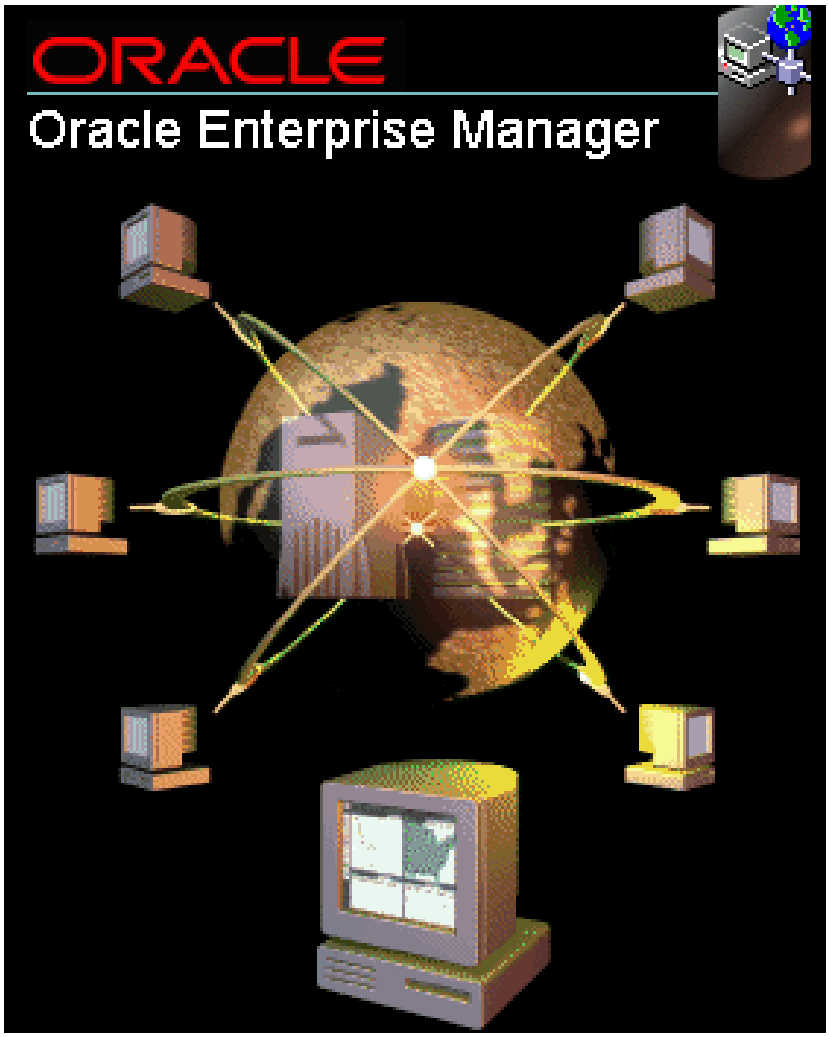
Kiến trúc và quản trị

cơ sở dữ liệu ORACLE

ORACLE 9i INTERNET



HÀ NỘI – 07/ 2002

MỤC LỤC

Chương 1. CÁC ĐIỂM MỚI TRONG ORACLE 9i

Phiên bản Oracle9*i* Release 1 (9.0.1) được đưa ra thị trường vào đầu năm 2001 và được cải tiến, bổ sung thêm một số chức năng, đặc điểm mới. Các đặc điểm này đã làm cho việc quản lý database trở nên mềm dẻo, linh hoạt và hiệu quả hơn. Dưới đây, ta sẽ xem xét một số đặc điểm mới này:

Cho phép định nghĩa lại cấu trúc của tables đang online

Chức năng này được cung cấp trong gói package `DBMS_REDEFINITION` do Oracle cung cấp, cho phép người dùng có thể định nghĩa lại cấu trúc của một table thông quan câu lệnh DML ngay khi nó đang online. Với các phiên bản trước, Oracle 8i, ta cũng có thể định nghĩa lại cấu trúc của table nhưng trước đó cần phải đặt chế độ offline cho nó. Điều này không thuận tiện cho việc quản trị.

Cho phép thực hiện lệnh `ANALYZE VALIDATE STRUCTURE` tức thời

Có thể thực hiện lệnh `ANALYZE` để tối ưu table ngay cả khi đang có lệnh DML thực hiện trên table.

Điều khiển lưu trữ sau

Oracle cung cấp cơ chế điều khiển switching đối với các online redo log group dựa theo thời gian (`time-based`). Trong cấu hình `primary/standby`, tất cả các noncurrent logs tại primary site sẽ được lưu trữ rồi vận chuyển tới standby database. Việc này sẽ hiệu quả khi hạn chế số lượng các redo records.

Tạm treo database

Oracle9*i* cung cấp chức năng `suspend/resume`. Quản trị viên sử dụng lệnh `ALTER SYSTEM SUSPEND` để tạm treo database, dừng mọi thao tác truy xuất vào ra đối với các datafiles và control files. Khi database ở trạng thái tạm treo, các thao tác vào ra (`I/O operations`) đang thực hiện sẽ được kết thúc và những truy cập vào database mới phát sinh sẽ được đẩy vào queue. Thực hiện lệnh `ALTER SYSTEM RESUME` để khôi phục lại tình trạng bình thường của database.

Đặt chế độ hoạt động tĩnh cho database

Oracle9*i* cho phép đưa database vào chế độ hoạt động tĩnh(quiesced state). Theo đó chỉ có các DBA transactions, queries, và các lệnh PL/SQL là được phép thực hiện. Trạng thái này cho phép người dùng thực hiện các thao tác quản trị một cách an toàn. Sự dụng câu lệnh `ALTER SYSTEM QUIESCE RESTRICTED` để đưa database về chế độ hoạt động tĩnh.

Khả năng khôi phục và cấp phát lại không gian

Oracle sẽ tự động thực hiện tạm treo (suspending) và sau đó khôi phục (resuming) lại việc thực hiện các thao tác database tốn kém (large database operations) trong trường hợp có lỗi cấp phát không gian. Nhờ vậy mà Oracle database server sẽ có thể tự thực hiện các thao tác hợp lý thay vì việc trả về thông báo lỗi như ở các phiên bản trước. Sau khi các lỗi này được khắc phục database lại được tự động khôi phục bình thường.

Cho phép lưu trữ trên nhiều đích lưu trữ

Số lượng đích lưu trữ tối đa mà ta có thể sử dụng để lưu trữ các online redo log được tăng lên từ 5 tới 10.

Tự động quản lý vùng không gian

Oracle9*i* cho phép quản lý tự động việc giải phóng và sử dụng các vùng không gian có trong các segments được lưu trữ trong các locally managed tablespaces thông qua việc sử dụng mệnh đề `SEGMENT SPACE MANAGEMENT` có trong câu lệnh `CREATE TABLESPACE`. Quản trị viên có thể sử dụng chế độ `AUTO` hoặc `MANUAL` để chỉ rõ kiểu quản lý không gian mà Oracle sẽ sử dụng.

Cập nhật lại các global indexes mỗi khi thực hiện thao tác bảo trì partition

Theo mặc định, có thể có một vài phần của một bảng được phân khu (partitioned tables) ở trạng thái không sử dụng (đánh dấu `UNUSABLE`) sẽ được nạp vào trong global indexes. Và ta cần xây dựng lại (rebuild) toàn bộ global index. Oracle9*i* cho phép thực hiện tự động công việc rebuild này thông qua mệnh đề `UPDATE GLOBAL INDEX` có trong câu lệnh `ALTER TABLE` khi thực hiện bảo trì.

Cho phép sử dụng đồng thời nhiều kích cỡ block

Oracle cho phép sử dụng đồng thời nhiều kích cỡ blocks (multiple block sizes). Kích thước chuẩn (standard block size) được quy định trong tham số khởi tạo `DB_BLOCK_SIZE` nhưng cũng có thể mở rộng thêm 4 giá trị kích thước block phi chuẩn nữa (nonstandard block sizes). Các kích thước blocks phi chuẩn được chỉ rõ mỗi khi tạo tablespaces. Kích thước block chuẩn được sử dụng cho `SYSTEM` tablespace và hầu hết các tablespaces khác. Việc hỗ trợ sử dụng nhiều kích cỡ block sẽ cho phép thực hiện trao đổi các tablespaces của các database mà không có cùng một kích thước block.

Quản lý động buffer cache

Kích thước của buffer cache có trong vùng nhớ System Global Area được quản lý động. Điều này có nghĩa là giá trị của tham số `DB_BLOCK_BUFFERS` (trong file tham số khởi tạo) có thể

được thay thế bởi giá trị có trong tham số khác, tham số DB_CACHE_SIZE. Trong Oracle 9i, buffer cache lại được phân chia thành nhiều bộ đệm con (subcaches) nếu có sử dụng chế độ multiple block sizes. Bốn giá trị kích cỡ block được chỉ ra trong 4 tham số DB_ nK_CACHE_SIZE tương ứng .

Quản lý động vùng nhớ SGA

Các tham số khởi tạo có thể tác động tới kích cỡ của vùng nhớ SGA. Và ta có thể thay đổi kích cỡ của SGA dễ dàng thông qua câu lệnh ALTER SYSTEM SET.

Quản lý việc khôi phục (undo) tự động

Oracle sử dụng rollback segments để lưu trữ các thông tin cho khôi phục. Việc phục hồi (undo) bao gồm roll back, undo, và thay đổi (changes) đối với database mỗi khi cần. Oracle 9i cho phép ta tạo riêng một undo tablespace để lưu trữ các thông tin phục hồi này. Việc sử dụng undo tablespace sẽ làm giảm bớt tính phức tạp của việc quản trị vùng không gian rollback segment, và cho phép phục hồi lại các thông tin dài mà không sợ bị trùng lên nhau.

Quản lý files trong Oracle

Một điểm mới trong Oracle 9i là quản lý files. Thông qua các tham số khởi tạo DB_CREATE_FILE_DEST và DB_CREATE_ONLINE_LOG_DEST_ n ta có thể chỉ ra cho hệ thống các đường dẫn cụ thể lưu trữ các file thuộc tablespace, online redo log file hay control file. Oracle luôn đảm bảo quản lý file duy nhất trong hệ thống.

Tự động xoá các datafiles

Oracle9*i* cung cấp một lựa chọn cho phép tự động xoá bỏ (remove) các datafiles mỗi khi tablespace tương ứng bị huỷ thông qua câu lệnh DROP TABLESPACE. Tuỳ chọn tương tự trong câu lệnh ALTER DATABASE TEMPFILE cũng được sử dụng để xoá các temporary file tương ứng.

Metadata API

Một PL/SQL package mới, DBMS_METADATA.GET_DDL, được đưa vào Oracle 9i cho phép ta lấy được các siêu dữ liệu (metadata) – Các thông tin tổng hợp về các schema object.

Các bảng ngoài - External tables

Oracle9*i* cho phép ta truy cập theo kiểu chỉ đọc các dữ liệu trong các bảng ngoài (external tables). External tables là các tables mà không nằm trong database, và có thể ở các khuôn dạng (format) nào đó. Câu lệnh CREATE TABLE ... ORGANIZATION EXTERNAL được sử dụng để chỉ ra metadata mô tả cho external table tương ứng. Oracle cung cấp điều khiển truy

cập ORACLE_LOADER, qua đó cung cấp khả năng ánh xạ dữ liệu tương ứng với cú pháp lệnh trong control file.

Tăng cường cho constraint

Ta sử dụng mệnh đề USING INDEX trong câu lệnh CREATE TABLE hay ALTER TABLE để cho phép ta chỉ rõ index mỗi khi sử dụng ràng buộc unique key hay primary key. Thêm vào đó, ta cũng có thể ngăn cản việc huỷ (dropping).

File tham số trên server

Oracle lưu trữ các tham số khởi tạo cho session trong file tham số dưới khuôn dạng văn bản và được đặt tại các client machine.

Các tham số khởi tạo của server nằm trong file tham số trên server thường ở khuôn dạng nhị phân và có thể được lưu trong database.

Temporary tablespace mặc định

Có thể thêm vào mệnh mới DEFAULT TEMPORARY TABLESPACE vào câu lệnh CREATE DATABASE để cho phép tạo temporary tablespace ngay trong thời gian tạo database. Tablespace này sẽ được sử dụng như temporary tablespace mặc định.

Đặt tên cho transaction

Oracle cho phép ta gán tên cho mỗi một transaction. Tên của transaction rất có ích cho việc phân biệt giảm thiểu việc nhầm lẫn giữa các transactions.

Một số thay đổi trong Oracle Database Configuration Assistant

Oracle Database Configuration Assistant có một số thay đổi trong thiết kế. Theo đó, nó cung cấp các mẫu (templates) giúp cho việc tiết kiệm, giảm bớt việc định nghĩa các object trong database.

Người dùng cũng có thể tạo ra các mẫu này thông qua việc sửa đổi các mẫu có sẵn. Khi tạo database bằng công cụ Database Configuration Assistant ta cũng có thể thêm vào ngay hoặc sau đó các khuôn mẫu gọi là các Oracle's new Sample Schemas. Những schemas này là những ví dụ tài liệu cơ bản trong Oracle.

Quản lý việc sử dụng index

Ta thêm mệnh đề MONITORING USAGE vào trong câu lệnh ALTER INDEX để có thể xác định và quản lý index khi nó được thực hiện.

Liệt kê các phân vùng

Oracle 9i giới thiệu sử dụng liệt kê các phân vùng, nó cho phép ta chỉ ra một danh sách các giá trị rời rạc tương ứng với các partitioning column của mỗi phân vùng. Phương thức liệt kê phân vùng (list partitioning method) được đưa ra nhằm mục đích mô hình hoá dữ liệu phân tán đối với các giá trị rời rạc. Việc này khó thực hiện được bằng các phương pháp range partitioning (phân khu theo khoảng giá trị) hay hash partitioning (phân khu theo hàm băm).

Phân khu theo hàm băm cho các index-organized tables

Oracle 9i cho phép sử dụng phương pháp băm khi phân khu các index-organized tables. Ở các phiên bản trước, việc phân khu cho index-organized tables vẫn thực hiện được nhưng chỉ bằng phương pháp range method.

Xử lý các job queue process linh hoạt

Các job queue process được tạo một cách linh hoạt và nó chỉ cần tới số hiệu của processes được tạo để thực hiện các jobs của process đó đang sẵn sàng cho việc thực hiện. Tiến trình nền (background process) có tên là `CJQ` sẽ đảm nhiêm công việc này.

Điểm mới trong Database Resource Manager

Có một số chức năng mới được thêm vào Database Resource Manager:

- Có khả năng tạo một active session pool, là nơi lưu chứa được một số lượng lớn nhất các user sessions đồng thời đang được thực hiện. Nếu có nhiều hơn số lượng lớn nhất các sessions cùng được thực hiện thì các sessions mới này sẽ được đưa vào hàng đợi để chờ thực hiện sau. Tuy nhiên ta cũng có thể đưa ra một khoảng thời gian trễ (timeout) để cho phép thực hiện hay huỷ việc thực hiện các sessions mới bổ sung này.
- Tự động chuyển users từ một nhóm này sang một nhóm khác tuỳ theo sự điều chỉnh của quản trị viên (administrator). Nếu một session được tạo bởi member thuộc một nhóm users nào đó thực hiện trong khoảng thời gian dài hơn thời gian cho phép thì session đó có thể được tự động chuyển sang một nhóm khác với những yêu cầu tài nguyên khác.
- Có khả năng ngăn chặn thực hiện các thao tác mà được dự kiến là sẽ phải chạy trong một thời gian dài hơn là khoảng thời gian cho phép.
- Có khả năng tạo một undo pool, là nơi chứa một số lượng nhất định vùng không gian dành cho việc khôi phục thông tin (undo).

Cơ chế xác thực và nhờ xác thực (Proxy authentication and authorization)

Oracle9*i* cho phép một server nằm ở lớp giữa (middle-tier) xác nhận hộ một client. Ta có thể thực hiện việc này bằng cách đưa vào mệnh đề `GRANT CONNECT THROUGH` trong câu lệnh `ALTER USER`. Ta cũng có thể chỉ rõ vai trò của lớp giữa (middle tier) trong việc kết nối tới client.

Application roles

Oracle cho phép gán roles cho các application users mà được kích hoạt bằng cách sử dụng PL/SQL package. Sử dụng mệnh đề `IDENTIFIED USING` package trong câu lệnh `CREATE ROLE` để thực hiện việc này.

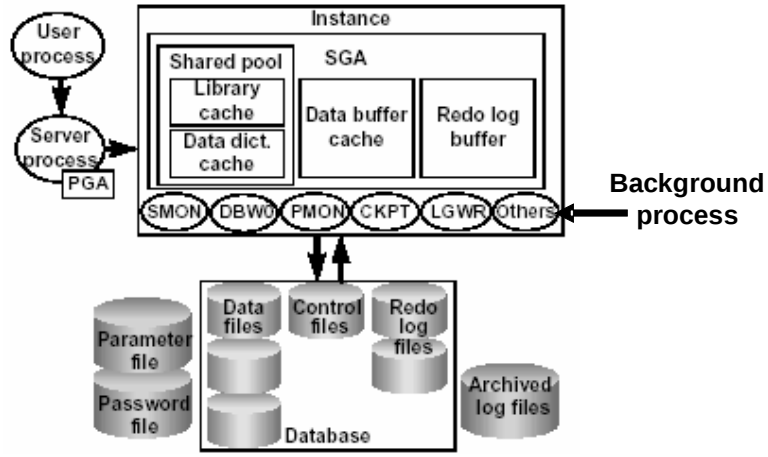
Chương 2. CÁC THÀNH PHẦN KIẾN TRÚC

2.1. KIẾN TRÚC ORACLE SERVER

Oracle server là một hệ thống quản trị cơ sở dữ liệu đối tượng-quan hệ cho phép quản lý thông tin một cách toàn diện. *Oracle server bao gồm hai thành phần chính là Oracle instance và Oracle database*.

2.1.1. Oracle Instance

Oracle instance bao gồm một cấu trúc bộ nhớ ***System Global Area (SGA)*** và các ***background processes (tiến trình nền)*** được sử dụng để quản trị cơ sở dữ liệu. Oracle instance được xác định qua tham số môi trường `ORACLE_SID` của hệ điều hành.



Hình vẽ 1. Kiến trúc Oracle Server

System Global Area - SGA

SGA là vùng bộ nhớ chia sẻ được sử dụng để lưu trữ dữ liệu và các thông tin điều khiển của Oracle server. SGA được cấp phát (allocated) trong bộ nhớ của máy tính mà Oracle server đang hoạt động trên đó. Các User kết nối tới Oracle sẽ chia sẻ các dữ liệu có trong SGA, việc mở rộng không gian bộ nhớ cho SGA sẽ làm nâng cao hiệu suất của hệ thống, lưu trữ được nhiều dữ liệu trong hệ thống hơn đồng thời giảm thiểu các thao tác truy xuất đĩa (disk I/O).

SGA bao gồm một vài cấu trúc bộ nhớ chính:

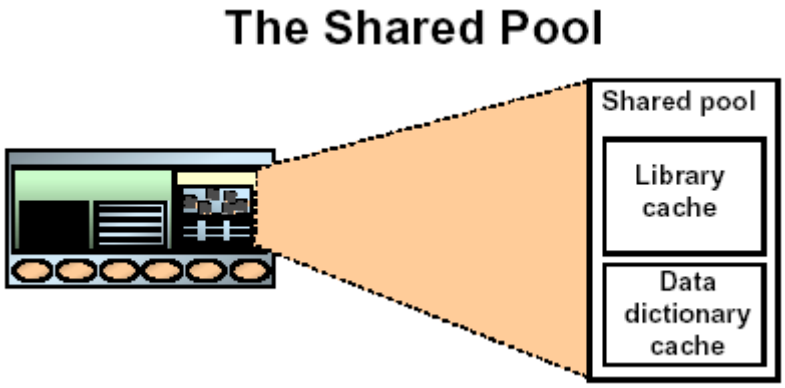
- Shared pool: Là một phần của SGA lưu các cấu trúc bộ nhớ chia sẻ.
- Database buffer cache: Lưu trữ các dữ liệu được sử dụng gần nhất.
- Redo log buffer: Được sử dụng cho việc dò tìm lại các thay đổi trong cơ sở dữ liệu và được thực hiện bởi các background process.

Để chi tiết hơn, ta sẽ xem xét cụ thể từng thành phần.

Share Pool

Shared pool là một phần trong SGA và được sử dụng khi thực hiện phân tích câu lệnh (parse phase). Kích thước của Shared pool được xác định bởi tham số `SHARED_POOL_SIZE` có trong parameter file (file tham số).

Các thành phần của Shared pool gồm có: Library cache và Data dictionary cache.



Hình vẽ 2. Cấu trúc Share Pool

Library Cache

Library cache lưu trữ thông tin về các câu lệnh SQL được sử dụng gần nhất bao gồm:

- Nội dung của câu lệnh dạng text (văn bản).
- Parse tree (cây phân tích) được xây dựng tuỳ thuộc vào câu lệnh.
- Execution plan (sơ đồ thực hiện lệnh) gồm các bước thực hiện và tối ưu lệnh.

Do các thông tin trên đã được lưu trữ trong Library cache nên khi thực hiện lại một câu lệnh truy vấn, trước khi thực hiện câu lệnh, Server process sẽ lấy lại các thông tin đã được phân tích mà không phải phân tích lại câu lệnh. Do vậy, Library cache có thể giúp nâng cao hiệu suất thực hiện lệnh.

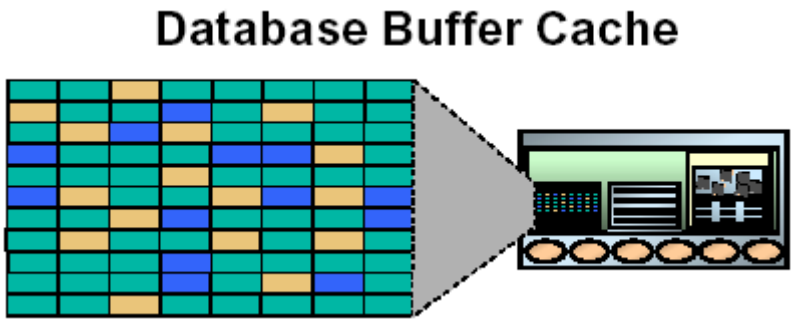
Data Dictionary Cache

Data dictionary cache là một thành phần của Shared pool lưu trữ thông tin của dictionary cache được sử dụng gần nhất như các định nghĩa các bảng, định nghĩa các cột, usernames, passwords, và các privileges (quyền).

Trong giai đoạn phân tích lệnh (parse phase), Server process sẽ tìm các thông tin trong dictionary cache để xác định các đối tượng trong câu lệnh SQL và để xác định các mức quyền tương ứng. Trong trường hợp cần thiết, Server process có thể khởi tạo và nạp các thông tin từ các file dữ liệu.

Data buffer cache

Khi thực hiện một truy vấn, Server process sẽ tìm các blocks cần thiết trong database buffer cache. Nếu không tìm thấy block trong database buffer cache, Server process mới đọc các block từ data file và tạo luôn một bản sao của block đó vào trong vùng nhớ đệm (buffer cache). Như vậy, với các lần truy xuất tới block đó sau này sẽ không cần thiết phải truy xuất vào datafile nữa.



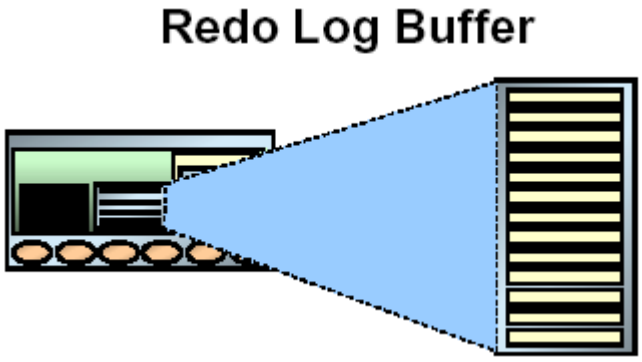
Hình vẽ 3. Database buffer cache

Database buffer cache là vùng nhớ trong SGA sử dụng để lưu trữ các block dữ liệu được sử dụng gần nhất. Tương tự như kích thước của blocks dữ liệu được xác định bởi tham số `DB_BLOCK_SIZE`, kích thước của vùng đệm trong buffer cache cũng được xác định bởi tham số `DB_BLOCK_BUFFERS`.

Oracle server sử dụng giải thuật least recently used (LRU) algorithm để làm tươi lại vùng nhớ. Theo đó, khi nạp mới một block vào bộ đệm, trong trường hợp bộ đệm đã đầy, Oracle server sẽ loại bớt block ít được sử dụng nhất ra khỏi bộ đệm để nạp block mới vào bộ đệm.

Redo log buffer

Server process ghi lại các thay đổi của một instance vào redo log buffer, đây cũng là một phần bộ nhớ SGA.



Hình vẽ 4. Redo log buffer

Có một số đặc điểm cần quan tâm của Redo log buffer:

- Kích thước được xác định bởi tham số `LOG_BUFFER`.
- Lưu trữ các redo records (bản ghi hồi phục) mỗi khi có thay đổi dữ liệu.
- Redo log buffer được sử dụng một cách thường xuyên và các thay đổi bởi một transaction có thể nằm đan xen với các thay đổi của các transactions khác.
- Bộ đệm được tổ chức theo kiểu circular buffer (bộ đệm nối vòng) tức là dữ liệu thay đổi sẽ tiếp tục được nạp lên đầu sau khi vùng đệm đã được sử dụng hết.

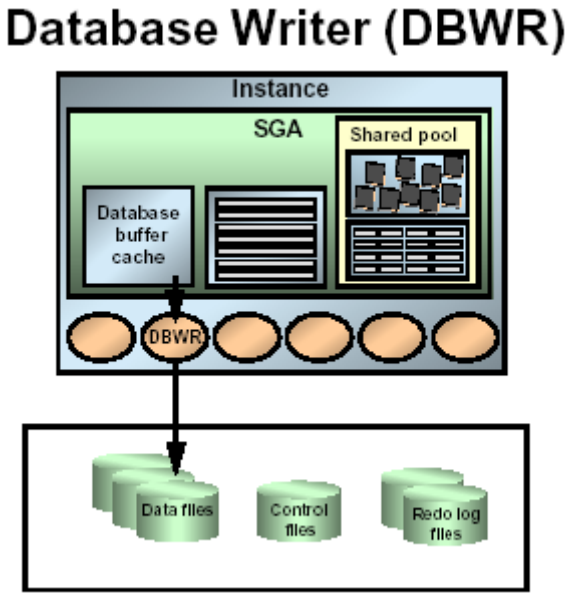
Background process

Background process (các tiến trình nền) thực hiện các chức năng thay cho lời gọi tiến trình xử lý tương ứng. Nó điều khiển vào ra, cung cấp các cơ chế xử lý song song nâng cao hiệu quả và độ tin cậy. Tùy theo từng cấu hình mà Oracle instance có các Background process như:

- Database Writer (DBW0): Ghi lại các thay đổi trong data buffer cache ra các file dữ liệu.
- Log Writer (LGWR): Ghi lại các thay đổi được đăng ký trong redo log buffer vào các redo log files.
- System Monitor (SMON): Kiểm tra sự nhất quán trong database.
- Process Monitor (PMON): Dọn dẹp lại tài nguyên khi các tiến trình của Oracle gặp lỗi.
- Checkpoint Process (CKPT): Cập nhật lại trạng thái của thông tin trong file điều khiển và file dữ liệu mỗi khi có thay đổi trong buffer cache.

Database Writer (DBW0)

Server process ghi lại các dữ liệu thay đổi để rollback và dữ liệu của các block trong buffer cache. Database writer (DBWR) ghi các thông tin được đánh dấu thay đổi từ database buffer cache lên các data files nhằm đảm bảo luôn có khoảng trống bộ đệm cần thiết cho việc sử dụng.



Hình vẽ 5. Database Writer (DBWR)

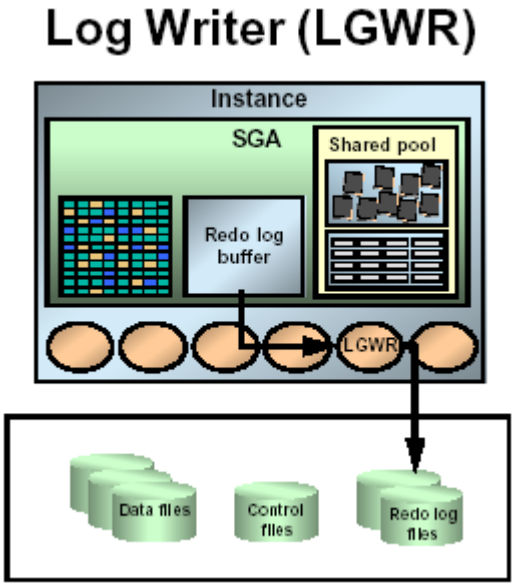
Với việc sử dụng này, hiệu suất sử dụng database sẽ được cải thiện do Server processes chỉ tạo các thay đổi trên buffer cache, DBWR ghi dữ liệu vào các data file cho tới khi:

- Số lượng buffers đánh bị dấu đạt tới giá trị ngưỡng.
- Tiến trình duyệt tất cả buffer mà vẫn không tìm thấy dữ liệu tương ứng.

- Quá thời gian quy định.

Log Writer

Log Writer (LGWR) là một trong các background process có trách nhiệm quản lý redo log buffer để ghi lại các thông tin trong Redo log buffer vào Redo log file. Redo log buffer là bộ đệm dữ liệu được tổ chức theo kiểu nối vòng.



Hình vẽ 6. Log Writer (LGWT)

LGWR ghi lại dữ liệu một cách tuần tự vào redo log file theo các tình huống sau:

- Khi redo log buffer đầy
- Khi xảy ra timeout (thông thường là 3 giây)
- Trước khi DBWR ghi lại các blocks bị thay đổi trong data buffer cache vào các data files.
- Khi commit một transaction.

System Monitor (SMON)

Tiến trình*system monitor (*SMON*)* thực hiện phục hồi các sự cố (crash recovery) ngay tại thời điểm instance được khởi động (startup), nếu cần thiết. SMON cũng có trách nhiệm dọn dẹp các temporary segments không còn được sử dụng nữa trong dictionary-managed tablespaces. SMON khôi phục lại các transactions bị chết mỗi khi xảy ra sự cố. SMON đều đặn thực hiện kiểm tra và khắc phục các sự cố khi cần.

Trong môi trường Oracle Parallel Server, SMON process của một instance có thể thực hiện khôi phục instance trong trường hợp instance hay CPU của máy tính đó gặp sự cố.

Process Monitor (PMON)

Tiến trình *process monitor (*PMON*)* thực hiện tiến trình phục hồi mỗi khi có một user process gặp lỗi. PMON có trách nhiệm dọn dẹp database buffer cache và giải phóng tài nguyên mà

user process đó sử dụng. Ví dụ, nó thiết lập lại (reset) trạng thái của các bảng đang thực hiện trong transaction, giải phóng các locks trên bảng này, và huỷ bỏ process ID của nó ra khỏi danh sách các active processes.

PMON kiểm tra trạng thái của nơi gửi (dispatcher) và các server processes, khởi động lại (restarts) mỗi khi xảy ra sự cố. PMON cũng còn thực hiện việc đăng ký các thông tin về instance và dispatcher processes với network listener.

Tương tự như SMON, PMON được gọi đến mỗi khi xảy ra sự cố trong hệ thống.

Checkpoint Process (CKPT)

Cập nhật lại trạng thái của thông tin trong file điều khiển và file dữ liệu mỗi khi có thay đổi trong buffer cache. Xảy ra checkpoints khi:

- Tất cả các dữ liệu trong database buffers đã bị thay đổi tính cho đến thời điểm checkpointed sẽ được background process $DBWR_n$ ghi lên data files.
- Background process CKPT cập nhật phần headers của các data files và các control files.

Checkpoints có thể xảy ra đối với tất cả các data files trong database hoặc cũng có thể xảy ra với một data files cụ thể.

Checkpoint xảy ra theo các tình huống sau:

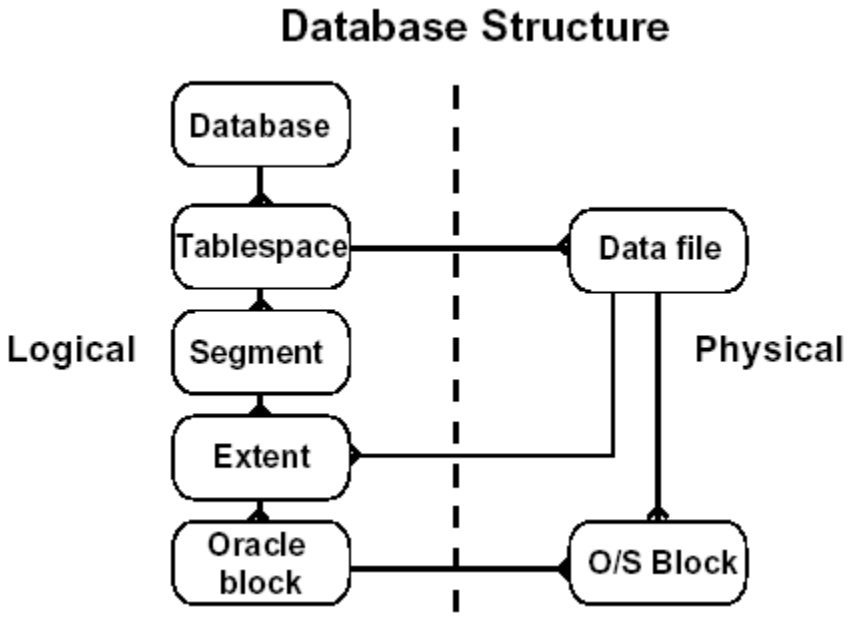
- Mỗi khi có log switch
- Khi một shut down một database với các chế độ trừ chế độ abort
- Xảy ra theo như thời gian quy định trong các tham số khởi tạo LOG_CHECKPOINT_INTERVAL và LOG_CHECKPOINT_TIMEOUT
- Khi có yêu cầu trực tiếp của quản trị viên

Thông tin về checkpoint được lưu trữ trong Alert file trong trường hợp các tham số khởi tạo LOG_CHECKPOINTS_TO_ALERT được đặt là TRUE. Và ngược lại với giá trị FALSE.

2.1.2. Oracle database

Oracle *database* là tập hợp các dữ liệu được xem như một đơn vị thành phần (Unit). Database có nhiệm vụ lưu trữ và trả về các thông tin liên quan. Database được xem xét dưới hai góc độ *cấu trúc logic* và *cấu trúc vật lý* . Tuy vậy, hai cấu trúc dữ liệu này vẫn tồn tại tách biệt nhau, việc quản lý dữ liệu theo cấu trúc lưu trữ vật lý không gây ảnh hưởng tới cấu trúc logic

Oracle database được xác định bởi tên một tên duy nhất và được quy định trong tham số DB_NAME của parameter file.



Hình vẽ 7. Cấu trúc database

Cấu trúc vật lý database

Cấu trúc vật lý bao gồm tập hợp các control file, online redo log file và các datafile:

Datafiles

Mỗi một Oracle database đều có thể có một hay nhiều *datafiles*. Các database datafiles chứa toàn bộ dữ liệu trong database. Các dữ liệu thuộc cấu trúc logic của database như tables hay indexes đều được lưu trữ dưới dạng vật lý trong các datafiles của database.

Một số tính chất của datafiles:

- Mỗi datafile chỉ có thể được sử dụng trong một database.
- Bên cạnh đó, datafiles cũng còn có một số tính chất cho phép tự động mở rộng kích thước mỗi khi database hết chỗ lưu trữ dữ liệu.
- Một hay nhiều datafiles tạo nên một đơn vị lưu trữ logic của database gọi là tablespace.
- Một datafile chỉ thuộc về một tablespace.

Dữ liệu trong một datafile có thể đọc ra và lưu vào vùng nhớ bộ đệm của Oracle. Ví dụ: khi một user muốn truy cập dữ liệu trong một table thuộc database. Trong trường hợp thông tin yêu cầu không có trong cache memory hiện thời, nó sẽ được đọc trực tiếp từ các datafiles ra và lưu trữ vào trong bộ nhớ.

Tuy nhiên, việc bổ sung hay thêm mới dữ liệu vào database không nhất thiết phải ghi ngay vào các datafile. Các dữ liệu có thể tạm thời ghi vào bộ nhớ để giảm thiểu việc truy xuất tới bộ nhớ ngoài (ổ đĩa) làm tăng hiệu năng sử dụng hệ thống. Công việc ghi dữ liệu này được thực hiện bởi DBW*n* background process.

Redo Log Files

Mỗi Oracle database đều có một tập hợp từ 02 *redo log files* trở lên. Các redo log files trong database thường được gọi là database's *redo log*. Một redo log được tạo thành từ nhiều redo entries (gọi là các *redo records*).

Chức năng chính của redo log là ghi lại tất cả các thay đổi đối với dữ liệu trong database. Redo log files được sử dụng để bảo vệ database khỏi những hỏng hóc do sự cố. Oracle cho phép sử dụng cùng một lúc nhiều redo log gọi là *multiplexed redo log* để cùng lưu trữ các bản sao của redo log trên các ổ đĩa khác nhau.

Các thông tin trong redo log file chỉ được sử dụng để khôi phục lại database trong trường hợp hệ thống gặp sự cố và không cho phép viết trực tiếp dữ liệu trong database lên các datafiles trong database. Ví dụ: khi có sự cố xảy ra như mất điện bất chợt chẳng hạn, các dữ liệu trong bộ nhớ không thể ghi trực tiếp lên các datafiles và gây ra hiện tượng mất dữ liệu. Tuy nhiên, tất cả các dữ liệu bị mất này đều có thể khôi phục lại ngay khi database được mở trở lại. Việc này có thể thực hiện được thông qua việc sử dụng ngay chính các thông tin mới nhất có trong các redo log files thuộc datafiles. Oracle sẽ khôi phục lại các database cho đến thời điểm trước khi xảy ra sự cố.

Công việc khôi phục dữ liệu từ các redo log được gọi là *rolling forward*.

Control Files

Mỗi Oracle database đều có ít nhất một *control file*. Control file chứa các mục thông tin quy định cấu trúc vật lý của database như:

- Tên của database.
- Tên và nơi lưu trữ các datafiles hay redo log files.
- Time stamp (mốc thời gian) tạo lập database, ...

Mỗi khi nào một instance của Oracle database được mở, control file của nó sẽ được sử dụng để xác định data files và các redo log files đi kèm. Khi các thành phần vật lý cả database bị thay đổi (ví dụ như, tạo mới datafile hay redo log file), Control file sẽ được tự động thay đổi tương ứng bởi Oracle.

Control file cũng được sử dụng đến khi thực hiện khôi phục lại dữ liệu.

Cấu trúc logic databse

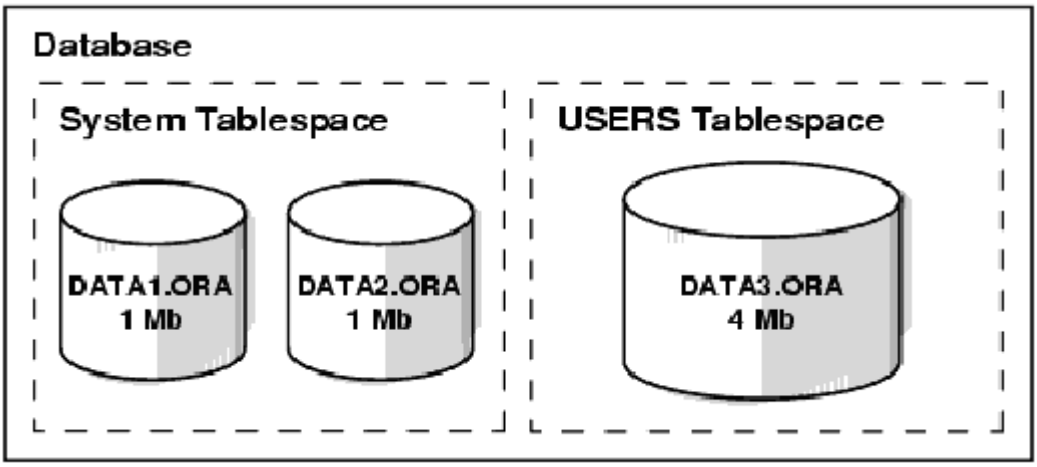
Cấu trúc logic của Oracle database bao gồm các đối tượng tablespaces, schema objects, data blocks, extents, và segments.

Tablespaces

Một database có thể được phân chia về mặt logic thành các đơn vị gọi là các *tablespaces*, Tablespaces thường bao gồm một nhóm các thành phần có quan hệ logic với nhau.

Databases, Tablespaces, và Datafiles

Mối quan hệ giữa các databases, tablespaces, và datafiles có thể được minh hoạ bởi hình vẽ sau:



Hình vẽ 8. Quan hệ giữa database, tablespace và datafile

Có một số điểm ta cần quan tâm:

- Mỗi database có thể phân chia về mặt logic thành một hay nhiều tablespace.
- Mỗi tablespace có thể được tạo nên, về mặt vật lý, bởi một hoặc nhiều datafiles.
- Kích thước của một tablespace bằng tổng kích thước của các datafiles của nó. Ví dụ: trong hình vẽ ở trên SYSTEM tablespace có kích thước là 2 MB còn USERS tablespace có kích thước là 4 MB.
- Kích thước của database cũng có thể xác định được bằng tổng kích thước của các tablespaces của nó. Ví dụ: trong hình vẽ trên thì kích thước của database là 6 MB.

Schema và Schema Objects

S*chema* là tập hợp các đối tượng (objects) có trong database. *Schema objects* là các cấu trúc logic cho phép tham chiếu trực tiếp tới dữ liệu trong database. Schema objects bao gồm các cấu trúc như tables, views, sequences, stored procedures, synonyms, indexes, clusters, và database links.

Data Blocks, Extents, and Segments

Oracle điểu khiển không gian lưu trữ trên đĩa cứng theo các cấu trúc logic bao gồm các data blocks, extents, và segments.

Oracle Data Blocks

Là mức phân cấp logic thấp nhất, các dữ liệu của Oracle database được lưu trữ trong các *data blocks*. Một data block tương ứng với một số lượng nhất định các bytes vật lý của database trong không gian đĩa cứng. Kích thước của một data block được chỉ ra cho mỗi

Oracle database ngay khi database được tạo lập. Database sử dụng, cấp phát và giải phóng vùng không gian lưu trữ thông qua các Oracle data blocks.

Extents

Là mức phân chia cao hơn về mặt logic các vùng không gian trong database. Một *extent* bao gồm một số data blocks liên tiếp nhau, cùng được lưu trữ tại một thiết bị lưu giữ. Extent được sử dụng để lưu trữ các thông tin có cùng kiểu.

Segments

Là mức phân chia cao hơn nữa về mặt logic các vùng không gian trong database. Một *segment* là một tập hợp các extents được cấp phát cho một cấu trúc logic . Segment có thể được phân chia theo nhiều loại khác nhau:

Data segment	Mỗi một non-clustered table có một data segment. Các dữ liệu trong một table được lưu trữ trong các extents thuộc data segment đó. Với một partitioned table thì mỗi each partition lại tương ứng với một data segment. Mỗi Cluster tương ứng với một data segment. Dữ liệu của tất cả các table trong cluster đó đều được lưu trữ trong data segment thuộc Cluster đó.
index segment	Mỗi một index đều có một index segment lưu trữ các dữ liệu của nó. Trong partitioned index thì mỗi partition cũng lại tương ứng với một index segment.
rollback segment	Một hoặc nhiều rollback segments của database được tạo lập bởi người quản trị database để lưu trữ các dữ liệu trung gian phục vụ cho việc khôi phục dữ liệu. Các thông tin trong Rollback segment được sử dụng để: ▪ Tạo sự đồng nhất các thông tin đọc được từ database ▪ Sử dụng trong quá trình khôi phục dữ liệu ▪ Phục hồi lại các giao dịch chưa commit đối với mỗi user
temporary segment	Temporary segments được tự động tạo bởi Oracle mỗi khi một câu lệnh SQL statement cần đến một vùng nhớ trung gian để thực hiện các công việc của mình như sắp xếp dữ liệu. Khi kết thúc câu lệnh đó, các extent thuộc temporary segment sẽ lại được hoàn trả cho hệ thống.

Oracle thực hiện cấp phát vùng không gian lưu trữ một cách linh hoạt mỗi khi các extents cấp phát đã sử dụng hết.

Các cấu trúc vật lý khác

Ngoài ra, Oracle Server còn sử dụng các file khác để lưu trữ thông tin. Các file đó bao gồm:

- Parameter file: Parameter file chỉ ra các tham số được sử dụng trong database. Người quản trị database có thể sửa đổi một vài thông tin có trong file này. Các tham số trong parameter file được viết ở dạng văn bản.
- Password file: Xác định quyền của từng user trong database. Cho phép người sử dụng khởi động và tắt một Oracle instance.
- Archived redo log files: Là bản off line của các redo log files chứa các thông tin cần thiết để phục hồi dữ liệu.

2.1.3. Quản trị cơ sở dữ liệu Oracle

Quản trị cơ sở dữ liệu là công việc bảo trì và vận hành Oracle server để nó có thể tiếp nhận và xử lý được tất cả các yêu cầu (requests) từ phía Client. Để làm được điều này, người quản trị viên cơ sở dữ liệu cần phải hiểu được kiến trúc của Oracle database.

2.1.4. Thiết lập các tham số khởi tạo ảnh hưởng tới kích cỡ bộ nhớ SGA

Tham số khởi tạo ảnh hưởng tới kích thước bộ nhớ cấp phát cho vùng System Global Area. Ngoại trừ tham số SGA_MAX_SIZE, còn lại các tham số khác đều là tham số động tức là có thể thay đổi giá trị của chúng ngay trong lúc database đang chạy thông qua câu lệnh ALTER SYSTEM. Kích thước của SGA cũng có thể thay đổi được trong quá trình chạy database.

Thiết lập tham số cho Buffer Cache

Tham số khởi tạo buffer cache quy định kích thước của buffer cache là một phần của SGA. .

Ta sử dụng các tham số DB_CACHE_SIZE và một trong những tham số DB_*n*K_CACHE_SIZE để cho phép sử dụng chế độ multiple block sizes đối với database. Oracle sẽ tự động gán các giá trị mặc định cho tham số the DB_CACHE_SIZE, còn tham số DB_*n*K_CACHE_SIZE sẽ được gán mặc định bằng 0.

Kích thước của buffer cache sẽ có ảnh hưởng nhiều tới hiệu suất thực hiện của hệ thống. Kích thước càng lớn thì càng giảm bớt việc đọc và ghi đĩa. Tuy nhiên, kích thước của cache lớn sẽ tốn nhiều bộ nhớ và sẽ có nhiều tốn kém trong việc thực hiện paging (phân trang) hay swapping (trao đổi) bộ nhớ.

Tham số DB_CACHE_SIZE

Tham số khởi tạo DB_CACHE_SIZE được sử dụng thay thế cho tham số DB_BLOCK_BUFFERS của các phiên bản Oracle trước kia. Tham số DB_CACHE_SIZE quy định kích thước của block buffers chuẩn. Kích thước của một block chuẩn lại được quy định trong tham số DB_BLOCK_SIZE.

Tuy vậy, tham số DB_BLOCK_BUFFERS vẫn được sử dụng để tương thích với các phiên bản trước, tuy nhiên giá trị của nó không được sử dụng cho các tham số động.

Tham số DB_*n*K_CACHE_SIZE

Chỉ ra kích cỡ là bội số nguyên lần kích thước của block buffers. Nó được chỉ ra bởi các tham số:

- DB_2K_CACHE_SIZE
- DB_4K_CACHE_SIZE
- DB_8K_CACHE_SIZE
- DB_16K_CACHE_SIZE
- DB_32K_CACHE_SIZE.

Mỗi tham số chỉ ra kích cỡ của buffer cache tương ứng với kích cỡ của block.

Ví dụ:

```
DB_BLOCK_SIZE=4096
DB_CACHE_SIZE=12M
DB_2K_CACHE_SIZE=8M
DB_8K_CACHE_SIZE=4M
```

Ở ví dụ này, các tham số chỉ ra kích thước block chuẩn của database là 4K. Kích thước cache tương ứng với kích thước block chuẩn là 12M. Các kích thước mở rộng của cache là 2K và 8K sẽ được đặt lại với giá trị tương ứng là 8M và 4M.

Điều chỉnh kích cỡ của Shared Pool

Tham số SHARED_POOL_SIZE trong phiên bản Oracle 9i là tham số động, tức là có thể thay đổi được giá trị của nó (điều này không thể thực hiện được trong các phiên bản trước). Nó cho phép ta thay đổi kích thước của shared pool là một trong các thành phần của SGA. Theo mặc định Oracle cũng chọn một giá trị mặc định phù hợp cho tham số này.

Điều chỉnh kích cỡ của Large Pool

Tương tự như SHARED_POOL_SIZE, tham số LARGE_POOL_SIZE cũng là một tham số động, nó cho phép ta điều chỉnh kích cỡ của large pool, đây cũng là một thành phần trong SGA. .

Giới hạn kích cỡ của SGA

Tham số SGA_MAX_SIZE quy định kích cỡ lớn nhất của System Global Area . Ta cũng có thể thay đổi kích cỡ của buffer caches, shared pool và large pool, tuy nhiên việc thay đổi này nên là mở rộng giá trị kích thước cho các thành phần của SGA. Giá trị mở rộng thêm này cũng không nên đặt tới ngưỡng của SGA_MAX_SIZE.

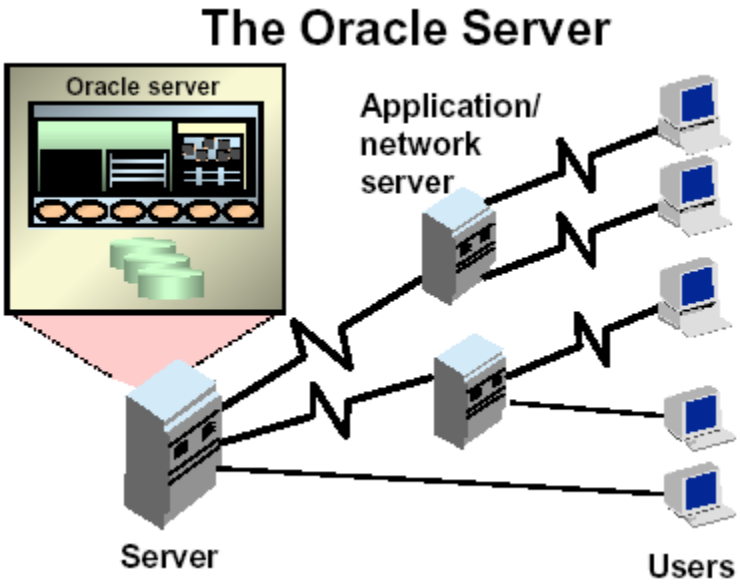
Trong trường hợp ta không chỉ rõ giá trị của SGA_MAX_SIZE thì Oracle sẽ tự động gán giá trị này bằng tổng số kích cỡ của các thành phần của SGA lúc ban đầu.

2.2.KẾT NỐI TỚI ORACLE SERVER

2.2.1. Mô hình kết nối

Các Client có thể kết nối tới Oracle Server thông qua 03 cách sau:

- Kết nối trực tiếp: kết nối mà Client nằm trên cùng một máy chủ Oracle server.
- Kết nối hai lớp (two-tiered) client-server: Client nằm trên một máy tính khác và kết nối trực tiếp tới máy chủ Oracle Server.
- Kết nối ba lớp (three-tiered): Client nằm trên máy tính khác với máy chủ Oracle Server, nó giao tiếp với một ứng dụng hay một máy chủ mạng (network server) và điều khiển ứng dụng hay máy chủ này kết nối tới Oracle server.



Hình vẽ 9. Kết nối tới Oracle server

2.2.2. Một số khái niệm cơ bản liên quan đến kết nối

Connection (liên kết)

Liên kết là đường liên lạc giữa một user process và một Oracle server. Trong trường hợp user sử dụng các tool hoặc các ứng dụng ngay trên cùng một máy với Oracle server, đường liên lạc sẽ được tạo lập ngay trên máy đó. Trong trường hợp user nằm trên một máy khác thì liên kết sẽ sử dụng đường mạng để kết nối tới Oracle server.

Session (phiên)

Một phiên tương ứng với một liên kết cụ thể của một user tới một Oracle server. Phiên bắt đầu khi user kết nối tới Oracle Server đã được kiểm tra hợp lệ và kết thúc khi user thực hiện log out khỏi Oracle Server hoặc user kết thúc một cách bất thường. Một user cùng một lúc có thể có nhiều phiên làm việc để kết nối tới Oracle Server thông qua các ứng dụng hay các tool khác nhau. Ví du: User có thể đồng thời có các phiên làm việc giữa SQL*Plus, Developer/2000 Form,... tới Oracle Server.

Lưu ý: Phiên chỉ tạo lập được khi Oracle Server đã sẵn sàng cho việc kết nối của các client.

2.2.3. Kết nối tới database

Các bước thực hiện kết nối

Để kết nối tới database trước tiên, cần phải tạo liên kết tới Oracle Server. Liên kết tới Oracle Server được tạo theo các bước sau:

- User sử dụng công cụ SQL*Plus hay sử dụng các công cụ khác của Oracle như Developer/2000 Forms để khởi tạo tiến trình. Trong mô hình Client-Server, các công cụ hay ứng dụng này được chạy trên máy Client.
- User thực hiện log in vào Oracle server với việc khai báo username, password và tên liên kết tới database. Các ứng dụng tools sẽ tạo một tiến trình để kết nối tới Oracle server qua các tham số này. Tiến trình này được gọi là tiến trình phục vụ. Tiến trình phục vụ sẽ giao tiếp với Oracle server thay cho tiến trình của user chạy trên máy Client.

Ví dụ thực hiện kết nối tới database

Để hiểu rõ hơn về các bước thực hiện kết nối, ta hãy xem xét một ví dụ mô tả việc kết nối tới Oracle database thực hiện bởi một user tại một máy tính khác có kết nối tới máy tính mà Oracle server đang chạy trên đó. Việc kết nối được thực hiện thông qua đường mạng bằng cách sử dụng dịch vụ Oracle Net8.

1. Tại máy chủ, cần đảm bảo Oracle server đang chạy và sẵn sàng đón nhận các tín hiệu từ phía Client. Máy chủ này được gọi là *host* hay *database server*.
2. Tại một máy trạm có chạy các ứng dụng (gọi là *local machine* hay *client workstation*) sẽ thực hiện các user process để kết nối tới database. Client application thực hiện thiết lập một kết nối tới server thông qua Net8 driver.
3. Máy chủ server trên đó có các Net8 driver. Server sẽ thực hiện việc nghe và dò tìm tất cả các yêu cầu gửi đến từ phía client và sau đó sẽ tạo một server process tương ứng với user process.
4. Khi user thực hiện một câu lệnh SQL hay commit một transaction. Ví dụ như user dữ liệu trên một dòng trong một table.
5. Server process sẽ nhận về câu lệnh gửi tới từ Client, kiểm tra và phân tích câu lệnh, việc này được thực hiện trong shared pool. Tiếp theo đó, Server process sẽ kiểm tra quyền truy nhập dữ liệu của user.
6. Server process trả về các giá trị dữ liệu yêu cầu từ các dữ liệu có trong datafile hay trong system global area.
7. Server process thay đổi các dữ liệu có trong system global area. DBW*n* process ghi lại các blocks đã thay đổi ra ổ đĩa. LGWR process sẽ ghi lại ngay lập tức các bản ghi thay đổi vào online redo log file ngay khi transaction được commit.
8. Trong trường hợp transaction thực hiện thành công, server process sẽ gửi thông báo hoàn tất qua đường mạng tới Client. Ngược lại, sẽ có một error message gửi tới Client.

Chương 3. CÁC CÔNG CỤ QUẢN TRỊ ORACLE

3.1.CÁC CÔNG CỤ QUẢN TRỊ CƠ SỞ DỮ LIỆU ORACLE

Oracle hiện tại đã cung cấp rất nhiều công cụ cho phép quản trị cơ sở dữ liệu. Có thể kể ra sau đây một số công cụ cơ bản sau:

Công cụ	Diễn giải
Server Manager Line Mode	Công cụ quản trị cơ sở dữ liệu Oracle theo chế độ dòng lệnh như start (khởi động), shutdown (dừng) database,...
SQL*Plus Line Mode	Đây là một tiện ích sử dụng cho việc công quản trị như starting up, shutting down, hoặc phục hồi database.
Oracle Enterprise Manager	Là công cụ có giao diện đồ họa để thực hiện việc quản trị, điều khiển và thay đổi một hoặc nhiều database.
SQL*Loader	Tiện ích sử dụng để nạp các file bên ngoài vào trong các bảng của Oracle .
Công cụ Export và Import	Tiện ích sử dụng để exporting hoặc importing dữ liệu theo khuôn dạng của Oracle.
Password File	Tiện ích sử dụng để tạo file mật khẩu trong database.

Để thuận tiện, tài liệu sẽ trình bày hai công cụ thường dùng nhất để quản trị cơ sở dữ liệu là:

- Server Manager Line Mode
- Oracle Enterprise Manager

3.2.SERVER MANAGER LINE MODE

3.2.1. Truy nhập Server Manager Line Mode

User (người sử dụng) có thể vào Server Manager Line Mode theo hai cách:

```
C:\svrmgrl
```

Cách này chỉ vào Server Manager Line Mode mà chưa thực hiện kết nối cụ thể tới database

Lưu ý: trong các phiên bản cũ, ta gõ `svrmgrl30` thay vì `svrmgrl`

Hoặc:

```
C:\svgrmrl command=”connect internal/admin”
C:\svgrmrl command=@c:\example.sql
```

Vào Server Manager Line Mode đồng thời thực hiện lệnh luôn.

Kí tự sử dụng trong Server Manager Line Mode

Với Server Manager Line Mode, ta có thể thực hiện câu lệnh SQL hoặc đoạn lệnh PL/SQL. Các câu lệnh được kết thúc bởi ký tự chấm phẩy ‘ ; ’

Sử dụng ký tự ‘ /’ để kết thúc câu lệnh trong trường hợp đã bấm phím Enter để xuống dòng.

Ngoài ra, ta có thể chạy file script chứa các câu lệnh SQL và PL/SQL. Bằng cách sử dụng ký tự ‘ @ ‘ ở trước tên file script.

3.2.2. Phân nhóm các lệnh trong Server manager

Loại lệnh	Tên lệnh
Lệnh không cần kết nối tới database	EXIT REMARK SET SHOW SPOOL
Các lệnh cần đến mức quyền truy nhập	CONNECT/DISCONNECT DESCRIBE EXECUTE SHOW ERRORS SHOW PARAMETER SHOW SGA
Các lệnh cần đến mức quyền truy nhập đặc biệt	CONNECT… AS SYSDBA CONNECT… AS SYSOPER ARCHIVE LOG RECOVER DATABASE STARTUP/SHUTDOWN

3.2.3. Diễn giải các lệnh trong Server manager

Tên lệnh	Diễn giải
EXIT	Đóng SQL Worksheet, thoát khỏi Server Manager
REMARK	Thêm vào lời chú dẫn, thường hay sử dụng trong file SQL script
SET	Thiết lập hoặc thay đổi các tính chất có trong phiên làm việc hiện thời.
SHOW	Hiển thị các thiết đặt hiện thời
SPOOL	Cho phép hoặc thôi cho phép chuyển hướng kết xuất dữ liệu ra file
CONNECT/ DISCONNECT	Kết nối hoặc huỷ kết nối tới database
DESCRIBE	Xem cấu trúc của một function, package, package body, procedure, table, object, view
EXECUTE	Thực hiện một dòng lệnh PL/SQL

SHOW ERRORS	Hiển thị các lỗi phát sinh của thủ tục, hàm hay package
SHOW PARAMETER	Hiển thị gái trị hiện thời của một hay nhiều tham số đã khởi tạo
SHOW SGA	Hiển thị thông tin về SGA của Instance hiện thời
CONNECT/AS SYSDBA	Kết nối tới database với đặc quyền quản trị
ARCHIVE LOG	Khởi động và dừng việc lưu trữ tự động đối với các file online redo log files, redo log file
RECOVER DATABASE	Phục hồi lại một hay nhiều tablespaces
STARTUP/ SHUTDOWN	Khởi động hoặc tắt Oracle instance

3.3.ORACLE ENTERPRISE MANAGER

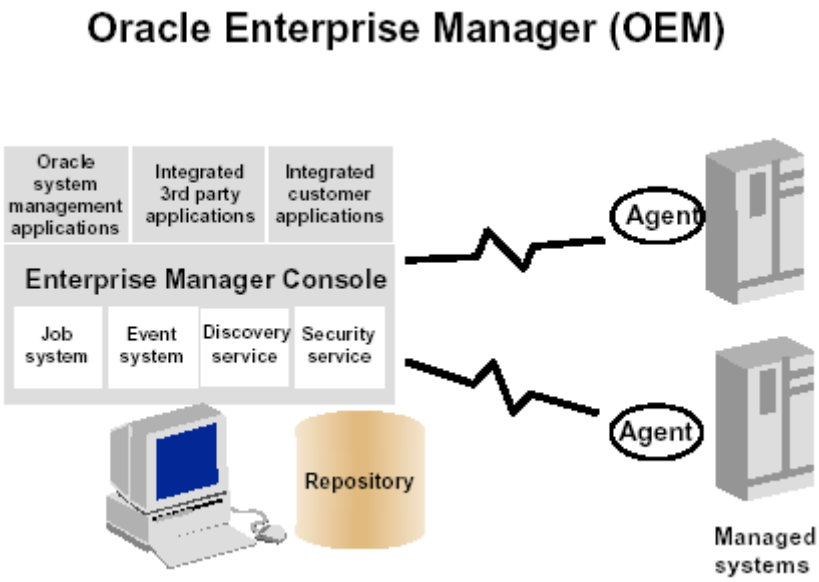
Oracle Enterprise Manager (OME) là phương tiện cho phép có được cái nhìn tổng thể về toàn bộ hệ thống. Trong đó có cây phân cấp và các hình ảnh đồ họa về các đối tượng và quan hệ giữa chúng trong hệ thống.

OME có các tiến trình Intelligent Agent processes cho phép quản lý từ xa các dịch vụ chung - common services như jobs, events,… một cách dễ dàng.

OME cũng bao gồm cả những ứng dụng quản lý chuyên biệt: DBA Management Pack, Advanced Management Packs.

Bên cạnh đó, OME còn cung cấp một lượng lớn các hàm API cho phép tích hợp với các hệ thống quản lý ứng dụng khác. Bao gồm cả các hệ thống quản lý của Oracle và không phải của Oracle.

OEM Console có trong cả Windows NT và Windows 95.



Hình vẽ 10. Oracle Enterprise Manager

3.3.1. Kiến trúc OME

Kiến trúc OME là mở rộng của kiến trúc Client/Server, nó có kiến trúc ba lớp.

- Lớp thứ nhất chứa các Java-based console và các ứng dụng tích hợp cho phép cài đặt và chạy bởi các Web browser.
- Lớp thứ hai là Oracle Management Server - OMS. Chức năng chính của OMS là xử lý và quản trị tất cả các tác vụ của hệ thống, tập trung quản lý và phân phối điều khiển giữa các clients và các nút điều khiển - managed nodes. OEM sử dụng Oracle Enterprise Manager repository để duy trì dữ liệu hệ thống, dữ liệu ứng dụng và các trạng thái của các thực thể điều khiển phân tán trong hệ thống, cho phép người dùng có thể truy cập và chia sẻ các vùng dữ liệu lưu trữ.
- Lớp thứ ba bao gồm các đích như databases, nodes và các dịch vụ quản lý khác.

3.3.2. Các dịch vụ chung

OEM có các dịch vụ cho phép quản lý các nodes trên mạng (network)

- Dịch vụ phát hiện - Discovery service: OEM tự động phát hiện (định vị) tất cả các database và các dịch vụ chạy trên các nodes, một khi các nodes được xác định. Các dịch vụ này bao gồm Web servers, listeners, machines, parallel servers, video servers, và các services khác.
- Job Scheduling System: cho phép thực hiện tự động lặp lại các tác vụ. Hệ thống cho phép tạo và quản lý các jobs, lên kế hoạch thực hiện chúng và cho phép xem, chia sẻ thông tin xác định Jobs.
- Event Management System: cho phép quản lý môi trường mạng (network environment) xử lý các trường hợp mất dịch vụ, thiếu hoặc hết vùng lưu trữ, và các vấn đề khác như sử dụng tài nguyên CPU. Mỗi khi các events được phát hiện, người quản trị có thể thông báo hoặc sửa nó.
- Bảo mật - Security: các tham số bảo mật xác định cho từng dịch vụ (services), đối tượng (objects), và từng user quản trị (administrators).
- Dịch vụ kho lưu trữ chia sẻ (Shared Repository)

OEM là một hệ thống đa người dùng - multiuser system. Mỗi quản trị viên có một account riêng để đăng nhập vào hệ thống. Tùy theo việc thiết đặt quyền hạn, mà quản trị viên có thể truy cập vào các dữ liệu lưu trong kho trung tâm, kho được chia sẻ cho tất cả các quản trị viên của OEM để thực hiện công việc quản lý.

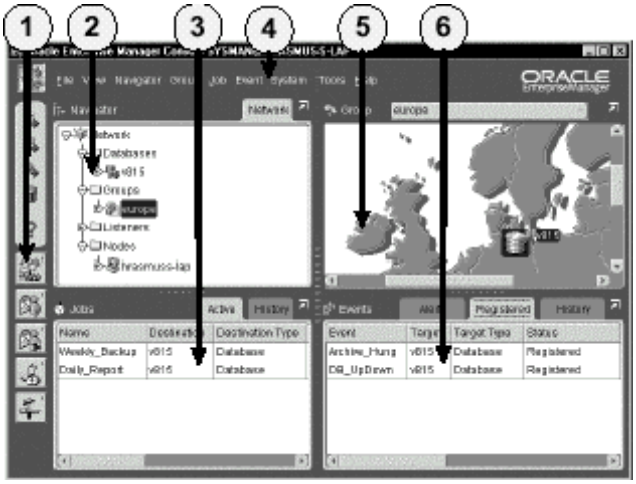
3.3.3. Oracle Configuration Assistant

Configuration Assistant là công cụ cho phép tạo các shared repositories, đặt lại cấu hình cho database và thiết đặt cấu hình cho các local console (đơn vị điều khiển cục bộ). Configuration Assistant được tự động khởi động ngay sau khi hoàn tất quá trình cài đặt của Universal Installer. Ta cũng có thể khởi động ứng dụng này bằng tay (chạy lệnh `%emrepmgr` từ dấu nhắc hệ thống).

3.3.4. Oracle Enterprise Manager Console

Bao gồm cả cây phân cấp và hình ảnh đồ họa biểu diễn các đối tượng trong hệ thống.

1. **Các nút có biểu tượng:** cho phép gọi các ứng dụng khác để cùng thực hiện việc quản trị các tác vụ (task). Việc này cũng có thể thực hiện thông qua mục chọn tương ứng trên menu.
2. **Navigator** hay **object explorer:** được tổ chức dưới dạng cây phân cấp. Nó cho phép xem các Oracle services trong mạng làm việc. Navigator cho phép quản trị viên có thể browse các Oracle services, như databases, listeners, nodes, và name servers, qua đó có thể sửa đổi các tính chất của các đối tượng; ví dụ: người dùng có thể thay đổi nội dung của bảng.
3. **Job system:** cho phép thực hiện các tác vụ từ xa liên quan tới listeners, databases. Job system dựa trên các thủ tục trong Tool Control Language (TCL) engine.



Hình vẽ 11. Oracle Enterprise Manager

4. **Menu** cho phép khởi tạo các ứng dụng quản trị khác và thực hiện nhiều tác vụ khác nhau.
5. **Map** hay **topographical view** cho phép các Oracle services có thể được gộp lại tùy theo quan hệ về không gian, chức năng, hay cả hai. Map view cho phép người sử dụng tập trung vào các đối tượng cần quản lý.
6. **Event system** điều khiển và thông báo các trạng thái của hệ thống.

3.4.CÁC CÔNG CỤ QUẢN TRỊ KHÁC

Ngoài hai công cụ chính như đã kể trên, Oracle còn hỗ trợ bộ các công cụ chuẩn khác như:

- **Instance Manager:** dùng để điều khiển database định nghĩa và khởi tạo các tham số liên quan tới các tính chất của instance.
- **Schema Manager:** dùng để tạo lập và quản lý các đối tượng như tables, indexes, và views.
- **Security Manager:** dùng để quản lý các users và phân quyền cho các users này

- **Storage Manager:** dùng để tổ chức các database files và quản lý các rollback segments.
- **SQL Worksheet:** giao tiếp theo kiểu dòng lênh, nó cho phép thực hiện các câu lệnh SQL và PL/SQL cũng như là các câu lệnh của Server Manager
- **Backup Manager:** dùng để sao lưu, phục hồi và bảo trì databases, quản lý các redo log files.
- **Data Manager:** dùng để nạp và tổ chức lại dữ liệu trong databases.

Ngoài các các công cụ kể trên, Oracle còn hỗ trợ các công cụ làm tăng cường hiệu suất làm việc của DATABASE.

- **Performance Manager:** biểu diễn hiệu suất làm việc của database dưới dạng biểu đồ đồ hoạ.
- **Top-Session Manager:** hiển thị thông tin chi tiết về các session của 10 session có sử dụng tài nguyên hệ thống, sắp xếp theo thứ tự giảm dần. Công cụ này còn cho phép kill session.
- **Loock Manager:** cho biết các thông tin liên quan đến việc khoá (lock) các đối tượng trong database. Các thông tin được biểu diễn dưới dạng đồ hoạ.
- **Tablespaces Manager:** công cụ giúp cho dễ dàng quản lý các tablespace có trong database.

Chương 4. TẠO DATABASE

4.1.CÁC BƯỚC TẠO DATABASE

Oracle hiện đã hỗ trợ một công cụ cho phép tạo database trên hệ điều hành Windows một cách trực quan. Đó là công cụ Oracle Database Assistant. Tuy nhiên, trong một số trường hợp công cụ này tỏ ra không được thuận tiện lắm.

Bên cạnh việc sử dụng công cụ cung cấp sẵn của Oracle để tạo database, Oracle còn cho phép user có thể tạo database mà không sử dụng các công cụ của Oracle. Phương pháp này gọi là tạo database bằng tay – manually.

Việc tạo database được tiến hành theo các bước:

1. Quyết định chọn lựa tên instance và tên database duy nhất. Chọn character set – tập ký tự sử dụng trong database.
2. Thiết lập các biến hệ thống.
3. Chuẩn bị file tham số, tạo file mật khẩu (nên có thao tác này).
4. Chuẩn bị instance phục vụ quản trị
5. Tạo database.
6. Chạy scripts để tạo các dictionary cho database.

4.2.CHUẨN BỊ MÔI TRƯỜNG

4.2.1. Chuẩn bị hệ điều hành

Để tạo database, quản trị viên trước tiên phải có thể truy nhập vào hệ điều hành với đầy đủ quyền.

Trước khi tạo database, cần tính toán dung lượng bộ nhớ cho database căn cứ vào cấu hình của Server và đảm bảo có đủ bộ nhớ để thực hiện các tiến trình của Oracle một cách hiệu quả.

Tính toán lượng đĩa trống cần thiết cho việc lưu trữ các data files, các control files, các redo log file và các files khác...

4.2.2. Lên kế hoạch bố trí các file thông tin

Để bảo vệ an toàn cho database, ta cần có kế hoạch bố trí các file thông tin.

Control files

Để đảm bảo an toàn, một database cần ít nhất 02 control files và được đặt tại hai chỗ khác nhau. Các control files nên được đặt tên khác nhau sao cho dễ dàng có thể phân biệt.

Tên của Control files nên được đặt kèm với tên của database cho dễ nhớ, như sau:

```
CTL<n><database_name>.ORA
```

Với:

```
n               là số thứ tự của control file
database_name   tên của database
```

Trong parameter file, tên của các control files được đặt phân cách nhau bởi các dấu phẩy.

Ví dụ:

```
control_files =  (“C:\ORANT\DATABASE\CTL1KTKB.ORA”,
                  ”C:\ORANT\DATABASE\CTL2KTKB.ORA”)
```

Online redo log files

Online redo log files thông thường bao gồm nhiều nhóm các online redo log files khác nhau. Với mỗi nhóm chứa các bản sao của các redo log file. Tương tự như control file. Các online redo log file cũng nên được đặt ở các nới khác nhau.

Cũng giống như Control files, việc đặt tên cho các Online redo log files nên được đặt kèm với tên của database cho dễ nhớ, như sau:

```
LOG<n><database_name>.ORA
```

Với:

```
n               là số thứ tự của control file
database_name   tên của database
```

Tên của các control files được đặt phân cách nhau bởi các dấu phẩy.

Ví dụ:

```
logfile =  ‘C:\ORANT\DATABASE\LOG1KTKB.ORA’ SIZE 1024K,
           ‘C:\ORANT\DATABASE\LOG2KTKB.ORA’ SIZE 1024K
```

Datafiles

Tên của datafiles nên được đặt theo như nội dung của nó.

Đối với các data files, ta cần quan tâm tới một số tính chất sau:

- Giảm thiểu việc phân đoạn trong các data files.
- Tách riêng các đối tượng trong database như tách các application data, temporary data trên các tablespaces khác nhau.

Các datafile được phân chia theo các segment khác nhau. Tên của chúng thường được đặt với đuôi là **.DBF** còn phần đầu sẽ được phân theo từng loại segment tương ứng.

Ví dụ:

```
C:\ORANT\DATABASE\KTKB\SYSTEM01.DBF
C:\ORANT\DATABASE\KTKB\RBS01.DBF
C:\ORANT\DATABASE\KTKB\RBS02.DBF
C:\ORANT\DATABASE\KTKB\USERS01.DBF
C:\ORANT\DATABASE\KTKB\TEMP01.DBF
```

```
C:\ORANT\DATABASE\KTKB\TOOLS01.DBF
C:\ORANT\DATABASE\KTKB\INDX01.DBF
```

4.2.3. Optimal Flexible Architecture – OFA

Điều quan trọng khi tạo database là tổ chức các file hệ thống sao cho dễ dàng cho việc quản trị, thêm mới và bổ sung các dữ liệu vào database tận dụng hiệu quả các thao tác vào ra của hệ thống.

OFA với các tiện ích giúp cho việc bảo trì database được đơn giản.

Cáu trúc của OFA:

1. Đặt tên các thiết bị để nó có thể chứa đựng các dữ liệu Oracle server giống như một tập hợp.
2. Phân biệt các file sản phẩm, bao gồm các phần mềm và các công cụ Oracle server, các file quản trị, file script khởi tạo,...
3. Lưu lại các phiên bản của các sản phẩm Oracle server
4. Tạo các thư mục lưu trữ dữ liệu Oracle server.

4.2.4. Cấu trúc thư mục phần mềm Oracle

Thư mục	Diễn giải
Bin	Chứa các file sản phẩm ở dạng nhị phân
Dbs	Chứa các file dữ liệu
Lib	Chứa các file thư viện sản phẩm của Oracle
Orainst	Chứa chương trình và các file phục vụ cho việc cài đặt
Rdbms	Các file server, các file thư viện và các file khác cần thiết cho database
Plsql	PL/SQL và các sản phẩm liên quan
Sqlplus	SQL*Plus
Network	Các sản phẩm Oracle Net8
Svrmgrl	Server manager

Cấu trúc thư mục con

Thư mục	Diễn giải
Admin	File scripts quản trị
Demo	File dữ liệu và các scripts minh hoạ
Doc	README file
Install	Các file phục vụ cho việc cài đặt
Lib	Các thư viện sản phẩm

Log	Các file log

4.2.5. Biến môi trường

Trên hệ điều hành Windows, ta thiết lập các biến môi trường. Các biến này tương ứng với các tham số trong registry như: ORACLE_HOME, ORACLE_SID, NLS_LANG.

Để tạo mới database, cần tạo mới biến môi trường ORACLE_SID:

```
C:\set ORACLE_SID = U16
```

4.3.CHUẨN BỊ CÁC THAM SỐ TRONG PARAMETER FILE

Khi tạo mới một database, ta cần quan tâm tới việc tạo parameter file. Parameter file chứa các thông tin cần thiết trong database, trong đó quan tâm nhất là các tham số sau:

Tham số	Diễn giải
DB_NAME	Tên định danh của database, tối đa 8 ký tự. Tên database phải trùng với giá trị của biến môi trường ORACLE_SID.
CONTROL_FILES	Liệt kê danh sách các control file sử dụng trong database. Tối thiểu có 01 control file trong database. Tuy nhiên, ta nên tạo 02 control files trở lên để đề phòng hỏng file. Các control files không cần thiết phải tồn tại. Khi tạo database, Oracle sẽ tạo các control files này
DB_BLOCK_SIZE	Xác định kích thước của một block sử dụng trong database. Kích thước này sẽ không thay đổi được sau khi database đã được tạo lập. Kích thước của các block được tính theo đơn vị K (Kilobytes). Kích thước của block thường được đặt bằng số nguyên lần luỹ thừa của 2. để tương ứng với số nguyên lần các block vật lý của hệ điều hành. Do đó, có thể tối ưu được số lần truy xuất đĩa cứng. Ví dụ: 2K, 4K, 8K, 16K, 32K, tuỳ theo phiên bản của Oracle và hệ điều hành.

Thông thường, khi chuẩn bị parameter file của một database sắp được tạo, ta có thể sao chép lại nội dung của parameter file mẫu rồi chỉnh sửa lại một vài thông số trong đó như db_name, control_files,...

Parameter file mẫu của oracle thường được đặt ở thư mục:

```
<%ORACLE_HOME%>\ADMIN\SAMPLES\PFILE
```

Ví dụ về nội dung của file tham số: file InitU16.ora

```
db_name = U16
db_files = 1020
control_files = ("C:\ORANT\database\ctl1U16.ora",
"C:\ORANT\database\ctl2U16.ora")
db_file_multiblock_read_count = 16
db_block_buffers = 2000
shared_pool_size = 30000000
log_checkpoint_interval = 8000
processes = 100
dml_locks = 200
log_buffer = 65536
sequence_cache_entries = 30
sequence_cache_hash_buckets = 23
#audit_trail = true
#timed_statistics = true
background_dump_dest = C:\ORANT\rdbms80\trace
user_dump_dest = C:\ORANT\rdbms80\trace
db_block_size =8192
compatible = 8.0.4.0.0
sort_area_size = 65536
log_checkpoint_timeout = 0
remote_login_passwordfile = shared
max_dump_file_size = 10240
```

4.4.CHUẨN BỊ INSTANCE PHỤC VỤ QUẢN TRỊ

Sử dụng công cụ ORADIM để tạo instance phục vụ cho việc tạo database. ORADIM sẽ tạo một service dành riêng cho database. Đây là một công cụ thực hiện ở chế độ dòng lệnh. Công cụ này chỉ cần thiết khi user tạo mới, sửa đổi hay huỷ instance của database bằng tay. Trong trường hợp sử dụng công cụ Oracle Database Configuration Assistant để can thiệp vào database thì không cần thiết phải biết tới công cụ này.

ORADIM	Oracle Database Configuration Assistant
Có thể tạo mới, start, stop, sửa đổi hay xoá bỏ instances. Không can thiệp tới database files	Chỉ có thể tạo mới hay huỷ bỏ databse. Không thể start hay stop database
Có thể sử dụng để sửa đổi instance	Không thể để sửa đổi instance
Dùng để tạo password file và service liên quan. Không tạo database được	Dùng để tạo password file và service liên quan, instance và cả database

Lưu ý: Ở các phiên bản trước của Oracle, công cụ ORADIM có tên là ORADIM80

4.4.1. Tạo một instance

Cú pháp:

```
C:\>ORADIM -NEW -SID SID | -SRVC SERVICE_NAME [-INTPWD
           INTERNAL_PWD] - SHUTTYPE SRVC | INST | SRVC, INST [-
           MAXUSERS NUMBER][-STARTMODE AUTO | MANUAL][-PFILE
           FILENAME]
```

Với:

```
-NEW            Tạo mới instance phục vụ cho database.
-SID SID        Tên của instance được tạo (tên này thường được
                lấy chính là tên của database).
-SRVC SERVICE_NAME
                Tên của service phục vụ database.
-INTPWD INTERNAL_PWD
                Mật khẩu của Internal account sử dụng để quản
                trị database
-MAXUSERS NUMBER
                Số lượng user tối đa định nghĩa trong password
                file
-STARTMODE AUTO, MANUAL
                Đặt chế độ khởi động instance phục vụ (khởi
                động service trên máy chủ server)
-PFILE FILENAME
                Chỉ rõ parameter file INIT<Database_name>.ORA
-SHUTTYPE SRVC, INST
                Dừng instance phục vụ (stop service)
```

Ví dụ:

```
C:\> ORADIM -NEW -SID PROD -INTPWD MYPASSWORD1 -STARTMODE AUTO
           -PFILE C:\ORACLE\ADMIN\PROD\PFILE\INIT.ORA
```

4.4.2. Khởi động instance

Cú pháp:

```
C:\ORADIM -STARTUP -SID SID [-USRPWD USER_PWD] [-STARTTYPE SRVC
           | INST | SRVC, INST] [-PFILE FILENAME]
```

Với:

```
-STARTUP        Khởi động instance phục vụ sẵn sàng cho việc
                tạo database.
-SID SID        Tên của instance được tạo (tên này thường được
                lấy chính là tên của database).
-USERPWD USER_PWD
                Mật khẩu.
-STARTTYPE SRVC, INST
```

```
                    Chế độ khởi động là service hay instance
```

Ví dụ:

```
C:\> ORADIM      -STARTUP -SID PUMA -STARTTYPE SRVC
                 -PFILE C:\ORACLE\ADMIN\PROD\PFILE\INIT.ORA
```

4.4.3. Dừng instance

Cú pháp:

```
C:\>ORADIM -SHUTDOWN -SID SID  [-USRPWD USER_PWD] [-SHUTTYPE
           SRVC | INST | SRVC, INST] [-SHUTMODE A | I | N]
```

Với:

```
-SHUTDOWN        Dừng (stop) instance phục vụ.
-SID SID         Tên của instance được tạo (tên này thường được
                 lấy chính là tên của database).
-USERPWD USER_PWD
                 Mật khẩu.
-SHUTMODE        Xác định chế độ dừng: A – abort mode,        I I
                 – Immediate mode, N – Normal mode
```

Ví dụ:

```
C:\> ORADIM -SHUTDOWN -SID PUMA -SHUTTYPE SRVC INST
```

4.4.4. Huỷ instance

Cú pháp:

```
C:\>ORADIM -DELETE -SID sid
```

Ví dụ:

```
C:\> ORADIM -DELETE -SID PUMA
```

4.5.TẠO DATABASE

4.5.1. Khởi động Instance

Sử dụng user với mức quyền DBA. Dùng công cụ ORADIM để tạo Instance.

Khởi động Instance ở chế độ NOMOUNT và chỉ rõ file tham số sử dụng trong chương trình:

```
SVRMGR> STARTUP NOMOUNT \
      > PFILE=initU16.ora
```

4.5.2. Lệnh tạo database

Sử dụng câu lệnh CREATE DATABASE để tạo database

Cú pháp:

```
CREATE DATABASE [database]
      [CONTROLFILE REUSE]
      [LOGFILE [GROUP integer] filespec
      [, [GROUP integer] filespec]...]
      [MAXLOGFILES integer]
      [MAXLOGMEMBERS integer]
      [MAXLOGHISTORY integer]
      [MAXDATAFILES integer]
      [MAXINSTANCES integer]
      [ARCHIVELOG|NOARCHIVELOG]
      [CHARACTER SET charset]
      [NATIONAL CHARACTER SET charset]
      [DATAFILE filespec [autoextend_clause]
      [, filespec [autoextend_clause]...]]

filespec :== 'filename' [SIZE integer][K|M] [REUSE]

autoextend_clause :==
      [AUTOEXTEND {OFF
            |ON [NEXT integer[K|M]]
            [MAXSIZE {UNLIMITED|integer[K|M]}]
            }
      ]
```

Với:

Database	Tên của CSDL cần tạo (tên này giống với tên của tham số DB_NAME trong parameter file)
CONTROLFILE REUSE	Tên file tham số đã tồn tại được tái sử dụng
LOGFILE GROUP	Tên của log file được sử dụng
MAXLOGFILES	Số lượng tối đa các log file group cho CSDL
MAXLOGMEMBERS	Số lượng tối đa các log file member đối với một log file group
MAXLOGHISTORY	Số lượng tối đa các redo log trong một group
DATAFILE filespec	Tên file dữ liệu được sử dụng
AUTOEXTEND	Cho phép hoặc không cho phép mở rộng tự động các file dữ liệu
MAXDATAFILES	Số lượng tỗi đa các datafiles trong database
MAXINSTANCES	Số lượng lớn nhất các instance có thể đồng thời mount và open database

ARCHIVELOG
: Xác định ràng redo log cần để ở chế độ archive trước khi được dùng lại

NOARCHIVELOG
: Xác định ràng redo log cần được dùng lại mà không cần đặt chế độ archive

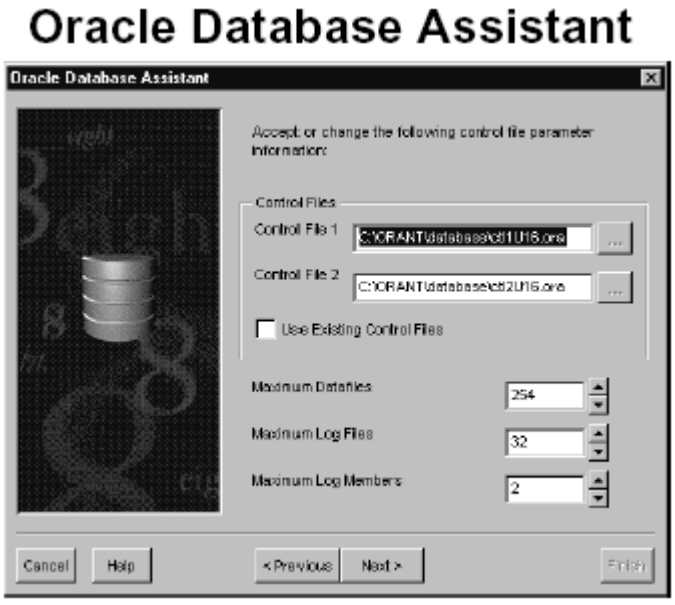
CHARACTER SET, NATIONAL CHARACTER SET
: Chuẩn ký tự mà CSDL sử dụng để lưu trữ các dữ liệu

Ví dụ: tạo database

```
SPOOL creU16.log

STARTUP NOMOUNT PFILE=initU16.ora

CREATE DATABASE U16
      MAXLOGFILES 5
      MAXLOGMEMBERS 5
      MAXDATAFILES 100
      MAXLOGHISTORY 100
      LOGFILE
      GROUP 1 ('/DISK3/log1a.rdo',/DISK4/log1b.rdo') SIZE 1 M,
      GROUP 2 ('/DISK3/log2a.rdo',/DISK4/log2b.rdo') SIZE 1 M
      DATAFILE
      '/DISK1/system01.dbf' size 50M autoextend on
      CHARACTER SET WE8ISO8859P1;
```

4.5.3. Oracle Database Assistant

Để tạo database, Oracle hỗ trợ công cụ rất tiện lợi giúp người quản trị dễ dàng tạo database hơn thông qua giao diện đồ hoạ, đó là công cụ Oracle Database Assistant.



Hình vẽ 12. Công cụ tạo hỗ trợ database – Oracle Database Assistant

Với công cụ này, người quản trị chỉ việc khai báo các tham số cần thiết cho database. Oracle Database Assistant sẽ tự động kết sinh ra câu lệnh SQL tương ứng với các tham số đã được khai báo. Các câu lệnh SQL có thể được chạy luôn hoặc cũng có thể được lưu lại thành các script files sử dụng sau này.

4.5.4. File script ví dụ tạo một database

File sqlu16.bat

```
set ORACLE_SID=U16
C:\ORANT\bin\oradim -new -sid U16 -intpwd oracle -startmode
auto -pfile C:\ORANT\database\initU16.ora
C:\ORANT\bin\oradim -startup -sid U16 -starttype srvc,inst
-usrpwd oracle -pfile C:\ORANT\database\initU16.ora
C:\ORANT\bin\svrmgr @U16run.sql
```

File U16run.sql

```
spool C:\ORANT\database\spoolmain
set echo on
connect INTERNAL/oracle
startup nomount pfile=C:\ORANT\database\initU16.ora
CREATE DATABASE U16
LOGFILE 'C:\ORANT\database\logU161.ora' SIZE 1024K,
'C:\ORANT\database\logU162.ora' SIZE 1024K
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXLOGHISTORY 1
DATAFILE 'C:\ORANT\database\Sys1U16.ora' SIZE 50M
MAXDATAFILES 100
MAXINSTANCES 1
CHARACTER SET WE8ISO8859P1
NATIONAL CHARACTER SET WE8ISO8859P1;
spool off
```

File initU16.ora

```
db_name = U16
db_files = 1020
control_files = ("C:\ORANT\database\ctl1U16.ora",
"C:\ORANT\database\ctl2U16.ora")
db_file_multiblock_read_count = 16
db_block_buffers = 2000
shared_pool_size = 30000000
log_checkpoint_interval = 8000
processes = 100
dml_locks = 200
log_buffer = 65536
```

```
sequence_cache_entries = 30
sequence_cache_hash_buckets = 23
#audit_trail = true
#timed_statistics = true
background_dump_dest = C:\ORANT\rdbms80\trace
user_dump_dest = C:\ORANT\rdbms80\trace
db_block_size =8192
compatible = 8.0.4.0.0
sort_area_size = 65536
log_checkpoint_timeout = 0
remote_login_passwordfile = shared
max_dump_file_size = 10240
```

4.5.5. Lỗi xảy ra khi tạo database

Lỗi xảy ra khi tạo database phần lớn do các nguyên nhân sau:

- Lỗi cú pháp lệnh tạo database
- Các file dữ liệu cần tạo lập đã tồn tại
- Lỗi do hệ điều hành, không có đủ quyền, không đủ chỗ trống,...

4.5.6. Kết quả sau khi tạo database

Kết thúc các bước trên ta thu được một database với:

- 02 data files được đặt trong SYSTEM tablespace.
- Các control files và các redo log files phục vụ cho database
- Hai user quản trị database và mật khẩu tương ứng là: `SYS/change_on_install` và `SYSTEM/manager`
- 01 Rollback segment SYSTEM
- Các bảng dữ liệu internal với dữ liệu trống

4.6.TẠO DATA DICTIONARY CHO DATABASE

Trong trường hợp tạo database bằng tay, sau khi tạo xong database, Oracle server sẽ tạo cho ta một database hoàn toàn trống. Các bảng trong database này đều được lưu trữ dưới dạng mã và ta không thể nào quan sát các thông tin trong nó được. Để có thể quan sát được các thông tin trong database. Ta cần tạo data dictionary cho database này.

Data dictionary hay còn gọi là từ điển dữ liệu của database là tập hợp các views được thiết lập trong database cung cấp các thông tin về database.

Các file tạo data dictionary cho database được Oracle cung cấp sẵn và thường được đặt trong thư mục `<%ORACLE_HOME%>\RDBMS\ADMIN`

Các dictionary views được phân loại và đặt trong các file SQL khác nhau.

Một số file SQL hay dùng:

Tên file SQL	Diễn giải

CATALOG.SQL	Tạo các dictionary views cơ bản, trigger và store procedure cơ sở
CATPROC.SQL	Tạo các package cơ sở
CATREP.SQL	Tạo các chức năng Replication cho database

Ngoài ra còn có rất nhiều file script khác.

Chương 5. QUẢN TRỊ ORACLE DATABASE

5.1.PHÂN LOẠI USERS

Oracle là một hệ quản trị cơ sở dữ liệu lớn, chạy trên môi trường mạng. Để vận hành hệ thống được tốt, có thể có nhiều người sẽ cùng tham gia vào hệ thống với những vai trò khác nhau gọi là các user. Có thể phân ra làm một số loại user chính sau:

- Database Administrators
- Security Officers
- Application Developers
- Application Administrators
- Database Users
- Network Administrators

5.1.1. Database Administrators

Do hệ thống Oracle database có thể là rất lớn và có nhiều users cùng tham gia vào hệ thống, và khi đó sẽ có một hay một số người chịu trách nhiệm quản lý hệ thống. Những người có vai trò như vậy được gọi là *database administrator* (DBA). Mỗi một database cần ít nhất 01 người để thực hiện công việc quản trị.

Một database administrator có trách nhiệm thực hiện một số công việc sau:

- Cài đặt và nâng cấp Oracle server và các công cụ ứng dụng khác.
- Phân phối hệ thống lưu trữ và lên kế hoạch lưu trữ cho hệ thống cơ sở dữ liệu trong tương lai.
- Tạo những cấu trúc lưu trữ cơ bản như tablespaces phục vụ cho việc phát triển và hoạt động của các ứng dụng.
- Tạo các đối tượng trong database như tables, views, indexes sử dụng cho các ứng dụng được thiết kế.
- Thay đổi cấu trúc database khi cần thiết tuỳ theo các thông tin của các application.
- Quản lý các users và đảm bảo bảo mật hệ thống.
- Đảm bảo tương thích về bản quyền, phiên bản với hệ thống Oracle
- Điều khiển và quản trị các user access truy xuất tới database.
- QUản lý và tối ưu các truy xuất tới database.
- Lên kế hoạch backup (sao lưu) và recovery (phục hồi) các thông tin có trong database.
- Lưu trữ các archive data.
- Sao lưu và khôi phục database.
- Cập nhật các công nghệ mới đưa ra các câu hỏi bổ ích.

5.1.2. Security Officers

Trong một số trường hợp, hệ thống đòi hỏi chế độ bảo mật cao. Khi đó cần đến một hay một nhóm người chuyên thực hiện công tác bảo vệ database gọi là security officers. Security officer có thể kết nối tới database, điều khiển và quản lý việc truy cập database của các users và bảo mật hệ thống.

5.1.3. Application Developers

Application developer là người thiết kế và viết các ứng dụng database. Application developer có trách nhiệm thực hiện một số yêu cầu sau:

- Thiết kế và phát triển ứng dụng database.
- Thiết kế cấu trúc database cho từng ứng dụng.
- Đánh giá yêu cầu lưu trữ cho ứng dụng.
- Quy định các hình thức thay đổi cấu trúc database của ứng dụng.
- Thiết lập biện pháp bảo mật cho ứng dụng được phát triển.

5.1.4. Database Users

Database users tương tác với database thông qua các ứng dụng và các tiện ích. Một user điển hình có thể thực hiện được một số công việc sau:

- Truy nhập, sửa đổi, và xoá huỷ các dữ liệu được phép
- Tạo các báo cáo đối với dữ liệu

5.1.5. Network Administrators

Đối với database Oracle hoạt đông trên môi trường mạng, khi đó cần có một user thực hiện công việc quản trị mạng. User này có trách nhiệm đảm bảo các ứng dụng Oracle hoạt động trên môi trường mạng được tốt.

5.2.PHƯƠNG THỨC XÁC NHẬN ĐẶC QUYỀN TRUY NHẬP

Việc phân quyền sử dụng là cần thiết trong công việc quản trị. Có hai user account được tự động tạo ra ngay từ khi tạo database và được gán quyền DBA là: SYS và SYSTEM.

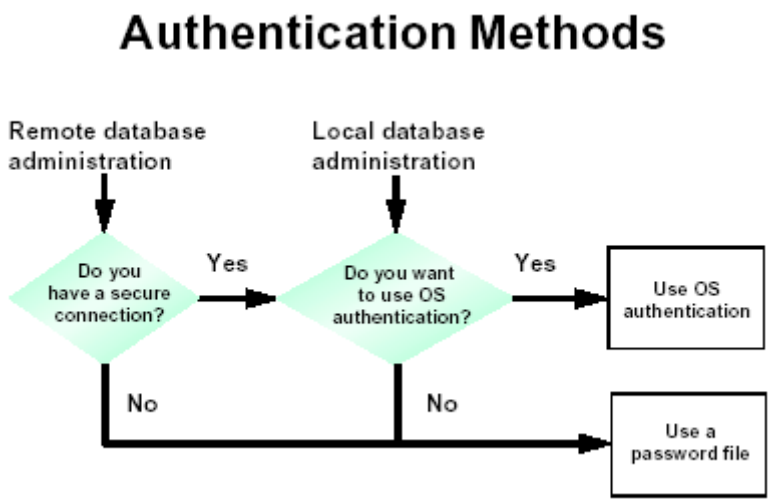
- **SYS:** được tạo tự động và gán quyền DBA. Mật khẩu mặc định là *change_on_install*. Có quyền sở hữu các bảng và các từ điển dữ liệu trong database.
- **SYSTEM:** được tự động tạo ra với mật khẩu ban đầu là *manager* và cũng được gán quyền DBA. Tuy nhiên, SYSTEM còn được sở hữu cả một số table, view mở rộng chứa các thông tin sử dụng cho các tools của Oracle.
- Quyền **DBA**: Ngay khi tạo database, Oracle đã tạo sẵn một quyền gọi là "DBA". Quyền này cho phép thực hiện các thao tác quản trị đối với database.

Lưu ý: Với quyền DBA, các users này sẽ có thể can thiệp được tới các quyền của các user khác sử dụng trong hệ thống. Vì thế, những quản trị viên database cần thay đổi mật khẩu của mình tránh sử dụng mật khẩu mặc định do Oracle cung cấp vì user khác có thể biết và sử dụng để truy nhập vào hệ thống một cách trái phép, gây xáo trộn hệ thống.

5.2.1. Phương thức xác nhận quyền

Trong một số trường hợp quản trị viên database cần đến phương thức xác nhận quyền truy nhập đặc biệt do có thể lúc đó database chưa được mở, ví dụ như với các trường hợp shutdown hoặc startup database.

Tuỳ thuộc vào việc quản trị database trên cùng một máy hay ở máy khác mà ta có thể sử dụng cơ chế xác nhận quyền truy nhập database bởi hệ điều hành hay hay password files.



Hình vẽ 13. Phương thức xác nhận quyền

5.2.2. Xác nhận quyền bởi hệ điều hành

Việc xác nhận quyền bởi hệ điều hành được tiến hành theo các bước:

1. Trong hệ điều hành Windows NT tạo một user's group với tên `ORA_<SID>_DBA` và một nhóm khác `ORA_<SID>_OPER` với `<SID>` tương ứng với tên của instance, hoặc `ORA_DBA` và `ORA_OPER` (khi này ta không quan tâm tới instance).
2. Thêm một user vào group để khi truy cập vào hệ điều hành, user có thể tự động được xác định quyền DBA.
3. Đặt tham số `REMOTE_LOGIN_PASSWORDFILE` trong parameter file là `NONE`.
4. Kết nối tới database với mức quyền SYSDBA hay SYSOPER:

```
CONNECT / AS { SYSDBA|SYSOPER }
```

Ghi chú:

- NET8 được cài đặt trên các hệ điều hành Windows 95, Windows NT để giúp cho việc xác nhận quyền.

- Các phiên bản trước của Oracle sử dụng lệnh: CONNECT INTERNAL với cú pháp: CONNECT INTERNAL/pw AS SYSDBA. Lệnh: CONNECT INTERNAL hiện tại vẫn được sử dụng.
- Với việc xác nhận quyền truy nhập bởi hệ điều hành, ta không cần quan tâm tới các mức quyền (privilege) thay vào đó, ta cần quan tâm tới hai quyền được cung cấp bởi hệ điều hành là OSDBA và OSOPER

OSOPER: là quyền cho phép user có thể STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, và RECOVER, ngoài ra còn có thêm cả quyền RESTRICTED SESSION.

OSDBA: là quyền cho phép user có thể có được tất cả các quyền của OSOPER, ngoài ra còn có thêm một số mức quyền phục vụ quản trị database là ADMIN OPTION, và CREATE DATABASE

5.2.3. Xác nhận quyền bằng file mật khẩu

Oracle hỗ trợ các tiện ích password cho phép kết nối tới Oracle Server sử dụng username và password. Việc truy cập vào database sử dụng password file được hỗ trợ bởi lệnh GRANT.

Sử dụng file mật khẩu:

1. Tạo file mật khẩu bằng lệnh:

   ```
   orapwd file=<fname> password=<password> entries=<entries>
   ```

   Với:

fname	là tên file mật khẩu
password	là mật khẩu của SYS hay INTERNAL
entries	là số lượng tối đa các quản trị viên được phép

2. Đặt tham số REMOTE_LOGIN_PASSWORDFILE là EXCLUSIVE hoặc SHARED.

   Với:

EXCLUSIVE	chỉ một instance có thể sử dụng file mật khẩu
SHARED	nhiều instance có thể dùng file mật khẩu

3. Gán quyền cho user

   ```
   GRANT SYSDBA TO admin;
   GRANT SYSOPER TO admin
   ```

4. Kết nối tới database theo cú pháp:

   ```
   SVRMGRL>CONNECT internal/admin AS SYSDBA
   ```

Xem thông tin về các member trong file mật khẩu

Thông tin về các member trong file mật khẩu được lưu trong view: V$PWFILE_USER. Nó cho biết có những user nào được gán quyền SYSDBA hay SYSOPER.

Diễn giải một số cột trong V$PWFILE_USER:

USERNAME	Tên user
SYSDBA	Cột này nhận giá trị TRUE thì User này được gán quyền SYSDBA
SYSOPER	Cột này nhận giá trị TRUE thì User này được gán quyền SYSOPER

Khi kết nối với database theo mức quyền SYSDBA hay SYSOPER, user đó sẽ được kết nối tới các schema mặc định, với SYSDBA thì shema mặc định là SYS, với SYSOPER thì shema mặc định là PUBLIC.

5.2.4. Thay đổi mật khẩu internal

Sử dụng tiện ích ORADIM để tạo lại file mật khẩu.

```
C:\>ORADIM -NEW -SID sid [-INTPWD internal_pwd][SRVC svrcname]
           [MAXUSERS n][STARTMODE auto, manual][-PFILE filename]
```

Với:

sid	tên instance
internal_pwd	mật khẩu internal account
svrcname	tên service
n	số lượng tối đa file mật khẩu
auto or manual	chế độ khởi động service là: manual hay automatic
filename	cho phép sử dụng file mật khẩu không phải là mặc định

Để thay đổi mật khẩu INTERNAL ta thực hiện theo các bước sau:

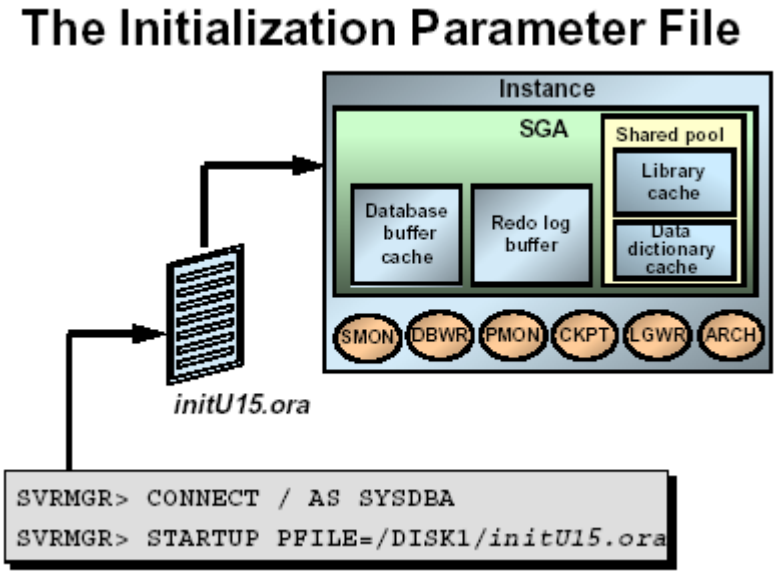
1. Xoá mật khẩu cũ

   ```
   C:\> ORADIM -DELETE -SID sid
   ```

2. Tạo mật khẩu mới

   ```
   C:\> ORADIM -NEW -SID sid -INTPWD internal_pwd - MAXUSERS N
   ```

5.3.TẠO PARAMETER FILE



Hình vẽ 14. Khởi tạo tham số

File tham số thông thường có tên `init<SID>.ora`.

Theo mặc định, file tham số được đặt trong thư mục %ORACLE_HOME%\DATABASE.

File tham số chỉ được đọc một lần khi khởi động instance. Khi thay đổi nội dung của file tham số, để sử dụng được các giá trị mới cần shut down rồi sau đó restart lại instance.

5.3.1. Sử dụng các tham số

Các tham số có thể ảnh hưởng tới hiệu quả sử dụng database. Các thông số trong file tham số bao gồm:

- Kích thước của vùng System Global Area (SGA) để tối ưu hiệu suất.
- Đặt mặc định cho database và instance.
- Đặt các hạn chế đối với user hay process.
- Đặt các hạn chế đối với tài nguyên database.
- Xác định các thuộc tính vật lý của database, như kích thước của block.
- Chỉ ra các control files, archived log files, Alert file, và trace file locations.

5.3.2. Một số quy tắc đối với các tham số

125. Các giá trị được chỉ ra theo khuôn dạng: `<Keyword> = <Giá trị>`.
126. Một số tham số đều là tuỳ chọn và một số khác là bắt buộc ví dụ như DB_NAME.
127. Server đều có giá trị mặc định đối với mỗi tham số. Các giá trị này là tuỳ theo hệ điều hành và tuỳ theo tham số.
128. Các tham số có thể được chỉ ra không cần phải tuân theo một thứ tự nào cả (đặt trước, sau không quan trọng).
129. Phần chú dẫn được bắt đầu bằng ký hiệu #.
130. Các tham số là ký tự được đặt trong dấu nháy kép.
131. Cũng có thể included các file bởi từ khoá IFILE.

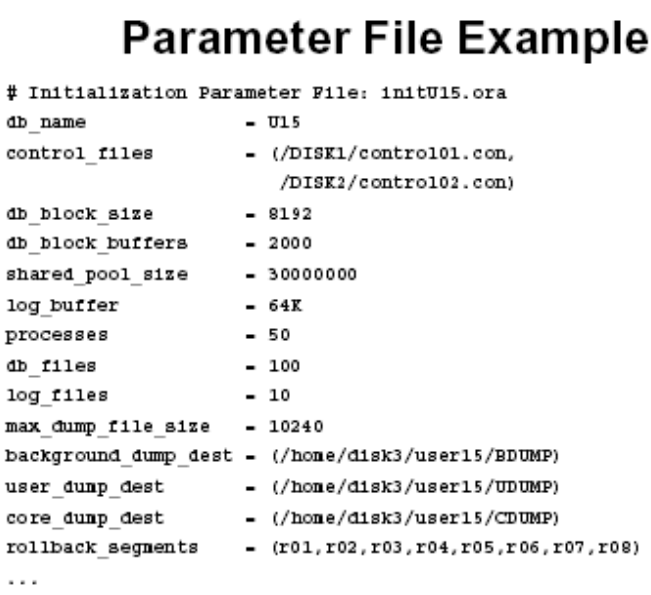
132. Các giá trị là tập hợp được đặt trong dấu ngoặc đơn ‘(,)’và được ngăn cách nhau bởi dấu phẩy (,).

5.3.3.

5.3.4. Các tham số cơ bản

Tham số	Diễn giải
CONTROL_FILES	Tên của các control files.
DB_BLOCK_BUFFERS	Số lượng các data blocks được cach trong SGA.
DB_BLOCK_SIZE	Kích thước của một data block. Kích thước này nên được chọn bằng số số nguyên lần mũ 2, có thể là 2K, 4K, 8K, 16K và 32K tuỳ theo phiên bản của Oracle và của Hệ điều hành.
DB_NAME	Định danh database từ 8 ký tự trở xuống. Tham số này chỉ cần thiết khi tạo mới một database.
IFILE	Tên của file tham số được include vào file tham số hiện thời. Cho phép có thể được lồng tối đa là ba cấp.
LOG_BUFFER	Số byte được cấp phát cho redo log buffer trong SGA.
MAX_DUMP_FILE_SIZE	Kích thước tối đa của trace files, được xác định bằng số lượng block của hệ điều hành.
OPEN_CURSOR	Số lượng cursor tối đa được đồng thời mở.
ROLLBACK_SEGMENTS	Số lượng rollback segments được sử dụng cho mỗi instance
PROCESSES	Số lượng tối đa các tiến trình hệ điều hành có thể kết nối với instance.
SHARED_POOL_SIZE	Kích thước của Shared Pool

Ví dụ một parametersfile:



Hình vẽ 15. File tham số ví dụ

5.4.START VÀ SHUT DOWN DATABASE

5.4.1. Các bước Start và Shut down database

Start Instance ở chế độ Nomount

Ta có thể khởi động một Instance mà không cần thiết phải gắn với một database cụ thể. Khi khởi động Instance, các công việc sau đây sẽ được thực hiện:

- Đọc file tham số : `init<SID>.ora`
- Thu xếp vùng bộ nhớ SGA
- Khởi động các background process
- Mở các trace file và các Alert file

Lưu ý: Tên database nằm trong tham số `DB_NAME` của file tham số.

Câu lệnh:

```
STARTUP NOMOUNT;
```

Start Instance ở chế độ mount

Để thực hiện một vài thao tác đặc biệt khi vận hành database, ta có thể khởi động một instance và mount database nhưng chưa mở database.

Ví dụ như:

- Đổi tên datafiles
- Enable hoặc Disable các redo log files
- Thực hiện phục hồi dữ liệu (recovery).

Các công việc khi mount database:

- Gắn database với một instance đã khởi động

- Định vị và mở các control files theo như thông số có trong file tham số
- Đọc nội dung của control file và xác định trạng thái cho các data files và các redo log files.

Câu lệnh:

```
STARTUP MOUNT;
```

Start Instance ở chế độ open

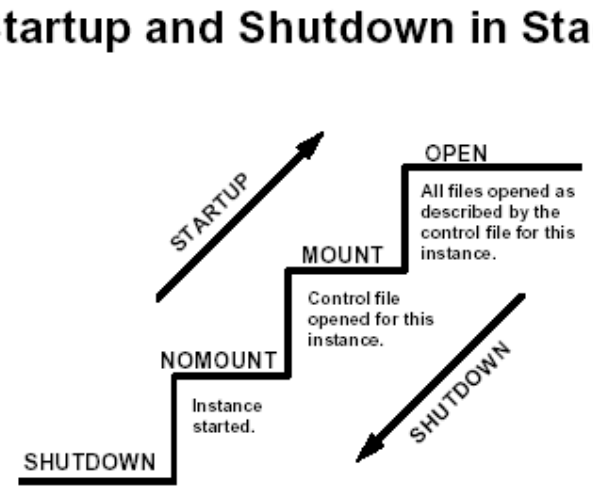
Sau khi databsae đã được mở, những người sử dụng hợp lệ có thể kết nối tới database và thực hiện các thao tác truy nhập vào database.

Việc mở database diễn ra theo hai bước:

- Mở các online data files
- Mở các online redo log files.

Câu lệnh:

```
STARTUP OPEN;
```



Hình vẽ 16. Các bước khởi động và dừng Instance

Khôi phục Instance

Trong một số trường hợp Instance có thể gặp lỗi và không thể làm việc được. Ví dụ như: có lỗi hệ thống xảy ra. Việc khôi phục Instance sẽ được thực hiện theo các bước sau:

- Khôi phục lại tất cả các dữ liệu có thể khôi phục được (dữ liệu chưa được lưu vào data files nhưng đã lưu vào trong online redo log files)
- Mở database.
- Khôi phục lại tất cả các transaction chưa được commit.

Close database

Đây là bước đầu tiên khi tắt hẳn một database. Sau khi đóng database, tất cả các dữ liệu còn trong bộ đệm (redo log buffer cache) sẽ được ghi ra file (online redo log file). Các control file vẫn được mở.

Dismount database

Dissmount database sẽ đóng nốt các control file thuộc database đang mở.

Shoutdown Instance

Đây là bước cuối cùng, intance sẽ được tắt hẳn. Các trace file và Alert file của instance bị đóng. Các background process bị dừng và vùng nhớ SGA cấp cho instance bị thu hồi.

5.4.2. Start database

Cú pháp:

```
STARTUP [FORCE] [RESTRICT] [PFILE=filename]
      [EXCLUSIVE | PARALLEL | SHARED]
      [OPEN [RECOVER][database]|MOUNT |NOMOUNT]
```

Với:

OPEN	cho phép các users truy cập vào database.
MOUNT	mounts database sẵn sàng cho các thao tác DBA, người sử dụng chưa truy cập được database.
NOMOUNT	Bố trí SGA và khởi động các background process, chưa sẵn sàng cho DBA.
EXCLUSIVE	chỉ cho phép instance hiện thời truy cập vào database.
PARALLEL	cho phép nhiều instances cùng được gắn với database (sử dụng Oracle Parallel Server)
SHARED	tương tự như PARALLEL.
PFILE=parfile	cho phép sử dụng file tham số không phải là mặc định để xác định cấu hình cho instance.
FORCE	huỷ bỏ các instance đang chạy trước đó, khởi động instance bình thường.
RESTRICT	chỉ cho phép các users truy cập với chế độ RESTRICTED.
SESSION	quyền truy nhập vào database.
RECOVER	bắt đầu khôi phục dữ liệu khi database.

5.4.3. Thay đổi tính sẵn dùng của database hiện thời

Khởi động database ở chế độ NOMOUNT

Thực hiện sửa đổi database theo lệnh:

```
ALTER database { MOUNT | OPEN |
                 OPEN READ ONLY | OPEN READ WRITTE}
```

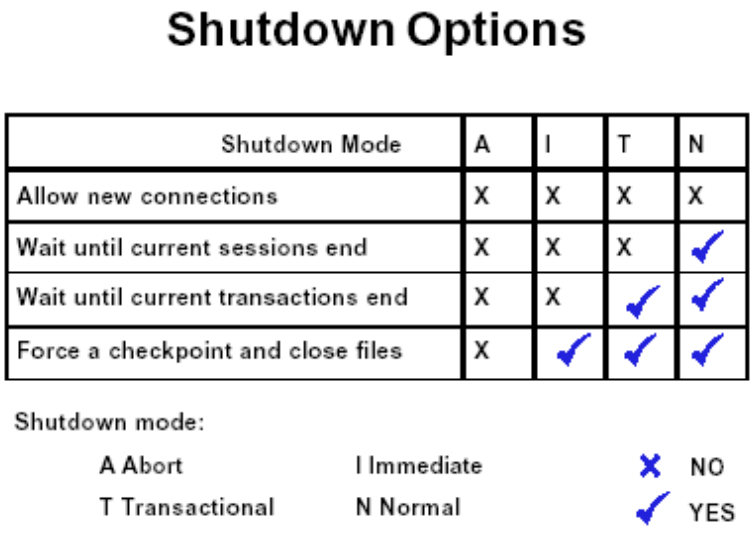
Với:

MOUNT	Gắn database với instance. Lúc này ta chỉ có thể thực hiện các thao tác quản trị trên database mà chưa thể sử dụng database được.

OPEN READ WRITE	Mở database, sẵn sàng cho việc sử dụng database, cả đọc lẫn ghi.
OPEN READ ONLY	Mở database nhưng chỉ cho đọc database như sử dụng các câu lệnh truy vấn chẳng hạn. Các thao tác ghi không thể thực hiện được. Tuỳ chọn này được sử dụng khi ta cần sao chép các redo log files của database.
OPEN	Tương tự như OPEN READ ONLY, đây là biểu diễn mặc định của OPEN READ WRITE.

5.4.4. Shut down database

Có một số chế độ tắt database tương ứng với các khả năng khác nhau.



Hình vẽ 17. So sánh các chế độ tắt database

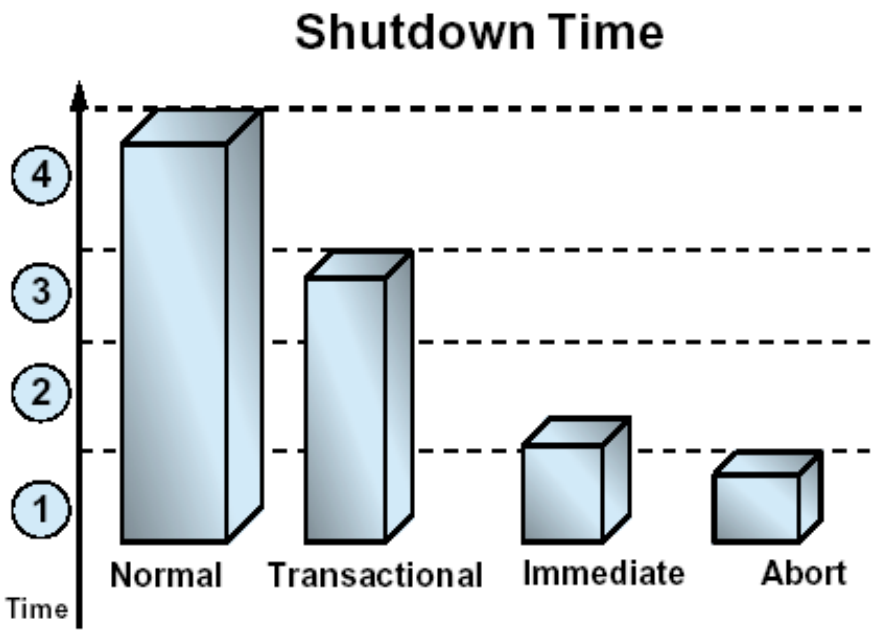
Cú pháp:

```
SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT ]
```

Với:

NORMAL	Không cho tạo thêm các connection tới database, chờ cho connection hiện thời kết thúc thì shutdown database.
TRANSACTION	Không cho phát sinh thêm các transaction, chờ cho transaction hiện thời kết thúc thì shutdown database.
IMMEDIATE	Kết thúc luôn transaction hiện thời nhưng vẫn chờ hệ thống commit hay rollback rồi mới shutdown database.
ABORT	Shutdown database tức thời không đòi hỏi bất cứ điều kiện gì.

Tương ứng với các cách tắt database trên, ta có biểu đồ về thời gian như sau:



Hình vẽ 18. So sánh thời gian giữa các cách tắt database

Hình vẽ trên so sánh tiêu tốn về thời gian khi thực hiện một thao tác chuyển đổi dữ liệu:

1. Thực hiện truy vấn để lấy dữ liệu
2. Thực hiện lệnh `INSERT` và `DELETE` để cập nhật và chuyển đổi dữ liệu
3. Phát lệnh `COMMIT` để cập nhật dữ liệu vào database
4. Huỷ bỏ liên kết tới database.

5.4.5. Thay đổi trạng thái của database

Cú pháp:

```
ALTER system { SUSPEND | RESUME }
SUSPEND          Đưa database vào trạng thái treo. Tạm thời
                 không cho phép thực hiện các thao tác vào ra
                 đối với datafiles và control files. Thao tác
                 này được thực hiện khi ta chuẩn bị backup
                 database.
RESUME           Ngược lại với SUSPEND, thao tác này sẽ đưa
                 database trở lại trạng thái bình thường sau khi
                 đã backup xong database.
```

Ví dụ:

```
SQL> ALTER SYSTEM SUSPEND;
System altered
SQL> SELECT database_status FROM v$instance;
DATABASE_STATUS
---------
SUSPENDED

SQL> ALTER SYSTEM RESUME;
System altered
```

```
SQL> SELECT database_status FROM v$instance;
DATABASE_STATUS
---------
ACTIVE
```

5.4.6. Tạm treo và phục hồi Database

Oracle9*i* cung cấp chức năng suspend/resume. Quản trị viên sử dụng lệnh ALTER SYSTEM SUSPEND để tạm treo database, dừng mọi thao tác truy xuất vào ra đối với các datafiles và control files. Khi database ở trạng thái tạm treo, các thao tác vào ra (I/O operations) đang thực hiện sẽ được kết thúc và những truy cập vào database mới phát sinh sẽ được đẩy vào queue. Thực hiện lệnh ALTER SYSTEM RESUME để khôi phục lại tình trạng bình thường của database.

Ta sử dụng lệnh ALTER SYSTEM SUSPEND để tạm treo một database, ngăn thực hiện các thao tác vào ra (I/O) đối với các datafiles và control files. Do đó, cho phép database có thể dễ dàng thực hiện các thao tác back up. Khi thực hiện việc treo database tất cả các thao tác vào ra đang có sẽ được tiếp tục cho phép thực hiện cho đến khi hoàn tất, các phép thao tác vào ra mới phát sinh sau này sẽ được tạm thời đưa vào queue chờ xử lý sau.

Lệnh suspend (tạm treo) database được thực hiện đối với database chứ không phải chỉ đối với instance. Do vậy, ở trong môi trường Oracle Real Application Clusters, một khi lệnh suspend được phát ra thì sau đó một cơ chế khoá sẽ được thiết lập và chặn tất cả các yêu cầu gửi tới instance.

Sử dụng lệnh ALTER SYSTEM RESUME để phục hồi (resume) lại các hoạt động thông thường của database. Ta cũng có thể chỉ rõ SUSPEND và RESUME từ các instances khác nhau. Ví dụ, nếu các instances 1, 2, và 3 đang chạy, và ta phát lệnh ALTER SYSTEM SUSPEND từ instance 1, sau đó ta cũng có thể phát lệnh RESUME từ các instances 1, 2, hay 3 đều như nhau.

Khả năng suspend/resume là rất hữu ích cho hệ thống nó cho phép ta thực hiện mirror một ổ đĩa hay một file rồi sau đó sử dụng vào việc sao lưu, phục hồi dữ liệu cho toàn bộ hệ thống.
.

Tuy vậy, đặc điểm suspend/resume không thay thế cho các thao tác normal shutdown database vì khi đó việc sao chép database được suspend có thể chứa cả các dữ liệu cập nhật chưa được commit.

Cau lệnh sau minh hoạ việc sử dụng lệnh ALTER SYSTEM SUSPEND/RESUME. Sử dụng thông tin cung cấp trong V$INSTANCE để biết được trạng thái của database.

```
SQL> ALTER SYSTEM SUSPEND;
System altered
SQL> SELECT DATABASE_STATUS FROM V$INSTANCE;
DATABASE_STATUS
---------
SUSPENDED


SQL> ALTER SYSTEM RESUME;
```

```
System altered
SQL> SELECT DATABASE_STATUS FROM V$INSTANCE;
DATABASE_STATUS
---------
ACTIVE
```

5.4.7. Đặt chế độ hoạt động tĩnh cho database

Oracle9*i* cho phép đưa database vào chế độ hoạt động tĩnh (quiesced state), Theo đó chỉ cáo các DBA transactions, queries, và các lệnh PL/SQL là được phép thực hiện. Trạng thái này cho phép người dùng thực hiện các thao tác quản trị một cách an toàn. Sự dụng câu lệnh ALTER SYSTEM QUIESCE RESTRICTED để đưa database về chế độ hoạt động tĩnh.

5.5. ĐẶT TRẠNG THÁI TĨNH CHO DATABASE

Có nhiều khi ta cần phải đưa database vào trạng thái mà chỉ có các DBA transactions, queries (truy vấn), fetches (tìm kiếm dữ liệu), hay các câu lệnh PL/SQL là được phép thực hiện. Chế độ này được gọi là quiesced state - tạm dịch là chế độ tĩnh. Chế độ này cho phép quản trị viên có thể thực hiện một số thao tác không an toàn lắm trên database bao gồm các thao tác sau đây:

- Các thao tác có thể gặp lỗi nếu đồng thời có một user transactions truy cập vào cùng một đối tượng. Ví dụ như khi thay đổi table, thêm mới cột dữ liệu vào một table đang có và không yêu cầu khoá (no-wait lock is required).
- Các thao tác không mong muốn gây ảnh hưởng tức thì giữa các user transactions xảy ra đồng thời. Ví dụ khi có một thủ tục chứa nhiều bước thao tác trên một table chẳng hạn như table ban đầu được export dữ liệu, rồi bị xoá đi và cuối cùng lại được import dữ liệu trở lại. Cùng lúc đó có user khác muốn truy cập vào table và ngay tại thời điểm table vừa bị huỷ. Khi này sẽ phát sinh lỗi hệ thống.

Nếu không áp dụng trạng thái tĩnh cho database, thì ta cần phải shutdown database rồi open lại nó ở chế độ restrict. Và việc này sẽ trở nên nghiêm trọng hơn khi hệ thống yêu cầu phải chạy liên tục 24 x 7. Áp dụng chế độ tĩnh cho database sẽ giảm bớt đi các hạn chế vì restriction vì nó loại bớt đi được các xấu xảy ra với database.

5.5.1. Đưa Database vào trạng thái tĩnh

Để đưa database vào trạng thái tĩnh, đơn giản ta chỉ cần sử dụng lệnh:

```
ALTER SYSTEM QUIESCE RESTRICTED
```

Tất cả các non-DBA active sessions sẽ được tiếp tục xử lý cho tới khi chúng chuyển sang trạng thái inactive. Một session được xem là active nếu lúc đó nó đang có các phép thực như transaction, query, fetch, hay đang xử lý một câu lệnh PL/SQL; hoặc cũng có thể là session đó đang nắm giữ phần tài nguyên chia sẻ (shared resources).

Khi tất cả các non-DBA sessions chuyển sang trạng thái inactive, câu lệnh `ALTER SYSTEM QUIESCE RESTRICTED` kết thúc và database được xem như là chuyển sang trạng thái tĩnh quiesce state. Trong môi trường `Oracle Real Application Clusters`, câu lệnh này có ảnh hưởng tới tất cả các instances, chứ không chỉ là đối với instance nơi phát ra câu lệnh.

Lệnh `ALTER SYSTEM QUIESCE RESTRICTED` có thể phải chờ trong một thời gian khá dài để cho active sessions chuyển sang trạng thái inactive. Nếu ta huỷ bỏ yêu cầu, hoặc nếu session bị kết thúc một cách đột ngột vì nhiều lý do khác nhau thì Oracle sẽ tự động phục hồi lại (undo) trạng thái trước khi thực hiện lệnh.

Nếu một truy vấn được đưa ra bởi các Oracle Call Interface (OCI), thì câu lệnh `ALTER SYSTEM QUIESCE RESTRICTED` sẽ không chờ fetch hết tất cả các dữ liệu mà chỉ chờ fetch xong dòng dữ liệu hiện thời mà thôi.

Khi ở trạng thái quiesce state, ta không sử dụng hệ điều hành để sao chép các file trong hệ thống giống như khi thực hiện backup lạnh đối với database, cho dù ta có các checkpoint tại mỗi một instance. Lý do là vì khi ở trạng thái quiesce state thì các file headers của online datafiles vẫn luôn được liên tục truy cập.

5.5.2. Phục hồi hệ thống trở lại hoạt động như bình thường

Thực hiện câu lệnh sau:

```
ALTER SYSTEM UNQUIESCE
```

Khi này tất cả các non-DBA activity sẽ được tiếp tục thực hiện. Trong môi trường Oracle Real Application Clusters, ta có thể phát lệnh này từ bất kỳ một instance nào có kết nối tới server không nhất thiết phải là instance phát lệnh đặt trạng thái tĩnh. Trong trường hợp session phát lệnh `ALTER SYSTEM UNQUIESCE` gặp lỗi, Oracle database server sẽ luôn đảm bảo việc thực hiện unquiesce sẽ kết thúc.

5.5.3. Xem trạng thái của database

Ta có thể xem trạng thái của database qua các thông tin có trong `V$INSTANCE` . Các thông tin này được lưu trong cột `ACTIVE_STATE` với các nội dung như sau:

ACTIVE_STATE	Diễn giải
NORMAL	Trạng thái thông thường
QUIESCING	Đang ở trạng thái tĩnh – quiesce state, nhưng các active non-DBA sessions vẫn được thực hiện
QUIESCED	Ở trạng thái quiesce state, và không có bất kỳ một active non-DBA sessions nào được phép thực hiện

5.6.LẤY CÁC THÔNG TIN VỀ HỆ THỐNG

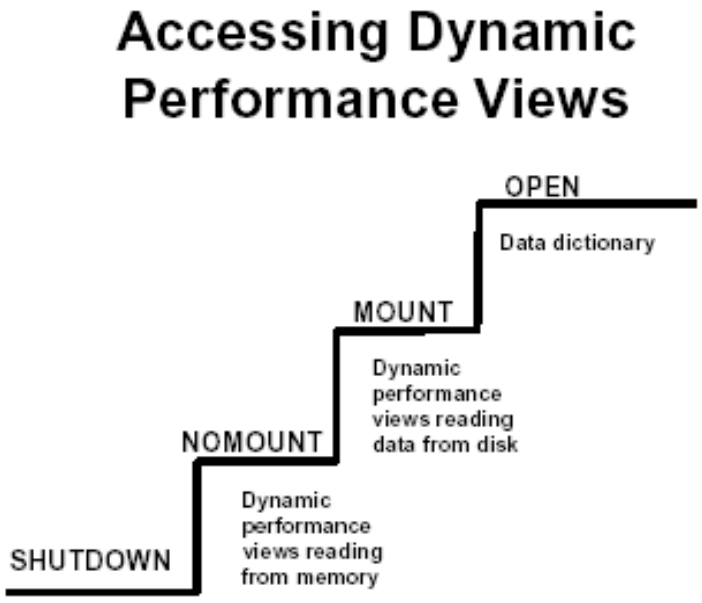
Các thông số hệ thống được đặt trong các tables hệ thống. Ta có thể quan sát và truy xuất tới chúng thông qua các view gọi là Dynamic performance views. Các view này thường có

tên viết đầu là V_$. Oracle thường tạo ra các Sysnonym tương ứng với các view này với tên có đầu là V$.

Khi khởi động database ở chế độ NOMOUNT, user quản trị có thể đọc được các dữ liệu có trong các view này. Thông tin trong view này là cần thiết cho việc mount database.

View V$FIXED_TABLE chứa tên của tất cả các view V$ có trong hệ thống.

Biểu đồ dưới đây diễn tả các mức độ truy cập các view của hệ thống



Hình vẽ 19. Các mức độ truy cập view hệ thống

5.6.1. Một số views cần quan tâm

Dynamic Performance View	Diễn giải
V$PARAMETER	Thông tin về các tham số khởi tạo
V$SGA	Thông tin tổng hợp về SGA
V$OPTION	Các tuỳ chọn cho Oracle server đã được cài đặt
V$PROCESS	Thông tin về các hoạt động của process hiên thời
V$SESSION	Thông tin về session
V$VERSION	Thông tin về phiên bản của các thành phần Oracle
V$INSTANCE	Thông tin về trạng thái của Instance hiện thời
V$THREAD	Thông tin về các thread trong hệ thống
V$CONTROLFILE	Liệt kê tên của các control files
V$DATABASE	Thông tin về database
V$DATAFILE	Thông tin về các data file được sử dụng
V$DATAFILE_HEADER	Thông tin header của các data file được sử dụng

V$LOGFILE	Thông tin về các online redo log files

5.6.2. Hiển thị giá trị của các thông số hệ thống

Ta có thể xem thông tin hệ thống bằng hai cách:

- Sử dụng lệnh xem tham số của Server manager.

```
SVRMGRL> SHOW PARAMETER control
```

- Truy xuất trực tiếp vào view hệ thống

```
SELECT name, type from v$control WHERE name like '%control
    %';
```

Với hai cách trên ta đều thu được một kết quả:

```
SVRMGR> SHOW PARAMETER control
NAME
                                 TYPE        VALUE
-------------------------------  -------   --------------------
control_file_record_keep_time    integer   7
control_files                    string    /DISK1/control01.con
```

5.6.3. Tham số hệ thống động (có thể thay đổi)

Trong các tham số hệ thống, có một vài tham số là động và ta có thể thay đổi được các tham số này. Thông qua các lệnh:

- `ALTER SESSION`: chỉ thay đổi giá trị của các tham số trong session hiện thời
- `ALTER SYSTEM`: thay đổi giá trị trong toàn bộ hệ thống nói chung.
- `ALTER SYSTEM DEFERRED`: chỉ thay đổi tham số hệ thống của các session sẽ kết nối vào database sau này, kể từ sau thời điểm thay đổi.

Cú pháp:

```
ALTER SESSION SET parameter_name = value
ALTER SYSTEM SET parameter_name = value [DEFERRED]
```

Ví dụ:

```
ALTER SESSION SET SQL_TRACE=true;
ALTER SYSTEM SET TIMED_STATISTICS=true;
ALTER SYSTEM SET SORT_AREA_SIZE=131072 DEFERRED;
```

Xem lại thông tin mà ta vừa thay đổi:

```
SVRMGR> SELECT isses_modifiable,issys_modifiable,
3> ismodified, name
2> FROM v$system_parameter
4> WHERE ismodified != 'FALSE';
ISSES      ISSYS_MOD ISMODIFI NAME
-----      --------- -------- ------------------------------
TRUE       IMMEDIATE MODIFIED timed_statistics
1 row selected.
```

5.6.4. Quản lý session

Restrict session

Restrict session cần thiết khi bảo trì cơ sở dữ liệu, import, export và sửa đổi cấu trúc của database.

Ta có thể đặt chế độ cho restrict session cho database thông qua lệnh:

```
ALTER SYSTEM {ENABLE|DISABLE}RESTRICTED SESSION
```

Với:

```
ENABLE RESTRICTED
                    chỉ cho phép các users  có quyền
                    RESTRICTED SESSION  truy nhập
DISABLE RESTRICTED SESSION
                    cho phép tất cả các users truy nhập vào
                    database
```

Kết thúc session

Ta có thể kết thúc (Terminate) các session của một Instance đã ở trong chế độ restrict, trước khi thực hiện các thao tác quản trị.

Cú pháp:

```
ALTER SYSTEM KILL SESSION 'integer1,integer2'
```

Với:

```
KILL SESSION     tên session cần kết thúc
integer1         giá trị của cột SID trong view v$session
integer2         giá trị của cột SERIAL# trong view v$session
```

Chú ý: hai giá trị integer1 và integer2 dùng để xác định session

Với lệnh `KILL SESSION` background process `PMON` sẽ thực hiện các công việc sau:

- Rollback transaction hiện thời của user
- Giải phóng tất cả các lock trên các table thực hiện bởi user đó
- Giải phóng các tài nguyên sử dụng bởi user

5.6.5. Trace file và ALERT file

Trace file lưu trữ các thao tác bởi background process. Các thông tin về lỗi trong hệ thống sẽ được lưu vào đây. Điều này là rất hữu ích khi thực hiện dò tìm và khắc phục lỗi xảy ra trong hệ thống.

Trong khi chạy Oracle Instance, tất cả các message phát ra đối với hệ thống đều được lưu vào Alert file. Trong quá trình khởi động database, Oracle sẽ tự tạo ra Alert file nếu nó chưa tồn tại.

Trong trường hợp có lỗi xảy ra, các background process sẽ thực hiện ghi lại các thông tin dump vào trace file.

Ta có thể đặt lại chế độ ghi lỗi ra trace file thông qua lệnh:

```
SQL>ALTER SESSION SET sql_trace=TRUE;
```

Đường dẫn tới các trace file và Alert có thể được chỉ ra bởi các tham số:

`BACKGROUND_DUMP_DEST`
: Xác định nơi đặt của các trace file và ALERT.

`USER_DUMP_DEST`
: Xác định nơi tạo các trace files.

`MAX_DUMP_FILE_SIZE`
: Số lượng block của hệ điều hành quy định kích thước của trace files.

Chương 6. DATA DICTIONARY, VIEWS VÀ PACKAGES

6.1.DATA DICTIONARY VÀ VIEWS

6.1.1. Data Dictionary

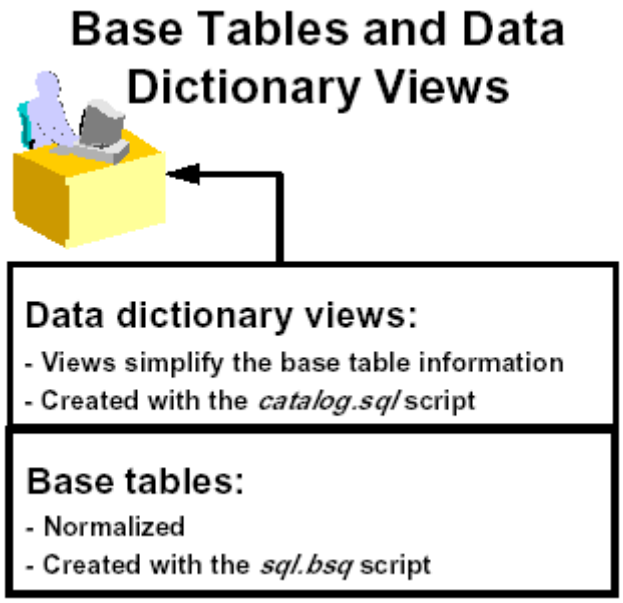
Data dictionary hay từ điển dữ liệu hệ thống là phần rất quan trọng trong Oracle database. Đó là một tập hợp các table và các view sử dụng cho việc tham chiếu đến các thông tin liên quan tới database. Data dictionary được tạo bởi file script *sql.bsq* trong quá trình tạo database.

Data dictionary bao gồm các thông tin trung tâm của Oracle server.

Data dictionary được Oracle server tự động cập nhật mỗi khi thực hiện lệnh định nghĩa dữ liệu (Data Definition Language – DDL).

Data dictionary đặt trong tablespace SYSTEM do User SYS quản lý. Data dictionary bao gồm hai loại sau:

- Base tabes
- Data dictionary Views



Hình vẽ 20. Dictionarytrong database

Base tables

Thông tin trong data dictionary được xác định từ các thông tin có trong các base tables (bảng cơ sở). Nội dung của các bảng này do Oracle server cập nhật. User thuộc database hầu như không thể cập nhật các thông tin này do chúng là các thông tin đã được chuẩn hoá và được mã hoá. Ví dụ: ta chỉ có thể truy xuất tới các thông tin có trong bảng `IND$` để biết được các thông tin về các indexes đã được định nghĩa trong database, hoặc lấy các thông tin trong bảng `OBJ$` để biết được các objects đã được định nghĩa trong database.

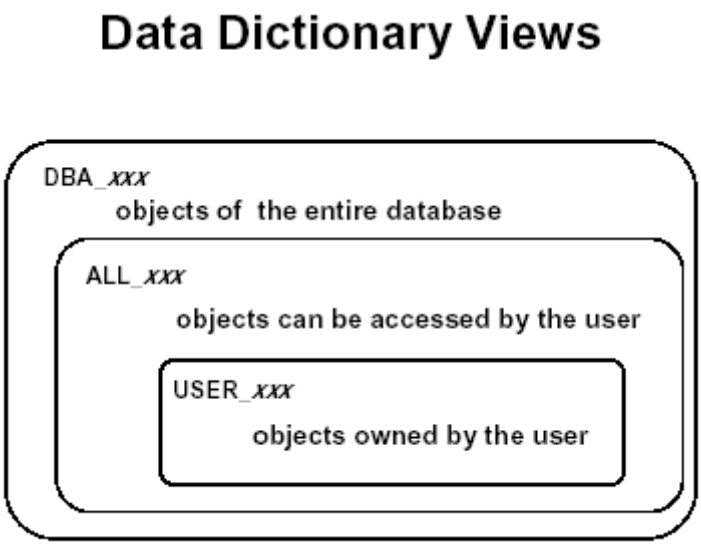
Ta không thể sử dụng các câu lệnh thao tác dữ liệu như INSERT, UPDATE, hay DELETE để thay đổi nội dung thông tin trong các bảng cơ sở một cách trực tiếp ngoại trừ bảng AUD$ (Xem thêm phần kiểm tra - Auditing).

Data Dictionary Views

Data dictionary views được tạo ra bởi các câu lệnh có trong file script *catalog.sql*. Các views này giải mã và tổng hợp các thông tin có trong các base tables. Để dễ dàng truy xuất các thông tin này, các data dictionary thường được tạo các synonyms tương ứng.

Phần lớn các thông tin hệ thống được User lấy về từ các data dictionary views hơn là lấy trực tiếp từ các base tables.

6.1.2. Data Dictionary views



Hình vẽ 21. Dictionary views

Data dictionary views được phân ra làm ba loại chứa các thông tin tương tự nhau nhưng ở các mức độ khác nhau. Các loại data dictionary views này được phân biệt bởi các tiếp đầu ngữ khác nhau.

Tiếp đầu ngữ USER

Các views có tiếp đầu ngữ USER chứa thông tin về các objects do User hiện thời sở hữu. Ví dụ: USER_TABLES sẽ chứa thông tin về các bảng dữ liệu của User hiện thời.

Tiếp đầu ngữ ALL

Các views có tiếp đầu ngữ ALL chứa thông tin về các objects có thể truy cập bởi User hiện thời, bao gồm cả các đối tượng do User đó sở hữu và cả các đối tượng khác mà User được

gán quyền truy nhập. Ví dụ: ALL_TABLES sẽ chứa thông tin về các bảng dữ liệu mà User hiện thời có thể truy nhập.

Tiếp đầu ngữ DBA

Các views có tiếp đầu ngữ DBA chứa thông tin về các objects có trong database. Các views này là cần thiết cho quản trị viên database. Một User bất kỳ cũng có thể xem được thông tin trong các views DBA nếu user đó được cấp quyền SELECT ANY TABLE.

Phân loại một số loại views

Tên View	Diễn giải
DICTIONARY DICT_COLUMNS	Thông tin chung
DBA_TABLES DBA_OBJECTS DBA_LOBS DBA_TAB_COLUMNS DBA_CONSTRAINTS	Thông tin liên quan tới các đối tượng của User như: table, Column, Constraint,...
DBA_USERS DBA_SYS_PRIVS DBA_ROLES	Thông tin về mức quyền của User

Tên View	Diễn giải
DBA_EXTENTS DBA_FREE_SPACE DBA_SEGMENTS	Tình hình cấp phát không gian cho các đối tượng trong database.
DBA_ROLLBACK_SEGS DBA_DATA_FILES DBA_TABLESPACES	Thông tin về cấu trúc database
DBA_AUDIT_TRAIL DBA_AUDIT_OBJECTS DBA_AUDIT_OBJ_OPTS	Các thông tin kiểm tra

Ví dụ: Để lấy các thông tin chung trong từ điển dữ liệu, ta có thể truy vấn trong Các views DICTIONARY hoặc DICT_COLUMNS.

```
SVRMGR>SELECT *
2> FROM dictionary
3> WHERE table_name LIKE '%TABLE%';
 TABLE_NAME            COMMENTS
```

```
-------------------  ---------------------------------------
ALL_ALL_TABLES       Description of all object and relational
                     tables accessible to the user
ALL_NESTED_TABLES    Description of nested tables in tables
                     accessible to the user
ALL_OBJECT_TABLES    Description of all object tables
                     accessible to the user
ALL_PART_TABLES
ALL_TABLES           Description of relational tables
                     accessible to the user
ALL_UPDATABLE_COLUMNS Descriptionofallupdatablecolumns
DBA_ALL_TABLES       Description of all object and relational
                     tables in the database
DBA_NESTED_TABLES    Description of nested tables contained
                     in all tables
DBA_OBJECT_TABLES    Description of all object tables in the
                     database
...
```

Xây dựng dictionary views

Sau khi tạo database, ta truy cập vào database theo user: SYS và chạy các scripts: *catalog.sql* và catprog.sql để tạo các dictionary views. Thông thường, các scripts này nằm trong thư mục: %ORACLE_HOME%\RDBMS80\ADMIN

Catalog.sql

CATALOG.SQL script dùng để tạo các view dựa trên các base tables (bảng cơ sở) của database. Các view này sẽ được tạo synonym (một tên khác với tên của objects được dùng để truy cập objects) tương ứng để dễ dàng truy vấn các dữ liệu từ đó hơn. Scripts này còn gọi tới các scripts khác để tạo các views và các đối tượng khác phục vụ cho các tiện ích Server Manager, cho việc kiểm tra, cho các tiện ích Export và Import dữ liệu,... Scripts STANDARD.SQL được gọi đến trong đó để tạo các môi trường PL/SQL tuân theo chuẩn.

Ví dụ: Scripts tạo mẫu giao tiếp cho 01 hàm built-in có tên BITAND:

```
function BITAND (LEFT binary_integer, RIGHT binary_integer)
return binary_integer;
```

Catproc.sql

CATPROC.SQL script dùng để tạo các hàm PL/SQL, các packages PL/SQL sử dụng trong RDBMS. Ngoài ra, CATPROC.SQL script còn tạo Các views mở rộng khác.

6.1.3. Sripts quản trị

Các scripts quản trị được đặt trong thư mục: %ORACLE_HOME%\RDBMS80\ADMIN

Các scripts này được phân nhóm và đặt trong từng file riêng biệt.

Các quy định về tên có trong Script quản trị

Quy ước	Diễn giải
Cat*.sql	Các thông tin Catalog và từ điển dữ liệu
Dbms*.sql	Phần khai báo (specification) của các packages trong database
Prvt*.plb	Phần thân cua packages đã được mã hoá và đóng gói
Utl*.sql	Các views và table tiện ích trong database

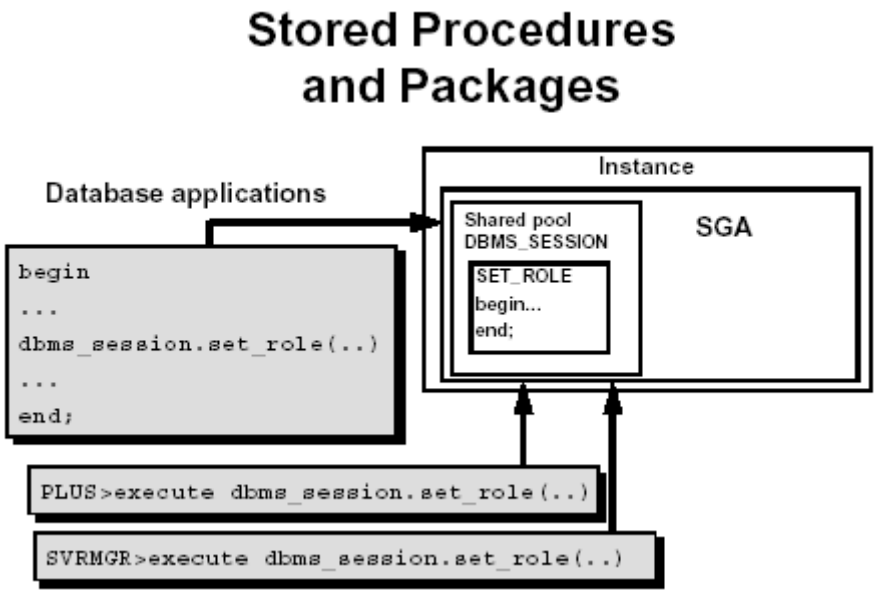
6.2.STORED PROCEDURES VÀ CÁC PACKAGES CHUẨN

6.2.1. Giới thiệu chung

Stored procedures và các packages là các đối tượng trong database, đó là tập hợp các đoạn mã lệnh PL/SQL để thực hiện một chức năng nào đó.

Stored procedures bao gồm cả các procedures (thủ tục), functions (hàm) và các packages được viết gộp thành một program unit (đơn vị chương trình).

Stored procedures có thể được tạo và huỷ bởi các lệnh CREATE và DROP



Hình vẽ 22. Stored procedures và các Packages chuẩn

Lợi ích của Stored procedures

- Các Stored procedures được nạp vào shared pool, do đó có thể giảm bớt việc truy xuất đĩa khi thực hiện thủ tục.

- Đảm bảo an toàn cho dữ liệu, ngăn không cho các users truy cập trực tiếp vào dữ liệu mà phải thông qua các thủ tục và hàm giao tiếp đã được cung cấp.
- Cho phép nhiều users có thể cùng sử dụng các bản sao của Stored procedures để thực hiện.

6.2.2. Stored procedures

Stored procedures là các functions hay procedures được tạo lập và lưu ngay trong dictionary giống như một schema object. Đây là tập hợp các câu lệnh SQL và PL/SQL. Sau khi Stored procedures được biên dịch, nó sẽ được gán tên và có thể thực hiện trực tiếp mà không cần phải biên dịch lại thêm bất cứ một lần nào nữa.

Sử dụng Stored procedures, ta có thể nạp trực tiếp vào ngay biểu thức thuộc câu lệnh SQL giống như là các hàm built-in có sẵn của Oracle như UPPER hay SUBSTR.

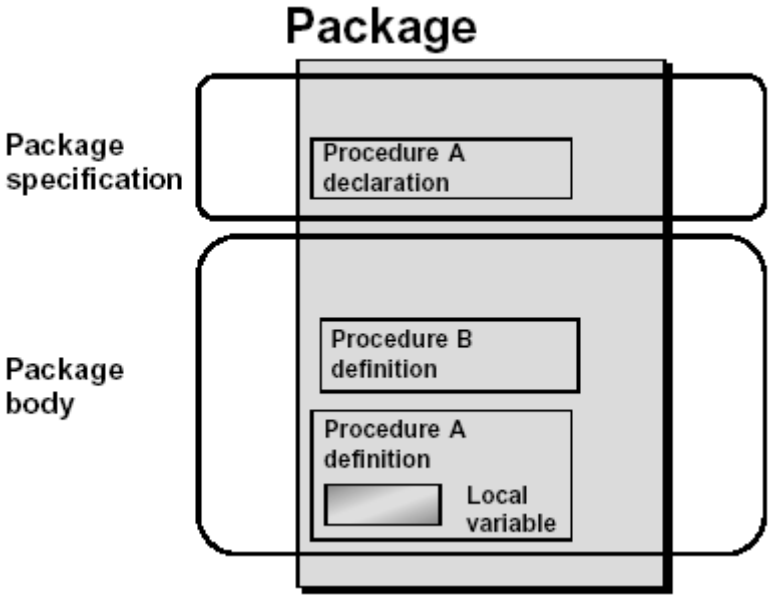
Các functions và procedures cho phép sử dụng tham số dưới dạng tham số vào (`IN`) và tham số ra (`OUT`) hoặc cũng có thể sử dụng tham số vừa vào vừa ra (`IN OUT`). Theo mặc định, các tham số được xác định ở chế độ vào `IN`.

6.2.3. Packages chuẩn

Một packages thông thường gồm hai phần: specification (phần đặc tả hay còn gọi là phần khai báo) và body (phần thân). Chúng được lưu riêng biệt trong cùng một database.

- Phần specification là phần giao tiếp với các ứng dụng. Phần này chứa các lời khai báo, các kiểu, biến, hằng, exceptions, cursors, và các khai báo hàm để sử dụng.
- Phần body là phần cài đặt cụ thể (implementation) của các khai báo trong phần specification.

Chức năng của packages cũng tương tự như Stored procedures. Một khi packages được biên dịch, packages đó có thể được sử dụng bởi nhiều ứng dụng khác nhau. Tuy nhiên, có một lợi ích lớn nhất khi sử dụng packages là ngay lần đầu tiên gọi đến packages, toàn bộ packages sẽ được nạp vào trong bộ nhớ.



Hình vẽ 23. Packages trong cơ sở dữ liệu

6.2.4. Giới thiệu một số packages chuẩn do Oracle cung cấp

Oracle cung cấp một số packages chuẩn, ngay sau khi tạo database:

- `DBMS_LOB`: cung cấp các thủ tục cho phép làm việc trên kiểu dữ liệu `BLOB` và `CLOB`, được định nghĩa trong file script catprog.sql.
- `DBMS_SESION`: cung cấp các câu lệnh SQL liên quan đến session như `ALTER SESSION, SET ROLE`, ... packages này được định nghĩa trong file *dbmsutil.sql* và *prvtutil.sql*
- `DBMS_UTILITY`: chứa các thủ tục tiện ích, được đặt trong file *dbmsutil.sql* và *prvtutil.sql*
- `DBMS_SPACE`: cung cấp các thông tin về khoảng trống của segment.
- `DBMS_ROWID`: cung cấp các thông tin về `ROWID`
- `DBMS_SHARE_POOL`: lưu trữ và huỷ bỏ các thông tin có trong share pool.

Packages	Thủ tục trong packages	Diễn giải
DBMS_SESSION	SET_ROLE	Kích hoạt việc thực hiện Roles của user
	SET_SQL_TRACE	Thiết lập chế độ dò tìm thực hiện lệnh
	SET_NLS	Chọn chuẩn hỗ trợ ngôn ngữ
	CLOSE_DATABASE_LINK	Đóng database link.
	UNIQUE_SESSION_ID	Trả về mã duy nhất cả các session hiên đang connect tới database.

	IS_ROLE_ENABLED	Xác đinh xem role có được kích hoạt trong session không.
	IS_SESSION_ALIVE	Xác định xem session có còn hay không.
	SET_CLOSE_CACHED_OPEN_CURSORS	Bật hoặc tắt close_cached_open_cursors
	FREE_UNUSED_USER_MEMORY	Giải phóng vùng bộ nhớ không còn sử dụng
DBMS_UTILITY	ANALYZE_SCHEMA	Phân tích các objects trong schema như: functions, procedures, packages, triggers,..
	COMPILE_SCHEMA	Biên dịch các objects trong schema
	DB_VERSION	Xác định phiên bản của database
DBMS_ROWID	ROWID_INFO	Thông tin về dòng dữ liệu
DBMS_SPACE	UNUSED_SPACE	Vùng không gian không sử dụng
	FREE_BLOCKS	Các blocks rỗi
DBMS_SHARED_POOL	KEEP	Lưu trữ các object trong shared pool
	UNKEEP	Thôi lưu giữ các object
	SIZES	Kích thước bộ nhớ trong shared pool
DBMS_SQL	OPEN_CURSOR	Trả về số hiệu cursor (ID number)
	PARSE	Phân tích câu lệnh
	BIND_VARIABLE	Binds một giá trị biến.
	BIND_ARRAY	Binds một giá trị biến mảng.
	EXECUTE Function	Executes a given cursor.
	EXECUTE_AND_FETCH	Thực hiện lệnh và lấy về các dòng dữ liệu.
	FETCH_ROWS	Lấy về các dòng dữ liệu của một cursor.
	COLUMN_VALUE	Lấy về dữ liệu của cột
	IS_OPEN	Xác định Cursor đã mở hay chưa.
	CLOSE_CURSOR	Đóng cursor và giải phóng bộ nhớ.

	LAST_ERROR_POSITION	Trả về lỗi thực hiện câu lệnh SQL
	LAST_ROW_COUNT	Trả về số lượng dòng dữ liệu lấy về
	LAST_ROW_ID	Trả về mã dòng dữ liệu xử lý ROWID
	LAST_SQL_FUNCTION_CODE	Trả về mã hàm SQL

6.2.5. Package DBMS_METADATA

Một PL/SQL package mới, DBMS_METADATA, được đưa vào Oracle 9i cho phép ta lấy được các siêu dữ liệu (metadata) – Các thông tin tổng hợp về các schema object.

- DBMS_METADATA là package mới bổ sung, nó cho phép thực hiện các thao tác DDL trên objects trong database.
- Package này làm việc được với các tables, indexes, views, packages, functions, procedures, triggers, synonyms, và types.
- DBMS_METADATA có các hàm cơ bản:
  - DBMS_METADATA.GET_DDL(object_type, name, schema)
  - DBMS_METADATA.GET_XML(object_type, name, schema)

Ví dụ:

```
SELECT DBMS_METADATA.GET_DDL('TABLE', 'EMP', 'SCOTT') from dual;
  CREATE TABLE "SCOTT"."EMP"
   (    "EMPNO" NUMBER(4,0),
        "ENAME" VARCHAR2(10),
        "JOB" VARCHAR2(9),
        "MGR" NUMBER(4,0),
        "HIREDATE" DATE,
        "SAL" NUMBER(7,2),
        "COMM" NUMBER(7,2),
        "DEPTNO" NUMBER(2,0),
         CONSTRAINT "PK_EMP" PRIMARY KEY ("EMPNO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645 PCTINCREASE 0
  FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE
"USERS"  ENABLE,
         CONSTRAINT "FK_DEPTNO" FOREIGN KEY ("DEPTNO")
  REFERENCES "SCOTT"."DEPT" ("DEPTNO") ENABLE NOVALIDATE
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645 PCTINCREASE 0
  FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE
"USERS"
```

```
SELECT DBMS_METADATA.GET_XML('TABLE', 'EMP', 'SCOTT') from dual;
<?xml version="1.0"?>
<ROWSET>
 <ROW>
  <TABLE_T>
   <VERS_MAJOR>1</VERS_MAJOR>
   <VERS_MINOR>0</VERS_MINOR>
   <OBJ_NUM>5543</OBJ_NUM>
   <SCHEMA_OBJ>
    <OBJ_NUM>5543</OBJ_NUM>
    <DATAOBJ_NUM>5543</DATAOBJ_NUM>
    <OWNER_NUM>25</OWNER_NUM>
    <OWNER_NAME>SCOTT</OWNER_NAME>
    <NAME>EMP</NAME>
    <NAMESPACE>1</NAMESPACE>
    <MINEXTS>1</MINEXTS>
    <MAXEXTS>2147483645</MAXEXTS>
    <EXTSIZE>128</EXTSIZE>
    <EXTPCT>0</EXTPCT>
    …
```

6.2.6. Package dbms_redefinition

Package này cung cấp 05 thủ tục cho phép chỉnh sửa các objects online .

- CAN_REDEF_TABLE
- START_REDEF_TABLE
- FINISH_REDEF_TABLE
- ABORT_REDEF_TABLE
- SYNC_INTERIM_TABLE

6.3.THÔNG TIN VỀ CÁC STORED PROCEDURES

Khi lưu trữ các Stored procedures hay packages, Oracle sẽ tự động lưu lại trạng thái của nó là VALID hay INVALID.

- VALID: Stored procedures hay packages có trạng thái là VALID nếu nó đã được biên dịch và không có lỗi xảy ra. Khi này, nó sẵn sàng cho việc sử dụng.
- INVALID: là trạng thái ngược lại với trạng thái VALID. Stored procedures hay Packages vẫn còn lỗi khi biên dịch. Khi này, ta chưa thể sử dụng được ngay.

Cú pháp lệnh yêu cầu biên dịch lại Stored procedures:

```
ALTER PROCEDURE [schema_name].<procedure_name> COMPILE [DEBUG];
```

Với:

schema_name	tên schema chứa procedure cần biên dịch lại
procedure_name	tên của procedure biên dịch lại.
COMPILE	chỉ định yêu cầu biên dịch lại procedure
DEBUG	chỉ định chương trình biên dịch mã lệnh PL/SQl của procedure sẽ sinh mã lệnh phù hợp để chương trình PL/SQL debugger có thể đọc. User có thể sử dụng chương trình này để dò tìm và gỡ lỗi cho procedure.

Ví dụ:

```
ALTER PROCEDURE henry.close_acct  COMPILE;
```

Tương tự như đối với procedure, cú pháp lệnh yêu cầu biên dịch lại Stored function có dạng:

```
ALTER FUNCTION [schema_name].<function_name> COMPILE [DEBUG];
```

Ví dụ:

```
ALTER FUNCTION merriweather.get_bal COMPILE;
```

Đối với package, lệnh yêu cầu biên dịch lại cũng tương tự nhưng có thêm một bổ sung là user phải khai báo rõ từng phần của package sẽ được biên dịch lại.

Cú pháp:

```
ALTER PACKAGE [schema_name].<package_name>
      COMPILE [DEBUG] <PACKAGE | SPECIFICATION | BODY>;
```

Các khai báo bổ sung cho phép user yêu cầu biên dịch lại phần SPECIFICATION hay phần BODY hoặc là biên dịch lại cả hai phần trên.

Ví dụ:

```
ALTER PACKAGE blair.accounting
         COMPILE PACKAGE;

Hoặc:
ALTER PACKAGE blair.accounting
         COMPILE BODY;
```

Để xác định được trạng thái của các Stored procedures, ta có thể thực hiện truy vấn dựa trên dictionary DBA_OBJECTS.

```
SVRMGR> SELECT object_name, object_type, status
     2> FROM dba_objects WHERE object_name like 'DBMS_%'
OBJECT_NAME           OBJECT_TYPE     STATUS
--------------------- --------------- --------
DBMS_ALERT            PACKAGES        VALID
DBMS_ALERT            PACKAGES        BODY VALID
DBMS_ALERT_INFO       TABLE           VALID
DBMS_APPLICATION_INF PACKAGES         VALID
DBMS_APPLICATION_INF PACKAGES         BODY VALID
```

```
DBMS_AQ                PACKAGES        VALID
DBMS_AQ                PACKAGES        BODY VALID
...
```

Hoặc ta cũng có thể sử dụng lệnh DESCRIBE để lấy thông tin

```
SVRMGR> DESCRIBE dbms_session.set_role
procedure SET_ROLE (ROLE_CMD VARCHAR2);
```

```
svrmgr> describe dbms_session
packages dbms_session is
------------
-- OVERVIEW
-- This packages provides access to SQL "alter session"
-- statements, and other session information from, stored
-- procedures.
----------------------------
-- PROCEDURES AND FUNCTIONS
procedure set_role(role_cmd varchar2);
-- Equivalent to SQL "SET ROLE ...".
-- Input arguments:
-- role_cmd
-- This text is appended to "set role " and then executed as
-- SQL.
procedure set_sql_trace(sql_trace boolean);
-- Equivalent to SQL "ALTER SESSION SET SQL_TRACE ..."
-- Input arguments:
-- sql_trace
-- TRUE or FALSE. Turns tracing on or off.
procedure set_nls(param varchar2, value varchar2);
```

Stored procedures hay Packages nhận trạng thái INVALID khi các câu lệnh trong Stored procedures hay Packages bị lỗi.

Chương 7. QUẢN TRỊ CONTROL FILES

7.1.CONTROL FILES

7.1.1. Giới thiệu control file

Control file là file thông tin dạng nhị phân được sử dụng cho việc khởi tạo và vận hành database một cách hiệu quả.

Mỗi khi instance được MOUNT (gắn) với một Oracle database, các thông tin trong control file sẽ được đọc ra, từ đó xác định các data files và các online redo log files.

Control file được cập nhật liên tục vào database trong suốt quá trình sử dụng và nó luôn ở trạng thái sẵn sàng (available) mỗi khi database được OPEN (mở) hay được MOUNT (gắn) với instance.

Control file cung cấp các thông tin một cách đồng nhất trong database được sử dụng trong quá trình khôi phục (recovery).

Mỗi control file tại một thời điểm chỉ phục vụ cho một database. Khi đã có một database sử dụng control file thì các database khác sẽ không thể truy cập tới control file đó nữa.

7.1.2. Cách thức đặt tên control file

Tên control file được xác định trong tham số CONTROL_FILES của parameter file. Tên của các control files được đặt phân cách bởi dấu phẩy (,). Instance phục vụ database sẽ mở các control file và lấy các thông tin từ đó để có thể điều khiển hoạt động của database. Trong quá trình hoạt động, Instance cũng sẽ ghi lại các tình trạng của database.

Để đảm bảo an toàn, một database cần ít nhất 02 control files và được đặt tại hai chỗ khác nhau. Các control files nên được đặt tên khác nhau sao cho có thể phân biệt dễ dàng.

Tên của Control files nên được đặt kèm với tên của database cho dễ nhớ, như sau:

```
CTL<n><database_name>.ORA
```

Với:

```
n               là số thứ tự của control file
database_name   tên của database
```

Trong parameter file, các tên của control files được đặt phân cách nhau bởi các dấu phẩy.

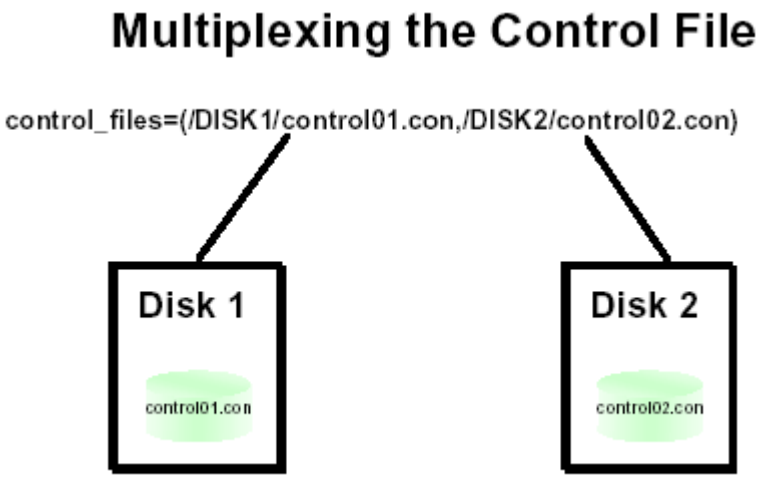
Ví dụ:

```
control_files =  (“C:\ORANT\DATABASE\CTL1KTKB.ORA”,
                      ”C:\ORANT\DATABASE\CTL2KTKB.ORA”)
```

7.1.3. Kết hợp nhiều control files

Khi tạo database, ta có thể sử dụng cùng lúc nhiều control files thông qua việc chỉ rõ tên các control files trong tham số khởi tạo CONTROL_FILES. Oracle server tạo và cập nhật tất cả danh sách các file liên quan mỗi khi tạo database.

Oracle khuyến cáo sử dụng ít nhất 02 control files. Các control files nên được đặt riêng biệt trên các ổ đĩa khác nhau đề phòng sự cố. Nếu một control file bị hỏng, ta có thể sao chép lại file này rồi khởi động lại instance.



Hình vẽ 24. Kết hợp sử dụng nhiều control file

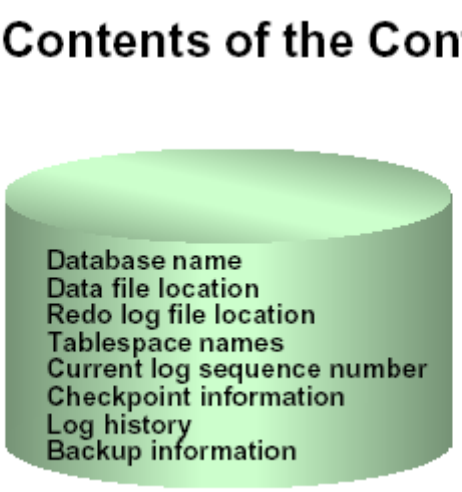
Để thêm mới một control file hoặc thay đổi số lượng cũng như nơi đặt các control file, ta thực hiện theo các bước sau:

1. Shutdown database.
2. Sử dụng lệnh của hệ điều hành để sao chép thêm một bản sao của control file và nên lưu trữ trên một thiết bị khác.
3. Sửa đổi hoặc thêm mới tham số `CONTROL_FILES` và tên (có đường dẫn) tương ứng với các control files.
4. Khởi động lại database.

7.1.4. Nội dung của control file

Các thông tin chứa trong control file bao gồm:

- Tên database và các định danh (identifications)
- Tên và nơi chứa các data files, các redo log files
- Tên các tablespaces trong database
- Nhãn thời gian tương ứng lúc tạo database
- Giá trị số hiệu của log sequence hiện thời
- Thông tin về checkpoint
- Các thông tin lịch sử (log history)
- Các thông tin sao lưu của tiện ích Recovery Manager



Hình vẽ 25. Nội dung control file

Control file có thể được chia làm hai loại chính:

- Có thể tái sử dụng (reused)
- Không thể tái sử dụng (unreused)

7.1.5. Các tham số ảnh hưởng tới kích thước của control file

Có một số tham số hệ thống liên quan tới kích thước của control file

- MAXLOGFILES
- MAXLOGMEMBERS
- MAXLOGHISTORY
- MAXDATAFILES
- MAXINSTANCES

Các control files được xác định tự động dựa theo các tham số khởi tạo tại thời điểm tạo lập database:

```
CONTROL_FILES = (“C:\ORANT\DATABASE\CTL1KTKB.ORA”,
                 ”C:\ORANT\DATABASE\CTL2KTKB.ORA”)
```

Tên file kèm theo đường dẫn được đặt luôn trong tham số tạo database.

Các tham số được chỉ ra trong database có ảnh hưởng tới control file. Quản trị viên database có thể tạo lại các control file hay thay đổi các tham số trong database để có thể tăng, giảm kích thước của control file.

Việc tạo mới control file đòi hỏi phải thay đổi kích thước của control file. Control file lưu trữ các thông tin cần thiết cho Recovery Manager. Vì thế, khi sử dụng Recovery Manager những phần không tái sử dụng được trong control file có thể được mở rộng dựa theo số lượng các thành phần.

7.2.QUẢN TRỊ CONTROL FILE

7.2.1. Tạo mới control file

Việc tạo mới control files đối với database đôi khi là cần thiết. Ta hãy xét các tình huống:

- Tất cả các control files của database hiện thời đều bị lỗi và ta không có bản backup của chúng.
- Ta muốn thay đổi một hay nhiều tham số được thiết lập đối với database mà các tham số này được chỉ ra ngay từ câu lệnh CREATE DATABASE như tên *database*, MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, và MAXINSTANCES.

Ví dụ, ta muốn đổi tên database để khỏi xảy ra xung đột với một database đang có trong hệ thống nhưng trùng tên.

Ta có thể tạo mới control file cho một database thông qua câu lệnh SQL.

Cú pháp:

```
CREATE CONTROLFILE [REUSE]
    [SET] DATABASE database
    LOGFILE [GROUP integer] filespec [, [GROUP integer]
           filespec] ...
    {RESETLOGS | NORESETLOGS}
    DATAFILE filespec [, filespec] ...
    [MAXLOGFILES integer]

    [MAXLOGMEMBERS integer]
    [MAXLOGHISTORY integer]
    [MAXDATAFILES integer]
    [MAXINSTANCES integer]
    [ARCHIVELOG | NOARCHIVELOG]
```

Với:

REUSE	Cho biết CONTROL_FILES có thể được tái sử dụng, ta không cần quan tâm tới các tham số thuộc loại tuỳ chọn.
SET DATABASE	Thay đổi tên của database. Lưu ý: <Tên>
DATABASE	Tên của database.
LOGFILE	danh sách tên của các redo log file groups
MAXLOGFILES	Số lượng tối đa các redo log file groups
MAXLOGMEMBERS	Số lượng tối đa các members trong một redo
MAXLOGHISTORY	Số lượng tối đa các archived redo log file groups
MAXDATAFILES	Số lượng tối đa các datafiles
MAXINSTANCES	Số lượng tối đa các instances có thể kết nối tới database.
ARCHIVELOG	Thiết lập chế độ archiving lưu trữ các redo log files

Ví dụ:

```
CREATE CONTROLFILE
       SET DATABASE prod
       LOGFILE GROUP 1 ('logfile1A', 'logfile1B') SIZE 50K,
```

```
GROUP 2 ('logfile2A', 'logfile2B') SIZE 50K
NORESETLOGS
DATAFILE 'datafile1' SIZE 3M, 'datafile2' SIZE 5M
MAXLOGFILES 50
MAXLOGMEMBERS 3
MAXDATAFILES 200
MAXINSTANCES 6
ARCHIVELOG;
```

7.2.2. Tạo mới control file cho một database đã có sẵn

Việc tạo mới control file được thực hiện theo các bước sau:

1. Thiết lập danh sách các datafiles và online redo log files sử dụng trong database. Trong trường hợp backup database, ta có thể dễ dàng xác định được danh sách các file này dựa vào thông tin trong dictionary view: V$CONTROLFILE, V$DATAFILE, V$LOGFILE. Trong trường hợp database bị lỗi, quản trị viên database cần cố gắng xác định đầy đủ các datafiles và online redo log files. Nếu thiếu bất kỳ một trong số các file trên thì tablespace SYSTEM sẽ không thể khôi phục lại được và do đó ta không thể khôi phục lại được database.
2. Shut down (tắt) database nếu nó đang được mở. Thực hiện shut down ở chế độ normal. Trong trường hợp không thể tắt normal được thì hãy tắt database theo chế độ IMMEDIATE hoặc ABORT.
3. Sao lưu (Backup) tất cả các datafiles và online redo log files của database.
4. Startup instance trở lại ở chế độ nomount.
5. Tạo mới control file thông qua lệnh tạo CONTROL FILES. Khi tạo mới control file, sử dụng tuỳ chọn RESETLOGS nếu database bị mất bất kỳ một nào online redo log groups. Trong trường hợp này ta cần khôi phục lại các redo logs bị mất. Ngược lại, ta sử dụng tuỳ chọn NORESETLOGS.
6. Sao lưu control file mới tạo.
7. Sửa đổi các tham số trong parameter file mà có sử dụng đến trong các control files bao gồm tham số CONTROL_FILES và DB_NAME.
8. Thực hiện khôi phục database nếu cần. Ta sẽ bỏ qua bước này trong trường hợp không cần phải khôi phục database. Nếu control file mới tạo có sử dụng tuỳ chọn NORESETLOGS, thì ta có thể khôi phục lại toàn bộ database. Trong trường hợp tuỳ chọn sử dụng là RESETLOGS, ta cần chỉ ra thêm một tuỳ chọn nữa là USING BACKUP CONTROL FILE. Thủ tục này sẽ thực hiện khôi phục lại các online hoặc archived redo logs hoặc datafiles.
9. Open database với control file vừa tạo. Nếu không thực hiện recovery thì có thể open database ở chế độ normally.
10. Nếu có sử dụng RESETLOGS trong lúc tạo control file, thì cần sử dụng thêm câu lệnh ALTER DATABASE , với tuỳ chọn RESETLOGS.

7.2.3. Một số lỗi đối với các Control Files

Sau khi thực hiện lệnh CREATE CONTROLFILE, ta có thể ta gặp một số lỗi cơ bản sau:

Thiếu file

Sau khi tạo một control file và sử dụng nó để mở database, kiểm tra alert log để biết liệu Oracle có xác định được có thông tin gì không đồng nhất giữa data dictionary và control file hay không? Ví dụ như datafile có kèm theo cả data dictionary nhưng không có danh sách các data dictionary đi kèm.

Nếu một datafile đã tồn tại trong data dictionary nhưng chưa có trong control file mới tạo, Oracle sẽ tạo một placeholder entry trong control file với tên là MISSING*nnnn* (trong đó *nnnn* là một con số viết dưới dạng thập phân).

Ta xét hai trường hợp có thể xảy ra như sau:

- Sử dụng tuỳ chọn RESETLOGS trong câu lệnh CREATE CONTROLFILE sẽ cho phép mở database mà không cần tới tuỳ chọn RESETLOGS. Điều này chỉ có thể xảy ra nếu tất cả các online redo logs đang trong tình trạng sẵn sàng.
- Sử dụng tuỳ chọn RESETLOGS trong câu lệnh CREATE CONTROLFILE để bắt buộc phải mở database cùng với tuỳ chọn RESETLOGS, datafile tương ứng với MISSING*nnnn* ở chế độ chỉ đọc hay OFFLINE.

Khi mở database có sử dụng tuỳ chọn RESETLOGS, và MISSING*nnnn* tương ứng với datafile không ở chế độ chỉ đọc hay offline, ta sẽ không thể truy xuất vào datafile đó. Trong trường hợp này, tablespace chứa datafile cần được huỷ bỏ (DROP).

Xử lý lỗi xảy ra đối với lệnh CREATE CONTROLFILE

Oracle gửi trả về mã lỗi(các mã lỗi hay xảy ra là ORA-01173, ORA-01176, ORA-01177, ORA-01215 hoặc ORA-01216) khi ta cố gắng thực hiện mount và open database sau khi tạo mới một control file. Tình huống hay xảy ra nhất là trong câu lệnh CREATE CONTROLFILE mà ta quên một file hoặc có đưa vào tên file nhưng nó vẫn chưa có trong danh sách. Trong trường hợp này, ta cần phải khôi phục (RESTORE) lại các files đã được backup ở bước 3 (phía trên) và lặp lại các thủ tục ở bước 4 (phía trên) lưu ý sử dụng đúng tên các files.

7.2.4. Huỷ bỏ Control Files

Ta có thể huỷ bỏ các control files khỏi database. Ví dụ, ta thực hiện việc này khi đường dẫn tới các control file không còn phù hợp nữa. Có một điều lưu ý là tại bất kỳ thời điểm nào database cũng cần phải có ít nhất là 2 control files.

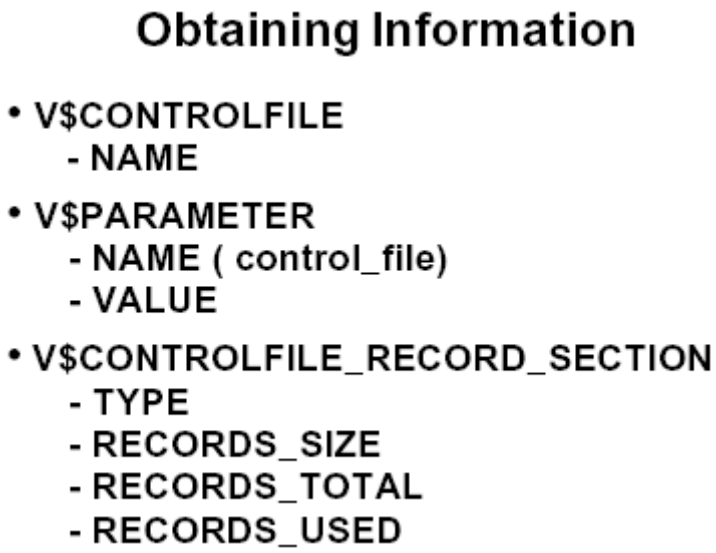
Các bước thực hiện

1. Shut down (tắt) database.
2. Sửa lại tham số CONTROL_FILES trong parameter file, xoá tên control file cũ và thay vào đó tên control file mới.

3. Restart (khởi động lại) database.

7.3.THÔNG TIN TRẠNG THÁI CỦA CONTROL FILES

Ta có thể xem được các thông tin về control file dựa trên dictionary views có trong database.



Ví dụ:

```
SVRMGR> SELECT name
2>FROM v$controlfile;
NAME
------------------------
/DISK1/control01.con
/DISK2/control02.con
2 rows selected.

SVRMGR> SELECT value
2>FROM v$parameter WHERE name ='control_files';
VALUE
------------------------
/DISK1/control01.con
```

```
/DISK2/control02.con
2 rows selected.
```

V$CONTROLFILE_RECORD_SECTION chứa các thông tin về các section.

Ví dụ:

```
SVRMGR>SELECT type, record_size, records_total, records_used
2> FROM v$controlfile_record_section
3> WHERE type='DATAFILE';
TYPE                 RECORD_SIZ  RECORDS_TO RECORDS_US
-------------- ----------         ---------- ----------
DATAFILE             180         30          4
1 row selected.
```

Cột dữ liệu RECORDS_TO chỉ ra số lượng các bản ghi được cấp phát cho một section.

Chương 8. QUẢN LÝ REDO LOG FILES

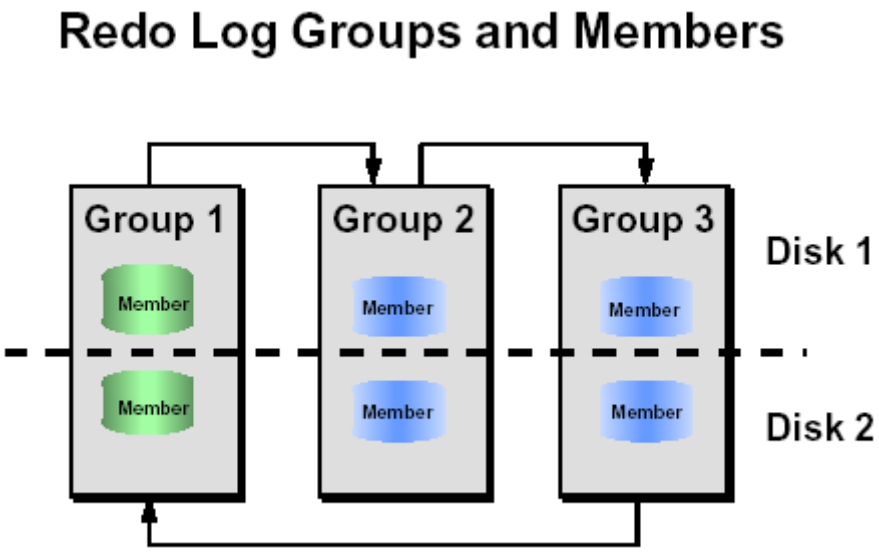
8.1.SỬ DỤNG CÁC REDO LOG FILES

8.1.1. Redo log file

Oracle server sử dụng các online redo log files để giảm thiểu việc mất mát dữ liệu trong database. Redo log files ghi lại tất cả các thay đổi trong database buffer cache trừ một vài ngoại lệ ghi dữ liệu trực tiếp.

Redo log files được sử dụng đến khi instance gặp sự cố và ta muốn khôi phục lại các dữ liệu đã commit nhưng chưa kịp ghi lên data files. Redo log files chỉ được sử dụng trong trường hợp khôi phục dữ liệu.

Quản trị viên cần thiết lập các bản sao các online redo log files của database để tránh việc mất mát thông tin trong database do việc sử dụng một file duy nhất.



Hình vẽ 26. Nhóm các redo log

8.1.2. Online Redo Log Groups

193. Là nhóm các bản sao riêng biệt của các online redo log files được gọi là online redo log *group*.

194. Background process `LGWR` thực hiện việc ghi đồng thời các thông tin tương tự nhau vào các member thuộc cùng một group. Khi một group đầy sẽ tiếp tục chuyển sang ghi dữ liệu trên group tiếp theo.

195. Oracle server, thông thường, cần ít nhất 02 online redo log file groups để có thể vận hành một database.

8.1.3. Online Redo Log Members

196. Mỗi một online redo log file trong một group được gọi là một *member (thành viên)*.

197. Mỗi member trong một nhóm có một số thứ tự *(log sequence numbers)* phân biệt và các member này có cùng một kích thước. Số thứ tự được gán mỗi khi Oracle server bắt đầu ghi dữ liệu vào log group để có thể phân biệt được các redo log file duy nhất. Số log sequence number được lưu trữ trong control file và trong phần header của tất cả các data files.

8.1.4. Nội dung của Online Redo Log Files (Members)

Online redo log files lưu trữ các *redo records* hay còn được gọi là các *redo entries.* Mỗi *redo record* là một nhóm các *change vectors (vector thay đổi dữ liệu)*, trong đó mỗi vector đặc trưng cho một sự thay đổi trên một block dữ liệu thuộc database. Ví dụ, khi ta thay đổi giá trị lương trong bảng employee, Oracle sẽ tạo ra một redo record lưu trữ lại việc thay đổi dữ liệu của data segment block, rollback segment block và transaction table tương ứng với thay đổi dữ liệu nói trên.

Các redo entries lưu trữ lại các dữ liệu để từ đó ta có thể tái tạo lại các thay đổi dữ liệu trong database, bao gồm cả rollback segments. Khi thực hiện phục hồi (recover) database sử dụng redo data, Oracle sẽ đọc các change vectors có trong các redo records rồi áp các thay đổi này vào các blocks tương ứng.

Các redo records được lưu trữ trong bộ nhớ đệm SGA. Mỗi khi thực hiện commit một transaction, LGWR sẽ ghi lại các redo records của transaction đó từ các redo log buffer thuộc SGA vào một online redo log file, và gán một số hiệu *system change number* (SCN) cho transaction đã được commit đó. Chi khi các redo records thuộc transaction đã được lưu trữ an toàn trên đĩa thì user process mới được nhận thông báo: transaction has been committed.

Các redo records có thể được ghi vào online redo log file trước khi transaction tương ứng được commit. Khi redo log buffer đầy, hoặc khi transaction commit, LGWR sẽ đẩy tất cả các redo log entries trong redo log buffer ra online redo log file, ngay cả khi redo records có thể chưa được commit để khi cần, Oracle có thể khôi phục (roll back) lại các thay đổi này.

8.1.5. Active và Inactive Online Redo Log Files

Tại mỗi một thời điểm, Oracle chỉ sử dụng một trong số các online redo log files để lưu trữ các redo records có trong redo log buffer. Online redo log file đó ở trạng thái sẵn sàng cho việc ghi dữ liệu, nó được gọi là *current* online redo log file.

Các online redo log files cần thiết cho việc khôi phục instance được gọi là *active* online redo log files. Trái lại, các online redo log files không cần thiết cho việc khôi phục instance được gọi là *inactive*.

Khi quản trị viên database đặt chế độ enable archiving, Oracle sẽ không thể tái sử dụng hay ghi đè lên các active online log file cho tới khi ARC*n* lưu trữ hết các nội dung của nó. Trong trường hợp disable archiving, khi online redo log file cuối cùng được điền đầy, việc lưu ra file sẽ được tiếp tục thực hiện đối với active file đầu tiên.

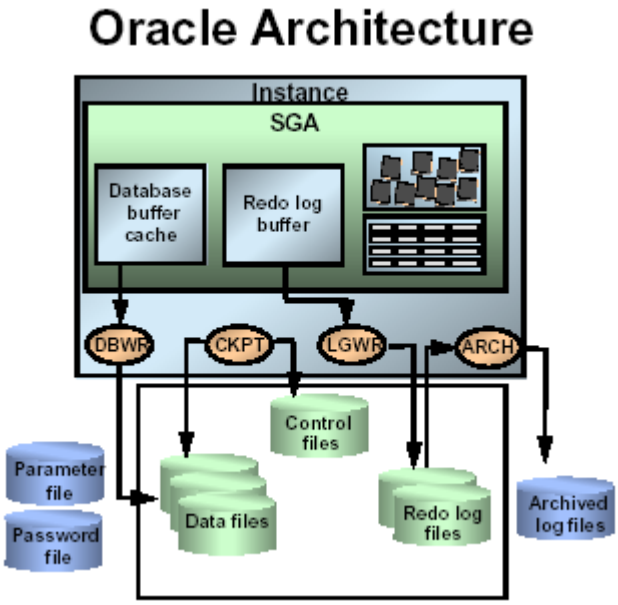
8.1.6. Thiết lập các Redo Log Files khởi tạo

Việc khởi tạo ban đầu tập hợp các online redo log file bao gồm các groups và các members được thực hiện trong quá trình tạo database.

Các tham số dưới đây xác định các giới hạn và số lượng của online redo log files:

- Tham số MAXLOGFILES trong lệnh CREATE DATABASE xác định số lượng tối đa các online redo log groups. Số lượng tối đa cho MAXLOGFILES là 255.
- Tham số MAXLOGMEMBERS trong lệnh CREATE DATABASE quy định số lượng tối đa các members có trong mỗi group.
- Tham số khởi tạo LOG_FILES xác định số lượng tối đa các log groups có thể được mở trong database tại thời điểm hiện thời. Giá trị này không được vượt quá giá trị MAXLOGFILES*MAXLOGMEMBERS.

8.2.LGWR, LOG SWITCHES VÀ CHECKPOINTS



Hình vẽ 27. Tổ chức các redo log files

8.2.1. Redo Log Buffer và Background process LGWR

Oracle Server sẽ tuần tự ghi lại các thay đổi đối với database có trong redo log buffer. Redo *log buffer* được sử dụng theo kiểu xoay vòng. Theo đó, các redo entries sẽ được tiên trình nền LGWR ghi vào một trong các online redo log groups gọi là online redo log group hiện thời (current) theo các tình huống sau:

- Khi commit một transaction
- Khi redo log buffer đã đầy
- Khi LGWR vượt quá thời gian timeout (3 giây)
- Trước khi DBWR ghi các blocks bị thay đổi trong database buffers cache vào trong các data files

Các members trong một redo log group được tiến trình LGWR ghi lên đó với cùng một nội dung dữ liệu. Cho nên không có khác biệt giữa các members trong một log group mà chỉ có sự khác nhau giữa các members ở các log group khác nhau.

8.2.2. Log Switches

LGWR ghi dữ liệu lên các online redo log files một cách tuần tự, tức là mỗi khi online redo log group được ghi đầy, LGWR sẽ lại chuyển sang ghi lên group tiếp theo. Khi online redo log file cuối cùng được ghi đầy, LGWR sẽ lại quay trở về online redo log group đầu tiên và lại bắt đầu quá trình ghi.

Log switch là sự kiện xảy ra khi LGWR dừng việc ghi trên một online redo log group và chuyển sang ghi trên online redo log group khác. Quản trị viên database cũng có thể thực hiện các log switches bằng tay. Mỗi khi xảy ra log switch, LGWT sẽ ghi dữ liệu lên log group mới và nó gán một số hiệu duy nhất để xác định được các redo entries vừa lưu giữ.

Mỗi khi xảy ra sự kiện log switch đồng thời một sự kiện checkpoint cũng sẽ được khởi tạo.

8.2.3. Checkpoints

Khi có checkpoints thì:

- Tất cả các dữ liệu trong database buffers đã bị thay đổi, tính cho đến thời điểm xảy ra checkpoint, sẽ được Background process DBWR ghi lên datafiles.
- Background process CKPT cập nhật phần headers của các data files và các control files.

Checkpoints có thể xảy ra đối với tất cả các data files trong database hoặc cũng có thể xảy ra với một data files cụ thể.

Checkpoint xảy ra theo các tình huống sau:

- Mỗi khi có log switch
- Khi một shut down một instance với các chế độ trừ chế độ abort
- Xảy ra theo như thời gian quy định trong các tham số khởi tạo LOG_CHECKPOINT_INTERVAL và LOG_CHECKPOINT_TIMEOUT
- Khi có yêu cầu trực tiếp của quản trị viên

Thông tin về checkpoint được lưu trữ trong Alert file trong trường hợp các tham số khởi tạo LOG_CHECKPOINTS_TO_ALERT được đặt là TRUE. Và ngược lại với giá trị FALSE.

8.3.LÊN KẾ HOẠCH SỬ DỤNG REDO LOG FILES

8.3.1. Xác định số lượng Online redo log files

Để xác định số lượng các online redo log files sử dụng cho phù hợp với database ta cần phải kiểm tra với nhiều cầu hình khác nhau.

Trong một số trường hợp, một database instance chỉ cần tới 02 groups. Tuy nhiên, trong một số trường hợp khác, một database instance lại có thể cần tới nhiều groups hơn để có thể luôn đảm bảo có các groups sẵn dùng cho LGWR. Ví dụ, khi các thông điệp ghi trong trace file hay Alert file cho biết LGWR thường xuyên phải chờ một group do vẫn chưa kết thúc được checkpoint, hoặc do group vẫn chưa được lưu trữ (archived) thì lúc này là lúc ta cần thêm mới các groups.

Mặc dù Oracle server cho phép sử dụng nhiều groups với số lượng members trong nó là khác nhau, ta vẫn nên cố gắng xây dựng một cấu hình cân đối (số lượng các members trong các group nên là bằng nhau).

8.3.2. Nơi đặt các Online Redo Log Files

Khi sử dụng đồng thời nhiều online redo log files, ta nên đặt các members của một group trên các phần đĩa khác nhau. Một điều lưu ý là khi một member nào đó không sẵn dùng (available) mà các members khác là sẵn dùng thì instance cũng không thể shut down được.

Việc tách biệt các archive log files và online redo log files trên các phần đĩa khác nhau, có thể làm giảm bớt xung đột giữa các background process ARCH và LGWR.

Các data files và online redo log files nên đặt trên các phần đĩa khác nhau để giảm bớt xung đột giữa LGWR và DBWR hạn chế việc mất dữ liệu ở cả data files và online redo log files trong trường hợp hỏng ổ đĩa.

8.3.3. Xác định kích thước cho các Online Redo Log Files

Kích thước tối thiểu của một online redo log file là 50 K còn kích thước tối đa thì tuỳ thuộc vào hệ điều hành. Các members thuộc các groups khác nhau có thể có các kích thước khác nhau; Tuy nhiên ta nên đặt kích thước giống nhau giữa các members này.

Việc sử dụng các groups có kích thước khác nhau chỉ nên thực hiện một cách tạm thời khi ta muốn thay đổi kích thước của các members. Trong trường hợp này, ta cần tạo các online redo log groups mới với kích thước khác, rồi sau đó loại bỏ (remove) các groups cũ đi.

Một số tình huống ảnh hưởng tới cấu hình của các online redo log files:

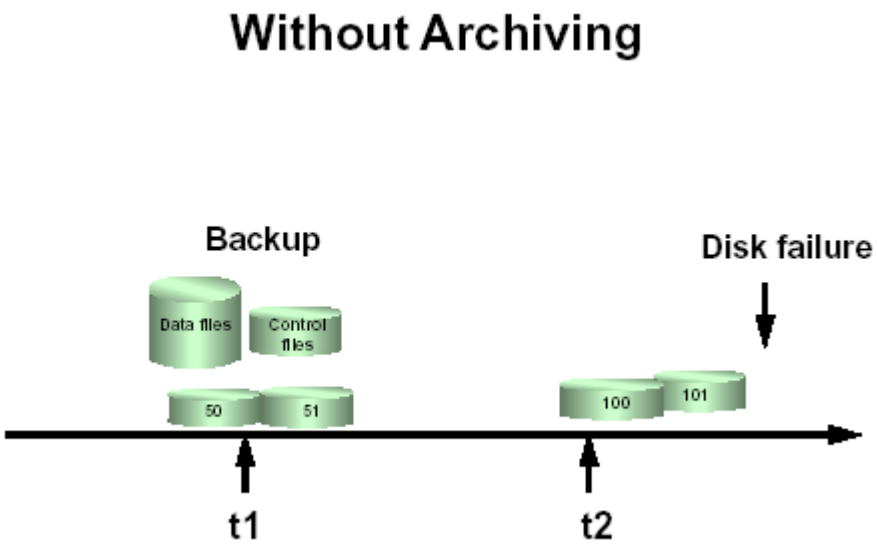
- Số lượng các log switches và checkpoints
- Số lượng và độ lớn của các redo entries
- Độ lớn của vùng không gian lưu trữ thứ cấp

8.3.4. Lưu trữ các redo log files

Quản trị viên database cần phải quyết định đặt chế độ ARCHIVELOG hay chế độ NOARCHIVELOG cho database.

Chế độ NOARCHIVELOG

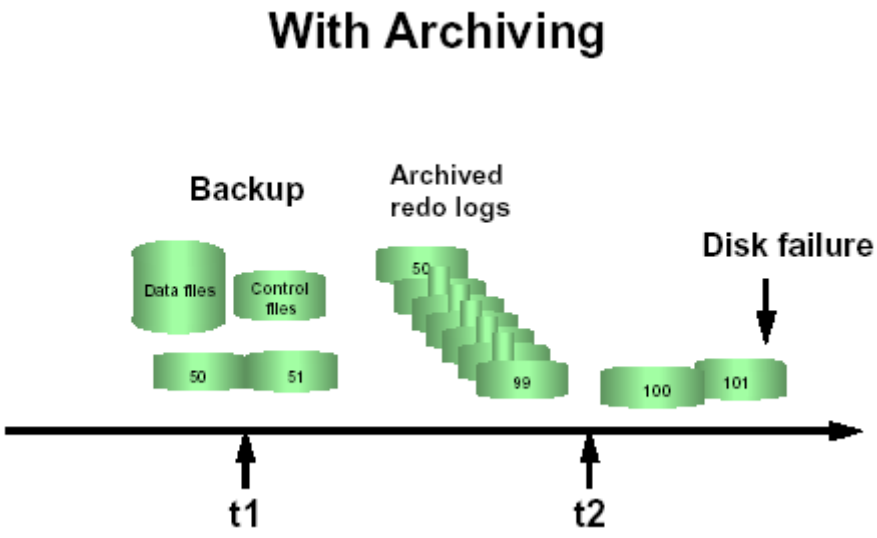
Với chế độ NOARCHIVELOG, các online redo log files sẽ bị ghi đè mỗi khi online redo log file đã ghi đầy và xảy ra log switches. LGWR sẽ không ghi đè lên redo log group cho tới khi kết thúc checkpoint của group đó



Hình vẽ 28. Lưu trữ dữ liệu ở chế độ NOARCHIVING

Chế độ ARCHIVELOG

Trong trường hợp database được thiết lập ở chế độ ARCHIVELOG, các groups đã đầy, mặc dù ở trạng thái inactive sẽ vẫn được lưu giữ. Do tất cả các thay đổi trong database đều được ghi lại trong các online redo log files, quản trị viên database có thể sử dụng phương pháp sao chép vật lý (physical backup) và có thể khôi phục lại các dữ liệu đã commit trong database mà không sợ bị mất dữ liệu.



Hình vẽ 29. Lưu trữ dữ liệu ở chế độ ARCHIVING

Có hai hình thức lưu trữ các online redo log files:

- Thực hiện lưu trữ bằng tay (manually). Lưu trữ các redo log file đã đầy theo lệnh của quản trị viên database.
- Lưu trữ tự động (automatically). Lưu trữ các redo log file đã đầy mỗi khi xảy ra log switch.

Tham số LOG_ARCHIVE_START trong parameter file xác định các chế độ lưu trữ này.

- LOG_ARCHIVE_START = TRUE, thực hiện lưu trữ ở chế độ tự động
- LOG_ARCHIVE_START = FALSE, thực hiện lưu trữ ở chế độ manually

8.4.ĐIỀU KHIỂN LƯU TRỮ SAU ĐỐI VỚI PRIMARY/STANDBY

Oracle cung cấp cơ chế điều khiển switch các online redo log group dựa theo thời gian (time-based). Trong cấu hình primary/standby, tất cả các noncurrent logs tại primary site sẽ được lưu trữ rồi vận chuyển tới standby database. Việc này sẽ hiệu quả khi hạn chế số lượng các redo records.

Việc thực hiện lưu trữ sau là vì standby database cho tất cả các thay đổi trên online redo log tại primary database được lưu trữ sau. Để điều khiển việc lưu trữ sau này, ta cần sử dụng tham số ARCHIVE_LAG_TARGET. Việc thiết lập tham số này cho phép ta hạn chế, cũng như xác định được khoảng thời gian được sử dụng cho lưu trữ sau.

8.4.1. Thiết lập tham số ARCHIVE_LAG_TARGET

Khi thiết lập tham số khởi tạo ARCHIVE_LAG_TARGET, Oracle sẽ kiểm tra theo định kỳ thời gian các online redo log của instance hiện thời và phát sinh các log switch theo các điều kiện sau:

- Giả sử ban đầu, current log được tạo sau *n* giây và sau đó lại mất *m* giây để lưu current log ra đĩa. Khi này khoảng thời gian n + m sẽ tương ứng với giá trị của tham số ARCHIVE_LAG_TARGET.
- Current log chứa các redo records.

Tham số ARCHIVE_LAG_TARGET cho biết giới hạn trên về thời gian (tính theo đơn vị giây) mà current log cần sử dụng. Do thời gian lưu trữ không chính xác bằng khoảng thời gian log switch.

Tham số khởi taon này nên được thiết lập với giá trị khoảng 30 giây.

ARCHIVE_LAG_TARGET = 1800

Giá trị 0 tương ứng với việc không thực hiện chức năng log switching. Đây là giá trị thiết lập mặc định.

Ta có thể đặt giá trị cho tham số ARCHIVE_LAG_TARGET ngay cả khi database không ở trong chế độ sao lưu (standby database). Ví dụ, tham số ARCHIVE_LAG_TARGET có thể được thiết lập để bắt buộc các logs phải thực hiện thao tác switch và lưu trữ lên ổ đĩa.

ARCHIVE_LAG_TARGET là một tham số động và ta có thể thay đổi giá trị của tham số này thông qua câu lệnh ALTER SYSTEM SET.

8.4.2. Các yếu tố ảnh hưởng tới tham số ARCHIVE_LAG_TARGET

Có một số yếu tố cần được xem xét khi ta thiết lập giá trị cho tham số ARCHIVE_LAG_TARGET.

- Tổng thời gian switch (xem như là thời gian lưu trữ) các logs
- Tần suất thực hiện switch các log khi nó đầy
- Lượng dữ liệu có thể redo bị mất khi database làm việc ở chế độ standby

Tham số ARCHIVE_LAG_TARGET sẽ trở nên không hữu dụng khi log được switch trong một khoảng thời gian quá ngắn. Tuy nhiên, trong trường hợp các redo được tạo ra với tốc độ không đều như nhau, thì khoảng thời gian ngắt quãng (interval) sẽ đưa ra giới hạn trên đối với current log. Khi database ở trong trạng thái nghỉ (idle) và redo records không được tạo ra thì, sau khoảng thời gian interval, log switch sẽ xảy ra và đẩy và ghi tất cả các redo records lên standby database.

Trong trường hợp ARCHIVE_LAG_TARGET được thiết lập với giá trị quá thấp thì cũng không tốt cho hệ thống về mặt hiệu suất. Là vì hệ thống liên tục phải thực hiện các log switches. Do vậy ta nên chọn giá trị hợp lý để nâng cao hiệu suất hệ thống.

8.5.XÁC ĐỊNH CHẾ ĐỘ LƯU TRỮ

Để biết được các thông tin về việc lưu trữ, ta có thể sử dụng một số cách sau:

8.5.1. Sử dụng lệnh Server Manager

Câu lệnh này cho biết chế độ log của database.

Ví dụ:

```
SVRMGR> ARCHIVE LOG LIST
Database log mode No Archive Mode
Automatic archival Disabled
Archive destination ?/dbs/arch
Oldest online log sequence 688
Current log sequence 689
```

8.5.2. Sử dụng thông tin trong data dictionary

Ta cũng có thể sử dụng thông tin trong các data dictionary views: V$DATABASE và V$INSTANCE.

Ví dụ:

```
SVRMGR> SELECT name, log_mode
      2> FROM v$database;
NAME            LOG_MODE
```

```
---------       ---------
U15             NOARCHIVELOG
1 row selected.

SVRMGR> SELECT archiver
     2> FROM v$instance;
ARCHIVE
---------
STOPPED
1 row selected.
```

Ta cũng có thể xem các thông tin liên quan đến các groups và các members thông qua views data dictionary V$THREAD, V$LOG.

Các thông tin cần quan tâm:

- V$THREAD: GROUPS, CURRENT_GROUP#, SEQUENCE#
- V$LOG: GROUP#, MEMBERS, STATUS, SEQUENCE#, BYTES

Ví dụ:

```
SVRMGR>SELECT groups, current_group#,sequence#
     2>FROM v$thread;
GROUPS           CURRENT_GR  SEQUENCE#
----------       ----------  ----------
2                1           689
1 row selected.


SVRMGR>SELECT group#,sequence#,bytes,members,status
     2>FROM v$log;
GROUP#    SEQUENCE#   BYTES    MEMBERS   STATUS
--------- ----------  -------- --------- -------
1         688         1048576  1         CURRENT
2         689         1048576  1         INACTIVE
2 rows selected.
```

Trong câu lênh ở trên, giá trị của cột STATUS được biểu hiện như sau:

- UNUSED chỉ ra online redo log group vẫn chưa được sử dụng. Trạng thái này tương ứng với việc online redo log file mới được thêm vào.
- CURRENT chỉ ra rằng online redo log group đang được sử dụng. Nó cũng ngầm đinh luôn trạng thái active đối với các online redo log group này.
- ACTIVE: trạng thái này ứng với the online redo log group vẫn đang được sử dụng nhưng không phải là online redo log group hiện thời.
- INACTIVE chỉ ra online redo log group không còn cần thiết cho việc khôi phục instance.

Để xác định tên của tất cả các member trong một group, ta có thể tra cứu thông tin trong V$LOGFILE: GROUP#, STATUS, MEMBER

Ví dụ:

```
SVRMGR>SELECT *
2>FROM v$logfile;
GROUP#           STATUS   MEMBER
----------       -------  ------------------------------
1                         /DISK3/log1a.rdo
2                         /DISK4/log2a.rdo
```

8.6.ĐIỀU KHIỂN CÁC LOG SWITCHS VÀ CHECKPOINTS

8.6.1. Thực hiện log switches

Log switches và checkpoint là các sự kiện xảy ra một cách tự động mỗi khi online redo log group đầy. Tuy nhiên, ta vẫn có thể phát sinh các Log switchs thông qua lệnh của Server Manager.

```
SVRMGR>ALTER SYSTEM SWITCH LOGFILE;
```

Trong Oracle Enterprise Manager – OEM, ta làm theo các bước sau:

1. Sử dụng Backup Manager
2. Chọn Subsystem
3. Chọn Logfile --> Switch logfile

8.6.2. Thực hiện checkpoint

Ta cũng có thể phát sinh các Checkpoints thông qua lệnh:

```
SVRMGR>ALTER SYSTEM CHECKPOINT;
```

Trong Oracle Enterprise Manager – OEM, ta làm theo các bước sau:

1. Sử dụng Backup Manager
2. Chọn Subsystem
3. Chọn Logfile --> Force checkpoint

8.6.3. Điều chỉnh các ngắt quãng checkpoints

Trong trường hợp database sử dụng các online redo log files lớn, ta có thể điều chỉnh lại các ngắt quãng đối với online redo log file đó thông qua các tham số:

- `LOG_CHECKPOINT_INTERVAL`: Số lượng blocks (tính theo số block của hệ điều hành) lớn nhất để thực hiện một checkpoint
- `LOG_CHECKPOINT_TIMEOUT`: Khoảng thời gian lớn nhất (tính theo đơn vị giây) để thực hiện một checkpoint.

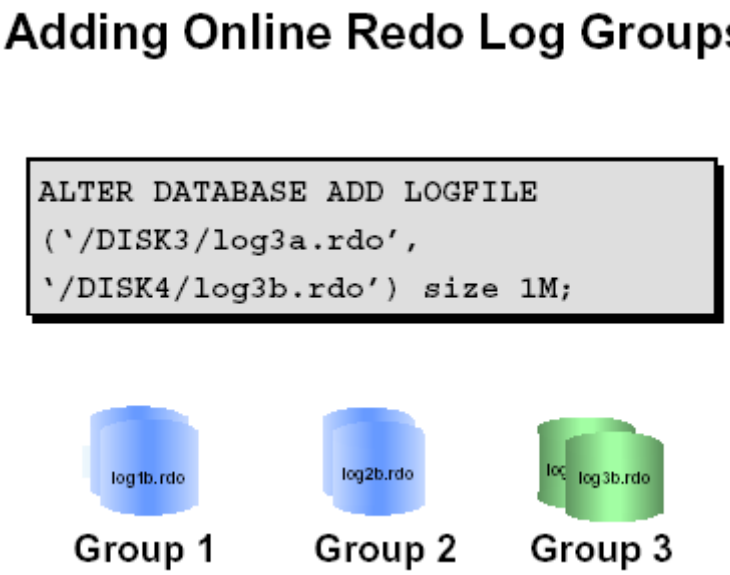
8.7.QUẢN TRỊ CÁC REDO LOG FILES

8.7.1. Bổ sung các online redo log groups

Trong một vài trường hợp, ta có thể cần tới việc nạp thêm các log groups hay các log members.

Cú pháp:

```
ALTER DATABASE [database]
ADD LOGFILE [GROUP integer] filespec
[, [GROUP integer] filespec]...]
```



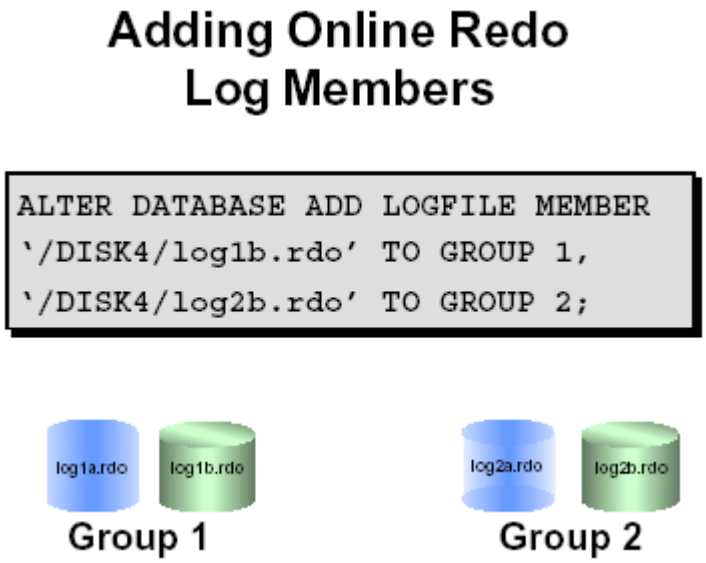
Hình vẽ 30. Bổ sung online redo log groups

Với câu lệnh trên, ta cần chỉ ra tên và đường dẫn của các members trong từng group cụ thể. Giá trị của tham số GROUP được chọn tương ứng với mỗi redo log file group. Trong trường hợp bỏ qua tham số này, Oracle server sẽ tự động sinh ra các giá trị thích hợp.

Trong Oracle Enterprise Manager – OEM, ta làm theo các bước sau:

1. Sử dụng Backup Manager
2. Chọn Subsystem
3. Chọn Logfile --> Add Logfile Group

8.7.2. Bổ sung các online redo log members



Hình vẽ 31. Bổ sung online redo log members

Tương tự như các group, ta cũng có thể thêm mới các member cho từng group bằng câu lệnh SQL

```
ALTER DATABASE [database]
      ADD LOGFILE MEMBER
      [ 'filename' [REUSE]
      [,'filename' [REUSE]]...
      TO {GROUP integer
      |('filename'[, 'filename']...)
      }
      ]...
```

Lưu ý: tên file được chỉ ra cần kèm theo đường dẫn đầy đủ. Trong trường hợp không có đường dẫn, file sẽ được xem như được đặt trong thư mục mặc định. Nếu file thêm mới đã tồn tại, ta cần thêm vào tuỳ chọn REUSE.

Trong Oracle Enterprise Manager – OEM, ta làm theo các bước sau:

1. Sử dụng Backup Manager
2. Chọn Subsystem
3. Chọn Logfile --> Add Logfile Member

8.7.3. Định lại chỗ cho các redo log file

Trong một vài trường hợp, ta cần phải dịch chuyển các file redo log tới một vị trí khác, để đảm bảo an toàn chẳng hạn. Khi này, ta cần thực hiện theo các bước sau:

1. Tắt database.
2. Sao chép các online redo log files tới một địa điểm mới.

3. Restart database ở chế độ mount.
4. Thực hiện lệnh `ALTER DATABASE RENAME FILE` để thay đổi con trỏ trong control file, trỏ tới một đường dẫn file mới.
5. Mở lại database (Lệnh: `ALTER DATABASE OPEN`).

Câu lệnh đổi tên file:

```
ALTER DATABASE [database]
      RENAME FILE 'filename'[, 'filename']...
      TO 'filename'[, 'filename']...
```

Lưu ý: Phải tồn tại file ở đường dẫn mới chỉ ra.

Trong Oracle Enterprise Manager – OEM, ta làm theo các bước sau:

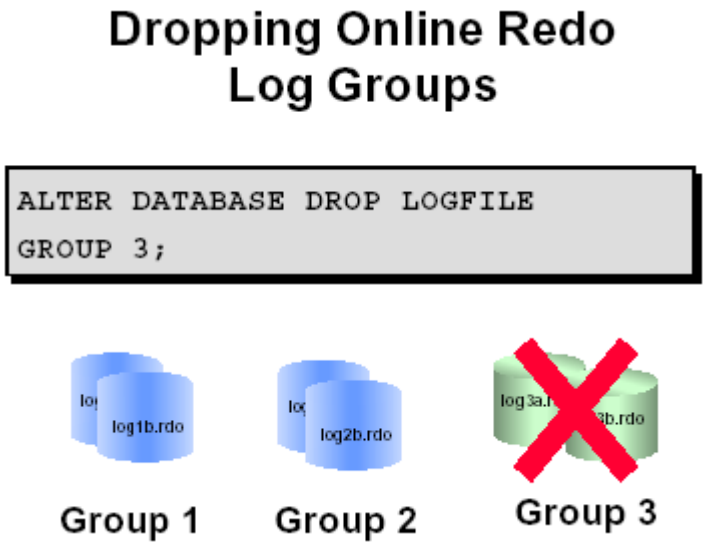
1. Sử dụng Backup Manager
2. Chuyển tới nút Logfile Group
3. Chọn log file group tương ứng
4. Thay đổi tên file trong trường thuộc tính.

8.7.4. Ngừng sử dụng các Online redo log groups

Để có thể thay đổi kích thước các online redo log groups, ta có thể thêm mới các online redo log group và xoá bỏ các online redo log group đã có.

Sử dụng lệnh của Server Manager để ngưng sử dụng online redo log group:

```
ALTER DATABASE [database]
      DROP LOGFILE
      {GROUP integer|('filename'[, 'filename']...)}
      [,{GROUP integer|('filename'[, 'filename']...)}]...
```



Hình vẽ 32. Ngừng sử dụng Online redo log groups

Trong Oracle Enterprise Manager – OEM, ta làm theo các bước sau:

1. Sử dụng Backup Manager
2. Chuyển tới nút Logfile Group
3. Chọn log file group tương ứng
4. Chọn Logfile --> Drop Logfile Group
5. Bấm nút OK.

Một số điểm cần lưu ý khi xoá log groups

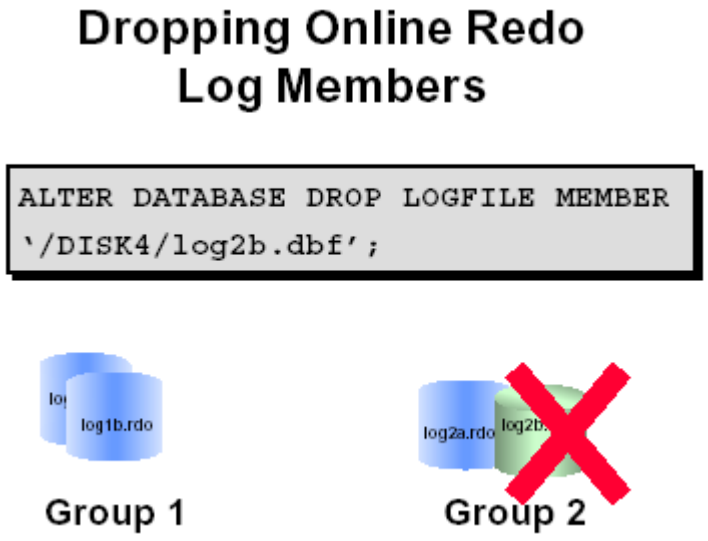
- Một instance cần ít nhất hai nhóm (group) các online redo log files.
- Không thể huỷ (drop) group đang ở trạng thái active.
- Khi huỷ một online redo log group, thực chất ta chỉ huỷ về mặt logic mà thôi. Oracle sẽ không tiếp tục quản lý nó nữa. Tuy nhiên, các file sẽ vẫn còn và không bị xoá bởi hệ điều hành.

8.7.5. Ngừng sử dụng các Online redo log members

Tương tự như các log group, đối với các log members ta cũng có thể ngưng sử dụng.

Sử dụng lệnh của Server Manager để ngưng sử dụng online redo log member:

```
ALTER DATABASE [database]
      DROP LOGFILE MEMBER 'filename'[, 'filename']...
```



Hình vẽ 33. Ngừng sử dụng Online redo log members

Trong Oracle Enterprise Manager – OEM, ta làm theo các bước sau:

1. Sử dụng Backup Manager
2. Chuyển tới nút Logfile Group
3. Chọn log file group tương ứng

4. Chọn Logfile --> Drop Logfile Member
5. Bấm nút OK.

Một số điểm cần lưu ý khi xoá log members

- Không thể ngừng sử dụng member của group mà có trạng thái là VALID.
- Nếu group đang trong trạng thái active, ta cần phải thực hiện log switch để chuyển sử dụng sang một log group khác trước khi ngưng sử dụng các member của group hiện thời.
- Khi huỷ một online redo log member, thực chất ta chỉ huỷ về mặt logic các file vẫn không bị xoá bởi hệ điều hành.

8.7.6. Xoá rỗng Online redo log file

Trong một vài trường hợp các members bị lỗi, quản trị viên database có thể xử lý bằng cách khởi tạo lại các log file thông qua lệnh SQL để khởi tạo lại:

```
ALTER DATABASE CLEAR LOGFILE
```

Cú pháp:

```
ALTER DATABASE [database]
      CLEAR [UNARCHIVED] LOGFILE
      {GROUP integer|('filename'[, 'filename']...)}
      [,{GROUP integer|('filename'[, 'filename']...)}]...
```

Sử dụng lệnh này cũng tương đương với việc thêm mới các online redo log file và xoá bỏ các redo log file hiện thời.

Lưu ý:

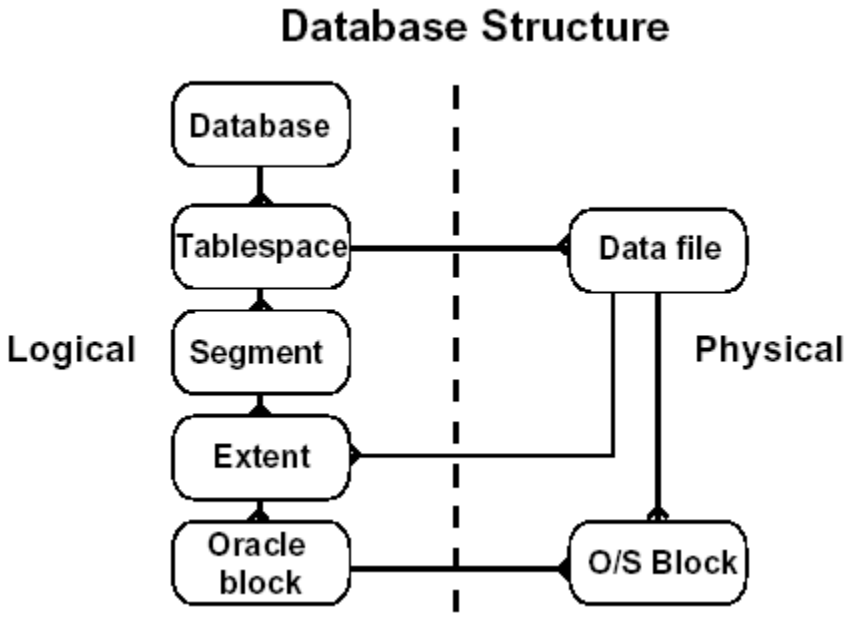
Khi xoá rỗng logfile mà nó không dùng để lưu trữ, ta cần bổ sung từ khoá UNARCHIVED.

Chương 9. QUẢN TRỊ TABLESPACES VÀ DATA FILES

9.1.CẤU TRÚC CỦA DATABASE

Cấu trúc database bao gồm cấu trúc logic và cấu trúc vật lý.

Cấu trúc vật lý bao gồm tập hợp các control files, online redo log files và các data files. Cấu trúc logic bao gồm các schema objects tablespaces, segments, extents và data blocks.



Hình vẽ 34. Cấu trúc database

9.1.1. Quan hệ giữa database với các tablespaces và data files

Về mặt logic, một database có thể phân nhỏ thành nhiều phần gọi là các tablespaces.

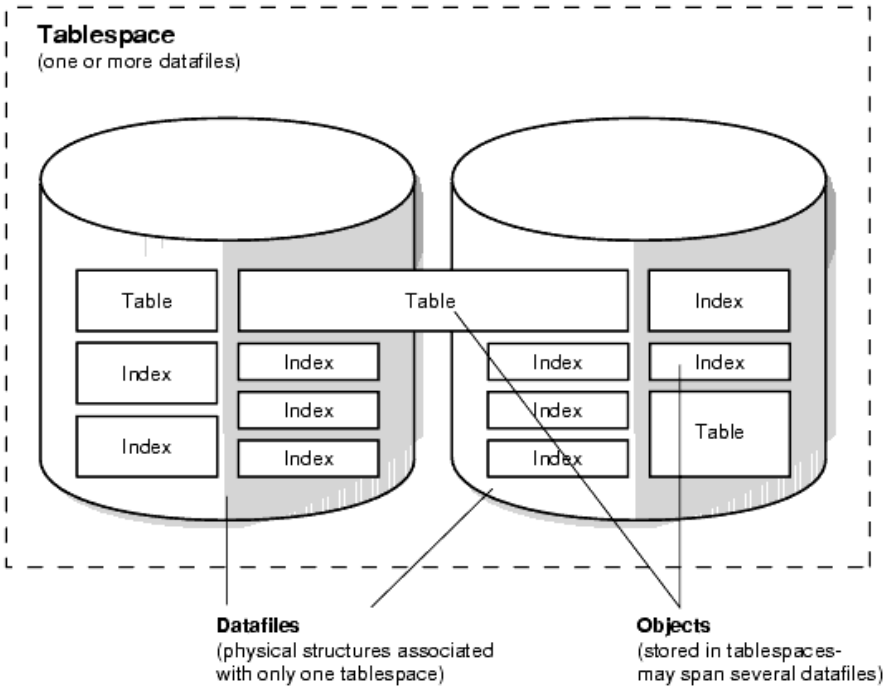
Tablespace

- Một tablespace chỉ thuộc một database.
- Mỗi tablespace có thể chứa một hay nhiều data file thuộc hệ điều hành.
- Tablespaces có thể đặt ở trạng thái online hay offline trong lúc database đang chạy.
- Ngoại trừ tablespace `SYSTEM` hay tablespace chứa rollback segments đang có trạng thái `ACTIVE`, các tablespaces đều có thể chuyển về trạng thái offline trong lúc database đang chạy.
- Các tablespaces cũng có thể chuyển đổi trạng thái read-write hay read-only.

Sử dụng tablespace

- Để điều khiển vùng không gian cấp phát và gán cho mỗi users
- Với việc đặt chế độ online hay offline cho các tablespace, ta có thể thay đổi tính sẵn dùng (availability) của các dữ liệu trong các tablespace

- Ta cũng có thể phân biệt các dữ liệu lưu trữ giữa các thiết bị để tăng hiệu suất sử dụng database.
- Thực hiện sao lưu và phục hồi dữ liệu từng phần, nâng cao hiệu suất hệ thống



Hình vẽ 35. Quan hệ giữa tablespace và datafile

Data files

Mỗi một tablespace có thể bao gồm một hay nhiều data files, là các file thuộc hệ điều hành dùng để lưu trữ dữ liệu trong tablespace. Các data files có một số tính chất chính sau:

- Một data file chỉ thuộc về một tablespace.
- Quản trị viên database có thể thay đổi kích thước của data file ngay cả khi nó đã được tạo lập, làm tăng tính năng động cho các đối tượng có trong tablespace.

9.1.2. Quan hệ giữa segment với các extent và các blocks

Oracle cho phép điều chỉnh không gian đĩa thông qua việc thay đổi kích thước của các cấu trúc lưu trữ logic như: tablespaces, segments, extents và blocks.

Setgments

Một segment là vùng không gian cấp phát tương ứng với một kiểu cấu trúc logic có trong một tablespace. Ta có thể phân ra làm một số loại segment chính sau:

- Data segments
- Index segments
- Temporary segments
- Rollback segments

Một segment cụ thể là một data segment có thể được trải rộng trên nhiều datafiles thuộc một tablespace.

Extents

Extent là một cấp độ phân chia về mặt logic tiếp theo của databse. Một extent là tập hợp liên tiếp các blocks dữ liệu. Mỗi kiểu segment được quy đinh bao gồm một hay nhiều extents. Khác với segments, một extent chỉ được nằm duy nhất trên một data file.

Data Blocks

Đây là đơn vị lưu trữ *(lưu ý không phải là đơn vị quản lý)* dữ liệu nhỏ nhất trong database Oracle. Một block dữ liệu sẽ tương ứng với một hay nhiều blocks của hệ điều hành. (Ví dụ: hệ điều hành Windows 32, 1 block hệ điều hành = 32 kbytes = 32*1024 bytes). Kích thước của block dữ liệu được xác định bởi tham số khởi tạo `DB_BLOCK_SIZE` ngay khi database được tạo. Block trong database cũng là đơn vị vào ra nhỏ nhất.

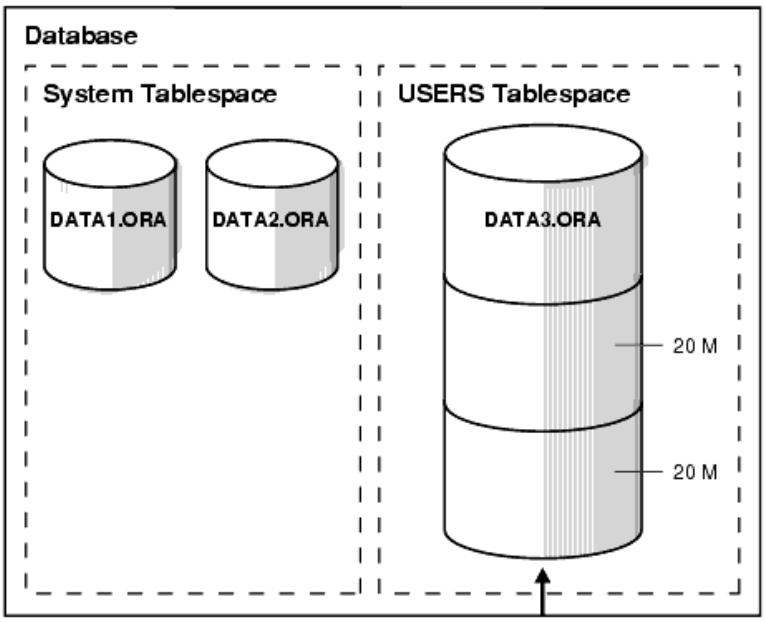
9.2.PHÂN LOẠI CÁC TABLESPACES

9.2.1. Tablespace SYSTEM và non-SYSTEM

Một database gồm có ít nhất một tablespace là tablespace SYSTEM, là nơi lưu trữ các thông tin của hệ thống. Ngoài ra, database còn có thể thêm vào các tablespace khác, đó là các non-SYSTEM tablespaces, chứa dữ liệu của các user.

Tablespace SYSTEM

- Có trong tất cả các database
- Chứa thông tin về các data dictionary views, các định nghĩa của stored procedures, packages, và các database triggers dưới dạng PL/SQL program units.
- Chứa SYSTEM rollback segment
- Không nên chứa dữ liệu người dùng trong tablespace này mặc dù có thể.



Hình vẽ 36. Dữ liệu người dùng nên đặt trong tablespace riêng

Non-SYSTEM Tablespace

- Chứa các rollback segments
- Chứa các temporary segments
- Chứa các data segments
- Chứa các index segments

9.2.2. Tablespaces read-only / read-write

Tablespaces read-only

Mục đích chính của việc sử dụng các tablespaces read-only (chỉ đọc) là hạn chế các thủ tục cần thiết khi thực hiện sao lưu và phục hồi một phần lớn dữ liệu không bị thay đổi (static) của database. Oracle không thực hiện cập nhật các files nằm trong tablespace read-only, vì thế các files có thể được đặt trong thiết bị chỉ đọc như CD ROMs hay ổ đĩa WORM drives (Write Once-Read Many).

Mỗi khi tạo mới một tablespace, hệ thống sẽ tạo cho ta một tablespace có đủ cả quyền đọc và quyền ghi. Ta có thể thay đổi lại thuộc tính tablespace thành read-only thông qua mệnh đề `READ ONLY` trong câu lệnh `ALTER TABLESPACE`. Việc này sẽ dẫn tới tất cả các datafiles thuộc tablespace đó sẽ được đặt lại thuộc tính là read-only.

Câu lệnh `ALTER TABLESPACE ... READ ONLY` sẽ đặt tablespace vào chế độ chuyển tiếp (*transitional read-only)* và chờ cho tất cả các transactions trên đó kết thúc (commit hoặc roll back). Chế độ chuyển tiếp này sẽ không cho phép bất kỳ một thao tác ghi mới nào được thực hiện trên tablespace ngoại trừ việc rollback các transactions hiện thời và thay đổi dữ liệu trong các blocks trong tablespace. Do đó, chế độ chuyển tiếp của tablespace cũng hệt như

tablespace read-only đối với các câu lệnh mới của người dùng ngoại trừ lệnh ROLLBACK. Sau khi tất cả các transactions hiện thời đã kết thúc thì câu lệnh ALTER TABLESPACE ... READ ONLY mới được xem là kết thúc và tablespace được đặt ở chế độ read-only.

Đặt chế độ read-only cho tablespace không làm ảnh hưởng tới trạng thái offline hay online của tablespace đó. Các Offline datafiles vẫn không thể truy xuất được. Việc đưa một datafile trong tablespace read-only vào chế độ online sẽ cho phép user có thể đọc dữ liệu trong file đó. File này vẫn không thể viết dữ liệu vào trừ phi tablespace tương ứng được đặt lại ở chế độ cho phép đọc và ghi.

Read-only tablespaces không thể bị sửa đổi. Để cập nhật dữ liệu trong một read-only tablespace, trước tiên ta cần đặt lại chế độ cho tablespace là read-write. Sau đó, thực hiện cập nhật dữ liệu trong tablespace rồi đặt lại chế độ read-only cho tablespace đó.

Do các read-only tablespaces không bị sửa đổi nên ta cũng không cần thiết phải thực hiện việc backup dữ liệu trên nó nhiều lần. Và ta cũng không cần thiết phải phục hồi lại các read-only tablespaces, do dữ liệu trong đó không bị thay đổi.

Ta không thể bổ sung các datafiles vào tablespace read-only, ngay cả khi đã đặt chế độ cho tablespace là offline. Bởi vì, khi bổ sung một datafile, Oracle sẽ phải cập nhật phần thông tin header trong khi đó thao tác ghi lên tablespace này là không được phép.

Tablespace read-write

Trái với tablespace read-only, với các tablespace read-wirte, ta có thể thực hiện các thao tác đọc và ghi trên đó.

Ta cũng có thể sử dụng mệnh đề READ WRITE trong câu lệnh ALTER TABLESPACE để thay đổi trạng thái tablespace read-only thành trạng thái read-write.

9.2.3. Temporary tablespace / permanent tablespace

Temporary tablespaces được sử dụng để dành riêng cho các thao tác sắp xếp dữ liệu. Trong temporary tablespace không có bất cứ segments dữ liệu nào nằm trong đó.

Sort segments có thể cùng được chia sẻ sử dụng khi nhiều thao tác sắp xếp cùng được thực hiện. Một sort segment được sử dụng cho tất cả các instance có thực hiện thao tác sắp xếp trên một tablespace.

Việc sử dụng các temporary tablespaces cho phép nâng cao hiệu suất thực hiện mỗi khi có nhiều thao tác sắp xếp được thực hiện trên một vùng nhớ lớn và không phù hợp với kích thước của bộ nhớ trong của máy tính. Sort segment thuộc temporary tablespace được tạo ra vào ngay thời điểm đầu của thao tác sắp xếp. Sort segment sẽ được cấp thêm vùng nhớ và mở rộng dần cho tới khi kích thước của segment ngang bằng hoặc lớn hơn tổng số kích thước lưu trữ cần thiết cho việc thực hiện tất cả các thao tác sắp xếp của instance.

Các tablespaces không phải là *temporary tablespaces* được gọi là các permanent tablespaces. Các permanent tablespace được sử dụng để lưu trữ dữ liệu trong database.

9.3.QUẢN LÝ KHÔNG GIAN TRONG TABLESPACES

Tablespaces cấp phát vùng không gian theo các *extents*. Tablespaces sử dụng hai phương pháp khác nhau để cấp phát và giải phóng vùng không gian lưu giữ:

- Quản lý các extents qua data dictionary (dictionary-managed tablespaces)
- Quản lý các extents qua tablespace (locally-managed tablespaces)

Ngay khi tạo tablespace, ta cần lựa chọn luôn phương pháp quản lý vùng không gian sẽ được áp dụng cho tablespace đó. Khi đã chọn rồi, ta không thể thay đổi phương pháp quản lý không gian nữa.

9.3.1. Dictionary-Managed Tablespaces

Trong phương pháp này tablespace sử dụng data dictionary để quản lý các extents của nó. Oracle cập nhật từng tables trong data dictionary mỗi khi cấp phát, giải phóng hay sử dụng lại một extent. Oracle cũng lưu lại các thông tin rollback của việc cập nhật các dictionary tables.

Theo mặc định, phương pháp quản lý này sẽ được áp dụng cho các tablespaces có trong database. Trong các phiên bản Oracle 8.0 hoặc sớm hơn, chỉ có một phương pháp đó chính là phương pháp này.

9.3.2. Locally-Managed Tablespaces

Bên cạnh đó, tablespace cũng có thể quản lý các extents của nó thông qua một bitmap (ánh xạ bit) trong từng datafile từ đó xác định được trạng thái của các blocks trong datafile là đang sử dụng hay đã được giải phóng. Mỗi một bit trong bitmap sẽ tương ứng với một block hay một nhóm các blocks. Mỗi khi có một extent được cấp phát, giải phóng hay tái sử dụng, Oracle sẽ thay đổi giá trị của bitmap theo đúng như trạng thái mới của các blocks. Việc thay đổi này sẽ không làm phát sinh các thông tin trong rollback do không có thao tác cập nhật dữ liệu nào trong các tables của data dictionary (Ngoại trừ trường hợp đặc biệt liên quan đến các thông tin hạn mức (quota) của tablespace).

Locally-managed tablespaces có một số ưu điểm hơn so với dictionary-managed tablespaces là:

- Quản lý cục bộ các extents tránh các thao tác quản lý không gian theo kiểu đệ quy. Việc này có thể xảy ra khi sử dụng phương pháp dictionary-managed tablespaces nếu việc sử dụng hay giải phóng không gian là kết quả của các thao tác sử dụng hay giải phóng không gian trong rollback segment hay data dictionary table.
- Quản lý cục bộ các extents một cách tự động các vùng không gian giải phóng liền kề với nhau. Điều này là cần thiết khi thực hiện công việc hợp nhất các extents rỗi.

Kích thước của các extents được quản lý cục bộ có thể được xác định tự động bởi hệ thống. Mặt khác, tất cả các extents có thể có cùng một kích cỡ như nhau trong phương pháp locally-managed tablespace.

Mệnh đề LOCAL trong phần EXTENT MANAGEMENT của câu lệnh CREATE TABLESPACE sẽ chỉ rõ phương thức quản lý không gian:

- Với các permanent tablespaces và temporary tablespaces, ta có thể sử dụng mệnh đề EXTENT MANAGEMENT LOCAL.
- Trong phiên bản 8i, phương pháp quản lý này vẫn chưa được áp dụng cho tablespace SYSTEM. Nếu áp dụng, hệ thống sẽ phát sinh lỗi 809225.

9.4.THIẾT LẬP TRẠNG THÁI CHO TABLESPACES

Quản trị viên database có thể thiết lập trạng thái cho các tablespaces là *online* (có thể sử dụng) hay *offline* (không thể sử dụng) ngoại trừ tablespace SYSTEM mỗi khi mở database. Tablespace SYSTEM luôn ở trạng thái online mỗi khi database được mở bởi vì Oracle luôn phải sử dụng các dữ liệu trong dictionary.

Một tablespace thông thường ở chế độ online khi đó, các dữ liệu trong nó là sẵn sàng đối với các database users. Tuy nhiên, quản trị viên database có thể đặt chế độ offline cho tablespace:

- Khi này một phần của database sẽ không thể truy xuất được, trong khi phần còn lại vẫn có thể truy xuất bình thường.
- Thực hiện offline tablespace khi backup dữ liệu (mặc dù ta vẫn có thể backup dữ liệu ngay khi database đang chạy và các tablespace ở trạng thái online).

Lưu ý: ta không thể đặt chế độ offline cho tablespace nếu nó có chứa các rollback segments đang được sử dụng.

Đặt Offline cho tablespace

Khi một tablespace được đưa ra offline, Oracle sẽ không cho phép thực hiện các câu lệnh SQL có tham chiếu tới các objects lưu trữ trong tablespace này. Oracle lưu lại các dữ liệu rollback tương ứng khi thực hiện câu lệnh SQL trong một rollback segment khác có trong tablespace SYSTEM thay vì là rollback segment có trong tablespace được offline nếu có. Tablespace được đưa về online trở lại, Oracle sẽ áp lại các dữ liệu rollback đang có trong tablespace SYSTEM vào tablespace đó.

Ta chỉ có thể đưa một tablespace thành online trong chính database mà nó được tạo, không thể đặt online cho tabalespace trong một database khác được. Việc này được giám sát bởi các thông tin có trong dictionary.

Oracle tự động thực hiện chuyển chế độ từ online thành offline đối với tablespaces mỗi khi xảy ra sự cố hệ thống. Ví dụ như: tiến trình DBW*n* gặp lỗi.

9.5.TRAO ĐỔI CÁC TABLESPACES GIỮA DATABASES

Ta có thể sử dụng chức năng *transportable tablespaces* để dịch chuyển một phần của một database sang một database Oracle khác. Việc trao đổi các tablespaces giữa các database là rất hữu ích cho:

- Việc dịch chuyể dữ liệu từ hệ thống xử lý trực tuyến (OLTP – online transaction processing systems) sang thành dữ liệu của hệ thống kho dữ liệu (data warehouse staging systems).
- Cập nhật kho dữ liệu (data warehouses) và các dữ liệu thuộc hệ thống.
- Nạp các dữ liệu từ các kho cơ sở dữ liệu trung tâm (central data warehouses).
- Lưu trữ các dữ liệu của hệ thống OLTP and data warehouse systems efficiently.
- Cung cấp dữ liệu cho các khách hàng hoặc người sử dụng nội bộ.

Dịch chuyển dữ liệu thông qua việc trao đổi các tablespaces cho phép di chuyển dữ liệu nhanh chóng và hiệu quả hơn các cách dịch chuyển dữ liệu khác như export/import hay unload/load đối với cùng một dữ liệu, Do việc trao đổi các tablespace chỉ đòi hỏi phải sao chép các datafiles rồi tích hợp thông tin về cấu trúc của tablespace vào database mới. Có thể sử dụng phương pháp trao đổi các tablespaces để dịch chuyển các index data, do đó, để tránh việc tái tạo lại (rebuilds) các index, ta có thể thực hiện công việc này để nạp dữ liệu trong các bảng.

9.5.1. Một số hạn chế trong việc trao đổi các tablespace:

269. Database nguồn và đích phải được chạy trên cùng một nền phần cứng (hardware platform). Ví dụ, có thể trao đổi các tablespaces giữa database Oracle chạy trên hệ điều hành Sun Solari, hoặc trao đổi các tablespaces giữa các databases Oracle chạy trên hệ điều hành NT. Tuy vậy, ta không thể trao đổi các tablespace giữa database Oracle chay trên SUN Solaris với các database Oracle chạy trên NT.
270. Database nguồn và đích phải có cùng một kích thước của data block.
271. Database nguồn và đích phải sử dụng cùng một tập ký tự sử dụng trong database (national character set).
272. Không thể chuyển đổi tablespace sang database đích khi database này đã có một tablespace có cùng tên.
273. Việc chuyển đổi tablespaces không được hỗ trợ:

- Snapshot/replication
- Function-based indexes
- Scoped REFs
- Domain indexes (Một kiểu index mới, cho phép mở rộng việc đánh chỉ số)

9.5.2. Các bước thực hiện chuyển đổi một tablespace giữa các database

1. Chỉ có thể thực hiện trao đổi các tablespaces mà nó không chứa các tham chiếu tới tablespace khác.

2. Tạo một transportable tablespace set.

Transportable tablespace set chứa các datafiles ứng với tập các tablespaces được sử dụng để chuyển đổi các file có chứa thông tin cấu trúc của các tablespaces dịch chuyển.

(Xem minh hoạ việc tạo một transportable tablespace set ở phia dưới).

3. Chuyển đổi tablespace.

   Sao chép các datafiles và export file sang database đích. Có thể sử dụng các công cụ sao chép file thông thường của hệ điều hành để thực hiện công việc này

4. Đưa tablespace vào sử dụng (plug-in).

   Thực hiện công việc Import để đưa các tablespaces vào database đích.

Minh hoạ việc trao đổi tablespace

1. Để biết tablespace SALES_1 và SALES_2 có chứa các tham chiếu trong nó không, ta thực hiện câu lệnh:

```
EXECUTE dbms_tts.transport_set_check('sales_1,sales_2', TRUE);
```

Câu lệnh này sinh ra kết quả và lưu trong view có tên là: TRANSPORT_SET_VIOLATIONS. Sử dụng câu lệnh truy vấn để xem kết quả:

```
SELECT * FROM transport_set_violations;
```

Lệnh truy vấn kết xuất kết quả rỗng cho biết tablespace không chứa các tham chiếu tới tablespace bên ngoài.

2. Tạo transportable tablespace set

Phát lệnh thay đổi trạng thái của tablespace về trạng thái read-only để không cho phép cập nhật dữ liệu vào tablespace này, chuẩn bị cho việc trao đổi tablespace.

```
ALTER TABLESPACE sales_1 READ ONLY;
```

Sử dụng công cụ tiện ích Export của Oracle để kết xuất các tablespace này:

```
EXP TRANSPORT_TABLESPACE=y TABLESPACES=(sales_1,sales_2)
      TRIGGERS=y/n CONSTRAINTS=y/n GRANTS=y/n FILE=expdat.dmp
```

TRIGGERS=Y – cho phép kết xuất; N – không cho phép kết xuất.

GRANTS=Y – kết xuất cả các quyền trên mỗi bảng thuộc tablespace đó; N – không kết xuất.

CONSTRANTS=Y – các ràng buộc tham chiếu sẽ được kết xuất; N – không kết xuất các ràng buộc tham chiếu.

3. Thực hiện sao chép các datafile của tablespace vừa được kết xuất ra một vị trí khác.

4. Đặt lại trạng thái bình thường cho tablespace vừa được xem xét.

```
ALTER TABLESPACE sales_1 READ WRITE;
```

5. Đưa bản sao của các datafile vừa được sao chép vào vị trí tương ứng với database đích.

6. Connect vào database mới với mức quyền SYSDBA.

7. Đưa các tablespaces đã được kết xuất vào database mới

```
IMP       TRANSPORT_TABLESPACE=y
  DATAFILES=('c:\db\sales_jan','c:\db\sales_feb',...)
  TABLESPACES=(sales_1,sales_2) TTS_OWNERS=(dcranney,jfee)
  FROMUSER=(dcranney,jfee) TOUSER=(smith,williams) FILE=expdat.dmp
```

9.6.TẠO TABLESPACE

9.6.1. Lệnh tạo tablespace

Ta có thể sử dụng câu lênh SQL để tạo một tablespace.

Cú pháp:

```
CREATE TABLESPACE tablespace
      DATAFILE filespec [autoextend_clause]
      [, filespec [autoextend_clause]]...
      [MINIMUM EXTENT integer[K|M]]
      [DEFAULT storage_clause]
      [PERMANENT|TEMPORARY]
      [ONLINE|OFFLINE]

storage_clause:= =
STORAGE ( [INITIAL integer[K|M]]
            [NEXT integer[K|M]]
            [MINEXTENTS integer]
            [MAXEXTENTS {integer|UNLIMITED}]
            [PCTINCREASE integer]
      )
```

Với:

tablespace	tên của tablespace được tạo
DATAFILE	tên data files của tablespace được tạo
DEFAULT STORAGE	tham số lưu trữ mặc định cho tất cả các đối tượng được tạo lập trong tablespace
MINIMUM EXTENT	kích thước tối thiểu của extent được sử dụng value
ONLINE	đặt chế độ sử dụng (Online) cho tablespace ngay từ khi tạo lập
OFFLINE	đặt chế độ chưa sử dụng (Offline) cho tablespace ngay từ khi tạo lập
PERMANENT	tablespace có thể sử dụng để lưu trữ các đối tượng thường trú
TEMPORARY	tablespace chỉ sử dụng để lưu trữ các đối tượng trung gian (temporary objects). Ví dụ: sử dụng

để lưu trữ dữ liệu khi sắp xếp theo câu lệnh ORDER BY

Ví dụ:

```
CREATE TABLESPACE app_data
      DATAFILE '/DISK4/app01.dbf' SIZE 100M,
      '/DISK5/app02.dbf' SIZE 100M
      MINIMUM EXTENT 500K
      DEFAULT STORAGE (INITIAL 500K NEXT 500K
      MAXEXTENTS 500 PCTINCREASE 0);
```

Cũng tương tự, ta có thể thực hiện trong Oracle Enterprise Manager – OEM:

1. Chạy Oracle Storage Manager.
2. Chọn Tablespace—>Create.
3. Trong General page của bảng thuộc tính, nhập vào tên tablespace rồi chọn ADD.
4. Trong bảng thuộc tính Create Datafile, chỉ ra các data file.
5. Trong phần Extents page, nhập vào các thông tin lưu giữ
6. Chọn mục Create.

Hạn chế

Số lượng tối đa các tablespaces trên mỗi database là 64.

Số lượng tối đa các data files trong mỗi tablespace là 1023.

9.6.2. Chế độ quản lý các tablespaces

Với câu lệnh tạo tablespace thông thường như ở trên, Oracle server sẽ tạo tablespace với chế độ quản lý là Dictionary-Managed Tablespaces

Để thực hiện quản lý tablespace theo phương pháp Locally-Managed Tablespaces ta cần đưa thêm vào câu lệnh mệnh đề: `MANAGEMENT LOCAL AUTOLOCATE`.

Ví dụ:

```
CREATE TABLESPACE lmtbsb
      DATAFILE 'c:\data\lmtbsb01.dbf' SIZE 50M
      EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

9.6.3. Tạo temporary tablespace

Quản trị viên database có thể tạo một temporary tablespace sử dụng cho việc sắp xếp các dữ liệu không dùng để lưu trữ thường trú các dữ liệu.

Để tạo temporary tablespace, ta có thể sử dụng lệnh SQL giống như lệnh tạo tablespace thông thường, nhưng có thêm từ khoá `TEMPORARY` ở cuối.

Ví dụ:

```
CREATE TABLESPACE sort
      DATAFILE '/DISK2/sort01.dbf' SIZE 50M
      MINIMUM EXTENT 1M
      DEFAULT STORAGE (INITIAL 2M NEXT 2M
      MAXEXTENTS 500 PCTINCREASE 0)
      TEMPORARY;
```

Với Oracle Enterprise Manager, ta làm theo các bước sau:

1. Chạy Oracle Storage Manager.
2. Chọn Tablespace—>Create.
3. Trong General page, nhập vào tên tương ứng rồi chọn ADD để hiển thị mục Create Datafile.
4. Trong Create Datafile chỉ ra từng data file cụ thể.
5. Chọn TEMPORARY trong nhóm chọn radio button.
6. Bấm nút Create.

9.6.4. Các tham số lưu trữ

Lượng không gian dùng cho một tablespace được xác định trong mệnh đề lưu trữ (storage clause). Các tham số này được xác định ngay tại thời điểm tạo tablespace. Trong trường hợp không chỉ rõ các tham số này trong lệnh tạo lập (CREATE), các tham số sẽ được sử dụng các giá trị theo mặc định.

Có một số tham số lưu trữ cần quan tâm sau:

- INITIAL quy định kích thước của extent đầu tiên. Kích thước nhỏ nhất của extent đầu tiên là 02 block = (2*DB_BLOCK_SIZE). Mặc định, kích thước này là 5 blocks = (5* DB_BLOCK_SIZE).
- NEXT ứng với kích thước của extent thứ hai. Kích thước tối thiểu là 01 block. Mặc định, kích thước này là 5 blocks = (5* DB_BLOCK_SIZE).
- MINEXTENTS số lượng extent được tạo lập mỗi khi segment được tạo lập. Mặc định giá trị này là 1.
- PCTINCREASE phần trăm tăng kích thước extent. Kích thước của một extent được xác định theo kích thước:

$$\text{Size}_n = \text{NEXT} \times \left(1 + \frac{\text{PCTINCREASE}}{100}\right)^{(n-2)}$$

  Với: Size_n kích thước của extent thứ n

  Ví dụ: NEXT = 200K, PCTINCREASE = 50. Ta tính được extent thứ hai = 200K, extent thư ba = 300K, extent thứ tư = 450K

- MAXEXTENTS xác định số lượng tối đa các extents có trong một segment. Giá trị nhỏ nhất là 1. Giá trị lớn nhất theo mặc định phụ thuộc vào kích thước của block

dữ liệu. Giá trị này cũng có thể được xác định thông qua giá trị UNLIMITED, tương đương với giá trị là 2147483645.

9.7.CÁC THAY ĐỔI ĐỐI VỚI TABLESPACE

9.7.1. Chuyển đổi một tablespace thành một temporary tablespace

Ta có thể thay đổi các tablespaces đang tồn tại để biến nó thành một temporary tablespace.

Ví dụ:

```
ALTER TABLESPACE tbsa TEMPORARY;
```

9.7.2. Thêm mới các tablespace

Để mở rộng không gian của tablespace ta có thể thực hiện theo hai cách sau:

- Thêm mới các data file vào tablespace
- Thay đổi dung lượng các data files

Hoặc ta cũng có thể sử dụng câu lệnh SQL can thiệp như sau:

```
ALTER TABLESPACE tablespace
      ADD DATAFILE filespec [autoextend_clause]
      [, filespec [autoextend_clause]]...
```

Với Oracle Enterprise Manager, ta làm theo các bước sau:

1. Chạy Oracle Storage Manager.
2. Chọn Tablespace—>Create.
3. Trong General page, nhập vào tên tương ứng rồi chọn ADD để hiển thị mục Create Datafile.
4. Trong Create Datafile chỉ ra từng data file cụ thể.

9.7.3. Mở rộng data files

Ta có thể thực hiện mở rộng (thay đổi) kích thước data file theo hai cách:

- Mở rộng theo chế độ tự động. Sử dụng từ khoá: AUTOEXTENTED
- Mở rộng theo chế độ can thiệp trực tiếp (manually). Sử dụng lệnh ALTER TABLESPACE, ALTER DATABASE

Thiết lập chế độ AUTOEXTENT trong khi tạo file

Cú pháp:

```
ALTER TABLESPACE tablespace
      ADD DATAFILE filespec [autoextend_clause]
      [, filespec [autoextend_clause]]...
```

Ví dụ:

```
ALTER TABLESPACE app_data
      ADD DATAFILE
      '/DISK6/app04.dbf' SIZE 200M
      AUTOEXTEND ON NEXT 10M
      MAXSIZE 500M;
```

Trong OEM ta thực hiện các bước sau:

1. Chạy Oracle Storage Manager.
2. Chuyển tới nút Tablespace.
3. ChọnTablespace—>Add Datafile.
4. Trong General page nhập vào các thông tin của file.
5. Trong Autoextend page nhập vào các thông tin tương ứng.
6. Bấm nút Create.

Thiết lập chế độ AUTOEXTENT khi data file đã tồn tại

Cú pháp:

```
ALTER DATABASE [database]
      DATAFILE 'filename'[, 'filename']...
      autoextend_clause
```

Trong OEM ta thực hiện các bước sau:

1. Chạy Oracle Storage Manager.
2. Chuyển tới nút Tablespace node.
3. Chọn data file.
4. Trong phần Autoextend page, bấm vào nút Enable Auto Extend.
5. Bấm nút Apply.

9.7.4. Thay đổi kích thước data file

Thay vì mở rộng kích thước của database bằng cách thêm vào các data file, quản trị viên cũng có thể mở rộng bằng cách điều chỉnh tăng kích thước của data file.

Sử dụng câu lệnh SQL sau để thay đổi kích thước của data file

```
ALTER DATABASE [database]
      DATAFILE 'filename'[, 'filename']...
      RESIZE integer[K|M]
```

Với:

```
integer          Kích thước tuyệt đối của file data file
```

Sử dụng câu lệnh SQL sau để thay đổi nơi lưu trữ mặc định:

```
ALTER TABLESPACE tablespace
      {MINIMUM EXTENT integer[K|M]
```

```
        |DEFAULT storage_clause
        }
```

Ví dụ:

```
ALTER TABLESPACE app_data
      MINIMUM EXTENT 2M;

ALTER TABLESPACE app_data
      DEFAULT STORAGE
      (INITIAL 2M NEXT 2M
      MAXEXTENTS 999);
```

9.7.5. Chuyển đổi chế độ ONLINE và OFFLINE

User chỉ có thể truy xuất vào tablespace nếu nó đang ở trạng thái online. Trong một vài trường hợp, quản trị viên database có thể thay đổi trạng thái database thành offline với mục đích:

- Di chuyển các data files tới vị trí khác
- Chỉ cho phép user truy xuất phần dữ liệu còn lại trong database.

Để chuyển đổi chế độ `ONLINE` và `OFFLINE`, ta có thể thực hiện câu lệnh SQL sau:

```
ALTER TABLESPACE tablespace
      {ONLINE
      |OFFLINE [NORMAL|TEMPORARY|IMMEDIATE]
      }
```

Chế độ OFFLINE

Oracle server không cho phép thực hiện câu lệnh SQL đối với các đối tượng có trong tablespace đã được `OFFLINE`.

Oracle server thực hiện checkpoint đối với tất cả các data files thuộc tablespace trước khi chuyển sang chế độ `OFFLINE`.

Mỗi khi database được mở, quản trị viên database có thể chuyển chế độ offline cho tất cả các tablespace ngoại trừ SYSTEM và các tablespace tương ứng với các active rollback segments hay temporary segments.

Trong OEM ta có thể thực hiện theo các bước sau:

1. Chạy Oracle Storage Manager.
2. Chuyển tới nút Tablespace.
3. Chọn tablespace tương ứng.
4. Trong General page, đặt chế độ trong Offline radio button.
5. Bấm nút Apply

9.7.6. Di chuyển các data file

Tuỳ thuộc kiểu tablespace, ta có thể di chuyển các data files theo các phương thức khác nhau.

Lệnh ALTER TABLESPACE

Lệnh này chỉ áp dụng cho các tablespace không phải là SYSTEM tablespace, và không chứa rollback segments hay temporary segments.

Câu lệnh:

```
ALTER TABESPACE tablespace
      RENAME DATAFILE 'filename'[, 'filename']...
      TO 'filename'[, 'filename']...
```

Ví dụ:

```
ALTER TABLESPACE app_data RENAME
      DATAFILE ‘/DISK4/app01.dbf’ TO
      ‘/DISK5/app01.dbf’;
```

Ta thực hiện theo các bước sau:

1. Chuyển chế độ offline cho tablespace.
2. Di chuyển các data files tương ứng bằng lệnh của hệ điều hành.
3. Thực hiện lệnh `ALTER TABLESPACE RENAME DATAFILE`.
4. Chuyển lại chế độ online cho tablespace đó.
5. Sử dụng lệnh của hệ điều hành để xoá data file cũ nếu cần thiết.

Lệnh ALTER DATABASE

Lệnh này chỉ áp dụng cho các tablespace không là SYSTEM và không chứa rollback segments hay temporary segments.

Câu lệnh:

```
ALTER DATABASE [database]
      RENAME FILE 'filename'[, 'filename']...
      TO 'filename'[, 'filename']...
```

Ví dụ:

```
ALTER DATABASE RENAME FILE
      ‘/DISK1/system01.dbf’ TO
      ‘/DISK2/system01.dbf’;
```

Ta thực hiện theo các bước sau:

1. Shutdown database.
2. Di chuyển data files bằng lệnh của hệ điều hành.
3. Mount lại database.
4. Thực hiện lệnh `ALTER DATABASE RENAME FILE`.
5. Mở lại database.

Trong OEM ta làm như sau

1. Chạy Oracle Storage Manager.
2. Chuyển tới nút Tablespace rồi chọn data file tương ứng.
3. Trong phần General page, thay đổi lại các thông tin thích hợp.
4. Bấm nút Apply.

9.7.7. Tablespace chỉ đọc

Sử dụng lệnh SQL để thiết lập các chế độ này.

Cú pháp:

```
ALTER TABLESPACE tablespace
      READ{ONLY|WRITE}
```

Ví dụ:

```
ALTER TABLESPACE app_data READ ONLY;
```

Trong OEM ta thực hiện theo các bước sau:

1. Chạy Oracle Storage Manager.
2. Chuyển tới nút Tablespace.
3. Chọn tablespace.
4. Chọn Tablespace—>Make Read Only.
5. Bấm nút OK.

Thiết lập chế độ chỉ đọc cho tablespace

Ta có thể thiết lập chế độ chỉ đọc cho tablespace khi nó đảm bảo một số điều kiện sau:

- Tablespace phải đang online
- Không có transaction nào xảy ra đối với tablespace đó
- Tablespace không chứa các rollback segments
- Hiện thời không có online backup trên tablespace

9.7.8. Huỷ tablespace

Trong một vài trường hợp ta có thể huỷ tablespace khỏi database.

Việc này có thể thực hiện bởi câu lệnh SQL sau:

```
DROP TABLESPACE tablespace
      [INCLUDING CONTENTS [CASCADE CONSTRAINTS]]
```

Với

```
tablespace tên của tablespace được huỷ
INCLUDING CONTENTS
```

huỷ luôn các segment có trong tablespace

CASCADE CONSTRAINTS

Huỷ luôn cả các ràng buộc liên quan tới các bảng bên ngoài có tham chiếu duy nhất tới các bảng thuộc tablespace bị huỷ

Ví dụ:

```
DROP TABLESPACE app_data
     INCLUDING CONTENTS;
```

Trong OEM ta thực hiện theo các bước sau

1. Chạy Oracle Storage Manager.
2. Chuyển tới nút Tablespace chọn tablespace tương ứng.
3. Chọn Tablespace—>Remove.
4. Bấm nút OK.

9.8.THÔNG TIN VỀ CÁC TABLESPACES

Một số views thông tin chung

Tên tham số	Diễn giải
DBA_TABLESPACES, USER TABLESPACES	Diễn giải của các tablespaces.
DBA_SEGMENTS, USER_SEGMENTS	Thông tin về segment có trong các tablespaces.
DBA_EXTENTS, USER_EXTENTS	Thông tin về data extents có trong các tablespaces.
DBA_FREE_SPACE, USER_FREE_SPACE	Thông tin về free extents có trong các tablespaces.
V$DATAFILE	Thông tin về tất cả các datafiles, bao gồm cả số hiệu tablespace và user sở hữu tablespace.
V$TEMPFILE	Thông tin về các tempfiles, bao gồm cả số hiệu tablespace và user sở hữu tablespace.
DBA_DATA_FILES	Hiển thị các datafiles thuộc các tablespaces.
DBA_TEMP_FILES	Hiển thị các tempfiles thuộc các temporary tablespaces.
V$TEMP_EXTENT_MAP	Thông tin của các extents trong các locally managed temporary tablespaces.
V$TEMP_EXTENT_POOL	Thông tin của các locally managed temporary tablespaces bao gồm: trạng thái của temporary space cached (vùng không gian đệm trung gian) được sử dụng bởi mỗi instance.

V$TEMP_SPACE_HEADER	Hiển thị vùng không gian used/free của mỗi tempfile.
DBA_USERS	Các tablespaces mặc định và temporary tablespaces của các users.
DBA_TS_QUOTAS	Hạn mức sử dụng tablespace của các users.
V$SORT SEGMENT	Thông tin về sort segment đối với mỗi instance.
V$SORT_USER	Vùng không gian sắp xếp trung gian được sử dụng bởi user và temporary/permanent tablespace.

9.8.1. Xem thông tin tablespace

Để xem thông tin về tablespace, ta có thể lấy trong data dictionary views. View DBA_TABLESPACES lưu trữ các thông tin này.

Một số thông tin quan tâm:

Tên tham số	Diễn giải
TABLESPACE_NAME	Tên tablespace
NEXT_EXTENT	Kích thước của các extent mở rộng tính theo bytes
MAX_EXTENTS	Số lượng tối đa các extents trong một segment
PCT_INCREASE	Phần trăm tăng trưởng kích thước của các extents
MIN_EXTENTS	Số lượng tối thiểu các extents trong một segment
STATUS	Trạng thái của tablespace là Online hay Offline
CONTENTS	Phân loại tablespace là permanent hay temporary

Ví dụ:

```
SVRMGR> SELECT tablespace_name, initial_extent, next_extent,
2 > max_extents, pct_increase, min_extlen
3 > FROM dba_tablespaces;

TABLESPACE_NAME  INITIAL_EX NEXT_EXT  MIN_EXTENT MAX_EXTENT PCT_I  MIN_EXTLEN
---------------  ---------- --------  ---------- ---------- -----  ---------
SYSTEM           1240       10240     1          121        50     0
RBS              10240      10240     1          121        50     0
TEMP             262144     262144    1          999        50     131072
DATA01           204800     204800    1          999        50     51200
4 rows selected.


SVRMGR> SELECT tablespace_name, contents,status
     2> FROM dba_tablespaces;
```

```
TABLESPACE_NAME    CONTENTS    STATUS
---------------    ---------   ------
SYSTEM             PERMANENT   ONLINE
RBS                PERMANENT   ONLINE
TEMP               TEMPORARY   ONLINE
DATA01             PERMANENT   ONLINE
4 rows selected.
```

9.8.2. Xem thông tin data files

Để xem thông tin về data files, ta có thể lấy trong dictionary views. View DBA_DATA_FILES lưu trữ các thông tin này.

Một số thông tin quan tâm:

Tên tham số	Diễn giải
FILE_NAME	Tên file (có kèm đường dẫn) tương ứng với datafile
TABLESPACE_NAME	Tên của tablespace ứng với datafile đó
BYTES	Dung lượng tính theo bytes của tablespace hiện thời
AUTOEXTENSIBLE	Chế độ tự động mở rộng dung lượng của datafile
MAXBYTES	Dung lượng tối đa
INCREMENT_BY	Chỉ số tăng tự động trong hệ thống

Ví dụ:

```
SVRMGR> SELECT file_name, tablespace_name, bytes,
2> autoextensible, maxbytes, increment_by
3> FROM dba_data_files;

FILE_NAME            TABLESPACE_NAME  BYTES     AUT   MAXBYTES  INCREMENT_BY
---------            ---------------  --------  ----  --------  ----------
/DISK1/system01.dbf  SYSTEM           31457280  NO    0         0
/DISK2/rbs01.dbf     RBS              5242880   NO    0         0
/DISK3/temp01.dbf    TEMP             5242880   NO    0         0
/DISK4/data01.dbf    DATA01           5242880   NO    0         0
/DISK5/data02.dbf    DATA01           512000    YES   15728640  512
5 rows selected.
```

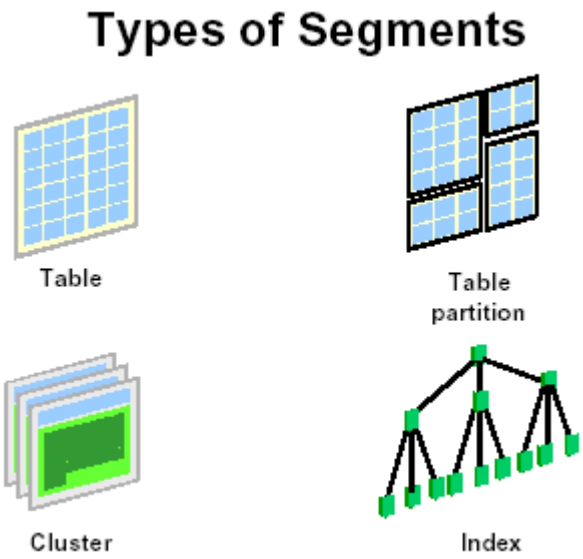
Chương 10. CẤU TRÚC LƯU TRỮ

10.1.CÁC LOẠI SEGMENTS

Segments là các vùng không gian của các objects (đối tượng) trong database. Dưới đây, ta sẽ xem xét một số loại segments cụ thể.

10.1.1. Table

Table (bảng), là nơi lưu giữ dữ liệu trong database. Dữ liệu trong một table được lưu giữ không theo một thứ tự bắt buộc. Các dữ liệu trong một table thuộc loại nonpartitioned (không phân khu) sẽ phải lưu giữ trong cùng một tablespace.



Hình vẽ 37. Các loại segments

10.1.2. Table partition

Có thể có một số table trong database có số lượng truy cập lớn và đồng thời. Khi đó, dữ liệu trong table đó sẽ được lưu thành nhiều partition (phân khu), mỗi partition có thể nằm trên các tablespace khác nhau. Oracle server hỗ trợ việc phân chia này bằng các giá trị khoá. Khi một table được phân khu, mỗi partition đó được xem như một segment.

10.1.3. Cluster

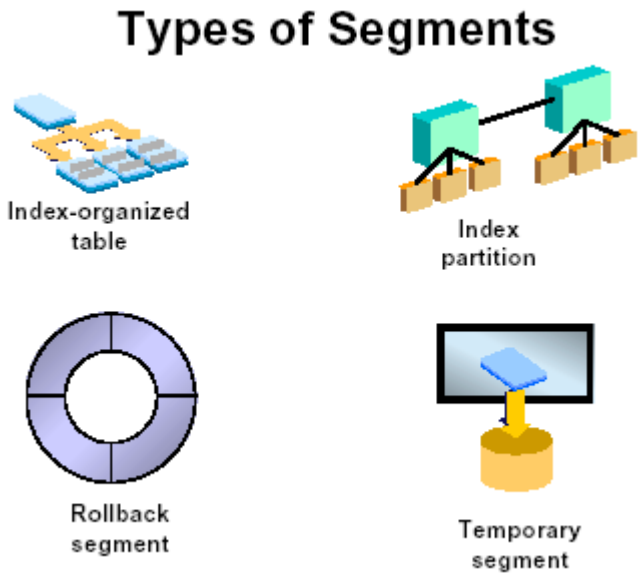
Các dòng dữ liệu trong một cluster được lưu trữ theo các giá trị của trường khoá (key column). Một cluster có thể chứa một hay nhiều tables và nó được xem là một kiểu đoạn dữ liệu (type of data segment). Các tables trong một cluster thuộc về cùng một đoạn và có chung các tính chất lưu trữ.

10.1.4. Index

Tất cả các đầu mục (entries) ứng với một index cụ thể được lưu trữ trong một index segment. Một table có tới bao nhiêu indexes, thì sẽ có bấy nhiêu index segments được sử dụng. Mục đích của segment này là tìm kiếm và định vị các dòng dữ liệu trong một table dựa trên một khoá được chỉ ra.

10.1.5. Index-Organized Table

Trong một index-organized table, các dữ liệu trong một index được lưu trữ dựa vào giá trị khoá. Một index-organized table không cần thiết đến một table dùng để tìm kiếm (lookup), các dữ liệu có thể được trả về ngay trực tiếp từ cây index (index tree).



Hình vẽ 38. Các loại segments (tiếp theo)

10.1.6. Index Partition

Một index có thể được partitioned (phân khu) và trải rộng trên nhiều tablespaces khác nhau. Khi đó, mỗi partition của một index sẽ tương ứng với segment (đoạn) và không được phép nằm dài trên nhiều tablespaces. Mục đích chính của việc sử dụng index partition là để giảm thiểu những tranh chấp vào ra I/O.

10.1.7. Rollback Segment

Rollback segment được sử dụng trong transaction (giao dịch) để tạo các thay đổi trong database. Trước khi thay đổi các dữ liệu hay các index blocks, các giá trị cũ sẽ được lưu giữ vào rollback segment. Việc làm này cho phép user có thể phục hồi lại các thay đổi.

10.1.8. Temporary Segment

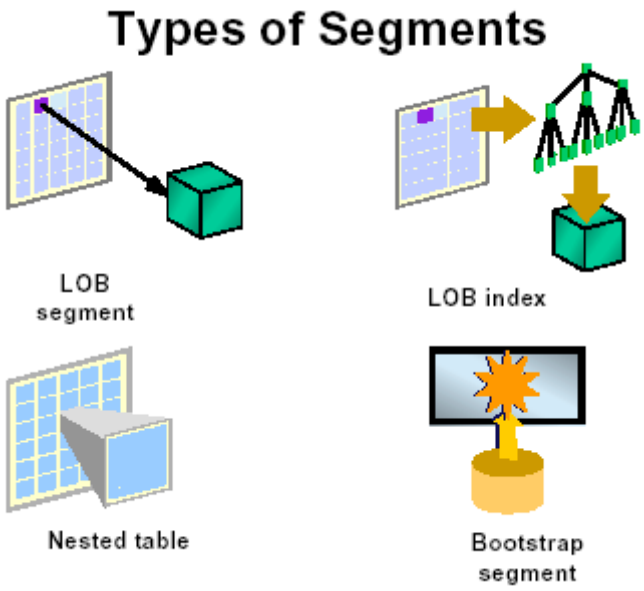
Khi một user thực hiện các lệnh như `CREATE INDEX`, `SELECT DISTINCT`, và `SELECT GROUP BY`, Oracle sẽ cố gắng thực hiện công việc sắp xếp ngay trong bộ nhớ. Khi công việc sắp xếp cần đến nhiều không gian hơn, các kết quả này sẽ được ghi trực tiếp lên đĩa. Temporary segments sẽ được dùng đến trong trường hợp này.

10.1.9. LOB Segment

Khi một hay nhiều cột trong table lưu giữ các đối tượng lớn (large objects - LOBs) như các văn bản tài liệu, hình ảnh, hay videos. Các cột chứa dữ liệu lớn này sẽ được Oracle server lưu giữ trong các segments riêng được biết đến như là LOB segments. Table sẽ chỉ lưu giữ các giá trị dùng để định vị, xác định nơi lưu giữ các dữ liệu LOB tương ứng.

10.1.10. LOB Index

Một LOB index segment được tạo ngầm định mỗi khi LOB segment được tạo lập. Các tính chất lưu giữ của LOB index có thể được quy định bởi quản trị viên database. Mục đích của việc sử dụng LOB index segment là cho phép tìm kiếm các giá trị cụ thể trong cột dữ liệu loại LOB.



Hình vẽ 39. Các loại segments (tiếp theo)

10.1.11. Nested Table

Cột dữ liệu trong table có thể được tạo lập từ một user-defined table (bảng do người dùng định nghĩa). Trong trường hợp này, bảng dữ liệu tương ứng với phần tử thuộc cột dữ liệu (inner table), được biết đến như một nested table và được lưu giữ trong một segment riêng biệt.

10.1.12. Bootstrap Segment

Bootstrap segment, được biết đến như một cache segment, được tạo bởi file script *sql.bsq* sau mỗi khi database được tạo. Segment giúp cho việc khởi tạo data dictionary cache mỗi khi database được mở bởi một instance. Dữ liệu trong bootstrap segment không thể xem hay sửa

chữa, cập nhật được. Quản trị database cũng không cần thiết phải quan tâm tới segment này.

10.2.QUẢN LÝ EXTENTS

10.2.1. Cấp phát và thu hồi các extents

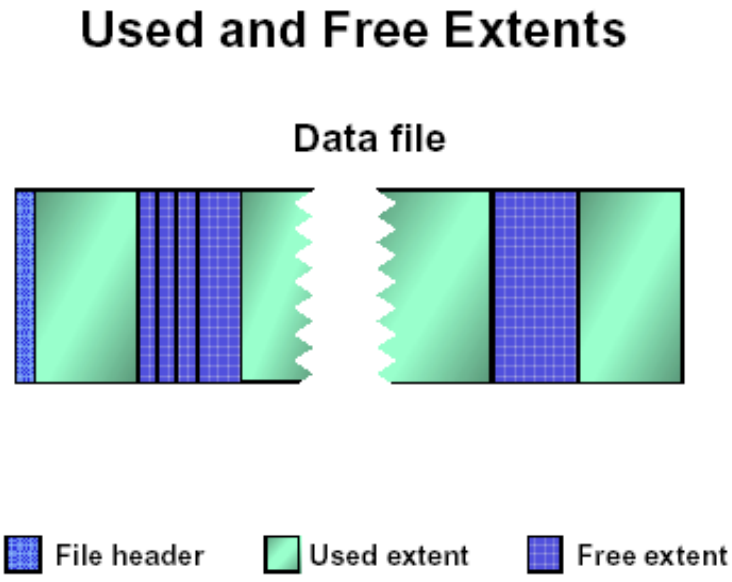
Việc cấp phát các extent xảy ra mỗi khi segment được tạo mới, được mở rộng hay bị thay đổi (altered).

Và nó sẽ bị thu hồi khi segment bị huỷ, bị thay đổi, bị cắt bớt (truncated). Riêng đối với các rollback segments, các extent có thể bị tự động thu hồi.

10.2.2. Sử dụng và giải phóng các extent

Khi một tablespace được tạo, các data files thuộc tablespace sẽ chứa các phần thông tin sau:

- Header block, tương ứng với block đầu tiên của file
- Phần còn lại của data file là các phần còn trống



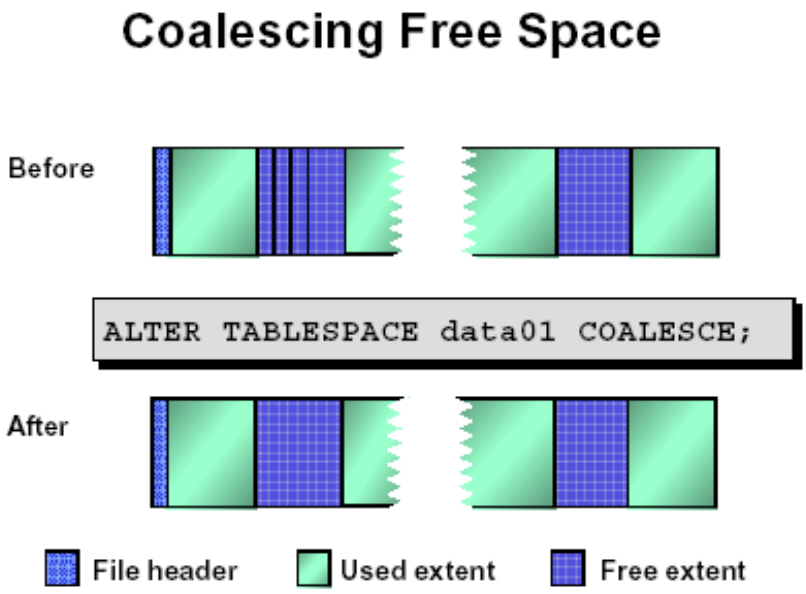
Hình vẽ 40. Sử dụng và giải phóng các extents

Mỗi khi segments được tạo lập, nó sẽ được cấp phát một vùng không thích hợp từ những extents còn trống trong tablespace. Segment sẽ cố gắng sử dụng nhiều nhất các vùng không gian liên tiếp nhau. Sau khi cấp phát, extent đó sẽ được xem là *used extent* (extent đã được sử dụng). Khi các segments giải phóng vùng không gian, các extents tương ứng với nó sẽ được giải phóng và đưa vào vùng free extents (extents rỗi) của tablespace. Với việc cấp phát và giải phóng các extents có thể gây nên hiện tượng phân đoạn vùng dữ liệu trong các data files của tablespace.

10.2.3. Kết hợp các vùng không gian trống

Ta có thể thực hiện việc kết hợp các vùng không gian trống liên tiếp nhau mỗi khi các extents trong cùng một tablespace được giải phóng. Điều này rất dễ xảy ra, ví dụ: khi có hai table bị huỷ (dropped). Các extents trống này có thể được kết hợp lại thành một extent trong các điều kiện:

- Khi tiến trình SMON khởi tạo một *space transaction* để kết hợp các extents trống.
- Khi Oracle server cần phải cấp phát vùng trống mà nó cần tới lượng không gian trống lớn hơn không gian của một extent.
- Kết hợp theo yêu cầu của quản trị viên database.



Hình vẽ 41. Kết hợp các vùng không gian trống

Lưu ý

Tiến trình SMON sẽ chỉ kết hợp các extent trong cùng tablespaces khi mà PCTINCREASE lớn hơn 0. Trong storage clause mặc định của tablespaces, đặt PCTINCREASE=1 khi đó các user objects có thể được tự động kết hợp các vùng trống mỗi khi nó được giải phóng.

Yêu cầu kết hợp vùng trống

View DBA_FREE_SPACE_COALESCED được dùng để xem tablespace nào có các extents rỗng có thể kết hợp được với nhau. Sử dụng câu lệnh truy vấn sau đây để lấy các thông tin:

```
SVRMGR> SELECT tablespace_name, total_extents,
2> percent_extents_coalesced
3> FROM dba_free_space_coalesced
4> WHERE percent_extents_coalesced <> 100;
TABLESPACE_NAME TOTAL_EXTE PERCENT_EX
--------------- ---------- ----------
RBS                      3         33
```

```
DATA01                9          22
2 rows selected.
```

Thực hiện kết hợp các vùng không gian trống trong tablespace bằng lệnh dưới đây:

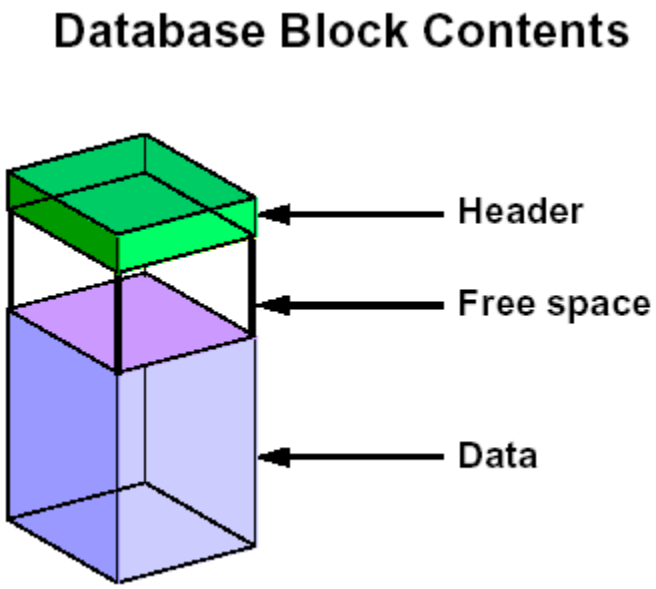
```
ALTER TABLESPACE tablespace COALESCE;
```

Trong OEM, ta thực hiện theo các bước sau

1. Sử dụng công cụ Oracle Tablespace Manager.
2. Chuyển tới nút Expand Tablespaces.
3. Chọn tablespace tương ứng.
4. Chọn mục Tools—>Coalesce Free Extents.

10.3.BLOCK DỮ LIỆU

10.3.1. Cấu trúc của block dữ liệu



Hình vẽ 42. Cấu trúc của Block dữ liệu

Các Blocks dữ liệu của Oracle được cấu thành từ các phần sau:

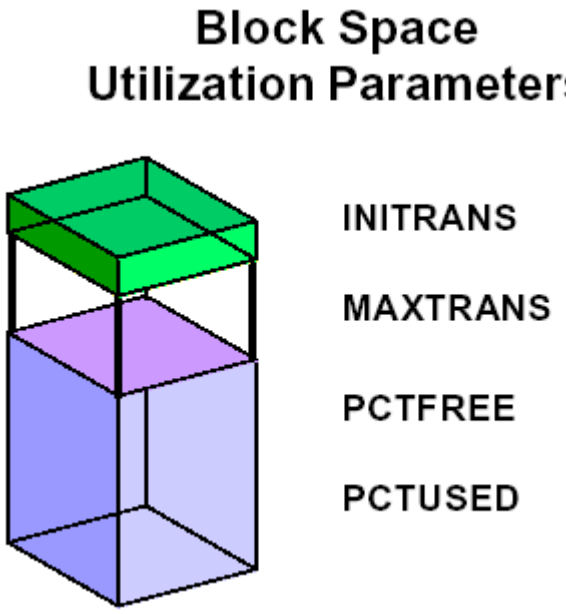
- Block header (vùng đầu): Header chứa địa chỉ của block dữ liệu, thông tin về table directory, row directory, và các transaction slots. Thông tin trong Block headers sẽ tăng dần theo hướng từ trên xuống dưới.
- Data space (vùng dữ liệu): Các dòng dữ liệu được nạp vào block theo hướng từ dưới lên.
- Free space (vùng trống): Vùng trống trong block là vùng nằm giữa vùng header và vùng không gian lưu trữ dòng dữ liệu. Ban đầu, vùng không gian trống là liên tiếp với

nhau.Tuy nhiên sau một thời gian sử dụng, vùng không gian trống trong một block có thể bị phân đoạn do việc xoá và cập nhật, thay đổi các dòng dữ liệu. Để giải quyết vấn đề này, Oracle server cho phép thực hiện kết hợp các phân đoạn dữ liệu.

10.3.2. Các tham số sử dụng không gian trong block

Các tham số sử dụng không gian trong block được dùng để điều khiển việc sử dụng vùng không gian dữ liệu và index trong các segments.

Các tham số điều khiển song song



Hình vẽ 43. Các tham số sử dụng không gian trong block

Các tham số `INITRANS` và `MAXTRANS` chỉ ra số lượng khởi tạo, số lượng lớn nhất các transaction slots, được tạo trong mỗi index block hay data block. Các transaction slots được

sử dụng để lưu giữ các thông tin về các transactions làm thay đổi các block tại cùng một thời điểm. Mỗi transaction chỉ sử dụng một transaction slot.

INITRANS được gán giá trị mặc định bằng 1 cho data segment, và 2 cho index segment.

MAXTRANS được gán giá trị mặc định là 255, dùng để tạo ngưỡng đối với các transactions đồng thời có làm thay đổi các block dữ liệu hay index block. Khi thiết lập giá trị này, vùng không gian cho các transaction slots sẽ được đảm bảo để có thể thực hiện các transaction một cách hiệu quả.

Tham số điều khiển vùng lưu trữ dữ liệu

PCTFREE trong một data segment chỉ lượng phần trăm vùng trống trong mỗi data block để dành cho việc tăng lên của dữ liệu do việc cập nhật các dòng dữ liệu trong block. Theo mặc đinh, PCTFREE là 10 phần trăm.

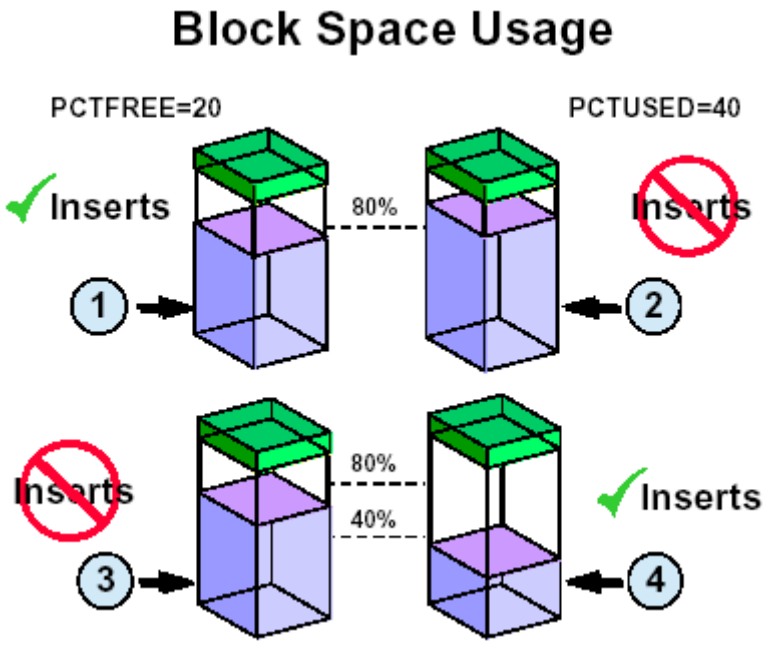
PCTUSED trong một data segment chỉ lượng phần trăm tối thiểu của vùng không gian sử dụng, theo đó Oracle Server lưu giữ các block dữ liệu của table. Một block sẽ được nạp lại vào free list (danh sách trống) mỗi khi PCTUSED giảm xuống. Free list của một segment là danh sách các blocks sẵn dùng cho việc cấp phát mỗi khi có dòng dữ liệu được insert. Theo mặc định mỗi free list sẽ được tạo tương ứng với mỗi segment. Tham số FREELISTS xác định số lượng free list. Mặc định, PCTUSED bằng 40 phần trăm.

PCTFREE và PCTUSED được tính toán theo phần trăm vùng không gian của dữ liệu, tức là vùng không gian của Block còn lại trừ đi vùng không gian header.

10.3.3. Sử dụng không gian trong block

Để cụ thể, ta theo dõi các bước việc sử dụng các vùng không gian trong block đối với một table có PCTFREE=20 và PCTUSED=40:

- **Phase 1**: Các dòng dữ liệu được nạp vào block cho tới khi đủ 80% (100-PCTFREE). Lúc này, ta không thể insert thêm dữ liệu vào Block.



Hình vẽ 44. Sử dụng vùng không gian trong block

- **Phase 2**: 20% không gian còn lại sử dụng cho việc tăng kích thước của các dòng dữ liệu do việc cập nhật lại các dòng dữ liệu này.
- **Phase 3**: Khi xoá dòng dữ liệu trong block, vùng không gian trống trong block sẽ tăng lên. Tuy nhiên, lúc này ta vẫn chưa thể insert dữ liệu vào block được.
- **Phase 4**: Khi vùng trống trong block đạt tới mức PCTUSED, ta lại có thể insert dữ liệu vào Block. Ta lại bắt đầu từ bước 01.

10.3.4. Phân loại mức độ phân đoạn đối với từng loại segment

Tablespace	Phân loại sử dụng	Mức độ phân đoạn
SYSTEM	Data dictionary	Không xảy ra
TOOLS	Applications	Rất ít
DATAn	Data segments	Ít
INDEXn	Index segments	Ít
RBSn	Rollback segments	Nhiều
TEMPn	Temporary segments	Rất nhiều*

Ký hiệu * có nghĩa là chỉ đúng với các tablespaces thuộc loại PERMANENT

Hiện tượng phân đoạn dữ liệu xảy ra với mức độ khác nhau đối với các loại segments khác nhau. Oracle khuyến cáo nên lưu trữ dữ liệu trên nhiều tablespaces khác nhau để giảm thiểu việc sử dụng lãng phí các vùng không gian.

Phân loại các Objects và phân đoạn

Các loại objects khác nhau được liệt kê dưới đây theo mức độ tăng dần về phân đoạn:

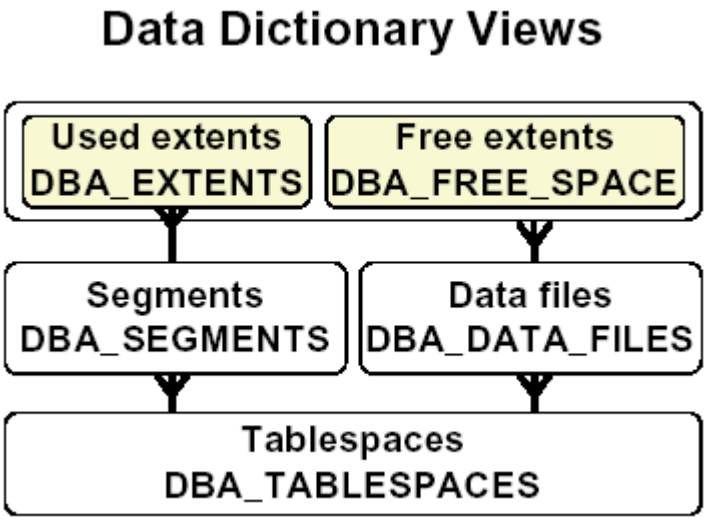
- Các data dictionary objects, ngoại trừ các audit table (bảng kiểm tra), đều không bao giờ bị dropped hay truncated. Vì thế chúng không bị phân đoạn trong tablespace.
- Vùng không gian sử dụng cho việc lưu trữ các ứng dụng luôn được cấp phát và thu hồi trong quá trình tái cấu trúc lại bộ nhớ. Vì thế, các tables lưu trữ này có mức độ phân đoạn là thấp.
- Data segment và index segements được sử dụng cho việc lưu trữ dữ liệu người dùng thuộc các ứng dụng. Các đối tượng này thường có mức độ phân đoạn cao.
- Do các rollback segments được phân bổ lại extents một cách tự động, chúng dễ gây ra hiện tượng phân đoạn dữ liệu trong hệ thống.
- Temporary segments trong các permanent tablespaces thường xuyên được bị xảy ra hiện tượng phân đoạn.

10.4.THÔNG TIN VỀ CẤU TRÚC LƯU TRỮ

10.4.1. Các view lưu trữ thông tin

Thông tin về các tablespaces, data files, segments, và extents (thông tin về cả phần sử dụng lần phần còn trống) đều có thể lấy từ các từ điển dữ liệu.

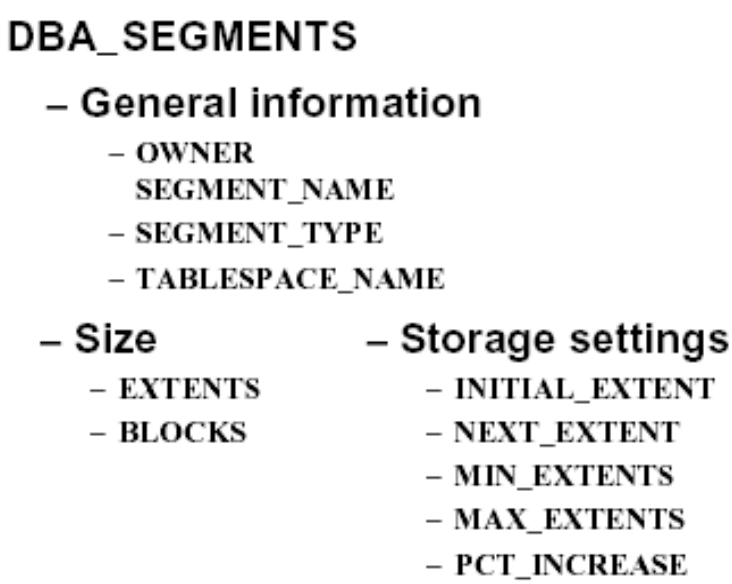
Thông tin về tablespace có thể được lưu trong `DBA_TABLESPACES`. Thông tin về các file dữ liệu của database được lưu trong `DBA_DATA_FILES`. Thông tin về các vùng trống trong các data file, vùng trống của extent được lưu trong `DBA_FREE_SPACE`. View `DBA_SEGMENTS` lưu giữ thông tin về các segment. Tương tự như vậy, `DBA_EXTENTS` lưu giữ thông tin về các extent.



Hình vẽ 45. Các views chứa thông tin về cấu trúc lưu trữ

10.4.2. Xem thông tin về các segments

Thông tin được lưu trong DBA_SEGMENTS.



Hình vẽ 46. Phân loại các thông tin chính có trong DBA_SEGMENTS

Ta có thể lấy thông tin về các segments theo các loại sau:

- Thông tin tổng hợp: User sở hữu, tên segment, loại segment, tên tablespace.
- Thông tin về kích cỡ: extents, blocks.
- Thông tin lưu trữ: INITIAL_EXTENT, NEXT_EXTENT, MIN_EXTENT, MAX_EXTENT, PCT_INCREASE

Ví dụ: Xem số lượng các extents và blocks được cấp phát cho từng segment do user SCOTT sở hữu.

```
SVRMGR> SELECT segment_name,tablespace_name,extents,blocks
2> FROM dba_segments
3> WHERE owner='SCOTT';

SEGMENT_NAME       TABLESPACE_NAME EXTENTS    BLOCKS
------------       --------------- --------- --------
EMP                DATA01           5         55
DEPT               DATA01           1         5
BONUS              DATA01           1         5
SALGRADE           DATA01           1         5
DUMMY              DATA01           1         5
5 rows selected.
```

10.4.3. Thông tin về các extents

Thông tin được lưu trong DBA_EXTENTS.

Hình vẽ 47. Phân loại các thông tin chính có trong DBA_EXTENTS

Ta có thể lấy thông tin về các extents theo các loại sau:

- Thông tin nhận dạng: User sở hữu, tên segment, mã hiệu extent
- Thông tin về kích cỡ và nơi đặt: TABLESPACE_NAME, RELATIVE_FNO, FILE_ID, BLOCK_ID, BLOCKS

Ví dụ: Xem thông tin chi tiết về các extents có trong một segment cho trước

```
SVRMGR> SELECT extent_id,file_id,block_id,blocks
2> FROM dba_extents
3> WHERE owner='SCOTT'
4> AND segment_name='EMP';

EXTENT_ID       FILE_ID  BLOCK_ID  BLOCKS
---------       -------  --------  ------
0               4        2         5
1               4        27        5
2               4        32        10
3               4        42        15
4               4        57        20
5 rows selected.
```

10.4.4. Thông tin về các vùng trống

Thông tin về các vùng trống được lưu trong DBA_FREE_SPACE.

Hình vẽ 48. Phân loại các thông tin chính có trong DBA_FREE_SPACE

View này chứa các thông tin về

Ví dụ:

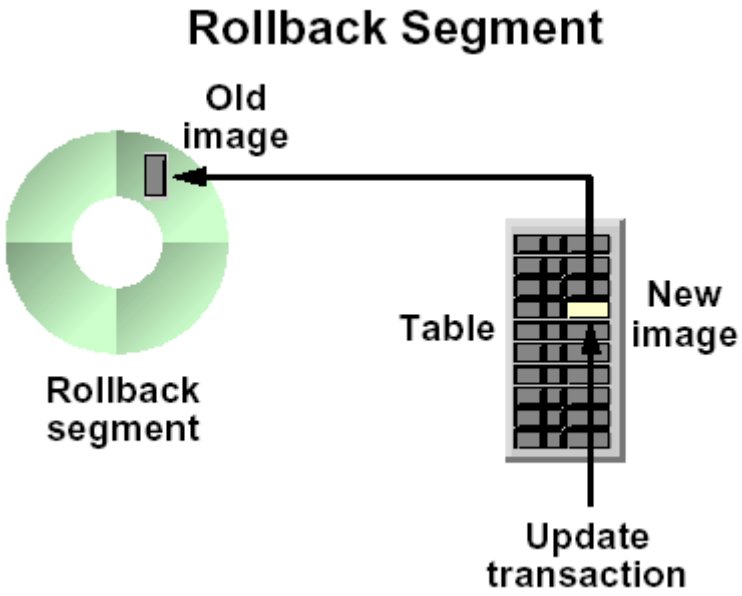
```
SVRMGR> SELECT tablespace_name, count(*),
2> max(blocks), sum(blocks)
3> FROM dba_free_space
4> GROUP BY tablespace_name;
TABLESPACE_NAME COUNT(*) MAX(BLOCKS SUM(BLOCKS
--------------- -------- ---------- ----------
DATA01                 2       1284       1533
RBS                    3       2329       2419
SORT                   1       1023       1023
SYSTEM                 1       5626       5626
TEMP                   1       2431       2431
5 rows selected.
```

Chương 11. QUẢN LÝ ROLLBACK SEGMENTS

11.1.GIỚI THIỆU ROLLBACK SEGMENTS

11.1.1. Khái niệm

Mỗi khi có sự thay đổi dữ liệu trong database, các dữ liệu cũ đều được lưu lại để có thể khôi phục lại trạng thái của dữ liệu trước khi thay đổi. Rollback segment được dùng để lưu trữ các giá trị cũ đó. Rollback segment lưu giữ các thông tin về block như block ID, và các dữ liệu đã sửa đổi của block.



Hình vẽ 49. Rollback segment

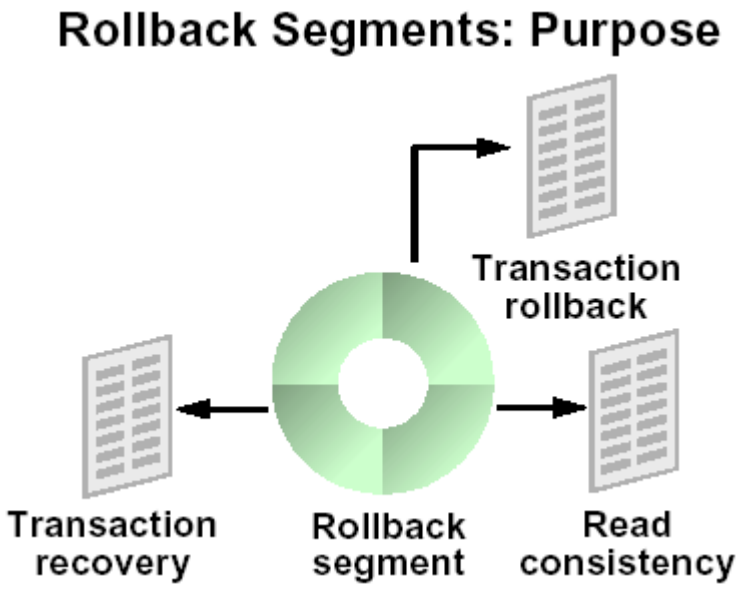
Phần đầu (header) của một rollback segment chứa một transaction table là nơi lưu giữ thông tin về các giao dịch hiện thời có sử dụng tới rollback segment đang xem xét. Mỗi transaction chỉ có thể sử dụng duy nhất một rollback segment để lưu giữ các dữ liệu dùng để khôi phục.

Nhiều transactions có thể đồng thời ghi dữ liệu lên rollback segment.

11.1.2. Mục đích sử dụng segment

Transaction rollback

Khi một transaction thực hiện các thay đổi đối với các dòng dữ liệu trong table, các giá trị ban đầu (old image) sẽ được lưu giữ vào rollback segment. Khi transaction đó được rolled back (lấy lại), các dữ liệu cũ lưu trong rollback segment sẽ được lấy ra và đè lên dữ liệu hiện tại trong block, phục hồi lại các giá trị nguyên thuỷ.



Hình vẽ 50. Mục đích của rollback segment

Phục hồi các Transaction

Trong trường hợp một instance gặp lỗi khi các transactions đang thực hiện, Oracle server cần phải khôi phục lại các dữ liệu chưa commit. Rollback trong trường hợp này được gọi là phục hồi dữ liệu. Việc này chỉ thực hiện được khi các thay đổi đối với các rollback segments đã được kết hợp bảo vệ bởi các redo log files.

Nhất quán trong việc đọc dữ liệu

Khi một thực hiện các transactions, các users trong database sẽ không thể thấy được các dữ liệu đã bị thay đổi mà chưa được commit bởi transactions. Các dữ liệu cũ lưu trong rollback segments sẽ vẫn được sử dụng để cung cấp cho các users khác nhằm đảm bảo nhất quán dữ liệu cho các user đó.

11.1.3. Phân loại rollback segment

SYSTEM Rollback Segment

SYSTEM rollback segment được tạo ngay trong SYSTEM tablespace mỗi khi một database được tạo lập. Rollback segment này chỉ được sử dụng đối với các thay đổi dữ liệu của các đối tượng nằm trong SYSTEM tablespace.

Non-SYSTEM Rollback Segments

Một database có thể có nhiều tablespaces và nên có ít nhất một non-SYSTEM rollback segment. Các non-SYSTEM rollback segment do quản trị viên database tạo lập có thể được sử dụng để lưu giữ các thay đổi trên các đối tượng có trong các non-SYSTEM tablespace khác. Có hai loại non-SYSTEM rollback segments.

- Private: Private rollback segments là các segments được sử dụng riêng cho mỗi instance.
- Public: Public rollback segments là một phần của rollback segments có trong database. Public rollback segments có thể được sử dụng bởi Oracle Parallel Server.

11.2.SỬ DỤNG ROLLBACK SEGMENT

11.2.1. Sử dụng rollback segment trong các transaction

Cấp phát các Rollback Segment

Đối với các transaction phải xử lý một khối lượng lớn các dữ liệu, ta cần gán transaction này với một rollback segment riêng chuyên làm nhiệm vụ lưu giữ các trạng thái ban đầu của dữ liệu.

Chú ý gán rollback segments cho một transaction:

- Lường trước khối lượng thông tin trong transaction cần rollback phù hợp (fit) với kích thước của vùng trống (extents) hiện thời của rollback segment.
- Cấp phát vừa đủ các vùng trống và không cần cấp phát bổ các vùng trống (extents) cho rollback segments đã được gán cho transaction vì điều này có thể dẫn đến việc giảm hiệu suất thực hiện của hệ thống.

Để gán một transaction cho một rollback segment một cách tường minh thì rollback segment đó cần phải đang ở trạng thái online. Cần thực lệnh lệnh `SET TRANSACTION USE ROLLBACK SEGMENT` trước khi thực hiện các lệnh trong transaction đó. Nếu trạng thái của rollback segment là offline hoặc câu lệnh `SET TRANSACTION USE ROLLBACK SEGMENT` không được đặt ở vị trí đầu tiên của transaction thì hệ thống sẽ phát sinh một lỗi.

Ví dụ: sử dụng lệnh gán rollback segment cho transaction tại thời điểm bắt đầu transaction:

```
SET TRANSACTION USE ROLLBACK SEGMENT large_rs1;
```

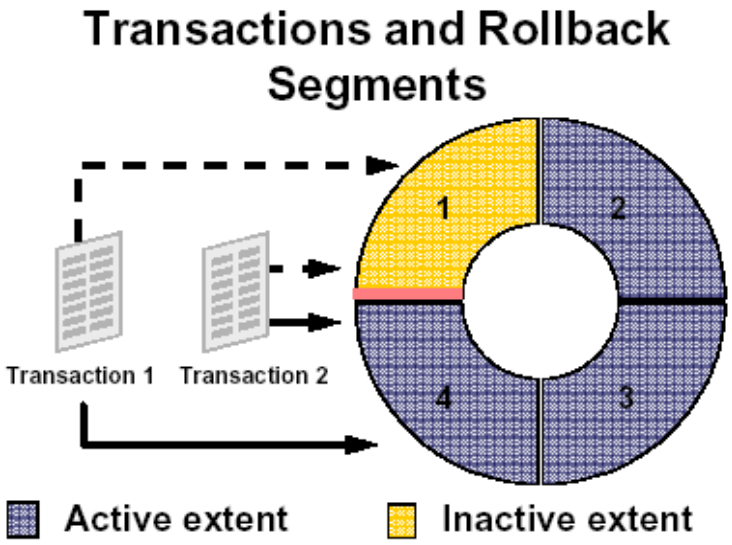
Sau khi transaction được commit, rollback segment này lại được Oracle đưa về trạng thái sẵn sàng sử dụng. Oracle sẽ tự động gán transaction tiếp theo cho một rollback segment bất kỳ nào đang còn rỗi (available) trừ phi transaction này lại được tiếp tục gán cho rollback segment bằng tay bởi user.

Sử dụng các extents

Các transactions sử dụng extents trong rollback segment theo một trình tự xoay vòng. Theo đó, transaction sẽ ghi dữ liệu thay đổi vào extent hiện thời, rồi tiếp tục chuyển tới các extent kế tiếp. Khi extent cuối cùng được sử dụng đầy, nó lại quay trở về extent 1.

Để rõ hơn, ta xem xét một ví dụ sau:

Có hai transaction cùng sử dụng một rollback segments có 04 extents.

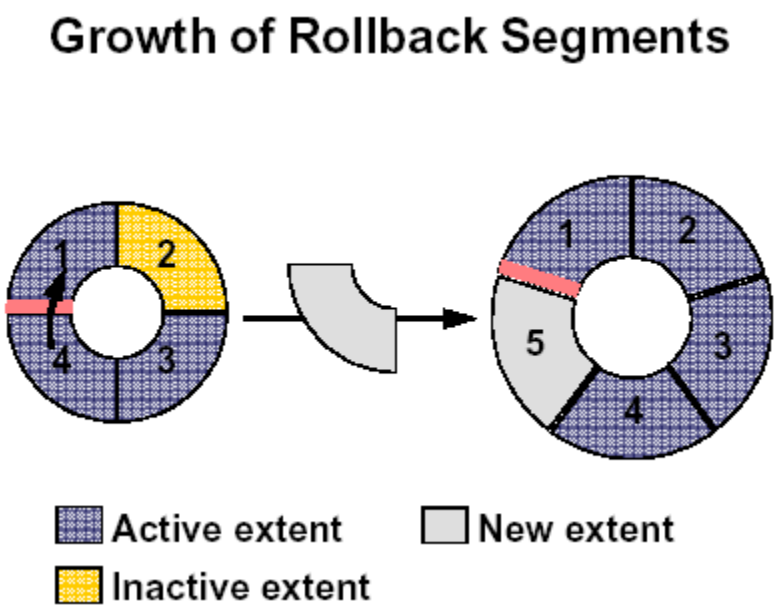


Hình vẽ 51. Sử dụng dữ liệu trong rollback segment

1. Tại thời điểm bắt đầu, giao dịch bắt đầu ghi dữ liệu vào Extent 3
2. Trong khi thực hiện, các transaction sẽ ghi dữ liệu vào Extent 3 cho tới khi đầy rồi tiếp tục chuyển sang ghi dữ liệu lên Extent 4.
3. Khi Extent 4 cũng đầy, nó tiếp tục lại quay trở lại ghi dữ liệu vào extent 1 nếu extent này ở trạng thái rỗi hoặc inactive. Một extent là rỗi hoặc inactive nếu hiện thời nó không bị sử dụng bởi bất kỳ một transaction nào.

11.2.2. Tăng trưởng đối với các rollback segments

Rollback segment có con trỏ để xác định extent đang làm việc. Khi extent làm việc đầy, con trỏ sẽ chuyển sang extent kế tiếp để thực hiện việc ghi dữ liệu. Cứ như vậy cho đến extent cuối cùng rồi lại quay trở về extent đầu tiên nếu extent này đang rỗi. Tuy nhiên, có nhiều khả năng extent đầu tiên này cũng đang không rỗi. Khi đó, con trỏ không thể nhảy cách mà bỏ qua extent 1 để chuyển sang extent 2 được. Để tiếp tục duy trì hoạt động cho transaction, cần phải bổ sung thêm một extent nữa vào sau extent cuối cùng. Việc này tạo nên sự tăng trưởng đối với các rolback segments. Việc tăng trưởng đối với các rolback segments sẽ tiếp tục xảy ra cho tới khi số lượng các extents tăng kịch khung quy định trong tham số MAXEXTENTS.



Hình vẽ 52. Tăng kích thước Rollback Segment

Sau khi rollback segments đã được tạo lập, quản trị viên database vẫn có thể thay đổi tham số lưu trữ của rollback segments. Để thay đổi, quản trị viên chỉ cần điều chỉnh các tham số `OPTIMAL` hay `MAXEXTENTS` cho phù hợp.

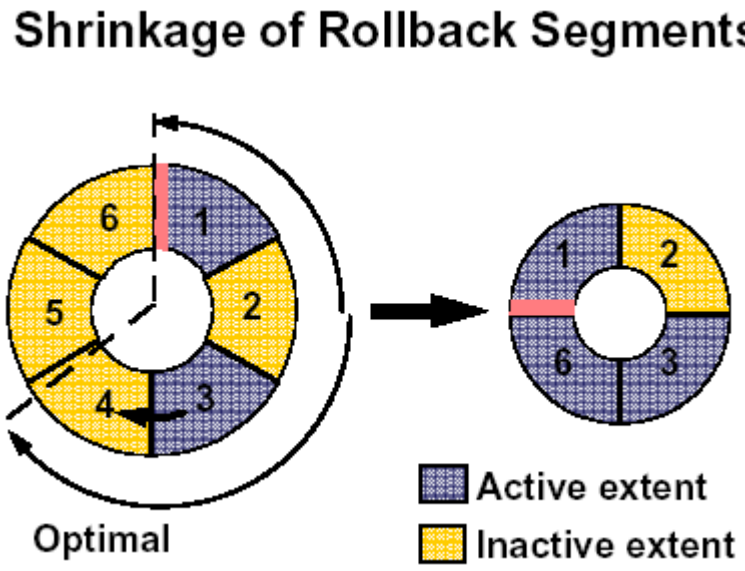
Ví dụ: Câu lệnh sau thay đổi số lượng tối đa các extents cấp phát cho rollback segments `RBS_01`.

```
ALTER ROLLBACK SEGMENT rbs_01
    STORAGE (MAXEXTENTS 120);
```

Với câu lệnh thay đổi này, ta cũng có thể điều chỉnh với rollback segment SYSTEM , bao gồm cả tham số `OPTIMAL`.

11.2.3. Tối ưu các rollback segments

Khi kết thúc hoặc commit các transaction, nó sẽ giải phóng vùng không gian đã sử dụng để lưu các dữ liệu dùng để phục hồi. Các extent trong rollback được đưa trở lại trạng thái inactive. Để tiết kiệm không gian lưu trữ trong rollback segment, ta có thể tối ưu lại rollback segment đó thông qua tham số `OPTIMAL`.



Hình vẽ 53. Giảm kích thước của Rollback segment

Oracle server sẽ thu hồi lại các extent đã cấp phát khi:

- Kích thước của rollback segment hiện tại được điều chỉnh tới giá trị của tham số `OPTIMAL`.
- Khi có nhiều hơn 02 extent rỗi liên tiếp cạnh nhau.

Một điều lưu ý là khi thu hồi lại các extent, Oracle server sẽ thu hồi extent chứa dữ liệu lâu nhất trước đó.

Ta có thể thực hiện giảm bớt kích thước của rollback segments thông qua việc sử dụng câu lệnh `ALTER ROLLBACK SEGMENT`. Lưu ý, khi này rollback segment được thu nhỏ nhất thiết phải đang ở trạng thái online.

Ví dụ: Thu nhỏ kích thước rollback segment `RBS1` bằng `100K`:

```
ALTER ROLLBACK SEGMENT rbs1 SHRINK TO 100K;
```

Câu lệnh phía trên thực hiện rút bớt kích thước của rollback segment tới kích thước như đã được chỉ ra. Tuy nhiên, việc rút gọn sẽ dừng lại khi có một extent không thể bị thu hồi do bất kỳ nguyên nhân nào.

11.3.QUẢN LÝ ROLLBACK SEGMENTS

11.3.1. Sử dụng rollback segment

Kích thước của rollback segment

Kích thước của rollback được xác định tuỳ thuộc vào hai yếu tố sau:

- Loại transaction được thực hiện (insert, update, delete, ...)
- Lượng dữ liệu được xử lý

Thông thường, việc thêm mới bản ghi vào bảng cần ít không gian lưu giữ thông tin phục hồi hơn là việc xoá dữ liệu khỏi bảng. Với thao tác thêm mới, chỉ cần lưu giữ ROWID vào rollback, trong khi thao tác delete lại cần phải lưu giữ toàn bộ dòng dữ liệu.

Đánh giá kích thước của rollback segment căn cứ theo transaction dài nhất có sử dụng rollback segment.

Số lượng các Extents

Với các rollback segment có quá nhiều các extents sẽ gây ra lãng phí không gian lưu trữ dữ liệu, để giảm bớt lãng phí, ta có thể điều chỉnh tham số `MINEXTENTS` cho phù hợp.

Oracle khuyến nghị, thông thường, `MINEXTENTS` nên đặt giá trị là 20.

11.3.2. Tạo rollback segment

Ta có thể tạo rollback segment thông qua câu lệnh SQL:

Cú pháp:

```
CREATE [PUBLIC] ROLLBACK SEGMENT rollback_segment
      [TABLESPACE tablespace]
      [STORAGE ([INITIAL integer[K|M]]
            [NEXT integer[K|M]]
            [MINEXTENTS integer]
            [MAXEXTENTS {integer|UNLIMITED}]
            [OPTIMAL {integer[K|M]|NULL}]
            )
      ]
```

Lưu ý:

- Một rollback segment có thể là `PUBLIC` hoặc `PRIVATE` (mặc định) việc gán này được thực hiện ngay lúc tạo và không thể thay đổi sau này.
- `MINEXTENTS>=2` đối với các rollback segment.
- `PCTINCREASE` được bỏ qua đối với các rollback segment và được gán bằng 0.
- `OPTIMAL`, nếu có chỉ ra thì không được nhỏ hơn giá trị kích thước khởi tạo của rollback segment được xác định trong tham số `MINEXTENTS`.
- `INITIAL=NEXT` để đảm bảo các extent trong rollback segment có cùng một kích thước.
- Không nên gán giá trị cho `MAXEXTENTS` là `UNLIMITIED` vì như vậy sẽ dẫn đến việc mở rộng các extent một cách không cần thiết.
- Nên đặt rollback segment trong một tablespace riêng biệt để giảm bớt hiện tượng phân đoạn dữ liệu trong database.

Ví dụ:

```
CREATE ROLLBACK SEGMENT rbs01
```

```
TABLESPACE rbs
STORAGE (
INITIAL 100K NEXT 100K OPTIMAL 4M
MINEXTENTS 20 MAXEXTENTS 100);
```

Trong Oracle Enterprise ta thực hiện theo các bước sau:

1. Chạy Oracle Storage Manager.
2. Chọn Rollback—>Create.
3. Trong phần General page, nhập vào tên, tablespace, và kiểu tương ứng. Chọn mục Online trong radio button.
4. Trong phần Extents, nhập vào các thông tin về rollback segments.
5. Bấm nút Create.

11.3.3. Thay đổi trạng thái của Rollback segments

Rollback segments có thể nhận một trong hai trạng thái `ONLINE/OFFLINE`

Khi rollback segment có trạng thái *online* thì nó sẵn sàng sử dụng cho các transactions, ngược lại, trạng thái *offline* cho biết nó không sẵn sàng cho các transactions. Thông thường, rollback segments là online và sẵn dùng cho các transactions.

Trong một số tình huống nhất định, ta cần đặt trạng thái online hay ofline đối với các rollback segments:

- Khi trạng thái của tablespace là online, nếu tablespace có chứa các rollback segments, ta sẽ không thể đặt trạng thái tablespace thành offline nếu có bất kỳ một transaction nào vẫn còn đang sử dụng các rollback segments thuộc tablespace đó. Để xử lý được tình huống này, ta cần thay đổi trạng thái của rollback segments thành offline để ngăn không cho sử dụng các rollback segments trước khi thay đổi trạng thái của tablespace là offline.
- Khi ta muốn drop (huỷ) các rollback segments, nhưng không thể thực hiện được do vẫn còn transactions đang sử dụng nó. Để xử lý được tình huống này, ta cần ngăn không cho sử dụng rollback segment thông qua việc đặt lại trạng thái rollback segments là offline.

Sau khi tạo mới một rollback segment, nó sẽ có trạng thái offline và chưa thể sử dụng ngay được. Để có thể sẵn dùng cho các transaction, rollback segment cần được chuyển trạng thái thành online thông qua câu lệnh `ALTER ROLLBACK SEGMENT`

Cú pháp:

```
ALTER ROLLBACK SEGMENT rollback_segment ONLINE | OFFLINE
```

Rollback segment sẽ có trạng thái online cho tới khi instance bị tắt (shutdown).

Đặt trạng thái online cho rollback segments ngay khi startup database

Để đảm bảo cho các rollback segments luôn nhận trạng thái online ngay khi khởi động (startup) database, ta cần chỉ rõ tên của rollback segments trong tham số ROLLBACK_SEGMENTS của parameter file.

Ví dụ:

```
ROLLBACK_SEGMENTS=(rbs01, rbs02, rbs03)
```

Lưu ý: Số lượng tối đa các rollback segment online đối với một instance được xác định bởi tham số MAX_ROLLBACK_SEGMENT.

Trong OEM ta có thể thực hiện theo các bước sau:

1. Chạy Oracle Storage Manager.
2. Chuyển tới nút Rollback Segments.
3. Chọn rollback segment tương ứng.
4. Chọn Rollback—>Place Online/ Take Offline.
5. Trong hộp thoại xác nhận, bấm nút Yes.

11.3.4. Instance sử dụng rollback segment

Để cụ thể, ta xem xét các bước thực hiện khi một instance sử dụng rollback segment

1. Instance sử dụng tất cả các rollback segments có tên trong phần tham số ROLLBACK_SEGMENTS.
2. Tham số TRANSACTIONS và TRANSACTIONS_PER_ROLLBACK_SEGMENT được sử dụng để tính toán số lượng rollback segments cần thiết cho một instance:

$$N = \frac{T}{TPR}$$

Với:

N	Số lượng rollback segment cần thiết
T	Giá trị tham số TRANSACTIONS
TRP	Giá trị tham số TRANSACTIONS_PER_ROLLBACK_SEGMENT

3. Trong trường hợp N nhỏ hơn hay bằng số lượng non-SYSTEM rollback segments có được, instance cũng sẽ không cần tới nhiều rollback segments hơn.
4. Khi giá trị của N lớn hơn hay bằng số các non-SYSTEM rollback segments dành cho instance, khi đó đòi hỏi phải sử dụng thêm cả các public rollback segments.

11.3.5. Điều chỉnh khả năng lưu trữ của rollback segment

Ta có thể điều chỉnh các tính chất lưu trữ của từng rollback segment thông qua lệnh ALTER ROLLBACK SEGMENT

Cú pháp:

```
ALTER ROLLBACK SEGMENT rollback_segment
[STORAGE ( [NEXT integer[K|M]]
      [MINEXTENTS integer]
      [MAXEXTENTS {integer|UNLIMITED}]
      [OPTIMAL {integer[K|M]|NULL}]
            )
]
```

Trong OEM ta thực hiện theo các bước sau:

1. Chạy Oracle Storage Manager.
2. Chuyển tới nút Rollback Segments.
3. Chọn rollback segment tương ứng.
4. Trong phần Extents page, đặt lại các tham số tương ứng.
5. Bấm nút Apply.

11.3.6. Giảm bớt độ rộng của rollback segment

Trong trường hợp tham số `OPTIMAL` được chỉ rõ, Oracle server sẽ có gắng thực hiện cấp phát và giải phóng vùng không gian dựa theo giá trị của tham số `OPTIMAL`. Ngược lại, ta có thể thực hiện cấp phát không gian thông qua lệnh trực tiếp:

```
ALTER ROLLBACK SEGMENT rollback_segment
      SHRINK [ TO integer [ K|M ]];
```

Trong trường hợp tham số integer không được chỉ rõ, Oracle sẽ giảm lượng không gian rollback segment về tới giá trị `OPTIMAL`

Trong OEM ta có thể thực hiện theo các bước sau:

1. Chạy Oracle Storage Manager.
2. Chuyển tới nút Rollback Segments.
3. Chọn rollback segment tương ứng.
4. Chọn Rollback—>Shrink.
5. Trong hộp thoại Shrink Rollback Segment, chọn Optimal Size rút gọn kích thước rollback segment theo kích thước đã được tối ưu. Hoặc chọn Size rồi nhập vào giá trị kích thước cho vùng không gian tương ứng.
6. Bấm nút OK.

11.3.7. Hủy bỏ rollback segment

Trong một số trường hợp không cần sử dụng các rollback segment, ta có thể hủy các rollback segment thông qua câu lệnh SQL:

```
DROP ROLLBACK SEGMENT rollback_segment;
```

Trong OEM, ta làm theo các bước sau:

1. Chọn Oracle Storage Manager.
2. Chuyển tới nút Rollback Segments.
3. Chọn rollback segment tương ứng.
4. Chọn Rollback—>Remove. Ta chỉ có thể hủy các rollback segment đã ở trạng thái offline.
5. Bấm nút Yes trong hộp thoại xác nhận.

11.3.8. Quản lý undo tự động

Khả năng quản lý undo tự động (Automatic Undo Management - AUM) là một đặc điểm khá mới của Oracle 9i. Cung cấp cơ chế tin cậy hơn cho DBA khi tạo, thay đổi kích thước và điều chỉnh rollback segments trong database. Theo đó, Rollback segments có thể được tạo, xoá hay điều chỉnh kích thước một cách tự động bởi instance.

Dữ liệu rollback data được quản lý nhờ vào undo tablespace.

Ví dụ: tạo undo tablespace

```
CREATE UNDO TABLESPACE “UNDO_TBS”
   DATAFILE ‘/u01/oradata/freeney9/undo_tbs01.ora’ SIZE 100M
   AUTOEXTEND ON NEXT 10M MAXSIZE 700M
```

Một số tham số khởi tạo chính:

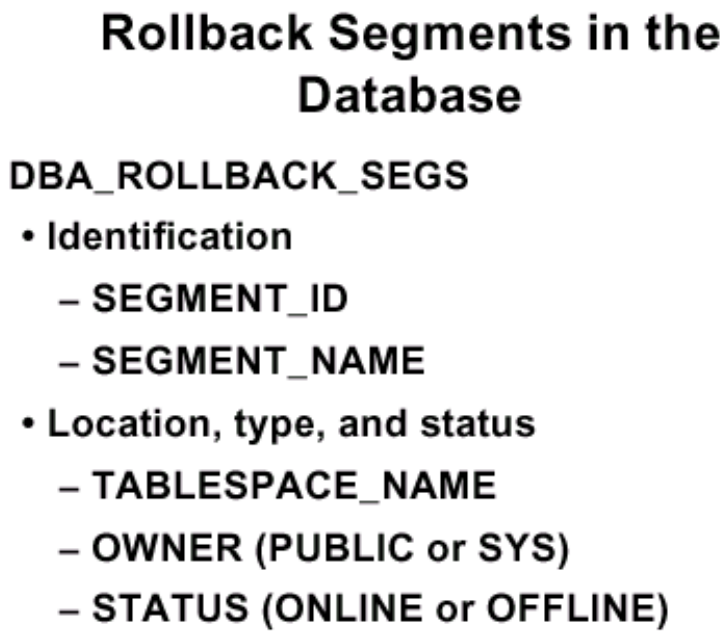
- `UNDO_MANAGEMENT (MANUAL / AUTO)`: Cho biết database có sử dụng cơ chế AUM hay không. Default = MANUAL
- `UNDO_TABLESPACE (valid tablespace)`: Chỉ rõ tên undo tablespace sử dụng.
- `UNDO_RETENTION (in seconds default=30)`: Cho biết thời gian trễ để thực hiện committed undo.
- `UNDO_SUPPRESS_ERRORS (TRUE / FALSE)`: Cho biết hệ thống có trả về exception hay không khi “SET TRANSACTION USE ROLLBACK SEGMENT” phát lỗi. Default = TRUE

11.4.THÔNG TIN VỀ CÁC ROLLBACK SEGMENT

Thông tin về các rollback segment được lưu giữ trong từ điển dữ liệu.

11.4.1. Xem thông tin chung về các rollback segment

Thông tin chung về rollback segment được lưu trong view `DBA_ROLLBACK_SEGS`.



Hình vẽ 54. Các thông tin chính về rollback segments

Các thông tin bao gồm:

- SEGMENT_ID: Mã hiệu của segment
- SEGMENT_NAME: Tên segment
- TABLESPACE_NAME: Tên tablespace chứa segment
- OWNER (PUBLIC/SYS): Tên user sở hữu segment
- STATUS (ONLINE/OFFLINE): Trạng thái của segment

Ví dụ: Xem thông tin chung về segment

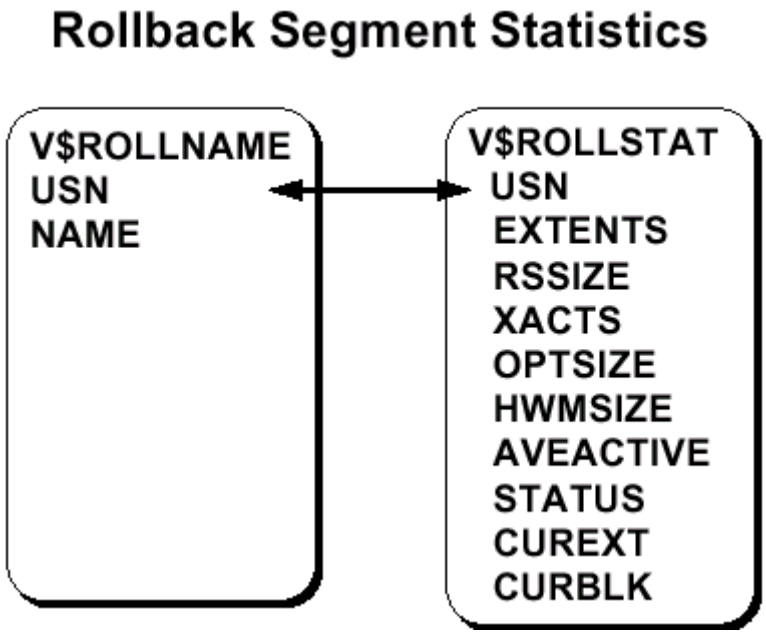
```
SVRMGR> SELECT segment_name, tablespace_name, owner, status
2> FROM dba_rollback_segs;
SEGMENT_NAME               TABLESPACE_NAME  OWNER STATUS
---------------            ---------------  ------
               ---------------
SYSTEM         SYSTEM           SYS         ONLINE
RBS1           RBS              SYS         ONLINE
RBS2           RBS              SYS         ONLINE
RBS3           RBS              SYS         OFFLINE
4 rows selected.
```

Dữ liệu trong cột OWNER nhận các giá trị:

- SYS: Rollback đó thuộc loại private
- PUBLIC: Rollback đó thuộc loại public

11.4.2. Xem thông tin thống kê về rollback segment

Ta lấy được các thông tin này từ các view V$ROLLSTAT và V$ROLLNAME.



Hình vẽ 55. Các thông tin thống kê về segments

Ví dụ: Xem các thông tin thông kê về segments

```
SVRMGR> SELECT n.name, s.extents, s.rssize, s.optsize,
2> s.hwmsize, s.xacts, s.status
3> FROM v$rollname n, v$rollstat s
4> WHERE n.usn = s.usn;

NAME   EXTENTS  RSSIZE    OPTSIZE   HWMSIZE   XACTS  STATUS
-----  -------  --------  --------  --------  ------ ---------
SYSTEM 43       2199552             2199552   0      ONLINE
RBS1   20       202752    204800    417792    0      ONLINE
RBS2   4        38912               38912     0      PENDING
                                               OFFLINE
3 rows selected.
```

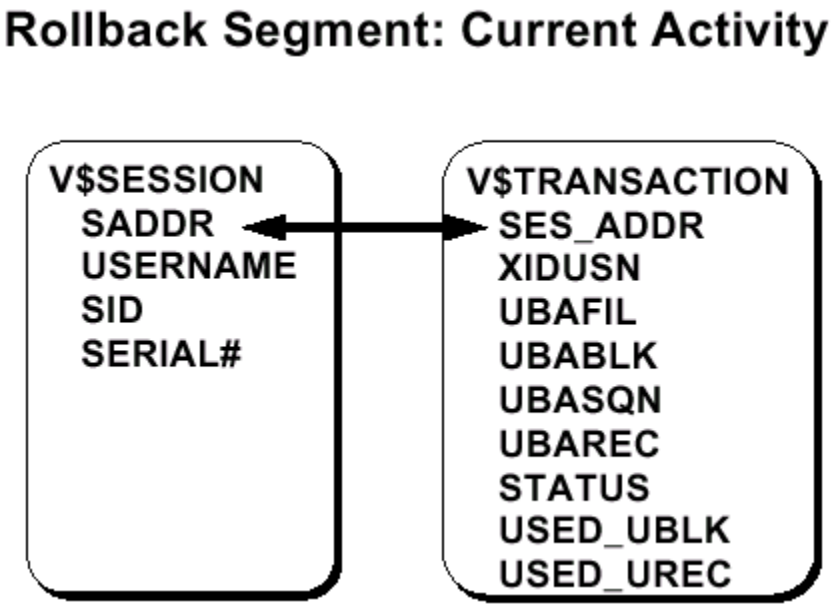
Diễn giải một số cột dữ liệu trong view V$ROLLSTAT

Tên cột	Diễn giải
USN	Là số hiệu của rollback segment (Rollback segment number)
EXTENTS	Số lượng các extents có trong rollback segment
RSSIZE	Kích thước của segment hiện thời tính theo đơn vị bytes
XACTS	Số lượng các transaction sử dụng rollback segment
OTPSIZE	Giá trị OPTIMAL của rollback segment
HWMSIZE	Hight warter mark; kích thước tối đa tính theo bytes, khi rollback segment tăng

AVEACTIVE	Kích thước của extent hiện thời,
STATUS	Trạng thái của rollback segment

11.4.3. Thông tin về rollback segment đang active

Ta có thể kết hợp thông tin trong hai bảng V$TRANSACTION và V$SESSION.



Hình vẽ 56. Thông tin về các thao tác trên các segments

Ví dụ:

```
SVRMGR> SELECT s.username, t.xidusn, t.ubafil,
2> t.ubablk, t.used_ublk
3> FROM v$session s, v$transaction t
4> WHERE s.saddr = t.ses_addr;

USERNAME     XIDUSN     UBAFIL     UBABLK     USED_UBLK
-----------  ---------- ---------  ---------- ----------
SYSTEM       2          2          7          1
SCOTT        1          2          163        1
2 rows selected.
```

Diễn giải một số cột dữ liệu

Tên cột	Diễn giải
SES_ADDR	Địa chỉ của session, lấy được từ V$SESION.SADDR
XIDUSN	Số hiệu của Rollback segment được sử dụng bởi transaction
UBAFIL, UBABLK, UBASQN,UBAREC	Vị trí hiện thời của rollback segment mà transaction sẽ ghi vào
USED_UBLK	Số hiệu block undo được tạo ra bởi transaction

START_UEXT, START_UBAFIL, START_UBABLK	Số hiệu của extent (file, block) thuộc rollback segment mà transaction bắt đầu ghi dữ liệu

11.5.CÁC VẤN ĐỀ LIÊN QUAN TỚI ROLLBACK SEGMENT

11.5.1. Thiếu không gian cho các transactions

Nguyên nhân

Do một transaction không được sử dụng nhiều rollback segments nên có thể xảy ra tình trạng thiếu vùng không gian cho các rollback segment và gây ra lỗi (ORA-01562). Nguyên nhân có thể là một trong các trường hợp sau:

- Không có đủ không gian trong tablespace (ORA-01560)
- Số lương các extents trong rollback segment đã đạt tới giá trị MAXEXTENTS và không thể bổ sung thêm các extent vào rollback segment (ORA-01628)

Giải pháp

Với lỗi ORA-01560:

- Mở rộng thêm các data files trong tablespace
- Đặt chế độ cho các data files là AUTOEXTEND
- Bổ sung mới data file vào tablespace

Với lỗi ORA-01628:

- Tăng tham số MAXEXTENTS của rollback segment
- Huỷ và tạo lại rollback segment với kích thước của extent lớn hơn

11.5.2. Lỗi đọc dữ liệu không đồng nhất

Nguyên nhân

Oracle server cố gắng đảm bảo các câu lệnh sẽ chỉ xử lý trên các dữ liệu đã được commit. Vì thế, các dữ liệu chưa commit sẽ không được sử dụng. Trong trường hợp Oracle server không tạo được các bản lưu giá trị cũ các dữ liệu (read-consistent image of data), user sẽ nhận được lỗi ORA-01555 snapshot too old. Lỗi này xảy ra khi transaction thay đổi các dữ liệu đã được commit và:

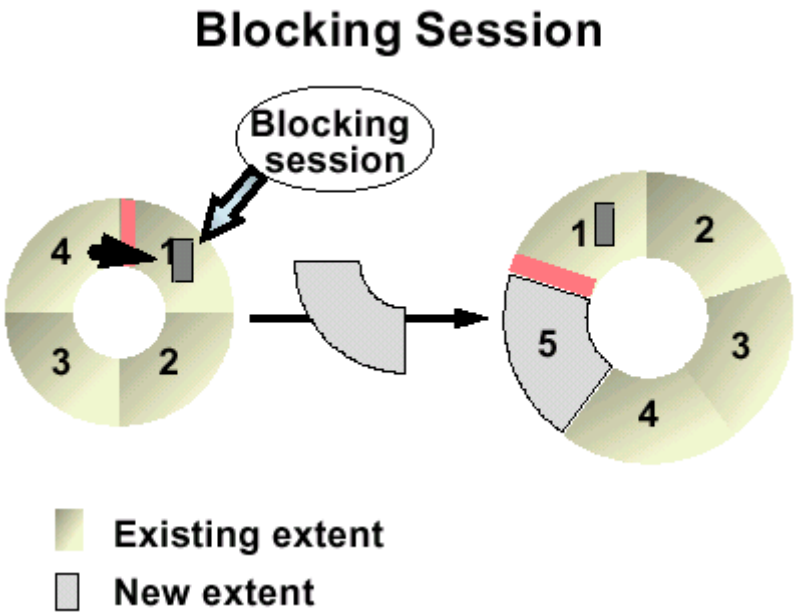
- Transaction slot có trong phần rollback header đang được sử dụng
- Giá trị ban đầu (before-image) trong rollback segment được ghi đè lên bởi một transaction khác

Giải pháp

- Tăng chỉ số MINEXTENTS
- Mở rộng kích thước extent

- Tăng giá trị OPTIMAL

11.5.3. Chặn session



Hình vẽ 57. Chặn session

Vấn đề

Khi một extent trong rollback segment được ghi đầy, Oracle server sẽ tiếp tục sử dụng extent kế tiếp theo cơ chế xoay vòng. Trong trường hợp extent kế tiếp vẫn đang trong tình trạng active, transaction sẽ không sử dụng được nó. Mặt khác, nó cũng không thể bỏ qua extent kế tiếp để chuyển tới extent sau nữa nếu nó rỗi. Khi đó, rollback segment sẽ được bổ sung thêm các extent. Việc làm này làm cho rollback segment ngày một mở rộng và quản trị viên cần phải can thiệp để hạn chế việc mở rộng này.

Giải pháp

Quản trị viên database cần thực hiện kiểm tra thông tin của các transaction đang được thực hiện thông qua việc lấy thông tin từ các view V$ROLLSTAT, V$TRANSACTION, V$SESSION để phát hiện các transaction đang bị cản trở, từ đó thực hiện việc điều chỉnh cho phù hợp. Công việc kiểm tra và giám sát này được thực hiện bằng tay bởi người quản trị database.

Ví dụ: Xem thông tin về các transactions đang được thực hiện

```
SVRMGR> SELECT s.sid, s.serial#, t.start_time, t.xidusn,
s.username
2> FROM v$session s, v$transaction t, v$rollstat r
3> WHERE s.saddr = t.ses_addr
4> AND t.xidusn = r.usn
5> AND ((r.curext = t.start_uext-1) OR
6> ((r.curext = r.extents-1) AND t.start_uext=0));

SID   SERIAL# START_TIME              XIDUSN USERNAME
```

```
---   ------- -------------           ------ --------
9     27      10/30/97 21:10:41       2      SYSTEM
1 row selected.
```

Chương 12. QUẢN LÝ TEMPORARY SEGMENTS

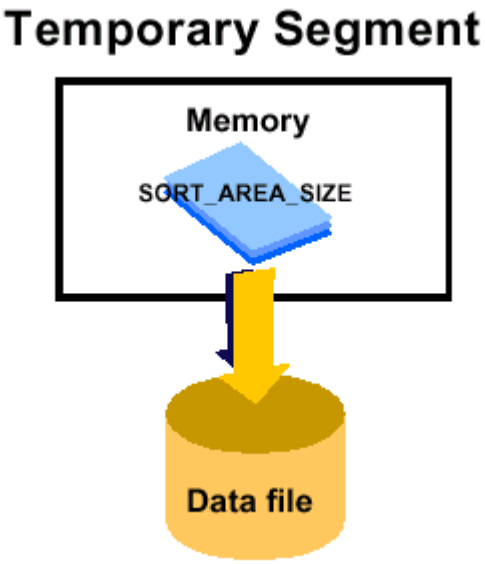
12.1. TEMPORARY SEGMENTS

Temporary segments được sử dụng khi Oracle server thực các hiện câu lệnh sắp xếp mà không thể sử dụng vùng không gian trong bộ nhớ do không đủ, ví dụ như:

- `SELECT. . . ORDER BY`
- `CREATE INDEX`
- `SELECT DISTINCT`
- `SELECT. . . GROUP BY`
- `SELECT. . . UNION`

Dung lượng bộ nhớ cần thiết cho tiến trình sắp xếp được xác định dựa trên tham số khởi tạo `SORT_AREA_SIZE`. Trong một số trường hợp, nhiều thao tác sắp xếp cùng được sử dụng và cần nhiều bộ nhớ hơn. Khi này bộ nhớ trong của máy là không thể đáp ứng được và kết quả của việc sắp xếp đó cần phải được tạm thời lưu lên đĩa. Vùng đĩa lưu trữ các dữ liệu trung gian này chính là temporary segments.

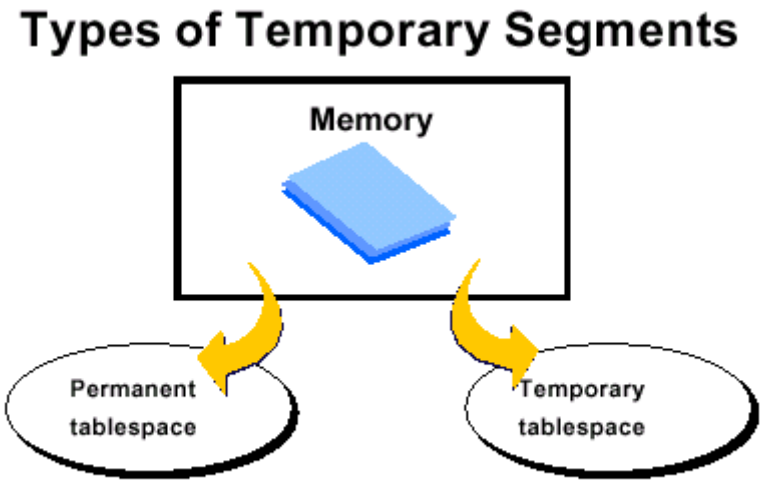
Temporary segments trong tablespace được Oracle server tạo lập với mục đích sử dụng làm vùng nhớ trung gian hỗ trợ thao tác sắp xếp.



Hình vẽ 58. Temporary segment

12.1.1. Phân loại temporary segments

Các temporary segments có thể được tạo trên một permanent tablespace hoặc trên một temporary tablespace. User có thể sử dụng một trong các kiểu tablespaces này để sắp xếp.



Hình vẽ 59. Phân loại temporary segment

Temporary Tablespace

Một temporary tablespace được sử cho các temporary segments tương ứng và không chứa bất kỳ segment nào có kiểu khác. Ta có thể tạo các temporary tablespace theo câu lệnh SQL sau:

```
CREATE TABLESPACE tablespace_name TEMPORARY
      DATAFILE filespec [autoextend_clause]
      [ , filespec [autoextend_clause]] ...
```

Một permanent tablespace có thể chuyển đổi thành dạng temporary tablespace bằng cách sử dụng câu lệnh:

```
ALTER TABLESPACE tablespace_name TEMPORARY
```

Lưu ý: với câu lệnh trên, tablespace không được phép chứa bất kỳ một đối tượng thường trú nào (như: table, store procedure, ...). Một temporary tablespace có thể chuyển đổi lại thành permanent tablespace thông qua câu lệnh SQL dưới đây:

```
ALTER TABLESPACE tablespace_name PERMANENT
```

Oracle server có thể tạo một temporary segment trong một permanent tablespace với số điều kiện sau:

- User thực hiện câu lệnh sắp xếp cần đến vùng không gian trên đĩa.
- User chạy câu lệnh mà nó đã được gán cho một permanent tablespace để thực hiện sắp xếp.

Khi một permanent tablespace được sử dụng cho việc sắp xếp, một instance có thể có một hoặc nhiều temporary segment trong tablespace.

Một temporary segment sẽ được hủy bởi tiến trình nền SMON khi kết thúc câu lệnh sắp xếp và vùng không gian đã cấp phát sẽ được giải phóng để cho các đối tượng khác của database sử dụng. Permanent tablespaces được sử dụng cho việc sắp xếp, có ba vùng không gian trong tablespace có thể được phân vùng khác nhau. Thông thường, mỗi tablespace nên được sử dụng cho từng tiến trình sắp xếp khác nhau.

Khi một temporary tablespaces được sử dụng cho các temporary segments, Instance chỉ tạo một segment dùng để sắp xếp cho tablespace. Một vài transactions cần đến sắp xếp trên ổ đĩa có thể sử dụng cùng segment. Tuy nhiên, một extent thì không thể cùng chia sẻ đồng thời cho nhiều transactions khác nhau.

12.1.2. Sử dụng các Sort Segments

Sort segment được tạo bởi câu lệnh sắp xếp đầu tiên sử dụng tới temporary tablespace cho việc sắp xếp. Và sort segment chỉ bị hủy khi tắt (shutdown) database. Việc này làm giảm bớt số lần cấp phát và thu hồi các sort segments phục vụ cho công việc sắp xếp, làm tăng năng suất hệ thống. Oracle không hạn chế số lượng các extents cấp phát cho mỗi sort segment thuộc một temporary tablespace.

12.1.3. Sort Extent Pool

Oracle server lưu lại chi tiết sort segment trong vùng Sort Extent Pool của vùng nhớ SGA, mỗi câu lệnh cần tới các vùng trống để thực hiện sắp xếp có thể tìm các extent rỗi trong vùng nhớ này.

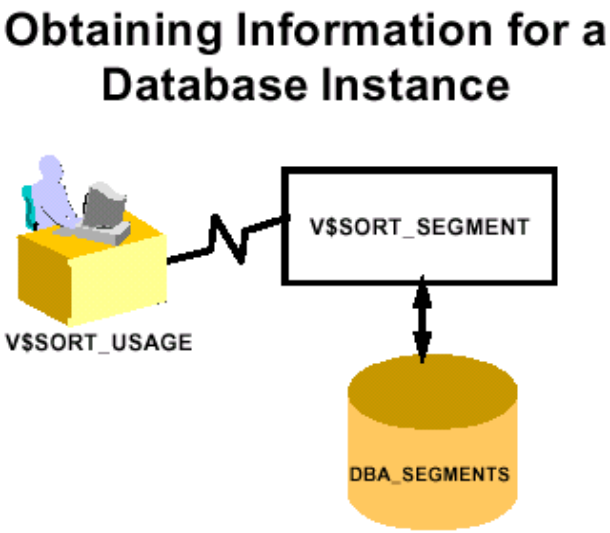
12.2.CẤP PHÁT KHÔNG GIAN CHO TEMPORARY SEGMENT

Temporary tablespaces được sử dụng để tăng hiệu quả sắp xếp dữ liệu. Kích thước của các extents trong temporary segment được xác định bởi `DEFAULT STORAGE` clause của tablespace tương ứng.

Do lượng dữ liệu ghi lên temporary segment bằng phần nguyên lần giá trị `SORT_AREA_SIZE`. Do vậy, ta nên đặt `INITIAL = NEXT = (n*SORT_AREA_SIZE)+ DB_BLOCK_SIZE`

Giá trị `PCTINCREASE=0`, để đảm bảo các extents có cùng kích thước.

12.3.THÔNG TIN VỀ CÁC TEMPORARY SEGMENT



Hình vẽ 60. Thu nhận thông tin về database instance

Ta có thể lấy được các thông tin về temporary segment trong một số bảng từ điển dữ liệu:

DBA_SEGMENTS: chứa thông tin về tất cả các loại segments trong database.

V$SORT_SEGMENT: cho biết trạng thái của các sort extent pool (vùng không gian sắp xếp). Với từ điển dữ liệu này, ta có thể biết được những thông tin sau:

Tên cột	Diễn giải
TABLESPACE_NAME	Tên temporary tablespace
EXTENT_SIZE	Kích thước của extent
TOTAL_EXTENTS	Tổng số các extents
TOTAL_BLOCKS	Tổng số các blocks

USED_EXTENTS	Số lượng extents đã sử dụng
USED_BLOCKS	Số lượng blocks đã sử dụng
FREE_EXTENTS	Số lượng extents còn trống
FREE_BLOCKS	Số lượng blocks còn trống
MAX_SORT_SIZE	Kích thước tối đa của vùng dữ liệu sắp xếp
MAX_SORT_BLOCKS	Số lượng blocks tối đa dùng để sắp xếp dữ liệu

Ví dụ:

```
SVRMGR> SELECT tablespace_name, extent_size,
      2> total_extents, max_sort_blocks
      3> FROM v$sort_segment;

TABLESPACE_NAME EXTENT_SIZ        TOTAL_EXTE  MAX_SORT_B
--------------- ----------        ----------  ----------
TEMP                   128        1           128
1 row selected.
```

MAX_SORT_SIZE và MAX_SORT_BLOCKS là số lượng các extents và các blocks sử dụng bởi phép sắp xếp lớn nhất. Thông tin này là hữu ích trong việc điều chỉnh kích thước của temporary tablespace

V$SORT_USAGE: cho biết thông tin về các sắp xếp hiện có của instance, ta kết hợp với V$SESSION để biết thêm các thông tin:

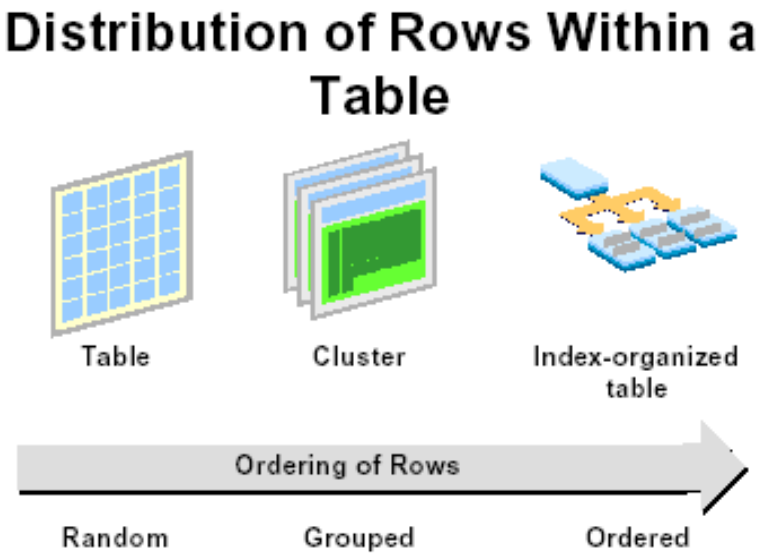
Ví dụ:

```
SVRMGR> SELECT s.username, u."USER", u.tablespace,
      2> u.contents, u.extents, u.blocks
      3> FROM v$session s,v$sort_usage u
      4> WHERE s.saddr=u.session_addr;

USERNAME         USER   TABLESPACE  CONTENTS   EXTENTS   BLOCKS
--------         -----  ----------  ---------  -------   ------
SYSTEM           SYS    TEMP        TEMPORARY  1         128
1 row selected.
```

Chương 13. CLUSTERS VÀ INDEX-ORGANIZED TABLES

13.1.TỔNG QUAN VỀ CLUSTERS VÀ INDEX-ORGANIZED TABLES

Trong một bảng thông thường người sử dụng có rất nhiều giới hạn về điều khiển phân tán các hàng trong bảng. Khi một bảng được tạo ra lần đầu tiên nói chung các hàng được chèn vào trong đoạn bắt đầu của block đầu tiên trong extent đầu tiên. Nhưng một khi các câu lệnh DML đã đưa ra một vài yếu tố như trật tự của các block trong danh sách các block tự do, các hàng bị migration sẽ làm cho các câu lệnh này rất khó trong việc săp xếp các hàng trong bảng.



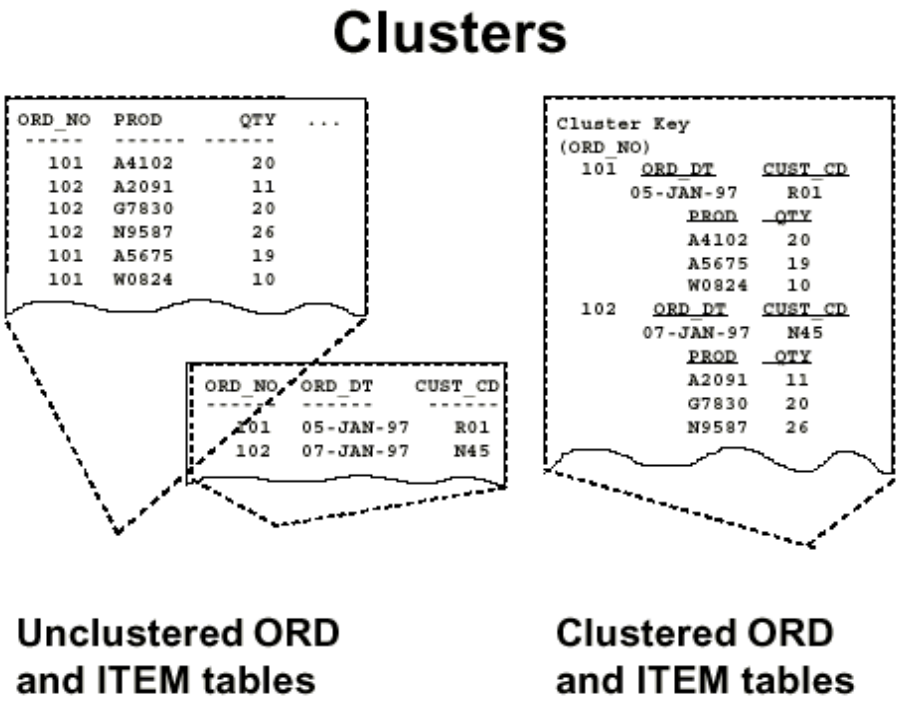
Hình vẽ 61. Lưu trữ các dòng dữ liệu trong một table

Cluster hỗ trợ một vài quá trình điều khiển các hàng được lưu trữ. Khi một cluster được sử dụng, Oracle server sẽ lưu tất cả các hàng mà có cùng giá trị khoá trong cùng một Block nếu có thể.

Khi bảng Index-Organized được sử dụng, người sử dụng có thể điều khiển trật tự của các hàng. Dữ liệu trong bảng index-organized sẽ ở trong trật tự của khoá chỉ định.

13.1.1. Cluster

Một cluster dùng để chứa tập các hàng có quan hệ trong một Oracle block.



Hình vẽ 62. Cluster

Sự khác nhau giữa bảng được cluster và bảng không cluster

Nếu hàng dữ liệu được lưu trong các bảng thông thường thì các bảng khác nhau sẽ sử dụng tập các block khác nhau của riêng bảng đó. Một block mà sử dụng để lưu các hàng từ bảng `ORDER` sẽ không chứa dữ liệu trong bảng `ITEM` và ngược lại.

Nếu hai bảng `ORDER` và bảng `ITEM` được lưu trong một cluster chúng sẽ chia sẻ cùng một phân đoạn cluster. Một block trong phân đoạn này có thể lưu các hàng của cả hai bảng. Nếu một bảng được lưu trong một cluster, cluster sẽ trở thành đơn vị vật lý lưu trữ và bảng sẽ là thực thể logic.

Các đặc tính của cluster

Các cluster có đặc tính sau:

- Mỗi cluster bao giờ cũng có cluster key (khoá của cluster) dùng để xác định các hàng cần phải nhóm lại với nhau.
- Cluster key có thể dựa trên một cột hay nhiều cột.

- Các bảng trong một cluster phải có các cột tương ứng với cluster key.
- Cluster là cơ chế trong suốt đối với ứng dụng sử dụng các bảng. Dữ liệu trong các bảng được cluster có thể được thao tác giống như trên các bảng thông thường.
- Cập nhật một cột trong khoá của cluster có thể dẫn đến việc thiết lập lại vị trí của hàng đó.
- Cluster key độc lập với primary key (khoá chính) của bảng. Các bảng trong cluster có thể có primary key. Primary key này có thể tham gia vào cluster key.
- Cluster hay được dùng để hỗ trợ hiệu năng. Truy xuất ngẫu nhiên dữ liệu trong cluster có thể nhanh hơn, nhưng khi sử dụng truy vấn trên toàn bảng thì các bảng được cluster sẽ chậm hơn.
- Cluster sẽ quyết định giá trị các tham số lưu trữ của tất cả các tables trong cluster đó.

13.1.2. Xem xét và chọn lựa Cluster

Chọn các tables cho cluster

Sử dụng các clusters để lưu trữ một hay nhiều tables mà chúng thường xuyên được sử dụng để truy vấn dữ liệu (ít thực hiện các thao tác insert hay update dữ liệu) và các tables này thường được kết hợp với nhau qua phép kết nối trong câu lệnh truy vấn dữ liệu.

Chọn lựa các cột sử dụng cho Cluster Key

Việc chọn lựa các cột dữ liệu sử dụng cho cluster key là khá quan trọng. Khi có nhiều cột dữ liệu được sử dụng trong các phép kết nối giữa nhiều tables của câu lệnh truy vấn, ta nên sử dụng cluster cho các tables trên với cluster key là nhóm của tất cả các cột dữ liệu đó. Thông thường, ta nên đặt index cho cluster tương ứng với các cột dữ liệu trong cluster key.

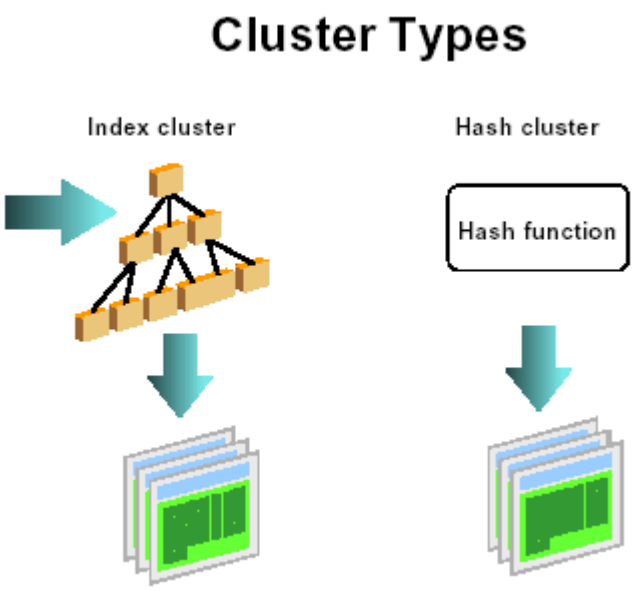
Một cluster key tốt là cluster key mà sử dụng nó ta có thể phân đều số lượng các rows (dòng) trên mỗi nhóm. Tránh chuyện có cluster key thì tương ứng với nhiều rows trong khi lại có những cluster key khác lại có quá ít các rows.

Khi có quá nhiều rows tương ứng với một cluster key sẽ làm cho việc tìm kiếm một dòng dữ liệu nào đó trong cluster trở nên tốn kém hơn. Mặt khác, đặt các Cluster keys đối với những cột có miền giá trị ít cũng không hiệu quả. Ví dụ như đặt cluster key đối với các cột nhận giá trị logic, miền giá trị của cột này chỉ là true và false cho nên cluster key trên cột dữ liệu này cũng không phát huy được hiệu quả.

13.1.3. Các kiểu cluster

Có hai kiểu cluster

- Index Cluster
- Hash Cluster



Hình vẽ 63. Các kiểu Cluster

Index cluster

Một index cluster sử dụng một index để bảo trì dữ liệu trong một cluster, các index này được gọi là cluster index:

- Cluster index phải có sẵn cho việc lưu trữ, truy xuất và bảo trì dữ liệu trong một index cluster.
- Index cluster được sử dụng để trỏ đến một block mà chứa các hàng với một giá trị khoá.
- Cấu trúc của cluster index tương tự như một normal index, mặc dầu normal index không chứa giá trị khoá `NULL`, nhưng cluster index có thể chứa giá trị khoá `NULL`. Chỉ có một điểm vào (entry) cho mỗi khoá trong cluster index, vì vậy chúng có thể nhỏ hơn so với index thông thường trong cùng một tập giá trị khoá.
- Để lưu trữ và truy xuất các hàng từ cluster, oracle server sử dụng cluster index để xác định hàng đầu tiên tương ứng với giá trị khoá sau đó truy xuất các hàng cho giá trị khoá.
- Nếu một vài hàng trong index cluster có cùng giá trị khoá, khoá cluster không lặp lại cho mỗi hàng. Trong một bảng với số lớn các hàng cùng giá trị khoá, sử dụng index cluster có thể làm giảm số lượng không gian đĩa cần lưu trữ.

Hash Cluster

Hash cluster sử dụng một chức năng tính toán vị trí của các hàng. Hàm băm (hash) sử dụng khoá cluster và có thể được người sử dụng định nghĩa hay hệ thống sinh ra.

Khi một hàng được chèn vào trong bảng trong một hash cluster:

- Cột khoá hash được sử dụng để tính giá trị băm (hash).

- Hàng được lưu dựa vào giá trị băm.

Hàm băm sử dụng để xác định các hàng trong khi trong khi truy xuất dữ liệu từ các bảng được băm, Hash cluster cho một hiệu năng lớn hơn so với index cluster.

13.1.4. Chọn lựa kiểu cluster

Lưu trữ table trong hash cluster là một cách nhằm làm tăng hiệu suất của hệ thống, tăng cường tốc độ trả về các dữ liệu của câu lệnh truy vấn. Sử dụng hash cluster là một trong số các cách chọn lựa giữa non-clustered tables có kèm theo index trên tables tương ứng, sử dụng index cluster và sử dụng hash cluster. Với một table có sử dụng index hay một index cluster, khi định vị dòng dữ liệu trong table, Oracle server sẽ sử dụng các key values (giá trị khoá) trong index để định vị. Với việc sử dụng hash cluster (cluster có sử dụng phương pháp băm), Oracle sẽ lưu trữ các rows của table trong một hash cluster và định vị dòng dữ liệu này thông qua một *hash function (hàm băm)*.

Oracle sử dụng hash function để tạo ra các giá trị dạng số, gọi là các *hash values (giá trị băm).* Các hash values được xác định dựa vào các giá trị của các cột dữ liệu. Giống như khoá của index cluster, khoá của một hash cluster cũng có thể bao gồm một hoặc nhiều cột dữ liệu.

Để xác định được một dòng dữ liệu trong indexed table thì ít nhất cần phải thực hiện 02 thao tác vào ra với:

- Cần tới ít nhất là một thao tác vào ra để xác định được key value trong index
- Và cần một thao tác vào ra nữa để đọc hoặc ghi dòng dữ liệu trong table hay cluster

Trái lại, với việc sử dụng hash function để xác định các row trong một hash cluster sẽ không cần phải sử dụng bất cứ một thao tác vào ra nào cả. Như vậy là dùng cách này có thể giảm bớt được các thao tác vào ra để đọc hay ghi dòng dữ liệu trong một hash cluster.

So sánh ưu nhược điểm của phương pháp Hashing và Indexing

Cả hai phương pháp indexing và hashing đều có những ưu và nhược điểm riêng. Do đó, ta cần căn cứ vào những trường hợp cụ thể riêng để chọn lựa phương pháp sử dụng cho hợp lý.

Ưu và nhược điểm của các phương pháp hashing và indexing:

- Các câu lệnh truy vấn sử dụng mệnh đề so sánh bằng đối với cluster key: `SELECT ... WHERE cluster_key = ...;` Trong những trường hợp này cluster key trong các điều kiện so sánh bằng sẽ được áp dụng hàm băm, giá trị băm - hash key tương ứng sẽ dùng để xác định dòng dữ liệu trong table. Trong khi đó, với indexed

table giá trị key value sẽ được tìm trong index trước, và sau đó mới đọc dòng dữ liệu thực sự trong table.

- Các tables trong hash cluster thường có kích thước ít thay đổi cho nên ta có thể xác định số lượng các rows và mức độ không gian sử dụng của các tables trong cluster. Khi các tables trong một hash cluster yêu cầu vùng không gian nhiều hơn là vùng không gian cấp phát cho cluster thì hiệu suất thực hiện của hệ thống sẽ bị giảm đi đáng kể do yêu cầu quá nhiều.
- Đối với các truy vấn trên các tables trả về nhiều dòng dữ liệu không bằng với key values (điều kiện truy vấn thuộc loại so sánh không bằng). Ví dụ, duyệt trên toàn bộ table để lấy ra các dòng dữ liệu dựa vào so sánh like (so sánh like áp dụng trên các chuỗi). Khi này, hash function không thể sử dụng để đưa ra các hash keys giống như trong trường hợp so sánh bằng được: `SELECT ... WHERE cluster_key<...;` Với index cluster, các key values được sắp xếp trong index, cho nên dễ dàng có thể đưa ra các cluster key values thoả mãn mệnh đề so sánh `WHERE` không tương đương, làm giảm bớt các thao tác truy xuất vào ra.
- Khi table có kích thước thường xuyên được tăng trưởng mà không bị giới hạn trên thì vùng không gian dành cho nó sẽ không cần thiết phải xác định ngay từ đầu. Với phương pháp hashing sẽ gây ra khó khăn vì kích thước của các cluster bao giờ cũng được xác định từ đầu. Trong khi sử dụng indexing, ta có thể điều chỉnh để tăng lượng không gian dành cho table khi cần thiết.
- Khi các ứng dụng thường xuyên thực hiện các thao tác duyệt toàn bộ table, nếu sử dụng phương pháp băm sẽ gây ra tốn kém hơn vì phải liên tục áp dụng hàm này.

13.2.QUẢN LÝ CLUSTER

13.2.1. Tạo cluster

Cú pháp:

- Tạo Cluster

```
CREATE CLUSTER [ schema. ]
      cluster (column dattype [, column datatype ] ... )
      [ PCTFREE integer ]
      [ PCTUSED integer ]
      [ INITRANS integer ]
      [ MAXTRANS integer ]
      [ SIZE integer [ K | M ] ]
      [ storage-clause ]
      [ TABLESPACE tablespace ]
      [ INDEX ]
```

Với:

schema	là owner của bảng
column	tên của cột làm khoá cho cluster
cluster	tên của cluster
data type	kiểu dữ liệu và kích thước của cột làm khoá

SIZE	chỉ định không gian yêu cầu bởi toàn bộ các hàng tương ứng với giá trị khoá, đơn vị tính byte, kilobyte hay megabyte.
INDEX	Chỉ định cluster tạo ra là một index cluster.

Lưu ý:

Nếu kích thước (SIZE) không được định nghĩa thì giá trị mặc định sẽ được sử dụng: 1 block. Oracle server sử dụng giá trị này để đánh giá số lượng lớn nhất giá trị khoá có thể chèn vào trong một block của cluster. Ví dụ nếu SIZE được khởi tạo giá trị 200, trong khi kích thước của một block là 2048 bytes, sau khi insert phần header còn lại khoảng 1900 bytes của block cho phép chèn dữ liệu.Vì vậy một block có thể chứa được 9 giá trị khoá.

- Tạo Index Cluster

```
CREATE INDEX [ schema. ] index
      ON CLUSTER [ schema. ] cluster
      [ PCTFREE integer ]
      [ INITRANS integer ]
      [ MAXTRANS integer ]
      [ TABLESPACE tablespace ]
      [ storage-clause ]
```

Cột khoá không cần thiết phải chỉ định cho cluster index vì chúng đã được định nghĩa khi tạo cluster. Đặt index cluster trong các tablespace khác nhau mà được sử dụng để tạo cluster.

- Tạo table trong Cluster

Để tạo một bảng trong một index cluster thực hiện các bước sau:

1. Tạo cluster
2. Tạo index cho cluster (cluster index)
3. Tạo bảng trên cluster

Có thể tạo bảng trên cluster trước khi tạo index. Ba bước trên cần thực hiện trước khi chèn dữ liệu vào trong bảng.

```
CREATE TABLE [schema.]table
      ( column_definition
      [, column_definition ]...
      [, [CONSTRAINT constraint] out_of_line_constraint ]...
      )
      CLUSTER [schema.]cluster (column [, column ]... )
```

Mệnh đề CLUSTER chỉ định rằng bảng phải được đặt trong một cluster.

Một bảng mà được đặt trong một cluster không thể có các thuộc tính vật lý của riêng nó bởi vì segment không phải của riêng nó mà là một phần của cluster.

Lưu ý: phải có cluster index thì mới thao tác được dữ liệu trong tables nằm trong cluster

Ví dụ:

- Tạo Cluster

```
CREATE CLUSTER scott.ord_clu
      (ord_no NUMBER(3))
      SIZE 200 TABLESPACE DATA01
      STORAGE(INITIAL 5M NEXT 5M PCTINCREASE 0);
```

- Tạo Index cho Cluster

```
CREATE INDEX scott.ord_clu_idx
      ON CLUSTER scott.ord_clu
      TABLESPACE INDX01
      STORAGE(INITIAL 1M NEXT 1M PCTINCREASE 0);
```

- Tạo table trong Cluster

```
CREATE TABLE scott.ord
      (ord_no NUMBER(3)
      CONSTRAINT ord_pk PRIMARY KEY,
      ord_dt DATE, cust_cd VARCHAR2(3))
      CLUSTER scott.ord_clu(ord_no);

CREATE TABLE scott.item
      (ord_no NUMBER(3) CONSTRAINT item_ord_fk
      REFERENCES scott.ord,
      prod VARCHAR2(5), qty NUMBER(3),
      CONSTRAINT item_pk PRIMARY KEY(ord_no,prod))
      CLUSTER scott.ord_clu(ord_no);
```

13.2.2. Tạo Hash Cluster

Để tạo một hash cluster và đặt bảng lên trên nó, thực hiện các bước sau:

- Tạo cluster
- Tạo bảng

Câu lệnh cho việc tạo một hash cluster tương tự như câu lệnh cho tạo index cluster.

Cú pháp:

```
CREATE CLUSTER [ schema. ]
      cluster (column datatype [, column datatype ] ... )
      HASHKEYS integer
      [ HASH IS expression ]
      [ PCTFREE integer ]
      [ PCTUSED integer ]
      [ INITRANS integer ]
      [ MAXTRANS integer ]
      [ SIZE integer [ K | M ] ]
      [ storage-clause ]
      [ TABLESPACE tablespace ]
```

Ví dụ:

- Tạo Cluster

```
CREATE CLUSTER scott.off_clu
      (country VARCHAR2(2),postcode VARCHAR2(8))
      SIZE 500 HASHKEYS 1000
```

```
                TABLESPACE DATA01
                STORAGE(INITIAL 5M NEXT 5M PCTINCREASE 0);
```

- Tạo các tables trong cluster

```
        CREATE TABLE scott.office(
                office_cd NUMBER(3), cost_ctr NUMBER(3),
                country VARCHAR2(2), postcode VARCHAR2(8))
                CLUSTER scott.off_clu(country,postcode);
```

Mệnh đề HASHKEYS chỉ định rằng cluster được tạo là một hash cluster.

Mệnh đề HASH IS có thể được sử dụng để định nghĩa một hàm băm (hash function) do người sử dụngđịnh nghĩa.

13.2.3. Xác định giá trị SIZE cho cluster

SIZE dùng để định nghĩa không gian sử dụng bởi tất cả các hàng cho khoá.

Được sử dụng cho cả hai kiểu cluster để đánh giá :

- Số cực đại giá trị khoá cho một block trong index cluster.
- Giá trị chính xác số giá trị khoá cho block trong hash cluster.

Tham số SIZE

Tham số SIZE định nghĩa không gian sử dụng bởi toàn bộ các hàng cho giá trị khoá của cluster. Ví dụ đối với index cluster nếu mỗi đơn đặt hàng có 10 mặt hàng và kích thước của mỗi hàng trong bảng là 20 bytes, một hàng trong bảng mặt hàng có kích thước là 18 bytes, tham số SIZE sẽ được tính như sau:

$$1 \text{ ORD row x } 20 \text{ bytes/ row} + 10 \text{ ITEM rows x } 18 \text{ bytes/ row} = 200 \text{ bytes}$$

Khi quyết định kích thước của SIZE cho Hash Cluster thì nên chọn giá trị lớn hơn giá trị tính một it để tránh việc không đủ không gian lưu trữ.

13.2.4. Các tham số chỉ định cho hash cluster

Có hai tham số chính được chỉ ra cho các hash cluster là

- HASHKEYS: số giá trị khoá
- HASH IS : Tuỳ chọn cho hàm băm do người dùng định nghĩa.

HASHKEYS

Từ khoá HASHKEYS chỉ định số giá trị khoá dùng cho hash cluster. Tham số này cùng với tham số SIZE được sử dụng bởi Oracle Server để đánh giá kích thước ban đầu của cluster. Không gian được thiết lập cho cluster trong quá trình tạo được xác đinh theo công thức:

$$\text{CEIL}\left(\frac{\text{HASHKEYS}}{\text{TRUNC}(\text{Available Block Space}/\text{SIZE})}\right)$$

Với:

$$\text{Available Block Space} = (\text{DB\_BLOCK\_SIZE} - \text{Header}) \times \left(1 - \frac{\text{PCTFREE}}{100}\right)$$

Các extent được thiết lập khi định nghĩa trong mệnh đề STORAGE sẽ tiếp tục đến khi tổng kích thước của các extent đã được thiết lập bằng hay lớn hơn giá trị tính toán.

HASH IS

Hàm băm có thể được định nghĩa bởi user, hàm được chọn cần xem xét tính phân tán của giá trị khoá :

- Có khả năng phân tán lớn nhất các hàng trong trong số các giá trị khoá.
- Cố gắng giảm thiểu số đụng độ (collision). Đụng độ sẽ xảy ra khi một vài giá trị khoá băm ra cùng một kết quả. Khi tình trạng đụng độ xảy ra sẽ gây ra không đủ không gian chứa tât cả các hàng , một con trỏ sẽ được đặt vào trong block, một bản ghi mới được đặt qua block mới.
- Định giá trị đến một giá trị nguyên dương.

Một hàm băm có thể chứa bất kì câu lệnh SQL hợp lệ nào, các hàm do user định nghĩa không thể tham gia vào hàm băm.

13.2.5. Sửa đổi các Cluster

Các sửa đổi đối với Cluster bao gồm

- Thay đổi tham số lưu trữ và các tham số liên quan đến sử dụng block.
- Thay đổi giá trị của tham số SIZE cho index cluster.
- Cấp phát và lấy lại không gian lưu trữ.
- Các giá trị HASH IS và HASHKEYS không thể thay đổi được bằng câu lệnh ALTER cho hash cluster.

Ví dụ:

```
ALTER CLUSTER scott.ord_clu
      SIZE 300 STORAGE (NEXT 2M);
```

Bảng quy định các thao tác có thể thực hiện cho index và hash cluster

Lệnh	Thao tác	Index Cluster	Hash Cluster
ALTER	Thay đổi các tham số sử dụng khối	Có	Có

CLUSTER	Thay đổi các thuộc tính lưu trữ (trừ INITIAL)	Có	Có
	Cấp phát vùng trống	Có	Có
	Thu hồi vùng không gian không sử dụng	Có	Có
	Thay đổi SIZE	Có	Không
	Thay đổi HASHKEYS và HASH IS	-	Không
TRUNCATE CLUSTER	Giải phóng không gian	Có	Không
	Tái sử dụng không gian	Có	Không

Các câu lệnh ALTER CLUSTER, TRUNCATE CLUSTER và ANALYZE CLUSTER có cùng một cú pháp tương ứng với câu lệnh cho bảng.

13.2.6. Xoá Cluster

Một cluster chỉ có thể xoá nếu như tất cả các bảng trên cluster đã được xoá đi, điều này có thể thực hiện bằng cách sử dụng mệnh đề tuỳ chọn INCLUDING TABLES trong câu lệnh xoá CLUSTER hay bằng cách xoá bảng trước khi xoá CLUSTER.

Cú pháp:

```
DROP CLUSTER [ schema. ] cluster
      [ INCLUDING TABLES [ CASCADE CONSTRAINTS ]]
```

Mệnh đề CASCADE CONSTRAINTS phải được sử dụng nếu bất cứ bảng nào trong cluster bị tham chiếu đến bởi một ngoại khoá (Foreign Key) của bảng mà không thuộc vào cluster.

Ví dụ:

- Sử dụng mệnh đề INCLUDE TABLES để xoá các bảng trong Cluster

```
DROP CLUSTER scott.ord_clu
      INCLUDING TABLES;
```

- Xoá các bảng trước khi xoá Cluster

```
DROP TABLE scott.ord;
      DROP TABLE scott.item;
      DROP CLUSTER scott.ord_clu;
```

Các tình huống sử dụng cluster

Index cluster hay Hash cluster được sử dụng trong các trường hợp khác nhau bảng dưới đây liệt kê nơi sẽ sử dụng index và hash cluster:

Chuẩn	Index	Hash
Phân tán khoá đồng nhất (Uniform key distribution)	X	X
Mở rộng sự đồng đều các giá trị khoá (**Evenly spread key values**)		X
Các khoá ít được cập nhật (**Rarely updated key**)	X	X
Các bảng kết nối Master-Detail	X	

Xác nhận số lượng các giá trị khoá		X
Các truy vấn sử dụng các khoá tương đương		X

Tính chất của dữ liệu

Hai đoạn sau đây mô tả liên quan giữa tính chất của dữ liệu trong cluster.

- Sự phân tán của giá trị khoá: Các cluster thì khá thích hợp trong tình huống tính tuần tự của giá trị khoá là như nhau. Xem trật tự xử lí của ví dụ sau: nếu các đơn đặt hàng chứa hầu hết số mặt hàng như nhau khi đó cluster có thể có ích. Nếu các đơn đặt hàng có chỉ một hay hai mặt hàng trong khi đó các đơn đặt hàng khác có hàng trăm mặt hàng trong trường hợp này sử dụng cluster là không thích hợp.
- Vùng giá trị khoá: Giá trị khoá phân tán trong vùng giá trị là không thích hợp cho khoá băm, bởi vì như vậy sẽ lãng phí rất nhiều không gian trong một cluster. Mặc dù thuật toán băm đã được thiết kế cẩn thận có thể giải quyết vấn đề này, rất khó khi thiết kế một hàm băm thích hợp để phát triển.

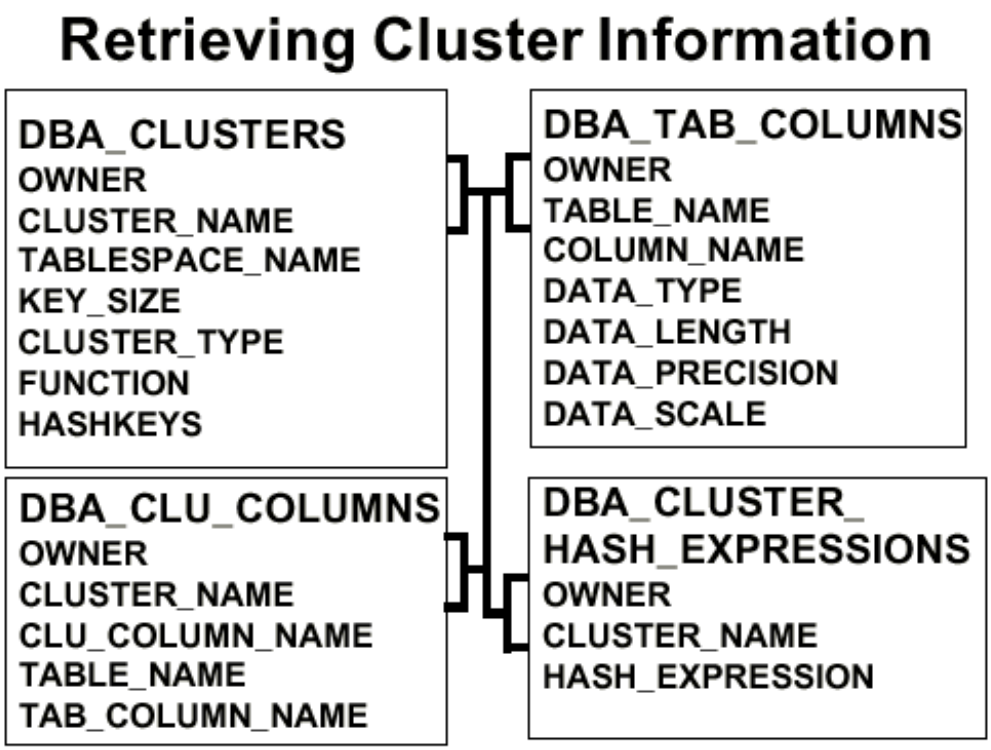
Các hoạt động trên dữ liệu trong các bảng

Phương thức dữ liệu đươc lưu trữ cần được xem xét trong khi thiết lập cluster.

- Tính tuần tự của việc cập nhật các cột khoá: Khi các giá trị khoá trong cluster được cập nhật, hàng có thể bị dịch chuyển vật lí từ block này sang block khác, điều này có thể làm giảm hiệu năng. Vì vậy các cột cần được cập nhật thường xuyên không thích hợp tham gia vào khoá của cluster.
- Tính tuần tự của join (kết nối): Các bảng mà thường xuyên tham gia vào kết nối trong câu lệnh query sử dụng quan hệ ngoại khoá là thích cho việc thiết lập cluster. Bởi vì các hàng dùng làm giá trị khoá được lưu trữ cùng nhau, sử dụng cluster có thể làm giảm số block cần đọc thoả mãn một yêu cầu nào đó. Trong một số trường hợp việc đặt bảng chi tiết (detail) trong cluster là rất có lợi, hãy xem xét trường hợp thông tin về các nhân viên luôn được truy xuất bởi các phòng ban và việc quét toàn bộ bảng phòng ban là thường xuyên xảy ra. Trong trường hợp này chỉ nên dùng bảng nhân viên trong cluster.
- Đánh giá số giá trị khoá: trong hash cluster, các hàm băm dựa trên tham số chỉ định bởi người dùng vào thời điểm tạo chúng. Việc đánh giá giá trị khoá hay kích thước chính xác là phức tạp bởi vì các bảng này thay đổi. Một lựa chọn sai cho `HASHKEYS` và `SIZE` có thể dẫn đến một hàm băm không hiệu quả và có thể dẫn đến lãng phí không gian lưu trữ, vì vậy hash cluster chỉ nên sử dụng khi mà giá trị của `SIZE` và `HASHKEYS` là có thể đánh giá được trước.
- Điều kiện query: Hàm băm có thể được sử dụng truy xuất dữ liệu chỉ khi giá trị khoá là được biết. Vì lí do này mà tìm kiếm trong một khoảng không thể sử dụng hàm băm. Nếu câu lệnh truy vấn sử dụng dựa vào tính ngang bằng trong khoá của cluster chúng có thể hỗ trợ cho quá trình băm. Tuy nhiên có thể tạo ra một index trên khoá của hash cluster, nếu nhiều truy vấn sử dụng tìm kiếm khoảng trong cột khoá của hash cluster.

13.3.THÔNG TIN VỀ CÁC CLUSTERS

Thông tin về Cluster được lấy trong từ điển dữ liệu: `DBA_CLUSTERS, DBA_TAB_COLUMNS, DBA_CLU_COLUMNS, DBA_CLUSTER_HASH_EXPRESSIONS`



Hình vẽ 64. Thông tin về Cluster

13.3.1. Xác định Cluster và các cột khoá Cluster

Để tìm tên của index cluster thuộc về user SCOTT và các cột làm khoá trong cluster sử dụng câu lệnh query sau:

```
SVRMGR> SELECT c.cluster_name, c.cluster_type, c.key_size,
2> cc.column_name, cc.data_type,
3> decode(cc.data_type, 'NUMBER',
4> decode(cc.data_precision,NULL,NULL,
5> cc.data_precision||','||cc.data_scale),
6> 'DATE', NULL, cc.data_length) AS "COLSIZE"
7> FROM dba_clusters c, dba_tab_columns cc
8> WHERE c.owner = cc.owner
9> AND c.cluster_name = cc.table_name
10> AND c. owner= 'SCOTT';

CLUSTER_NAME  CLUST    KEY_SIZE  COLUMN_NAME DATA_TYPE COLSIZE
------------  ------   -------   ----------   --------- -------
OFF_CLU       HASH     500       COUNTRY      VARCHAR2   2
OFF_CLU       HASH     500       POSTCODE     VARCHAR2   8
ORD_CLU       INDEX    300       ORD_NO       NUMBER     3,0
3 rows selected.
```

13.3.2. Lấy thông tin cột khoá của cluster và các cột trong bảng

Truy vấn cột DBA_CLU_COLUMNS để lấy thông tin về danh sách các cluster, các bảng trong cluster và tên các cột:

```
SVRMGR> SELECT *
2> FROM dba_clu_columns
3> WHERE owner='SCOTT'
4> ORDER BY cluster_name, table_name;

OWNER    CLUSTER_NAME    CLU_COLUMN_NA   TABLE_NAME TAB_COLUMN_NA
------   ------------    -------------   ---------- -------------
SCOTT    OFF_CLU         COUNTRY         OFFICE     COUNTRY
SCOTT    OFF_CLU         POSTCODE        OFFICE     POSTCODE
SCOTT    ORD_CLU         ORD_NO          ITEM       ORD_NO
SCOTT    ORD_CLU         ORD_NO          ORD        ORD_NO
4 rows selected.
```

13.3.3. Lấy thông tin cho hash cluster

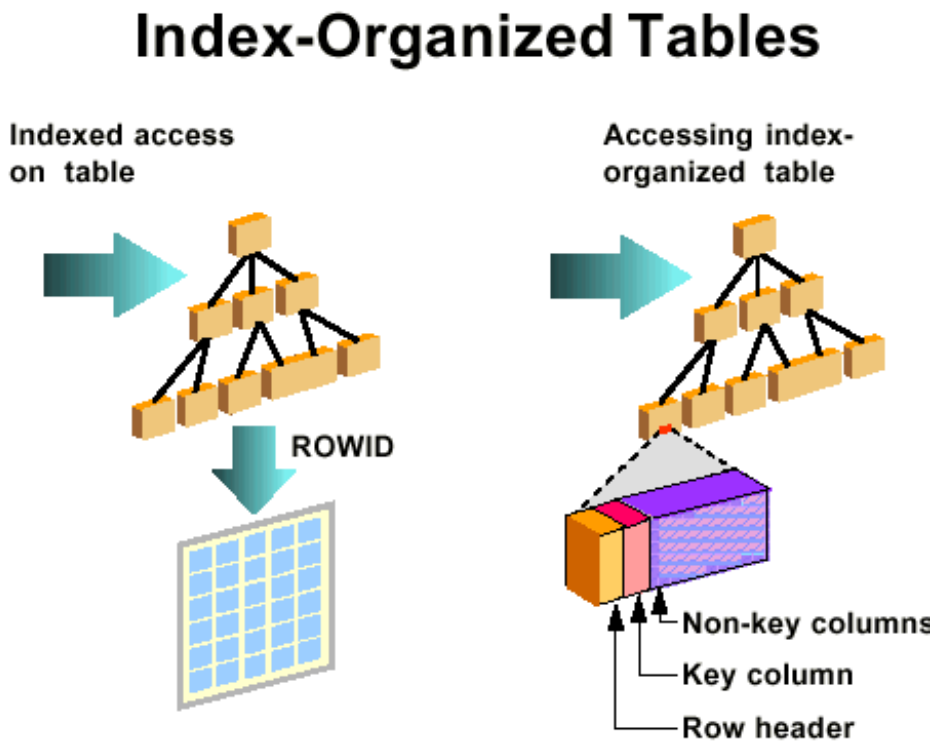
Để tìm số khoá hash và kiểu của hàm băm sử dụng, sử dụng câu lệnh query sau:

```
SVRMGR> SELECT c.cluster_name, c.hashkeys, c.function,
2> h.hash_expression
3> FROM dba_clusters c, dba_cluster_hash_expressions h
4> WHERE c.owner = h.owner(+)
5> AND c.cluster_name = h.cluster_name(+)
6> AND c.owner='SCOTT'
```

```
7> AND c.cluster_type='HASH';

CLUSTER_NAME        HASHKEYS   FUNCTION      HASH_EXPRESSION
-------------       ---------- ------------- ---------------
OFF_CLU             1009       DEFAULT2
1 row selected.
```

13.4.INDEX-ORGANIZED TABLE

13.4.1. Tính chất chung



Hình vẽ 65. Bảng được tổ chức theo kiểu index

Cấu trúc lưu trữ

Một index-organized table lưu toàn bộ dữ liệu cho bảng trong một cấu trúc B-TREE, B-TREE index dựa trên khoá chính của bảng và nó được tổ chức giống như một index. Các block ở phần lá trong cấu trúc này chứa các cột không phải là khoá thay vì các `ROWID`. Vì vậy việc sử dụng index-organized table cần 2 segments khác nhau một cho table và một cho index.

Truy xuất một index-organized table

Truy xuất index đối với các bảng thông thường yêu cầu một hay nhiều khối block index được đọc để truy xuất `ROWID` và việc vào ra trên bảng dựa vào giá trị của `ROWID`. Ngược lại đọc các bảng index-organized yêu cầu chỉ các khối index cần được đọc bởi vì toàn bộ hàng có sẵn trong phần là của node.

Một index-organized table có thể truy xuất sử dụng khoá chính hay nhiều cột kết hợp tham gia vào khoá chính. Việc thay đổi dữ liệu bảng như : thêm , xoá hay cập nhật các hàng dẫn đến kết quả cập nhật các index.

So sánh giữa index-organized table và các table

Index-Organized Table	Table
Khoá xác định duy nhất là `ROWID`	Xác định duy nhất bởi khoá chính
`ROWID` được ngầm định sử dụng cho các index	Không có `ROWID` Không có các index phụ

FTS trả về các dòng dữ liệu không theo một trật tự	Duyệt tất cả chỉ số và trả về các dòng dữ liệu theo thứ tự khoá chính
Cho phép các ràng buộc duy nhất	Không hỗ trợ các ràng buộc duy nhất
Hỗ trợ việc phân vùng, phân tán và lan truyền (replication) dữ liệu	Không hỗ trợ việc phân vùng, phân tán và lan truyền (replication) dữ liệu

Một index-organized table khác với table như sau:

- ROWID có thể là xác định duy nhất cho một hàng trong bảng thông thường, trong khi hàng trong index-organized table thì được xác định bởi khoá chính của bảng.
- Một bảng thông thường có thể có nhiều index. Các index đó lưu giá trị ROWID tương ứng cho giá trị khoá của index, trong khi đó đối với index-organnized table không dùng ROWID nên index thứ cấp sẽ không thể tạo ra trên kiểu bảng này. Bất cứ một câu lệnh SQL truy xuất giá trị ROWID của bảng index-organized table sẽ nhận được một lỗi.
- Khi thực hiện việc quét trên toàn bảng trên bảng thông thường trật tự các hàng trả về là không thể biết trước được. Ngược lại quá trình truy vấn trên toàn bộ bảng index-organized table các hàng trả về theo thứ tự của khoá chính trong bảng.
- Một index-organized table hỗ trợ tất cả các loại constraint ngoại trừ unique constraint.
- Một index-organized table không tham gia trong các distributed transactions (giao dịch phân tán), nó không thể phân khu (partition) hay nhân bản (replicate).

13.4.2. Tạo một index-organized table

Một index-organized table sử dụng trong trường hợp :

- Các ứng dụng thu hồi thông tin
- Các ứng dụng không gian
- Các ứng dụng xử lí phân tích trực tuyến (OLAP)
- Các bảng thường xuyên truy xuất sử dụng khoá chính của bảng.

Cú pháp:

```
CREATE TABLE [ schema. ] table
      (column-definition [, column-definition ] ...
      [,       out-of-line-constraint       [,       out-of-line-
      constraint ] ... ] )
      ORGANIZATION INDEX
      [ PCTFREE integer ]
      [ INITRANS integer ]
      [ MAXTRANS integer ]
      [ storage-clause ]
      [ TABLESPACE tablespace ]
      [ PCTTHRESHOLD integer
      [ INCLUDING column ] ]
      [ OVERFLOW segment_attributes_clause ]
```

Với:

schema	là owner của bảng
table	là tên của bảng
ORGANIZATION INDEX	chỉ định nó là một index-organized table
PCTTHRESHOLD	chỉ định không gian dành riêng trong index block cho các hàng của index-organized table (nếu các hàng vượt quá giá trị cơ sở đã được tính toán , toàn bộ các hàng trong cột đã được đặt tên trong mệnh đề INCLUDING được chuyển vào phân đoạn overflow, nếu phân đoạn overflow không được chỉ định các hàng vượt quá giá trị PCTTHRESHOLD sẽ bị từ chối, giá trị mặc định của nó là 50 và giá trị của nó phải nằm trong khoảng từ 0-50.
INCLUDING	column chỉ định cột tại đó chia một hàng của bảng index-organized table thành hai phần index và overflow, tất cả các cột tiếp theo giá trị column được lưu trong phần phân đoạn overflow.
OVERFLOW	chỉ định rằng các hàng dữ liệu trong index-organized table vượt quá giá trị chỉ định của PCTTHRESHOLD thì sẽ được đặt trong phân đoạn định nghĩa bởi segment_attributes_clause, mệnh đề này sẽ định nghĩa tablespace, vùng lưu trữ và các tham số về sử dụng block.

Ví dụ:

```
CREATE TABLE scott.sales
      ( office_cd NUMBER(3),
      qtr_end DATE,
      revenue NUMBER(10,2),
      review VARCHAR2(1000),
      CONSTRAINT sales_pk
      PRIMARY KEY(office_code, qtr_end))
      ORGANIZATION INDEX TABLESPACE data01
      PCTTHRESHOLD 20
      OVERFLOW TABLESPACE data02;
```

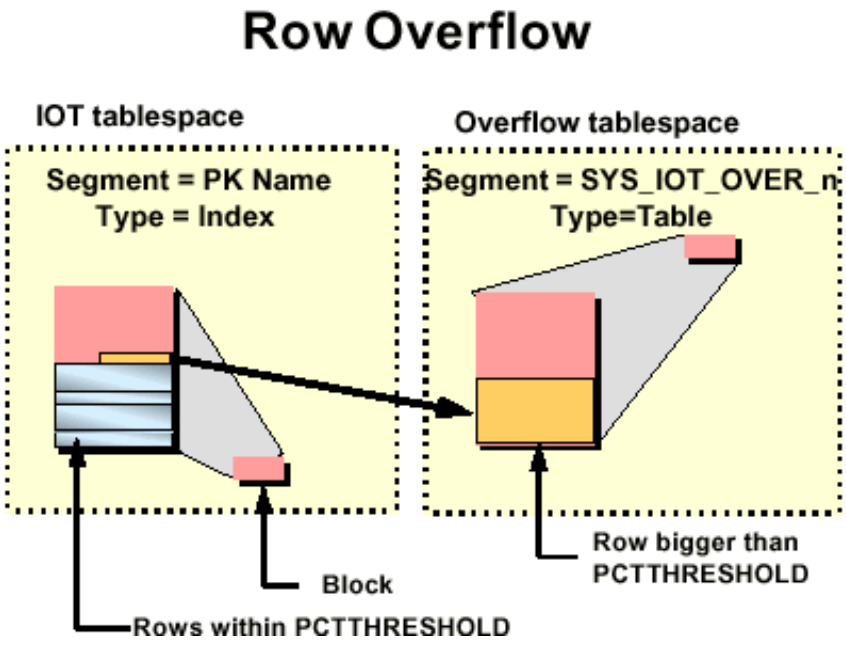
Một số giới hạn

Khoá chính phải chỉ định cho index-organized table, nếu tạo index-organized table trên bảng mà không có khoá chính sẽ sinh ra lỗi.

Nếu tham số PCTTHRESHOLD được định nghĩa và phân đoạn overflow không được chỉ định các hàng vượt quá giá trị PCTTRESHOLD sẽ bị từ chối và sẽ sinh ra một lỗi.

13.4.3. Hiện tượng ROW OVERFLOW (tràn dòng dữ liệu)

Một hàng lớn trong index-organized table có thể bị phá huỷ vùng lưu trữ dày đặc của các hàng trong index, vấn đề này có thể khắc phục được bằng cách sử dụng một vùng overflow.



Hình vẽ 66. Tràn dòng dữ liệu

Một hàng lớn trong một index-organzed table có thể phá huỷ mật độ vùng lưu trữ của các hàng trong index. Vấn đề này có thể giải quyết bằng cách sử dụng vùng dữ liệu overflow.

Tạo phân đoạn (segment) cho một index-organized table

Khi một index-organized table được tạo bằng cách chỉ định mệnh đề `OVERFLOW` các vùng sau đây được tạo:

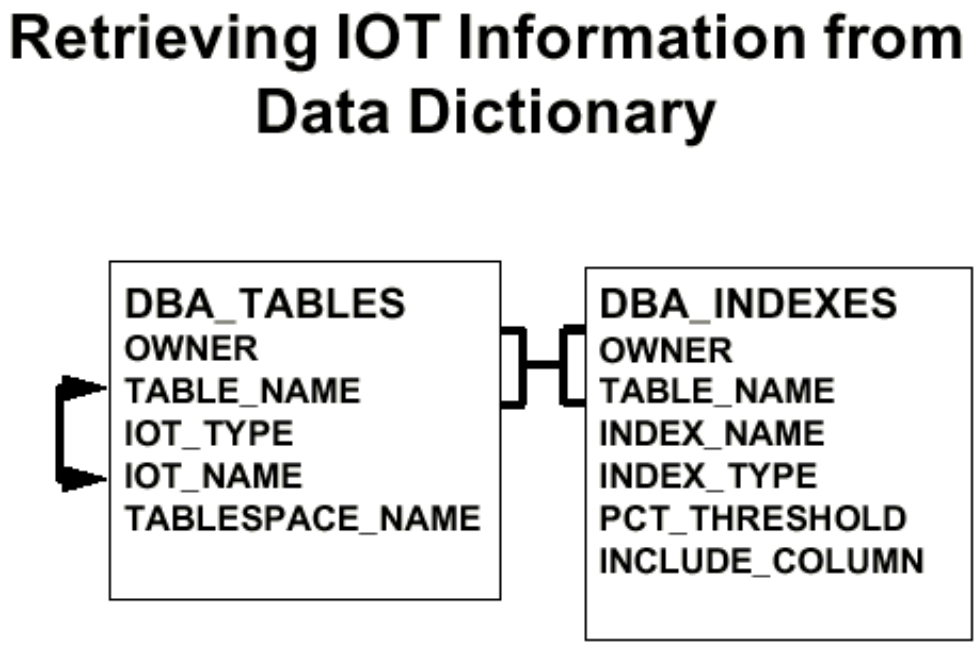
- Một bảng "logical" với tên định nghĩa trong mệnh đề `CREATE TABLE` được tạo ra. Vì toàn bộ các hàng được lưu trong index, không có phân đoạn mà tương ứng với bảng .
- Một index cùng tên với khoá chính của bảng nằm trong tablespace định nghĩa bởi câu lệnh `CREATE TABLE` , sử dụng các tham số lưu trữ cho khối chỉ định được tạo ra.
- Một bảng tiếp nhận các hàng bị overflow được tạo ra, tên của bảng này là `SYS_IOT_OVER_n`, trong đó n là giá trị tuần tự của object `OBJECT_ID` của index-organnized table.

Các hoạt động có thể trên index-organized table

Một index-organized table có thể sử dụng giống như các bảng thông thường, tất cả các câu lệnh giống như ALTER TABLE, ANALYZE TABLE, TRUNCATE TABLE và DROP TABLE và tất cả các câu lệnh DML đều có thể thực hiện trên index-organized table.

13.4.4. Lấy thông tin IOT (Index Orrganized Table)

Thông tin về IOT được lấy trong từ điển dữ liệu. Hai views cần quan tâm là DBA_TABLES và DBA_INDEXES



Hình vẽ 67. Thông tin về IOT

Sử dụng câu lệnh truy vấn sau đây liệt kê thông tin về các index-organized table và cấu trúc của chúng :

```
SVRMGR> SELECT t.table_name AS "IOT", o.table_name AS
"Overflow",
2> i.index_name AS "Index",
3> o.tablespace_name AS "Overflow TS",
4> i.tablespace_name AS "Index TS", i.pct_threshold
5> FROM dba_tables t, dba_tables o, dba_indexes i
6> WHERE t.owner=o.owner
7> AND t.table_name = o.iot_name
8> AND t.owner = i.owner
9> AND t.table_name = i.table_name
```

```
      10> AND t.owner= 'SCOTT';

IOT    Overflow          Index     Overflow TS  Index TS  PCT_THRESHOLD
-----  ----------------- --------  -----------  --------  -------------
SALES  SYS_IOT_OVER_2049 SALES_PK  DATA02       DATA01    20
```

Chương 14. QUẢN LÝ CÁC TABLES

14.1.TỔNG QUAN VỀ TABLES

14.1.1. Phân loại các tables

Có một số phương pháp lưu trữ dữ liệu người sử dụng. Trong Oracle database, dữ liệu có thể được lưu trong những đối tượng sau:

- Regular Tables
- Partition Tables
- Index_Organized Tables
- Clustered Tables

Regular Tables

Regular table (thường được goi là table) là hình thức thường hay được sử dụng để lưu trữ dữ liệu. Đây là những bảng dữ liệu được sử dụng theo mặc định và là đối tượng được tập trung nghiên cứu trong chương này. Quản trị viên database có thể điều khiển giới hạn các dòng dữ liệu phân tán trong một unclustered table. Các dòng dữ liệu có thể lưu trữ theo một trật tự tuỳ thuộc vào các thao tác dữ liệu được thực hiện trên bảng đó.

Partitioned Tables

Một partitioned table (bảng phân khu) cho phép xây dựng một ứng dụng ổn định. Partition table có một số đặc tính sau:

- Một partition table có thể có một hay nhiều partition, mỗi partition chứa các dòng dữ liệu thuộc vào một dãy giá trị của key value (giá trị khoá).
- Mỗi partition trong một parttioned table gọi là một segment (phân đoạn) và có thể đặt chúng trong các tablespaces khác nhau.
- Partition thường được sử dụng cho các tables có số lượng bản ghi lớn hay sử dụng các câu lệnh truy vấn dữ liệu và có nhiều thao tác dữ liệu đòi hỏi sử dụng đồng thời nhiều proccess (tiến trình).
- Có thể thực hiện trên đó một số câu lệnh đặc biệt nhằm hỗ trợ việc quản lý và thực hiện các thao tác dữ liệu trong các partition của partitioned table.

Hai loại Index_Organized Tables và Clustered Tables sẽ được xem xét chi tiết hơn trong các chương tiếp theo.

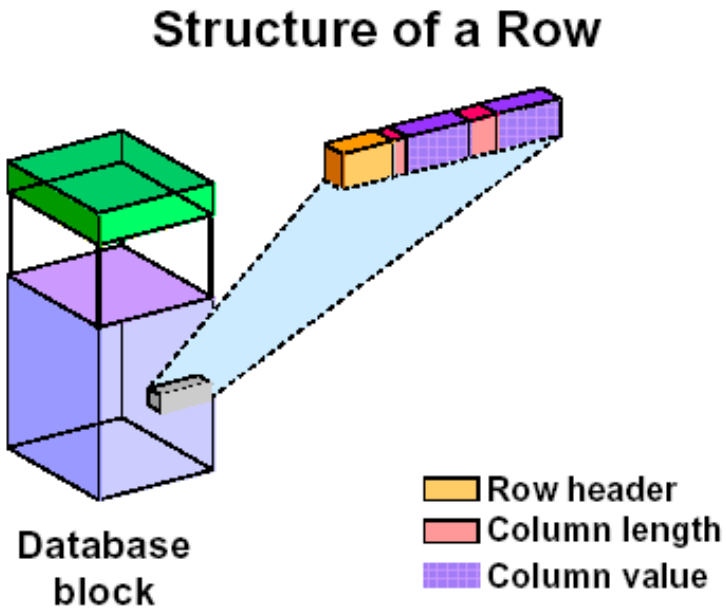
14.1.2. Cấu trúc các dòng dữ liệu (row data)

Các dòng dữ liệu (rows data) được lưu trong các database blocks. Các trường dữ liệu trên mỗi row được lưu trữ theo một trật tự giống như trật tự của các cột dữ liệu (columns) khi định

nghĩa table. Các trường có giá trị NULL sẽ không được lưu trữ. Mỗi row trong table có thể có số lượng các trường dữ liệu khác nhau.

Mỗi row trong table đều có:

- Row header (phần thông tin đầu của dòng dữ liệu): lưu trữ số lượng các trường trong dòng dữ liệu đó, ngoài ra còn có thông tin về chaining và thông tin về trạng thái khoá của dòng dữ liệu đó.
- Row data (nội dung của dòng dữ liệu lưu trữ): đối với mỗi trường dữ liệu, oracle lưu độ dài của trường dữ liệu và giá trị của trường dữ liệu đó (có một byte dành riêng dùng để lưu độ dài của trường dữ liệu nếu độ dài của của nó không vượt quá 250 bytes). Giá trị của cột sẽ lưu ngay tiếp theo thông tin về độ dài của cột.



Hình vẽ 68. Cấu trúc dòng dữ liệu dữ liệu

Các rows được lưu trữ liền kề nhau và không cần bất cứ khoảng cách nào giữa chúng. Mỗi row trong block đều có các thông tin (slot) trong danh mục các rows. Danh mục các thông tin này trỏ đến phần đầu của mỗi row.

Oracle 8i cung cấp một vài kiểu dữ liệu xây dựng sẵn dùng để lưu trữ các dữ liệu có kiểu vô hướng, kiểu tập hợp và kiểu quan hệ.

14.2.CÁC KIỂU DỮ LIỆU TRONG TABLE

14.2.1. Kiểu dữ liệu vô hướng

Dũ liệu kí tự (character)

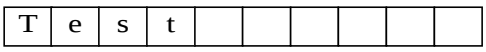
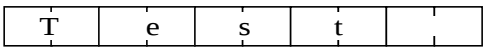
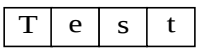
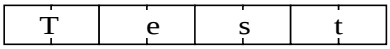
Dữ liệu kí tự có thể lưu trữ theo kiểu chuỗi có độ dài không đổi (fixed length) hoặc có độ dài thay đổi được (variable length) trong database.

Kiểu kí tự có độ dài không thay đổi như là CHAR, NCHAR được lưu trữ gắn thêm các khoảng trống (blanks). NCHAR là kiểu dữ liệu NLS (kiểu dữ liệu có hỗ trợ đặc tính ngôn ngữ của từng quốc gia) cho phép lưu trữ các tập kí tự có độ rộng không thay đổi hay có độ rộng thay đổi (fixed width, variable width). Kích thước cực đại được quyết định bởi số bytes dùng để lưu trữ một kí tự, với một giới hạn trên là 2000 bytes cho một row.

Kiểu dữ liệu CHAR phù hợp với việc lưu trữ xâu ký tự thuộc bảng mã ASCII. Trong khi kiểu dữ liệu NCHAR phù hợp với việc lưu trữ xâu ký tự thuộc bảng mã phức tạp hơn, bảng mã unicode chẳng hạn. Khi này, mỗi ký tự lưu trữ có thể có kích thước lớn hơn một byte (Ví dụ: ký tự chữ Trung Quốc, Nhật Bản,..).

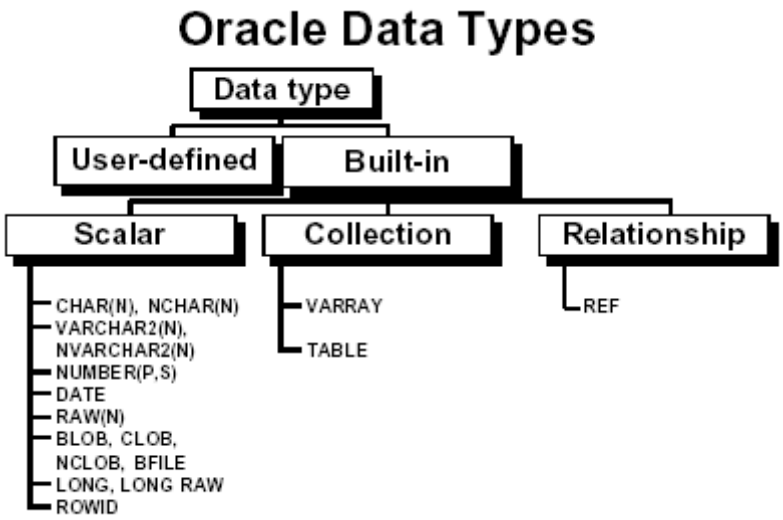
Lưu trữ dữ liệu bằng kiểu CHAR, NCHAR nhiều khi gây ra hiện tượng tốn kém bộ nhớ một cách không cần thiết. Kiểu dữ liệu chuẩn VARCHAR và NVARCHAR có thể khắc phục được nhược điểm này. Với việc sử dụng các kiểu dữ liệu VARCHAR và NVARCHAR để lưu trữ dữ liệu, nếu nội dung của dữ liệu lưu trữ ít hơn kích thước khai báo thì hệ thống sẽ chỉ cấp vừa đủ bộ nhớ để lưu trữ xâu ký tự mà thôi.

Ví dụ minh hoạ việc lưu trữ dữ liệu xâu chữ giữa các kiểu dữ liệu khác nhau

Dữ liệu	Bảng mã	Kiểu dữ liệu	Lưu trữ dữ liệu
Test	8 bits	CHAR(10)	10 bytes 
Test	16 bits	NCHAR(5)	10 bytes 
Test	8 bits	VARCHAR(10)	4 bytes 
Test	16 bits	NVARCHAR(5)	8 bytes 

VARCHAR và NVARCHAR là hai kiểu dữ liệu chuẩn được Oracle hỗ trợ từ các phiên bản đầu. Với các phiên bản sau của Oracle, có hỗ trợ thêm kiểu dữ liệu VARCHAR2 và NVARCHAR2. Hai kiểu dữ liệu này cũng tương tự như VARRCHAR và NVARCHAR, tuy nhiên hai kiểu dữ liệu này được hỗ trợ xử lý tốt hơn và còn được tiếp tục hỗ trợ trong các phiên bản tiếp theo của Oracle. Oracle khuyến cáo người sử dụng nên dùng kiểu dữ liệu VARCHAR2 và NVARCHAR2 thay cho kiểu các dữ liệu cũ là VARCHAR và NVARCHAR.

Kiểu dữ liệu kí tự có độ dài thay đổi sẽ chỉ sử dụng một số bytes cần thiết để lưu trữ giá trị thực sự của cột và có thể thay đổi kích thước cho mỗi hàng. VARCHAR2 và NVARCHAR2 là ví dụ của kiểu dữ liệu kí tự có độ dài thay đổi.



Hình vẽ 69. Các kiểu dữ liệu trong Oracle

Dữ liệu kiểu số (numeric)

Dữ liệu kiểu số trong Oracle Database luôn được lưu trữ với kiểu dữ liệu có độ dài thay đổi. Chúng có thể lưu trữ được những con số lên tới 38 chữ số.

Dữ liệu kiểu số có:

- Một byte để lưu phần mũ
- Một byte để lưu hai con số phần định trị.
- Một byte để lưu số âm.

Kiểu dữ liệu ngày tháng (date)

Oracle server lưu dữ liệu kiểu date trong một trường có độ dài không thay đổi là 7 bytes. Dữ liệu kiểu date của Oracle bao giờ cũng bao gồm thời gian đầy đủ: thế kỷ, năm, tháng, ngày, giờ, phút, giây và phần trăm của giây.

Kiểu dữ liệu thô (raw)

Kiểu dữ liệu này cho phép lưu trữ các dữ liệu nhị phân nhỏ. Oracle server không thực hiện chuyển đổi tập kí tự mỗi khi dữ liệu kiểu raw được chuyển qua lại giữa các máy trong mạng (khi dữ liệu kiểu raw được dịch chuyển từ database này sang database khác sử dụng công cụ của Oracle).

Kiểu dữ liệu lưu trữ đối tượng lớn (LOB)

LONG, LONG RAW	LOB
Một cột cho một bảng	Nhiều cột cho một bảng
Kích thước có thể tới 2GB	Có thể lên tới 4GB
SELECT trả về dữ liệu	SELECT trả về locator

Dữ liệu in-line (có thể lưu trữ trong một dòng của table)	Dữ liệu có thể là in-line hay out-of-line (dữ liệu không lưu trữ được trong bảng mà phải lưu riêng trong một tablespace, có sử dụng LOB locator để xác định dữ liệu LOB)
Không hỗ trợ Object type	Hỗ trợ kiểu object
Truy nhập tuần tự các bó (chunk)	Truy xuất chunk không tuần tự

Ngoài ra, Oracle cung cấp 6 kiểu dữ liệu cho việc lưu trữ các đối tượng lớn:

- `CLOB` và `LONG` để dữ liệu kí tự có độ rộng không đổi.
- `NCLOB` để lưu dữ liệu kí tự NLS có độ rộng không đổi.
- `BLOB` và `LONG RAW` cho các dữ liệu phi cấu trúc .
- `BFILE` để lưu trữ các dữ liệu phi cấu trúc trong hệ điều hành.

`LONG` và `LONG RAW`, trước đây, thường được sử dụng để lưu trữ dữ liệu phi cấu trúc như image, document hay các thông tin vật lý. Ở các phiên bản gần đây, Oracle 8i, kiểu dữ liệu này được thay thế bằng kiểu dữ liệu `LOB`. Kiểu dữ liệu `LOB` khác với dữ liệu `LONG` và `LONG RAW` cho nên chúng không thể dùng lẫn với nhau. `LOB` không hỗ trợ cho các chương trình viết với `LONG` và ngược lại.

Kiểu dữ liệu RowID

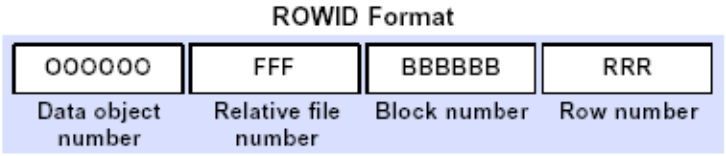
RowID là toán tử giả có thể được sử dụng trong câu lệnh truy vấn cùng với các cột dữ liệu có trong bảng. RowID có một số đặc tính sau:

- RowID là định danh duy nhất cho một row trong database.
- RowID không lưu trữ rõ ràng như các cột giá trị.
- Mặc dù RowID không phải là địa chỉ vật lý của một row nhưng nó vẫn có thể sử dụng để xác định vị trí của một row.
- Sử dụng RowID cho phép truy xuất nhanh chóng các rows của một table.
- RowID còn được lưu trữ trong Index để chỉ định rõ từng rows tương ứng với từng giá trị khoá (key values).

Định dạng của RowID

Cần 10 bytes để lưu trữ một giá trị RowID trên đĩa và hiển thị nó bởi 18 kí tự. Định dạng của một RowID bao gồm các thành phần:

- Data object number: được gán cho mỗi data object, ví dụ như: table hay index. Khi các Objects này được tạo lập, giá trị data object number tương ứng sẽ được khởi tạo và được quy định duy nhất trong database.
- Relative file number: là số hiệu duy nhất ứng với mỗi file trong một tablespace.
- Block number: dùng xác định vị trí của Block chứa dòng dữ liệu trong file.
- Row number: để xác định vị trí của từng row trong danh mục các rows thuộc block header.



Hình vẽ 70. Định dạng của một RowID

Trong đó, data object number cần 32 bits, relative file number cần 10 bits, block number cần 22 bits và row number cần 16 bits tổng số bits dùng để lưu thông tin về RowID là 80 bits hay 10 bytes.

Khi hiển thị một RowID theo bộ mã 64, ta cần tới 6 vị trí cho data object number, 3 vị trí cho relative file number, 6 vị trí tiếp theo cho block number và 3 vị trí cuối cùng cho row number.

Bộ mã 64 sử dụng các kí tự " A-Z", "a-z","0-9","+ / " tổng cộng là 64 kí tự.

Ví dụ:

```
SVRMGR> SELECT deptno, ROWID
2> FROM scott.dept;

DEPTNO     ROWID
--------   ------------------
10         AAAArsAADAAAAUaAAA
20         AAAArsAADAAAAUaAAB
30         AAAArsAADAAAAUaAAC
40         AAAArsAADAAAAUaAAD
4 rows selected.
```

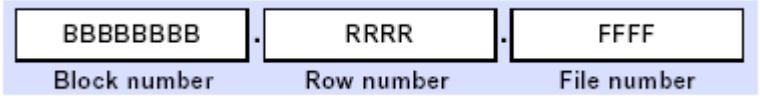
Trong đó :

- AAAArs là giá trị của Data object number
- AAD là giá trị của Relative file number
- AAAAUa là giá trị của Block number
- AAA là giá trị của Row number

Xác định vị trí của row nhờ giá trị của RowID

Vì một segment chỉ có thể nằm trong một tablespace nên ta có thể sử dụng Data Object Number để xác định tablespace chứa row. Giá trị Relative File Number trong tablespace dùng để xác định file. Giá trị Block Number dùng để xác định Block chứa row và giá trị Row Number xác định chính xác row trong danh mục các rows.

RowID bị giới hạn (Restricted RowID)



Hình vẽ 71. Giới hạn của RowID

Phiên bản trước đây của Oracle có sử dụng định dạng Restricted RowID. Một Restricted RowID sử dụng 6 bytes và không chứa giá trị Data Object Number. Định dạng này sử dụng trong Oracle 7 và các phiên bản trước đó. Do File Number là duy nhất trong database nên tại các phiên bản trước của Oracle không cho phép có nhiều hơn 1024 data file.

Mặc dù Oracle 8 đã khắc phục giới hạn trên bằng cách sử dụng Tablespace_Relative nhưng File Number trong mỗi Restricted RowID vẫn được sử dụng trong một Objects (Ví dụ như nonpartitioned indexes). Khi này, các Index tham chiếu đến các rows cũng nằm trên cùng một segment chứa các rows.

14.2.2. Tập hợp (collection)

Có hai kiểu dữ liệu tập hợp sử dụng để lưu trữ các dữ liệu có tính lặp lại trong các rows của một table. Việc chọn lựa kiểu dữ liệu tập hợp thích hợp là một việc làm cần thiết.

Mảng biến (varying arrays)

Varying arrays dùng để lưu các thông tin danh sách chứa một số lượng nhỏ các yếu tố, ví dụ như số điện thoại của các khách hàng.

Varying arrays có các đặc tính sau:

- Là một mảng có thứ tự các yếu tố.
- Tất cả các thành phần trong mảng có cùng kiểu dữ liệu.
- Mỗi thành phần có một chỉ số (index), đó là con số tương ứng với vị trí của thành phần trong mảng, chúng được đánh số từ 0 đến n-1.
- Số lớn nhất của các thành phần trong mảng chính là kích thước của mảng.
- Oracle cho phép mảng có thể có kích thước thay đổi. Vì vậy, người ta gọi chúng là các VARRAYs. Kích thước cực đại của VARRAYS cần được chỉ định khi mô tả
- Các phần tử trong mảng được đánh chỉ số một cách liên tục.

Các bảng lồng nhau (nested table)

Các nested table cung cấp một phương thức định nghĩa một bảng như là một cột dữ liệu trong một bảng khác. Có thể sử dụng phương pháp này để lưu trữ một lượng lớn các bản ghi (ví dụ như các items (mục) trong một đơn hàng).

Nested table có các đặc tính sau:

- Một nested table là một tập không sắp xếp các bản ghi hay rows.
- Các rows trong nested table có cùng một cấu trúc.
- Các rows trong nested table được lưu trữ tách rời với với bảng cha và có một con trỏ trỏ tới với row tương ứng trên bảng cha.
- Các tham số lưu trữ cho nested table có thể được chỉ định bởi người quản trị database.
- Không có giới hạn việc lồng các bảng.

14.2.3. Kiểu quan hệ (REF)

Kiểu quan hệ được dùng như là con trỏ trong database. Việc sử dụng kiểu này đòi hỏi tuỳ chọn OBJECT. Ví dụ, mỗi Item trong đơn đặt hàng có thể trỏ đến hay tham chiếu đến một hàng trong bảng PRODUCTS, mà không cần lưu trữ mã của các sản phẩm.

Kiểu dữ liệu do người sử dụng định nghĩa (User Defined Type)

Oracle database cho phép người sử dụng định nghĩa kiểu dữ liệu và sử dụng chúng trong ứng dụng, sử dụng đặc tính này cần chọn tuỳ chọn OBJECT.

14.2.4. Kiểu dữ liệu TIMESTAMP

Trong phiên bản Oracle 9i, ta có thêm mộ kiểu dữ liệu mới, gọi là kiểu TIMESTAMP. Kiểu dữ liệu này cho phép ta lưu trữ dữ liệu dates, time với cấp chính xác 9 số thân phân của đơn vị giây.

Oracle cung cấp một số hàm phục vụ cho việc chuyển đổi kiểu liên quan:

- TO_TIMESTAMP: chuyển đổi String sang Timestamp.
- TO_TIMESTAMP_TZ: chuyển đổi String thành Timestamp có kèm Time Zone.
- TO_DSINTERVAL: chuyển đổi String thành Interval Day to Second.
- TO_YMINTERVAL: chuyển đổi String thành Interval Year to Month
- TO_CHAR: chuyển đổi sang khuôn dạng characters.
- EXTRACT: trả về các giá trị yêu cầu (dạng một số - number) tử một giá trị datetime hay interval datatype. Options are Year, Month, Day, Hour, Minute, Second, Timezone_Hour, Timezone_Minute, Timezone_Region, or Timezone_ABBR.

Ví dụ:

```
SELECT EXTRACT(YEAR FROM SYSDATE) FROM DUAL;
```

14.3. QUẢN LÝ CÁC TABLES

14.3.1. Tạo table

Ta có thể tạo table thông qua câu lệnh SQL

Cú pháp:

```
CREATE TABLE [schema.] table
      (column datatype[ , column datatype ].....)
      [TABLESPACE tablespace]
      [PCTFREE interger]
      [PCTUSED interger]
      [INITTRANS interger]
      [MAXTRANS interger]
      [STORAGE storage_clause]
```

```
[LOGGING|NOLOGGING]
[CACHE|NOCACHE]
```

Với:

schema	tương ứng với user sở hữu table.
table	tên của bảng tạo
column	tên của cột trong bảng cần tạo
datatype	kiểu dữ liệu cho cột tương ứng
TABLESPACE	tên tablespace chứa bảng
PCTFREE	không gian dành riêng trong mỗi block (tính bằng đợn vị %). Sử dụng chứa khi các hàng lớn lên do update.
PCTUSED	xác định giới hạn dưới của không gian sử dụng trong block
INITRANS	xác định số giao dịch được thiết lập cho mỗi block
MAXTRANS	xác định số giao dịch lớn nhất có thể thiết lập cho block mặc định là 255.
STORAGE	quy định kích thước của không gian lưu trữ, xác định xem có bao nhiêu extents sẽ được thiết lập cho bảng.
LOGGING	chỉ định việc tạo bảng sẽ được ghi vào trong redo log file.
NOLOGGING	chỉ định việc tạo bảng và nạp dữ liệu vào bảng sẽ không được ghi vào redo log file.
CACHE	chỉ định việc truy xuất các blocks của bảng được thiết lập trong vùng đệm khi có thực hiện full scan trên table.
NOCACHE	chỉ định các blocks được truy xuất trên bảng này không được đặt vào trong danh sách LRU trong vùng đệm khi có thực hiện full scan trên table.

Ví dụ:

```
CREATE TABLE employees(
      empno NUMBER(4),
      last_name VARCHAR2(30)
      deptno NUMBER(2))
      PCTFREE 20 PCTUSED 50
      STORAGE(INITIAL 200K NEXT 200K
      PCTINCREASE 0 MAXEXTENTS 50)
      TABLESPACE data01;
```

Lưu ý:

- Mỗi table nên có một Primary Key.
- Nếu giá trị `MINIMUM EXTENT` được chỉ ra cho tablespace thì khi mở rộng kích thước bảng, giá trị kích thước sẽ được làm tròn lên một bội số lần giá trị của `MINIMUM EXTENT`.
- Nếu bỏ qua mệnh đề `[NO]LOGGING` thì thuộc tính logging của bảng sẽ được đặt mặc định theo thuộc tính logging của tablespace chứa bảng đó.
- Nếu giá trị `MINEXTENT` được chỉ định bởi một giá trị lớn hơn 1 và tablespace chứa nhiều data file, quá trình mở rộng sẽ thực hiện trên nhiều data files khác nhau tương ứng với tablespace.

- Nên đặt các bảng trên các tablespace riêng, không đặt các bảng trên Rollback Tablespace, Temporary Tablespace hay Index Tablespace.
- Sử dụng chuẩn về kích thước mở rộng (extent size) là một bội số của 5*DB_BLOCK_SIZE để giảm thiểu sự phân mảnh trong database.
- Để nâng cao hiệu suất thực hiện truy vấn trên toàn bộ bảng, cần thiết lập thông số extent size với giá trị bằng giá trị DB_FILE_MULTIBLOCK_READ_COUNT. Đây là tham số quy định số lượng các blocks được đọc mỗi khi server proccess thực hiện việc đọc dữ liệu qua phép truy xuất file dữ liệu của hệ điều hành.
- Mệnh đề CACHE chỉ dùng cho các bảng có kích thước nhỏ và thường xuyên được truy vấn.

Trong OEM, ta thực hiện theo các bước sau

1. Chạy Oracle Schema Manager.
2. Chọn Object—>Create.
3. Chọn Table từ danh sách rồi bấm nút OK.
4. Chọn Create Table Manually trong phần New Table.
5. Bấm nút OK.
6. Nhập vào các thông tin trong phần General, Storage, và Options.
7. Bấm nút Create.

14.3.2. Thiết lập giá trị PCTFREE và PCTUSED

PCTFREE

Khi giá trị PCTFREE lớn thì không gian dành cho insert dữ liệu sẽ lớn hơn không gian cho update dữ liệu. Thiết lập giá trị này lớn để dự phòng cho một số trường hợp:

- Table có nhiều cột dữ liệu nhận giá trị NULL lúc đầu nhưng sau đó nó lại được cập nhật bởi một giá trị khác NULL.
- Các cột dữ liệu trong table được mở rộng kích thước mỗi khi nó được cập nhật bởi một giá trị khác có độ rộng lớn hơn.

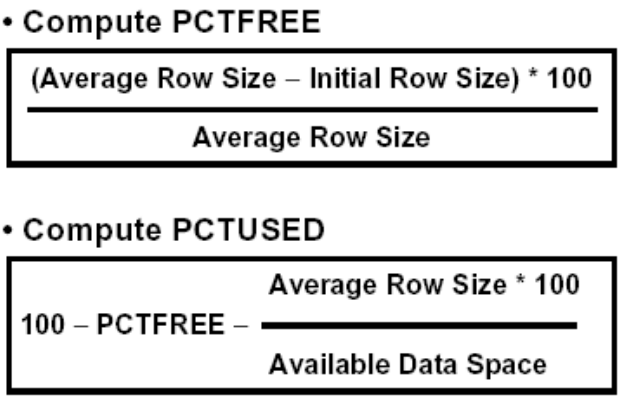
Một giá trị PCTFREE lớn sẽ làm cho mật độ hàng trong block thấp đi. Mỗi block sẽ cho phép có ít hàng hơn được lưu trữ.

PCTUSED

Tham số PCTUSED được xác định nhằm đảm bảo cung cấp đủ số lượng block trống phục vụ công việc lưu trữ dữ liệu của table. Các blocks cung cấp cho table được lấy từ một danh sách các block rỗng. Khi table cần thêm block để lưu trữ, Oracle server sẽ tìm một Block trống tiếp theo trong danh sách các block rỗng này để cung cấp cho table. Quá trình tìm kiếm tuyến tính xảy ra cho đến khi hoặc là tìm thấy một block rỗng trong danh sách hoặc tìm đến cuối cùng của danh sách.

Ta có thể sử dụng công thức dưới đây để xác định giá trị của tham số PCTFREE. Giá trị của tham số được tính bởi công thức này có thể làm giảm thời gian tìm kiếm trong danh sách các block rỗng và tăng khả năng tìm kiếm khi cần thêm không gian sử dụng.

Công thức tính cho các giá trị PCTFREE và PCTUSED



Hình vẽ 72. Công thức tính PCTFREE và PCTUSED

Chú ý:

Giá trị kích thước trung bình (average row size) có thể đánh giá từ việc sử dụng câu lệnh ANALYZE TABLE.

14.3.3. Migration (di trú) và Chaining các dòng dữ liệu

Migration (di trú) dòng dữ liệu

Nếu giá trị PCTFREE được khởi tạo bởi một giá trị nhỏ thì sẽ có thể không đủ không gian cần thiết trong quá trình tăng trưởng của các blocks (ví dụ như update dữ liệu trong các rows bởi một dữ liệu khác có độ rộng lớn hơn). Khi đó, oracle server sẽ chuyển toàn bộ row sang một block mới và thay đổi con trỏ từ block cũ sang block mới. Quá trình này goi là quá trình migration (di trú) của một row. Khi thực hiện di trú một row, hiệu năng tìm kiếm các rows sẽ bị giảm đi do Oracle server cần phải quét 2 block dữ liệu để xác định row dữ liệu cần tìm.

Chaining dòng dữ liệu

Hiện tượng chaining các rows xảy ra khi insert một row quá lớn vào một block. Điều này xảy ra khi row đó chứa các cột dữ liệu có kích thước lớn. Trong trường hợp này, Oracle server sẽ chia các rows thành nhiều đoạn nhỏ (gọi là chunk). Mỗi chunk được lưu trữ trong một block cùng với thông tin con trỏ để truy xuất nó. Tập hợp nhiều chunks cho phép lưu được toàn bộ dữ liệu của row.

Có thể giảm thiểu hiện tượng chaining các rows bằng cách đặt giá trị kích thước của block là lớn hoặc cũng có thể tách các table thành nhiều tables nhỏ hơn mà tại các tables nhỏ này có ít cột hơn.

14.3.4. Sao chép một tables

Ta có thể sử dụng câu lệnh `CREATE TABLE` để sao chép một table đang tồn tại:

Cú pháp:

```
CREATE TABLE [schema.]table
      [LOGGING|NOLOGGING]
      ....
      AS
      Subquery
```

Ví dụ:

```
CREATE TABLE new_emp
      STORAGE(INITTIAL 200K NEXT 200K PCTINCREASE 0 MAXEXTENTS
      50)
      NOLOGGING
      TABLESPACE data01
      AS
      SELECT * FROM scott.emp;
```

Lưu ý:

Các mệnh đề `TABLESPACE, STORAGE` hay thông tin sử dụng các blocks có thể được chỉ ra khi tạo bảng. Sử dụng mệnh đề `NOLOGGING` nếu muốn bỏ qua việc sinh ra các thông tin log trong redo log file và tăng tốc độ tạo bảng.

Khi thực hiện sao chép các table, các contraints (ràng buộc), triggers (một thủ tục được tự động kích hoạt khi có thao tác trên dữ liệu) và privileges (quyền) trên table gốc sẽ không được sao chép sang table mới. Để có được những thứ này, ta phải tạo bằng tay.

Nếu một column đã được quy định là `NOT NULL` trong table gốc, các cột tương ứng trong bảng mới cũng sẽ được quy định là `NOT NULL`.

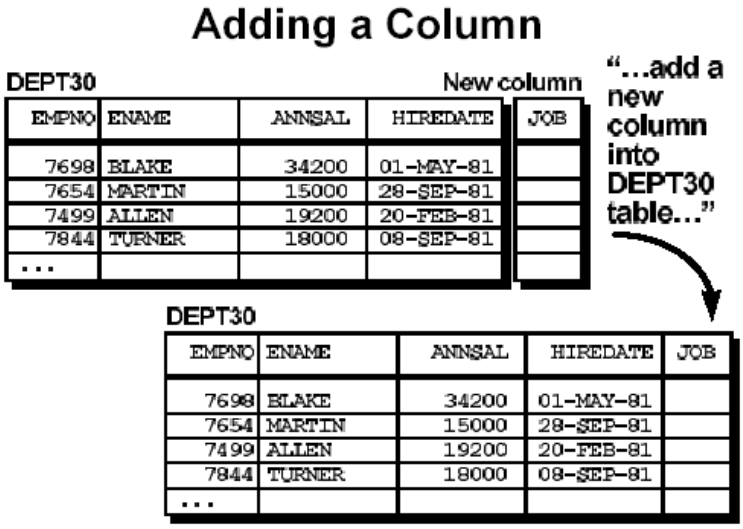
14.3.5. Quản trị columns trong table

Ta có thể thay đổi cấu trúc của table thông qua việc can thiệp vào cấu trúc của table. Sử dụng câu lệnh `ALTER TABLE` để sửa đổi cấu trúc của table.

Việc thay đổi cấu trúc của table bao gồm:

- Thêm mới columns

- Thay đổi cấu trúc columns
- Xoá bỏ columns
- Đặt chế độ UNUSED cho columns



Hình vẽ 73. Thay đổi cấu trúc của table

Thêm mới columns

Cú pháp:

```
ALTER TABLE table
      ADD   (column_name     datatype [DEFAULT     expr],
            [column_name     datatype],...);
```

Ví dụ:

```
SQL>ALTER TABLE Dept
2  >ADD    (job        varchar2(30));
Table altered.
```

Thay đổi cấu trúc một column

Cú pháp:

```
ALTER TABLE table
      MODIFY (column_name    datatype [DEFAULT     expr],
             [column_name    datatype],...);
```

Ví dụ:

```
SQL>ALTER TABLE Dept
2  >MODIFY (ename      varchar2(50));
Table altered.
```

Xoá bớt column

Bắt đầu từ phiên bản 8.1.0.0.0 trở đi, Oracle cho phép ta có thể xoá bớt các cột dữ liệu không còn cần sử dụng đến trong một table, bao gồm cả index-organized table. Việc này sẽ làm

giải phóng một phần vùng không gian trong database. Để có thể thực hiện được việc này User cần phải được cấp quyền ALTER ANY TABLE trên table có cột cần xoá.

Tuy vậy, ta không thể xoá tất cả các cột trong một table hay xoá các cột dữ liệu trong các table do user SYS sở hữu.

Để xoá columns ta thực hiện câu lệnh ALTER TABLE...DROP COLUMN.

Ví dụ: Xoá cột SAL trong bảng emp:

```
ALTER TABLE emp DROP COLUMN sal;
```

Xoá cột SAL và COMM trong bảng emp:

```
ALTER TABLE emp DROP (sal, comm);
```

Đánh dấu không sử dụng – Unused

Giống như việc xoá columns, thao tác này cũng chỉ thực hiện được bắt đầu từ phiên bản Oracle 8.1.0.0.0.

Để thực hiện việc này ta sử dụng câu lệnh ALTER TABLE...SET UNUSED. Khi một cột được đánh dấu là Unused, tên của nó sẽ không còn trong data dictionary views và ta có thể sử dụng lại tên này để đặt cho một cột dữ liệu mới bổ sung, mặt khác, tất cả các constraints, indexes, trên cột được đánh dấu Unused sẽ bị xoá bỏ.

Ví dụ: Đánh dấu Unused cho cột SAL và COMM.

```
ALTER TABLE emp SET UNUSED (sal, comm);
```

Để xem thông tin về tình hình sử dụng các cột dữ liệu đang trong trạng thái unused, ta cần truy vấn dữ liệu trong các views: USER_UNUSED_COL_TABS, ALL_UNUSED_COL_TABS và DBA_UNUSED_COL_TABS. Cột COUNT cho biết số lượng các unused columns trong table.

```
SELECT * FROM dba_unused_col_tabs;

OWNER                TABLE_NAME           COUNT
-------------------- -------------------- ----------
SCOTT                EMP                  1
1 row selected.
```

Xoá các cột dữ liệu đã được đánh dấu Unused

Để xoá hẳn cộ dữ liệu đã đánh dấu Unused ta có thể sử dụng câu lệnh ALTER TABLE...DROP UNUSED COLUMNS.

Ví dụ:

```
ALTER TABLE emp DROP UNUSED COLUMNS;
```

14.3.6. Chuyển một Table tới Segment hay Tablespace mới

Trong quá trình thực hiện chương trình, dữ liệu trong table thường xuyên được thêm mới, cập nhật,... Việc này sẽ làm cho dữ liệu trong table tăng nhanh. Khi nó vượt quá hạn mức mà

quản trị viên đã cấp phát ban đầu, ta cần phải chuyển table tới segment hay tablespace mới để hệ thống tiếp tục thực hiện được.

Ta sử dụng câu lệnh `ALTER TABLE...MOVE` thực hiện công việc này.

Ví dụ: Chuyển table `EMP` tới một segment mới với các tham số lưu trữ mới phù hợp hơn.

```
ALTER TABLE  emp MOVE
      STORAGE ( INITIAL 20K
                NEXT 40K
                MINEXTENTS 2
                MAXEXTENTS 20
                PCTINCREASE 0 );
```

14.3.7. Định nghĩa lại một table đang online

Trong hệ thống cấp cao, đôi khi ta cần ta cần phải định nghĩa lại (redefine) các table (gọi là "hot" tables) để nâng cao hiệu suất sử dụng của câu lệnh truy vấn cũng như các lệnh thao tác dữ liệu khác. Ở phiên bản Oracle 9i có cung cấp cơ chế để định nghĩa lại tables ngay cả khi nó đang hoạt động - online.

Khi định nghĩa lại một table đang trong tình trạng online, các câu lệnh DML vẫn có thể được trên table đó. Table sẽ bị khoá (locked) ở chế độ exclusive.

Với khả năng mới được cung cấp này, ta có thể:

- Thay đổi các tham số lưu trữ đối với table.
- Di chuyển table sang một tablespace khác trong cùng một schema.
- Bổ sung các hỗ trợ cho việc truy vấn song song.
- Hỗ trợ việc thêm và huỷ partitioning .
- Tạo lại table để làm giảm sự phân đoạn.
- Thay đổi cấu trúc của một table thông thường hay một index-organized table
- Thêm và huỷ column trong table

Cơ chế thực hiện việc định nghĩa lai trong chế độ online được cung cấp trong PL/SQL `DBMS_REDEFINITION`. Quyền cho phép thực hiện công việc này có tên là `EXECUTE_CATALOG_ROLE`. Để có được quyền này, user cần được cấp các quyền khác sau:

- `CREATE ANY TABLE`
- `ALTER ANY TABLE`
- `DROP ANY TABLE`
- `LOCK ANY TABLE`
- `SELECT ANY TABLE`

Các bước thực hiện việc định nghĩa lại các table:

1. Kiểm tra table có thể online redefine (định nghĩa lại khi đang thực hiện) bằng cách gọi thủ tục `DBMS_REDEFINITION.CAN_REDEF_TABLE()`. Trong trường hợp table không

thể thực hiện online redefine thủ tục đó sẽ trả về một lỗi cho biết lý do không thể thực hiện online redefine.

2. Tạo một table tạm thời (interim table). Table này bắt buộc phải thuộc cùng schema với table đang được online redefin với tất cả các thuộc tính tương ứng.

3. Bắt đầu tiến trình redefine bằng việc gọi thủ tục:
`DBMS_REDEFINITION.START_REDEF_TABLE()`.

4. Tạo các triggers, indexes, và tạo các contraints tương ứng trên interim table. Các constraints có liên quan trên interim table cần được tạo lập và đặt trạng thái disabled. Cho tới khi tiến trình redefine kết thúc (complete hoặc aborte). Sau khi quá trình redefine kết thúc, các triggers, constraints, indexes và các quyền gắn với interim table sẽ được tiếp tục redefine. Quá trình redefine các constraints liên quan tới interim table được thực hiện sau cùng và chuyển trạng thái của table này thành enable. Quá trình redefine kết thúc.

5. Thực hiện thủ tục `DBMS_REDEFINITION.FINISH_REDEF_TABLE()` để kết thúc việc redefine table. Trong quá trình thực hiện thủ tục này, table ban đầu sẽ bị locke ở chế độ exclusive.
6. Việc đổi tên các indexes được tạo trong interim table ở bước 4 sẽ được thực hiện đối với table đã được redefine.

Ví dụ:

Ví dụ sau minh hoạ công việc online redefine của một table có tên là `emp`. Đây là table thuộc loại nonpartitioned, và có các cột dữ liệu có tên: `empno, name, salary, phone`. Table sẽ được redefine theo các bước sau:

- Cột `salary` is multiplied by a factor of 1.10 và được đổi tên thành `sal`.
- Cột `phone` bị xoá bỏ.
- Một cột dữ liệu mới, cột `deptno` được thêm vào với giá trị mặc định là 10.
- Thực hiện redefine lại table được phân khu theo khoảng giá trị của cột `empno`.

Giả sử đã chạy xong thủ tục `DBMS_REDEFINITION.CAN_REDEF_TABLE()` và table `emp` đã sẵn sàng cho việc redefine.

Các bước thực hiện redefine:

Tạo một interim table với tên là int_emp.

```
CREATE TABLE int_emp
     (empno       NUMBER PRIMARY KEY,
      name        VARCHAR2(100),
      sal         NUMBER,
      deptno      NUMBER DEFAULT 10)
      PARTITION BY RANGE(empno)
```

```
              (PARTITION emp1000 VALUES LESS THAN (1000) TABLESPACE
tbs_1,PARTITION emp2000 VALUES LESS THAN (2000) TABLESPACE tbs_2);
```

Khởi động tiến trình redefine.

```
DBMS_REDEFINITION.START_REDEF_TABLE('u1', 'emp', 'int_emp',
       'empno empno, name name, salary*1.10 sal');
```

Tạo các triggers, indexes và constraints trên bảng int_emp. Tất cả các ràng buộc tham chiếu tới int_emp đều được đặt là disable. Tiếp theo, ta quy định các quyền cấp phát trên int_emp giống như là các quyền trong emp.

Thực hiện đồng bộ hai table: int_emp và emp.

```
DBMS_REDEFINITION.SYNC_INTERIM_TABLE('u1', 'emp', 'int_emp');
```

Kết thúc việc redefine.

```
DBMS_REDEFINITION.FINISH_REDEF_TABLE('u1', 'emp', 'int_emp');
```

Table `emp` bị khoá ở chế độ exclusive. Sau khi table `emp` is được redefine với các thuộc tính mới.

Xoá table trung gian.

14.3.8. Bảng ngoài – External table

External tables là các files lưu trữ dữ liệu bên ngoài database mà Oracle xem nó như là một table. Dữ liệu trong external table thường là read-only và không có indexes trên đó.Quyền trên các Object này chỉ là '`SELECT TABLE`' và '`READ DIRECTORY`'.

Tham số `UTL_FILE_DIR` được sử dụng để xác định đường dẫn tới file.

Ví dụ:

```
CREATE DIRECTORY external_tables AS
           'c:\oracle\oradata\external';

CREATE TABLE EMP_EXT (EMPNO NUMBER(4,0), ENAME VARCHAR2(10), JOB
           VARCHAR2(9), MGR NUMBER(4,0), HIREDATE DATE, SAL
           NUMBER(7,2), COMM NUMBER(7,2), DEPTNO NUMBER(2,0))
   ORGANIZATION EXTERNAL
   (TYPE oracle_loader
   DEFAULT DIRECTORY external_tables
   ACCESS PARAMETERS
     (RECORDS DELIMITED BY NEWLINE
      BADFILE external_tables:'bad_emp_ext.txt'
      LOGFILE external_tables:'log_emp_ext.txt'
      FIELDS TERMINATED BY ','
      MISSING FIELD VALUES ARE NULL)
      LOCATION ('emp.txt'))
```

```
REJECT LIMIT UNLIMITED;
```

Khi table metadata đã được tạo lập, table này có thể được sử dụng để truy vấn dữ liệu giống hệt như các table khác. Ta vẫn có thể sử dụng các hàm hay các câu lênh join trên table này, …

Ta cũng có thể tham khảo các thông tin về external tables trong các view sau:

- DBA_EXTERNAL_TABLES cho biết các thuộc tính của external table trong database.
- DBA_EXTERNAL_LOCATIONS cho biết đường dẫn tới các files và thư mục tương ứng lưu giữ chúng.

14.4.CÁC RÀNG BUỘC (CONSTRAINTS) ĐỐI VỚI TABLES

14.4.1. Ràng buộc đối với tables

Khi nạp dữ liệu vào table, oracle không chỉ quan tâm tới việc cho phép đưa các dữ liệu phù hợp với cấu trúc của table (như: cùng kiểu dữ liệu với cột tương ứng, độ lớn của dữ liệu đưa vào nằm trong khoảng cho phép,...) mà còn quan tâm tới tính phù hợp dữ liệu về mặt logic của các dữ liệu lưu trữ trong table (ví dụ như không thể có hai người khác nhau mà lại có cùng một mã số lưu trong hệ thống, giá trị độ tuổi không thể nhận giá trị âm,...).

Để đảm bảo tính logic và phù hợp với yêu cầu nghiệp vụ của từng bài toán cụ thể, Oracle server cho phép người thiết kế và quản trị database có thể tạo ra các ràng buộc dữ liệu phù hợp nhất thông qua việc sử dụng các table constraints (ràng buộc đối với tables).

Sử dụng table constraints nhằm đáp ứng được một số yêu cầu:

- Thiết lập các quy tắc nghiệp vụ đối với dữ liệu trong từng table ở nhiều mức độ khác nhau: kiểm tra tính logic của dữ liệu trước khi thực hiện các thao tác insert, update hay delete từng dòng dữ liệu trên table.
- Ngăn cản việc xoá dữ liệu trên table khi dữ liệu này có liên quan tới các dữ liệu thuộc các tables khác.

Các loại ràng buộc toàn vẹn dữ liệu

Ràng buộc	Diễn giải
NOT NULL	Không cho phép cột dữ liệu trong table nhận giá trị rỗng
UNIQUE	Không cho phép có trùng lặp dữ liệu tại columns tương ứng giữa các dòng dữ liệu khác nhau
PRIMARY KEY	Khoá chính dùng để xác định, phân biệt các dòng dữ liệu khác nhau trong table
FOREIGN KEY	Ràng buộc dữ liệu giữa hai tables khác nhau. Đảm bảo dữ liệu thuộc table này phải tương ứng với dữ liệu trong một table khác

CHECK	Kiểm tra dữ liệu nhập vào table tuân theo một quy tắc nhất định

Các constraints có thể được tạo lập trong quá trình tạo table hoặc sau khi table đã được tạo. Thông tin về các constraints được cập nhật và lưu trữ trong data dictionary.

14.4.2. Null / Not Null

Là ràng buộc đối với dữ liệu trong column là trống (null) hoặc không trống (not null).

Ví dụ mệnh đề ràng buộc:

```
CREATE TABLE DEPT (
 DEPTNO              NUMBER(2) NOT NULL,
 DNAME               CHAR(14),
 LOC                 CHAR(13),
 CONSTRAINT DEPT_PRIMARY_KEY PRIMARY KEY (DEPTNO));
```

14.4.3. Unique

Chỉ ra ràng buộc duy nhất, các giá trị của column chỉ trong mệnh đề `UNIQUE` trong các row của table phải có giá trị khác biệt. Giá trị `NULL` là cho phép nêu `UNIQUE` dựa trên một cột.

Ví dụ:

```
CREATE TABLE DEPT (
 DEPTNO              NUMBER(2),
 DNAME               CHAR(14),
 LOC                 CHAR(13),
 CONSTRAINT UNQ_DEPT_LOC UNIQUE(DNAME, LOC));
```

14.4.4. Primary Key

Chỉ ra ràng buộc duy nhất (giống `UNIQUE`), tuy nhiên khoá là dạng khoá `UNIQUE` cấp cao nhất. Một table chỉ có thể có một `PRIMARY KEY`. Các giá trị trong `PRIMARY KEY` bắt buộc phải `NOTT NULL`.

Cú pháp khi đặt `CONSTRAINT` ở mức `TABLE`:

```
[CONSTRAINT constraint_name] PRIMARY KEY (column, Column..)
```

Cú pháp khi đặt `CONSTRAINT` ở mức `COLUMN`

```
[CONSTRAINT constraint_name] PRIMARY KEY
```

14.4.5. Foreign Key (Referential Key)

Chỉ ra mối liên hệ ràng buộc tham chiếu giữa table này với table khác, hoặc trong chính 1 table. Nó chỉ ra mối liên hệ cha-con và chỉ ràng buộc giữa `FOREIGN KEY` bảng này với `PRIMARY KEY` hoặc `UNIQUE KEY` của bảng khác.

Ví dụ: quan hệ giữa `DEPT` và `EMP` thông qua trường `DEPTNO`.

```
CREATE TABLE EMP (
 EMPNO      NUMBER(4),
 ENAME      VARCHAR2(10) NOT NULL,
```

```
JOB        VARCHAR2(9),
MGR        NUMBER(4),
HIREDATE   DATE,
SAL        NUMBER(7,2),
COMM       NUMBER(7,2),
DEPTNO     NUMBER(7,2) NOT NULL,
CONSTRAINT EMP_DEPTNO_FK FOREIGN KEY (DEPTNO)
           REFERENCES DEPT (DEPTNO) ON DELETE CASCADE);
```

Từ khoá `ON DELETE CASCADE` chỉ định trong dạng khoá này nhằm mục đích khi dữ liệu cha bị xoá (trong bảng `DEPT`) thì dữ liệu con cũng tự động bị xoá theo (trong bảng `EMP`).

14.4.6. Check

Ràng buộc kiểm tra giá trị.

Ví dụ:

```
CREATE TABLE EMP
 (EMPNO NUMBER NOT NULL CONSTRAINT PK_EMP PRIMARY KEY,
  ENAME VARCHAR2(10) CONSTRAINT NN_ENAME NOT NULL
      CONSTRAINT UPPER_ENAME CHECK (ENAME = UPPER(ENAME)),
  JOB VARCHAR2(9),
  MGR NUMBER CONSTRAINT FK_MGR REFERENCES
      SCOTT.EMP(EMPNO),
  HIREDATE DATE DEFAULT SYSDATE,
  SAL NUMBER(10,2) CONSTRAINT CK_SAL   CHECK(SAL>500),
  COMM NUMBER(9,0) DEFAULT NULL,
  DEPTNO NUMBER(2) CONSTRAINT NN_DEPTNO NOT NULL
       CONSTRAINT FK_DEPTNO REFERENCES SCOTT.DEPT(DEPTNO);
```

14.5.QUẢN LÝ KHÔNG GIAN LƯU TRỮ TRONG TABLE

14.5.1. Thay đổi thông tin lưu trữ và tham số sử dụng Block

Một số thông tin của tham số lưu trữ và tham số sử dụng block có thể thay đổi bằng cách sử dụng câu lệnh `ALTER TABLE`.

Cú pháp

```
ALTER TABLE [schema.]table
      {[storage_clause ]
      [PCTFREE integer]
      [PCTUSED integer]
      [INITRANS integer]
      [MAXTRANS integer]
```

Ví dụ:

```
ALTER TABLE scott.emp
PCTFREE 30
PCTUSED 50
STORAGE(NEXT 500K MINEXTENTS 2 MAXEXTENTS 100);
```

Trong OEM, ta thực hiện theo các bước sau

1. Chạy Oracle Schema Manager.
2. Chuyển tới nút Tables, rồi tiếp tục chuyển tới schema tương ứng
3. Chọn table.
4. Thay đổi các giá trị trong phần Storage tab
5. Bấm nút Apply.

Ảnh hưởng của việc thay đổi các tham số lưu trữ:

Các tham số có thể thay đổi và ảnh hưởng của việc thay đổi đó như sau:

- `NEXT`: khi Oracle server thiết lập các extents cho bảng thì giá trị mới sẽ được áp dụng, kích thước mở rộng tuần tự sẽ được tăng lên bởi `PCTINCREASE`.
- `PCTINCREASE`: khi thay đổi bởi tham số này, thông tin thay đổi sẽ được ghi nhận trong data dictionary. Giá trị mới sẽ được sử dụng để tính lại giá trị của tham số `NEXT` khi thiết lập các extents mới. Xét ví dụ: table có 2 extents với `NEXT=10K` và `PCTINCREASE=0`. Khi tăng giá trị của `PCTINCREASE` lên thành `100`, extent thứ 3 sẽ được thiết lập `10K`, và extent thứ 4 sẽ là `20K` và cứ tiếp tục như vậy khi thêm các extents nữa.
- `MINEXTENTS`: giá trị của `MINEXTENTS` có thể thay đổi tới giá trị bất kì nào nhỏ hơn hay bằng giá trị cuả số extent hiện thời của bảng. Giá trị này sẽ không ảnh hưởng ngay khi thay đổi mà sẽ ảnh hưởng khi bảng bị truncate.
- `MAXEXTENTS`: có thể nhận bất kì giá trị nào lớn hơn hay bằng số extents hiện thời đang có trong table.

Giới hạn:

- Không thể thay đổi tham số `INITIAL` thông qua lệnh `ALTER TABLE`
- Giá trị `NEXT` chỉ định sẽ được làm tròn lên đến một giá trị là bội số nguyên lần kích thước của một Block.

Các tham số sử dụng block:

Thay đổi các tham số sử dụng block nhằm:

- Tiết kiệm không gian sử dụng.
- Giảm thiểu quá trình migration và chaining của block.

Ảnh hưởng của việc thay đổi các tham số đó:

- `PCTFREE`: thay đổi tham số này sẽ làm ảnh hưởng đến quá trình insert dữ liệu trong tương lai. Các blocks mà không được sử dụng cho việc insert do chúng được điền đầy (`100-PCTFREE`) sẽ không bị ảnh hưởng đến khi chúng được đưa vào danh sách các block trống (free list).

- PCTUSED: bất cứ một thay đổi nào của tham số này đều ảnh hưởng đến tất cả các blocks trong table. Khi cập nhật hay xoá một row, block chứa row đó sẽ được đánh dấu. Việc sử dụng hay tái sử dụng các blocks có thể thực hiện được đối với thao tác insert dữ liệu nếu như mức độ sử dụng các blocks giảm xuống dưới giá trị PCTUSED.
- INITRANS: việc thay đổi giá trị INITRANS chỉ ảnh hưởng đến các block mới.
- MAXTRANS: thay đổi giá trị này sẽ ảnh hưởng đến tất cả các blocks trong table.

14.5.2. Cấp phát các extents bằng tay (manually)

Các extents có thể cấp phát bằng cách sử dụng câu lệnh cấp phát trực tiếp (gọi là phương pháp manually - bằng tay) nhằm:

- Điều khiển quá trình phân tán các extents của table trên các file khác nhau.
- Tránh hiện tượng mở rộng tự động kích thước table trước khi nạp dữ liệu vào table .

Cú pháp:

```
ALTER TABLE [schema.]table
      ALLOCATE EXTENT [([ SIZE integer K|M]]
            [ DATAFILE 'filename ' ])]
```

Ví dụ:

```
ALTER TABLE scott.emp
      ALLOCATE EXTENT(SIZE 500K DATAFILE
      'D:\Orant\oradata\orcl\data01.dbf');
```

Nếu bỏ qua tham số SIZE, Oracle server sẽ sử dụng giá trị NEXT_EXTENT có trong DBA_TABLES để thiết lập giá trị cho các extents mới.

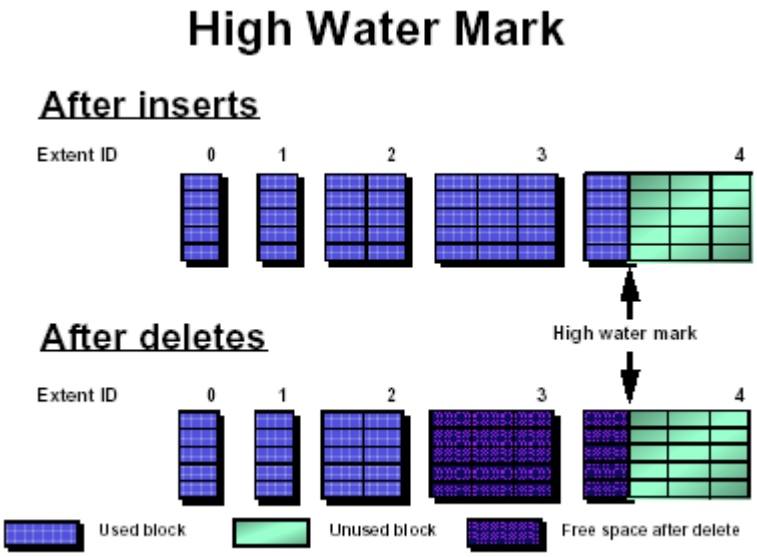
File được chỉ định trong mệnh đề DATAFILE phải thuộc về tablespace mà chứa table đang xem xét, nếu không câu lệnh sẽ sinh ra lỗi. Nếu mệnh đề DATAFILE không được sử dụng thì Oracle server sẽ thiết lập extent mới trong một datafile thuộc tablespace chứa table đang xem xét.

Lưu ý: Giá trị NEXT_EXTENT trong DBA_TABLES không làm ảnh hưởng đến quá trình thiết lập extent bằng tay. Oracle server không tính lại giá trị của extent tiếp theo khi thực hiện câu lệnh.

14.5.3. High Water Mark

518. Giá trị của High Water Mark cho một table chỉ định Block cuối cùng đã từng được sử dụng cho table.
519. Khi dữ liệu đã được insert vào trong table thì High Water Mark được chuyển đến block cuối cùng được sử dụng.
520. High Water Mark không được khởi tạo lại giá trị khi xoá các rows trong table.
521. Giá trị của High Water Mark được lưu trữ trong phần Header của table đó.

522. Khi Oracle server truy vấn dữ liệu trên toàn bộ table, nó đọc tất cả các blocks theo trình tự từ dưới lên trên cho tới khi đạt đến giá trị High Water Mark.



Hình vẽ 74. High water mark

Ví dụ: đoạn mã lệnh PL/SQL sau đây có thể dùng để tìm ra và hiển thị số block đã sử dụng trong table và số block không được sử dụng.

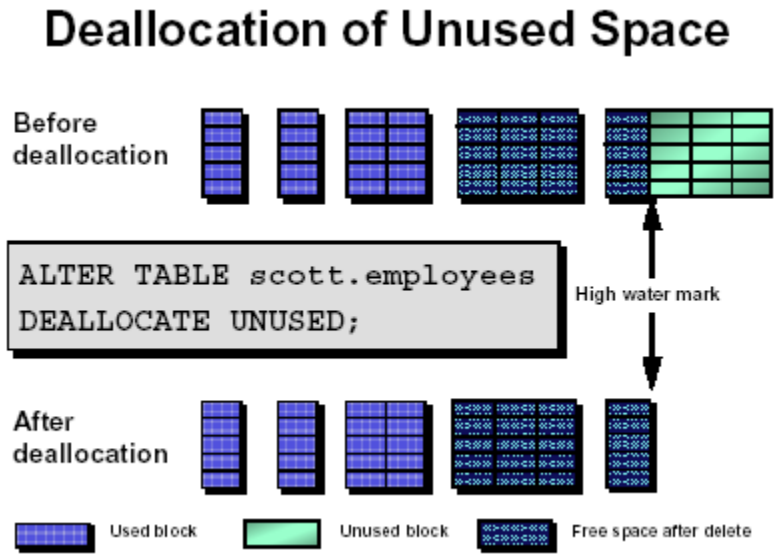
```
SVRMGR> DECLARE
2> v_owner VARCHAR2(30) := 'SCOTT' ;
3> v_segment_name VARCHAR2(30) := 'EMPLOYEES';
4> v_segment_type VARCHAR2(30) := 'TABLE';
5> v_total_blocks NUMBER;
6> v_total_bytes NUMBER;
7> v_unused_blocks NUMBER;
8> v_unused_bytes NUMBER;
9> v_last_used_extent_file_id NUMBER;
10> v_last_used_extent_block_id NUMBER;
11> v_last_used_block NUMBER;
12>
13> BEGIN
14> dbms_space.unused_space(v_owner,
15> v_segment_name,
16> v_segment_type,
17> v_total_blocks,
18> v_total_bytes,
19> v_unused_blocks,
20> v_unused_bytes,
21> v_last_used_extent_file_id,
22> v_last_used_extent_block_id,
23> v_last_used_block
24> );
25> dbms_output.put_line(INITCAP(v_segment_type)||' :
'||v_owner||'.'||v_segment_name);
26> dbms_output.put_line('Total Blocks :
'||TO_CHAR(v_total_blocks));
27> dbms_output.put_line('Blocks above HWM :
'||TO_CHAR(v_unused_blocks));
28> END;
29> /
```

```
Statement processed.
Table : SCOTT.EMPLOYEES
Total Blocks : 25
Blocks above HWM : 23
```

Chú ý: thủ tục có gọi tới package dbms_space. Package này được tạo lập khi chạy script dbmsutil.sql và thủ tục prvtutil.plb.

14.5.4. Thu hồi không gian không sử dụng

Nếu như đã cấp phát một lượng lớn các extents cho table nhưng nó chưa được sử dụng hết thì ta có thể lấy lại vùng không gian còn trống đó. Không gian này sau khi được thu hồi sẽ lại sẵn sàng cho các segments khác sử dụng.



Hình vẽ 75. Thu hồi không gian không sử dụng

Cú pháp:

```
ALTER TABLE [schema.]table
DEALLOCATE UNUSED [ KEEP integer [K|M]]
```

Giá trị KEEP chỉ số bytes trên mức High Water Mark cần để lại.

Nếu High Water Mark nằm tại một extent nhỏ hơn giá trị của MINEXTENTS, thì Oracle server sẽ giải phóng các extents nằm phía trên giá trị MINEXTENTS.

Ví dụ: khi MINEXTENTS<=4, Oracle server sẽ lấy lại tất cả các blocks không được sử dụng trên mức High Water Mark. Chú ý rằng extent thứ 5 (với ID=4) bây giờ sẽ chứa 5 blocks. Nếu giá trị MINEXTENTS là 5 đối với table thì Oracle server sẽ không thu hồi không gian từ extent thứ 5.

Lưu ý:

Do việc thu hồi không gian bởi câu lệnh trên sẽ giải phóng không gian không sử dụng nên việc sử dụng tuần tự câu lệnh này có thể dẫn đến phân mảnh không gian trong data file. Để

tránh hiện tượng đó cần khởi tạo giá trị `MINIMUM EXTENT` cho tablespace. Để giải phóng vùng không gian bên dưới High Water Mark, thậm chí khi High Water Mark là dưới giá trị `MINEXTENTS` cần sử dụng mệnh đề `KEEP 0`.

14.5.5. Truncate một table

Truncate một table sẽ xoá toàn bộ các row dữ liệu trong table và giải phóng không gian sử dụng.

Cú pháp:

```
TRUNCATE TABLE [schema.]table
[{DROP|REUSE} STORAGE]
```

Ảnh hưởng của việc sử dụng câu lệnh:

- Tất cả các rows trong table đều bị xoá .
- Không thể rollback được khi đã thực hiện câu lệnh vì câu lệnh này bắt buộc phải commit.
- Các Indexes tương ứng của table sẽ được xoá đi.
- Một table tham chiếu bởi các ngoại khoá (`FOREIGN KEY`) không thể `TRUNCATE`.
- Các triggers đi kèm với table sẽ không bị xoá khi thực hiện câu lệnh.
- Nếu sử dụng mệnh đề `DROP`: tất cả các extents ngoại trừ các extents chỉ định bởi `MINEXTENTS` được loại bỏ
- High Water Mark được khởi tạo sẽ trỏ đến block đầu tiên trong table. Giá trị của `NEXT_EXTENT` trong table được khởi tạo lại đến kích thước của extent có giá trị extent_id nhỏ nhất trong số các extents đã bị thu hồi. Tức là nếu `MINEXTENTS=2` thì giá trị `NEXT_EXTENT` sẽ được khởi tạo đến giá trị của extent thứ 3 của table.
- Sử dụng `REUSE` nhằm tái sử dụng toàn bộ không gian đã sử dụng bởi table.
- Sử dụng mệnh đề `REUSE` hay `DROP` đều dẫn đến việc xoá các Indexes.

14.5.6. Xoá table

Một table có thể bị xoá khi không cần thiết sử dụng hay khi muốn tổ chức lại nó.

Cú pháp:

```
DROP TABLE [schema.]table
[CASCADE CONSTRAINTS]
```

Khi một table bị xoá đi, các extents sử dụng bởi table này sẽ được giải phóng. Nếu các extents đó là liên tục thì chúng có thể được nhập lại tự động hoặc bằng tay.

Chú ý:

Mệnh đề `CASCADE CONSTRAINTS` là cần thiết nếu table là bảng cha trong quan hệ ngoại khoá.

14.5.7. Kiểm tra cấu trúc bảng

Oracle server thực hiện kiểm tra tính toàn vẹn của mỗi data block. Sử dụng mệnh đề CASCADE để kiểm tra cấu trúc của các indexes trên table và thực thi việc tham chiếu chéo giữa các table và index của table đó.

Mục đích chính của câu lệnh này là thống kê các thông tin về table. Từ đó, sử dụng thông tin này nhằm mục đích tối ưu hoá việc sử dụng không gian lưu trữ.

Một số cách sử dụng khác là:

- Xoá thông tin thống kê về các bảng trong data dictionary.
- Kiểm tra cấu trúc các bản.
- Xác định mức độ Migration và Chaining của các rows trong table.

Kiểm tra cấu trúc table

Sau khi kiểm tra cấu trúc table lưu trữ dữ liệu, tất cả các blocks trong bảng đều được kiểm tra tính toàn vẹn.

Oracle server kiểm tra xem block có bị hỏng hay không ngay tại mỗi lần đọc block đó. Tham số DB_BLOCK_CHECKSUM=TRUE sẽ yêu cầu thực hiện tính toán checksum đối với phần header của block dữ liệu trước khi ghi block dữ liệu lên đĩa.

Cú pháp: sử dụng câu lệnh sau khi kiểm tra tính toàn vẹn của các block trong table:

```
ANALYZE TABLE [schema.]table
VALIDATE STRUCTURE [CASCADE]
```

Chú ý: Sủ dụng các thủ tục sau đây để phân tích các Objects:

```
DBMS_DDL.ANALYZE_OBJECT để phân tích một đối tượng chỉ định.
```

- DBMS_UNTILITY.ANALYZE_SCHEMA để phân tích tât cả các đối tượng thuộc về user.
- DBMS_UTILITY.ANALYZE_DATABASE dùng phân tích tất cả các đối tuộng trong database.

14.5.8. Phát hiện các rows bị migration

Câu lệnh ANALYZE cũng còn có thể được sử dụng để kiểm tra các rows bị migration hoặc chaining trong table hay không.

Cú pháp:

```
ANALYZE TABLE [schema.]table
[ COMPUTE STATISTICS]
[ ESTIMATE STATISTICS] SAMPLE integer ROWS | PERCENT]
```

Tuỳ chọn COMPUTE STATISTICS sẽ sinh ra thông tin thống kê dựa vào thông tin của toàn bộ table, còn tuỳ chọn ESTIMATE STATISTICS sẽ sinh ra thông tin thống kê dựa vào một số hàng làm mẫu.

Khi thông tin về thống kê đã kết sinh thì nó sẽ được cập nhật vào bảng DBA_TABLES, trường CHAIN_CNT sẽ được cập nhật với thông tin về số rows bị chaining và migration trong bảng.

Nếu có một số lượng lớn các rows trong table bị chaining hay migration thì table đó cần được tổ chức và đánh giá lại thông qua câu lệnh ANALYZE như ở trên.

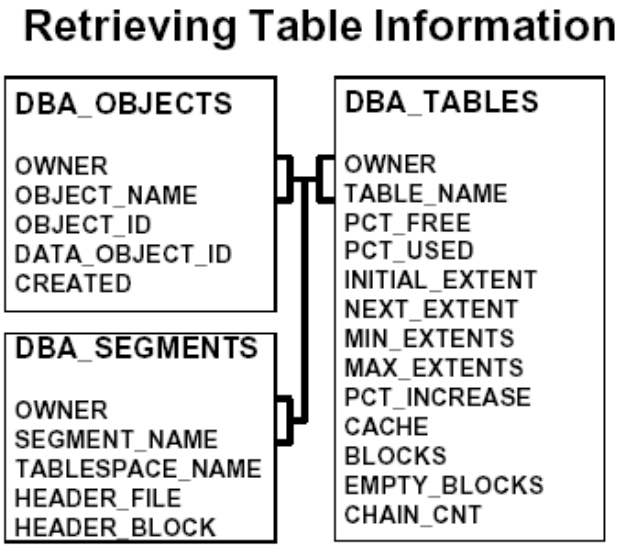
Ví dụ:

```
ANALYZE TABLE VOUCHER
      ESTIMATE STATISTICS;

Tương đương với lệnh:
ANALYZE TABLE VOUCHER
      ESTIMATE STATISTICS;
      SAMPLE 1024 ROWS
```

Khi sử dụng mệnh đề ESTIMATE STATISTICS, theo mặc định, nó sử dụng 1024 rows

14.6.THÔNG TIN VỀ TABLES

Thông tin về các tables có thể lấy từ dictionary views.



Hình vẽ 76. Thông tin về các tables trong database

14.6.1. Thông tin chung về các tables

Để lấy thông tin về bảng: số đối tượng dữ liệu, vị trí của phần header của bảng cho tất cả các bảng thuộc về user SCOTT ta dùng câu lênh sau:

```
SVRMGR> SELECT t.table_name, o.data_object_id,
2> s.header_file, s.header_block
3> FROM dba_tables t, dba_objects o, dba_segments s
4> WHERE t.owner=o.owner
5> AND t.table_name=o.object_name
6> AND t.owner=s.owner
7> AND t.table_name=s.segment_name
8> AND t.owner='SCOTT';
TABLE_NAME   DATA_OBJEC HEADER_FIL HEADER_BLO
---------- ---------- ---------- ----------
BONUS        1812       4          12
```

```
DEPARTMENTS 1811        4          7
DUMMY       1814        4          22
EMPLOYEES   1810        4          2
SALGRADE    1813        4          17
5 rows selected.
```

14.6.2. Thông tin về sử dụng block và thông tin chaining

Sử dụng câu lệnh query sau đây để lấy thông tin về số hàng bị chaining hay migration trong bảng, số block được sử dụng tới giá trị High Water Mark và giá trị của số block trên mức High Water Mark

```
SVRMGR> SELECT blocks AS HWM, empty_blocks, chain_cnt AS
"Chained Blocks"
2> FROM dba_tables
3> WHERE owner='SCOTT'
4> AND table_name='EMPLOYEES';

HWM   EMPTY_BLOC Chained Bl
---- ---------- ----------
1     23         0
1 row selected.
```

Lưu ý: dữ liệu trong trong dba_tables được cập nhật thực hiện lệnh ANALYZE (xem phần kiểm tra cấu trúc bảng - ở trên).

Chương 15. QUẢN LÝ CÁC INDEXES

15.1.PHÂN LOẠI INDEXES

Index là cấu trúc hình cây cho phép truy xuất trực tiếp một row trong table. Indexes có thể chia ra làm hai loại chính là logic và vật lý. Index theo kiểu logic dẫn xuất từ ứng dụng, còn index theo kiểu vật lý thì được phân chia theo cách thức mà index được lưu trữ.

15.1.1. Index trên một column và Index trên nhiều columns

Một index trên một column thì chỉ có một column đó tham gia vào `INDEX KEY`. Ví dụ index trên trường `EMPNO` của bảng `EMP` chỉ có một cột tham gia vào khoá của index.

Index trên nhiều columns hay còn gọi là index phức hợp, những indexes này được tạo thành từ nhiều columns trong table, các columns tạo thành index không cần phải ở cạnh nhau. Ví dụ index tạo nên từ hai cột `DEPNO` và `JOB` trong bảng `EMP`.

Số columns cực đại cho một index trên nhiều columns là 32. Tuy nhiên kích thước kết hợp của tất cả các columns không vượt quá 1/3 kích thước của một Block.

15.1.2. Unique index và Non-unique index

Một unique index (index duy nhất) đảm bảo rằng không có hai rows nào thuộc table có cùng một giá trị trên các columns có trong index. Khoá của unique index chỉ có thể trỏ đến một row duy nhất trong table.

Khác với unique index, với non-unique index (index không duy nhất), một giá trị khoá của index sẽ tương ứng với một nhóm các rows.

15.1.3. Partitioned index và non-partitioned index

Các table thông thường trong database đều thuộc loại non-partitioned table.

Partitioned index (index phân khu) dùng cho các table lớn, lưu các mục index (index entries) của index này có thể nằm trên nhiều segments.

Việc phân khu sẽ cho phép một index có thể trải rộng trên nhiều tablespaces, giảm bớt tình trạng quá tải khi index được truy xuất và quản lý.

Các partitioned index hay được sử dụng cùng với các partitioned table (bảng được phân khu) để tăng cường hiệu năng và dễ quản lý. Partitioned index sẽ được tạo ra ứng với mỗi partitioned table.

Tài liệu này đề cập tới hai loại index hay được sử dụng là **B-TREE index** và **BIPMAP index**.

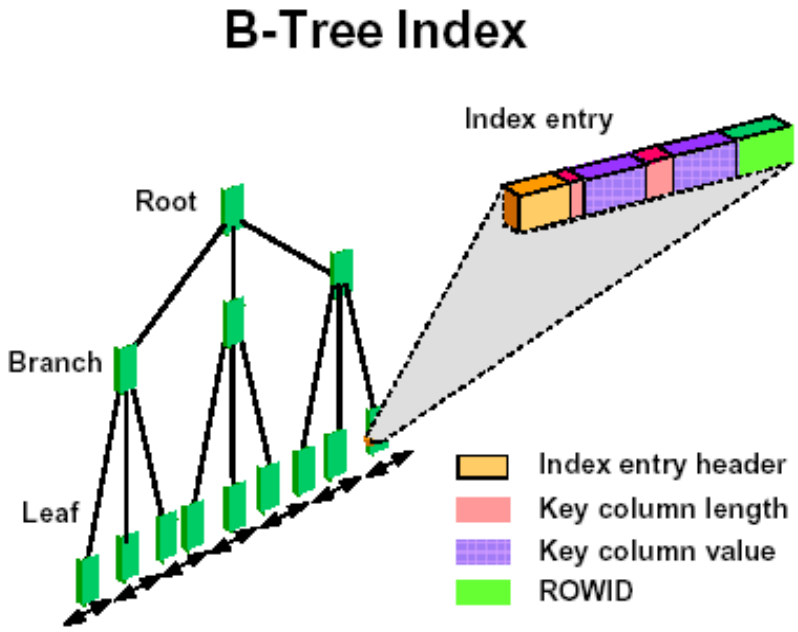
15.2.TỔ CHỨC INDEX

15.2.1. B-TREE index

Mặc dù hầu hết các indexes đều sử dụng B-TREE index, song thuật ngữ B-TREE thường sử dụng kết hợp với một index có lưu trữ danh sách `ROWID` tại mỗi khoá của index đó.

Cấu trúc của B_TREE

Đỉnh của index hay còn gọi là gốc (root). Gốc chứa các điểm vào (entry) trỏ đến mức tiếp theo của index. Ở mức tiếp theo là các block nhánh (branch). Block nhánh này lại trỏ đến các block tiếp theo của index. Ở mức thấp nhất là lá (leaf). Lá sẽ chứa thông tin điểm vào trỏ đến các rows trong table.



Hình vẽ 77. B_TREE index

Các khối lá là kết nối kép thuận tiện cho việc truy xuất index trong trật tự giảm hay tăng của giá trị khoá.

Định dạng của lá index

Một điểm vào của index sẽ được tạo thành bởi các thành phần sau đây:

- Entry header: thông tin lưu trữ số column và thông tin khoá của các hàng trong bảng.
- Key column length_value pair: chứa thông tin về kích thước column tham gia vào khoá và tiếp theo là kích thước của đó.
- ROWID: là giá trị của ROWID chứa giá trị khoá của index.

Đặc tính của index leaf entry

Một B-TREE index trong một non-partitioned table:

- Giá trị khoá bị lặp lại nếu như có nhiều hàng có cùng giá trị khoá.

- Không có index entry tương ứng với các rows mà giá trị của tất cả các cột khoá đều bằng NULL.
- ROWID được giới hạn sử dụng để trỏ đến các rows của table, bởi vì tất cả các rows đều thuộc về cùng một segment.

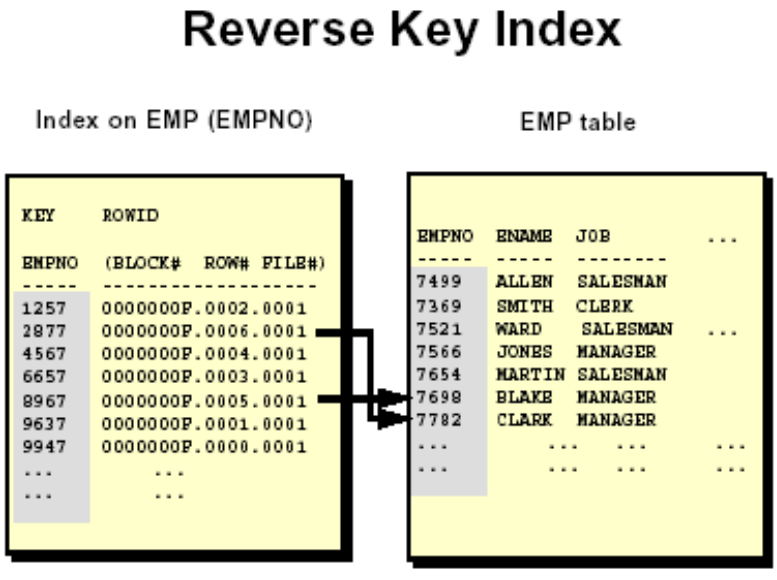
Ảnh hưởng của việc thực thi câu lệnh DML đối với Index.

Oracle server xem xét index của table mỗi khi có một câu lệnh DML thực thi trên table đó. Dưới đây là một số ảnh hưởng của câu lệnh DML đối với index:

- Kết quả của câu lệnh insert (thêm mới) dòng dữ liệu sẽ chèn thêm một index entry vào index tương ứng.
- Việc delete (xoá) các rows trong table sẽ dẫn tới việc xoá các index entries tương ứng trong block. Không gian sử dụng bởi các rows bị xoá sẽ không dùng được cho các entries (điểm vào mới) cho tới khi toàn bộ các entries (điểm vào) của block bi xoá.
- Việc cập nhật các cột khoá là kết quả của quá trình delete hay insert. Giá trị PCTFREE không ảnh hưởng đến index ngoại trừ vào thời điểm tạo index. Một điểm vào mới có thể được thêm vào block của index ngay cả khi không đủ không gian chỉ định bởi PCTFREE.

15.2.2. Reverse Key Index

Trái ngược với B-TREE index, reverse key index (khoá index ngược) sử dụng cấu trúc khoá index theo thứ tự các bytes ngược với thứ tự trong B-TREE (ngoại trừ ROWID). Tuy nhiên, trật tự của các columns trong khoá vẫn được giữ nguyên.



Hình vẽ 78. Reverse Key Index

Khi chèn một bản ghi trong trật tự tăng dần của khoá, ví dụ như việc hệ thống sinh ra số empno cho bảng EMP, thì có thể xảy ra hiện tượng thắt cổ chai trên các index vì tất cả các

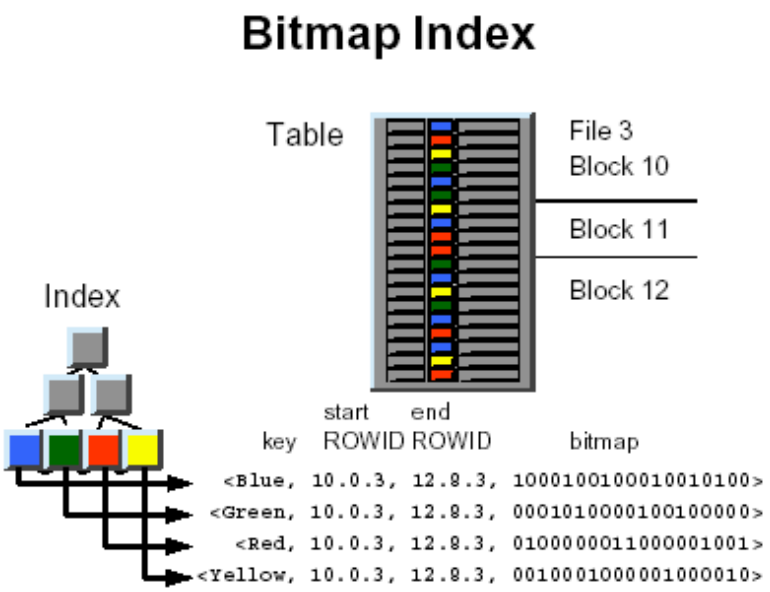
index được cập nhật xảy ra ở cùng một vị trí trong cây index, phương pháp dùng khoá ngược sẽ trải và phân tán các index cập nhật qua nhiều cây index.

Ví dụ: khi insert mã nhân viên 7698 vào trong bảng, một khoá 8967 (khoá ngược với 7698) sẽ được lưu trong index, và nhân viên tiếp theo 7782 được nhập và thì một khoá 2877 (khoá ngược với 7782) sẽ được lưu vào của index. Vì vậy các mục (entries) của index được lưu trên trên nhiều bock khác nhau của index.

Như vậy việc sắp xếp đó có ý nghĩa đặc biệt trong việc tránh giảm hiệu năng trong index môi trường Oracle Paralell.

Index dùng khoá ngược hay được sử dụng cho các query mà có các giá trị giống nhau, bởi vì các khoá về mặt từ vựng cạnh nhau sẽ không được lưu trữ gần nhau khi sử dụng khoá ngược.

15.2.3. Bitmap Index



Hình vẽ 79. Bitmap index

Bitmap Index (Index theo kiểu ánh xạ bits) là một kiểu index hay được sử dụng trong một số trường hợp sau:

- Khi table có nhiều rows và các cột khoá có giá trị khác nhau rất ít. Điều đó có nghĩa là có rất ít sự khác nhau trong giá trị của các cột. Ví dụ Bitmap Index thích hợp hơn đối với các cột giới tính (Nam hay Nữ).
- Khi truy vấn có kết hợp sử dụng nhiều mệnh đề trong phần điều kiện WHERE. Mệnh đề truy vấn sử dụng các phép toán logic OR.
- Khi các cột khoá là read-only (chỉ đọc) hay có rât ít hoạt động cập nhật các cột khoá.

Cấu trúc của Bitmap Index

Một Bitmap index cũng được tổ chức như là B-TREE index, nhưng phần lá của mỗi node lưu một dãy các bit cho mỗi khoá thay vì danh sách các ROWID. Mỗi bit trong danh sách Bitmap

đó tương ứng với một ROWID, và nếu giá trị bit đó được khởi tạo, điều đó có nghĩa là hàng có ROWID tương ứng sẽ chứa giá trị khoá.

Sử dụng Bitmap index

Bitmap-TREE index sử dụng để thiết lập phần lá của các node, phần này sẽ chứa đoạn bitmap được sử dụng để xác định hàng chứa giá trị khoá.

Khi có thay đổi trên các cột khoá trong table, các chuỗi bitmap cần được thay đổi theo. Kết quả là sẽ sinh ra các khoá trên các bitmap segment liên quan do quá trình phân đoạn các khoá này đòi hỏi thực hiện trên toàn bộ bitmap segment. Một row quản lý bởi bitmap sẽ không thể cập nhật bởi các transaction khác đến khi transaction đầu kết thúc.

15.2.4. So sánh giữa B-TREE index và Bitmap index

Bảng so sánh giữa B-TREE và Bitmap index

B-tree	Bitmap
Thích hợp với các cột dữ liệu trên tập giá trị lớn	Thích hợp với các cột dữ liệu trên tập giá trị nhỏ
Việc cập nhật dựa trên khoá quan hệ nên không đắt	Cập nhật dựa vào trường khoá nên khá đắt
Không hiệu quả cho các truy vấn có sử dụng mệnh đề OR	Hiệu quả cho các truy vấn có sử dụng mệnh đề OR
Hữu ích đối với OLTP (Online transsaction processing - dịch vụ xử lý trực tuyến)	Hữu ích đối với DSS (Decision support system - hệ thống hỗ trợ quyết định)

Bảng trên đây so sánh giữa B-TREE và Bitmap Index, Bitmap index được sử dụng nhiều hơn trong trường hợp các cột có giá trị khác nhau rất ít.

Việc cập nhật các cột làm khoá trong Bitmap index thì sẽ chậm hơn bởi vì Bitmap index sử dụng phương pháp khoá đoạn bitmap (bitmap segment level locking), trong khi đó trong một B-TREE index khoá thực hiện trên các điểm vào tương ứng với từng row riêng lẻ trên table.

Bitmap index có thể thực hiện các hoạt động với các toán hạng logic OR. Khi đó Oracle Server sử dụng hai phân đoạn bitmap để thực hiện việc so sánh từng bit trong toán hạng OR

và trả về kết quả là một chuỗi Bitmap. Tính chất này cho phép sử dụng hiệu quả chuỗi Bitmap trong câu lệnh truy vấn có sử dụng toán hạng logic OR.

Nói chung B-TREE index thích hợp hơn trong môi trường OLTP cho việc truy vấn các bảng động. Trong khi đó, Bitmap index thích hợp hơn trong môi trường DSS có sử dụng nhiều câu lệnh truy vấn phức tạp trên các table lớn (large) và tĩnh (static).

15.3.QUẢN LÝ INDEX

15.3.1. Tạo các index

Một index có thể tạo hoặc trên acount của user là owner của bảng hay tạo trên một acount khác.

Cú pháp:

```
CREATE [ UNIQUE ]INDEX [schema.] index
      ON [schema.] table
      (column [ ASC | DESC ] [ , column [ASC | DESC ] ] ...)
      [ TABLESPACE tablespace ]
      [ PCTFREE integer ]
      [ INITRANS integer ]
      [ MAXTRANS integer ]
      [ storage-clause ]
      [ LOGGING| NOLOGGING ]
      [ NOSORT ]
```

Với:

UNIQUE	được sử dụng chỉ định một unique index (non-unique index là mặc định).
schema	là owner của bảng chứa index.
index	là tên của index.
table	là tên của bảng chứa index
column	là tên cột dùng làm index
ASC/DESC	được cung cấp để tương thích về cú pháp cho database khác.
TABLESPACE	tên tablespace mà index sẽ được tạo trên đó
PCTFREE	không gian dành riêng trong mỗi block, được sử dụng khi một có điểm vào mới của index (new entries) được tạo ra.
INITRANS	chỉ định số giao dịch thiết lập ban đầu cho mỗi block
MAXTRANS	giới hạn số giao dịch có thể thiết lập cho mỗi block (giá trị mặc định là 255).
STORAGE	tham số lưu trữ, quy định có bao nhiêu extents sẽ cấp phát cho index.
LOGGING	chỉ định việc tạo các index và các hoạt đông tuần tự trên trên index sẽ được ghi vào trong các redo log file
NOLOGGING	chỉ định việc tạo và các hoạt động tuần tự trên index không được ghi vào các log file.
NOSORT	chỉ định các row được lưu trong database theo trật tự tăng dấn và vì thế oracle server không cấn sắp xếp các hàng trong khi tạo index.

Ví dụ:

```
CREATE INDEX scott.emp_lname_idx
      ON scott.employees(last_name)
      PCTFREE 30
      STORAGE(INITIAL 200K NEXT 200K
      PCTINCREASE 0 MAXEXTENTS 50)
      TABLESPACE indx01;
```

Chú ý:

- Nếu giá trị của MINIMUM EXTENT đã được định nghĩa cho tablespace, kích thước của extent dùng cho index sẽ được làm tròn lên thành một bội số lần giá trị của MINIMUM EXTENT.
- Nếu mệnh đề [NO]LOGGING bị bỏ qua, thuộc tính log của index mặc định sẽ phụ thuộc vào thuộc tính log của tablespace mà index được tạo trên đó.
- PCTUSED không được chỉ định cho index, vì các điểm vào index phải được lưu trữ theo một trật tự nhất định, người dùng không được phép điều khiển khi một index block được sử dụng cho việc insert.
- Nếu từ khoá NOSORT được sử dụng khi dữ liệu không được săp xếp trong trường khoá thì câu lệnh sẽ kết thúc với một lỗi, lỗi này giống như việc thực hiện nhiều câu lệnh DML trên bảng.
- Oracle server sẽ sử dụng index đang tồn tại để tạo bảng nếu có thể. Việc này xảy ra khi khoá cho một index mới tương ứng với phần đầu của khoá của một index đã tồn tại.

Các hướng dẫn khi tạo index:

Hãy xem xét các hướng dẫn khi tạo index:

- Index làm tăng tốc độ của các câu lệnh truy vấn nhưng làm chậm tốc độ của các câu lệnh DML. Vì thế, cần giảm tối thiểu index trên các bảng hay xảy ra thay đổi.
- Đặt index trong các tablespace riêng biệt, không đặt index trong tablespace chứa rollback segment, temporary segment và table.
- Để làm giảm sự phân mảnh trong các tablespace dùng chứa index sử dụng chuẩn kích thước extent là bội số của 5*DB_BLOCK_SIZE.
- Hiệu năng có thể tăng lên nếu không sử dụng mệnh đề LOGGING. Vì vậy, cần xem xét khi sử dụng mệnh đề LOGGING khi tạo các index lớn.
- Vì các điểm vào cho index là nhỏ hơn đối với các rows được index nên index block sẽ có nhiều điểm vào cho một block. Vì vậy, giá trị của INITRANS đối với index nói chung nên đặt lớn hơn là giá trị của tham số này trên table sử dụng index.

Để phát huy hiệu quả của việc sử dụng index, ta tạo index cho từng cột hoặc nhóm cột tham gia trong mệnh đề WHERE của câu lệnh truy vấn.

Ví dụ:

```
1. Tạo index tăng tốc độ truy vấn tên nhân viên:
SELECT * FROM emp WHERE UPPER(emp_name) LIKE 'JOH%';

Lệnh tạo index
CREATE INDEX idx ON emp (UPPER(emp_name));
```

```
2. Với câu lệnh truy vấn trên biểu thức:
SELECT a FROM t WHERE a + b * (c - 1) < 100;

Lệnh tạo index
CREATE INDEX idx ON t (a + b * (c - 1), a, b);

3. Tạo index hỗ trợ sắp xếp chuỗi ký tự dựa trên ngôn ngữ của
từng quốc gia:
SELECT * FROM t_table ORDER BY name;

Lệnh tạo index
CREATE INDEX nls_index
          ON t_table NLSSORT(name, 'NLS_SORT = German'));

4. Sử dụng index trên nhiều cột khác nhau:
SELECT * FROM emp
     WHERE UPPER(emp_name) LIKE 'JOH%'
     ORDER BY name;

Lệnh tạo index
CREATE INDEX empi
          ON emp UPPER ((ename), NLSSORT(ename));
```

Index và giá trị PCTFREE

Tham số PCTFREE cho index làm việc khác với cho table. Tham số này được sử dụng chỉ trong quá trình tạo các index. Tham số này dành riêng không gian cho các điểm vào của idex. Các điểm vào của index sẽ không được cập nhật. Khi cột khoá được cập nhật, các điểm vào cũ sẽ được xoá và chèn vào đó một điểm vào mới. Sử dụng giá trị của tham số PCTFREE thấp cho index trên các cột mà giá trị của nó tăng đều đặn. Ví dụ như hệ thống sinh số hoá đơn. Khi này, điểm vào của index mới sẽ luôn luôn dựa vào điểm vào đã tồn tại trước đó và vì vậy không cần chèn thêm vào các điểm vào mới giữa hai điểm vào đã tồn tại.

Ở đây giá trị cho một cột được index của hàng được chèn vào bảng có thể nhận bất cứ giá trị nào. Vì vậy giá trị mới có thể không nằm trong dãy giá trị đã có. Cho nên cần phải chỉ định một giá trị PTCFREE cao. Ví dụ tạo index trên trường mã khách hàng trên bảng hoá đơn. Trong trường hợp này cần sử dụng công thức tính sau để xác định giá trị của PCTFREE:

$$\frac{\text{Maximum number of rows} - \text{Initial number of rows}}{\text{Maximum number of rows}} \times 100$$

Giá trị cực đại có thể lấy trong khoảng thời gian cụ thể chẳng hạn như một năm.

15.3.2. Một số cách sử dụng index

Sử dụng index một cách tường minh

Với cách này, người dùng chỉ việc tạo index một cách tường minh thông qua câu lệnh SQL.

Mệnh đề ON sẽ cho biết cột dữ liệu trong bảng được sử dụng index.

Ví dụ:

```
CREATE INDEX emp_ename ON emp(ename)
      TABLESPACE users
      STORAGE (INITIAL 20K
      NEXT 20k
      PCTINCREASE 75)
      PCTFREE 0;
```

Index được gắn liền ngay trong ràng buộc (constraint)

Khi tạo các ràng buộc UNIQUE hoặc PRIMARY KEY cho các cột dữ liệu trong table, Oracle sẽ tự động tạo ra các index tương ứng với cột dữ liệu này.

Tuy nhiên trong một số trường hợp, user sở hữu table muốn tạo các indexes cho table nằm trên một tablespace riêng để tiện cho việc quản trị. Khi này, việc tạo index cần được gắn liền một cách tường minh ngay trong câu lệnh tạo bảng.

Ví dụ:

```
CREATE TABLE emp (
     empno NUMBER(5) PRIMARY KEY, age INTEGER)
     ENABLE PRIMARY KEY USING INDEX
     TABLESPACE users
     PCTFREE 0;
```

Index Online

Thông thường trong khi tạo index, các câu lệnh DML tác động lên cột dữ liệu có liên quan đều tạm thời không thực hiện được cho đến khi hoàn tất việc tạo index.

Để có thể cho phép thực hiện câu lệnh DML tác động lên các cột dữ liệu được index, Oracle có hỗ trợ Index online (Index trực tuyến) khi này, ta cần bổ sung thêm mệnh đề ONLINE vào trong câu lệnh.

Ví dụ:

```
ALTER INDEX emp_name REBUILD ONLINE;
CREATE INDEX emp_name ON emp (mgr, emp1, emp2, emp3) ONLINE;
```

Index theo giá trị hàm

Không chỉ cho phép thực hiện các index trực tiếp trên các cột dữ liệu trong table, Oracle còn cho phép thực hiện các index dựa vào giá trị của các hàm áp dụng trên các cột dữ liệu của table.

Ví dụ:

```
CREATE INDEX idx ON t (a + b * (c - 1), a, b);

Sử dụng cho câu lệnh truy vấn:
SELECT a FROM t WHERE a + b * (c - 1) < 100;
```

Lưu ý: Loại Index này chỉ được sử dụng trong các phiên bản Oracle 8i trở lên. Để sử dụng được index này, ta cần phải thiết lập một số thông số khởi tạo trong parameter file:

- QUERY_REWRITE_INTEGRITY đặt là TRUSTED
- QUERY_REWRITE_ENABLED đặt là TRUE
- COMPATIBLE phải được đặt là 8.1.0.0.0 hoặc lớn hơn

Để sử dụng được index này, table cần được phải được thực hiện cấu trúc (ANALYZE) sau khi đã tạo xong index.

15.3.3. Tạo Index khoá ngược (reverse key index)

Ta có thể tạo các reverse key index bằng câu lện CREATE INDEX:

Cú pháp:

```
CREATE [ UNIQUE ]INDEX [schema.] index
      ON [schema.] table
      (column [ ASC | DESC ] [ , column [ASC | DESC ] ] ...)
      [ TABLESPACE tablespace ]
      [ PCTFREE integer ]
      [ INITRANS integer ]
      [ MAXTRANS integer ]
      [ storage-clause ]
      [ LOGGING| NOLOGGING ]
      REVERSE
```

Ví dụ:

```
CREATE UNIQUE INDEX scott.ord_ord_no_idx
      ON scott.ord(ord_no) REVERSE
      PCTFREE 30
      STORAGE(INITIAL 200K NEXT 200K
      PCTINCREASE 0 MAXEXTENTS 50)
      TABLESPACE indx01;
```

Câu lệnh này tạo index khoá ngược tương tự như cho index thông thường ngoại trừ việc thêm mệnh đề REVERSE.

Chú ý: từ khoá NOSORT không được sử dụng trong câu lệnh tạo index khoá ngược.

15.3.4. Tạo Bitmap index

Cú pháp:

```
CREATE BITMAP INDEX [schema.] index
      ON [schema.] table
      (column [ ASC | DESC ] [ , column [ASC | DESC ] ] ...)
      [ TABLESPACE tablespace ]
      [ PCTFREE integer ]
      [ INITRANS integer ]
      [ MAXTRANS integer ]
      [ storage-clause ]
      [ LOGGING| NOLOGGING ]
      [ NOSORT ]
```

Ví dụ :

```
CREATE BITMAP INDEX scott.ord_region_id_idx
      ON scott.ord(region_id)
      PCTFREE 30
      STORAGE(INITIAL 200K NEXT 200K
      PCTINCREASE 0 MAXEXTENTS 50)
      TABLESPACE indx01;
```

Chú ý: Bitmap index không thể là Unique.

Tham số khởi tạo CREATE_BITMAP_AREA_SIZE quyết định không gian sẽ được sử dụng cho việc lưu trữ các bitmap segment trong bộ nhớ, giá trị mặc định của nó là 8MB, một giá trị lớn hơn có thể làm cho việc tạo index nhanh hơn.

15.3.5. Thay đổi tham số lưu trữ cho index

Một số tham số lưu trữ và tham số sử dụng block có thể thay đổi bằng cách sử dụng câu lệnh ALTER INDEX.

Cú pháp:

```
ALTER INDEX [schema.]index
      [ storage-clause ]
      [ INITRANS integer ]
      [ MAXTRANS integer ]
```

Ví dụ:

```
ALTER INDEX scott.emp_lname_idx
      STORAGE(NEXT 400K
      MAXEXTENTS 100);
```

Ảnh hưởng của việc thay đổi tham số lưu trữ cho một index giống như việc thay đổi tham số cho bảng, cách sử dụng chung nhất của việc thay đổi tham số là tăng giá trị của MAXEXTENTS cho index.

15.3.6. Cấp phát và thu hồi không gian sử dụng của index

Thiết lập không gian sử dụng bằng tay

Công việc này cần thiết khi thêm các extent cho một index trước khi một quá trình chèn một lượng lớn các hàng vào trong bảng . Thêm các extent bằng tay ngăn việc tự động thêm các extent của index.

Lấy lại không gian cấp phát cho index bằng tay

Sử dụng mệnh đề DEALLOCATE của câu lệnh ALTER INDEX để giải phóng không gian không được sử dụng ở trên mức High Water Mark trong một index.

Cú pháp:

```
ALTER INDEX [schema.]index
      {ALLOCATE EXTENT ([SIZE integer [K|M]]
      [ DATAFILE 'filename' ])
```

```
        | DEALLOCATE UNUSED [KEEP integer [ K|M ] ] }
```

Ví dụ:

```
ALTER INDEX scott.ord_region_id_idx
ALLOCATE EXTENT (SIZE 200K
DATAFILE '/DISK6/indx01.dbf');

ALTER INDEX scott.ord_ord_no_idx
DEALLOCATE UNUSED;
```

Chú ý:

Không gian index được thu hồi khi bảng trên đó index được xây dựng bị truncate(xoá). Khi thực hiện việc xoá bảng bằng lệnh Truncate thì các index trên bảng đó cũng bị xoá theo.

15.3.7. Xây dựng lại (Rebuild) các index

Các index được xây dựng lại nhằm mục đích:

- Một index mới được xây dựng trên cơ sở một index đã tồn tại .
- Quá trình săp xếp không cần thiết khi một index được xây dựng trên một index đã tồn tại, kết quả là quá trình sẽ tạo ra hiệu năng cao hơn.
- Index cũ bị xoá đi sau khi một index mới được tạo. Trong quá trình xây dựng lại Index không gian cần thiết là không gian cho cả index cũ và index mới khi được tạo thành.
- Các truy vấn có thể tiếp tục sử dụng các index đang tồn tại trong khi các index mới đang được xây dựng.

Các tình huống có thể phải xây dựng lại index

- Các index đang tồn tại cần được chuyển tới một tablespace mới,công việc này cần thiết khi các index ở trong cùng một tablespace vì các bảng hay các object khác cần phân tán trên nhiều đĩa.
- Một index chứa nhiều điểm vào bi xoá, hiện tượng này xảy ra với các index trượt, ví dụ như index trên trường số thứ tự đặt hàng của bảng đặt hàng. Khi các đơn đặt hàng đã hoàn thành sẽ bị xoá đi và một đơn đặt hàng mới được thêm vào trong bảng với số đặt hàng lớn hơn.
- Một index thông thường cần chuyển đổi sang index với khoá ngược.

Cú pháp:

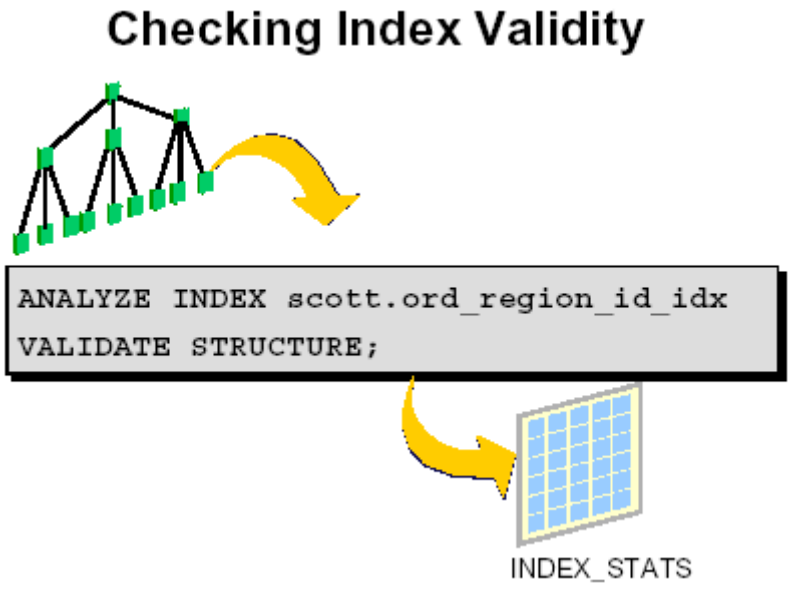
```
ALTER INDEX [schema.] index REBUILD
      [ TABLESPACE tablespace ]
      [ PCTFREE integer ]
      [ INITRANS integer ]
      [ MAXTRANS integer ]
      [ storage-clause ]
      [ LOGGING| NOLOGGING ]
      [ REVERSE | NOREVERSE ]
```

Câu lệnh Rebuild index không thể dùng để chuyển đổi một Bitmap index thành một B-Tree và ngược lại. Các mệnh đề REVERSE và NOREVERSE chỉ sử dụng với B-tree index.

Ví dụ:

```
ALTER INDEX scott.ord_region_id_idx
      REBUILD
      TABLESPACE indx02;
```

15.3.8. Kiểm tra tính hợp lệ của index



Hình vẽ 80. Kiểm tra tính hợp lệ của Index

Câu lệnh phân tích index thực hiện các công việc sau:

- Kiểm tra tất cả index blocks và tìm xem có block hỏng không. Câu lệnh này không kiểm tra xem index có tương ứng với dữ liệu trong bảng hay không.
- Thiết lập view INDEX_STATS với thông tin về index

Ở phiên bản Oracle 9i, ta có thể thực hiện lệnh ANALYZE VALIDATE STRUCTURE để tối ưu ngay cả khi đang có lệnh DML thực hiện trên table.

Cú pháp :

```
ANALYZE INDEX [schema.]index VALIDATE STRUCTURE
```

Sau khi thực hiện câu lệnh ANALYZE INDEX truy vấn view INDEX_STATS để lấy thông tin về index như trong ví dụ dưới đây:

```
SVRMGR> SELECT blocks, pct_used, distinct_keys
2> lf_rows, del_lf_rows
3> FROM index_stats;
BLOCKS      PCT_USED    LF_ROWS     DEL_LF_ROWS
------      ---------   --------    ------------
25          11          14          0
1 row selected.
```

Tổ chức lại index nếu nó có tỷ lệ các hàng bị xoá cao, ví dụ khi tỷ lệ DEL_LF_ROWS với LF_ROWS vượt quá 30% .

15.3.9. Xoá các index

Một index cần được xoá đi trong những trường hợp sau đây:

- Một index không cần thiết cho ứng dụng nữa.
- Một index có thể được xoá đi khi thực hiện load nhiều dữ liệu, và tạo lại sau khi đã load xong dữ liệu.
- Một index có thể được đánh dấu không hợp lệ (INVALID) khi có một instance hỏng trong quá trình nào đó ví dụ như load dữ liệu. Trong trường hợp đó index cần được xoá đi và tạo lại.
- Index bị hỏng.

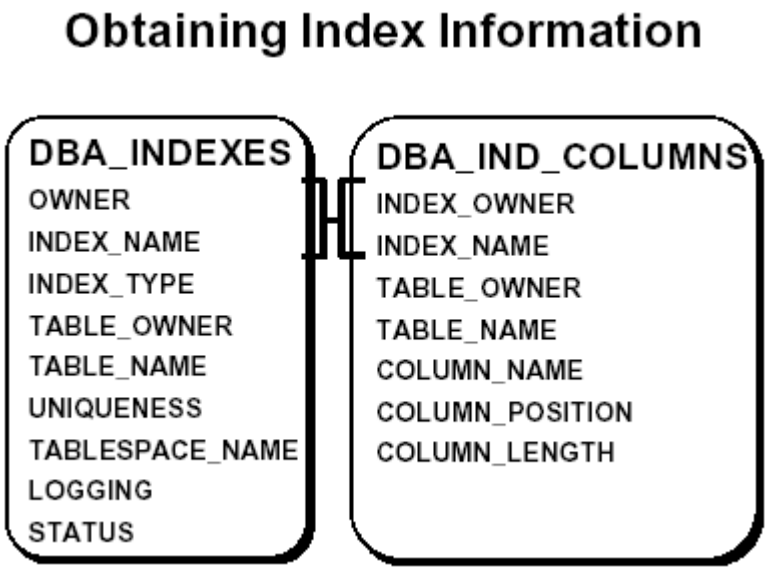
Cú pháp:

```
DROP INDEX [schema.]index;
```

15.4.THÔNG TIN VỀ CÁC INDEX

15.4.1. Xem thông tin về các index

Data dictionary có các view dùng để xem thông tin về index, hai view thông dụng nhất hay sử dụng là DBA_INDEXES và DBA_IND_COLUMNS.



Hình vẽ 81. Thông tin về Index

Sử dụng câu lệnh sau đây kiểm tra tên kiểu và trạng thái của index của user SCOTT:

```
SVRMGR> SELECT index_name, tablespace_name, index_type,
2> uniqueness, status
3> FROM dba_indexes
4> WHERE owner='SCOTT';

INDEX_NAME          TABLESPACE_NAME  INDEX_TYPE   UNIQUENES  STATUS
------------        ---------------  ----------   ---------  ------
EMP_LNAME_IDX       INDX01           NORMAL       NONUNIQUE  VALID
ORD_ORD_NO_IDX      INDX01           NORMAL       UNIQUE     VALID
ORD_REGION_ID_IDX   INDX02           BITMAP       NONUNIQUE  VALID
3 rows selected.
```

Cột INDEX_TYPE chỉ định index là Bitmap hay Normal, sử dụng câu lệnh sau liệt kê tên của tất cả các index khoá ngược:

```
SVRMGR> SELECT o.object_name
2> FROM dba_objects o
3> WHERE owner='SCOTT'
4>    AND o.object_id IN (SELECT i.obj#
5>    FROM ind$ i
6>    WHERE BITAND(i.property,4) = 4);
OBJECT_NAME
-----------------
ORD_ORD_NO_IDX
1 row selected.
```

15.4.2. Tìm các cột trong một index

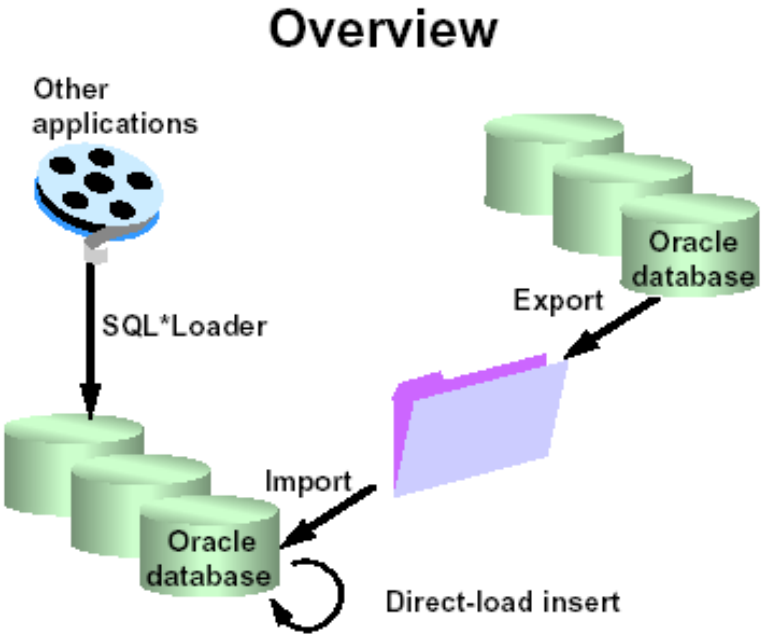
Câu lệnh truy vấn sau đây liệt kê các index của user SCOTT và chỉ ra các bảng và cột trên đó index được xây dựng:

```
SVRMGR> SELECT index_name, table_owner, table_name, column_name
2> FROM dba_ind_columns
3> WHERE index_owner = 'SCOTT'
4> ORDER BY index_name, column_position;

INDEX_NAME          TABLE_OWNER   TABLE_NAME       COLUMN_NAME
----------------- -------------  ---------------  ----------
EMP_LNAME_IDX       SCOTT         EMP              LAST_NAME
ORD_ORD_NO_IDX      SCOTT         ORD              ORD_NO
ORD_REGION_ID_IDX   SCOTT         ORD              REGION_ID
3 rows selected.
```

Chương 16. NẠP VÀ TỔ CHỨC LƯU TRỮ DỮ LIỆU

16.1.GIỚI THIỆU CHUNG

16.1.1. Tổng quan việc nạp dữ liệu



Hình vẽ 82. Tổng quan về việc nạp và lưu trữ dữ liệu

Có một số phương pháp khác nhau để có thể load (nạp) dữ liệu vào trong tables của oracle database, các phương pháp được đề cập trong chương này bao gồm.

- Công cụ direct load insert: nạp dữ liệu trực tiếp.
- SQL*loader: nạp dữ liệu từ file text, khuôn dạng tự do
- Công cụ Import và Export: nạp dữ liệu từ file lưu trữ với khuôn dạng do Oracle quy định.

Direct load insert

Direct load insert có thể được sử dụng để sao chép (copy) dữ liệu từ một bảng sang một bảng khác trong cùng một database. Sử dụng phương pháp này có thể tăng tốc độ của quá trình insert dữ liệu do có thể bỏ qua vùng đệm dữ liệu, dữ liệu được ghi trực tiếp vào trong database.

SQL loader

SQl * loader là công cụ được sử dụng để load dữ liệu vào trong oracle database sử dụng các file dữ liệu bên ngoài, công cụ này thường dùng chuyển dữ liệu từ hệ thống khác (như FoxPro, Access,...) vào trong Oracle.

Eport và Import

Công cụ Export cho phép các users tách thông tin trong dictionary views và dữ liệu trong Oracle Database và chuyển chúng vào trong một file của hệ điều hành theo định dạng file nhị phân của Oracle.

File sinh ra bởi công cụ Export có thể đọc bởi công cụ Import để đưa dữ liệu đọc được vào trong cùng một database hoặc vào một database khác.

16.1.2. Nạp dữ liệu trực tiếp

Khi nạp dữ liệu trực tiếp (direct load insert) có thể sử dụng mệnh đề APPEND như sau:

Cú pháp:

```
INSERT /*+APPEND*/ INTO [schema.]table
      [ [NO]LOGGING]
      sub_query ;
```

Trong đó:

```
schema          là owner của bảng
table           là tên của bảng
sub_query       là câu lệnh query để lấy các hàng theo yêu cầu
```

Direct insert load chỉ được dùng khi câu lệnh `INSERT INTO SELECT` được sử dụng. Tuỳ chọn này không có khi câu lệnh `INSERT INTO VALUES` được sử dụng. Direct load insert đươc dùng cho cả nonpartitioned table và partitioned table. Công cụ này cho phép kiểm tra các index và các constraint của bảng.

Chế độ logging

Khi insert sử dụng tuỳ chọn `LOGGING` (là giá trị mặc định) câu lệnh này sinh ra các điểm vào cho redo log, thực hiện việc recovery dữ liệu đầy đủ nếu trong quá trình load có lỗi.

Nếu `NOLOGGING` được sử dụng, tất cả các thay đổi trong dữ liệu sẽ không được lưu trong vùng đệm redo log, một số thông tin nhỏ về logging được đưa vào redo log cho các câu lệnh mở rộng vùng lưu trữ

Chế độ `NOLOGGING` sử dụng nếu như các thuộc tính đã được khởi tạo cho bảng.

Nếu một số thao tác update online đối với dữ liệu trên bảng xảy ra thường xuyên nên khởi tạo thuộc tính `NOLOGGING` khi load và khởi tạo lại `LOGGING` khi load dữ liệu đã hoàn thành.

Direct load insert cho phép các giao dịch khác đồng thời tạo thay đổi trên bảng

Toàn bộ dữ liệu đưa vào bảng theo phương pháp này sẽ được load vào vùng chỉ định bởi High Water Mark. Nếu bảng chứa nhiều block nơi các hàng đã bị xoá, không gian có thể sẽ không được sử dụng, việc truy vấn trên toàn bộ bảng có thể sẽ chậm đi.

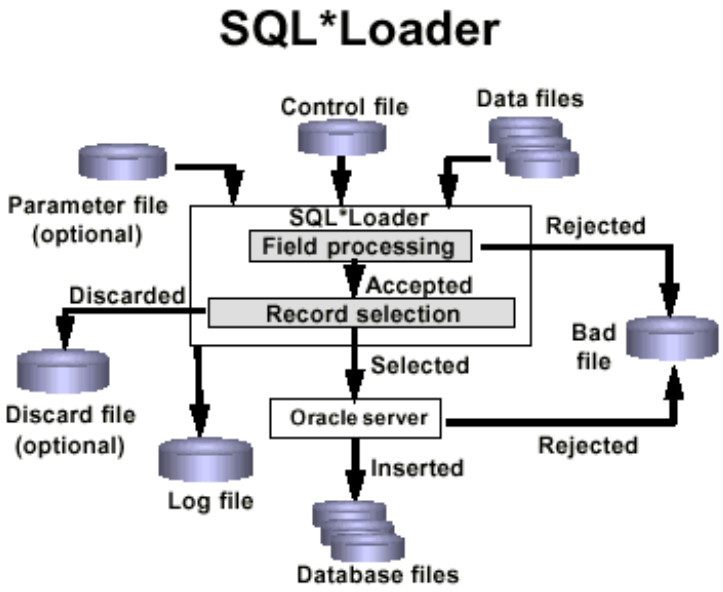
16.2.NẠP DỮ LIỆU

16.2.1. Nạp dữ liệu bằng SQL* Loader

Sql* Loader nạp dữ liệu từ một file bên ngoài database vào trong Oracle database.

Các đặc tính của SQL loader:

- Có thể sử dụng một hay nhiều file đầu vào
- Một vài bản ghi đầu vào có thể được kết hợp vào trong một bản ghi logic trong quá trình nạp.
- Các trường đầu vào có thể có độ dài thay đổi hoặc như nhau.
- Dữ liệu đầu vào có thể có các định dạng khác nhau – kí tự, nhị phân, date.
- Dữ liệu có thể được load từ các phương tiện lưu trữ khác nhau như đĩa, băng từ.
- Dữ liệu có thể được load vào một hoặc nhiều bảng trong một lần chạy.
- Có tuỳ chọn cho phép thay thế hay nối tiếp dữ liệu vào trong các bảng.
- Các hàm SQL có thể được sử dụng ngay trên dữ liệu đầu vào trước khi dữ liệu được lưu trong database.
- Giá trị các cột có thể tự động được sinh ra dựa trên một nguyên tắc nào đó, ví dụ: giá trị khoá tuần tự có thể được sinh ra và lưu trong các cột của bảng.
- Dữ liệu có thể được load trực tiếp vào trong bảng mà không cần phải sử dụng đến vùng đệm.



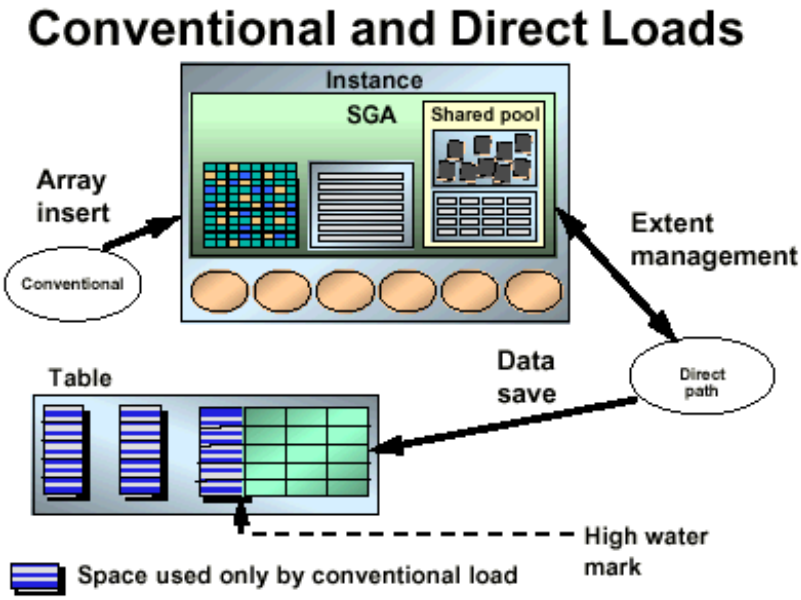
Hình vẽ 83. SQL*loader

SQL*Loader sử dụng các file sau đây

- Control file (file điều khiển): dùng để định dạng cho dữ liệu đầu vào, bảng đầu ra và các điều kiện tuỳ chọn có thể được sử dụng để load một phần các bản ghi tìm thấy trong file dữ liệu đầu vào.
- Data file (file dữ liệu): chứa dữ liệu đầu vào theo định dạng được định nghĩa bởi control file.
- Parameter file (file tham số) là file tuỳ chọn có thể sử dụng chứa các tham số dòng lệnh cho quá trình load.

- Bad File: được sử dụng bởi công cụ load dùng để ghi các bản ghi bị loại bỏ trong quá trình load (hiện tượng này có thể xảy ra khi các bản ghi được load vào trong bảng và bị loại ra sau khi kiểm tra tính hợp lệ đối với các trường)
- Log File: được tạo bởi SQL*loader chứa một bản ghi các thông tin trong quá trình load dữ liệu.
- Discard File: là một file có thể được tạo khi cần thiết, file này chứa tất cả các bản ghi không thoả mãn điều kiện lựa chọn.

16.2.2. Phương pháp nạp dữ liệu



Hình vẽ 84. Phương thức nạp dữ liệu

SQL* Loader sử dụng hai phương pháp load dữ liệu:

- Conventional load - nạp dữ liệu thông qua mảng
- Direct load - nạp dữ liệu trực tiếp

Conventional load

Phương pháp conventional load xây dựng một mảng các hàng được insert và sử dụng câu lệnh `INSERT` để load dữ liệu. Trong quá trình load dữ liệu theo phương pháp conventional load các bản ghi đưa vào sẽ được phân tích dựa vào các trường được chỉ định. Một mảng của

các bản ghi được tạo lập và chèn vào trong bảng theo chỉ định của file điều khiển. Các bản ghi không thoả mãn điều kiện của các trường được chỉ định thì sẽ bị loại bỏ và các bản ghi không thoả mãn điều kiện lựa chọn đặt ra sẽ bị từ chối.

Có thể sử dụng phương pháp conventional load để load dữ liệu vào trong cả các bảng cluster hay các bảng không được cluster.

Thông tin redo log được sinh ra và điều khiển bởi thuộc tính `LOG` cho các bảng được load.

Direct load

Theo phương pháp direct load, oracle server xây dựng các block dữ liệu trên bộ nhớ và cất các block này trực tiếp vào trong các vùng extent được cấp phát cho bảng được dùng trong quá trình load .

Redo log không được sinh ra trừ khi database đang ở chế độ `ARCHIVE LOG`, Direct load sử dụng các trường đã chỉ định để xây dựng toàn bộ các blocks của dữ liệu và trực tiếp ghi toàn bộ các block đó vào trong các datafile. Quá trình load dữ liệu này có thể bỏ qua vùng đệm dữ liệu trên bộ nhớ, việc truy xuất vùng nhớ SGA chỉ để quản lý việc mở rộng các extents và hiệu chỉnh giá trị High Water Mark.

Phương pháp direct load cho phép nạp dữ liệu nhanh hơn so với phương pháp conventional load. Nhưng phương pháp này không sử dụng được trong một số tình huống nhất định. Phần tiếp theo sẽ trình bày so sánh giữa hai phương pháp load dữ liệu.

16.2.3. So sánh hai phương pháp nạp dữ liệu

Bảng sau đây so sánh hai phương pháp load dữ liệu

Conventional load	Direct load
Sử dụng lện `COMMIT` để ghi nhận thay đổi dữ liệu thường trú	Sử dụng các lệnh lưu trữ dữ liệu
Luôn có các redo log tương ứng	Chỉ tạo redo log trong một số trường hợp đặc biệt
Thoả mãn tất cả các ràng buộc	Chỉ cần thoả mãn khoá chính, khoá duy nhất và điều kiện `NOT NULL`
Thực hiện các trigger `INSERT` kèm theo (nếu có)	Không thực hiện các trigger `INSERT` kèm theo
Có thể nạp dữ liệu vào clustered tables	Không cho phép nạp dữ liệu vào clustered tables
User khác vẫn có thể thay đổi dữ liệu trên table, trong khi nạp.	Trong khi nạp, các user khác không được phép sửa đổi dữ liệu của bảng đó

Phương pháp lưu trữ dữ liệu

Phương pháp conventional load sử dụng câu lệnh SQL và COMMIT cho việc cất dữ liệu, quá trình chèn một mảng dữ liệu và tiếp theo là câu lệnh COMMIT. Mỗi dữ liệu load có thể liên quan đến một vài giao dịch.

Phương pháp direct load sử dụng phương pháp ghi các block dữ liệu vào trong oracle data file, các đặc tính khác nhau sau đây giữa hai quá trình cất dữ liệu trên hai phương pháp:

- Trong quá trình dữ liệu được cất, toàn bộ các block được cất vào trong oracle database.
- Các block này được ghi vào trong sau giá trị High Water Mark của bảng.
- Một quá trình cất dữ liệu sẽ không kết thúc giao dịch.
- Các index không được cập nhật trong mỗi lẫn cất dữ liệu.

Lưu lại các thay đổi

Conventional load sinh ra điểm vào redo log giống như các câu lệnh DML, khi sử dụng direct load, các điểm vào đó không được sinh ra nếu như database ở trong chế độ:

- Database trong chế độ NOARCHIVELOG
- Database trong chế độ ARCHIVELOG nhưng tham số LOGGING=DISABLE (chế độ logging bị disable khi khởi tạo thuộc tính NOLOG cho bảng hay sử dụng mệnh đề UNRECOVERABLE trong file điều khiển.

Thiết lập các ràng buộc

Trong quá trình sử dụng conventional load tất cả các constraint được enable sẽ được thiết lập, các constraint này được sử dụng trong quá trình thực hiện các câu lệnh DML.

Khi thực hiện Direct load các constraints được sử dụng như sau:

- Các NOT NULL constraint được kiểm tra khi các mảng được xây dựng.
- Ngoại khoá(Foreign Key) và các CHECK constraint bị DISABLE và được ENANBLE thực hiện xong quá trình load dữ liệu bằng cách sử dụng câu lệnh tương ứng trong tệp điều khiển. Ngoại khoá bi disable bơi vì chúng tham chiếu đến các hàng khác hay các bảng khác. Các CHECK constraint bị DISABLE bởi vì chúng có thể sử dụng các hàm SQL, nếu một số lượng nhỏ các hàng được chèn vào trong một bảng lớn nên sử dụng phương thức load conventional load.
- Khoá chính và khoá duy nhất (unique) được kiểm tra trong quá trình load và khi kết thúc quá trình load chúng có thể bị disable nếu chúng không hợp lệ.

Thực hiện các Trigger Insert

Trong khi các trigger insert được thực hiện trong quá trình conventional load thì chúng lại bị DISABLE trước khi thực hiện việc load bằng phương thức direct. Chúng có thể vẫn trong trạng thái DISABLE nếu như đối tượng được tham chiếu tới không thể truy xuất khi kết thúc quá trình chạy. Cần xem xét việc sử dụng phương thức conventional load khi load dữ liệu vào trong bảng với trigger insert.

Load dữ liệu vào trong clustered table

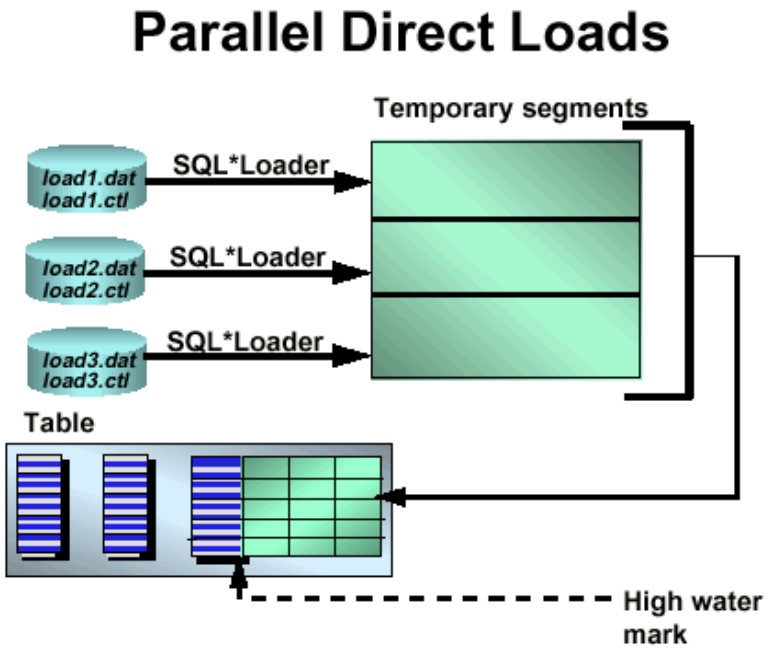
Phương thức direct load không thể sử dụng cho các clustered table, chỉ có thể sử dụng phương thức conventional load cho các bảng clustered table.

Quá trình khoá (Locking)

Trong khi thực hiện direct load, các transactions (giao dịch) khác không thể ghi lại bất kỳ thay đổi nào trên bảng đang được load, ngoại trừ việc sử dụng phương thức parallel direct load.

16.2.4. Nạp dữ liệu đồng thời (Parallel direct load)

Parallel direct load cho phép sử dụng một vài session direct load đồng thời dữ liệu vào trong một bảng.



Hình vẽ 85. Nạp dữ liệu đồng thời

Tính tuần tự của các câu lệnh

Sử dụng các file dữ liệu vào khác nhau cho mỗi session trong khi sử dụng parallel direct load, khi các sesion trong parallel direct load được khởi tạo, quá trình load thực hiện các bước sau đây:

1. Mỗi session sử dụng một temporary segment để load dữ liệu từ tệp dữ liệu đầu vào. Những temporary segment này được tạo trong tablespace mà bảng nằm trên đó. Nếu tablespace chứa một vài datafile thì một user có thể chỉ cho mỗi session file nơi temporary segment được tạo. Tham số lưu trữ cho các segment này có thể được chỉ định bởi user. Theo mặc định, các segment này sử dụng các tham số lưu trữ giống với bảng đang được load.
2. Extent cuối cùng trong mỗi temporary segment sẽ được cắt đi để thu hồi không gian không được sử dụng khi session kết thúc.

3. Toàn bộ các temporary segment được kết hợp lại hình thành nên một segment vào cuối của quá trình load dữ liệu.
4. Segment đó được thêm vào segment của bảng.

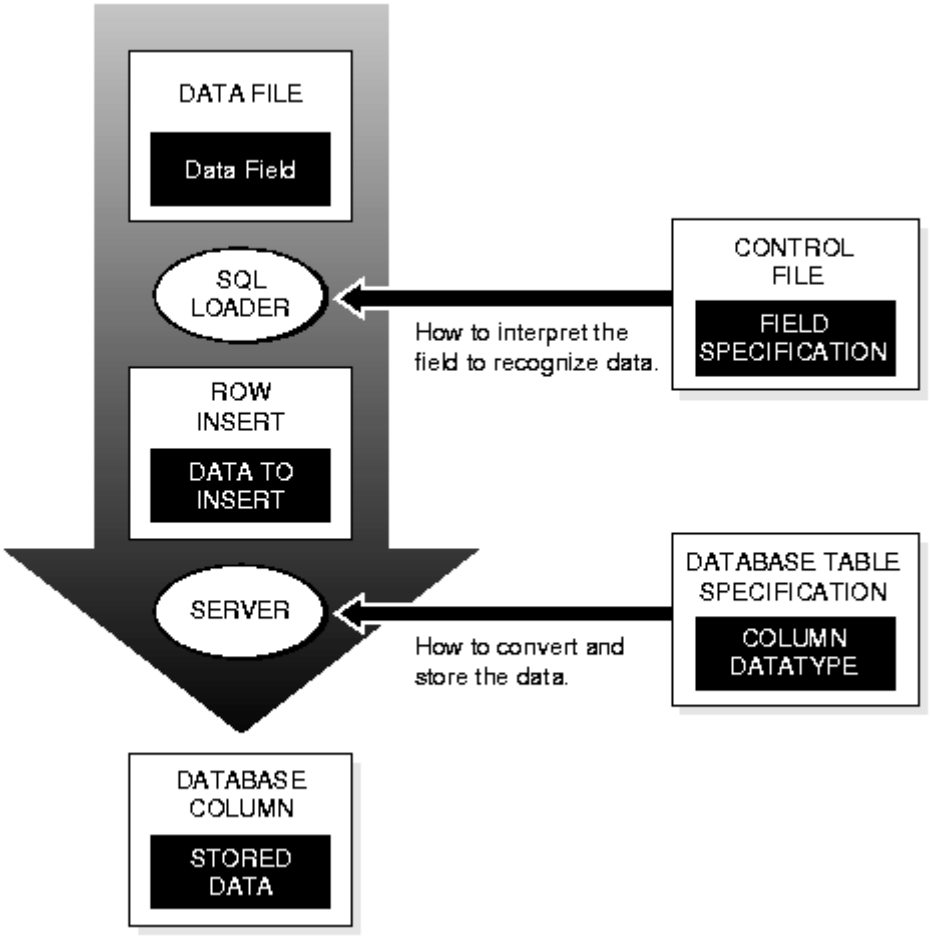
Các hạn chế

Việc sử dụng parallel direct load có các hạn chế sau đây:

- Các indexes không được xem xét trong quá trình load, xoá các indexes trước khi load sử dụng paralell và tạo lại chúng sau khi load xong.
- Tính toàn vẹn của tham chiếu, `CHECK` constraint và các trigger phải được `DISABLE` và được `ENABLE` lại bằng tay sau đó.
- Các hàng chỉ có thể nối tiếp vào với dữ liệu tồn tại trong bảng do các quá trình load riêng lẻ không thể kết hợp được với nhau. Nếu dữ liệu trong bảng cần thay thế thì phải sử dụng câu lệnh `TRUNCATE` bảng trước khi sử dụng parallel load.

*16.3.NẠP DỮ LIỆU BẰNG SQL*LOADER*

16.3.1. Sử dụng SLQ*LOADER



Hình vẽ 86. Nạp dữ liệu bằng SQL*Loader

Sử dụng câu lệnh sau đây để thực hiện load dữ liệu:

Cú pháp:

```
C:\> sqlldr[keyword=]value[[[,]keyword=]value]...
```

Với:

keyword	là từ khoá
value	là giá trị được gán cho từ khoá

Ví dụ:

```
$sqlldr scott/tiger \
> control=ulcase6.ctl \
> log=ulcase6.log direct=true
```

Chú ý:

Nếu từ khoá không được chỉ định thì giá trị cần được chỉ đúng như trật tự trong câu lệnh.

Các từ khoá được sử dụng khi nạp dữ liệu

Từ khoá	Diễn giải
USERID	Username và mật khẩu tương ứng
CONTROL	Tên của control file
LOG	Tên của log file
BAD	File lưu trữ các bản ghi nạp hỏng (tên mặc định là controlfile.bad)
DATA	Tên file dữ liệu đầu vào
DISCARD	Bỏ qua file chứa các bản ghi đã được lưu trữ nhưng không được chọn
DISCARDMAX	Số lượng tối đa các discards được phép
SKIP	Số lượng bản ghi được bỏ qua, sử dụng đến trong trường hợp tiếp tục nạp dữ liệu khi gặp lỗi
LOAD	Số lượng các bản ghi tiếp tục được nạp sau khi SKIP
ERRORS	Số lượng tối đa các bản ghi lỗi
ROWS	Số dòng dữ liệu trong mảng được tạo trước khi nạp dữ liệu (đối với phương pháp nạp thông thường)
BINDSIZE	Dung lượng tối đa (tính theo đơn vị bytes) dùng để tạo mảng các dòng dữ liệu nạp vào database, đối với trường hợp nạp thông thường
DIRECT	SQL*Loader sử dụng phương pháp nạp trực tiếp nếu tham số này được đặt là TRUE.
PARFILE	Tên của các file chứa các tham số nạp dữ liệu
PARALLEL	Tham số chỉ dùng khi nạp dữ liệu trực tiếp. Chỉ số lượng luồng nạp trực tiếp có thể thực hiện đồng thời.
FILE	File chứa temporary segment sử dụng đến khi nạp trực tiếp

Các tham số còn có thể định nghĩa trong file điều khiển.

16.3.2. Parameter file (tệp tham số)

Trong trường hợp thực hiện lệnh nạp dữ liệu với nhiều tham số tuỳ chọn khác nhau, khi này ta có thể gom các tham số tuỳ chọn này vào trong cùng một file tham số.

Tệp tham số có thể được sử dụng để chỉ định các tham số cho quá trình load dữ liệu, sử dụng tệp tham số để lưu các thông số được sử dụng cho quá trình load. Tệp này sử dụng định dạng sau đây để định nghĩa tham số:

```
<KEYWORD> = <VALUE>
```

Tham số PARFILE được dùng để xác định tên của file tham số.

Ví dụ:

```
SQLLDR PARFILE=example.par
```

```
Với nội dung của file tham số example.par là:

userid=scott/tiger
control=example.ctl
errors=9999
log=example.log
```

Lưu ý:

Trong ví dụ trên, example.ctl là tên của control file - sẽ được nhắc tới trong phần sau.

Các từ khoá sử dụng trong file tham số chính là các từ khoá dùng để nạp dữ liệu như đã nói ở trên.

16.3.3. Control file (tệp điều khiển)

Tệp điều khiển bao gồm các thành phần sau:

- Tên của tệp dữ liệu đầu vào sử dụng mệnh đề `INFILE`.
- Sự hợp thành các bản ghi logic từ một bản ghi vật lí trong file dữ liệu đầu vào, sử dụng mệnh đề như `CONCATENATE` và `CONTINUEIF`.
- Các trường chỉ định bao gồm vị trí, kiểu dữ liệu, delimiter sử dụng mệnh đề `FIELDS`.
- Tên của bảng và phương pháp load dữ liệu, xác định dữ liệu có được load vào bảng trống hay chèn các bản ghi mới sau khi xoá các bản ghi đã tồn tại, hoặc gắn thêm các hàng vào bảng đã tồn tại dữ liệu, sử dụng mệnh đề `INTO TABLE`.
- Các bản ghi được bỏ qua cho mỗi bảng sử dụng mệnh đề `CONTINUE_LOAD`.
- Điều kiện có thể được sử dụng cho việc lựa chọn các hàng được load sử dụng mệnh đề `WHEN`.
- Các cột được load
- Quy tắc cho việc sinh ra các giá trị cột, sử dụng mệnh đề `RECNUM`, `SYSDATE` và áp dụng các hàm SQL
- Các tham số load sử dụng trong mệnh đề `OPTIONS`
- Chỉ định các tham số lưu trữ cho phân đoạn temporary được tạo khi sử dụng parallel load.
- Các comment (chú dẫn) sử dụng tiền tố “--“
- Các tuỳ chọn cho direct load như: `SINGLEROW` (bảo trì các index trên hàng dựa vào hàng cơ sở) `REENABLE` (để thiết lập lại các constraint khi quá trình chạy kết thúc), `SORTED_INDEXES` (chỉ định dữ liệu được xắp sếp trước), `UNRECOVERABLE` (không sinh ra các thông tin redo log).

Chú ý:

- Khởi tạo giá trị `NOLOG` cho bảng đang sử dụng từ khoá `NOLOGGING` tương đương với việc sử dụng tuỳ chọn `RECOVERABLE` trong tệp điều khiển.

- Dữ liệu có thể được đặt chung vào trong tệp điều khiển bằng cách chỉ định tham số INFILE * và sử dụng từ khoá BEGINDATA để đánh dấu phần bắt đầu của dữ liệu, nếu tuỳ chọn này được sử dụng thì tất cả các bản ghi đặt sau từ khoá BEGINDATA sẽ được biên dịch như là dữ liệu.

Một số từ khoá hay sử dụng trong Control file:

Từ khoá	Diễn giải
INFILE	Xác định tên file chứa dữ liệu nạp vào database, dấu * cho biết dữ liệu sẽ được lấy ngay trong control file, phía sau của từ khoá BEGINDATA
BEGINDATA	Từ khoá xác định điểm bắt đầu chứa dữ liệu, sau từ khoá này là dữ liệu cần nạp vào database.
READBUFFERS	Yêu cầu sử dụng bộ nhớ đệm để nạp dữ liệu vào database, sử dụng trong phương pháp direct load.
BADFILE	Tên của các file lưu các dữ liệu không thể nạp được vào database do phát sinh lỗi trong quá trình nạp dữ liệu.
DISCARDFILE	Tên của file lưu các bản ghi bị bỏ qua không nạp vào database do không đúng với tiêu chuẩn nạp dữ liệu
CHARACTERSET	Tên tập ký tự sử dụng trong datafile
INSERT	Thêm mới 1 dòng dữ liệu trong database
APPEND	Chèn thêm một dòng dữ liệu vào cuối cùng của bảng
TRAILING NULLCOLS	Điền giá trị null vào cột. TRAILING NULLCOLS được sử dụng cùng với WHITESPACE
POSITION	Từ khoá dùng để xác định vị trí của dữ liệu cần nạp
CONSTANT	Đặt giá trị hằng số cho cột dữ liệu
RECNUM	Đếm số lượng dòng dữ liệu đã được nạp
SYSDATE	Trả về giá trị ngày giờ hiện thời
TERMINATED	Từ khoá xác định phân cách kết thúc
ENCLOSED	Từ khoá dùng để xác định đường bao dữ liệu

Ví dụ: Nội dung của một control file

```
1. Dữ liệu được nạp trực tiếp
      LOAD DATA
      INFILE *
      INTO TABLE dept
      FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
      (deptno, dname, loc)
      BEGINDATA
      12,RESEARCH,"SARATOGA"
```

```
10,"ACCOUNTING",CLEVELAND
11,"ART",SALEM
13,FINANCE,"BOSTON"
21,"SALES",PHILA.
22,"SALES",ROCHESTER
42,"INT'L","SAN FRAN"
```

Ví dụ ở trên, sử dụng dấu phẩy (,) để ngăn cách giữa các trường dữ liệu nạp vào database.

2. Dữ liệu cần nạp đặt trong file 'ulcase2.dat'

```
LOAD DATA
INFILE 'ulcase2.dat'
INTO TABLE emp
 (empno         POSITION(01:04)   INTEGER  EXTERNAL,
 ename          POSITION(06:15)   CHAR,
 job            POSITION(17:25)   CHAR,
 mgr            POSITION(27:30)   INTEGER EXTERNAL,
 sal            POSITION(32:39)   DECIMAL  EXTERNAL,
 comm           POSITION(41:48)   DECIMAL  EXTERNAL,
 deptno         POSITION(50:51)   INTEGER  EXTERNAL)
```

Dữ liệu trong file 'ulcase2.dat' là:

```
7782 CLARK      MANAGER   7839 2572.50           10
7839 KING       PRESIDENT      5500.00           10
7934 MILLER     CLERK     7782  920.00           10
7566 JONES      MANAGER   7839 3123.75           20
7499 ALLEN      SALESMAN  7698 1600.00   300.00  30
7654 MARTIN     SALESMAN  7698 1312.50  1400.00  30
```

Ở ví dụ này, việc xác định các trường dữ liệu để nạp vào database dựa vào vị trí của cột dữ liệu trên mỗi dòng. Toán tử POSITION (Vị trí đầu: Vị trí cuối) sẽ thực hiện công việc này.

16.3.4. Data file

File dữ liệu chứa các bản ghi được xử lý theo một định dạng đã định nghĩa trong control file.

Dữ liệu trong data file thường là các dữ liệu ở dạng text. Thông thường thì các dữ liệu này có được do việc export từ các database khác loại như FoxPro, Access,...

Ví dụ data file: xem ví dụ phía trên

16.3.5. Các thành phần của log file

Log file luôn được tạo ra và nếu quá trình load dữ liệu hoàn thành mà log file không được tạo ra thì đó là do user thiếu quyền hoặc khoong đủ không gian đĩa.

Log file bao gồm các thông tin sau:

- Phần header: bao gồm thông tin về thời gian chạy, phiên bản của phần mềm.
- Các thông tin toàn cục: tên của input file và output file, các tham số dòng lệnh.
- Các thông tin bảng: tên bảng, điều kiện load và phương pháp load.
- Thông tin về các trường và cột.

- Thông tin về tệp dữ liệu: chỉ ra các bản ghi bị từ chối và loại bỏ và lí do bị từ chối hay loại bỏ.
- Thông tin load các bảng: số các hàng đã được load, số các hàng bị từ chối vì lỗi dữ liệu, số các hàng bị loại bỏ.
- Thông tin tổng hợp: hiển thị dữ liệu sau: số lượng không gian được thiết lập cho mảng, thông tin thống kê cho tất cả các data file.
- Thời gian bắt đầu và thời gian kết thúc quá trình load.

16.3.6. Các file đầu ra khác

Bad file

Bad file chứa các bản ghi bị từ chối trong quá trình xử lý vì một trong các lí do sau:

- Các bản ghi đầu vào có lỗi. Ví dụ như định dạng sai, độ rộng của trường quá lớn.
- Không thể chèn thêm các bản ghi vào table chẳng hạn như dữ liệu nạp vào không hợp lệ, các constraints bị vi phạm.

Discard file

Discard file chứa dữ liệu như định dạng của tệp dữ liệu đưa vào, nó chứa các bản ghi không thoả mãn điều kiện load.

16.3.7. Các hướng dẫn khi sử dụng load

Sử dụng các hướng dẫn sau đây khi dùng SQL* Loader để load dữ liệu giảm thiểu được lỗi và tăng hiệu năng trong quá trình load:

- Sử dụng tệp tham số để rút bớt các tham số dòng lệnh.
- Tách tệp điều khiển và tệp dữ liệu để cho phép sử dụng lại tệp điều khiển cho các session load khác nhau.
- Thiết lập không gian lưu dữ liệu dựa vào kích thước của dữ liệu tránh việc thiết lập các extents lưu trữ tự động trong khi load.
- Khi sử dụng direct load,các segment temporary được sử dụng sinh ra các index cho dữ liệu mới. Các index này được trộn với các index đã có của bảng vào lúc load. Bằng cách xếp dữ liệu đầu vào trong khoá của các index lớn nhất có thể làm giảm không gian dùng cho việc sort dữ liệu.
- Với parallel direct load có thể chỉ định vị trí của các phân đoạn tạm thời được sử dụng cho việc insert dữ liệu. Đối với mỗi session load dữ liệu nên chỉ định các datafile khác nhau.

Các lỗi xảy ra khi load dữ liệu

Khi quá trình load dữ liệu kết thúc không bình thường, toàn bộ dữ liệu đã được load đến thời điểm kết thúc đó sẽ được commit. Sau khi hiệu chỉnh lỗi, quá trình load có thể thực hiện tiếp như sau để hoàn thành quá trình load:

- Nếu đang load vào một bảng hay nhiều bảng có cùng số bản ghi được xử lý , sử dụng tham số SKIP để tiếp tục quá trình load.
- Nếu nhiều bảng đã được load và số các bản ghi đã được xử lý là không như nhau đối với các bảng , sử dụng mệnh đề CONTINUE_LOAD trong tệp điều khiển chỉ định số bản ghi bị bỏ qua cho mỗi bảng
- Kiểm tra các index được đánh dấu là Unusable . Hiện tượng này xảy ra khi index không nhất quán với bảng, xoá index có trạng thái như vậy tạo lại chúng khi load xong dữ liệu.

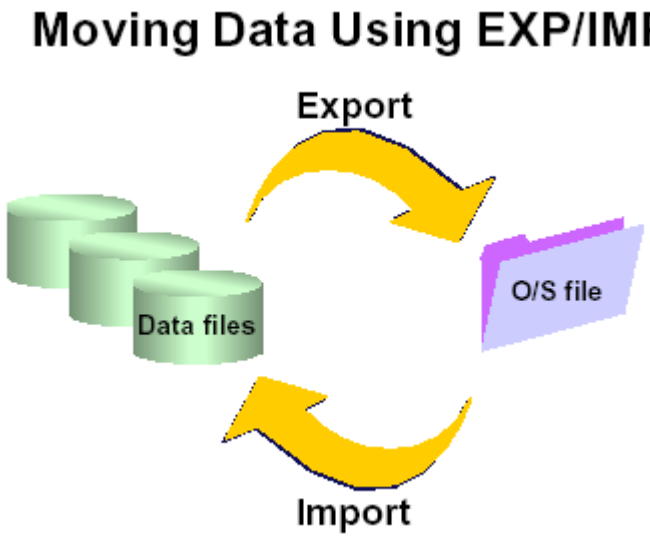
Giải quyết các lỗi khi load

- Không đủ không gian để load dữ liệu : hiện tượng này xảy ra khi đĩa cứng bị đầy hay khi giá trị các extent cấp phát đến giá trị MAXEXTENTS
- Hỏng Instance: restart lại instance và load lại dữ liệu.
- Dữ liệu không ở trong trật tự chỉ định: dữ liệu đã được load xong index ở trong trạng thái unusable, xoá và tạo lại các index.
- Trùng lặp giá trị trong primary key hay unique key: hiện tượng này không làm hỏng quá trình load song chúng có thể DISABLE các các constraint và index ở trạng thái UNUSABLE. Trong trường hợp các constraint bị lỗi, ta có thể sử dụng các bảng exception để bắt lỗi và sửa chúng. Trong trường hợp các unique index ở trạng thái UNUSABLE, cần tìm lỗi bằng cách tạo unique constraint trên cột làm index. Bắt các lỗi trên các bảng exception và sửa chúng.
- BINSIZE quá nhỏ: sử dụng giá trị lớn hơn cho việc load dữ liệu.
- Lỗi vì vượt quá giá trị discard được khởi tạo: hiện tượng này xảy ra khi số các bản ghi không hợp lệ vượt quá giá trị ERRORS hay DISCARDMAX được khởi tạo, nguyên nhân chung của hiện tượng này là sử dụng tệp dữ liệu đầu vào không đúng, kiểm tra và sửa lại tệp dữ liệu đầu vào.

16.4.TỔ CHỨC LẠI DỮ LIỆU BẰNG CÔNG CỤ EXPORT VÀ IMPORT

16.4.1. Công cụ dịch chuyển dữ liệu

Export và Import là công cụ cho phép người quản trị Oracle Database có thể chuyển dữ liệu giữa hai Oracle database vào chính bên trong Oracle database giữa các tablespace khác nhau hay giữa các user khác nhau.



Hình vẽ 87. Dịch chuyển dữ liệu

Công cụ Export

Công cụ Export có thể được sử dụng để tạo ra các bản copy logic của các đối tượng được định nghĩa và dữ liệu thành các tệp nhị phân. Export có thể ghi dữ liệu ra tệp trên đĩa hay trên băng từ.

Công cụ Import

Công cụ Import có thể đọc các tệp được tạo bằng công cụ Export, copy các đối tượng được định nghĩa và dữ liệu vào trong Oracle database. Công cụ import không thể đọc các text file hay các file được tạo trong bất kì đinh dạng nào khác.

Sử dụng công cụ Export và Import

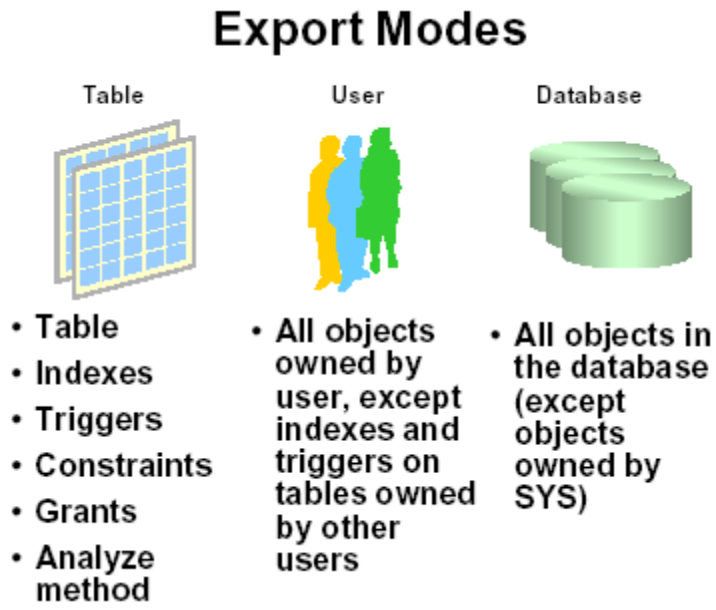
Công cụ Export và Import có thể được sử dụng trong những trường hợp sau:

- Tổ chức lại các bảng: dữ liệu của các bảng cần được chuyển từ tablespace này sang tablespace khác. Các bảng chứa nhiều hàng bị migration, một bảng có nhiều blocks có rất ít các hàng được chèn vào.
- Chuyển dữ liệu từ user này sang user khác: điều này cần thiết khi một user cần được thay đổi. Một dữ liệu được export bởi một user có thể import vào một user khác.
- Chuyển dữ liệu giữa các database: Các đối tượng xác định có thể được chuyển từ quá trình phát triển sang sản phẩm khi sử dụng export lấy phần thông tin cấu trúc và loại bỏ phần dữ liệu. Export và Import còn được dùng để lấy dữ liệu từ một ứng dụng OLTP vào trong Oracle data Warehouse.
- Chuyển dữ liệu giữa các Platform hay các phiên bản khác nhau của Oracle database.
- Thực thi quá trình Logical backup: toàn bộ hay một vài đối tượng trong database có thể được export, các export file có thể được sử dụng như là một Logical backup.

16.4.2. Các chế độ Export

Công cụ export cung cấp 3 kiểu export:

- User
- Table
- Database



Hình vẽ 88. Các chế độ export dữ liệu

Chế độ Table

Tất cả các user có thể sử dụng chế độ bảng để export các bảng thuộc về user đó, các user được cấp quyền có thể export bất cứ bảng thuộc về bất cứ một user nào trong database, sử dụng chế độ bảng để export:

- Cấu trúc các bảng
- Dữ liệu trong bảng
- Tất cả các index của bảng(chỉ thực hiện được khi dùng user được phân quyền).
- Tất cả các trigger trong bảng (chỉ thực hiện được khi dùng user được phân quyền).
- Các constraint trên bảng
- Tất cả các quyền trên bảng

Chế độ export theo user

Chế độ export theo user sử dụng để export dữ liệu thuộc về user:

- User có thể export các đối tượng do user đó sở hữu. Khi này các objects thuộc về user đều được export ngoại trừ các indexes và triggers do user đó sở hữu nhưng lại được áp dụng cho table của một user khác. Hoặc các indexes và triggers trên table do user này sở hữu nhưng các triggers và indexes này lại do user khác sở hữu.
- Các trigger và Index được tạo bởi các user khác nhưng trên bảng thuộc về user được export.

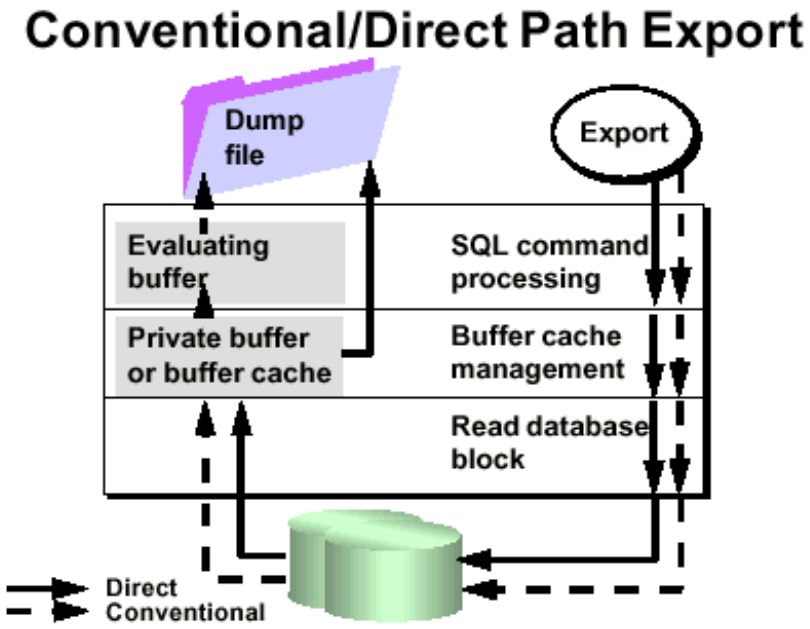
- Các user không được cấp quyền chỉ có thể export được các đối tượng thuộc về user đó.

Chế độ Full

Khi sử dụng chế độ full thì toàn bộ các đối tượng thuộc về database ngoại trừ các các đối tượng thuộc SYS sẽ được export. Chế độ này đòi hỏi quyền đặc biệt và không thể thực hiện với mọi user. Chú ý rằng trong ba chế độ export quyền của user được sủ dụng là role: EXP_FULL_DATABASE.

16.4.3. Export dữ liệu trực tiếp và Export dữ liệu thông thường

Trong Oracle sử dụng hai phương pháp load khác nhau : conventional và direct load



Hình vẽ 89. Các phương thức Export dữ liệu

Cách thông thường (Conventional load)

Conventional load là phương pháp mặc định được sử dụng bởi Oracle Loader để định dạng dữ liệu và ghi dữ liệu vào trong database.

Conventional load sử dụng câu lệnh SQL để lấy dữ liệu trong table. Dữ liệu được đọc từ đĩa vào trong vùng đệm. Sau đó, các hàng được chuyển vào vùng đệm kiểm tra giá trị. Dữ liệu sau khi kiểm tra giá trị sẽ được chuyển vào các tệp export.

Cách trực tiếp (Direct path)

Direct path load lấy dữ liệu nhanh hơn so với phương pháp conventional path export. Với phương pháp direct path load dữ liệu được đọc từ đĩa vào trong vùng đệm sau đó các hàng

được chuyển trực tiếp vào trong các export process. Vùng kiểm tra đánh giá bị bỏ qua do dữ liệu trong các block không được ghi nhận mang các hàng cùng với nó . Dữ liệu luôn sẵn sàng trong định dạng của các tệp export yêu cầu, nó tránh được quá trình chuyển đổi định dạng dữ liệu. Dữ liệu được chuyển vào các export process và các process này ghi dữ liệu vào trong các export file.

Công cụ Import có thể sử dụng các export file được tạo bởi bất kì một trong hai phương pháp trên. Phương pháp export không ảnh hưởng đến thời gian của quá trình import.

16.5.CÔNG CỤ EXPORT

16.5.1. Sử dụng công cụ Export

Công cụ export có thể gọi theo chế độ:

- Dòng lệnh
- Chế độ tương tác
- Chế độ graphic

Chế độ dòng lệnh (command line)

Cú pháp:

```
$exp [keyword=]{value|(value, value ...)}
[ [ [,] keyword=]{value|(value, value ...)} ] ...
```

Với:

```
keyword          là từ khoá được sử dụng trong câu lệnh export.
value            là giá trị được gán cho từ khoá.
```

Các tham số dòng lệnh sử dụng khi Import dữ liệu

Từ khoá	Mặc định	Diễn giải
ANALYZE	Y	Cho phép thực hiện lệnh ANALYZE đối với database trước khi import dữ liệu
BUFFER	Tuỳ theo hệ điều hành	Kích thước của bộ đệm sử dụng khi export. Kích thước buffer được xác định theo công thức: `buffer_size = rows_in_array * maximum_row_size`
COMPRESS	Y	Giá trị là Y, khi import dữ liệu, kích thước của extent khởi tạo được đặt bằng với kích thước của segment hiện thời.
CONSISTENT	N	Giá trị là Y, tất cả các thao tác export chỉ được thực hiện trong một read-only transaction.
CONSTRAINTS	Y	Giá trị là Y, các ràng buộc cũng được exported

		cùng với dữ liệu trong bảng.
DIRECT	N	Giá trị là Y, export trực tiếp
FEEDBACK	0	Hiển thị mức độ export, số dòng dữ liệu được export, tại mỗi lần. Ví dụ: FEEDBACK=10, Công cụ export sẽ hiển thị tiến trình cho biết mỗi lần thực hiện sẽ export được 10 dòng dữ liệu
FILE	expdat.dmp	Tên file dữ liệu ra. Tệp dữ liệu có phần mở rộng là .dmp
FULL	N	Giá trị là Y, export toàn bộ database. Để thực hiện được việc này, user cần được cấp quyền EXP_FULL_DATABASE
GRANTS	Y	Giá trị là Y, export các đối tượng trong database cùng với cả các quyền đang được áp dụng trên đối tượng đó
HELP	N	Giá trị là Y, hiển thị danh sách tham số và ý nghĩa tương ứng của chúng
INCTYPE		Các chế độ incremental export (Các chế độ này sẽ được nói cụ thể ở phần sau)
INDEXES	Y	Giá trị là Y, export cả các index.
LOG		Tên file lưu trữ các thông báo khi export. Ví dụ: Exp system/manager LOG=export.log
OWNER		Đưa ra danh sách các users sở hữu các đối tượng được export
PARFILE		Tên file chứa các tham số export
QUERY		Export dữ liệu thoả mãn một điều kiện nào đó. Ví dụ: exp scott/tiger tables=emp query = \"where job=\'SALESMAN\' and sal\<1600\"
RECORD	Y	Ghi lại chế độ dữ liệu là incremental hay cumulative vào trong các bảng hệ thống: SYS.INCEXP, SYS.INCFIL và SYS.INCVID
RECORDLENGTH	Tuỳ theo hệ điều hành	Kích thước tính theo bytes của bản ghi được export
ROWS	Y	Cho biết có export từng dòng dữ liệu trong bảng hay không.
STATISTICS	ESTIMATE	Phân tích và thống kê số lượng các dữ liệu đã export. Có một số chế độ: ESTIMATE, COMPUTE, và NONE
TABLES		Liệt kê danh sách các bảng export. Ví dụ: TABLES=(EMP#, DEPT, MYDATA)

TRANSPORT_TABLESPACE	N	Có export các transportable tablespace hay không
TABLESPACES		Khi tham số TRANSPORT_TABLESPACE = Y, TABLESPACES sẽ dùng để liệt kê danh sách tên các tablespace đã export
TRIGGERS	Y	Có export cả các trigger hay không?
USERID		Username và mật khẩu của user thực hiện export. Chúng được thể hiện theo khuôn dạng: username/password AS SYSDBA hay username/password@instance AS SYSDBA

Ví dụ:

```
1. Export trực tiếp bảng dữ liệu
      $exp scott/tiger tables=(dept,emp) \
      > file=emp.dmp log=exp.log compress=n \
      > direct=y recordlength=32768

2. Export dữ liệu sử dụng file tham số
      > exp system/manager parfile=params.dat
Nội dung của params.dat
      FILE=dba.dmp
      GRANTS=y
      FULL=y
      ROWS=y
```

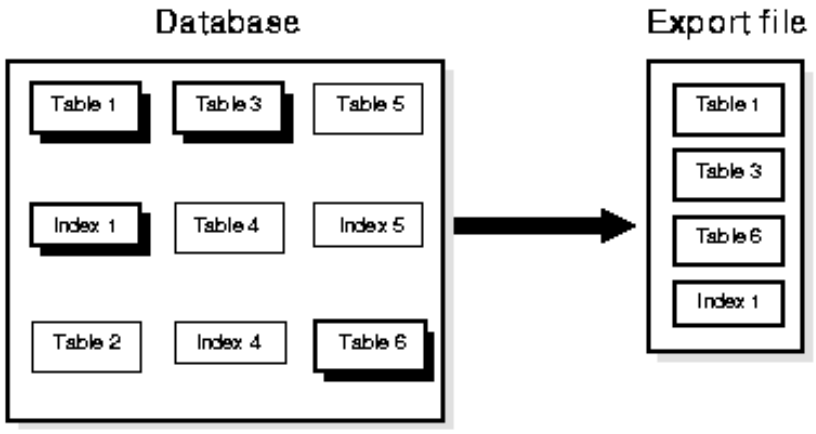
16.5.2. Giới thiệu một số chế độ export

Quá trình backup (sao lưu) dữ liệu cần được thường xuyên được thực hiện. Công việc Backup có thể được thực hiện theo định kỳ. Tuy vậy, với lần đầu tiên, ta vẫn phải thực hiện việc backup dữ liệu một cách đầy đủ. Tại các lần backup dữ liệu sau đó, dữ liệu trong database có thể có những biến đổi không nhiều so với dữ liệu của lần backup đầu tiên. Vì thế ta có các chế độ thực hiện backup dữ liệu khác nhau và có thể áp dụng các chế độ backup này tuỳ thuộc vào tình huống cụ thể.

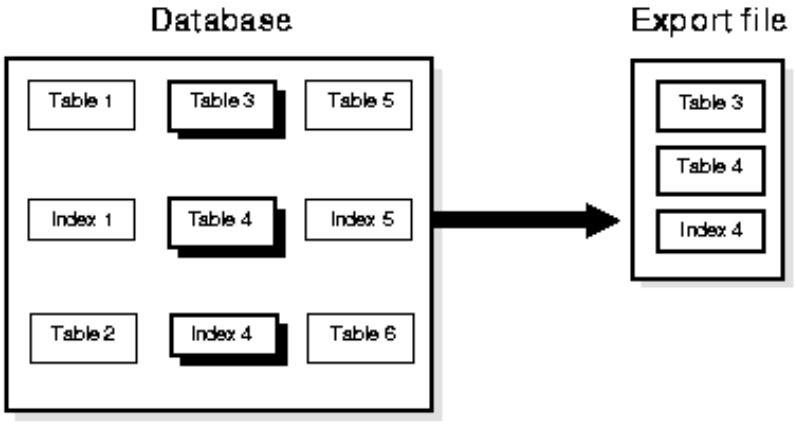
Incremental Exports (chế độ tăng trưởng)

Chế độ *incremental* chỉ thực hiện backup đối với các bảng có sự thay đổi so với lần thực hiện backup gần đó nhất. Trong chế độ này, toàn bộ dữ liệu của bảng và cả định nghĩa các bảng cũng sẽ được exports không phân biệt các dòng dữ liệu có thay đổi hay không có thay đổi (miễn là bảng có thay đổi, thì sẽ export toàn bộ bảng). Chế độ incremental Exports là chế độ thường hay được thực hiện nhất trong số các chế độ export.

Ví dụ: Giả sử lần Export đầu tiên, ta đã thực hiện export toàn bộ thông tin về bảng. Khi này tại lần thứ hai, sử dụng chế độ incremental Export, sau khi bảng dữ liệu đã có sự thông tin ở một dòng nào đó. Tất cả các bảng dữ liệu có sự thay đổi thông tin và cả các indexes đi kèm với bảng đó sẽ được export.



Trong lần thứ ba, có thay đổi trong table 3 và table 4. Khi này Chế độ Incremental Export sẽ tiếp tục export dữ liệu như sau:

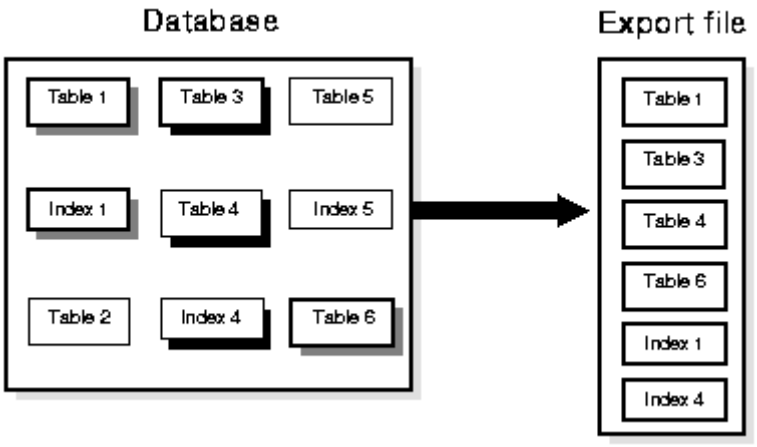


Hình vẽ 90. Export ở chế độ Incremental

Cumulative Exports (Chế độ tích luỹ)

Chế độ *cumulative* export sẽ thực hiện backup đối với các bảng có sự thay đổi kể từ lần thực hiện backup trước đó ở chế độ cumulative hay complete Export. Chế độ cumulative export sẽ thực hiện nén tất cả các lần thực hiện backup ở chế độ incremental exports vào những file đơn riêng lẻ. Ta không cần phải lưu trữ các file backup ở chế độ incremental export vì khi thực hiện backup theo chế độ cumulative export các file backup ở trên sẽ bị ghi đè lên bằng một file mới tương ứng.

Ví dụ: Trong chế độ cumulative Export ở lần thứ tư, khi có sự thay đổi tại table 1 và table 6 thì tất cả các bảng có thay đổi từ lần export toàn bộ gần nhất sẽ được export.



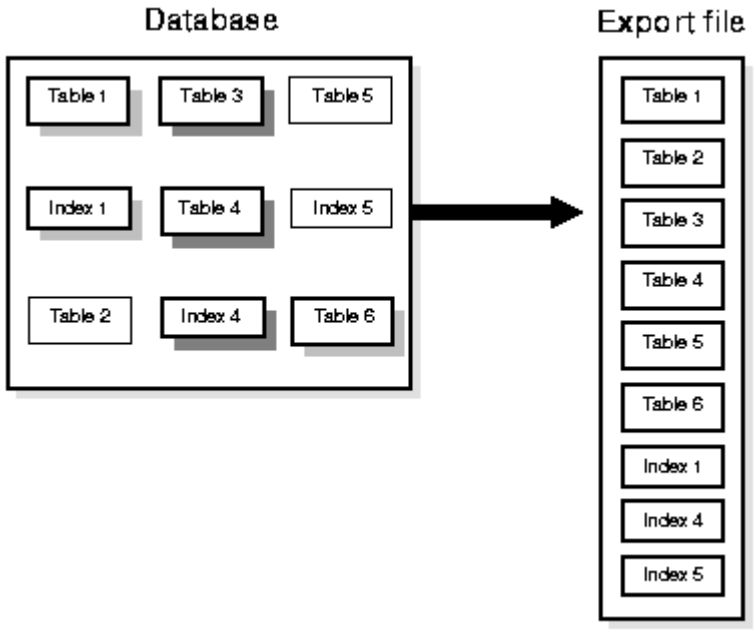
Hình vẽ 91. Export ở chế độ Cumulative

Ta quan sát thấy dữ liệu trong table 4 tuy có được thực hiện backup ở lần thứ 2 nhưng vẫn tiếp tục được Backup.

Complete Exports (Chế độ toàn bộ)

Chế độ *complete* Export sẽ thiết lập việc backup toàn bộ dữ liệu có trong database.

Ví dụ: Trong lần 5, thực hiện backup dữ liệu ở chế độ toàn bộ.



Hình vẽ 92. Export ở chế độ toàn bộ

Ta thấy ngay trong chế độ này, table 4 tuy không có thay đổi gì nhưng cũng vẫn bị export.

Ví dụ: thực hiện Export dữ liệu ở chế độ Incremental

```
> exp system/manager full=y inctype=incremental
```

16.5.3. Các tablespaces trao đổi

Oracle cho phép quản trị viên database có thể chuyển một tablespace từ một database sang một database khác, thông qua đặc điểm transportable tablespace.

Để di chuyển hay sao chép một tập hợp các tablespaces, trước tiên ta cần đưa các tablespaces về trạng thái read-only, rồi sao chép các datafiles tương ứng với các tablespaces này, và sử dụng công cụ Export/Import để di chuyển các thông tin database được lưu trữ trong data dictionary gọi là metadata (thông tin của các thông tin). Cả các datafiles và metadata được export sẽ được sao chép sang database đích. Tiếp theo, quản trị viên database cần import các metadata này vào database mới rồi thiết lập lại trạng thái cho các tablespace vừa được sao chép.

Một số từ khoá được sử dụng để thực hiện export các transportable tablespace metadata.

- TRANSPORT_TABLESPACE
- TABLESPACES

16.5.4. Một số thông báo khi export: Warning, Error, và Completion Messages

Log File

Ta có thể lưu lại tất cả các thông báo (messages) do công cụ Export phát ra vào trong một log file, thông qua việc sử dụng tham số LOG. Công cụ Export sẽ ghi lại chi tiết các thông tin thành công và cả các thông tin lỗi xảy ra trong quá trình thực hiện export.

Warning Messages (thông báo cảnh báo)

Đối với các lỗi xảy ra không nghiêm trọng (nonfatal errors) tiện ích Export sẽ không dừng việc export dữ liệu ngay. Ví dụ, khi có một lỗi xảy ra trong quá trình exporting một table, export sẽ hiển thị các thông báo lỗi error message rồi bỏ qua table hiện tại để tiếp tục chuyển sang các table khác. Các lỗi không nghiêm trọng được gọi là các *warnings (cảnh báo)*.

Export sẽ phát ra cảnh báo mỗi khi có lỗi không nghiêm trọng xảy ra.

Ví dụ, yêu cầu export một table không tồn tại, tiện ích Export sẽ đưa ra thông báo không có table tương ứng và tiếp tục thực hiện các table còn lại.

```
> exp scott/tiger tables=xxx,emp

Export: Release 8.1.6.0.0 - Production on Wed Oct 6 15:25:15
1999

(c) Copyright 1999 Oracle Corporation.  All rights reserved.

Connected to: Oracle8i Enterprise Edition Release 8.1.6.0.0 -
Production
With the Partitioning and Java options
PL/SQL Release 8.1.6.0.0 - Production
Export done in WE8DEC character set and WE8DEC NCHAR character
set

About to export specified tables via Conventional Path ...
```

```
EXP-00011: SCOTT.XXX does not exist
. . exporting table                          EMP         14
rows exported
Export terminated successfully with warnings.
```

Fatal Error Messages (thông báo lỗi nghiêm trọng)

Khi có các lỗi nghiêm trọng (*fatal errors)* xảy ra trong quá trình Export, công cụ sẽ ngừng session ngay lập tức. Thông thường, các lỗi này xảy ra do các lỗi hệ thống (internal problem) do thiếu tài nguyên, thiếu bộ nhớ... Ví dụ, khi chạy CATEXP.SQL, file script không chạy, công cụ Export sẽ phát sinh thông báo lỗi hệ thống (fatal error):

```
EXP-00024: Export views not installed, please notify your DBA
```

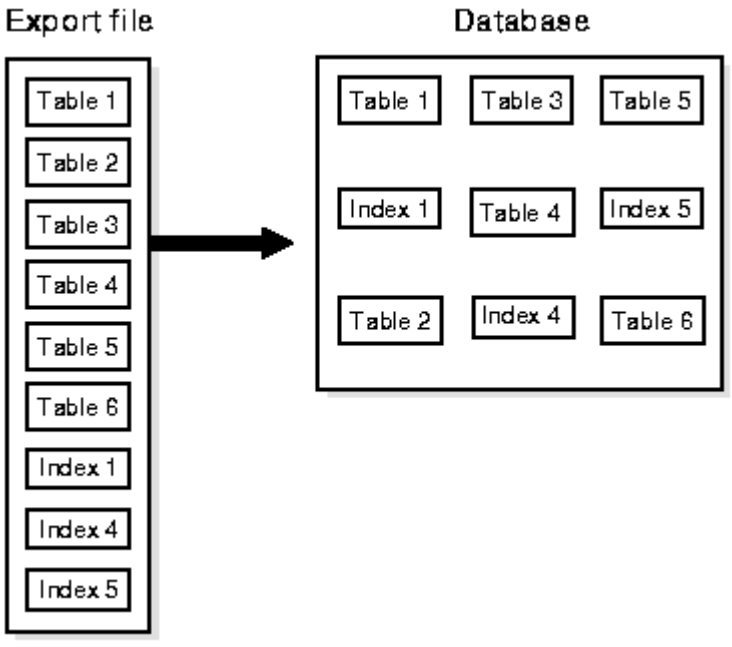
Completion Messages (Thông báo hoàn tất)

Khi Export đã hoàn tất và không có lỗi xảy ra, Export sẽ hiển thị thông báo "Export terminated successfully without warnings". Nếu có bất kỳ lỗi không nghiêm trọng nào xảy ra trong quá trình thực hiện, Export sẽ kết thúc với thông báo "Export terminated successfully with warnings". Khi có lỗi nghiêm trọng khiến hệ thống phải dừng, khi này hệ thống sẽ trả về thông báo "Export terminated unsuccessfully" .

16.6.CÔNG CỤ IMPORT

16.6.1. Sử dụng công cụ Import

Công cụ import dùng để đưa dữ liệu từ các export file vào database



Hình vẽ 93. Sử dụng công cụ Import để đưa dữ liệu vào database

Khi thực hiện Import dữ liệu, Users cần được được gán role IMP_FULL_DATABASE. Công cụ import có thể được thực hiện ở các chế độ sau:

Từ khoá	Diễn giải
TABLES	Chế độ này cho phép thực hiện import các tables và partitions vào database.
FROMUSER	Với chế độ này, user có thể import tất cả các objects do user đó sở hữu như tables, grants, indexes, và procedures.
FULL	Chỉ có các users được gán IMP_FULL_DATABASE role mới có thể thực hiện import ở chế độ này. Khi này, user có thể import toàn bộ database
TRANSPORT_TABLESPACES	Cho phép user có quyền tương ứng có thể di chuyển và import tập hợp các tablespace từ database này sang database khác

Chế độ dòng lệnh

Cú pháp:

```
$imp [keyword=]{value|(value, value ...)}
[ [ [,] keyword=]{value|(value, value ...)} ] ...
```

Với:

```
keyword          là từ khoá sử dụng
value            là giá trị được gán cho từ khoá
```

Các tham số dòng lệnh

Từ khoá	Mặc định	Diễn giải
ANALYZE	Y	Cho phép thực hiện lệnh ANALYZE đối với database trước khi import dữ liệu
BUFFER	Tuỳ theo hệ điều hành	Kích thước của bộ đệm sử dụng khi import. Kích thước buffer được xác định theo công thức: `buffer_size = rows_in_array * maximum_row_size`
COMMIT	N	Phát lần commit sau mỗi lần đưa dữ liệu vào database trong quá trình import. Với cách này, dữ liệu sẽ được đảm bảo đưa vào database, để đổi lại, thời gian import sẽ bị kéo dài hơn.
CONSTRAINTS	Y	Cho biết có import cả các ràng buộc vào database hay không
DATAFILES		Liệt kê danh sách các datafile sẽ được đưa vào

		database.
FEEDBACK	0	Hiển thị mức độ import, số dòng dữ liệu được import, tại mỗi lần. Ví dụ: FEEDBACK=10, Công cụ import sẽ hiển thị tiến trình cho biết mỗi lần thực hiện sẽ đưa được 10 dòng dữ liệu vào database
FILE	expdat.dmp	Tên file dữ liệu vào. Tệp dữ liệu có phần mở rộng là .dmp
FROMUSER		Import các objects vàocác schemas thuộc FROMUSER
FULL	N	Giá trị là Y, import toàn bộ database. Để thực hiện được việc này, user cần được cấp quyền IMP_FULL_DATABASE
GRANTS	Y	Giá trị là Y, import các đối tượng trong database cùng với cả các quyền đang được áp dụng trên đối tượng đó
HELP	N	Giá trị là Y, hiển thị danh sách tham số và ý nghĩa tương ứng của chúng
IGNORE	N	Có bỏ qua các lỗi xảy ra trong quá trình import để tiếp tục import dữ liệu mới hay không.
INCTYPE		Các chế độ incremental Import
INDEXES	Y	Giá trị là Y, import cả các index.
LOG		Tên file lưu trữ các thông báo khi import.
PARFILE		Tên file chứa các tham số export
RECORDLENGTH	Tuỳ theo hệ điều hành	Kích thước tính theo bytes của 01 bản ghi được import
ROWS	Y	Cho biết có import từng dòng dữ liệu trong bảng hay không.
SKIP_UNUSABLE_INDEXES	N	Cho biết có bỏ qua việc tạo các index và đặt nó về trạng thái Unusable trong quá trình Import hay không.
TABLES		Liệt kê danh sách các bảng import. Ví dụ: `imp system/manager TABLES=(jones.accts, scott.emp,scott.dept)`
TRANSPORT_TABLESPACE	N	Có import các transportable tablespace hay không
TABLESPACES		Khi tham số TRANSPORT_TABLESPACE = Y, TABLESPACES sẽ dùng để liệt kê danh sách tên các tablespace đã import

TOUSER		Hiển thị danh sách các user sở hữu các objects sẽ được import.
USERID		Username và mật khẩu của user thực hiện export. Chúng được thể hiện theo khuôn dạng: `username/password AS SYSDBA` hay `username/password@instance AS SYSDBA`

Chú ý: Chỉ có một tham số `FULL=Y` hay `OWNER=user` hay `TABLES=schema.table` được chỉ định.

Ví dụ:

```
> imp system/manager parfile=params.dat

Nội dung của file params.dat:
      FILE=blake.dmp
      SHOW=n
      IGNORE=n
      GRANTS=y
      ROWS=y
      FROMUSER=blake
      TOUSER=scott
      TABLES=(unit,manager)
```

Thứ tự của quá trình import

Khi thực hiện import, các objects (đối tượng) sẽ lần lượt được đưa vào database theo thứ tự sau:

1. Các định nghĩa về kiểu dữ liệu (Type definitions)
2. Các định nghĩa các bảng (Table definitions)
3. Dữ liệu trong các bảng (Table data)
4. Các Index tương ứng với từng bảng (Table indexes)
5. Các Integrity constraints, views, procedures và triggers
6. Các đối tượng mở rộng khác như Bitmap, functional và domain indexes

Tính tuần tự này ngăn dữ liệu bị từ chối vì trật tự trên đó bảng được import. Trật tự này còn ngăn các trigger thực hiện hai lần trên cùng một dữ liệu. Tuy nhiên một số đối tượng như thủ tục có thể được kiểm tra khi import bởi vì chúng được import trước khi các đối tượng được tham chiếu. Kiểm tra các đối tượng với `STATUS=INVALID` và compile lại chúng.

Import vào một bảng đã tồn tại

Khi import dữ liệu vào một bảng đã tồn tại, trật tự của import có thể còn tạo ra lỗi tham chiếu. Tình huống tương tự cũng xảy ra khi có constraints trên một bảng mà có tham chiếu tới chính nó.

Vì lý do như đã nói trên đây, cách tốt nhất là disable các tham chiếu bởi constraint khi import dữ liệu vào trong một bảng đã tồn tại. Các constraint có thể được enable lại sau khi import thành công.

Ta cũng có thể phân chia quá trình Import ra làm nhiều lần thay vì 1 lần để tránh việc check constraints mất nhiều thời gian.

Tablespace được sử dụng cho một đối tượng

Nếu một user có đủ quota cần thiết các bảng sẽ được import vào trong cùng một tablespace mà chúng đã được export. Tuy nhiên nếu tablespace không tồn tại hay user không đủ quota trên tablespace, import sẽ tạo bảng trên tablespace mặc định của user. Nếu user không truy xuất được tablespace mặc định, các bảng sẽ không được import.

Một phân đoạn LOB chỉ có thể được import vào cùng một tablespace từ đó chúng đã được export. Một bảng đang chứa LOB sẽ không được tạo nếu như owner của các bảng không thể tạo các đối tượng trên tablespace ở đó các phân đoạn LOB đã được export.

Chú ý

Một bảng không chứa LOB có thể được chuyển từ một tablespace sang một tablespace khác bằng cách sử dụng phương thức import.

Các hướng dẫn trong sử dụng export và import

Sử dụng các tệp tham số để lưu các tham số chung trong danh sách tham số.

- Nếu có nhiều hoạt động update trên các bảng đang được export nên sử dụng tham số CONSISTENT=Y, nói chung nên chạy các quá trình export dữ liệu lớn khi có ít các hoạt động trong bảng được export, nên tạo các rollback segment lớn cho quá trình import.
- Tuỳ chọn COMPRESS=Y sẽ sinh ra đoạn mã để tạo initial extent , initial extent này bằng tổng kích thước của tất cả các extent hiện đang được cấp phát cho một đối tượng. Nếu đối tượng có nhiều hàng được xoá hay extent cuối cùng có nhiều block không được sử dụng , thì không cần thiết phải cấp phát nhiều không gian cho đối tượng đó.
- Cấp phát vùng đệm lớn nếu hệ điều hành và nguồn của hệ thống cho phép.

16.6.2. Chuyển đổi character set

Chuyển đổi character set khi export

Phương pháp conventional export sẽ sử dụng character set chỉ định cho session của user.

Phương pháp direct path export chỉ sử dụng charater set của database, nếu character set của export session không giống như character set của database khi export được khởi tạo, export sẽ hiển thị một thông báo, cần chỉ định character set cho session và khởi động lại quá trình export.

Export file có chứa một cờ (flag) chỉ định character set được sử dụng.

Chuyển đổi character set khi Import

Import session và character set của database đích có thể khác với character set của database nguồn, trong tình huống này cần có sự chuyển đổi của các tập character set. Dữ liệu được chuyển đổi sang character set của user session trong quá trình import sau đó được chuyển thành database character set.

Trong quá trình chuyển đổi, bất cứ character set nào trong export file không tương đương với character set đích đều sẽ được thay thể bởi character set mặc định. Đó là character set được định nghĩa bởi database. Để chuyển đổi 100% thành công thì tập kí tự trong database đích phải là tập cha của tập kí tự trong tệp nguồn.

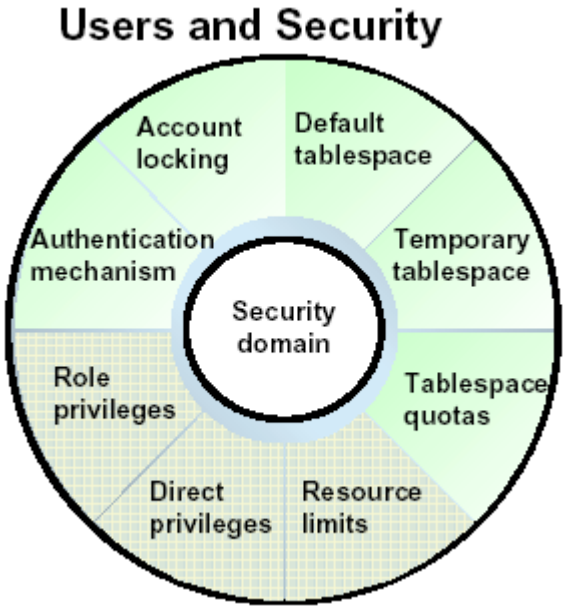
Chương 17. QUẢN LÝ USER

17.1.USER TRONG DATABASE

17.1.1. User và những thành phần liên quan

Security Domain

Quản trị viên database định nghĩa các tên User. Qua đó, cho phép người sử dụng có thể truy nhập vào database thông qua các tên user này. Security domain (bảo mật theo miền) định nghĩa các quyền truy nhập nhất định trên các đối tượng trong database và áp dụng các quyền này cho từng user có trong database.



Hình vẽ 94. Các thành phần bảo mật

Authentication Mechanism (Cơ chế xác nhận)

Mỗi user truy cập vào database đều trải qua bước xác nhận quyền truy nhập. Việc này có thể được thực hiện bởi:

- Database
- Hệ điều hành
- Xác nhận quyền thông qua đường mạng

Tuy nhiên, trong tài liệu này ta chỉ quan tâm tới việc xác nhận bởi database.

Tablespace Quotas (hạn mức tablespace)

Tablespace quotas điều khiển số lượng table space ứng với khả năng lưu trữ vật lý được phép đối với mỗi user trong database.

Default Tablespace (tablespace mặc định)

Là tablespace mặc định chứa các segments do tiến trình của user sử dụng để lưu trữ dữ liệu trong trường hợp User không chỉ rõ tên tablespace ngay khi tạo segment.

Temporary Tablespace (tablespace trung gian)

Temporary tablespace là nơi Oracle server cấp phát các extents phục vụ cho công việc sắp xếp (sort) mỗi khi thực hiện lệnh sắp xếp của User đó.

Account Locking (khoá account)

Các Accounts có thể bị khoá (locked) để ngăn cản việc user thâm nhập vào database. Việc này có thể được thực hiện một cách tự động hoặc do điều khiển của quản trị viên database.

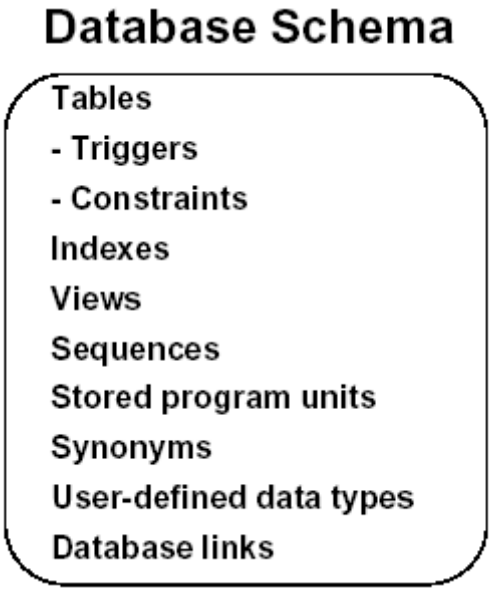
Resource Limits (hạn chế tài nguyên)

Là những giới hạn được đưa ra cho mỗi user về các tài nguyên của hệ thống như: thời gian sử dụng CPU, truy xuất vào ra I/O, số lượng các sessions được mở tối đa,... Những giới hạn về tài nguyên sẽ được bàn kỹ trong chương sau.

17.1.2. Database schema

Schema được xem như một tập hợp các đối tượng như tables, views, clusters, procedures, và packages, cùng có một quan hệ gắn liền với một user nào đó. Mỗi khi user trong database được tạo, một schema tương ứng với user cũng sẽ được tạo lập với cùng tên. Mỗi user chỉ có thể gắn liền với một schema có cùng tên, vì thế *username và schema* nhiều khi có thể dùng lẫn thay cho nhau.

Hình dưới đây sẽ liệt kê các đối tượng trong schema của mỗi users Oracle database.



Hình vẽ 95. Database schema

17.2.QUẢN LÝ USER

17.2.1. Các bước thực hiện khi tạo mới user

1. Lựu chọn username (tên user dùng để truy cập database) và cơ chế xác nhận đối với user này.
2. Chỉ ra các tablespaces cho user dùng để lưu trữ dữ liệu.
3. Phân bổ hạn mức sử dụng trên từng tablespace.
4. Gán các default tablespace và temporary tablespace.
5. Tạo user.
6. Phân quyền truy nhập (privileges - quyền; roles - chức danh) cho user vừa tạo lập.

17.2.2. Tạo mới user với cơ chế xác nhận bởi database

Với cơ chế này, database sẽ sử dụng mật khẩu của user để xác nhận mỗi khi user kết nối tới database. Mỗi mật khẩu của user, sẽ được Oracle server lưu trữ ngay trong data dictionary và nó có thể kiểm tra rất dễ dàng mỗi khi User kết nối tới database.

Cú pháp:

```
CREATE USER user
      IDENTIFIED {BY password | EXTERNALLY}
      [ DEFAULT TABLESPACE tablespace ]
      [ TEMPORARY TABLESPACE tablespace ]
      [ QUOTA {integer [K | M ] | UNLIMITED } ON tablespace
      [ QUOTA {integer [K | M ] | UNLIMITED } ON
      tablespace ] ...]
      [ PASSWORD EXPIRE ]
      [ ACCOUNT { LOCK | UNLOCK }]
      [ PROFILE { profile | DEFAULT }]
```

Với:

user	tên truy nhập của user
BY *password*	*xác* định cơ chế xác nhận user bởi database với mật khẩu truy nhập là *password*
EXTERNALLY	*xác* định cơ chế xác nhận user bởi hệ điều hành
DEFAULT/TEMPORARY TABLESPACE	xác định default/temporary tablespace cho user
QUOTA	xác định lượng không gian tối đa cấp phát cho user để lưu trữ các đối tượng trong từng tablespace tương ứng. Từ khoá UNLIMITED cho biết không hạn định số lượng không gian cấp phát.
PASSWORD EXPIRE	bắt buộc user phải chỉ rõ mật khẩu mỗi khi user thực hiện kết nối tới database thông qua SQL*PLUS (Phương thức này chỉ có tác dụng khi user sử dụng cơ chế xác nhận bởi database)
ACCOUNT LOCK/ UNLOCK	sử dụng tuỳ chọn này để lock/unlock đỗi với mỗi user một cách tường minh (mặc định là UNLOCK).
PROFILE	dùng để điều khiển tài nguyên của user.

Lưu ý:

Khi thiết lập tuỳ chọn PASSWORD EXPIRE trong lệnh tạo user, khi user sử dụng SQL*PLUS để kết nối tới database, mỗi lần kết nối user lại phải nạp mới mật khẩu. Việc truy cập thông thường sẽ nhận được thông báo:

```
ERROR:
ORA-28001: the account has expired
Changing password for PETER
Old password:
New password:
Retype new password:
Password changed
```

Trong OEM ta có thể thực hiện theo các bước sau

1. Sử dụng Oracle Security Manager.
2. Chọn User—>Create.
3. Nhập vào thông tin của trong phần General page.
4. Chỉ rõ hạn mức sử dụng trong phần Quotas.
5. Bấm nút Create.

17.2.3. Thay đổi thuộc tính của user

Ta sử dụng câu lệnh ALTER USER khi thực hiện các thay đổi user như mật khẩu, lock. Sử dụng khi:

- Thay đổi mật khẩu khi user quên mật khẩu.
- Lock/Unlock đối với các account của user.
- Thay đổi mật khẩu theo từng phiên làm việc.

Cú pháp:

```
ALTER USER user
      [ IDENTIFIED {BY password | EXTERNALLY }]
      [ PASSWORD EXPIRE]
      [ ACCOUNT {LOCK | UNLOCK }] ;
```

Ví dụ:

```
ALTER USER peter
      IDENTIFIED BY hisgrandpa
      PASSWORD EXPIRE;
```

Lưu ý: khi user đã bị lock mà vẫn cố gắng kết nối tới database. Oracle server sẽ phát sinh lỗi

```
ERROR:
ORA-28000: the account is locked
Warning: You are no longer connected to ORACLE.
```

Trong OEM ta có thể thực hiện theo các bước sau

1. Chạy Oracle Security Manager.
2. Chuyển tới nút Users.
3. Chọn username tương ứng.

4. Chọn User—>Change Account Status.
5. Thực hiện thay đổi các trạng thái Unlock, Lock, or Expire.
6. Nhập vào các thông tin trong phần General page.
7. Bấm nút Apply.

17.2.4. Thay đổi hạn mức (quota) sử dụng tablespace

Trong một số trường hợp, ta có thể thay đổi hạn mức sử dụng tablespace khi:

- Các tables của user đó không thể mở rộng để lưu trữ thêm được nữa
- Các ứng dụng được cải tiến đòi hỏi bổ sung thêm các tables hay indexes.
- Các đối tượng được tổ chức lại và được đặt trên nhiều tablespaces khác nhau.

Cú pháp:

```
ALTER USER user
      [ DEFAULT TABLESPACE tablespace]
      [ TEMPORARY TABLESPACE tablespace]
      [ QUOTA {integer [K | M] | UNLIMITED } ON tablespace
      [ QUOTA {integer [K | M] | UNLIMITED } ON tablespace ] ...
      ]
```

Ví dụ:

```
ALTER USER peter
      QUOTA 0 ON data01;
```

Trong OEM, ta thực hiện theo các bước sau

1. Chạy Oracle Security Manager.
2. Chuyển tới nút Users.
3. Chọn username tương ứng.
4. Nhập vào các thông tin thích hợp trong phần Quotas.
5. Bấm nút Apply.

17.2.5. Huỷ User

Huỷ bỏ user khỏi database

Cú pháp:

```
DROP USER user [CASCADE]
```

Ví dụ:

```
DROP USER peter;

Hoặc
DROP USER peter CASCADE;
```

Lưu ý:

- CASCADE sẽ huỷ tất cả các đối tượng trong schema trước khi xoá User. Nó cần được chỉ rõ khi schema chứa nhiều đối tượng.
- Ta không thể huỷ được các user hiện đang kết nối tới Oracle server.

17.3.THÔNG TIN VỀ USER

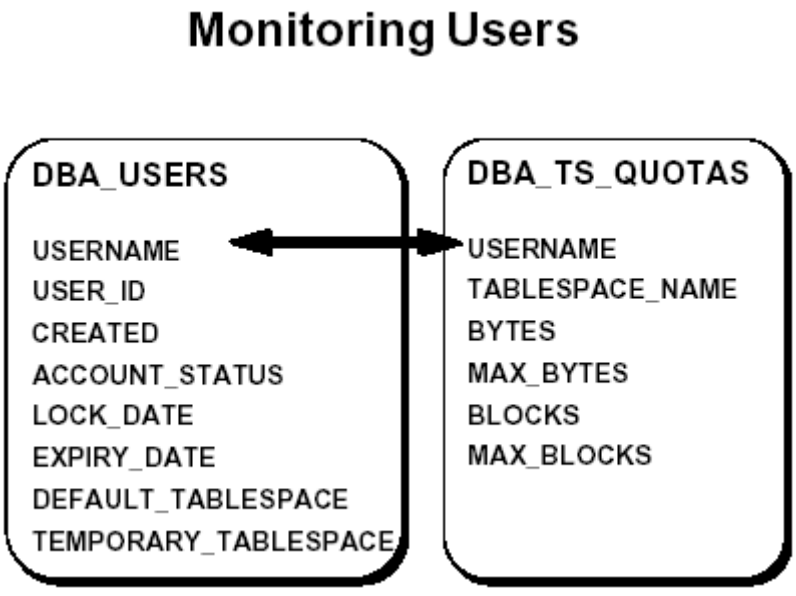
Ta có thể lấy các thông tin liên quan tới user trong data dictionary DBA_USERS và DBA_TS_QUOTAS.

Với mỗi user, ta có thể xác định được các thông tin về hạn mức:

Ví dụ:

```
SVRMGR> SELECT tablespace_name, blocks, max_blocks,bytes,
max_bytes
2> FROM dba_ts_quotas
3> WHERE username = 'SCOTT';

TABLESPACE_NAME  BLOCKS    MAX_BLOCKS BYTES     MAX_BYTES
---------------  -------   ---------- -------   -----------
DATA01           10        -1         20480     -1
1 row selected.
```

Giá trị -1 trong cột MAX_BLOCKS và MAX_BYTES cho biết cột này chưa được gán giá trị tường minh.



Hình vẽ 96. Thông tin về User trong data dictionary

Hoặc ta cũng có thể lấy các thông tin về Account của user

Ví dụ:

```
SVRMGR> SELECT username, account_status, temporary_tablespace
2> FROM dba_users;

USERNAME          ACCOUNT_STATUS      TEMPORARY_TABLESPACE
```

```
--------------  ----------------  --------------------
SYS             OPEN              TEMP
SYSTEM          OPEN              TEMP
DBSNMP          OPEN              TEMP
SCOTT           OPEN              TEMP
4 rows selected.
```

Chương 18. QUẢN LÝ THÔNG TIN PROFILES

18.1.GIỚI THIỆU PROFILE

Thông tin profile là tập hợp các thông tin về tài nguyên hệ thống và thông tin liên quan tới mật khẩu của user bao gồm:

- Thời gian sử dụng CPU (CPU time)
- Các thao tác vào ra (I/O operations)
- Thời gian nghỉ (Idle time)
- Thời gian kết nối (Connect time)
- Dung lương bộ nhớ (Memory space)
- Các phiên làm việc đồng thời (Concurrent sessions)
- Số lần sử dụng và thời gian sử dụng một mật khẩu (Password aging and expiration)
- Lịch sử thay đổi mật khẩu (Password history)
- Cơ chế xác nhận mật khẩu (Password complexity verification)
- Khoá account truy nhập (Account locking)

Quản trị viên database gán các profile cho từng user mà theo đó Oracle server phân bổ tài nguyên cho user theo thông tin có trong profile.

Oracle server tự động tạo default profile (profile mặc định) mỗi khi tạo database. Ban đầu, các thông tin trong default profile được đặt không hạn chế. Quản trị viên database có thể điều chỉnh lại các tham số này đối với từng user.

Sử dụng Profile

- Hạn chế users thực hiện các thao tác đòi hỏi sử dụng nhiều tài nguyên hệ thống.
- Đảm bảo cắt kết nối của users với database mỗi khi session của user đó không hoạt động nữa.
- Cho phép các users như nhau trong một nhóm sử dụng tài nguyên như nhau.
- Quản lý việc sử dụng tài nguyên dạng dữ liệu lớn và phức tạp trong hệ thống database có nhiều người dùng.
- Điều chỉnh việc sử dụng mật khẩu của user.

Sử dụng profile để quản lý tài nguyên

Ta có thể quản lý tài nguyên hệ thống thông qua việc sử dụng tài nguyên. Để làm được điều đó, ta thực hiện theo các bước sau:

1. Tạo một profile bằng lệnh `CREATE PROFILE` qua đó xác định các giới hạn tài nguyên và giới hạn mật khẩu.
2. Gán profile đó cho user thông qua lệnh `CREATE USER` hay `ALTER USER`.

3. Thực hiện giới hạn các tài nguyên sử dụng bằng cách dùng lệnh ALTER SYSTEM hay điều chỉnh lại các thông số có trong file tham số khởi tạo. Cách này đòi hỏi phải dừng rồi khởi động lại database.

18.2.QUẢN LÝ PROFILE

18.2.1. Tạo Profile

Tạo profile lưu trữ các giới hạn tài nguyên sử dụng thông qua câu lệnh:

```
CREATE PROFILE profile LIMIT
      [SESSIONS_PER_USER max_value]
      [CPU_PER_SESSION max_value]
      [CPU_PER_CALL max_value]
      [CONNECT_TIME max_value]
      [IDLE_TIME max_value]
      [LOGICAL_READS_PER_SESSION max_value]
      [LOGICAL_READS_PER_CALL max_value]
      [COMPOSITE_LIMIT max_value]
      [PRIVATE_SGA max_bytes]

max_value :== {integer|UNLIMITED|DEFAULT}
max_bytes :== {integer[K|M]|UNLIMITED|DEFAULT}
```

Với:

profile	là tên của profile
UNLIMITED	xác định user được gán profile tương ứng có thể sử dụng không hạn chế tài nguyên
DEFAULT	mục profile để hạn chế sử dụng tài nguyên được chỉ ra trong mệnh đề tương ứng thuộc DEFAULT profile
COMPOSITE_LIMIT	giới hạn sử dụng tài nguyên tổng cộng bao gồm cả các trọng số tương ứng. Giới hạn được tính bằng tổng số: CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION, PRIVATE_SGA

Lưu ý:

View RESOURCE_COST trong data dictionary cho biết các trọng số tương ứng được gán cho các tài nguyên khác nhau. Ta cũng có thể sử dụng lệnh ALTER RESOURCE COST để biết được các trọng số cho từng tài nguyên khác nhau.

Ví dụ:

```
CREATE PROFILE developer_prof LIMIT
      SESSIONS_PER_USER 2
      CPU_PER_SESSION 10000
      IDLE_TIME 60
```

```
CONNECT_TIME 480;
```

Trong OEM, ta thực hiện các bước sau

1. Chạy Security Manager.
2. Chọn Profile—>Create.
3. Trong phần General page, nhập vào tên và các thông tin chi tiết ứng với từng mục tài nguyên.
4. Bấm nút Apply.

18.2.2. Thiết lập các giới hạn về tài nguyên

Giới hạn về tài nguyên cho từng session level

Tài nguyên	Diễn giải
CPU_PER_SESSION	Tổng lượng thời gian CPU, được xác định theo đơn vị trăm giây.
SESSIONS_PER_USER	Số lượng tối đa các session có thể sử dụng đồng thời bởi cùng một user
CONNECT_TIME	Thời gian kết nối tối đa, tính theo đơn vị phút
IDLE_TIME	Thời gian trễ, tính theo đơn vị phút
LOGICAL_READS_PER_SESSION	Số lượng block dữ liệu được đọc
PRIVATE_SGA	Vùng không gian giành riêng trong SGA, tính theo đơn vị byte

Giới hạn về tài nguyên cho từng Call level

Tài nguyên	Diễn giải
CPU_PER_CALL	Thời gian sử dụng CPU cho mỗi lần gọi, tính theo đơn vị trăm giây.
LOGICAL_READS_PER_CALL	Số lượng block được đọc tối đa

Ghi chú:

- IDLE_TIME được áp dụng chỉ cho các tiến trình server. Giới hạn IDLE_TIME không ảnh hưởng gì đối với các câu lệnh truy vấn dài hay các thao tác khác.
- LOGICAL_READS_PER_SESSION là giới hạn các lần đọc block dữ liệu từ cả bộ nhớ lẫn ổ đĩa.
- PRIVATE_SGA áp dụng khi chạy multithreaded server (MTS).

18.2.3. Gán Profile cho User

Việc gán profile có thể được thực hiện ngay trong lện CREATE USER hay lệnh ALTER USER.

Ví dụ: sử dụng lệnh CREATE USER để tạo một user USER3 với mật khẩu là USER3, sau đó gán profile có tên là DEVELOPER_PROF cho user vừa tạo.

```
CREATE USER user3 IDENTIFIED BY user3
      DEFAULT TABLESPACE data01
      TEMPORARY TABLESPACE temp
      QUOTA unlimited ON data01
      PROFILE developer_prof;
```

Điều chỉnh user SCOTT và gán profile DEVELOPER_PROF cho User này.

```
ALTER USER scott
      PROFILE developer_prof;
```

Trong OEM, ta có thể làm theo các bước sau

1. Chạy Security Manager.
2. Chọn Profile—>Assign Profile to users.
3. Trong phần Assign Profile page, chọn user tương ứng.
4. Bấm nút OK.

Một số tính chất của Profile

- Profile được gán không ảnh hưởng tới sessions hiện thời.
- Profiles chỉ có thể được gán cho users không được gán cho roles hay cho các profiles khác.
- Trong trường hợp ta không gán profile cho user ngay từ khi mới tạo user, profile mặc định (default profile) sẽ được tự động gán cho user đó.

18.2.4. Đặt giới hạn tài nguyên

Ta có thể đặt giới hạn sử dụng tài nguyên thông qua tham số khởi tạo RESOURCE_LIMIT hay sử dụng câu lện ALTER SYSTEM.

Sử dụng tham số khởi tạo RESOURCE_LIMIT

- Cho phép hoặc không cho phép áp dụng các giới hạn tài nguyên. Với cách này, ta cần khởi động lại instance khi thay đổi các giá trị trong file tham số khởi tạo.
- Giá trị của tham số là TRUE nếu cho phép giới hạn tài nguyên. Ngược lại, giá trị sẽ là FALSE (đây là giá trị mặc định).

Sử dụng lệnh ALTER SYSTEM

- Cho phép hoặc không cho phép áp dụng các giới hạn tài nguyên.
- Các điều chỉnh do lệnh `ALTER SYSTEM` sẽ có tác dụng ngay cho tới khi có một điều chỉnh khác hoặc khi database bị tắt.
- Ta sử dụng lệnh này trong trường hợp muốn hạn chế sử dụng tài nguyên mà lại không được tắt database.

18.2.5. Thay đổi thông tin trong profile

Ta có thể thay đổi các thông tin trong profile thông qua câu lệnh `ALTER PROFILE`.

Cú pháp:

```
ALTER PROFILE profile LIMIT
      [SESSIONS_PER_USER max_value]
      [CPU_PER_SESSION max_value]
      [CPU_PER_CALL max_value]
      [CONNECT_TIME max_value]
      [IDLE_TIME max_value]
      [LOGICAL_READS_PER_SESSION max_value]
      [LOGICAL_READS_PER_CALL max_value]
      [COMPOSITE_LIMIT max_value]
      [PRIVATE_SGA max_bytes]
```

Ví dụ:

```
ALTER PROFILE default LIMIT
      SESSIONS_PER_USER 5
      CPU_PER_CALL 3600
      IDLE_TIME 30;
```

Trong OEM, ta thực hiện theo các bước sau

1. Chạy Security Manager.
2. Chuyển tới nút Profile.
3. Chọn profile tương ứng.
4. Trong phần General page thay đổi các thông số trong từng mục.
5. Bấm nút Apply.

18.2.6. Huỷ profile

Thực hiện việc huỷ profile theo lệnh `DROP PROFILE`.

Cú pháp:

```
DROP PROFILE profile [CASCADE]
```

Với:

```
profile          tên của profile bị huỷ
CASCADE          huỷ tất cả các profile đã được gán cho user.
```

Ví dụ:

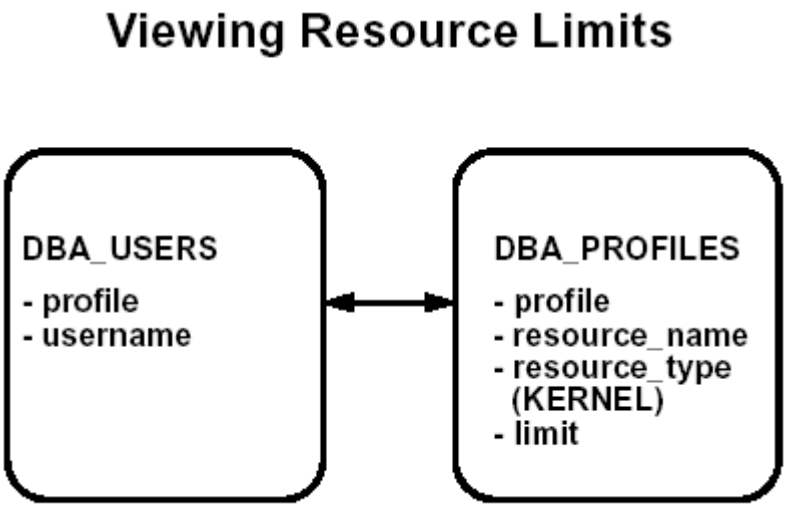
```
DROP PROFILE developer_prof;

Hoặc
DROP PROFILE developer_prof CASCADE;
```

Trong OEM, ta thực hiện theo các bước sau

1. Chạy Security Manager.
2. Chuyển tới nút Profile.
3. Chọn profile tương ứng.
4. Chọn Profile—>Remove.
5. Bấm nút OK.

18.2.7. Thông tin về các giới hạn tài nguyên

Để xem thông tin về các giới hạn sử dụng tài nguyên, ta xem xét trong các data dictionary DBA_USERS và DBA_PROFILE



Hình vẽ 97. Thông tin về giới hạn tài nguyên

Ví dụ: xem giới hạn tài nguyên của User SCOTT

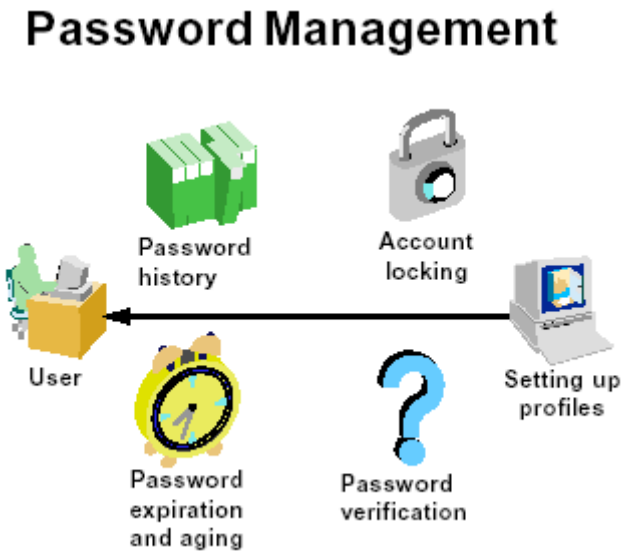
```
SVRMGR1> SELECT p.profile, p.resource_name, p.limit
2> FROM dba_users u, dba_profiles p
3> WHERE p.profile=u.profile AND username='SCOTT' AND
4> p.resource_type='KERNEL';

PROFILE          RESOURCE_NAME                LIMIT
---------------  ---------------------------  -------
DEVELOPER_PROF   COMPOSITE_LIMIT              DEFAULT
DEVELOPER_PROF   SESSIONS_PER_USER            2
```

```
DEVELOPER_PROF   CPU_PER_SESSION              10000
DEVELOPER_PROF   CPU_PER_CALL                 DEFAULT
DEVELOPER_PROF   LOGICAL_READS_PER_SESSION    DEFAULT
DEVELOPER_PROF   LOGICAL_READS_PER_CALL       DEFAULT
DEVELOPER_PROF   IDLE_TIME                    60
DEVELOPER_PROF   CONNECT_TIME                 480
DEVELOPER_PROF   PRIVATE_SGA                  DEFAULT
9 rows selected.
```

18.3.QUẢN LÝ MẬT KHẨU

Để có thể đảm bảo việc bảo mật cho toàn bộ database, hệ thống quản lý mật khẩu Oracle cho phép quản trị viên database quản lý mật khẩu thông qua các profile. Việc quản lý mật khẩu này có một số đặc điểm sau:

- Account locking: cho phép tự động khoá account tương ứng với user mỗi khi user đó không thể kết nối tới hệ thống.
- Gắn thời hạn sử dụng cho mỗi mật khẩu: mỗi mật khẩu sẽ có một thời hạn sử dụng nhất định, quá thời hạn trên, mật khẩu cần được thay đổi.
- Lưu trữ lịch sử thay đổi mật khẩu (Password history): kiểm tra các mật khẩu mới để đảm bảo mật khẩu mới không trùng với các mật khẩu trước đó.
- Cơ chế xác nhận mật khẩu mềm dẻo: cho phép kiểm tra và xác nhận mật khẩu một cách mềm dẻo nhưng vẫn đảm bảo tính chặt chẽ chống lại được những hình thức cố tình xâm nhập vào hệ thống bằng cách phá khoá, đoán mật khẩu.



Hình vẽ 98. Quản lý mật khẩu

Cũng tương tự như việc đưa ra các giới hạn về tài nguyên sử dụng, ta cũng có thể sử dụng các profile để thiết lập các giới hạn về mật khẩu và gán cho mỗi user thông qua lệnh CREATE hay ALTER USER.

Thực hiện quản lý mật khẩu, ta có thể sử dụng các câu lệnh CREATE USER hay ALTER USER để lock hay unlock account của user đó.

18.3.1. Tạo profile quản lý mật khẩu

Tạo profile quản lý mật khẩu thông qua lệnh `CREATE PROFILE`

Cú pháp:

```
CREATE PROFILE profile LIMIT
      [FAILED_LOGIN_ATTEMPTS max_value]
      [PASSWORD_LIFE_TIME max_value]
      [ {PASSWORD_REUSE_TIME
      |PASSWORD_REUSE_MAX} max_value]
      [ACCOUNT_LOCK_TIME max_value]
      [PASSWORD_GRACE_TIME max_value]
      [PASSWORD_VERIFY_FUNCTION
      {function|NULL|DEFAULT} ]
```

Ví dụ:

```
CREATE PROFILE grace_5 LIMIT
      FAILED_LOGIN_ATTEMPTS 3
      PASSWORD_LIFE_TIME 30
      PASSWORD_REUSE_TIME 30
      PASSWORD_VERIFY_FUNCTION verify_function
      PASSWORD_GRACE_TIME 5;
```

Trong OEM, ta thực hiện theo các bước sau:

1. Chạy Security Manager.
2. Chọn Profile—>Create.
3. Trong phần Password Property sheet nhập vào các tham số mật khẩu của account.
4. Bấm nút Apply.

18.3.2. Các tham số điều chỉnh mật khẩu

Tài nguyên	Diễn giải
FAILED_LOGIN_ATTEMPTS	Số lần kết nối hỏng (do nhập sai tên hay mật khẩu) tối đa
PASSWORD_LOCK_TIME	Số ngày lock account của user kể từ khi hết hạn sử dụng mật khẩu
PASSWORD_LIFE_TIME	Số ngày sử dụng mật khẩu mới kết từ khi bắt đầu thay đổi mật khẩu
PASSWORD_GRACE_TIME	Thời gian gia hạn cho việc thay đổi mật khẩu
PASSWORD_REUSE_TIME	Thời gian tối thiểu, tính theo ngày, có thể tái sử dụng mật khẩu cũ
PASSWORD_REUSE_MAX	Số lần tối đa có thể sử dụng lại một mật khẩu.
PASSWORD_VERIFY_FUNCTION	Hàm PL/SQL thực hiện mã hoá và kiểm tra mật khẩu trước khi nó được sử dụng

18.3.3. Một số đặc điểm chính trong quản lý mật khẩu

Account Locking

Oracle server tự động locks (khoá) account của user khi user đó cố tình truy nhập vào account với số lần truy nhập sai đạt đến giá trị FAILED_LOGIN_ATTEMPTS. Sau khoảng thời gian PASSWORD_LOCK_TIME, account đó lại được tự động unlocked hoặc có thể được unlocked bởi quản trị viên database với lệnh ALTER USER.

Ngoài ra, database account cũng có thể được locked một cách tường minh thông qua lệnh ALTER USER. Khi này, account sẽ không được tự động unlocked giống như trường hợp trên.

Thời hạn sử dụng mật khẩu

Tham số PASSWORD_LIFE_TIME xác định thời gian tối đa cho việc sử dụng một mật khẩu, sau thời gian này, mật khẩu cần được thay đổi.

Quản trị viên database có thể chỉ ra thời gian gia hạn (PASSWORD_GRACE_TIME). Thời gian này bắt đầu sau khi mật khẩu truy cập hệ thống hết hạn. Trong thời gian này, mỗi lần truy cập vào hệ thống sẽ hiện lên một thông báo nhắc nhở (warning message) thay đổi mật khẩu. User cần phải thực hiện thời hạn thay đổi mật khẩu trong khoảng thời gian gia hạn, quá thời gian trên account của user sẽ bị locked.

Lịch sử sử dụng mật khẩu (Password history)

Password history thực hiện việc kiểm tra để đảm bảo không cho phép user được tái sử dụng một mật khẩu nào đó trong nhiều lần. Các quy định này được thiết lập trong các tham số:

- PASSWORD_REUSE_TIME tương ứng với thời gian tối thiểu, tính theo ngày, có thể tái sử dụng mật khẩu cũ.
- PASSWORD_REUSE_MAX là số lần tối đa có thể sử dụng lại một mật khẩu.

Khi một tham số được thiết lập giá trị là DEFAULT hay UNLIMITED, các tham số khác sẽ được gán giá trị là UNLIMITED.

18.3.4. Hàm cung cấp mật khẩu cho người sử dụng

Khi đưa hàm cung cấp mật khẩu cho người sử dụng vào trong databse, quản trị viên database cần quan tâm tới một số đặc điểm sau:

- Hàm được khai báo theo định dạng:

```
function_name(
userid_parameter IN VARCHAR2(30),
password_parameter IN VARCHAR2(30),
old_password_parameter IN VARCHAR2(30))
RETURN BOOLEAN
```

- Giá trị trả về của hàm là TRUE nếu thành công hay FALSE trong trường hợp gặp lỗi.
- Khi hàm mật khẩu phát sinh lỗi (raise an exception), lỗi đó sẽ làm ngừng thực hiện lệnh ALTER USER hay CREATE USER.

- Hàm mật khẩu bắt buộc phải khai báo trong schema của user SYS.

Oracle cung cấp sẵn hàm xác nhận mật khẩu, hàm này được gọi là VERIFY_FUNCTION đặt trong file script *utlpwdmg.sql*, thuộc thư mục %ORACLE_HOME%\RDBMS\ADMIN.

18.3.5. Thông tin về mật khẩu

Ta có thể kết hợp các thông tin về mật khẩu có trong các data dictionary DBA_USERS và DBA_PROFILES.

- DBA_USERS: profile, username, account_status, lock_date, expiry_date
- DBA_PROFILES: profile, resource_name, resource_type (PASSWORD), limit

Ví dụ:

Xem thông tin về thời hạn và trạng thái lock của mật khẩu.

```
SVRMGR> SELECT username, password, account_status,
2>lock_date, expiry_date
3> FROM dba_users;
USERNAME   PASSWORD         ACCOUNT_STATUS LOCK_DATE EXPIRY_DA
---------- ---------------- -------------- --------- ---------
SYS        D4C5016086B2DC6  OPEN                     19-DEC-97
SYSTEM     D4DF7931AB130E3  OPEN                     19-DEC-97
TEST       7A0F2B316C212D6  OPEN                     31-JAN-98
SCOTT      F894844C34402B6  OPEN                     19-DEC-97
DBSNMP     E066D214D5421CC  OPEN                     19-DEC-97
USER3      94152F9F5B35B10  OPEN                     12-FEB-98
6 rows selected.
```

Xem thông tin profile của mật khẩu

```
SVRMGR> SELECT * FROM dba_profiles
2>WHERE resource_type='PASSWORD';
PROFILE           RESOURCE_NAME              LIMIT
---------------   ------------------------   ---------------
DEFAULT           FAILED_LOGIN_ATTEMPTS      3
DEVELOPER_PROF    FAILED_LOGIN_ATTEMPTS      DEFAULT
DEFAULT           PASSWORD_LIFE_TIME         60
DEVELOPER_PROF    PASSWORD_LIFE_TIME         DEFAULT
DEFAULT           PASSWORD_REUSE_TIME        1800
DEVELOPER_PROF    PASSWORD_REUSE_IME         DEFAULT
DEFAULT           PASSWORD_REUSE_MAX         UNLIMITED
DEVELOPER_PROF    PASSWORD_REUSE_MAX         DEFAULT
DEFAULT           PASSWORD_VERIFY_FUNCTION   VERIFY_FUNCTION
DEVELOPER_PROF    PASSWORD_VERIFY_FUNCTION   DEFAULT
DEFAULT           PASSWORD_LOCK_TIME         .0006
DEVELOPER_PROF    PASSWORD_LOCK_TIME         DEFAULT
DEFAULT           PASSWORD_GRACE_TIME        10
DEVELOPER_PROF    PASSWORD_GRACE_TIME        DEFAULT
14 rows selected.
```

Chương 19. CÁC QUYỀN HỆ THỐNG

19.1.PHÂN LOẠI QUYỀN

Oracle database có khoảng 80 quyền hệ thống và con số này đang tiếp tục tăng lên.

Các quyền hệ thống có thể chia ra như sau:

- Các quyền cho phép thực hiện các thao tác mức độ rộng trên hệ thống ví dụ như: CREATE SESSION, CREATE TABLESPACE.
- Các quyền cho phép quản lý các đối tượng thuộc về một user ví dụ: CREATE TABLE
- Các quyền cho phép quản lý các đối tượng trong bất cứ một schema nào ví dụ câu lệnh: CREATE ANY TABLE.

Có thể điều khiển các quyền bằng cách câu lệnh GRAND hay REVOKE.

Chú ý:

Các users có quyền ANY đều có thể truy xuất các bảng thuộc data dictionary ngoại trừ các tiền tố USER_ALL và bất cứ views nào trên đó các quyền được gán cho PUBLIC.

19.1.1. Các quyền hệ thống

Phân loại	Ví dụ
INDEX	CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX
TABLE	CREATE TABLE CREATE ANY TABLE ALTER ANY TABLE DROP ANY TABLE SELECT ANY TABLE UPDATE ANY TABLE DELETE ANY TABLE
SESSION	CREATE SESSION ALTER SESSION RESTRICTED SESSION
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE UNLIMITED TABLESPACE

- Không có quyền CREATE INDEX
- Các quyền như CREATE TABLE, CREATE PROCEDURE, CREATE CLUSTER bao gồm cả các quyền xoá các đối tượng đó.
- CREATE TABLE bao gồm các quyền CREATE INDEX và ANALYZE, Các user cần có đủ quota trên tablespace hay phải được gán UNLIMITED TABLESPACE .
- Để truncate các bảng thì quyền DROP ANY TABLE được sử dụng.

19.1.2. Gán các quyền hệ thống

Sử dụng cú pháp sau đây để gán quyền hệ thống cho user

Cú pháp:

```
GRANT {system_priv|role}
      [, {system_priv|role} ]...
      TO {user|role|PUBLIC}
      [, {user|role|PUBLIC} ]...
      [WITH ADMIN OPTION]
```

Với:

```
system_priv      chỉ định quyền hệ thống sử dụng
role             chỉ định tên chức danh được gán
PUBLIC           gán quyền hệ thống cho tất cả các user
WITH ADMIN OPTION
                 cho phép user được gán quyền có thể gán
                 quyền hay chức danh đó cho user khác.
```

Ví dụ:

```
GRANT CREATE SESSION, CREATE TABLE
      TO user1;
Hoặc
GRANT CREATE SESSION TO scott
      WITH ADMIN OPTION;
```

Một số hướng dẫn

- Để có quyền hệ thống, user cần được gán quyền có kèm thêm tuỳ chọn `WITH ADMIN OPTION`.
- Người được gán với tuỳ chọn `WITH ADMIN OPTION` có thể tiếp tục gán cho một user khác quyền hệ thống với `WITH ADMIN OPTION`.
- Bất cứ một user nào có quyền `GRANT ANY ROLE` có thể gán bất kì quyền nào trong database.
- Người được gán với tuỳ chọn `WITH ADMIN OPTION` có thể gán quyền hay lấy lại các quyền từ bất cứ user hay role nào trong database.

19.1.3. Xác nhận user bằng password file

Trong chương quản lý database, ta đã biết đến hai quyền hệ thống là SYSDBA và SYSOPER. Các quyền này được sử dụng chỉ định rằng việc xác lập user thông qua password file.

Chỉ có người quản trị hệ thống mới có thể có khả năng connect vào database với quyền quản trị. Kết nối với quyền SYSDBA sẽ cho phép user có quyền thực thi không giới hạn đối với các hoạt động trên database hay đối tượng trong database.

Bảng sau đây phân biệt giữa hai loại quyền SYSDBA và SYSOPER

Phân loại	Ví dụ
SYSOPER	STARTUP SHUTDOWN ALTER DATABASE OPEN | MOUNT ALTER DATABASE BACKUP CONTROLFILE ALTER TABLESPACE BEGIN/END BACKUP RECOVER DATABASE, ALTER DATABASE ARCHIVELOG RESTRICTED SESSION
SYSDBA	SYSOPER privileges WITH ADMIN OPTION CREATE DATABASE RECOVER DATABASE UNTIL

Sau khi tạo password file bằng công cụ password và khởi tạo tham số trong initialization file:

```
REMOTE_LOGIN_PASSWORD_FILE=EXCLUSIVE
```

Người quản trị hệ thống có thể thêm các user vào trong password file bằng cách gán các quyền SYSDBA và SYSOPER cho user.

Tuỳ chọn WITH ADMIN OPTION không được sử dụng cho việc gán các quyền này. Chỉ có user hiện đang được nối đến database với quyền SYSDBA mới có thể gán và lấy lại SYSDBA hay SYSOPER cho các user khác. Các quyền này không thể được gán cho một role bởi vì các role không tồn tại khi database startup.

View V$PWFILE_USERS chứa thông tin về các user được gán quyền SYSDBA và SYSOPER.

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

USERNAME              SYSDB      SYSOP
------------------    ---------  -----------
INTERNAL              TRUE       TRUE
SYS                   TRUE       TRUE
2 rows selected.
```

19.1.4. Thông tin về các quyền

Thông tin về các quyền được lấy từ các view data dictionary: DBA_SYS_PRIVS và SESSION_PRIVS.

Các thông tin bao gồm:

- DBA_SYS_PRIVS: GRANTEE, PRIVILEGE, ADMIN OPTION
- SESSION_PRIVS: PRIVILEGE

Ví dụ:

Liệt kê các quyền hệ thống được gán cho user và role:

```
SVRMGR>SELECT * FROM DBA_SYS_PRIVS;

GRANTEE      PRIVILEGE                ADM
```

```
----------   ---------------        -----
...
SCOTT        SELECT ANY TABLE       NO
SYS          DELETE ANY TABLE       NO
SYS          EXECUTE ANY TYPE       NO
SYS          INSERT ANY TABLE       NO
SYS          SELECT ANY SEQUENCE    NO
SYS          SELECT ANY TABLE       YES
SYS          UPDATE ANY TABLE       NO
SYSTEM       UNLIMITED TABLESPAC    YES
...
```

View SESSION_PRIVS liệt kê các quyền có sẵn cho session hiện tại cho một user.

```
SVRMGR> SELECT * FROM session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
ALTER SESSION
CREATE TABLE
SELECT ANY TABLE
CREATE CLUSTER
CREATE SYNONYM
CREATE VIEW
CREATE SEQUENCE
CREATE DATABASE LINK
CREATE PROCEDURE
CREATE TRIGGER
CREATE TYPE
12 rows selected.
```

19.2.QUẢN LÝ QUYỀN

19.2.1. Thu hồi các quyền hệ thống

Sử dụng cú pháp sau đây để lấy lại các quyền hệ thống.

Cú pháp:

```
REVOKE {system_priv|role}
      [, {system_priv|role} ]...
      FROM {user|role|PUBLIC}
      [, {user|role|PUBLIC} ]...
```

Ví dụ:

```
REVOKE CREATE TABLE FROM user1;

Hoặc:
REVOKE CREATE SESSION FROM scott;
```

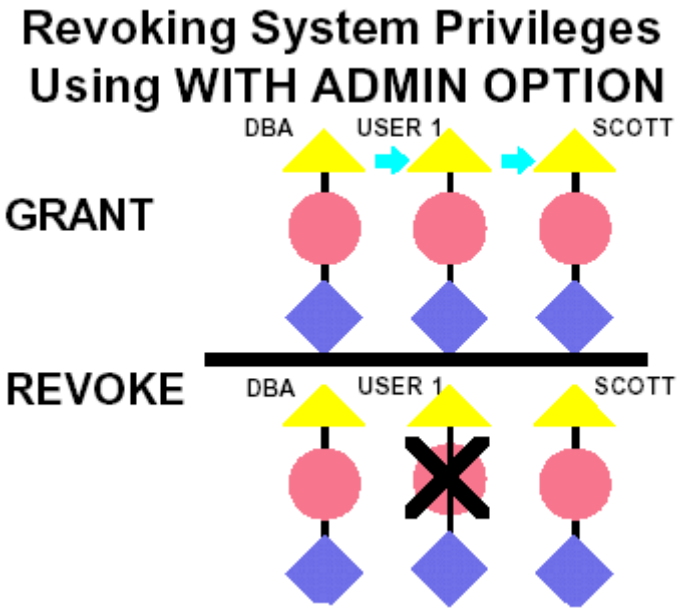
Chú ý:

- Câu lệnh REVOKE chỉ có thể lấy lại quyền của mà đã được gán trực tiếp bằng câu lệnh GRANT.

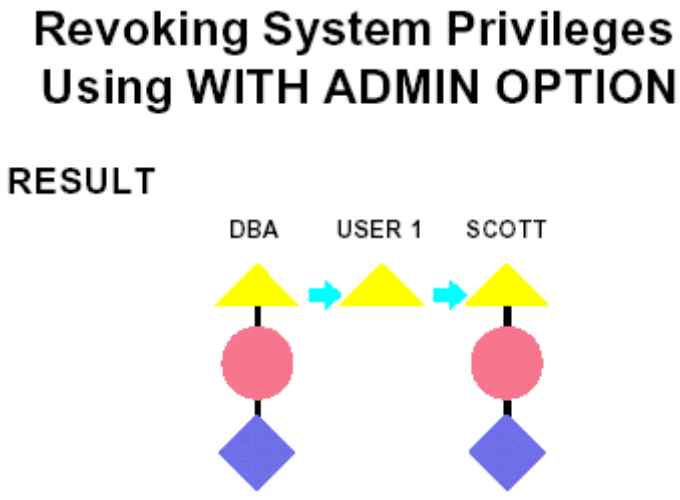
- Thu hồi các quyền hệ thống có thể ảnh hưởng đến một số các đối tượng phụ thuộc. Ví dụ: nếu quyền `SELECT ANY TABLE` được gán cho user và user đó được gán các thủ tục hay view mà sử dụng các bảng thuộc về các user khác thì việc lấy lại các quyền sẽ làm cho các thủ tục hay view đó trở nên không hợp lệ.

Thu hồi các quyền sử dụng có WITH ADMIN OPTION

Hai hình dưới đây sử minh hoạ việc thu hồi các quyền hệ thống sử dụng `WITH ADMIN OPTION`.



Hình vẽ 99. Thu hồi quyền – trước khi thu hồi



Hình vẽ 100. Thu hồi quyền – sau khi thu hồi

Không có sự ảnh hưởng lan truyền khi thu hồi quyền hệ thống, khi sử dụng tuỳ chọn `WITH ADMIN OPTION`.

19.2.2. Quyền trên các đối tượng

Mỗi quyền trên đối tượng được gán cho phép người dùng được gán thực thi một số thao tác trên đối tượng, bảng dưới đây liệt kê các quyền có thể được gán trên một đối tượng.

Quyền	Table	View	Sequence	Procedure
ALTER	X		X	
DELETE	X	X		
EXECUTE				X
INDEX	X			
INSERT	X	X		
REFERENCES	X			
SELECT	X	X	X	
UPDATE	X	X		

19.2.3. Gán các quyền trên đối tượng

Sử dụng cú pháp sau đây để gán một quyền trên đối tượng.

Cú pháp:

```
GRANT { object_priv [(column_list)]
      [, object_priv [(column_list)] ]...
      |ALL [PRIVILEGES]}
      ON [schema.]object
      TO {user|role|PUBLIC}
      [, {user|role|PUBLIC} ]...
      [WITH GRANT OPTION]
```

Với:

`object_priv` chỉ định quyền đối tượng được gán column_list chỉ định các cột của một bảng hay view (tuỳ chọn này chỉ sử dụng khi gán các quyền INSERT,REFERENCES hay UPDATE.

`ALL` gán tất cả các quyền cho đối tượng mà đã được gán với WITH ADMIN OPTION.

`ON object` chỉ định đối tượng trên đó các quyền được gán .

`WITH GRANT OPTION` cho phép người được gán quyền có thể gán các quyền đó cho một user khác.

Lưu ý:

- Để gán các quyền, các đối tượng phải thuộc về schema của user gán hoặc cần có quyền `WITH GRANT OPTION.`
- Mặc định nếu một object thuộc về một user nào đó thì user đó có đầy đủ các quyền trên đối tượng đó.
- Tuỳ chọn `WITH ADMIN OPTION` không dùng cho việc gán các quyền cho các chức danh.

19.2.4. Thông tin về các quyền

Thông tin về các quyền được lưu trữ trong các data dictionary.

Một số thông tin ta cần quan tâm:

- `DBA_TAB_PRIVS: GRANTEE, OWNER, TABLE_NAME, GRANTOR, PRIVILEGE, GRANTABLE`

- DBA_COL_PRIVS: GRANTEE, OWNER, TABLE_NAME, COLUMN_NAME, GRANTOR, PRIVILEGE, GRANTABLE

Truy vấn thông tin trên bảng DBA_TAB_PRIVS để lấy thông tin về các quyền trên đối tượng được gán cho user.

```
svrmgr> SELECT * FROM dba_tab_privs
      2> WHERE GRANTEE='SCOTT'

GRANTEE    OWNER  TABLE_NAME   GRA     PRIVILEGE  GRA
---------  -----  -----------  -----   ---------- -----
SCOTT      SYS    RESUMES      SYS     READ       NO
1 row selected.
```

19.2.5. Thu hồi các quyền trên đối tượng

Sử dụng cú pháp sau đây để lấy lại quyền đã cấp.

Cú pháp:

```
REVOKE { object_priv
      [, object_priv ]...
      |ALL [PRIVILEGES] }
      ON [schema.]object
      FROM {user|role|PUBLIC}
      [, {user|role|PUBLIC} ]...
      [CASCADE CONSTRAINTS]
```

Với:

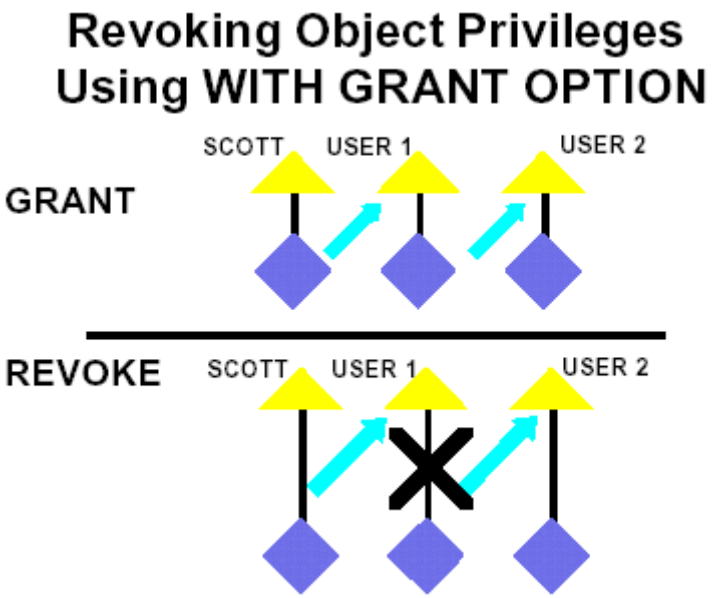
```
object_pri        chỉ định quyền trên đối tượng đã được gán
ALL               thu hồi toàn bộ các quyền trên đối tượng đã
                  được gán cho user.
ON                chỉ định đối tượng trên đó các quyền trên đối
                  tượng được thu hồi.
FROM              chỉ định user hay role bị thu hồi quyền.
CASCADE CONSTRAINTS
                  xoá tất cả các tham chiếu mà việc thu hồi đã
                  được định nghĩa do sử dụng quyền REFERENCES hay
                  ALL.
```

Ví du:

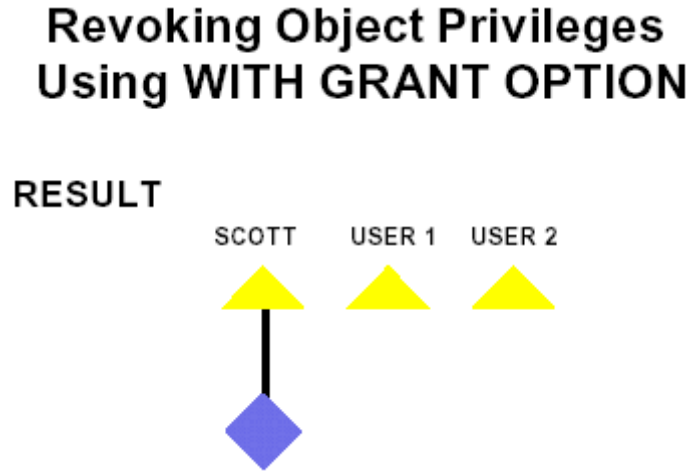
```
REVOKE execute ON dbms_pipe
      FROM scott;
```

Lấy lại các quyền đối tượng khi sử dụng mệnh đề WITH GRANT OPTION

Việc thu hồi các quyền đối tượng sẽ dẫn đến các việc thu hồi các quyền lan truyền. Hình sau minh hoạ quá trình lan truyền đó:



Hình vẽ 101. Thu hồi quyền trên đối tượng – trước khi thu hồi

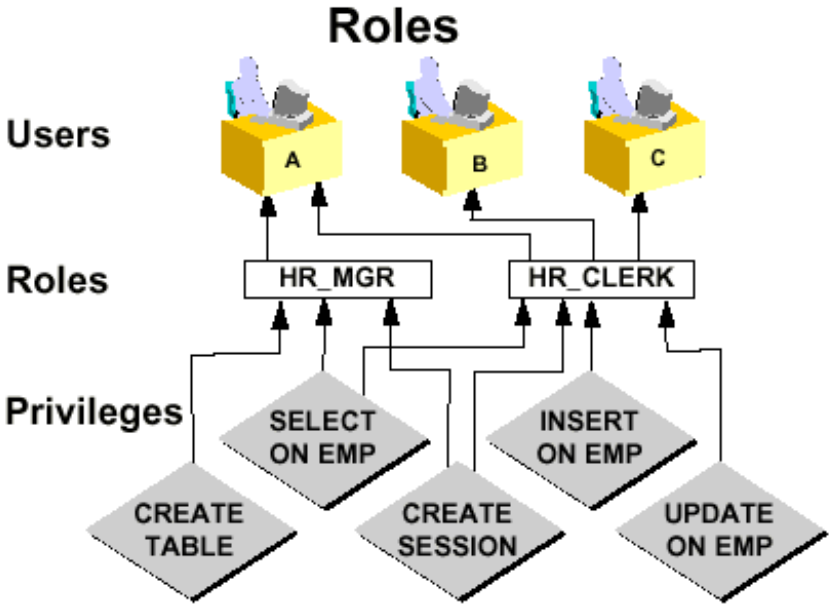


Hình vẽ 102. Thu hồi quyền trên đối tượng – sau khi thu hồi

Chương 20. QUẢN LÝ CHỨC DANH (ROLE)

20.1.CHỨC DANH (ROLE) TRONG DATABASE

Oracle cung cấp công cụ cho phép quản lý một cách dễ dàng các quyền thông qua việc sử dụng chức danh (Roles). Chức danh là một nhóm các quyền được đặt tên có liên quan đến nhau và được gán cho một user hay một chức danh khác. Chức danh được đưa ra nhằm làm dễ dàng quản lý các quyền trong hệ thống.



Hình vẽ 103. Chức danh trong database

20.1.1. Các tính chất của chức danh

740. Được gán và lấy lại từ user với cùng câu lệnh.

741. Có thể gán cho bất cứ user nào một chức danh ngoại trừ cho chính nó.

742. Có thể bao gồm cả quyền hệ thống (system privileges) và quyền đối tượng (object privileges) trong chức danh.

743. Có thể enable và disable các chức danh được gán cho các user.

744. Có thể yêu cầu password khi cần enable các chức danh.

745. Tên các chức danh phải là duy nhất trong các user đang tồn tại và tên các chức danh đang tồn tại.

746. Không thuộc về bất cứ user nào và không thuộc về bất cứ schema nào.

747. Có các mô tả đặt trong data dictionary.

20.1.2. Lợi ích của việc sử dụng chức danh

Giảm công việc gán các quyền

Sử dụng các chức danh đơn giản hoá việc quản lý các chức danh, bằng cách gán một tập các quyền cho người dùng. Có thể gán các quyền cho một chức danh và sau đó gán chức danh đó cho các user.

Quản lý các quyền một cách linh động

Khi thay đổi các quyền có trong một chức danh thì quyền của tất cả các user đã được gán các chức danh đó sẽ bị thay đổi theo.

Chọn các quyền đã có sẵn

Một chức danh có thể được enable hay disable tạm thời để cho phép hay cấm các quyền.

Không có hiện tượng lan truyền khi lấy lại các chức danh.

Các quyền đối tượng có thể lấy lại mà không gây ra hiện tượng lan truyền.

20.2.QUẢN LÝ CHỨC DANH

20.2.1. Tạo và sửa chữa các Chức danh

Sử dụng câu lệnh sau đây thực hiện việc tạo các chức danh

Cú pháp:

```
CREATE ROLE role_name [NOT IDENTIFIED | IDENTIFIED
      {BY password | EXTERNALLY }]
```

Với:

```
role_name              tên của chức danh
NOT IDENTIFIED         chỉ định không cần kiểm tra chức danh khi
                       enable chức danh
BY password            mật khẩu người dùng cần cung cấp khi
                       enable chức danh
EXTERNALLY             chỉ định user phải được xác lập bởi dịch vụ
                       bên ngoài (như hệ điều hành hay dịch vụ
                       bên thứ ba) trước khi enable chức danh.
```

Ví dụ:

```
CREATE ROLE sales_clerk;

Hoặc:
CREATE ROLE hr_clerk
      IDENTIFIED BY bonus;

Hoặc:
CREATE ROLE hr_manager
      IDENTIFIED EXTERNALLY;
```

Chú ý:

Câu lệnh `CREATE ROLE IDENTIFIED GLOBALLY` chỉ định rằng việc kiểm tra xác lập chức danh thông qua Oracle Security Server.

20.2.2. Các chức danh được định nghĩa sẵn

Các chức danh được liệt kê dưới đây được định nghĩa tự động trong Oracle Database. Connect và Resource là chức danh cung cấp để phù hợp với các version trước đây của Oracle database.

Các chức danh EXP_FULL_DATABASE và IMP_FULL_DATABASE dùng cho các công cụ export và import.

Các chức danh có tên là DELETE_CATALOG_ROLE, EXCUTE_CATALOG_ROLE và SELECT_CATALOG_ROLE cho phép thực hiện truy xuất tới các views và các packages trong data dictionary. Các chức danh này có thể gán cho user không có quyền DBA nhưng muốn xem thông tin trong các bảng và view thuộc data dictionary.

Tên chức danh	Diễn giải
CONNECT RESOURCE	Chức danh cung cấp sẵn để tương thích với các phiên bản trước đó
DBA	Tất cả các quyền hệ thống, có tuỳ chọn: WITH ADMIN OPTION
EXP_FULL_DATABASE	Quyền export dữ liệu của database
IMP_FULL_DATABASE	Quyền import dữ liệu vào database
DELETE_CATALOG_ROLE	Quyền xoá dữ liệu
EXECUTE_CATALOG_ROLE	Quyền thực hiện một thủ tục
SELECT_CATALOG_ROLE	Quyền lấy dữ liệu

Một số chức danh đặc biệt khác:

Oracle còn có một số chức danh khác để xác lập người quản trị database, trên nhiều hệ điều hành khác nhau, các chức danh này được goi là OSOPER và OSDBA tên của chúng có thể khác biệt trên các hệ điều hành. Các chức danh khác được định nghĩa bởi SQL script được cung cấp cùng với database.

20.2.3. Sửa chữa các chức danh

Sử dụng cú pháp sau đây để sửa chữa một chức danh đã tồn tại.

Cú pháp:

```
ALTER ROLE role_name {NOT IDENTIFIED | IDENTIFIED
     {BY password | EXTERNALLY }};
```

Với:

```
role_name        tên của chức danh cần thay đổi.
```

```
NOT IDENTIFIED   chỉ định không cần xác nhận khi enable chức danh
IDENTIFIED       chỉ định cần xác nhận khi enable các chức danh
BY password      cung cấp mật khẩu xác nhận khi enable chức danh
EXTERNALLY       chỉ định user cần được xác nhận bởi dịch vụ bên
                 ngoài (cơ chế xác nhận bởi hệ điều hành)
```

Ví dụ:

```
ALTER ROLE sales_clerk
      IDENTIFIED BY commission;

Hoặc:
ALTER ROLE hr_clerk
      IDENTIFIED EXTERNALLY;

Hoặc:
ALTER ROLE hr_manager
      NOT IDENTIFIED;
```

20.2.4. Gán các chức danh

Sử dụng cú pháp sau đây để gán một chức danh cho một user:

Cú pháp :

```
GRANT role_name [, role_name ]...
TO {user|role|PUBLIC}
[, {user|role|PUBLIC} ]...
[WITH ADMIN OPTION]
```

Với :

```
role_name        tên của chức danh gán
user             tên của user được gán chức danh
role             tên của chức danh được gán
PUBLIC           chỉ định gán chức danh cho tất cả các user
WITH ADMIN OPTION
                 cho phép user được gán chức danh có thể gán
                 chức danh cho user khác.
```

Ví dụ:

```
GRANT sales_clerk TO scott;

Hoặc:
GRANT hr_clerk,
      TO hr_manager;

Hoặc:
GRANT hr_manager TO scott
      WITH ADMIN OPTION;
```

User tạo chức danh được mặc định gán tuỳ chọn `WITH ADMIN OPTION.`

20.2.5. Thiết lập chức danh mặc định

Một user có thể có nhiều chức danh được gán. Chức danh mặc định là một tập con các chức danh được tự động enable khi user log on vào database. Giới hạn các chức danh mặc định bằng câu lệnh ALTER USER.

Cú pháp :

```
ALTER USER user DEFAULT ROLE
      {role [,role]... | ALL [EXCEPT role [,role]... ] | NONE}
```

Với:

```
user              tên của user được gán các chức danh
role              tên của chức danh được thiết lập mặc định
ALL               đặt tất cả các chức danh được gán cho user là
                  mặc định ngoại trừ các chức danh nằm sau mệnh
                  đề EXCEPT.
EXCEPT            chỉ định các chức danh đi sau mệnh đề này không
                  thuộc vào các chức danh mặc định.
NONE              không chức danh nào trong số chức danh được gán
                  cho user là chức danh mặc định.
```

Ví dụ:

```
ALTER USER scott
      DEFAULT ROLE hr_clerk, sales_clerk;
Hoặc:
ALTER USER scott DEFAULT ROLE ALL;
Hoặc:
ALTER USER scott DEFAULT ROLE ALL
      EXCEPT hr_clerk;
Hoặc:
ALTER USER scott DEFAULT ROLE NONE;
```

Vì các chức danh cần được gán trước khi chúng có thể được thiết lập mặc định nên không thể khởi tạo các chức danh mặc định ngay khi sử dụng câu lệnh CREATE USER.

20.2.6. Enable và Disable các chức danh

Enable và Disable các chức danh sẽ tạm thời cho phép hoặc không cho phép các quyền kết hợp với các chức danh. Để enable các chức danh trước hết cần gán các chức danh cho user.

Khi một chức danh được enable, user có thể sử dụng các quyền được gán cho chức danh đó và ngược lại trừ khi các quyền đó được gán trực tiếp cho user hay được gán cho chức danh khác nhưng chức danh đó được enable cho user đó.

Chỉ định các chức danh được enable

Câu lệnh SET ROLE và thủ tục DBMS_SESSION.SET_ROLE cho phép enable tất cả các chức danh và disable các chức danh khác .

Cú pháp:

```
SET ROLE {role [ IDENTIFIED BY PASSWORD ]
      [, role [ IDENTIFIED BY PASSWORD ]]...
```

```
        | ALL [ EXCEPT role [, role ] ...]
        | NONE }
```

Với:

```
role            là tên của chức danh
IDENTIFIED BY password
                chỉ định mật khẩu xác nhận khi enable chức danh

ALL             enable tất cả các chức danh được gán cho user
                hiện thời ngoại trừ các chức danh trong danh
                sách sau mệnh đề EXCEPT
EXCEPT          danh sách các chức danh không được enable.
NONE            disable tất cả các chức danh cho session hiện
                thời.
```

Ví dụ:

```
SET ROLE sales_clerk
     IDENTIFIED BY commission;

Hoặc:
SET ROLE hr_clerk;

Hoặc:
SET ROLE ALL EXCEPT
     sales_clerk;
Hoặc:
SET ROLE NONE;
```

20.2.7. Thu hồi các chức danh từ các user

Sử dụng cú pháp sau đây để thu hồi các chức danh từ các user .

Cú pháp :

```
REVOKE role_name [, role_name ]...
     FROM {user|role|PUBLIC}
     [, {user|role|PUBLIC} ]...
```

Với:

```
role_name       tên của các chức danh cần thu hồi.
user            tên user bị thu hồi chức danh.
role            tên của các chức danh bị thu hồi chức danh.
PUBLIC          thu hồi các quyền hay chức danh từ tất cả các
                user.
```

Ví dụ:

```
REVOKE sales_clerk FROM scott;

Hoặc:
REVOKE hr_manager FROM PUBLIC;
```

20.2.8. Xoá các chức danh

Để xoá các chức danh từ database sử dụng câu lệnh sau.

Cú pháp:

```
DROP ROLE role_name;
```

Với:

```
role_name        là tên của chức danh bị xoá.
```

Ví dụ:

```
DROP ROLE hr_manager;
```

20.3.THÔNG TIN VỀ CÁC CHỨC DANH

Các bảng thông tin về chức danh

Thông tin về các chức danh được lấy trong data dictionary. Có rất nhiều tables và views chứa thông tin về các quyền được gán cho user.

Tên view	Diễn giải
DBA_ROLES	Tất cả các chức danh trong database
DBA_ROLE_PRIVS	Các chức danh đã được gán quyền cho user hay chức danh khác
ROLE_PRIVS	Các chức danh đã được gán quyền cho chức danh khác
DBA_SYS_PRIVS	Quyền hệ thống gán cho user hay chức danh
ROLE_SYS_PRIVS	Quyền hệ thống gán cho chức danh
ROLE_TAB_PRIVS	Quyền trên table được gán cho chức danh
SESSION_ROLES	Các chức danh được phép của user hiện thời

Ví dụ: Thông tin về các quyền cấp phát cho user

```
SVRMGRL> SELECT chức danh, password_required FROM dba_chức
danhs;

ROLE                           PASSWORD
------------------------------ -----------
CONNECT                        NO
RESOURCE                       NO
DBA                            NO
AQ_USER_ROLE                   NO
SELECT_CATALOG_ROLE            NO
EXECUTE_CATALOG_ROLE           NO
DELETE_CATALOG_ROLE            NO
AQ_ADMINISTRATOR_ROLE          NO
RECOVERY_CATALOG_OWNER         NO
IMP_FULL_DATABASE              NO
EXP_FULL_DATABASE              NO
SNMPAGENT                      NO
SALES_CLERK                    YES
HR_CLERK                       EXTERNAL
14 rows selected.
```

Chương 21. TÍNH NĂNG HỖ TRỢ NGÔN NGỮ QUỐC GIA

21.1.NGÔN NGỮ QUỐC GIA

21.1.1. Các đặc điểm chính

NLS (National Language Support) cho phép các công cụ database và các thông báo lỗi, trật tự sắp xếp, thông tin ngày tháng, tiền tệ, con số và lịch được chuyển đổi tự động phù hợp với ngôn ngữ quốc gia người sử dụng.

Các thao tác trên ngôn ngữ phụ thuộc được điều khiển bởi một số tham số và biến môi trường trên cả 2 phía Client và Server.

Server và Client có thể chạy trên cùng hoặc khác computer. Trong trường hợp client và server sử dụng các tập character set khác nhau, Oracle sẽ chuyển đổi các tập character set tự động.

Một số đặc điểm chính của NLS:

- Hỗ trợ nhiều loại ngôn ngữ khác nhau.
- Hỗ trợ nhiều quốc gia, lãnh thổ khác nhau.
- Hỗ trợ nhiều loại tập ký tự (character set) sử dụng trong database.
- Cho phép sắp xếp ký tự trên các loại ngôn ngữ của từng quốc gia.
- Thông báo sử dụng tiếng địa phương.
- Định dạng thời gian, ngày tháng theo định dạng của từng quốc gia.
- Định dạng số theo khuôn dạng của từng quốc gia.
- Định dạng tiền tệ theo khuôn dạng của từng quốc gia.

Hỗ trợ ngôn ngữ quốc gia

- Các user có thể tương tác, lưu trữ, xử lývà truy xuất dữ liệu trong ngôn ngữ quốc gia của họ bao gồm các vùng sau: Tây Âu, Đông Âu, Trung Đông, Đông Nam Á và Bắc Á.
- Các nước khác nhau và các vùng địa lí khác nhau được chỉ định bằng các chuyển đổi văn hoá khác nhau ảnh hưởng trực tiếp đến định dạng của dữ liệu.
- Có rất nhiều tập kí tự khác nhau bao gồm tập kí tự đơn (single_byte) và tập kí tự nhiều byte được hỗ trợ.
- Các công cụ của database và các thông báo lỗi có thể hiển thị trong ngôn ngữ quốc gia của người dùng. Oracle hỗ trợ khoảng 26 ngôn ngữ khác nhau.
- Các định dạng ngày tháng được định dạng cùng với các chuẩn ISO cho ngày tháng năm và giờ.
- Dữ liệu số được định dạng tương ứng trên định dạng địa phương.
- Các biểu tượng tiền tệ phản ánh kinh tế địa phương và chuyển đổi sang chuẩn ISO.

21.1.2. Chọn tập kí tự cho database

Bộ mã kí tự chỉ định mã kí tự tương ứng mà một máy tính có thể hiển thị và nhận.

Bộ mã kí tự được dùng để biên dịch dữ liệu thành các biểu tượng có ý nghĩa hiển thị trên các máy tính.

Oracle hỗ trợ các bộ kí tự sau:

- Single_byte
- Varying_width
- Fixed_width
- Unicode

Tập kí tự Single_Byte

Tập kí tự Single_byte, trong mỗi kí tự chiếm một byte như vậy với bộ mã kí tự này có thể mã được 127 kí tự nếu bộ mã 7 bits, 256 kí tự nếu bộ mã 8 bits. Ví dụ các tập kí tự single_byte như US7ASCII, WE8ISO8859P1...

Tập kí tự Varying_Width nhiều byte

Một tập kí tự nhiều byte sử dụng nhiều byte cho một kí tự, các tập kí tự nhiều byte được sử dụng để hỗ trợ các ngôn ngữ châu Á. Một số tập kí tự nhiều byte sử dụng một số bits dấu (significant bits) để xác định tập kí tự sử dụng thuộc loại single_byte hay Varying_Width.

Một số ví dụ về các tập kí tự nhiều byte có độ rộng thay đổi: Japanese Extended Unix Code(JEUC), Chinese GB2312-80.

Tập kí tự Fixed_Width nhiều byte

Tập kí tự nhiều byte có độ rộng không thay đổi được sử dụng giống như tập kí tự nhiều byte ngoại trừ là nó được định dạng với số bits không thay đổi cho một kí tự. Ví dụ các tập kí tự : JA 16EUCFIXED, 16 bits Japanese, JA 16SJISFIXED, 16 bits Japanese.

Tập kí tự Unicode

Unicode là tập kí tự chuẩn được sử dụng để hiển thị tất cả các kí tự của máy tính bao gồm các biểu tượng kĩ thuật và tập các kí tự dùng để hiển thị, tập các kí tự Unicode có thể hiển thị các kí tự dưới dạng mã khác nhau. Ví dụ: UCS2 được định dạng 2 bytes và có độ rộng không đổi, UTF8 là bộ mã nhiều byte có độ rộng thay đổi.

21.1.3. Tập kí tự và tập kí tự quốc gia của database

Câu lệnh `CREATE DATABASE` có các mệnh đề: `CHARACTER SET` và `NATIONAL CHARACTER SET` để mô tả tập kí tự được sử dụng trong database. Cả hai tập kí tự này không thể thay đổi khi đã tạo database. Nếu mệnh đề `NATIONAL CHARACTER SET` không được chỉ ra trong câu lệnh tạo database thì database sẽ sử dụng tập kí tự tương ứng với `CHARACTER SET`.

Vì tập kí tự của database được sử dụng để xác định và lưu trữ các câu lệnh SQL và mã PL/SQL nên nó tương thích với tập ký tự EBCDIC hay 7-bit ASCII. Vì vậy, không thể sử dụng

các tập kí tự nhiều bytes (multibyte character set) hay tập ký tự có độ rộng không đổi (fixed-width) như là tập kí tự sử dụngcho database. Ta chỉ có thể sử dụng các tập ký tự này như là tập ký tự thuộc loại `NATIONAL CHARACTER SET` mà thôi.

Các kiểu dữ liệu như NCHAR, NVARCHAR2 và NCLOB được cung cấp để khai báo các cột như là các biến thể của các kiểu cơ bản như CHAR, VARCHAR2 và CLOB. Chúng được lưu trữ và sử dụng với tập ký tự national character set chứ không phải là database character set.

Tập ký tự database (Database Character Set)	**Tập ký tự quốc gia (National Character Set)**
Xác định ngay khi tạo database	Xác định ngay khi tạo database
Không thể thay đổi lại sau khi đã tạo database	Không thể thay đổi lại sau khi đã tạo database
Lưu trữ dữ liệu theo các kiểu: `CHAR, VARCHAR2, CLOB, LONG`	Lưu trữ dữ liệu theo các kiểu: `NCHAR, NVARCHAR2, NCLOB`
Lưu trữ tập dữ liệu với độ dài thay đổi	Lưu trữ tập dữ liệu có độ dài cố định và cả thay đổi

- Để mô tả một đối tượng kí tự có độ dài không thay đổi sử dụng national character set, sử dụng kiểu dữ liệu `NCHAR[size]`
- Để mô tả một đối tượng kí tự có độ dài thay đổi mà sử dụng national character set, sử dụng kiểu `NVARCHAR2[size]`.
- Để mô tả một đối tượng kiểu `CLOB` chứa các kí tự nhiều byte dộ rộng không đổi sử dụng national character set, sử dụng kiểu dữ liệu `NCLOB[size]`.

Các hướng dẫn

Tập kí tự của database và tập kí tự quốc gia có thể quan hệ rất gần gũi, ví dụ: các khách hàng người nhật sẽ chọn tập kí tự JA!16EUC làm tập kí tự cho database và JA16EUCFIXED như là tập kí tự cho quốc gia.

Tìm kiếm trên các chuỗi kí tự với kiểu kí tự có độ rộng không đổi sẽ nhanh hơn so với tập kí tự có độ rông thay đổi.

Tập kí tự có độ rộng thay đổi sẽ sử dụng không gian hiệu quả hơn.

21.2.CÁC THAM SỐ NLS

21.2.1. Lựa chọn tham số

Có ba cách chỉ định các tham số NLS:

- Chỉ định tham số NLS thông qua tham số khởi tạo nằm trong parameter file của Server .

- Sử dụng biến môi trường tại client. Thông số tại Client sẽ chép đè lên (overriding) tham số mặc định tương ứng tại server.
- Chỉ định tham số thông qua lệnh ALTER SESSION. Giá trị mới của tham số sẽ chép đè lên giá trị cũ của tham số tương ứng của server.

Chỉ định các tham số phụ thuộc ngôn ngữ phía server

Các tham số khởi tạo NLS_LANGUAGE quy định các giá trị đặc trưng cho ngôn ngữ (language-dependent) như:

- Ngôn ngữ sử dụng trong các thông báo của Oracle.
- Ngôn ngữ sử dụng đối với tên của ngày tháng.
- Các biểu tượng cho AM, PM , A.D và B.C.
- Thứ tự sắp xếp dữ liệu kí tự theo mặc định.

Các tham số khởi tạo NLS_TERRITORY quy định các giá trị đặc trưng cho từng vùng lãnh thổ (territory).

Bao gồm:

- Định dạng ngày tháng theo mặc định.
- Dấu chấm thập phân, dấu phân cách hàng nghìn.
- Biểu tượng tiền tệ quốc gia.
- Biểu tượng tiền tệ chuẩn ISO
- Ngày bắt đầu của tuần.

Chú ý:

Khi sử dụng các tham số đặc trưng cho từng vùng lãnh thổ nếu giá trị có khoảng trống trong thì cần đặt chúng trong dấu ngoặc kép. Ví dụ: “ The Netherlands”

21.2.2. Ngôn ngữ phụ thuộc và giá trị territory mặc định

Các giá trị mặc định

Tham số	Giá trị
NLS_LANGUAGE	**AMERICAN**
NLS_DATE_LANGUAGE	AMERICAN
NLS_SORT	BINARY
NLS_TERRITORY	**AMERICA**
NLS_CURRENCY	$
NLS_ISO_CURRENCY	AMERICA
NLS_DATE_FORMAT	DD-MON-YY
NLS_NUMERIC_CHARACTERS	,.

Tham số khởi tạo NLS _LANGUAGE quyết định giá trị mặc định cho các tham số sau đây:

Tham số	Diễn giải
NLS_DATE_LANGUAGE	Thay đổi tường minh các định dạng về thời gian, ngày tháng
NLS_SORT	Thay đổi trật tự sắp xếp các ký tự trong database

NLS_TERRITORY quyết định giá trị mặc định cho các tham số sau đây:

Tham số	Diễn giải
NLS_CURRENCY	Thay đổi tường minh các ký tự biểu diễn tiền tệ
NLS_ISO_CURRENCY	Ký tự biểu diễn tiền tệ của quốc gia theo chuẩn ISO
NLS_DATE_FORMAT	Chỉ rõ định dạng ngày tháng
NLS_NUMERIC_CHARACTERS	Chỉ ra tường minh các ký tự biểu diễn trong số thập phân

21.2.3. Xác định các biến môi trường

Chỉ định biến môi trường NLS_LANG

```
NLS_LANG=<language>_<territory>.<charset>
```

Thêm các biến môi trường :

```
NLS_DATE_FORMAT
NLS_DATE_LANGUAGE
NLS_SORT
NLS_NUMERIC_CHARACTERS
NLS_CURRENCY
NLS_ISO_CURRENCY
NLS_CALENDAR
```

Biến môi trường NLS_LANG

Sử dụng biến môi trường NLS_LANG để chép đè lên giá trị NLS mặc định sử dụng cho user. Xác định giá trị của tham số NLS_LANG:

```
NLS_LANG=<language>_<territory>.<charset>
```

Với:

```
language        tên ngôn ngữ quốc gia sử dụng (là giá trị của
                tham số NLS_LANGUAGE).
territory       đặc trưng vùng lãnh thổ (là giá trị của tham số
                NLS_TERRITORY).
charset         bộ mã kí tự sử dụng bởi ứng dụng phía client.
```

Ví dụ:

```
NLS_LANG=GERMAN_GERMAN.W8EISO8850P1
```

Tham số NLS_LANG quy định tập kí tự cho client. Các clients khác nhau có thể sử dụng các bộ mã kí tự khác nhau. Dữ liệu truyền đi từ client tới server sẽ được chuyển đổi tự động từ tập kí tự sử dụng tại client sang tập kí tự sử dụng tại server. Việc chuyển đổi này là trong suốt đối với các ứng dụng tại client.

Các biến môi trường mở rộng

Các tham số khởi tạo NLS đều có thể khai báo dưới dạng biến môi trường, chúng quy định các đặc trưng ngôn ngữ sử dụng tại mỗi clients. Tham số NLS_CALENDAR được sử dụng để chọn hệ thống lịch mà Oracle sẽ sử dụng.

Một số tham số có thể chỉ định dưới dạng biến môi trường tại clients: NLS_LIST_SEPARATOR, NLS_DISPLAY, NLS_CREDIT, NLS_DEBIT, NLS_LANG, NLS_MONETARY,NLS_CHAR.

21.2.4. Chỉ định đặc trưng ngôn ngữ (Language-Dependent) cho từng session

Để thay đổi các đặc tính NLS cho một session ta sử dụng câu lệnh ALTER SESSION. Tất cả các biến môi trường có thể được khởi tạo ở cả phía client và server đều có thể thay đổi được thông qua câu lệnh ALTER SESSION .

Ví dụ: định dạng của date được thay đổi cho session.

```
ALTER SESSION SET
      NLS_DATE_FORMAT='DD.MM.YYYY';
```

Ta cũng có thể sử dụng công cụ SQL*Plus để thiết lập các biến môi trường thay cho câu lệnh ALTER SESSION. Có thể sử dụng package DBMS_SESSION.SET_NLS để khởi tạo các tham số NLS cho session.

Ví dụ:

```
DBMS_SESSION.SET_NLS('NLS_DATE_FORMAT',''DD.MM.YYYY'');
```

21.2.5. Tham số NLS và các hàm SQL

Sắp xếp nhị phân là phương thức săp xếp thường được sử dụng. Trong đó, các phần tử được sắp xếp theo các giá trị nhị phân trong bộ mã kí tự. Vị trí của các kí tự có thể thay đổi theo ngôn ngữ sử dụng.

Để loại bỏ giới hạn của sắp xếp nhị phân, Oracle cung cấp phương thức sắp xếp theo ngôn ngữ bằng cách khởi tạo tham số NLS_SORT.

Ví dụ:

```
SQL> ALTER SESSION SET NLS_SORT=BINARY;
Session altered.
```

```
SQL> SELECT letter FROM letters ORDER BY letter;
LETTER
------
-a
b
c
z
ỹ
o
6 rows selected.
```

Tham số NLS trong các hàm SQL

Oracle cung cấp một số hàm SQL sử dụng trên các tập kí tự single_byte và multibyte.

Các hàm SQL cho phép chỉ định các tham số NLS giống như một phần trong danh sách tham số của chúng. Các tham số này sẽ thay thế cho các giá trị NLS chỉ định bởi các biến môi trường.

Ví dụ: sử dụng tham số ngôn ngữ trong hàm SQL.

```
SVMRGRL> SELECT TO_CHAR(hiredate,'dd.mon.yyyy',
      2> 'NLS_DATE_LANGUAGE=GERMAN')
      3> FROM emp

TO_CHAR(HIR
-----------
17.dez.1980
20.feb.1981
22.feb.1981
02.apr.1981
28.sep.1981
01.mai.1981
09.jun.1981
19.apr.1987
17.nov.1981
08.sep.1981
23.mai.1987
03.dez.1981
03.dez.1981
23.jan.1982
14 rows selected.


SVRMGR> SELECT ename,
2> TO_CHAR(sal,'99G999D99','NLS_NUMERIC_CHARACTERS='',.''')
3> FROM emp;

ENAME         TO_CHAR(SA
-------       ----------
SMITH         800,00
ALLEN         1.600,00
WARD          1.250,00
JONES         2.975,00
MARTIN        1.250,00
```

```
BLAKE          2.850,00
CLARK          2.450,00
SCOTT          3.000,00
KING           5.000,00
TURNER         1.500,00
ADAMS          1.100,00
JAMES          950,00
FORD           3.000,00
MILLER         1.300,00
14 rows selected.
```

Các hàm SQL sau đây sử dụng tham số NLS

Tên hàm	Tham số NLS
TO_DATE	NLS_DATE_LANGUAGE NLS_CALENDAR
TO_NUMBER	NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY
TO_CHAR	NLS_DATE_LANGUAGE NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY NLS_CALENDAR
NLS_UPPER, NLS_LOWER, NLS_INITCAP, NLSSORT	NLS_SORT

Một số giá trị được sử dụng giống như các mặt nạ (mask) định dạng trong các hàm như TO_CHAR, TO_DATE, và TO_NUMBER.

- "D" cho dấu phân chấm thập phân
- "G" dấu phân cách hàng nghìn
- "L" cho biểu tượng tiền tệ
- "C" cho biểu tượng tiền tệ theo chuẩn ISO

Mặt nạ định dạng cho ngày tháng:

- "RM, rm" định dạng tháng theo kiểu roman
- "IW" định dạng tuần theo chuẩn ISO
- "IYYY, IYY, IY," và "I" định dạng năm theo chuẩn ISO

21.3.THÔNG TIN VỀ CÁC GIÁ TRỊ NLS ĐƯỢC KHỞI TẠO

21.3.1. Thông tin về tập ký tự sử dụng

Thông tin được lấy trong từ điển dữ liệu, bảng NLS_DATABASE_PARAMETERS

Sử dụng câu lệnh truy vấn sau đây để lấy thông tin về các giá trị NLS được khởi tạo

```
SVRMGR> SELECT parameter, value FROM nls_database_parameters
2> WHERE parameter LIKE '%CHARACTERSET%';
PARAMETER                      VALUE
------------------------       ------------------------------
NLS_CHARACTERSET               WE8ISO8859P1
NLS_NCHAR_CHARACTERSET         US7ASCII
2 rows selected.
```

View này hiển thị giá trị các tham số được chỉ định bắt buộc trong tệp khởi tạo:

```
SVRMGR> SELECT * FROM nls_instance_parameters;

PARAMETER                        VALUE
--------------------------       ------------------------------
NLS_LANGUAGE                     AMERICAN
NLS_TERRITORY                    AMERICA
NLS_SORT
NLS_DATE_LANGUAGE
NLS_DATE_FORMAT
NLS_CURRENCY
NLS_NUMERIC_CHARACTERS
NLS_ISO_CURRENCY
8 rows selected.
```

View sau đây hiển thị các thông tin về NLS khởi tạo cho session:

```
SVRMGR> SELECT * FROM nls_session_parameters;

PARAMETER                      VALUE
------------------------ ------------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_CURRENCY                   $
NLS_ISO_CURRENCY               AMERICA
NLS_NUMERIC_CHARACTERS         .,
NLS_CALENDAR                   GREGORIAN
NLS_DATE_FORMAT                DD-MON-YY
NLS_DATE_LANGUAGE              AMERICAN
NLS_SORT BINARY
9 rows selected.
```

21.3.2. Thông tin về các thiết lập thông số NLS

Thông tin NLS được đặt trong các data dictionary views: NLS_INSTANCE_PARAMETERS và NLS_SESSION_PARAMETERS.

NLS_INSTANCE_PARAMETERS

- Tham số (tham số khởi tạo NLS được thiết lập một cách tường minh)

- Giá trị

NLS_SESSION_PARAMETERS

- Tham số (Các tham số NLS của từng phiên)
- Giá trị

Hiển thị các tham số NLS sử dụng trong parameter file tại server

```
SVRMGR> SELECT * FROM nls_instance_parameters;
PARAMETER                      VALUE
-----------------------------  ------------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_SORT
NLS_DATE_LANGUAGE
NLS_DATE_FORMAT
NLS_CURRENCY
NLS_NUMERIC_CHARACTERS
NLS_ISO_CURRENCY
8 rows selected.
```

Hiển thị nội dung các tham số môi trường đang sử dụng trong session

```
SVRMGR> SELECT * FROM nls_session_parameters;
PARAMETER                      VALUE
-----------------------------  ------------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_CURRENCY                   $
NLS_ISO_CURRENCY               AMERICA
NLS_NUMERIC_CHARACTERS         .,
NLS_CALENDAR                   GREGORIAN
NLS_DATE_FORMAT                DD-MON-YY
NLS_DATE_LANGUAGE              AMERICAN
NLS_SORT                       BINARY
9 rows selected.
```

Sử dụng câu lệnh truy vấn sau đây lấy thông tin về các tham số khởi tạo hợp lệ:

```
SVRMGR> SELECT * FROM v$nls_valid_values
2> WHERE parameter='LANGUAGE';
PARAMETER       VALUE
---------       --------------
LANGUAGE        AMERICAN
LANGUAGE        GERMAN
LANGUAGE        FRENCH
LANGUAGE        CANADIAN FRENCH
LANGUAGE        SPANISH
LANGUAGE        ITALIAN
LANGUAGE        DUTCH
LANGUAGE        SWEDISH
LANGUAGE        NORWEGIAN
LANGUAGE        DANISH
...
```

Hiển thị giá trị hiện thời của giá trị NLS khởi tạo

```
SVRMGR> SELECT * FROM v$nls_parameters;
PARAMETER                VALUE
------------------       ------------------
NLS_LANGUAGE             AMERICAN
NLS_TERRITORY            AMERICA
NLS_CURRENCY             $
NLS_ISO_CURRENCY         AMERICA
NLS_NUMERIC_CHARAC       .,
NLS_CALENDAR             GREGORIAN
NLS_DATE_FORMAT          DD-MON-YY
NLS_DATE_LANGUAGE        AMERICAN
NLS_CHARACTERSET         WE8ISO8859P1
NLS_SORT                 BINARY
10 rows selected.
```

PHỤ LỤC

A – DANH MỤC CÁC HÌNH VẼ