**MỞ ĐẦU**

Hiện nay, công nghệ thông tin là một ngành phát triển không ngừng. Khối lượng dữ liệu cần xử lý dần dần tích lũy nhiều hơn đòi hỏi chúng ta phải xữ lý khối dữ liệu đó. Khái niệm “dữ liệu lớn ” ra đời, theo đó là sự phát triển về khoa học dữ liệu nhằm đáp ứng các nhu cầu cho việc kinh doanh, nghiện cứu khoa học,…Các công nghệ về khai phá dữ liệu lớn được ra đời nhằm giải quyết các vấn đề về dữ liệu và trở thành nhu cầu tất yếu của xã hội.

Do thời gian, kiến thức và kinh nghiệm còn hạn chế nên bài báo cáo còn nhiều thiếu sót, vì vậy em rất mong nhận được những ý kiến góp ý của thầy cô để bài báo cáo có thể hoafan thiện hơn.

Em xin chân thành cảm ơn tới các thầy cô phụ trách học phần: *TS. Nguyễn Thị Thanh Huyền, Ths.Nguyễn Tuấn Dũng và Ths.Nguyễn Danh Tú* đã hướng dẫn, chỉ bảo và tạo điều kiện giúp em có thể hoàn thành học phần và báo cáo môn học.

Em xin chân thành cảm ơn !!!

**CHƯƠNG 1**

**CƠ SỞ DỮ LIỆU LỚN**

1. **Định nghĩa**

*Cơ sở dữ liệu lớn(****Big Data****)*là một thuật ngữ rộng cho việc xử lý một tập hợp dữ liệu rất lớn và phức tạp mà các ứng dụng xử lý dữ liệu truyền thống không xử lý đươc. Dữ liệu lớn bao gồm cac thách thức như phân tích, thu nhập, giám sát dữ liệu, tìm kiếm, chia sẻ, lưu trữ, truyền nhận, trực quan, truy vấn và tính riêng tư. Thuật ngữ này thường chỉ đơn giản đề cập đến việc sử dụng các phân tích dự báo, phân tích hành vi người dùng, hoặc một số phương pháp phân tích dữ liệu tiên tiến khác trích xuất giá trị từ dữ liệu mà ít khi đề cập đến kích thước của bộ dữ liệu.

Dữ liệu lớn thường bao gồm tập hợp dữ liệu vối kích thước vượt xa khả năng của các công cụ phần mềm thông thường để thu thập, hiển thị, quản lý và xử lý dữ liệu trong một thời gian có thể chấp nhận được. Kích thước dữ liệu lớn là một mục tiêu liên tục thay đổi. Như năm 2012 thì phạm vi một vài tá terabytes tới nhiều petabytes dữ liệu. Dữ liệu lớn yêu cầu một tập các kỹ thuật và công nghệ được tích hợp theo hình thức mới để khai phá từ tập dữ

1. **Đặc trưng cơ bản**

Dữ liệu lớn có 5 đặc trưng có bản sau:

* Khối lượng dữ liệu (Volume): Đây là đặc điểm tiêu biểu nhất của dữ liệu lớn, khối lượng dữ liệu rất lớn. Kích cỡ của Big data đang từng ngày tăng lên, và tính đến năm 2012 thì nó có thể nằm trong khoảng vài chục terabyte cho đến nhiều petabyte (1 petabyte = 1024 terabyte) chỉ cho một tập hợp dữ liệu. Dữ liệu truyền thống có thể lưu trữ trên các thiết bị đĩa mềm, đĩa cứng. Nhưng với dữ liệu lớn chúng ta sẽ sử dụng công
* Tốc độ (Velocity): Tốc độ có thể hiểu theo 2 khía cạnh: (a) Khối lượng dữ liệu gia tăng rất nhanh (mỗi giây có tới 72.9 triệu các yêu cầu truy cập tìm kiếm trên web bán hàng của Amazon); (b) Xử lý dữ liệu nhanh ở mức thời gian thực (real-time), có nghĩa dữ liệu được xử lý ngay tức thời ngay sau khi chúng phát sinh (tính đến bằng mili giây). Các ứng dụng phổ biến trên lĩnh vực Internet, Tài chính, Ngân hàng, Hàng không, Quân sự, Y tế- Sức khỏe như hiện nay phần lớn dữ liệu lớn được xử lý real-time. Công nghệ xửxử lý dữ liệu lớn ngày nay đã cho phép chúng ta xử lý tức thì trước khi được lưu trữ vào cơ sở dữ liệu.
* Đa dạng (Variety): Đối với dữ liệu truyền thống chúng ta hay nói đến dữ liệu có cấu

trúc, thì ngày nay hơn 80% dữ liệu được sinh ra là phi cấu trúc (tài liệu, blog, hình ảnh, video, bài hát, dữ liệu từ thiết bị cảm biến vật lý, thiết bị chăm sóc sức khỏe . . . ). Big data cho phép liên kết và phân tích nhiều dạng dữ liệu khác nhau. Ví dụ, với các bình luận của một nhóm người dùng nào đó trên Facebook với thông tin video được chia sẻ.

* Độ tin cậy/chính xác (Veracity): Một trong những tính chất phức tạp nhất của Dữ liệu lớn là độ tin cậy/chính xác của dữ liệu. Với xu hướng phương tiện truyền thông xã hội (Social Media) và mạng xã hội (Social Network) ngày nay và sự gia tăng mạnh mẽ tính tương tác và chia sẻ của người dùng Mobile làm cho bức tranh xác định về độ tin cậy và chính xác của dữ liệu ngày một khó khăn hơn. Bài toán phân tích và loại bỏ dữ liệu
* Giá trị (Value): Giá trị là đặc điểm quan trọng nhất của dữ liệu lớn, vì khi bắt đầu triển khai xây dựng dữ liệu lớn thì việc đầu tiên chúng ta cần phải làm đó là xác định được giá trị của thông tin mang lại như thế nào, khi đó chúng ta mới có quyết định có nên triển khai dữ liệu lớn hay không. Nếu chúng ta có dữ liệu lớn mà chỉ nhận được 1% lợi ích từ nó, thì không nên đầu tư phát triển dữ liệu lớn. Kết quả dự báo chính xác thể hiện rõ nét nhất về giá trị của dữ liệu lớn mang lại. Ví dụ, từ khối dữ liệu phát sinh trong quá trình khám, chữa bệnh sẽ giúp dự báo về sức khỏe được chính xác hơn, sẽ

1. **Các ứng dụng cơ sơ dữ liệu lớn**

Báo cáo của Viện nghiên cứu Toàn cầu McKinsey năm 2011 mô tả các thành phần chính

* Các kỹ thuật để phân tích dữ liệu, chẳng hạn như kiểm thử A/B, học máy và xử lý ngôn ngữ tự nhiên
* Công nghệ dữ liệu lớn, như thông tin kinh doanh, điện toán đám mây và cơ sở dữ liệu
* Công cụ trực quan như biểu đồ, đồ thị và các phương pháp thể hiện khác của dữ liệu

*Trong chính chị*

Dữ liệu lớn đã được chứng minh là rất hữu ích trong lĩnh vực chính trị. Nó đóng vai trò lớn trong chiến dịch tái trnh cử thành công năm 2012 của tổng thống Mỹ Barack Obama. Gần đây nhất, phân tích dữ liệu lớn chịu trách nhiệm chính cho BJP và các đồng minh của mình để giành chiến thắng trong cuộc Tổng tuyển cử Ấn Độ rất thành công năm 2014. Chính phủ Ấn Độ sử dụng rất nhiều kỹ thuật để xác định cách thức cử tri Ấn Độ phản ứng với hành động của chính phủ, cũng như ý tưởng cho chính sách tăng cường.

*Trong nông nghiệp*

Các công ty công nghệ sinh học đã sử dụng dữ liệu cảm biến để tối ưu hóa hiệu quả cây trồng. Các công ty này trồng cây thử nghiệm và chạy mô phỏng để đo lường cách thức thực vật phản ứng với những thay đổi trong các điều kiện khác nhau. Các giá trị môi trường được thay đổi, từ nhiệu độ, độ ẩm, lượng nước tười, đất, … Những mô phỏng này cho phép tìm ra môi trường tối ưu cho các loại gen cụ thể.

1. **Cơ sở dữ liệu NoSQL**

* *Tổng quan về cơ sở dữ liệu NoSQL*

Khi làm việc với database, chúng ta đã quá quen với SQLServer, MySQL, PostgreSQL, Oracle,… Điểm chung của những database này là sử dụng ngôn ngữ SQL để truy vấn dữ liệu. Nhưng có 1 dạng database khác với những đặc tính khác biệt được gọi chung dưới cái tên là NoSQL. Thuật ngữ NoSQL được giới thiệu lần đầu vào năm 1998 sử dụng làm tên gọi chung cho các lightweight open source relational database (cơ sở dữ liệu quan hệ mã nguồn mở nhỏ) nhưng không sử dụng SQL cho truy vấn. Vào năm 2009, Eric Evans, nhân viên của Rackspace giới thiệu lại thuật ngữ NoSQL trong một hội thảo về cơ sở dữ liệu nguồn mở phân tán. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ database mới: distributed (phân tán) + non-relational (không ràng buộc). Đây là 2 đặc tính quan trọng nhất.

Cơ sở dữ liệu NoSQL có khả năng lưu trữ dữ liệu với lượng cực lớn, truy vấn dữ liệu với tốc độ cao mà không đòi hỏi quá nhiều về năng lực phần cứng cũng như tài nguyên hệ thống và tăng khả năng chịu lỗi. Đây là những vấn đề mà các relational database không thể giải quyết được. Và cơ sở dữ liệu NoSQL dược sử dụng rộng rãi hiện nay vì lượng dữ liệu mà các hệ thống cần phải

* *Một số đặc điểm:*
* High Scalability: Gần như không có một giới hạn cho dữ liệu và người dùng trên hệ
* High Scalability: Gần như không có một giới hạn cho dữ liệu và người dùng trên hệ
* Atomicity: Độc lập data state trong các operation.
* Consistency: chấp nhận tính nhất quán yếu, có thể không thấy ngay được sự thay đổi
* Durability: dữ liệu có thể tồn tại trong bộ nhớ máy tính nhưng đồng thời cũng được
* Deployment Flexibility: việc bổ sung thêm/loại bỏ các node, hệ thống sẽ tự động nhận biết để lưu trữ mà không cần phải can thiệp bằng tay. Hệ thống cũng không đòi hỏi
* Modeling flexibility: Key-Value pairs, Hierarchical data (dữ liệu cấu trúc), Graphs.
* Query Flexibility: Multi-Gets, Range queries (load một tập giá trị dựa vào một dãy các khóa)
* *Phân loại cơ sở dữ liệu NoSQL*

Có bốn loại chung (loại phổ biến nhất) của cơ sở dữ liệu NoSQL. Mỗi loại đều có các thuộc tính và giới hạn riêng. Không có một giải pháp duy nhất nào tốt hơn tất cả các giải pháp khác, tuy nhiên có một số cơ sở dữ liệu tốt hơn để giải quyết các vấn đề cụ thể. Để làm rõ cơ sở dữ liệu NoSQL, hãy thảo luận các loại phổ biến nhất:

* Key-value stores
* Column-oriented databases (column-family).
* Graph databases
* Document Oriented databases

**CHƯƠNG 2**

**CƠ SỞ DỮ LIỆU PHÂN TÁN**

1. **Tổng quan về cơ sở dữ liệu phân tán**
   1. **Các khái niệm cơ bản**

**Cơ sở dữ liệu phân tán:** Một tuyển tập dữ liệu có quan hệ logic với nhau, được phân bố trên máy tính của một mạng máy tính.

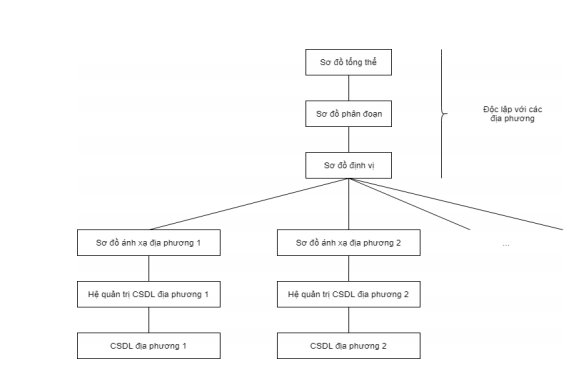
**Hệ quản trị CSDL phân tán:** Hệ thống phần mềm cho phép quản lý CSDL phân tán và đảm bảo tính trong suốt về sự phân tán đối với người dung.

**Ứng dụng cục bộ:** được yêu cầu và thực hiện trên máy tính ở một nút trong hệ CSDL phân tán và chỉ liên quan đến CSDL tại nút đó

**Ứng dụng toàn cục:** yêu câu truy nhập dữ liệu ở nhiều nút thông qua hệ thống truyền thông.

* 1. **Ưu nhược điểm của CSDL phân tán**
* *Ưu điểm*
* Đáp ứng nhanh hầu hết các ứng dụng sử dụng dữ liệu tại trạm.
* Tăng cường các đơn thể ứng dụng và CSDl mà không làm cản trở người sử dụng hiện tại.
* Kiểm soát dữ liệu địa phương theo hướng hoàn thiện sự tích hợp và quản trị dữ liệu từ xa.
* Tăng cường khả năng của hệ thống liên quan đế dư thừa dữ liệu.
* *Nhược điểm*
* Thiết kế cơ sở dữ liệu phức tạp hơn.
* Khó kiểm soát tính toàn vẹn dữ liệu.
* Khó phát hiện, khó thử lỗi.
* Giá thành.
* Bảo mật.
* Thiếu chuẩn mực.
* Thiếu kinh nghiệm.
  1. **Kiến trúc của một hệ CSDL phân tán**

Do sự đa dạng, không có kiến trúc nào được công nhận tương đương với kiến trúc 3 mức ANSI/SPARC. Sơ đồ dưới đây là một kiến trúc cơ bản để tổ chức cho một CSDL phân tán



Hình 1: Sơ đồ tổ chức kiến trúc CSDL phân tán

* **Sơ đồ tổng thể**: Sơ đồ này xác định tất cả các dữ liệu sẽ được lưu trữ trong CSDL phân tán. Sơ đồ tổng thể có thể được định nghĩa một cách chính xác theo cách như trong CSDL không phân tán. Ở đây sẽ sử dụng mô hình quan hệ để hình thành nên sơ đồ này. Sử dụng mô hình này, sơ đồ tổng thể bao gồm định nghĩa của một tập các quan hệ tổng thể.
* **Sơ đồ phân đoạn**: Mỗi quan hệ tổng thể có thể chia thành một vài phần nhỏ hơn không giao nhau được gọi là đoạn (fragments). Có nhiều cách khác nhau để thực hiện việc phân chia này. Sơ đồ tổng thể mô tả các ánh xạ giữa các quan hệ tổng thể và các đoạn được định nghĩa trong sơ đồ phân đoạn. Ánh xạ này là một- nhiều. Có thể có nhiều đoạn liên kết tới một quan hệtổng thể, nhưng mỗi đoạn chỉ liên kết tới nhiều nhất là một quan hệ tổng thể. Các đoạn được chỉ ra bằng tên của quan hệ tổng thể cùng với tên của chỉ mục đoạn.
* **Sơ đồ định vị**: Các đoạn là các phần logic của một quan hệ tổng thể được định vị trên một hoặc nhiều vị trí vật lý trên mạng. Sơ đồ định vị xác định đoạn nào ở các trạm nào. Lưu ý rằng, kiểu ánh xạ được định nghĩa trong sơ đồ định vị quyết định CSDL phân tán là dư thừa hay không. Tất cả các đoạn liên kết với cùng một quan hệ tổng thể R và được định vị tại cùng một trạm j cấu thành ảnh vật lý của quan hệ tổng thể R tại trạm j. Bởi vậy, có thể ánh xạ một-một giữa một ảnh vật lý và một cặp (quan hệ tổng thể, trạm). Các ảnh vật lý có thể được chỉ ra bằng tên của một quan hệ tổng thể và một chỉ mục trạm.
* **Sơ đồ ánh xạ địa phương**: Ánh xạ các ảnh vật lý tới các đối tượng được các hệ quản trị CSDL địa phương thao tác tại các trạm. Ánh xạ này phụ thuộc vào các hệ quản trị CSDL địa phương. Do vậy, trong một hệ thống không đồng nhất, phải có các kiểu ánh xạ địa phương khác nhau tại các

1. **Hệ quản trị cơ sở dữ liệu phân tán**
   1. **Hệ quản trị CSDL phân tán**

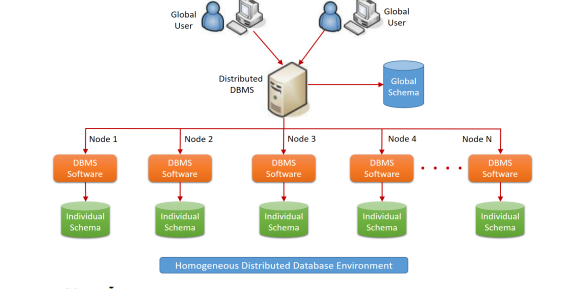
Hệ quản trị CSDL phân tán cung cấp công cụ như tạo lập và quản lý CSDL phân tán. HQTCSDLPT có chức năng hỗ trợ việc tạo và bảo trì CSDL phân tán, chúng có các thành phần tương tự như một hệ quản trị CSDL tập trung và các thành phần hỗ trợ trong việc chuyển tải dữ liệu đến các trạm và ngược lại.

Các thành phần sau đây đòi hỏi một HQTCSDLPT thương mại phải có:

* Quản trị dữ liệu (database management): DB
* Truyền thông dữ liệu (data communication): DC
* Từ điển dữ liệu (data dictionary): DD dùng để mô tả thông tin về sự phân tán của dữ liệu trên mạng.
* Cơ sở dữ liệu phân tán (distributed database): DDB
  1. **Các loại CSDL phân tán**

1. **CSDL phân tán thuần nhất**

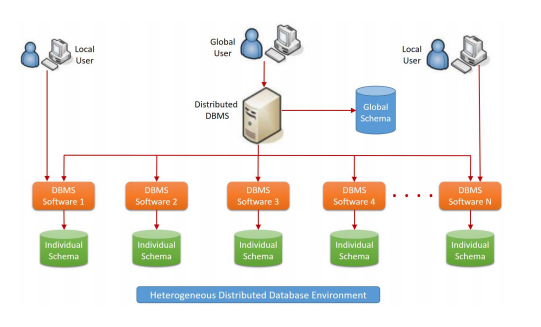
* Tất cả các nút cùng sử dụng một loại hệ quản trị CSDL.
* Thuận lợi cho việc tăng trưởng và cho phép nâng cao hiệu năng.
* Có được bằng cách chia một CSDL thành một tập CSDL cục bộ



Hình 2: Sơ đồ CSDL phân tán thuần nhất

1. **CSDL hỗn tạp**

* Các nút có thể thực hiện trên các hệ quản trị CSDL khác nhau.
* Xảy ra khi các nút đã cài đặt CSDL riêng.
* Có được bằng cách tích hợp các CSDL cục bộ đã có



Hình 3: Cơ sở dữ liệu phân tán hỗn hợp

1. **Kỹ thuật phân đoạn**

Phân hoạch cơ sở dữ liệu thành các đoạn (fragments): sự phân đoạn cho phép phân chia một đối tượng đơn lẻ thành hai hay nhiều mảnh. Thông tin phân đoạn được lưu trữ trong catalog dữ liệu phân tán. Phần mềm xử lý giao tác sẽ truy nhập thông tin ở đây để xử lý các yêu cầu của người dùng.

Việc chia quan hệ tổng thể thành các đoạn có thể thực hiện bằng cách áp dụng các kiểu phân đoạn sau:

* Phân đoạn ngang.
* Phân đoạn dọc.
* Phân đoạn hỗn hợp: là sự kết hợp giữa phân đoạn ngang và phân đoạn dọc.
  1. **Các ràng buộc trong thiết kế phân đoạn**

Một phương pháp thiết kế phân đoạn đúng đắn phải thỏa mãn ba ràng buộc sau:

* **Tính đầy đủ**: Toàn bộ dữ liệu của quan hệ tổng thể phải được ánh xạ vào các đoạn quan hệ và ngược lại. Điều này có nghĩa là, không tồn tại một mục dữ liệu nào thuộc vào quan hệ tổng thể mà không thuộc vào bất kỳ một đoạn nào.
* **Xây dựng lại**: Quan hệ tổng thể có thể được xây dựng lại từ các đoạn mà nó đã tách ra. Điều kiện này là hiển nhiên, bởi vì trong thực tế chỉ có các đoạn được lưu trữ trong CSDL phân tán, và quan hệ tổng thể phải được xây dựng lại thông qua các đoạn khi cần thiết.
* **Tính rời nhau**: các đoạn được tách ra từ quan hệ tổng thể phải là rời nhau. Vì vậy, việc tạo các bản sao phải rõ ràng với các đoạn được chia. Tuy nhiên, điều kiện này chỉ áp dụng chính vào việc phân đoạn ngang, trong khi việc phân đoạn dọc nhiều khi vẫn được phép vi phạm điều kiện này.
  1. **Các phương pháp phân đoạn**

1. **Phân đoạn ngang**

Phân đoạn ngang (horizontal fragmentation partitioning) là tách quan hệ tổng thể R thành các tập con R1, R2, …, Rn. Mỗi tập con chứa một số n\_ bộ của R, điều này rất hữu ích trong CSDL phân tán, nơi mỗi tập con bao gồm các dữ liệu có các thuộc tính địa lý chung. Mỗi n\_bộ thuộc vào một trong các thành phần để có thể khôi phục được quan hệ tổng thể R khi cần thiết. Việc khôi phục quan hệ R được thực hiện bằng phép hợp các quan hệ: R = R1 ∪ R2 ∪ … ∪ Rn

Ta có thể sử dụng các phép toán chọn lọc dựa trên quan hệ tổng thể để tách quan hệ tổng thể

1. **Phân đoạn dọc**

Phân đoạn dọc (vertical fragmentation partitioning) của một quan hệ tổng thể là việc chia các thuộc tính của nó thành các nhóm. Điều này là hữu ích trong các CSDL phân tán, khimà mỗi nhóm các thuộc tính có thể chứa dữ liệu có chung những thuộc tính địa lý.

Việc phân đoạn là đúng đắn nếu như mỗi thuộc tính của quan hệ tổng thể được ánh xạ thành thuộc tính của ít nhất một đoạn con, và phải đảm bảo điều kiện là quan hệ tổng thể có thể được xây dưng lại từ các phân đoạn mà nó chia ra bằng các phép kết nối.

Phân rã theo chiều dọc quan hệ tổng thể R thành các quan hệ Ri: Ri= Chiếu thuộc tính Ri (R)

Quan hệ ban đầu được khôi phục nhờ các phép kết nối tự nhiên:

Việc đưa các khóa của quan hệ tổng thể vào trong mỗi đoạn được tách ra là cách dễ nhất để có

* 1. **Tính trong suốt của CSDL phân tán**

Tính trong suốt của một hệ phân tán được hiểu như là việc che khuất đi các thành phần riêng

biệt của hệ đối với người sử dụng và những người lập trình ứng dụng.

* Các loại trong suốt trong hệ phân tán:
* Trong suốt phân đoạn (fragmentation transparency)
* Trong suốt về vị trí (location transparency)
* Trong suốt ánh xạ địa phương (local mapping transparency)
* Trong suốt nhân bản (replication transparency)
* **Trong suốt phân đoạn**

Trong suốt phân đoạn: là cấp độ cao nhất của mức độ trong suốt, người sử dụng hoặc chương

trình ứng dụng chỉ làm việc trên các quan hệ của cơ sở dữ liệu. Khi dữ liệu đã được phân đoạn thì việc truy cập vào CSDL được thực hiện bình thường như là chưa bị phân tán và không ảnh hưởng tới người sử dụng.

Ví dụ: Xét quan hệ tổng thể NCC (Id, Tên, Tuổi) và các phân đoạn được tách ra từ

nó:

NCC1 (Id, Tên, Tuổi)

NCC2 (Id, Tên, Tuổi)

NCC3 (Id, Tên, Tuổi)

Giả sử DDBMS cung cấp tính trong suốt về phân đoạn, khi đó ta có thể thấy tính trong suốt này được thể hiện như sau:

Khi muốn tìm một người có Id="Id1" thì chỉ cần tìm trên quan hệ tổng thể NCC mà không cần biết quan hệ NCC có phân tán hay không.

* **Trong suốt về vị trí**

Người sử dụng không cần biết về vị trí vật lý của dữ liệu mà có quyền truy cập đến cơ sở dữ liệu tại bất cứ vị trí nào. Các thao tác để lấy hoặc cập nhật một dữ liệu từ xa được tự động thực hiện bởi hệ thống tại điểm đưa ra yêu cầu.

Tính trong suốt về vị trí rất hữu ích, nó cho phép người sử dụng bỏ qua các bản sao dữ liệu đã tồn tại ở mỗi vị trí. Do đó có thể di chuyển một bản sao dữ liệu từ một vị trí này đến một vị trí khác và cho phép tạo các bản sao mới mà không ảnh hưởng đến các ứng dụng.

* **Trong suốt địa phương**

Là một đặc tính quan trọng trong một hệ thống DBMS không đồng nhất. Ứng dụng tham chiếu đến các đối tượng có các tên độc lập từ các hệ thống cục bộ địa phương.

Ứng dụng được cài đặt trên một hệ thống không đồng nhất nhưng được sử dụng như một hệ thống đồng nhất.

* **Trong suốt nhân bản**

Mức trong suốt bản sao liên quan chặt chẽ tới mức trong suốt định vị. Mức trong suốt bản sao có nghĩa là người sử dụng không biết bản sao của đoạn đặt ở vị trí nào. Mức trong suốt bản sao tương đương mức trong suốt định vị. Tuy nhiên, trong những trường hợp thực tế người sử dụng không có mức trong suốt định vị nhưng lại có mức trong suốt bản sao.

**CHƯƠNG 3**

**HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ORACLE**

1. **Lecture 1**
2. **Lecture 2**

**Practice 1:**  Create the DEPT table based on the following table instance chart. Place the syntax in a script called lab\_09\_01.sql, then execute the statement in the script to create the table. Confirm that the table is created.

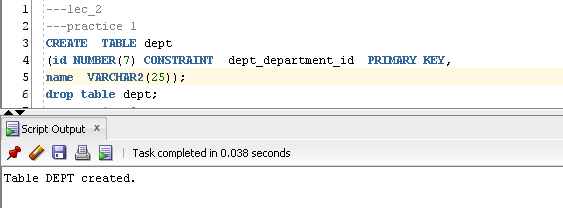
**Mã nguồn**

*CREATE TABLE dept*

*(id NUMBER(7) CONSTRAINT dept\_department\_id PRIMARY KEY,*

*name VARCHAR2(25));*

**Kết quả**



**Practice 2:** Populate the DEPT table with data from the DEPARTMENTS table. Include only columns

that you need.

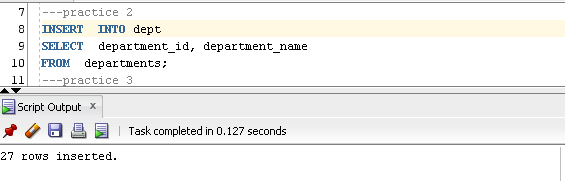
**Mã nguồn**

*INSERT INTO dept*

*SELECT department\_id, department\_name*

*FROM departments;*

**Kết quả**



**Practice 3:** Create the EMP table based on the following table instance chart. Place the syntax in a script called lab\_09\_03.sql, and then execute the statement in the script to create the table.

Confirm that the table is created.

**Mã nguồn**

*CREATE TABLE emp*

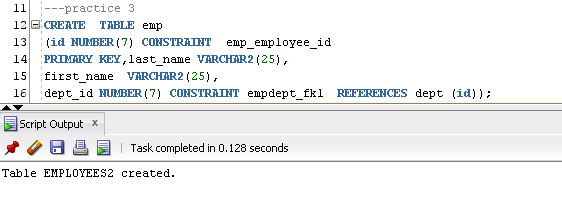
*(id NUMBER(7) CONSTRAINT emp\_employee\_id*

*PRIMARY KEY,last\_name VARCHAR2(25),*

*first\_name VARCHAR2(25),*

*dept\_id NUMBER(7) CONSTRAINT empdept\_fk1 REFERENCES dept (id));*

**Kết quả**



**Practice 4:** Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPARTMENT\_ID columns. Name the columns in your new table ID, FIRST\_NAME, LAST\_NAME,

SALARY, and DEPT\_ID, respectively

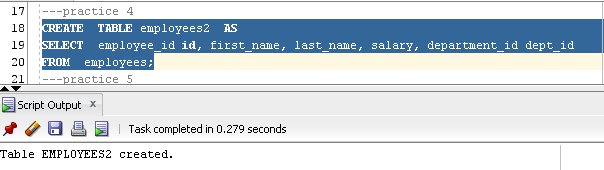
**Mã nguồn**

*CREATE TABLE employees2 AS*

*SELECT employee\_id id, first\_name, last\_name, salary, department\_id dept\_id*

*FROM employees;*

**Kết quả**



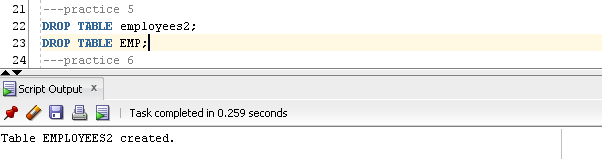
**Practice 5:** Drop the EMP table.

**Mã nguồn**

*DROP TABLE employees2;*

*DROP TABLE EMP;*

**Kết quả**

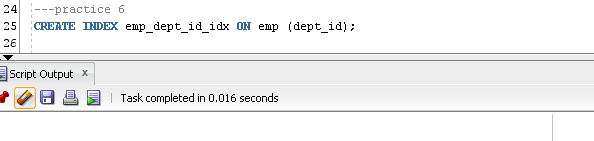


**Practice 6 :** Create a nonunique index on the DEPT\_ID column in the DEPT table.

**Mã nguồn**

*CREATE INDEX emp\_dept\_id\_idx ON emp (dept\_id);*

**Kết quả**



1. **Lecture 3**

**Practice 1:**The staff in the HR department wants to hide some of the data in the EMPLOYEES table. They want a view called EMPLOYEES\_VU based on the employee numbers, employee names, and department numbers from the EMPLOYEES table. They want the heading for the employee name to be EMPLOYEE.

**Mã nguồn**

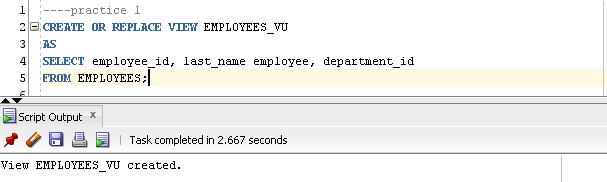
*CREATE OR REPLACE VIEW EMPLOYEES\_VU*

*AS*

*SELECT employee\_id, last\_name employee, department\_id*

*FROM EMPLOYEES;*

**Kết quả**

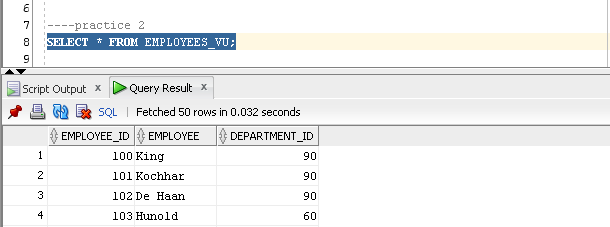


**Practice 2:** Confirm that the view works. Display the contents of the EMPLOYEES\_VU view.

**Mã nguồn**

*SELECT \* FROM EMPLOYEES\_VU;*

**Kết quả**



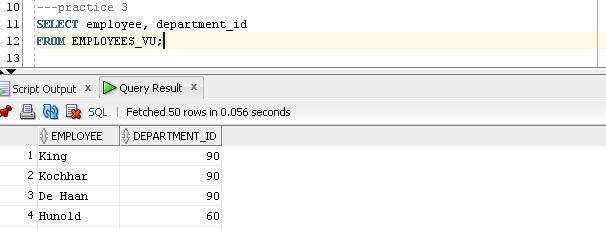
**Practice 3:** Using your EMPLOYEES\_VU view, write a query for the HR department to display all employee names and department numbers.

**Mã nguồn**

*SELECT employee, department\_id*

*FROM EMPLOYEES\_VU;*

**Kết quả**



**Practice 4:**

*•* Department 50 needs access to its employee data. Create a view named DEPT50 that contains the employee numbers, employee last names, and department numbers for all employees in department 50. You have been asked to label the view columns EMPNO, EMPLOYEE, and DEPTNO. For security purposes, do not allow an employee to be reassigned to another department through the view.

*•* Display the structure and contents of the DEPT50 view.

*•* Test your view. Attempt to reassign Mohammed to department 80.

**Mã nguồn**

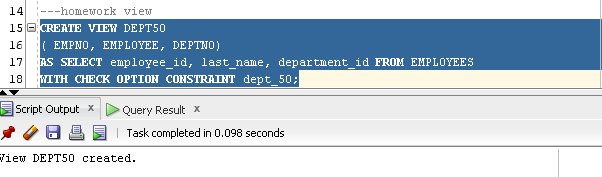
*CREATE VIEW DEPT50*

*( EMPNO, EMPLOYEE, DEPTNO)*

*AS SELECT employee\_id, last\_name, department\_id FROM EMPLOYEES*

*WITH CHECK OPTION CONSTRAINT dept\_50;*

**Kết quả**



**Practice 5:**

*•* You need a sequence that can be used with the primary key column of the DEPT table. The sequence should start at 200 and have a maximum value of 1,000. Have your sequence increment by 10. Name the sequence DEPT\_ID\_SEQ.

*•* To test your sequence, write a script to insert two rows in the DEPT table. Be sure to use the sequence that you created for the ID column. Add two departments: Education and Administration. Confirm your additions. Run the commands in your script.

**Mã nguồn**

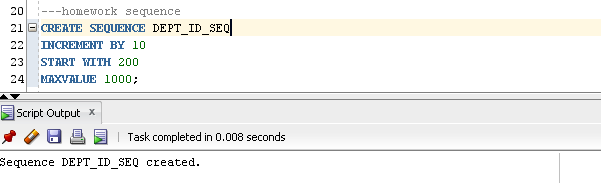
*CREATE SEQUENCE DEPT\_ID\_SEQ*

*INCREMENT BY 10*

*START WITH 200*

*MAXVALUE 1000;*

**Kết quả**

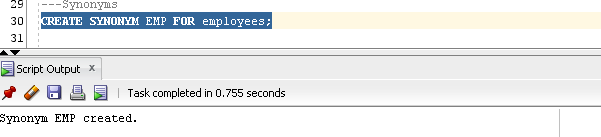


**Practice 6:** Create a synonym for your EMPLOYEES table. Call it EMP.

**Mã nguồn**

*CREATE SYNONYM EMP FOR employees;*

**Kết quả**



1. **Lecture 4**

**Practice 1:**

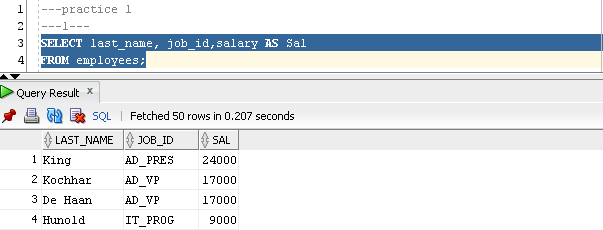
**--1--** The following SELECT statement executes successfully:

*SELECT last\_name, job\_id, salary AS Sal*

FROM employees;

True/False

**Kêt quả**



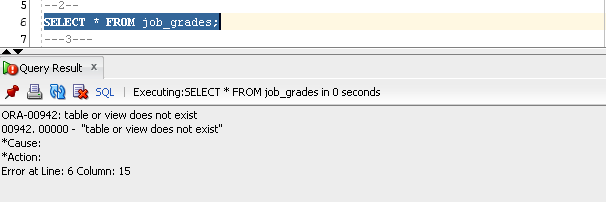
**--2-**- The following SELECT statement executes successfully:

SELECT \*

FROM job\_grades;

True/False

**Kết quả**



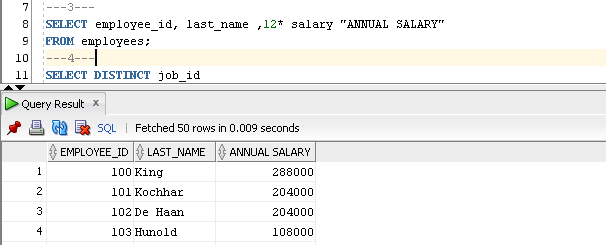
---**3**---There are four coding errors in the following statement. Can you identify them?

SELECT employee\_id, last\_name

sal x 12 ANNUAL SALARY

FROM employees;

**Kết quả**



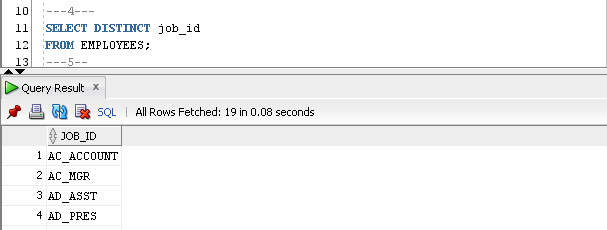
-**--4**---The HR department needs a query to display all unique job codes from the EMPLOYEES table.

**Mã nguồn**

*SELECT DISTINCT job\_id*

*FROM EMPLOYEES;*

**Kết quả**



---**5-**--The HR department has requested a report of all employees and their job IDs. Display

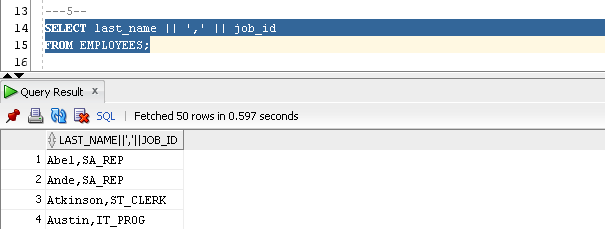
the last name concatenated with the job ID (separated by a comma and space) and name the column Employee and Title.

**Mã nguồn**

*SELECT last\_name || ',' || job\_id*

*FROM EMPLOYEES;*

**Kết quả**



**Practice 2**

**---1---**The HR departments needs to find high-salary and low-salary employees. Display the last name and salary of employees who earn between $5,000 and $12,000 and are in department 20 or 50. Label the columns Employee and Monthly Salary, respectively.

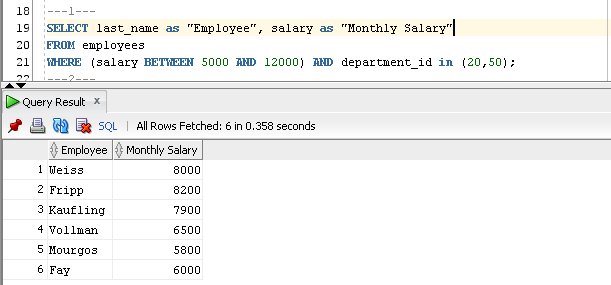
**Mã nguồn**

*SELECT last\_name as "Employee", salary as "Monthly Salary"*

*FROM employees*

*WHERE (salary BETWEEN 5000 AND 12000) AND department\_id in (20,50);*

**Kết quả**



**---2---**Create a report to display the last name, salary, and commission of all employees who earn commissions. Sort data in descending order of salary and commissions.

**Mã nguồn**

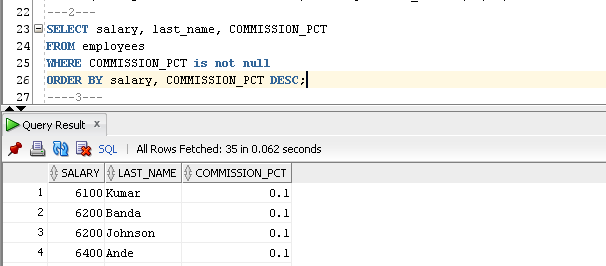
*SELECT salary, last\_name, COMMISSION\_PCT*

*FROM employees*

*WHERE COMMISSION\_PCT is not null*

*ORDER BY salary, COMMISSION\_PCT DESC;*

**Kết quả**



**--3--**Display the last name of all employees who have both an a and an e in their last name.

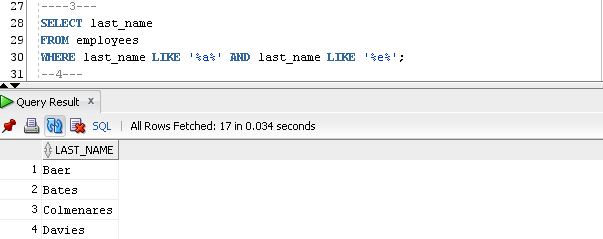
**Mã nguồn**

*SELECT last\_name*

*FROM employees*

*WHERE last\_name LIKE '%a%' AND last\_name LIKE '%e%';*

**Kết quả**



**---4---**Display the last name, job, and salary for all employees whose job is SA\_REP or ST\_CLERK and whose salary is not equal to $2,500, $3,500, or $7,000.

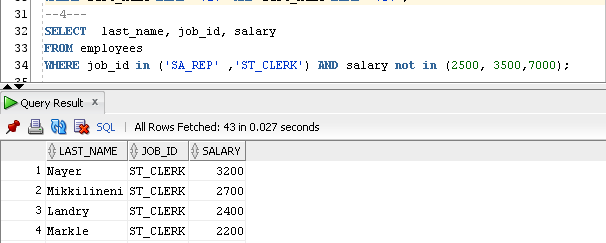
**Mã nguồn**

*SELECT last\_name, job\_id, salary*

*FROM employees*

*WHERE job\_id in ('SA\_REP' ,'ST\_CLERK') AND salary not in (2500, 3500,7000);*

**Kết quả**



1. **Lecture 5**

**Practice 1:**

**---1---**Write a query that displays the last name (with the first letter uppercase and all other letters lowercase) and the length of the last name for all employees whose name starts with the letters J, A, or M. Give each column an appropriate label. Sort the results by the employees’ last names.

**Mã nguồn**

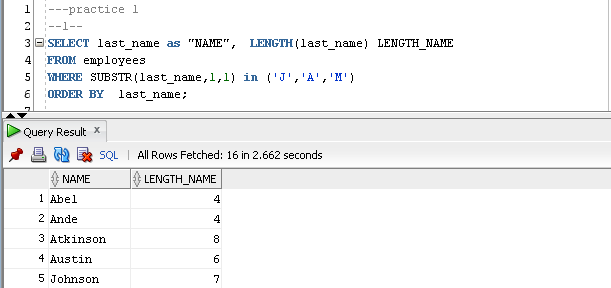
*SELECT last\_name as "NAME", LENGTH(last\_name) LENGTH\_NAME*

*FROM employees*

*WHERE SUBSTR(last\_name,1,1) in ('J','A','M')*

*ORDER BY last\_name;*

**Kết quả**



**---2---**The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number

of months between today and the date on which the employee was hired. Label the column MONTHS\_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

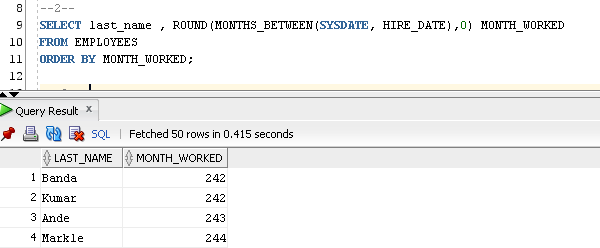
**Mã nguồn**

*SELECT last\_name , ROUND(MONTHS\_BETWEEN(SYSDATE, HIRE\_DATE),0) MONTH\_WORKED*

*FROM EMPLOYEES*

*ORDER BY MONTH\_WORKED;*

**Kết quả**



**---3---**Display each employee’s last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to “Monday, the Thirty-First of July, 2000.”

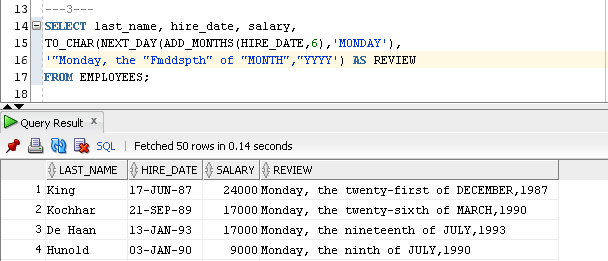
**Mã nguồn**

*SELECT last\_name, hire\_date, salary,*

*TO\_CHAR(NEXT\_DAY(ADD\_MONTHS(HIRE\_DATE,6),'MONDAY'),'"Monday, the "Fmddspth" of "MONTH","YYYY') AS REVIEW*

*FROM EMPLOYEES;*

**Kết quả**



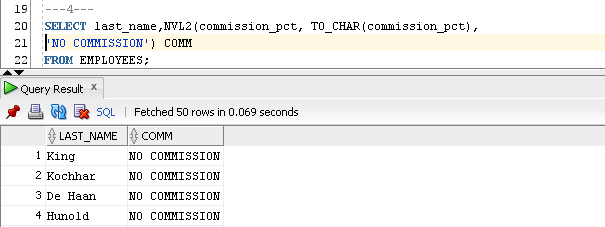
**---4----**Create a query that displays the employees’ last names and commission amounts. If an employee does not earn commission, show “No Commission.” Label the column COMM

**Mã nguồn**

*SELECT last\_name,NVL2(commission\_pct, TO\_CHAR(commission\_pct),'NO COMMISSION') COMM*

*FROM EMPLOYEES;*

**Kết quả**



**---5---**Using the DECODE function, write a query that displays the grade of all employees based on the value of the column JOB\_ID, using the following data:

**Job Grade**

AD\_PRES A

ST\_MAN B

IT\_PROG C

SA\_REP D

ST\_CLERK E

None of the above 0

**Mã nguồn**

*SELECT DECODE (job\_id,'AD\_PRES', 'A',*

*'ST\_MAN','B',*

*'IT\_PROG', 'C',*

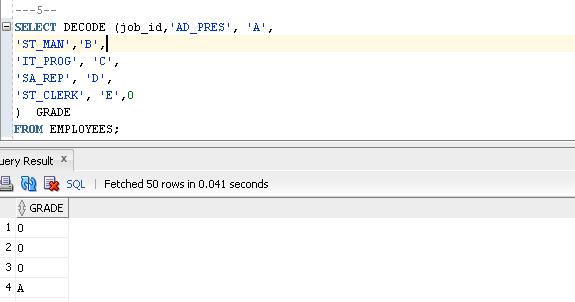
*'SA\_REP', 'D',*

*'ST\_CLERK', 'E',0*

*) GRADE*

*FROM EMPLOYEES*

**Kết quả**



**---6---**Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H when prompted for a letter, then the output should show all employees whose last name starts with the letter H.

**Mã nguồn**

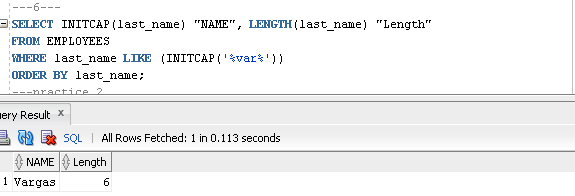
*SELECT INITCAP(last\_name) "NAME", LENGTH(last\_name) "Length"*

*FROM EMPLOYEES*

*WHERE last\_name LIKE (INITCAP('%var%'))*

*ORDER BY last\_name;*

**Kết quả**



**Practice 2**

**---1--** Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number.

**Mã nguồn**

*SELECT ROUND(AVG(salary),0) Average,*

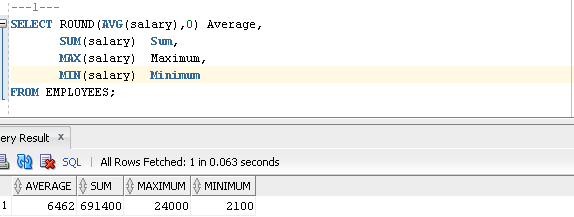
*SUM(salary) Sum,*

*MAX(salary) Maximum,*

*MIN(salary) Minimum*

*FROM EMPLOYEES;*

**Kết quả**



**---2---**Modify the query in Exercise 1 to display the minimum, maximum, sum, and average salary for each job type.

**Mã nguồn**

*SELECT job\_id,ROUND(AVG(salary),0) Average,*

*SUM(salary) Sum,*

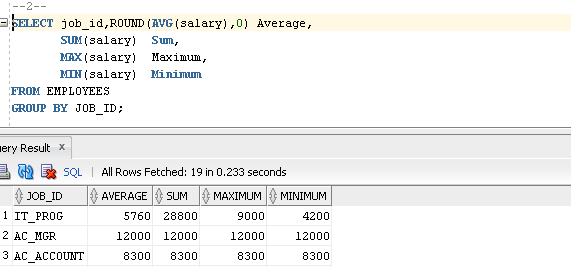
*MAX(salary) Maximum,*

*MIN(salary) Minimum*

*FROM EMPLOYEES*

*GROUP BY JOB\_ID;*

**Kết quả**



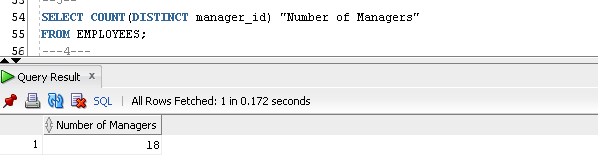
**---3---**Determine the number of managers without listing them. Label the column Number of Managers. Hint: Use the MANAGER\_ID column to determine the number of managers.

**Mã nguồn**

*SELECT COUNT(DISTINCT manager\_id) "Number of Managers"*

*FROM EMPLOYEES;*

**Kết quả**



**---4----**Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.

**Mã nguồn**

*SELECT manager\_id, MIN(salary)*

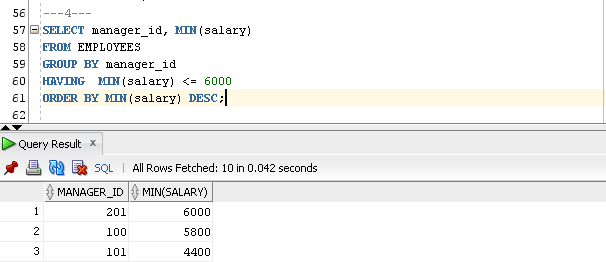
*FROM EMPLOYEES*

*GROUP BY manager\_id*

*HAVING MIN(salary) <= 6000*

*ORDER BY MIN(salary) DESC;*

**Kết quả**



1. **Leture 6**

**Practice 1**

**---1---** The HR department needs a report of all employees. Write a query to display the last name, department number, and department name for all employees.

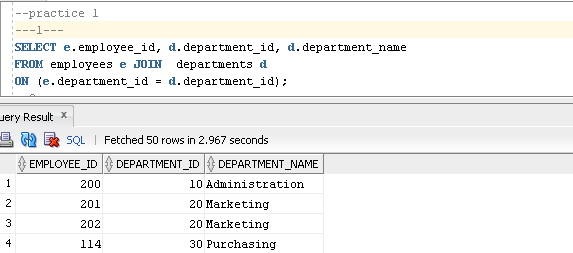
**Mã nguồn**

*SELECT e.employee\_id, d.department\_id, d.department\_name*

*FROM employees e JOIN departments d*

*ON (e.department\_id = d.department\_id)***;**

**Kết quả**



**---2A---**

*•* Create a report to display employees’ last name and employee number along with their manager’s last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively.

*•* Modify Part A to display all employees including King, who has no manager. Order the results by the employee number.

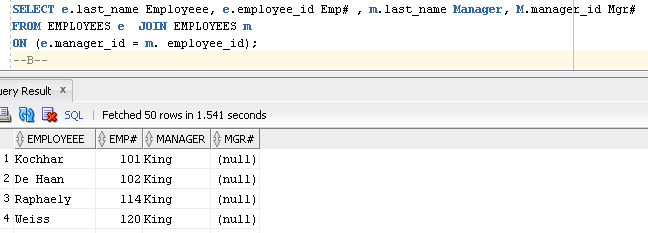
**Mã nguồn**

*SELECT e.last\_name Employeee, e.employee\_id Emp# , m.last\_name Manager, M.manager\_id Mgr#*

*FROM EMPLOYEES e JOIN EMPLOYEES m*

*ON (e.manager\_id = m. employee\_id);*

**Kết quả**



**---2B---**The HR department needs to find the names and hire dates for all employees who were hired before their managers, along with their managers’ names and hire dates.

**Mã nguồn**

*SELECT e.last\_name Employeee, e.employee\_id Emp# , m.last\_name Manager, M.manager\_id Mgr#*

*FROM EMPLOYEES e JOIN EMPLOYEES m*

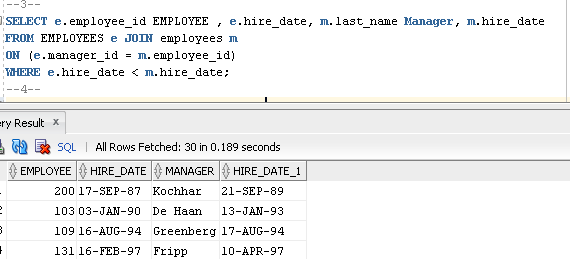
*ON (e.manager\_id = m. employee\_id)*

*WHERE e.last\_name = 'King'*

*and m.manager\_id is null*

*ORDER BY Emp#;*

**Kết quả**



**---3---**Display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains a u.

**Mã nguồn**

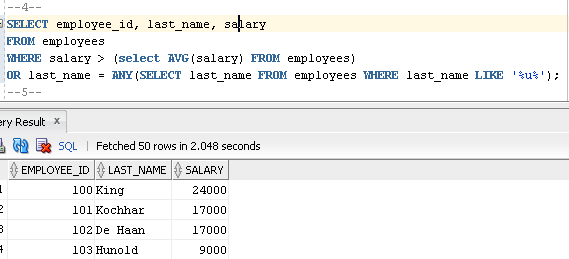
*SELECT employee\_id, last\_name, salary*

*FROM employees*

*WHERE salary > (select AVG(salary) FROM employees)*

*OR last\_name = ANY(SELECT last\_name FROM employees WHERE last\_name LIKE '%u%');*

**Kết quả**



**----4----**The HR department needs a report with the following specifications:

*•* Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department

*•* Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them.

Write a compound query to accomplish this.

**Mã nguồn**

*SELECT last\_name , department\_id*

*FROM employees*

*UNION ALL*

*SELECT TO\_CHAR(Null), department\_id*

*FROM DEPARTMENTS;*

*SELECT DEPARTMENT\_ID, DEPARTMENT\_NAME*

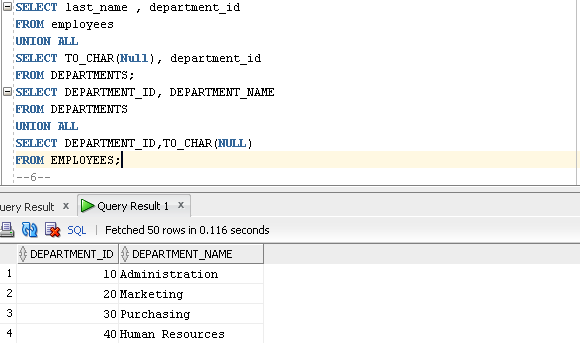
*FROM DEPARTMENTS*

*UNION ALL*

*SELECT DEPARTMENT\_ID,TO\_CHAR(NULL)*

*FROM EMPLOYEES;*

**Kết quả**



---5----Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

**Mã nguồn**

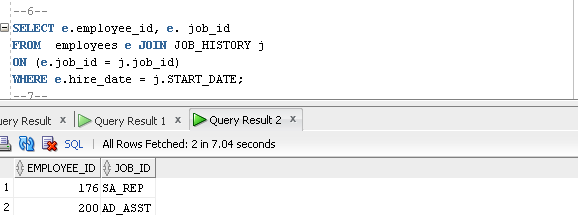
*SELECT e.employee\_id, e. job\_id*

*FROM employees e JOIN JOB\_HISTORY j*

*ON (e.job\_id = j.job\_id)*

*WHERE e.hire\_date = j.START\_DATE;*

**Kết quả**



**---7---**The HR department needs a list of countries that have no departments located in them. Display the country ID and the name of the countries. Use set operators to create this report.

**Mã nguồn**

*SELECT country\_id, country\_name*

*FROM countries*

*MINUS*

*SELECT c.country\_id, c.country\_name*

*FROM countries c*

*JOIN LOCATIONS l ON c.country\_id = l.country\_id*

*JOIN departments d ON d.location\_id = l.location\_id;*

**Kết quả**



1. **Lecture 7**
2. **Lecture 8**

**---1---**Write a query to display the following for those employees whose manager ID is less than

120:

– Manager ID

– Job ID and total salary for every job ID for employees who report to the same manager

– Total salary of those managers

– Total salary of those managers, irrespective of the job IDs

**Mã nguồn**

*SELECT manager\_id, job\_id, Sum(salary) as SUM\_SAL*

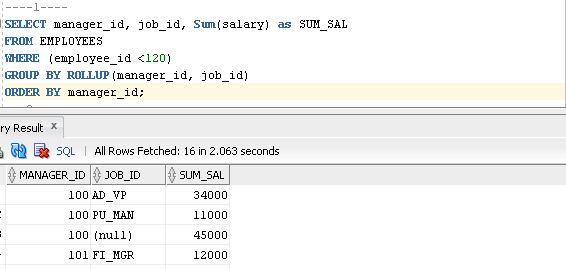
*FROM EMPLOYEES*

*WHERE (employee\_id <120)*

*GROUP BY ROLLUP(manager\_id, job\_id)*

*ORDER BY manager\_id;*

**Kết quả**



**---2---**Observe the output from question 1. Write a query using the GROUPING function to determine whether the NULL values in the columns corresponding to the GROUP BY expressions are caused by the ROLLUP operation.

**Mã nguồn**

*SELECT JOB\_ID,MANAGER\_ID, SUM(SALARY) AS SALARY\_SUM,*

*GROUPING (JOB\_ID) AS GRP\_JOB\_ID,*

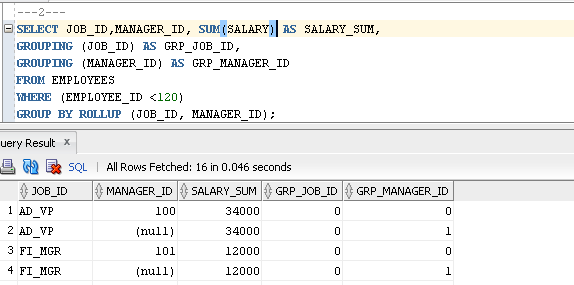
*GROUPING (MANAGER\_ID) AS GRP\_MANAGER\_ID*

*FROM EMPLOYEES*

*WHERE (EMPLOYEE\_ID <120)*

*GROUP BY ROLLUP (JOB\_ID, MANAGER\_ID);*

**Kết quả**



**---3---**Write a query to display the following for those employees whose manager ID is less than 120:

– Manager ID

– Job and total salaries for every job for employees who report to the same manager

– Total salary of those managers

– Cross-tabulation values to display the total salary for every job, irrespective of the

manager

– Total salary irrespective of all job titles

**Mã nguồn**

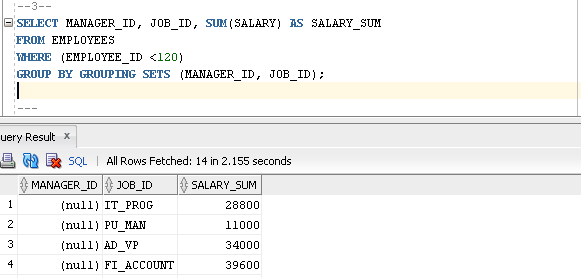
*SELECT MANAGER\_ID, JOB\_ID, SUM(SALARY) AS SALARY\_SUM*

*FROM EMPLOYEES*

*WHERE (EMPLOYEE\_ID <120)*

*GROUP BY GROUPING SETS (MANAGER\_ID, JOB\_ID);*

**Kết quả**



**---4----**Using GROUPING SETS, write a query to display the following groupings:

– department\_id, manager\_id, job\_id

– department\_id, job\_id

– manager\_id, job\_id

The query should calculate the sum of the salaries for each of these groups.

**Mã nguồn**

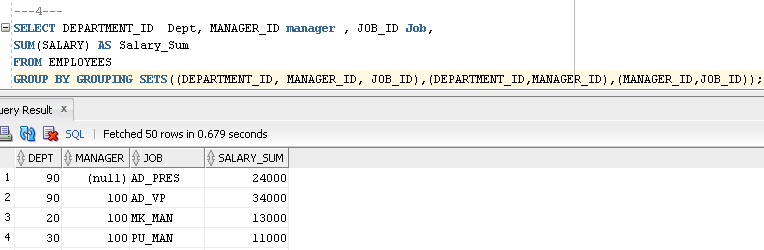
**S***ELECT DEPARTMENT\_ID Dept, MANAGER\_ID manager , JOB\_ID Job,*

*SUM(SALARY) AS Salary\_Sum*

*FROM EMPLOYEES*

*GROUP BY GROUPING SETS((DEPARTMENT\_ID, MANAGER\_ID, JOB\_ID),(DEPARTMENT\_ID,MANAGER\_ID),(MANAGER\_ID,JOB\_ID));*

**Kết quả**



1. **Lectture 9**
2. **Lecture 10**
3. **Lecture 11**

**Practice 1:**

*•* Produce a report showing an organization chart for Mourgos’s department.

*•* Print last names, salaries, and department IDs.

**Mã nguồn**

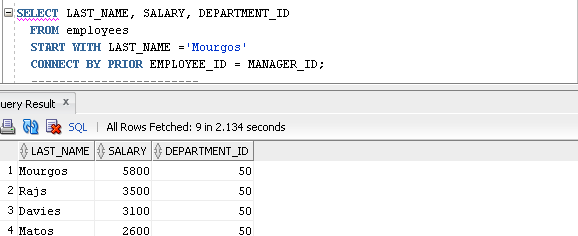
*SELECT LAST\_NAME, SALARY, DEPARTMENT\_ID*

*FROM employees*

*START WITH LAST\_NAME ='Mourgos'*

*CONNECT BY PRIOR EMPLOYEE\_ID = MANAGER\_ID;*

**Kết quả**



**Practice 2:**

*•* Create a report that shows the hierarchy of the managers for the employee Lorentz.

*•* Don’t display Lorentz.

*•* Display his immediate manager first.

**Mã nguồn**

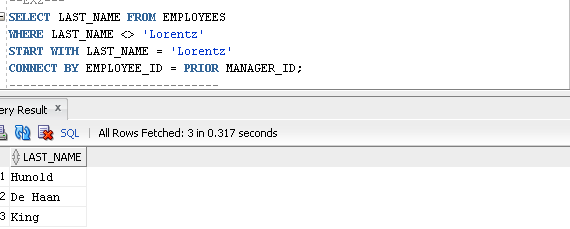
*SELECT LAST\_NAME FROM EMPLOYEES*

*WHERE LAST\_NAME <> 'Lorentz'*

*START WITH LAST\_NAME = 'Lorentz'*

*CONNECT BY EMPLOYEE\_ID = PRIOR MANAGER\_ID;*

**Kết quả**



**Practice 3**

Create an indented report showing the management hierarchy starting from the employee whose LAST\_NAME is Kochhar. Print the employee’s last name, manager ID, and department ID.

**Mã nguồn**

*SELECT LPAD(LAST\_NAME, LENGTH(LAST\_NAME)+ (LEVEL+2)-2,'\_') AS NAME,*

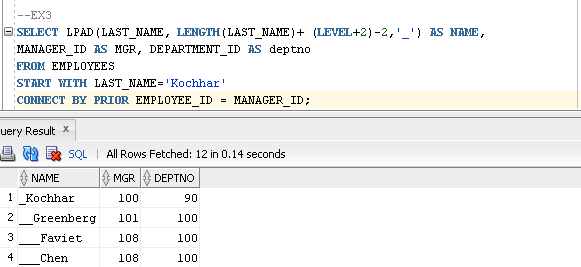
*MANAGER\_ID AS MGR, DEPARTMENT\_ID AS deptno*

*FROM EMPLOYEES*

*START WITH LAST\_NAME='Kochhar'*

*CONNECT BY PRIOR EMPLOYEE\_ID = MANAGER\_ID;*

**Kết quả**



**CHƯƠNG 4**

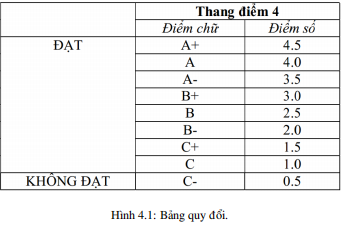
**BÀI TẬP KẾT THÚC MÔN**

1. **Bài 1**

**Đề bài:** Kiểm tra 1 sinh viên đã đủ điều kiện tốt nghiệp chưa biết rằng các điều kiện để một sinh

viên tốt nghiệp là:

1. Tích lũy đủ số tín chỉ
2. Điểm phẩy tốt nghiệp không nhỏ hơn 1.0 biết bảng đổi điểm như sau:



* **Oracle**
* **Mã nguồn**

*create or replace view THONG\_TIN\_SV ("ID","NAME","COURSE\_ID", "YEAR","SEC\_ID","SEMESTER","SEMESTER\_ID","GRADE","POINT","CREDITS")*

*as*

*select t.id, s.name, t.course\_id, t.year, t.sec\_id,t.semester,*

*case*

*when t.semester = 'Spring' then 1*

*else 2*

*end semester\_id, t.grade,*

*case*

*when t.grade = 'A+' then 4.5*

*when t.grade = 'A' then 4.0*

*when t.grade = 'A-' then 3.5*

*when t.grade = 'B+' then 3.0*

*when t.grade = 'B' then 2.5*

*when t.grade = 'B-' then 2.0*

*when t.grade = 'C+' then 1.5*

*when t.grade = 'C' then 1.0*

*when t.grade = 'C-' then 0.5*

*else 0.0*

*end as Point, c.credits*

*from takes t*

*join student s*

*on t.id = s.id*

*join course c*

*on c.course\_id =t.course\_id;*

*-- Tao ham tinh so tin chi tich luy cua sinh vien*

*CREATE OR REPLACE FUNCTION TC\_TICH\_LUY (STUDENT\_ID in STUDENT.ID%TYPE*

*)*

*RETURN number*

*IS tc\_tich\_luy number(3):=0;*

*BEGIN*

*SELECT*

*CASE*

*WHEN SUM(credits) is null then 0*

*ELSE SUM(credits)*

*END INTO tc\_tich\_luy*

*FROM*

*(SELECT course\_id, credits, Max(Point) point*

*FROM (SELECT \* FROM THONG\_TIN\_SV*

*WHERE THONG\_TIN\_SV.ID = STUDENT\_ID )*

*GROUP BY course\_id , credits*

*HAVING Max(Point) > 0.5);*

*RETURN tc\_tich\_luy;*

*END TC\_TICH\_LUY;*

*-- Tao ham tinh diem tich luy cua sinh vien*

*CREATE OR REPLACE FUNCTION DIEM\_TICH\_LUY (STUDENT\_ID IN STUDENT.ID%TYPE)*

*RETURN NUMBER*

*IS diem\_tich\_luy number(3,2):= 0.00;*

*BEGIN*

*SELECT*

*CASE*

*WHEN SUM(credits) is null then 0*

*ELSE SUM(credits\*point)/sum(credits)*

*END into diem\_tich\_luy*

*FROM (*

*SELECT course\_id, credits, Max(Point) point*

*FROM (SELECT \* FROM THONG\_TIN\_SV*

*WHERE THONG\_TIN\_SV.ID = STUDENT\_ID)*

*GROUP BY course\_id, credits);*

*RETURN diem\_tich\_luy;*

*END DIEM\_TICH\_LUY;*

*-- Tao ham kiem tra sinh vien co du dieu kien tot nghiep hay khong?*

*CREATE OR REPLACE PROCEDURE KIEM\_TRA\_DKTN(STUDENT\_ID VARCHAR)*

*IS*

*BEGIN*

*IF(TC\_TICH\_LUY(STUDENT\_ID)>128and DIEM\_TICH\_LUY(STUDENT\_ID) >1.0)*

*then dbms\_output.put\_line('SINH VIEN CO ID: ' ||STUDENT\_ID|| ' DU DIEU KIEN TOT NGHIEP');*

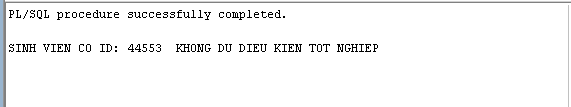
*ELSE DBMS\_OUTPUT.PUT\_LINE ('SINH VIEN CO ID: ' ||STUDENT\_ID||' KHONG DU DIEU KIEN TOT NGHIEP');*

*END IF;*

*END;*

* **Kết quả**

exec KIEM\_TRA\_DKTN('44553');



* **SQL Server**
* **Mã nguồn**

*create view THONG\_TIN\_SV*

*as*

*select t.id, s.name, t.course\_id, t.year, t.sec\_id,t.semester,*

*case*

*when t.semester = 'Spring' then 1*

*else 2*

*end semester\_id, t.grade,*

*case*

*when t.grade = 'A+' then 4.5*

*when t.grade = 'A' then 4.0*

*when t.grade = 'A-' then 3.5*

*when t.grade = 'B+' then 3.0*

*when t.grade = 'B' then 2.5*

*when t.grade = 'B-' then 2.0*

*when t.grade = 'C+' then 1.5*

*when t.grade = 'C' then 1.0*

*when t.grade = 'C-' then 0.5*

*else 0.0*

*end as Point, c.credits*

*from takes t*

*join student s*

*on t.id = s.id*

*join course c*

*on c.course\_id =t.course\_id;*

*-- Tao ham tinh so tin chi tich luy cua sinh vien --*

*CREATE FUNCTION TC\_TICH\_LUY (@STUDENT\_ID varchar(5))*

*RETURNS DECIMAL(3,2)*

*AS*

*BEGIN*

*declare @tc\_tich\_luy decimal(3,2);*

*select @tc\_tich\_luy =*

*CASE*

*WHEN sum(t.credits) is null then 0*

*else sum(t.credits) end int) point*

*from (select \* from THONG\_TIN\_SV*

*where THONG\_TIN\_SV.id = @STUDENT\_ID) s*

*group by course\_id, credits*

*having Max(Point) > 0.5 ) t;*

*return @tc\_tich\_luy;*

*end;*

*-- Tao ham tinh diem tich luy cua sinh vien--*

*create function DIEM\_TICH\_LUY (@STUDENT\_ID varchar(5))*

*returns decimal (3,2)*

*as*

*begin*

*declare @diem\_tich\_luy decimal(3,2);*

*select @diem\_tich\_luy =*

*case*

*when sum(credits) is null then 0*

*else sum(point\*credits)/ sum(credits)*

*end*

*from (select course\_id, credits, Max(Point) point*

*from (select \* from THONG\_TIN\_SV*

*where THONG\_TIN\_SV.id = @STUDENT\_ID) s*

*group by course\_id, credits ) t;*

*return @diem\_tich\_luy;*

*end ;*

*-- Tao ham kiem tra sinh vien co du dieu kien tot nghiep ko? --*

*CREATE PROCEDURE KIEM\_TRA\_DKTN(@STUDENT\_ID varchar(5))*

*as*

*begin*

*declare @tc\_tich\_luy int;*

*declare @diem\_tich\_luy decimal (3,2);*

*declare @kiem\_tra char;*

*set @tc\_tich\_luy = dbo.TC\_TICH\_LUY(@STUDENT\_ID);*

*set @diem\_tich\_luy = dbo.DIEM\_TICH\_LUY(@STUDENT\_ID);*

*if (@tc\_tich\_luy > 128 and @diem\_tich\_luy >= 1.0)*

*print 'SINH VIEN CO ID:' + convert(varchar, @STUDENT\_ID) + ' DU DIEU KIEN TOT NGHIEP';*

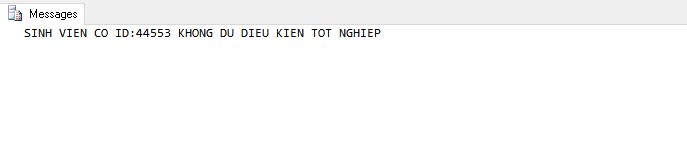
*else*

*print 'SINH VIEN CO ID:' + convert(varchar,@STUDENT\_ID ) + ' KHONG DU DIEU KIEN TOT NGHIEP';*

*end;*

* **Kết quả**

*exec dbo.KIEM\_TRA\_DKTN'44553';*



* *Nhận xét:*
* *Bước 1:* tạo hàm chuyển đổi điểm từ dạng chữ sang dạng điểm số
* *Bước 2*: tạo view xem thông tin của sinh viên
* *Bước 3 :* tạo hàm tính CPA
* *Bước 4:* tạo hàm tính tín chỉ tích lũy
* *Bước 5* : tạo thủ tục kiểm tra sinh viên có đủ điều kiện tốt nghiệp hay không
* *Bước 6:* test kết quả
* Lưu ý: Do một sinh viên có thể học nhiều lần 1 học phần nên khi tính điểm ta lấy điểm cao nhất (max(Point)).

1. **Bài 2**

* **Đề bài:**Viết thủ tục SP\_LOC\_DU\_LIEU cho phép nhập vào tên trường bất kỳ và một giá trị củatrường (Ví dụ: SP\_LOC\_DU\_LIEU *‘dept\_name’, ‘Physics’*). Kết quả trả về là dữ liệu sau khi lọc theo giá trị của trường dữ liệu đó. Bảng kết quả trả về gồm các trường: *Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng virn, Kkhoa viện.*
* **Oracle**
* **Mã nguồn**

*--Tao bang moi co thoi gian da duoc ghep---*

*CREATE TABLE TIME*

*AS*

*(SELECT TIME\_SLOT\_ID,DAY||','||START\_HR||':'||START\_MIN||'-'||END\_HR||':'||END\_MIN TIMES FROM TIME\_SLOT);*

*-- Tao bang view voi tat ca thuoc tinh de bai yeu cau--*

*CREATE OR REPLACE VIEW VW\_BAI\_2*

*AS*

*select tk.id ID, st.name NAME, se.year YEAR, se.semester SEMESTER,se.course\_id COURSEID,ts.TIMES TIMES,*

*se.room\_number ROOMNUMBER, ins.name TEACHERNAME,st.dept\_name DETPNAME*

*FROM section se*

*JOIN course cr on se.course\_id = cr.course\_id*

*join teaches te on se.course\_id = te.course\_id*

*and se.sec\_id = te.sec\_id*

*and se.semester = te.semester*

*and se.year = te.year*

*join instructor ins on te.id = ins.id*

*join takes tk on se.sec\_id = tk.sec\_id*

*and se.semester = tk.semester*

*and se.year = tk.year*

*and se.course\_id = tk.course\_id*

*join student st on tk.id = st.id*

*join time ts on se.time\_slot\_id = ts.time\_slot\_id;*

*--Tao ham loc du lieu--*

*CREATE OR REPLACE PROCEDURE SP\_LOC\_DU\_LIEU*

*( field\_name IN varchar, field\_value IN varchar)*

*IS*

*cur sys\_refcursor;*

*cur\_id VARCHAR(20);*

*cur\_name VARCHAR(20);*

*cur\_year VARCHAR(20);*

*cur\_semester VARCHAR(20);*

*cur\_courseid VARCHAR(20);*

*cur\_times VARCHAR(20);*

*cur\_roomnumber VARCHAR(20);*

*cur\_teachername VARCHAR(20);*

*cur\_deptname VARCHAR(20);*

*BEGIN*

*IF field\_name IN*

*('ID', 'NAME', 'YEAR', 'SEMESTER','COURSEID','TIMES',*

*'ROOMNUMBER','TEACHERNAME','DETPNAME')*

*THEN*

*OPEN cur FOR*

*'SELECT \* FROM VW\_BAI\_2*

*WHERE ' ||field\_name||'='''||field\_value||'''';*

*DBMS\_OUTPUT.PUT\_LINE(rpad('Ma sv',10)||'|'||rpad('Ho ten',10)||'|'|| rpad('Namhoc',10)||'|'||rpad('Kyhoc',10)||'|'|| rpad('Khoahoc',30)||'|'||rpad('Thoigian',10)||'|'|| rpad('Phonghoc',10)||'|'||rpad('Giangvien',11)||'|'|| rpad('Khoa vien',10));*

*DBMS\_OUTPUT.PUT\_LINE(lpad('-',130,'-'));*

*LOOP*

*FETCH cur INTO cur\_id,cur\_name,cur\_year,cur\_semester,*

*cur\_courseid,cur\_times,cur\_roomnumber,cur\_teachername,cur\_deptname;*

*EXIT WHEN cur%NOTFOUND;*

*DBMS\_OUTPUT.PUT\_LINE(rpad(cur\_id,10)||'|'||rpad(cur\_name,10)||'|'||*

*rpad(cur\_year,10)||'|'||rpad(cur\_semester,10)||'|'||*

*rpad(cur\_courseid,30)||'|'||rpad(cur\_times,10)||'|'||*

*rpad(cur\_roomnumber,10)||'|'||rpad(cur\_teachername,11)||'|'||*

*rpad(cur\_deptname,10));*

*END LOOP;*

*CLOSE cur;*

*ELSE DBMS\_OUTPUT.PUT\_LINE('KHONG TIM THAY KET QUA');*

*END IF;*

*END;*

* **Kết quả**

*DECLARE*

*FIELD\_NAME VARCHAR2(200);*

*FIELD\_VALUE VARCHAR2(200);*

*BEGIN*

*FIELD\_NAME := 'ID';*

*FIELD\_VALUE := '13506';*

*SP\_LOC\_DU\_LIEU(*

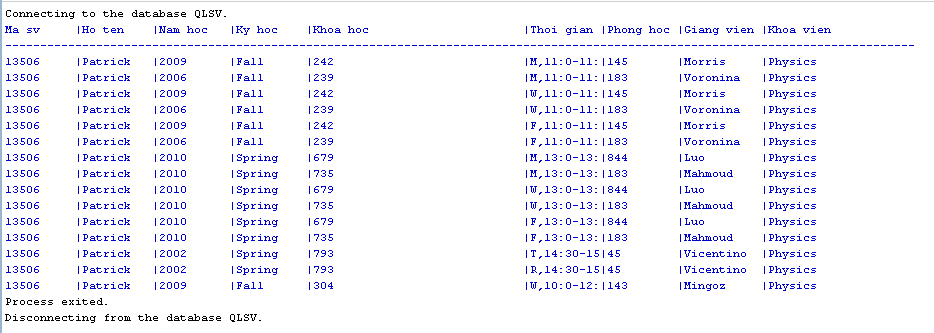
*FIELD\_NAME => FIELD\_NAME,*

*FIELD\_VALUE => FIELD\_VALUE*

*);*

*--rollback;*

*END;*



* **SQLL Server**
* **Mã nguồn**

*---Tao bang moi chua thoi gian da duoc ghep chuoi---*

*SELECT TIME\_SLOT\_ID,CONCAT(DAY,',',START\_HR,':',START\_MIN,'-',END\_HR,':',END\_MIN) AS TIMES*

*INTO THOI\_GIAN*

*FROM time\_slot;*

*---- Tao view chung chua cac thuoc tinh can loc ---*

*CREATE VIEW BAI\_2*

*AS*

*SELECT tk.id ID, st.name NAME,se.year YEAR, se.semester SEMESTER,*

*se.course\_id COURSEID, se.room\_number ROOMNUMBER, ins.name TEACHERNAME,*

*ts.times TIMES,st.dept\_name DETPNAME*

*FROM section se*

*JOIN course cr ON se.course\_id = cr.course\_id*

*JOIN teaches te ON se.course\_id = te.course\_id*

*AND se.sec\_id = te.sec\_id*

*AND se.semester = te.semester*

*AND se.year = te.year*

*JOIN instructor ins ON te.id = ins.id*

*JOIN takes tk ON se.sec\_id = tk.sec\_id*

*AND se.semester = tk.semester*

*AND se.year = tk.year*

*AND se.course\_id = tk.course\_id*

*JOIN student st ON tk.id = st.id*

*JOIN THOI\_GIAN ts ON se.time\_slot\_id = ts.time\_slot\_id;*

*-- Tao ham loc du lieu--*

*CREATE PROCEDURE SP\_LOC\_DU\_LIEU\_1(@field\_name varchar(30), @field\_value varchar(30))*

*AS*

*BEGIN*

*IF @field\_name in ('ID','NAME','YEAR','SEMESTER','COURSEID','ROOMNUMBER',*

*'TEACHERNAME','TIMES', 'DETPNAME')*

*DECLARE @query varchar(200)*

*SET @query =*

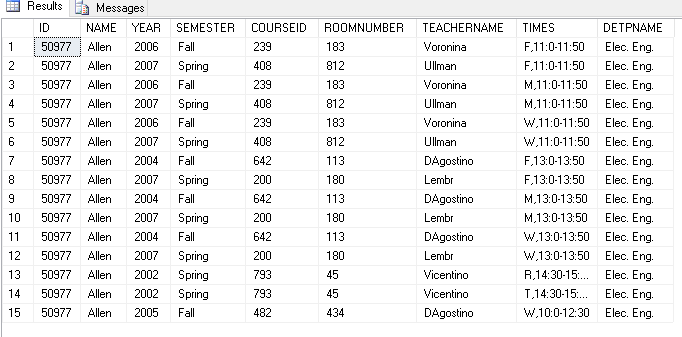
*'SELECT \* FROM BAI\_2 WHERE '+ @field\_name + '=' + ''''+@field\_value+''''*

*EXEC (@query)*

*END*

* **Kết quả**

*EXECUTE SP\_LOC\_DU\_LIEU\_1 @field\_name='id' , @field\_value = '50977';*



* *Nhận xét:*
* *Bước 1:* tạo view gồm tất cả các thông tin đầu ra bằng cách join các bảng có chứa thông tin thành 1 bảng
* *Bước 2:* tạo thủ tục lọc dữ liệu
* *Lưu ý :* thời gian học được tạo thành bằng cách ghép chuỗi của các cột có thông tin liên quan về thời gian.

1. **Bài 3**

* **Đề bài:** Viết thủ tục SP\_LOC\_DU\_LIEU cho phép nhập vào một biến kiểu table gồm 2 trường: tên trường và một giá trị của trường. Kết quả trả về là dữ liệu sau khi lọc theo danh sách các giá trị của các trường dữ liệu đó. Bảng kết quả trả về gồm các trường: *Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng viên, Khoa Viện.*
* **Oracle**
* **Mã nguồn**

*CREATE OR REPLACE VIEW VW\_BAI\_2*

*AS*

*select tk.id ID, st.name NAME, se.year YEAR, se.semester SEMESTER,se.course\_id COURSEID,ts.TIMES TIMES,*

*se.room\_number ROOMNUMBER, ins.name TEACHERNAME,st.dept\_name DETPNAME*

*FROM section se*

*JOIN course cr on se.course\_id = cr.course\_id*

*join teaches te on se.course\_id = te.course\_id*

*and se.sec\_id = te.sec\_id*

*and se.semester = te.semester*

*and se.year = te.year*

*join instructor ins on te.id = ins.id*

*join takes tk on se.sec\_id = tk.sec\_id*

*and se.semester = tk.semester*

*and se.year = tk.year*

*and se.course\_id = tk.course\_id*

*join student st on tk.id = st.id*

*join time ts on se.time\_slot\_id = ts.time\_slot\_id;*

*CREATE OR REPLACE TYPE field AS OBJECT (field\_name varchar(50),field\_value varchar(50));*

*CREATE OR REPLACE TYPE tablefield IS TABLE OF field;*

*-- tao ham loc du lieu ----*

*CREATE OR REPLACE PROCEDURE SP\_LOC\_DU\_LIEU\_1(i\_table tablefield)*

*AS*

*cur sys\_refcursor;*

*query varchar(1000) := 'SELECT \* FROM VW\_BAI\_2 WHERE 1=1 ';*

*cur\_id VARCHAR(20);*

*cur\_name VARCHAR(20);*

*cur\_year VARCHAR(20);*

*cur\_semester VARCHAR(20);*

*cur\_courseid VARCHAR(20);*

*cur\_times VARCHAR(20);*

*cur\_roomnumber VARCHAR(20);*

*cur\_teachername VARCHAR(20);*

*cur\_deptname VARCHAR(20);*

*BEGIN*

*FOR i IN 1..i\_table.LAST*

*LOOP*

*IF i\_table(i).field\_name IN*

*('ID', 'NAME', 'YEAR', 'SEMESTER','COURSEID','TIMES',*

*'ROOMNUMBER','TEACHERNAME','DETPNAME')*

*THEN*

*query := query ||' AND '|| i\_table(i).field\_name ||*

*'=''' || i\_table(i).field\_value || '''';*

*ELSE*

*DBMS\_OUTPUT.PUT\_LINE('NHAP KHONG DUNG');*

*END IF;*

*END LOOP;*

*OPEN cur FOR query;*

*DBMS\_OUTPUT.PUT\_LINE(rpad('Ma sv',10)||'|'||rpad('Ho ten',10)||'|'||*

*rpad('Nam hoc',10)||'|'||rpad('Ky hoc',10)||'|'||*

*rpad('Khoa hoc',30)||'|'||rpad('Thoi gian',10)||'|'||*

*rpad('Phong hoc',10)||'|'||rpad('Giang vien',11)||'|'||*

*rpad('Khoa vien',10));*

*DBMS\_OUTPUT.PUT\_LINE(lpad('-',130,'-'));*

*LOOP*

*FETCH cur INTO*

*cur\_id,cur\_name,cur\_year,cur\_semester,*

*cur\_courseid,cur\_times,cur\_roomnumber,cur\_teachername,cur\_deptname;*

*EXIT WHEN cur%NOTFOUND;*

*DBMS\_OUTPUT.PUT\_LINE(rpad(cur\_id,10)||'|'||rpad(cur\_name,10)||'|'||*

*rpad(cur\_year,10)||'|'||rpad(cur\_semester,10)||'|'||*

*rpad(cur\_courseid,30)||'|'||rpad(cur\_times,10)||'|'||*

*rpad(cur\_roomnumber,10)||'|'||rpad(cur\_teachername,11)||'|'||*

*rpad(cur\_deptname,10));*

*END LOOP;*

*CLOSE cur;*

*END;*

* **Kết quả**

*DECLARE*

*I\_TABLE C##TRANGDAO.TABLEFIELD:= tablefield(field('ID','13506'),field('NAME','Patrick'));*

*BEGIN*

*-- Modify the code to initialize the variable*

*-- I\_TABLE := NULL;*

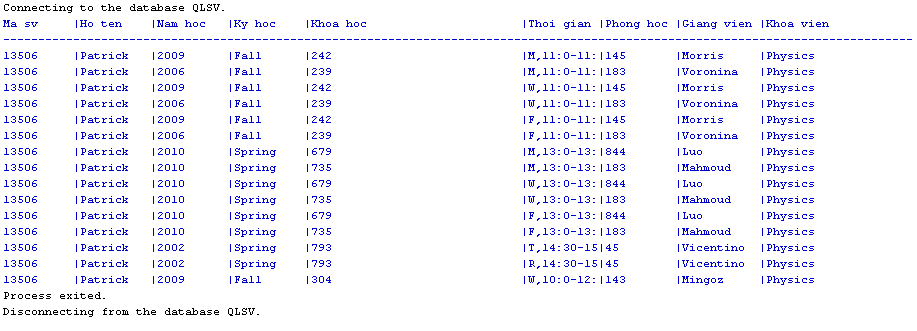
*SP\_LOC\_DU\_LIEU\_1(*

*I\_TABLE => I\_TABLE*

*);*

*--rollback;*

*END;*



* **SQL Server**
* **Mã nguồn**

*--TAO DU LIEU KIEU TABLE - dinh nghia loai bang --*

*CREATE TYPE object*

*as table(field\_name varchar(20), field\_value varchar(30))*

*--- Thuc hien loc du lieu nhu bai 2 ---*

*CREATE PROCEDURE SP\_LOC\_DU\_LIEU\_2 (@tab object READONLY)*

*AS*

*BEGIN*

*DECLARE @str\_query varchar(100);*

*DECLARE @i int =1;*

*DECLARE @count int =0;*

*DECLARE @table table(row int, col varchar(50), search varchar(50));*

*DECLARE @field\_name varchar(20)*

*DECLARE @field\_value varchar(20)*

*INSERT INTO @table SELECT ROW\_NUMBER() OVER (ORDER BY field\_name), \* FROM @tab*

*SET @count = (SELECT COUNT(\*) FROM @tab)*

*IF @count > 0 SET @str\_query = 'SELECT \* FROM BAI\_2 WHERE 1=1';*

*BEGIN*

*WHILE (@i <= @count)*

*BEGIN*

*SELECT @field\_name = col , @field\_value = search FROM @table*

*WHERE row = @i*

*SET @str\_query = @str\_query + 'and ' + ISNULL(@field\_name, '')+*

*' LIKE ''%' + isnull(@field\_value, '') + '%'''*

*SET @i = @i +1*

*END*

*END*

*EXECUTE (@str\_query);*

*END*

* **Kết quả**

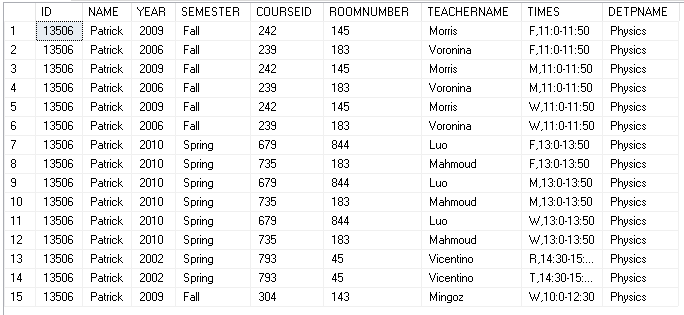
*--- Test ket qua ---*

*DECLARE @cur object*

*INSERT INTO @cur values('id','13506')*

*INSERT INTO @cur values('name','Patrick')*

*EXEC SP\_LOC\_DU\_LIEU\_2 @tab = @cur*



* *Nhận xét:*
* *Bước 1 :* sử dụng view thông tin của bài 2
* *Bước 2:* tạo kiểu dữ liệu object và table
* *Bước 3:* thực hiện lọc dữ liệu giống như bài 2

1. **Bài 4**

**Đề bài:** Sinh viên A muốn học môn *‘Mobile Computing’* cần phải học qua những môn gì?

* **Oracle**
* **Mã nguồn**

*SELECT prereq\_id "prereq learn " , course.title*

*FROM prereq JOIN course*

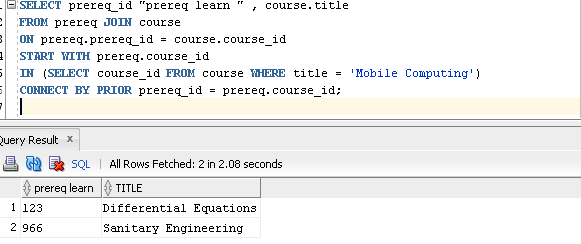
*ON prereq.prereq\_id = course.course\_id*

*START WITH prereq.course\_id*

*IN (SELECT course\_id FROM course WHERE title = 'Mobile Computing')*

*CONNECT BY PRIOR prereq\_id = prereq.course\_id;*

* **Kết quả**



* **SQL Server**
* **Mã nguồn**

*SELECT course\_id ID, title TITLE FROM course WHERE course\_id in*

*(SELECT prereq\_id FROM prereq WHERE course\_id*

*IN (SELECT course\_id FROM course WHERE title = 'Mobile Computing') );*

* **Kết quả**



* *Nhận xét:*
* Oracle: sử dụng truy vấn phân cấp (tree)
* SQL: dùng đệ quy và các truy vấn lồng

1. **Bài 5**

* **Đề bài:** Cài đặt Trigger kiểm tra số lượng sinh viên đăng ký vượt quá sức chứa của phòng. Đưa ra thông báo không thành công khi sinh viên đăng ký môn học. Rollback khi có lỗi xảy ra.
* **Oracle**
* **Mã nguồn**

*CREATE OR REPLACE TRIGGER KIEM\_TRA\_DANG\_KY\_LOP*

*BEFORE INSERT ON takes*

*FOR EACH ROW*

*DECLARE*

*max\_cap number ;*

*cur\_cap number;*

*BEGIN*

*-- lay ra dung luong toi da cua phong --*

*SELECT capacity INTO max\_cap FROM classroom c*

*JOIN section se*

*ON c.building = se.building*

*AND c.room\_number = se.room\_number*

*WHERE course\_id = :new.course\_id*

*AND sec\_id= :new.sec\_id*

*AND semester =:new.semester*

*AND se.year = :new.year;*

*--- Lay ra so luong dang ky hien tai--*

*SELECT count(id) INTO cur\_cap FROM takes t*

*WHERE course\_id = :new.course\_id*

*AND sec\_id= :new.sec\_id*

*AND semester = :new.semester*

*AND t.year = :new.year;*

*IF (cur\_cap >= max\_cap )*

*THEN*

*dbms\_output.put\_line ('LOP HOC DA DAY');*

*ROLLBACK;*

*RAISE\_APPLICATION\_ERROR(-20002, 'LOP HOC DA DAY');*

*ELSE*

*dbms\_output.put\_line('DANG KY THANH CONG!');*

*END IF;*

*END KIEM\_TRA\_DANG\_KY\_LOP;*

* **Kết quả**

*DECLARE*

*BEGIN*

*insert into TAKES*

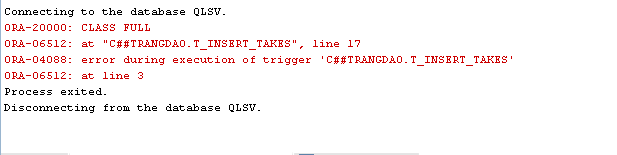
*(ID,COURSE\_ID,SEC\_ID,SEMESTER,YEAR,GRADE)*

*values*

*('1000','376','1','Fall','2006',NULL);*

*rollback;*

*END;*



* **SQL Server**
* **Mã nguồn**

*CREATE TRIGGER KIEM\_TRA\_DANG\_KY\_LOP*

*ON takes*

*FOR INSERT*

*AS*

*BEGIN*

*DECLARE @max\_cap numeric;*

*DECLARE @cur\_cap numeric;*

*SELECT @max\_cap = capacity FROM classroom c , section se,*

*inserted i*

*WHERE c.room\_number = se.room\_number*

*AND se.course\_id = i.course\_id*

*AND se.sec\_id = i.sec\_id*

*AND se.year= i.year*

*AND se.semester = i.semester;*

*SELECT @cur\_cap = COUNT(\*) FROM takes t, inserted i*

*WHERE t.course\_id = i.course\_id*

*AND t.year = i.year*

*AND t.sec\_id = i.sec\_id*

*AND t.semester = i.semester ;*

*IF (@max\_cap >= @cur\_cap +1)*

*PRINT ('DANG KY THANH CONG!');*

*ELSE*

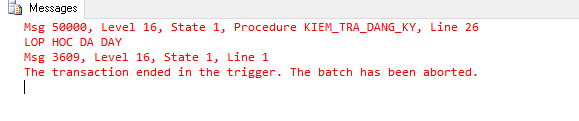
*RAISERROR ('LOP HOC DA DAY !',16,1);*

*ROLLBACK TRANSACTION;*

*END ;*

* **Kết quả**

*insert into takes values ('1000','376','1','Fall','2006',null);*



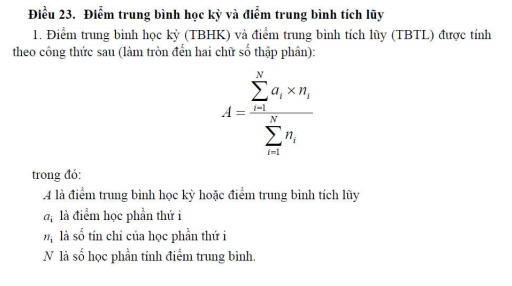
* *Nhận xét:*
* Cài đặt trigger kiểm tra số lượng sinh viên đăng ký vượt quá sức chứa của phòng.
* Tính toán tổng số người đăng ký hiện tại và so sánh với sức chứa tối đa của phòng.

1. **Bài 6**

* **Đề bài:**Viết thủ tục cho biết kết quả học tập của một sinh viên với:

– **Đầu vào:** Mã sinh viên

– **Đầu ra:** Mã sinh viên, Tên sinh viên, Số tín chỉ tích lũy, Điểm trung bình học kỳ và điểm trung bình tích lũy theo từng học kỳ.



* **Oracle**
* **Mã nguồn**

*create or replace view THONG\_TIN\_SV ("ID","NAME","COURSE\_ID", "YEAR","SEC\_ID","SEMESTER","SEMESTER\_ID","GRADE","POINT","CREDITS")*

*as*

*select t.id, s.name, t.course\_id, t.year, t.sec\_id,t.semester,*

*case*

*when t.semester = 'Spring' then 1*

*else 2*

*end semester\_id, t.grade,*

*case*

*when t.grade = 'A+' then 4.5*

*when t.grade = 'A' then 4.0*

*when t.grade = 'A-' then 3.5*

*when t.grade = 'B+' then 3.0*

*when t.grade = 'B' then 2.5*

*when t.grade = 'B-' then 2.0*

*when t.grade = 'C+' then 1.5*

*when t.grade = 'C' then 1.0*

*when t.grade = 'C-' then 0.5*

*else 0.0*

*end as Point, c.credits*

*from takes t*

*join student s*

*on t.id = s.id*

*join course c*

*on c.course\_id =t.course\_id;*

*--TAO ham tinh so tctl hien tai---*

*CREATE OR REPLACE FUNCTION TCTLHT (student\_id varchar, cur\_semester varchar, cur\_year varchar)*

*RETURN number*

*IS tctl number(3,0);*

*BEGIN*

*SELECT sum(t.credits) INTO tctl*

*FROM (SELECT course\_id, credits, max(Point) point*

*FROM (SELECT \* FROM THONG\_TIN\_SV tb*

*WHERE tb.id = student\_id*

*AND tb.year <= cur\_year*

*AND tb.semester\_id <= cur\_semester)*

*GROUP BY course\_id,credits*

*HAVING max(Point) > 0.5) t;*

*RETURN tctl;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN RETURN 0 ;*

*END ;*

*--- Tinh diem trung binh hoc ky GPA --*

*CREATE OR REPLACE FUNCTION GPA (student\_id varchar, cur\_semester varchar, cur\_year varchar)*

*RETURN number*

*AS gpa number(3,2);*

*BEGIN*

*SELECT sum(point\*credits)/sum(credits) INTO gpa*

*FROM (SELECT course\_id, credits, max(Point) point FROM THONG\_TIN\_SV tb*

*WHERE tb.id = student\_id*

*AND tb.year= cur\_year*

*AND tb.semester\_id = cur\_semester*

*GROUP BY course\_id, credits);*

*RETURN gpa;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN RETURN 0 ;*

*END ;*

*--- TInh diem trung binh tich tuy den hoc ky hien tai CPA--*

*CREATE OR REPLACE FUNCTION CPA (student\_id varchar, cur\_semester varchar, cur\_year varchar)*

*RETURN number*

*AS cpa number(3,2);*

*BEGIN*

*SELECT sum(point\*credits)/sum(credits) INTO cpa*

*FROM (SELECT course\_id, credits, max(Point) point FROM THONG\_TIN\_SV tb*

*WHERE tb.id = student\_id*

*AND tb.year <= cur\_year*

*AND tb.semester\_id <= cur\_semester*

*GROUP BY course\_id,credits);*

*RETURN cpa;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN RETURN 0 ;*

*END ;*

*---Tao ham lay ra ket qua hoc tap cua sinh vien---*

*CREATE OR REPLACE PROCEDURE KET\_QUA\_HOC\_TAP(student\_id IN varchar)*

*AS*

*BEGIN*

*DBMS\_OUTPUT.PUT\_LINE(rpad('MSSV',8,' ')||rpad('TENSV',20,' ')||rpad('NAM HOC',20,' ')||rpad('SO TC',10,' ')||rpad('GPA',10,' ')||rpad('CPA',10,' '));*

*DBMS\_OUTPUT.PUT\_LINE(rpad('-',80,'-'));*

*FOR t in (*

*SELECT DISTINCT id AS ID, NAME, YEAR, TCTLHT(id,semester\_id,year) AS CREDITS,*

*GPA(id,semester\_id,year) AS GPA, CPA(id,semester\_id,year) AS CPA*

*FROM THONG\_TIN\_SV WHERE id=student\_id*

*ORDER BY CREDITS)*

*LOOP*

*DBMS\_OUTPUT.PUT\_LINE(rpad(t.ID,8,' ')||rpad(t.NAME,20,' ')||rpad(t.YEAR,20,' ')||rpad(t.CREDITS,10,' ')||rpad(t.GPA,10,' ')||rpad(t.CPA,10,' '));*

*END LOOP;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*DBMS\_OUTPUT.PUT\_LINE('KHONG CO THONG TIN');*

*END;*

* **Kết quả**

*DECLARE*

*STUDENT\_ID VARCHAR2(200);*

*BEGIN*

*STUDENT\_ID := '13506';*

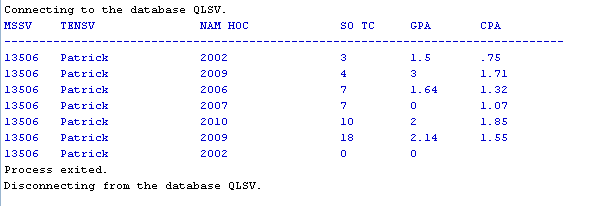
*KET\_QUA\_HOC\_TAP(*

*STUDENT\_ID => STUDENT\_ID*

*);*

*rollback;*

*END;*



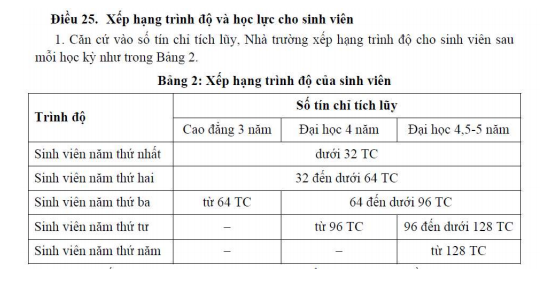
* **SQL Server**
* **Mã nguồn**
* **Kết quả**
* *Nhận xét:*

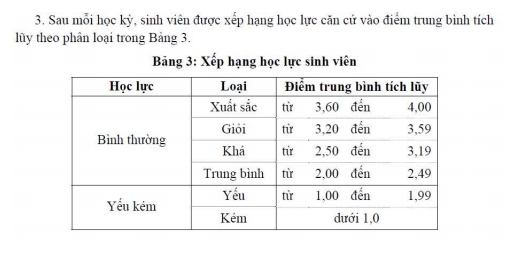
1. **Bài 7**

**Đề bài :** Viết thủ tục đánh giá kết quả học tập của một sinh viên với:

– *Đầu vào:* Mã sinh viên

– *Đầu ra:* Xếp hạng trình độ sinh viên và xấp hạng học lực của sinh viên, biết rằng:





* **Oracle**
* **Mã nguồn:**

*---Tao ham quy doi hoc luc sinh vien ---*

*CREATE OR REPLACE FUNCTION HOC\_LUC (student\_id varchar, cur\_semester varchar, cur\_year varchar)*

*RETURN VARCHAR*

*AS hocluc VARCHAR(10);*

*cpa number(3,2);*

*BEGIN*

*cpa := CPA\_1(student\_id, cur\_semester, cur\_year);*

*IF (cpa < 1.0) THEN hocluc:='Kem';*

*ELSIF (cpa >= 1.0 AND cpa < 2.0) THEN hocluc :='Yeu';*

*ELSIF (cpa >= 2.0 AND cpa < 2.5) THEN hocluc :='TrungBinh';*

*ELSIF (cpa >= 2.5 AND cpa < 3.2) THEN hocluc :='Kha';*

*ELSIF (cpa >= 3.2 AND cpa < 3.6) THEN hocluc :='Gioi';*

*ELSIF (cpa >= 3.6 AND cpa <= 4.0) THEN hocluc :='XuatSac';*

*END IF ;*

*RETURN hocluc;*

*END;*

*--Tao ham quy doi trinh do sinh vien ---*

*CREATE OR REPLACE FUNCTION TRINH\_DO(student\_id varchar, cur\_semester varchar, cur\_year varchar)*

*RETURN number*

*AS*

*trinhdo number(1);*

*tctl number(3);*

*BEGIN*

*tctl := TCTLHT(student\_id, cur\_semester, cur\_year);*

*IF (tctl < 32) THEN trinhdo := 1 ;*

*ELSIF (tctl >= 32 AND tctl <64) THEN trinhdo := 2;*

*ELSIF (tctl >= 64 AND tctl <96) THEN trinhdo := 3;*

*ELSIF (tctl >= 96 AND tctl <128) THEN trinhdo := 4;*

*ELSE trinhdo := 5;*

*END IF;*

*RETURN trinhdo;*

*END TRINH\_DO;*

*--Tao ham danh gia ket qua sinh vien --*

*CREATE OR REPLACE PROCEDURE DANH\_GIA (student\_id varchar)*

*AS*

*BEGIN*

*DBMS\_OUTPUT.PUT\_LINE(rpad('MSSV',8,' ')||rpad('TENSV',15,' ')||rpad('NAM HOC',10,' ')||rpad('SO TC',10,' ')||rpad('CPA',10,' ')||rpad('TRINH DO',10,' ')||rpad('HOC LUC',10,' '));*

*DBMS\_OUTPUT.PUT\_LINE(rpad('-',140,'-'));*

*FOR t in (*

*SELECT id AS ID, NAME, YEAR, TCTLHT(id,semester\_id,year) AS CREDITS,*

*CPA(id,semester\_id,year) AS CPA, TRINH\_DO(student\_id,semester\_id, year) AS TRINHDO, HOC\_LUC(id, semester\_id, year) AS HOCLUC*

*FROM THONG\_TIN\_SV WHERE id=student\_id*

*ORDER BY CREDITS)*

*LOOP*

*DBMS\_OUTPUT.PUT\_LINE(rpad(t.ID,8,' ')||rpad(t.NAME,15,' ')||rpad(t.YEAR,10,' ')||rpad(t.CREDITS,10,' ')||rpad(t.CPA,10,' ')||rpad(t.TRINHDO,10,' ')||rpad(t.HOCLUC,10,' '));*

*END LOOP;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*DBMS\_OUTPUT.PUT\_LINE('KHONG CO THONG TIN');*

*END;*

* **Kết quả**

*DECLARE*

*STUDENT\_ID VARCHAR2(200);*

*BEGIN*

*STUDENT\_ID := '13506';*

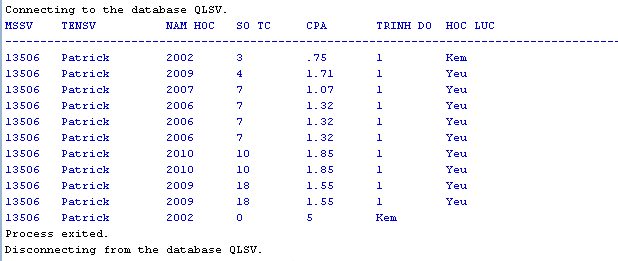
*DANH\_GIA(*

*STUDENT\_ID => STUDENT\_ID*

*);*

*rollback;*

*END;*



* **SQL Server**
* **Mã nguồn**
* **Kết quả**
* *Nhận xét:*

1. **Bài 8**

* **Đề bài:** Đánh chỉ mục các bảng takes, student, advisor. So sánh tốc độ truy vấn sau khi đã thực hiện đánh chỉ mục.
* **Oracle**
* **Mã nguồn**

*CREATE INDEX idx\_takes ON takes(grade);*

*SELECT \* FROM takes WHERE takes.grade = 'B+';*

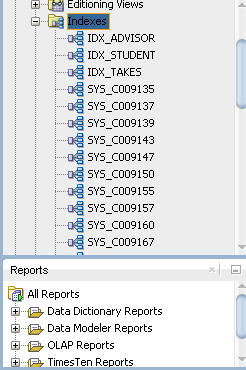
*CREATE INDEX idx\_student ON student (name);*

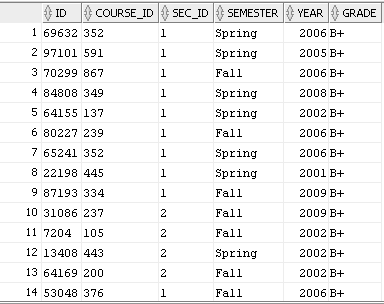
*SELECT \* FROM student WHERE student.name ='Allen';*

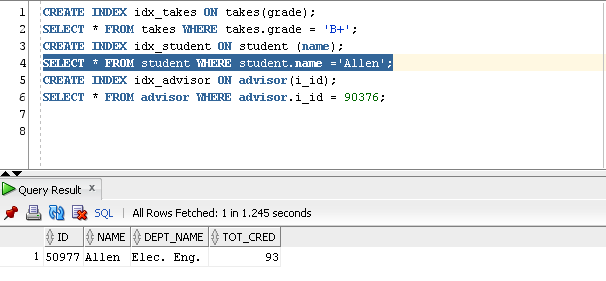
*CREATE INDEX idx\_advisor ON advisor(i\_id);*

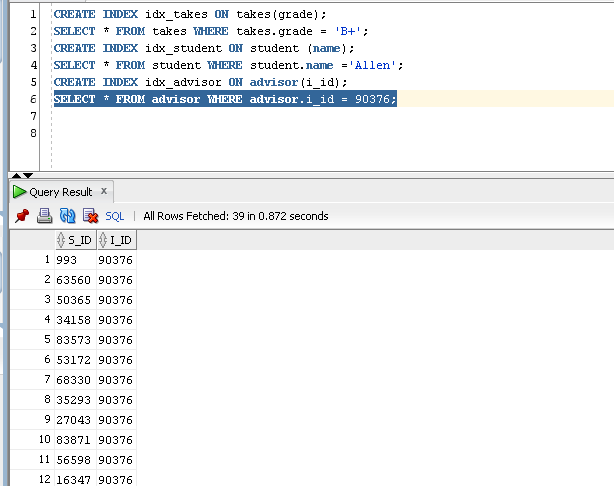
*SELECT \* FROM advisor WHERE advisor.i\_id = 90376;*

* **Kết quả**









* **SQL Server**
* **Mã nguồn**

*CREATE INDEX idx\_takes ON takes(grade);*

*SELECT \* FROM takes WHERE takes.grade = 'B+';*

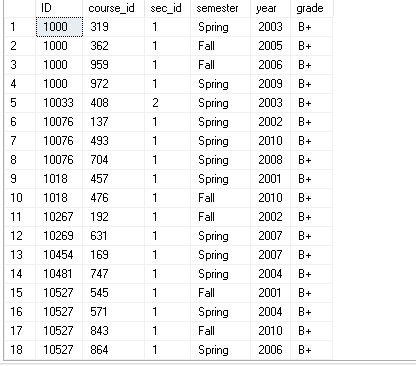
*CREATE INDEX idx\_student ON student (name);*

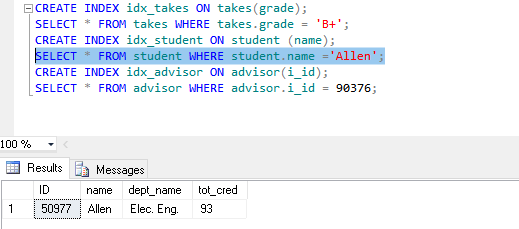
*SELECT \* FROM student WHERE student.name ='Allen';*

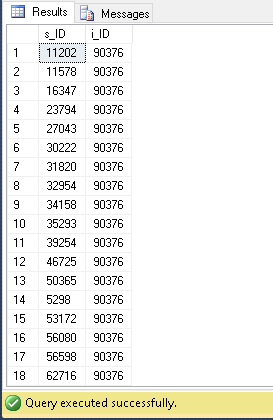
*CREATE INDEX idx\_advisor ON advisor(i\_id);*

*SELECT \* FROM advisor WHERE advisor.i\_id = 90376;*

* **Kết quả**







* *Nhận xét:*

1. **Bài 9**

* **Đề bài:** Viết thủ tục cho phép sinh viên đăng ký khóa học với lựa chọn phòng và thời gian nào đó. Cài đặt các TRANSACTION để đảm bảo toàn vẹn dữ liệu và đưa ra thông báo lỗi khi có lỗi xảy ra.
* **Oracle**
* **Mã nguồn**

*CREATE OR REPLACE PROCEDURE DANG\_KY\_LOP (*

*student\_id varchar , course\_id int, sec\_id int,*

*year varchar, semester varchar*

*)*

*IS*

*BEGIN*

*SAVEPOINT a;*

*INSERT INTO takes VALUES (student\_id, course\_id, sec\_id, semester, year, null);*

*COMMIT;*

*DBMS\_OUTPUT.PUT\_LINE ('DANG KY THANH CONG ');*

*EXCEPTION*

*WHEN others THEN DBMS\_OUTPUT.PUT\_LINE ('DANG KY KHONG THANH CONG');*

*ROLLBACK TO a;*

*RAISE ;*

*END;*

* **Kết quả**

*DECLARE*

*STUDENT\_ID VARCHAR2(200);*

*COURSE\_ID NUMBER;*

*SEC\_ID NUMBER;*

*YEAR VARCHAR2(200);*

*SEMESTER VARCHAR2(200);*

*BEGIN*

*STUDENT\_ID := '13506';*

*COURSE\_ID := '376';*

*SEC\_ID := '1';*

*YEAR := '2006';*

*SEMESTER := 'Spring';*

*DANG\_KY\_LOP(*

*STUDENT\_ID => STUDENT\_ID,*

*COURSE\_ID => COURSE\_ID,*

*SEC\_ID => SEC\_ID,*

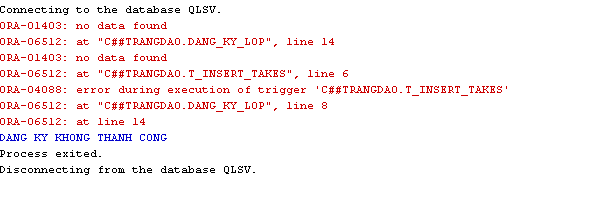
*YEAR => YEAR,*

*SEMESTER => SEMESTER*

*);*

*rollback;*

*END;*



* **SQL Server**
* **Mã nguồn**

*CREATE PROCEDURE DANG\_KY\_LOP*

*(@student\_id varchar(5), @course\_id numeric, @sec\_id int,*

*@semester varchar, @year varchar*

*)*

*AS*

*BEGIN TRY*

*BEGIN TRANSACTION*

*INSERT INTO takes VALUES (@student\_id, @course\_id, @sec\_id, @semester, @year,NULL);*

*COMMIT TRANSACTION*

*END TRY*

*BEGIN CATCH*

*IF @@TRANCOUNT > 0*

*ROLLBACK TRANSACTION*

*DECLARE @mess varchar (2000) = error\_message()*

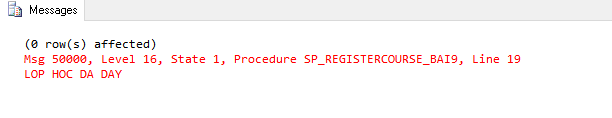
*RAISERROR( @mess,16, 1 );*

*END CATCH*

*RETURN 0*

* **Kết quả:**

*EXEC DANG\_KY\_LOP '1000', '376', '1', 2006, 'Fall';*



* *Nhận xét:*

1. **Bài 10 :**

* **Đề bài:**

*Yêu cầu chung*: Thiết kế và cài đặt giao diện ứng dụng trên môi trường WinForms kết nối với cơ sở dữ liệu, bằng ngôn ngữ lập trình C# cho phép sinh viên có thể đăng ký học tập theo học chế tín chỉ.

*Mô tả chi tiết*: Ứng dụng gồm 2 Forms:

+ Form01: Chứa 1 Data Grid 1 bộ lọc. Data Grid hiển thị đầy đủ các trường thông tin: Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng viên, Khoa viện. Bộ lọc hỗ trợ lọc theo các trường: Mã sinh viên, Họ tên, sinh viên, Năm học, Kỳ học, Khóa học. Thực hiện các chức năng xem, thêm, xóa, sửa đối với từng sinh viên, thao tác trên Form 02 dưới đây.

+ Form02: Chứa các thông tin về Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng viên, Khoa viện của từng sinh viên. Cài đặt mối liên kết dữ liệu trên giao diện: chẳng hạn chọn khoa viện thì lọc được danh sách sinh viên thuộc khoa viện, chọn khóa học thì lọc được danh sách giảng viên có thể dạy khóa học đó. Thực hiện cập nhật dữ liệu trên giao diện và lưu trữ vào cơ sở dữ liệu.

Lưu ý: sinh viên lựa chọn và thiết kế các Controls trên Form phù hợp với dữ liệu được mô tả trong cơ sở dữ liệu đã cho trước.

* **Oracle**
* **Mã nguồn**
* **Kết quả**
* **SQL Server**
* **Mã nguồn**
* **Kết quả**
* *Nhận xét:*