячими» блоками (к которым происходят особо частые обращения):

```
WITH objects_statistics_desc AS (
  SELECT
    owner
  , object_name
  , object_type
  , statistic_name
  , SUM ( value )
  FROM v$segment_statistics
  GROUP BY owner, object_name, object_type, statistic_name
  ORDER BY SUM ( value ) DESC
  )
SELECT * FROM objects_statistics_desc WHERE ROWNUM <= 10
;
```

10.2.1.3. Структуризация буфера блоков

С версии 9 используются новые параметры для указания размеров секций KEEP и RECYCLE в буфере блоков основного для БД размера:

```
DB_CACHE_SIZE = 100M
DB_KEEP_CACHE_SIZE = 75M
DB_RECYCLE_CACHE_SIZE = 500K
```

Параметры динамически изменяемые.

10.2.2. Настройка областей PGA

10.2.2.1. Атоматическое управление размерами PGA процессов

В версии 9 появились два новых параметра СУБД, позволяющие отойти от использования параметров *_AREA_SIZE, а вместо этого указать *желаемый общий* объем памяти для PGA серверных процессов. В этом случае фактические значения *_AREA_SIZE процессов (то есть рабочие области для запросов SQL) Oracle будет определять динамически и самостоятельно.

- PGA_AGGREGATE_TARGET. Параметр задает рекомендуемый максимум суммарного объема областей PGA *всех* имеющихся *назначенных* (dedicated) серверных процессов. Значение = 0 автоматически приведет к WORKAREA_SIZE_POLICY = AUTO.
- WORKAREA_SIZE_POLICY. Если = AUTO, вступает в силу автоматическое распределение памяти PGA, а если = MANUAL, максимальные размеры PGA будут регулироваться значениями *_AREA_SIZE.

Например, можно выставить:

PGA_AGGREGATE_TARGET = 200M
WORKAREA_SIZE_POLICY = AUTO # необязательно, т. к. PGA_AGGREGATE_TARGET > 0

В этом случае администратор должен регулировать эффективное использование памяти сеансов, измененяя единственно величину PGA_AGGREGATE_TARGET. Оба параметра динамически изменяемы для СУБД, а последний — и для сеанса, позволяя перевести его на статичное выделение памяти, например перед выполнением большой загрузки данных, например:

```
ALTER SESSION SET WORKAREA_SIZE_POLICY = MANUAL;
ALTER SESSION SET SORT_AREA_SIZE = 2097152;
ALTER SESSION SET SORT_AREA_RETAINED_SIZE = 2097152;
```

Показатели текущей эффективности можно определять из таблиц:

V$PGASTAT
V$SQL_WORKAREA
V$SQL_WORKAREA_ACTIVE
V$SQL_WORKAREA_HISTOGRAM
V$TEMPSEG_USAGE

10.2.3. Советы СУБД по выбору новых значений

Новый в версии 9 аппарат «экспертных советов» (advices) автоматически формирует «эскпертные заключения» СУБД по поводу возможного выигрыша или проигрыша при изменении размеров основных областей памяти, а так же желательного времени восстановления после сбоя. Решая, на какую величину изменить текущие значения, АБД может ориентироваться на эти советы. Наблюдать эти советы — экспертные заключения самой СУБД — можно в специальных таблицах V$%ADVICE%:

Таблица	***Помогает выбрать параметр СУБД ...***
V$SHARED_POOL_ADVICE	SHARED_POOL_SIZE
V$DB_CACHE_ADVICE	DB_*n*K_CACHE_SIZE DB_KEEP_CACHE_SIZE DB_RECYCLE_CACHE_SIZE
V$MTTR_TARGET_ADVICE	FAST_START_MTTR_TARGET
V$PGA_TARGET_ADVICE	PGA_AGGREGATE_TARGET
V$PGA_TARGET_ADVICE_HISTOGRAM	

Так, таблица V$SHARED_POOL_ADVICE дает администратору экспертное «мнение СУБД» по поводу выигрыша или издержек, связанных с возможным выбором нескольких значений SHARED_POOL_SIZE, больше либо меньше текущего. Пример запроса:

```
SELECT
    shared_pool_size_for_estimate
  , shared_pool_size_factor
  , estd_lc_memory_object_hits
FROM
  v$shared_pool_advice
;
```

Таблица V$DB_CACHE_ADVICE позволит администратору мотивированно принять решение о переустановке размеров разделов DEFAULT, KEEP и RECYCLE буфера блоков данных.

Таблица V$PGA_TARGET_ADVICE дает возможность администратору получить «мнение СУБД» по поводу выбора отличных от существующего значений PGA_AGGREGATE_TARGET. Например, ненулевое значение столбца ESTD_OVERALLOC_COUNT означает, что PGA_AGGREGATE_TARGET желательно выбрать больше.

Таблица V$PGA_TARGET_ADVICE_HISTOGRAM дает возможность администратору получить оценочное «мнение СУБД» по поводу влияния выбора иного значения PGA_AGGREGATE_TARGET на распределение статистических показателей в V$SQL_WORKAREA_HISTOGRAM.

Сбор статистики для выработки СУБД этих оценок расходует процессорное время и память в shared pool. Поэтому дано право его регулировать, правда только в отношении сведений об областях памяти в V$DB_CACHE_ADVICE. Для этого служит (динамический) параметр DB_CACHE_ADVICE:

```
ALTER SYSTEM SET DB_CACHE_ADVICE = ON;
```

По умолчанию его значение зависит от параметра СУБД STATISTICS_LEVEL, но чаще всего DB_CACHE_ADVICE = ON.

10.3. Настройка в версиях 10+

Версия 10 позволяет использовать все основные возможности настройки версии 9, но добавляет к ним новшества, развивающие динамику, автоматику и экспертизу, появившиеся в версии 9. Ниже перечисляются некоторые добавленные возможности.

Для типичной БД, не имеющей больших особенностей по устройству и эксплуатации, методика настройки может ограничиваться следующими действиями:

- администратор следит за регулярно обновляющимися рекомендациями встроенного советника ADDM;
- администратор выполняет рекомендации ADDM.

10.3.1. Настройка SGA

Возможность версии 9 динамического перераспределения памяти в версии 10:

1. расширена на другие участки SGA (см. ниже),
2. доведена до автоматизации.

Так, значения новых параметров СУБД:

SGA_TARGET > 0
STATISTICS_LEVEL <> BASIC

позволяют включить *автоматическое* перераспределение памяти (ASMM, automatic shared memory management, или иногда AMM) среди областей, управляемых следующими параметрами СУБД:

DB_CACHE_SIZE

SHARED_POOL_SIZE
LARGE_POOL_SIZE
JAVA_POOL_SIZE
STREAMS_POOL_SIZE

Если теперь этим параметрам присвоить значения > 0, они начинают восприниматься как минимально допустимые для соответствующих областей. По-прежнему вручную выставляются следующие параметры СУБД для участков в SGA:

DB_KEEP_CACHE_SIZE и DB_RECYCLE_CACHE_SIZE
DB_*n*K_CACHE_SIZE (*n* = 2, 4, 8, 16, 32)
LOG_BUFFER

В автоматическом перераспределении участвует память, указанная параметром SGA_TARGET, за вычетом суммы перечисленных выше параметров. Сам параметр SGA_TARGET является динамическим. С версии 10.2 в его выборе можно руководствоваться советом из таблицы V$SGA_TARGET_ADVICE. Однако верхний возможный для динамической установки предел SGA_TARGET устанавливается статическим параметром SGA_ MAX_SIZE.

Текущие величины участков SGA с учетом их динамического переопределения можно узнать, например, запросом:

```
SELECT component, current_size, min_size, max_size
FROM   v$sga_dynamic_components
;
```

Размер резервной области shared pool при автонастройке SGA устанавливается также автоматически как заданный процент размера shared pool. Величину этой доли shared pool можно задать скрытым параметром _SHARED_POOL_RESERVED_PCT (умолчательно 5%).

Технически автоматическое перераспределение осуществляется специальным фоновым процессом СУБД MMAN. Параметр STATISTICS_LEVEL позволяет регулировать объем собираемой статистики, на основании которой будут приниматься конкретные решения об очередном перераспределении.

10.3.2. Настройка PGA

Параметр PGA_AGGREGATE_TARGET обрел силу для областей PGA *всех* имеющихся серверных процессов (как dedicated, так и shared).

10.3.3. Экспертные советы СУБД по выбору новых значений

Перечень таблиц с экспертными советами СУБД по поводу возможных последствий выбора новых значений некоторых параметров в версии 10 расширился следующими:

V$PX_BUFFER_ADVICE
V$SGA_TARGET_ADVICE
V$JAVA_POOL_ADVICE
V$STREAMS_POOL_ADVICE

10.3.4. Автоматический сбор статистики и авторегулирование

Новое средство версии 10, «репозитарий автоматически собираемых сведений о загруженности СУБД», или Automatic Workload Repository (AWR), представляет собой развитие идеи пакета STATSPACK. AWR представляет собой место хранения в БД статистической информации о работе СУБД плюс механизм ее регулярного сбора и анализа. По умолчанию «снимки» (снятия показаний) делаются ежечасно, а данные хранятся 7 или 8 дней. Физически данные, собираемые AWR, хранятся в табличном пространстве SYSAUX.

Управление AWR возможно как программно, так и через OEM. Пример самостоятельного снятия показаний СУБД в AWR:

```
EXECUTE DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT ( 'TYPICAL' )
```

Текущие данные AWR пожно наблюдать в таблицах 'DBA_HIST_%' ESCAPE '\'. Например, узнать о снятых в AWR (вручную, или же автоматически системой) показателях загрузки СУБД можно запросом:

```
SELECT
  snap_id
, snap_level
FROM
  dba_hist_snapshot
ORDER BY
  begin_interval_time
;
```

Те же сведения в графическом виде предоставляет страница OEM *Administration* → *Automatic Workload Repository*.

10.3.5. Аппарат «советников»

Версия 10 ввела аппарат «советников» (advisors), способных давать развернутое экспертное заключение по поводу использования ряда элементов СУБД и БД, а также предлагать пути решения проблем, если они есть. Представление об областях, покрываемых аппаратом советников, можно получить запросом:

```
SQL> SELECT * FROM dba_advisor_definitions;

ADVISOR_ID ADVISOR_NAME                     PROPERTY
---------- ------------------------------ ----------
         1 ADDM                                    1
         2 SQL Access Advisor                     15
         3 Undo Advisor                            1
         4 SQL Tuning Advisor                      7
         5 Segment Advisor                         3
         6 SQL Workload Manager                    0
         7 Tune MView                             31
```

Собственно настройкой СУБД наиболее полно занимается советник ADDM (Automatic Database Diagnostic Monitor). Для своих заключений он пользуется данными, накопленными в AWR.

Работа по получению от системы советов и применении рекомендаций может строиться как программно (например, с помощью пакетов DBMS_ADVISOR или DBMS_SQL_TUNE), так и через OEM.

Программный просмотр результатов работы ADDM осуществляется с привлечением следующих таблиц:

DBA_ADVISOR_FINDINGS
DBA_ADVISOR_OBJECTS
DBA_ADVISOR_RECOMMENDATIONS
DBA_ADVISOR_RATIONALE

10.4. Настройка в версии 11

Версия 11 позволяет использовать все основные возможности настройки версии 10, но развивает далее возможности динамики, автоматики и экспертизы версии 10. Ниже приводится пример.

10.4.1. Настройка SGA и PGA

Статический параметр MEMORY_MAX_TARGET позволяет задать желаемое максимальное суммарное значение памяти SGA + PGA.

Динамический параметр MEMORY_TARGET позволяет устанавливать суммарный размер SGA + PGA в пределах MEMORY_MAX_TARGET по мере необходимости.

10.5. Настройка формирования контрольных точек

Слишком частое формирование контрольных точек (а) тормозит работу СУБД и (б) уменьшает время восстановления данных после сбоя СУБД в оперативной памяти, и наоборот.

Используются следующие параметры.

- LOG_CHECKPOINT_TIMEOUT — время в секундах до следующего автоматического выполнения контрольной точки.
- LOG_CHECKPOINT_INTERVAL — количество блоков ОС в журнале, заполняемое до следующего автоматического выполнения контрольной точки.
- FAST_START_IO_TARGET[8-9] — максимальное количество операций ввода/вывода (файлы данных БД), которое СУБД будет выполнять при восстановлении по журналу.
- FAST_START_MTTR_TARGET[9-] — примерное число секунд, в которое должна уложиться СУБД при восстановлении по журналу.

[8-9] В версиях 8.x появился, в версии 9 устарел.
[9-] Начиная с версии 9.

Параметр FAST_START_MTTR_TARGET естественнее прочих соотносится с требованиями приложения; если же его использовать, параметр LOG_CHECKPOINT_INTERVAL лучше отключить, например задав:

```
LOG_CHECKPOINT_INTERVAL = 0
LOG_CHECKPOINT_TIMEOUT = 1800
FAST_START_MTTR_TARGET = 600
```

Оценочные значения для параметров можно получить из V$INSTANCE_RECOVERY.

10.6. Настройка журнализации

Некоторые посылки для планирования настройки:

- большое значение LOG_BUFFER сокращает в целом число обращений LGWR к диску;
- частое выполнение COMMIT увеличивает число обращений к диску;
- медленная работа LGWR (медленное обращение к журнальному файлу) чаще приводит к *полному* заполнению буфера, а следовательно ко внутренним задержкам.

10.6.1. Диагностика

Работа механизма журнализации отражена:

- в таблицах статистик (СУБД, сеансов, служб БД) показателями класса 2 (столбец CLASS таблицы V$STATNAME);
- в таблицах внутренних ожиданий с именами LIKE '%log%' или LIKE '% redo%' (столбец NAME таблицы V$EVENT_NAME.

Когда процессор заполняет журнальный буфер быстрее, чем СУБД сбрасывает оттуда записи на диск, фиксируется ожидание 'redo log space requests'. Это свидетельствует о том, что параметр СУБД LOG_BUFFER маловат, или что дисковая подсистема медленно переключается на новый файл. Следующий показатель не должен превышать 1/5000:

```
SELECT r.value / e.value "Wait For Space Ratio"
FROM   v$sysstat r, v$sysstat e
WHERE  r.name = 'redo log space requests'
  AND  e.name = 'redo entries'
;
```

Сеансы, которые поставляют журнальную информацию быстрее, чем процесс LGWR успевает освободить потребное место в буфере и время их простоя можно узнать запросом:

```
SELECT sid, event, seconds_in_wait, state
FROM   v$session_wait
WHERE  event = 'log buffer space'
;
```

Другой пример: когда процесс LGWR отстает в работе или происходит переключение журнального файла, может выполняться повторная попытка размещения зхаписей в буфере: 'redo buffer allocation retries'. Если следующее отношение больше 1%, можно попытаться (а) увеличить размер LOG_BUFFER, или (б) снизить нагрузку со стороны выполнения контрольных точек параметрами СУБД LOG_CHECKPOINT_INTERVAL, LOG_CHECKPOINT_TIMEOUT или FAST_START_MTTR_TARGET):

```
SELECT a.value / w.value "Retries Statictics Ratio %"
FROM   v$sysstat a, v$sysstat w
WHERE  a.name = 'redo buffer allocation retries'
  AND  w.name = 'redo blocks written'
;
```

10.6.2. Настройка СУБД и БД

Основные решения:

- увеличить LOG_BUFFER
- откорректировать внутреннюю структуру буфера
- перенести журналы на более быстрый диск
- увеличить размеры журналов
- уменьшить частоту контрольных точек

Откорректировать внутреннюю структуру буфера можно, увеличив число защелок, используемых буфером, параметром СУБД LOG_PARALLELISM (начиная с версии 9.2, но в версии 10 переведен в разряд скрытых).

Задержки работы LGWR могут быть вызваны медленными переключениями журнальных файлов. Проверка:

```
SELECT
 event, total_waits, time_waited, average_wait
FROM
 v$system_event
WHERE
 event LIKE 'log%switch%'
;
```

Возможное решение: добавить журнальные группы или увеличить их размер.

Задержки могут возникать из-за слишком частых контрольных точек. Проверка: наличие записей «CHECKPOINT NOT COMPLETE» в файле *ALERT.LOG*.

Если ожидается большой поток изменений (загрузка данных) при включенном режиме архивирования журнальных файлов, можно заранее увеличить динамический параметр LOG_ARCHIVE_MAX_PROCESSES.

10.6.3. Решения на уровне приложения

Заведение данных в режиме NOLOGGING способно сокращать объем порождаемой журнальной информации.

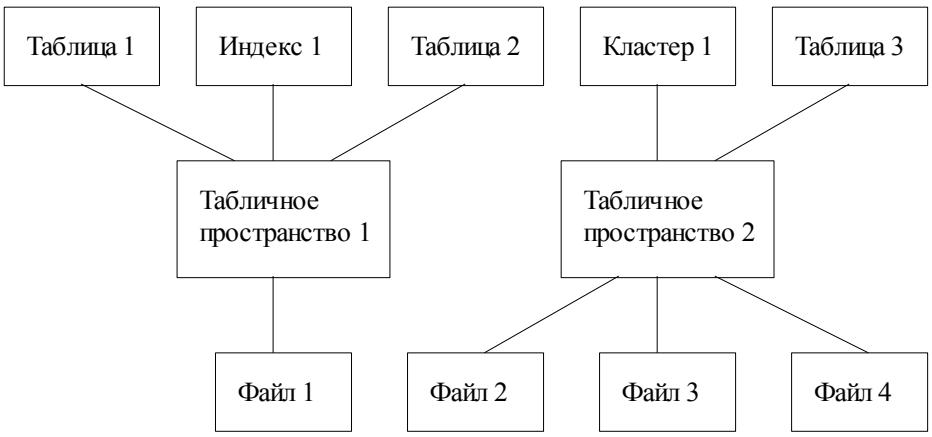
Если БД используется в режиме архивирования, NOLOGGING автоматически возникает при прямоточной загрузке (если этого не запретить специально), иначе его нужно указывать явно.

11. Организация хранения данных в Oracle

Ниже рассматриваются особенности хранения данных БД в табличных пространствах.

11.1. Хранение объектов БД на диске

Следующий рисунок иллюстрирует связь между единицами хранения, табличными пространствами и файлами:



Единицы хранения в табличном пространстве соответствуют логическим объектам БД, требующим собственной структуры для размещения данных, иначе говоря, *сегмента*, как-то:

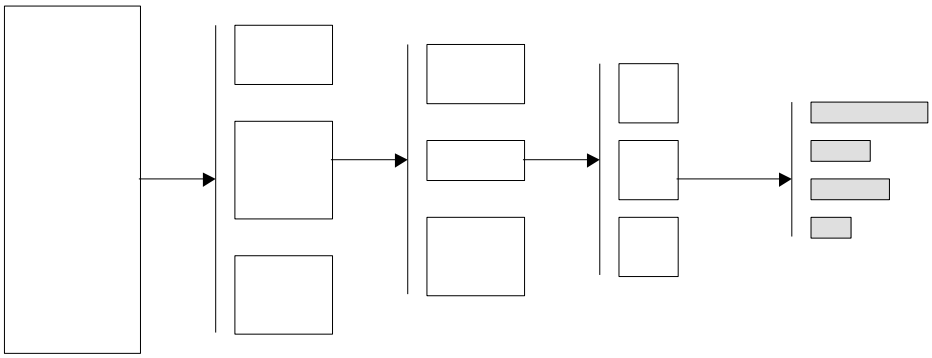
- таблице,
- индексу,
- кластеру,
- разделу таблицы или индекса,
- сегменту отмены (отката) и временных данных,
- таблице с индексной организацией,
- объекту и индеку LOB,
- значениям для вложенных таблиц.

11.1.1. Внутренняя организация хранения данных в табличных пространствах

Основными структурами хранения данных являются:

- Табличные пространства
- Сегменты
- Экстенты («сплошные» куски)
- Блоки

Состав табличного пространства поясняется следующим рисунком:



Табличное пространство	Сегменты	Экстенты	Блоки	Записи
Задается параметрами CREATE/ALTER TABLESPACE...	Создаются для хранимых объектов БД (особый случай: секционированные таблицы)	Параметры задаются конструкцией STORAGE	Параметры задаются конструкцией CREATE/ALTER TABLE/INDEX...	

С версии 11 по умолчанию действует отложенное создание сегментов, когда фактически сегмент будет заводиться в табличном пространстве не при создании таблицы, а при поступлении в нее первой строки (DEFERRED_SEGMENT_CREATION = TRUE).

Сведения об имеющихся сегментах и экстентах доступна в таблицах словаря-справочника DBA(USER)_SEGMENTS и DBA(USER)_EXTENTS.

Пример просмотра полного числа блоков в SQL*Plus:

```
SELECT blocks
FROM dba_segments
WHERE
    owner = UPPER ( '&object_owner' )
AND segment_name = UPPER ( '&object_name' )
;
```

Те же сведения можно получить в OEM.

Сведения о незанятых областях в файлах табличных пространств имеются в таблице словаря-справочника DBA_FREE_SPACE. Общую сводку всех, занятых и свободных от данных блоков можно вычислить так:

```
WITH
"all"  AS ( SELECT   tablespace_name, SUM ( blocks ) blocks
            FROM     dba_data_files
            GROUP BY tablespace_name
          ),
"used" AS ( SELECT   tablespace_name, SUM ( blocks ) blocks
            FROM     dba_segments
            GROUP BY tablespace_name
          ),
"free" AS ( SELECT   tablespace_name, SUM ( blocks ) blocks
            FROM     dba_free_space
            GROUP BY tablespace_name
          )
SELECT
  tablespace_name
, "all" .blocks  "all"
, "used".blocks  "used"
, "free".blocks  "free"
```

```
FROM
   "all"
   LEFT JOIN "used" USING ( tablespace_name )
   LEFT JOIN "free" USING ( tablespace_name )
;
```

Во всех использованных таблицах имеется столбец BYTES. Если в последнем запросе заменить BLOCKS на BYTES, получим те же сведения в байтах.

Статистистические показатели для сегментов способна раскрыть таблица V$SEGMENT_STATISTICS. Например, общую картину по статистике даст запрос:

```
SELECT   statistic_name, COUNT ( * )
FROM     v$segment_statistics
GROUP BY statistic_name
;
```

11.1.2. Управление размещением сегментов в табличных пространствах

Увеличение объема данных в объекте ведет к автоматическому росту сегмента за счет выделения у в файлах табличного пространства места для новых экстентов.

С версии 8.1.5 в Oracle имеется два механизма роста сегментов: «старый» и «новый» (современный). Техническое различие между ними в том, что в «старом» механизме новые экстенты для растущего сегмента выделяются централизовано, через словарь-справочник, а в «новом» — на основе локальной карты свободной памяти внутри самого табличного пространства. Старый механизм по отношению к новому:

- медленнее обрабатывает резервирование памяти в табличном пространстве при росте объекта (например, таблицы),
- более склонен к фрагментированию табличного пространства.

Для заведенных пространств возможно (если фактические размеры экстентов это допускают) преобразование пространства из пространства «старого типа» в пространство «нового типа» — с помощью специального системного пакета DBMS_SPACE_ADMIN.

Выбор централизованного или локального способа управления сегментами производится при создании табличного пространства и является окончательным. Начиная с версии 9.2 все табличные пространства в БД по умолчанию создаются с локальным управлением.

11.1.2.1. Централизованный механизм выделения экстентов

В обычном случае выделение памяти для новых экстентов в процессе роста сегментов управляется следующими параметрами конструкции STORAGE (указывается для сегмента или же для табличного пространства, но в качестве значения по умолчанию):

- INITIAL (размер первого экстента)
- NEXT (размер второго экстента)
- PCTINCREASE (процент роста каждого следующего экстента по отношению к предыдущему, начиная с третьего)
- MAXEXTENTS (максимально допустимое для этого сегмента число экстентов)
- MINEXTENTS (число экстентов, создаваемых в сегменте при постановке первой записи)

Начиная с версии 8.1.5 создание таких пространств *может* уточняться фразой EXTENT MANAGEMENT DICTIONARY, а с версии 9 — *обязано* уточняться:

```
CREATE TABLESPACE ts1
 DATAFILE 'c:\oracle\oradata\prima\ts101.dbf ' SIZE 100M
 EXTENT MANAGEMENT DICTIONARY
;
```

11.1.2.2. Локальный механизм выделения экстентов в версиях 8.1.5+

В версиях 8.1.5 и 8.1.6 (слегка подправлено) появилась дополнительная возможность управления экстентами. Табличные пространства могут быть категории Locally Managed Tablespaces (пространства с локально хранимой картой свободной памяти), в противовес старым Dictionary Managed Tablespaces.

В пространствах с локальным управлением экстентов последние выделяются без обращения к словарю-справочнику Oracle. Имеется *две разновидности* таких пространств: в первой размеры экстентов при росте сегмента получают поэтапно значения 64K, 1M, 8M и 64M; во второй все экстенты всегда одинаковы. В обоих случаях в самом табличном пространстве имеется поразрядная карта распределения памяти внутри пространства, что может существенно ускорить выделение экстентов под растущие объекты.

Пример создания табличного пространства первой разновидности:

```
CREATE TABLESPACE misc
 DATAFILE 'c:\oracle\oradata\prima\misc01.dbf' SIZE 100M
 EXTENT MANAGEMENT LOCAL
 AUTOALLOCATE
;
```

Пример создания табличного пространства второй разновидности:

```
CREATE TABLESPACE medium_tables
 DATAFILE 'd:\oracle\oradata\prima\medium_tables01.dbf' SIZE 100M
 EXTENT MANAGEMENT LOCAL
 UNIFORM SIZE 400K
;
```

Для обслуживания такого рода пространств используется системный пакет DBMS_SPACE_ADMIN.

11.1.2.3. Искусственное наращивание или сокращение сегмента

Добавить к сегменту новый экстент из табличного пространства можно вручную. Это бывает полезно для повышения скорости работы с объектом, так как:

- загодя выделив сегменту экстент(ы) можно ускорить грядущий рост данных объекта, так как СУБД не будет потом отвлекаться на автоматическое наращивание сегмента
- новый экстент можно выделить в другом файле табличного пространства (если оно состоит из нескольких файлов), получив выигрыш в скорости доступа к объекту за счет чередования запросов к одним данным к разным дисковым устройствам (data striping)

Примеры выделения экстента вручную:

```
ALTER TABLE emp ALLOCATE EXTENT;

ALTER TABLE emp ALLOCATE EXTENT ( SIZE 1M );

ALTER TABLE emp
 ALLOCATE EXTENT ( DATAFILE 'f:\oracle\oradata\orcl\USERS01.DBF')
;
```

С другой стороны «лишние» экстенты (неиспользованные до сих пор), а также место в недозаполненном экстенте («справа» от верхней отметки заполнения, см. ниже) можно вернуть табличному пространству:

```
ALTER TABLE emp DEALLOCATE UNUSED;

ALTER TABLE emp DEALLOCATE UNUSED KEEP 64K;
```

Для локально управляемых пространств с единым размером экстента (UNIFORM SIZE) фразы SIZE и KEEP в предложениях выше игнорируются, а для таких же пространств с поэтапным автоматическим ростом размера экстента (AUTOALLOCATE) — нет.

Упражнение. Проверить, что искусственное наращивание или сокращение сегмента для табличных пространств типа EXTENT MANAGEMENT LOCAL AUTOALLOCATE способно привести к появлению экстентов длиною, отличной от 64K, 1M, 8M или 64M.

11.1.3. Управление размещением данных в сегментах

Заполнение блоков сегмента данными может также осуществляться по-разному:

- по правилу «списка свободных блоков» (free list — «старое» правило) и
- с использованием «автоматического управления местом в сегменте» (automatic segment storage management, ASSM — «новое», или современное правило).

Для табличных пространств с централизованным управлением действует только «старое» правило, а для пространств с локальным управлением возможет выбор. Выбор делается при создании табличного пространства и является окончательным.

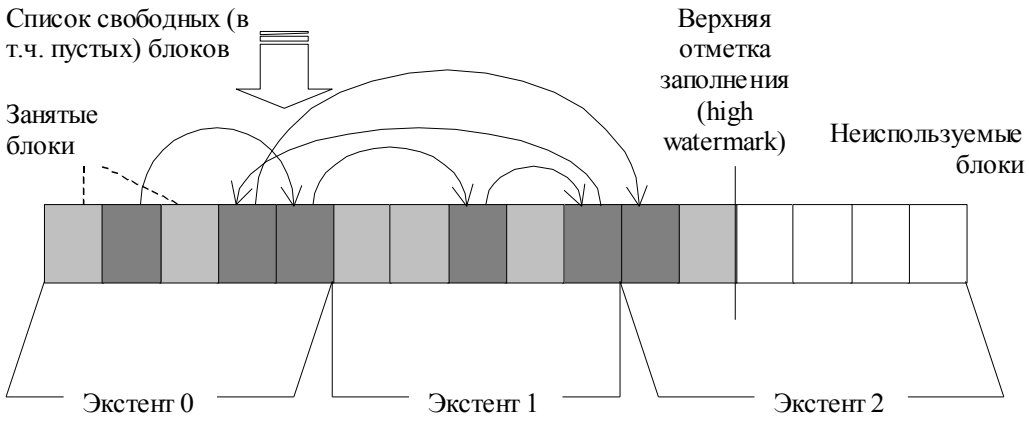
В целом механизм ASSM считается более предпочтительным (повышает производительность обработки обращений к общим данным несколькоих программ), однако Oracle не отказывается и от употребления списков свободных блоков из-за того, что они способны давать лучшую производительность в некоторых случаях, как например, при особенно интенсивном потоке команд INSERT, UPDATE и DELETE. Кроме того в версии 9 ASSM не подходит для сегментов типа LOB.

11.1.3.1. Механизм списка свободных блоков

В этом случае для каждого сегмента Oracle поддерживает *список свободных блоков* (а может быть и сразу несколько списков для повышения скорости работы при интенсивных вставках данных). Когда появляется новый свободный блок (после удаления данных или в результате захвата нового экстента) он «включается» в конец списка. Такой блок будет проверяться на возможность приема данных в последнюю очередь.

Также для каждого объекта хранится наибольший номер из всех использовавшихся блоков («верхняя отметка заполнения», highwater mark). При обычной работе с таблицей эта отметка (по определению) может только расти и сбрасывается она только (а) командой TRUNCATE, (б) указанием DEALLOCATE UNUSED в ALTER TABLE (в последних версиях), (в) командой ALTER TABLE … SHRINK SPACE (начиная с версии 10). ⇨ Полезность перезагрузки данных (при выдаче COUNT(*), например, будут обязательно просматриваться *все* блоки от первого до последнего из использовавшихся, независимо от их заполненности данными).

Логически картинка распределения памяти сегмента в табличном пространстве выглядит примерно так:



Размер списков свободных блоков и местонахождение верхней отметки заполнения можно посмотреть только с помощью системного пакета DBMS_SPACE (в таблицах словаря-справочника эта информация отсутствует).

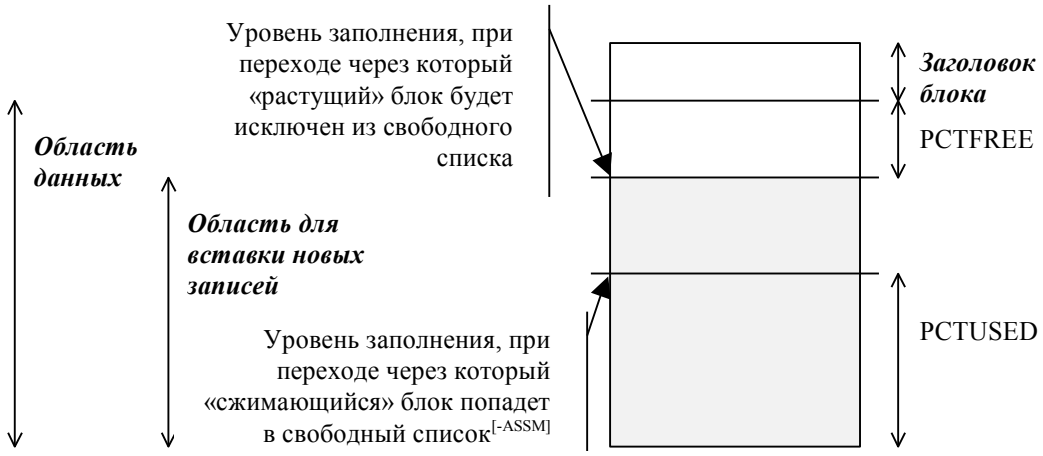
11.1.3.2. Механизм ASSM автоматического управления памятью в сегменте

Начиная с версии 9 в табличных пространствах с локальным управлением употребление списка свободных блоков возможно, умолчательно не подразумевается, а если используется, то должно указываться явочным порядком при заведении пространства (CREATE TABLESPACE ... **SEGMENT SPACE MANAGEMENT MANUAL**).

Альтернативой для таких табличных пространств является автоматического управления местом в сегменте (ASSM; задаетсякак CREATE TABLESPACE ... **SEGMENT SPACE MANAGEMENT AUTO**). В этом случае параметры FREELISTS и FREELIST_GROUPS игнорируются, а выбор блока для новой записи определяется по *спискам* специальных *служебных блоков* в сегменте (bitmapped blocks, BMB), содержащих *поразрядную карту занятости* блоков в экстентах.

11.1.4. Управление памятью в блоках с данными

Заполнение блока данных иллюстрируется следующей схемой и контролируется указанными на рисунке параметрами:



[-ASSM] Параметр не имеет силы и значение игнорируется в блоках локально управляемых табличных пространств установленным свойством ASSM (что допускается с версии 9).

Значение PCTFREE > 0 провоцирует недозаполненность блоков, однако способно сократить количество разорваных записей, требующих дополнительных затрат на прочтение. Фактическое количество разорваных записей дает анализ таблицы (пакет DBMS_STATS или команда SQL ANALYZE), а полный список адресов таких записей дает команда ANALYZE; например:

```
ANALYZE TABLE emp LIST CHAINED ROWS INTO chained_rows;
```

Здесь таблица CHAINED_ROWS была получена прогоном сценария *utlchain.sql* в *%ORACLE_HOME %\rdbms\admin* (другой сценарий, *utlchn1.sql*, даст список адресов в «универсальном» формате, распространяющимся помимо обычных таблиц на индексно-организованные).

В табличных пространствах с ASSM изменение PCTFREE вступит в силу на уже занятых блоках (а не новых занимаемых) только после обработки сегмента процедурой SEGMENT_FIX_STATUS из пакета DBMS_REPAIR. В табличных пространствах с использованием механизма списка свободных блоков изменение PCTFREE учитывается только на новых блоках списка.

В блоках индекса PCTFREE СУБД учитывает только при создании индекса.

Заголовок состоит из постоянной и переменной частей.

Параметр блока INITRANS позволяет задать начальный размер таблицы транзакций в заголовке блока (ITL) и соответствует количеству транзакций, могущих одновременно править разные записи в одном блоке. При необходимости эта таблица динамически увеличивается вплоть до значения MAXTRANS (еще один параметр блока), а начиная с версии 10, где MAXTRANS был упразднен, до 256. О задержках из-за недостаточно большой таблицы транзакций (а возможно, чрезмерного заполнения блока записями при большом количестве работающих с ним транзакций) сообщит ответ на запрос:

```
SELECT owner, object_name
FROM   v$segment_statistics
WHERE  statistic_name = 'ITL waits'
AND    value > 0
;
```

(Набор показателей статистики использования сегментов дает таблица V$SEGSTAT_NAME.)

К версии 10 (при выборе ASSM для табличного пространства) осталось только два параметра блока: PCTFREE и INITRANS.

Посмотреть структуру конкретного блока можно командой:

```
ALTER SYSTEM DUMP DATAFILE номер_или_имя_файла BLOCK номер_блока;
```

Результат поступит в файл трассировки процесса сеанса.

12. Резервное копирование и восстановление

Технические процедуры резервного копирования и восстановления (иными словами снятия запасных списков, копий, и восстановления данных по этим спискам) предназначаются для решения одной общей задачи: защиты данных от потерь. Единственной техники выполнения этих процедур нет, а вместо этого для них существует определенное многообразие способов и средств.

12.1. Виды резервного копирования

Резервное копирование в целом (а не только в Oracle и не только в базах данных) можно классифицировать по нескольким независимым «направлениям» характеристик:

- физическое — логическое
- холодное — горячее
- полное — частичное (разностное, разностное накопительное)

12.1.1. Физическое резервирование

Копирование файлов БД на ленту, другой диск или другую машину.

Достоинства:
Процесс восстановления обычно осуществляется быстрее, чем при логическом резервировании (простое копирование файлов).

Ограничения:
Можно восстанавливать данные только на той же или на однотипной ЭВМ (в Oracle 10+ это уже не так строго). Нельзя восстанавливать БД от другой версии СУБД.

12.1.2. Логическое резервирование

Резервирование отдельных объектов, хранимых в БД.

Достоинства:
Объекты БД можно восстанавливать по-отдельности. Часто восстановление можно осуществлять на других платформах или версиях СУБД.

Ограничения:
Восстановление может происходить медленно, при больших объемах данных в таблицах и в силу необходимости воспроизводить индексы. Порядок восстановления объектов БД может противоречить ограничениям целостности схем в БД. Восстановление возможно только по состоянию на момент времени, когда снималась копия.

12.1.3. Резервирование изменений (частичное)

Позволяет копировать только изменения, произошедшие со времени предыдущего резервирования: последнего (разностное; инкрементальное копирование, incremental) или еще более раннего (разностно-накопительное; кумулятивное, cumulative).

Достоинства:
Требует меньше места для хранения данных.

Ограничения:
- Восстановление требует применения по очереди всех накопленных копий изменений.

- В зависимости от схемы резервирования, восстановление может требовать определенного жесткого порядка применения копий изменений.
- Потеря одной копии изменений делает невозможным дальнейшее восстановление.

12.1.4. Холодное/горячее резервирование

Холодное резервирование: копирование файлов данных, контрольных файлов и файлов журнала при неработающей системе.

Достоинства:
- Простейший в исполнении метод.
- Может осуществляться безотносительно к использованию режима ARCHIVELOG (Oracle).

Ограничения:
- На время копирования БД должна быть остановлена.
- В режиме NOARCHIVELOG вся база должна копироваться и восстанавливаться целиком (Oracle).

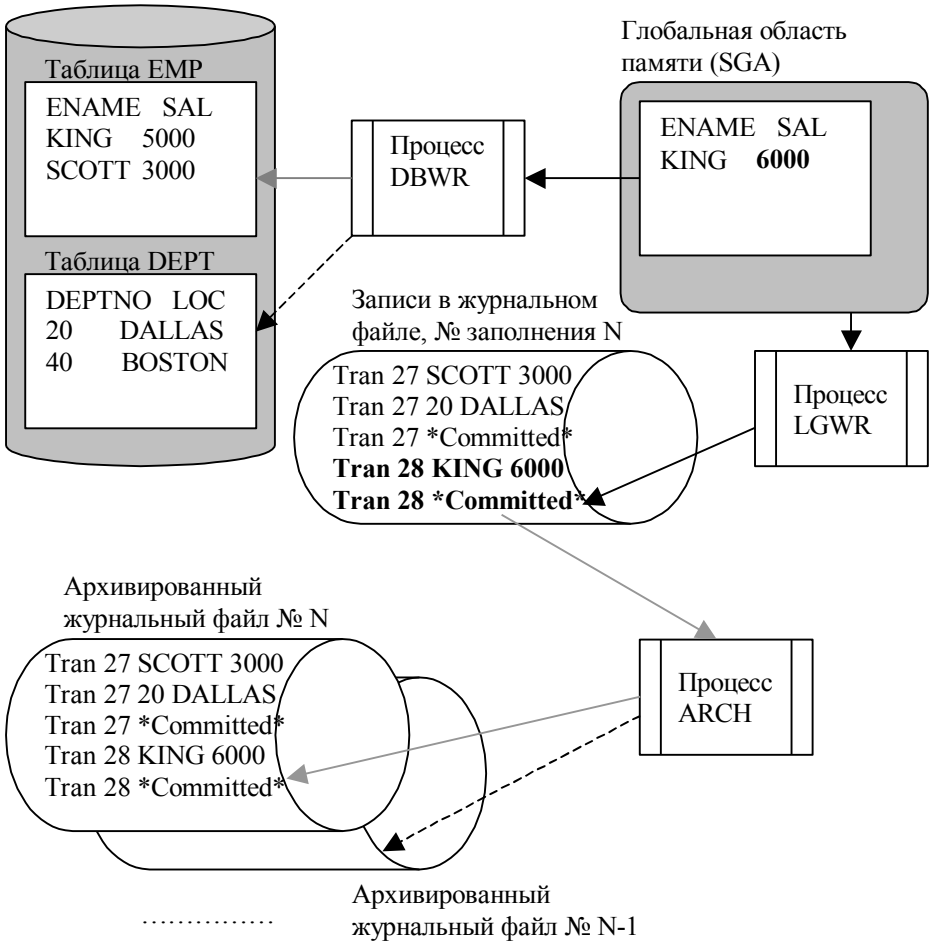
Горячее резервирование: выполняется без останова работы системы.

Достоинства:
Наиболее богатый по своим возможностям способ резервирования. Не требует остановки системы. Позволяет восстановить данные на произвольный момент времени.

Ограничения:
Должен быть включен режим архивирования, замедляющий обычную работу.

12.2. Режим ARCHIVELOG работы БД

Включает процесс СУБД ARCH (технически это несколько процессов ARC*n*), который постоянно переписывает в специально создаваемые «архивные файлы» содержание журнальных файлов, освобождающихся по мере работы процесса СУБД LGWR (протоколирующего вносимые в БД изменения).



Достоинства:
В сочетании с физическими копиями файлов позволяет осуществлять восстановление БД более точно.

Ограничения:
Может замедлять скорость работы СУБД. Если процесс ARCH не в состоянии осуществлять запись, СУБД прекращает внесение изменений в БД. За накоплением архивных копий журналов (организационно) приходится следить.

12.3. Основные сценарии физического резервирования

Ниже описаны возможные сценарии выполнения полного холодного и горячего резервирования и восстановления на физическом уровне.

12.3.1. Холодное резервирование

Для выполнения холодного резервирования нужно предпринять следующие действия:

1) Остановить Oracle, чтобы привести БД в согласованное состояние. Команды SHUTDOWN, SHUTDOWN IMMEDIATE, SHUTDOWN TRANSACTIONAL.
2) Скопировать файлы БД в предназначенное место.
3) Запустить Oracle снова. Команда STARTUP.

Замечания:

- Если при восстановлении холодной копии выбрано новое месторасположение файлов БД, перед открытием БД следует выдать команду ALTER DATABASE RENAME FILE.
- Список необходимых для копирования файлов можно узнать, выдав запрос к словарю-справочнику:

```
SELECT name FROM v$datafile
UNION ALL
SELECT name FROM v$tempfile
UNION ALL
SELECT member FROM v$logfile
UNION ALL
SELECT name FROM v$controlfile
;
```

  Местонахождение файлов параметров СУБД — умолчательное для *INIT.ORA* и стандартное для *SPFILE.ORA* — в *$ORACLE_HOME/dbs* (Unix) и в *%ORACLE_HOME%\database* (Windows).
- Копирование файлов журнала при нормальном (не ABORT) закрытии БД необязательно. Отказавшись от него, получим экономию места хранения копии, но одновременно усложним процедуру восстановления.
- Копирование файлов табличного пространства в состоянии READ ONLY или OFFLINE также можно не выполнять, если имеются их копии от предыдущих операций резервирования.

12.3.2. Пример автоматизации

Снятие холодной копии легко может быть автоматизировано. Рассмотрим пример БД в Windows, работающей по *SPFILE.ORA*. Подготовим файл *savefiles.sql*:

```
STORE SET /temp/sqlplussettings.sql REPLACE
SET VERIFY OFF  PAGESIZE 0  FEEDBACK OFF
SPOOL /temp/docopy.sql REPLACE
SELECT 'host copy ' || name || ' &1' FROM
(
SELECT name FROM v$datafile
UNION ALL
SELECT name FROM v$tempfile
UNION ALL
SELECT member FROM v$logfile
UNION ALL
SELECT name FROM v$controlfile
UNION ALL
SELECT value FROM v$parameter WHERE name='spfile'
)
;
SPOOL OFF
@/temp/sqlplussettings.sql
```

Еще один файл, *docold.sql*:

```
@/temp/savefiles &1
SHUTDOWN IMMEDIATE
@/temp/docopy
STARTUP
```

Поместим файлы в каталог *\temp*. Тогда снять холодную копию с работающей БД можно в SQL*Plus следующим образом:

```
SQL> CONNECT / AS SYSDBA
SQL> HOST mkdir cold_copy
SQL> @/temp/docold cold_copy
```

Исправления для Unix касаются только операций работы с файлами (*copy*, *mkdir*).

12.3.3. Включение режима архивирования

12.3.3.1. Включение во всех версиях

Способ устаревший, но действующий.

Перед включением режима архивирования, или же сразу после, нужно осуществить полное резервирование БД (иначе архивирование журналов бессмыслено). Ниже приводится традиционная последовательность действий, сохраняющая свою силу во всех версиях Oracle:

1) Остановить Oracle.
2) Снять «холодную» копию БД.
3) Отредактировать файл параметров СУБД (*INIT.ORA* или *SPFILE.ORA)*. Проставить следующие значения:

   LOG_ARCHIVE_START = TRUE
   LOG_ARCHIVE_DEST = *полное_имя_каталога_для_архива*
   LOG_ARCHIVE_FORMAT = "*имя_базы_данных*S%ST%T.ARC"

4) «Смонтировать» Oracle командой `STARTUP MOUNT` (открыть контрольный файл).
5) Перевести БД в режим архивирования командой `ALTER DATABASE ARCHIVELOG;` (проставить в контрольный файл признак архивации для БД).
6) Открыть базу данных: `ALTER DATABASE OPEN;`

Проверить действенность выполненного включения можно по-разному:

(а) выдать команду SQL*Plus `ARCHIVE LOG LIST` и посмотреть ответ;
(б) выдать в SQL*Plus команду SQL `ALTER SYSTEM SWITCH LOGFILE;` и наблюдать появление в каталоге *полное_имя_каталога_для_архива* появление файла с архивированой копией.

Замечание. Поскольку информация о включенном режиме архивирования заносится не только в файл параметров СУБД, но и в контрольный файл БД, логичнее снимать холодную копию сразу *после* включения этого режима. В алгоритме выше это предложено делать *до* включения только ради надежности при выполнении действий начинающим администратором.

12.3.3.2. Включение архивирования с версии 8.1

С версии 8.1 процедура включения архивирования журнальных файлов может выполняться как и прежде, но может и отличаться в отношении указания места рассылки архивированного образа.

Версия 8.1 ввела на смену параметру LOG_ARCHIVE_DEST группу параметров

LOG_ARCHIVE_DEST_*n* = *место_расположения_архива*

где *n* от 1 до 10. При этом местом расположения может быть по-прежнему каталог, и тогда указывается location=*имя_каталога*, но может быть и другая БД, и тогда указывается service=*имя_другой_БД*. Например:

LOG_ARCHIVE_DEST_1 = "**location=**c:\oracle\oradata\orcl"
LOG_ARCHIVE_DEST_2 = "**service=**standby.class"

Возможность service=… позволяет автоматически рассылать архивированные копии журнала при устройстве горячего резерва БД.

Вдобавок, в *месте_расположения_архива* можно указать желаемую реакцию на возможные ошибки копирования и некоторые другие подробности.

У каждого параметра LOG_ARCHIVE_DEST_*n* есть парный *динамический* параметр LOG_ARCHIVE_DEST_STATE_*n*, позволяющий по ходу работы СУБД активировать или разактивировать конкретное место рассылки, например:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = DEFER;
```

Параметры LOG_ARCHIVE_DEST_*n* и LOG_ARCHIVE_DEST_STATE_*n* нельзя признать устаревшими. Они используются для повышения надежности накопления резервных файлов и для организации горячего резерва.

12.3.3.3. Включение архивирования с версии 10

С версии 10 процедура включения архивирования журнальных файлов может выполняться как и прежде, но может и отличаться в отношении указания места размещения архивированного образа и организационно.

А) Версия 10 ввела новые обозначения для маски имени файла с архивированной копией журнала: %a, %d, %r, причем последнее условное обозначение включать в маску обязательно. Например:

LOG_ARCHIVE_FORMAT = "**%d**S%ST%Tr**%r**.ARC"

Б) Версия 10 сделала необязательным указание LOG_ARCHIVE_START = TRUE, так как процесс ARC*n* СУБД будет запущен автоматически по информации из контрольного файла.

В) Версия 10 ввела новые параметры на замену LOG_ARCHIVE_DEST% и LOG_ARCHIVE_FORMAT:

DB_RECOVERY_FILE_DEST = *общий_каталог_для_всех_данных_резервирования*
DB_RECOVERY_FILE_DEST_SIZE = *предельный_объем_данных_в_каталоге*

Эти параметры назначают т. н. «область быстрого восстановления» на диске (flash recovery area, FRA, а с версии 11 название изменили на fast recovery area). Пример:

```
DB_RECOVERY_FILE_DEST = "c:\oracle\flash_recovery_area"
DB_RECOVERY_FILE_DEST_SIZE = 2G
```

Достоинство использования области FRA в том, что оно подразумевает: (а) накопление в одном месте (FRA) *разного* рода файлов, необходимых для восстановления, (б) автоматическое формирование по мере надобности подкаталогов и файлов, и автоматическое их именование, (б) автоматическое удаление файлов, ставших ненужными вследствие снятия более поздних резервных копий — при исчерпании выделенной параметром DB_RECOVERY_FILE_DEST_SIZE квоты дискового пространства.

Если указанные два параметра в версии 10 установлены, включение и выключение архивирования фактически сводится к выдаче команды `ALTER DATABASE [NO]ARCHIVELOG;`

Справочная информация об области быстрого восстановления FRA содержится в таблицах:

V$RECOVERY_FILE_DEST
V$FLASH_RECOVERY_AREA_USAGE[10–11.1]
V$RECOVERY_AREA_USAGE[11.2-)
[10–11.1] С версии 11.2 имя полагается устаревшим.
[11.2-) С версии 11.2.

Обращение командой SELECT ко второй таблице может сопровождаться диагностическим сообщением о превышении установленного предела заполненности FRA данными резервирования.

12.3.3.4. Выключение режима архивирования

Выполняется та же последовательность действий, но «с обратным знаком»:

1) Остановить Oracle.
2) Снять холодную копию БД.
3) `START MOUNT`
4) `ALTER DATABASE NOARCHIVELOG;`
5) `ALTER DATABASE OPEN;`

С этого момента циклическая журнализация изменений БД будет продолжаться без архивирования и восстановить базу данных можно будет только по холодной копии.

В версии 10 при использовании FRA для выключения режима архивирования достаточно перевести систему в состояние MOUNT и выдать команду:

```
ALTER DATABASE NOARCHIVELOG;
```

12.4. Ручное горячее физическое копирование и восстановление

12.4.1. Горячее резервирование

Горячее резервирование может осуществляться только при включенном режиме архивирования воизбежание рассогласованости данных.

«Ручной» способ снятия копии здесь приводится в основном для понимания процесса. В своей полноте он сейчас используется редко, хотя отдельные его элементы все же могут пригодиться.

12.4.1.1. Резервирование данных табличных пространств

Перед выполнением резервирования данных табличного пространства выдается команда:

```
ALTER TABLESPACE имя_пространства BEGIN BACKUP;
```

По этой команде:

- измененные в оперативной памяти блоки с данными этого табличного пространства сбрасываются на диск, как при контрольной точке,
- само табличное пространство помечается как находящееся «в процессе резервирования»,
- информация в журнальные файлы начинает заноситься блоками, а не побайтово (а значит значительно большими объемами).

Затем следует скопировать файлы табличного пространства штатными средствами ОС в Unix или же программой *ocopy.exe* в Windows (входит в состав ПО Oracle). По завершению копирования файлов выдается команда:

```
ALTER TABLESPACE имя_пространства END BACKUP;
```

которая возвращает пространство в нормальный режим работы.

После этого рекомендуется переключить журнальные файлы командой:

```
ALTER SYSTEM SWITCH LOGFILE;
```

Внимание ! Копирование файлов БД при работающей СУБД без перевода в режим BEGIN BACKUP, хотя технически и возможно, но некорректно и может приводить к ошибкам.

В версии 9 в состоянии БД MOUNT (а с версии 10 — и в состоянии OPEN) можно выдать более общую команду:

```
ALTER DATABASE END BACKUP;
```

В версии 10 появилась симметричная команда для перевода всех табличных пространств в режим BEGIN BACKUP:

```
ALTER DATABASE BEGIN BACKUP;
```

По завершении копирования рекомендуется сбросить текущие журнальные записи в архив:

```
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

12.4.1.2. Резервирование контрольного файла

Резервирование контрольного файла выполняется командой:

```
ALTER DATABASE BACKUP CONTROLFILE TO имя_файла-копии REUSE;
```

или же:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

В первом случае будет снята «горячая физическая копия». Ключевое слово REUSE позволит затереть возможно имеющуюся ранее снятую копию с тем же именем. Во втором случае в файле трассировки процесса, обслуживающего сеанс пользователя, появится сценарий воссоздания контрольного файла. Вместо трассировочного файла процесса можно использовать явное указание имени файла, например:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE AS '@_control.sql' REUSE;
```

12.4.1.3. Резервирование журнальных файлов

Резервирование оперативных (online) журнальных файлов требуется только в случае снятия полной холодной копии (и, таким образом, выполняется не специально, а заодно с прочими файлами БД). Заботиться следует о текущем состоянии журнальных файлов.

Если снимается горячая резервная копия (то есть БД работает в режиме ARCHIVELOG), надежнее будет сбросить журнальную информацию в архив (например, явно переключиться на следующий журнальный файл) до и после резервирования.

12.4.2. Основные сценарии восстановления на физическом уровне

С учетом описанного здесь, а еще более — в документации и других источниках, Oracle предоставляет чрезвычайно много возможностей для копирования и восстановления в конкретных случаях. Ниже приводятся наиболее популярные схемы физического восстановления данных после потерь.

Признаком разрушения носителей данных в Oracle (файлов) может служить ошибка ORA-01157 при переходе из состояния БД MOUNTED в состояние OPEN, то есть при открытии файлов БД по информации из контрольного файла. Уточнить причину отказа СУБД открыть базу можно по файлу *ALERT.LOG* или запросом:

```
SELECT
  name, error
FROM
  v$datafile JOIN v$recover_file USING ( file# )
;
```

С версии 11 для просмотра файла *ALERT.LOG* (фактически *log.xml*) удобно воспользоваться программой *adrci*.

12.4.2.1. Восстановление по полной холодной копии

Осуществляется восстановлением *всех* файлов БД (собственно формирующих холодную копию) по «своим местам» и простым открытием базы.

12.4.2.2. Общая схема восстановления с использованием архивных копий журналов

В наиболее распространенном варианте общая схема восстановления выглядит следующим образом:

1) Восстановить файлы БД по имеющимся копиям.
2) Убедиться в наличии архивированных файлов журнала с момента времени, соответствующего восстановленным файлам, до момента восстановления.
3) Перевести БД в состояние MOUNTED (то есть, открыть только контрольный файл).
4) С помощью команды RECOVER в SQL*Plus применить последовательно изменения в БД, зафиксированные в архивных копиях.
5) Остановить Oracle (SHUTDOWN).
6) Перевести БД в состояние OPEN (открыть БД).

Шаг 5 рекомендуется фирмой Oracle, хотя часто после шага 4 можно сразу открыть БД.

12.4.2.3. Восстановление всей БД

12.4.2.3.1. Законченое восстановление

Выполняются следующие действия:

1) Если БД работает, ее нужно закрыть; остановить СУБД.
2) Скопировать на старые места сделанные ранее копии файлов БД *за исключением контрольного файла и журнальных файлов*. (Если контрольный файл потерян или испорчен, его иногда можно восстановить с помощью команды CREATE CONTROL FILE).

3) Смонтировать БД. Выдать:

```
>sqlplus /NOLOG
SQL> CONNECT / AS SYSDBA
SQL> STARTUP MOUNT
```

4) Выдать в SQL*Plus команду `RECOVER DATABASE`. В типичном случае на каждый последующий вопрос СУБД достаточно будет отвечать возвратом каретки.
5) Получив сообщение о завершении восстановления можно закрыть базу командой `SHUTDOWN` и открыть как обычно, или же открыть сразу командой `ALTER DATABASE OPEN`.

База восстановлена.

12.4.2.3.2. Восстановление на момент времени в прошлом («недоконченое»)

В отличие от «окончательного» восстановления командой RECOVER DATABASE (complete), можно выполнить «недоконченое» восстановление (incomplete), например:

```
RECOVER DATABASE UNTIL TIME '2003-09-21:12:00:00'
```

(Варианты: вместо UNTIL TIME можно указать UNTIL CHANGE *nnnnn*, то есть номер SCN, или UNTIL CANCEL, то есть пока не кончатся наличные архивные копии журнала).

При восстановлении БД на момент времени в прошлом открывать базу придется со сбросом содержимого текущего журнального файла:

```
ALTER DATABASE OPEN RESETLOGS;
```

Если речь идет о рабочей БД, то после такого восстановления с нее необходимо снять ее полную физическую копию, начинающую историю хранения физических копий файлов БД сызнова, так как номер переключений журнальных файлов сбрасывается в 1, и все прежние резервные копии оказываются бесполезными для будущих восстановлений (до версии 10). Такое открытие БД носит специальное название *инкарнация*.

Поскольку инкарнация — ответственный этап жизни БД, прежде чем осуществлять `ALTER DATABASE OPEN RESETLOGS`, можно выполнить `ALTER DATABASE OPEN READ ONLY` и удостовериться, что БД доведена до требуемого состояния.

12.4.2.4. Восстановление данных табличного пространства

Если БД велика, а данные оказались потеряны всего в одном табличном пространстве, более разумно восстановить его отдельно, а не восстанавливать всю БД целиком. Это будет эффективнее и *не* потребует общего закрытия БД.

Последовательность действий:

1) Перевести табличное пространство в нерабочее состояние:

   ```
   ALTER TABLESPACE имя_пространства OFFLINE;
   ```
2) Восстановить файлы табличного пространства (если их несколько) из состава резервных копий. Если файлу предназначено новое местонахождение, следует выдать команду для обновления информации в контрольном файле:

   ```
   ALTER TABLESPACE имя_пространства RENAME
   DATAFILE 'полное_старое_имя_файла'
   TO 'полное_новое_имя_файла'
   ;
   ```
3) Теперь можно начинать восстановление содержимого файлов. Следует обеспечить наличие всех необходимых заархивированных журнальных файлов.
4) В SQL*Plus смонтировать БД и выдать команду `RECOVER TABLESPACE имя_пространства`. Если потребуется, в простейшем случае отвечать возвратом каретки на каждый последующий вопрос.
5) Получив сообщение о завершении восстановления содержимого файлов, можно снова включить табличное пространство:

   ```
   ALTER TABLESPACE имя_пространства ONLINE;
   ```

12.4.2.5. Пробное восстановление

Если есть подозрения в разрушении файлов пезервных или архивных копий или в достаточности последних, перед реальным восстановлением можно выполнить имитацию, пробное восстановление, которое отлично от реального единственно тем, что Oracle не будет вносить изменения в имеющиеся файлы. Пример пробного восстановления БД:

```
RECOVER DATABASE TEST
```

Пример пробного восстановления БД с допуском пяти испорченых блоков:

```
RECOVER DATABASE TEST ALLOW 5 CORRUPTION
```

Еще примеры:

```
RECOVER DATABASE TEST UNTIL CANCEL

RECOVER TABLESPACE users TEST
```

Сведения об ошибках при пробном восстановлении помещаются в файл *ALERT.ORA*.

12.4.2.6. Режим автовосстановления

Выполняя в SQL*Plus восстановление БД или табличного пространства, можно избавиться от необходимости отвечать на вопросы системы по восстановлению каждого отдельного архивного файла в диалоге. Если указать в SQL*Plus `SET AUTORECOVERY ON` (по умолчанию этот режим отключен), Oracle будет выполнять обращения к архивным файлам автоматически. Того же эффекта можно достичь на уровне команды: `RECOVER` **`AUTOMATIC`** *`далее_как_обычно`*.

Эти возможности удобны в случае, если файлы архивных копий журналов не «ужаты» (скомпрессированы) внешними средствами и находятся в доступности на диске.

12.5. Физическое копирование и восстановление с помощью RMAN

RMAN — программа, входящая в ПО Oracle на всех платформах, начиная с версии 8. Она написана для выполнения физического резервирования и восстановления, но от базовых средств физического резервирования и восстановления *принципиально* отличается тем, что умеет работать не только на уровне файлов, но и блоков данных в табличных пространствах. Вдобавок она поддерживает более естественные для резервирования и восстановления организационную логику и набор понятий для этих процедур. Это обеспечивает ее улучшенную функциональность по отношению к базовым средствам резервирования.

Ее возможности включают следующие:

- выполнение холодного/горячего резервирования, причем во втором случае табличные пространства не переводятся в режим BACKUP, что позволяет избежать дополнительной нагрузки на журнал;
- выполнение полного резервирования и резервирования изменений;
- обнаружение поврежденных блоков или даже нарушений формата хранения данных таблиц и индексов;
- паралельное выполнения операций ввода/вывода.

Уровни выполнения копирования/восстановления с помощью RMAN:

- база данных
- конкретные табличные пространства
- конкретные файлы табличных пространств
- служебные файлы БД (контрольные, архивные, *SPFILE.ORA*)

12.5.1. Пример запасного сохранения (резервирования) и восстановления базы данных

Предположим наличие в файловой системе каталога *d:\oracle\oradata\prima\rman-backups*. Пусть на СУБД целевой БД указывает *ORACLE_SID*. Пусть в силе доверительное подключение к СУБД пользователя SYS.

12.5.1.1. Вход в RMAN и соединение с целевой БД

Перед сохранением и перед восстановлением БД требуется войти в программу RMAN и соединиться с целевой БД. Это можно сделать двумя способами:

```
>rman NOCATALOG
RMAN> CONNECT TARGET /
RMAN>
```

Указание NOCATALOG при вызове RMAN необязательно: оно умолчательно, в противовес своей противоположности CATALOG, требующего указания; далее NOCATALOG будет опускаться.

Второй способ:

```
>rman TARGET /
RMAN>
```

Отсутствие привычного для SYS указания AS SYSDBA вызвано тем, что выполнение сохранения и восстановления по определению долго допускалось лишь обладателю SYSDBA, и явное указание в RMAN этой привилегии становится излишним. С версии 12, однако, существует вторая привилегия, разрешающая эти действия, SYSBACKUP. Она не дает обладателю избыточных прав сверх необходимых для сохранения/восстановления, и поэтому фирма Oracle рекомендует к использованию именно ее. Ее указывать при соединении с целевой БД обязательно:

```
>rman TARGET "'/ AS SYSBACKUP'"
```

Кавычки " и ' можно поменять местами. Они относятся к провозглашению содержимого текста единой лексемой для (а) командной оболочки ОС и (б) программы RMAN.

12.5.1.2. Сохранение и восстановление БД

Простейшие примеры использования RMAN — резервное сохранение БД и ее восстановление. Пример сохранения для полного варианта в синтаксисе версии 8:

```
RMAN> RUN {
2> ALLOCATE CHANNEL d1 TYPE DISK;
3> BACKUP DATABASE
4> FORMAT 'd:\oracle\oradata\prima\rman-backup\%d_%U.bus';
5> }
```

В указанном каталоге ОС появился файл *резервного набора PRIMA_ 02DGA6F0_1_1.BUS* (реальное имя может варьироваться).

Теперь в целях упражнения можно удалить файлы *табличных пространств*, перечисленных в протоколе выполнения программой RMAN резервирования. Это можно сделать, перейдя в состояние MOUNT, для чего с равным успехом годится и RMAN, и SQL*Plus:

```
RMAN> SHUTDOWN IMMEDIATE
RMAN> STARTUP MOUNT
```

После этого можно удалить файлы табличных пространств и выполнить восстановление:

```
RMAN> RUN {
2> ALLOCATE CHANNEL d1 TYPE DISK;
3> RESTORE DATABASE;
4> RECOVER DATABASE;
5> ALTER DATABASE OPEN;
6> }
```

База восстановлена и открыта.

12.5.1.3. Упрощения синтаксиса в версии 9

С версии Oracle 9 в RMAN для выполнения приведенного выше резервирования можно обойтись более краткими формулировками:

```
RMAN> BACKUP DATABASE
2> FORMAT 'd:\oracle\oradata\prima\rman-backup\%d_%U.bus';
```

Аналогично пример упрощенной записи действий по восстановлению:

```
RMAN> RESTORE DATABASE;
RMAN> RECOVER DATABASE;
RMAN> ALTER DATABASE OPEN;
```

Этот синтаксис подразумевает использование *неявного* канала, так что открывать канал отдельной командой стало необязательным. Но если ради производительности администратор захочет воспользоваться двумя или более каналами, придется их указать или в параметрах настройки RMAN, или явными командами.

12.5.1.4. Некоторые общие замечания по технологии копирования/восстановления

- Холодное резервирование/восстановление выполняется в состоянии СУБД MOUNT.
- При работе с каталогом прежде копирования и восстановления понадобится установить связь с каталогом (CONNECT CATALOG).
- Для установки соединения с локальной целевой базой удобно пользоваться переменной среды ОС *ORACLE_SID*.
- Команды RECOVER DATABASE и ALTER DATABASE OPEN выше можно выдать отдельно или даже в SQL*Plus.
- Для Unix полные имена файлов записываются по правилам этой ОС.
- Список возможных условных обозначений в маске имени файла (%d, %U и прочих) приводится в документации.

12.5.2. Другие примеры

Резервирование данных всех файлов табличного пространства:

```
RMAN> BACKUP TABLESPACE system, users;
```

Резервирование данных из конкретных файлов табличного пространства:

```
RMAN> BACKUP DATAFILE 1, 4;
```

или

```
RMAN> BACKUP DATAFILE
2> 'd:\oracle\oradata\prima\system01.dbf',
3> 'd:\oracle\oradata\prima\users01.dbf';
```

Ссылаться на файл можно с равным успехом по номеру и по полному имени.

Резервирование контрольного файла:

```
RMAN> BACKUP CURRENT CONTROLFILE;
```

Резервирование файла параметров СУБД:

```
BACKUP SPFILE;
```

Резервирование архивных копий журнала:

```
RMAN> BACKUP ARCHIVELOG ALL DELETE INPUT;
```

Примеры восстановления:

```
RMAN> RESTORE TABLESPACE users;
RMAN> RECOVER TABLESPACE users;

RMAN> RESTORE DATAFILE 4;
RMAN> RECOVER DATAFILE 4;

RMAN> RESTORE CONTROLFILE;
```

Технологичный пример восстановление на момент в прошлом записывается посредством пакетного задания:

```
RMAN> RUN {
2> SET UNTIL TIME = '2004-05-28:11:44:00';
3> RESTORE DATABASE;
4> RECOVER DATABASE;
5> ALTER DATABASE OPEN RESETLOGS;
6> }
```

Другие примеры копирования и восстановления с помощью RMAN могут быть почерпнуты из документации или из отдельного учебного курса.

13. Дополнительные базовые программные средства для администрирования

Программы логического копирования и восстановления можно использовать для выполнения логического резервного копирования и восстановления, для реорганизации хранения данных и переноса из базы в базу.

13.1. *exp* и *imp*

В составе ПО Oracle имеются две симметричные программы для копирования и восстановления данных на логическом уровне: *exp* (экспорт) и *imp* (импорт).

13.1.1. Общие принципы работы программ *exp* и *imp*

Программы могут исполняться в двух режимах: диалоговом и пакетном. Диалоговый режим проигрывает пакетному в том, что не позволяет воспользоваться всем разнообразием параметров экспорта и импорта. При использовании пакетного режима небольшое количество параметров можно указывать непосредственно в командной строке ОС при вызове программы, а пространный их перечень можно вынести в специальный файл, сославшись на него к вомандной строке.

Обе программы взаимодействуют с БД по схеме клиент-сервер, что требует правильной установки кодировки перед обращении к программам в ОС, например:

```
>set NLS_LANG=.RU8PC866
```

Экспорт и импорт допускается с разными уровнями детализации. Так, программа *exp* позволяет экспортировать в указанный файл:

1) всю БД
2) все объекты из указанного табличного пространства (возможно что разных пользователей)
3) все объекты конкретного пользователя
4) конкретные таблицы указанного пользователя
5) нужные строки конкретных таблиц

13.1.2. Некоторые типовые сценарии

Ниже приводятся некоторые командные сценарии экспорта и импорта. Пакетные экспорт и импорт могут быть дополнены указанием в командной строке PARFILE = *файл_с_параметрами*:

```
>exp scott/tiger PARFILE=export_parameters.par
```

Экспорт всех объектов, хранящихся в табличном пространстве USERS:

```
>exp system/manager TABLESPACES=users
```

Экспорт всех объектов пользователя SCOTT:

```
>exp scott/tiger OWNER=scott
```

Экспорт только сотрудников 10 отдела:

```
>exp scott/tiger TABLES=emp QUERY='WHERE deptno=10'
```

Ускоренный вариант экспорта, при котором данные из БД в файл экспорта переписываются блоками, минуя буфер блоков в СУБД:

```
>exp scott/tiger TABLES=(emp, dept) DIRECT=Y
```

Экспорт, выполняемый в рамках транзакции READ ONLY, предотвращающий рассогласование данных в связанных таблицах из-за работы с ними других пользователей:

```
>exp scott/tiger TABLES=(emp, dept) CONSISTENT=Y
```

Экспорт с указанием имен файлов экспорта и протокола:

```
>exp system/manager OWNER=scott FILE=exp_scott.dmp LOG=exp_scott.log
```

Экспорт без индексов и без строк (только описания только таблиц):

```
>exp scott/tiger INDEXES=N ROWS=N
```

Импорт объектов SCOTT в схему YARD:

```
>imp system/manager FROMUSER=scott TOUSER=yard FILE=exp_scott.dmp
```

Импорт данных без индексов; создать сценарий заведения индексов и сохранить в файле *scott_indexes.sql*:

```
>imp scott/tiger INDEXES=N INDEXFILE=scott_indexes.sql
```

13.1.3. Некоторые параметры настройки

Ниже перечисляются некоторые параметры программ *exp* и *imp*, позволяющие повысить их производительность.

DIRECT

DIRECT = Y позволяет программе *exp* читать блоки данных напрямую из базы («прямоточно»), в обход области буферизации блоков с данными в SGA. Это может дать заметное ускорение. Параметры NLS_LANG обязаны совпадать с имеющимися в БД.

RECORDLENGTH

Размер в байтах массива, которым будет производиться обмен с диском (чтение для *imp* и запись для *exp*). Разумно выбирать кратным размеру блока данных и блока файловой системы.

BUFFER

Размер внутренней служебной области, используемой при передаче данных. В случае *imp* в комбинации с COMMIT = Y будет вызывать автоматический COMMIT после импорта очередной группы строк из этой области, что снижает затраты на импорт больших таблиц.

INDEXES

Параметр позволяет отказаться от импорта индексов, существенно ускоряя импорт. Индексы можно будет воссоздать позже.

13.1.4. Полный экспорт и экспорт изменений

(В версиях *exp*/*imp* для Oracle 9 возможность экспорта изменений упразднена).

Разные значения параметра INCTYPE позволяют указывать разные виды экспорта:

- FULL (полный экспорт) — экспортируются все объекты БД
- CUMULATIVE (кумулятивный экспорт) — экспортируются все объекты БД, измененные с момента последнего полного или кумулятивного экспорта

- INCREMENTAL (разностный экспорт) — экспортируются все объекты, измененные с момента последнего экспорта

13.1.5. Таблицы словаря-справочника для записи информации об экспорте

Если при экспорте указывать параметр RECORD = Y, то информация об этих действиях будет заноситься в специальные представления словаря-справочника, которые впоследствии дадут возможность определить, какие файлы нужны для восстановления требуемой таблицы.

- DBA_EXP_FILES — записи о каждом удачном экспорте с момента полного экспорта
- DBA_EXP_OBJECTS — записи о каждом экспортированном объекте в последнем успешном экспорте
- DBA_EXP_VERSION — номер последнего успешного экспорта (одно число).

Так как представления обновляются при экспорте, для хранения истории таблицы можно копировать в другие таблицы.

13.1.6. Дополнительные достоинства экспорта/импорта

Помимо резервирования и восстановления данных, экспорт и импорт полезны при следующих операциях:

- Перенесение данных с платформы на платформу
- Изменение параметров хранения таблицы. Таблицу можно экспортировать, удалить из базы, создать с новыми параметрами хранения и перезагрузить.
- Перенесение данных одной схемы в другую (в Oracle в открытую не разрешено переподчинять объекты другому пользователю).

13.2. *expdp* и *impdp*

В версии 10 на замену программам *exp* и *imp* введены программы *expdp* и *impdp* («Oracle Data Pump», буквально — насос, то есть перекачка данных), отчасти повторяющие функциональность программ *exp* и *imp*, отчасти подправляющие ее, а отчасти дополняющие. Старые программы *exp* и *imp* продолжают поставляться, но оставлены для выполнения обменов данными с прежними версиями.

Работать с программами *expdp* и *impdp* можно (а) через командную строку ОС (пакетно или же в диалоге), (б) в OEM и (в) в программе с помощью средств пакета DBMS_DATAPUMP.

13.2.1. Общие сведения

Основное технологическое отличие в том, что *expdp* и *impdp* исполняются не на клиенте, а на сервере, не загружая сеть пересылкой данных. Технически загрузка/выгрузка осуществляется отдельным заданием, исполняемым автономным процессом СУБД, и не требует присутствия пользователя у терминала в ожидании своего завершения.

В целях защиты, доступ к файловой системе сервера регулируется посредством объекта вида DIRECTORY в БД; он должен сообщать программе каталог ОС для размещения файла экспорта. Допустим, такой объект типа DIRECTORY уже имеется под названием DATA_PUMP_DIR. Тогда задание на экспорт можно построить так:

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> GRANT READ, WRITE ON DIRECTORY data_pump_dir TO SCOTT;

Grant succeeded.

SQL> HOST expdp scott/tiger DIRECTORY=data_pump_dir TABLES=emp
```

```
Export: Release 10.2.0.1.0 - Production on Thursday,        ...

Copyright (c) 2003, 2005, Oracle.  All rights reserved.

Connected to: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 -
Production
With the Partitioning, OLAP and Data Mining options
Starting "SCOTT"."SYS_EXPORT_TABLE_01":  scott/********      ...
... ... ...
```

Если используется DIRECTORY под названием DATA_PUMP_DIR и имеются приволегии (роль) EXP_FULL_DATABASE, это имя можно не указывать. Пример экспорта выше равносилен следующему:

```
>expdp scott/tiger TABLES=emp
```

Импорт выполняется аналогично.

Выполнение экспорта и импорта требует времени, и если в процессе работы этих программ нажать *Ctrl-C*, они перейдут в режим диалога, позволяющий отслеживать выполнение и останавливать их:

```
SQL> HOST expdp scott/tiger DIRECTORY=data_pump_dir TABLES=emp

Export: Release 10.2.0.1.0 - Production on Thursday,        ...
... ... ...
Ctrl-C
Export> HELP
------------------------------------------------------------

The following commands are valid while in interactive mode.
Note: abbreviations are allowed.

ADD_FILE
Add dumpfile to dumpfile set.

CONTINUE_CLIENT
Return to logging mode. Job will be restarted if idle.
... ... ...
```

В отличие от *exp*, при подготовке файла экспорта для переноса данных под другую версию СУБД программа *expdp* требует явного указания целевого номера версии, например VERSION=10.2.0.

13.2.2. Выборочный экспорт/импорт

Уровни выполнения экспорта и импорта программами *expdp* и *impdp* изначально были те же, что у *exp* и *imp*, однако позже появилась дополнительная избирательность.

Параметр PARFILE обеих программ позволяет сослаться на файл с параметрами экспорта или импорта. В нем удобно использовать прочие параметры выполнения операции, например EXCLUDE — исключить определенные категории объектов, или QUERY — исключить при экспорте строки.

Пример:

```
SCHEMAS=SCOTT
DUMPFILE=scott_exp_monday.dmp
LOGFILE= scott_exp_monday.log
REUSE_DUMPFILES=yes
PARALLEL=2
EXCLUDE=TABLE/INDEX
QUERY=SCOTT.EMP:"where mgr in (select empno from scott.emp where ename =
'KING')"
```

Если этот текст поместить в файл *parexp.exp*, экспорт можно выполнить так:

```
>expdp system/oracle PARFILE=parexp.exp
```

Разнообразие фильтров исключения/включения объектов чрезвычайно велико. Списки дают таблицы DATABASE_EXPORT_OBJECTS, SCHEMA_EXPORT_OBJECTS и TABLE_EXPORT_OBJECTS. Пример справочного запроса:

```
SELECT object_path, named FROM database_export_objects
;
```

Фрагмент ответа:

```
...
TABLE/INDEX                                                              Y
TABLE/INDEX/STATISTICS                                                   N
TABLE/INSTANCE_CALLOUT                                                   N
TABLE/MATERIALIZED_VIEW_LOG                                              Y
TABLE/POST_INSTANCE/GRANT/PROCDEPOBJ_GRANT                               N
TABLE/POST_INSTANCE/PROCDEPOBJ                                           Y
TABLE/POST_INSTANCE/PROCDEPOBJ_AUDIT                                     N
TABLE/POST_TABLE_ACTION                                                  N
TABLE/PRE_TABLE_ACTION                                                   N
...
```

Пример экспорта схемы без таблиц и генераторов чисел (sequences):

```
>expdp system/oracle SCHEMAS=scott EXCLUDE=TABLE EXCLUDE=SEQUENCE
```

Автоматический импорт sequences в случае схемы был когда-то старым неудобством *exp* и *imp.*

13.2.3. Некоторые особые свойства

Ряд параметров *expdp*/*impdp* не имеет аналогов среди параметров *exp*/*imp* по технологическим или же по версионным причинам. В частности, к ним относятся следующие.

JOB_NAME

Выполнение экспорта и импорта программами *expdp*/*impdp* осуществляется путем автоматического запуска задания на сервере. Если не позаботиться об имени, СУБД назовет задание по-своему (смотри пример выше). Параметр JOB_NAME позволяет указать имя задания программисту на свое усмотрение в виде строки текста длиною до 30 байтов (версии 11-). Строка может содержать пробелы, но тогда ее следует заключить в кавычки, иначе необязательные. Например:

```
>expdp scott/tiger DIRECTORY=data_pump_dir JOB_NAME='SCOTT export'
```

ENCRYPTION
ENCRYPTION_ALGORITHM
ENCRYPTION_PASSWORD
ENCRYPTION_MODE

Параметры для шифрования данных в файле экспорта. Технология, подобно применяемой в RMAN, подразумевает прозрачное шифрование, шифрование явным паролем и двойную возможность (значения TRANSPARENT, PASSWORD и DUAL параметра ENCRYPTION_MODE). Пример:

```
>expdp scott/tiger ENCRYPTION=ALL ENCRYPTION_PASSWORD='oracle'
ENCRYPTION_ALGORITHM=AES192
```

13.2.4. Перенос данных

Подобно программам *rman*, *exp* и *imp*, программы *expdp* и *impdp* приспособлены не только для сохранения и восстановления данных, но и переноса. Собственно перенос осуществляется посредством передачи файла экспорта на машину с принимающей БД, однако здесь эта передача может быть возложена на Oracle Net. Это осуществимо при наличии в БД связи с посторонней базой. Пример:

```
CREATE PUBLIC DATABASE LINK oradb.office
 CONNECT TO system
 IDENTIFIED BY oracle
 USING 'oraserver:1521/orcl.office'
;
```

Теперь получить на *своей* машине файл экспорта всех объектов схемы SCOTT *посторонней* базы ORCL.OFFICE можно, выдав в командной строке:

```
>expdp system/oracle DIRECTORY=data_pump_dir NETWORK_LINK="oradb.office"
SCHEMAS=scott COMPRESSION=ALL
```

Параметр COMPRESSION=ALL необязателен и служит для сокращения объема файла экспорта. Кавычки вокруг имени связи обязательны лишь при составном имени.

Следующая команда позволит через Oracle Net через связь ORCL.OFFICE получить описание и данные таблицы *из посторонней* БД и импортировать и то, и другое *в свою*:

```
>impdp system/oracle NETWORK_LINK="oradb.office" TABLES=scott.emp
```

Обратите внимание на отсутствие необходимости обозначать директорию для файла экспорта.

Передача данных между базами описанным выше способом через Oracle Net удобна, однако при больших объемах будет не столь эффективна в сравнении с передачей файла экспорта средствами ОС (через *ftp*, или иным способом).

13.2.5. Программный запуск

Программы *expdp* и *impdp*, будучи выполняемы СУБД, имеют программный интерфейс обращения. Он реализован пакетом DBMS_DATAPUMP и доступен для употребления при наличии у пользователя приписанных ему ролей DATAPUMP_EXP_FULL_DATABASE и DATAPUMP_IMP_FULL_DATABASE.

Пример экспорта:

```
CONNECT / AS SYSDBA
SET SERVEROUTPUT ON

DECLARE
  dp_handle NUMBER;   -- переменная для ссылки на структуру описания перекачки данных
BEGIN
  dp_handle := DBMS_DATAPUMP.OPEN (       -- получили переменную «перекачки данных»
    operation   => 'EXPORT'
  , job_mode    => 'SCHEMA'
  , remote_link => NULL
  , job_name    => 'SCOTT export'
  , version     => 'LATEST'
  );
  DBMS_DATAPUMP.SET_PARALLEL (                -- примеры уточнений ...
    handle => dp_handle
  , degree => 2
  );
  DBMS_DATAPUMP.ADD_FILE (
```

```
    handle    => dp_handle
  , filename  => 'SCOTT.dmp'
  , directory => 'DATA_PUMP_DIR'
  );
  DBMS_DATAPUMP.METADATA_FILTER (
    handle => dp_handle
  , name   => 'SCHEMA_EXPR'
  , value  => q'[= 'SCOTT']'
  );
  DBMS_DATAPUMP.START_JOB ( dp_handle ); -- запустили задание на перекачку
  DBMS_DATAPUMP.DETACH    ( dp_handle ); -- освободили переменную перекачки
END;
/

SELECT * FROM dba_datapump_jobs;
-- проверка состояния задания
```

13.2.6. Драйвер перекачки данных в таблицах с внешним хранением данных

Встроенная в программы *expdp* и *impdp* машина перекачки данных находит применение также в виде драйвера ORACLE_DATAPUMP при создании таблиц с внешним хранением данных («внешних таблиц»). Заполнив однократно такую таблицу данными, программист может после этого ее произвольное количество раз читать, но не изменять. Сами же данные при создании таблицы экспортируются во внешний файл, а при обращении к ее содержимому импортируются из файла; то есть в БД будет храниться только описание таблицы, но не ее данные. Тем самым предоставляется возможность создавать в разные моменты времени снимки содержимого одной и той же таблицы, и обращаться к этим снимкам, не требуя для них места в БД и ресурсов буферизации блоков в СУБД. Если перенести файл с экспортированными данными на другую установку, их можно будет читать в табличной форме и в другой БД.

Тем самым может достигаться частичное решение задачи сохранения данных на логическом уровне и их переноса.

Примеры создания таблицы с внешним хранением данных с применением драйвера ORACLE_DATAPUMP:

```
CREATE TABLE ext_emp
ORGANIZATION EXTERNAL (
    TYPE ORACLE_DATAPUMP
    DEFAULT DIRECTORY data_pump_dir
    LOCATION ( 'emp_export.dmp' )
)
AS SELECT * FROM emp
;
```

В результате приведенной команды в каталоге ОС, на который ссылается каталог Oracle по имени DATA_PUMP_DIR, появится файл с данными. Формат файла закрытый и понятен только драйверу ORACLE_DATAPUMP, однако описательная часть его представляет собою текст в формате XML.

```
SQL> SELECT COUNT ( * ) FROM ext_emp;

  COUNT(*)
----------
        14
```

Обращаться к таблице EXT_EMP командой SELECT можно без ограничений.

Пример, показывающий, как таблицы подобного рода можно использовать для переноса данных путем копирования файла экспорта (в нашем случае это *emp_export.dmp*):

```
CREATE TABLE ext_emp2 (
  empno    NUMBER ( 4 )
, ename    VARCHAR2 ( 10 )
, job      VARCHAR2 ( 9 )
, mgr      VARCHAR2 ( 10 )
, hiredate DATE
, sal      NUMBER ( 7, 2 )
```

```
, comm     NUMBER ( 7, 2 )
, deptno   NUMBER ( 2 )
)
ORGANIZATION EXTERNAL (
    TYPE ORACLE_DATAPUMP
    DEFAULT DIRECTORY data_pump_dir
    LOCATION ( 'emp_export.dmp' )
);
--  проверка:
SELECT COUNT ( * ) FROM ext_emp2;
```

Потребительскую ценность подобных таблиц увеличивает, сверх приведенных основных свойств, ряд дополнительных. Так, допускается ссылка на несколько файлов экспорта:

```
ALTER TABLE ext_emp2 LOCATION ( 'emp_export.dmp', 'emp_export.dmp' );
```

Проверка:

```
SQL> SELECT COUNT ( * ) FROM ext_emp2;

  COUNT(*)
----------
        28
```

Файлы не обязаны находиться в общем каталоге, но ссылку на неумолчательный каталог следует обозначить явно, например:

```
ALTER TABLE ext_emp2
    LOCATION ( 'emp_export.dmp', my_work_dir:'emp_export.dmp' )
;
```

Создание и чтение данных можно распараллелить, что особенно выгодно при наличии нескольких файлов с данными:

```
ALTER TABLE ext_emp2 PARALLEL 2;
```

Примеры указания «предела терпимости» количества ошибок доступа к данным и отключения протоколирования ошибок доступа:

```
ALTER TABLE ext_emp2 REJECT LIMIT 10
;
ALTER TABLE ext_emp2 ACCESS PARAMETERS ( NOLOGGIN )
;
```

С версии 11.2 разрешено также при создании подобных таблиц задать уплотненное хранение данных в файлах и шифровать содержимое:

```
CREATE TABLE ext_emp3
ORGANIZATION EXTERNAL (
    TYPE ORACLE_DATAPUMP
    DEFAULT DIRECTORY data_pump_dir
    LOCATION ( 'emp_export3.dmp' )
    ACCESS PARAMETERS ( COMPRESSION ENABLED
                        ENCRYPTION ENABLED )
)
AS SELECT * FROM emp
;
```

Замечание. Шифрование в этом случае выполняется средствами TDE («прозрачного шифрования данных»), а оно требует наличия бумажника (Oracle wallet), находящегося в открытом состоянии во время выполнения действий по шифрованию и дешифрованию.

Программа *expdp*, экспортируя таблицу, во время работы фактически отрабатывает выполнение следующей команды:

```
CREATE TABLE <рабочее название> (<список столбцов>)
ORGANIZATION EXTERNAL
(TYPE ORACLE_DATAPUMP
```

```
 DEFAULT DIRECTORY ...
 LOCATION ('bogus.dat')
)
 ... REJECT LIMIT UNLIMITED
AS SELECT <выборка строк из таблицы в БД>
```

13.3. SQL*Loader

13.3.1. Общая информация

SQL*Loader — средство загрузки данных в систему, удобное для загрузки больших объемов. Ниже перечисляются некоторые его полезные свойства:

- SQL*Loader позволяет выполнять загрузку из множества разных файлов, в том числе в разных форматов
- Позволяет использовать во входном файле фиксированный формат, формат с разделителем и формат переменной длины строки
- Выполнять указанные SQL-предложения перед загрузкой строки в БД
- Комбинировать несколько входных записей в одну запись БД
- Автоматически создавать значения уникальных ключей при загрузке

При работе SQL*Loader может, или обязан использовать файлы разных типов:

- *Контрольный файл*. Файл с информацией, задающих режим работы SQL*Loader (имена, места расположения, характеристики и форматы)
- *Журнальный файл*. Выходной файл с перечислением режимов загрузки и встреченных ошибок
- *Файл некорректно оформленных записей*. Выходной файл с записями, которые не загружались по причине некорректного формата
- *Файл записей, отклоненных БД*. Выходной файл с записями, отклоненными загрузчиком (отфильтрованными по заданным критериям, например из-за нарушения ограничений целостности).

13.3.2. Загрузка данных в гибком формате

Пусть надо загрузить данные в таблицу EMPLOAD с той же структурой, что и у EMP. Пусть в каталоге *c:\loaderdata* имеется файл *employee.txt* со следующими данными:

```
1000,"Clinton","President",,21/12/1999,3000,,40
1010,"Al Gore","VicePres",1000,23/12/1999,2000,,40
```

Пусть в этом же каталоге имеется управляющий файл *load.ctl* со следующим содержанием:

```
LOAD DATA
INFILE 'c:\loaderdata\employee.txt'
INTO TABLE empload
(
 empno    INTEGER EXTERNAL TERMINATED BY ',',
 ename    CHAR TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"',
 job      CHAR TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"',
 mgr      INTEGER EXTERNAL TERMINATED BY ',',
 hiredate DATE "DD/MM/YYYY" TERMINATED BY ',',
 sal      DECIMAL EXTERNAL TERMINATED BY ',',
 comm     DECIMAL EXTERNAL TERMINATED BY ',',
 deptno   INTEGER EXTERNAL TERMINATED BY ','
)
```

Создадим пустую таблицу EMPLOAD:

```
SQL> DROP TABLE empload;
```

```
SQL> CREATE TABLE empload AS SELECT * FROM emp WHERE 1 = 2;
```

Войдем в каталог *c:\loaderdata* и выдадим команду:

```
>sqlldr scott/tiger CONTROL=load
```

По ней произойдет загрузка таблицы EMPLOAD.

Упражнение. Создать файлы *employee.txt*, *load.ctl*, пустую таблицу EMPLOAD и выполнить ее загрузку данными. Посмотреть содержание автоматически образовавшегося файла *load.log*, а также таблицы EMPLOAD.

13.3.3. Загрузка данных в фиксированном формате

Пусть имеется файл *employeef.txt* со следующими данными:

```
1000 Clinton   President      21/12/1999 +3000.50          40
1010 Gore      Vice-Pres 1000 23/12/1999 +2000.00          40
```

Пусть имеется управляющий файл *loadf.ctl* со следующим содержанием:

```
LOAD DATA
INFILE 'c:\loaderdata\employeef.txt'
REPLACE INTO TABLE empload
(
 empno    POSITION (1:4)   INTEGER EXTERNAL,
 ename    POSITION (6:15)  CHAR,
 job      POSITION (16:24) CHAR,
 mgr      POSITION (26:29) INTEGER EXTERNAL NULLIF mgr=BLANKS,
 hiredate POSITION (31:40) DATE "DD/MM/YYYY",
 sal      POSITION (42:49) DECIMAL EXTERNAL,
 comm     POSITION (51:58) DECIMAL EXTERNAL NULLIF comm=BLANKS,
 deptno   POSITION (60:61) INTEGER EXTERNAL
)
```

(*Замечание*: указание REPLACE затрет перед загрузкой уже имеющиеся данные таблицы, выдав DELETE. Если необходимо затереть имеющиеся данные SQL-командой TRUNCATE, вместо REPLACE следует указать TRUNCATE)

Находясь в каталоге ОС *c:\loaderdata* выдадим команду:

```
>sqlldr scott/tiger CONTROL=loadf
```

Произойдет загрузка таблицы EMPLOAD.

Упражнение. Создать файлы *employeef.txt*, *loadf.ctl*, пустую таблицу EMPLOAD и выполнить загрузку ее данными. Посмотреть содержание автоматически образовавшегося файла *loadf.log*, а также таблицы EMPLOAD.

13.3.4. Загрузка данных в столбцы типа LOB

SQL*Loader позволяет загружать не только данные, представленные в текстовом формате, в «обычные» столбцы, но и данные в столбцы типа LOB.

Добавим в EMPLOAD столбец:

```
SQL> ALTER TABLE empload ADD ( eface BLOB DEFAULT EMPTY_BLOB ( ) );
```

Создадим файл *c:\loaderdata\newemps.txt*:

```
1020,Bush,NewPres,,13/03/2001,5000,c:\loaderdata\bush.jpg
1021,Powell,Secretary,1020,14/03/2001,4000,c:\loaderdata\powell.jpg
```

Управляющий файл *loadlob.ctl* создадим так:

```
LOAD DATA
INFILE 'c:\loaderdata\newemps.txt'
APPEND INTO TABLE empload
FIELDS TERMINATED BY ','
(
 empno, ename, job, mgr, hiredate, sal,
 external_filename FILLER,
 eface             LOBFILE(external_filename) TERMINATED BY EOF
)
```

(*Замечание*: в этом примере формат некоторых полей опущен; поле заполнителя external_filename играет вспомогательную роль и в базу не грузится; подробнее см. документацию по SQL*Loader).

Тогда загрузка может быть выполнена по следующей команде:

```
>sqlldr scott/tiger CONTROL=loadlob
```

13.3.5. Таблицы с драйвером ORACLE_LOADER загрузчика данных

Иногда удобно не загружать из файла данные в таблицу, а оставлять их в файле, но обращаться в Oracle как к данным обычной таблицы. С этой целью можно воспользоваться таблицами с внешним хранением данных («внешними таблицами»). Связь с данными в таких таблицах обеспечивается драйверами двух видов: ORACLE_LOADER и ORACLE_DATAPUMP. Драйвер ORACLE_LOADER использует код загрузчика SQL*Loader.

Для таблиц, построенных с помощью драйвера ORACLE_LOADER, исходные данные берутся чаще всего из текстового файла. С версии 11.2, кроме того, в определении таких таблиц разработчику разрешено сослаться на «препроцессор», то есть на свою собственную программу преобразования, так что исходные данные уже не обязаны иметь текстовый вид.

Далее приводится пример одностороннего отображения «файл → таблица» с использованием драйвера загрузчика данных. Подразумевается, что действия выполняются на сервере (единственное место, где это используется — при правке содержимого файла с исходными данными командами HOST в SQL*Plus и *echo* командной оболочки ОС).

Для обращения к файлу с данными *employee.txt* требуется создать в БД каталог, например, следующим образом:

```
CONNECT / SA SYSDBA
CREATE DIRECTORY extfiles_dir AS ' c:\loaderdata';
GRANT READ ON DIRECTORY extfiles_dir TO scott;
```

Заведем в схеме SCOTT таблицу с внешним хранением:

```
CREATE TABLE emp_load
  ( empno    NUMBER ( 4 )
  , ename    VARCHAR2 ( 10 )
  , job      VARCHAR2 ( 9 )
  , mgr      NUMBER ( 4 )
  , hiredate DATE
  , sal      NUMBER ( 7, 2 )
  , comm     NUMBER ( 7, 2 )
  , deptno   NUMBER ( 2 )
  )
ORGANIZATION EXTERNAL
 ( TYPE ORACLE_LOADER
   DEFAULT DIRECTORY extfiles_dir
```

```
   ACCESS PARAMETERS
    ( RECORDS DELIMITED BY NEWLINE
      NOBADFILE
      NOLOGFILE
      FIELDS TERMINATED BY ','
      MISSING FIELD VALUES ARE NULL
        ( empno
        , ename    CHAR ( 10 ) ENCLOSED BY '"'
        , job      CHAR ( 9 ) ENCLOSED BY '"'
        , mgr
        , hiredate CHAR ( 10 ) DATE_FORMAT DATE MASK "dd/mm/yyyy"
        , sal      DECIMAL EXTERNAL
        , comm     DECIMAL EXTERNAL
        , deptno
        )
    )
   LOCATION ( 'employee.txt' )
 )
;
```

Проверка в SQL*Plus:

```
SQL> SELECT * FROM emp_load;

    EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
--------- ---------- --------- ---------- --------- ---------- ---------- ----------
     1000 Clinton    President            21-DEC-99       3000                    40
     1010 Al Gore    VicePres        1000 23-DEC-99       2000                    40

SQL> HOST echo 1020,"Hussein","Manager",1000,15/11/2005,1000,2000,30 >> employee.txt

SQL> /

    EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
--------- ---------- --------- ---------- --------- ---------- ---------- ----------
     1000 Clinton    President            21-DEC-99       3000                    40
     1010 Al Gore    VicePres        1000 23-DEC-99       2000                    40
     1020 Hussein    Manager         1000 15-NOV-05       1000       2000         30

SQL> SELECT SUM ( sal ) FROM emp_load;

  SUM(SAL)
----------
      6000
```

Во втором запросе результат вычисляется по ходу прочтения файла *employee.txt*, то есть не требуется полной загрузки данных этого файла в БД для вычисления результата.

Конструкция LOCATION в определении таблицы допускает задание списка файлов или же переопределение файлов-источников, например:

```
SQL> HOST echo 1030,"Laden","Analyst",1010,13/03/2005,,1500.50 > centralasia.txt

SQL> ALTER TABLE emp_load LOCATION ( 'employee.txt', 'centralasia.txt' );

Table altered.

SQL> SELECT * FROM emp_load;

    EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
--------- ---------- --------- ---------- --------- ---------- ---------- ----------
     1000 Clinton    President            21-DEC-99       3000                    40
     1010 Al Gore    VicePres        1000 23-DEC-99       2000                    40
     1020 Hussein    Manager         1000 15-NOV-05       1000       2000         30
     1030 Laden      Analyst         1010 13-MAR-05                  1500.5
```

Это позволяет переориентировать таблицу EMP_LOAD на другие файлы по мере подготовки их внешними программами приложения. Переопределять разрешено и DEFAULT DIRECTORY.

Еще пример свойств. Указания NOBADFILE и NOLOGFILE можно заменить на противоположные (и подразумеваемые молчаливо), в результате чего в каталоге начнут появляться протокольные файлы доступа к данным из БД. Это, однако, потребует дополнительно привилегии WRITE для SCOTT на использование каталога EXTFILES_DIR (для предыдущих действий хватало привилегии READ). Указание REJECT LIMIT сообщит предельное количество игнорируемых нарушений формата в записях из файлов-источников, обнаруживаемых в процессе выполнгения SELECT:

```
ALTER TABLE emp_load ACCESS PARAMETERS (
   RECORDS DELIMITED BY NEWLINE
   BADFILE
   LOGGING
);

ALTER TABLE emp_load REJECT LIMIT 20;
```

Здесь, пока нарушений формата количеством менее 21-го, СУБД не будет порождать ошибку доступа, а вместо этого информацию о нарушениях будет заносить о протокольный файл.

14. Объекты словаря-справочника

14.1. Статические таблицы, используемые в администрировании

Исходные, основные таблицы: OBJ$, TAB$, COM$, IND$, AUD$, PROPS$ и другие.

Некоторые *виртуальные* таблицы («представления») словаря-справочника с информацией об объектах БД:

Таблица	***Содержание***
… начинающаяся с USER_	Информация об объектах пользователя USER
… начинающаяся с ALL_	Информация об объектах, к которым пользователю USER предоставлен доступ
… начинающаяся с DBA_	Информация о всех объектах БД
DICTIONARY	Комментированный список имен таблиц словаря-справочника
...	...

14.2. Динамические таблицы, используемые в администрировании

«Динамические таблицы производительности» дают АБД значения внутренних переменных (структур, массивов) SGA СУБД (в малой части — записей из контрольного файла, например, V$DATABASE, и файла паролей, например, V$PWFILE_USERS) в табличной форме, то есть в виде, допускающем к ним обращение запросами SQL.

Исходно значения внутренних переменных предоставляют таблицы с префиксом X$ («X$-таблицы»). Их недостаток есть продолжение их преимуществ — эти значения неудобны для понимания АБД. X$-таблицы предназначены фирмой Oracle для разработчиков; их состав и структура могут слегка меняться от версии к версии, и иногда (хотя и нечасто) такие таблицы в новых версиях пропадают.

Для пользователей Oracle фирма-изготовитель предназначает «V$-таблицы» и «GV$-таблицы». Это уже выводимые таблицы, представления, построенные на основе обращений к X$-таблицам. Они намного более понятны обычному пользователю и в целом имеют более стабильный жизненный цикл. GV$-таблицы фактически дублируют V$-таблицы и представляют собою варианты последних для кластерной конфигурации БД (RAC). (Но есть и исключения: например, таблица V$OBJECT_USAGE не имеет GV$-варианта.)

Ввиду своеобразной природы X$/V$/GV$-таблиц (они не хранят данные своих строк в сегментах табличных пространств), справочная информация о них выдается также особым образом: не через таблицы словаря справочника OBJ$, TAB$, VIEW$, USER_OBJECTS и др., а через таблицы:

- V$FIXED_TABLE — описание X$/V$/GV$-таблиц;
- V$FIXED_VIEW_DEFINITION — описание определений V$/GV$-таблиц;
- X$KQFTA — отдельно описание X$-таблиц.

Примеры:

```
SYS> SELECT name FROM v$fixed_table WHERE name LIKE 'V$%ADVICE%';

NAME
------------------------------
V$SHARED_POOL_ADVICE
V$JAVA_POOL_ADVICE
V$STREAMS_POOL_ADVICE
V$PX_BUFFER_ADVICE
```

```
V$MEMORY_TARGET_ADVICE
V$SGA_TARGET_ADVICE
V$DB_CACHE_ADVICE
V$MTTR_TARGET_ADVICE
V$PGA_TARGET_ADVICE_HISTOGRAM
V$PGA_TARGET_ADVICE

SYS> COLUMN view_definition FORMAT A70 WORD
SYS> SELECT view_definition
  2  FROM   v$fixed_view_definition
  3  WHERE  view_name = 'V$SHARED_POOL_ADVICE'
  4  ;

VIEW_DEFINITION
----------------------------------------------------------------------
select shared_pool_size_for_estimate, shared_pool_size_factor,
estd_lc_size, estd_lc_memory_objects, estd_lc_time_saved,
estd_lc_time_saved_factor, estd_lc_load_time,
estd_lc_load_time_factor, estd_lc_memory_object_hits  from
gv$shared_pool_advice where inst_id = USERENV('Instance')

  1  SELECT view_definition
  2  FROM   v$fixed_view_definition
  3* WHERE  view_name = 'GV$SHARED_POOL_ADVICE'
  4  /

VIEW_DEFINITION
----------------------------------------------------------------------
select inst_id, sp_size, round(sp_size / basesp_size, 4),
kglsim_size, kglsim_objs, kglsim_timesave,
decode(kglsim_basetimesave, 0, to_number(null),
round(kglsim_timesave / kglsim_basetimesave, 4)),  kglsim_parsetime,
decode(kglsim_baseparsetime, 0, to_number(null),
round(kglsim_parsetime / kglsim_baseparsetime, 4)),  kglsim_hits from
x$kglsim
```

Хотя в основном для работы АБД хватает V$/GV$-таблиц, изредка за нужными сведениями приходится обращаться напрямую к X$-таблицам. Вот несколько примеров.

Список последних 20-ти извещений, помещенных в файлы *ALERT.LOG*/*log.xml*:

```
COLUMN indx                  FORMAT 9999999
COLUMN originating_timestamp FORMAT A32
COLUMN message_text          FORMAT A70 WORD
SELECT *
FROM   ( SELECT   indx, originating_timestamp, message_text
         FROM     x$dbgalertext
         ORDER BY originating_timestamp DESC)
WHERE ROWNUM <= 20
/
```

Общее количество недокументированных параметров СУБД (их имена начинаются с '_'):

```
COLUMN name        FORMAT A42
COLUMN value       FORMAT A40 WORD_WRAPPED
COLUMN description FORMAT A60 WORD_WRAPPED

WITH params AS (
SELECT
     x.ksppinm      AS name
   , y.kspftctxvl   AS value
   , x.ksppdesc     AS description
   , y.kspftctxdf   AS isdefault
   , DECODE ( BITAND ( y.kspftctxvf, 7 ), 1, 'MODIFIED', 4, 'SYSTEM_MOD', 'FALSE' )
                    AS ismod
   , DECODE ( BITAND ( y.kspftctxvf, 2 ), 2, 'TRUE', 'FALSE' )
                    AS isadj
```

```
FROM
     sys.x$ksppi   x
   , sys.x$ksppcv2 y
WHERE
     x.inst_id  = USERENV ( 'Instance' )
 AND y.inst_id  = USERENV ( 'Instance' )
 AND x.indx + 1 = y.kspftctxpn
ORDER BY TRANSLATE ( x.ksppinm, ' _', ' ' )
)
SELECT COUNT ( * ) FROM params
/
```

Упражнение. Выдать названия, текущие значения и краткие определения скрытых параметров СУБД, имена которых начинаются с '__'. (Преимущественно это параметры, способные динамически изменять свои значения по ходу работы СУБД.)

Длительность нахождения каждого блока с данными в буфере в SGA определяется собственным счетчиком доступа. Счетчик увеличивает свое значение при обращениях СУБД к блоку и уменьшает при отсутствии обращений. Высокое значение счетчика блока делает маловероятным его скорый уход из SGA и говорит о большой востребованности данных в этом блоке в текущий момент. Значение счетчика каждого блока дает столбец TCH таблицы X$BH, и никак иначе, в том числе из V$BH, его не узнать. Запрос на десятку самых горячих блоков в буфере SGA и их принадлежности можно составить следующим образом:

```
COLUMN object FORMAT A60
WITH touched AS  (
   SELECT   tch, file#, dbablk, obj
   FROM     x$bh
   WHERE    state <> 0
   ORDER BY tch DESC
)
SELECT
      tch
    , file#
    , dbablk
    , CASE WHEN obj = 4294967295 THEN 'UNDO block'
           ELSE ( SELECT MAX ( object_type || ': ' || owner || '.' || object_name )
                         || CASE COUNT ( * ) WHEN 1 THEN NULL ELSE ' -probably' END
                  FROM   dba_objects
                  WHERE  data_object_id = obj )
      END object
FROM  touched
WHERE ROWNUM <= 10
/
```

Номер блока X$BH.OBJ = 4294967295 фиктивный, он соответствует шестнадцатеричному значению FFFFFFFF, которое используется в SGA для блоков UNDO (максимум из восьмибайтового целого):

```
SQL> SELECT TO_CHAR ( 4294967295, 'XXXXXXXX' ) hexadecimal FROM dual;

HEXADECIM
---------
 FFFFFFFF
```

15. Планирование автоматического выполнения заданий

Некоторые задания по администрированию системы, например резервное копирование или сбор статистики, желательно поставить на автоматическую основу. Это можно сделать:

- средствами Oracle:
  - простой вариант — с использованием системного пакета DBMS_JOB, запускающего программы на PL/SQL наподобие cron в Unix,
  - средствами намного более развитого пакета DBMS_SCHEDULER[10-),
  - частично посредством программы *adrci*[11-)
- средствами ОС,
- посредством Oracle Enterprise Manager (OEM).

[10-) С версии Oracle 10.
[11-) С версии Oracle 11.

15.1. Пример автоматического выполнения средствами ОС

Ниже приводится одна из возможных схем организации такого выполнения заданий, отличная от прочих своей простотой и универсальностью (так как выглядит одинаково, с точностью до некоторых технических расхождений, в среде и Windows, и Unix).

Пусть требуется раз в день осуществлять полный экспорт данных всех пользователей, исключая системных. Для этого можно выполнить следующие шаги (пример приведен для Windows).

Для порождения задания на экспорт программой *exp* можно создать следующий файл под названием *genexp.txt*:

```
set heading off
spool c:\exports\lastexp.bat

SELECT 'exp userid=system/manager FILE=c:\exports\'
      || username || to_char ( SYSDATE, 'ddmmyyyy' ) || '.dmp OWNER='
      || username
FROM dba_users
WHERE username NOT IN ( 'SYS', 'SYSTEM', 'DBSNMP', 'SYSMAN' );
```

Для подготовки файла *genexp.txt* и выполнения экспорта можно завести файл *doexp.bat* со следующим содержимым:

```
start /wait sqpplus –s c:\exports\genexp.txt
start /wait c:\exports\lastexp.bat
```

Подготовить задание *doexp.bat* для автозапуска можно командой ОС:

```
>at \\winserv 20:15 /every:Monday,Thursday "c:\exports\doexp.bat"
```

Наконец, поставить задание в очередь можно, выдав в командной строке ОС:

```
>at
```

(*Замечание*: команда *at* имеется как в Unix, так и в Windows, различаясь некоторыми деталями своего оформления. В Unix, помимо *at*, можно воспользоваться более популярной для этой ОС командой *cron*).

Этот пример легко модифицируется для выполнения, к примеру, горячего резервирования и даже холодного, включающего останов и старт СУБД (файлы *savefiles.sql* и *docold.sql* в основном тексте материала).

15.2. Автоматическое выполнение средствами Oracle

15.2.1. Использование пакета DBMS_JOB

Пакет DBMS_JOB дает возможность программно планировать задания с помощью «старого» планировщика заданий, действующего ныне по правилу сохранения обратной совместимости. Если пакет не установлен при создании БД (что практически исключено), его можно установить с помощью сценария *dbmsjob.sql* в *%ORACLE_HOME%\rdbms\admin*.

В последних версиях Oracle пакет поддерживается только для обратной совместимости, однако его достоинством остается простота.

Пусть имеется программа на PL/SQL, требующая регулярного выполнения. Примером может служить программа *statspack.snap* из среды Statspack (версии 10-). Поставить ее на ежедневное автоматическое выполнение можно так:

```
DECLARE
   jobno NUMBER;
BEGIN
   DBMS_JOB.SUBMIT (
     what      => 'BEGIN STATSPACK.SNAP; END;'
   , next_date => SYSDATE
   , interval  => 'SYSDATE + 1 / 24'
   , job       => jobno
   );
   COMMIT;
   DBMS_OUTPUT.PUT_LINE ( 'New job system ID is: ' || jobno );
END;
/
```

Для того, чтобы это работало, в *INIT.ORA* или в *SPFILE.ORA* должно быть:

JOB_QUEUE_PROCESSES = *n*

где *n* от 1 до 36. (Другие два параметра, имеющие отношение к работе пакета DBMS_JOB — JOB_QUEUE_INTERVAL и JOB_QUEUE_KEEP_CONNECTIONS. В последних версиях Oracle первый из них сделан скрытым, а второй устарел).

Следить за работой заданий Oracle можно с помощью таблиц DBA_JOBS, USER_JOBS и DBA_JOBS_RUNNING (создаются сценарием *catjobj.sql*).

15.2.2. Использование пакета DBMS_SCHEDULER

В версии 10 в СУБД появился более развитый планировщик заданий, программная работа с которым обеспечивается средствами пакета DBMS_SCHEDULER. Планировщик версии 10 позволяет запускать на исполнение не только внутренние процедуры, но и программы в ОС.

Планировщик использует следующие понятия:

- *Schedule* (расписание)
- *Program* (программа)
- *Job* (плановое задание = расписание + программа)
- *Window* (интервал для включения разных ресурсных планов)

- Некоторые другие.

«Программа» может быть хранимой процедурой PL/SQL или Java, внешней процедурой на C, блоком PL/SQL или же командой ОС. Запускать задания можно по принципу пакета DBMS_JOB, но с бо́льшим разнообразием формулировок, например:

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name   => 'MY_JOB'
, job_type   => ' STORED_PROCEDURE'
, job_action => 'PERFSTAT.STATSPACK.SNAP'
, enabled    => TRUE
);
END;
/
```

Указанная команда ОС будет выполнена однократно, после чего задание автоматически исчезнет. Добавление следующих параметров приведет к тому, что процедура будет выполняться в текущее (на момент запуска) время суток каждое воскресение:

```
, start_date      => SYSTIMESTAMP
, repeat_interval => 'FREQ=WEEKLY; BYDAY=SUN'
```

В то же время можно заранее создать «программу» и «расписание» процедурами CREATE_PROGRAM и CREATE_SCHEDULE, после чего выдать:

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name      => 'MY_JOB'
, program_name  => 'MY_PROGRAM'
, schedule_name => 'MY_SCHEDULE'
);
END;
/
```

Справочная информация — в таблицах %SHEDULER%. Для работы планировщика в ОС должна быть запущена программа *extjob*. На Windows она запускается службой *OracleJobScheduler*ORACLE_SID. Возможность формулировать плановые задания и осуществлять сопутствующие действия регулируется серией системных привилегий (например, CREATE JOB) и предопределенной ролью SCHEDULER_ADMIN.

15.3. Автоматическое выполнение заданий с использованием Oracle Enterprise Manager

Имеющаяся в составе OEM консоль содержит пункты меню для формулирования и запуска заданий (jobs) для баз данных, распределенных в сети. Эти задания соответствуют «новому» планировщику (версии 10), систему понятий которого и поддерживает консоль OEM.

15.4. Автоматическое выполнение заданий комбинированными средствами

При необходимости можно организовать сочетание внутренних и внешних по отношению к СУБД способов запуска плановых заданий. Это позволяет сделать пакет DBMS_PIPE, разрешающий взаимодействие процессов СУБД Oracle и ОС.

16. Работа с файлами диагностики при помощи программы *adrci*

Программа *adrci* была введена в версии Oracle 11 в помощь АБД для работы с файлами диагностики серверного ПО. В этой версии появилась новая техника ADR организации накопления и обработки диагностической информации. Последняя собирается в виде разнообразных файлов в подкаталогах опорного каталога *$ORACLE_BASE/diag*. На его полное название указывает параметр СУБД DIAGNOSTIC_DEST. В частности, в подкаталогах *$ORACLE_BASE/diag* хранятся файлы диагностики работы процесса *listener*, файлы СУБД и экземпляра ASM, включая *ALERT.LOG* (*log.xml*), **.trc* и файлы содержимого памяти в аварийных ситуациях, файлы «инцидентов». В случае кластерной конфигурации RAC диагностические файлы накапливаются в подкаталогах *$ORACLE_BASE/diag* по каждому работающему экземпляру СУБД отдельно.

С версии 11 данные из *ALERT.LOG* дублируются в ADR в файле *log.xml* в формате XML. *adrci* чаще всего обращается именно к нему.

Программа *adrci* позволяет, пользуясь собственным командным языком, выявлять из обилия диагностических файлов нужную информацию, но в добавок она еще берет на себя задачу автоматического удаления устаревших файлов диагностики из файловой системы. Последнее обстоятельство крайне полезно для АБД.

Программа *adrci* разработана в соответствии с описанием управления инцидентами (incident management), определенном библиотекой службами IT ITSM (IT Service Management; осуществление услуг, предоставляемых клиенту, и управление ими). Заимствованными понятиями являются *инцидент*, *проблема*, *известная ошибка* и некоторые другие.

Основные понятия *adrci*:

- *Проблема* — критическая ошибка, способная возникнуть по ходу работы ПО Oracle (например ORA-00600, -07445, -04031 и др.). Такие ошибки протоколируются в ADR, в случае СУБД в том числе в *ALERT.LOG*/*log.xml*.
- *Инцидент* — конкретное возникновение критической ошибки. Одна и та же проблема проявляет себя в одном или нескольких инцидентах; в то же время повторяющийся инцидент образует проблему.
- *Пакет инцидентов* — логическое собрание данных об инцидентах одной или нескольких проблем, формируемое для отсылки в службу поддержки Oracle (Oracle support service).

Далее приводятся некоторые полезные возможности этой программы.

16.1. Настройки и просмотр сведений

Команда HELP дает подсказку к употреблению прочих команд. Опорный каталог:

```
adrci> SHOW BASE
ADR base is "c:\app\oracle"
```

Пример сведений о каталогах с файлами имеющихся компонент ПО:

```
adrci> SHOW HOME
ADR Homes:
diag\rdbms\orcl\orcl
diag\tnslsnr\winxp\backlistener
diag\tnslsnr\winxp\listener
```

Настройка на работу с файлами определенной компоненты:

```
adrci> SET HOME diag\rdbms\orcl\orcl
adrci>
```

Показать последние 15 строк файла *ALERT.LOG*:

```
adrci> SHOW ALERT -tail 15
SMON: enabling tx recovery
Database Characterset is CL8MSWIN1251
2013-02-21 15:49:24.796000 +03:00
...
```

Если добавить ключ –f (SHOW ALERT –tail 15 **-f**), программа не вернет управление пользователю, но будет сама автоматически перевыдавать данные по мере поступления новых строк в *ALERT.LOG*/*log.xml*.

Для извлечения из *ALERT.LOG*/*log.xml* не всего подряд, а определенной категории данных, имеются набор условных обозначений этих категорий и правила построения условных выражений отбора. Например:

```
adrci> SHOW ALERT -p "message_text like '%ORA-%'"

ADR Home = c:\app\oracle\diag\rdbms\orcl\orcl:
*************************************************************************
Output the results to file:
c:\docume~1\admin\locals~1\temp\alert_1692_3424_orcl_1.ado
```

Чтобы выдача шла не в файл (в нашем случае *alert_1692_3424_orcl_1.ado*), а на экран, следует дополнительно указать ключ –term. Другие примеры:

```
adrci> SHOW ALERT -p "originating_timestamp >= systimestamp - 1 / 24"

adrci> SHOW ALERT -p "message_text like '%ORA-600%' and
originating_timestamp >= systimestamp - 30"

adrci> SHOW ALERT -p "message_text like '%DROP%'" -term
```

Последняя команда *adrci* сообщит о выполнявшихся командах SQL DROP. Такие команды будут отмечаться в *ALERT.LOG*/*log.xml*, если до этого был установлен соответствующим образом параметр СУБД ENABLE_DDL_LOGGING:

```
SQL> ALTER SYSTEM SET ENABLE_DDL_LOGGING = TRUE;

SQL> ALTER SESSION SET ENABLE_DDL_LOGGING = TRUE;
```

Просмотр файлов трассировки процессов СУБД:

```
adrci> SHOW TRACEFILE -rt
   27-FEB-13 17:00:01  diag\rdbms\orcl\orcl\trace\orcl_ckpt_420.trc
   27-FEB-13 16:58:36  diag\rdbms\orcl\orcl\trace\orcl_lgwr_416.trc
   27-FEB-13 15:01:09  diag\rdbms\orcl\orcl\trace\orcl_vktm_324.trc
...
adrci> SHOW TRACE c:\app\oracle\diag\rdbms\orcl\orcl\trace\orcl_ckpt_420.trc
Output the results to file:
c:\docume~1\admin\locals~1\temp\utsout_1692_3424_18.ado
adrci>
```

Данные о проблемах и инцидентах:

```
adrci> SHOW PROBLEM

adrci> SHOW INCIDENT

adrci> SHOW INCIDENT -mode detail -p "incident_id=12345"
```

Допускается выдача нескольких команд одной строкой, например:

```
adrci> SHOW PROBLEM; SHOW ALERT -p "message_text like '%ORA-%'"
```

Команда HOST выполняет вызов командной оболочки ОС (временный «выход в систему»).

16.2. Чистка данных

Часть файлов диагностики подвергается инфраструктурой ADR автоматической чистке. В этом отношении файлы делятся на две категории: «долгие» (LONGP_POLICY) и «короткие» (SHORTP_POLICY). Срок хранения первых (данные об инцидентах, *ALERT.LOG*) по умолчанию составляет 365 дней; для вторых (файлы трассирования процессов, аварийная распечатка памяти) — 30 дней. И то, и другое можно уточнить и переустановить:

```
adrci> SHOW CONTROL

adrci> SET CONTROL (LONGP_POLICY = 1440)

adrci> SET CONTROL (SHORTP_POLICY = 336)
```

Время задается в часах. Помимо этого, чистку можно устроить вручную:

```
adrci> PURGE -age 1440 -type ALERT

adrci> PURGE -age 2880 -type TRACE

adrci> PURGE -i 12345
```

Время задается в минутах. Последняя команда вычистит данные об инциденте с указанным номером.

16.3. Пакеты инцидентов для отсылки в службу поддержки Oracle

Пакет создается как логическое объединение сведений об одном или нескольких инцидентов. В него объединяются диагностические данные из ADR: копии образов участков памяти, отчеты монитора здоровья и прочее. В нужный момент АБД выдает отдельную команду порождения файла **.zip* с физическим содержанием пакета. Файл служит для отправки в службу техподдержки и компонуется системой так, чтобы содержать всю имеющуюся по инцидентам и проблемам диагностическую информацию.

Для управления пакетами служит ряд команд IPS (Incident Packaging Service, служба пакетирования инцидентов).

Примеры.

```
adrci> IPS CREATE PACKAGE
Created package 1 without any contents, correlation level typical
 adrci> IPS CREATE PACKAGE PROBLEM 2 CORRELATE ALL
Created package 2 based on problem id 1, correlation level all
```

Просмотр:

```
adrci> IPS SHOW PACKAGE
```

Ручное добавление файла в пакет с номером 1:

```
adrci> ips add file
c:\app\oracle\diag\rdbms\orcl\orcl\trace\orcl_ckpt_420.trc  package 1
```

Пример физического создания пакета с номером 1:

```
adrci> IPS GENERATE PACKAGE 1 IN \temp
Generated package 1 in file \temp\IPSPKG_20130227173205_COM_1.zip, mode
complete
adrci> HOST "unzip -l /temp/IPSPKG_20130227173205_COM_1.zip"
Archive:  /temp/IPSPKG_20130227173205_COM_1.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
     2818  27.02.2013 18:00
diag/rdbms/orcl/orcl/incpkg/pkg_1/seq_1/export/IPS_CONFIGURATION.dmp
      460  27.02.2013 18:00   diag/rdbms/orcl/orcl/incpkg/pkg_1/seq_1/export/IPS_PACKAGE.dmp
...
```

Удаление:

```
adrci> IPS DELETE PACKAGE 1
Deleted package 1
adrci> IPS DELETE PACKAGE 2
Deleted package 2
```

16.4. Другие возможности

Программа *adrci* позволяет запускать на исполнение командные файлы.

Подготовим файл *\temp\adrscript.adr*:

```
SPOOL /temp/alert_log_errors.log
ECHO "ALERT LOG ERRORS:"; SET HOMEPATH diag/rdbms/orcl/orcl; SHOW PROBLEMS
SHOW ALERT -term -p "MESSAGE_TEXT like '%ORA-%'"
SPOOL OFF
```

Теперь с равным успехом можно выдать в *adrci* и в ОС:

```
adrci> @\temp\adrscript.adr
```

```
>adrci script=/temp/adrscript.adr
```

Программа *adrci* позволяет следить за исполнением заданий монитора здоровья:

```
adrci> SHOW HM_RUN

ADR Home = c:\app\oracle\diag\rdbms\orcl\orcl:
*************************************************************

**********************************************************
HM RUN RECORD 1
**********************************************************
   RUN_ID                        41
   RUN_NAME                      DICTIONARY_CHECK_test
   CHECK_NAME                    Dictionary Integrity Check
...
adrci> SHOW REPORT HM_RUN DICTIONARY_CHECK_test
<?xml version="1.0" encoding="US-ASCII"?>
<HM-REPORT REPORT_ID="DICTIONARY_CHECK_test">
    <TITLE>HM Report: DICTIONARY_CHECK_test</TITLE>
    <RUN_INFO>
        <CHECK_NAME>Dictionary Integrity Check</CHECK_NAME>
        <RUN_ID>41</RUN_ID>
        <RUN_NAME>DICTIONARY_CHECK_test</RUN_NAME>
...
```

Прочие подробности использования программы содержит документация по Oracle.

Страница для заметок