BÀI GIẢNG MÔN ORACLE

Trường Đại Học Kỹ Thuật Công Nghệ
Khoa Công Nghệ Thông Tin
Bộ môn: Hệ Thống Thông Tin
Giáo viên: Trần Hồ Lệ Phương Đan

Nội dung

- **Phần 1: Tổng quan về Oracle,**
- **Phần 2: Ngôn ngữ SQL**
- **Phần 3: Ngôn ngữ PL/SQL**
- **Phần 4: Quản trị Oracle**

Phần 1: Tổng quan về Oracle

- Kiến thức: cung cấp cho sinh viên kiến thức tổng quát về cấu trúc hoạt động của Oracle Server, các đặc điểm, điểm mạnh của hệ quản trị Oracle

Phần 1: Cơ Bản Về Oracle
Giới thiệu Oracle

- 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 cơ sở dữ liệu bên dưới.
- Là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mang tính mềm dẻo, linh động, thích ứng cao với các quy mô xử lý giao dịch, an toàn hệ thống. Cung cấp các công cụ xây dựng và quản lý cơ sở dữ liệu.
- Tích hợp Web: kết nối ứng dụng với công nghệ Web được tích hợp trong Oracle WebServer.

Phần 1: Cơ Bản Về Oracle
Giới thiệu Oracle

- Do Relational Software Inc phát triển năm 1977
- Oracle v1: 1978, Oracle v2: 1980, Oracle v3 released: 1982, Oracle v4: 1984, Oracle v5: 1986,
- 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).

Phần 1: Cơ Bản Về Oracle
Ưu điểm của Oracle

- Tính bảo mật cao
- Tính an toàn dữ liệu cao
- Cơ chế quyền hạn rõ ràng, ổn định
- Giá rẻ hơn MSSQLServer
- 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.
- Có thể cài đặt trên nhiều hệ điều hành khác như Solaris, Linux, …

Phần 1: Cơ Bản Về Oracle
Các điểm mới của Oracle

- Cho phép định nghĩa lại cấu trúc của tables đang online
- Tạm treo database
- Đặt chế độ hoạt động tĩnh cho database
- Khả năng khôi phục và cấp phát lại không gian
- Tự động quản lý vùng không gian
- Quản lý động vùng nhớ SGA(System Global Area)

Oracle Server là gì?

Oracle Server:

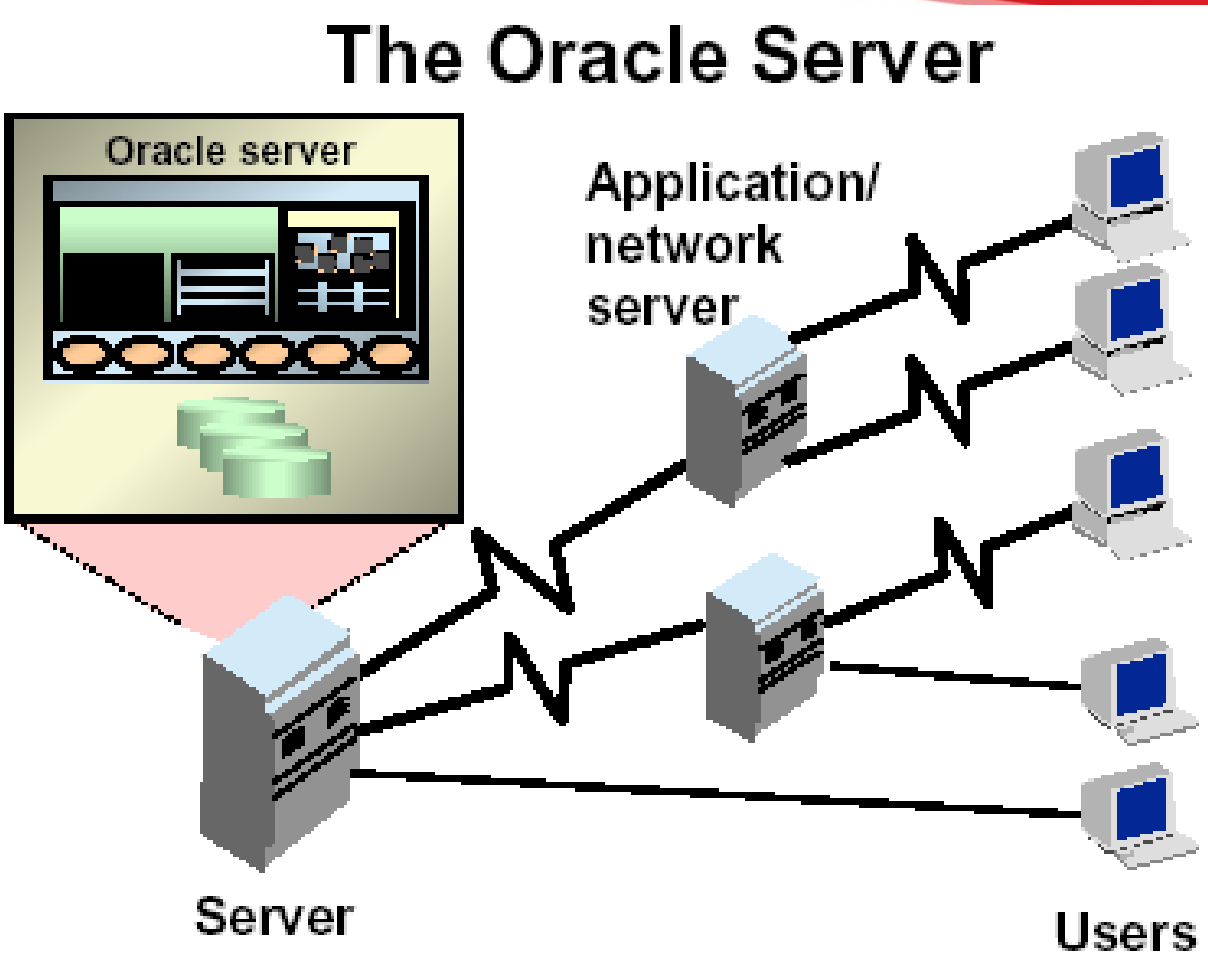
- Là hệ thống quản trị cơ sở dữ liệu đối tượng- quan hệ, tập hợp các file, tiến trình (processes) và cấu trúc bộ nhớ trong Oracle Server.

Kết nối tới Oracle Server

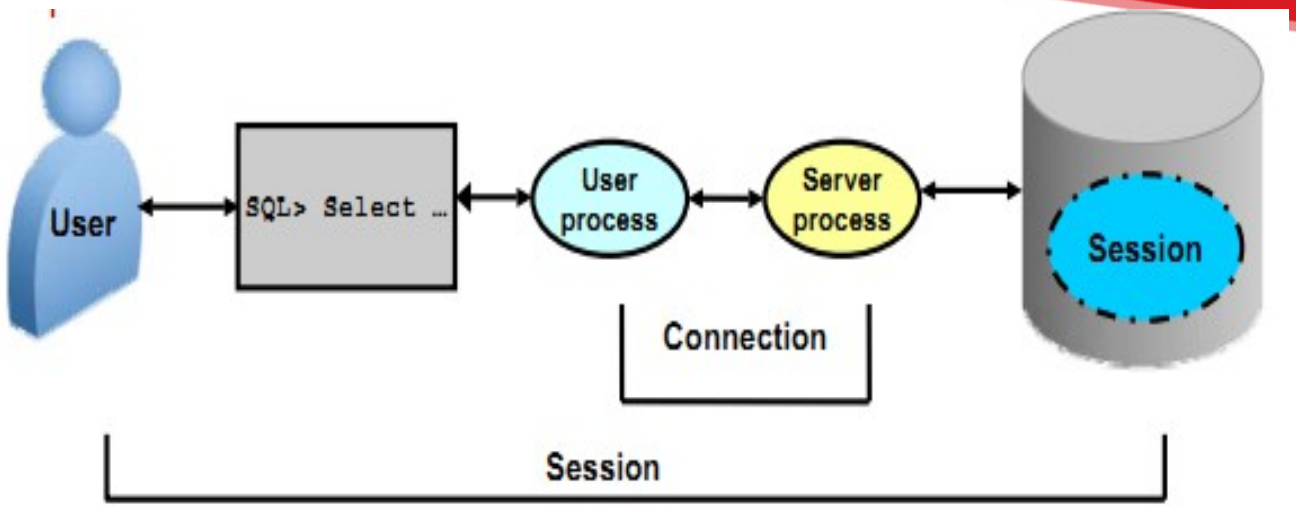
3 cách kết nối tới Oracle Server:

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

Kết nối tới Oracle Server



Kết nối tới Oracle Server



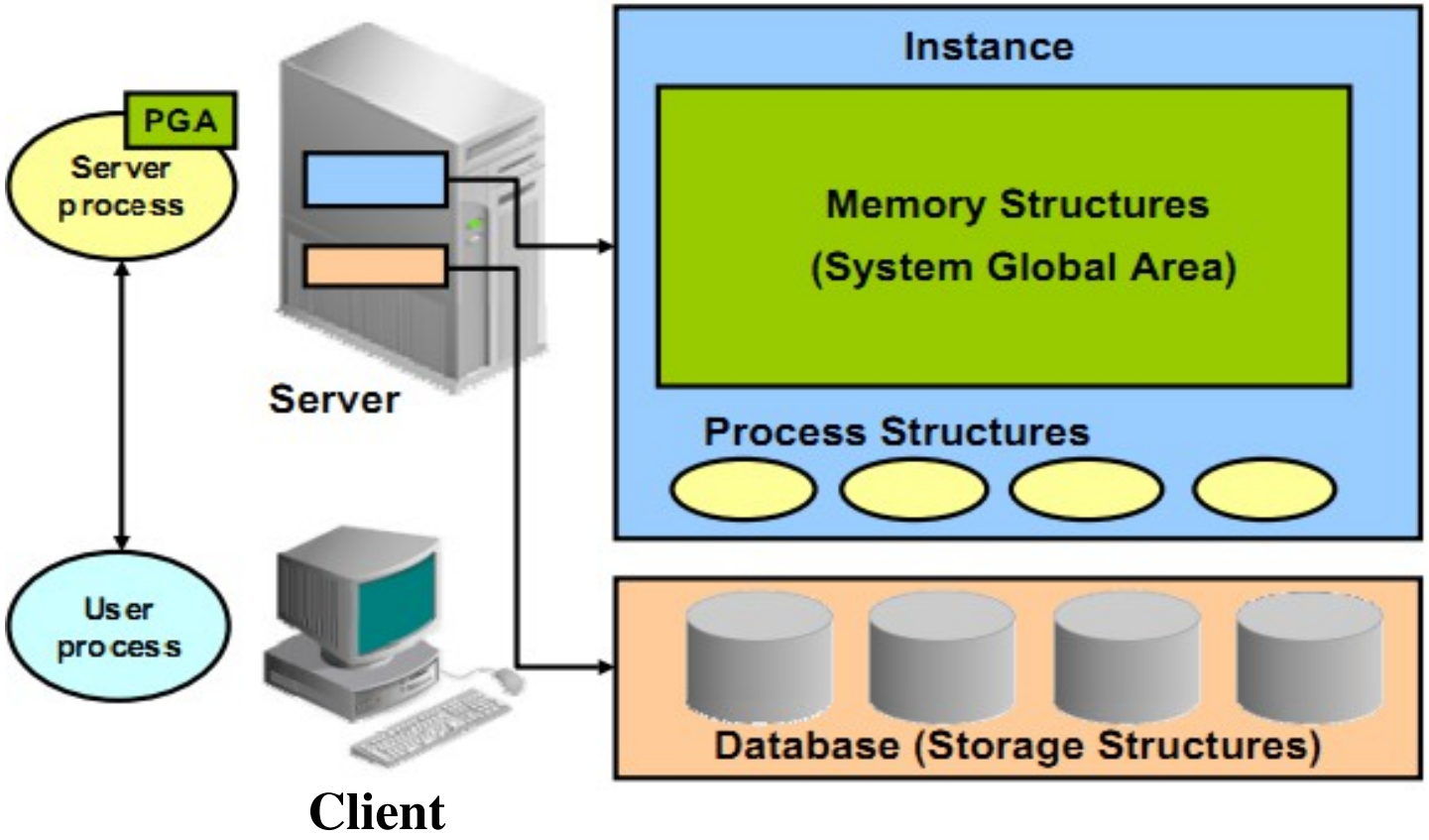
User không thể thao tác trực tiếp trên cơ sở dữ liệu được, mà User sẽ tạo ra các yêu cầu (gọi là User process), các yêu cầu này sẽ được gửi tới Server và Server sẽ thực hiện các yêu cầu này (Server Process) để tác động lên cơ sở dữ liệu.

Kết nối tới Oracle Server

Connection: Là quá trình giao tiếp giữa một **User Process** và một **Instance**.

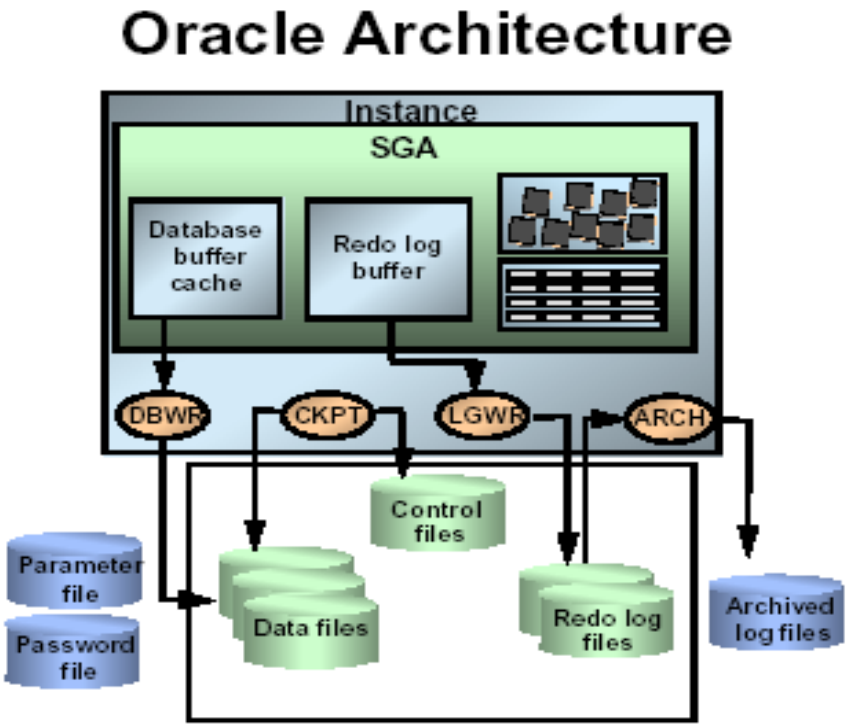
Session: Là một kết nối cụ thể từ một User tới một Instance thông qua User Process. Ví dụ khi một User sử dụng SQL*Plus đăng nhập vào Oracle Server, nếu quá trình đăng nhập thành công, thì một **Session** (phiên làm việc) sẽ được thiết lập từ đây. Session sẽ tồn tại cho tới khi User ngắt kết nối khỏi hệ thống.

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle



Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

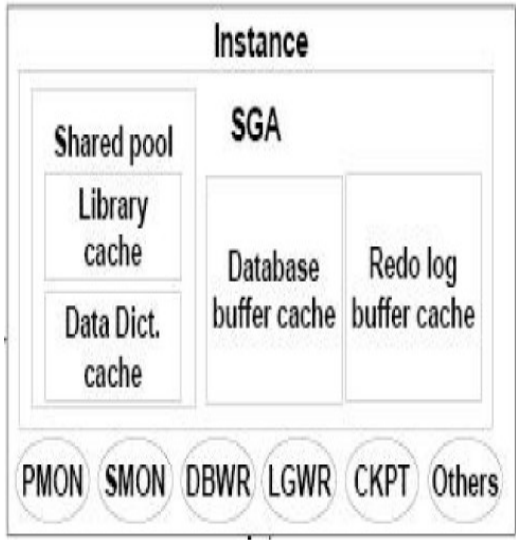
- Bao gồm 2 thành phần chính
  - Oracle Instance
  - Oracle Database



Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

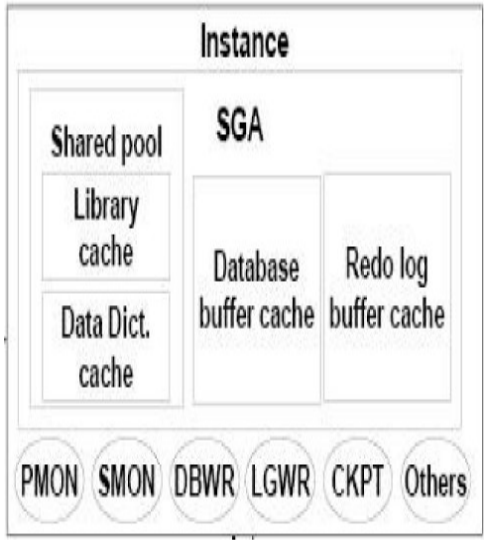
- Oracle Instance:
  - Dùng để quản trị cơ sở dữ liệu
  - Được xác định qua tham số môi trường Oracle_SID
  - Bao gồm một cấu trúc bộ nhớ System Global Area (SGA) và các background process (tiến trình nền)

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle



- SGA: là vùng bộ nhớ chia sẻ, dùng để lưu trữ dữ liệu và các thông tin điều khiển của Oracle Server
- SGA bao gồm các vùng bộ nhớ chính:
  - Shared pool: Là một phần của SGA lưu các cấu trúc bộ nhớ chia sẻ.
  - Database buffer cache: Lưu trữ các dữ liệu được sử dụng gần nhất.
  - Redo log buffer: Được sử dụng cho việc dò tìm lại các thay đổi trong cơ sở dữ liệu và được thực hiện bởi các background process.

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle



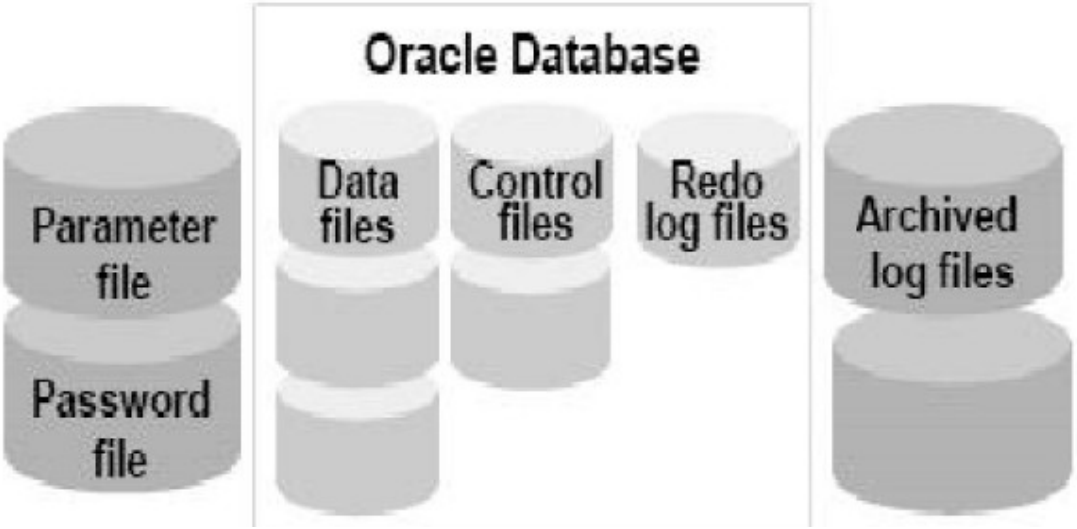
- SGA bao gồm các vùng bộ nhớ chính:
  - Shared pool
    - Library Cache: lưu trữ thông tin về các câu lệnh SQL gần nhất, giúp nâng cao hiệu suất thực hiện lệnh
    - Data Dictionnary Cache: lưu trữ thông tin dictionnary cache được sử dụng gần nhất như định nghĩa các bảng, các cột, usernames, passwords và các privileges
- Bacground process: điều khiển vào ra, cung cấp cơ chế xử lý song song nâng cao hiệu quả và độ tin cậy

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

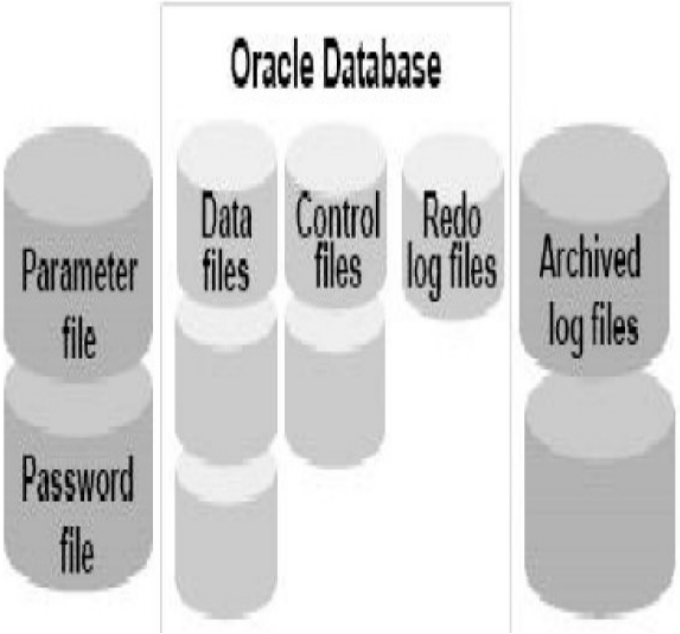
- Oracle Database:
  - Tập hợp file hệ thống, file dữ liệu, lưu trữ và trả về các thông tin liên quan
  - Được lưu trữ dưới hai cấu trúc: vật lý và luận lý

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

- Oracle Database:
  - Cấu trúc vật lý: database là tập hợp các control file, online redo log file, và các datafile



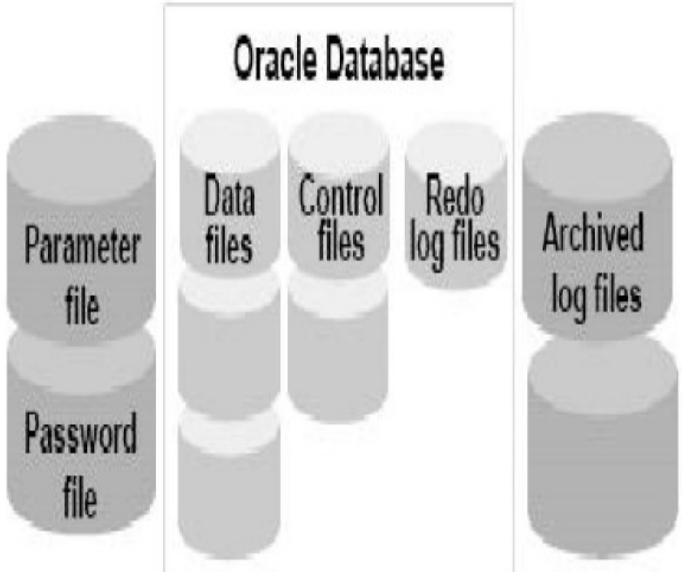
Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle



Cấu trúc vật lý của Oracle Database

- Datafiles: chứa toàn bộ dữ liệu trong database, có thể tự động mở rộng kích thước mỗi khi database hết chỗ lưu trữ dữ liệu. Một hay nhiều datafiles tạo nên một đơn vị logic của database gọi là tablespace

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

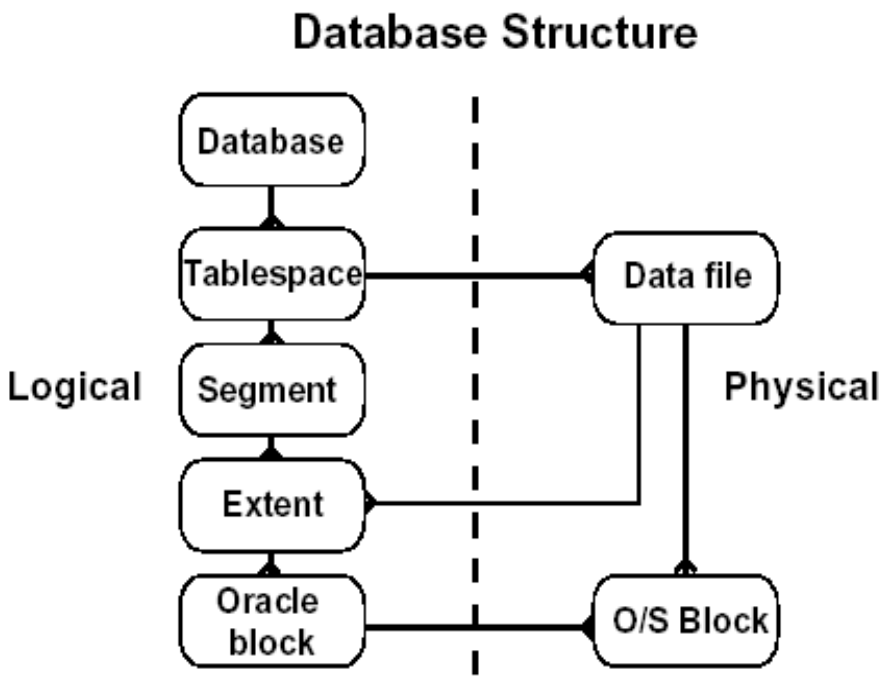


Cấu trúc vật lý của Oracle Database

- Control files: chứa các mục thông tin quy định cấu trúc vật lý của database như tên database, tên và nơi lưu trữ các datafiles hay redo log files, time stamp tạo lập database
- Redo log files: ghi lại tất cả các thay đổi dữ liệu trong database. Bảo vệ database khỏi những hỏng hóc do sự cố

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

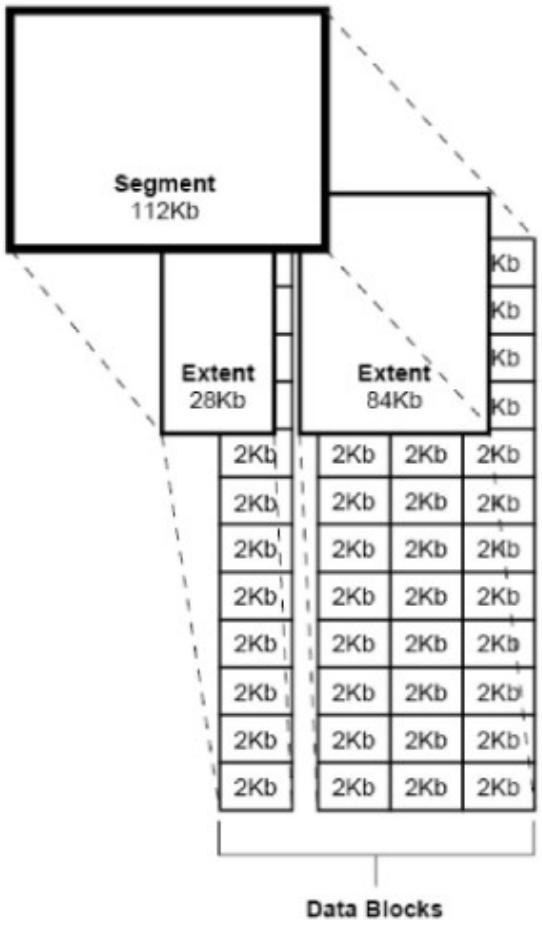
- Cấu trúc luận lý: gồm các đối tượng tablespaces, schema objects, data blocks, extents, và segments



Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

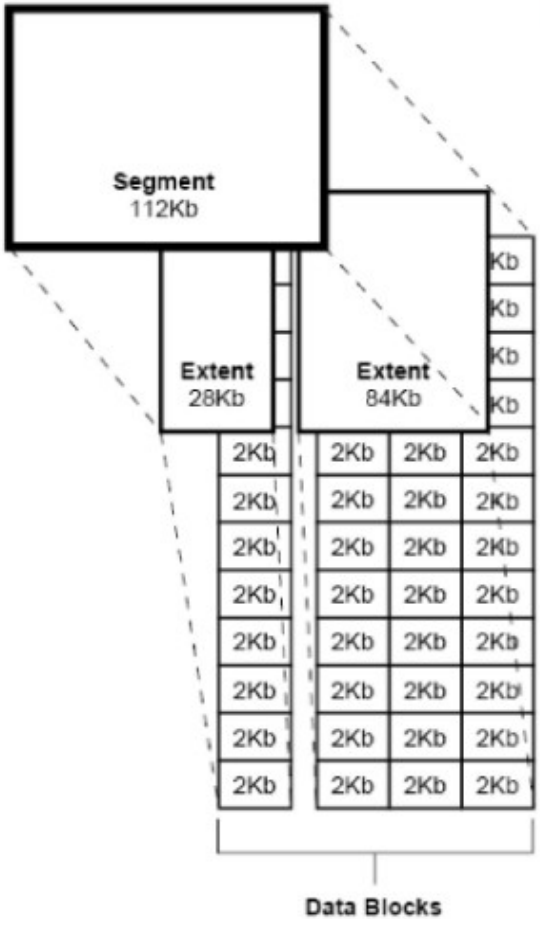
- Tablespaces: tablespaces lưu trữ toàn bộ cơ sở dữ liệu về mặt logic. Một database được lưu trữ trong một hoặc nhiều đơn vị lưu trữ logic tablespace và mỗi tablespace có thể được tạo nên, về mặt vật lý, bởi một hay nhiều datafiles
- Schema là tập hợp các đối tượng (objects) có trong database. Schema objects là các cấu trúc logic cho phép tham chiếu trực tiếp tới dữ liệu trong database. Schema objects bao gồm các cấu trúc như tables, views, sequences, stored procedures, synonyms, indexes, clusters, và database links

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle



- Cấu trúc luận lý:
  - Data Blocks, Extents, and Segments: Oracle điểu khiển không gian lưu trữ trên đĩa cứng theo các cấu trúc logic bao gồm các data blocks, extents, và segments
  - Oracle Data Blocks: Là mức phân cấp logic thấp nhất, các dữ liệu của Oracle database được lưu trữ trong các data blocks

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

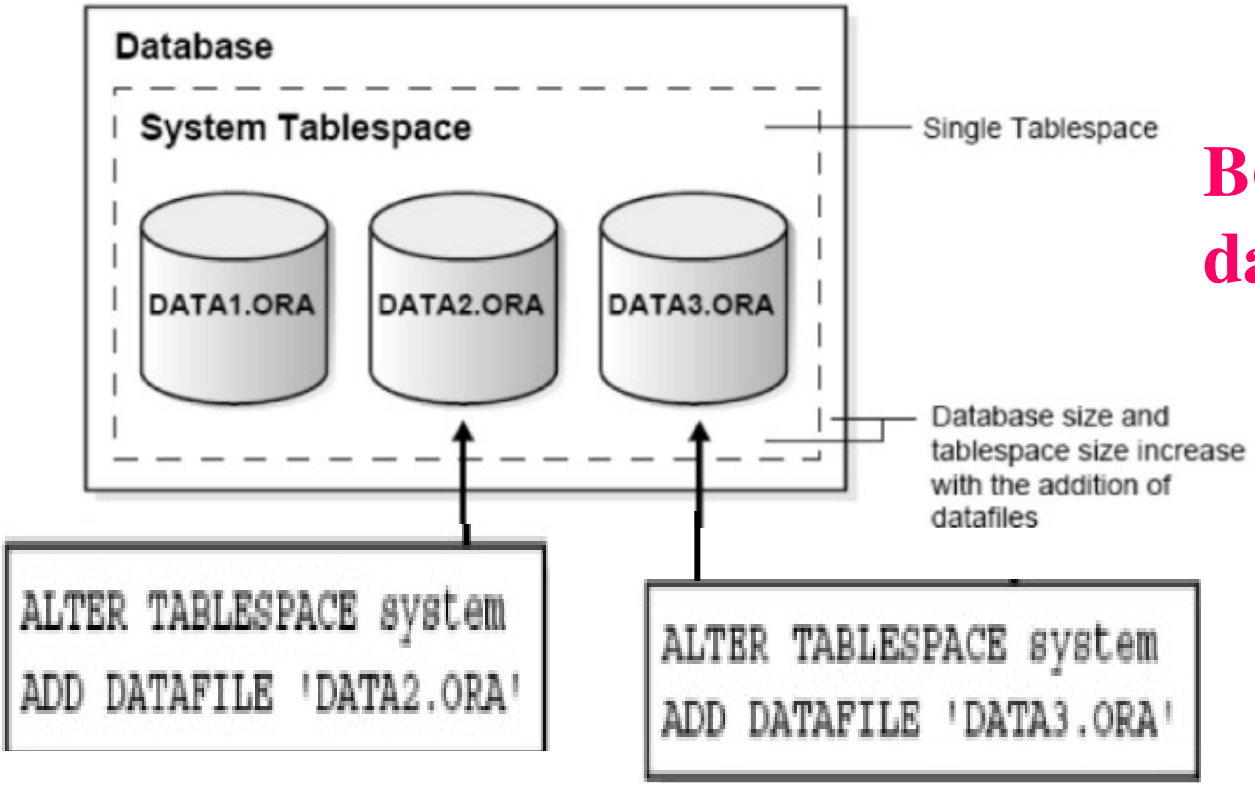


Cấu trúc luận lý:

- Extents: Là mức phân chia cao hơn về mặt logic các vùng không gian trong database. Một extent bao gồm một số data blocks liên tiếp nhau, cùng được lưu trữ tại một thiết bị lưu giữ. Extent được sử dụng để lưu trữ các thông tin có cùng kiểu
- Segments: Là mức phân chia cao hơn nữa về mặt logic các vùng không gian trong database. Một segment là một tập hợp các extents được cấp phát cho một cấu trúc logic

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

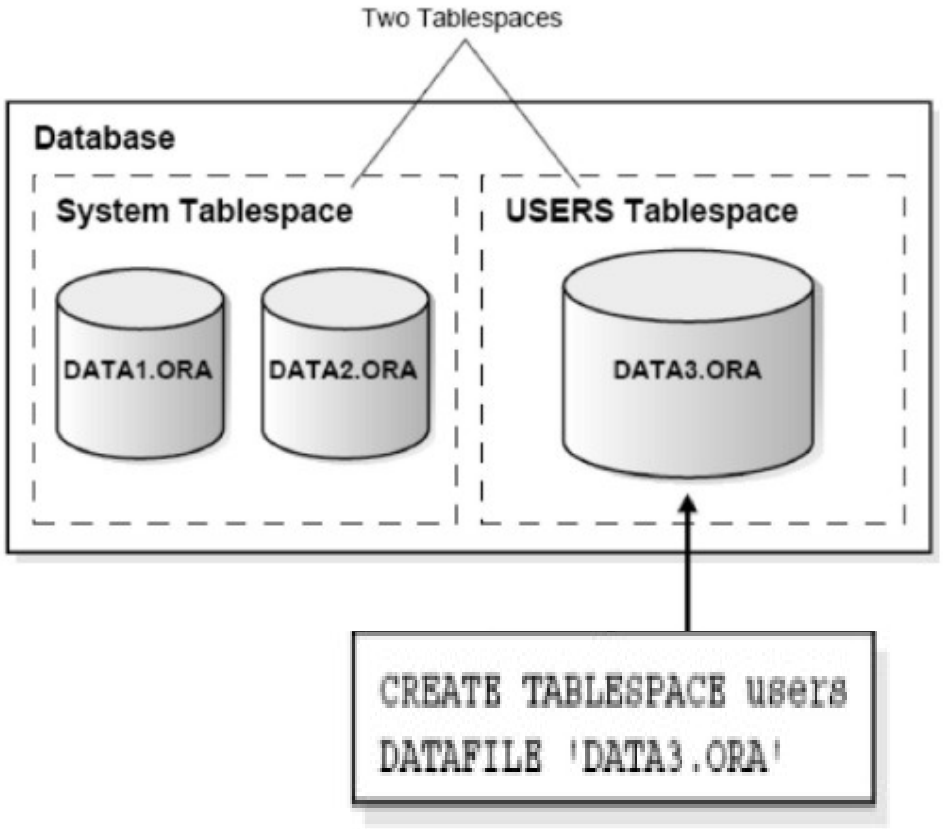
- Cấp phát thêm vùng nhớ cho cơ sở dữ liệu:



Bổ sung thêm datafile

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

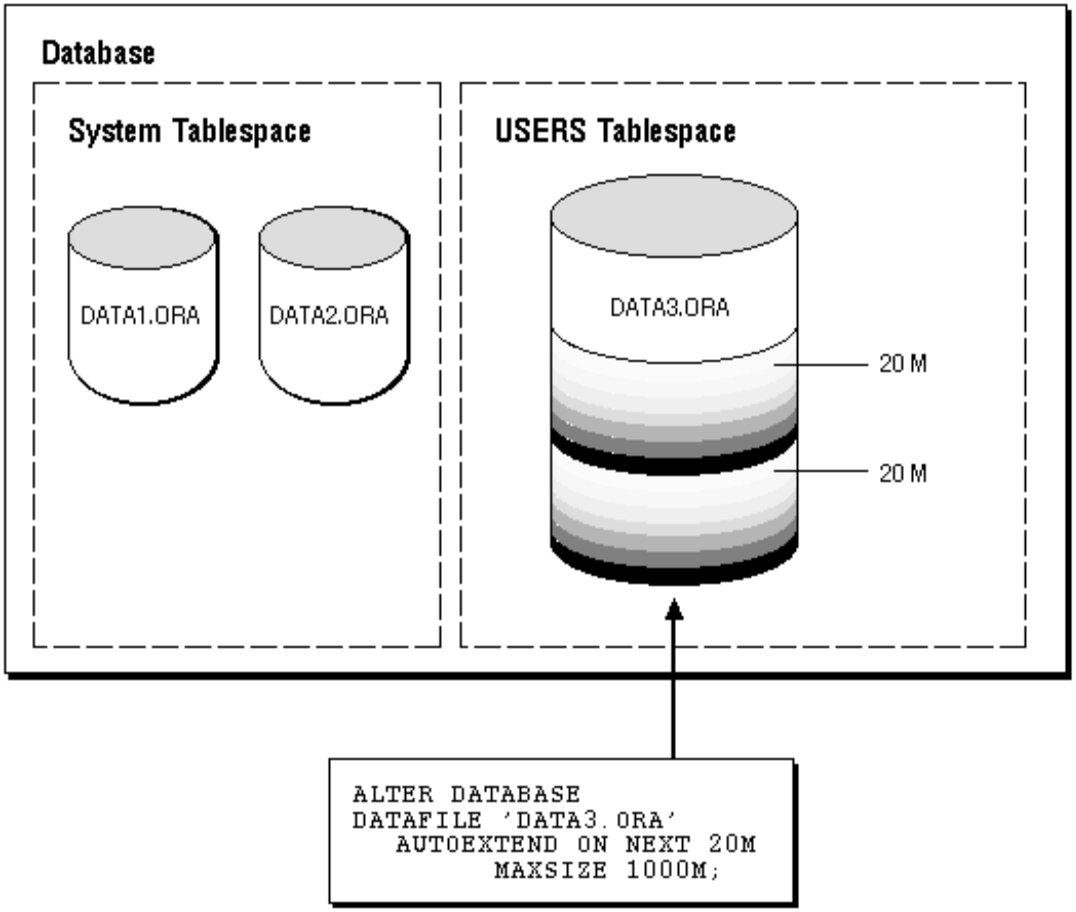
- Cấp phát thêm vùng nhớ cho cơ sở dữ liệu:



Tạo tablespace mới

Phần 1: Cơ Bản Về Oracle
Cấu trúc của Oracle

- Cấp phát thêm vùng nhớ cho cơ sở dữ liệu:



Tăng thêm vùng nhớ của datafile

Các khái niệm

- **Data Dictionary / Từ điển dữ liệu:** nơi lưu trữ thông tin về cấu trúc vật lý, luận lý cả CSDL
  - Người dùng
  - Tên, kiểu dữ liệu, các cột trong bảng dữ liệu
  - Ràng buộc toàn vẹn dữ liệu
  - Vùng nhớ cấp phát

SELECT * FROM DICTIONARY

Các khái niệm

- **Schema:** tập hợp các schema object thuộc về một người dùng.

  Quan hệ người dùng – schema: quan hệ 1-1
- **Schema Object:** đại diện cho một người dùng về các quyền trên dữ liệu, bảng … trong một CSDL. Schema Object là cấu trúc luận lý liên quan trực tiếp đến dữ liệu của CSDL

System Tables

System Table	Description
ALL_ARGUMENTS	Arguments in object accessible to the user
ALL_CATALOG	All tables, views, synonyms, sequences accessible to the user
ALL_COL_COMMENTS	Comments on columns of accessible tables and views
ALL_CONSTRAINTS	Constraint definitions on accessible tables
ALL_CONS_COLUMNS	Information about accessible columns in constraint definitions
ALL_DB_LINKS	Database links accessible to the user
ALL_ERRORS	Current errors on stored objects that user is allowed to create
ALL_INDEXES	Descriptions of indexes on tables accessible to the user
ALL_IND_COLUMNS	COLUMNs comprising INDEXes on accessible TABLES
ALL_LOBS	Description of LOBs contained in tables accessible to the user

System Tables

ALL_OBJECTS	Objects accessible to the user
ALL_OBJECT_TABLES	Description of all object tables accessible to the user
ALL_SEQUENCES	Description of SEQUENCEs accessible to the user
ALL_SNAPSHOTS	Snapshots the user can access
ALL_SOURCE	Current source on stored objects that user is allowed to create
ALL_SYNONYMS	All synonyms accessible to the user
ALL_TABLES	Description of relational tables accessible to the user
ALL_TAB_COLUMNS	Columns of user's tables, views and clusters
ALL_TAB_COL_STATISTICS	Columns of user's tables, views and clusters
ALL_TAB_COMMENTS	Comments on tables and views accessible to the user

System Tables

ALL_TRIGGERS	Triggers accessible to the current user
ALL_TRIGGER_COLS	Column usage in user's triggers or in triggers on user's tables
ALL_TYPES	Description of types accessible to the user
ALL_UPDATABLE_COLUMNS	Description of all updatable columns
ALL_USERS	Information about all users of the database
ALL_VIEWS	Description of views accessible to the user
DATABASE_COMPATIBLE_LEVEL	Database compatible parameter set via init.ora
DBA_DB_LINKS	All database links in the database
DBA_ERRORS	Current errors on all stored objects in the database
DBA_OBJECTS	All objects in the database
DBA_ROLES	All Roles which exist in the database
DBA_ROLE_PRIVS	Roles granted to users and roles
DBA_SOURCE	Source of all stored objects in the database

System Tables

DBA_TABLESPACES	Description of all tablespaces
DBA_TAB_PRIVS	All grants on objects in the database
DBA_TRIGGERS	All triggers in the database
DBA_TS_QUOTAS	Tablespace quotas for all users
DBA_USERS	Information about all users of the database
DBA_VIEWS	Description of all views in the database
DICTIONARY	Description of data dictionary tables and views
DICT_COLUMNS	Description of columns in data dictionary tables and views
GLOBAL_NAME	global database name
NLS_DATABASE_PARAMETERS	Permanent NLS parameters of the database
NLS_INSTANCE_PARAMETERS	NLS parameters of the instance

System Tables

NLS_SESSION_PARAMETERS	NLS parameters of the user session
PRODUCT_COMPONENT_VERSION	version and status information for component products
ROLE_TAB_PRIVS	Table privileges granted to roles
SESSION_PRIVS	Privileges which the user currently has set
SESSION_ROLES	Roles which the user currently has enabled.
SYSTEM_PRIVILEGE_MAP	Description table for privilege type codes. Maps privilege type numbers to type names
TABLE_PRIVILEGES	Grants on objects for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee
TABLE_PRIVILEGE_MAP	Description table for privilege (auditing option) type codes. Maps privilege (auditing option) type numbers to type names

HUTECH
ĐẠI HỌC KỸ THUẬT CÔNG NGHỆ TP.HCM

Nội Dung Trình Bày

- Phần 1: Cơ bản về Oracle
- Phần 2: Ngôn ngữ hỏi SQL
- Phần 3: Ngôn ngữ PL/SQL
- Phần 4: Quản trị Oracle

- Kiến thức nền: sinh viên đã học môn Cơ sở dữ liệu, Phân tích thiết kế hệ thống thông tin
- Kiến thức đạt được: sau phần này, sinh viên có thể thực hiện thao tác trên database, bảng. Truy vấn dữ liệu trên database, điều khiển các giao tác, các đối tượng.
- Thời lượng: 9 tiết

Giới thiệu

- Ngôn ngữ SQL (Structured Query Language): là ngôn ngữ chuẩn để thao tác trên cơ sở dữ liệu quan hệ
- Khởi nguồn của SQL là SEQUEL(Structured English Query Language) ra đời năm 1974
- Các chuẩn của SQL:
  - SQL89 (SQL1): ra đời năm 1989
  - SQL92(SQL2): ra đời năm 1992
  - SQL99 (SQL3): ra đời năm 1999

Phân loại

- Ngôn ngữ SQL có thể được phân loại như sau:
  - Ngôn ngữ định nghĩa dữ liệu: các câu lệnh cho phép can thiệp vào cấu trúc bảng như tạo, xóa, đổi tên, chỉnh sửa cấu trúc table (bảng), view (khung nhìn), xóa RBTV.
  - Ngôn ngữ thao tác dữ liệu: các câu lệnh cho phép can thiệp vào dữ liệu như thêm, xóa, sửa dữ liệu, và truy vấn dữ liệu.

Phân loại

- Ngôn ngữ SQL có thể được phân loại như sau:
  - Ngôn ngữ điều khiển dữ liệu: các câu lệnh cho phép tạo, xóa quyền người dùng, start (khởi động) hoặc stop () các quyền người dùng quyền. Các lệnh tạo, đổi mật khẩu, xóa người dùng, cấ´p quyền và thu hồi quyền sử dụng của người dùng trên cơ sở dữ liệu.
  - Ngoài ra còn có các lệnh điều khiển giao tác.
  - Với Oracle, có thêm các leệnh thao tác trên các thành phần CSDL khác: Synonym, Index và Sequence

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

1. Lệnh tạo mới bảng dữ liệu:

Cú pháp:

CREATE [GLOBAL TEMPORARY] **TABLE Table_name**

(**Column_name Datatype** [CONSTRAINT constraint_def

DEFAULT default_exp]

[, column_name type [CONSTRAINT constraint_def

DEFAULT default_exp]...]

[,table_constraint [,…]]

)

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

```
CREATE
[GLOBAL TEMPORARY]
 TABLE   table_name
( column_name datatype
[CONSTRAINT
constraint_def
DEFAULT default_exp]
[, column_name type
[CONSTRAINT constraint_def
DEFAULT default_exp]...]
[,table_constraint [,…]]
)
```

Trong đó:

- **Global Temporary: nếu có từ khóa này bảng được tạo sẽ là bảng tạm**
- **Table_name: tên bảng do người dùng đặt**
- **Column_name: tên cột trong bảng**
- **Datatype: kiểu dữ liệu của cột**
- **Constraint: từ khóa cho phép tạo ràng buộc trên cột**
- **Contraint_def: ràng buộc của cột**
- **Default: từ khóa cho phép xác định giá trị mặc định cho cột**
- **Default_exp: dữ liệu của cột nhận giá trị default_exp nếu người dùng không nhập dữ liệu vào cột**
- **Table_constraint: ràng buộc của toàn bảng dữ liệu**

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

- **Các kiểu datatype:**

Loại dữ liệu	Mô tả
VARCHAR2(n)	Dữ liệu kiểu ký tự, n<=4000
CHAR(n)	Dữ liệu kiểu ký tự, kích thước cố đinh, n<=2000
NUMBER	Kiểu số nguyên, số ký số tối đa là 38 ký số
NUMBER(p)	Kiểu số nguyên, với số ký số tối đa là p
NUMBER(p,s)	Kiểu số thực, tối đa p ký số, s số thập phân. p≤38, -84 ≤ s ≤ 127.Ví dụ: số 7456123, khai báo kiểu number (7, -2) = 7456100
DATE	Kiểu ngày, lưu ngày từ 1/1/4712 BC -> 31/12/9999
LONG	Kiểu số nguyên
RAW	Chuỗi nhị phân dài tối đa 2000 bytes
LONG RAW	Chuỗi nhị phân dài tối đa 2GB
BLOB	(Bynary Large Object) có độ dài ≤ 4GB
CLOB	(Character Large Object) có độ dài ≤ 4GB
BFILE	Chứa con trỏ chỉ đến một tập tin nhị phân ở ngoài DB

Ngôn ngữ định nghĩa dữ liệu
DDL (Data Definition Language)

Ví dụ: tạo bảng

1. KHOA(MAKHOA, TENKHOA)

```
CREATE TABLE   KHOA
(   MAKHOA varchar2(30)
    CONSTRAINT pk_khoa Primary key,
    TENKHOA varchar(50) NOT NULL ENABLE
)
```

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

Ví dụ: tạo bảng

2. SINHVIEN(MA_SV, HOTEN, PHAI, NGAY_SINH, MA_KHOA)

```
CREATE TABLE   SINHVIEN
(   MA_SV varchar2(30) CONSTRAINT pk_khoa Primary key,
    HOTEN varchar(50) NOT NULL ENABLE,
    PHAI varchar(5) CONSTRAINT check_phai CHECK
                        (PHAI Between 'Nam' and 'Nu'),
    NGAY_SINH date,
    MA_KHOA varchar2(30) CONSTRAINT fk_sv
                    REFERENCES KHOA(MAKHOA)
)
```

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

- **Các loại ràng buộc dữ liệu trên cột**
  - Mô tả: CONSTRAINT <constraint_name>[<constraint_type>]
  - Các loại ràng buộc

Kiểu ràng buộc	Diễn giải
NULL/NOT NULL	Cho phép/không cho phép cột chứa giá trị rỗng
CHECK (*Criteria*)	Giá trị nhập vào cột phải thỏa *Criteria*
UNIQUE	Giá trị nhập vào cột phải là duy nhất

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

- **Các loại ràng buộc dữ liệu trên cột**
  - Mô tả: CONSTRAINT <constraint_name>[<constraint_type>]
  - Các loại ràng buộc

Kiểu ràng buộc	Diễn giải
PRIMARY KEY	Ràng buộc khóa chính cho cột
REFERENCES <*Table_name1*>(*Column_name 1*)	Ràng buộc khóa ngoại cho cột. Cột làm khóa ngoại sẽ tham chiếu tới *Column_name* đã tồn tại trong bảng *Table_name*.

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

2. Lệnh chỉnh sửa cấu trúc bảng

- **Cú pháp**:

  ALTER TABLE Table_name

  – **Thêm cột vào bảng**

  ALTER TABLE Table_name ADD Column_name Datatype

- **Diễn giải**: thêm cột *column_name* với kiểu dữ liệu *datatype* vào bảng *table_name*
- **Ví dụ**: Thêm cột QUEQUAN vào bảng

  SINHVIEN(MA_SV, HOTEN, PHAI, NGAY_SINH, MA_KHOA)

  ALTER TABLE SINHVIEN ADD QUEQUAN varchar2(50)

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

2. Lệnh chỉnh sửa cấu trúc bảng

– Thêm cột vào bảng

SINHVIEN(MA_SV, HOTEN, PHAI, NGAY_SINH, MA_KHOA)

SINHVIEN(MA_SV, HOTEN, PHAI, NGAY_SINH, MA_KHOA, QUEQUAN)

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

2. Lệnh chỉnh sửa cấu trúc bảng

- **Cú pháp**:

  ALTER TABLE Table_name

  – **Chỉnh sửa cột trong bảng**

  ALTER TABLE Table_name MODIFY Column_name Datatype {SET DEFAULT <expression>|DROP DEFAULT}

- **Diễn giải**: chỉnh sửa cấu trúc cột *column_name* trong bảng *table_name*
- **Ví dụ**: SINHVIEN(MA_SV, HOTEN, PHAI, NGAY_SINH, MAKHOA, QUEQUAN)

  Thay đổi kiểu dữ liệu cột QUEQUAN thành varchar2(80)

  ALTER TABLE SINHVIEN MODIFY QUEQUAN varchar2(80)

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

2. Lệnh chỉnh sửa cấu trúc bảng

- **Cú pháp:**

  ALTER TABLE Table_name

  – **Thêm ràng buộc toàn vẹn cho cột**

  ALTER TABLE Table_name

  ADD CONSTRAINT constraint_name constraint_type

- **Diễn giải:** thêm ràng buộc toàn vẹn có tên *constraint_name* trong bảng *table_name*
- **Ví dụ**: Thêm ràng buộc khóa ngoại cho cột MAKHOA trong bảng SINHVIEN(MA_SV, HOTEN, PHAI, NGAY_SINH, MAKHOA, QUEQUAN), tham chiếu đến cột MAKHOA của bảng KHOA

ALTER TABLE SINHVIEN ADD CONSTRAINT fk_sv

FOREIGN KEY (MAKHOA) REFERENCES KHOA(MAKHOA)

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

2. Lệnh chỉnh sửa cấu trúc bảng

- **Cú pháp**:

  ALTER TABLE Table_name

  – **Xóa cột trong bảng**

  ALTER TABLE Table_name

  DROP COLUMN column_name

- **Diễn giải:** xóa cột *column_name* trong bảng *table_name*
- **Ví dụ**: Xóa cột QUEQUAN trong bảng SINHVIEN(MA_SV, HOTEN, PHAI, NGAY_SINH, MAKHOA, QUEQUAN)

ALTER TABLE SINHVIEN DROP COLUMN QUEQUAN

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

2. Lệnh chỉnh sửa cấu trúc bảng

- **Cú pháp**:

  ALTER TABLE Table_name

  – **Xóa ràng buộc toàn vẹn trong bảng**

  ALTER TABLE Table_name

  DROP CONSTRAINT constraint_name

- **Diễn giải:** xóa ràng buộc toàn vẹn có tên *constraint_name* trong bảng *table_name*
- **Ví dụ**: Xóa ràng buộc khóa ngoại fk_sv trong bảng SINHVIEN

ALTER TABLE SINHVIEN DROP CONSTRAINT fk_sv

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

3. Xóa bảng

- **Cú pháp**:

  DROP TABLE Table_name

  [CASCADE CONSTRAINT]

- **Diễn giải** : xóa bảng *table_name*, nếu có CASCADE CONSTRAINT sẽ xóa luôn tất cả các ràng buộc toàn vẹn liên quan đến bảng cần xóa
- Ví dụ: Xóa bảng KHOA(MAKHOA, TENKHOA)

DROP TABLE KHOA CASCADE CONSTRAINT

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

4. Thêm dữ liệu vào bảng

- **Cú pháp**:

  INSERT INTO Table_name [Column_name1, Column_name2, …] VALUES (value1, value2, …)

- **Diễn giải:** thêm một dòng dữ liệu vào bảng *Table_name* các giá trị *value1*, *value2*… theo thứ tự các cột được liệt kê *column_name1*, *column_name2*…
- **Ví dụ**: Thêm dữ liệu vào bảng KHOA(MAKHOA, TENKHOA)

INSERT INTO KHOA(MAKHOA, TENKHOA) VALUES (‘CNTT’, ‘CONG NGHE THONG TIN’)

INSERT INTO KHOA VALUES (‘QTKD’, ‘QUAN TRI KINH DOANH’)

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

5. Chỉnh sửa dữ liệu trong bảng

- **Cú pháp**:

  UPDATE Table_name SET (colum_name1 = value1, column_name2 = value2, …) WHERE <criteria>

- **Diễn giải**: chỉnh sửa dữ liệu cột *column_name1* thành *value1*, *column_name2* thành *value2* với các dòng dữ liệu thỏa *criteria*
- **Ví dụ**: đổi tên khoa thành ‘QUAN TRI KINH DOANH VA DU LICH’ với khoa có mã là ‘QTKD’

  UPDATE KHOA SET TENKHOA='QUAN TRI KINH DOANH VA DU LICH' WHERE MAKHOA='QTKD'

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language)

5. Xóa dữ liệu trong bảng

- **Cú pháp**:

  DELETE FROM Table_name WHERE <criteria>

- **Diễn giải**: xóa các dòng dữ liệu trong bảng *table_name* thỏa *criteria*
- **Ví dụ**:

DELETE FROM KHOA WHERE MAKHOA='QTKD'

Bài tập

- SAN_PHAM (MA_SP, TEN_SP, DVT, DIENGIAI)
- NHAP_KHO (HD_NHAP, NGAY_NHAP)
- NHAP_KHO_CT (*HD_NHAP*, MA_SP, DON_GIA, SOLUONG)
- XUAT_KHO(HD_XUAT, NGAY_XUAT)
- XUAT_KHO_CT (*HD_XUAT*, MA_SP, SOLUONG, DON_GIA)
- TON_KHO(MA_SP, DON_GIA, SOLUONG)

Yêu cầu:

- Tạo các bảng trên
- Tạo các RBTV
- Nhập dữ liệu vào bảng

NGÔN NGỮ SQL

TRUY VẤN CƠ SỞ DỮ LIỆU

Truy vấn trên nhiều bảng

- Nội dung
  - Tạo định danh (SYSNONYM)
  - Các lỗi thường gặp
  - Truy vấn phức tạp

SYNONYM

- Mục đích
  - Tạo định danh (tên gọi tắt) tham chiếu đến một đối tượng của một schema
- Cú pháp:

Tạo định danh:

CREATE SYNONYM <tên định danh>

FOR <tên schema>.<tên object>

Xóa định danh:

DROP SYNONYM <tên định danh>

SYNONYM

- Ví dụ:

  PHONG(MA_PHONG, TEN_PHONG)

  CREATE SYNONYM P FOR PHONG;

  - INSERT INTO P(MA_PHONG, TEN_PHONG)

  VALUES(1,'Dao tao');

  - SELECT * FROM P; # SELECT * FROM PHONG;

Lỗi thường gặp

- Ví dụ: PHONG(MA_PHONG, TEN_PHONG)

SELECT MA, TEN FROM PHONG;

Error starting at line 1 in command:
SELECT MA, TEN FROM PHONG
Error at Command Line:1 Column:11
Error report:
SQL Error: ORA-00904: "TEN": **invalid identifier**
00904. 00000 - "%s: invalid identifier“

Lỗi thường gặp - **ORA-00942**

- Ví dụ: PHONG(MA_PHONG, TEN_PHONG)

  SELECT MA_PHONG, TEN_PHONG FROM PHG;

Error starting at line 1 in command:
SELECT MA_PHONG, TEN_PHONG FROM PHG
Error at Command Line:1 Column:32
Error report:
SQL Error: ORA-00942: **table or view does not exist**

Lỗi thường gặp - **ORA-00918**

- Ví dụ: PHONG(MA_PHONG, TEN_PHONG)

  NHANVIEN(MA_NV, HTEN_NV, MA_PHONG)

```
SELECT MA_PHONG, MA_NV, HTEN_NV
FROM PHONG P, NHANVIEN NV
WHERE P.MA_PHONG = NV.MA_PHONG;
```

Error at Command Line:1 Column:7
Error report:
SQL Error: ORA-00918: **column ambiguously defined**

Lỗi thường gặp - **ORA-01722**

- Ví dụ: PHONG

MA_PHONG	NUMBER
TEN_PHONG	VARCHAR2(100 BYTE)
TRUONG_PHONG	NUMBER

```
SELECT MA_PHONG, TEN_PHONG
FROM PHONG WHERE MA_PHONG = 'PDT';
```

Error starting at line 25 in command:
….
Error report:
SQL Error: ORA-01722**: invalid number**

Bài tập
Thiết lập ràng buộc giữa các bảng

- **Xác định khóa chính?**
  - **CONGNHAN** (HOTEN_CN, NAMS_CN, NAM_VAO_N, CH_MON)
  - **THAMGIA** (HOTEN_CN, STT_CTR, NGAY_TGIA, SO_NGAY)
  - **KTRUCSU** (HOTEN_KTS, NAMS_KTS, PHAI, NOI_TN,
  - DCHI_LL_KTS)
  - **THIETKE** (HOTEN_KTS, STT_CTR, THU_LAO)
  - **CGTRINH** (STT_CTR, TEN_CTR, DIACHI_CTR, TINH_THANH,
  - KINH_PHI, TEN_CHU, TEN_THAU, NGAY_BD)

Bài tập
Thiết lập ràng buộc giữa các bảng

- **Xác định khóa ngoại?**
  - **CONGNHAN** (HOTEN_CN, NAMS_CN, NAM_VAO_N, CH_MON)
  - **THAMGIA** (HOTEN_CN, STT_CTR, NGAY_TGIA, SO_NGAY)
  - **KTRUCSU** (HOTEN_KTS, NAMS_KTS, PHAI, NOI_TN, DCHI_LL_KTS)
  - **THIETKE** (HOTEN_KTS, STT_CTR, THU_LAO)
  - **CGTRINH** (STT_CTR, TEN_CTR, DIACHI_CTR, TINH_THANH,

Bài tập
Tạo Synonym

CREATE SYNONYM <tên định danh>
FOR <tên schema>.<tên object>

CONGNHAN => CN

THAMGIA => TG

KTRUCSU => KTS

THIETKE => TK

CGTRINH => CgTr

CHUTHAU =>

Bài tập
Truy vấn thông tin qua định danh

- Tìm tên và địa chỉ liên lạc các chủ thầu thi công công trình ở Cần Thơ

CGTRINH (STT_CTR, TEN_CTR, DIACHI_CTR, TINH_THANH, KINH_PHI, TEN_CHU, TEN_THAU, NGAY_BD)

CHUTHAU (TEN_THAU, TEL, DCHI_THAU)

Bài tập
Truy vấn thông tin qua định danh

- Tìm tên và địa chỉ liên lạc các chủ thầu thi công công trình ở Cần Thơ do kiến trúc sư Lê Kim Dung thiết kế

THIETKE (HOTEN_KTS, STT_CTR, THU_LAO)

CGTRINH (STT_CTR, TEN_CTR, DIACHI_CTR, TINH_THANH, KINH_PHI, TEN_CHU, TEN_THAU, NGAY_BD)

CHUTHAU (TEN_THAU, TEL, DCHI_THAU)

Bài tập

Truy vấn thông tin qua định danh

- Tìm nơi tốt nghiệp của KTS thiết kế ks Quốc Tế Cần Thơ

KTRUCSU (HOTEN_KTS, NAMS_KTS, PHAI, NOI_TN, DCHI_LL_KTS)

THIETKE (HOTEN_KTS, STT_CTR, THU_LAO)

CGTRINH (STT_CTR, TEN_CTR, DIACHI_CTR, TINH_THANH, KINH_PHI, TEN_CHU, TEN_THAU, NGAY_BD)

Bài tập
Truy vấn thông tin qua định danh

- Tìm tên, năm sinh, năm vào nghề của công nhân

có chuyên môn hàn hoặc điện tham gia công

trình của chủ thầu Lê Văn Sơn

CONGNHAN (HOTEN_CN, NAMS_CN,

NAM_VAO_N, CH_MON)

THAMGIA (HOTEN_CN, STT_CTR, NGAY_TGIA, SO_NGAY)

CGTRINH (STT_CTR, TEN_CTR, DIACHI_CTR,

SELECT lồng nhau

- Nội dung
  - Các hàm kết hợp min, max, sum và avg
  - Xây dựng câu select truy vấn lồng nhau

Các hàm kết hợp min, max, sum và avg

- **min**(<tên cột>): tìm giá trị nhỏ nhất của cột
- **max**(<tên cột>): tìm giá trị lớn nhất của cột
- **sum**(<tên cột>): cộng tộng tất cả các giá trị trong cột
- **avg**(<tên cột>): tính trung bình các giá trị trong cột

Các hàm kết hợp min, max, sum và avg

Ví dụ:

- Kinh phí cao nhất cho một công trình là bao nhiêu?
- Kinh phí trung bình cho một công trình là bao

nhiêu?

Các hàm kết hợp min, max, sum và avg

KI...
450
200
1000
30
3000
40
65
100

- SELECT MAX(KINH_PHI) FROM CGTRINH

=>3000

- SELECT SUM(KINH_PHI) FROM CGTRINH

=>4885

- SELECT AVG(KINH_PHI) FROM CGTRINH

=>6010

Select chứa select con (select lồng nhau)

- **Cú pháp**

  SELECT <Tên các cột>

  FROM <Tên bảng>

  WHERE <điều kiện chứa select con>

  (SELECT <Tên các cột>

  FROM <Tên bảng> [**WHERE** <Điều kiện>] **)**

  [AND | OR <điều kiện chứa select con>

  (SELECT <Tên các cột>

  FROM <Tên bảng> [**WHERE** <Điều kiện>] **)**

  [...]]

Select chứa select con (select lồng nhau)

- Mục đích:
  - Điều kiện tìm kiếm lấy từ kết quả của câu select khác
  - Giúp trả lời các câu hỏi
    - Tìm dự án thi công dài nhất?
    - Tìm kiến trúc sư có thù lao cao nhất?
    - Tìm các tác giả có nhiều bài viết nhất?
    - ...

Các dạng so sánh với select con

<tên cột> <so sánh> (<select con>)

- Điều kiện đúng khi giá trị của cột so sánh đúng với giá trị trả về từ select con (1 rows)
- Ví dụ: cho biết tên công trình có kinh phí cao nhất

```
Select TEN_CTR From CGTRINH
Where KINH_PHI = (Select
  max(KINH_PHI)
```

Các dạng so sánh với select con

<tên cột> <so sánh> ALL(<select con>)

- Điều kiện đúng khi giá trị của cột so sánh đúng với tất cả các giá trị trả về từ select con (1-n rows)
- Ví dụ: cho biết tên chủ thầu thi công xây dựng có kinh phí lớn hơn tất cả các công trình của chủ thầu 586

```
Select TEN_CTR From CGTRINH
Where KINH_PHI > ALL (Select KINH_PHI
                      From CGTRINH
```

Các dạng so sánh với select con

<tên cột> <so sánh> ANY | SOME(<select con>)

- Điều kiện đúng khi giá trị của cột so sánh đúng với bất kỳ một giá trị nào trả về từ select con (1-n rows)
- Ví dụ: cho biết tên KTS tham gia bất kỳ công trình nào ở Cần Thơ

```
Select HOTEN_KTS  From THIETKE
Where STT_CTR = ANY  (Select STT_CTR
                      From CGTRINH
                      Where TINH_THANH = N'Cần Thơ');
```

Các dạng so sánh với select con

<tên cột> <so sánh> [NOT] IN(<select con>)

- Điều kiện đúng khi giá trị của cột nằm trong tập hợp các giá trị trả về của select con (1-n rows)
- Ví dụ: tìm họ tên và chuyên môn của các công nhân tham gia các công trình do KTS Lê Thanh Tùng thiết kế

```
Select a.HOTEN_CN, CH_MON
From CONGNHAN a, THAMGIA b
Where a.HOTEN_CN = b.HOTEN_CN
and b.STT_CTR IN (Select STT_CTR From
```

Các dạng so sánh với select con

[NOT] EXISTS(<select con>)

- Điều kiện đúng khi kết quả trả về của select con khác rỗng (1-n rows)
- Ví dụ: tìm họ tên công nhân không làm việc cho công trình khách sạn Quốc tế tỉnh Cần Thơ

```
select HOTEN_CN, CH_MON from CONGNHAN CN
where not exists ( select * from THAMGIA TG, CGTRINH CT
        where CN. HOTEN_CN = TG.HOTEN_CN
        and TG.STT_CTR = CT.STT_CTR
        and CT. TEN_CTR = 'khach san quoc te'
```

Bài tập
Quản lý mua bán hàng

- **HANGHOA(MA_HANG,TEN_HG)**
- **DAILY(STT_DL, TEN_DL, DCHI_DL)**
- **MUA(MA_HANG, STT_DL, NGAY_MUA, SOLG_MUA, TRIGIA_MUA)**
- **BAN(MA_HANG, STT_DL, NGAY_BAN, SOLG_BAN,**

Bài tập
Quản lý mua bán hàng

- Tìm tên những đại lý vừa có bán coca vừa có bán pepsi
- Tìm tên những mặt hàng được mua nhưng chưa bán
- Tìm tên và địa chỉ những đại lý có mua cùng mặt hàng với Vạn Lợi mua
- Tìm tổng giá trị mua Coca Cola ở đại lý Tân Hiệp Hưng
- Tìm tổng giá trị bán Coca Cola ở đại lý Tân Hiệp Hưng

Bài tập
Quản lý mua bán hàng

- Tìm tên những mặt hàng được bán ở đại lý Tân Hiệp Hưng vào cả 2 ngày 15/12/2004 và 31/12/2004
- Có bao nhiêu mặt hàng được mua vào nhưng chưa được bán ra tại đại lý Tân Hiệp Hưng trong tháng 12/2004

GROUP BY trong SELECT

- Nội dung
  - Bảng tạm
  - Câu truy vấn GROUP BY

Bảng tạm

- Cú pháp:

  WITH <tên bảng tạm> **AS** (<câu truy vấn con>)

  [, <tên bảng tạm> AS (<câu truy vấn con>) [, ...]]

  SELECT ……

  FROM ……

  WHERE ……;

- Mục đích: đơn giản trong sử dụng lại nhiều lần cùng một câu truy vấn trong select tổng

Bảng tạm

- Ví dụ:

  WITH t1 **AS** (select * from congnhan)

  Select * from t1

	HOTEN_CN	NAMS_CN	NAM_VAO_N	CH_MON
1	nguyen thi suu	45	60	ho
2	vi chi a	66	87	han
3	le manh quoc	56	71	moc
4	vo van chin	40	52	son

- Ghi chú: các bảng tamh chỉ có ý nghĩa trong câu truy vấn ngay sau đó

GROUP BY

- **Cú pháp**

  SELECT <các cột phân nhóm>,

  <hàm-kết-tập(<biểu thức>)>

  FROM <tên bảng>

  [WHERE <Điều kiện>]

  GROUP BY <Cột để phân nhóm>

  [, <Cột để phân nhóm>

  [,...]]

  HAVING <điều kiện chọn trên nhóm>

GROUP BY

Hàm kết tập theo nhóm:

- Count: đếm số mẫu tin thuộc nhóm này
- Min: giá trị nhỏ nhất trong nhóm
- Max: giá trị lớn nhất trong nhóm
- Avg: giá trị trung bình của nhóm
- Sum: tổng giá trị của nhóm

GROUP BY

Ví dụ: tìm tổng kinh phí của tất cả các công trình theo từng chủ thầu

CGTRINH(STT_CTR, TEN_CTR, DIACHI_CTR, TINH_THANH, KINH_PHI, TEN_CHU, TEN_THAU, NGAY_BD)

Select TEN_THAU, sum(KINH_PHI) As TongKP

From CGTRINH

Group by TEN_THAU;

GROUP BY

Ví dụ: họ tên KTS có tổng thù lao thiết kế công trình lớn hơn 25 triệu

THIETKE(HOTEN_KTS, STT_CTR, THU_LAO)

Select HOTEN_KTS, sum(THU_LAO) As TongTL

From THIETKE

Group by HOTEN_KTS

GROUP BY

Ví dụ: số lượng KTS có tổng thù lao thiết kế công trình lớn hơn 25 triệu

THIETKE(HOTEN_KTS, STT_CTR, THU_LAO)

Select count(*) As Solg_KTS

From

(*Select HOTEN_KTS, sum(THU_LAO) As TongTL*

From THIETKE Group by HOTEN_KTS

GROUP BY

Ví dụ: tìm tên, địa chỉ công trình có nhiều công nhân nhất

THAMGIA(HOTEN_CN, STT_CTR, NGAY_TGIA, SO_NGAY)

CGTRINH(STT_CTR, TEN_CTR, DIACHI_CTR, TINH_THANH, KINH_PHI, TEN_CHU, TEN_THAU, NGAY_BD)

```
with t1 as (
Select STT_CTR, count(*) As TongSoCN
From THAMGIA
```

GROUP BY

```
with t1 as (
Select STT_CTR, count(*) As TongSoCN
From THAMGIA
Group by STT_CTR)
Select TEN_CTR, DIACHI_CTR,
   TongSoCN
From t1 a, CGTRINH b
Where a.STT_CTR = b.STT_CTR
and TongSoCN = (Select Max(TongSoCN)
From t1 );
```

Bài tập

- Cho biết tên thành phố và kinh phí trung bình cho mỗi công trình của từng thành phố
- Số lượng công nhân cho mỗi công trình
- Số lượng công nhân cho mỗi công trình kèm theo tên công trình

CHƯƠNG 3

CÁC HÀM CÓ SẴN TRONG ORACLE

Các hàm có sẵn trong Oracle

- Nội dung
  - Hàm xử lý chuỗi
  - Hàm xử lý ngày tháng
  - Biểu thức Case

Hàm xử lý chuỗi

- LENGTH (<chuỗi>)
  - Trả về chiều dài chuỗi
  - Select LENGTH(‘lop 08DTHH’) LEN From Dual

    =>10

Hàm xử lý chuỗi

- INSTR(<chuỗi a>, <chuỗi con b>
  <vị trí bắt đầu tìm n>,<số lần xuất hiện m>)
  - Tìm vị trí chuỗi b trong chuỗi a bắt đầu từ vị trí n, lần xuất hiện thứ m.
  - n<0: tìm từ phải sang trái
  - Select INSTR('CORORATE FLOOR','OR',3,2) From Dual => 14
  - Select INSTR('CORPORATE FLOOR','OR',-3,1)

Hàm xử lý chuỗi

- SUBSTR(<chuỗi a>

  <vị trí bắt đầu cắt lấy n>,<số ký tự cần cắt m>)

  – Cắt lấy chuỗi con của chuỗi a, lấy từ vị trí n về phải m ký tự, nếu không chỉ m thì lấy cho đến cuối chuỗi .

  – n<0: vị trí cắt đếm từ phải sang trái

  – Select SUBSTR('ORACLE',1,3) Substring From Dual => 'ORA'

  – Select SUBSTR('Lop 08DTH',-3,3) Substring

Hàm xử lý chuỗi

- CONCAT(<chuỗi a>,<chuỗi b>)
  - Nối chuỗi a và chuỗi b .
  - Select CONCAT(‘Lop 08DTH’,’ Khoa CNTT’) Substring From Dual => ‘Lop 08DTH Khoa CNTT’
- LOWER/UPPER/INITCAP (<chuỗi>)
  - Chuyển chuỗi thành chuỗi viết thường /hoa /ký tự đầu các từ viết hoa
  - INITCAP(‘khoa cong nghe thong tin’) => ’Khoa Cong Nghe Thong Tin’

Hàm xử lý chuỗi

- LTRIM/RTRIM(<chuỗi a>, <chuỗi b>)
  - Loại bỏ chuỗi b ở bên trái/ phải chuỗi a
  - Select LTRIM('khoa cong nghe thong tin', 'khoa') From Dual;=> 'cong nghe thong tin'
  - Select RTRIM('LAST WORDx', 'xy') FROM Dual;

    => 'LAST WORD'

Hàm xử lý chuỗi

- REPLACE(chuỗi a, chuỗi b [,chuỗi c])
  - Thay tất cả các chuỗi b có trong chuỗi a bằng chuỗi c
  - SELECT JOB, REPLACE(JOB, ‘SALESMAN’, ‘SALESPERSON’) JOB_RE, ENAME, REPLACE(ENAME, ‘CO’,’PR’) ENAME_RE

    FROM EMP WHERE DEPTNO =30 OR DEPTNO =20;

JOB	**JOB_RE,**	**ENAME**	**ENAME_RE**
MANAGER	MANAGER	JONES	JONES
SALESMAN	SALESPERSON	MARTIN	MARTIN
SALESMAN	SALESPERSON	ALLEN	ALLEN
CLERK	CLERK	JAMES	JAMES

Hàm xử lý ngày tháng

- SYSDATE
  - Hàm lấy ngày tháng hiện hành
  - SELECT sysdate FROM Dual;
- EXTRACT(YEAR|MONTH|DAY FROM <chuỗi ngày>)
  - Tách lấy năm/ tháng/ ngày của <chuỗi ngày>
  - Select EXTRACT(Month FROM DATE'2001-05-07') FROM Dual; => 5
  - Select EXTRACT(DAY FROM Sysdate) FROM Dual;

Hàm xử lý ngày tháng

- MONTHS_BETWEEN(<ngày 1>, <ngày 2>)
  - Cho biết số tháng giữa 2 ngày <ngày 1> và <ngày 2>
  - Select MONTHS_BETWEEN (DATE'1981-01-02',DATE'1980-01-02') FROM Dual
    => 2
- ADD_MONTHS(<ngày>, n)
  - Thêm n tháng vào <ngày>
  - Select ADD_MONTHS(DATE'1981-01-02', 2) FROM Dual;

Hàm xử lý ngày tháng

- NEXT_DAY(<ngày>, <thứ>)
  - Cho biết <thứ> sau <ngày> là ngày nào
  - Select NEXT_DAY(DATE'2011-02-10', 'friday') FROM Dual; => 07-10-2011
- LAST_DAY(<ngày>)
  - Cho biết ngày cuối cùng trong tháng chỉ bởi <ngày>
  - Select last_day(date'2011-10-02') FROM Dual;

    => 31-10-2011

Hàm xử lý ngày tháng

- Bài tập:
  - Tính tuổi người có ngày sinh ‘1980-04-01’
  - **CONGNHAN** (HOTEN_CN, NAMS_CN, NAM_VAO_N, CH_MON)

    Viết RBTV cho cột NAMS_CN: tuổi của công nhân phải từ 18 đến 60

Chuyển kiểu

- TO_CHAR(<số>)
  - Chuyển số sang dạng chuỗi
- TO_CHAR(<ngày>, <chuỗi định dạng>)
  - Chuyển ngày sang chuỗi theo định dạng
  - Select TO_CHAR(DATE'2011-09-20', 'DD-MM-YYYY') FROM Dual; => ‘20-09-2011’
- TO_NUMBER(<chuỗi ký số>)
  - Chuyển chuỗi ký số sang dạng số
  - Select TO_NUMBER('123')+TO_NUMBER('34') FROM Dual; => 157

Chuyển kiểu

- TO_DATE(<chuỗi dàng ngày tháng>, <chuỗi định dạng>)
  - Chuyển chuỗi ngày sang dạng ngày theo định dạng
  - Select TO_DATE('02-02-2011','DD-MM-YYYY') FROM Dual;
    => 02-02-2011
  - Select MONTHS_BETWEEN(TO_DATE('02-02-2011','DD-MM-YYYY'), TO_DATE('01-01-2011','DD-MM-YYYY')) FROM Dual;

Biểu thức Case

- CASE WHEN <biểu thức điều kiện 1> THEN <KQ trả về 1>

  [WHEN <biểu thức điều kiện 1> THEN <KQ trả về 2>…]

  [ELSE <KQ trả về ngoại lệ>]

  END

  – Đánh giá danh sách các điều kiện và trả về giá trị phù hợp

  – Select CASE WHEN extract(day from sysdate)>15 THEN 'cuoi thang' ELSE 'Dau thang'

Biểu thức Case

- Bài tập:

  NHANVIEN(MA_NV, HO_TEN, GIOI_TINH)

  Trong đó GIOI_TINH =1 nếu là nam, =0 nếu là nữ

  Hiển thị danh sách gồm:

  – Mã
  – Họ tên
  – Giới tính ghi rõ nam hay nữ

Chương 5:

NGÔN NGỮ THỦ TỤC PL/SQL

Nội dung

- PL/SQL là gì?
- Cấu trúc một chương trình PL/SQL
- Các kiểu dữ liệu trong PL/SQL
- Các mệnh đề thông dụng trong PL/SQL
- Cấu trúc lập trình điều khiển trong PL/SQL
- Sử dụng Cursor để xử lý dữ liệu trên kết quả câu truy vấn

PL/SQL

- PL/SQL: Procedural Language/SQL
  - Xử lý dữ liệu bằng các cú pháp lập trình thông thường
  - Thực hiện các câu truy vấn 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

- Mục đích:
  - Tăng thêm sức mạnh cho SQL
  - Xử lý kết quả truy vấn trên từng dòng
  - Phát triển chương trình ứng dụng trên cơ sở Module
  - Tái sử dụng những đoạn code (module)
  - Giảm chi phí cho việc bảo trì và thay đổi ứng dụng

Cấu trúc một chương trình PL/SQL

Anonymous

```
[DECLARE]


BEGIN
   --statements

[EXCEPTION]

END;
```

Procedure

```
PROCEDURE name
IS

BEGIN
   --statements

[EXCEPTION]

END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
   --statements
   RETURN value;
[EXCEPTION]

END;
```

Cấu trúc một chương trình PL/SQL

Anonymous

```
[DECLARE]

BEGIN
   --statements

[EXCEPTION]

END;
```

Procedure

```
PROCEDURE name
IS

BEGIN
   --statements

[EXCEPTION]

END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
   --statements
   RETURN value;
[EXCEPTION]

END;
```

Khối vô danh được phép lồng trong một khối khác

Khối định danh (thủ tục/ hàm)

Cấu trúc một chương trình PL/SQL

SQL Worksheet | History

0.006 seconds | thu

Worksheet | Query Builder

```
set serveroutput on;
declare
        today date default sysdate;
begin
        DBMS_OUTPUT.PUT_LINE('Today is: '||today);
end;
```

Script Output

Task completed in 0.006 seconds

```
anonymous block completed
Today is: 02-OCT-11
```

Cấu trúc một chương trình PL/SQL

SQL Worksheet | History

0.008 seconds | thu

Worksheet | Query Builder

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE show_the_date IS
      today DATE DEFAULT SYSDATE;
BEGIN
      DBMS_OUTPUT.PUT_LINE('Today is: '||today);
END;
```

Script Output

Task completed in 0.008 seconds

```
PROCEDURE show_the_date compiled
```

Các kiểu dữ liệu trong PL/SQL

- Các kiểu dữ liệu cơ bản:
  - BINARY_INTEGER: từ -231 đến 231-1
  - NUMBER[(precision, scale)]: precision<=38, Scale: -84 đến 127
  - DEC, DECIMAL, NUMERIC, DOUBLE, FLOAT: kiểu số thực
  - INTEGER, INT, SMALLINT: kiểu số nguyên có 38 chữ số

Các kiểu dữ liệu trong PL/SQL

- Các kiểu dữ liệu cơ bản:
  - BOOLEAN: kiểu luận lý, có 3 giá trị True, False, Null
  - CHAR(max_length): kiểu ký tự
  - DATE: thế kỷ, năm, tháng, ngày, giờ, phút, giây
  - VARCHAR2(max_length): max_length<=32767
  - LONG: <=32760 bytes

Các kiểu dữ liệu trong PL/SQL

- Tham chiếu kiểu %TYPE
  - Dùng để khai báo một biến mà nó ***tham chiếu đến một cột trong cơ sở dữ liệu. (có cấu trúc như một cột trong Table).***
  - **Khai báo:**

```
<variable_name>
<field_name | other_variable_name>%TYPE
```

Các kiểu dữ liệu trong PL/SQL

- Tham chiếu kiểu %TYPE
  - Ví dụ:

```
SET SERVEROUTPUT ON;
DECLARE
    Kinhphi_nhonhat  CGTRINH.Kinh_phi%TYPE;
BEGIN
    SELECT MIN(CT.Kinh_phi) INTO Kinhphi_nhonhat
    FROM CGTRINH CT;
    DBMS_OUTPUT.PUT_LINE(kinhphi_nhonhat);
END;
```

→ *Kiểu dữ liệu chính xác của kinhphi_nhonhat không cần biết*

→ *Khi kiểu của cột kinhphi thay đổi thì kiểu của kinhphi_nhonhat cũng thay đổi theo.*

Các kiểu dữ liệu trong PL/SQL

- Tham chiếu kiểu %ROWTYPE
  - Dùng để khai báo một biến mà nó *tham chiếu đến một dòng trong cơ sở dữ liệu (Có cấu trúc như một dòng trong Table).*
  - Khai báo:

```
<variable_name>  <table_name>%ROWTYPE
```

Các kiểu dữ liệu trong PL/SQL

- Tham chiếu kiểu %ROWTYPE
  - Ví dụ:

```
SET SERVEROUTPUT ON;
DECLARE
    row_cuahang cuahang%ROWTYPE;
BEGIN
    SELECT * INTO row_cuahang
    FROM cuahang WHERE mach = 22;
    DBMS_OUTPUT.PUT_LINE(row_cuahang.mach || ' : ' ||
    row_cuahang.tench );
END;
```

→ *Khi truy xuất đến từng dòng ta sử dụng giống như một bảng dữ liệu (trường hợp này bảng có 1 record) tham chiếu đến cột*

Các kiểu dữ liệu trong PL/SQL

- Tham chiếu kiểu %TABLE
  - Dùng để khai báo một biến mà nó *tham chiếu đến một bảng trong cơ sở dữ liệu (Có cấu trúc như một Table).*
  - Khai báo:

```
TYPE <table_name> IS
TABLE OF <type> [NOT NULL] INDEX BY
 BINARY_INTEGER;
<variable_name>  <table_name>;
```

Các kiểu dữ liệu trong PL/SQL

- Tham chiếu kiểu %TABLE
  - Ví dụ:

```
DECLARE
    TYPE cuahang_type IS TABLE OF cuahang
    %ROWTYPE
    INDEX BY BINARY_INTEGER;
    emp_tab cuahang_type;
BEGIN
    SELECT * INTO emp_tab(1) FROM cuahang
    WHERE mach = 22;
    DBMS_OUTPUT.PUT_LINE(emp_tab(1).mach || ' : ' ||
    emp_tab(1).tench );
END;
```

Khai báo biến và hằng PL/SQL

- Khai báo biến:

```
<variable_name> <type[(scale)]>;
```

  - Ví dụ: mucluong NUMBER(5);
- Khai báo hằng:

```
<variable_name> CONSTANT
<type[(scale)]>;
```

  - Ví dụ: heso CONSTANT NUMBER(3,2):= 1.86;

Khai báo biến và hằng PL/SQL

- Gán biến và biểu thức:

```
<variable_name>:= <expression]>;
```

  - Ví dụ:
    - luong:=luong+luong*10/100;

      kq BOOLEAN;

      kq:= mucluong>350000;

Lệnh xuất/nhập trong PL/SQL

- Lệnh xuất:
  - Cú pháp: DBMS_OUTPUT.PUT_LINE (‘Nội dung');

  → Lưu ý: trước khi thực hiện lệnh xuất ta phải chạy lệnh SET SERVEROUTPUT ON
- Lệnh nhập: 2 cách để nhập giá trị cho biến
  - Biến thay thế &: dấu & đặt trước biến. Biến được nhập giá trị lúc thực thi

  → Lưu ý: biến kiểu chuỗi, kiểu ngày đặt trong cặp dấu ‘ ’
  - Biến thay thế &&: dấu && đặt trước biến. Giá trị nhập vào được lưu trữ cho những lần sau.

Lệnh xuất/nhập trong PL/SQL

- Ví dụ:

```
SET SERVEROUTPUT ON
DECLARE
        x number;
BEGIN
        x:=&x;
        DBMS_OUTPUT.PUT_LINE('Gia tri
        x =');
        DBMS_OUTPUT.PUT(x);
END;
```

Các mệnh đề trong PL/SQL

- Mệnh đề SELECT:

```
 SELECT  field_name1,  field_name2...
      INTO  variable_name1,
          variable_name2...
      [cursor_var]

FROM table_name1, table_name2...

[WHERE condition1, condition2... ]
```

Các mệnh đề trong PL/SQL

- Mệnh đề SELECT:
  - Ví dụ:

```
SET SERSEROUTPUT ON;
DECLARE
    mach cuahang.mach%TYPE;
    tench cuahang.tench%TYPE;
BEGIN
    SELECT mach, tench INTO mach, tench
    FROM cuahang WHERE mach = 22;
    DBMS_OUTPUT.PUT_LINE(mach || ' : ' || tench );
END ;
```

Các mệnh đề trong PL/SQL

- Mệnh đề INSERT/ UPDATE/ DELETE:

INSERT INTO table_name[(field_name1, field_name2, …)] VALUES (variable_name1, variable_name2, …);

- Ví dụ:

```
INSERT INTO counts (sales_set, non_sales_set)
VALUES (v_sales_count, v_non_sales);
```

Các mệnh đề trong PL/SQL

- Mệnh đề INSERT/ UPDATE/ DELETE:
  - Ví dụ:

```
DECLARE
    mach NUMERIC(6,2);
    tench VARCHAR2(20);
BEGIN
    mach :=23; tench :='cua hang 114';
    INSERT INTO cuahang(mach, tench)
    VALUES (mach,tench);
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc rẻ nhánh: IF
  - Cú pháp 1:

```
IF <condition1> THEN
    statements1;
ELSE
    IF <condition2> THEN
            statements2;
    ELSE
    …;
    END IF;
END IF;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc rẻ nhánh: IF
  - Cú pháp 1:
    - Ví dụ:

```
IF n=1 THEN
        ngay :='Sunday';
ELSE
        IF n=2 THEN
                ngay :='Monday';
        END IF;
END IF;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc rẻ nhánh: IF
  - Cú pháp 2:

```
IF <condition1> THEN
statements1;
ELSIF <condition2> THEN
statements2;
ELSE
statements3
END IF;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc rẻ nhánh: IF
  - Cú pháp 2:
    - Ví dụ:

```
IF n=1 THEN ngay :='Sunday';
ELSIF n=2 THEN ngay :='Monday';
ELSIF n=3 THEN ngay :='Tuesday';
ELSIF n=4 THEN ngay :='Wedsday';
ELSIF n=5 THEN ngay :='Thursday';
END IF;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc rẻ nhánh: CASE

```
CASE selector
WHEN expression1 THEN
        sequence_of_statements1;
...
WHEN expressionN THEN
        sequence_of_statementsN;
ELSE sequence_of_statementsN+1;]
END CASE [label_name];
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc rẻ nhánh: CASE
  - Ví dụ:

```
SET SERVEROUTPUT ON;
DECLARE
      Loai varchar2(10);
BEGIN
       Loai:=&Loai;
       CASE Loai
       WHEN 'G' THEN DBMS_OUTPUT.PUT_LINE('Loai gioi');
       WHEN 'K' THEN DBMS_OUTPUT.PUT_LINE('Loai kha');
       WHEN 'T' THEN DBMS_OUTPUT.PUT_LINE('Loai trung binh');
       ELSE DBMS_OUTPUT.PUT_LINE('Loai yeu');
       END CASE;
END;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc rẻ nhánh: SEARCH CASE

```
CASE selector
WHEN search_condition1 THEN
        sequence_of_statements1;
...
WHEN search_conditionN THEN
        sequence_of_statementsN;
ELSE sequence_of_statementsN+1;]
END CASE [label_name];
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc rẻ nhánh: SEARCH CASE
  - Ví dụ:

```
SET SERVEROUTPUT ON;
DECLARE
        n int;
        Loai varchar2(20);
BEGIN
  n:=&n;
  CASE
  WHEN (n MOD 2=0)and(n>0) THEN Loai:='duong chan';
  WHEN (n MOD 2!=0)AND (n>0)then Loai:='duong le';
  WHEN (n MOD 2=0)and (n<0) THEN Loai:='am chan';
  ELSE Loai:='am le';
  END CASE;
  DBMS_OUTPUT.PUT_LINE(' la so: '||Loai);
END;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc lặp: LOOP
  - Cú pháp :

```
LOOP
        Statement;
        IF <condition> THEN
        ….
        EXIT;
        END IF;
END LOOP;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc lặp: LOOP
  - Ví dụ:

```
SET SERVEROUTPUT ON;
    DECLARE
                    n int;   i int:=0;   s int:=0;
    BEGIN
      n:=&n;
      LOOP
                      IF i>n THEN EXIT;
                      END IF;
                      s:=s+i;      i:=i+1;
       END LOOP;
      DBMS_OUTPUT.PUT_LINE('tong n so nguyen dau tien la: '||s);
    END;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc lặp: FOR
  - Cú pháp:

```
FOR    loop_index
IN [REVERSE] lowest_number  ..  highest_number
LOOP
        executable_statement(s);
END LOOP;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc lặp: FOR
  - Ví dụ:

```
SET SERVEROUTPUT ON;
DECLARE
   n int; s int:=0;
BEGIN
   n:=&n;
   FOR i IN 0..n LOOP
      s:=s+i;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('tong n so nguyen dau tien la: '||
s);
END;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc lặp: WHILE
  - Cú pháp:

```
WHILE condition
LOOP
        statement;
END LOOP;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Cấu trúc lặp: WHILE
  - Ví dụ:

```
SET SERVEROUTPUT ON;
DECLARE
 i int :=1;d int:=0;  n int;
BEGIN
 n:=&n;
 WHILE(i<=n) LOOP
  IF n mod i=0 THEN
          d:=d+1;
  END IF;
  i:=i+1;
 END LOOP;
```

```
 IF d=2 THEN
    DBMS_OUTPUT.PUT_LINE(n||' la
                          so nguyen to');
 ELSIF d!=2 THEN
    DBMS_OUTPUT.PUT_LINE(n||‘
                 khong la so nguyen to');
 END IF;
END;
```

Các cấu trúc điều khiển lập trình trong PL/SQL

- Bài tập: Xét khối lệnh sau

```
SET SERVEROUTPUT ON
DECLARE
    TYPE MY_DEP_TABLE IS TABLE OF
    departments.department_name%TYPE
    INDEX BY BINARY_INTEGER;
    my_dep MY_DEP_TABLE ;
    v_somautin INTEGER := 4 ;
    V_DEPID DEPARTMENTS.DEPARTMENT_ID%TYPE ;
BEGIN
  FOR I IN 1 .. V_SOMAUTIN LOOP
    IF i=1 THEN v_depid:=120 ;
    ELSIF i=2 THEN v_depid:=150 ;
    ELSIF i=3 THEN v_depid:=180 ;
    ELSIF i=4 THEN v_depid:=210 ;
    END IF ;
    SELECT department_name INTO my_dep(i)
    FROM DEPARTMENTS  WHERE DEPARTMENT_ID = V_DEPID ;
  END LOOP ;
  DBMS_OUTPUT.PUT_LINE('STT ' || ' TEN PHONG') ;
  FOR I IN 1 .. V_SOMAUTIN LOOP
    DBMS_OUTPUT.PUT_LINE(I ||' ' || ' '|| MY_DEP(I)) ;
  END LOOP ;
```

Khối lệnh thực hiện công việc gì??????????

Các cấu trúc điều khiển lập trình trong PL/SQL

- Lệnh: GOTO
  - Cú pháp:

```
<<label_name>>
 ...
statement ;
GOTO label_name ;
```

  - label_name: dùng để đặt tên cho một khối lệnh
  - label_name được định nghĩa trong cặp dấu “<< >>”

Các cấu trúc điều khiển lập trình trong PL/SQL

- Lệnh: GOTO
  - Câu lệnh GOTO rẽ nhánh không điều kiện đến một nhãn. Khi thực hiện, câu lệnh GOTO thay đổi luồng điều khiển trong một khối để chuyển đến thực hiện lệnh nằm trong nhãn.
  - GOTO không được phép trong trường hợp:
    - Từ một xử lý ngoại lệ vào trong khối hiện hành.
    - Nhảy ra ngoài chương trình con.

Các cấu trúc điều khiển lập trình trong PL/SQL

- Lệnh GOTO:

```
BEGIN
    <<outer_block>>
    Declare
        <khai báo biến>
    Begin
        <khối lệnh 1>
        GOTO  inner_block
        <khối lệnh 2>
        <<inner_block>>
        Declare
            <khai báo biến>
        Begin
            <khối lệnh 3>
        End;    /*End của <<inner_block>>*/
    End;    /*End của <<outer_block>>*/
END;
```

Khối lệnh 2 không được thực hiện

Các cấu trúc điều khiển lập trình trong PL/SQL

- **Ví dụ 1:**

```
Create Function Test_Block1 (m
    number) return number
As
Begin
    <<BlockA>>
    declare
        x number;
    begin
        x:=m;
        if x=5 then
          GOTO BlockB ;
        else
          GOTO BlockC ;
        end if;
        <<BlockB>>
        declare
            y number;
        Begin
            y:=100;
            return 5;
        End;  /*End cua <<BlockB>>*/
        <<BlockC>>
        return 0;
    end;   /*End cua <<BlockA>>*/
End;
```

Select Test_Block1 (5) from Dual;

=> KQ trả về: 5

Select Test_Block1(10) from Dual;

=> KQ trả về: 0

Kết quả của 2 TH?

Các cấu trúc điều khiển lập trình trong PL/SQL

- Xử lý ngoại lệ: EXCEPTION
  - Được xử lý khi một lỗi phát sinh, chương trình được chuyển tới *khối PL/SQL chứa phần xử lý ngoại lệ*.
  - Có 2 dạng ngoại lệ (exception)
    - Không tường minh (implicit)

      VD: Một ngoại lệ được sinh ra khi chia một số cho zero
    - Tường minh (explicit): do người dùng định nghĩa bằng cách sử dụng câu lệnh RAISE

Các cấu trúc điều khiển lập trình trong PL/SQL

```
DECLARE /*nếu là Block ngoài cùng của Function hoặc
        Procedure thì không dùng Declare */
        lỗi_ngoại_lệ EXCEPTION;
        ….
BEGIN
        …
        IF <điều kiện lỗi> THEN
                RAISE lỗi_ngoại_lệ; /*bật ngoại lệ*/
        END IF;
EXCEPTION
        WHEN lỗi_ngoại_lệ THEN
        ….
        WHEN OTHERS THEN
        ….
END;
```

Ví dụ:

```
DECLARE
   Ti_le NUMBER(3,1);
BEGIN
   SELECT price / earnings INTO ti_le FROM stocks
      WHERE symbol = 'XYZ';  -- co the xay ra loi chia cho 0
   INSERT INTO stats (symbol, ratio) VALUES ('XYZ', pe_ratio);
   COMMIT;
EXCEPTION  -- bat dau phan xu ly ngoai le
   WHEN ZERO_DIVIDE THEN  -- xu ly loi chia cho 0
      INSERT INTO stats (symbol, ratio) VALUES ('XYZ', NULL);
      COMMIT;
   WHEN OTHERS THEN  -- xu ly tat ca cac loi khac
      ROLLBACK;
END;  -- ket thuc khoi
```

Ví dụ:

```
Create Function Test_Exception (maso number) return number
As
    trung_ma_so  EXCEPTION;
BEGIN
    IF maso=5THEN
        RAISE  trung_ma_so; /*bat ngoai le*/
    ELSE
        return 2;
    END IF;
EXCEPTION
    WHEN trung_ma_so THEN
        return 1; /*da co ma so nay roi*/
    WHEN OTHERS then   /*sử dụng từ khóa OTHERS cho các lỗi khác past_due,
    việc sử dụng OTHERS đảm bảo không có ngoại lệ nào sẽ không được xử lý*/
        return 0; /*loi phat sinh*/
END;
```

THEN

Select Test_Exception (5) from Dual; ?

=> KQ trả về: 1

Select Test_Exception(10) from Dual; ?

=> KQ trả về: 2

CURSOR_ALREADY_OPEN	Mở một cursor, mà cursor đó đã ở trạng thái đang mở.
DUP_VAL_ON_INDEX	Khi có thao tác INSERT UPDATE vi phạm ràng buộc UNIQUE .
INVALID_CURSOR	Mở cursor chưa tạo,đóng một cursor mà nó chưa được mở.
INVALID_NUMBER	Lỗi chuyển kiểu dữ liệu từ string sang kiểu number.
LOGIN_DENIED	Đăng nhập sai username/password.
NO_DATA_FOUND	Câu lệnh SELECT INTO không trả về dòng nào.
NOT_LOGGED_ON	Một chương trình PL/SQL cần thao tác đến Cơ sở dữ liệu Oracle nhưng lại chưa đăng nhập vào Cơ sở dữ liệu.
PROGRAM_ERROR	Một số lỗi chương trình, ví dụ một hàm (function) không chứa mệnh đề RETURN để trả về giá trị.
STORAGE_ERROR	Lỗi bộ nhớ
TIMEOUT_ON_RESOURCE	Lỗi timeout xảy ra khi Oracle đang chờ tài nguyên.
TOO_MANY_ROWS	Câu lệnh SELECT INTO trả về nhiều hơn một dòng.
VALUE_ERROR	Lỗi chuyển kiểu dữ liệu hoặc thao tác vi phạm RBTV.
ZERO_DIVIDE	Lỗi chia một số cho zero.

Các cấu trúc điều khiển lập trình
Bài tập

Trong Database QuanLyXayDung

1.Viết khối lệnh PL/SQL lấy tên công trình với thù lao lớn hơn thù lao cho trước

2.Sử dụng lệnh DEFINE cung cấp thù lao. Nếu thù lao đưa vào cho kết quả nhiều hơn 1 dòng, điều khiển lỗi, xuất thông báo "Có nhiều công trình có mức thù lao cao hơn mức đã cho".

Các cấu trúc điều khiển lập trình
Bài tập

4. Nếu thù lao đưa vào không trả về dòng nào, điều khiển lỗi, xuất thông báo "Không có công trình nào có mức thù lao như yêu cầu"
5. Nếu thù lao đưa vào cho kết quả là một dòng, xuất tên công trình và tên KTS thiết kế công trình cùng với thù lao
6. Nếu xảy ra lỗi khác, xuất thông báo câu "Đã xảy ra lỗi khác"

Kiểu con trỏ trong PL/SQL

- Con trỏ (cursor): là đối tượng liên kết với một tập dữ liệu và cho phép làm việc với từng dòng của tập dữ liệu đó
- Quy trình sử dụng một con trỏ

Khai báo -> mở cursor -> lấy dữ liệu để xử lý -> đóng cursor

Kiểu con trỏ trong PL/SQL

```
--Khai báo một con trỏ
    CURSOR <cursor_name>
    [(<parameter_list>)] IS
    <SELECT statement>;
--Mở con trỏ
    OPEN [<cursor_name>
    [( <parameter_list>)] ];
--Lấy dữ liệu
    FETCH <Tên cursor> INTO <Tên biên>;
--Đóng con trỏ
    CLOSE <Tên cursor>;
```

Kiểu con trỏ trong PL/SQL

- Trong đó:
  - SELECT phải chỉ ra các cột cụ thể cần lấy cho con trỏ này.
  - Trong ngôn ngữ thủ tục PL/SQL, 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 tham khảo trực tiếp.

```
DECLARE
  CURSOR c_cgtrinh IS
    SELECT ten_ctr, diachi_ctr
    FROM cgtrinh;
v_cgtrinh c_cgtrinh%ROWTYPE;
BEGIN
  OPEN c_cgtrinh;
    dbms_output.put_line('ten_ctr | diachi_ctr');
    LOOP
      FETCH c_cgtrinh INTO v_cgtrinh;
        EXIT WHEN c_cgtrinh%NOTFOUND;
        dbms_output.put_line(v_cgtrinh.ten_ctr
            || ' | '
            || v_cgtrinh.diachi_ctr);
    END LOOP;
  CLOSE c_cgtrinh;
END;
```

1. Khai báo cursor để truy vấn dữ liệu trong câu SELECT

Kiểu con trỏ trong PL/SQL

```
DECLARE
  CURSOR c_cgtrinh IS
    SELECT ten_ctr, diachi_ctr
    FROM cgtrinh;
v_cgtrinh c_cgtrinh%ROWTYPE;
BEGIN
  OPEN c_cgtrinh;
    dbms_output.put_line('ten_ctr | diachi_ctr');
    LOOP
      FETCH c_cgtrinh INTO v_cgtrinh;
        EXIT WHEN c_cgtrinh%NOTFOUND;
        dbms_output.put_line(v_cgtrinh.ten_ctr
            || ' | '
            || v_cgtrinh.diachi_ctr);
    END LOOP;
  CLOSE c_cgtrinh;
END;
```

Khai báo biến để truy cập từng mẫu tin trong cursor

Kiểu con trỏ trong PL/SQL

```
DECLARE
  CURSOR c_cgtrinh IS
    SELECT ten_ctr, diachi_ctr
    FROM cgtrinh;
v_cgtrinh c_cgtrinh%ROWTYPE;
BEGIN
  OPEN c_cgtrinh;
    dbms_output.put_line('ten_ctr | diachi_ctr');
    LOOP
      FETCH c_cgtrinh INTO v_cgtrinh;
        EXIT WHEN c_cgtrinh%NOTFOUND;
        dbms_output.put_line(v_cgtrinh.ten_ctr
           || ' | '
           || v_cgtrinh.diachi_ctr);
    END LOOP;
  CLOSE c_cgtrinh;
END;
```

2. Mở cursor

Kiểu con trỏ trong PL/SQL

```
DECLARE
  CURSOR c_cgtrinh IS
    SELECT ten_ctr, diachi_ctr
    FROM cgtrinh;
v_cgtrinh c_cgtrinh%ROWTYPE;
BEGIN
  OPEN c_cgtrinh;
    dbms_output.put_line('ten_ctr | diachi_ctr');
    LOOP
      FETCH c_cgtrinh INTO v_cgtrinh;
        EXIT WHEN c_cgtrinh%NOTFOUND;
        dbms_output.put_line(v_cgtrinh.ten_ctr
            || ' | '
            || v_cgtrinh.diachi_ctr);
    END LOOP;
  CLOSE c_cgtrinh;
END;
```

3. Lấy mẩu tin trong cursor

Kiểu con trỏ trong PL/SQL

```
DECLARE
  CURSOR c_cgtrinh IS
    SELECT ten_ctr, diachi_ctr
    FROM cgtrinh;
v_cgtrinh c_cgtrinh%ROWTYPE;
BEGIN
  OPEN c_cgtrinh;
    dbms_output.put_line('ten_ctr | diachi_ctr');
    LOOP
      FETCH c_cgtrinh INTO v_cgtrinh;
        EXIT WHEN c_cgtrinh%NOTFOUND;
        dbms_output.put_line(v_cgtrinh.ten_ctr
           || ' | '
           || v_cgtrinh.diachi_ctr);
    END LOOP;
  CLOSE c_cgtrinh;
END;
```

Thoát khỏi vòng lặp khi con trỏ chỉ tới empty

Kiểu con trỏ trong PL/SQL

```
DECLARE
  CURSOR c_cgtrinh IS
    SELECT ten_ctr, diachi_ctr
    FROM cgtrinh;
v_cgtrinh c_cgtrinh%ROWTYPE;
BEGIN
  OPEN c_cgtrinh;
    dbms_output.put_line('ten_ctr | diachi_ctr');
    LOOP
      FETCH c_cgtrinh INTO v_cgtrinh;
        EXIT WHEN c_cgtrinh%NOTFOUND;
        dbms_output.put_line(v_cgtrinh.ten_ctr
           || ' | '
           || v_cgtrinh.diachi_ctr);
    END LOOP;
  CLOSE c_cgtrinh;
END;
```

Các câu lệnh xử lý

Kiểu con trỏ trong PL/SQL

```
DECLARE
  CURSOR c_cgtrinh IS
    SELECT ten_ctr, diachi_ctr
    FROM cgtrinh;
v_cgtrinh c_cgtrinh%ROWTYPE;
BEGIN
  OPEN c_cgtrinh;
    dbms_output.put_line('ten_ctr | diachi_ctr');
    LOOP
      FETCH c_cgtrinh INTO v_cgtrinh;
        EXIT WHEN c_cgtrinh%NOTFOUND;
        dbms_output.put_line(v_cgtrinh.ten_ctr
           || ' | '
           || v_cgtrinh.diachi_ctr);
    END LOOP;
  CLOSE c_cgtrinh;
END;
```

Đóng cursor

Kiểu con trỏ trong PL/SQL
Các thuộc tính

THUỘC TÍNH	DiỄN GiẢI
%ISOPEN	Trả về giá trị TRUE nếu con trỏ đang mở
%NOTFOUND	Trả về giá trị TRUE nếu lệnh FETCH vừa thực hiện trả về rỗng
%FOUND	Trả về giá trị TRUE nếu lệnh FETCH vừa thực hiện trả về khác rống
%ROWCOUNT	Trả về số mẫu tin đã được lệnh FETCH truy xuất

Kiểu con trỏ trong PL/SQL
Các mệnh đề trong Cursor

- SELECT FOR UPDATE
  - Cho phép khóa các mẫu tin đang truy xuất để tiến hành xử lý
  - Các mẫu tin tự động bị khóa khi mở Cursor và khóa khi được COMMIT/ ROLLBACK

```
SELECT ...
FROM ...
FOR UPDATE [OF column_reference];
```

Kiểu con trỏ trong PL/SQL
Các mệnh đề trong Cursor

- SELECT FOR UPDATE

```
DECLARE
    CURSOR c_cgtrinh IS
        SELECT ten_ctr, diachi_ctr
        FROM cgtrinh
        FOR UPDATE OF ten_ctr;
```

Kiểu con trỏ trong PL/SQL
Các mệnh đề trong Cursor

- WHERE CURRENT OF
  - Tiếp nối mệnh đề FOR UPDATE
  - Chỉ định câu lệnh update/delete thay đổi dữ liệu trên những dòng của Cursor hiện hành có định nghĩa FOR

```
[UPDATE statement | DELETE statement] ...
WHERE CURRENT OF cursor_name;
```

Kiểu con trỏ trong PL/SQL
Các mệnh đề trong Cursor

- WHERE CURRENT OF

```
DECLARE
    CURSOR c_cgtrinh IS
            SELECT ten_ctr, diachi_ctr
            FROM cgtrinh
            FOR UPDATE OF ten_ctr;

.....................................

        update cgtrinh
        set ten_ctr = 'du an cau Cantho'
        where current of c_cgtrinh;
```

Kiểu con trỏ trong PL/SQL
Bài tập

- Xét khối PL/SQL:

```
SET VERIFY OFF
-----------------------------------------------------------------------------------------------------------
DECLARE
v_num      NUMBER(3) := &p_num;  -- bien nhap tu ban phim
v_sal      employees.salary%TYPE;  -- khai bao kieu trung voi mot cot
CURSOR emp_cursor IS   -- dat ten cho cursor
    SELECT distinct salary  -– 3 dong in dam nay la cau lenh SQL
    FROM          employees
    ORDER BY salary DESC;
BEGIN
-----------------------------------------------------------------------------------------------------------
OPEN emp_cursor; - thuc thi truy van moi duoc tai
-----------------------------------------------------------------------------------------------------------
FETCH emp_cursor INTO v_sal; - lay gia tri dong hien tai gan vao bien v_sal
WHILE emp_cursor%ROWCOUNT <= ? AND emp_cursor%FOUND LOOP  - vong lap while
    INSERT INTO top_dogs (salary)
    VALUES (v_sal);
    FETCH emp_cursor INTO v_sal;  - gan lai gia tri v_sal moi
END LOOP;
-----------------------------------------------------------------------------------------------------------
CLOSE emp_cursor;
COMMIT;
END;
```

Kiểu con trỏ trong PL/SQL
Bài tập

- EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPNO)
- Viết khối PL/SQL cho phép:
  - Hiển thị tên, lương và mã số nhà quản lý của nhân viên đang làm việc trong phòng ban, với mã số phòng ban được nhập từ bàn phím
  - Tăng lương cho những nhân viên với mức tăng như sau:
    - Nếu lương <5000 thì tăng 25%
    - Nếu lương từ 5000 đến 10000 thì tăng 20%
    - Nếu lương từ 10000 đến 15000 thì tăng 15%
    - Ngoài ra thì tăng 12%

Chương 5:

FUNCTION - PROCEDURE - PACKAGE - TRIGGER

Nội dung

- PROCEDURE
- FUNCTION
- PACKAGE
- TRIGGER

PROCEDURE

- Đưa một khối lệnh thực hiện một chức năng vào một procedure
- Được biên dịch thành p-code nâng cao khả năng thực hiện
- Tăng khả năng xử lý, xử dụng chung, tái sử dụng, có tính bảo mật và an toàn dữ liệu cao
- Không có giá trị trả về

PROCEDURE

- Cú pháp:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(<param1> [IN | OUT | IN OUT] <datatype>
[,<param2> [IN | OUT | IN OUT] <datatype>[DEFAULT <default_value>]) ]
IS
        [<variable_name><datatype>
                [NULL | NOT NULL] [DEFAULT <default_value>] ;]
BEGIN
        [<statement>;] -- Đây là khối lệnh PL/SQL trong chương trình
        [EXCEPTION  --Phần ngoại lệ (nếu có)
         WHEN <exception_name> THEN
         [<statement>;] ]
END;
```

PROCEDURE

Ví dụ: **NHANVIEN(MANV, TENNV, NGAYSINH, MAPHONG, LUONG, MA_NQL)**

```
CREATE OR REPLACE
        PROCEDURE Tang_Luong
AS
      luong_cu Float;
      luong_moi float;
BEGIN
  SELECT LUONG INTO luong_cu FROM
        NHANVIEN WHERE MANV=123;
  IF SQL%FOUND THEN
luong_moi:=luong_cu+
                luong_cu*10/100;
        UPDATE NHANVIEN SET
                LUONG=luong_moi
        WHERE MANV=123;
      IF SQL%ROWCOUNT<>0
      THEN
DBMS_OUTPUT.PUT_LINE
            ('Luong  NV 123 duoc tang');
       END IF;
  END IF;
  EXCEPTION
  WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE
            ('Khong  tim thay nhan vien');
END;
```

Lưu ý

- Trong Procedure và Function:
  - Không chấp nhận kiểu các dữ liệu giới hạn độ rộng (vd: varchar2(n), number(n)…) trong tham số vào và trong trị trả về.
  - Không thể áp dụng cho các điều kiện thực hiện trên nhóm (mệnh đề GROUP).

FUNCTION

- Giống như Procedure nhưng trả về một giá trị ngay vị trí gọi
- Tham số sử dụng trong hàm chỉ có thể là loại IN, không chấp nhận giá trị OUT hay giá trị IN OUT.
- Không cho phép hàm trả về kiểu dữ liệu như RECORD, TABLE

FUNCTION

- Cú pháp:

```
CREATE [OR REPLACE] FUNCTION function_name
[(<param1> [IN] <datatype>
[,<param2> [IN] <datatype>[DEFAULT <default_value>]) ]
RETURN <datatype> IS
        [<variable_name><datatype>
                [NULL | NOT NULL] [DEFAULT <default_value>] ;]
BEGIN
        [<statement>;] -- Đây là khối lệnh PL/SQL trong chương trình
        RETURN <return_value>;
        [EXCEPTION  --Phần ngoại lệ (nếu có)
         WHEN <exception_name> THEN
         [<statement>;] ]
END;
```

FUNCTION
Ví dụ

```
CREATE OR REPLACE FUNCTION
  CountCredits (p_ID IN lecturer.ID%TYPE)
  RETURN NUMBER AS
  v_TotalCredits NUMBER := 0;
                    --Total number of credits
  v_CourseCredits NUMBER;
                    -- Credits for one course
  CURSOR c_RegisteredCourses IS
  SELECT department, course
 FROM myStudent
  WHERE student_id = p_ID;
BEGIN
FOR v_CourseRec  IN c_RegisteredCourses
LOOP
SELECT num_credits INTO v_CourseCredits
FROM session
WHERE department = v_CourseRec. Department
 AND course = v_CourseRec.course;
 v_TotalCredits := v_TotalCredits
                    + v_CourseCredits;
END LOOP;
RETURN v_TotalCredits;
END CountCredits;
```

FUNCTION

Ví dụ: **EMP**(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)

```
CREATE OR REPLACE Function FindEmpSal
        ( name_in IN varchar2 )
RETURN number
IS mySalary number;
   CURSOR c1 IS
      SELECT sal
      FROM emp WHERE ename = name_in;
```

```
BEGIN
      OPEN c1;
      Fetch c1 Into mySalary;
      IF c1%Notfound Then
            mySalary := 0;
      END IF;
      CLOSE c1;
      RETURN mySalary;
END;
```

Bài tập

1. CHINHANH(MACN, TENCN, DIACHI, DIENTHOAI)
2. NHANVIEN(MANV, TENNV, HOTEN, NGAYVAOLAM, LUONG, HOAHONG, MANQL)
3. KHOHANG(MAKHO, TENKHO, DCKHO, DTKHO, MACN, MANV)
4. KHACHHANG(MAKH, TENKH, DTKH, DCKH)
5. SANPHAM(MASP, TENSP, MOTA, GIABAN, DVT)
6. TONKHO(MASP, MAKHO, NGAYKIEM, SLTON)
7. DONHANG(MADH, MAKH, NGAYDH, NGAYGH, MANV)
8. CHITIETDH(MADH, MASP, DONGIA, SL)

Bài tập

1. Viết hàm cho biết tổng trị giá của đơn hàng (đơn giá*số lượng) có mã đơn hàng là tham số truyền vào
2. Tạo thủ tục cập nhật hoa hồng của nhân viên với điều kiện những đơn hàng có tổng tiền <=50.000.000 thì cập nhật hoa hồng là 10%. Những đơn hàng có tổng tiền từ 50.000.000 đến 15.000.000 thì cập nhật hoa hồng là 15%. Những đơn hàng có tổng tiền >200.000.000 thì cập nhật hoa hồng là 20%. Còn lại cập nhật hoa hồng của nhân viên là 0%.

Giải Bài tập 1

```
CREATE OR REPLACE FUNCTION TTDH
                                        (MA_HD IN CHITIETDH.MAHD%TYPE)
RETURN NUMBER
IS
    Tongtrigia NUMBER;
BEGIN
                    SELECT SUM(SL*GIABAN) INTO Tongtrigia FROM
                CHITIETDH
                    WHERE MAHD=MA_HD;
    RETURN Tongtrigia;
END;
```

Giải Bài tập 2

```
CREATE OR REPLACE PROCEDURE CAPNHAT_HH
IS
        CURSOR CR IS SELECT MANV,HOAHONG
         FROM NHANVIEN
        FOR UPDATE;
        CW CR%ROWTYPE;
        SOLUONG NUMBER;
BEGIN
          OPEN CR;
          LOOP
          FETCH CR INTO CW;
          EXIT WHEN CR%NOTFOUND;
          SELECT SUM(SL*GIABAN) INTO SOLUONG
           FROM CHITIETHOADON, HOADON
           WHERE
           CHITIETHOADON.MAHD=HOADON.MAHD
           AND HOADON.MANV=CW.MANV;
```

```
          CASE
            WHEN SOLUONG<500000 THEN
              CW.HOAHONG:=10;
            WHEN SOLUONG BETWEEN
              500000 AND 1000000 THEN
              CW.HOAHONG:=15;
            WHEN SOLUONG>1500000 THEN
              CW.HOAHONG:=20;
            ELSE CW.HOAHONG:=0;
            END CASE;
            UPDATE NHANVIEN SET
              HOAHONG=CW.HOAHONG
            WHERE CURRENT OF CR;
          END LOOP;
          CLOSE CR;
END;
```

PACKAGE

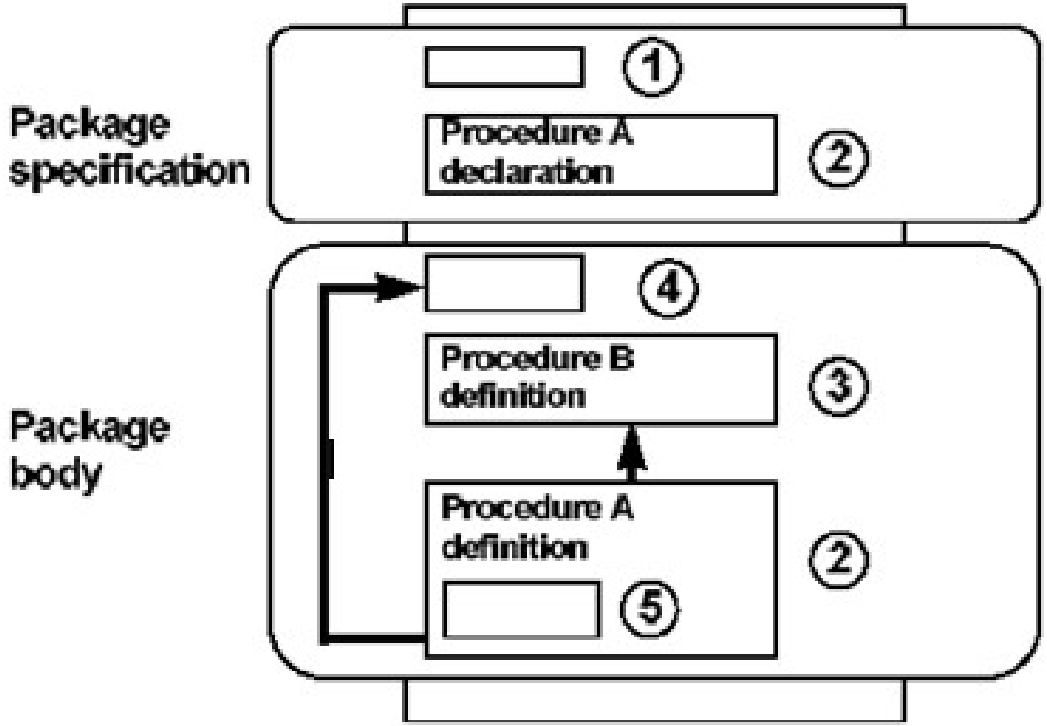
- Là một tập hợp các kiểu dữ liệu, biến lưu giữ giá trị và các thủ tục, hàm có cùng một mối liên hệ với nhau, được gộp chung lại.
- Khi một phần tử trong package được gọi thì toàn bộ nội dung của package sẽ được nạp vào trong hệ thống.
- Nâng cao tốc độ thực hiện lệnh của toàn bộ hàm, thủ tục có trong package.

PACKAGE

- Lợi ích của Package :
  - Tăng tính phân nhỏ của các thành phần
  - Đơn giản trong việc thiết kế ứng dụng
  - Ẩn dấu thông tin
  - Nâng cao hiệu suất sử dụng
  - Thực hiện quá tải

PACKAGE

Cấu trúc của Package



PACKAGE

- Cấu trúc của Package :
  1. Public variable (biến công cộng): là biến mà các ứng dụng bên ngoài có thể tham chiếu tới được.
  2. Public procedure (thủ tục công cộng): bao gồm các hàm, thủ tục của package có thể triệu gọi từ các ứng dụng bên ngoài.
  3. Private procedure (thủ tục riêng phần): là các hàm, thủ tục có trong package và chỉ có thể được triệu gọi bởi các hàm hay thủ tục khác trong package mà thôi.
  4. Global variable (biến tổng thể): là biến được khai báo dùng trong toàn bộ package, ứng dụng bên ngoài tham chiếu được tới biến này.
  5. Private variable (biến riêng phần): là biến được khai báo trong một hàm, thủ tục thuộc package. Nó chỉ có thể được tham chiếu đến trong bản thân hàm hay thủ tục đó.

PACKAGE

- Cú pháp: khai báo phần mô tả Package

```
CREATE [OR REPLACE] PACKAGE
package_name
IS | AS
    public types and item declarations,
            --- Phần khai báo các biến, hằng, cursor, ngoại lệ
và kiểu sử          --- dụng trong toàn bộ package
    subprogram specifications
            --- Khai báo các hàm, thủ tục PL/SQL
END package_name
```

PACKAGE

- Cú pháp: khai báo phần thân Package

```
CREATE [OR REPLACE] PACKAGE BODY
package_name
 IS | AS
    private type and item declarations
        --Khai báo các kiểu chỉ sử dụng riêng trong package
    subprogram bodies
        -- Nội dung của package
END package_name;
```

```
CREATE OR REPLACE PACKAGE
      PCK_NHANVIEN AS
   Procedure THEM_NHANVIEN(luong
number,  manv number);
   Procedure XOA_NHANVIEN(luong
number);
    --thủ tục có thể giao tiếp với bên ngoa
    nv number;
END PCK_NHANVIEN;

CREATE OR REPLACE PACKAGE
BODY
      PCK_NHANVIEN IS
  Procedure THEM_NHANVIEN(luong
                    number,manv number) IS
    luong_moi number;
  BEGIN
      luong_moi:= luong*6;
      INSERT INTO EMPLOYEES
            (employee_id,salary) VALUES
                        (manv,luong_moi);
      nv:=manv;
  END THEM_NHANVIEN;
  PROCEDURE XOA_NHANVIEN(luong
number) IS
   BEGIN
       DELETE FROM EMPLOYEES
        WHERE salary<luong OR
employee_id=nv
    END;
BEGIN---thân của package
       INSERT INTO EMPLOYEES
(employee_id,
       first_name)values(12, 'new');
END PCK_NHANVIEN;
```

PACKAGE

- Ví dụ 1:

```
CREATE OR REPLACE PACKAGE
      PCK_NHANVIEN AS
   Procedure DEM_NHANVIEN(luong number,
   manv number);
   Procedure THEM_NHANVIEN(luong number);
    --thủ tục có thể giao tiếp với bên ngoài
   em number;
END PCK_NHANVIEN;

CREATE OR REPLACE PACKAGE BODY
     PCK_NHANVIEN IS
  Procedure DEM_NHANVIEN(luong
                    number,manv number) IS
     luong_moi number;
```

```
  BEGIN
      luong_moi:= luong*6;
      INSERT INTO EMPLOYEES (employee_id,salary)
      VALUES (manv,luong_moi);
      em:=manv;
  END DEM_NHANVIEN;
  PROCEDURE THEM_NHANVIEN(luong number) IS
   BEGIN
       DELETE FROM EMPLOYEES
        WHERE salary<luong OR employee_id=em;
    END;
BEGIN---thân của package
       INSERT INTO EMPLOYEES (employee_id,
       first_name)values(12, 'new');
END PCK_NHANVIEN;
```

```
SET SEREROUTPUT ON;
CALL PCK_NHANVIEN.THEM_NHANVIEN(90,90);
CALL PCK_NHANVIEN.XOA_NHANVIEN(90);
SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID=90 OR
EMPLOYEE_ID=12
```

PACKAGE

```
create or replace package EMPLOYEEPACKAGE as
  procedure ADDEMPLOYEE(
      P_EMPLOYEEID in EMPLOYEES.EMPLOYEE_ID%type,
      P_FIRST_NAME in EMPLOYEES.FIRST_NAME%type,
      P_LAST_NAME  in EMPLOYEES.LAST_NAME%type);
      E_EMPLOYEENOTREGISTERED EXCEPTION;
      type T_EMPLOYEEIDTABLE is table of
      Employees.employee_id%TYPE INDEX BY BINARY_INTEGER;
  procedure EMPLOYEELIST(P_FIRST_NAME
      in  EMPLOYEES.FIRST_NAME%type,
      p_last_name IN Employees.last_name%TYPE,
      p_IDs OUT t_EmployeeIDTable,
      p_NumEmployees IN OUT BINARY_INTEGER);
  end EMPLOYEEPACKAGE;

  create or replace package body EMPLOYEEPACKAGE as
     procedure ADDEMPLOYEE(
        P_EMPLOYEEID in EMPLOYEES.EMPLOYEE_ID%type,
        P_FIRST_NAME in EMPLOYEES.FIRST_NAME%type,
        p_last_name IN Employees.last_name%TYPE) IS
```

PACKAGE

```
  begin
    insert into EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME)
    VALUES (p_EmployeeID, p_first_name, p_last_name);
    COMMIT;
  end ADDEMPLOYEE;
procedure EMPLOYEELIST(P_FIRST_NAME
  in  EMPLOYEES.FIRST_NAME%type,
  p_last_nameIN  Employees.last_name%TYPE,
  p_IDs OUT t_EmployeeIDTable,
  p_NumEmployees IN OUT BINARY_INTEGER) IS
  v_EmployeeID  Employees.Employee_id%TYPE;
  cursor C_REGISTEREDEMPLOYEES is
    select EMPLOYEE_ID FROM Employees
    WHERE first_name = p_first_name AND last_name = p_last_name;
 BEGIN
   p_NumEmployees := 0; OPEN c_RegisteredEmplcyees;
   LOOP        FETCH c_RegisteredEmployees INTO v_EmployeeID;
     EXIT WHEN c_RegisteredEmployees%NOTFOUND;
      p_NumEmployees := p_NumEmployees + 1;
     p_IDs(p_NumEmployees) := v_EmployeeID;
   END LOOP;
end EMPLOYEELIST;
end EMPLOYEEPACKAGE;
```

PACKAGE
Bài Tập

Tạo package có tên emp_info và thủ tục salary_emp cho biết thông tin nhân viên có lương cao nhất và một hàm có tên là sum_salary tính tổng lương nhân viên các phòng ban với mã phòng ban là tham số truyền vào

TRIGGER

- Là thủ tục được thực hiện ngầm định ngay khi thực hiện lệnh SQL như INSERT, DELETE, UPDATE nhằm đảm bảo các quy tắc logic phức tạp của dữ liệu
- Dùng để khai báo các ràng buộc toàn vẹn phức tạp không thể khai báo ở cấp table như NOT NULL, UNIQUE, PRIMARY KEY, CHECK…
- Chỉ sử dụng khi cần thiết vì có thể gây rối với các hệ thống lớn
- Không cho phép các lệnh quản lý giao tác (COMMIT, ROLLBACK, SAVE POINT…) bên trong thân trigger

TRIGGER

- Phân loại Trigger:
  - BEFORE TRIGGER: Trigger được kích hoạt trước khi thực hiện câu lệnh
  - AFTER TRIGGER: trigger được kích hoạt sau khi thực hiện câu lệnh
  - INSTEAD OF TRIGGER: trigger cho phép người dùng thay đổi một cách trong suốt dữ liệu của một số view

TRIGGER

- Phân loại Trigger:
  - Trigger theo câu lệnh kích hoạt: INSERT, UPDATE, DELETE
  - Trigger mức câu lệnh: trigger được kích hoạt mỗi khi thực hiện câu lệnh
  - Trigger mức dòng dữ liệu: Trigger được kích hoạt nhiều lần ứng với mỗi dòng dữ liệu chịu ảnh hưởng bởi thao tác thực hiện lệnh

TRIGGER

- Cú pháp: Trigger mức câu lệnh

```
CREATE [OR REPLACE] TRIGGER
<trigger_name>
<timing event1> [OR event2 OR event3]
ON <table_name>
BEGIN
    PL/SQL Block;
END;
```

TRIGGER

- Cú pháp: Trigger mức dòng dữ liệu

```
CREATE [OR REPLACE] TRIGGER
<trigger_name>
<timing event1> [OR event2 OR event3]
ON <table_name>
    [REFERENCING OLD AS old| NEW AS new ]
FOR EACH ROW
    [WHEN condition]
BEGIN
    PL/SQL Block;
END;
```

TRIGGER

- Cú pháp: Trigger mức dòng dữ liệu

```
CREATE [OR REPLACE] TRIGGER
<trigger_name>
<timing event1> [OR event2 OR event3]
ON <table_name>
    [REFERENCING OLD AS old| NEW AS new ]
FOR EACH ROW
    [WHEN condition]
BEGIN
    PL/SQL Block;
END;
```

Tên tigger

TRIGGER

- Cú pháp: Trigger mức dòng dữ liệu

```
CREATE [OR REPLACE] TRIGGER
<trigger_name>
timing event1 [OR event2 OR event3]
    [REFERENCING OLD AS old| NEW AS new ]
FOR EACH ROW
    [WHEN condition]
BEGIN
    PL/SQL Block;
END;
```

Thờ gian kích họat tigger: BEFORE/AFTER

TRIGGER

- Cú pháp: Trigger mức dòng dữ liệu

```
CREATE [OR REPLACE] TRIGGER
<trigger_name>
<timing event1> [OR event2 OR event3]
    [REFERENCING OLD AS old| NEW AS new ]
FOR EACH ROW
    [WHEN condition]
BEGIN
    PL/SQL Block;
END;
```

Loại câu lệnh kích họat tigger:

INSERT/UPDATE/DELETE

TRIGGER

- Cú pháp: Trigger mức dòng dữ liệu

```
CREATE [OR REPLACE] TRIGGER
<trigger_name>
<timing event1> [OR event2 OR event3]
ON <table_name>
                                        new ]
FOR EACH ROW
    [WHEN condition]
BEGIN
    PL/SQL Block;
END;
```

Tên bảng có gắn trigger trên đó

TRIGGER

- Cú pháp: Trigger mức dòng dữ liệu

```
CREATE [OR REPLACE] TRIGGER
<trigger_name>
<timing event1> [OR event2 OR event3]
ON <table_name>
    [REFERENCING OLD AS old| NEW AS new ]
    [WHEN condition]
BEGIN
    PL/SQL Block;
END;
```

Tên biến thay thế cho giá trị trước và sau thay đổi của dòng dữ liệu đang xử lý

TRIGGER

- Cú pháp: Trigger mức dòng dữ liệu

```
CREATE [OR REPLACE] TRIGGER
<trigger_name>
<timing event1> [OR event2 OR event3]
ON <table_name>
    [REFERENCING OLD AS old| NEW AS new ]
FOR EACH ROW
```

Trigger thuộc loại tác động lên từng dòng dữ liệu

```
BEGIN
    PL/SQL Block;
END;
```

TRIGGER

- Cú pháp: Trigger mức dòng dữ liệu

```
CREATE [OR REPLACE] TRIGGER
<trigger_name>
<timing event1> [OR event2 OR event3]
ON <table_name>
    [REFERENCING OLD AS old| NEW AS new ]
FOR EACH ROW
    [WHEN condition]
    PL/SQL Block;
END;
```

Điều kiện ràng buộc để thực hiện trigger

TRIGGER
Ví dụ

EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)

```
CREATE TRIGGER Tang_Bonus AFTER INSERT O
FOR EACH ROW
DECLARE
    v_sal  EMP.SAL%TYPE;
BEGIN
    IF :new.Sal IS NOT NULL THEN
        v_sal:= :new.Sal*10/100;
        INSERT INTO BONUS (empno, sal) VALUES (:new.MGR,v_sal) ;
    END IF;
END;
```

:new: là bảng tạm dùng để lưu dữ liệu tạm thời trong khi giao tác Insert thực hiện

:old: là bảng tạm dùng để lưu dữ liệu tạm thời trong khi giao tác Update,Delete thực hiện

TRIGGER
Thao Tác Trigger

- Cho phép/ không cho phép kích hoạt Trigger:

  ALTER TRIGGER tên-trigger DISABLE/ ENABLE;

- Không cho phép kích hoạt tất cả các Trigger trên bảng:

  ALTER TABLE table_name DISABLE/ ENABLE ALL TRIGGERS;

- Xóa trigger:

  DROP TRIGGER Tên-trigger;

TRIGGER
Bài Tập

1. Khi thêm mới hoặc cập nhật một nhân viên cần kiểm tra:
   - Lương phải cao hơn thưởng
   - Lương mới cập nhật phải cao hơn lương cũ
   - Không được tăng lương quá 10%
1. Khi thêm mới hoặc cập nhật lương cần kiểm tra: Lương nhân viên phải thấp hơn lương người quản lý

TRIGGER
Bài Tập

```
CREATE OR REPLACE TRIGGER T_LUONG
AFTER INSERT OR UPDATE OF SAL,COMM ON EMP
FOR EACH ROW
BEGIN
        ----luong phai nhieu hon thuong
        IF (:NEW.SAL<:NEW.COMM) THEN
          RAISE_APPLICATION_ERROR(-20001,'luong phai lon hon
             thuong');
        ELSIF (:NEW.SAL<:OLD.SAL) THEN
          RAISE_APPLICATION_ERROR(-20002,'luong moi phai cao
             hon luong cu');
        ELSIF (:NEW.SAL>:OLD.SAL*1.1) THEN
          RAISE_APPLICATION_ERROR(-20003,'luong moi khong
             duoc cao qua 10% luong cu');
        END IF;
END;
```