TS. TRẦN VĂN DŨNG (CHỦ BIÊN)
TH.S NGUYỄN VIỆT HƯNG

BÀI GIẢNG:
CÔNG NGHỆ ORACLE

(Học phần: 03 tín chỉ)

HÀ NỘI - 2017

LỜI MỞ ĐẦU

Hiện nay, hệ quản trị cơ sở dữ liệu được sử dụng phổ biến trên thế giới và tại Việt Nam là: Oracle, Microsoft SQL-Server và IBM-DB2 .v.v. . Trong đó, Oracle thường được các doanh nghiệp đang phát triển và doanh nghiệp lớn sử dụng, đặc biệt là các ngân hàng và các tập đoàn tài chính, bảo hiểm, thuế,… nơi mà ngoài tốc độ xử lý thì yêu cầu bảo mật, an toàn dữ liệu luôn được đặt lên hàng đầu.

Oracle không chỉ có một hệ quản trị CSDL mạnh mẽ mà còn cung cấp một hệ thống phần mềm và các giải pháp toàn diện cho phép giải quyết hầu hết các bài toán của doanh nghiệp.

Bài giảng "Công nghệ Oracle" là tài liệu được biên soạn để phục vụ cho việc giảng dạy, học tập của giảng viên, sinh viên ngành Công nghệ thông tin. Tài liệu cung cấp những kiến thức cơ sở về hệ quản trị cơ sở dữ liệu Oracle và xây dựng ứng dụng với các công cụ của Oracle.

Bài giảng "Công nghệ Oracle" được biên soạn 03 tín chỉ theo đề cương học phần "Công nghệ Oracle" do TS Trần Văn Dũng làm chủ biên và được trình bày trong 3 chương, trong đó:

Chương 1: Ngôn ngữ PL/SQL và chương 2: Quản trị cơ sở dữ liệu Oracle do TS Trần Văn Dũng biên soạn.

Chương 3: Xây dựng ứng dụng trên Oracle Developer Suite do ThS Nguyễn Việt Hưng phụ trách biên soạn.

Hy vọng rằng bài giảng này sẽ mang lại những kiến thức bổ ích và những thông tin thiết thực giúp sinh viên và độc giả quan tâm có khả năng vận dụng kiến thức để xây dựng các ứng dụng sử dụng các công nghệ của Oracle.

Mặc dù đã hết sức cố gắng, song do biên soạn lần đầu, bài giảng không tránh khỏi những thiếu sót. Kính mong đồng nghiệp và bạn đọc đóng góp ý kiến để bài giảng được hoàn thiện hơn.

Xin chân thành cảm ơn ./.

Nhóm tác giả

MỞ ĐẦU

GIỚI THIỆU CHUNG VỀ CÔNG NGHỆ ORACLE

Công nghệ Oracle là tập hợp các sản phẩm phần mềm phục vụ cho mục đích xây dựng và quản lý hệ thống thông tin, các ứng dụng giao tiếp với cơ sở dữ liệu.

Oracle là tên của một hãng phần mềm đồng thời là một hệ quản trị cơ sở dữ liệu phổ biến trên thế giới. Hãng Oracle ra đời đầu những năm 70 của thế kỷ 20 tại nước Mỹ. Khởi đầu với phần mềm quản trị Cơ sở dữ liệu cách đây hơn 50 năm. Hiện tại ngoài sản phẩm Oracle Database Server, Oracle còn cung cấp nhiều sản phẩm phục vụ doanh nghiệp khác.

❖ Các sản phẩm của Oracle

- Database Server (Server quản lý cơ sở dữ liệu)
- Công cụ thao tác cơ sở dữ liệu: SQL*Plus
- Công cụ phát triển ứng dụng: Oracle Developer Suite (Form, Report, ….), Oracle JDeveloper, ...
- Phân tích dữ liệu: Oracle Discoverer, Oracle Express, Oracle Warehouse Builder ...
- Oracle Application Server (OAS)
- Ứng dụng đóng gói: Oracle Human Resource, Oracle Financial Applications...
- Oracle Email, Oracle Calendar, Oracle Web Conferencing ...

❖ Lịch sử các phiên bản

- Oracle v1: 1978, Oracle v2: 1980, Oracle v3 released: 1982, Oracle v4: 1984, Oracle v5: 1986, (SQLNet: hệ thống khách/chủ (client/server)).
- 1988: phát hành Oracle v6, giới thiệu ngôn ngữ PL/SQL
- Oracle7 được phát hành năm 1992 (SQL*DBA).
- Năm 1999 Oracle giới thiệu Oracle8i (i:internet).
- Năm 2001-2002: 2 phiên bản Oracle9i (Release 1&2).
- Năm 2004-2005: 2 phiên bản Oracle10g (g:Grid) (Release 1&2).
- Năm 2008: Phiên bản 11g (Release 1&2).
- 1/7/2013: Phiên bản 12c (cloud)

❖ Tổng quan về hệ quản trị CSDL Oracle

- Cơ sở dữ liệu là gì?

- Cơ sở dữ liệu (CSDL) là một hệ thống các thông tin có cấu trúc được lưu trữ trên các thiết bị lưu trữ thông tin thứ cấp (như băng từ, đĩa từ …).

- Có thể thỏa mãn yêu cầu khai thác đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng với mục đích khác nhau.

- Hệ quản trị CSDL là gì?

Hệ quản trị cơ sở dữ liệu (database management system - DBMS) là một hệ thống phần mềm nhằm cung cấp cho người sử dụng một môi trường thích hợp, hiệu quả để khai thác CSDL theo các khía cạnh lưu trữ, sửa đổi và truy vấn thông tin. Một số hệ quản trị CSDL thường gặp: MS Access, MS SQL Server20xx, MySQL, Oracle, DB2, LDAP…

Hệ quản trị CSDL Oracle (gọi tắt là Oracle) là một trong những hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ nhất thế giới. Được thiết kế để triển khai cho mọi môi trường doanh nghiệp. Việc cài đặt, quản lý rất dễ dàng, cung cấp nhiều công cụ giúp phát triển các ứng dụng một cách hoàn thiện và nhanh chóng. Oracle phù hợp cho mọi loại dữ liệu, các ứng dụng và các môi trường khác nhau nh windows và linux. Kết nối ứng dụng với công nghệ Web được tích hợp trong Oracle Web Server.

Hơn hai phần ba trong số 500 tập đoàn công ty lớn nhất thế giới (Fortune 500) sử dụng Oracle. Ở Việt Nam hầu hết các đơn vị lớn thuộc các ngành ngân hàng, kho bạc, thuế, bảo hiểm, bưu điện, hàng không, dầu khí,… đều sử dụng hệ quản trị CSDL Oracle.

- Các đặc điểm của Oracle

- Tính an toàn dữ liệu cao

- Cơ chế quyền hạn rõ ràng, ổn định.

- Dễ cài đặt, dễ triển khai, bảo trì và nâng cấp lên phiên bản mới.

- Tích hợp thêm PL/SQL, là một ngôn ngữ lập trình thủ tục, thuận lợi để viết các Trigger, StoreProcedure, Package.

- Khả năng xử lý dữ liệu rất lớn, có thể lên đến hàng trăm terabyte (1 terabyte ~ 1,000 gigabyte ~ 1,000,000,000 kilobyte) mà vẫn đảm bảo tốc độ xử lý dữ liệu rất cao.

- Khả năng bảo mật rất cao, Oracle đạt độ bảo mật cấp c2 theo tiêu chuẩn bảo mật của bộ quốc phòng mỹ và công nghệ Oracle vốn được hình thành từ yêu cầu đặt hàng của các cơ quan an ninh FBI và CIA.

- Tương thích với nhiều platform (Unix, Linux, Solaris, Windows .v.v…)

- Một vài điểm so sánh Oracle với SQL Server

Bảng 1.*Một vài so sánh Oracle và SQL Server*

	SQL Sever	Oracle
Hardware requirements	Chỉ chạy trên chip Intel base and compatible, không chạy được trên các chíp mạnh khác như Power, PA-RISC, Itanium, SPARC ...	Chạy được trên hầu hết các kiến trúc phần cứng.

Operating system	Windows	Multi-platform (Windows, linux,unix,..)
Programming language database	T-SQL (Transact SQL)	PL/SQL (Procedural Language SQL)
Instance	Từ MSSQL 2000, mỗi máy có thể nhiều hơn 1 instance. Cụ thể MSSQL 2000 (16 instanes), 2005 (50 instances)	Trên mỗi máy có nhiều Instances, số lượng phụ thuộc vào từng OS
Login Name/ DB username	Mỗi Login name có thể "map" tới nhiều DB Username trong các Database, LoginName và DB Name không nhất thiết cùng tên. vd: LoginName là SA được map tới DB Username tên là DBO trong tất cả các Database.	Không có sự phân biệt LoginName/DB Name. Khi tạo 1 user, thì đó là vừa là LoginName vừa là Db Username.
Database/ Schema (*)	Mỗi Instance có nhiều Database, và mỗi Database có nhiều schema. Có thể phân quyền cho DB Username trên schema.	Mỗi Instance xem như chỉ có 1 Database !!! Trong Database có nhiều DB Username, tương ứng mỗi DB Username có 1 và chỉ 1 schema cùng tên với user. Vì vậy, trong Oracle không có khái niệm phân quyền trên schema (chỉ có phân quyền cho từng Objects trên schema đó).
Auto Commit	On	Off
Giao diện quản trị CSDL mặc đinh	Dễ sử dụng (SQL Server Managerment Studio)	Nặng nề, khó sử dụng OEM (Oracle enterprise manager)
Command line	Giao diện dòng lệnh dài dòng và phức tạp, khó dùng	Giao diện dòng lệnh dể sử dụng, đa số là Create/Alter/Drop. Như việc phân quyền chỉ cần Grant/revoke.

(*) Schema: User có thể làm việc trong phạm vi cho phép của mình mà Oracle gọi là "khung cảnh" (Schema) của user, hay còn gọi là lược đồ CSDL của user. Mỗi lược đồ CSDL là tập hợp các đối tượng như là table, view, trigger, function, procedure,…. Người dùng được cấp quyền trên lược đồ nào thì chỉ có thể tác động lên các đối tượng trong lược đồ đó.

Trong Oracle mỗi database có nhiều schema, tương ứng với mỗi schema sẽ có một và chỉ một user trùng tên với schema đó.

VD: Trong Oracle mặc định có 2 user là SYSTEM và SCOTT, tương ứng là 2 schema cùng tên. SYSTEM là user có quyền cao nhất trong hệ thống, người dùng khi đăng nhập vào SYSTEM có thể tác động đến bất kì đối tượng trên bất kỳ schema nào. Còn khi đăng nhập vào SCOTT thì chỉ có thể tác động trên schema SCOTT.

❖ Cài đặt Oracle 10G trên windows

- Yêu cầu về cấu hình

* Phần cứng:

- RAM: ≥ 4 GB
- FREE DISK SPACE: Ổ đĩa cài đặt Oracle còn trống từ 10 GB trở lên.

* Hệ điều hành:
Page file: 2 GB – 5 GB (Sinh viên tự tìm hiểu cách thiết lập page file)

- Phần mềm cần thiết

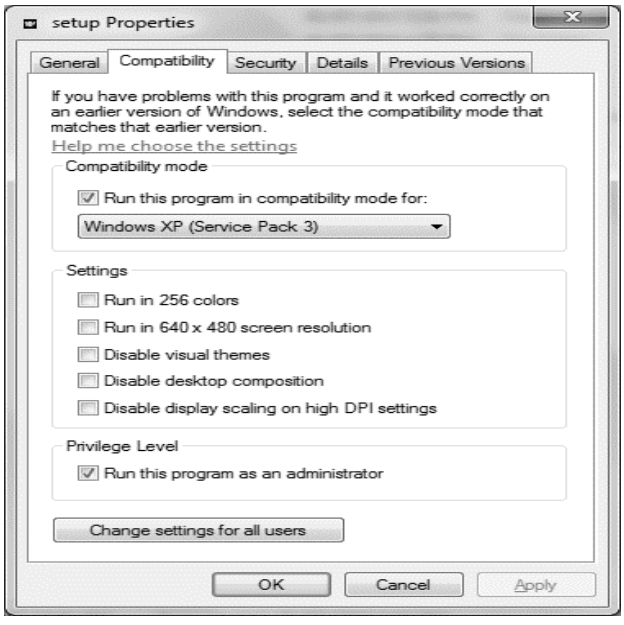
- Database 10gR2 - phần mềm cài đặt Oracle database server, có thể tải về tại địa chỉ: https://goo.gl/jwgfut
- PL/SQL developer – phần mềm hỗ trợ quản trị Oracle database, , có thể tải về tại địa chỉ: https://goo.gl/ljg1sq

- Các bước cài đặt

!Chú ý: Ngắt tất cả các kết nối mạng trước khi cài đặt!

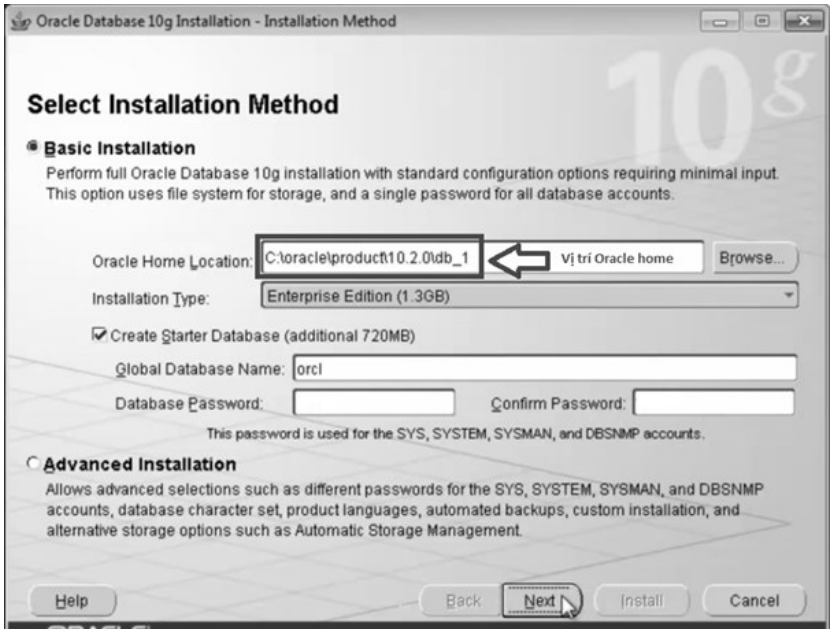
▪ *Bước 1. Thay đổi thuộc tính file setup.exe như hình dưới bao gồm:*

- Run this program in compatibility mode for: Windows XP
- Run this program as an administrator



Hình 1.*Thay đổi thuộc tính file setup*

- *Bước 2. Mở file cài đặt setup.exe*



Hình 2.*Giao diện cài đặt đầu tiên*

Trong đó:

Oracle Home Location: Thư mục chính chứa các file sau khi cài đặt của Oracle

Installation Type: Phiên bản cài đặt (chọn Enterprise Edition)

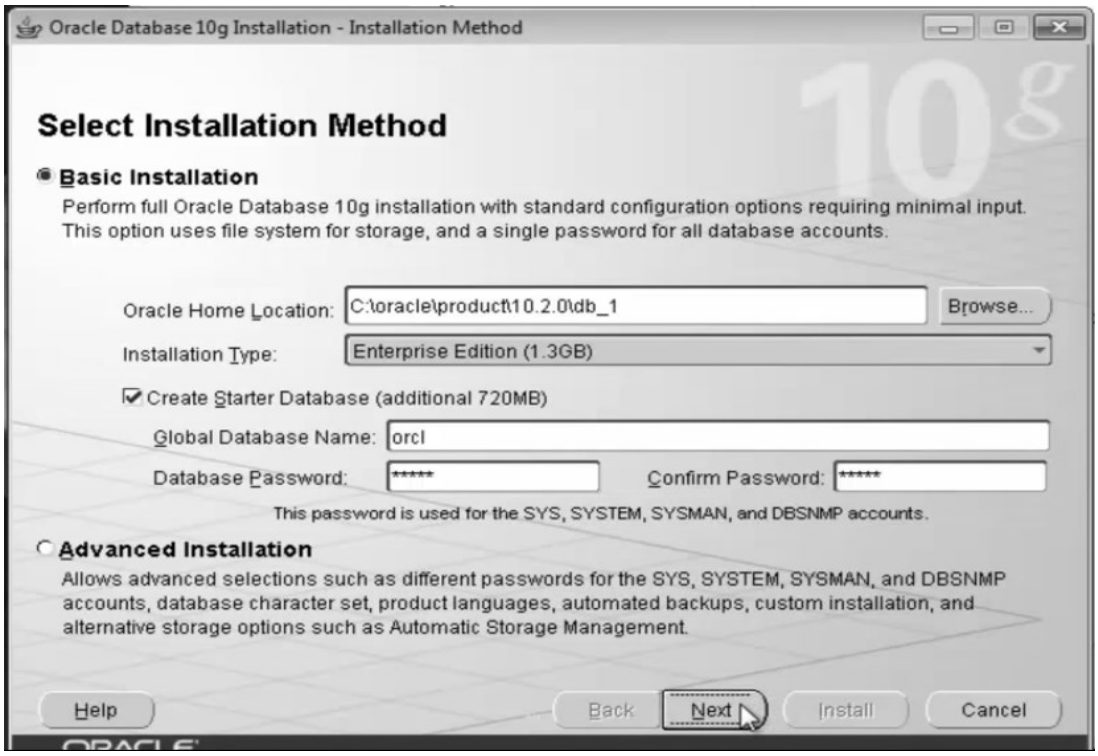
Global Database Name: Tên cơ sở dữ liệu mặc định sẽ được sau khi cài đặt xong.

Database Password: Mật khẩu đăng nhập vào CSDL (mật khẩu này dùng cho các tài khoản SYS, SYSTEM, SYSMAN, DBSNMP)

Confirm Password: Nhập lại mật khẩu đã nhập trong mục Database Password.

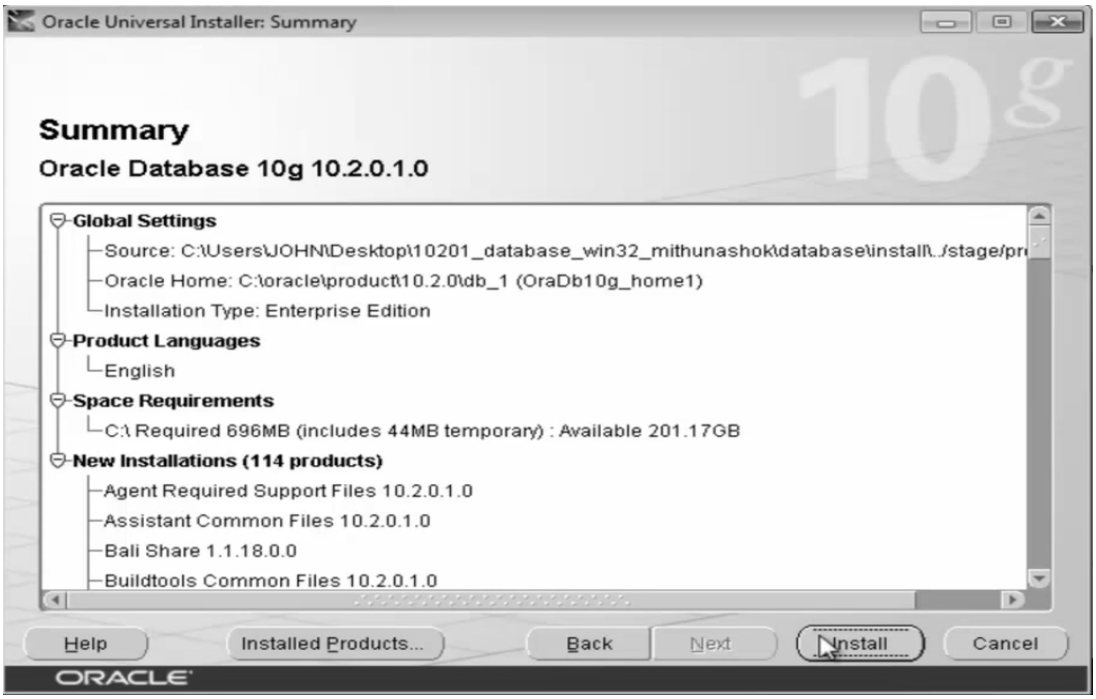
- Bước 3. Nhập mật khẩu cho database. ***(Ghi nhớ mật khẩu này!).*** Sau đó bấm next.

  Mật khẩu khuyến nghị: abc123



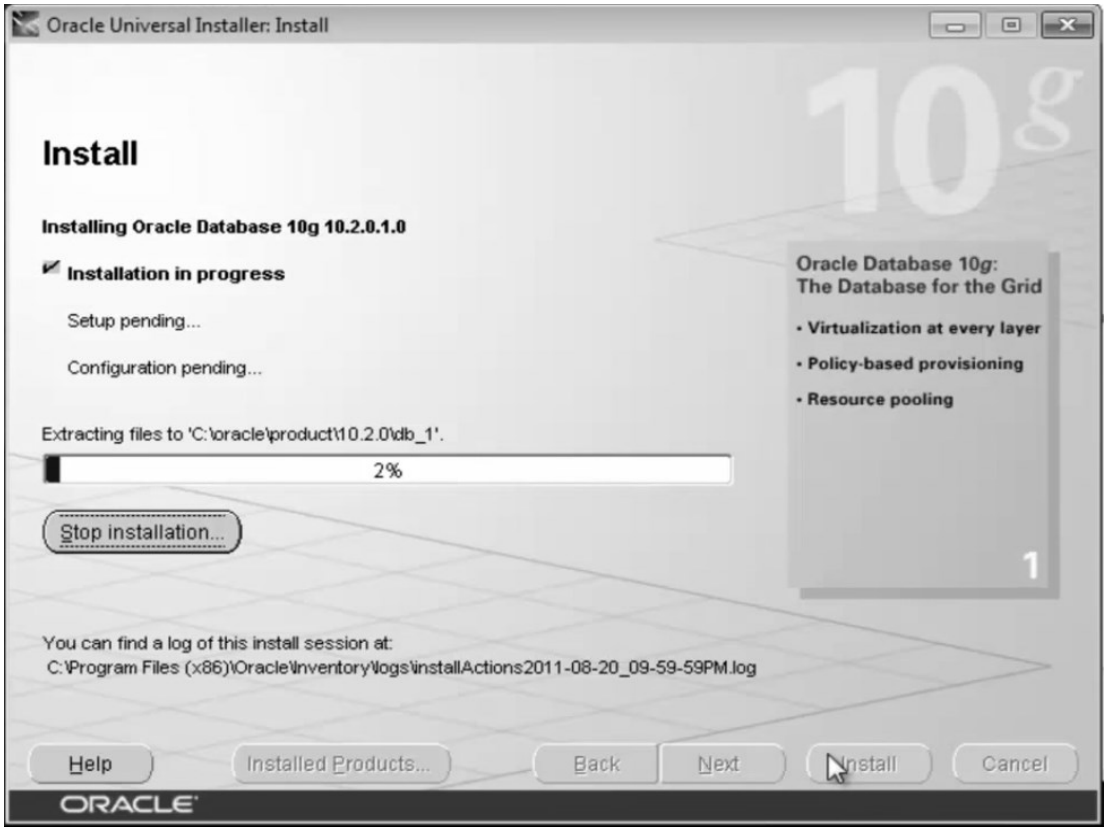
Hình 3.*Giao diện nhập mật khẩu*

- Bước 4. Bấm next liên tục cho đến khi hiện lên hình dưới thì bấm Install.



Hình 4.*Các thành phần sẽ cài đặt*

Quá trình tự động cài đặt bắt đầu. Thời gian chờ khoảng 15-20 phút.



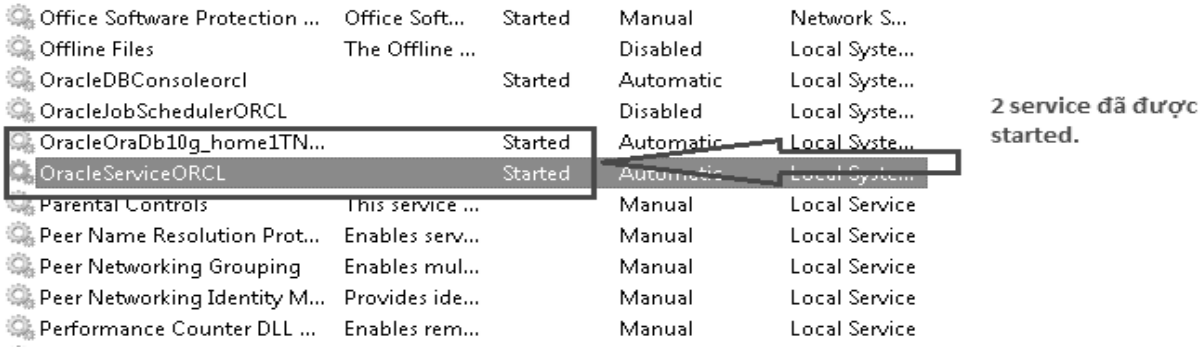
Hình 5.Quá trình cài đặt bắt đầu

❖ **Một số lưu ý quan trọng**

- Kiểm tra service mỗi khi làm việc với Oracle

Cần nhớ: Mỗi khi khởi động máy, để làm việc được với CSDL Oracle, ta tiến hành các công việc sau:

a. Kiểm tra các services của Oracle xem đã ở trạng thái started chưa. Nếu chưa thì start lên. Để xem các services trong window, vào start, gõ **services**



Hình 6.Các service của Oracle

Lời khuyên: nên thiết lập các service của Oracle ở chế độ khởi động là **Manual** thay vì Automatic để giảm thời gian khởi động windows. Khi cần làm việc với Oracle, ta tiến hành khởi động lần lượt 2 service là **OraclServiceORCL** và **OracleOraDb10g_home1TNSListener.**

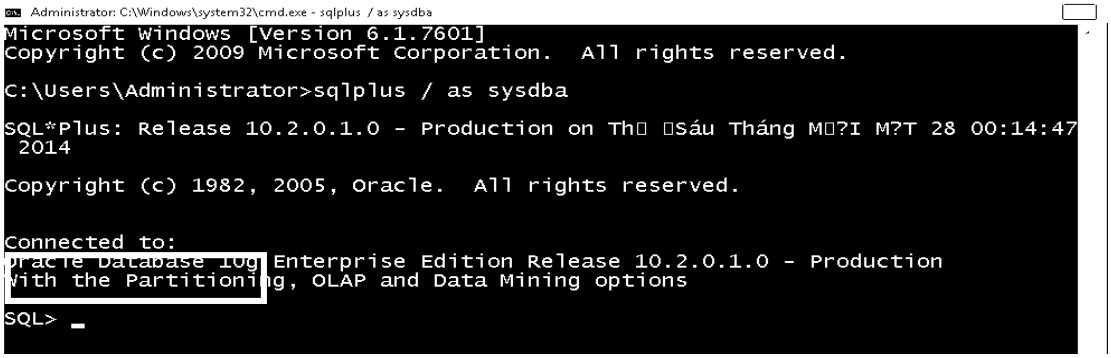
Chờ khoảng 1 phút để các tiến trình khởi động hoàn tất.

b. Đăng nhập vào sqlplus với quyền sysdba:

+ Start\cmd (Run as administrator)

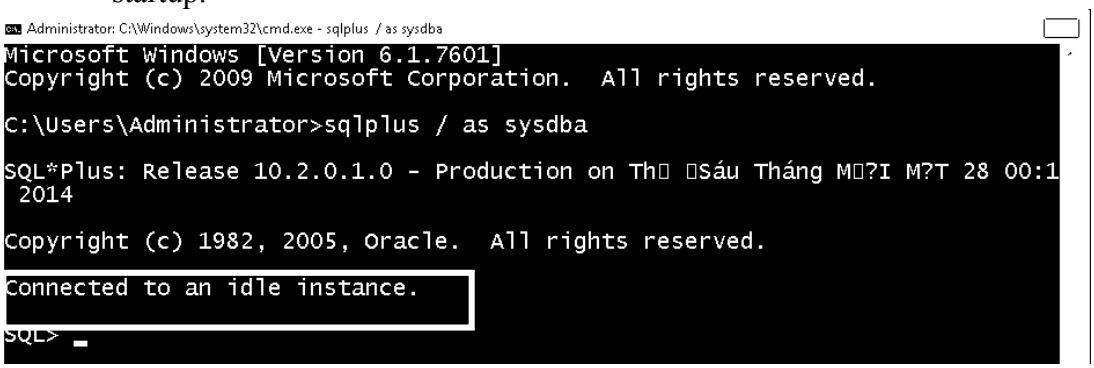
+ Gõ lệnh: sqlplus sys/abc123 as sysdba (abc123 là mật khẩu database lúc cài đặt)

- Nếu trạng thái là Connected to: … thì đã có thể làm việc được với CSDL.



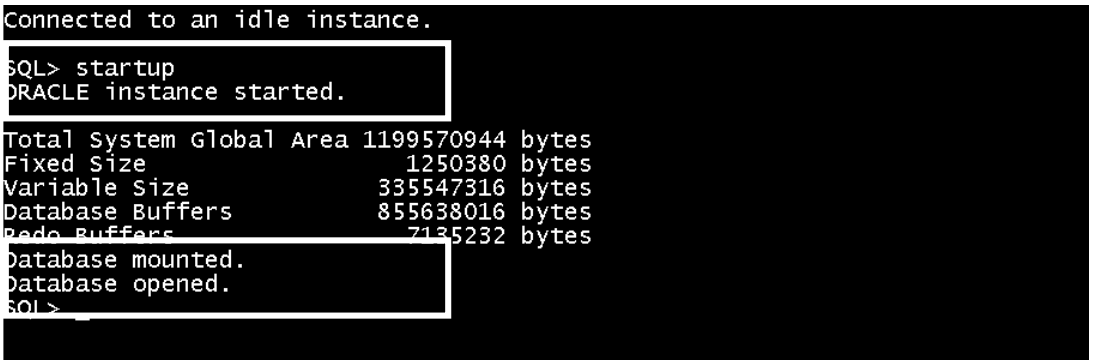
Hình 7.Trạng thái làm việc bình thường

- Nếu trạng thái là Connected to an idle instance, tức là Instance chưa được startup.



Hình 8.*Trạng thái instance chưa hoạt động*

Ta tiến hành khởi động instance bằng cách gõ lệnh: ***startup***



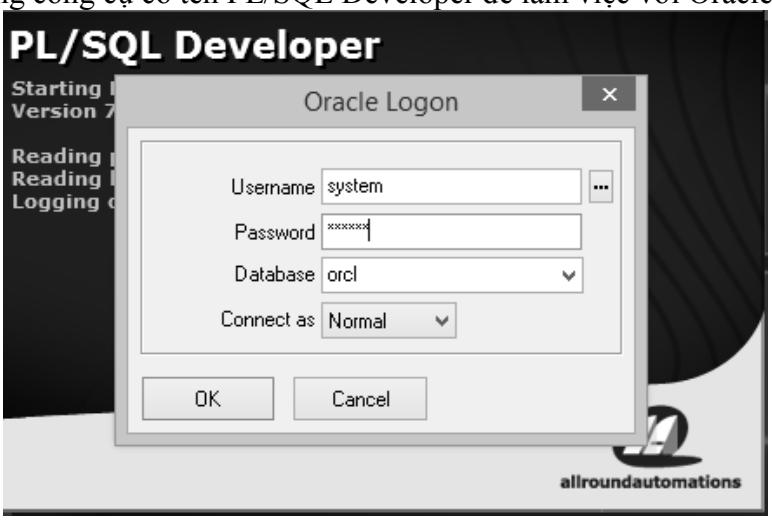
Hình 9.*Khởi động instance*

- Tắt service không cần thiết

Để tăng tốc cho hệ thống, ta chuyển Startup Type của service có tên bắt đầu là OracleDBConsole sang trạng thái Disabled.

❖ PL/SQL Developer

Khác với SQL Server hoặc MySQL có gói download mà khi cài đặt xong nó có sẵn công cụ trực quan để làm việc. Còn với Oracle, mặc định để làm việc với CSDL ta sử dụng Oracle Enterprise Manager (OEM) thông qua trình duyệt web, nhưng công cụ này khá nặng nề và cồng kềnh khi chạy trên các máy tính để bàn hoặc xách tay. Vì vậy ta sử dụng công cụ có tên PL/SQL Developer để làm việc với Oracle.



Hình 10.*Giao diện đăng nhập khi khởi động chương trình*

Thông tin đăng nhập bao gồm:

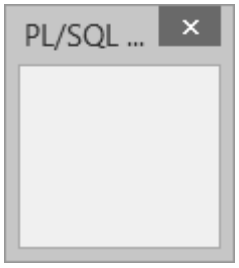
Username: tên schema/tên user (tên người dùng)

Password: mật khẩu tương ứng

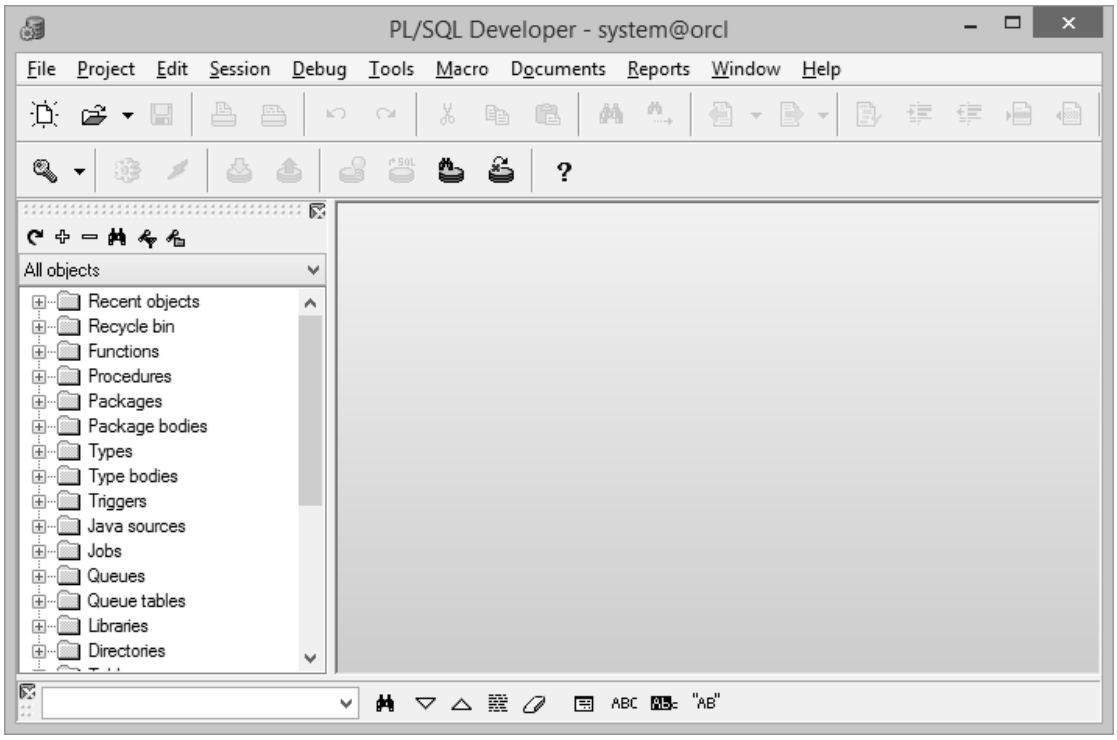
Database: tên cơ sở dữ liệu

Connect as: loại quyền kết nối đến database {Normal, SYSDBA, SYSOPER}

Chú ý: Phải chạy PL/SQL Developer với quyền admin, nếu không khi đăng nhập sẽ xảy ra lỗi như 11.



Hình 11.*Lỗi khi đăng nhập*



Hình 12.*Giao diện làm việc khi đăng nhập thành công*

Khi đăng nhập thành công, trên thanh tiêu đề của ứng dụng sẽ hiện tên user và tên database hiện đang được sử dụng. Ví dụ ở Hình 12 là: **system@orcl** tức là đã đăng nhập vào user **system** của cơ sở dữ liệu có tên **orcl**.

- Phiên làm việc

  Với mỗi một cửa sổ làm việc của PL/SQL Developer sẽ chỉ làm việc với một schema (hay còn gọi là một user) của một database trong một thời điểm.

  Một phiên làm việc trong PL/SQL Developer sẽ bắt đầu khi một user đăng nhập thành công và kết thúc khi một trong các sự kiện sau xảy ra:

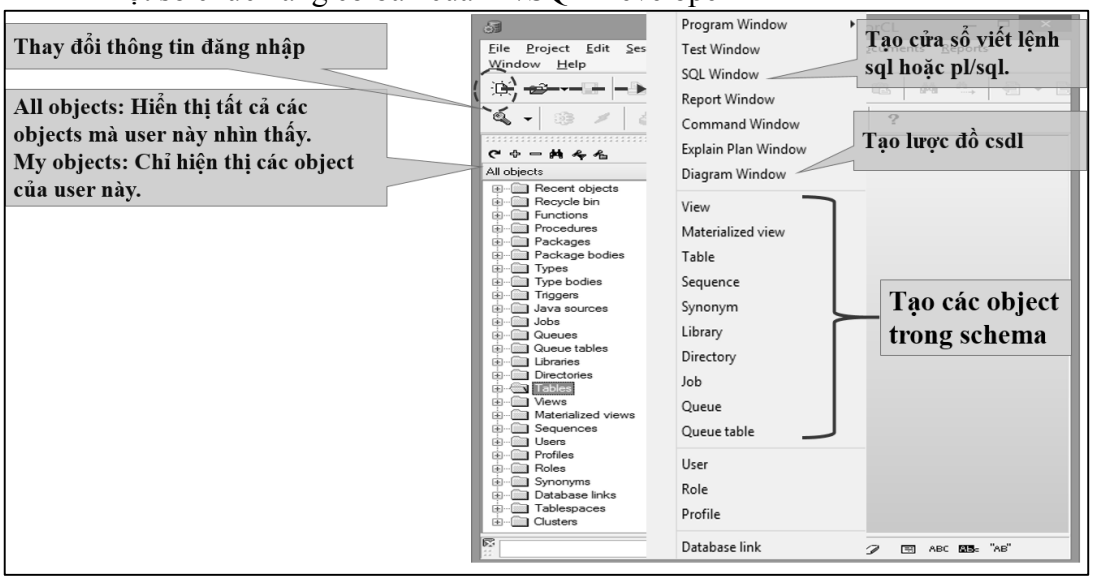
  - Một user khác đăng nhập

- Log off khỏi CSDL

- Tắt cửa sổ làm việc của PL/SQL Developer

PL/SQL Developer cho phép mở nhiều cửa sổ cùng lúc để có thể làm việc với nhiều schema cùng lúc.

- Một số chức năng cơ bản của PL/SQL Developer



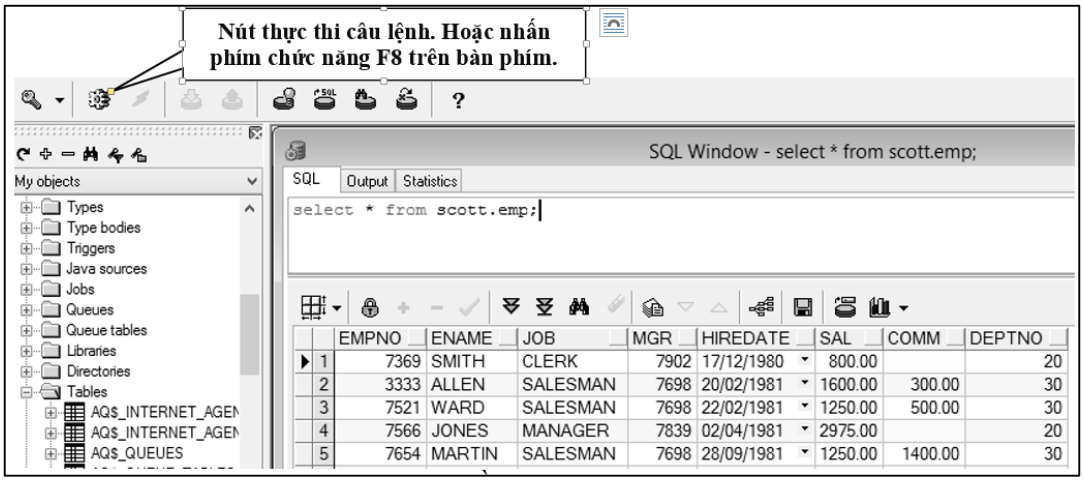
Hình 13.*Một số chức năng cơ bản*

➢ Thực thi câu lệnh SQL

Tạo một cửa sổ SQL Window để viết và thực thi các câu lệnh SQL.

VD: Truy vấn tất cả thông tin trong bảng emp của user scott.

Select * from scott.emp;



Hình 14.*Ví dụ về thực thi câu lệnh SQL*

❖ **Các lỗi thường gặp khi đăng nhập**

*** ORA-12541:** TNS:no listener

Nguyên nhân: service OracleOraDb10g_home1TNSListener chưa được start.

Khắc phục: start service này lên. Chờ khoảng 1 phút.

*** ORA-12514:** TNS:listener does not currently know of service requested in connect descriptor

Nguyên nhân: listener không tìm thấy service của database hoặc instance chưa được startup.

Khắc phục: start service OracleServiceORCL, chờ khoảng 1 phút để khởi động hết các tiến trình. Nếu vẫn không được thì đăng nhập vào sqlplus với quyền sysdba và startup database.

❖ **Bài tập thực hành**

Bài 1. Làm việc với CSDL thông qua PL/SQL Developer

a. Đăng nhập vào user system

b. Truy vấn tên các bảng được tạo trong user scott bằng câu lệnh:

HD: select table_name from dba_tables where owner='SCOTT';

c. Truy vấn thông tin trong bảng DEPT và EMP của user scott.

HD: select * from scott.dept; select * from scott.emp;

d. Hiển thị tên các nhân viên trong phòng ban có mã là 30.

HD: select ename from scott.emp where deptno=30;

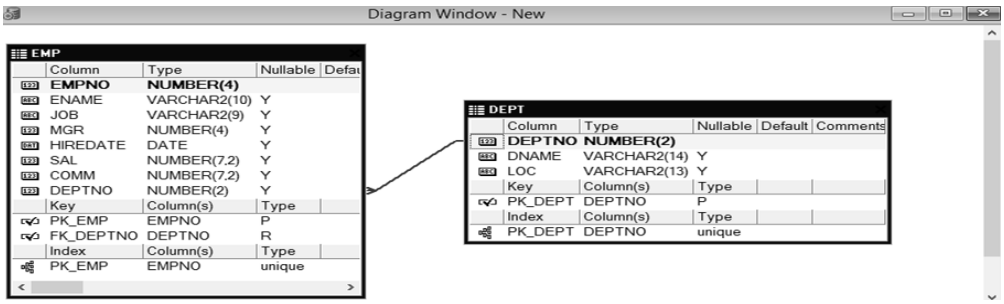
e. Đăng nhập vào user scott với mật khẩu là 123456 và đưa ra nhận xét.

f. Đăng nhập lại vào user system và tiến hành đổi mật khẩu của scott thành ***tiger*** và mở khóa nó.

HD: alter user scott identified by tiger account unlock;

g. Đăng nhập lại vào user scott với mật khẩu đã thay đổi ở câu f và thực hiện các truy vấn ở câu c,d.

h. Tạo một cửa sổ Diagram Window và kéo 2 bảng EMP và DEPT vào cửa sổ Diagram để xem mô hình kết nối giữa các 2 bảng.

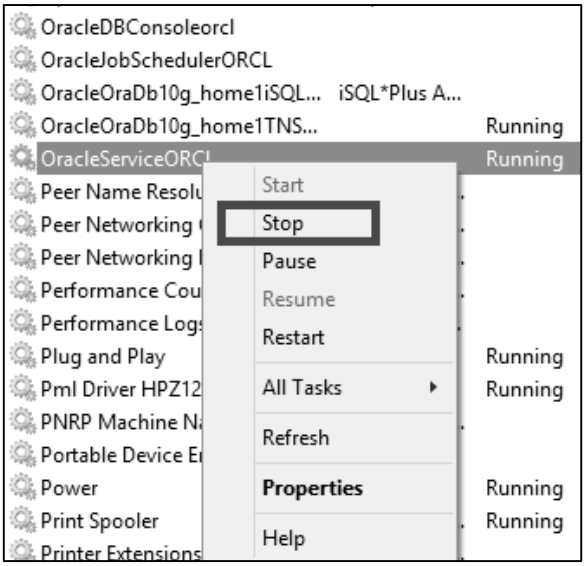


Bài 2. Tắt, bật CSDL

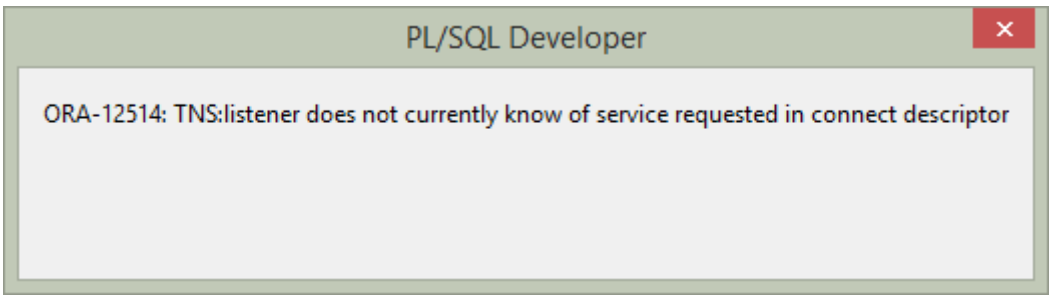
Hướng dẫn:

a. Tắt CSDL bằng cách: Stop service OracleServiceORCL. Sau đó thực hiện đăng nhập trên PL/SQL Developer và đưa ra nhận xét.

Để stop service, ta khởi động ứng dụng Services, tìm đến service có tên: OracleServiceORCL, chuột phải và chọn Stop.



- Sau khi stop service, thực hiện đăng nhập trên PL/SQL Developer, sẽ xuất hiện lỗi ORA-12154.



b. Bật CSDL bằng cách: Start service OracleServiceORCL. Đăng nhập lại trên PL/SQL Developer để kiểm tra.

Chương 1
NGÔN NGỮ PL/SQL

1.1. MỘT SỐ KHÁI NIỆM CƠ BẢN

1.1.1. Các khái niệm trong cơ sở dữ liệu

- Table là cấu trúc lưu trữ cơ bản nhất trong CSDL quan hệ (RDBMS), nó bao gồm 1 hoặc nhiều column và 0 hoặc nhiều row.

- Column hiển thị một loại dữ liệu trong bảng, ví dụ tên phòng ban trong bảng phòng ban. Người ta thể hiện nó thông qua tên column và giữ số liệu dưới các kiểu và kích cỡ nhất định.

- Row là tổ hợp những giá trị của Column trong bảng. Một row còn có thể được gọi là 1 bản ghi (record).

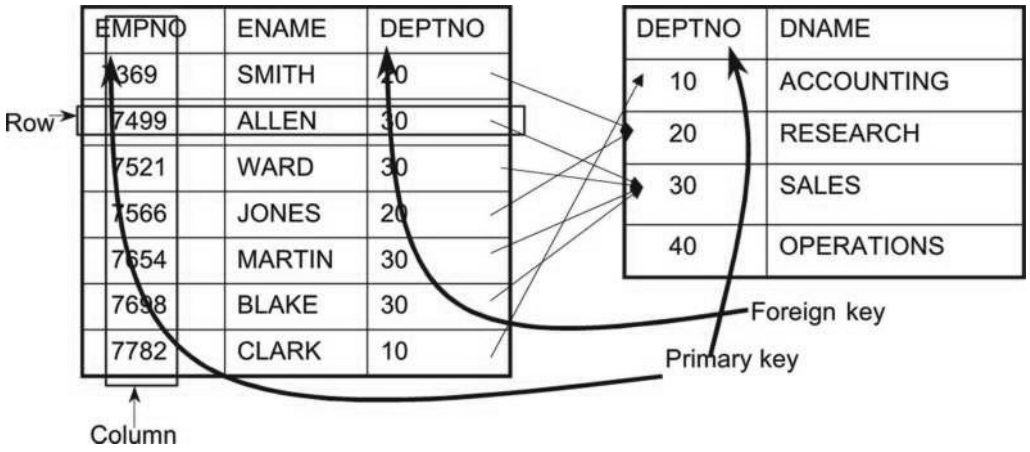
- Field là giao của column và row. Field chính là nơi chứa dữ liệu. Nếu không có dữ liệu trong field người ta nói field có giá trị là null.

- Primary Key là một column hoặc một tập các column xác định tính duy nhất của các row ở trong bảng. Ví dụ mã phòng ban. Primary Key nhất thiết phải có số liệu.

- Foreign Key là một column hoặc một tập các column tham chiếu một bảng khác hoặc tới chính bảng đó. Foreign Key xác định mối quan hệ giữa các bảng.

- Constraint là các ràng buộc dữ liệu, ví dụ Foreign Key, Primary Key...

Ví dụ:



Hình 15.*Minh họa các khái niệm trong CSDL*

Một số đối tượng khác trong CSDL:

- View là cấu trúc logic hiển thị dữ liệu từ một hoặc nhiều bảng

- Sequence dùng để sinh tự động các giá trị số tăng dần, thường dùng cho việc khóa chính tự động tăng. Index tăng tính thực thi của câu truy vấn.

- Synonym tên tương đương của đối tượng
- Program unit gồm Procedure, function, package...

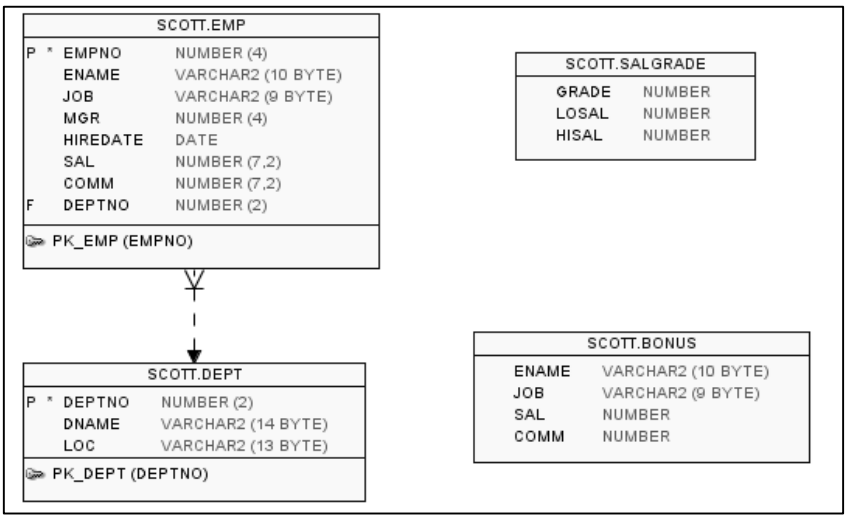
1.1.2. Các nhóm lệnh SQL cơ bản

Bảng 2.*Các nhóm lệnh SQL cơ bản*

SELECT	Là lệnh thông dụng nhất, dùng để lấy, xem dữ liệu trong CSDL
INSERT, UPDATE, DELETE	Là 3 lệnh dùng để nhập thêm những row mới, thay đổi nội dung dữ liệu trên các row hay xoá các row trong table. Những lệnh này được gọi là các lệnh thao tác dữ liệu DML (Data Manipulation Language)
CREATE, ALTER, DROP, RENAME, TRUNCATE	Là những lệnh dùng để thiết lập, thay đổi hay xoá bỏ cấu trúc dữ liệu như là table, view, index. Những lệnh này được gọi là các lệnh định nghĩa dữ liệu DDL (Data Definition Language)
COMMIT, ROLLBACK, SAVEPOINT	Quản lý việc thay đổi dữ liệu bằng các lệnh DML.
GRANT, REVOKE	2 lệnh này dùng để gán hoặc huỷ các quyền truy nhập vào CSDL Oracle và các cấu trúc bên trong nó. Những lệnh này được gọi là các lệnh điều khiển dữ liệu DCL (Data Control Language)

1.1.3. Truy vấn dữ liệu cơ bản

- Sơ đồ quan hệ cơ sở dữ liệu thực hành (Mặc định có sẵn trong user scott khi tạo cơ sở dữ liệu)



Hình 16.*Sơ đồ quan hệ cơ sở dữ liệu thực hành*

- **Mô tả dữ liệu**

Bảng 3.*Mô tả các bảng dữ liệu thực hành*

Tên	Kiểu	Khoá	Giải thích
DEPT			
DEPTNO	NUMBER(2) NOT NULL	PK	Mã phòng ban
DNAME	CHAR(14)		Tên phòng ban
LOC	CHAR(13)		Địa chỉ
SALGRADE			
GRADE	NUMBER	PK	Mức lương
LOSAL	NUMBER		Giá trị thấp
HISAL	NUMBER		Giá trị cao
EMP			
EMPNO	NUMBER(4) NOT NULL	PK	Mã nhân viên
ENAME	CHAR(10),		Tên nhân viên
JOB	CHAR(9),		Nghề nghiệp
MGR	NUMBER(4)	FK (EMP.EMPNO)	Mã người quản lý
HIREDATE	DATE		Ngày vào làm
SAL	NUMBER(7,2)		Lương
COMM	NUMBER(7,2)		Thưởng
DEPTNO	NUMBER(2) NOT NULL	FK(DEPT.DEPTNO)	Mã phòng ban

1.1.3.1. Truy vấn không điều kiện

+ Cú pháp:

SELECT [DISTINCT] {*, COLUMN [ALIAS], ...} FROM TABLE;

Với:

SELECT	Hiển thị nội dung của một hay nhiều cột
DISTINCT	Phân biệt nội dung giữa các dòng dữ liệu trả về
*	Lấy tất các các cột trong bảng
COLUMN	Tên cột dữ liệu cần trả về

ALIAS	phần tiêu đề của cột dữ liệu trả về
FROM TABLE	Tên bảng chứa dữ liệu truy vấn

+ Các thành phần khác có thể đưa vào mệnh đề SELECT trong câu lệnh truy vấn

- Biểu thức toán học
- Column alias
- Các column được ghép chuỗi
- Literal (Các chuỗi ký tự)
- Các hàm

Biểu thức toán học:

Trong mệnh đề SELECT biểu thức toán học có thể các giá trị (column hoặc hàng số), các toán tử, các hàm.

Các toán tử được dùng là (+), (-), (*), (/). Độ ưu tiên của các toán tử giống trong phần số học.

VD: SELECT ename, sal *12, comm FROM scott.emp;

Tiêu đề của cột (column alias):

Trong mệnh đề SELECT, column alias là phần nhãn hiển thị của column khi lấy số liệu ra. Trong column alias không được có dấu cách và viết cách sau tên column một dấu cách. Column alias được chấp nhận có dấu cách khi được đặt trong dấu nháy kép (“ ”).

VD: (ANUAL chính là column alias)

SELECT ename, SAL*12 ANUAL, comm FROM scott.emp;

Ghép tiếp các cột dữ liệu:

Toán tử ghép tiếp chuỗi (||) cho phép ghép tiếp dữ liệu trong các cột khác nhau của cùng một dòng dữ liệu với nhau thành một chuỗi.

Ta có thể có nhiều toán tử ghép chuỗi trong cùng một column alias.

VD: select empno||ename employee from scott.emp;

Ghép tiếp chuỗi ký tự

Trong mệnh đề SELECT, ta có thể thực hiện ghép tiếp bất kỳ ký tự nào, biểu thức hay số nào mà không phải là column hoặc column alias.

VD: select empno || ename || 'work in department' || deptno "employee detail" from scott.emp;

Hàm:

Trong mệnh đề select có thể chứa các hàm có sẵn hoặc do người dùng tạo ra.

VD: select sysdate from dual;--Hiển thị ngày giờ hệ thống

Chú ý: DUAL là bảng giả (dummy table) trong Oracle, bảng này chỉ có một trường và được sử dụng khi câu truy vấn không cần thiết tham chiếu đến một bảng thực trong cơ sở dữ liệu.

1.1.3.2. Truy vấn có điều kiện - Mệnh đề WHERE

Cú pháp:

SELECT [DISTINCT] {*, COLUMN [ALIAS],...} FROM TABLE [WHERE CONDITION (S)];

Mệnh đề WHERE dùng để đặt điều kiện trả về cho toàn bộ câu lệnh truy vấn. Trong mệnh đề WHERE có thể có các thành phần:

- Tên column
- Toán tử so sánh
- Tên column, hằng số hoặc danh sách các giá trị.

VD: select deptno, job, ename, sal from scott.emp where sal between 1000 and 2000;

1.1.3.3. Sắp xếp dữ liệu trả về - Mệnh đề ORDER BY

Cú pháp:

SELECT [DISTINCT] {*, COLUMN [ALIAS],...} FROM TABLE [WHERE CONDITION] **[ORDER BY EXPR/POSITION [DESC/ASC]];**

Mệnh đề ORDER BY dùng để sắp xếp số liệu được hiển thị và phải đặt ở vị trí sau cùng của câu lệnh truy vấn.

VD: select ename, job, sal*12, deptno from scott.emp order by ename;

Mệnh để ORDER BY mặc định sắp xếp theo thứ tự tăng dần ASC[ENDING]: số thấp trước, ngày nhỏ trước , ký tự theo bảng chữ cái.

Mệnh đề Order còn có thể sắp xếp nhiều column. Các column cần sắp xếp được viết thứ tự sau mệnh đề ORDER BY và cách bởi dấu phẩy (,). Column nào gần mệnh để ORDER BY hơn có mức độ ưu tiên khi sắp xếp cao hơn. Chỉ định cách thức sắp xếp ASC/DESC được viết sau column cách bởi một dấu cách.

VD: select deptno, job, ename, sal from scott.emp order by deptno, sal desc;

1.1.3.4. Nhóm dữ liệu trả về - Mệnh đề GROUP BY [HAVING]

Cú pháp:

SELECT [DISTINCT] {*, COLUMN [ALIAS],...} FROM TABLE [WHERE CONDITION] **[GROUP BY EXPR] [HAVING CONDITION]** [ORDER BY EXPR/POSITION [DESC/ASC]];

Mệnh đề GROUP BY sẽ nhóm các dòng dữ liệu có cùng giá trị của expr. Ví dụ GROUP BY JOB nghĩa là sẽ nhóm các nghề giống nhau. Thường dùng trong các bài toán thống kê như tính tổng, tìm lớn nhất, nhỏ nhất, trung bình… Mệnh đề GROUP

BY phải theo sau các điều kiện trong mệnh đề WHERE và phải đứng trước mệnh đề ORDER BY nếu được sử dụng.

VD: Hiển thị lương lớn nhất của từng chức vụ

select job, max(sal) from scott.emp group by job;

Mệnh đề HAVING là đặt điều kiện của nhóm dữ liệu. Có thể đặt ngay trước hoặc ngay sau mệnh đề GROUP BY. Mệnh đề này khác mệnh đề WHERE ở chỗ mệnh đề WHERE đặt điều kiện cho toàn bộ câu lệnh SELECT.

VD: select job, max(sal) from scott.emp group by job having max(sal)>=3000;--Hiển thị lương lớn nhất từ 3000 trở lên của từng chức vụ.

1.1.4. Truy vấn dữ liệu mở rộng

1.1.4.1. Kết hợp dữ liệu từ nhiều bảng

❖ *Mối liên kết tương đương (kết nối tự nhiên)*

Mối liên kết tương đương được thể hiện trong mệnh đề WHERE hoặc thông qua INNER JOIN trong mệnh đề FROM.

VD: Liệt kê các nhân viên có dname=RESEARCH

Cách 1: select emp.*, dname from scott.emp,scott.dept where emp.deptno = dept.deptno and dname='RESEARCH';

Cách 2: select emp.*, dname from scott.emp **inner join** scott.dept **on** emp.deptno=dept.deptno where dname='RESEARCH';--Có thể bỏ từ khóa inner

❖ *Kết nối trái(phải) OUTER JOIN (LEFT hoặc RIGHT):*

Nếu bảng A LEFT OUTER JOIN với bảng B thì kết quả gồm các bản ghi có trong bảng A, với các bản ghi không có mặt trong bảng B thì các cột từ B được điền NULL. Các bản ghi chỉ có trong B mà không có trong A sẽ không được trả về.

VD: select e.ename, d.deptno, d.dname from scott.dept d left join scott.emp e on d.deptno = e.deptno;

❖ *Liên kết của bảng với chính nó (tự thân)*

Có thể liên kết bảng với chính nó bằng cách đặt alias.

VD: Hiển thị thông tin bao gồm tên nhân viên, lương nhân viên, tên người quản lý của nhân viên đó, lương người quản lý đó với điều kiện lương của nhân viên lớn hơn lương người quản lý nhân viên đó.

Select e.ename ten_nhan_vien, e.sal luong, m.ename ten_quan_ly, m.sal luong_quan_ly from scott.emp e, scott.emp m where e.mgr = m.empno and e.sal > m.sal;

	TEN_NHAN_VIEN	LUONG	TEN_QUAN_LY	LUONG_QUAN_LY
1	SCOTT	3000	JONES	2975
2	FORD	3000	JONES	2975

Hình 17.*Kết quả trả về câu lệnh truy vấn*

1.1.4.2. Lệnh truy vấn lồng nhau

❖ *Câu lệnh SELECT lồng nhau*

+ Trong mệnh đề WHERE

VD: Tìm những nhân viên làm cùng nghề với BLAKE

```
Select ename, job from scott.emp where job = (select job from scott.emp where ename = 'BLAKE');
```

+ Trong mệnh đề HAVING

VD: Tìm những phòng có mức lương trung bình lớn hơn phòng 30.

```
Select dept.deptno, dname, avg(sal) from scott.emp, scott.dept where emp.deptno = dept.deptno group by dept.deptno, dname having avg(sal) > (select avg(sal) from scott.emp where deptno = 30);
```

❖ *Toán tử SOME/ANY/ALL/NOT IN/EXITS*

Bảng 4.*Mô tả các toán tử cơ bản*

TÊN TOÁN TỬ	DIỄN GIẢI
NOT IN / IN	Không thuộc / Thuộc
ANY và SOME	So sánh một giá trị với mỗi giá trị trong kết quả trả về của câu truy vấn con. Trả về đúng khi phép so sánh với bất kỳ giá trị nào là đúng.
ALL	So sánh một giá trị với mọi giá trị trong danh sách hay trong kết quả trả về của câu hỏi con. Trả về đúng khi phép so sánh với tất cả các giá trị đều đúng.
EXISTS	Trả về TRUE nếu có tồn tại.

VD: Hiển thị các nhân viên có lương lớn hơn các nhân viên ở phòng 30.

```
Select * from scott.emp where sal >= all (select distinct sal from scott.emp where deptno = 30) order by sal desc;
```

1.1.4.3. Các hàm xử lý dữ liệu

❖ *Các hàm xử lý chuỗi, ký tự*

+ CHR(number_code) : trả về ký tự theo mã ký tự, trong đó: number_code là mã ký tự trong bảng mã ASCII

VD: select CHR(116) from dual; => ‘t’

+ LENGTH*(<chuỗi>):* trả về độ dài chuỗi, nếu chuỗi là trống hoặc là NULL, thì hàm sẽ trả về NULL.

VD: select length('ORACLE') from dual; => 6

+ LOWER(*string1*): chuyển đổi tất cả các ký tự trong chuỗi *string1* thành ký tự thường.

VD: select LOWER('ORACLE') from dual; => oracle

+ UPPER(*string1*): ngược lại của LOWER

+ INSTR*(<chuỗi a>,<chuỗi con b>,<vị trí x>[,<i>]):* Trả về vị trí xuất hiện lần thứ i của chuỗi con b trong chuỗi a, bắt đầu tìm từ vị trí x. Nếu x < 0 thì tìm từ phải sang trái.

VD: SELECT INSTR('CORPORATE FLOOR', 'OR', 3, 2) FROM DUAL;

(=>14)

+ SUBSTR*(<chuỗi a>,<vị trí bắt đầu x> [số ký tự y,]*): Lấy y ký tự bắt đầu từ vị trí x của chuỗi a. Nếu x < 0 thì tìm từ phải sang trái. Nếu không có y sẽ lấy đến cuối chuỗi a.

VD: SELECT SUBSTR('ABCDEFG',3,4) "Substring" FROM DUAL;
(=>CDEF)

+ CONCAT(<chuỗi 1>,<chuỗi 2>) hay phép toán <chuỗi 1>||<chuỗi 2>: ghép chuỗi.

+ LTRIM (*<chuỗi a>,<chuỗi b>)* (hay RTRIM, TRIM): Cắt khỏi chuỗi a từ bên trái (hay từ bên phải, hay cả hai bên) những ky tự có trong chuỗi b.

VD: SELECT LTRIM('xyxXxyLAST WORD','xy') FROM DUAL;

(=> XxyLAST WORD)

- Các hàm số học:

+ ABS(n): Trị tuyệt đối của n

VD: SELECT ABS(-15) "Absolute" FROM DUAL;

Absolute

15

+ MOD(a,b): Lấy phần dư của a chia cho b

VD: SELECT MOD(11,4) "Modulus" FROM DUAL;

Modulus

3

+ ROUND(n,i): Làm tròn số *n* tới *i* chữ số thập phân.

+ POWER(n,i): lũy thừa *i* của *n*

- *Các hàm xử lý ngày tháng*

+ EXTRACT(YEAR | MONTH | DAY FROM <ngày>): Trả về thành phần, ngày, tháng, hoặc năm của một dữ liệu kiểu date.

VD: SELECT EXTRACT(YEAR FROM DATE '1998-03-07') FROM DUAL; (=>1998)

+ ADD_MONTHS(*<ngày x>,<số tháng n>):* Trả về ngày mới sau khi cộng n tháng vào ngày x.

VD: SELECT add_months(sysdate, 3) FROM DUAL;

+ MONTHS_BETWEEN(*<ngày 1>,<ngày 2>):* Số tháng giữa 2 ngày.

+ SYSDATE: Trả về ngày tháng hiện tại.

❖ *Các hàm chuyển đổi kiểu:*

+ TO_NUMBER (*<chuỗi số>*):Chuyển ký tự có nội dung số sang số

+ TO_CHAR*(<value>[, format_mask]):* chuyển đổi một giá trị số hoặc ngày tháng sang chuỗi. Trong đó, *value* là giá trị cần chuyển, *format_mask* là định dạng sẽ sử dụng để chuyển đổi.

VD: select TO_CHAR(1210.73, '9999.9') from dual;

=> 1210.7

select TO_CHAR(1210.73, '$9,999.00') from dual;

=> $1,210.73

select TO_CHAR(sysdate, 'yyyy/mm/dd') from dual;

+ TO_DATE(*<value> [, format_mask]*): chuyển đổi một giá trị số hoặc ngày tháng sang chuỗi.

Ví dụ: (Ngày hệ thống trong ví dụ là 7/5/2016)

SELECT To_char (sysdate, 'day, dd month yyyy') from dual;

Kết quả: saturday, 07 may 2016

SELECT To_char (sysdate, 'dd/MM/yyyy') from dual;

Kết quả: 07/05/2016

Hiển thị số ngày từ ngày 1/1/2016 đến 30/4/2016

SELECT To_date ('30/4/2016', 'dd/MM/yyyy')-to_date('1/1/2016', 'dd/MM/yyyy') from dual;

Kết quả: 120

Select extract(day from sysdate) from dual;

Kết quả: 7

Bảng 5.*Một số khuôn dạng ngày*

YYYY, YY	Năm, năm 2 ký tự số
YEAR	Chỉ năm theo cách phát âm của người anh;
Q	Quý trong năm
MM	Giá trị tháng với 2 số (01-12)
MONTH	Tên đầy đủ của tháng theo tiếng anh, độ dài 9
MON	Tháng với 3 ký tự viên tắt (JAN, FEB...)
ww, w	Tuần trong năm hoặc trong tháng
DDD, DD, D	Ngày trong năm, tháng hoặc tuần
DAY	Chỉ thứ trong tuần
DY	Chỉ thứ trong tuần với 3 ký tự viết tắt
"char"	Đoạn ký tự đặt trong nháy đúp được tự động thêm khi đặt trong khuôn dạng

1.1.5. Table và các lệnh SQL về table

1.1.5.1. Lệnh tạo table

Cú pháp tạo bảng:

CREATE TABLE TABLE_NAME (COLUMN DATATYPE [**DEFAULT** EXPR] [COLUMN_CONSTRAINT], .. , [TABLE_CONSTRAINT]) [**TABLESPACE** TABLESPACE_NAME] [**AS** SUBQUERY]

Trong đó:

TABLENAME	Tên bảng cần tạo
COLUMN DATATYPE	Tên column trong table Kiểu dữ liệu của column
DEFAULT EXPR	Giá trị mặc định của column trong trường hợp NULL là expr
COLUMN_CONSTRAINT	Ràng buộc của bản thân column, một cột có thể có nhiều ràng buộc, mỗi ràng buộc của một cột phân cách nhau bởi dấu cách.
TABLE_CONSTRAINT	ràng buộc của toàn bảng, mỗi ràng buộc cách nhau bởi dấu phẩy (,)
TABLESPACE TABLESPACE_NAME	TABLESAPCE lưu trữ bảng được tạo
AS SUBQUERY	tạo bảng có cấu trúc giống mệnh đề truy vấn

Các ví dụ tạo bảng dưới đây được thực hiện trong schema scott.

VD1:

create table empdemo(empno number not null constraint pk_empdeno primary key,ename varchar2(10) constraint nn_ename not null constraint upper_ename

check(ename = upper(ename)),job varchar2(9),

hiredate date default sysdate,

sal number(10,2) constraint ck_sal check(sal>500),

deptno number(2) constraint nn_deptno not null constraint fk_deptno2 references dept(deptno));

VD2: create table dept10 as select empno, ename, job, sal from emp where deptno = 10;

1.1.5.2. Một số quy tắc khi tạo table

a. Các quy tắc đặt tên object

(1) Tên dài từ 1 đến 30 ký tự, ngoại trừ tên CSDL không quá 8 ký tự và tên liên kết có thể dài đến 128 ký tự.

(2) Tên không chứa dấu nháy (").

(3) Không phân biệt chữ hoa chữ thường.

(4) Tên phải bắt đầu bằng ký tự chữ trong bộ ký tự của CSDL.

(5) Tên chỉ có thể chứa ký tự số trong tập ký tự của CSDL. Có thể dùng các ký tự_, $, #. ORACLE không khuyến khích dùng các ký tự $ và #.

(6) Tên không được trùng với các từ đã dùng bởi ORACLE.

(7) Tên không được cách khoảng trống.

(8) Tên có thể đặt trong cặp dấu nháy kép, khi đó tên có thể bao gồm các ký tự bất kỳ, có thể bao gồm khoảng trống, có thể dùng các từ khóa của ORACLE, phân biệt chữ hoa chữ thường.

(9) Tên phải duy nhất trong "không gian tên" nhất định. Các object thuộc cùng không gian tên phải có tên khác nhau.

b. Quy tắc khi tham chiếu đến Object

Sơ đồ chung khi tham chiếu các object hoặc thành phần của các object:

Schema.Object.Part.@dblink

Trong đó:

object: Tên object

schema: Schema chứa object

part: Thành phần của object

dblink: Tên CSDL chứa object

Tham chiếu đến các object không thuộc quyền sở hữu:

Để tham chiếu đến các object không thuộc schema hiện thời, phải chỉ ra tên của schema chứa object muốn truy cập: schema.object

Ví dụ: Để xóa table EMP trong schema SCOTT:

DROP TABLE scott.emp

Tham chiếu các object từ xa

Để truy cập đến một CSDL ở xa, sau tên object phải chỉ ra tên liên kết CSDL (database link) của CSDL chứa object muốn truy cập. Database link là một schema object, Oracle dùng để thâm nhập và truy xuất CSDL từ xa.

1.1.5.3. Các kiểu dữ liệu cơ bản

- **CHAR:** kiểu CHAR dùng để khai báo một chuỗi có chiều dài cố định, khi khai báo biến hoặc cột kiểu CHAR với chiều dài chỉ định thì tất cả các mục tin của biến hay cột này đều có cùng chiều dài được chỉ định. Các mục tin ngắn hơn Oracle sẽ tự động thêm vào các khoảng trống cho đủ chiều dài. Oracle không cho phép gán mục tin dài hơn chiều dài chỉ định đối với kiểu CHAR. Chiều dài tối đa cho phép của kiểu CHAR là 255 byte

- **VARCHAR2: k**iểu VARCHAR2 dùng để khai báo chuỗi ký tự với chiều dài thay đổi. Khi khai báo một biến hoặc cột kiểu VARCHAR2 phải chỉ ra chiều dài tối đa, các mục tin chứa trong biến hay cột kiểu VARCHAR2 có chiều dài thực sự là chiều dài của mục tin. Oracle không cho phép gán mục tin dài hơn chiều dài tối đa chỉ định đối với kiểu VARCHAR2. Chiều dài tối đa kiểu VARCHAR2 là 2000 byte.

- **NUMBER**: kiểu số của ORACLE dùng để chứa các mục tin dạng số dương, số âm, số với dấu chấm động.

NUMBER(p, s)

Trong đó:

p: số chữ số trước dấu chấm thập phân (precision), p từ 1 đến 38 chữ số

s: số các chữ số tính từ dấu chấm thập phân về bên phải (scale).

NUMBER(p): số có dấu chấm thập phân cố định với precision bằng p và scale bằng 0

NUMBER: số với dấu chấm động với precision bằng 38. Nhớ rằng scale không được áp dụng cho số với dấu chấm động.

DATE: Dùng để chứa dữ liệu ngày và thời gian. Mặc dù kiểu ngày và thời gian có thể được chứa trong kiểu CHAR và NUMBER.

Với giá trị kiểu DATE, những thông tin được lưu trữ gồm thế kỷ, năm, tháng, ngày, giờ, phút, giây. Oracle không cho phép gán giá trị kiểu ngày trực tiếp, để gán giá trị kiểu ngày, bạn phải dùng TO_DATE để chuyển giá trị kiểu chuỗi ký tự hoặc kiểu số. Nếu gán một giá trị kiểu ngày mà không chỉ thời gian thì thời gian mặc định là 12

giờ đêm. Nếu gán giá trị kiểu ngày mà không chỉ ra ngày, thì ngày mặc định là ngày đầu của tháng. Hàm SYSDATE cho biết ngày và thời gian hệ thống.

Tính toán đối với kiểu ngày

Đối với dữ liệu kiểu ngày, bạn có thể thực hiện các phép toán cộng và trừ.

Ví dụ:

SYSDATE+1 ngày hôm sau

SYSDATE-7 cách đây một tuần

SYSDATE+(10/1440) mười phút sau

1.1.5.4. Ràng buộc dữ liệu trong table

Các dạng constraint gồm:

- NULL/NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY (Referential Key)
- CHECK

a. NULL/NOT NULL

Là ràng buộc column trống hoặc không trống.

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));

b. 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)**);

c. 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 phải NOT 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

d. FOREIGN 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.

Từ khoá ON DELETE CASCADE được chỉ định trong dạng khoá này để chỉ khi dữ liệu cha bị xoá (trong bảng DEPT) thì dữ liệu con cũng tự động bị xoá theo (trong bảng EMP).

e. CHECK

Ràng buộc kiểm tra giá trị.

Ví dụ: create table emp_test (empno number primary key, ename varchar2(10) constraint nn_ename **constraint upper_ename check (ename = upper(ename)),** hiredate date default sysdate,sal number(10,2) **constraint ck_sal check(sal>500)**;

1.1.5.5. Chỉnh sửa cấu trúc table

Cú pháp:

ALTER TABLE tablename [ADD/MODIFY/DROP options ([column [column constraint) [ENABLE clause] [DISABLE clause]

Trong đó:

ADD: Thêm column hay constraint.

MODIFY: Sửa đổi kiểu các column

DROP: Bỏ constraint hoặc column.

ENABLE/DISABLE: Kích hoạt hoặc ngừng hoạt động các CONSTRAINT .

Chú ý:

- Khí dùng mệnh đề MODIFY không thể chuyển tính chất của COLUMN có nội dung là NULL chuyển thành NOT NULL;
- Không thể đưa thêm một cột NOT NULL nếu table đã có số liệu. Phải thêm cột NULL, điền đầy số liệu, sau đó chuyển thành NOT NULL.
- Không thể chuyển đổi kiểu khác nhau nếu column đã chứa số liệu
- Không thể dùng mệnh đề MODIFY để định nghĩa các CONSTRAINT trừ ràng buộc NULL/NOT NULL. Muốn sửa CONSTRAINT cần xoá chúng sau đó ADD thêm vào.

Ví dụ 1: Thêm một column có tên là short_name vào bảng emp:

alter table emp add short_name char(10);

Ví dụ 2: Chỉnh sửa kiểu dữ liệu của cột short_name từ char(10) sang nvarchar2(25)

alter table emp modify short_name nvarchar2(25);

Ví dụ 3: Thêm 1 ràng buộc unique: alter table emp add constraint unq_name unique(ename,short_name);

Ví dụ 4: Tạm ngừng hoạt động của ràng buộc unq_name:

alter table emp disable constraint unq_name;

1.1.5.6. Các lệnh DDL khác

- Xóa table

Cú pháp: DROP TABLE table_name [CASCADE CONSTRAINTS]

Trong đó: CASCADE CONSTRAINTS xóa tất cả các ràng buộc toàn vẹn liên quan đến table bị xóa.

Khi drop table thì:

- Xóa tất cả dữ liệu
- View và synonymliên quan vẫn còn nhưng không có giá trị
- Các giao dịch chưa giải quyết xong sẽ được commit
- Chỉ người tạo ra table hay DBA mới có thể xóa table

- Chú thích cho table

Dùng lệnh COMMENT để chú thích.

Ví dụ: COMMENT ON TABLE EMP IS ‘ THONG TIN NHAN VIEN’;

COMMENT ON COLUMN EMP.EMPNO IS ‘ MA SO NHAN VIEN’;

- Thay đổi tên object

Dùng lệnh RENAME để thay đổi tên object.

Cú pháp: RENAME old TO new

Trong đó: old Tên cũ new Tên mới

Ví dụ: RENAME emp TO employee

- Xóa dữ liệu của table

Dùng lệnh TRUNCATE TABLE để xóa dữ liệu của table, xóa tất cả các row trong table. Cú pháp: TRUNCATE TABLE table_name

1.1.6. Ngôn ngữ thủ tục PL/SQL

1.1.6.1. Tổng quan về PL/SQL

- PL/SQL là một ngôn ngữ lập trình với sự kết hợp giữa SQL và các cấu trúc điều khiển, các thủ tục (procedure), hàm (function), con trỏ (cursor), ngoại lệ (exception) và các lệnh giao tác.

- PL/SQL cho phép sử dụng tất cả lệnh thao tác dữ liệu gồm INSERT, DELETE, UPDATE và SELECT, COMMIT, ROLLBACK, SAVEPOINT, cấu trúc điều khiển như vòng lặp (for, while, loop), rẽ nhánh (if),…mà với SQL chúng ta không làm được.
- PL/SQL hỗ trợ cả lập trình hướng thủ tục và hướng đối tượng.
- Mục tiêu chính của PL/SQL là để:
  + Tăng thêm sức mạnh của ngôn ngữ SQL,
  + Xử lý kết quả của câu lệnh truy vấn trên từng dòng (dùng cursor),
  + Phát triển các chương trình ứng dụng trên CSDL dạng module,
  + Tái sử dụng những đoạn code (dùng procedure),
  + Giảm chi phí trong việc bảo trì và thay đổi ứng dụng.

1.1.6.2. Cấu trúc PL/SQL

Ngôn ngữ PL/SQL tổ chức các lệnh theo từng khối lệnh. Một khối lệnh PL/SQL cũng có thể có các khối lệnh con khác ở trong nó.

Cấu trúc đầy đủ của một khối lệnh PL/SQL bao gồm:

DECLARE /* Phần khai báo - Không bắt buộc */

Khai báo các biến sử dụng trong phần thân

BEGIN /* Phần thân */

Đoạn lệnh thực hiện;

EXCEPTION /* Phần xử lý lỗi - Không bắt buộc */

Xử lý lỗi xảy ra;

END;

Ví dụ: In ra màn hình dòng chữ “Hello, World!”

```
DECLARE
 message varchar2(20):= 'Hello, World!';
BEGIN
 dbms_output.put_line(message);
END;
```

1.1.6.3. Biến, hằng và nhập/xuất giá trị

- Khai báo biến:

```
mucluong NUMBER(5);
```

- Khai báo hằng:

```
heso CONSTANT NUMBER(3,2) := 1.86;
```

Ghi chú: Ký hiệu **:=** được sử dụng như là toán tử gán.

- Gán biến và biểu thức:

biến := biểu thức;

VD:

```
DECLARE
 a integer := 10;
 b integer := 20;
 c integer;
 f real;
BEGIN
 c := a + b;
 dbms_output.put_line('Value of c: ' || c);
 f := 70.0/3.0;
 dbms_output.put_line('Value of f: ' || f);
END;
```

- Phạm vi tác dụng của biến

PL/SQL cho phép sự lồng nhau của các khối lệnh, mỗi khối chương trình có thể chứa các khối lệnh bên trong. Nếu một biến được định nghĩa trong khối lệnh con, nó không có tác dụng ở bên ngoài khối lệnh đó. Nếu biến được định nghĩa ở khối lệnh ngoài, thì nó có tác dụng ở tất cả các khối con lồng trong nó.

Có 2 loại phạm vi biến:

+ Biến cục bộ (Local variable): Biến được định nghĩa trong1 khối lệnh con và không thể truy cập được khối lệnh ngoài.

+ Biến toàn cục (Global variable): Biến được định nghĩa ở khối lệnh ngoài cùng của chương trình hoặc của gói (package).

VD:

```
DECLARE
 -- Global variables
 num1 number := 95;
 num2 number := 85;
BEGIN
 dbms_output.put_line('Outer Variable num1: ' || num1);
```

```
dbms_output.put_line('Outer Variable num2: ' || num2);
DECLARE
-- Local variables
num1 number := 195;
num2 number := 185;
BEGIN
dbms_output.put_line('Inner Variable num1: ' || num1);
dbms_output.put_line('Inner Variable num2: ' || num2);
END;
END;
```

=> Kết quả:

Outer Variable num1: 95

Outer Variable num2: 85

Inner Variable num1: 195

Inner Variable num2: 185

- Gán giá trị trả về của câu lệnh truy vấn cho các biến:

Sử dụng mệnh đề SELECT INTO của SQL để gán giá trị cho các biến. Với mỗi trường giá trị trả về trong SELECT phải có một biến cùng kiểu dữ liệu với trường đó trong INTO.

VD:

```
DECLARE
 v_empno integer := 7788;
 v_ename emp.ename%type;/*Khai báo biến v_ename có kiểu dữ liệu giống của cột ename trong bảng emp.*/
 v_hiredate emp.hiredate%type;
 v_sal emp.sal%type;
BEGIN
 SELECT ename,hiredate,sal INTO v_ename, v_hiredate, v_sal
 FROM emp WHERE empno = v_empno;

 dbms_output.put_line
 ('Employer ' ||v_ename || ' - Hiredate: ' || v_hiredate || ' - Sal: ' || v_sal);
```

END;

Kết quả: Employer SCOTT - Hiredate: 19-APR-87 - Sal: 3000

- **Độ ưu tiên của toán tử:** ** (phép lũy thừa), NOT, *, /, +, -, || (phép nối chuỗi), =, <>, <=, >=, IS NULL, LIKE, BETWEEN, IN, AND, OR.
- Lệnh xuất một nội dung lên màn hình:

Cú pháp: DBMS_OUTPUT.PUT_LINE (chuỗi nội dung);

Ví dụ:

```
declare
x number(6) := 25;
begin
dbms_output.put_line('Gia tri x la: ' || x);
end;
```

Kết quả khi chạy : **Gia tri x la: 25**

- Lệnh nhập giá trị cho 1 biến

Biến thay thế &: dấu & đặt trước tên biến. Biến được yêu cầu nhập giá trị lúc thực thi câu SQL.

Lưu ý: biến kiểu chuỗi, kiểu ngày đặt trong cặp dấu nháy đơn ‘ ’

Ví dụ:

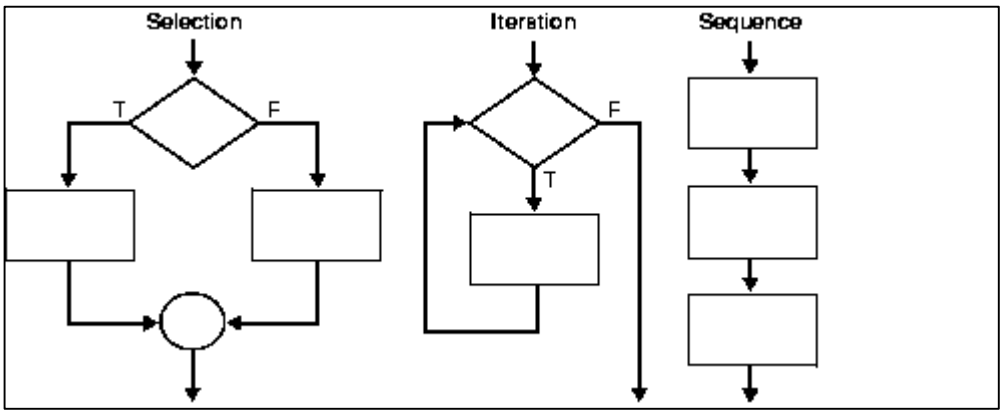
```
DECLARE
 x number;
BEGIN
 x := &x;
 dbms_output.put_line('Gia tri x = ' || x);
END;
```

Kí hiệu	Ý nghĩa
+, -, *, /	Phép tính cộng/trừ (âm)/nhân/chia
,	Phân cách cách thành phần
>,<,=,>=,<=	Các toán tử quan hệ (lớn hơn, nhỏ hơn, bằng, lớn hơn hoặc bằng, nhỏ hơn hoặc bằng)
<>,!=,^=	So sánh khác

;	Kết thúc một câu lệnh
:=	Toán tử gán
||	Toán tử nối chuỗi
**	Toán tử lũy thừa
/* */	Chú thích trên nhiều dòng
--	Chú thích trên 1 dòng
..	Toán tử danh sách

Bảng 6.Ý nghĩa một số ký hiệu

1.1.6.4. Các cấu trúc điều khiển trong PL/SQL



Hình 18.*Sơ đồ tổng quát các cấu trúc điều khiển*

a) Cấu trúc rẽ nhánh **IF**

- Mệnh đề **IF THEN**

Cú pháp:

```
IF condition THEN
 sequence_of_statements
END IF;
```

Ví dụ:

```
DECLARE
 no INTEGER(2) := 14;
BEGIN
 IF ( no = 14 ) THEN
```

```
 DBMS_OUTPUT.PUT_LINE('condition true');
 END IF;
END;
```

Result: condition true

- Mệnh đề IF THEN ELSE

```
IF ( condition ) THEN
  sequence_of_statements1
ELSE
  sequence_of_statements2
END IF;
```

Ví dụ:

```
DECLARE
 NO INTEGER(2) := 14;
BEGIN
 IF ( NO MOD 2 = 0 ) THEN
 DBMS_OUTPUT.PUT_LINE(NO || ' IS EVEN');
 ELSE
 DBMS_OUTPUT.PUT_LINE(NO || ' IS ODD');
 END IF;
END;
```

Result: 14 is even

- Mệnh đề IF THEN ELSIF

```
IF condition1 THEN
 sequence_of_statements1
ELSIF condition2 THEN
 sequence_of_statements2
ELSE
 sequence_of_statements3
```

```
END IF;
```

Ví dụ:

```
DECLARE
 grade char(1) := 'A';
BEGIN
 IF grade = 'A' THEN
 DBMS_OUTPUT.PUT_LINE('Excellent');
 ELSIF grade = 'B' THEN
 DBMS_OUTPUT.PUT_LINE('Very Good');
 ELSIF grade = 'C' THEN
 DBMS_OUTPUT.PUT_LINE('Good');
 ELSE
 DBMS_OUTPUT.PUT_LINE('no such grade');
 END IF;
END;
-----
Result: 'Excellent'
```

b) Cấu trúc rẽ nhánh **CASE**

Cú pháp 1: (Simple Case)

```
CASE selector
  WHEN value1
        THEN statement1;
  WHEN value2
        THEN statement2;
  ELSE
        statement3;
END CASE
```

Ý nghĩa: chương trình sẽ kiểm tra lần lượt từ trên xuống dưới trong các mệnh đề WHEN, khi gặp 1 giá trị value bằng selector thì sẽ thực hiện đoạn lệnh trong mệnh đề WHEN đó và bỏ qua các trường hợp bên dưới, khi tất cả các mệnh đề WHEN đều không đúng thì đoạn lệnh trong mệnh đề ELSE sẽ được thực hiện.

Ví dụ:

```
DECLARE
 a number := 7;
BEGIN
 CASE a
 WHEN 1 THEN
 DBMS_OUTPUT.PUT_LINE('value 1');
 WHEN 2 THEN
 DBMS_OUTPUT.PUT_LINE('value 2');
 WHEN 3 THEN
 DBMS_OUTPUT.PUT_LINE('value 3');
 ELSE
 DBMS_OUTPUT.PUT_LINE('no matching CASE found');
 END CASE;
END;
```

Result: no matching CASE found

Cú pháp 2: (Search Case)

```
CASE
  WHEN condition1 THEN
        statement1;
  WHEN condition2 THEN
        statement2;
  ELSE
        statement3;
END CASE;
```

Ý nghĩa: chương trình sẽ kiểm tra lần lượt từ trên xuống dưới trong các mệnh đề WHEN, khi gặp 1 biểu thức điều kiện đúng thì đoạn lệnh trong mệnh đề WHEN đó được sẽ thực hiện và bỏ qua các trường hợp bên dưới, khi tất cả các mệnh đề WHEN biểu thức điều kiện đều sai thì đoạn lệnh trong mệnh đề ELSE sẽ được thực hiện.

Ví dụ:

```
DECLARE
  a number := 3;
```

```
BEGIN
  CASE
        WHEN a = 1 THEN
 DBMS_OUTPUT.PUT_LINE('value 1');
 WHEN a = 2 THEN
 DBMS_OUTPUT.PUT_LINE('value 2');
 WHEN a = 3 THEN
 DBMS_OUTPUT.PUT_LINE('value 3');
 ELSE
 DBMS_OUTPUT.PUT_LINE('no matching CASE found');
 END CASE;
END;
```

Result: `value 3`

c) Cấu trúc lặp không định trước **LOOP .. EXIT WHEN**

Cú pháp:

```
LOOP
 Statements;
EXIT WHEN condition;
END LOOP;
```

Ý nghĩa: Đoạn lệnh `Statements` sẽ được thực hiện lặp đi lặp lại cho đến khi điều kiện thoát `condition` là đúng thì thoát khỏi vòng lặp.

EXIT WHEN được dùng để chỉ định điều kiện thoát khỏi vòng lặp cho tất cả các dạng lặp.

Ví dụ:

```
DECLARE
 no NUMBER := 5;
BEGIN
 LOOP
 DBMS_OUTPUT.PUT_LINE('Inside value: no = ' || no);
 no := no - 1;
```

```
 EXIT WHEN no = 0;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Outside loop end');
END;
```

Result: Inside value: no = 5

Inside value: no = 4

Inside value: no = 3

Inside value: no = 2

Inside value: no = 1

Outside loop end

d) Cấu trúc lặp không định trước **WHILE LOOP**

Cú pháp:

```
WHILE condition LOOP
 statement(s);
END LOOP;
```

Ý nghĩa: Khi gặp vòng lặp While, chương trình sẽ kiểm tra điều kiện `condition` , nếu đúng thì thực hiện đoạn lệnh trong while, sai thoát khỏi while.

Ví dụ: Hiển thị các số chính phương nhỏ hơn 1000

```
DECLARE
 i NUMBER := 0;
BEGIN
 WHILE i * i < 1000 LOOP
 DBMS_OUTPUT.PUT_LINE(i * i);
 i := i + 1;
 END LOOP;
END;
```

e) Cấu trúc lặp xác định **FOR … LOOP**

Cú pháp:

```
FOR current IN [ REVERSE ] lower_value..upper_value LOOP
 statement(s);
END LOOP;
```

Ý nghĩa: Khi gặp vòng lặp For, chương trình sẽ gán giá trị cho biến `current` lần lượt từ `lower_value` đến `upper_value` (nếu có từ khóa `reverse` thì sẽ gán ngược lại từ `upper_value` đến `lower_value`), sau mỗi lần gán sẽ thực hiện các lệnh trong LOOP 1 lần.

Chú ý:

- lower_value <= upper_value
- Biến `current` chỉ là biến duyệt giá trị, không cần phải định nghĩa trước, chỉ có tác dụng trong vòng for, không cho phép thay đổi giá trị bởi câu lệnh gán.

Ví dụ 1: Tổng các số từ 1 đến 9

```
DECLARE
 i NUMBER := 0;
 tong number := 0;
BEGIN
 FOR i IN 1 .. 9 LOOP
 tong := tong + i;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Sum :' || tong);
END;
```

Result: Sum :45

Cách dùng đặc biệt: có thể sử dụng vòng lặp for để duyệt các dòng trả về của câu lệnh select.

Ví dụ 2: Kiểm tra xem có nhân viên có mã ***xyz*** (xyz nhập từ bàn phím) trong công ty không, nếu có thì hiển thị thông tin nhân viên, nếu không thì hiển thị dòng chữ "Không có nhân viên có mã ***xyz*** trong công ty".

```
declare
 i number;
 checkExist boolean := false;
begin
 i := &input_empno;--Nhap ma nhan vien tu ban phim
 for j in (select * from emp where empno = i) loop
 dbms_output.put_line(j.empno || ' ' || j.ename || ' ' || j.job);
 checkExist := true;
```

```
 end loop;
 if not (checkExist) then
 dbms_output.put_line('Khong co nhan vien co ma ' || i ||' trong cong ty.');
 end if;
end;
```

1.1.6.5. Xử lý các ngoại lệ (Exception)

Khi một lỗi phát sinh, một ngoại lệ được đưa ra, việc thực hiện chương trình bình thường bị dừng lại và điều khiển được chuyển tới ***khối lệnh chứa phần xử lý ngoại lệ tương ứng.*** Nếu không tìm thấy phần xử lý ngoại lệ tương ứng với lỗi đó, chương trình sẽ bị dừng đột ngột và hiện thông báo lỗi lên màn hình.

❖ Có 2 dạng ngoại lệ (exception) :

- **Ngoại lệ không tường minh (implicit):** là những ngoại lệ được định nghĩa sẵn và sinh ra một cách tự động. VD: Nếu chia một số cho 0, ngoại lệ ZERO_DIVIDE sẽ tự động sinh ra.
- **Ngoại lệ tường minh (explicit):** là ngoại lệ do người dùng định nghĩa, và được sinh ra bằng cách sử dụng câu lệnh **RAISE**

Cú pháp:

```
DECLARE
  declaration statement(s);
BEGIN
  statement(s);
EXCEPTION
  WHEN built-in_exception_name_1 THEN
        statement(s);
  WHEN built-in_exception_name_2 THEN
        statement(s);
  ……
[WHEN OTHERS THEN
        statement(s);]
END;
```

Ví dụ 1: Nhập 2 số từ bàn phím, tính tổng, hiệu, tích, thương 2 số đó.

```
declare
 a number;
 b number;
begin
 a := &number1;
 b := &number2;
 dbms_output.put_line(a || '+' || b || '=' || (a + b));
 dbms_output.put_line(a || '-' || b || '=' || (a - b));
 dbms_output.put_line(a || '*' || b || '=' || (a * b));
 dbms_output.put_line(a || '/' || b || '=' || (a / b));
exception
 when zero_divide then
 dbms_output.put_line('Error divide by 0');
end;
```

Ví dụ 2: Kiểm tra sự tồn tại nhân viên trong công ty.

```
declare
 i number;
 trung_ma exception;-- Khai bao 1 ngoai le moi.
begin
 i := &input_empno; --Nhap ma nhan vien tu ban phim
 for j in (select empno from emp) loop
 if j.empno = i then
 raise trung_ma;
 end if;
 end loop;
exception
 when trung_ma then
 dbms_output.put_line('Da co nhan vien nay');
end;
```

1.1.6.6. Các kiểu dữ liệu cơ bản của PL/SQL

❖ *Kiểu dữ liệu một cột (%type)*

Là kiểu dữ liệu lưu trữ các giá trị chưa biết kiểu của một cột trong một bảng.

Cú pháp: tên_biến tên_bảng.tên_cột%type

Ví dụ: khai báo biến v_Manv có cùng kiểu dữ liệu với cột Manv trong bảng NHANVIEN

v_Manv NHANVIEN.Manv%TYPE;

Khai báo có điểm thuận lợi là: kiểu dữ liệu chính xác của biến v_Manv không cần được biết, nếu định nghĩa của cột Manv trong bảng NHANVIEN bị thay đổi thì kiểu dữ liệu của biến v_Manv thay đổi tương ứng.

❖ *Kiểu dữ liệu một dòng (%Rowtype)*

Kiểu dữ liệu này được sử dụng để lưu trữ các giá trị chưa biết kiểu dữ liệu trong tất cả các cột trong một bảng.

Cú pháp: v_Variable_name Table_Name%Rowtype;

Ví dụ: khai báo biến **v_nv** có kiểu dữ liệu là một dòng trong bảng **NHANVIEN**

v_nv NHANVIEN%ROWTYPE;

Khi truy xuất đến từng cột ta sử dụng giống như một bảng dữ liệu (trong trường hợp này chỉ gồm 1 record) tham chiếu đến một cột.

Cú pháp: Tên-biến.Tên-cột

Ví dụ: v_nv.HoTen

❖ *Kiểu dữ liệu Record*

Đây là kiểu dữ liệu do người dùng định nghĩa, nó có cấu trúc là gồm các biến có kiểu dữ liệu đã có.

Cú pháp:

```
TYPE Ten_kieu_Record IS
 RECORD (
 Col1 Kieu_Du_Lieu1,
 Col2 Kieu_Du_Lieu2,
 ...
 );


-- Khai báo biến sử dụng kiểu dữ liệu trên:
Ten_Bien Ten_kieu_Record;
```

Ví dụ:

```
declare
type diem is record(
 x float,
 y float
);
A diem;
B diem;
kc float;
begin
A.x:=1;A.y:=1;
B.x:=1;B.y:=3;
kc := sqrt((A.x-B.x)**2+(A.y-B.y)**2);
dbms_output.put_line('Khoang cach tu A den B la: '||kc);
        end;
```

❖ *Kiểu dữ liệu Table*

Kiểu dữ liệu TABLE cho phép định nghĩa ra một kiểu dữ liệu mới, lưu trữ được nhiều phần tử.

Các đặc điểm của kiểu TABLE:

- Kiểu dữ liệu TABLE tương tự kiểu mảng của ngôn ngữ lập trình có cấu trúc, nhưng có số phần tử không giới hạn.
- Chỉ số của kiểu TABLE không nhất thiết liên tục. Ví dụ TABLE có 3 phần tử tại chỉ số 1, 3, 5.

Cú pháp:

```
TYPE <Table_Name>
IS  TABLE  OF  <Data_Type> INDEX  BY  BINARY_INTEGER;
```

Ví dụ: Định nghĩa một kiểu TABLE chứa các phần tử kiểu Varchar2(50)

```
Declare
 -- Định nghĩa một kiểu TABLE.
 Type My_Tbl Is Table Of Varchar2(50) Index By Binary_Integer;
 -- Khai báo một biến sử dụng kiểu dữ liệu khai báo ở trên.
```

```
 v_Emps My_Tbl;
Begin
 v_Emps(1) := 'One';
 v_Emps(2) := 'Two';
 v_Emps(3) := 'Three';
 Dbms_Output.Put_Line('Element Count = '||v_Emps.Count);
 For i In v_Emps.First .. v_Emps.Last Loop
 Dbms_Output.Put_Line('Element at ' || i || ' = ' || v_Emps(i));
 End Loop;
End;
----
Result:
  Element Count = 3
Element at 1 = One
Element at 2 = Two
Element at 3 = Three
```

Bảng 7.*Các hàm của kiểu TABLE*

Tên hàm/Thuộc tính	**Ý nghĩa**	**Ví dụ sử dụng**
• DELETE	Xóa các dòng trong bảng	v_tbl.delete(3);
• EXISTS	Trả về TRUE nếu tồn tại phần tử chỉ định trong Table.	v_e:= v_tbl.exists(3);
• COUNT	Trả về số lượng phần tử trong table.	v_count:=v_tbl.count;
• FIRST	Trả về chỉ số của phần tử đầu tiên trong table.	v_first_idx:=v_tbl.first;
• LAST	Trả về chỉ số phần tử cuối cùng trong table.	v_last_idx:=v_tbl.last;

• NEXT	Trả về chỉ số của phần tử tiếp theo trong bảng so với chỉ số được chỉ định.	v_idx:= v_tbl.next(2);
• PRIOR	Trả về chỉ số phần tử đứng trước so với phần tử được chỉ định.	v_idx:=v_tbl.prior(2);

1.1.6.7. Con trỏ (Cursor)

- Cursor là kiểu biến có cấu trúc, cho phép ta xử lý dữ liệu gồm nhiều dòng. Số dòng phụ thuộc vào câu lệnh truy vấn dữ liệu sau nó.Trong quá trình xử lý, ta thao tác với cursor thông qua từng dòng dữ liệu. Dòng dữ liệu này được định vị bởi một con trỏ. Với việc dịch chuyển con trỏ, ta có thể lấy được toàn bộ dữ liệu của một dòng hiện tại.

- Để xử lý một câu SQL, PL/SQL mở một vùng làm việc có tên là vùng ngữ cảnh (context area). PL/SQL sử dụng vùng này để thi hành câu SQL và chứa kết quả trả về. Vùng ngữ cảnh đó là phạm vi hoạt động của con trỏ.

- Có hai loại con trỏ:

+ Con trỏ được khai báo tường minh (explicit cursor)

+ Con trỏ không được khai báo tường minh (implicit cursor) (hay còn gọi là con trỏ tiềm ẩn).

❖ Con trỏ tiềm ẩn

Một lệnh SQL được xử lý bởi Oracle và không được đặt tên bởi người sử dụng. Các lệnh SQL được thực hiện trong một con trỏ tiềm ẩn bao gồm SELECT .. INTO, UPDATE, INSERT, DELETE.

Có bốn thuộc tính:

- SQL%NOTFOUND: kết quả trả về là TRUE nếu câu lệnh SQL không tìm thấy dữ liệu
- SQL%FOUND: kết quả trả về là TRUE nếu câu lệnh SQL tìm thấy dữ liệu
- SQL%ROWCOUNT: kết quả trả về là số dòng dữ liệu mà câu lệnh SQL tìm thấy
- SQL%ISOPEN: kết quả trả về là TRUE nếu con trỏ đang ở trạng thái mở

Trước khi thi hành câu SQL, các thuộc tính của con trỏ tiềm ẩn có giá trị NULL.

Ví dụ 1: thuộc tính %NOTFOUND

```
DELETE FROM emp WHERE empno='222';
IF SQL%NOTFOUND THEN
 DBMS_OUTPUT.PUT_LINE ('Ko co nhan vien 222');
```

```
END IF;
```
Ví dụ 2: thuộc tính %FOUND
```
SELECT empno into v_eno FROM EMP WHERE empno=7788;
IF SQL%FOUND THEN
 DELETE FROM EMP WHERE empno=7788;
END IF;
```
Ví dụ 3: thuộc tính %ROWCOUNT
```
 UPDATE EMP SET SAL=5000 WHERE empno=7788;
 IF SQL%ROWCOUNT >0 THEN
 DBMS_OUTPUT.PUT_LINE (‘Luong moi’);
 END IF;
```
Ví dụ 4: Thủ tục tăng lương một nhân viên có mã truyền vào từ tham số.
```
CREATE OR REPLACE Procedure Tang_Luong(manv number) As
 old_luong number;
 new_luong number;
Begin
 select sal into old_luong from emp where empno=manv;
 if SQL%FOUND then
 new_luong:=old_luong+old_luong*10/100;
 update emp set sal=new_luong where empno=manv;
 if SQL%ROWCOUNT<>0 then
 DBMS_OUTPUT.PUT_LINE('Luong NV ' || manv ||' duoc tang 10%');
 end if;
 end if;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
 DBMS_OUTPUT.PUT_LINE('Khong tim thay nhan vien ' || manv);
END;
```

- ❖ Con trỏ tường minh

Là con trỏ được đặt tên bởi người sử dụng (câu SELECT được đặt tên).

Cú pháp: CURSOR tên-cursor IS câu-lệnh-SELECT;

- Trong đó, câu lệnh SELECT phải chỉ ra các cột cụ thể cần lấy cho con trỏ này.
- Phần khai báo này phải được đặt trong vùng khai báo biến (trước BEGIN của khối (Block)).
- Trong ngôn ngữ thủ tục PLSQL, để xử lý dữ liệu lưu trong cơ sở dữ liệu, đầu tiên dữ liệu cần được ghi vào các biến. Giá trị trong biến có thể được thao tác. Dữ liệu các bảng không thể được tham khảo trực tiếp.

Ví dụ: cursor c_nv is select empno,sal from emp;

- Các bước khai báo và sử dụng con trỏ tường minh:

- OPEN tên-cursor; /*Mở con trỏ thi hành câu truy vấn*/
- FETCH tên-cursor INTO biến1, biến2, …, biếnn;

hoặc FETCH tên-cursor INTO biến_có_kiểu_record; /*Lệnh FETCH dùng để gọi một dòng trong tập dữ liệu của con trỏ, có thể được lặp để gọi tất cả các dòng của con trỏ*/.

- CLOSE tên-cursor /*đóng con trỏ, giải phóng khỏi bộ nhớ*/

- Mọi con trỏ khai báo tường minh đều có bốn thuộc tính:

%NOTFOUND, %FOUND, %ROWCOUNT, %ISOPEN

Các thuộc tính này được thêm vào sau phần tên của con trỏ.

a) Thuộc tính %NOTFOUND (đi kèm lệnh Fetch)

Mang giá trị TRUE hoặc FALSE. %NOTFOUND bằng TRUE khi đã fetch đến dòng cuối cùng của con trỏ, ngược lại, bằng FALSE khi lệnh fetch trả về ít nhất một dòng hoặc chưa fetch đến dòng cuối cùng.

Ví dụ:

```
OPEN cur_first;
LOOP
FETCH cur_first INTO v_empno,v_sal;
EXIT WHEN cur_first%NOTFOUND;
END LOOP;
```

b) Thuộc tính %FOUND (đi kèm lệnh Fetch)

Ngược với thuộc tính %NOTFOUND.

Ví dụ:

```
OPEN cur_first;
LOOP
FETCH cur_first INTO v_empno,v_sal;
```

IF cur_first%FOUND THEN

……

ELSE

CLOSE cur_first;

EXIT;

END IF;

END LOOP;

c) Thuộc tính %ROWCOUNT (đi kèm lệnh Fetch)

Trả về số dòng con trỏ đã được FETCH.

Ví dụ:

OPEN cur_first;

LOOP

FETCH cur_first INTO v_empno,v_sal;

IF cur_first%ROWCOUNT=1000 THEN

EXIT;

END IF;

END LOOP;

d) Duyệt con trỏ tường minh sử dụng câu lệnh FOR .. LOOP

Cú pháp:

FOR tên_biến **IN** tên_cursor **LOOP**

Các câu lệnh;

END LOOP;

Ví dụ:

declare

cursor c_emp is select * from emp;/*Khai báo 1 con trỏ trả về tất cả các bản ghi của bảng emp*/

v_emp c_emp%rowtype; /*Khai báo 1 biến có kiểu dữ liệu là từng record của con trỏ c_emp*/

begin

for v_emp in c_emp loop/*Duyệt từng bản ghi trong con trỏ c_emp và lưu vào biến v_emp*/

dbms_output.put_line('Ma NV: ' || v_emp.empno || ' Ten NV: ' || v_emp.ename);

/*In mã và tên của từng nhân viên duyệt được*/

end loop;

end;

1.1.6.8. Hàm (Function)

Hàm (function) là nhóm các lệnh **PL/SQL** thực hiện chức năng nào đó. Hàm sẽ trả về một giá trị ngay tại lời gọi của nó.

- Cú pháp:

```
-- function_name: Tên hàm
-- argument: Tên tham số
-- mode: Loại tham số: IN hoặc OUT hoặc IN OUT, mặc định là IN
-- datatype: Kiểu dữ liệu của tham số

CREATE  [OR  REPLACE] FUNCTION  <function_name>
 [
 (argument1 [mode1] datatype1,
 argument2 [mode2] datatype2,
 ...)
 ]
 RETURN  datatype
 IS  | AS
BEGIN
 -- PL/SQL Block;
END;
```

Ví dụ 1: hàm trả về tổng 2 số truyền vào từ 2 tham số

```
CREATE OR REPLACE FUNCTION Sum(a Integer, b Integer)
RETURN Integer
AS
Begin
 return a + b;
End;
```

Ví dụ 2: hàm không tham số trả về thời gian hiện tại

```
CREATE OR REPLACE FUNCTION Get_Current_Datetime
RETURN Date
AS
Begin
 return sysdate;
End;
```

- Hủy Function

```
DROP FUNCTION <function_name>;
```

- Gọi hàm.

```
-- Khi gọi hàm phải khai báo một biến trả về
-- Khai báo một biến c.
c Integer;
...
-- Gọi hàm.
c := Sum(10, 100);
```

1.1.6.9. Thủ tục (Procedure)

Một nhóm các lệnh thực hiện chức năng nào đó có thể được gom lại trong một thủ tục (procedure) nhằm làm tăng khả năng xử lý, khả năng sử dụng chung, tăng tính bảo mật và an toàn dữ liệu,tiện ích trong phát triển.

Thủ tục có thể được lưu giữ ngay trong database như một đối tượng của database, sẵn sàng cho việc tái sử dụng. Thủ tục lúc này được gọi làStore procedure. Với các Store procedure, ngay khi lưu giữ Store procedure, chúng đã được biên dịch thành dạng pcode vì thế có thể nâng cao khả năng thực hiện.

Thủ tục không trả về giá trị trực tiếp như hàm.

- Cú pháp:

```
-- procedure_name: Tên thủ tục
-- argument: Tên tham số
-- mode: Loại tham số: IN hoặc OUT hoặc IN OUT, mặc định là IN
-- datatype: Kiểu dữ liệu của tham số

CREATE  [OR  REPLACE] PROCEDURE  <procedure_name>
```

```
[
(argument1 [mode1] datatype1,
argument2 [mode2] datatype2,
...)
]
IS  | AS
BEGIN
 -- PL/SQL Block;
END;
```

Loại tham số:

- IN: Tham số chỉ dùng để truyền giá trị từ ngoài vào trong thủ tục.
- OUT: Tham số chỉ dùng để truyền giá trị từ trong thủ tục ra ngoài.
- IN OUT: Tham số là kết hợp của cả IN và OUT.

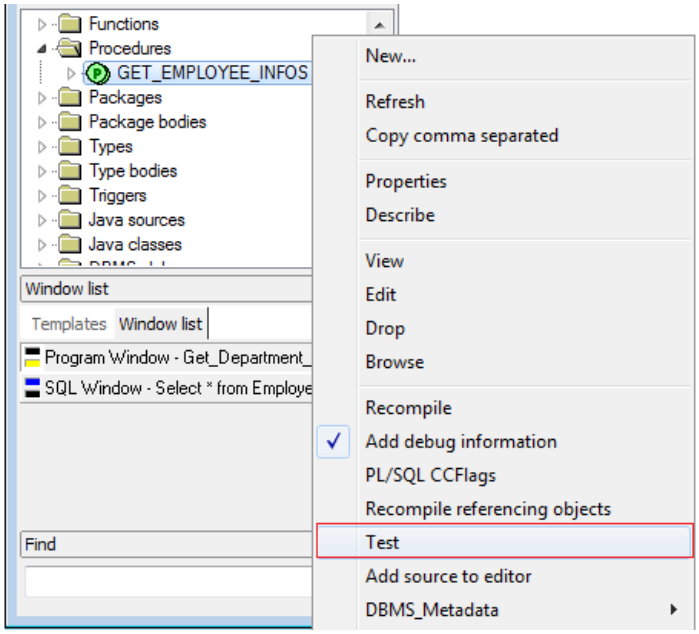
Ví dụ: Thủ tục truyền vào empno (mã nhân viên) và hiển thị lên màn hình thông tin nhân viên bao gồm tên, lương tương ứng với mã truyền vào trong bảng EMP.

```
Create Or Replace Procedure Get_Employee_Infos(p_Empno Number) Is
 v_Ename varchar2(100);
 v_Sal number;
Begin
 -- Nếu câu lệnh Select này nếu không có bản ghi nào
 -- nó sẽ ném ra Exception NO_DATA_FOUND:
 -- Câu lệnh Select ở này sẽ không trả về nhiều hơn 1 bản ghi vì Emp_Id là duy nhất
 -- trong bảng EMP;
 -- Do vậy không xay ra ngoại lệ TOO_MANY_ROWS
 Select ename, sal Into v_Ename, v_Sal From Emp Where empno = p_Empno;
 -- Ghi ra màn hình Console.
 Dbms_Output.Put_Line('Found Record!');
 Dbms_Output.Put_Line(' Empno ' || p_empno);
 Dbms_Output.Put_Line(' Ename ' || v_Ename);
 Dbms_Output.Put_Line(' Sal ' || v_Sal);
```

```
Exception
 When No_Data_Found Then
 -- Ghi ra màn hình Console.
 Dbms_Output.Put_Line('No Record found with empno = ' || p_empno);
End Get_Employee_Infos;
```

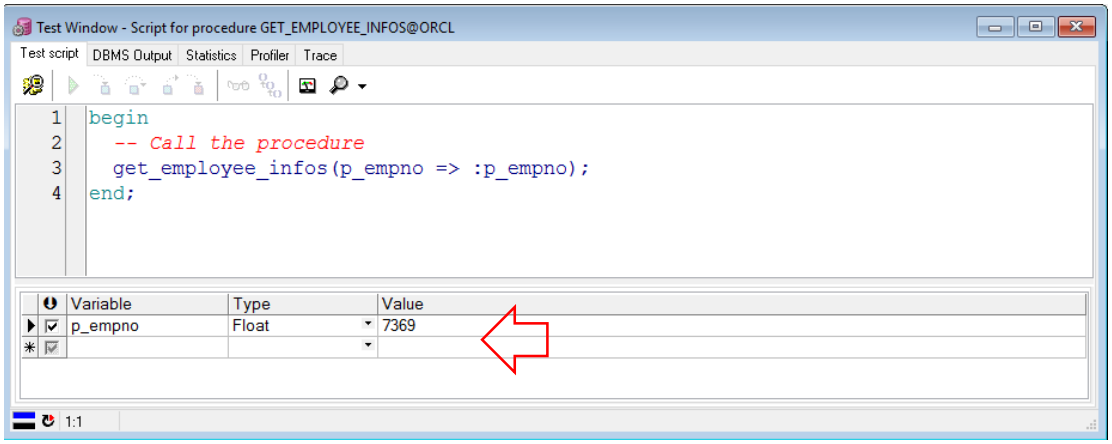
- Test thủ tục trên PL/SQL Developer

Nhấn phải chuột vào thủ tục **Get_Employee_Infos** chọn Test:



Hình 19.*Chức năng test thủ tục*

Nhập tham số đầu vào, ví dụ: P_Empno = 7369



Hình 20.*Nhập tham số cho thủ tục*

Kết quả thực thi thủ tục:

Hình 21.*Kết quả thực thi thủ tục*

1.2. BÀI TẬP THỰC HÀNH

1.2.1. Bài tập về SQL

1. Hiển thị tên và thu nhập trong một năm của các nhân viên.
2. Hiển thị thông tin nhân viên theo nội dung: Who, what and when, dữ liệu hiển thị như ví dụ dưới đây:
3. KING HAS HELP THE POSITION OF PRESIDENT IN DEPT 10 SINCE 17-11-1981
4. Hiển thị nhân viên trong bảng EMP có mức lương trên 1000 đến dưới 2000 (chọn các trường ENAME, DEPTNO, SAL).
5. Hiển thị thông tin những nhân viên làm công việc thư ký (cleck) tại phòng 20.
6. Hiển thị tất cả những nhân viên mà tên có các ký tự TH và LL.
7. Hiển thị tên nhân viên, mã phòng ban, ngày gia nhập công ty sao cho gia nhập công ty trong năm 1983.
8. Tìm lương thấp nhất, lớn nhất và lương trung bình của tất cả các nhân viên
9. Tìm lương nhỏ nhất và lớn nhất của mỗi loại nghề nghiệp.
10. Đếm xem có bao nhiêu quản lý(manager) trong danh sách nhân viên.
11. Tìm tất cả các phòng ban mà số nhân viên trong phòng > 3
12. Hiển thị tên nhân viên, vị trí địa lý(LOC), tên phòng với điều kiện lương >1500.
13. Hiển thị tên nhân viên, nghề nghiệp, lương, mức lương, tên phòng làm việc trừ nhân viên có nghề là cleck và sắp xếp theo mức lương tăng dần.
14. Hiển thị chi tiết về những nhân viên kiếm được 36000$ 1 năm hoặc nghề là cleck. (gồm các trường tên, nghề, thu nhập, mã phòng, tên phòng, mức lương)
15. Hiển thị những phòng không có nhân viên nào làm việc.
16. Tìm những nhân viên kiếm được lương cao nhất trong mỗi loại nghề nghiệp.
17. Tìm mức lương cao nhất trong mỗi phòng ban, sắp xếp theo thứ tự phòng ban.
18. Tìm nhân viên gia nhập vào phòng ban sớm nhất, sắp xếp theo mã phòng ban tăng dần.

19. **Hiển thị những nhân viên có mức lương lớn hơn lương TB của phòng ban mà họ làm việc.

20. Tìm ngày thứ 6 đầu tiên cách 2 tháng so với ngày hiện tại hiển thị ngày dưới dạng 09 February 1990.

1.2.2. Bài tập về tạo bảng

21. Tạo bảng PROJECT với các column được chỉ ra dưới đây, PROJID là primary key, và P_END_DATE > P_START_DATE.

Column name	Data Type	Size.
PROJID	NUMBER	4
P_DESC	VARCHAR2	20
P_START_DATE	DATE	
P_END_DATE	DATE	
BUDGET_AMOUNT	NUMBER	7,2
MAX_NO_STAFF	NUMBER	2

22. Tạo bảng ASSIGNMENTS với các column được chỉ ra dưới đây, đồng thời cột PROJID là foreign key tới bảng PROJECT, cột EMPNO là foreign key tới bảng EMP.

Column name	Data Type	Size.	
PROJID	NUMBER	4	NOT NULL
EMPNO	NUMBER	4	NOT NULL
A_START_DATE	DATE		
A_END_DATE	DATE		
BILL_AMOUNT	NUMBER	4,2	
ASSIGN_TYPE	VARCHAR2	2	

23. Thêm ràng buộc duy nhất (UNIQUE) cho 2 column PROJECT_ID và EMPNO của bảng ASSIGNMENTS.

24. Xem các thông tin về các ràng buộc trong USER_CONSTRAINTS.

25. Xem trong user hiện tại có tất cả bao nhiêu bảng.

26. Thêm dữ liệu vào bảng PROJECTS.

PROJID	1	2
P_DESC	WRITE C030 COURSE	PROOF READ NOTES
P_START_DATE	02-JAN-88	01-JAN-89
P_END_DATE	07-JAN-88	10-JAN-89
BUDGET_AMOUNT	500	600
MAX_NO_STAFF	1	1

27. Thêm dữ liệu vào bảng ASSIGNMENTS

PROJID	1	1	2
EMPNO	7369	7902	7844
A_START_DATE	01-JAN-88	04-JAN-88	01-JAN-89
A_END_DATE	03-JAN-88	07-JAN-88	10-JAN-89
BILL_RATE	50.00	55.00	45.50
ASSIGN_TYPE	WR	WR	PF
HOURS	15	20	30

28. Cập nhật trường ASIGNMENT_TYPE từ WR thành WT.

1.2.3. Bài tập về PL/SQL

29. Viết thủ tục giải phương trình bậc 2.
30. Viết chương trình liệt kê các số chính phương nhỏ hơn 1000.
31. Viết chương trình nhập vào một số nguyên dương n và thực hiện các công việc sau: (a) Kiểm tra n có phải là số nguyên tố không? (b) Nếu n không phải là số nguyên tố thì xác định số nguyên tố gần n nhất và bé hơn n.
32. Viết hàm tìm số Fibonaci thứ n.
33. Viết thủ tục liệt kê các nhân viên trong một phòng ban có mã phòng ban truyền vào từ tham số.
34. Viết chương trình hiển thị các nhân viên có lương cao nhất, thấp nhất trong công ty.
35. Viết thủ tục tăng lương thêm 10% lương cho các nhân viên có lương nhỏ hơn 2000.
36. Viết thủ tục liệt kê các nhân viên vào làm việc tính từ ngày abc truyền vào từ tham số.
37. Viết chương trình PL/SQL liệt kê các cột ENAME, HIREDATE, SAL với điều kiện EMPNO bằng giá trị biến &EMPLOYEE_NO được đưa vào, sau đó kiểm tra:
38. Có phải mức lương lớn hơn 1200
39. Tên nhân viên có phải có chứa chữ T
40. Ngày gia nhập cơ quan có phải là tháng 10 (DEC)
41. Hiển thị các kết quả lên màn hình.
42. Viết hàm kiểm tra password mới có đủ mạnh hay không. Giả sử 1 password mạnh phải thỏa các tiêu chí sau:

    + Không được trùng với username
    + Không được trùng với password cũ
    + Phải chứa ít nhất 1 ký số
    + Phải có độ dài ít nhất là 6 ký tự

Function sau sẽ nhận vào 3 tham số: tên user, password mới mà user sẽ đổi, password cũ. Function sẽ trả về false nếu vi phạm 1 trong các ràng buộc trên

Gợi ý:

```
CREATE OR REPLACE FUNCTION is_password_strong (
p_username VARCHAR2,
p_new_password VARCHAR2,
 p_old_password VARCHAR2)
```

```
-- trả về TRUE nếu pass đủ mạnh
RETURN BOOLEAN
AS
l_return_val BOOLEAN := TRUE;
BEGIN
-- Kiểm tra nếu password trùng với username
IF
THEN
raise_application_error(-20001,'Password same as user name');
END IF;
-- nếu password cũ trùng với password mới (không phân biệt hoa và thường)
IF
THEN
raise_application_error(-20002,'Password has to be different than old
password');
END IF;
-- nếu password không chứa ký số nào
IF (regexp_like (p_new_password, '[0123456789]')= FALSE)
THEN
raise_application_error(-20003,'Password needs at least one digit');
END IF;
-- make sure password is at least six characters
IF //nếu password có chiều dài nhỏ hơn 6
THEN
raise_application_error(-20004, 'Password too short');
END IF;
RETURN l_return_val;
END;
```

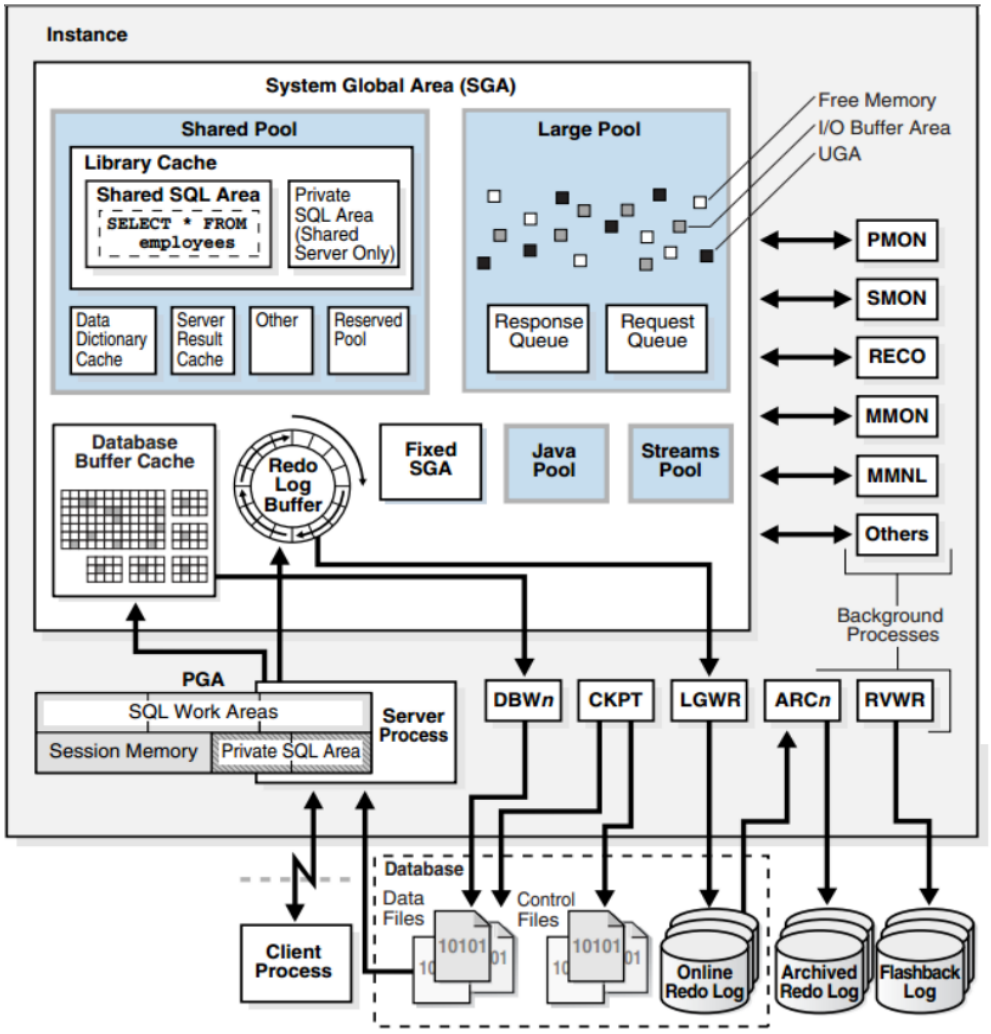
Chương 2
QUẢN TRỊ CƠ SỞ DỮ LIỆU ORACLE

Quản trị cơ sở dữ liệu là gì?

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 hệ quản trị cơ sở dữ liệu mà mình sử dụng.

2.1. KIẾN TRÚC ORACLE SERVER

Oracle Server: Là tập hợp các file, tiến trình (processes) và cấu trúc bộ nhớ trong Oracle Server. Oracle Server bao gồm 2 thành phần chính là: Oracle Instance và Oracle Database.



Hình 22.*Kiến trúc Oracle Server*

2.1.1. Oracle Database

Các HQTCSDL đều dùng cả bộ nhớ máy tính và các thiết bị lưu trữ như ổ cứng để hoạt động. Các ổ cứng cung cấp khả năng lưu trữ lâu dài và một không gian rộng lớn đủ chứa hàng triệu mẫu tin có thể lên đến hàng gigabyte. Tuy nhiên, truy cập dữ liệu từ ổ cứng chậm hơn nhiều so với truy cập từ bộ nhớ. Vì thế các hệ CSDL đều sử dụng bộ nhớ vào việc nạp trước dữ liệu nhằm tăng tốc độ truy vấn.

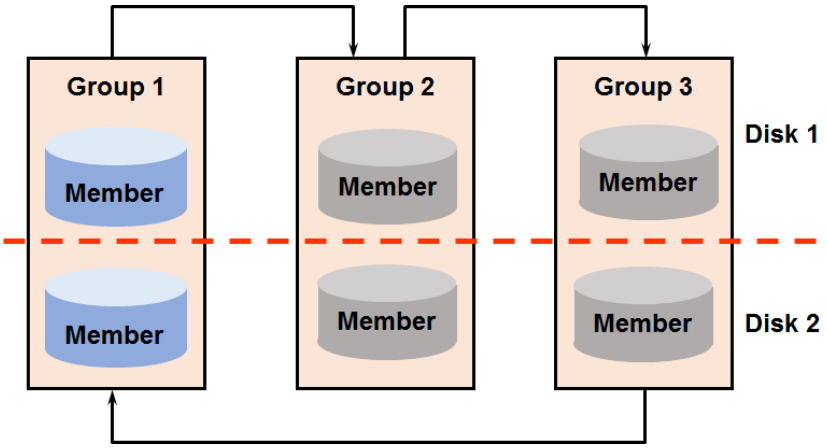
Trong Oracle, một CSDL (**database**) *là một tập hợp các tập tin hệ thống lưu trữ dữ liệu do người dùng hoặc chương trình đưa vào và thông tin về cấu trúc của CSDL* (**metadata**). Oracle database được xác định bởi tên một tên duy nhất và được quy định trong tham số DB_NAME của file tham số (parameter file). Oracle database bao gồm cấu trúc vật lý và cấu trúc logic.

a. Cấu trúc vật lý

Cấu trúc vật lý bao gồm tập hợp các Control file, Data file và Redo log file.

– ***Data file:*** chứa đựng tất cả các dữ liệu của CSDL có cấu trúc logic như các table, index,.. và chúng được lưu giữ vật lý trong các file CSDL. Mỗi datafile chỉ được sử dụng trong một database, và nó cho phép tự động mở rộng kích thước mỗi khi database hết chỗ lưu trữ dữ liệu. Một hay nhiều datafile tạo thành một đơn vị lưu trữ logic của database gọi là Tablespace. Dữ liệu trong datafile có thể đọc ra và lưu vào vùng nhớ đệm của Oracle. Để giảm thiểu việc truy xuất tới bộ nhớ ngoài và tăng khả năng sử dụng hệ thống, Background process sẽ không ghi dữ liệu ngay vào các datafile mà ghi vào bộ nhớ.

– ***Redo log file****:* ghi lại tất cả những thay đổi được tạo cho CSDL và chứa các thông tin cho việc khôi phục. Được tổ chức thành nhóm (group), trong các Group có các member, mỗi group có ít nhất 1 member, phải có ít nhất hai nhóm.



Hình 23.*Cấu trúc Redo logfile*

– ***Control file:*** ghi lại cấu trúc vật lý của CSDL như tên của database, tên và nơi lưu trữ các datafile hay redo log file, mốc thời gian tạo database…

Đặc điểm:

- Là file nhị phân (binary file).
- 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.
- Mỗi control file tại một thời điểm chỉ phục vụ cho một database.
- Oracle thường có ít nhất 2 control file và lưu trữ ở các vị trị khác nhau, khi xảy ra sự cố ở 1 control file, có thể sao chép lại để khôi phục.

b. Cấu trúc logic

Cấu trúc logic của Oracle database bao gồm các đối tượng tablespace, schema, segment, extent và datablock.

– ***Tablespace*:** một database có thể chia về mặt logic thành các đơn vị gọi là tablespace. Một tablespace thường gồm một hay một nhóm các datafile có quan hệ logic với nhau.

– ***Schema*:** schema là tập hợp các đối tượng có trong database. Các đối tượng Schema là các cấu trúc logic cho phép tham chiếu trực tiếp tới dữ liệu trong database như: table, view, sequence, stored procedure, synonym, index, cluster, database link.

– ***Datablock*:** là mức phân cấp logic thấp nhất, các dữ liệu của Oracle database được lưu trữ trong các data block. 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 data block.

– ***Extent*:** là mức phân chia cao hơn data block về mặt logic các vùng không gian trong database. Một extent bao gồm một số data block 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.

– ***Segment*:** 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

2.1.2. Oracle Instance

Để có thể truy vấn và cập nhật CSDL, Oracle phải khởi động một số tiến trình nền và cấp phát một vài vùng nhớ sử dụng trong suốt quá trình thao tác trên CSDL.

Khi một CSDL được khởi động (start), một SGA (System Global Area) được cấp phát và các tiến trình nền (Oracle background processes) được khởi động. Sự kết hợp giữa SGA và các tiến trình nền được gọi là thể hiện CSDL (Database Instance hoặc Oracle Instance).

Trong một server, nhiều CSDL có thể tồn tại song song. Vì vậy, để không bị lẫn lộn giữa các CSDL khác nhau, mỗi thể hiện CSDL được nhận dạng bằng một SID riêng biệt (System Identifier). *Một CSDL có thể được mở (open hay mount) bởi nhiều hơn một thể hiện, nhưng một thể hiện chỉ có thể mở nhiều nhất một CSDL mà thôi.*

a. SGA: Là vùng bộ nhớ chia sẻ được sử dụng để lưu trữ dữ liệu và các thông tin điều khiển của một Instance. Khi kết nối đến server, người dùng được chia sẻ các dữ liệu có trong SGA. Vùng nhớ này sẽ được giải phóng khi thể hiện được tắt (shutdown) và mỗi thể hiện có một SGA riêng biệt. SGA bao gồm các thành phần:

– ***Shared pool:*** Dùng để lưu trữ những đoạn SQL vừa được thực thi gần nhất và những định nghĩa dữ liệu được dùng gần nhất. Shared pool gồm Library cache để lữu trữ định nghĩa về các đoạn lệnh sql và pl/sql vừa được thực thi gần nhất; Data dictionary cache dùng để thu thập định nghĩa được dùng gần nhất trên CSDL bao gồm các thông tin về database file, table, index, column, user, privilege…

– ***Database buffer cache***: lưu trữ các bản copy của datablock được đọc từ datafiles. Khi một sql được thực thi thì trình xử lý sẽ đọc thông tin từ Database buffer cache để lấy các datablock cần thiết, làm cho tốc độ hoạt động cao hơn và nhanh hơn đọc trên đĩa cứng.

– ***Redo log buffer***: là một bản ghi tạm thời, ghi lại tất cả những thay đổi trên các datablock với mực đích để phục hồi dữ liệu, và được thực hiện bởi background process.

– ***Large pool*:** Cung cấp một vùng nhớ lớn, được cấp phát cho các trường hợp như: vùng nhớ cho UGA (user global area), xử lý I/O, sao lưu và phục hồi hệ thống.

– ***Java pool:*** Những yêu cầu về cú pháp đối với các câu lệnh Java.

b. Các tiến trình nền (background process):

– ***DBWR*** (Database writer process): Database writer ghi sự thay đổi blocks từ database buffer cache xuống data files.

– ***LGWR*** (Log writer process): Ghi lại tất cả những thay đổi tới CSDL trong vùng Redo log buffer xuống Online redo log files khi:

+ commit

+ Khi redo log buffer đầy 1/3

+ Khi có nhiều hơn 1 MB thay đổi trong redo log buffer

+ Sau mỗi 3 giây

+ Trước khi DBWn ghi

- **System monitor (SMON):** Có nhiệm vụ phục hồi lại những thay đổi trong redo log, mở database cho user truy xuất; phục hồi các transactions chưa được commit.

- **Process Monitor (PMON):** Thực hiện khôi phục process khi bị lỗi: phục hồi lại các giao dịch (roll back), giải phóng các tài nguyên.

- **Checkpoint (CKPT):** thay LGWR viết các thông tin dữ liệu từ vùng log buffer tới các header của các file dữ liệu và file điều khiển.

2.1.2.1. Từ điển dữ liệu Data Dictionary

- Mọi CSDL Oracle đều có một tự điển dữ liệu. Tự điển dữ liệu được tạo ra khi CSDL được tạo.
- Tự điển dữ liệu trong Oracle là một tập các bảng và view được sử dụng như một tham khảo dạng *chỉ đọc* (read-only) về bản thân CSDL đó.
- Từ điển dữ liệu nằm trên tablespace SYSTEM, thuộc schema của user SYS, bao gồm 2 loại:

✓ Các bảng cơ bản (Base table): là các bảng lưu trữ thông tin của tự điển dữ liệu. Dữ liệu được lưu trong các bảng này dưới dạng mã hóa.

✓ Các view dành cho người dùng truy xuất (User-accessible View): Tổng hợp và hiển thị thông tin được lưu trong các bảng cơ bản ở dạng người bình thường có thể đọc hiểu. Tùy vào quyền của mỗi user mà user đó có thể truy xuất view nào và truy xuất những dữ liệu nào của view đó.

✓ Một tự điển dữ liệu sẽ lưu trữ tất cả các thông tin về cấu trúc luận lý và cấu trúc vật lý của CSDL:

✓ Định nghĩa của tất cả các đối tượng schema trong CSDL.

✓ Các quy định, giới hạn về sử dụng tài nguyên của các user,v.v

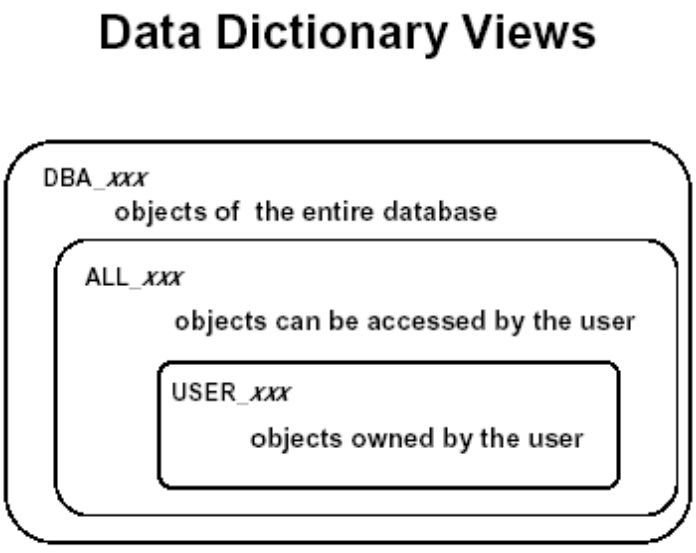
✓ Danh sách các user. Các quyền, role được cấp cho các user.

✓ Các ràng buộc toàn vẹn của dữ liệu

✓ Các thông tin CSDL tổng quát khác.

✓ Oracle tự động cập nhật từ điển dữ liệu để phản ánh chính xác trạng thái thực tế của CSDL.

✓ Data Dictionary views



Hình 24.*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`.

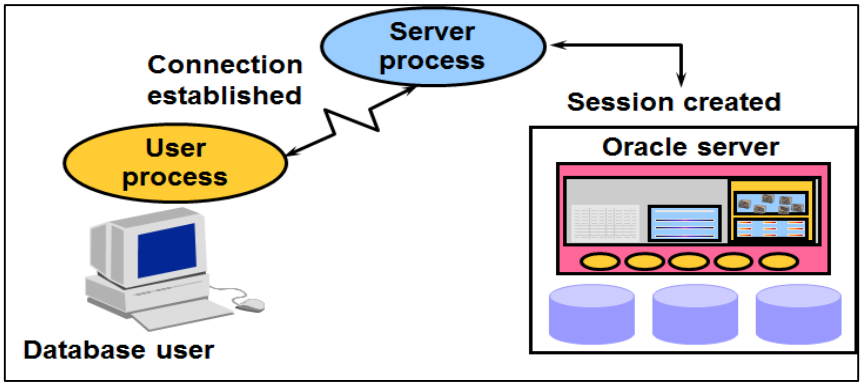
Ví dụ 1: Hiển thị tên các bảng có trong user SCOTT. (Đăng nhập vào SCOTT).

```
SQL> select table_name, tablespace_name from user_tables;

TABLE_NAME                     TABLESPACE_NAME
------------------------------ ------------------------------
DEPT                           USERS
EMP                            USERS
BONUS                          USERS
SALGRADE                       USERS
TEST                           USERS
```

Ví dụ 2: Liệt kê tất cả các user có quyền SELECT ANY TABLE.

```
SQL> select grantee, privilege from dba_sys_privs where privilege='SELECT ANY TABLE' and grantee not in
(select role from dba_roles);

GRANTEE                        PRIVILEGE
------------------------------ ----------------------------------------
SYS                            SELECT ANY TABLE
FLOWS_030000                   SELECT ANY TABLE
OLAPSYS                        SELECT ANY TABLE
SYSTEM                         SELECT ANY TABLE
```

2.1.2.2. Kết nối tới Oracle Server



Hình 25.*Mô hình chung khi client kết nối đế Oracle Server*

❖ **Một số khái niệm cơ bản liên quan đến kết nối**

a. User process

Tiến trình trên máy Client và được khởi động vào thời điểm một người sử dụng yêu cầu kết nối với Oracle Server.

b. Server process

Tiến trình trên máy Server, kết nối với Oracle Instance và được khởi động khi người sử dụng thiết lập một phiên làm việc (Session)[e].

c. Background processes

Các tiến trình nền khởi động trên Server khi một Oracle Instance khởi động.

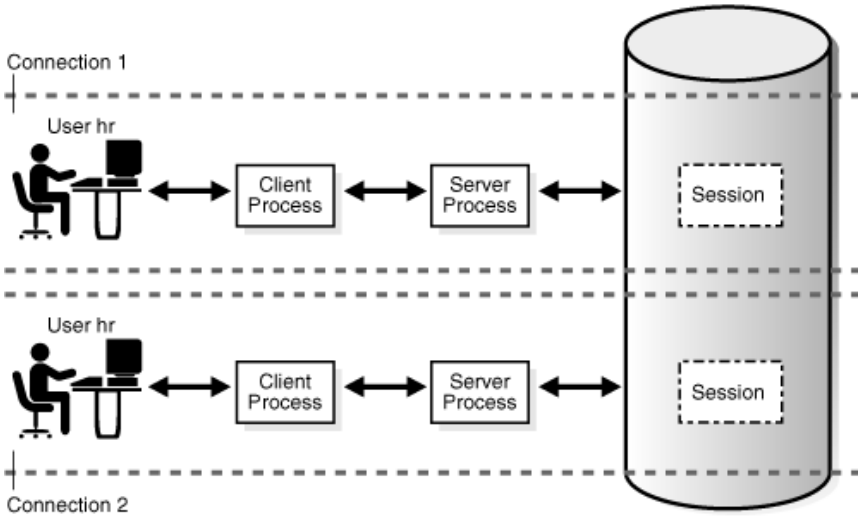
d. Connection (liên kết)

Connection 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 sử dụng ứng dụng nằm trên một máy khác thì liên kết sẽ sử dụng đường mạng để kết nối tới Oracle server.

e. Phiên làm việc (Session)

Session là một kết nối riêng của một user đến một Oracle Server. Session được bắt đầu khi một user xác thực thành công đến một Oracle Server, và kết thúc khi user đăng xuất hoặc bị kết thúc đột ngột.

Từ một máy client (database user), có thể có nhiều kết nối đến Oracle server khi người dùng sử dụng nhiều công cụ hoặc ứng dụng khác nhau đăng nhập vào Oracle Server.

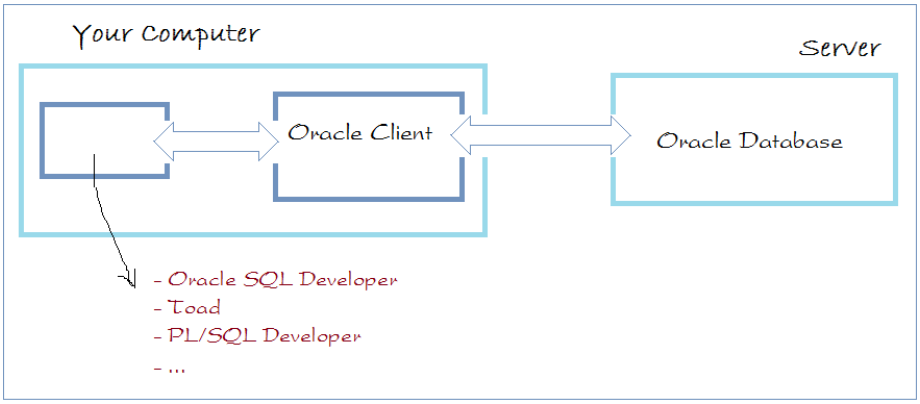


Hình 26.*Một Session – 1 kết nối*

❖ **Các mô hình kết nối**

a. Client – Server

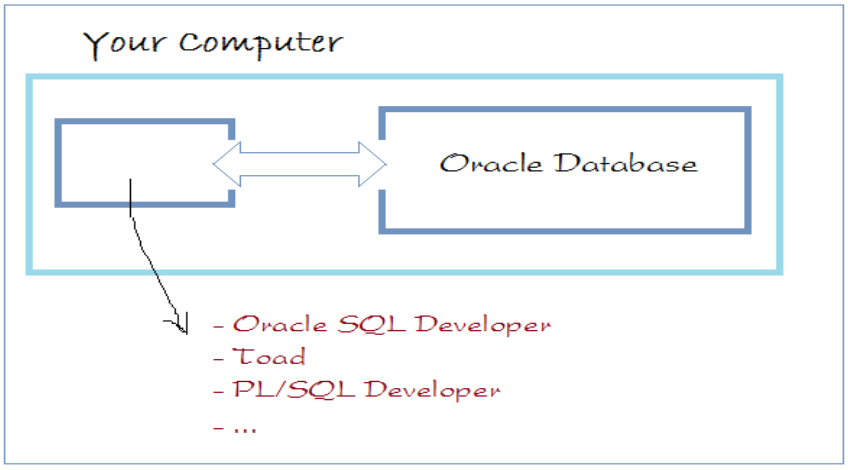
Chương trình trực quan trên một máy tính và kết nối tới một Oracle Server nằm trên một máy tính khác, khi đó máy tính cần kết nối đến Server phải cài đặt Oracle Client hoặc cài luôn một CSDL Oracle. Chú ý: Oracle Database đóng vai trò vừa là server vừa là Client.



Hình 27.*Mô hình Client – Server*

b. Host – Based

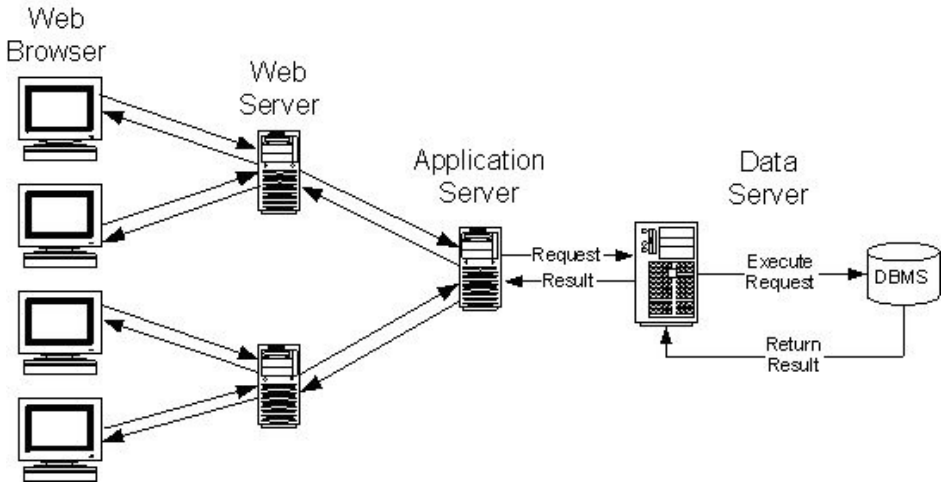
Chương trình trực quan nằm trên một máy tính và kết nối tới CSDL **Oracle** nằm trên cùng máy tính, lúc đó Database này vừa đóng vai trò là một **Oracle Server** vừa là **Oracle Client**. Không cần cài đặt thêm gì khác.



Hình 28.*Mô hình Host – Based*

c. Client – Application server – Server

User có thể truy cập vào cơ sở dữ liệu từ máy tính cá nhân của họ (Client) thông qua một ứng dụng máy chủ (application server), nơi sử dụng cho những yêu cầu chạy chương trình.

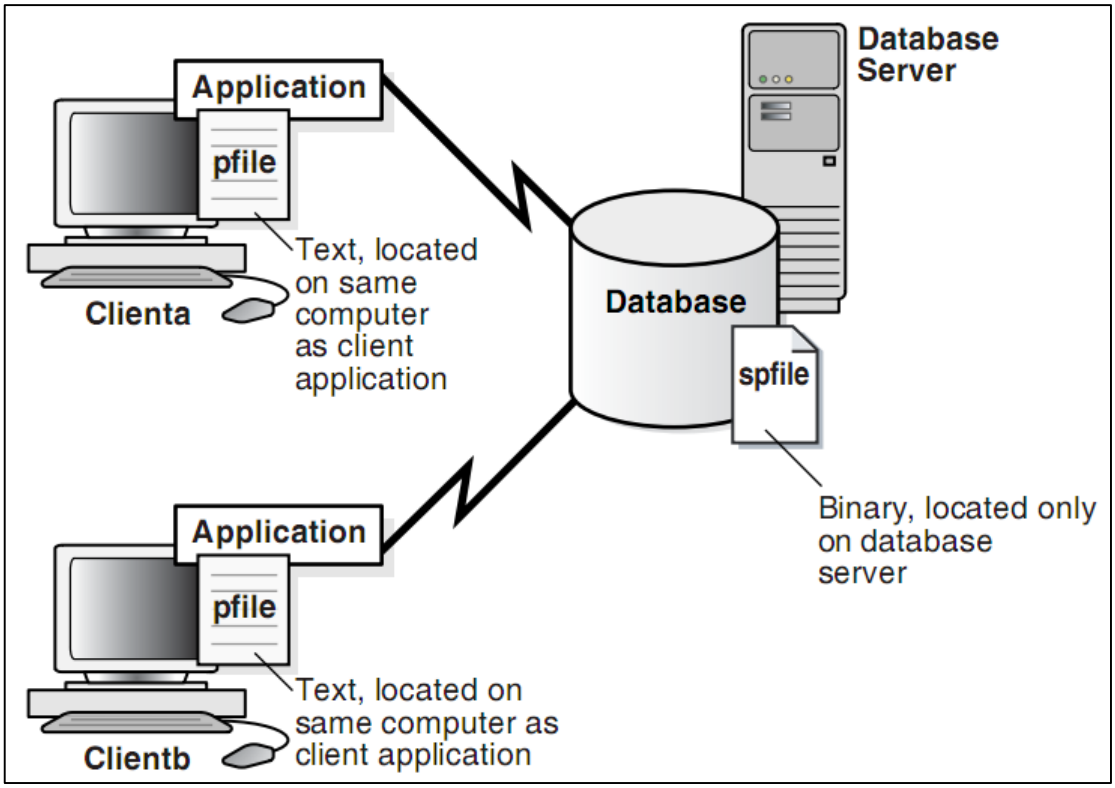


Hình 29.*Mô hình Client – Application server – Server*

2.1.3. Quản lý Instance

2.1.3.1. File tham số (Parameter file)

Để khởi động một Oracle Instance, bước đầu tiên Oracle Server sẽ đọc thông tin các tham số trong Parameter file, có 2 loại file tham số là PFILE và SPFILE.



Hình 30.*Đặc điểm của các file tham số*

Một số thông tin được lưu trong file tham số là:

- Tên của instance
- Kích thước bộ nhớ các thành phần trong SGA
- Tên và vị trí control files

❖ PFILE:

- File dạng text
- Điều chỉnh bởi chương trình soạn thảo của HĐH
- Các điều chỉnh được thực hiện bằng tay
- Các thay đổi có hiệu lực vào lần khởi động kế tiếp
- Chỉ mở khi instance khởi động
- Vị trí mặc định

%ORACLE_HOME%/database với Windown

$ORACLE_HOME\dbs với Unix

- Định dạng tên: initSID.ora

Ví dụ về PFILE:

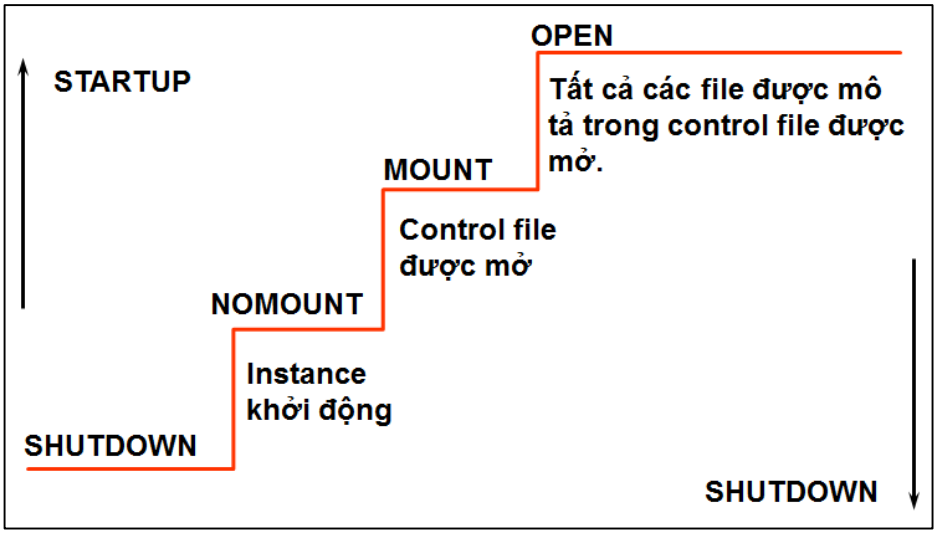
```
# Initialization Parameter File: initdba01.ora
db_name = dba01
instance_name = dba01
control_files = (
  /home/dba01/ORADATA/u01/control01dba01.ctl,
  /home/dba01/ORADATA/u02/control01dba02.ctl)
db_block_size = 4096
db_cache_size = 4M
shared_pool_size = 50000000
java_pool_size = 50000000
max_dump_file_size = 10240
background_dump_dest = /home/dba01/ADMIN/BDUMP
user_dump_dest = /home/dba01/ADMIN/UDUMP
core_dump_dest = /home/dba01/ADMIN/CDUMP
undo_management = AUTO
undo_tablespace = UNDOTBS
```

- SPFILE:
  - Binary file.
  - Được quản lý bởi Oracle Server.
  - Luôn luôn đặt ở máy chủ.
  - Có khả năng tạo ra các thay đổi mà không phải tắt và khởi động lại database.
  - Định dạng tên: spfileSID.ora

2.1.3.2. Start và Shutdown Database

- **Các bước Start và Shut down database**

Để Start và Shutdown database, đầu tiên cần sử dụng SQL*Plus và đăng nhập vào user sys với quyền **sysdba** hoặc **sysoper**.



Hình 31.*Quy trình start và shutdown database*

a. 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;`

b. 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;`

c. 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;`

d. 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.

e. Dismount database

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

f. Shutdown Instance

Đây là bước cuối cùng, instance 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.

2.1.3.3. Start database

Cú pháp:

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

Với:

Bảng 8.*Các tham số start database*

OPEN	Cho phép các users truy cập vào database.
MOUNT	Gắn 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=filename	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 (hạn chế).
RECOVER	Khởi động với chế độ khôi phục dữ liệu

2.1.3.4. Chuyển đổi các trạng thái database

CSDL Oracle có thể chuyển đổi trạng thái khởi động khi đang ở mức thấp lên trạng thái khởi động mức cao hơn hoặc có thể thay đổi tính sẵn dùng như là chỉ đọc (read only) hoặc có thể đọc và ghi bình thường (read write)

Thực hiện sửa đổi database theo lệnh:

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

Với:

Bảng 9.*Các tham số chuyển đổi trạng thái database*

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.
OPEN	Tương tự như OPEN READ ONLY, đây là biểu diễn mặc định của OPEN READ WRITE.

Ví dụ:

- CSDL khởi động ở trạng thái nomount, để chuyển lên trạng thái open, ta sử dụng câu lệnh: ***alter database open;***

- CSDL đang ở trạng thái open read write, để chuyển về sang trạng chỉ đọc ta thực hiện các bước:

Bước 1: shutdown immediate;

Bước 2: startup open read only;

2.1.3.5. Shutdown Database

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

Bảng 10.*So sánh các chế độ tắt database*

Shutdown Mode	ABORT	IMMEDIATE	TRANSACTIONAL	NORMAL
Cho phép tạo các kết nối mới	NO	NO	NO	NO
Đợi các phiên hiện thời kết thúc	NO	NO	NO	YES
Đợi các giao dịch hiện thời kết thúc	NO	NO	YES	YES
T.hiện một checkpoint, đóng các file	NO	YES	YES	YES

Cú pháp:

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

Với:

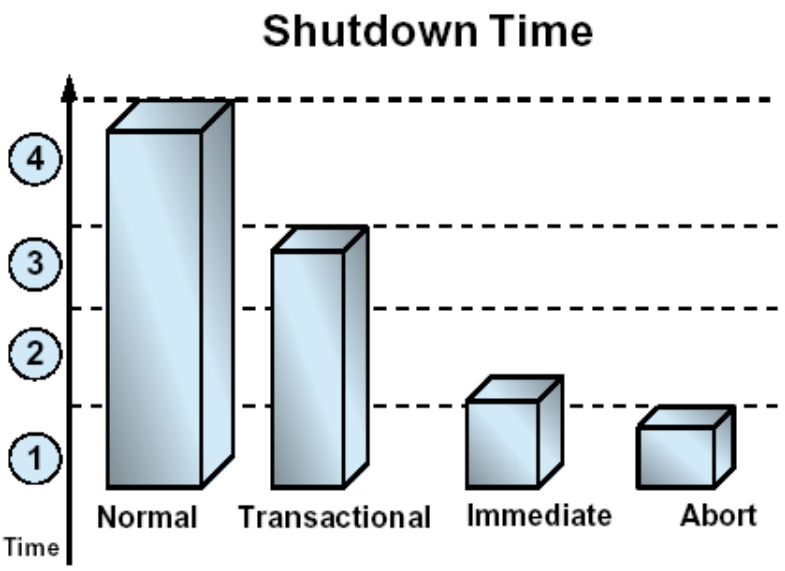
NORMAL Không cho tạo thêm các connection tới database, chờ cho connection hiện thời kết thúc thì shutdown database.

TRANSACTIONAL 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 32.*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:

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

2.1.4. Bài tập thực hành

1. Đăng nhập vào user sys và shutdown database.
2. Đăng nhập vào user sys và startup database.
3. Mở khóa user HR và thay đổi mật khẩu là *hr1234* như sau:

   SQL> alter user HR account unlock identified by hr1234;
4. Shutdown database và mở lại ở chế độ read-only.
5. Đăng nhập vào user HR và thực hiện insert vào bảng REGIONS như sau:

   INSERT INTO regions VALUES (5, 'Mars');

   Điều gì sẽ xảy ra?
6. Chuyển database sang chế độ read-write, thực hiện insert lại vào bảng REGIONS nhưng chưa commit;
7. Mở 1 session mới và đăng nhập vào user sys, thực hiện Shutdown database ở chế độ TRANSACTIONAL. Điều gì sẽ xảy ra ở phiên làm việc của user sys?
8. Rollback dữ liệu vừa insert vào bảng HR, điều gì sẽ xảy ra ở session của sys?

9. Tắt 2 session và tạo 1 session mới với user sys và startup database.

2.2. TẠO CƠ SỞ DỮ LIỆU

2.2.1. Tổng quan

2.2.1.1. Lên kế hoạch và tổ chức một CSDL

Lập kế hoạch cho CSDL là bước đầu tiên quản lý hệ thống CSDL.

- Xác định loại CSDL (data warehousing, high online transaction processing, general purpose)

- Vạch ra thiết kế kiến trúc của CSDL (How will data files, control files, and online redo log files be organized and stored?)

- Lựa chọn tên của CSDL. (Chú ý: Tên CSDL dài tối đa 8 kí tự với phiên bản oracle 10g, 12 kí tự với phiên bản oracle 11g)

2.2.1.2. Các điều kiện để thiết lập CSDL

Để tạo một CSDL mới, bạn cần phải có các điều kiện sau:

- Một account đủ quyền tạo CSDL.

- Bộ nhớ đủ để khởi động một instance.

- Đĩa đủ dung lượng cho CSDL đã lên kế hoạch.

2.2.1.3. Các cách để tạo 1 CSDL

- Chương trình cài đặt Oracle Universal Installer.

- Sử dụng công cụ tạo CSDL tự động Database Configuration Assistant (DBCA)

- Tạo thủ công bằng các dòng lệnh

2.2.2. Tạo và xóa CSDL sử dụng Database Configuration Assistant (DBCA)

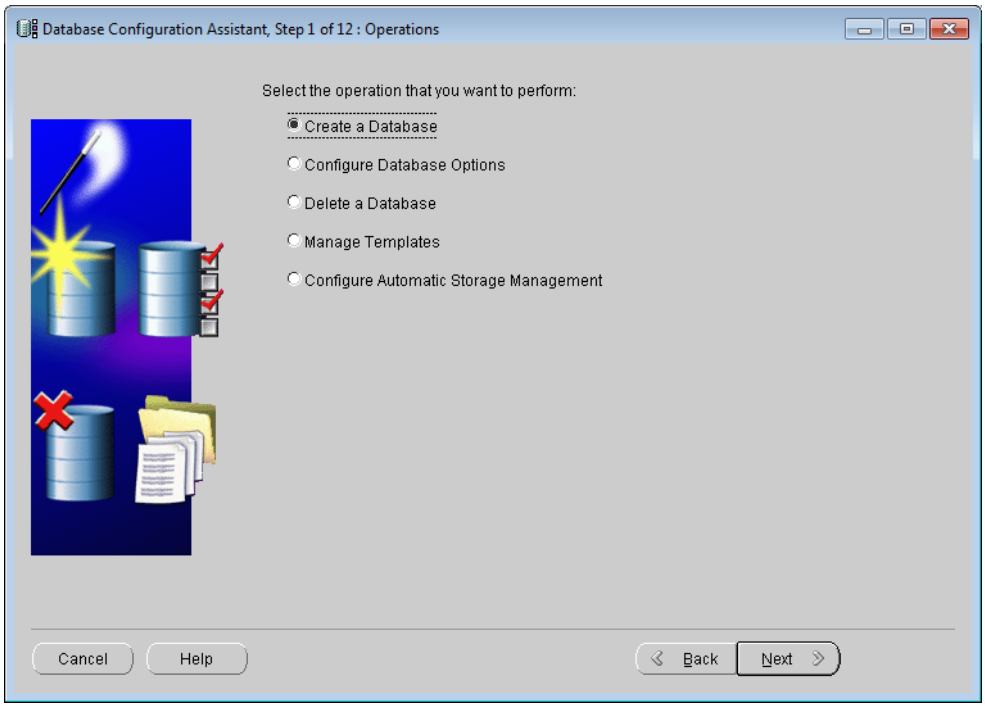
Các chức năng của Database Configuration Assistant:

- Tạo một CSDL.
- Cấu hình lại các thuộc tính của CSDL.
- Xóa một CSDL.

Để khởi động chương trình, tìm đến chương trình có tên Database Configuration Assistant và chạy với quyền administrator.

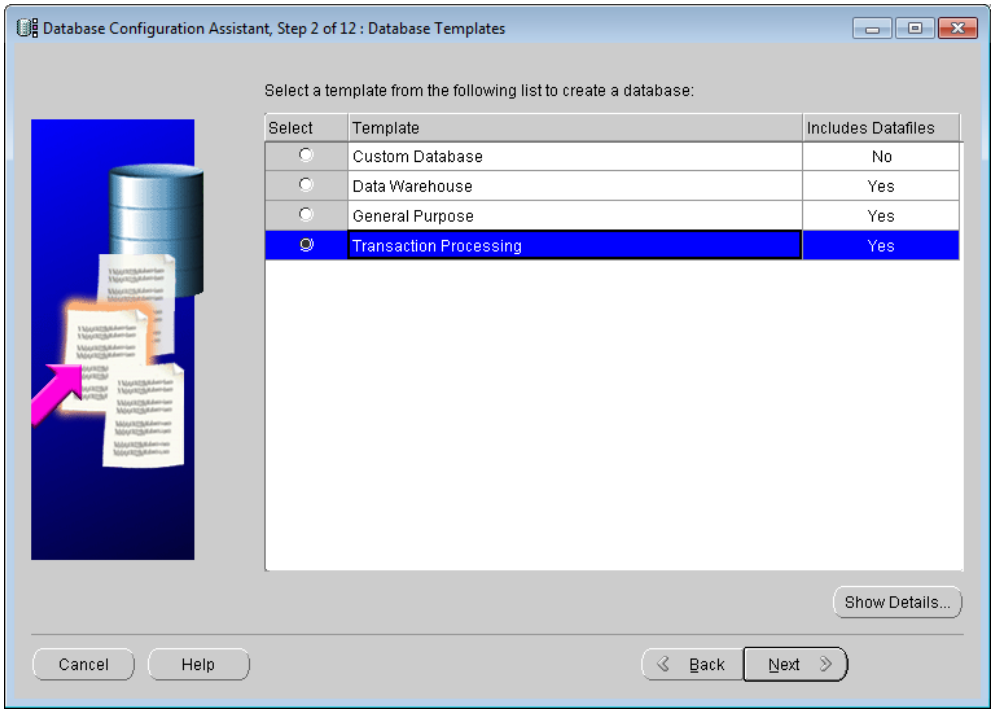
2.2.2.1. Các bước tạo CSDL sử dụng công cụ DBCA

❖ *Bước 1. Chọn chức năng đầu tiên "Create a Database"*



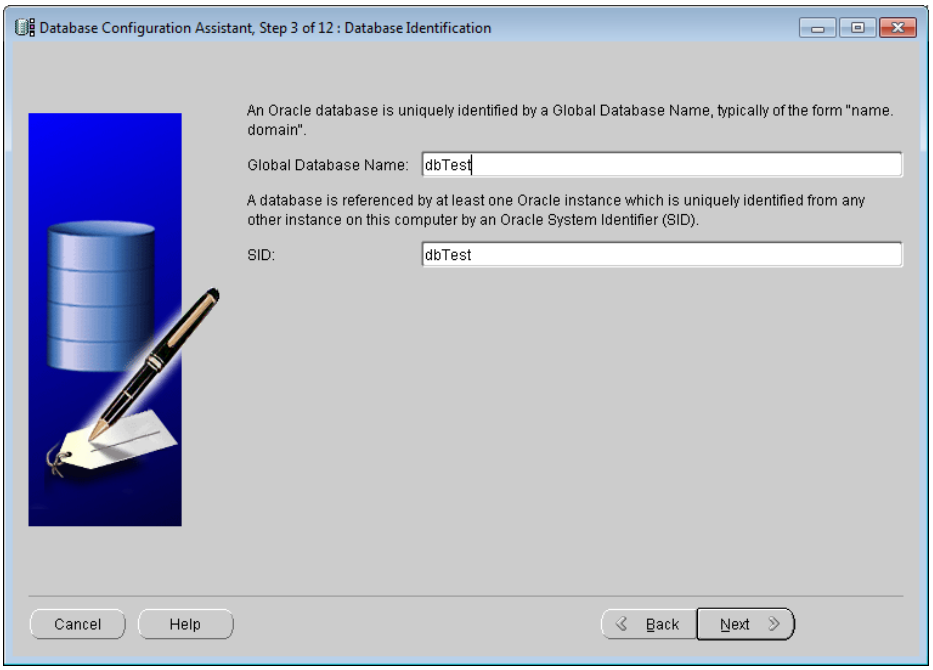
Hình 33.*Giao diện khởi động*

❖ *Bước 2. Chọn loại CSDL là "Transaction Procesing"*



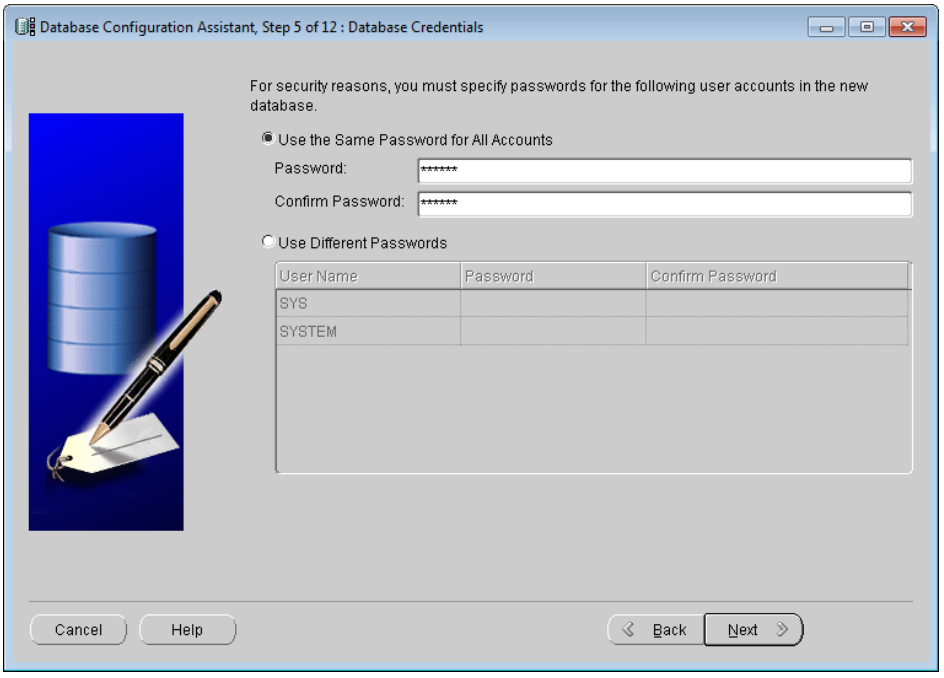
Hình 34.*Chọn loại CSDL*

❖ *Bước 3. Nhập tên Database (tên này là duy nhất, tối đa 8 kí tự)*



Hình 35.*Nhập tên CSDL cần tạo*

❖ *Bước 4. Nhập mật khẩu mặc định dùng cho các user hệ thống SYS, SYSTEM*

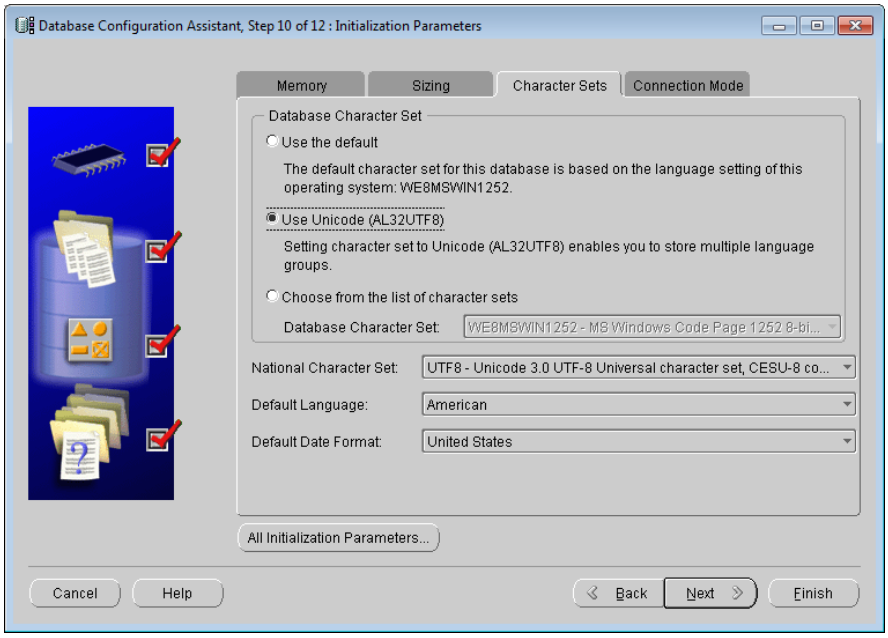


Hình 36.*Nhập mật khẩu*

❖ *Bước 5. Bỏ qua một số bước trung gian, Next đến bước thiết lập các tham số cho hệ thống (Step 10 of 12)*

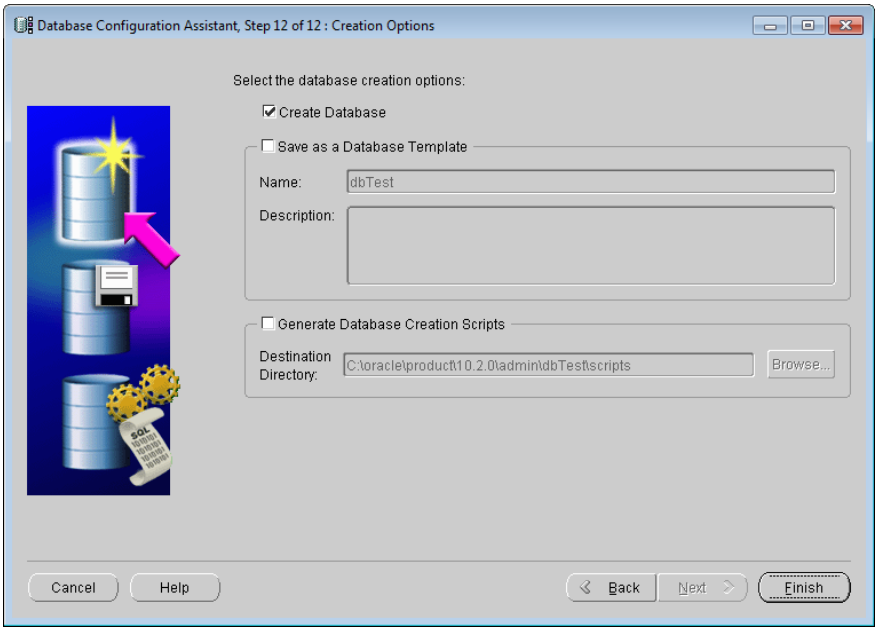
Ở step 10, chuyển sang tab **Character Sets** để thiết lập gõ Unicode như hình bên dưới.

- Database Character Set: **Unicode (AL32UTF8)**
- National Character Set: **UTF8**

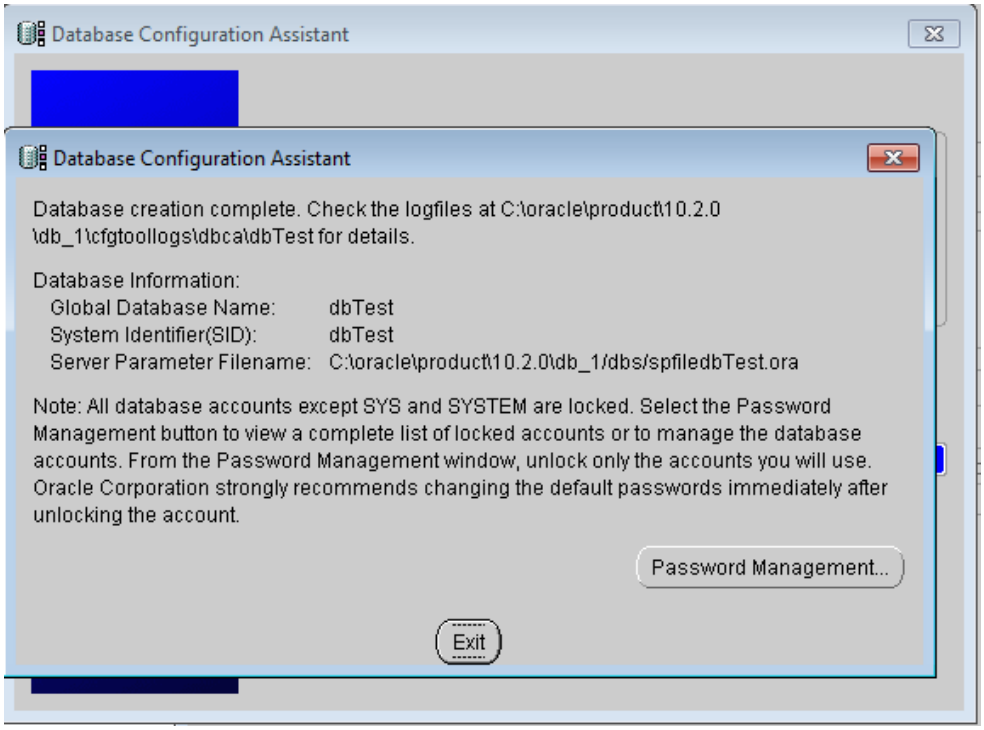


Hình 37.*Cấu hình tập kí tự cho Database*

❖ *Bước 6. Finish*



Hình 38.*Bấm Finish để bắt đầu tạo*



Hình 39.*Tạo CSDL tự động thành công*

2.2.2.2. Các bước xóa CSDL sử dụng công cụ DBCA

Bước 1. Chọn "Delete a Database"

Bước 2. Chọn tên CSDL cần xóa

Bước 3. Finish

2.2.3. Tạo một CSDL thủ công

❖ 1. Tạo file tham số (PFILE)

- Tạo thư mục con có tên mynewdb trong thư mục: C:\oracle\product\10.2.0\oradata\

- Nội dung file:

```
control_files =
(C:\oracle\product\10.2.0\oradata\mynewdb\control1.ctl,
C:\oracle\product\10.2.0\oradata\mynewdb\control2.ctl)
undo_management = AUTO
undo_tablespace = UNDOTBS1
db_name = mynewdb
db_block_size = 8192
sga_max_size = 1073741824 # 1GB
sga_target = 1073741824 #1GB
```

- File tham số được lưu với tên **initmynewdb.ora** ở thư mục : **%oracle_home%\database**

❖ 2. Thiết lập biến môi trường trong cmd

- Vào cmd, thiết lập biến Oracle_sid=tên_instance_chuẩn_bị_tạo, oracle_home=đường_dẫn_thư_mục_cài_đặt_oracle
- Cú pháp: Set oracle_sid=mynewdb

Set oracle_home=C:\oracle\product\10.2.0\db_1

❖ 3. Tạo file password

orapwd file=%oracle_home%\database\pwdmynewdb.ora password=abc123 entries=5

❖ 4. Tạo instance

oradim -new -sid mynewdb -startmode manual

❖ 5. Tạo SPFILE

sqlplus sys/abc123 as sysdba (Khởi động SQL*Plus và đăng nhập vào SYS)

SQL> create spfile from pfile;

❖ 6. Khởi động instance ở giai đoạn NOMOUNT.

SQL> Startup nomount;

❖ 7. Tạo file Script thực hiện lệnh CREATE DATABASE

```
create database mynewdb
logfile group 1 ('C:\oracle\product\10.2.0\oradata\mynewdb\g1_redo01.log',
'C:\oracle\product\10.2.0\oradata\mynewdb\g1_redo02.log') size 100M,
group 2 ('C:\oracle\product\10.2.0\oradata\mynewdb\g2_redo01.log',
'C:\oracle\product\10.2.0\oradata\mynewdb\g2_redo02.log') size 100M
character set UTF8
national character set AL16UTF16
datafile 'C:\oracle\product\10.2.0\oradata\mynewdb\system.dbf' size 500M
autoextend on next 10M maxsize unlimited extent management local
sysaux datafile 'C:\oracle\product\10.2.0\oradata\mynewdb\sysaux.dbf' size
100M autoextend on next 10M maxsize unlimited
undo tablespace undotbs1 datafile
'C:\oracle\product\10.2.0\oradata\mynewdb\undotbs1.dbf' size 100M
default temporary tablespace temp tempfile
'C:\oracle\product\10.2.0\oradata\mynewdb\temp01.dbf' size 100M;
```

- Lưu nội dung trên vào file có tên: **createmynewdb.sql** và đặt vào thư mục: **%oracle_home%\database**
- Thực hiện câu lệnh sau để tạo database: **@?\database\createmynewdb.sql**

❖ 8. Chạy các scripts để tạo data dictionary và hoàn thành các bước sau khi tạo CSDL

@?/rdbms/admin/catalog.sql

@?/rdbms/admin/catproc.sql

@?/sqlplus/admin/pupbld.sql

EXIT

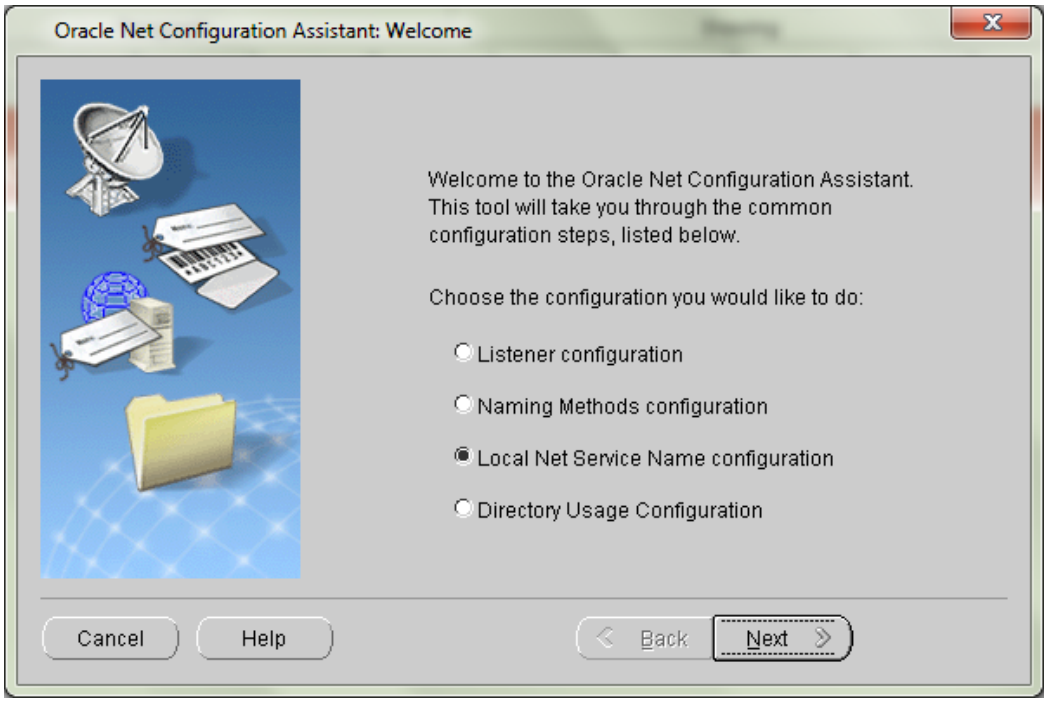
❖ 9. Cấu hình file tnsnames.ora để listener lắng nghe database

Sau khi tạo database bằng tay, chúng ta chưa thể sử dụng các công cụ trực quan (SQL | PL/SQL Deverloper,.v.v) để kết nối đến CSDL vì ta chưa cấu hình để Listener lắng nghe database mới tạo ra. Để làm điều này, có thể sử dụng các công cụ: Net Configuration Assistant, Net Manager.

Để sử dụng Net Configuration Assistant, ta vào:

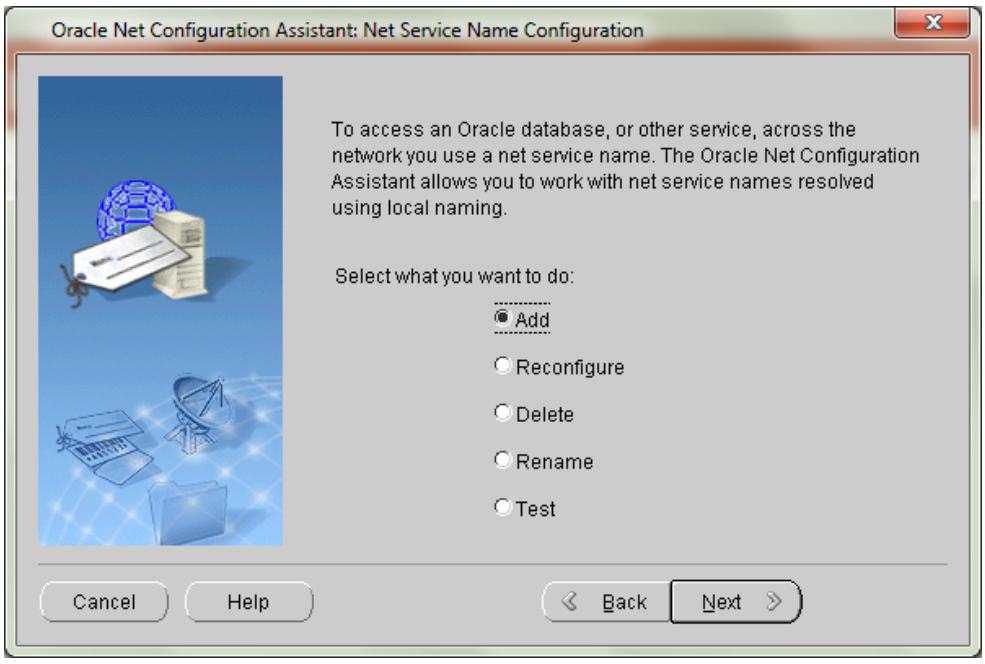
Start Menu\All Programs\Oracle - OraDb10g_home1\Configuration and Migration Tools\Net Configuration Assistant

➢ Chọn **Local Net Service Name configuration** để thêm tên CSDL cần Listener lắng nghe. Bấm Next



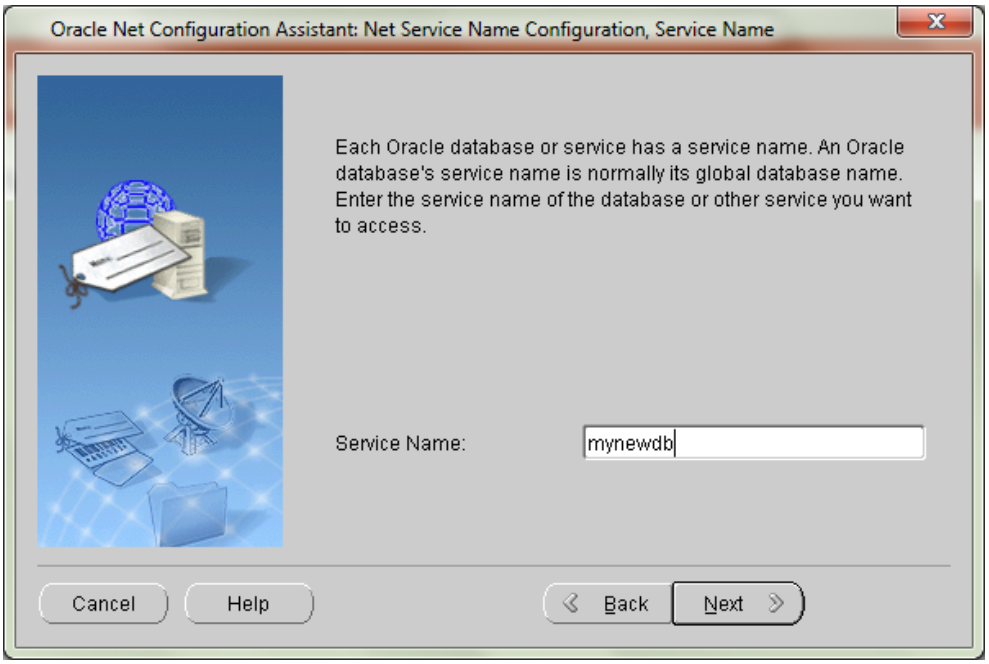
Hình 40.*Cấu hình listener bước 1*

- Chọn **Add.** Bấm Next



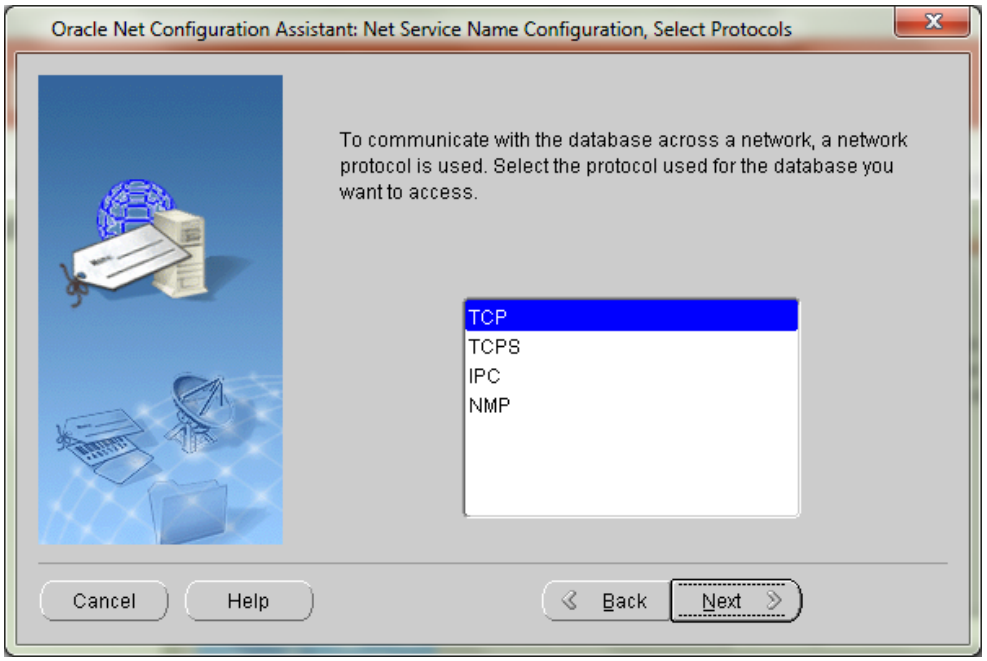
Hình 41.*Cấu hình listener bước 2*

- Gõ tên CSDL cần lắng nghe. Ở ví dụ này CSDL mới cần lắng nghe là: **mynewdb**



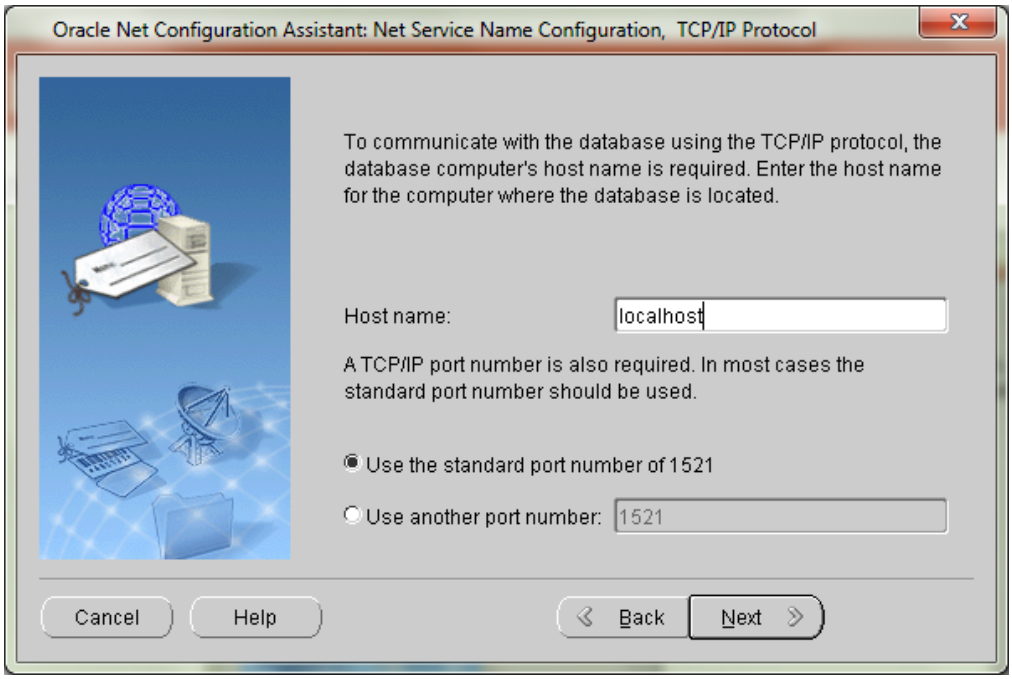
Hình 42.*Cấu hình listener bước 3*

- Chọn giao thức lắng nghe, thường để mặc định là TCP.



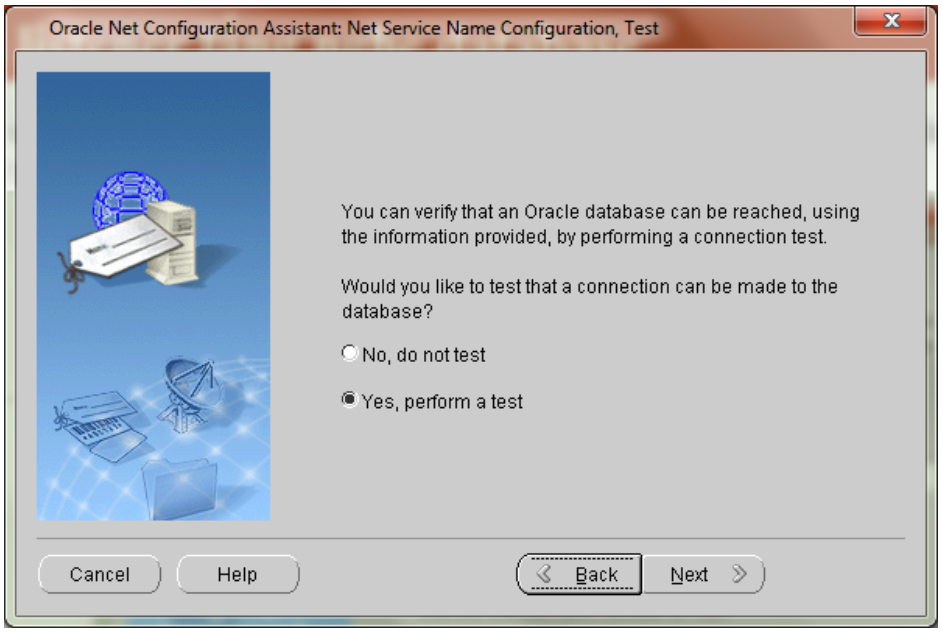
Hình 43.*Cấu hình listener bước 4*

- Nhập tên máy chủ lưu trữ CSDL. Ở đây Listener lắng nghe CSDL ngay trên cùng 1 máy nên có thể đặt là **localhost** hoặc tên máy.



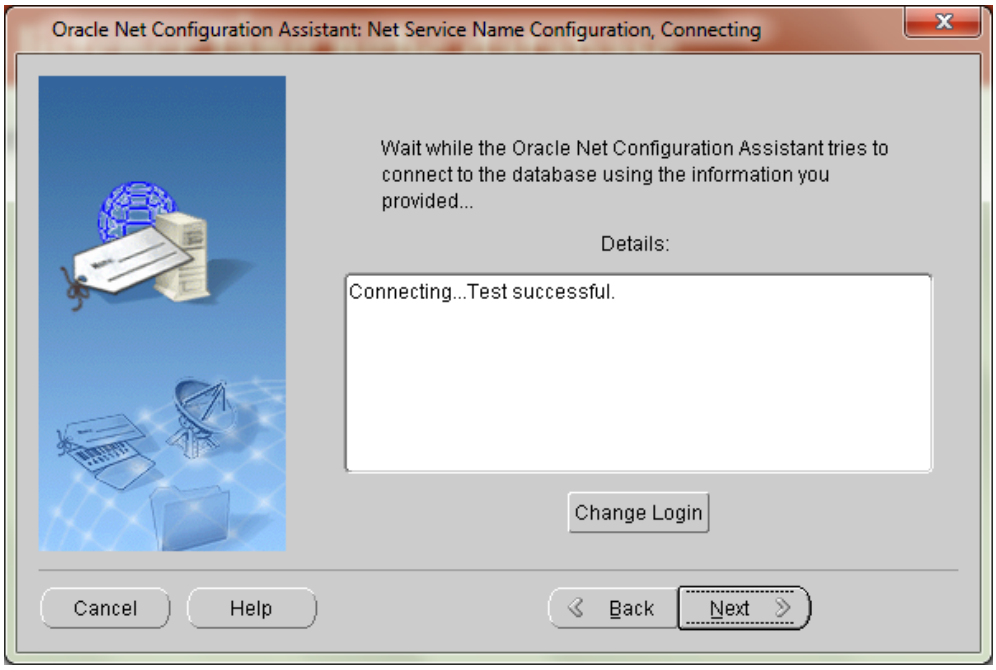
Hình 44.*Cấu hình listener bước 5*

- Chọn "Yes, perform a test" để kiểm tra kết quả.



Hình 45.*Cấu hình listener bước 6*

- Nếu hiện ORA-01017: invalid username/password; logon denied hoặc Test successful thì đã thành công.



Hình 46.Cấu hình listener thành công

Có thể sử dụng lệnh **lsnrctl status** để kiểm tra tình trạng lắng nghe các database của listener.

Để bật/tắt listener, sử dụng lệnh: **lsnrctl start/stop**

- Xóa CSDL bằng tay

Sử dụng câu lệnh DROP DATABASE để xóa CSDL. Điều kiện để xóa được CSDL bằng cách này là database phải startup ở chế độ sau:

+ MOUNT

+ EXCLUSIVE mode

+ RESTRICTED mode

Các bước thực hiện: VD xóa CSDL **mynewdb** vừa mới tạo ra

Khởi chạy cmd.

- set oracle_sid=mynewdb
- sqlplus sys/abc123 as sysdba
- shutdown immediate;
- startup mount exclusive restrict;
- drop database;
- Quit
- sc delete oracleservicemynewdb

2.2.4. Bài tập thực hành

Bài 1. Tạo CSDL bằng tay với tên theo cú pháp **YOURNAMEDB:** (chú ý: tên CSDL dài tối đa 8 ký tự): Ví dụ: **NAMDB**, **TRANGDB**

Bài 2. Sau khi tạo CSDL thành công, đăng nhập vào user sys truy vấn tên và ngày tạo database. Gợi ý: truy vấn trong bảng v$database. Để xem cấu trúc bảng, sử dụng lệnh: desc tên_bảng;

Bài 3. Xóa CSDL vừa tạo ra.

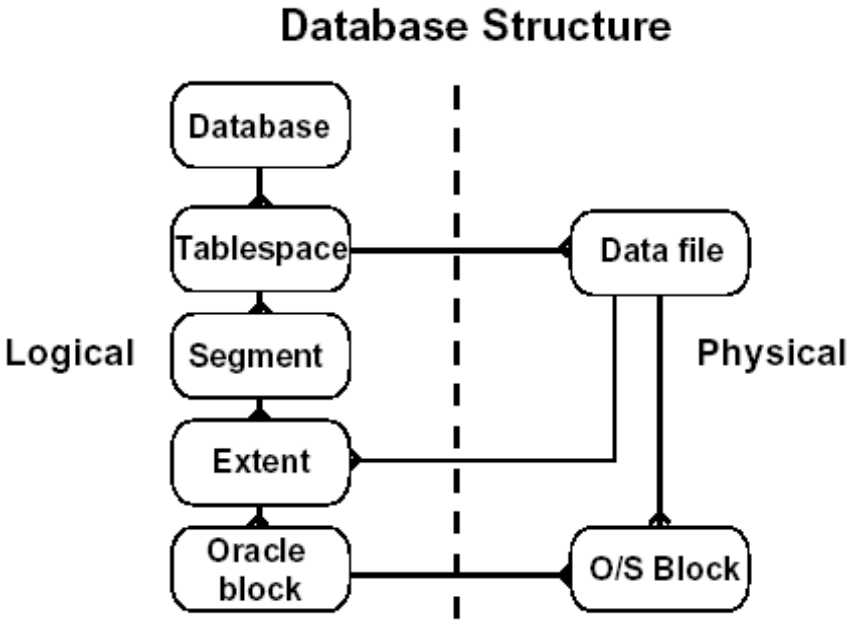
2.3. QUẢN TRỊ CƠ SỞ DỮ LIỆU ORACLE

2.3.1. Quản lý Tablespaces và Datafiles

2.3.1.1. 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.

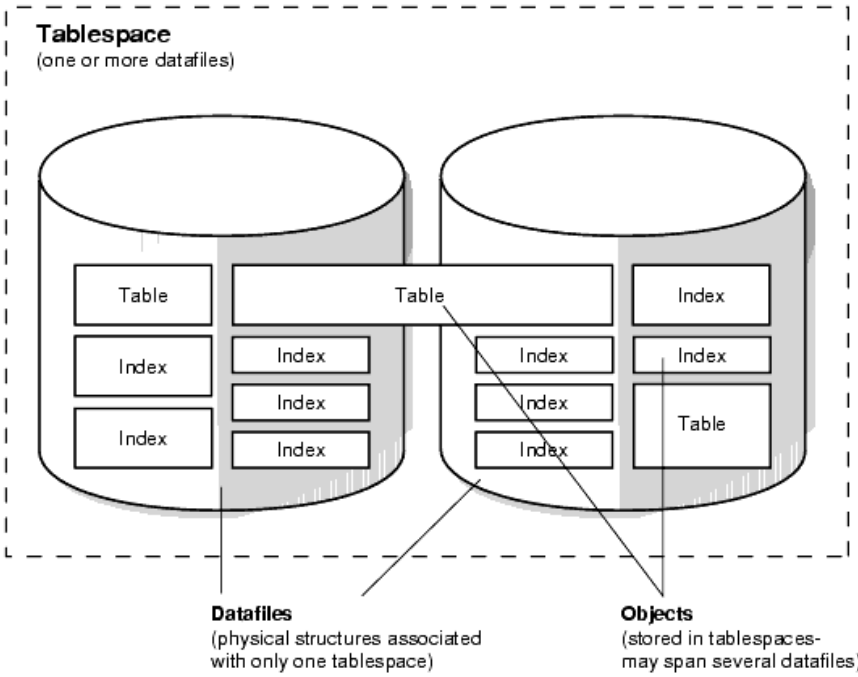


Hình 47.*Cấu trúc database*

❖ ***Quan hệ giữa database, tablespaces và data files***

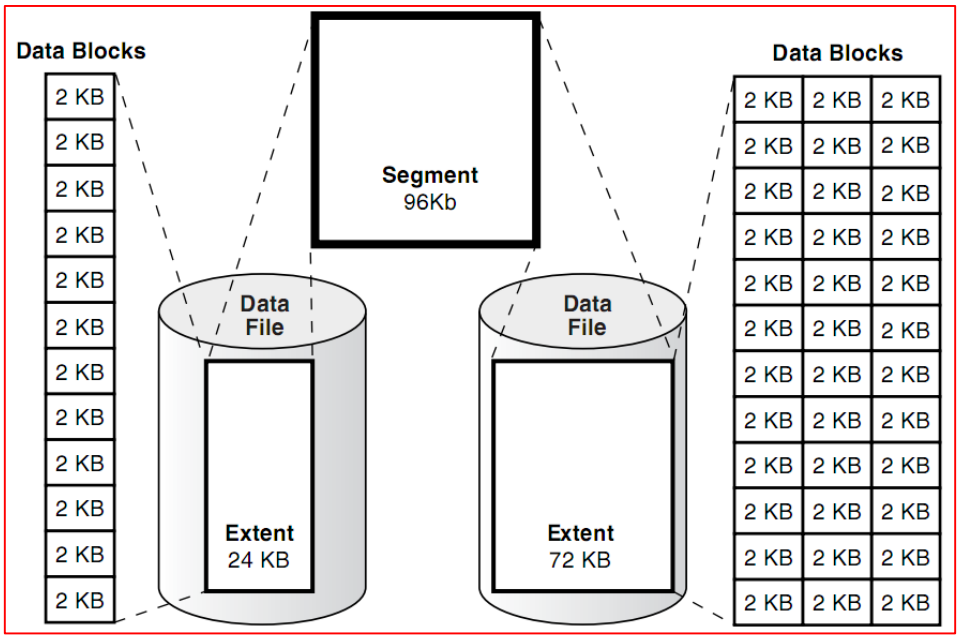
Về mặt logic, một database có thể phân nhỏ thành nhiều phần gọi là các tablespaces

- Tablespace:
  - Thuộc về chỉ một cơ sở dữ trong một thời điểm nhất định.
  - Bao gồm một hoặc nhiều data file.
  - Tách ra thành nhiều đơn vị lưu trữ logic.
- Data file:
  - Thuộc về một tablespace.
  - Là kho chứa cho lược đồ đối tượng dữ liệu.



Hình 48.*Quan hệ giữa tablespace và datafile*

❖ ***Quan hệ giữa segment, extent và các blocks trong tablespace***



Hình 49.*Minh họa quan hệ giữa các đơn vị logic trong tablespace*

❖ ***Data Blocks:***

Đây là đơn vị lưu trữ dữ liệu nhỏ nhất trong database Oracle. Một block dữ liệu

sẽ tương ứng với 1 số byte lưu trữ trong ổ đĩa. Kích thước của block dữ liệu được xác định bởi tham số khởi tạo DB_BLOCK_SIZE ngay khi database được tạo.

❖ ***Extents***

Một extent là 1 tập hợp các data block. Một extent chỉ nằm trên 1 datafile.

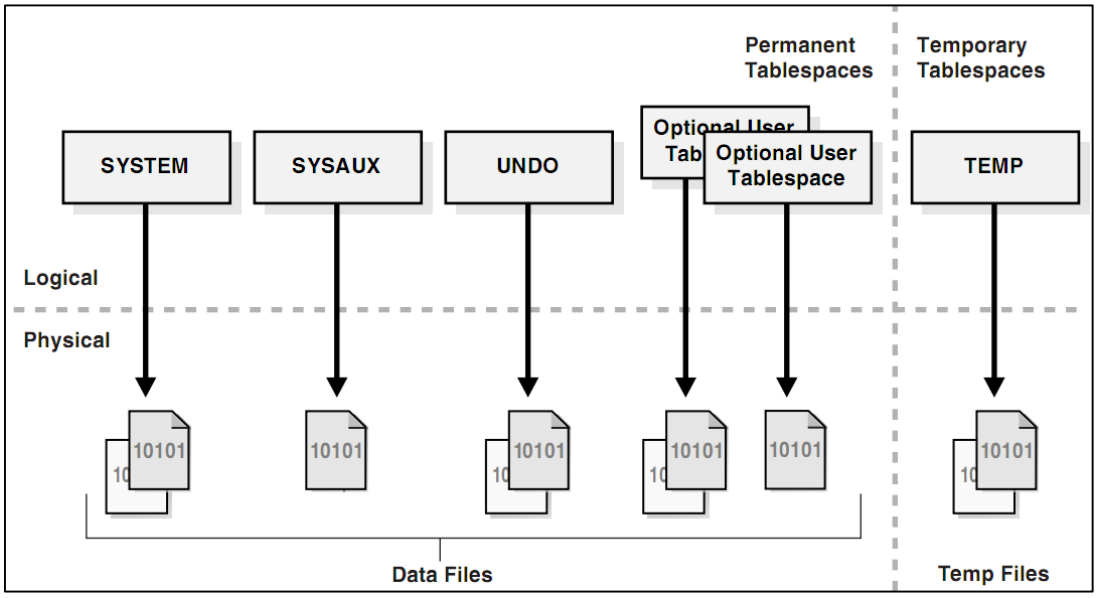
❖ ***Segments***

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

- Data segments
- Index segments
- Temporary segments
- Undo (Rollback) segments

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

2.3.1.2. Phân loại Tablespaces



Hình 50.*Sơ đồ phân loại Tablespaces*

a. Permanent Tablespaces

- Permanent Tablespaces là nhóm tablespaces lưu trữ các đối tượng dữ liệu lâu dài. Các segment dữ liệu của permanent tablespaces được lưu trữ trên ổ đĩa trong các datafiles.
- Mỗi user được gán một permanent tablespaces khi user được tạo ra. Mệnh đề DEFAULT TABLESPACE trong câu lệnh CREATE DATABASE sẽ quy định tablespace mặc định được gán cho user.
- Một Oracle database bắt buộc phải có SYSTEM và SYSAUX tablespaces.

- SYSTEM Tablespace
- Bắt buộc phải có trong mỗi database.
- Được sở hữu bởi user SYS và lưu trữ các thông tin sau:
- Data dictionary
- Table và view chứa thông tin quản trị database.
- Các định nghĩa của store procedure, trigger, package,...

❑ SYSAUX Tablespace

- Là tablespace bổ trợ cho SYSTEM tablespace.
- Sử dụng cho các thành phần như Oracle Enterprise Manager, Oracle Streams, Oracle Ultra Search, Oracle Data Mining,...

❑ UNDO Tablespace

- Là tablespace đặc biệt được sử dụng để lưu trữ các undo segment phục vụ cho việc khôi phục lại (Rollback) các transaction chưa commit.
- Không thể tạo bất kỳ một đối tượng nào trong tablespace này.
- Các extent được quản lý ở chế độ locally managed.
- Cú pháp tạo:

```
CREATE UNDO TABLESPACE undo1
```

❑ Optional User Tablespace

Là tablespace dùng cho việc lưu trữ các đối tượng trong lược đồ dữ liệu của người sử dụng như table, view, sequence, index, …

b. Temporary Tablespaces

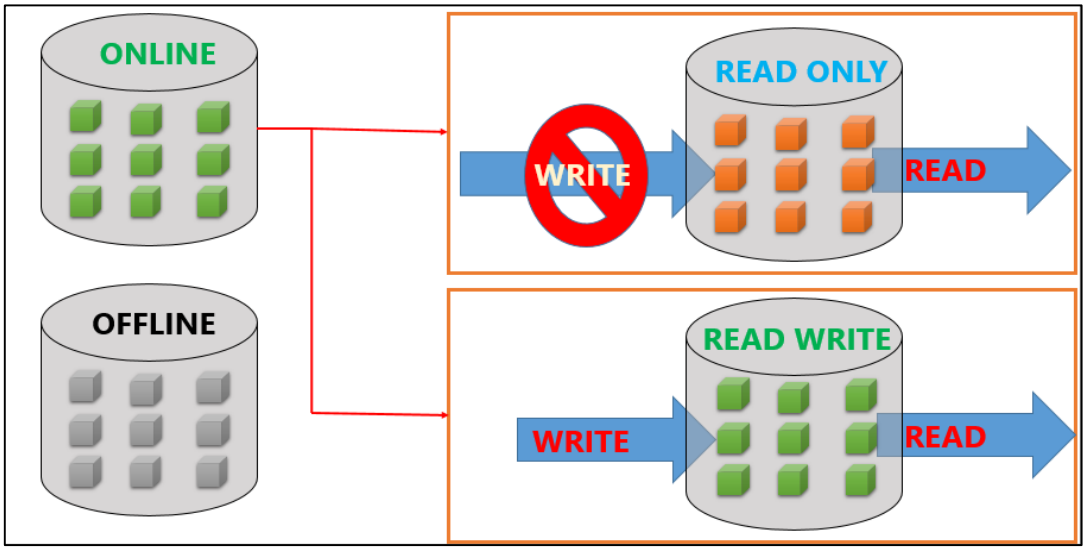
- Dữ liệu lưu trữ trong temporary tablespaces chỉ tồn tại trong một session.
- Dữ liệu trong temporary tablespaces được lưu trữ vật lý trong các temp files.
- Không chứa các đối tượng cố định (permanent objects)
- Được sử dụng để dành riêng cho các thao tác sắp xếp dữ liệu
- Nâng cao hiệu suất thực hiện 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

2.3.1.3. Các trạng thái của 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.



Hình 51.*Mô hình phân loại các trạng thái của Tablespaces*

- Không thể truy cập dữ liệu khi offline.
- Các Tablespaces không thể offline:
  - SYSTEM tablespace.
  - Default Undo Tablespace.
  - Temporary tablespace.
- Các Tablespaces không thể read only:
  - SYSTEM, SYSAUX tablespace.
  - Undo Tablespace.
  - Temporary tablespace.
- Cú pháp chuyển đổi các trạng thái của tablespaces:

```
ALTER TABLESPACE tablespace_name online|offline|read only|read write;
```

2.3.1.4. Thêm, sửa, xóa Tablespaces

a. Tạo mới tablespaces

Cú pháp: CREATE TABLESPACE ***tablespace_name*** DATAFILE ***clause***;

Trong đó:

tablespace_name : Tên tablespace

clause: 'đường_dẫn_file' SIZE kích_thước **K|M .** Có thể có nhiều datafile, phân cách nhau bởi dấu phẩy.

Ví dụ: Tạo permanent tablespace có tên **userdata** gồm 2 datafile kích thước lần lượt là 10M và 20M.

Create tablespace userdata datafile '%oracle_home%\oradata\usedata1.dbf' size 10M, '%oracle_home%\oradata\usedata2.dbf' size 20M;

b. Mở rộng kích thước tablespaces

Một tablespace có thể mở rộng kích thước bằng 2 cách:

❖ Cách 1: Thay đổi kích thước của data file:

+ Sử dụng tự động mở rộng kích thước với mệnh đề AUTOEXTEND.

Cú pháp: AUTOEXTEND {OFF|ON[NEXT integer[K|M]] [MAXSIZE UNLIMITED|integer[K|M]]}

Mặc định:

- Không tự động mở rộng data file khi không có mệnh đề này trong câu lệnh tạo tablespaces.
- Maxsize ở chế độ **unlimited** khi không thêm mệnh đề MAXSIZE
- Tự động mở rộng 1 MB khi không có mệnh đề NEXT

Mệnh đề đi sau các câu lệnh:

CREATE DATABASE

CREATE TABLESAPCE ... DATAFILE

ALTER TABLESAPCE ... ADD DATAFILE

ALTER DATABASE DATAFILE

Ví dụ: CREATE TABLESPACE user_data DATAFILE 'C:/userdata01.dbf' SIZE 20M AUTOEXTEND ON NEXT 1M MAXSIZE 50M;

+ Sử dụng mở rộng bằng tay ALTER DATABASE (RESIZE).

Cú pháp: ALTER DATABASE DATAFILE 'filename'[, 'filename'] RESIZE integer[K|M]

Ví dụ: ALTER DATABASE DATAFILE '/oradata/userdata02.dbf' RESIZE 200M;

❖ Cách 2: Thêm một data file vào tablespace

Cú pháp: ALTER TABLESPACE *tablespace_name* ADD DATAFILE 'filename'[, 'filename'] SIZE integer[K|M]

Ví dụ: ALTER TABLESPACE user_data ADD DATAFILE '/oradata/userdata03.dbf' SIZE 200M;

c. Đổi tên hoặc thay đổi vị trí của datafiles

❖ Cách 1. Sử dụng lệnh ALTER TABLESPACE

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

Ví dụ: ALTER TABLESPACE userdata RENAME DATAFILE '/u01/oradata/userdata01.dbf' TO '/u02/oradata/userdata01.dbf';

Thực hiện theo các bước sau:

1. Chuyển chế độ offline cho tablespace.
2. Di chuyển hoặc đổi tên các data files tương ứng bằng lệnh của hệ điều hành. (Sử dụng Windows Explorer)
3. Thực hiện lệnh ALTER TABLESPACE RENAME DATAFILE.
4. Chuyển lại chế độ online cho tablespace đó.

❖ Cách 2. Sử dụng lệnh ALTER DATABASE

Cú pháp: `ALTER DATABASE RENAME FILE 'filename'[, 'filename']... TO 'filename'[, 'filename']...`

Ví dụ: ALTER DATABASE RENAME FILE '/u01/oradata/system01.dbf'

TO '/u03/oradata/system01.dbf';

Ta thực hiện theo các bước sau:

1. Shutdown database.
2. Di chuyển hoặc đổi tên data files bằng lệnh của hệ điều hành.
3. Khởi động lại database ở chế độ Mount.
4. Thực hiện lệnh `ALTER DATABASE RENAME FILE`.
5. Mở lại database.(Alter database open)

d. Xóa tablespaces

- Không thể xóa tablespace nếu đó là:
  - SYSTEM, SYSAUX tablespace.
  - Defaut temporary hoặc undo tablespace
- Lệnh INCLUDING CONTENTS để xóa tablespace khi tablespace có dữ liệu.
- INCLUDING CONTENTS AND DATAFILES xóa cả các data file.
- CASCADE CONSTRAINTS hủy tất cả các ràng buộc có liên quan tới các bảng bên ngoài tablespace.

- Ví dụ: DROP TABLESPACE userdata INCLUDING CONTENTS AND DATAFILES;

2.3.1.5. Khôi phục datafile bị mất

B1. Tạo lại datafile

Cú pháp:

Alter database create datafile 'full_path_file_name';

B2. Khôi phục dữ liệu

Cú pháp:

Recover datafile 'full_path_file_name';

2.3.1.6. Truy vấn thông tin về Tablespaces

❖ **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
`STATUS`	Trạng thái của tablespace là Online hay Offline
`CONTENTS`	Phân loại tablespace là permanent hay temporary

VD: SELECT tablespace_name, contents,status from dba_tablespaces;

```
TABLESPACE_NAME                CONTENTS  STATUS
------------------------------ --------- ---------
SYSTEM                         PERMANENT ONLINE
UNDOTBS1                       UNDO      ONLINE
SYSAUX                         PERMANENT ONLINE
TEMP                           TEMPORARY ONLINE
USERS                          PERMANENT ONLINE
EXAMPLE                        PERMANENT ONLINE
```

❖ **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 datafile
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: `SELECT file_name, tablespace_name, bytes FROM dba_data_files;`

```
FILE_NAME
--------------------------------------------------
TABLESPACE_NAME                     BYTES
------------------------------ ----------
C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\USERS01.DBF
USERS                             5242880

C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\SYSAUX01.DBF
SYSAUX                          346030080

C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\UNDOTBS01.DBF
UNDOTBS1                         31457280

C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\SYSTEM01.DBF
SYSTEM                          597688320

C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\EXAMPLE01.DBF
EXAMPLE                         104857600
```

2.3.1.7. Bài tập thực hành

Bài 1. Tạo các permanent tablespaces với các thông tin như sau:

a. Tablespace name: **DATA01**
   Data file name: **data01.dbf**
   size: 5M
   location: %oracle_home%\oradata

b. Tablespace name: **DATA02**
   Data file name: **data02.dbf**

size: **10M**

location: %oracle_home%\oradata

c. Tablespace name: **INDEX01**

Data file name: **index01.dbf**

Size: **10M**

Location: c:\oracle\oradata

Tự động mở rộng 500K, dung lượng tối đa datafile là 50M

Bài 2. Cấp phát thêm 5MB dung lượng trống cho tablespace DATA02. Hiển thị kết quả thu được.

Bài 3. Di chuyển datafile trong tablespace INDEX01sang thư mục %oracle_home\oradata.

Bài 4. Đăng nhập vào user SYSTEM, tạo bảng TEST trong tablespace DATA01 như sau: *Create table TEST(id number(5)) tablespace DATA01;*

Chuyển trạng thái tablespace DATA01 sang READ ONLY. Insert dữ liệu vào bảng TEST. Điều gì xảy ra?

Bài 5. Chuyển trạng thái DATA01 sang READ WRITE .

Bài 6. Liệt Thông tin về tên vị trí lưu trữ của các datafile trong mỗi tablespace.

Bài 7. Hiển thị tên, số datafile, trạng thái của tablespace.

Bài 8. Kiểm tra về đặc tính AutoExtend của mỗi datafile.

Bài 9. Kiểm tra dung lượng, ngày tạo của mỗi datafile.

Bài 10. Hiển thị tổng dung lượng của các datafile có trong mỗi tablespace.

Bài 11. Liệt kê các datafile chứa trong thư mục C:\ORACLE. Gợi ý: truy vấn trong view dba_data_files.

Bài 12. Xóa tablespace DATA01.

Bài 13. Thông tin về tên vị trí lưu trữ của các datafile trong mỗi tablespace.

Bài 14. Hiển thị tên, số datafile, trạng thái của tablespace.

Bài 15. Kiểm tra về đặc tính AutoExtend của mỗi datafile.

Bài 16. Kiểm tra dung lượng, ngày tạo của mỗi datafile.

Bài 17. Hiển thị tổng dung lượng của các datafile có trong mỗi tablespace.

2.3.2. Quản lý quyền, chức danh

2.3.2.1. Quản lý quyền (Privilege)

❖ **Khái niệm:** Quyền của một user trong CSDL Oracle là một sự cho phép thực hiện 1 câu lệnh SQL hoặc được phép truy xuất đến một đối tượng nào đó.

VD: quyền tạo bảng CREATE TABLE, quyền đăng nhập vào cơ sở dữ liệu CREATE SESSION, quyền SELECT trên một bảng cụ thể nào đó,…).

❖ Phân loại quyền:

Oracle có 2 loại quyền cho user:

- **Quyền hệ thống (System Privilege):** Cho phép user thực hiện các thao tác cụ thể trong CSDL.
- **Quyền đối tượng (Object Privilege):** Cho phép user truy xuất và thao tác trên một đối tượng cụ thể.

❖ Quyền hệ thống (Sytem Privilege)

Với phiên bản Oracle 10G thì có hơn 100 quyền hệ thống khác nhau.

❖ 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: CREATE SESSION, CREATE TABLESPACE, CREATE USER.
- Các quyền cho phép quản lý các đối tượng thuộc về một user. Ví dụ: CREATE TABLE, CREATE TRIGGER,…
- Các quyền cho phép quản lý các đối tượng trong bất cứ một schema nào. Ví dụ câu lệnh: CREATE ANY TABLE, …

Bảng 11.*Một số quyền hệ thống*

Phân loại	Các quyền hệ thống
USERS	CREATE USER ALTER USER DROP USER
PROCEDURES	CREATE PROCEDURE CREATE ANY PROCEDURE ALTER ANY PROCEDURE DROP ANY PROCEDURE EXECUTE ANY PROCEDURE
PRIVILEGE	GRANT ANY OBJECT PRIVILEGE GRANT ANY PRIVILEGE
ROLE	CREATE ROLE ALTER ANY ROLE DROP ANY ROLE GRANT ANY ROLE

TABLE	CREATE TABLE CREATE ANY TABLE ALTER ANY TABLE DROP ANY TABLE SELECT ANY TABLE UPDATE ANY TABLE INSERT ANY TABLE DELETE ANY TABLE
SESSION	CREATE SESSION ALTER SESSION RESTRICTED SESSION
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE UNLIMITED TABLESPACE

Các quyền liên quan đến việc tạo các đối tượng như CREATE TABLE, CREATE PROCEDURE,.. thì bao gồm cả các quyền xoá các đối tượng đó. Để user có thể tạo đối tượng, ngoài được cấp quyền tạo đối tượng, Các user cần có đủ quota trên tablespace hay phải được gán quyền UNLIMITED TABLESPACE.

❖ **Gán các quyền hệ thống cho user:**

Cú pháp:

GRANT system_privilege[,system_privilege]... **TO {**user|Public**}** **[,**user**]...[WITH ADMIN OPTION]**

Trong đó:

System_privilege: Tên các quyền hệ thống

User: Tên user được gán quyền

Public: Chỉ định quyền được gán cho tất cả các user.

WITH ADMIN OPTION: Cho phép user được gán quyền có thể gán tiếp quyền đó cho user khác.

Ví dụ:

- Gán quyền đăng nhập vào CSDL cho emi:

  Grant create session to emi;

- Gán quyền tạo bảng cho emi sao cho emi có thể gán quyền cho user khác:

Grant create table to emi with admin option;

❖ **Thu hồi các quyền hệ thống của user:**

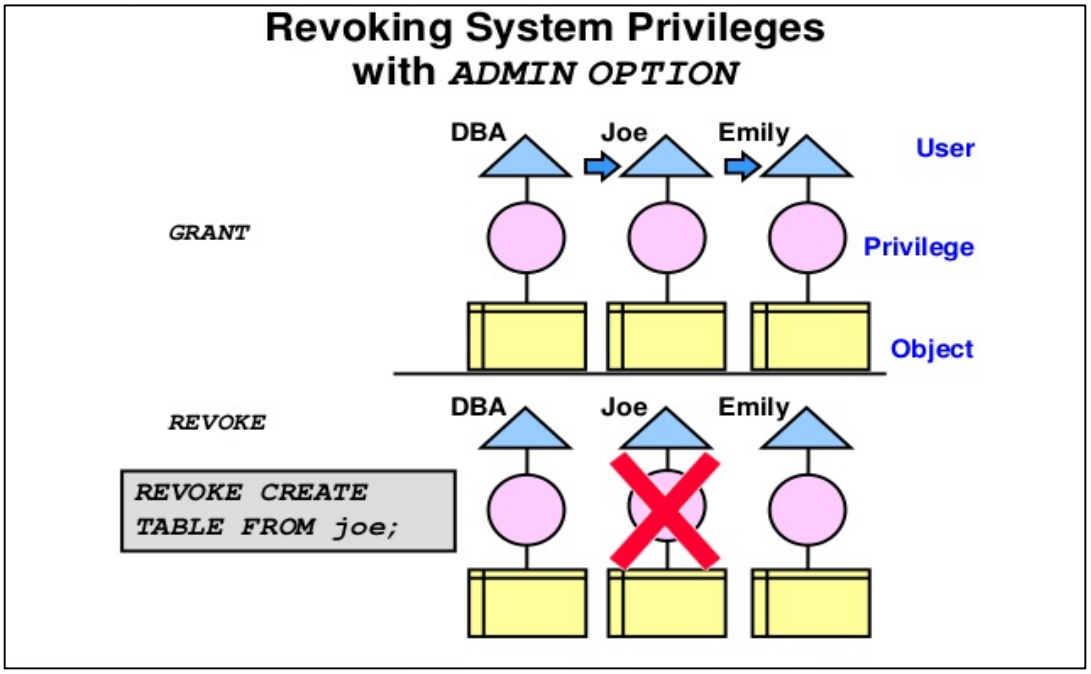
Cú pháp:

REVOKE system_privilege [,system_privilege]... **FROM** {user|**PUBLIC**} [, user]...

- Sử dụng câu lệnh REVOKE để thu hồi một quyền hệ thống khỏi user.
- Các user được gán quyền hệ thống với tùy chọn ADMIN OPTION có thể thu hồi quyền hệ thống đó của bất kỳ user.
- Ví dụ: Thu hồi quyền tạo bảng của emi: *Revoke create table from emi;*

Lưu ý:

- Chỉ các quyền được gán qua câu lệnh GRANT mới có thể bị thu hồi.
- 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 sở hữu 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.



Hình 52.*Minh hoạ việc thu hồi các quyền hệ thống*

Không có sự ảnh hưởng lan truyền khi thu hồi quyền hệ thống.

❖ **Thông tin về các quyền hệ thống**

Thông tin về các quyền hệ được lấy từ các view data dictionary: `DBA_SYS_PRIVS` và `SESSION_PRIVS`.

Các thông tin bao gồm:

- DBA_SYS_PRIVS: GRANTEE (tên user hoặc role được gán quyền), PRIVILEGE (tên quyền), ADMIN OPTION (có hay không tùy chọn admin)
- SESSION_PRIVS: PRIVILEGE

Ví dụ:

Liệt kê các quyền hệ thống được gán cho user và role:

```
SVRMGR>SELECT * FROM DBA_SYS_PRIVS;
```

View SESSION_PRIVS liệt kê các quyền có sẵn cho session hiện tại cho một user.

❖ **Quyền đối tượng (Object Privilege)**

Mỗi quyền trên đối tượng cho phép người dùng được gán thực thi một số thao tác trên đối tượng đó. Các đối tượng bao gồm table, view, index, synonym, sequences, PL/SQL functions, procedures, packages.

User sở hữu một đối tượng có tất cả các quyền đối tượng trên đối tượng đó, và những quyền này không thể thu hồi. User sở hữu đối tượng có thể cấp quyền trên đối tượng đó cho các user khác. Một người dùng với quyền ADMIN có thể cấp và thu hồi quyền đối tượng từ các user mà không sở hữu các đối tượng đó.

Quyền	Loại đối tượng	Mô tả
SELECT	Table, sequence, view, synonym	Cho phép một user select từ bảng, sequence, view, synonym.
INSERT	Table or synonym	Cho phép một user thêm dữ liệu vào bảng hoặc vào bảng thông qua synonym (bí danh).
UPDATE	Table	Cho phép một user cập nhật dữ liệu trên bảng.
DELETE	Table	Cho phép một user xóa dữ liệu của bảng.
EXECUTE	PL/SQL package, procedure, function	Cho phép một user thực thi một PL/SQL package, procedure hoặc function.

Bảng: Quyền trên các đối tượng.

❖ **Gán các quyền đối tượng cho user:**

Cú pháp:

GRANT {object_privilege[,object_privilege] … |ALL}

ON [schema.]object **TO {**user|**PUBLIC}[,**user**]...**

[WITH GRANT OPTION]

Gán quyền đối tượng phải nằm trong schema của người gán hoặc người gán phải có tùy chọn WITH GRANT OPTION với quyền đó hoặc người gán có quyền hệ thống GRANT ANY OBJECT PRIVILEGE.

Ví dụ 1: Gán quyền select dữ liệu trên bảng customers của emi cho jeff.

Grant select on emi.customers to jeff;

Ví dụ 2: Gán quyền update trên bản customers của emi cho jeff sao cho jeff có thể gán tiếp quyền đó cho bob.

Grant update on emi.customers to jeff with grant option;

❖ **Thu hồi các quyền đối tượng của user:**

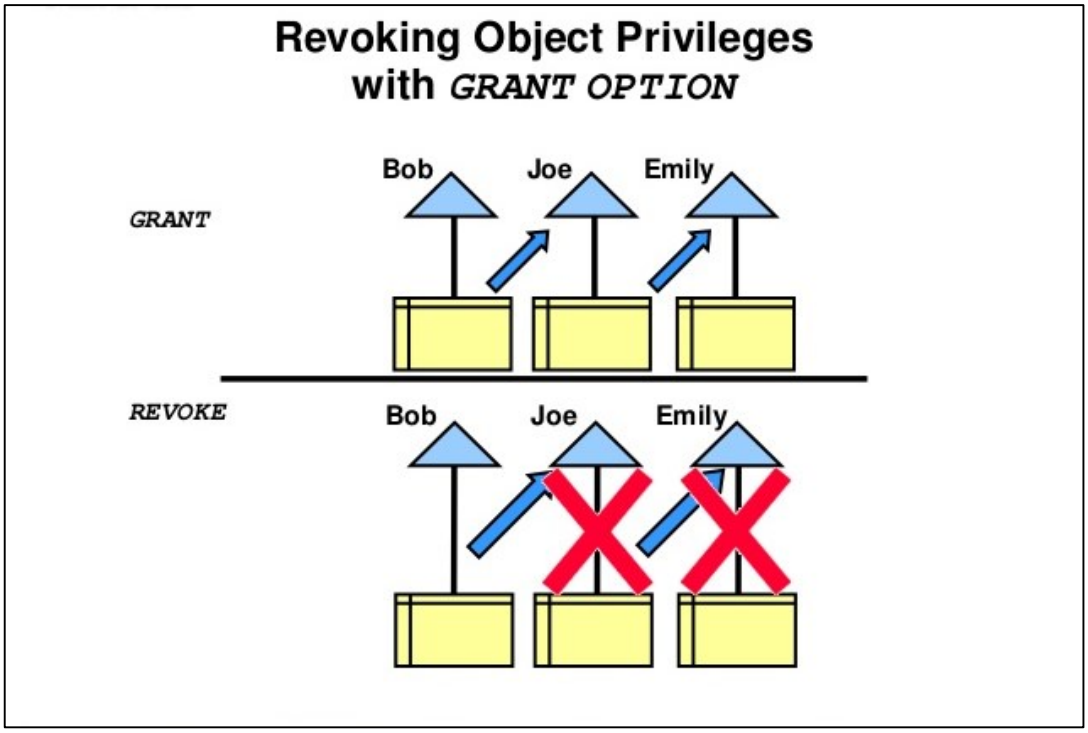
Cú pháp:

REVOKE object_privilege [,object _privilege]... **FROM** {user|**PUBLIC**} [, user]...

- Sử dụng câu lệnh REVOKE để thu hồi một quyền hệ thống khỏi user.
- Để một user REVOKE một quyền đối tượng từ một user khác, nó cần thỏa mãn một trong hai điều kiện sau:

  + Phải là user trực tiếp gán quyền đó

  + Có quyền hệ thống: GRANT ANY OBJECT PRIVILEGE
- Việc thu hồi các quyền đối tượng sẽ dẫn đến các việc thu hồi các quyền lan truyền.



Hình 53.*Minh họa việc thu hồi quyền đối tượng*

❖ **Thông tin về các quyền đối tượng**

Thông tin về các quyền đối tượng được lưu trữ trong các data dictionary.

Truy vấn thông tin trên bảng `DBA_TAB_PRIVS` để lấy thông tin về các quyền trên đối tượng được gán cho user.

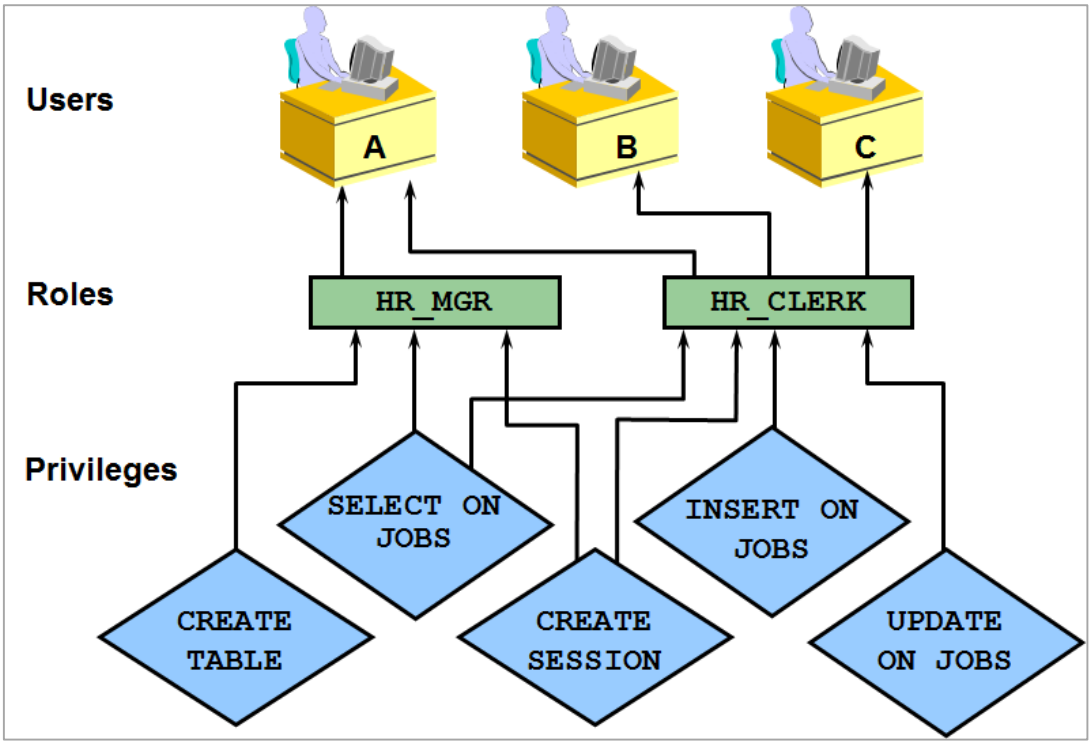
Bảng 12.*Thông tin quyền đối tượng*

Tên cột	Mô tả
GRANTEE	Tên user mà được gán quyền trên đối tượng.
OWNER	Tên user sở hữu đối tượng đó.
TABLE_NAME	Tên đối tượng.
GRANTOR	Tên user mà thực hiện việc gán. (Nếu là admin hoặc user có quyền grant any object privilege thì GRANTOR trùng với OWNER)
PRIVILEGE	Tên quyền được gán.
GRANTABLE	Cho biết user được gán quyền có thêm tùy chọn `GRANT OPTION` (`YES`) hay không (`NO`).

2.3.2.2. Quản lý chức danh (Role)

Khái niệm: Chức danh (Role) 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 54.*Minh họa chức danh trong Database*

❖ **Một số chức danh định sẵn**

ROLE NAME	**DESCRIPTION**
CONNECT	Từ phiên bản Oracle 10*g* Release 2, chức danh Connect chỉ còn 1 quyền là CREATE SESSION
RESOURCE	Bao gồm các quyền: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER,CREATE TYPE, UNLIMITED TABLESPACE (quyền này không hiện ra khi truy vấn trong data dictionary)
DBA	All system privileges WITH ADMIN OPTION

Bảng 13.*Một số chức danh định sẵn*

❖ **Các đặc điểm của chức danh**

1. Role có thể gán cho user hoặc role và thu hồi từ user hoặc role giống như gán và thu hồi quyền.
2. Role có thể gán cho bất kỳ user và role. Tuy nhiên, một role không thể gán cho chính nó hoặc gán vòng quanh.
3. Role có thể bao gồm cả quyền hệ thống và quyền đối tượng.
4. Tên của role phải duy nhất, không trùng tên với bất kỳ user hoặc role khác đã tồn tại trong CSDL.
5. Role không thuộc sở hữu của bất kỳ user và không được lưu trữ trong bất kỳ schema nào. Role được lưu trữ trong data dictionary.
6. Một user có thể enable, disable các role gán cho nó.
7. Khi một role có đặt mật khẩu, cần phải nhập mật khẩu khi enable role.

❖ **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.

❖ **Tạo và sửa chức danh**

Cú pháp:

CREATE/ALTER ROLE role_name [NOT IDENTIFIED|IDENTIFIED BY password]

Với:

CREATE/ALTER	Tạo/Sửa
role_name	Tên của chức danh

NOT IDENTIFIED chỉ định không cần nhập mật khẩu khi enable chức danh. Là mặc định nếu không chỉ rõ.

BY password mật khẩu người dùng cần cung cấp khi enable chức danh

Ví dụ: *CREATE ROLE sales_clerk;*

CREATE ROLE hr_clerk IDENTIFIED BY bonus;

Chú ý: Để tạo được chức danh, user phải có quyền hệ thống CREATE ROLE. Để chỉnh sửa được chức, user phải thõa mãn 1 trong các yêu cầu sau:

1. User phải có tùy chọn WITH ADMIN OPTION của chức danh đó.
2. User phải có quyền ALTER ANY ROLE

❖ Gán các chức danh

Cú pháp:

GRANT role [, role]... TO {user|role|PUBLIC}

[, {user|role|PUBLIC}]...[WITH ADMIN OPTION]

Để gán chức danh cho 1 user hoặc role khác, user phải có chức danh đó với tùy chọn WITH ADMIN OPTION hoặc có quyền hệ thống: GRANT ANY ROLE

Chú ý: User tạo ra chức danh thì mặc định có tùy chọn WITH ADMIN OPTION đối với chức danh đó.

Ví dụ: GRANT hr_manager TO scott WITH ADMIN OPTION;

GRANT sales_clerk TO scott;

❖ Đặc điểm của tùy chọn WITH ADMIN OPTION

Khi một quyền hoặc chức danh được cấp cho một user có thêm tùy chọn WITH ADMIN OPTION sẽ cho phép user đó:

- ✓ Cấp lại quyền/chức danh đó cho một user hoặc chức danh khác.
- ✓ Thu hồi lại quyền/chức danh đó từ một user hoặc chức danh bất kỳ.
- ✓ Thay đổi chức danh đó bằng lệnh ALTER ROLE.
- ✓ Xóa chức danh đó.

❖ **Gán quyền cho chức danh**

Gán quyền hệ thống cho chức danh

Cú pháp:

GRANT system_privilege [, system_privilege]... TO role [WITH ADMIN OPTION]

- Để gán một quyền hệ thống cho chức danh, user phải có tùy chọn WITH ADMIN OPTION với quyền hệ thống đó hoặc user phải có quyền GRANT ANY PRIVILEGE.
- VD: grant create table to manger_role;

Gán quyền đối tượng cho chức danh

Cú pháp:

GRANT object_privilege[,object_privilege]...TO role[,role]

Để gán 1 quyền đối tượng cho chức danh, user phải thỏa mãn 1 trong các yêu cầu sau:

- User sở hữu đối tượng đó.
- User có quyền đối tượng đó với tùy chọn WITH GRANT OPTION.
- User đó phải có quyền hệ thống GRANT ANY OBJECT PRIVILEGE.
- Không có tùy chọn GRANT OPTION khi gán 1 quyền đối tượng cho chức danh.

❖ **Thu hồi các chức danh khỏi User**

Thu hồi các chức danh khỏi user đòi hỏi tùy chọn ADMIN OPTION hoặc quyền GRANT ANY ROLE.

Cú pháp:

REVOKE role [,role]**FROM** {user|role|**PUBLIC**}[,user|role|**PUBLIC**}]

Ví dụ: Revoke oe_clerk from scott;

❖ **Xóa chức danh**

Cú pháp:

DROP ROLE role;

Khi xóa một chức danh thì:

- Hủy bỏ chức danh đó với tất cả user và các chức danh mà nó được gán.
- Hủy bó nó khỏi CSDL.

Để xóa được một chức danh, user phải có tùy chọn ADMIN OPTION với chức danh đó hoặc quyền DROP ANY ROLE.

❖ **Enable và Disable các chức danh**

Trong 1 session, user có thể enable và disable các role mà được gán cho nó qua mệnh đề SET ROLE.

Role có đặt mật khẩu cần phải chỉ rõ mật khẩu để enable role đó.

Cú pháp:

```
SET ROLE {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 hr_clerk;

SET ROLE oe_clerk IDENTIFIED BY order;

SET ROLE ALL EXCEPT oe_clerk;

❖ **Lấy thông tin 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.

Bảng 14.*Các view chứa thông tin về Role*

Tên view	Diễn giải
DBA_ROLES	Tất cả các chức danh trong database.
DBA_ROLE_PRIVS	Các chức danh đã được gán cho user hay chức danh khác
USER_ROLE_PRIVS	Các chức danh đã được gán cho user hiện tại.
DBA_SYS_PRIVS	Quyền hệ thống gán cho user hay chức danh
USER_SYS_PRIVS	Quyền hệ thống gán cho user hiện tại.
ROLE_TAB_PRIVS	Quyền đối tượng gán cho chức danh.
SESSION_ROLES	Các chức danh được enable của user hiện thời.

Ví dụ: Hiển thị các chức danh được gán cho user SCOTT:

```
SQL> select grantee, granted_role,admin_option from dba_role_privs where grantee='SCOTT';

GRANTEE                        GRANTED_ROLE                   ADM
------------------------------ ------------------------------ ---
SCOTT                          RESOURCE                       NO
SCOTT                          CONNECT                        NO
```

2.3.2.3. Bài tập thực hành

Bài 1. Hiển thị các quyền hệ thống của chức danh RESOURCE.

Bài 2. Hiển thị tất cả các user được gán quyền SELECT ANY TABLE. (Gợi ý: Truy vấn trong view dba_sys_privs, có liên quan đến view dba_roles).

Bài 3. Cho bảng Attendance

(ID INT PRIMARY KEY,

  Name NVARCHAR2

)

Làm các bước sau:

a. Tạo các role sau: DataEntry, Supervisor, và Management.

b. Gán John, Joe, và Lynn vào role DataEntry, gán Fred vào role Supervisor, và gán Amy và Beth vào role Management.

c. Cho role DataEntry các quyền SELECT, INSERT, và UPDATE trên bảng Attendance.

d. Cho role Supervisor các quyền SELECT và DELETE trên bảng Attendance.

e. Cho role Management quyền SELECT trên bảng Attendance.

f. Lần lượt kiểm tra kết quả phân quyền đã cấp cho các role

Bài 4. Tạo một user mới tên NameManager với password là pc123. Gán quyền update cho user này trên cột Name của bảng Attendance.

Bài 5. Tạo một chức danh gọi là DEV, mà sẽ cho phép người dùng được gán chức danh này có quyền tạo bảng và select dữ liệu từ bảng CUSTOMERS của Emi.

Bài 6. Cho đoạn code sau:

1 CONN giaovien/p123;

2 CREATE ROLE sinhvien;

3 GRANT select ON Attendance TO sinhvien WITH GRANT OPTION;

Đoạn code trên sẽ báo lỗi ở đâu và trong trường hợp nào?

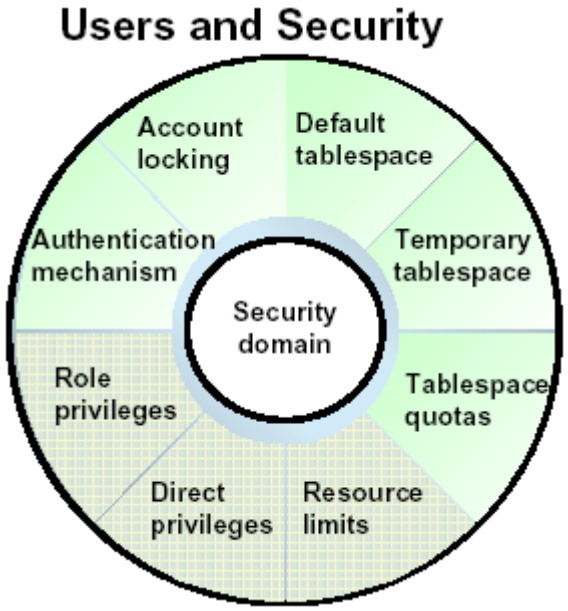
2.4. QUẢN TRỊ NGƯỜI DÙNG

2.4.1. User trong database

- **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 55.*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.

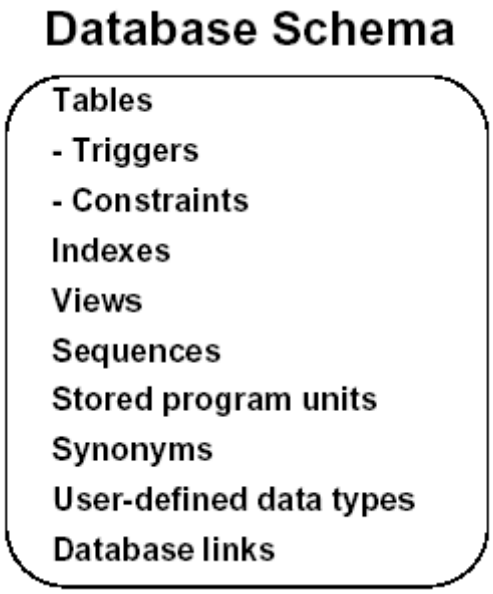
Resource Limits (hạn chế tài nguyên)

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,...

❖ **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 56.*Database schema*

2.4.2. Tạo mới User

❖ **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.

❖ **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. Mật khẩu của mỗi 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 [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 }]

Với:

Bảng 15.*Thông số khi tạo user*

USER	TÊN TRUY NHẬP CỦA USER
BY password	Xác định cơ chế xác nhận user bởi database với mật khẩu truy nhập là *password*. (*password* không phân biệt hoa thường)
DEFAULT TABLESPACE	Xác định tablespace mặc định được sử dụng để lưu trữ các đối tượng của user. **Chú ý:** nếu không chỉ rõ hạn mức trên default tablespace của user thì mặc định user được sử dụng unlimited không gian trên tablespace đó.
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.

ACCOUNT LOCK/ UNLOCK	Sử dụng tuỳ chọn này để lock/unlock đối với mỗi user một cách tường minh (mặc định là UNLOCK).

❖ **Gán quyền truy cập vào CSDL cho user:**

Khi 1 user mới được tạo ra, user đó chưa có bất cứ 1 quyền nào, chúng ta chưa thể đăng nhập vào user (vì chưa có quyền tạo session), vì vậy ta phải gán quyền create session hoặc chức danh connect cho user.

Cú pháp: Grant create session username;

❖ **Gán quyền tạo đối tượng cho user:**

Để có thể tạo bảng cũng như các đối tượng khác như sequence, trigger, procedure,… user phải được cấp quyền cấp quyền tương ứng.

Ví dụ: Cấp quyền tạo bảng cho user: ***Grant create table to*** *username****;***

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

2.4.3. Thay đổi thuộc tính của user

Ta sử dụng câu lệnh `ALTER USER` khi thực hiện các thay đổi đối với user:

+ Thay đổi mật khẩu
+ Khóa/Mở khóa user.
+ Thay đổi mật khẩu theo từng phiên làm việc.
+ Thay đổi hạn mức sử dụng trên các tablepsace.

Cú pháp:

ALTER USER user

[IDENTIFIED {BY password}]

[PASSWORD EXPIRE]

[ACCOUNT {LOCK | UNLOCK }]

[QUOTA {integer [K | M] | UNLIMITED } ON tablespace

[QUOTA {integer [K | M] | UNLIMITED } ON tablespace] ...];

Ví dụ 1:

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.

Khi đó người dùng chưa đăng nhập vào user nào.

Thay đổi hạn mức sử dụng trên các tablepsace.

Hạn mức sử dụng là phần không gian cấp phát cho một user được phép sử dụng trên 1 tablespace.

Ví dụ 1 : Thay đổi hạn mức sử dụng của user SCOTT trên tablespace USERS là 10MB.

ALTER USER scott QUOTA 10M ON users;

Ví dụ 2: Cho phép SCOTT sử dụng không giới hạn tablespace USERS:

ALTER USER scott QUOTA unlimited ON users;

Ví dụ 3: Thay đổi hạn mức sử dụng không gian trên tablespace data01 của user peter là 0 MB: **ALTER USER peter QUOTA 0 ON data01;**

Khi đó không thể cấp phát thêm không gian trống cho user để tạo các đối tượng trên tablespace data01.

Chú ý:

Sau khi thay đổi hạn mức của user trên tablespace là 0, các đối tượng thuộc sở hữu của user đó vẫn còn trong các tablespace, nhưng không thể cấp phát thêm một không gian mới để lưu trữ.

2.4.4. Hủy bỏ user

Huỷ bỏ user khỏi database. Cú pháp: `DROP USER user [CASCADE];`

Ví dụ:

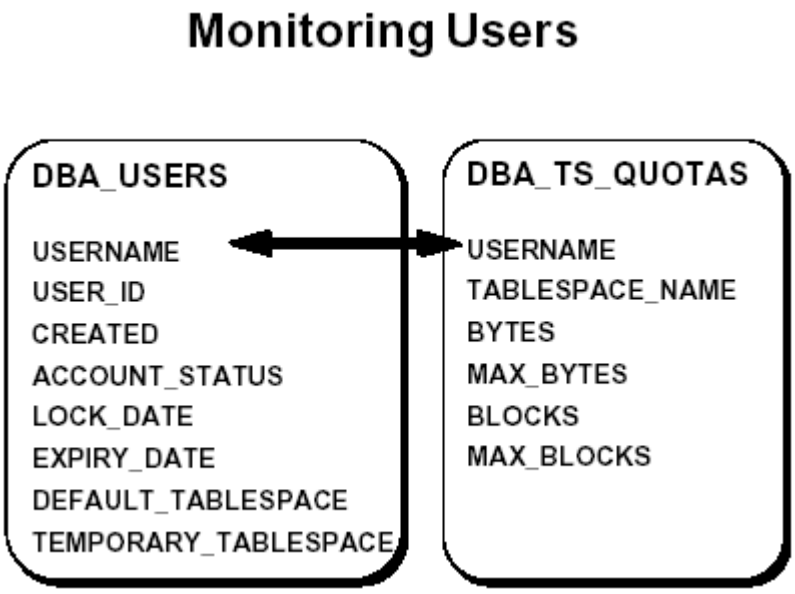
```
DROP USER peter;
Hoặc
DROP USER peter CASCADE;
```

Lưu ý:

Nếu schema có chứa đối tượng, cần phải có tùy chọn CASCADE, khi đó sẽ huỷ tất cả các đối tượng trong schema trước khi xoá User. Ta không thể huỷ được các user hiện đang kết nối tới Oracle server.

Ta không thể huỷ được các user hiện đang kết nối tới Oracle server.

2.4.5. Thông tin về user



Hình 57.*Thông tin về User trong data dictionary*

Ta có thể lấy các thông tin liên quan tới user trong data dictionary `DBA_USERS` và `DBA_TS_QUOTAS`.

Bảng 16.*Mô tả một số trường trong view DBA_USERS*

DBA_USERS	
USERNAME	Tên user
CREATED	Ngày tạo user
ACCOUNT_STATUS	Trạng thái của user (open, locked, expired,…)
DEFAULT_TABLESPACE	Tablespace mặc định cho user

Bảng 17.*Mô tả một số trường trong view DBA_TS_QUOTAS*

DBA_TS_QUOTAS	
USERNAME	Tên user
TABLESPACE_NAME	Tên tablespace mà user được cấp phát hạn mức sử dụng trên đó.
BYTES	Dung lượng tính theo byte user đã sử dụng trên tablespace.
MAX_BYTES	Dung lượng tối đa cho phép user sử dụng trên tablespace.

Ví dụ 1: Thông tin ngày tạo, trạng thái, tablespace mặc định của user SCOTT.

```
SQL> select username, created, account_status,default_tablespace from dba_users where username='SCOTT';

USERNAME                       CREATED   ACCOUNT_STATUS
------------------------------ --------- --------------------------------
DEFAULT_TABLESPACE
------------------------------
SCOTT                          30-AUG-05 OPEN
USERS
```

Hình 58.*Hình ví dụ 1*

Ví dụ 2: Hiển thị hạn mức sử dụng trên các tablespace của user SCOTT.

```
SQL> select username,tablespace_name,bytes,max_bytes from dba_ts_quotas where username='SCOTT';

USERNAME                       TABLESPACE_NAME                     BYTES
------------------------------ ------------------------------ ----------
 MAX_BYTES
----------
SCOTT                          USERS                              589824
   1048576
```

Hình 59.*Hình ví dụ 2*

2.4.6. Bài tập thực hành

Bài 1. Tạo người dùng Bob với mật khẩu là CRUSADER sao cho tất cả các đối tượng được tạo ra bởi Bob được nằm trong tablespace USERS và chắc chắn rằng Bob có thể đăng nhập và được phép tạo các đối tượng có tổng dung lượng là 1MB trong tablespace USERS.

Bài 2. Tạo một người dùng Emi với mật khẩu là MARY. Hiển thị các thông tin về Bob và Emi trong từ điển dữ liệu.

Bài 3. Hiển thị hạn mức sử dụng trong các tablespace mà Bob được sử dụng.

Bài 4. Thay đổi mật khẩu Bob thành SAM. (Gợi ý: Cách 1: Đăng nhập vào SYSTEM, thay đổi mật khẩu của Bob bằng mệnh đề alter user. Cách 2: Đăng nhập vào Bob với mật khẩu cũ, sử dụng mệnh đề alter user để thay đổi mật khẩu).

Bài 5. Đăng nhập vào tài khoản SYSTEM , hủy bỏ hạn mức sử dụng trên tablespace mặc định của Bob.

Bài 6. Xóa bỏ tài khoản Emi trong cơ sở dữ liệu.

Bài 7. Bob đã quên mật khẩu của mình. Quản trị viên hãy gán lại mật của anh ta thành OLINK và yêu cầu Bob thay đổi mật khẩu của mình khi anh ta đăng nhập vào CSDL.

Chương 3.
XÂY DỰNG ỨNG DỤNG TRÊN ORACLE DEVELOPER SUITE

3.1. ORACLE DESIGNER

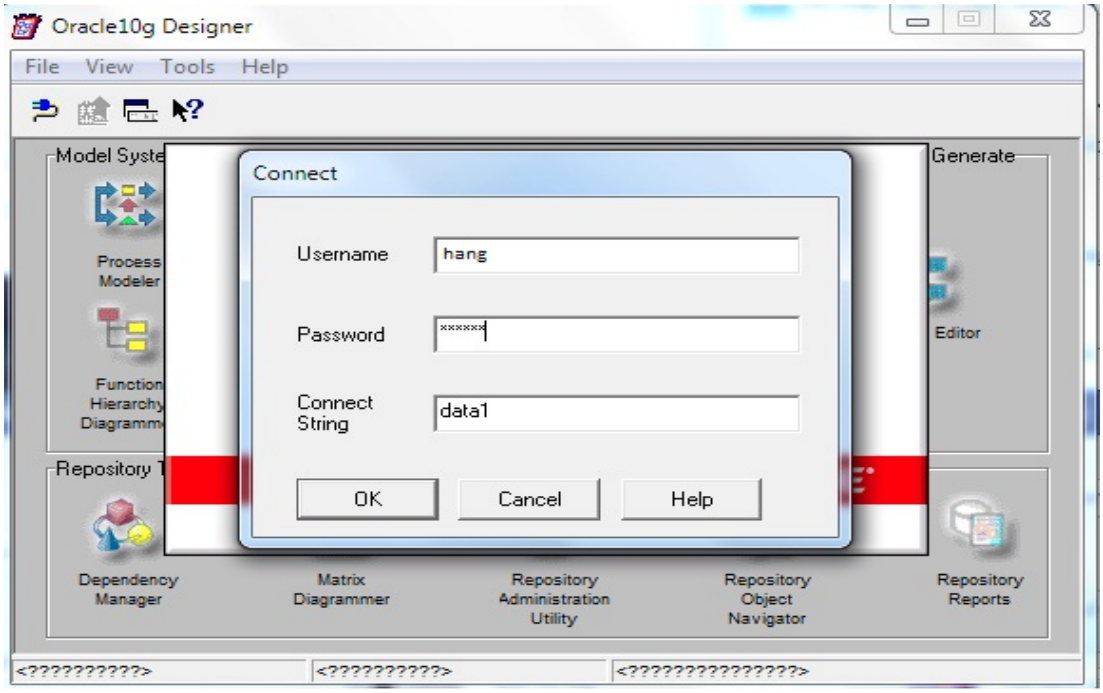
3.1.1. Khái niệm

Oracle Designer là một công cụ tích hợp hỗ trợ các nhiệm vụ trong các hoạt động phân tích và thiết kế, chiến lược CSDL, giúp người dùng thiết kế các vòng đời phát triển hệ thống và sau đó tạo ra các hình thức hoạt động đầy đủ. Giống như các công cụ khác trong Oracle, Oracle Designer có một kho lưu trữ tập trung.

3.1.2. Các thành phần

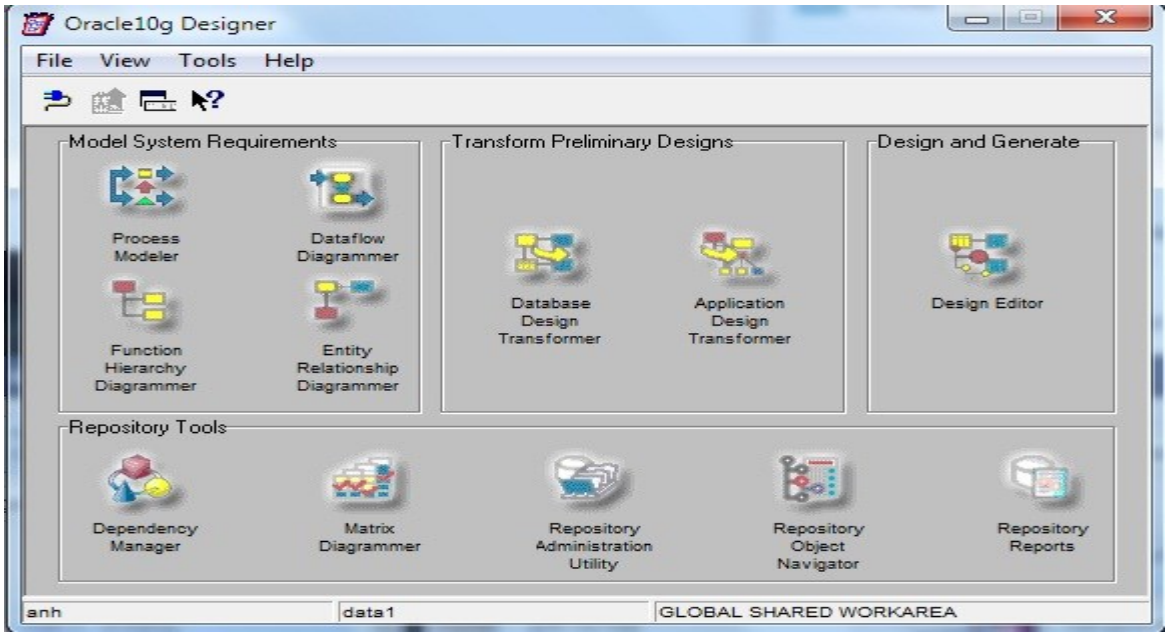
Các thành phần của Oracle designer bao gồm: Systems modeling, server designer ang generation, form design and generation, web PL/SQL design and generation, repository management.

Khi người dùng muốn sử dụng công cụ Oracle Designer thì phải đăng nhập vào hệ thống:



Hình 60.*Giao diện đăng nhập*

Sau khi đăng nhập thành công thì Oracle Designer có các công cụ hỗ trợ sau:



Hình 61.*Các thành phần của Oracle designer*

Các công cụ oracle designer được phân chia thành các khu vực phản ánh yêu cầu của người dùng.

Yêu cầu hệ thống mô hình hóa (Modeling System Requirement): Trong khu vực này sử dụng các công cụ để tiến trình các mô hình kinh doanh, tạo đại diện các sơ đồ kinh doanh, chi tiết bản ghi, mô tả các yêu cầu kinh doan, tạo mô hình sơ đồ liên kết chức năng, liên kết dữ liệu.

Tạo ra các thiết kế sơ bộ (Generating Preliminary Designes): Sử dụng các máy biến áo để tạo ra hiết kế sơ bộ từ các mô hình tạo ra trước nó.

Thiết kế và tạo (Designing and Generating): Trong khu vực này các công cụ được sử dụng để thiết kế một hệ thống đáp ứng các yêu cầu kinh doan của một tổ chức. Ngoài ra nó còn cung cấp môi trường cho nhà thiết kế hay kỹ sư tạo ra các thành phần phía máy swerver và các thành phần phía máy client từ các định nghĩa được ghi trong Repository.

Utilities: Trong khu vực này các công cụ dung để nhập và chỉnh sửa thông tin, hiển thị các mối quan hệ giữa các yếu tố trong Repository. Ngoài ra còn tạo báo cáo, viết truy vấn trong SQL …

3.1.3. Vai trò

Oracle Designer là công cụ phần mềm để phân tích các yêu cầu kinh doanh và thiết kế, tạo ra hệ thống Client/server đáp ứng những yêu cầu. Oracle design cung cấp một Repository nhiều người sử dụng và tích hợp chặt chẽ với Oracle Developer. Oracle cho phép các tổ chức thiết kế và cung caaos các hệ thống mở rộng mà hệ thống client/server có thể thích ứng với sự thay đổi của các nhà kinh doanh.

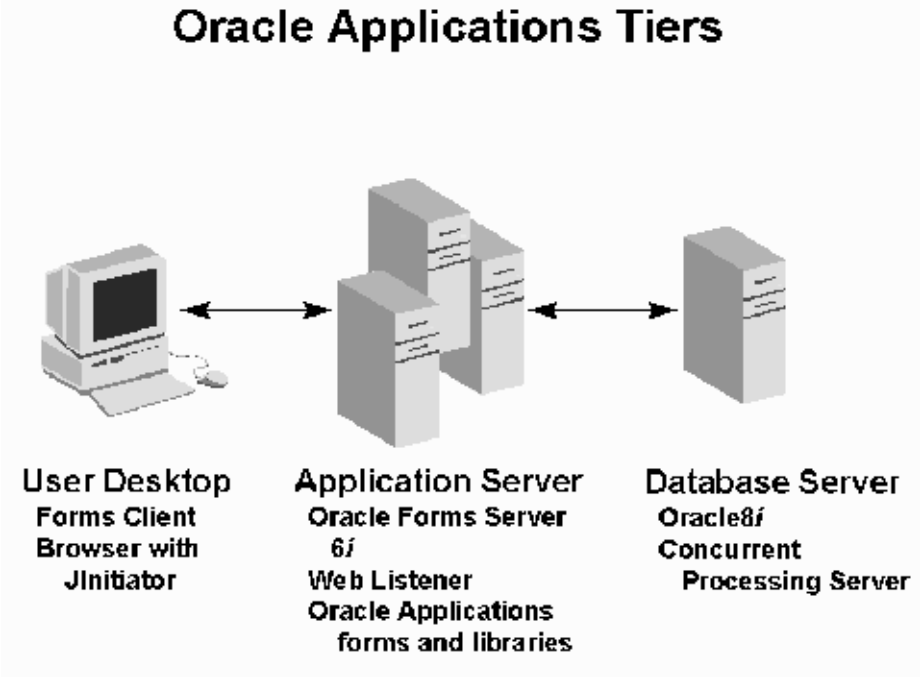
3.2. ORACLE FORM

Oracle Form là một trong những sản phẩm quan trọng trong bộ công cụ Oracle Developer Suite với mục đích dùng để thiết kế, xây dựng các ứng dụng trong doanh nghiệp. Các ứng dụng được xây dựng trên Oracle Form Builder được sử dụng trong các hệ thống máy tính lớn, hoạt động trên môi trường web theo mô hình client – server.

Oracle Form cung cấp các phương tiện phát triển giao diện, các xử lý, các thao tác với dữ liệu trong database và có khả năng kết nối, trao đổi thông tin với các ứng dụng khác của Oracle như là Oracle Report, Oracle Graphic.

Kiến trúc ứng dụng của hệ thống Oracle Application theo mô hình tập trung 3 lớp, đó là các lớp Desktop Client, Middle Tier (Form Server) và Database Server. Mọi thao tác ở Client dều thực hiện dựa trên hệ thống Java nhúng. Dữ liệu nhập liệu được truyền về Application Server, việc xử lí các chức năng được thực hiện ở Application Server. Việc thao tác xử lí dữ liệu thực hiện ở Database Server.

Kiến trúc tổng thể được mô tả như hình 50.



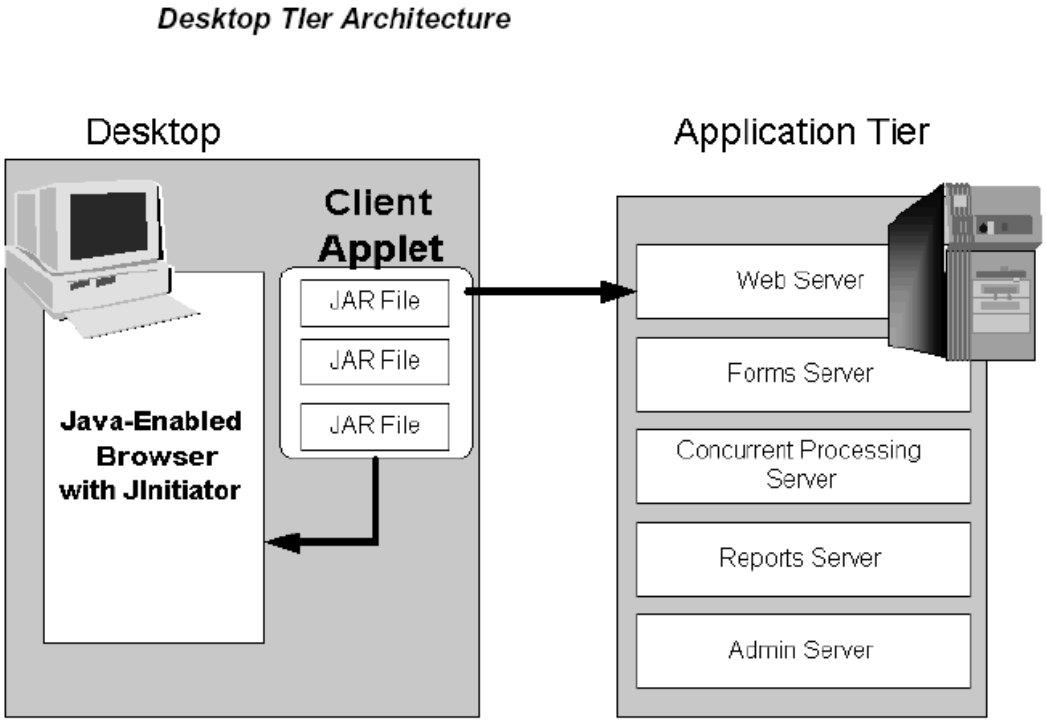
Hình 62.*Kiến trúc tổng thể Oracle Application*

- **Desktop Client :**

- Giao diện đồ họa, sử dụng trên các Web browser chuẩn có hỗ trợ Java nhúng chạy Jinitiator.

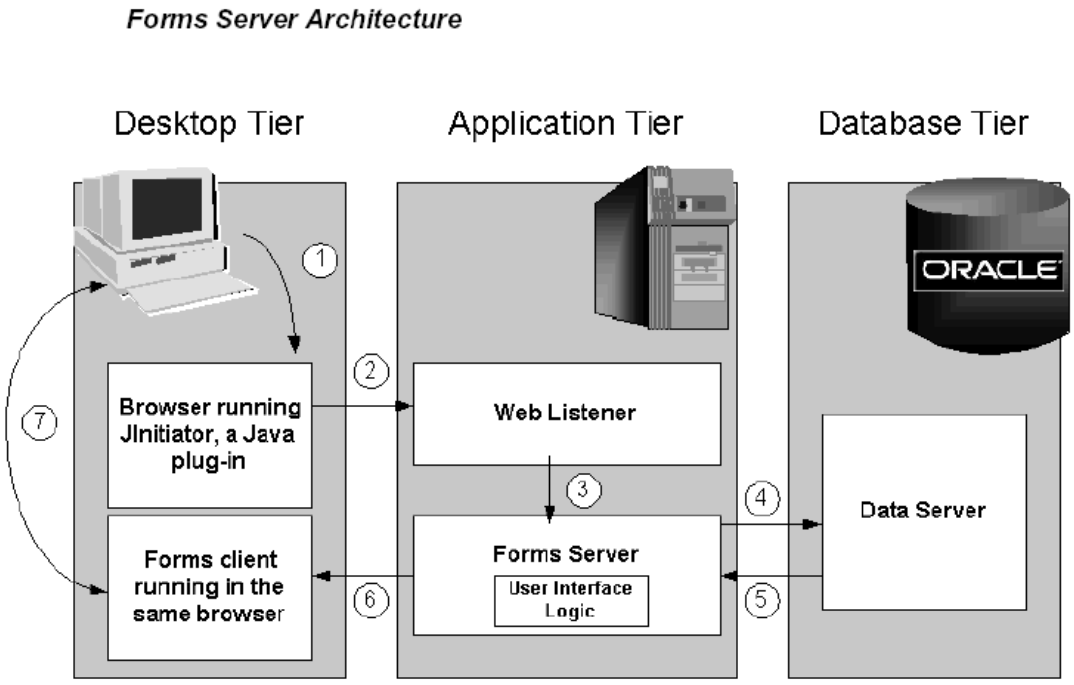
- Oracle JInitiator là phiên bản Oracle của JavaSofts Java Plug-In, cho phép xử lý JVM tại Web client thay vì dựa vào JVM mặc định của browser.

- Có thể tuong thích với mọi PC,network computer hoặc trên bất cứ giao diện nào có Java.



Hình 63.*Kiến trúc tầng Desktop*

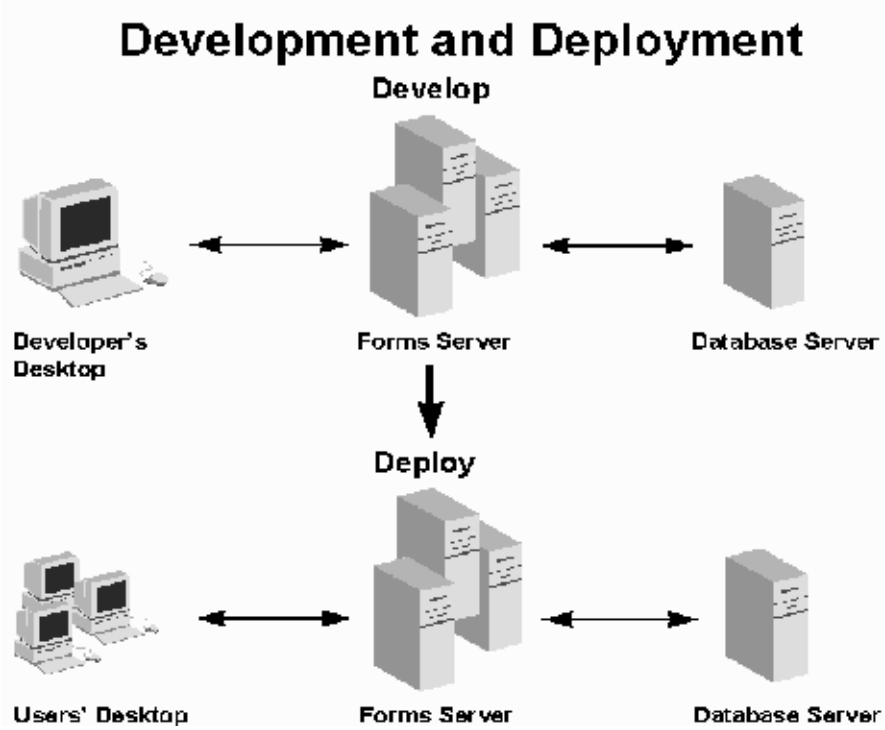
- **Middle Tier (Form Server):**
  - Xử lý các đáp ứng logic của Form.
  - Là nơi lưu giữ các dữ liệu cache cục bộ.
  - Trung gian giữa hai tầng.



Hình 64.*Kiến trúc tầng giữa*

- **Database Server:**
  - Sử dụng các lời gọi hàm từ xa (RPC) để giao tiếp với DB khi cần thiết (có thể thực hiện nhiều DB action cùng một lúc – các câu lệnh SQL).
  - Xử lý thao tác dữ liệu trên ứng dụng.
  - Lưu các thủ tục trên DB nên giao tiếp giữa các stored procedure và DB xảy ra trong bộ nhớ chứ không phải trên mạng.

❖ *Mô hình phát triển và triển khai Oracle Application*



Hình 65.*Mô hình phát triển và triển ứng dụng*

Quá trình phát triển form bao gồm ba bước chính là:

• Build

Generate

• Run/Test

Build Form : xây dựng form, thiết lập các thông số

Generating Your Form : (dịch Form trên Forms Server) Biên dịch file .fmb thành file chạy .fmx

Running Your Form for Testing : test trên môi trường Web chứ không phải trên Form developer

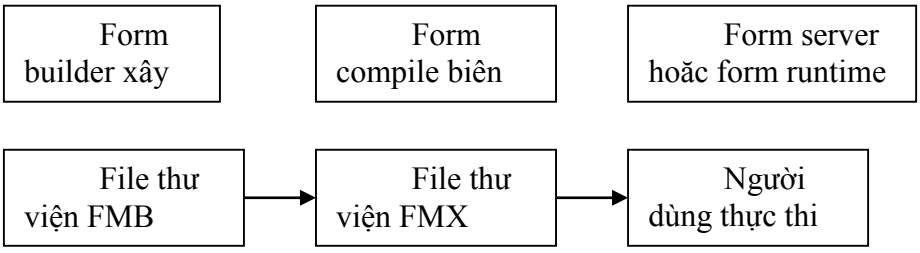
Trong Oracle Form 10g có 3 thành phần chính là:

Form Builder: hay còn gọi là môi trường phát triển tích hợp. công cụ này cung cấp các thành phần thiết kế mong muốn như là thiết kế giao diện, thiết kế menu, thiết kế library.

Form compiler: thành phần này giúp ta biên dịch các file tạo bởi Oracle builder thành các file có thể thực thi được.

Oracle Server: Thành phần này giúp ta thực thi file được tạo ra bởi Form compiler trong môi trường web. Nó là một sản phẩm trung gian để nhận các các yêu cầu từ một trình duyệt web và dọn sẵn một java applet dựa trên form cho trình duyệt.

Form Server có thể được gọi từ trình duyệt web hoặc có thể được gọi trong form builder



❖ *Môi trường phát triển của Oracle Form*

Các ứng dụng chạy trên nền Web và Internet đang được Oracle Form tập trung phát triển, xây dựng. Khi sử dụng Web mục đích chính của Oracle đó là loại bỏ được sự phụ thuộc và hệ điều hành. Và khi đó chúng chỉ phụ thuộc vào trình duyệt Web và Web Server.

Để thực thi một Web Form, người dùng yêu cầu 1 URL trong trình duyệt Web của mình. URL chỉ đến một ứng dụng được đăng kí bên trong Web Server. Listener trên Web Server nắm các yêu cầu của URL rồi chuyển nó đến Form Server. Form server tìm kiếm các file có dạng *.fmx rồi đổi thành một file java applet và cuối cùng gửi tới trình duyệt web. Và để thực thi form trong trình duyệt Web các client có một applet được gọi là Jinitiator. Khi Form Oracle thực thi thông qua Web, Web Server lưu trữ form và gửi Jinitiator tới trình duyệt Web của Client.

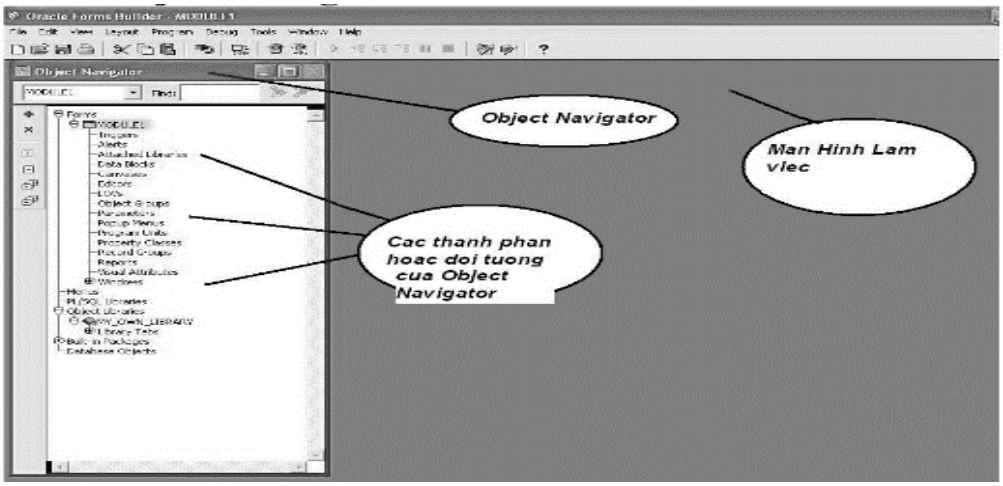
3.2.1. Các thành phần Form Builder

Form buider có các công cụ để xây dựng form: object navigator, layout editor, property palette và PL/SQL editor.

Có 2 cách để gọi Oracle Forms Builder

- Tìm đến biểu tượng Forms Builder và nháy đúp con trỏ trên biểu tượng.
- Chạy file frmbld.exe trong thư mục bin của thư mục đã cài bộ Oracle Developer Suite (ví dụ: C:\DevSuiteHome_1\BIN).

❖ **Object Navigator**



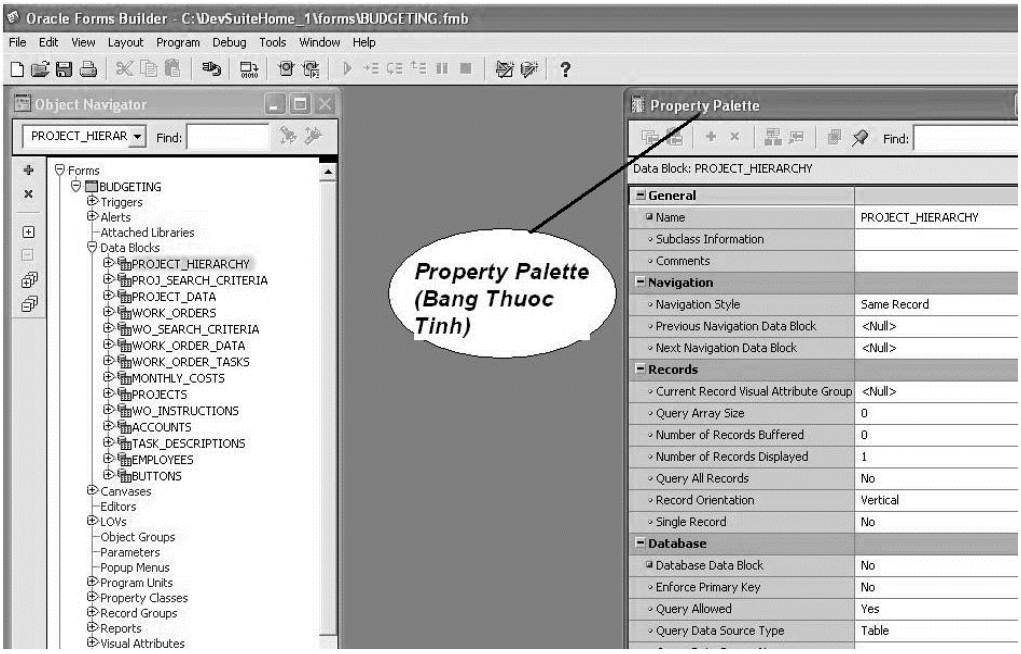
Hình 66.*Object Navigator*

Là nơi định vị và di chuyển tới bất kỳ thành phần nào của form. Click đúp bất kỳ đối tượng nào trong cửa sổ object navigator làm cho form builder thể hiện đối tượng được chọn bên trong một công cụ form builder thích hợp.

Object navigator cung cấp cấu trúc hình cây để hiển thị các đối tượng trên form và được sử dụng để tạo, xóa hoặc sao chép và đổi tiên các thành phần của form. Sử dụng để lựa chọn các đối tượng đã tồn tại.

❖ **Property Palette**

Là một công cụ của form builder được sử dụng để định nghĩa các đặc tính đặc biệt của các thành phần form. Mỗi thành phần có một bộ thuộc tính khác nhau. Ví dụ các thuộc tính thành phần bao gồm sự giới hạn, chiều dài giá trị, kiểu dữ liệu hoặc giá trị thông báo. Các thuộc tính này về thực chất có thể được sử dụng để điều khiển các hành vi của form. Double click vào đối tượng trong object navigator thường mở ra một công cụ property palette.

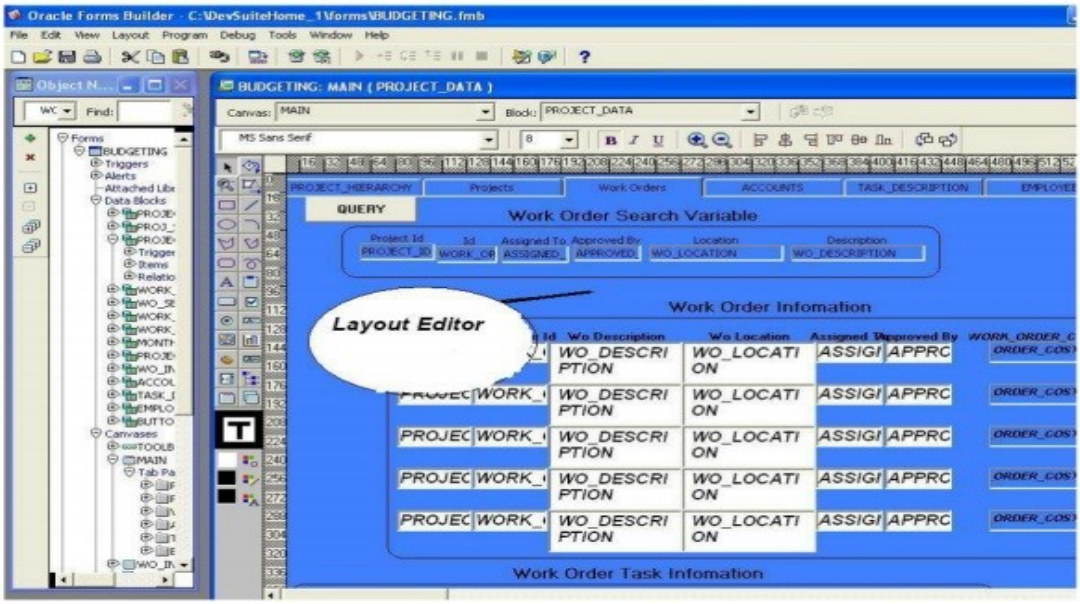


.

Hình 67.*Property Palette*

❖ Layout Editor

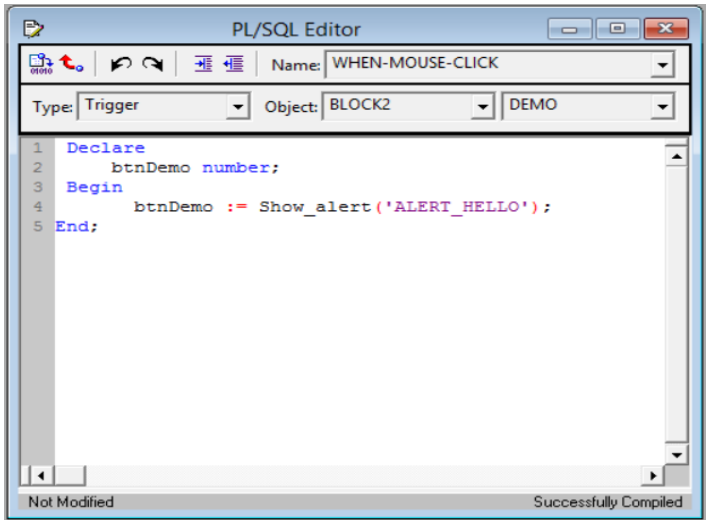
Là công cụ được dùng để thiết kế giao diện form. Công cụ này cho phép điều khiển các đối tượng hiển thị trên man hình, di chuyển và sắp xếp các đối tượng của form, thiết lập font chữ và tô màu, thêm các thành phần vào form và thêm một nhãn cho thành phần đó.



Hình 68.Layout Editor

❖ **PL/SQL Editor**

Công cụ này sử dụng để viết các kịch bản PL/SQL cần thiết cho một form hay là viết các đoạn mã lệnh được gọi là trigger để thêm mới các chức năng trên Form. Kịch bản PL/SQL này được đặt trong các trigger gắn với các đối tượng của form.



Hình 69.*Một đoạn code mẫu mô tả sử dụng trigger với ngôn ngữ PL/SQL trong form*

3.2.2. Cấu trúc Logic và hiển thị

3.2.2.1. Window

Khái niệm:

Window là một cửa sổ màn hình giống như một khung bức tranh rỗng (chưa có nội dung). Window có các chức năng cho phép phóng to, thu nhỏ, cuốn lên-xuống, di chuyển vị trí.

Một form có thể có nhiều window. Tất cả các form khi tạo mới sẽ tự động tạo một window ngầm định với tên là WINDOW0. Có thể tạo các window bằng cách chèn thêm từ Object Navigator.

Mỗi một window được tạo hầu như đồng thời với việc tạo một canvas-view. Canvas-view sẽ là nền cho giao diện để đặt các đối tượng. Cũng có thể đặt tương ứng canvas-view với window bằng cách đặt thuộc tính trong canvas-view.

Tại thời điểm chạy ứng dụng, window sẽ được hiển thị khi có lời gọi từ chương trình hoặc khi có sự định hướng xuất hiện (Navigation) của một item trên một canvas-view mà được gán tới window. Oracle Forms hiển thị window với nền canvas-view tương ứng. Trong cửa sổ thuộc tính của window ta có thể đặt các thuộc tính của window.

Cách tạo và xóa một window:

Để tạo mới một window ta chuyển hộp chọn trên cửa sổ Object Navigator vào đối tượng windows sau đó nhấn vào biểu tượng *Create*. Ta có thể nháy đúp chuột vào

window để gọi cửa sổ thuộc tính để có thể thay đổi tên ngầm định của window hoặc các thuộc tính khác.

Muốn xóa một window đặt con chuột vào biểu tượng window cần xoá sau đó nhấn phím *del* hoặc nhấn vào biểu tượng *delete* trên Object Navigator

3.2.2.2.Canvas

Khái niệm:

Canvas-View là vùng sẽ được hiển thị lúc chạy ứng dụng. Quan hệ giữa canvas và view (khung nhìn) của nó là một khái niệm cơ bản trong Oracle Forms. View giống như một hình chữ nhật trên canvas mà những gì chứa trong view sẽ được hiển thị trên window khi chạy ứng dụng.

Có thể dùng view của canvas để thay đổi kích thước vùng hiển thị. Khi view cùng kích thước với canvas thì tất cả nội dung trên canvas sẽ được hiển thị. Khi view nhỏ hơn thì chỉ có một phần canvas trong view được hiển thị. Canvas-View luôn hiển thị trên window mà ta đã gán tới. Có 3 loại Canvas-View: là content, stacked, toolbar

Với content hoặc toolbar canvas-view: khung nhìn được xác định bởi window tại đó canvas hiển thị trong nó. Việc thay đổi kích thước window tại thời điểm chạy sẽ ảnh hưởng đến việc hiển thị nội dung trong canvas.

Với stacked canvas-view: kích thước của view có thể được chỉ định tại lúc thiết kế bằng cách giá trị cho các thuộc tính tương ứng. Có thể cho ẩn hay hiện các stacked canvas-view khi thiết kế chương trình. Có thể đặt thuộc tính để hiển thị các thanh cuốn (có chức năng hiển thị các vùng khác trên canvas chưa được hiển thị).

Tạo và xóa canvas:

Để tạo mới một canvas-view: vào cửa sổ Object Navigator, vào đối tượng canvas-views sau đó nhấn vào biểu tượng *Create*. Ta có thể gọi cửa sổ thuộc tính để thay đổi các thuộc tính của canvas-view.

Xoá một canvas-views: đặt con chuột vào tên canvas-views cần xoá sau đó nhấn phím del hoặc nhấn chuột vào biểu tượng *Delete* để xoá.

Các thuộc tính cơ bản của canvas-views:

General:

- Name: tên canvas
- Canvas type: Kiểu canvas

Functional:

- Raise on Entry: xác định thứ tự hiển thị các canvas trong window
- Popup menu: gán popup menu cho canvas

Viewport:

- Viewport x/yosition: vị trí hiển thị canvas

- Viewport width/height: độ rộng và độ cao của canvas

Physical:

- Visible: xem có hiển thị canvas này hay không?
- Window: xác định window đặt canvas
- Bevel: chọn khung hiển thị canvas
- Corner style: Chọn kiểu góc của canvas
- Width style: Xác định kiểu độ rộng canvas(fixed, variable)

Visual Attributes:

- Visual attribute group: gắn 1 nhóm các thuộc tính trực quan đã được xác định trước vào canvas

Color:

- Foregroup/background color: Màu chữ và màu nền canvas

Font:

- Font name/size/weight/style: chọn font chữ, cỡ chữ độ rộng và kiểu chữ

3.2.2.3.Block Data

Khái niệm:

Block là một khối có chứa các Item. Tất cả các Item dù có quan hệ tới các bảng hoặc không đều phải nằm trong các Block.

Data blocks là một thành phần chứa dữ liệu của form có quan hệ tới các table (hoặc view) trên cơ sở dữ liệu. Data blocks có thể chứa item để nhận thông tin từ cơ sở dữ liệu, và cũng có thể chứa các item không nhận thôn tin từ cơ sở dữ liệu(control item)

Control block là khối mà không có quan hệ tới các bảng trên cơ sở dữ liệu và nó chỉ gồm các control item.

Tất cả Data blocks có thể là single-record blocks hoặc multi-record blocks. Một single-record block chỉ hiển thị một bản ghi tại một thời điểm. Một multi-record block hiển thị nhiều bản ghi tại một thời điểm.

Một base table block có thể là master block hoặc detail block. Master block hiển thị các master record. Detail block hiển thị các detail record.

Ta có thể tạo thêm một block bằng cách chèn thêm đối tượng từ Object Navigator và có thể thay đổi các thuộc tính của block bằng cách thay đổi các giá trị tương ứng trong cửa sổ thuộc tính của block.

Tạo một Control block

Để tạo một control block ta thực hiện như sau.

- Di chuyển chuột đến bảng Object Navigator.

- Bấm chuột chọn nhóm đối tượng Data block rồi di chuyển bấm vào nút Create để tạo block.

- Một Bảng xuất hiện với 2 nút radio button. Tích và chọn radio *Build a new data block manually*

- Đặt tên cho control block vừa tạo bằng cách kích 2 lần vào tên mạc định được tạo ra trong Object Navigator và viết vào tên mới. Hoặc là mở Property palette cho đối tượng block đó và thay đổi thuộc tính Name thành tên mới muốn đặt cho block

- Thêm các item mong muốn cho block vừa tạo

Tạo một Data block

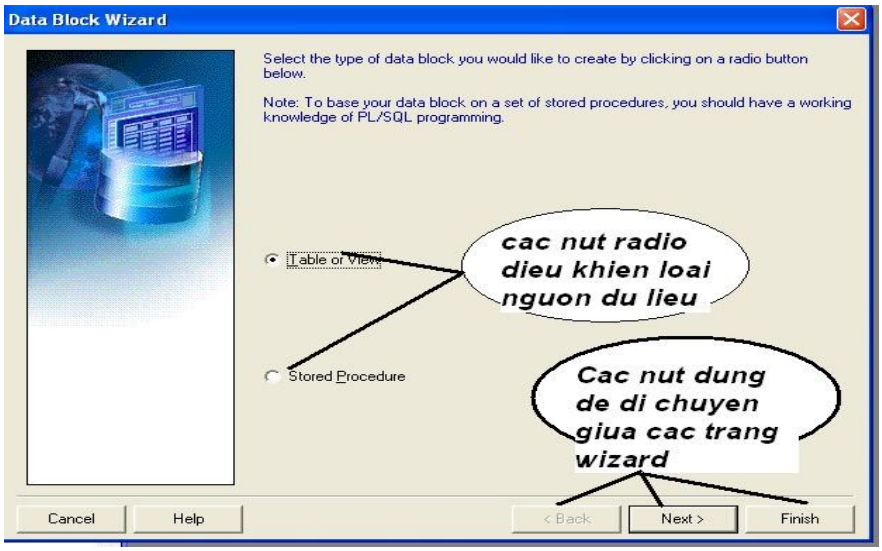
Để tạo một data block ta có hai cách:

Cách 1: Sử dụng trình Data Block Wizard:

Data block wizard là một trình tạo block đơn giản bằng giao diện của Oracle Form. Để tạo bằng Data block wizard ta có thể thực hiện như sau:

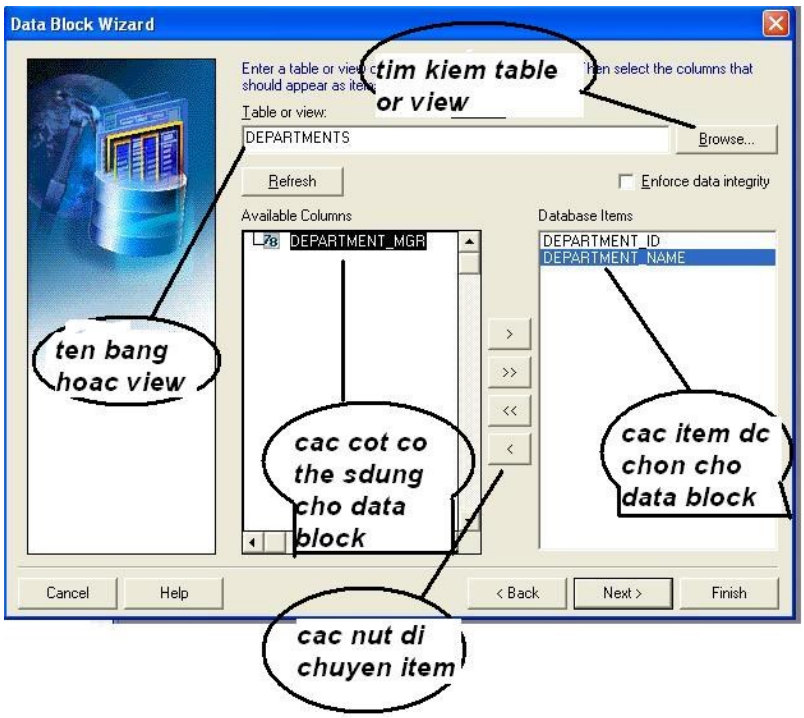
Khởi động*:* Bấm chuột phải vào vùng bất kỳ trong object navigator chọn Data block wizard hoặc chọn mục Data block trong object navigator và ấn vào nút create, một bảng tùy chọn xuất hiện, ta chọn tùy chọn *Use the Data block wizard* rồi chọn Nút **OK** để khởi động trình Data block wizard

Trang 1: trang chọn type. Nó được sử dụng để nhập vào loại dữ liệu nguồn cho block. Hộp thoại này có 2 nút là Table or view và Stored Procedure. Việc chọn Table or view thiết lập block dữ liệu dựa trên một bảng hoặc một view của Oracle. Một bảng là một đối tượng cơ sở dữ liệu để lưu trữ các record. View để lưu trữ các phát biểu select để tạo và trả về một tập các kết quả hoặc nhận được từ các bảng. Các store procedure được đặt tên là các kịch bản PL/SQL nằm trong cơ sở dữ liệu. chúng được sử dụng kết hợp với Ref Cursor và được sử dụng như một nguồn dữ liệu. Tùy chọn Table or view được thông dụng hơn nên trong phần này ta sẽ chọn Table or view để đến trang tiếp theo



Hình 70.*Trang chọn loại dữ liệu cho data block*

Trang 2:trang table. Trang này được sử dụng để thiết lập nguồn dữ liệu. trang này không xuất hiện nếu bạn chưa đăng nhập vào cơ sở dữ liệu nên là nếu trước đó bạn chưa đăng nhập vào cơ sở dữ liệu thì bạn sẽ được nhắc nhở đăng nhập.



Hình 71.*Trang 2 data block wizard: chọn nguồn data block*

- table or view: Mục này cung cấp tên nguồn dữ liệu là bảng hoặc view

- browse: nút này khởi động hộp thoại table được sử dụng để tìm kiếm xác định các bảng hoặc các view

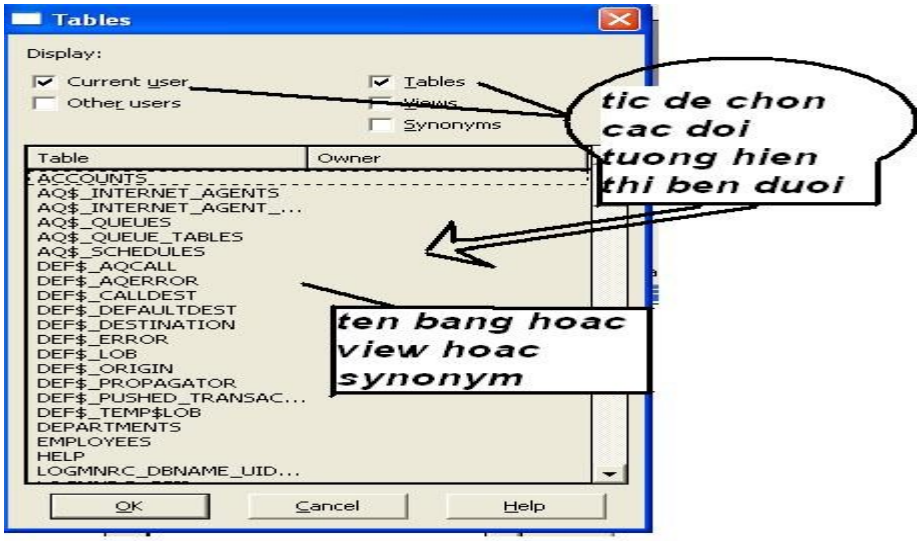
- refresh: nút này điền vào hộp danh sách Available column. Chỉ sử dụng nút này nếu khi chọn bảng hay là view rồi mà vẫn chưa thấy xuất hiện các cột trong hộp danh sách Available column

- Enforce data integrity: đánh dấu vào hộp kiểm này để thêm các rằng buộc cơ sở dữ liệu của bảng(view) được chọn vào thuộc tính của data block

- Available column: hộp liệt kê các cột trình có thể sử dụng cho data block

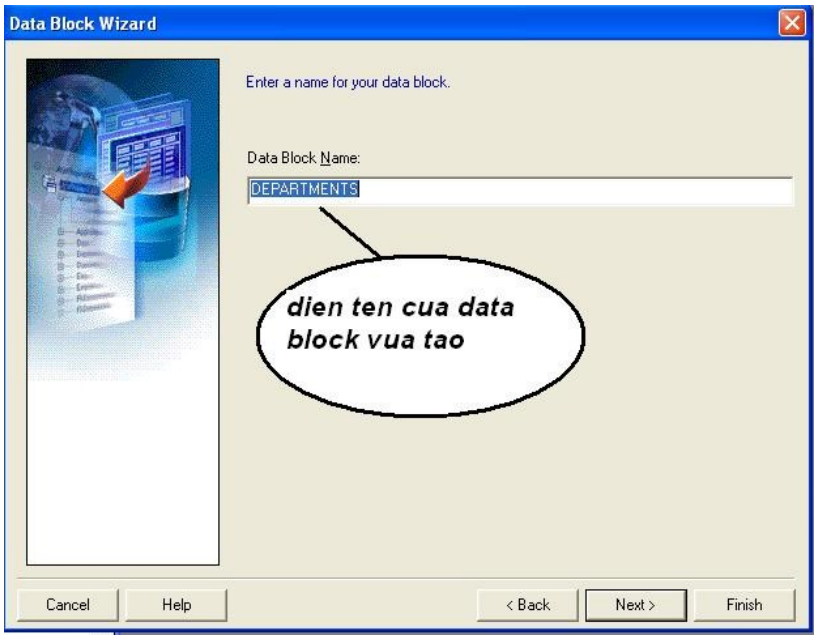
- Database items: hộp danh sách trình bày các cột có thể trình bày các cột sẽ xuất hiện trong data block

- Move buttons: bốn nút xuất hiện giữa 2 hộp danh sách. Chúng được sử dụng để thêm hoặc loại bỏ từ các cột danh sách



Hình 72.*Hộp chọn table hoặc view khi bấm vào nut browse của trang 2 trình data block wizard*

Trang 3: trang data block name. Đây là trang sử dụng để đặt tên cho block dữ liệu. Mạc định thì tên block dữ liệu sẽ là tên của table hoặc tên view đã chọn



Hình 73.*Trang data block name của trình dât block wizard*

Trang 4: trang chào mừng. Đây là trang chúc mừng bạn đã vừa tạo xong 1 data block. Trang này còn có 2 nút radio để chọn xem có gọi trình Layout Wizard không? Đây là trình cho phép ta đặt block dữ liệu lên canvas. Chúng ta sẽ xét đến trình này sau.

Hình 74.*trang chào mừng của trình data block wizard*

Cách 2: Tạo Data block bằng việc thay đổi các thuộc tính của Control block

Để tạo một data block bằng tay đầu tiên ta thực hiện tạo một control block như trên, rồi sau đó chuyển nó thành một data block (gắn nó với dữ liệu trong database) bằng cách thay đổi thuộc tính của block đó. Các thuộc tính và giá trị cần thay đổi là:

Database data block: Yes

Query data source name: có các tùy chọn là none, table, procedure, transactional trigger, from clause query. Tùy chọn thông thường là *table*. Nó sẽ lấy dữ liệu của một bảng nào đó. Tùy chọn *from clause query* sử dụng để ta viết vào một câu lệnh select truy vấn dữ liệu từ các bảng khác nhau

Query data source name: đây là nơi điền nguồn dữ liệu cho data block. Nếu chọn là *table* ở thuộc tính trên thì tại đây ta điền vào tên bảng mong muốn. Nếu chọn *from clause query* ở trên thì ta đánh vào câu lệnh select mong muốn.

Xóa một block (data và control block)

Để xóa một block ta thực hiện như sau:

Chọn data block muốn xóa trong object navigator. Rồi ấy Del hoặc chọn nút delete trong object navigator

Các thuộc tính cơ bản của block

General:

- Name: tên datablock
- Subclass information: điền vào một tập các giá trị thuộc tính đã xây dựng sẵn

Navigation:

- Navigation style: Kiểu di chuyển trong block
- Previous/next navigation data block: chỉ block trước và block tiếp theo block hiện thời

Record:

- Number of records displayed: số record hiển thị tại một thời điểm
- Record orientation: chỉ hướng của record là ngang hay học

Database:

- Database data block: xác định xem block là data hay control block
- Query/delete/insert/update allow: cho phép truy vấn, xóa, chèn hay update hay không?
- Query data source type: chọn kiểu nguồn dữ liệu

- Query data source name: điền nguồn dữ liệu cho block
- Where/ order by clause: tiêu chí hạn chế hặc xắp xếp dữ liệu trong block

Scrollbar:

- Show scroll bar: xác định xem có hiển thị scroll bar hay không?
- Scroll bar orientation: chọn hướng của scroll bar
- Scroll bar x/y position: tọa độ scroll bar
- Scroll bar width/length: chiều dài và rộng của scroll bar

Color:

- Fore/background color: chọn màu chữ, màu nền

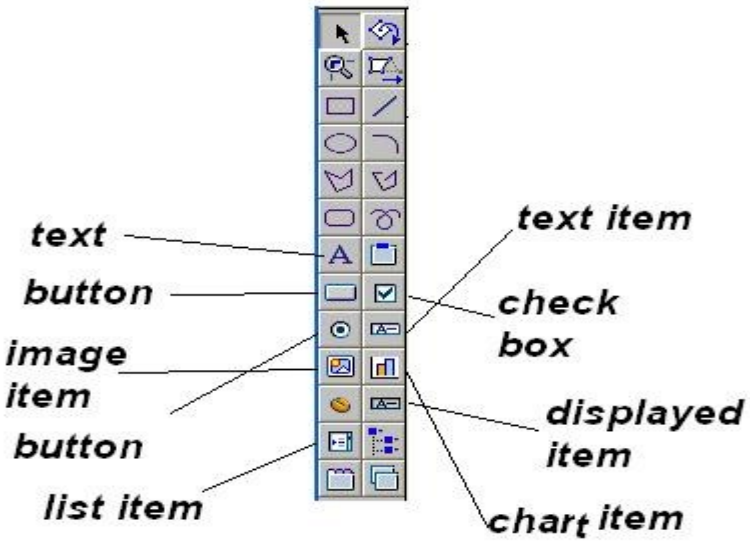
3.2.2.4.Item

Khái niệm

Items là các đối tượng giao diện mà hiển thị thông tin để thực hiện các thao tác và các tương tác với người sử dụng.

Oracle Forms cung cấp các kiểu giao diện item mà ta có thể sử dụng để xây dựng các giao diện ứng dụng. Gồm: button, chart item, check box, display item, image item, list item, radio group, text item, text. Mỗi một item trong form đều thuộc vào một block nào đó. Các item trong một block có thể nằm trên các canvas-views khác nhau và có thể hiển thị trên các window khác nhau.

Các item trên thanh toolbar của màn hình layout (Tools/Layout Editer)



Hình 75.*thanh tool bar của layout editor*

Cách tạo và xóa một item:

Trong màn hình layout để tạo một item, ta phải nhấn chuột vào biểu tượng của *item* tương ứng, sau đó kéo thả chuột vào màn hình layout. Thuộc tính *canvas* chỉ ra canvas nào item đang hiển thị và *block* chỉ ra item thuộc block nào. Có thể thay đổi các giá trị này.

Cũng có thể vào Object Nevigator chọn đối tượng blocks/[tên block]/items muốn đặt item, sau đó nhấn vào biểu tượng *Create*. Nháy đúp con trỏ chuột lên trên item vừa tạo sẽ hiện lên màn hình thuộc tính cho phép thay đổi các thuộc tính của item.

Muốn xoá một text item, đặt hộp chọn vào item cần xoá sau đó nhấn phím del hoặc biểu tượng *Delete* trên cửa sở Object Navigator để xoá.

Lấy giá trị item: <biến>:=:<tên block >.<tên item>

Gán giá trị cho item: :<tên block >.<tên item>:=<Giá trị>

3.2.2.5.Text Item:

Khái niệm: Text item Là đối tượng dạng text hiển thị các giá trị dạng string. Text item cho phép thực hiện các thao tác soạn thảo. Là kiểu ngầm định của oracle forms.

Cách tạo, xóa: giống với cách tạo xóa item chung

Các thuộc tính chung cơ bản của item:

General

- Name: tên của item
- Intem type: kiểu của item
- Subclass information: chèn một tập các thuộc tính với giá trị sẵn cho item

Function:

- Enabled: cho phép định hướng tới item bằng chuột
- Multi line: cho phép có nhiều dòng không?
- Case restriction: chọn chế độ hiển thị dữ liệu lên iem với người dùng là chữ hoa hay chữ thường hay là giữ đúng định dạng trong csdl hoặc đúng định dạng khi người sử dụng nhập vào giá trị
- Conceal data: giấu thông tin nhạy cảm bằng cách hiển thị các ký **. Thường được sử dụng để nhập giá trị password

Navigation:

- Keyboard navigable: có cho phép di chuyển đến item bằng phím tab của bàn phím hay không?

- Previous/next navigation item: item trước và item tiếp theo của item hiện thời

Data:

- Data type: chọn kiểu dữ liệu cho item
- Maximum length: độ dài tối đa khi hiển thị hay khi nhập vào item
- Required: xác định xem item này có nhất thiết phải nhập vào khi tìm kiếm không?
- Fomat mask: đặt định dạng cho item
- Copy value from item: giá trị của item sẽ được copy từ giá trị của 1 item khác nào đó
- Synchronize item: xác định item này được đồng bộ với item nào?

Calculation:

- Calculation mode: kiểu tính toán của item(none, formula- tính theo công thức,summary- tính theo các hàm sum, count…)
- Formula: là nơi viết vào công thức tính nếu ta chọn Calculation mode là formula
- Summary function: chọn hàm tính nếu Calculation mode được chọn là summary

Record:

- Current record visual attribute group: chèn một tập các thuộc tính trực quan được xây dựng sẵn cho bản ghi ở thời điểm hiện thời
- Distance between record: chọn khoảng cách giữa các record
- Number of item displayed: chọn số các item được hiển thị

Database:

- Database item: xác định xem item này là data item hay là control item
- Column name: tên 1 cột có trong data block mà item này chứa trong đó
- Primary key: xác định xem item này có phải là item chứa cột mạng khóa chính hay không?
- Query only: xác định xem item này có phải chỉ được sử dụng để query không?
- Query/insert/update/delete allowed: cho phép query, insert, update, delete hay không?

Physical:

- Canvas: xác định item nằm trên canvas nào
- Tab page: xác định item nằm trên tab nào nếu canvas thuộc kiểu tab can vas
- x/y position: tọa độ vị trí item
- width/height: độ rộng và độ cao của item
- show vertical scroll bar: xác định xem có hiển thị scroll bar cho item này hay không? Thuộc tính này chỉ chọn khi item có thuộc tính multi line là yes

Visual attributes:

- Visual attribute group: chọn nhóm các thuộc tính trực quan đã được xây dựng sẵn để gán cho item
- Prompt visual attribute group: chọn nhóm các thuộc tính đã xây dựng trước dành cho prompt của item này

Color:

- Background/foreground color: chọn màu nền và màu chữ cho item

Font:

- Font name/size/weight/style: chọn font chữ, cỡ chữ, độ rộng của form và kiểu của font

Prompt:

- Prompt: viết vào prompt cho item này
- Prompt display style: kiểu hiển thị prompt(1 record, tất cả các record, hay là hidden)
- Prompt alignment: chọn căn prompt là ở trên, dưới hay ở giữa item
- Prompt attachment edge: chọn chiều hiển thị prompt(start- chiều ngang với item, top - chiều dọc,…)

Prompt color:

- Prompt foreground color: chọn màu chữ cho prompt

Prompt font: chọn font cho prompt

Help:

- Tooltip: viết câu trợ giúp chức năng của item này khi ngươfi sử dụng dê chuột vào

- Tooltip visual attribute group: chọn mọt nhóm các thuộc tính đã xây dựng sẵn cho tooltip

3.2.2.6.Display item.

Khái niệm: Display item là đối tượng dùng để hiển thị các giá trị dưới dạng string. Display item không cho phép định hướng con trỏ tới và soạn thảo nội dung trong nó.

Cách tạo và xóa display item: tương tự như tạo và xóa một item

Các thuộc tính cơ bản: item type: *display item*

II.4.4.4.3. list item

Khái niệm: List item là một đối tượng dùng để hiển thị một danh sách các giá trị dưới dạng string cho phép chọn một giá trị.

Cách tạo và xóa list item: tương tự như tạo và xóa một item

Các thuộc tính cơ bản:

- Item type: list item

- List stype: chọn kiểu danh sách. Có các kiểu sau: poplist, Tlist, Combo box

- Element in list: xoạn thảo các thành phần của list item

- Mapping of other value: giá trị được gán ứng với các giá trị không có trong danh sách

II.4.4.4.4. Button

Khái niệm: Là một cái hộp có nhãn bên trong hoặc là một biểu tượng mà có thể nhấn vào đó để thực hiện các hành động

Cách tạo và xóa list item: tương tự như tạo và xóa một item

Thuộc tính cơ bản:

- item type: push button
- lable: nhãn hiển thị của button
- iconic: xác định xem có dùng một icon khác icon mạc định không?
- icon file name: tên file icon cho button

3.2.2.7.Check box

Khái niệm: Là một nhãn hiển thị với một hộp đánh dấu trạng thái.

Cách tạo và xóa check box: tương tự như các tạo và xoa item

Các thuộc tính cơ bản:

- Item type: check box

- Lable: nhãn của check box

- Value when checked: giá trị khi mà check box được tích

- Value when unchecked: giá trị khi check box không được tích

3.2.2.8.Image

Khái niệm: Là một hình hộp hình chữ nhật với một kích thước nào đó mà có thể hiển thị các ảnh được lưu trong cơ sở dữ liệu hoặc trên file. Các ảnh này chỉ xuất hiện trên màn hình trong lúc chạy chương trình và nhờ các lệnh

Cách tạo và xóa image: tương tự như tạo và xóa item

Các thuộc tính cơ bản:

Item type: image

Image format: định dạng của ảnh được sử dụng

Display qualtity: chất lượng hiển thị của ảnh

Sizing stype: chọn kiểu hiển thị của bức ảnh khi mà kích thước của nó không phù hợp với kích thước ảnh thật

3.2.2.9.Triggers

Khái niệm

Triggers là một khối chứa các mã lệnh nhằm thực hiện một chức năng nào đó trong chương trình ứng dụng. Tất các các trigger đều có tên và chứa một hoặc nhiều dòng lệnh PL/SQL tương ứng với sự kiện mà nó phải xử lý.

Tên Trigger thường tương ứng với sự kiện (ví dụ When-Button-Pressed trigger sẽ tương ứng với sự kiện nhấn vào Button.

Một trigger phải được gắn với một object xác định trên form: item, block hoặc chính trên form. Trigger gồm các mức (level) tương ứng:

Việc xác định mức đặt các trigger là rất quan trọng bởi vì nó xác định phạm vi hoạt động của trigger ứng với các sự kiện (events). Ví dụ Item-level trigger chỉ bật lên (fire) ứng với các sự kiện trên item. Nó không được bật lên ứng với sự kiện như vậy trên item khác cùng hay không cùng thuộc một block. Block-level trigger chỉ bật lên nếu có sự kiện xuất hiện trong block nhưng nó không bật lên khi xuất hiện các sự kiện giống như vậy thuộc block khác.

Mặt khác nếu định nghĩa các trigger ở các mức khác nhau cùng xử lý một sự kiện, mà xự kiện này đều nằm trong phạm vi của các trigger thì trigger ở mức thấp nhất sẽ được thực hiện. Ví dụ ta đặt trigger When-New-Item-Instance tới một block và một trigger When-New-Item-Instance khác tới text item trong block này, khi sự kiện xuất hiện trên text item thì trigger mức item sẽ được bật lên còn trigger mức block bị bỏ qua.

ở cùng một mức các trigger cũng xuất hiện và xử lý với các cách khác nhau. Có các loại trigger sau: Pre-, Post-,When-, On-, Key-.

Tạo và xóa một trigger

Trên cửa sổ Object Navigator ứng với mỗi object (item, block, form) sẽ có mục Triggers. Ta chuyển hộp chọn vào Triggers sau đó nhấn vào biểu tượng Create để tạo mới và chọn sự kiện tương ứng (ví dụ when-button-pressed) sau đó hiện lên cửa sổ viết mã.

Pre –triggers:

Các loại trigger này được kích hoạt trước các giao tác của cơ sở dữ liệu, hoặc các sự kiện của form. Chúng chứa đựng các phát biểu bổ xung, các phát biểu này được thực thi trước khi hoạt động xảy ra. Không giống như trigger on chúng không thay thế các hoạt động bình thường của form. Vd như trigger pre-insert được kích hoạt khi phát biểu insert được thực hiện

Một số trigger loại này hay dùng như:

pre-insert: được thực hiện trước khi quá trình insert thực hiện

pre-delete:được thực hiện trước khi quá trình delete thực hiện

pre-update: : được thực hiện trước khi quá trình update thực hiện

pre-form: trước khi form được khởi động

When – triggers:

Các loại trigger này được kích hoạt khi một hoạt động của form xảy ra. Ví dụ như trigger when-new-form-instance được kích hoạt khi form được khởi động đầu tiên

Một số trigger loại này hay được sử dụng:

- When-button-pressed: Được thực hiện khi có thao tác nhấn button bằng phím hoặc chuột

- When-clear-block:Thực hiện khi xoá dữ liệu từ block hiện thời

- When-create-record: Thực hiện khi tạo một bản ghi mới

- When-checkbox-change: Thực hiện khi check box thay đổi trạng thái

- When-radio-change: Thực hiện khi có thay đổi chọn các radio button

- When-list-change: Thực hiện khi thay đổi chọn giá trị trong danh sách.

- When-new-block-instance: Được thực hiện khi di chuyển input focus từ một block từ block khác

- When-new-item-instance: Được thực hiện khi con trỏ chuyển tới Item

- When-validate-item: Xuất hiện khi có những thay đổi giá trị của item từ người sử dụng hoặc từ các trigger và khi con trỏ di chuyển ra khỏi item.

Post – Triggers:

Các loại trigger này kích hoạt sau các giao tác của cơ sở dữ liệu hoặc các sự kiện của form. Chúng chứa đựng các phát biểu bổ xung, các phát biểu này được thực thi sau khi hoạt động xảy ra. Không giống như trigger on, chúng không thay thế các hoạt

động bình thường của form. Ví dụ trigger post-query được kích hoạt sau khi một record được đem về block dữ liệu

Trigger post-query thường được sử dụng để đưa thông tin diễn giải vào trong một block dữ liệu. ví dụ làm rõ tên phòng ban trong bảng nhân viên vì trong bảng nhân viên chỉ có mã phòng ban. Trigger này có thể làm trả về tên của phòng ban tương ứng với mã phòng ban đó

Một số Post-Trigger hay dùng

- Post-Change: Thực hiện khi dữ liệu của item được thay đổi với giá trị chấp nhận không phải là giá trị null.

- Post-form: Xuất hiện trong khi thoát khỏi form

- Post-query: Thực hiện sau khi dữ liệu được điền vào các record

- Post-record: Xuất hiện khi rời input focus từ bản ghi này tới bản ghi khác.

Key – Trigger:

Các loại trigger này dùng để thay thế các tính năng của phím chức nằng thông thường, chẳng hạn như trigger Key-Entqry được kích hoạt khi các phím chức năng dùng để chuyển một block dữ liệu vào trạng thái Enter query được ấn. chúng được dùng để thay thế hoặc là phát triển thêm các tính năng thông thường của phím đó

.Một số trigger hay dùng:

- Key-up: ứng với việc nhấn phím Page-up

- Key-down: ứng với việc nhấn phím Page-down

- Key-enter: ứng với nhấn phím enter

- Key-next-item: ứng với việc nhấn phím tab

On – Trigger:

Các loại trigger này được dùng để khởi động một giao tác, ví dụ như là các phát biểu trong trigger on-insert thay thế phát biểu insert được sử dụng bởi block dữ liệu. Các thao tác đặc biệt trong trigger on là các thao tác quản lý dữ liệu chẳng hạn như là xóa dữ liệu và điền dữ liệu lại vào một block dữ liệu chi tiết sau khi thay đổi các record trong block dữ liệu cha.

Các On-Triggers hay dùng

On-error: Thực hiện khi có một lỗi nào đó xuất hiện

On-mesage: Thực hiện khi hiện các mesage

3.2.2.10. LOVs

Khái niệm

LOV là viết tắt của List of value. Nó là một đối tượng đặc biệt của form được sử dụng để điền dữ liệu và kiểm tra tính hợp lệ của dữ liệu trong item văn bản.

Một LOV là một hộp thoại được gắn với các item rõ rang về form. Người dùng có thể mởi LOV và chọn một giá trị từ danh sách.

Một LOV có thể được sử dụng trong chế độ enter query để đồng nhất chế độ chọn lựa hoặc được sử dụng trong chế độ normal để điền và kiểm tra một giá trị. LOV sử dụng một record group là nguồn cung cấp giá tr trị cho nó. Record group thường sử dụng phát biểu select là nguồn dữ liệu vì thế LOV là một tập các giá trị động. giống như list item LOV có thể được dụng để nhận diện các giá trị, tuy nhiên LOV có khả năng thể hiện hàng trăm hàng ngàn giá trị. Nó cho phép người dùng cuộn qua các giá trị cũng như giới hạn việc hiển thị dựa trên các giá trị nhập vào

Cách Tạo LOV

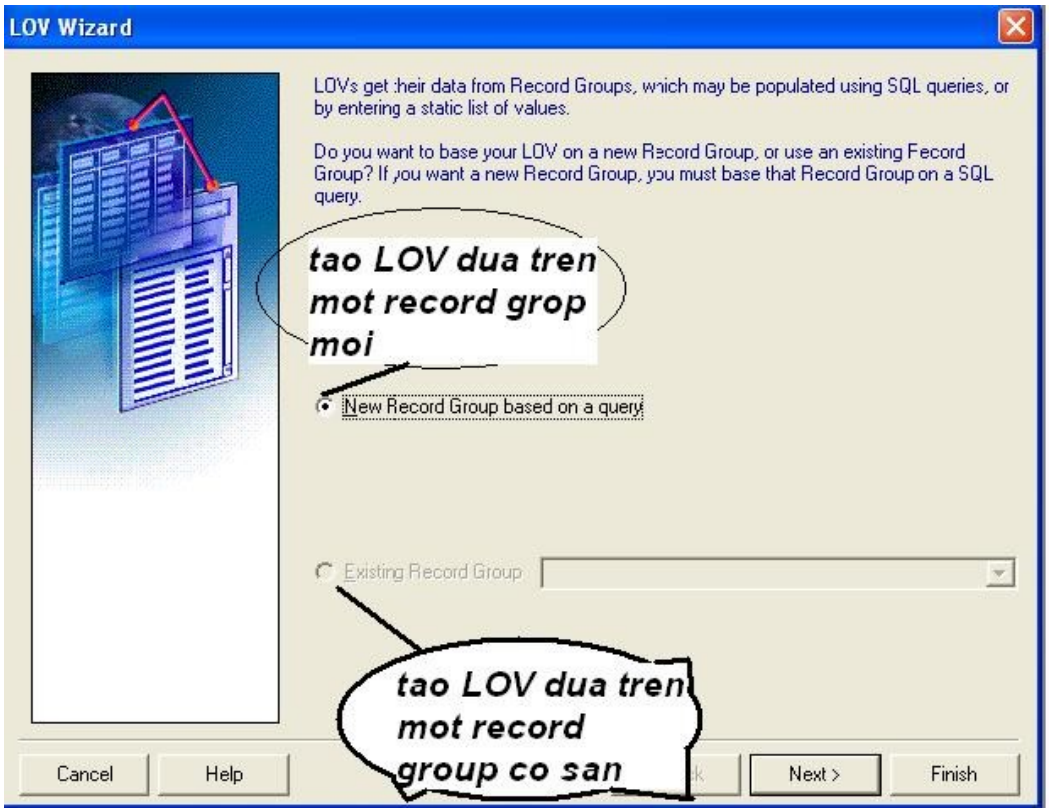
Có một cách vô cùng đơn giản để tạo các LOV là sử dụng trình LOV Wizard

Mở Object Navigator và chọn đối tượng LOV

Kích vào nút create. Thao tac này mởi một hộp thoại có 2 nút radio cho phép bạn chọn có sử dụng trình LOV Wizard hay không? Chọn nút radio báo ràng bạn muốn sử dụng trình wizard

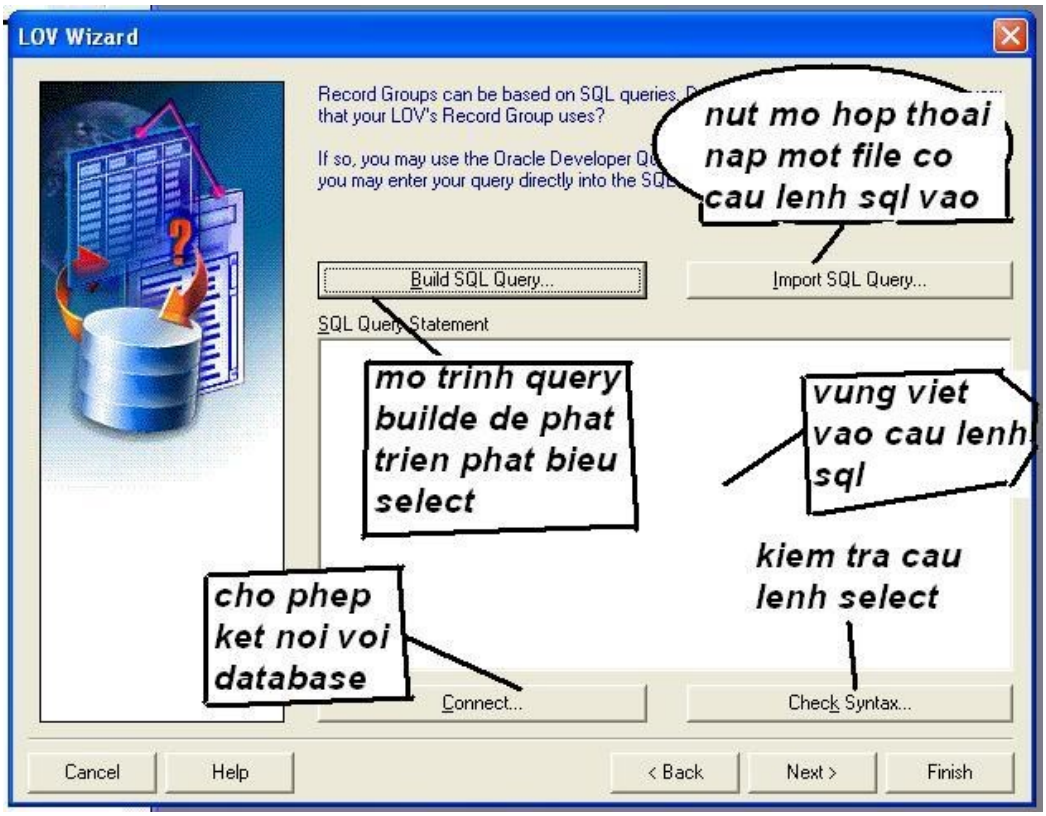
Hình 76.*LOV*

Trang đầu tiên của trình wizard này là trang Source. Trang này được sử dụng để xác định xem LOV sử dụng một nguồn dữ liệu là một record group mới hay là record group đã tồn tại



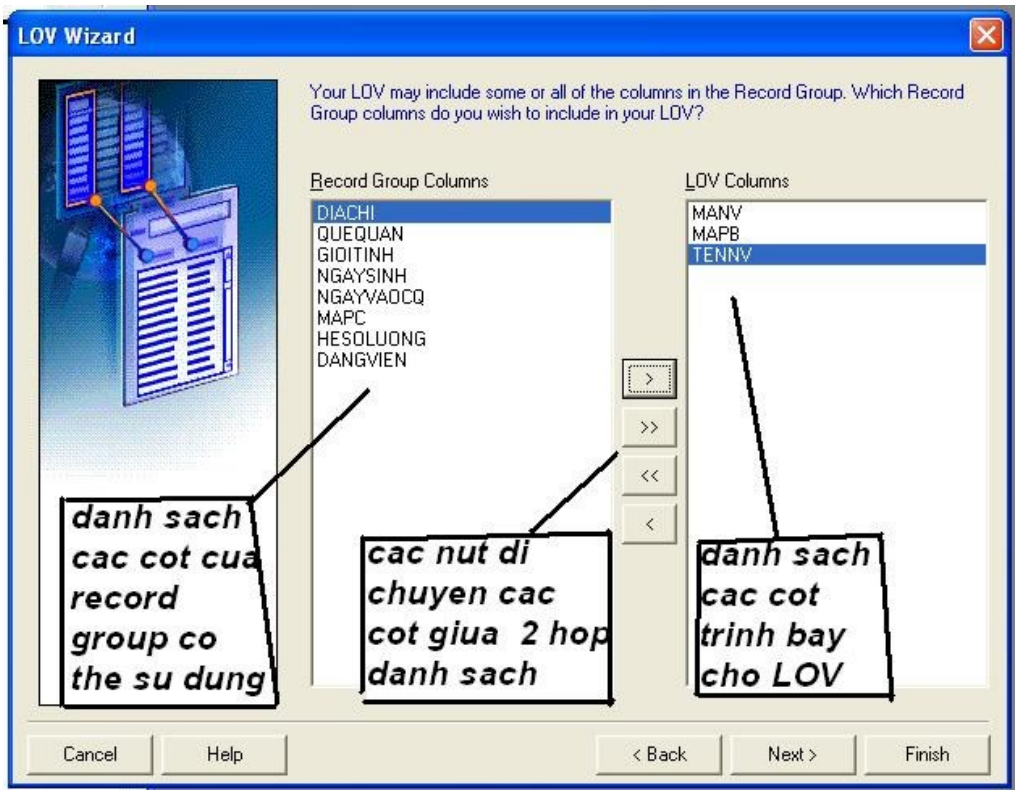
***Hình** 77.Trang Source của trình LOV Wizard*

Nếu chọn tùy chọn new record group trên trang source thì trang tiếp theo là trang sql query. Trang này được sử dụng để thêm phát biết select sẽ được sử dụng trong record group. Trong trang này có tùy chọn kết nối với database, kiểm tra cú pháp của phát biểu select. Form buider không cho phép bạn ra khỏi trang này trừ khi câu select là hợp lệ



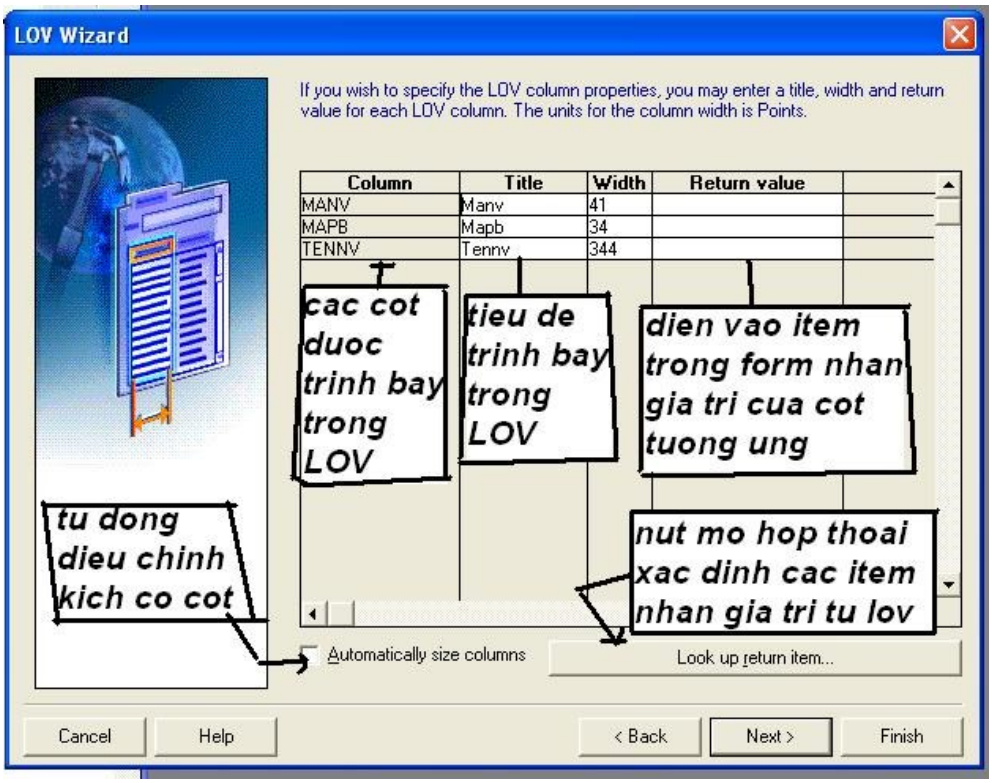
Hình 78.*Trang sql query của trình LOV Wizard*

Trang tiếp theo là trang column selection. Nó được sử dụng để xác định các cột nào xuất hiện trong LOV. Các cột được trình bày trong cột bên phải sẽ là các cột được trình bày trong LOV



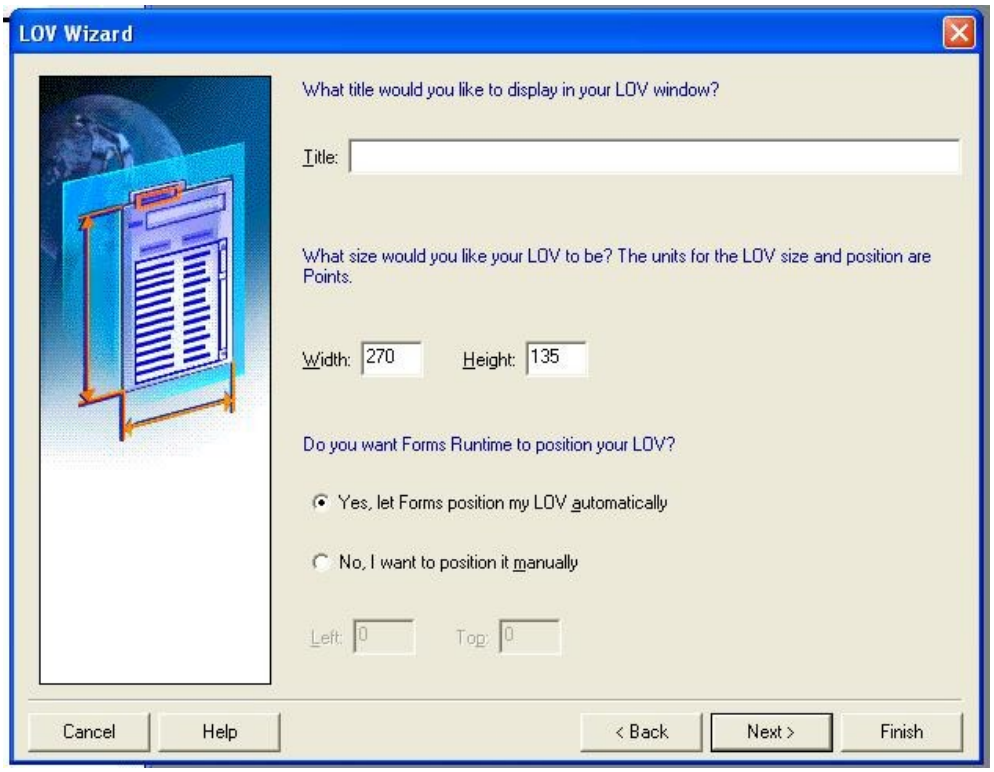
Hình 79.*Trang column selection cua trinh LOV Wizard*

Trang kế tiếp là trang Column display. Được sử dụng để người dùng nhập tiêu đề cột. ở đây có bốn cột là column, Title (tiêu đề), width(chiều rộng), return value(cột nhận giá trị trả về). Bạn có thể hiệu chỉnh bất kỳ cột nào trừ cột column. Return value được sử dụng để chọn item nào của form sẽ nhận giá trị trả về từ LOV



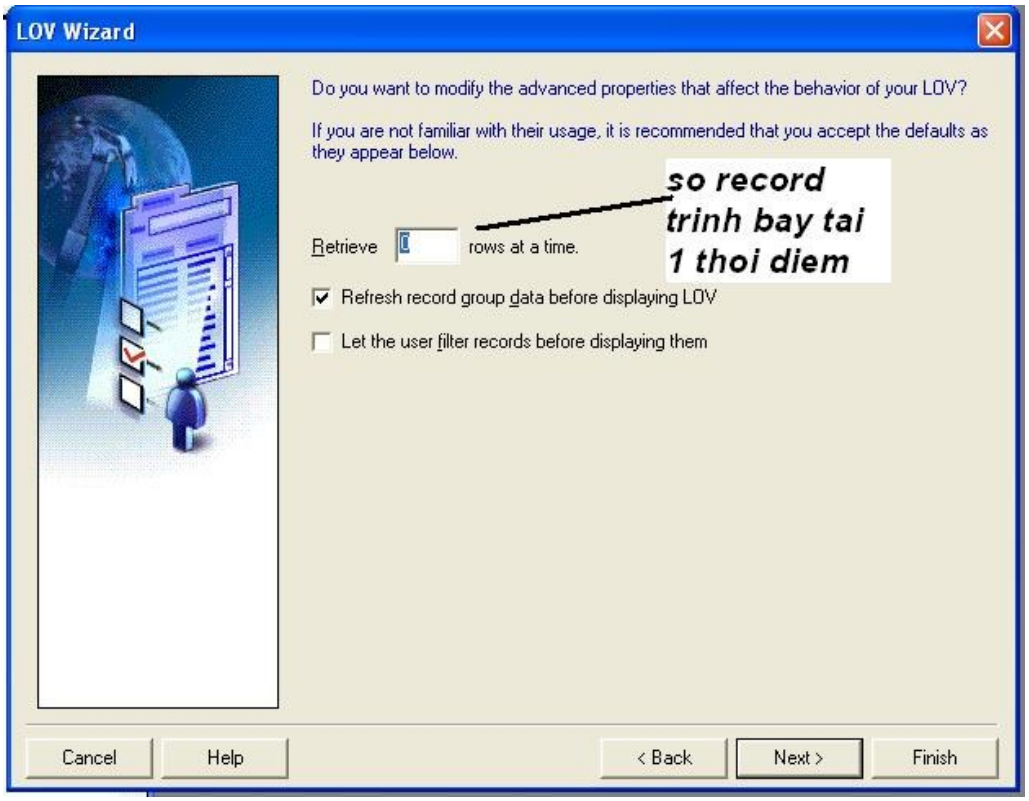
Hình 80.*Trang Column display của trinh LOV wizard*

Trang tiếp theo là trang display. Trang này xác định vị trí trình bày của LOV cũng như toán bộ kích thước và tiêu đề



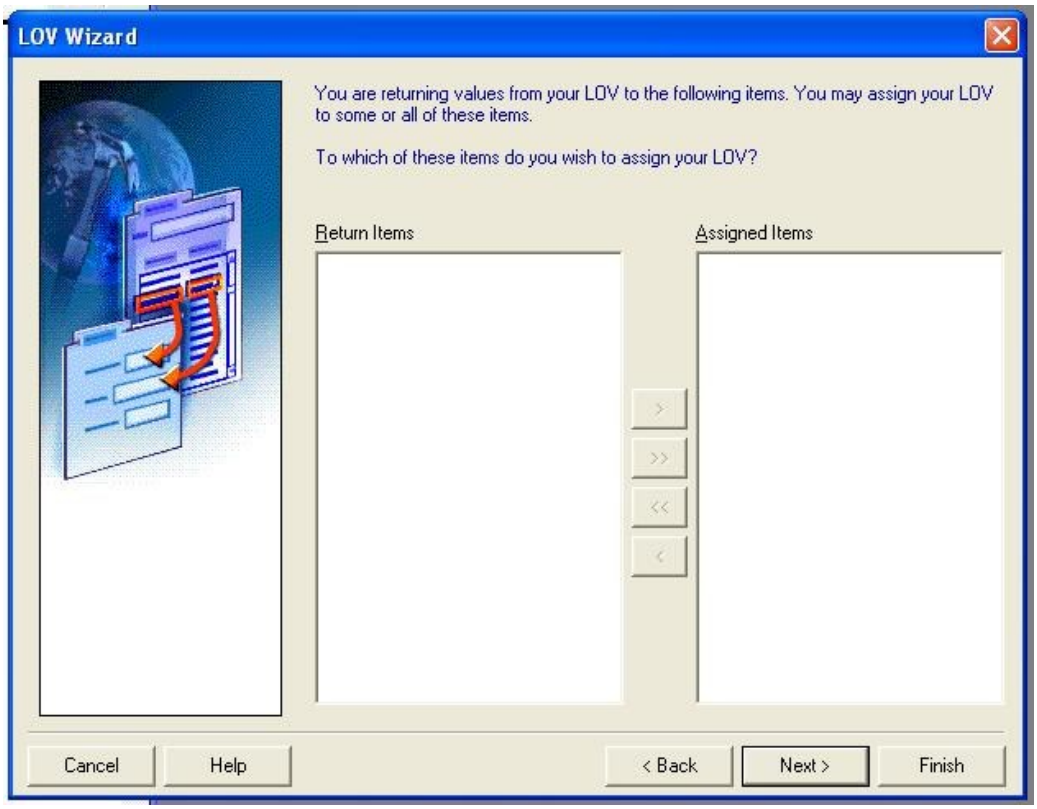
Hình 81.*Trang display của trình LOV wizard*

Trang tiếp theo là trang advanced option. Đây là trang thiết lập số lượng record xuất hiện trong 1 thời điểm.



Hình 82.*Trang advanced option của trinh LOV Wizard*

Trang cuối cùng là trang xác định item tren form được dùng để gắn LOV này với nó. Mạc định thì LOV sẽ được mở khi người dùng di chuyển vào item này



Hình 83.*Trang xác định item gắn với LOV*

3.2.2.11. Messages và Alerts

Messages: Là dòng thông báo xuất hiện tại dòng trạng thái của Window chỉ ra trạng thái của quá trình xử lý nào đó hoặc khi có lỗi.

Để hiển thị message ta dùng lệnh MESSAGE(message_string, user_response);

message_string: Chuỗi hiển thị cần thông báo

user_response: Chỉ hình thức hiển thị của message.

Thường hai trigger là: on-error và on-message hay liên quan đến quá trình hiển thị và xử lý các message.

Alert

Hiển thị dưới dạng một modal window chứa các thông tin cần thông báo và đợi trả lời từ phía người sử dụng. Tuỳ theo trả lời mà có thể thực hiện các xử lý tiếp theo.

Tạo một Alerts

Trong cửa sổ Object Navigator chọn mục Alerts sau đó nhấn biểu tượng *Create,* sau đó đặt thuộc tính theo yêu cầu.

Các thuộc tính cơ bản của Alert

Functional properties

Alert style: Chỉ kiểu hiển thị của alert (stop, caution, note)

Button 1, Button 2, Button 3: Nhãn cho các button. Phải có ít nhất một button có giá trị

Default alert button: Button ngầm định

Messeage: Nội dung thông báo cần hiển thị

Hiển thị Alert

Để hiển thị alert dùng lệnh Show_alert(‘ <Tên alert>’)

Để thay đổi nội dung thông báo của Alert dùng lệnh:

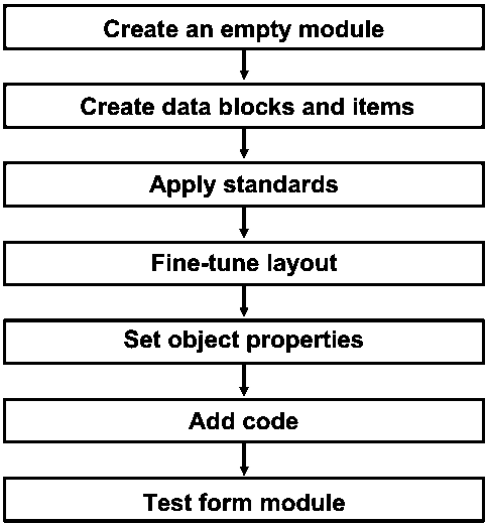
set_alert_properties(‘<tên alert>’, properties, mesage)

Ví dụ: Set_Alert_Property('My_Error_Alert', alert_message_text, ‘ Có lỗi xuất hiện khi ghi’);

Bt:=Show_Alert(al_id);

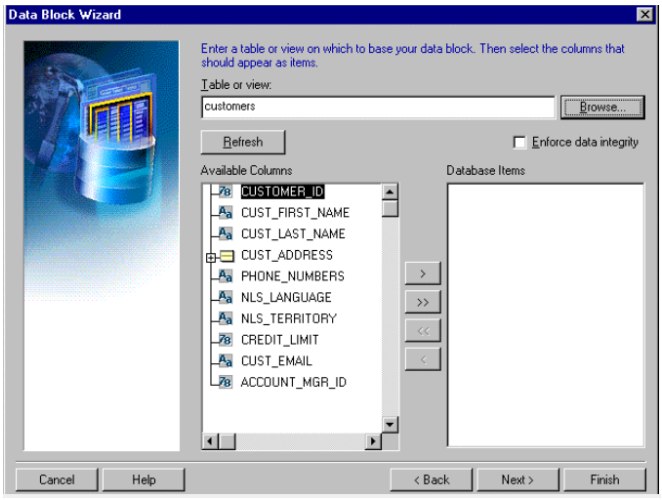
3.2.3. Lập trình Form

Quy trình tạo một module form:

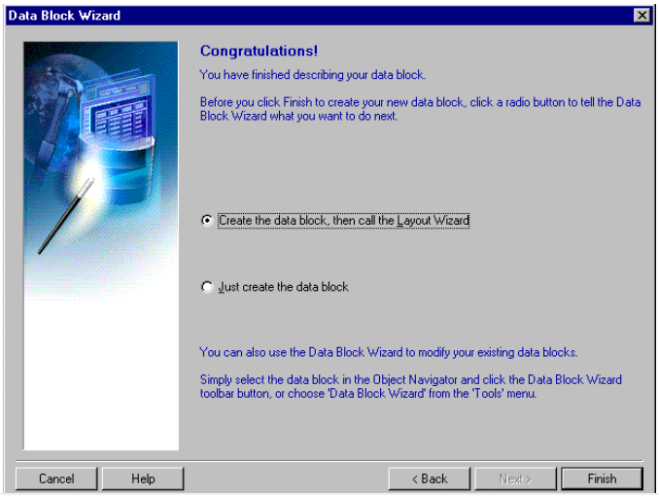


3.2.3.1.Tạo 1 form đơn giản từ 1 bảng

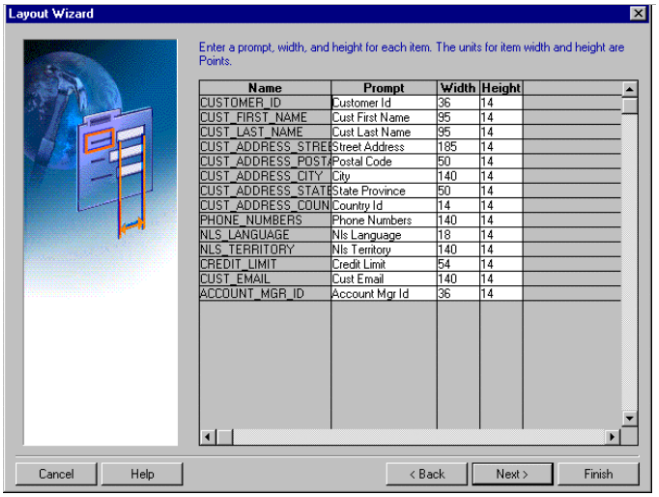
- **Tạo thủ công bằng tay**
- **Sử dụng Forms Builder Wizards:**
  - Data Block Wizard: tạo một datablock liên quan đến data source nhanh chóng và dễ dàng.
  - Layout Wizard: tạo ra một giao diện trực quan có các item từ datablock vừa tạo ra.



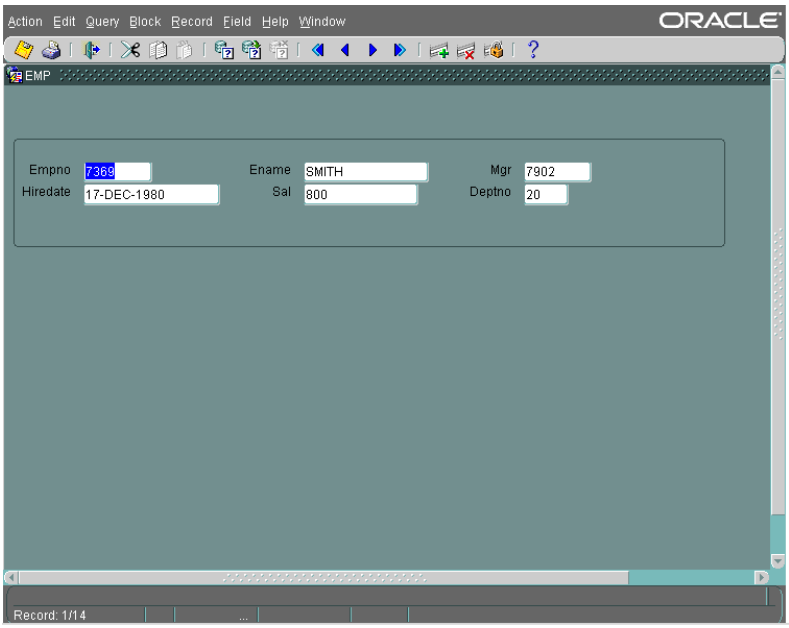
Hình 84.*Chọn một bảng hoặc view trong CSDL, bấm Next>*



Hình 85.*Thiết lập về data đã xong, chuyển sang thiết lập layout. Bấm Finish*



Hình 86.*Thiết lập nhãn, kích thước các item*



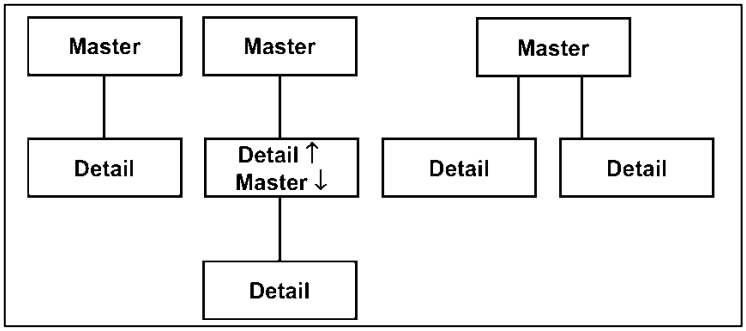
Hình 87.*Giao diễn form đơn giản từ 1 bảng demo*

Bài tập thực hành:

- Tạo một module mới tên là CUSTOMERS. Tạo một datablock mới bằng cách sử dụng wizard, truy xuất đến bảng customers trong schema summit.
- Sử dụng Layout Editor, đặt lại vị trí các item sao cho phù hợp.
- Lưu và chạy form module CUSTOMERS trên Web.
- Thực hành insert, update, delete dữ liệu trên form.

3.2.3.2. Tạo master-detail form

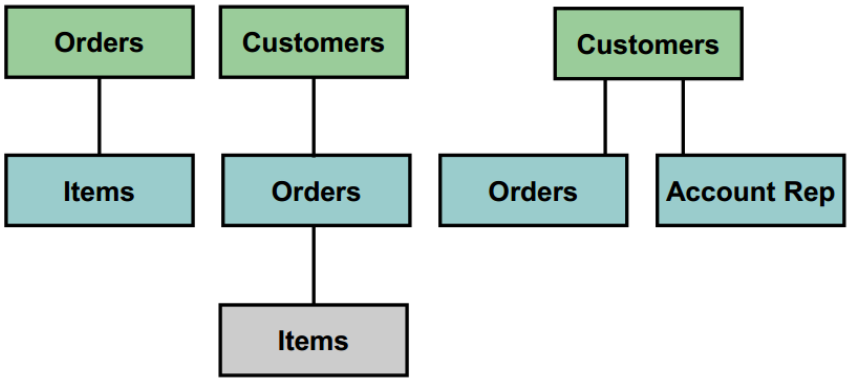
Quan hệ giữa các data block:



Hình 88.*Quan hệ giữa các data block*

- Mỗi module có thể chứa nhiều hơn 1 data block.
- Mỗi datablock có thể có quan hệ với datablock khác.
- **Master-Detail Relationship** là mối quan hệ giữa 2 datablock phản ánh mối quan hệ chính – ngoại giữa 2 bảng dữ liệu.

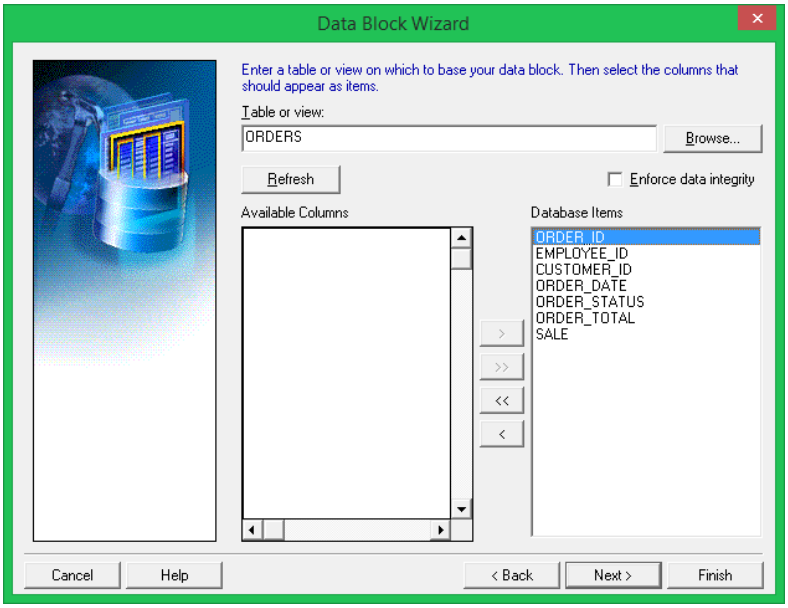
- **Master datablock** là bảng chứa khóa chính.
- **detail datablock** là bảng chứa khóa ngoại tham chiếu đến khóa chính của bảng ở master datablock.



Hình 89.*Ví dụ về một mô hình master detail form*

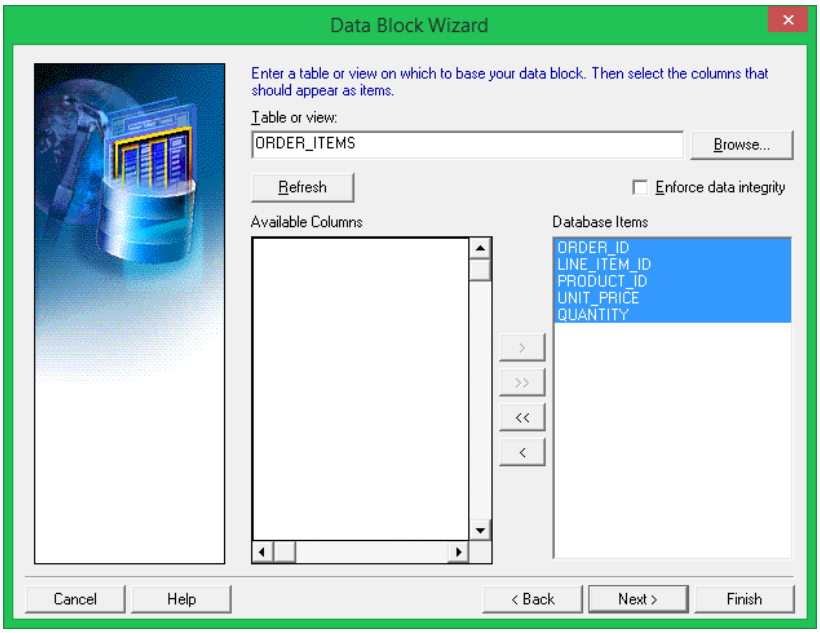
- **Quy trình tạo master-detail form**

Bước 1: Tạo master block (sử dụng bảng ORDERS)



Hình 90.*Tạo master block*

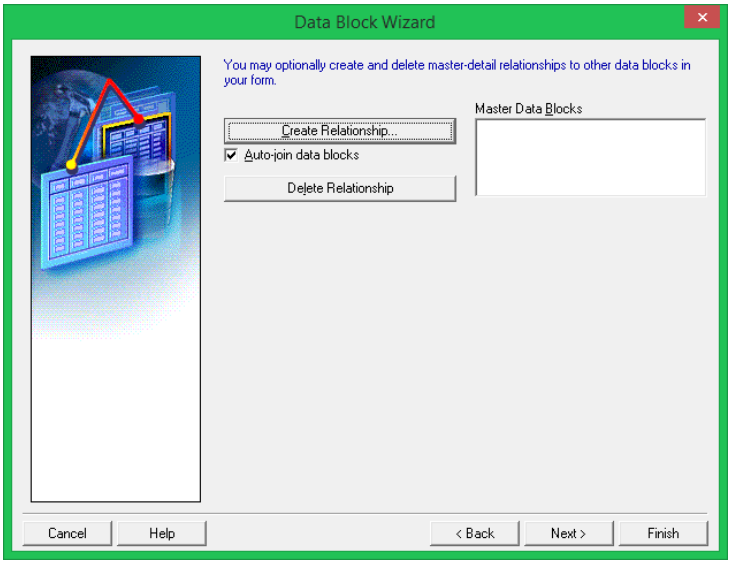
Bước 2: Tạo detail data block (sử dụng bảng ORDER_ITEMS)



Hình 91.*Tạo detail block sử dụng bảng order_item*

Bước 3: Tạo quan hệ với master data block

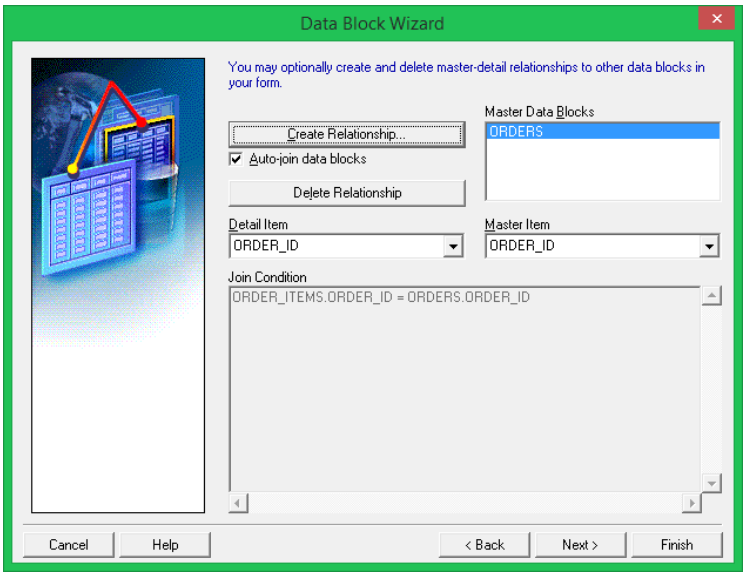
- Click chọn Create Relationship
- Chọn master data block cần tạo quan hệ với detail block đang tạo, bấm OK.



Hình 92.Tạo quan hệ với master block

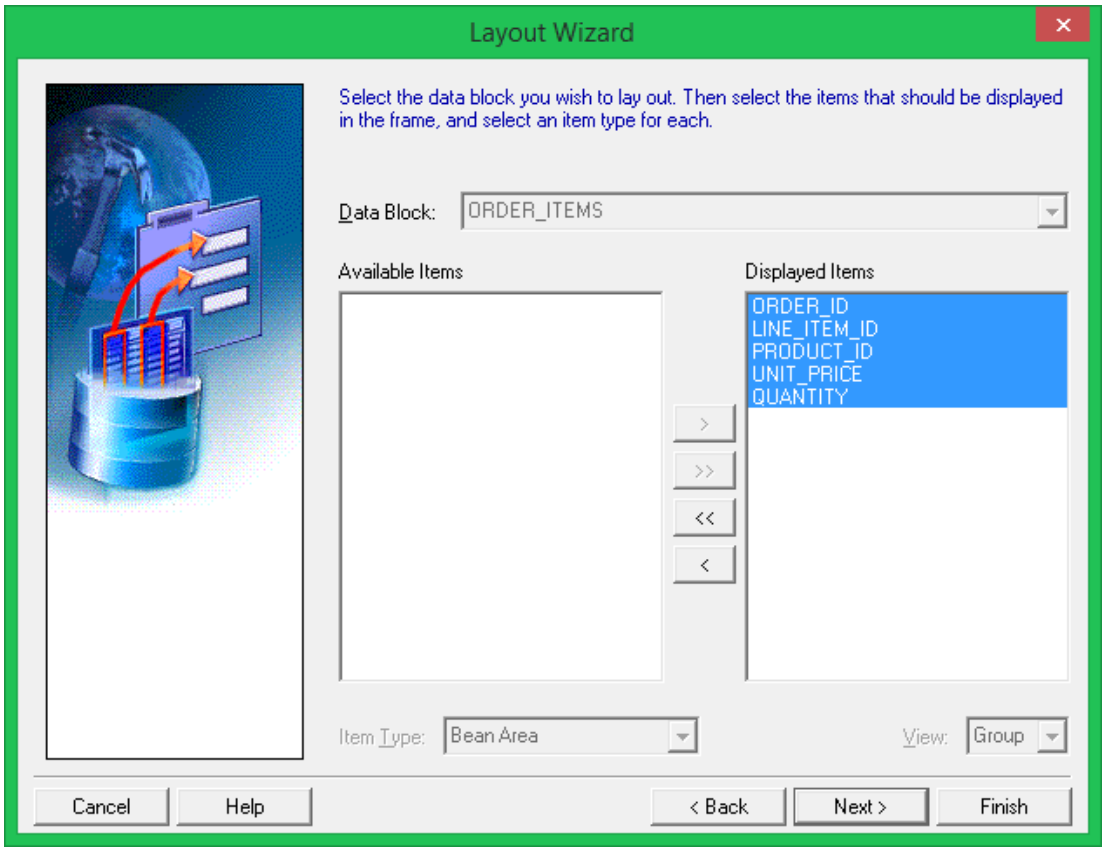
- Chương trình sẽ tự động tạo ra điều kiện kết nối giữa master data block và detail data block.

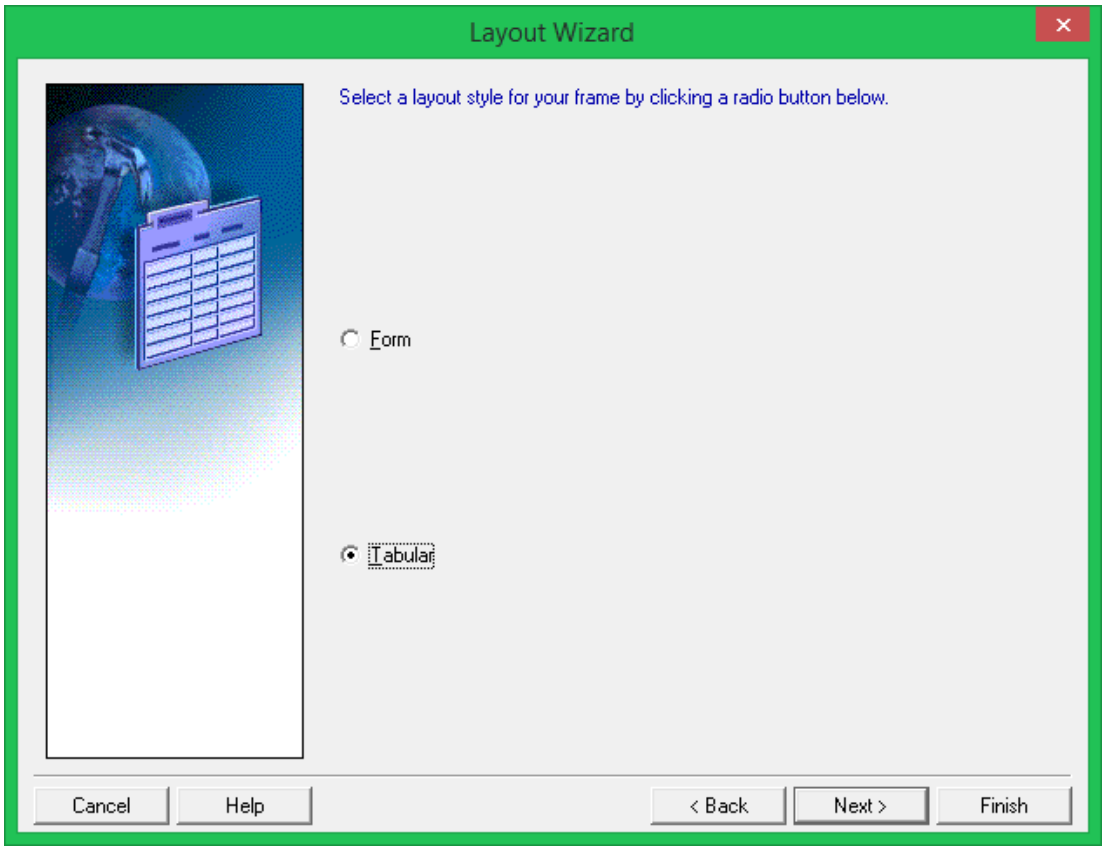
- Nếu Checkbox "Auto-join data blocks" không được đánh dấu, thì chương trình sẽ không tự động tạo điều kiện kết nối, khi đó chúng ta phải tạo điều kiện kết nối 2 data block bằng tay.

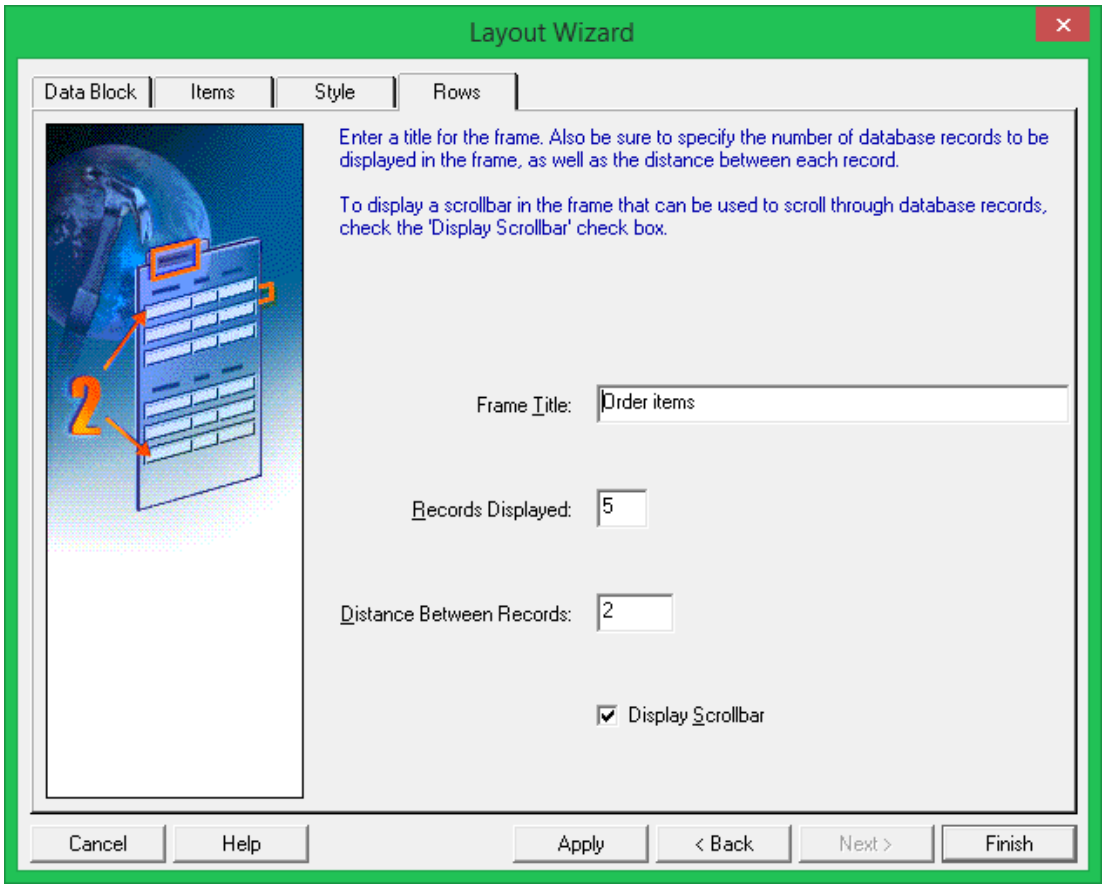


Hình 93.Quan hệ giữa master và detail block

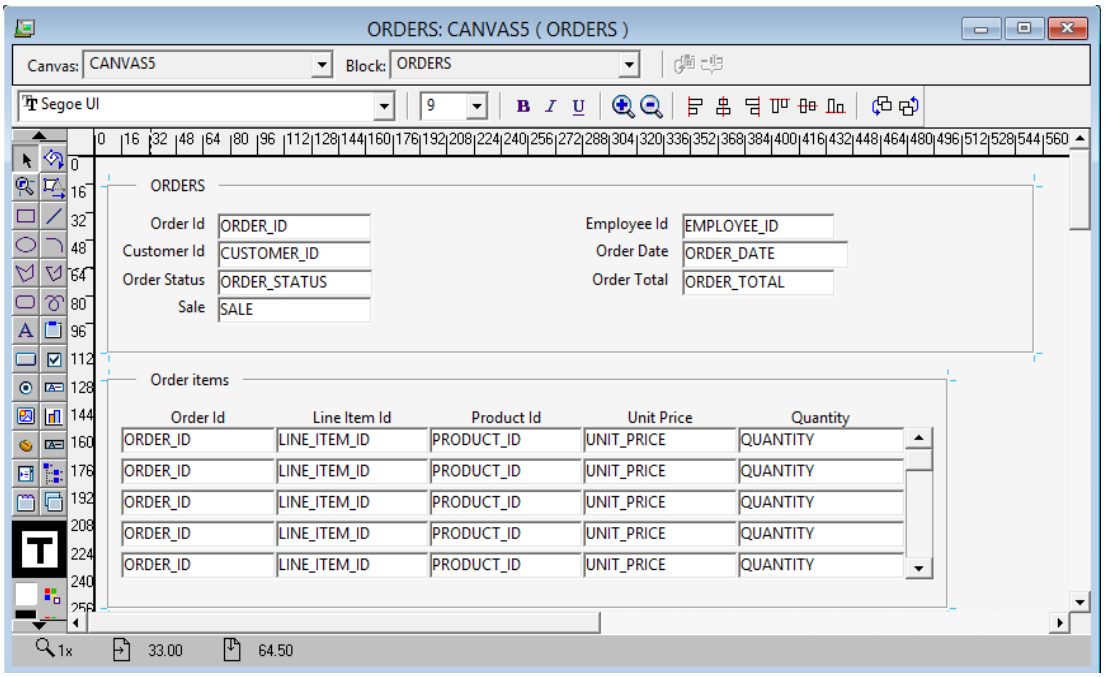
❖ ***<u>Bước 4:</u> Tạo layout để hiển thị detail block order_items***




Layout Wizard
Select a layout style for your frame by clicking a radio button below.
Form
Tabular
Cancel
Help
< Back
Next >
Finish



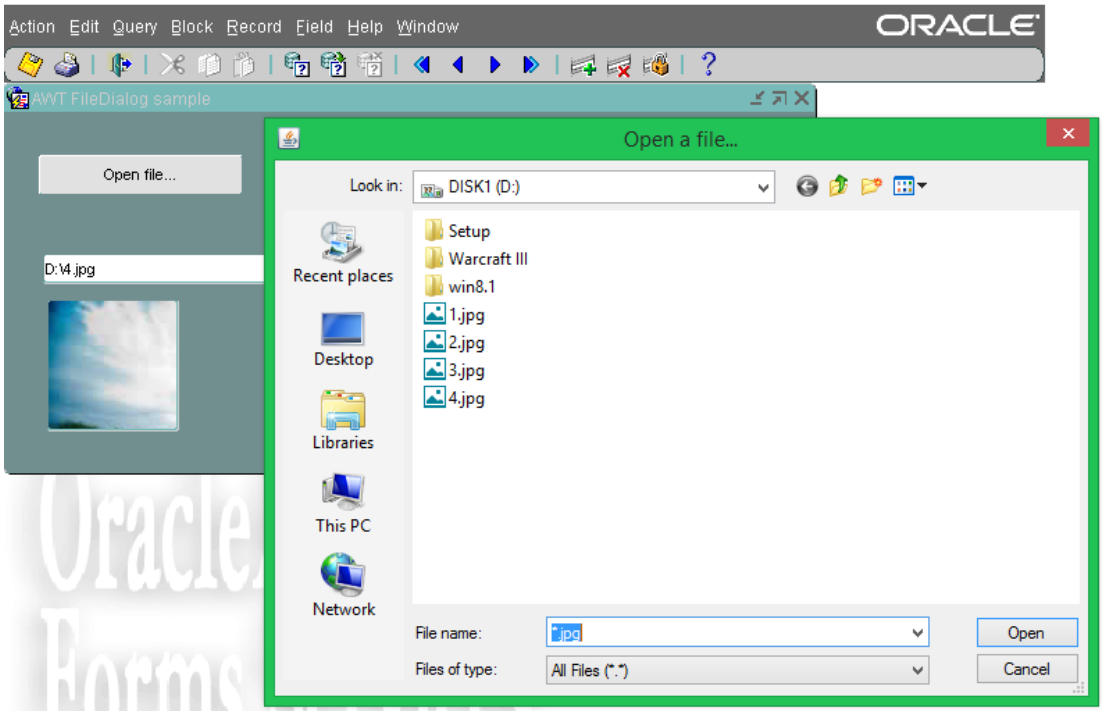
❖ ***Giao diện master-detail form: order-order items***



❖ Bài tập thực hành

1. Tạo mới 1 form module tên là ORDERS. Tạo mới một block sử dụng Data Block Wizard. Sử dụng bảng ORDERS bao gồm tất cả cả các cột ngoại trừ cột ORDER_TOTAL. Hiển thị các item trong block trên canvas tên là CV_ORDER và layout dạng form hiển thị từng bản ghi 1. Đặt tên frame trong canvas là Orders.
2. Tạo mới 1 block sử dụng Data Block Wizard sử dụng bảng ORDER_ITEMS bao gồm tất cả các cột. Tạo quan hệ với master data block ORDERS. Hiển thị tất cả các cột ngoại trừ cột ORDER_ID trên canvas CV_ORDER. Sử dụng layout dạng tabular hiện thị 6 bản ghi, có thanh scrollbar. Di chuyển block ORDER_ITEMS đặt sau block ORDERS. Đặt tên frame tương ứng với block này là Items.
3. Tạo mới một block sử dụng bảng INVENTORIES không có kết nối với 2 block trước và hiển thị trên 1 canvas mới tên là CV_INTENTORY có layout dạng tabular hiển thị 4 bản ghi, có thanh scrollbar. Sau đó di chuyển block INVENTORIES xuống sau block ORDER_ITEMS. Đặt tên frame là Stock.
4. Tạo một quan hệ tên là Order_Items_Inventories kết nối giữa block ORDER_ITEMS và INVENTORIES. Thiết lập quan hệ sao cho khi xóa các record bên block ORDER_ITEMS không ảnh hưởng gì đến bên INVENTORIES.
5. Trên block ORDER_ITEMS, thay đổi tên nhãn của item Line_Item_ID thành Item# sử dụng Layout Wizard.
6. Lưu trữ và Run Form.

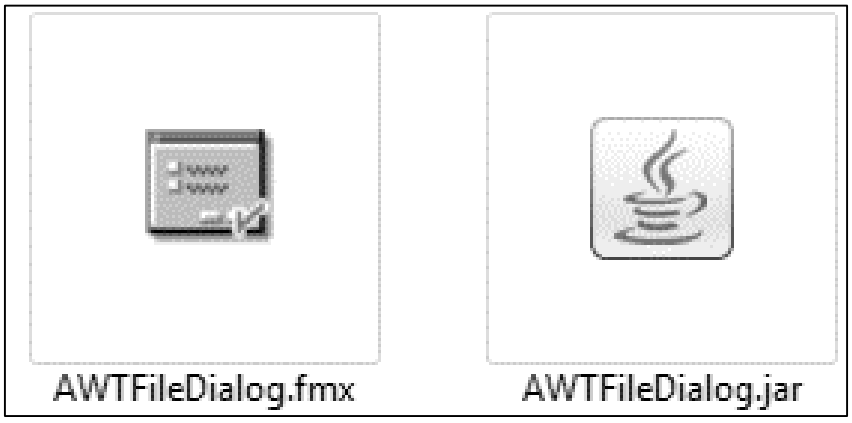
3.2.3.3.Cách tạo hộp thoại mở file trong form



Hình 94.*Ảnh minh họa open file dialog*

B1. Tải thư viện và cấu hình file formwebs.config và registry

Tải file AWTFileDialog.jar từ địa chỉ:
http://www.mediafire.com/file/866q4qw0mp43crb/AWTFileDialog.zip



Hình 95.*File form demo và file thư viện jar*

B2. Mở file formsweb.cfg bằng notepad và chỉnh sửa ở các vị trí như sau:

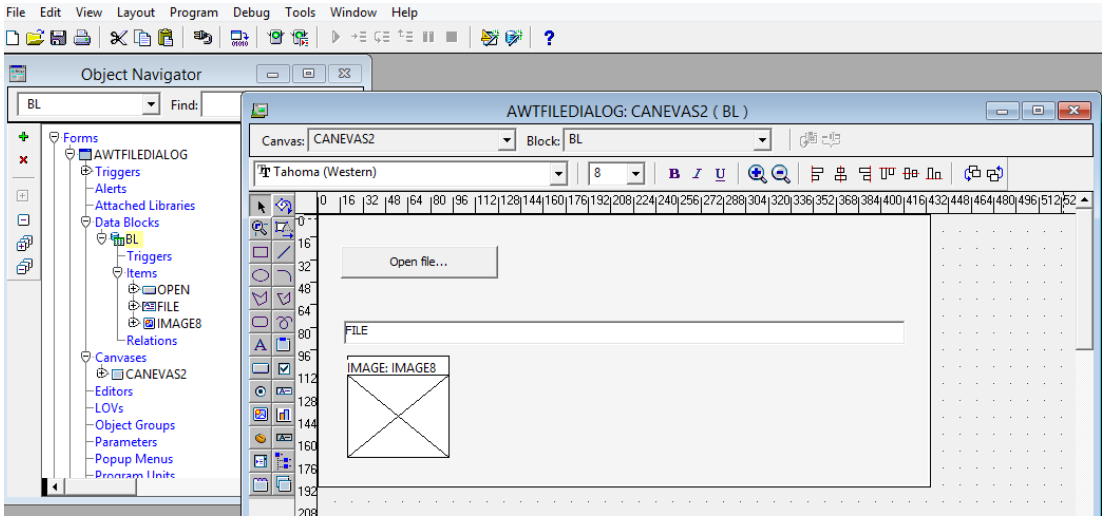
Bổ sung dòng chữ: *AWTFileDialog.jar* vào cuối các dòng bắt đầu bằng chữ *archive_jini* và *archive.* Sau đó lưu lại.

B3. Bổ sung đường dẫn vị trí đặt file vào biến môi trường FORMS_BUILDER_CLASSPATH trong Registry Editor HEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_OHXXXXXXX

Ví dụ: ;*C:\Oracle\Middleware\Oracle_FRHome1\forms\java\AWTFileDialog.jar*

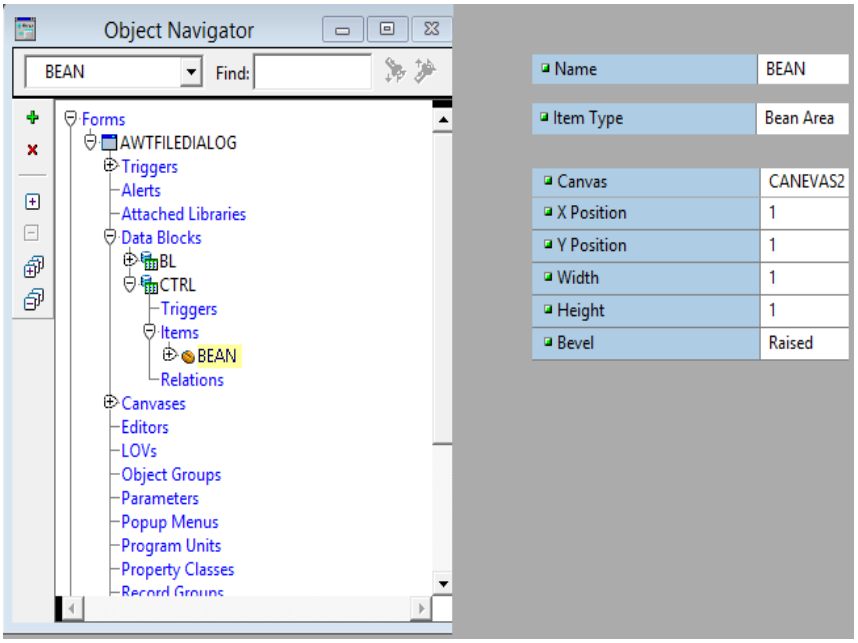
❖ **Quy trình tạo hộp thoại mở file**

- Tạo 1 data block gồm 1 button, 1 text item, 1 image item có canvas như hình dưới.



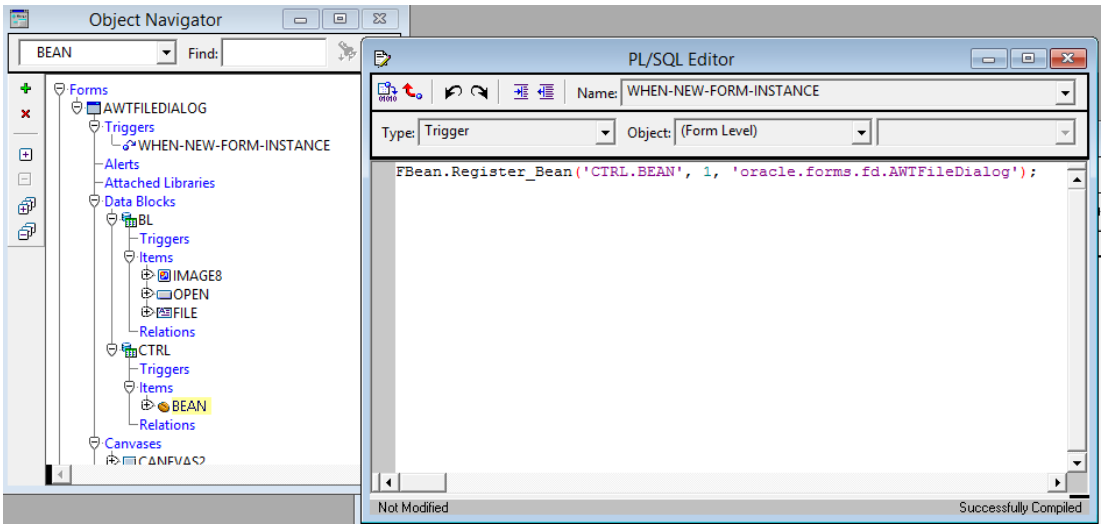
Hình 96.*Tạo giao diện*

- Tạo 1 data block CTRL chứa 1 bean area item



Hình 97.*tạo bean area item*

- Đăng ký Bean khi load form trong sự kiện WHEN-NEW-FORM-INSTANCE
FBean.Register_Bean('CTRL.BEAN', 1, 'oracle.forms.fd.AWTFileDialog');



Hình 98.*Đăng ký Bean khi load form*

➢ **Để trả về đường dẫn file được mở, ta gọi function:**

FBean.Invoke_Char('name_bean_area_item',1,'openFile','title_dialog,link,file_type_list');

Trong đó:

- **name_bean_area_item:** chính là tên bean area item chứa trong data block CTRL
- **openFile:** từ khóa xác định loại hộp thoại. Có thể thay thành saveFile để chuyển thành hộp thoại lưu file.
- **title_dialog:** tiêu đề khi hộp thoại được hiện lên.
- **link:** đường dẫn thư mục mặc định được chọn khi mở hộp thoại.
- **file_type_list:** Danh sách loại file muốn mở. Phân cách nhau bởi dấu **;**

(VD: *.* mở tất cả loại file hoặc *.jpg;*.gif: mở các file ảnh có đuôi jpg, gif)

VD: FBean.Invoke_Char('CTRL.BEAN', 1, ‘openFile' ,'Open a file...,D:\,*.jpg;*.gif');

➢ **Để đưa đường dẫn file đã mở vào 1 text item, ta thực hiện câu lệnh:**

:blockname.text_item_name:=FBean.Invoke_Char('name_bean_area_item',1,'openFile','title_dialog,link,file_type_list');

Ta viết 1 trigger để thử nghiệm:

VD: Khi bấm vào nút OPEN thì hiện lên hộp thoại mở file, ta viết trong trigger WHEN-BUTTON-PRESSED như sau:

:BL.FILE := FBean.Invoke_Char('CTRL.BEAN', 1, 'openFile' ,'Open a file...,D:\,.jpg;*.gif');*

➢ **Để hiển thị hình ảnh lên Image item từ một đường dẫn ảnh, ta thực hiện câu lệnh:**

read_image_file('Đường dẫn','file_type_list','TênBlock.TênImageItem');

Ta viết 1 trigger để thử nghiệm:

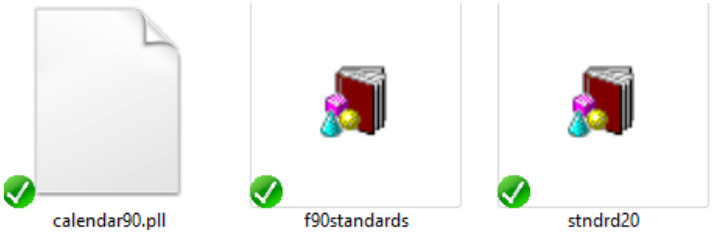
VD: Khi bấm vào nút OPEN thì hiện lên hộp thoại mở file, sau khi mở file ảnh thì hiển thị hình ảnh lên Image Item, ta viết trong trigger WHEN-BUTTON-PRESSED như sau:

:BL.FILE := FBean.Invoke_Char('CTRL.BEAN', 1, 'openFile' ,'Open a file...,D:\,.jpg;*.gif');*

read_image_file(:BL.FILE,'JPG','BL.IMAGE8');

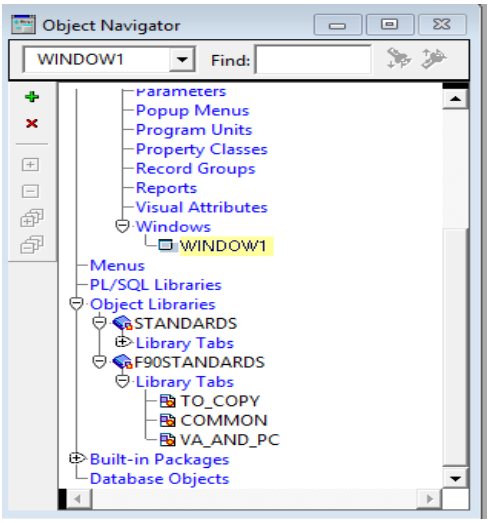
3.2.3.4.Calendar – tạo hộp thoại chọn ngày trên form

Bước 1: Tải thư viện calendar tại địa chỉ: http://1drv.ms/1zkAcQz



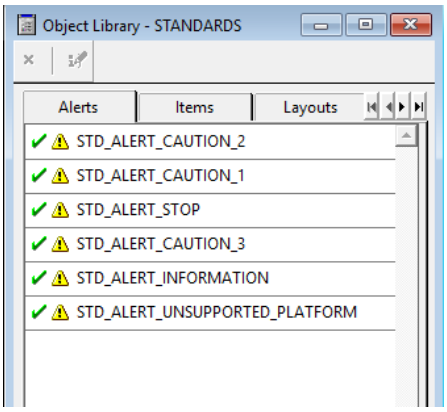
Hình 99.*Các file thư viện*

Bước 2: File/Open => Tìm chọn đến file: f90standards.olb



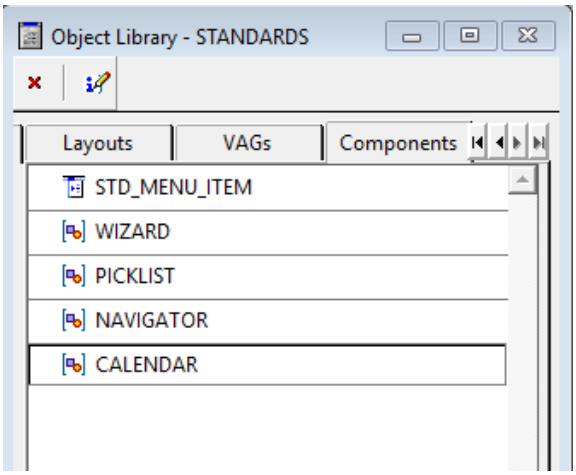
Hình 100.*Thêm thư viện f90standards.olb vào module*

Bước 3: Phải chuột vào STANDARDS trong Object Libraries, chọn Object Library



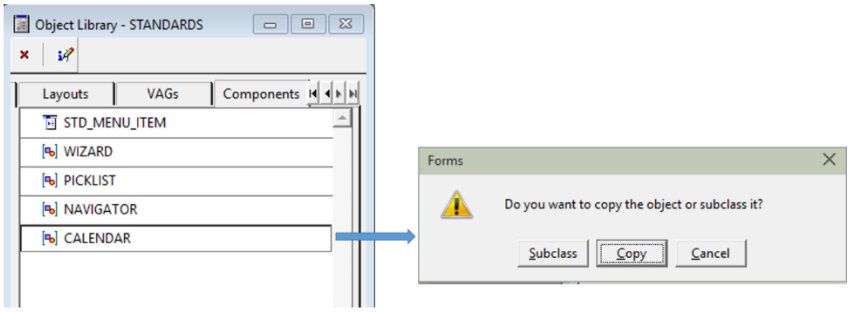
Hình 101.*Minh họa bước 3*

Bước 4: Chọn tab Componments

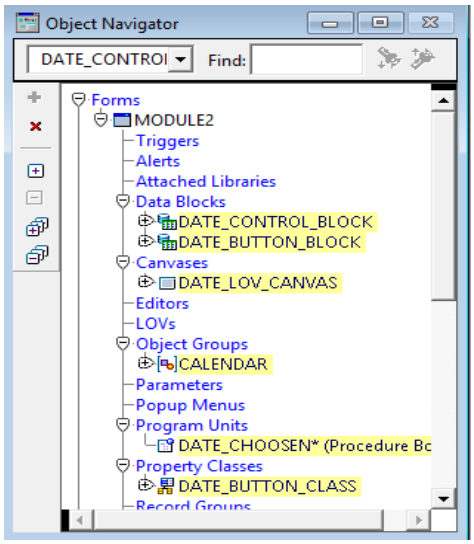


Hình 102.*Tab Componments*

Bước 5: Kéo thành phần CALENDAR thả vào module đang làm việc, hiện lên hộp thoại hỏi, ta chọn Copy

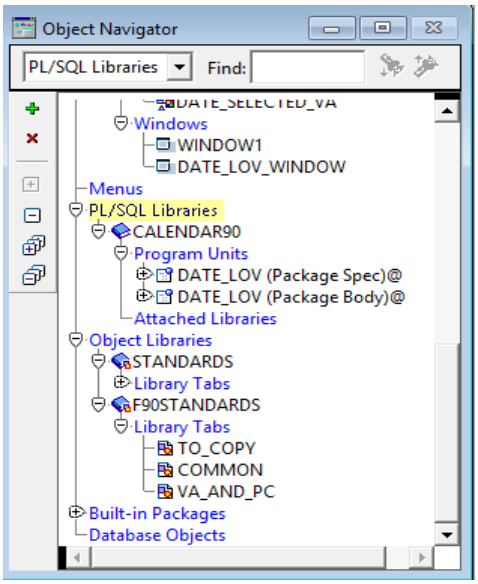


Hình 103.*Minh họa bước 5*



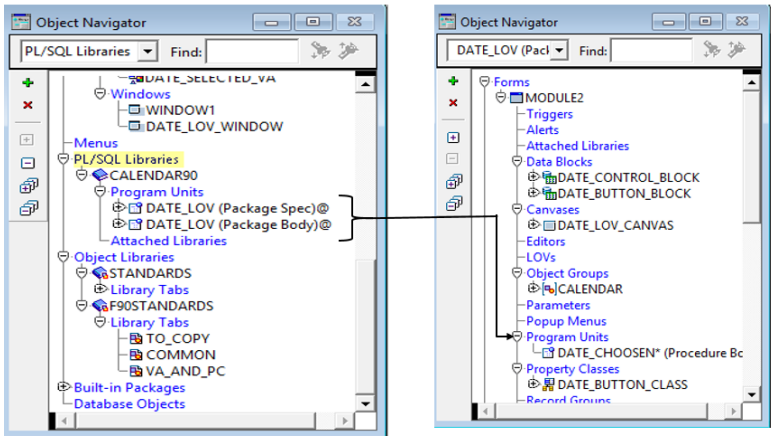
Hình 104.*Kết quả sau khi thực hiện bước 5*

Bước 6: File/Open => Tìm chọn đến file: calendar90.pll



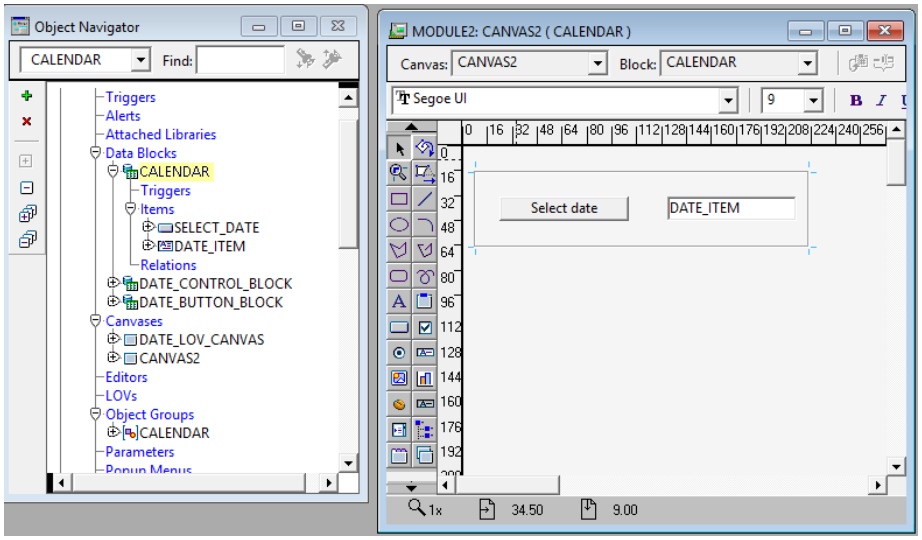
Hình 105.Minh họa bước 6

Bước 7: Kéo 2 package DATE_LOV trong Program units của CALENDAR90 vừa mở sang mục Program units của Module đang làm việc.



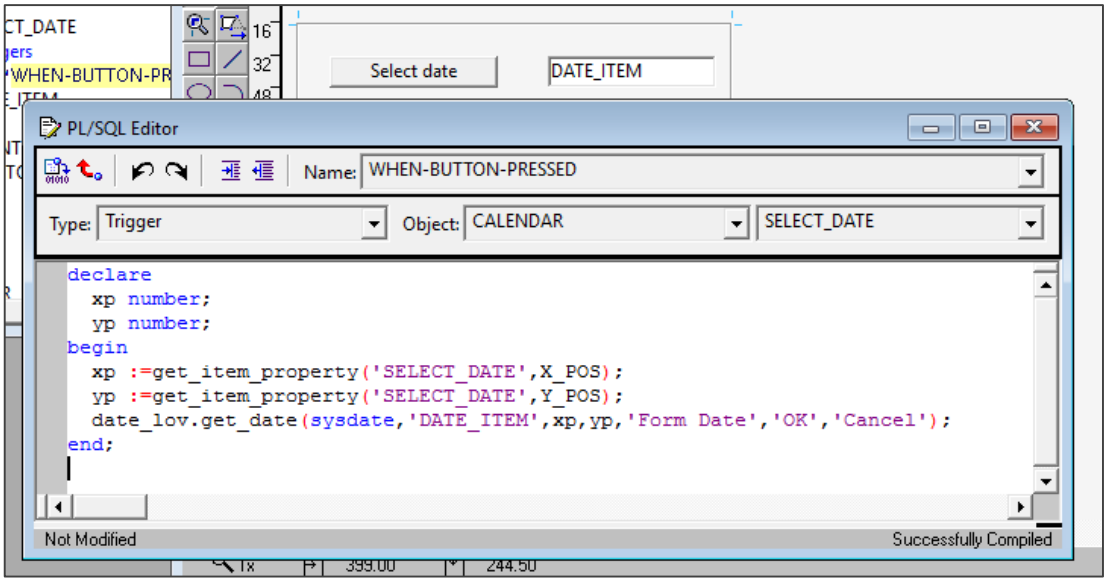
Hình 106.*Minh họa bước 7*

Bước 8: Tạo 1 block mới gồm 1 button và 1 text item, tạo canvans mới hiển thị các item của block mới như hình dưới.



Hình 107.*Minh họa bước 8*

Bước 9: Viết trigger cho sự kiện click vào button Select date như sau:



Hình 108.*Viết trigger cho button*

3.2.3.5.Sequences – tạo id tự động tự tăng

Sequence là danh sách tuần tự của con số, và được tạo bởi Oracle sever. Sequence dùng để tạo khóa chính một cách tự động cho dữ liệu.

Sequence thường dùng để tạo khóa chính trong sinh mã tự động. Có thể dùng chung cho nhiều đối tượng. Con số sequence này có chiều dài tối đa là 38 số.

Để tạo sequence, dùng lệnh CREATE SEQUENCE

Cú pháp:

CREATE SEQUENCE sequence_name

INCREMENT BY integer

START WITH integer

[MAXVALUE integer]

[MINVALUE integer]

[CYCLE/NO CYCLE];

Ví dụ:

CREATE SEQUENCE sample_sequence INCREMENT 1 START WITH 2

MAXVALUE 100;

Để làm việc với các sequence, dùng lệnh SQL với các cột giả sau:

CURRVAL Cho giá tri hiện thời của sequence

NEXTVAL Tăng giá tri hiện thời của sequence và cho giá trị sau khi tăng phải xác định tên sequence trước currval và nextval

sequence.CURRVAL

sequence.NEXTVAL

Ví dụ: Bảng demo có 2 trường: ID number, name nvarchar2(100). Câu lệnh sql thêm 1 bản ghi vào bảng có sử dụng sequence như sau:

insert into demo values(seqDemo.nextval,'text');

- **Lấy giá trị của sequence đưa lên text item trên form**

Thông tin data block demo:

- Tên data block: DemoSEQ
- Gồm 2 text item: ID, NAME

Để lấy giá trị ở sequence DEMO_SEQ đưa lên text item ID, ta sử dụng câu lệnh (Viết trong 1 trigger nào đó):

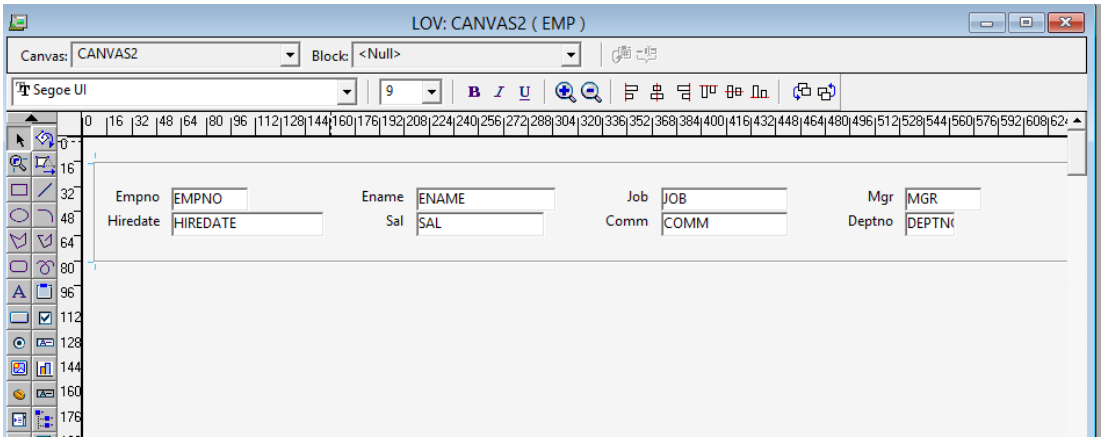
select DEMO_SEQ.nextval into :DemoSEQ.ID from dual;

Bài tập thực hành

Tạo một form có trường ID tự động tăng sử dụng sequence.

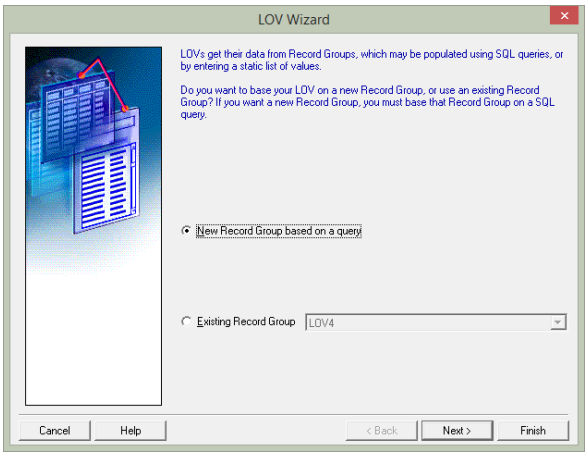
3.2.3.6.Tạo LOV (list of value)

Bước 1: Tạo một module có datablock cần tạo lov



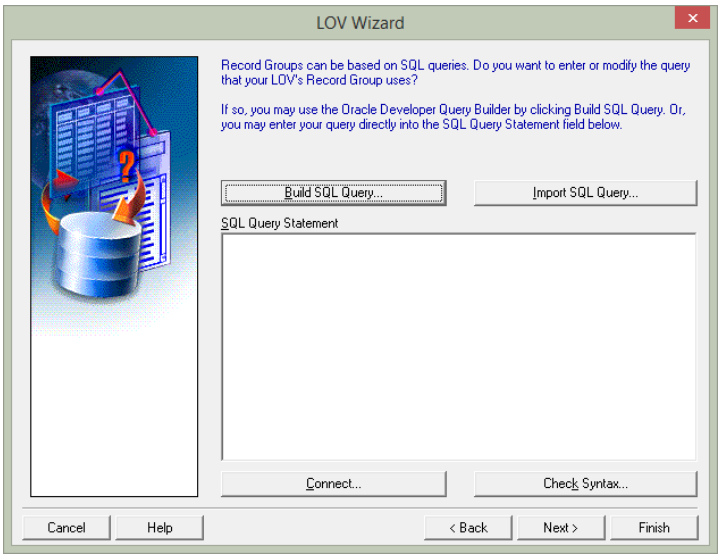
Hình 109.*Demo Form hiển thị thông tin bảng EMP*

Bước 2: Chuột phải vào module, chọn Lov Wizard



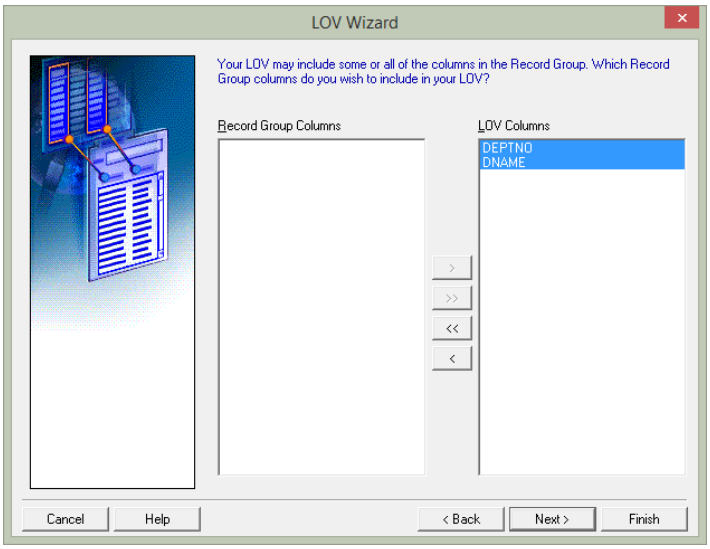
Hình 110.*Giao diện Lov Wizard*

Bước 3: Next, viết câu lệnh SQL. Có thể chọn Build SQL Query để chọn bảng dữ liệu



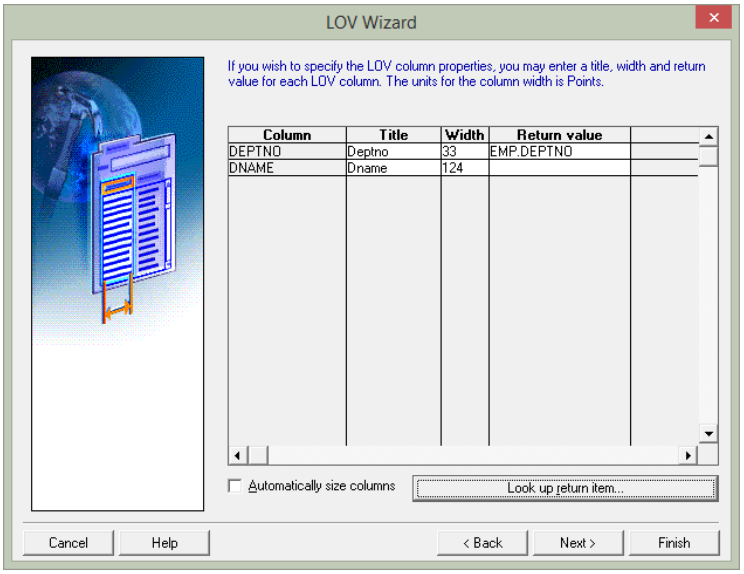
Hình 111.*Viết lệnh SQL cho LOV*

Bước 4: Next, Chọn các cột hiển thị trên lov



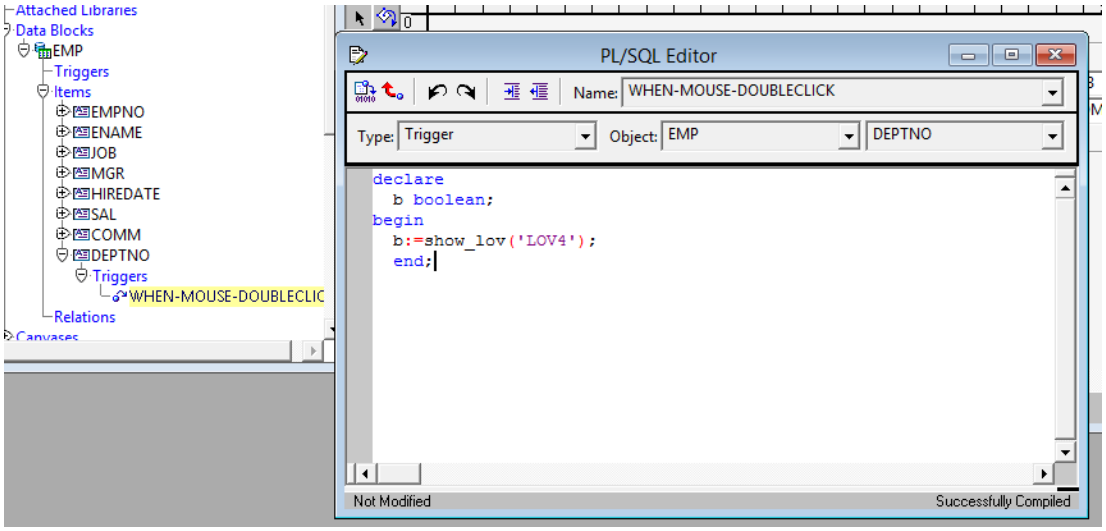
Hình 112.*Chọn các cột hiển thị trên LOV*

Bước 5: Next, Chọn cột cần trả về giá trị cho textitem trên datablock ở ô Return value. Có thể bấm vào Look up return item... Để tìm text-item cần trả về giá trị trên datablock.



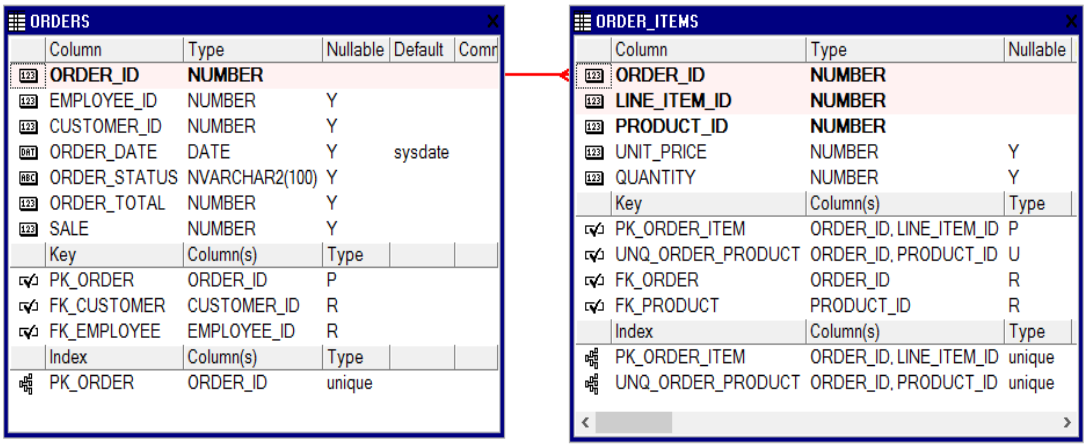
Hình 113.*Xác định item sẽ trả về giá trị*

Bước 6: Viết trigger cho textitem cần hiển thị LOV



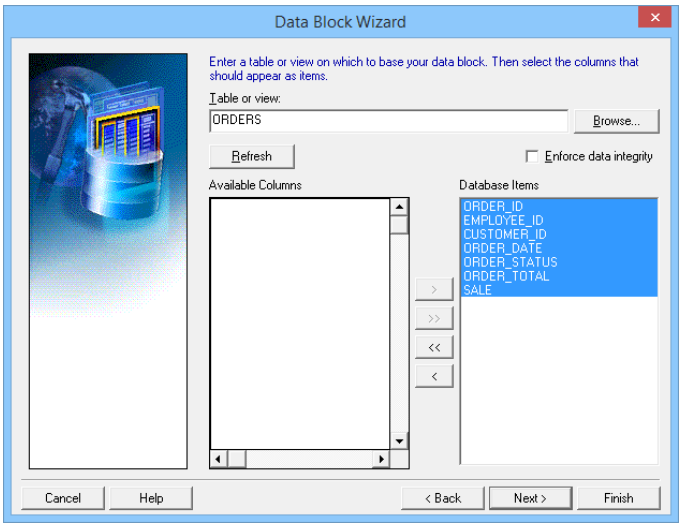
3.2.3.7.Hướng dẫn làm 1 form hóa đơn đơn giản

Mô hình quan hệ 2 bảng hóa đơn và chi tiết hóa đơn

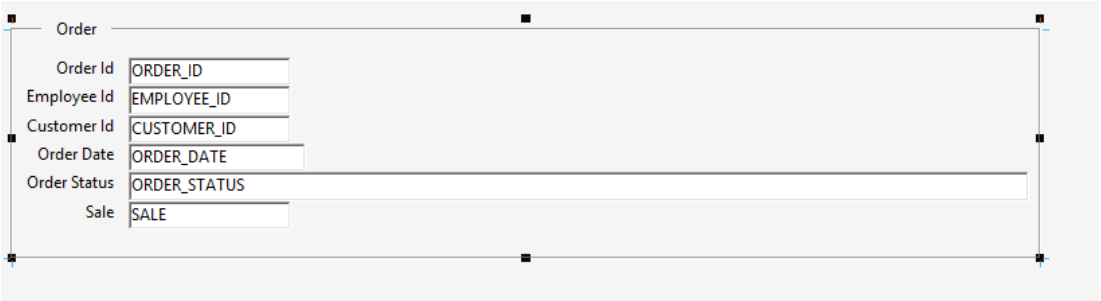


Hình 114.Mô hình quan hệ mẫu

Bước 1: Tạo Block và layout master sử dụng bảng dữ liệu ORDERS



Hình 115.Block Orders

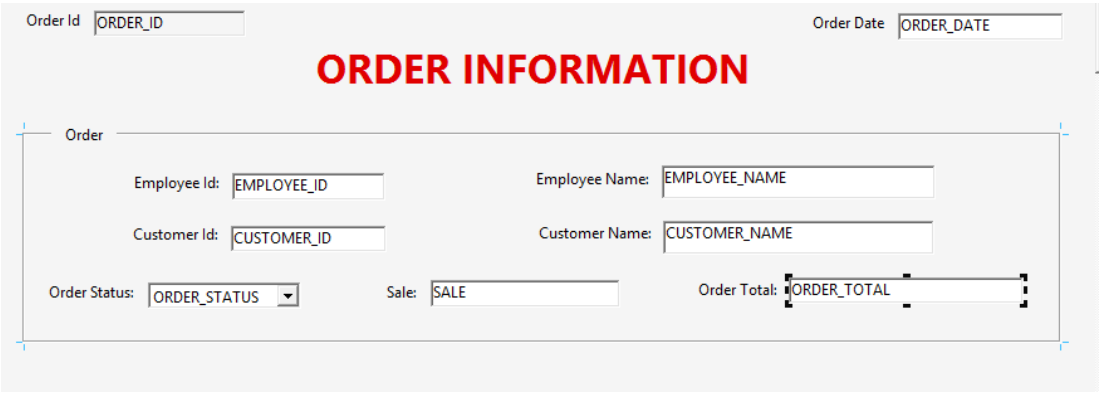


Hình 116.*Giao diện mặc định được tọa ra*

Bước 2: Căn chỉnh lại vị trí các item và chỉnh sửa giao diện

Một số chú ý cho việc căn chỉnh giao diện dễ dàng:

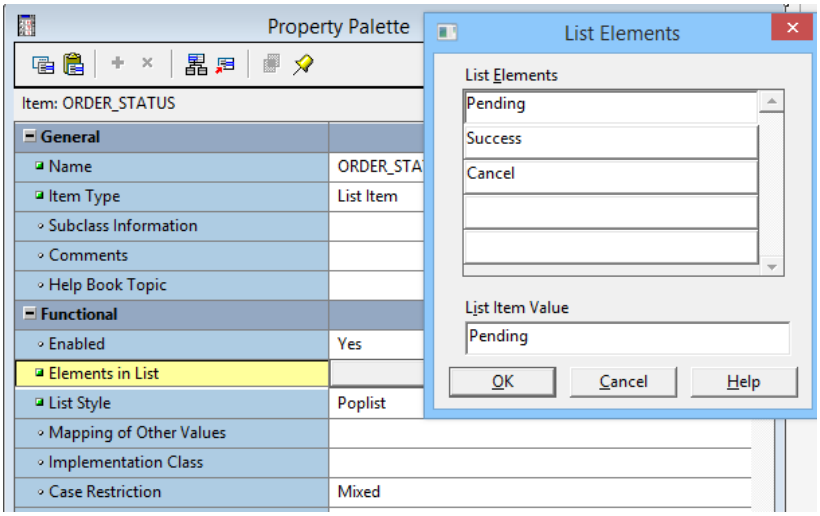
- Để di chuyển frame, đưa trỏ chuột vào dòng kẻ frame, sau đó bấm giữ chuột và kéo thả đến vị trí cần di đặt.
- Để thay đổi kích thước frame, bấm chuột vào dòng kẻ frame, sẽ thấy các chấm đen trên các vị trí chính giữa và đỉnh các dòng kẻ, bấm giữ chuột vào các vị trí chấm đen đó và kéo thả đến kích thước thích hợp.



Hình 117.Giao diện sau khi chỉnh sửa

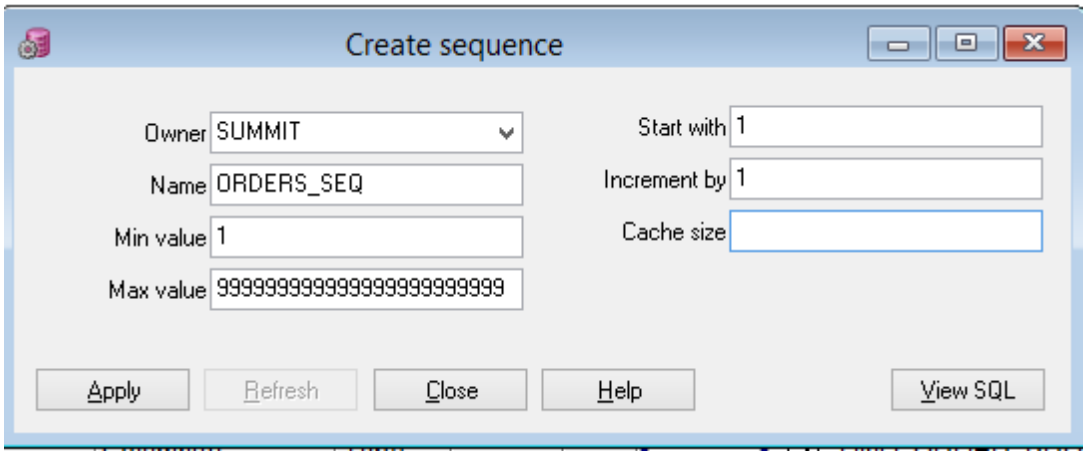
Bước 3: Xử lý chi tiết Block ORDERS

- *Bước 3.1. Thay đổi Item Type của ORDER_STATUS thành List Item và thêm các Elements như hình dưới.*



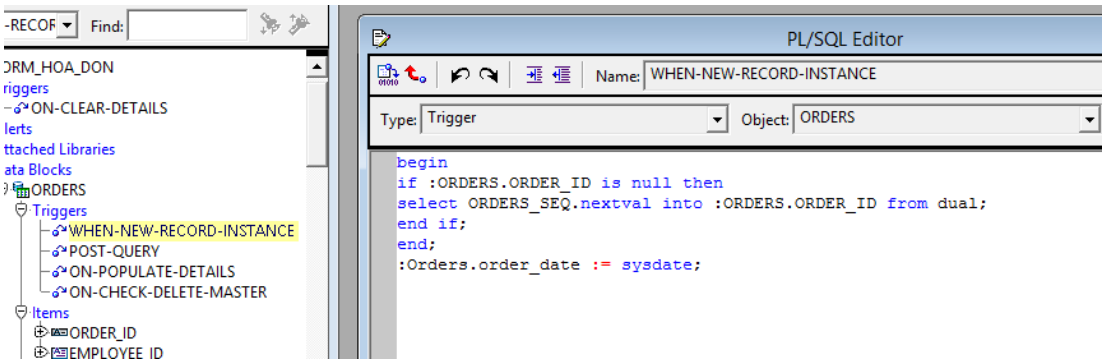
Hình 118.*Thay đổi thuộc tính của ORDER_STATUS*

- *Bước 3.2. Order_ID tự động tăng: - Sử dụng PL/SQL DEVELOPER để tạo 1 sequence nhanh chóng sử dụng để tăng id tự động cho order_id*



Hình 119.*Tạo ID tự động tăng*

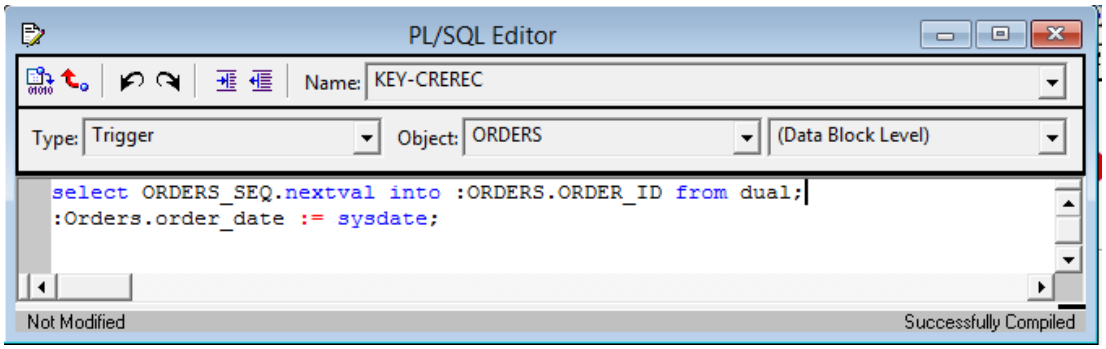
- *Tạo trigger WHEN-NEW-RECORD-INSTANCE trong block ORDERS để mỗi khi thêm mới 1 bản ghi thì tự động ID tăng lên.*



Hình 120.*Trigger tự động chèn ID vào ORDER_ID khi thêm bản ghi.*

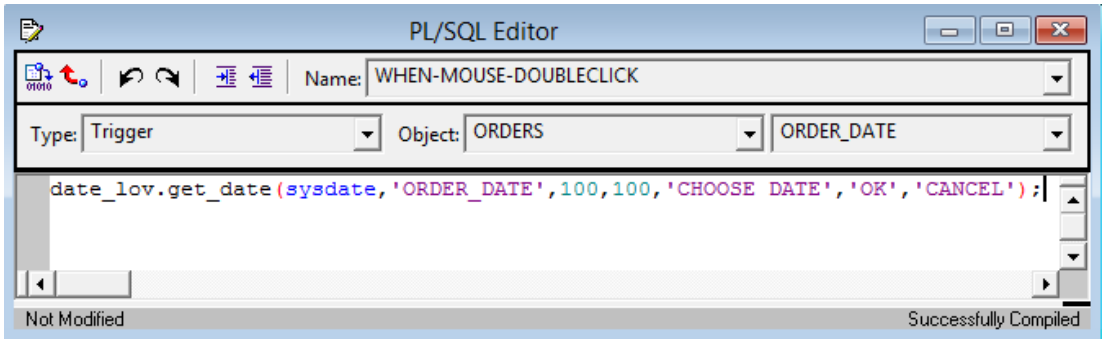
- *Bước 3.3. Ngày hiện tại cho ORDER_DATE*

Lấy ngày hiện tại cho Item ORDER_DATE bằng cách bổ sung câu lệnh gán ngày vào trigger KEY_CREREC như hình dưới.



Hình 121.*Trigger KEY-CRECEC để xử lý ID tự động tăng và lấy ngày hiện tại cho ORDER_DATE*

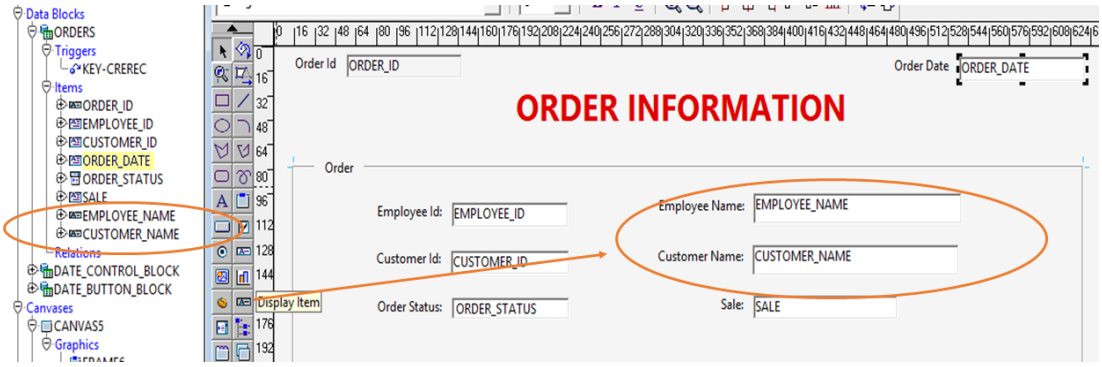
- *Bước 3.4. Tạo Calendar chọn ngày khi click đúp vào ORDER_DATE*
  - Các bước tạo calendar như đã học ở bài trước.
  - Tạo trigger WHEN-MOUSE-DOUBLECLICK



Hình 122.*Trigger WHEN-MOUSE-DOUBLECLICK để tạo chọn lịch khi click đúp vào ORDER DATE*

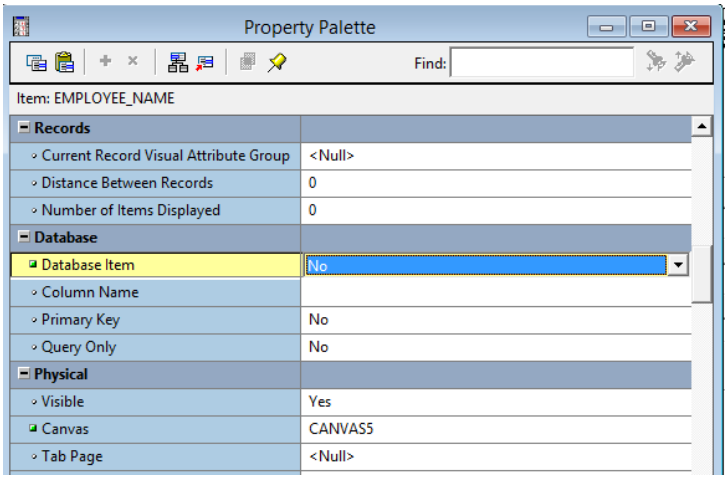
- *Bước 3.5. Bổ sung tên khách hàng, tên nhân viên vào block*

Vì trong bảng ORDER không có trường tên khách, tên nhân viên mà chỉ có ID tương ứng, nên ta sẽ tạo thêm 2 display item để hiển thị tên khách hàng, tên nhân viên tương ứng với ID có trong bảng Order.



Hình 123.*Bổ sung thêm 2 display item*

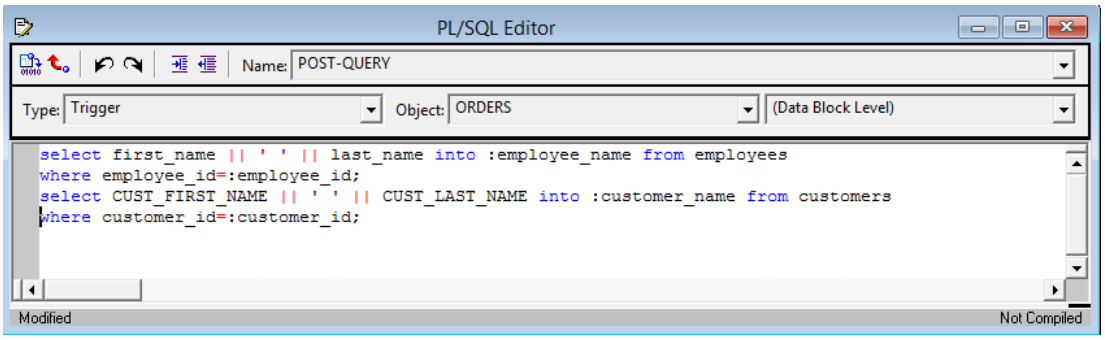
- Thay đổi thuộc tính Database Item của 2 Display Item vừa tạo thành **No** vì 2 Item này không có trong bảng ORDERS



Hình 124.*Thay đổi thuộc tính Database Item*

- *Bước 3.6. Tự động đổ dữ liệu tương ứng với ID nhân viên, ID khách hàng vào EMP_NAME và CUS_NAME khi next record trên form*

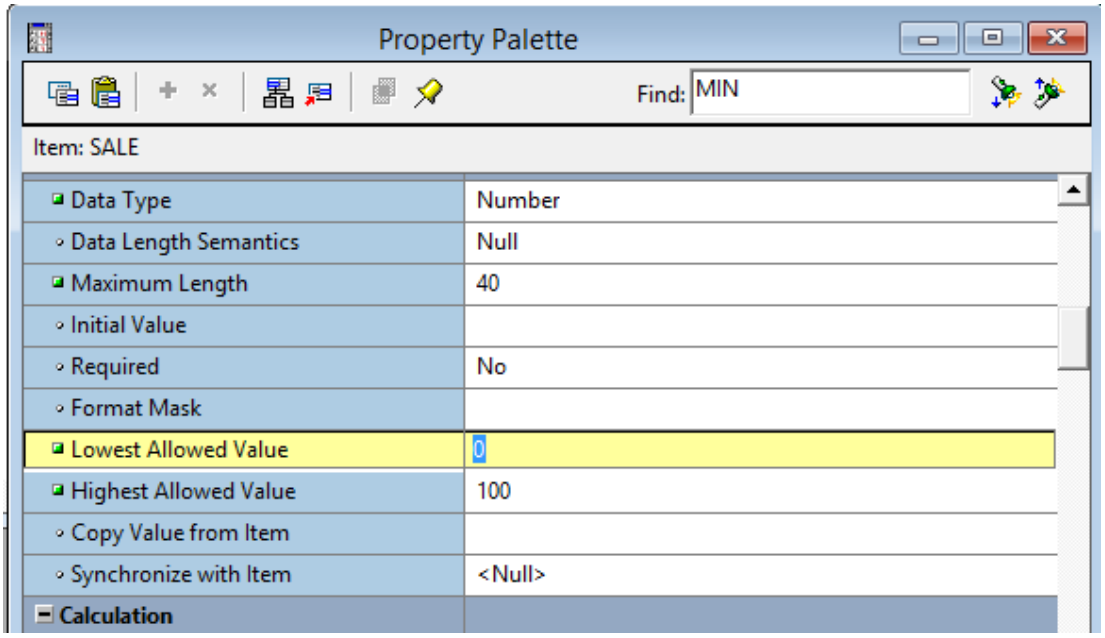
Viết trong trigger PORT-QUERY của block ORDERS các câu lệnh dưới để lấy dữ liệu tương ứng đổ vào 2 display item



Hình 125.*Trigger POST-QUERY để tự động đổ dữ liệu tương ứng cho 2 display item*

- *Bước 3.7. Thiết lập giá trị nhỏ nhất và lớn nhất cho Item SALE*

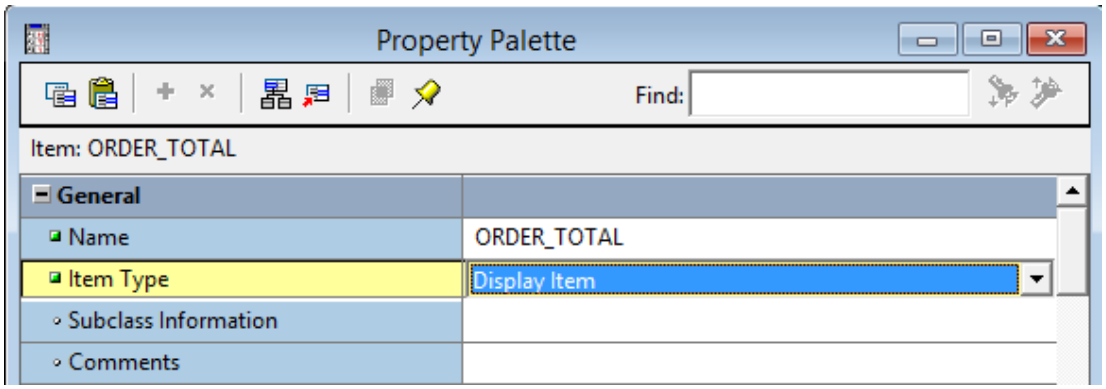
Vì giảm giá không thể âm hoặc lớn hơn 100% nên ta thiết lập các thuộc tính Lowest Allowed Value và Highest Allowed Value như hình dưới



Hình 126.*Thiết lập giá trị nhỏ nhất và lớn nhất cho Item SALE*

- *Bước 3.8. Thay đổi Item Type của ORDER_TOTAL thành Display Item*

Vì ORDER_TOTAL được tự động tính bằng tổng thành tiền trong ORDER_ITEMS nhân với SALE nên không cần phải nhập bằng tay nên ta chuyển nó thành **display item.**



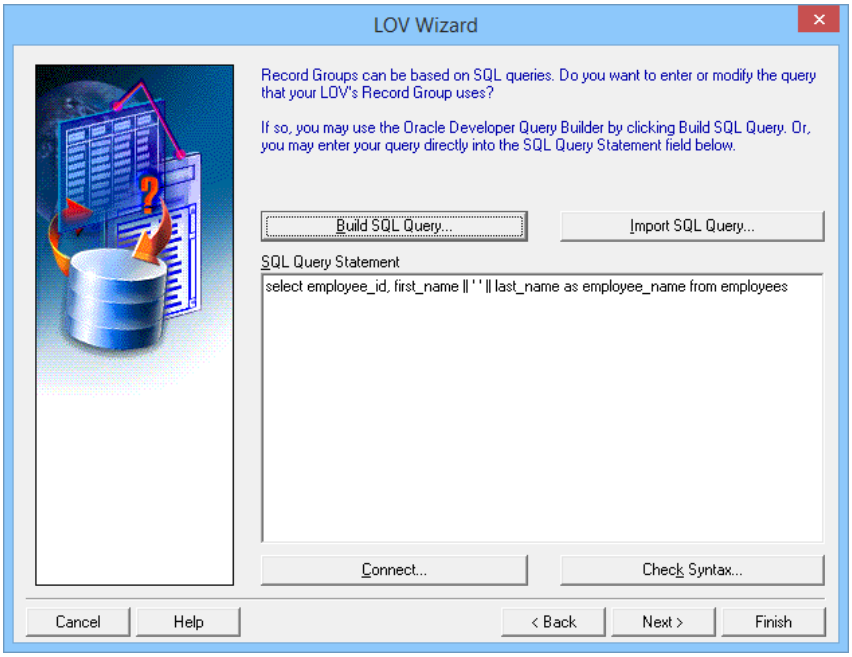
Hình 127.*Thay đổi Item Type của ORDER_TOTAL thành Display Item*

- *Bước 3.9. Tạo LOV cho EMPLOYEE VÀ CUSTOMER*

Câu lệnh sql ta chỉ cần lấy ID, first_name và last_name của bảng employees:

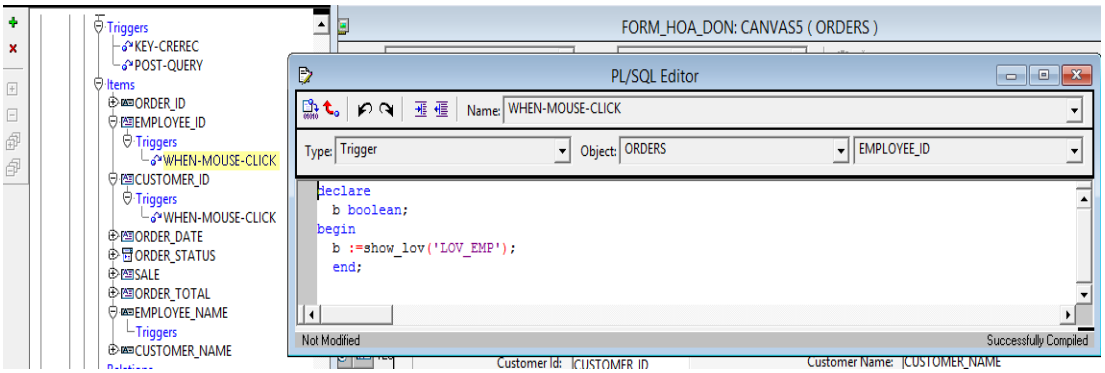
SQL:

select employee_id, first_name || ' ' || last_name as employee_name from employees;



Hình 128.*Tạo LOV cho EMPLOYEE*

- Viết trigger WHEN-MOUSE-CLICK cho 2 text item EMPLOYEE_ID và CUSTOMER_ID

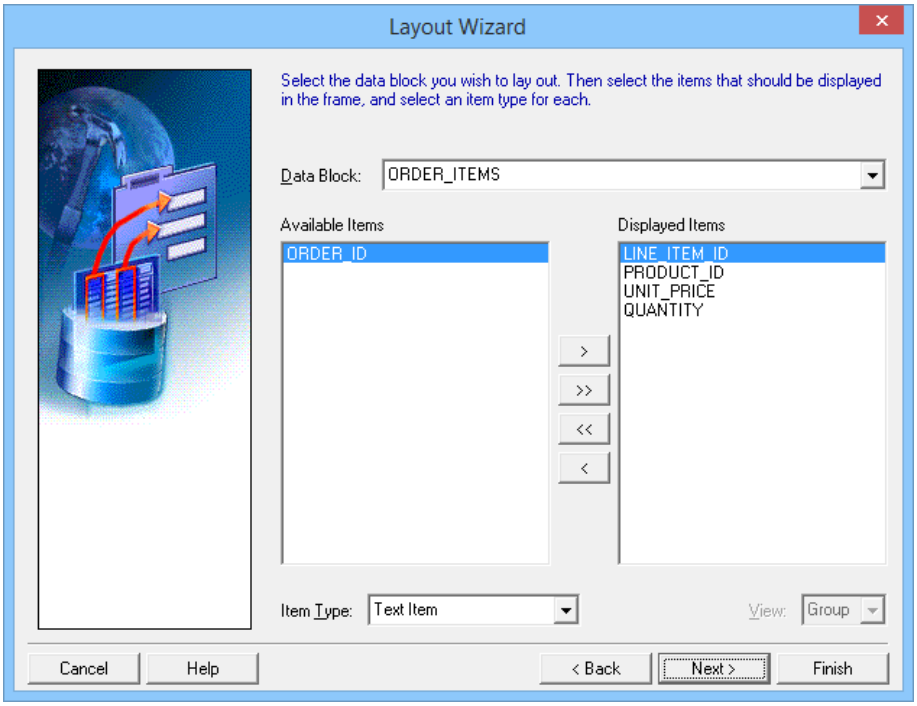


Hình 129.*Trigger show lov*

Bước 4: Tạo detail block ORDER_ITEMS

- *Bước 4.1. Tạo Block và Layout dạng tabular cho block ORDER_ITEMS*

Vì đây là chi tiết các sản phẩm mua trong 1 hóa đơn nên không cần hiển thị trường ORDER_ID vì đã có ở trên master-block.



Hình 130.*Chọn các trường sẽ hiển thị ra giao diện*

- *Bước 4.2. Chỉnh sửa giao diện cho detail block*

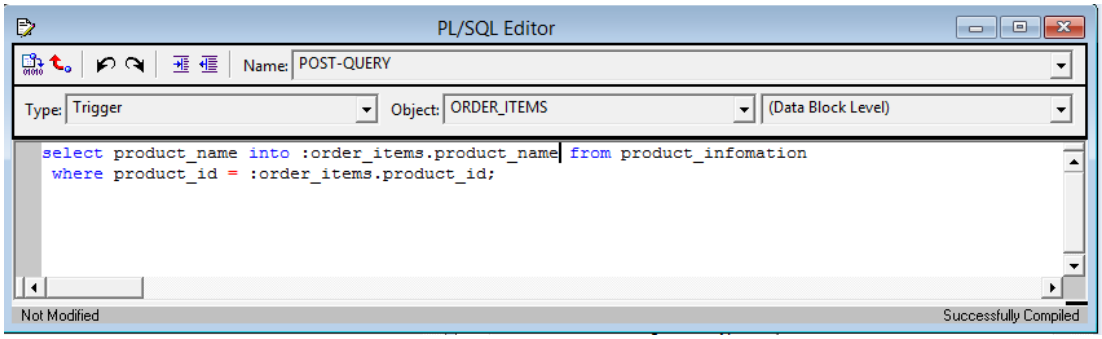
Bổ sung 2 text item vào block **order_items** là **PRODUCT_NAME** và **ITEM_TOTAL** để hiện thị tên sản phẩm tương ứng với ID và tính thành tiền cho từng sản phẩm.

Chú ý: thuộc tính Database Item của 2 item phải thiết lập là **No.**

Hình 131.*Giao diện sau khi chỉnh sửa*

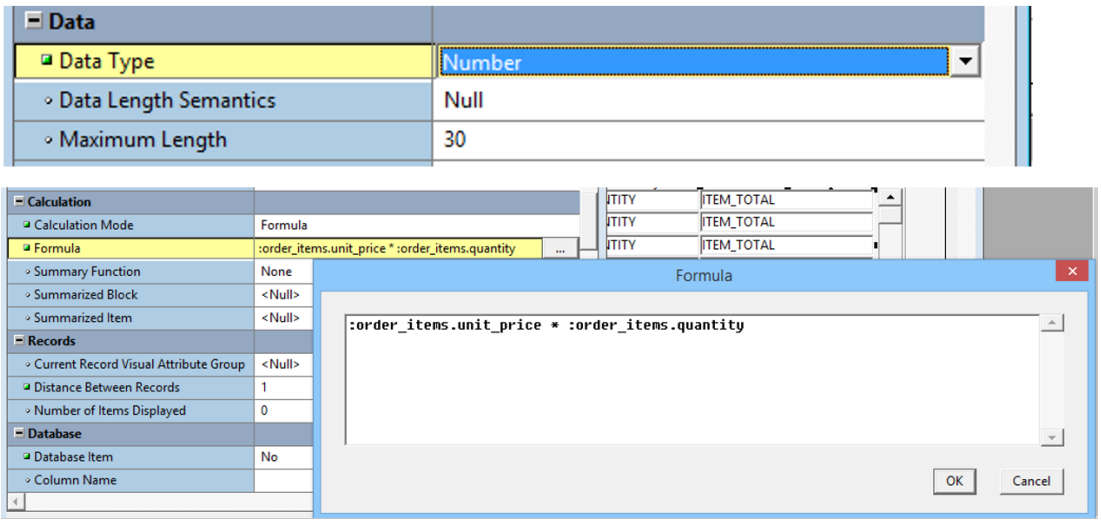
Bước 5: Xử lý chi tiết detail block ORDER_ITEMS

- *Bước 5.1. Viết trigger tự động đổ tên sản phẩm vào item PRODUCT_NAME tương ứng với PRODUCT_ID*



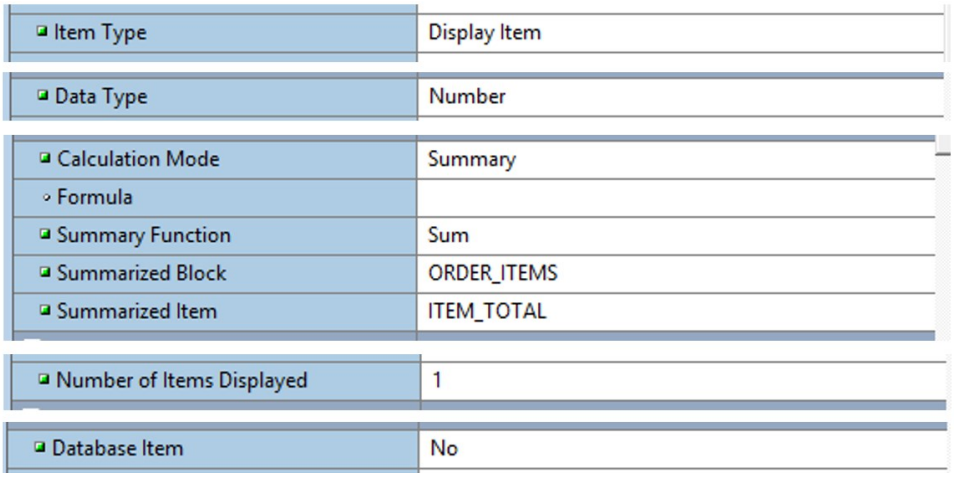
Hình 132.*Trigger tự động đổ tên sản phẩm vào item PRODUCT_NAME tương ứng với PRODUCT_ID*

- *Bước 5.2. Thiết lập các thuộc tính cho textitem ITEM_TOTAL để tính thành tiền tự động*



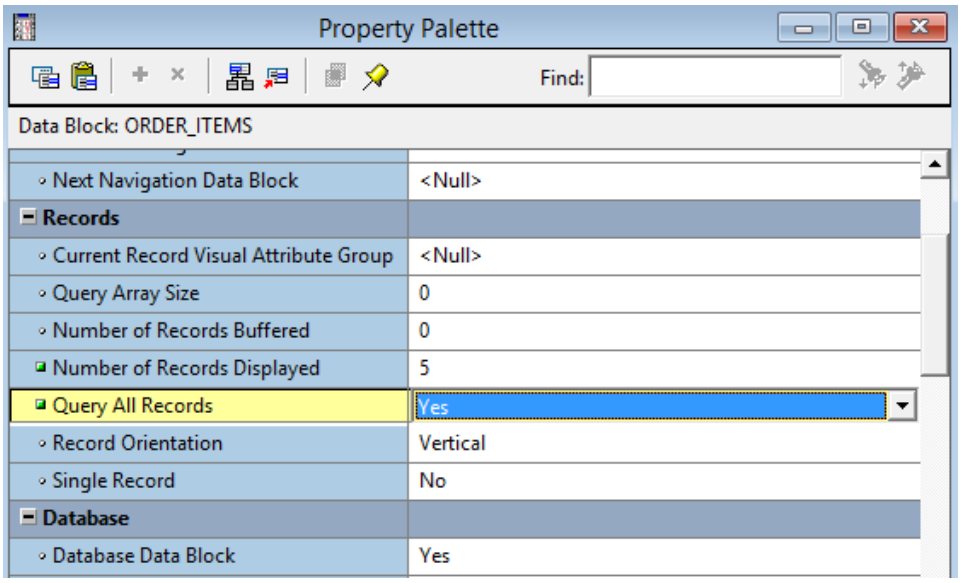
Hình 133.*Thiết lập các thuộc tính cho textitem ITEM_TOTAL để tính thành tiền tự động*

- *Bước 5.3. Tạo 1 DisplayItem ORDER_TOTAL trong block ORDER_ITEMS để tự động tính tổng tiền của hóa đơn.*



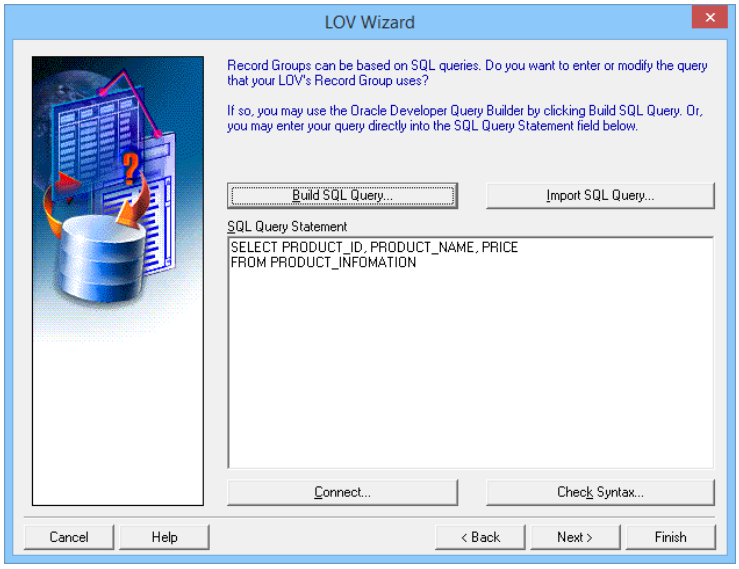
Hình 134.*Thay đổi các thuộc tính của ORDER_TOTAL như trên*

- *Bước 5.5. Thiết lập thuộc tính Query All Records của Datablock ORDER_ITEMS thành Yes*



Hình 135.*Thiết lập thuộc tính Query All Records*

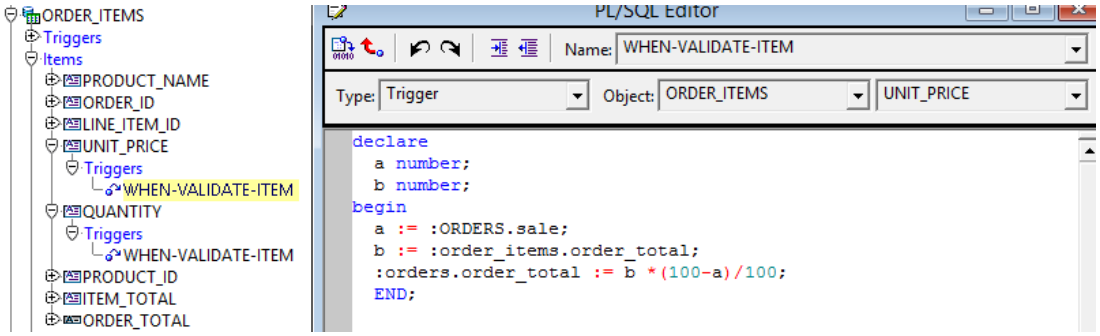
- *Bước 5.6. Tạo LOV chọn sản phẩm*



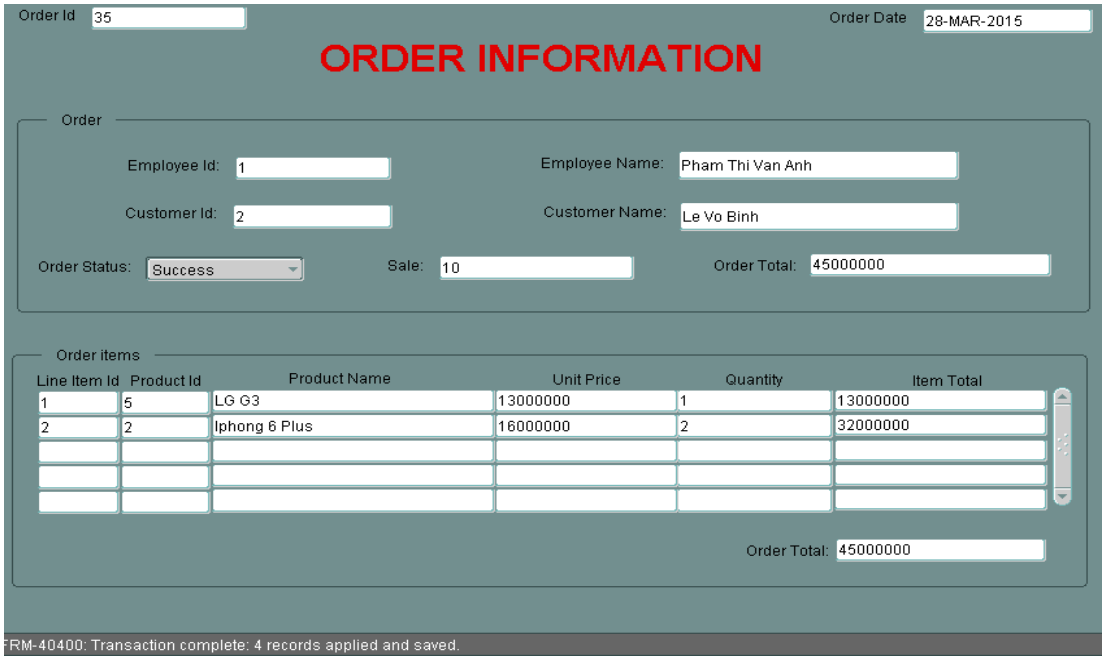
Hình 136.*Tạo LOV chọn sản phẩm*

- *Bước 5.7. Viết trigger tự động tính tổng tiền sau khi giảm giá*

Khi sale, unit_price, quantity thay đổi thì tổng tiền thay đổi, nên ta viết trigger When-Validate-Item đối với từng item để tính tổng tiền như sau:



Hình 137.*trigger tự động tính tổng tiền sau khi giảm giá*



Hình 138.*Giao diện hoàn chỉnh*

3.3. ORACLE REPORT

Oracle Reports cho phép tạo ra rất nhiều loại báo cáo khác nhau, từ đơn giản đến phức tạp bao gồm: master/detail reports, nested matrix reports, form letters và mailing labels.

Các tính năng chính trong Report Builder bao gồm:

- ***Data model :*** dùng để tạo dữ liệu trong report và layout editors dùng để thiết kế giao diện của report
- ***Object navigator:*** giúp bạn có một cái nhìn tổng thể về dữ liệu và các đối tượng trong report theo cấu trúc hình cây và có thể drill-down
- ***Packaged functions:*** được gán với các đối tượng trong báo cáo để tính toán hoặc điều khiển việc hiển thị

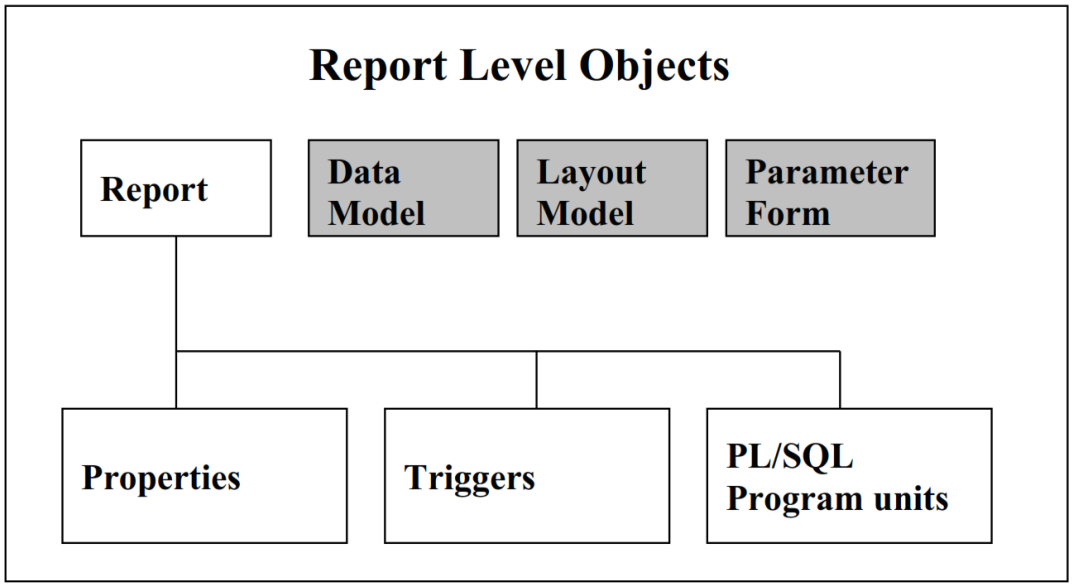
- ***Layout Editor:*** công cụ đồ hoạ cho phép tạo, tinh chỉnh giao diện báo cáo.
- ***Live Preview:*** báo cáo đầy đủ giống như khi được in ra
- ***Help:*** trợ giúp online theo đối tượng

3.3.1. Các thành phần Report Builder

❖ Có một số loại module khác nhau trong Report Builder

Kiểu Module	Mô tả
Report	Định nghia các đối tượng trong Report
Query	Định nghĩa dữ liệu lấy ra cho report
Template	Mẫu hiển thị của report được xây dựng sẵn để có thể sử dụng một cách dễ dàng.
PL/SQL Library	Thư viện độc lập chứa các chương trình con PL/SQL cho phép gọi từ report.

Bảng 18.Một số module của Report

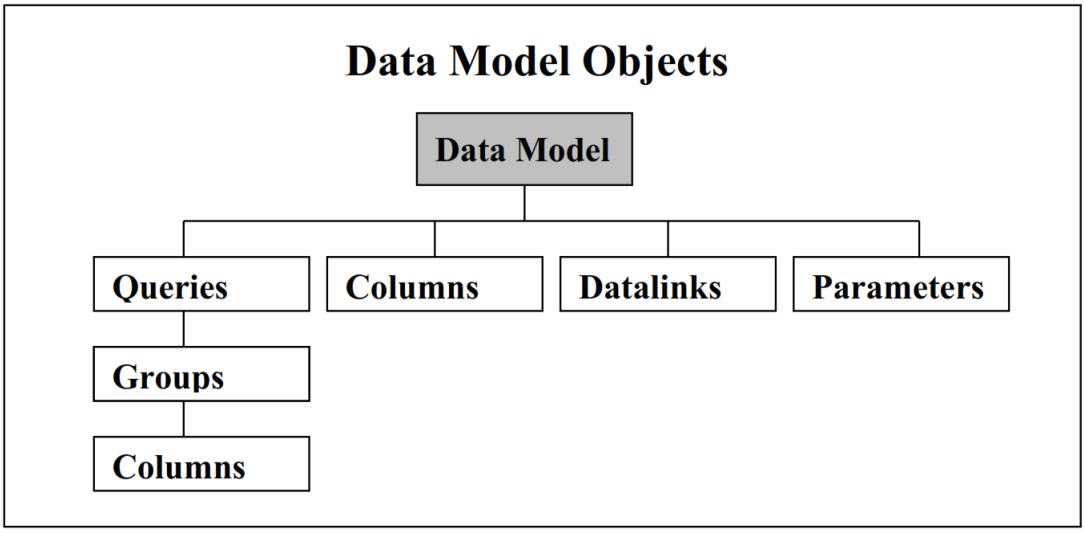


Hình 139.Các mức đối tượng của Report

Thành phần	Mô tả
Data Model	Thiết lập nên các dữ liệu cho một Report
Layout Model	Xây dựng Layout Thiết kế hiển thị cho các đối tượng

Live Previewer	Hiển thị report như dạng mà nó sẽ được in ra để có thể chỉnh sửa đơn giản các thành phần dữ liệu hiển thị.
Parameter Form	Thiết lập các tham số cần nhập vào cho report khi chạy.
Properties	Khai báo các thuộc tính của Report, ví dụ kích cỡ cho một trang in
Triggers	Các thủ tục sẽ được xử lý tại các giai đoạn khác nhau theo sự kiện khi vận hành Report
PL/SQL Program Units	Các chuong trình con PL/SQL mà có thể được gọi ra dể thực hiện.

❖ Các đối tượng trong data model:



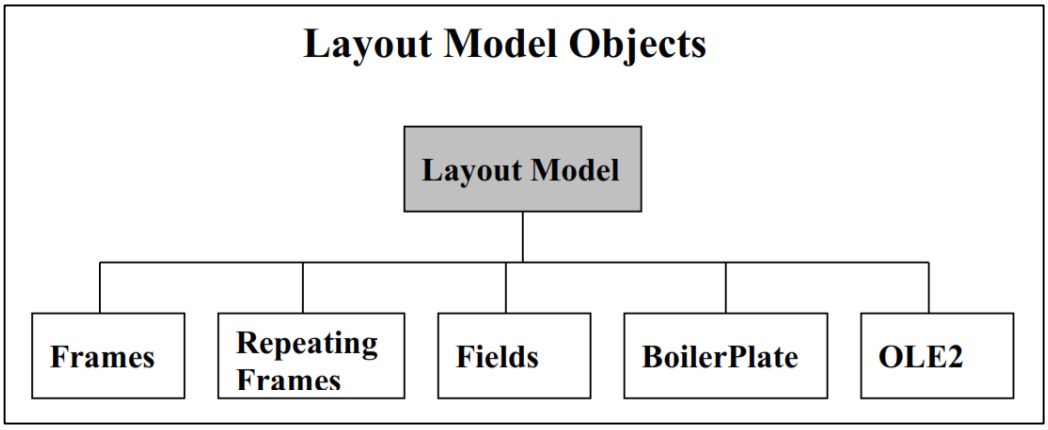
Hình 140.Các đối tượng trong Data Model

Trong đó các thành phần được mô tả như sau:

Thành phần	Mô tả
Query	Lấy dữ liệu ra cho report.
Group	Tổ chức cấu trúc dữ liệu cho Report
Columns	Chứa các giá trị để hiển thị ra kết quả hay là lưu các giá trị trung gian cho việc tính toán
Data Link	Liên kết các query theo các mối quan hệ trong cơ sở dữ liệu

Parameter	Các tham số cho phép người sử dụng report có thể nhập vào các giá trị khi report được vận hành

❖ Các đối tượng trong layout model



Hình 141.Các đối tượng trong layout model

Trong đó các thành phần được mô tả như sau:

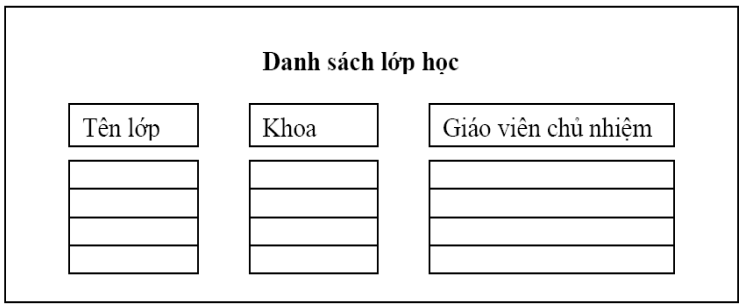
Thành phần	Mô tả
Frame	Chứa một hay một nhóm các dối tượng khác nhau
Field	Các trường chứa dữ liệu của Report
Boilerplate	Chứa text hay graphic hiển thị ở bất kỳ vị trí nào trong report
OLE2	*Nhúng các dối tượng OLE vào report*

❖ Một số kiểu Report thông thường

- Tabular
- Master-Detail
- Matrix.

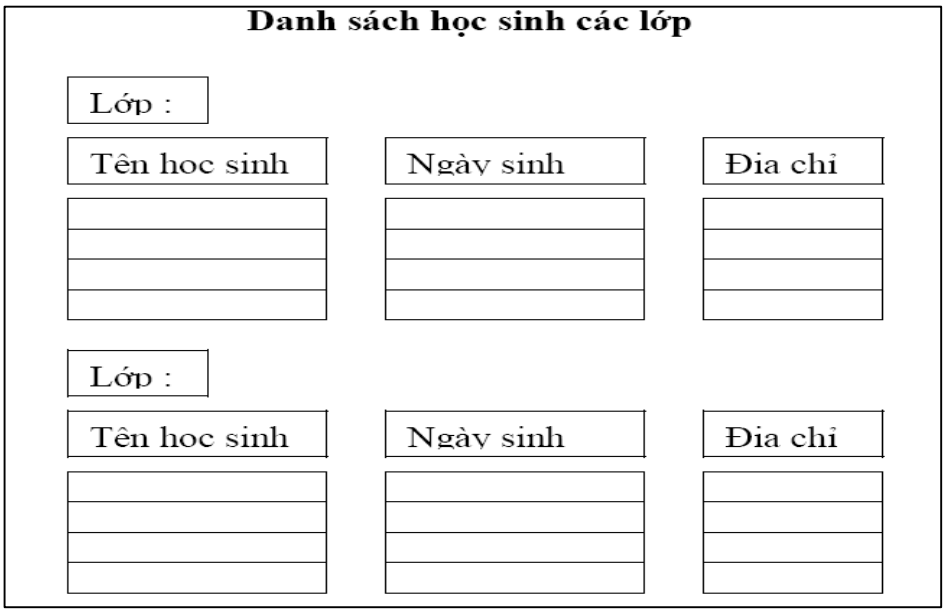
Ví dụ trong bài toán quản lý học sinh, ta có thể lập báo cáo theo các kiểu trên như sau:

+ Kiểu Tabular



Hình 142.Kiểu report tabular

+ Kiểu Master_Detail



Hình 143.Kiểu report Master_Detail

+ Kiểu Matrix: dữ liệu hiển thị dạng bảng trong đó cột và hang là các Master và nội dung hiển thị trong các ô là dữ liệu Detail.

Bảng số lượng các lớp trong khoa

Khoa Mã lớp	Toán	Lý	Sinh	Địa
K40	4	5		
K41	7			
K42				

Hình 144.Kiểu report matrix

3.3.2. Xây dựng báo cáo

3.3.2.1.Các bước tạo Oracle Report

Có 3 bước để tạo một Oracle Report mới

- Định nghĩa một report mới

Khi chạy Report Builder thì mạc định sẽ tạo cho ta một report mới. Nếu không chúng ta có thể tạo một report mới bằng cách chọn File -> New -> New Report từ menu chính của Oracle Report Builder.

Khi tạo xong một đối tượng report mới chúng ta có thể dễ dàng nhìn thấy panel đầu tiên nằm phía bên trái trong phần màn hình chính của Report Builder. Đây là một panel vô cùng quan trọng, nó thể hiện toàn bộ các đối tượng có trong report theo cấu trúc hình cây, và ta có thể di chuyển đến bất cứ đối tượng nào một cách dễ dàng. Các đối tượng trong report được nhóm theo từng nhóm riêng biệt, được tổ chức theo hình cây giúp ta dễ dàng tìm kiếm đối tượng cần thiết.

- ***Tạo data model*** gồm: chọn dữ liệu nào, mối liên hệ dữ liệu và các tính toán liên quan đến báo cáo

Data model là nơi chứa các đối tượng dữ liệu cấu trúc dữ liệu và các mối liên kết dữ liệu của report. Ta có thể tạo mới, sửa đổi, các đối tượng model trong data model editor. Các loại đối tượng có trong data model bao gồm:

- Queries: là một câu lệnh select. Chúng ta có thể lấy dữ liệu từ một hoặc nhiều bảng trên một hoặc nhiều CSDL khác nhau.

- Groups: Group xác định cấu trúc dữ liệu trong báo cáo. Oracle tự động tạo ra một group ứng với mỗi query nhưng ta có thể tạo thêm các group mới từ query đó.

Chúng ta sử dụng group kiểu cha – con để tạo ra các breack reports

- Columns: đây là nơi chứa dữ liệu của report. Cột mạc định tương ứng với các cột chứa trong câu lệnh select. Ta cũng có thể tạo ra các cột tổng, các cột công thức.

- Parameters: là các biến trong report cho phép điều khiển sự diễn thị trong runtime. Có 2 loại parameter là user parameter và system parameter. Oracle tự động tạo các system parameter tại thời điểm runtime còn user parameter là do người sử dụng tự định nghĩa.

- Data links: được dùng để tạo kết nối cha – con giữa các query và group.

- ***Tạo layout để thể hiện báo cáo***: đầu tiên dùng default layout tạo layout mạc định, rồi tu chỉnh mạc định để có layout riêng của bạn

Tạo layout cho report chính là xác định xem cái báo cáo của chúng ta trông sẽ như thế nào. Oracle report cung cấp cho chúng ta 6 layout styles mạc định bao gồm: tabular, master/detail, form letter, form, mailing label, và matrix. Bạn có thể chọn một trong các kiểu trên rồi tu chỉnh lại thành layout riêng của mình.

3.3.2.2.Report Wizard

Report Wizard là một tiện ích mà Report Builder cung cấp để thiết kế report.

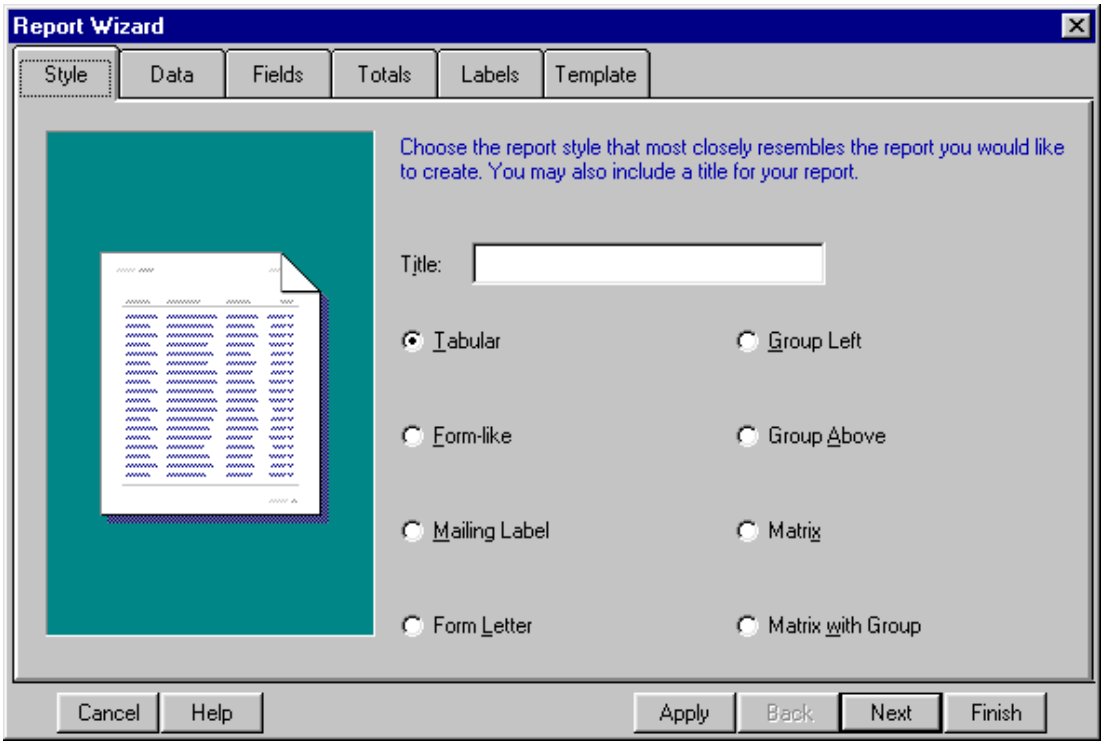
Để tạo được một Report sử dụng công cụ Report Wizard cần xây dựng theo thứ tự các bước như sau:

1. Kiểu của Report (Style)
2. Dữ liệu (Data)
3. Các trường giá trị và các trường hiển thị (Field)
4. Total (Các trường tính toán)
5. Labels (Các nhãn hiển thị đối với mỗi trường)
6. Template (Các mẫu Template có sẵn được cung cấp bởi Report Builder

❖ Tạo Report Tabular

Report được tạo ra với kiểu tabular là phổ biến nó có dạng hiển thị như một danh sách với các dòng và các cột. Sau đây là các bước chung cơ bản tạo ra một Report kiểu Tabular.

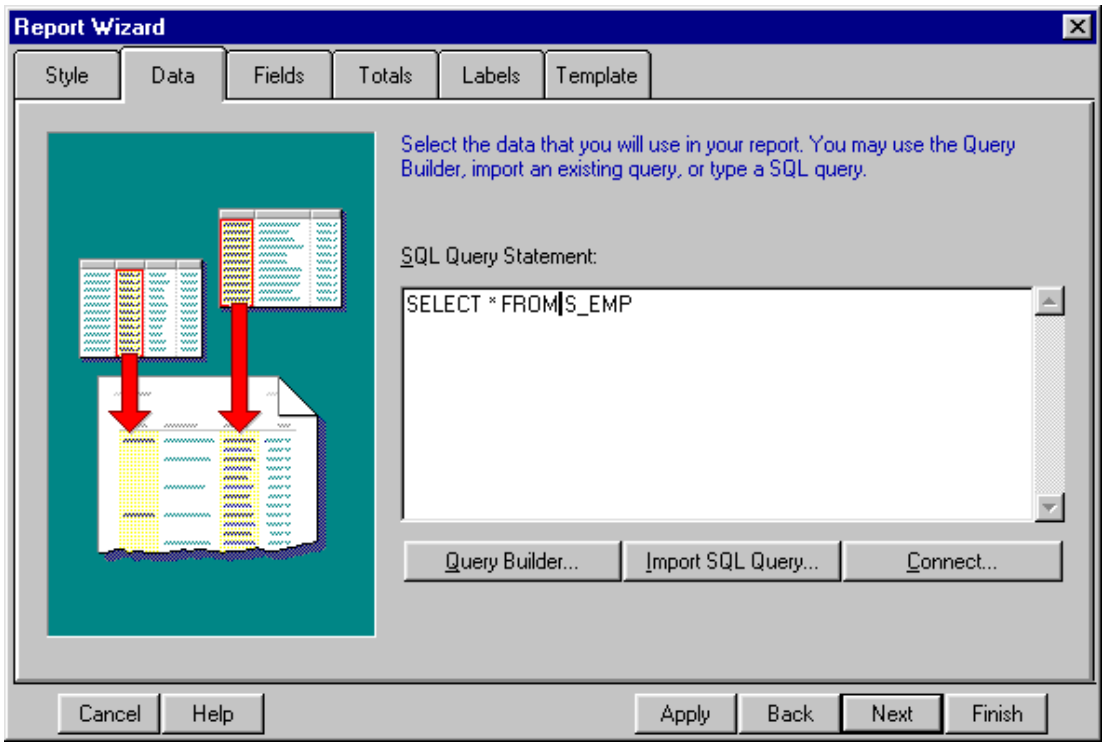
Kiểu của Report (Style)



Hình 145.*Kiểu của report*

Dữ liệu của Report

Trong phần này nhập vào câu lệnh select để xây dựng query lấy dữ liệu cho report.



Hình 146.*Dữ liệu của report*

Các trường giá trị và các trường hiển thị: Cho phép chọn các trường sẽ hiển thị trong Report. Trong đó :

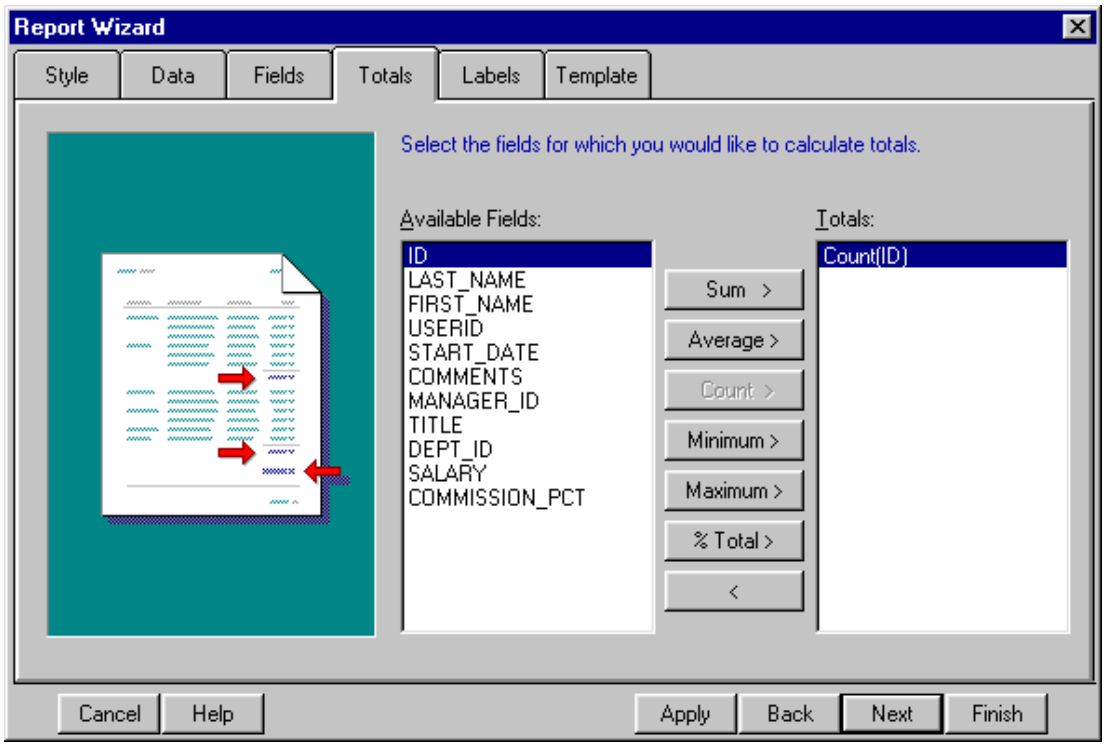
- Danh sách Available Fields: Là danh sách các trường đã được lựa chọn từ câu lệnh query trước đó

- Danh sách Displayed Fields: Là danh sách các trường sẽ hiển thị ra layout của report

Các trường tính toán (Count, Sum, ...)

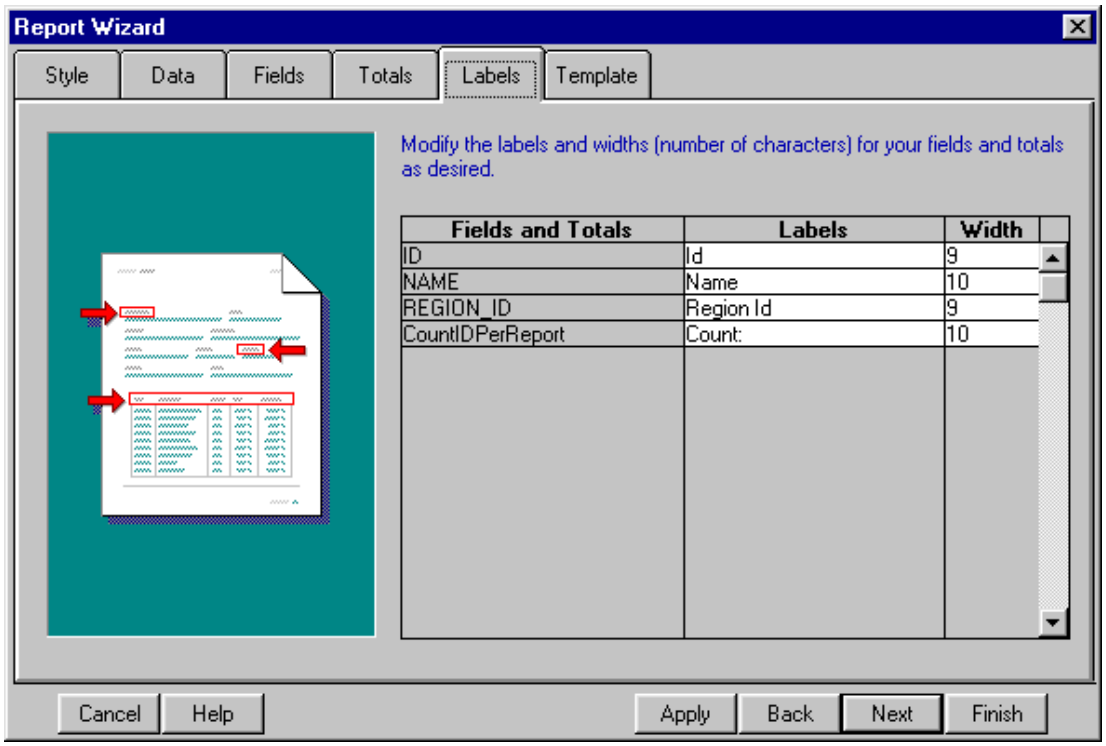
Trong đó :

Sẽ sử dụng các hàm nhóm (group function) đã liệt kê sẵn (count, sum, average ...) để tạo ra các trường tính toán (tổng, thứ tự) dựa trên các trường dữ liệu đã query sẵn(Available Fields).



Hình 147.*Tính toán trong report*

Khai báo thuộc tính cho các trường dữ liệu hiển thị

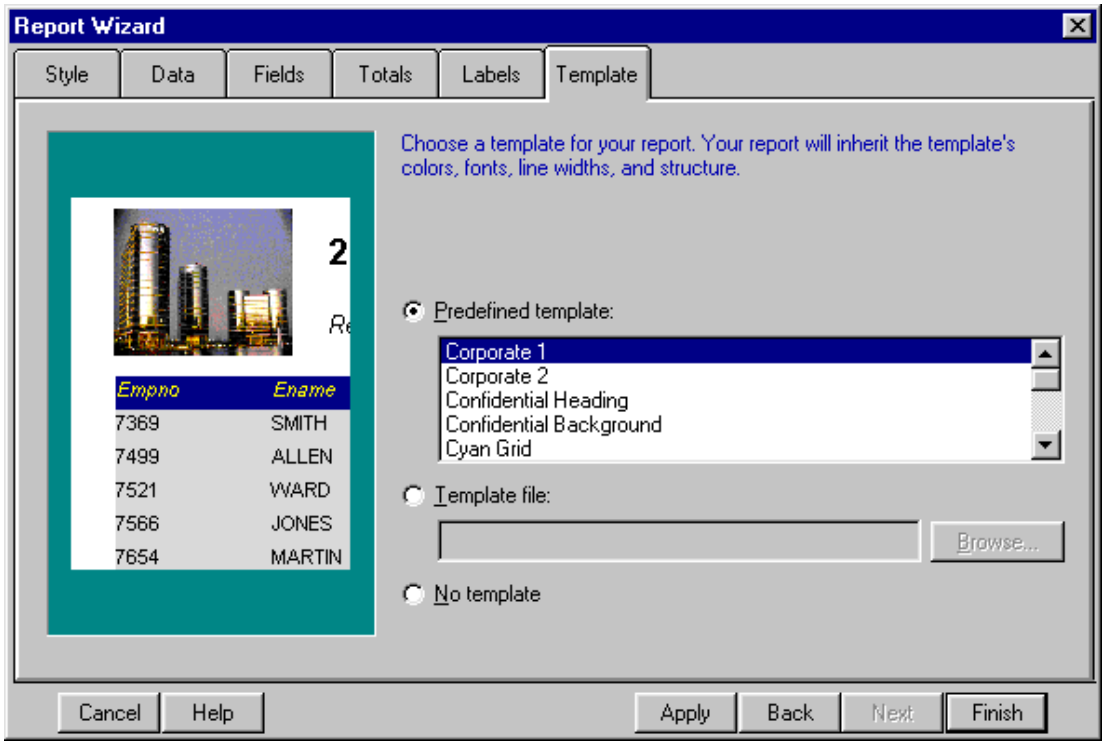


Hình 148.Các thuộc tính hiện thị

Trong đó:

- **Fields and totals**: Các trường và các cột sẽ hiển thị
- **Label:** Là nhãn của các trường hiển thị
- **Width**: Là độ rộng sẽ hiển thị của từng trường

Chọn mẫu hiển thị:



Hình 149.*Mẫu report*

Các mẫu hiển thị được cung cấp sẵn bởi Report Builder, có thể lựa chọn các cách hiển thị một báo cáo :

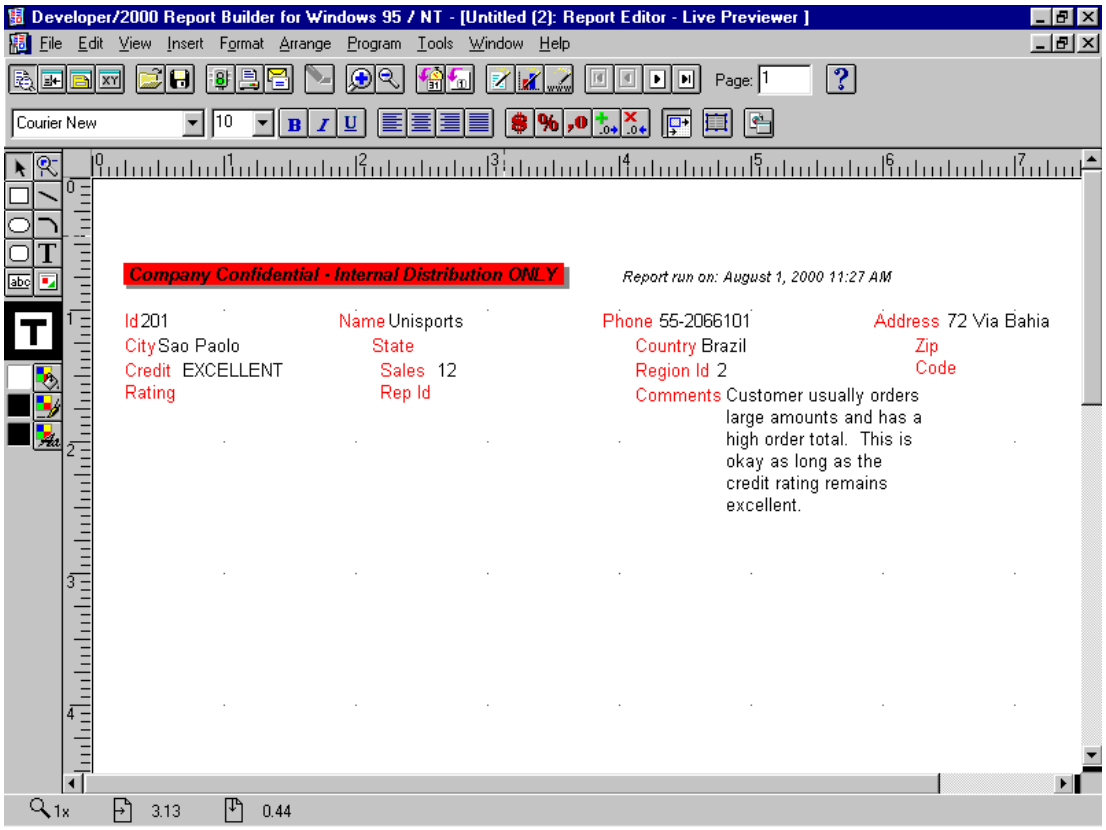
- **Predefined Template:** Để sử dụng các template đã được Report Builder xây dựng trước.
- **Template File**: Các template do người thiết kế đã tạo ra từ trước
- **No Template**: Không sử dụng theo mẫu tạo trước nào

❖ **Tạo report kiểu Form_Like**

Report kiểu Form_Like có 3 đặc trưng cơ bản khác với Report kiểu Tabular

- Các nhãn được đặt vị trí phía trái của mỗi trường và có màu khác
- Các trường được sắp xếp ngang ra theo trang giấy
- Mỗi bản ghi được đặt trên một trang của Report

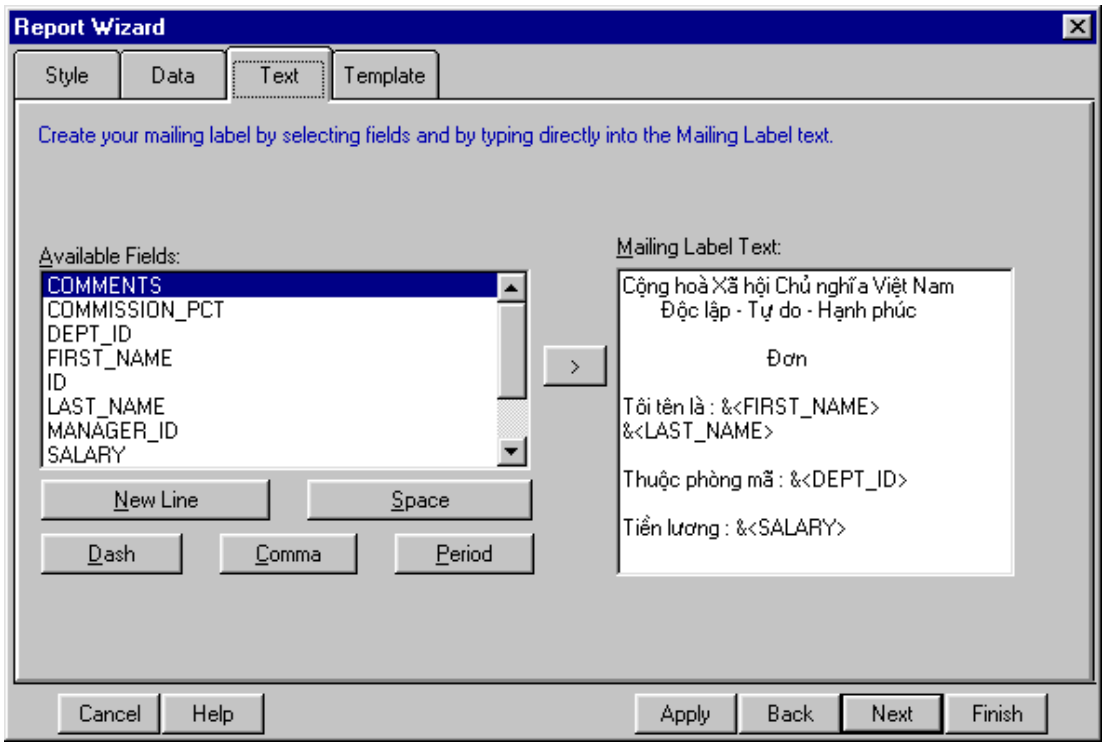
Report kiểu Form_like có dạng như sau:



Hình 150.*Report kiểu Form_Like*

❖ Tạo report kiểu Mailing Label và Form Letter

Viết nội dung báo cáo theo kiểu Mail, Form có sử dụng các trường dữ liệu đã được query.



Hình 151.*Tạo report kiểu Mailing Label và Form Letter*

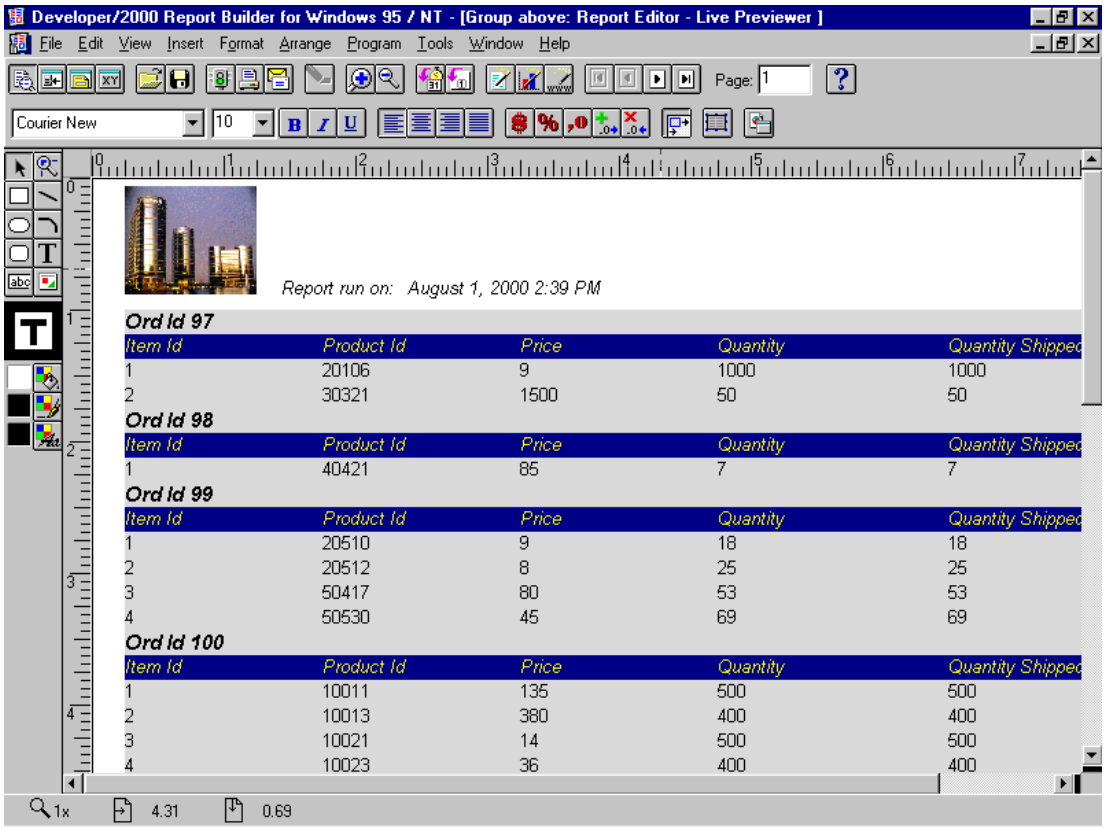
Trong đó các thành phần được mô tả cụ thể như sau:

Thành phần	**Mô tả**
Available field	Chứa các trường dữ liệu sinh ra từ câu lệng query
Mailing label text	Nội dung báo cáo
New line	Tạo một dòng mới cho báo cáo
Space	Tạo một dấu space
Dash	Tạo một dấu – (gạch ngang)
Comma	Tạo một dấu ‘ (phẩy)
Period	Tạo một dấu . (chấm)

Ghi chú: Report kiểu Form và Mailing chỉ khác nhau duy nhất ở một điểm đó là khi vận hành Report với kiểu mailing thì cho phép hiển thị nhiều Record trên một trang, còn Report Form Letter hiển thị mỗi bản ghi trên một trang khác nhau.

3.3.2.3.Live Previewer

Các thành phần trên live previewer:



Hình 152.*Live Previewer*

Trên Live previewer có 4 thành phần hoạt động:

- Toolbar: Chứa các nút điều khiển để chuyển trạng thái hoạt động của Report
- Stylebar: Chứa các công cụ định dạng kiểu dáng cho một đối tượng trên Report
- Tool palette: Chứa các công cụ làm nên một đối tượng trong Report
- Status bar: Hiển thị trạng thái (toạ độ) của vị trí chuột hay các đối tượng có lựa chọn.

Sắp xếp cho thẳng hàng các cột (align column)

Các bước để sắp xếp

- Chọn các cột muốn sắp xếp thẳng hàng
- Chọn biểu tượng "End justify" trên Stylebar

Đặt format mask

Có nhiều cách để đặt Format mask cho một đối tượng.

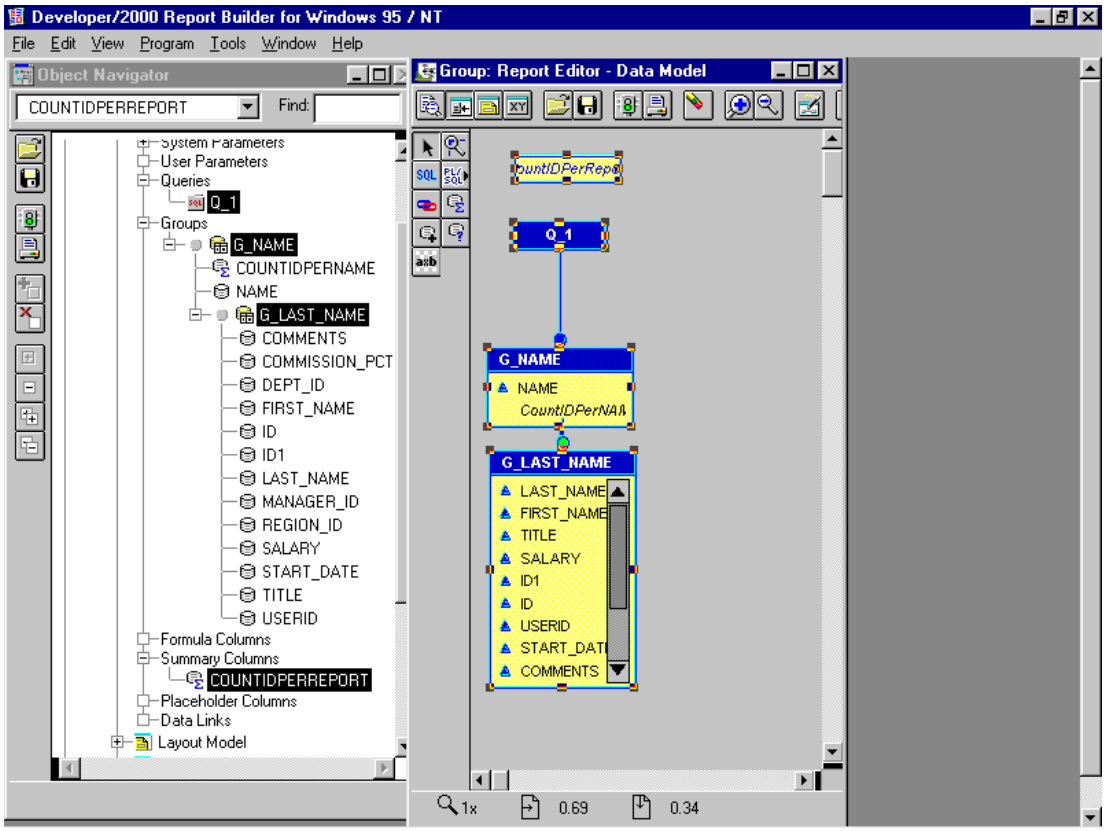
- Có thể đặt trong thuộc tính của từng đối tượng
- Dùng các công cụ trên thanh Stylebar

Với mỗi đối tượng có kiểu định dạng khác nhau giá trị Format mask là khác nhau.

Chèn số trang, thời gian vào report

Chọn Insert -> Date and Time.. hoặc -> Page number

3.3.2.4.Data Model



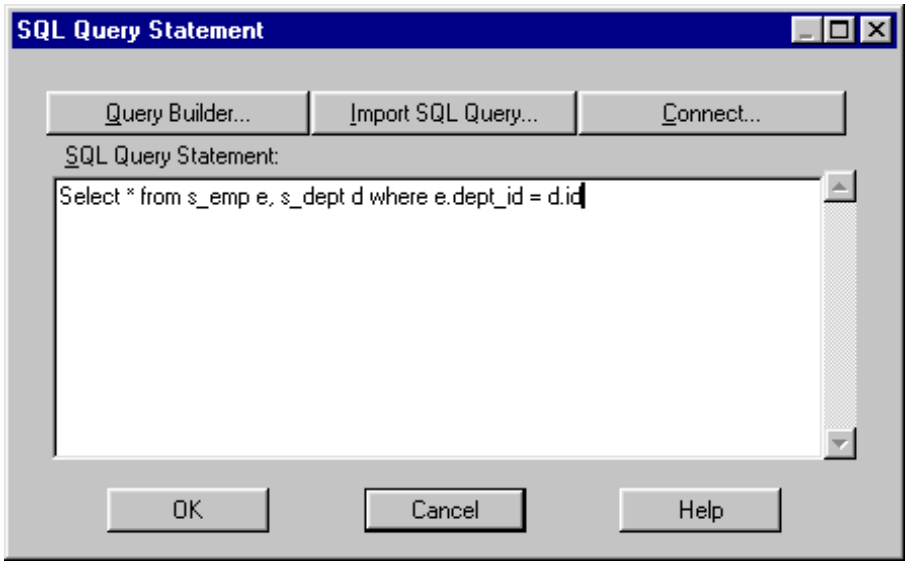
Hình 153.*Data model*

Các đối tượng trong data model

Đối tượng	Mô tả
Query	Bạn có thể tạo một Report với nhiều query
Group	Mỗi group xác định bởi một Query
Column	Mỗi Column thuộc một Group nào đó. Ngầm định là một Group chứa các cột được sinh ra từ Query
Link	Liên kết giữa nhóm master và nhóm detail. Bạn có thể tạo một liên kết giữa các nhóm từ các query khác nhau. Các liên kết không bao giờ được tự động tạo
Parameter	Tạo ra các tham số phục vụ trong quá trình sử dụng form theo mục đích của người thiết kế Report

Query

Query là câu lệnh select để lấy ra dữ liệu cho report. Hộp thoại Query có dạng như sau:



Hình 154.Query cho report

Một report có thể bao gồm nhiều query.

- Người thiết kế có thể thay đổi query, đổi tên query, bổ sung thêm các dòng chú thích (/* */ cho nhiều dòng và -- cho 1 dòng)
- Có thể hạn chế số record mà query lấy ra (fetch) từ cơ sở dữ liệu.
- Khi tạo 1 query thì ít nhất sẽ có 1 group tự động được sinh ra để mô tả dữ liệu trong câu lệnh query đó. Việc sắp xếp group (hay các group) này sẽ quyết định cấu trúc của report.

Group

Group quyết định cấu trúc dữ liệu, tần suất dữ liệu của report.

- Một group sẽ biểu diễn thông tin dữ liệu được lấy ra từ 1 query.
- Trong group có thể chứa nhiều column, ít nhất là một colume
- Các group được chia nhỏ (break) hay lồng vào nhau (matrix) để tạo nên những report có cấu trúc phức tạp theo yêu cầu.
- Người thiết kế có thể thay đổi tên của group, hạn chế số record của group.
- Có thể chia group bằng cách kéo 1 column của group sang trái, lên trên để tạo group cấp trên hay sang phải, xuống dưới để tạo group cấp dưới.

Một report kiểu group biểu diễn thông tin về phòng và nhân viên trong phòng sẽ bao gồm 2 group ở 2 cấp. Group ở trên chứa thông tin về phòng (s_dept) còn group ở dưới chứa thông tin về nhân viên (s_emp). Cấu trúc của report sẽ là dạng phân cấp, biểu diễn thông tin về phòng và với mỗi phòng lại đi vào cấp ở dưới biểu diễn thông

tin về các nhân viên trong phòng đó.

Data Link

Xác định 1 mối quan hệ Master/Detail (cha/con) giữa một group và một query dựa trên quan hệ primary key và foreign key. Khi đó query con sẽ thực hiện query dữ liệu mỗi khi một record của group cha được đưa ra (fetch) theo điều kiện đã xác định trong data link liên kết giữa chúng.

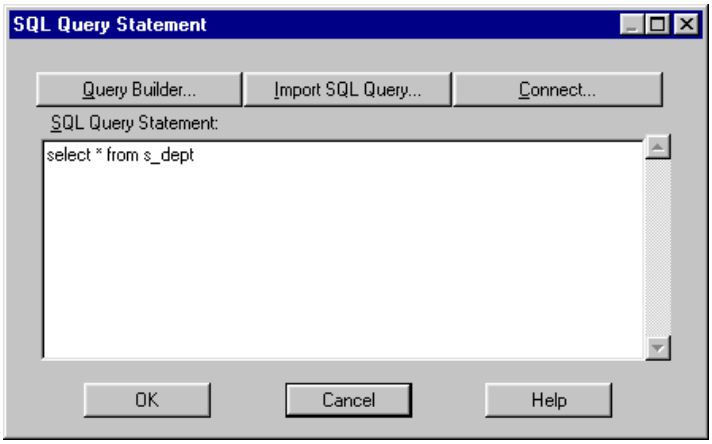
Cách tạo một Data link

- Trong Data Model chọn chuột vào biểu tượng data link
- Đặt chuột và kéo liên kết giữa 2 query

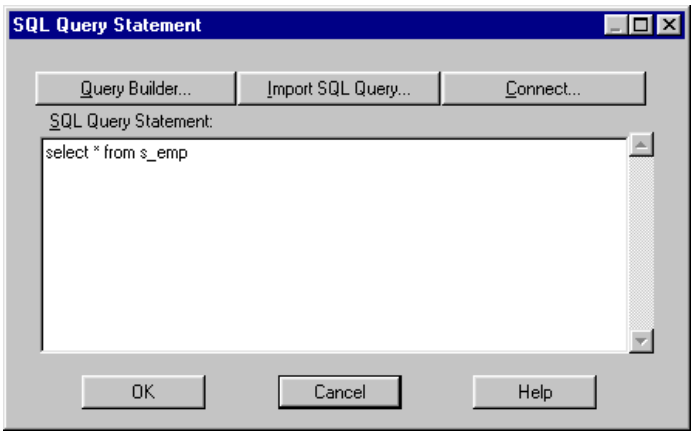
Ví dụ về thiết lập một data link:

Một report kiểu group biểu diễn thông tin về phòng và nhân viên trong phòng sẽ bao gồm 2 group ở 2 cấp. Group ở trên chứa thông tin về phòng (s_dept) còn group ở dưới chứa thông tin về nhân viên (s_emp). Cấu trúc của report sẽ là dạng phân cấp, biểu diễn thông tin về phòng và với mỗi phòng lại đi vào cấp ở dưới biểu diễn thông tin về các nhân viên trong phòng đó. Sử dụng data link để phân cấp dữ liệu.

Khi đó report này sẽ bao gồm 2 query, một query lấy dữ liệu về phòng ban (s_dept) và một query lấy dữ liệu về nhân viên (s_emp):

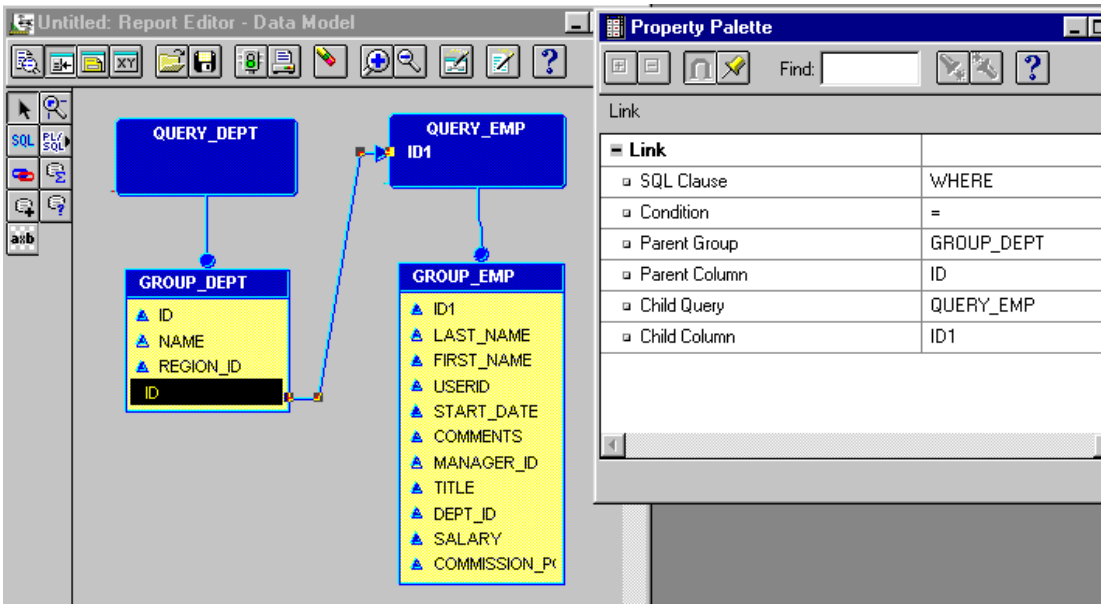


Hình 155.*Query về phòng ban*



Hình 156. *Query về nhân viên*

Sau đó tạo 1 datalink



Hình 157.*Tạo data link*

Data link này sẽ tạo ra mối liên hệ giữa column ID của group GROUP_DEPT và column DEPT_ID của group GROUP_EMP.

Các thuộc tính của data link sẽ bao gồm

Thuộc tính	Mô tả
SQL	clause Mệnh dề diều kiện WHERE, HAVING, START WITH.
Condition	Toán tử diều kiện =, >, <, like, ...
Parent Group	Group (Master) cha trong mối quan hệ tạo ra.
Parent Column	Column quan hệ trong Group cha.
Child Query	Query con trong mối quan hệ.
Child Column	Column quan hệ trong Query con.

TÀI LIỆU THAM KHẢO

1. Sara Woodhull, Mildred Wang, 11/2007, *Oracle Applications Developer's Guide*, Oracle.
2. Sheila Moore, 7/2014, *Oracle Database 2 Day Developer's Guide*, Oracle.
3. Fred Bethke, Joan Carter, 1/2000, *Form Builder Reference*, Oracle.
4. Nguyễn Quảng Ninh, Nguyễn Nam Thuận, 2009, *Giáo trình hướng dẫn lý thuyết kèm theo bài tập thực hành oracle 11g*, NXB Hồng Đức.

MỤC LỤC