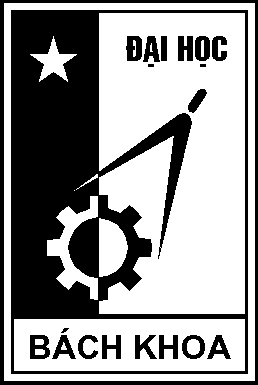
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**

****

**TIỂU LUẬN**

**CƠ SỞ DỮ LIỆU** **NÂNG CAO**

**Sinh viên thực hiện : Nguyễn Chí Thảo**

**Mã số sinh viên   : 20153445**

**Lớp   : Toán Tin 02 – K60**

**Hà Nội 2019**

Mục lục

[Phần 1: Cơ sử dữ liệu lớn 4](#_Toc10853972)

[1.1. Khái niệm dữ liệu lớn (Big Data) 4](#_Toc10853973)

[1.2. Các đặc trưng 5](#_Toc10853974)

[1.3. Ứng dụng của Big Data 6](#_Toc10853975)

[1.4. Các quan niệm sai lầm về Big Data 6](#_Toc10853976)

[1.5. Cơ sở dữ liệu NoSQL 7](#_Toc10853977)

[Phần 2: Cơ sở dữ liệu phân tán 8](#_Toc10853978)

[2.1. Khái niệm 8](#_Toc10853979)

[2.2. Ưu và nhược điểm của cơ sở dữ liệu phân tán 9](#_Toc10853980)

[2.3. Kiến trúc của một hệ cơ sở dữ liệu phân tán 9](#_Toc10853981)

[2.4. Tính trong suốt của cơ sở dữ liệu phân tán 10](#_Toc10853982)

[Phần 3: Hệ quản trị cơ sở dữ liệu Oracle 11](#_Toc10853983)

[Lecture1 11](#_Toc10853984)

[Lecture2 11](#_Toc10853985)

[Lecture 3 12](#_Toc10853986)

[Lecture 4 15](#_Toc10853987)

[Lecture 5 16](#_Toc10853988)

[Lecture 6 20](#_Toc10853989)

[Lecture 7 22](#_Toc10853990)

[Lecture 8 22](#_Toc10853991)

[Lecture 9 24](#_Toc10853992)

[Lecture 10 26](#_Toc10853993)

[Lecture 11 26](#_Toc10853994)

[Phần 4: Bài tập kết thúc môn 28](#_Toc10853995)

[Bài 1 28](#_Toc10853996)

[Bài 2 31](#_Toc10853998)

[Bài 3 32](#_Toc10853999)

[Bài 4 33](#_Toc10854000)

[Bài 5 34](#_Toc10854001)

[Bài 6 36](#_Toc10854002)

[Bài 7 38](#_Toc10854003)

[Bài 8 40](#_Toc10854004)

[Bài 9 41](#_Toc10854005)

[Bài 10 41](#_Toc10854006)

[Tài liệu tham khảo 43](#_Toc10854007)

# Phần 1: Cơ sử dữ liệu lớn

## Khái niệm dữ liệu lớn (Big Data)

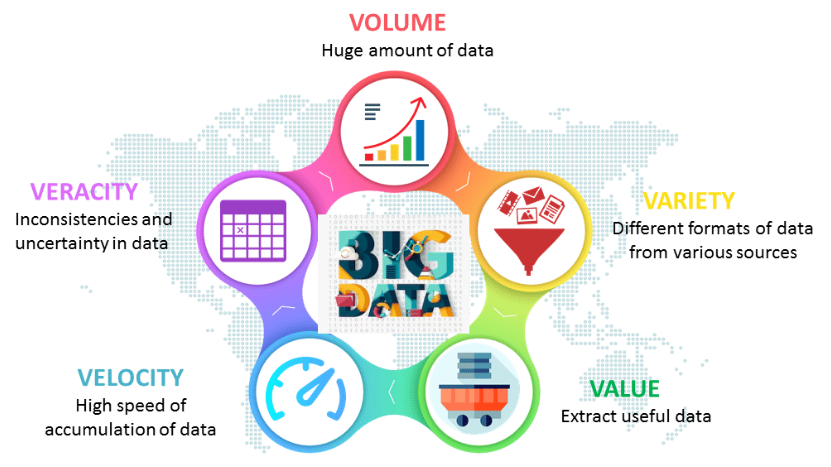
Dữ liệu lớn(Big Data) là một thuật ngữ cho việc xử lý một tập hợp dữ liệu rất lớn và phức tạp mà các ứng dụng xử lý dữ liệu truyền thống không xử lý được. Dữ liệu lớn bao gồm các thách thức như phân tích, thu thập, giám sát dữ liệu, tìm kiếm, chia sẻ, lưu trữ, truyền nhận, trực quan, truy vấn và tính riêng tư.

Dữ liệu lớn có thể được phân tích để có thông tin chi tiết dẫn đến những quyết định tốt hơn và các động thái kinh doanh chiến lược cũng đi đúng hướng hơn.

Các công nghệ “dữ liệu lớn” (“big data”) (như Hadoop, HBase, MongoDB) đã nhận được rất nhiều sự chú ý của truyền thông dạo gần đây. Dữ liệu lớn nói đến những tập dữ liệu quá lớn đối với những hệ thống xử lí dữ liệu truyền thống, và do đó yêu cầu công nghệ xử lí mới. Cùng với các công nghệ truyền thống, những công nghệ big data đang được sử dụng cho rất nhiều công việc, bao gồm cả kĩ thuật dữ liệu. Thỉnh thoảng, các công nghệ big data còn được sử dụng để implement những kĩ thuật data mining. Tuy nhiên, các công nghệ big data được biết đến nhiều hơn là xử lí dữ liệu trong việc *hỗ trợ* các kĩ thuật datamining và các hoạt động khoa học dữ liệu khác.

## Các đặc trưng

Ngày xưa, các công nghệ Big data được mô tả bởi 4 đặc trưng (4V’s of Big Data) nhưng hiện nay, nó đã có thêm 1 chữ V nữa thành 5V.



* Volume (Dung lượng): Lượng dữ liệu mà chúng ta có thể thu thập thực sự rất lớn và do đó dung lượng dữ liệu trở thành một yếu tố quan trọng trong phân tích Big Data.
* Variety (Đa dạng): Dữ liệu được tạo ra hoàn toàn không đồng nhất theo nghĩa nó có thể ở các định dạng khác nhau như video, văn bản, cơ sở dữ liệu, số, dữ liệu cảm biến,.... và do đó hiểu được sự đa dạng của Big Data là yếu tố chính để mở khóa tiềm năng, giá trị thực sự của nó.
* Velocity (Tốc độ): Tốc độ tạo dữ liệu mới nhờ vào sự phụ thuộc của chúng ta vào internet, cảm biến, dữ liệu từ máy này sang máy khác cũng rất quan trọng để phân tích dữ Big Data một cách kịp thời.
* Veracity (Tính xác thực): Biết được rằng liệu dữ liệu có đến từ một nguồn đáng tin hay không cũng là vô cùng quan trọng trước khi giải mã và triển khai phân tích Big Data.
* Value (Giá trị) : Việc trích xuất được những thông tin quan trọng là một trong những việc giúp xử lí dữ liệu chính xác hơn. Liệu dư liệu đấy đã cũ rồi hay vẫn còn mới, nó có nhiều giá trị thống kê hay không, nó có tương quan với nhau hay không…

## Ứng dụng của Big Data

Big Data là 1 trong những xu hướng hiện nay, nó có vô vàn ứng dụng trong thực tế, ví dụ như:

* Xử lí dữ liệu của người dân do chính phủ thu thập
* Dữ liệu mạng xã hội
* Các hệ gợi ý
* Internet vạn vật (Internet of Things – IoT)
* An ninh mạng

…

## Các quan niệm sai lầm về Big Data

* Big Data chỉ dành cho các tập dữ liệu cực lớn
* Chúng ta sẽ thay đổi tất cả các hệ thống hiện tại bằng Big Data
* Big Data là Hadoop
* Các dữ liệu giao dịch trước đây không còn ý nghĩa
* Các kho dữ liệu truyền thống đã là quá khứ
* Big Data chỉ dành cho các doanh nghiệp kinh doanh Internet, không dành cho các lĩnh vực kinh doanh truyền thống
* Chúng ta không cần và cũng không có ngân sách, không có kỹ năng nên không cần quan tâm

## 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. Giờ chúng ta hãy cùng tìm hiểu xem nó là cái gì, và tại sao nó lại rất phát triển và được nhiều người quan tâm đến vậy.

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 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.

* Sở dĩ người ta phát triển NoSQL suất phát từ yêu cầu cần những database 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.
* Lượng dữ liệu mà các hệ thống cần phải xử lý giờ đây ngày 1 lớn. Ví dụ như Google, Facebook phải lưu trữ và xử lý một lượng dữ liệu cực lớn mỗi ngày.

Một số đặc điểm chung:

* High Scalability: Gần như không có một giới hạn cho dữ liệu và người dùng trên hệ thống.
* High Availability: Do chấp nhận sự trùng lặp trong lưu trữ nên nếu một node (commodity machine) nào đó bị chết cũng không ảnh hưởng tới toàn bộ hệ thống.
* Atomicity: Độc lập data state trong các operation.
* Consistency: chấp nhận tính nhất quán yếu, có thể không thấy ngay được sự thay đổi mặc dù đã cập nhật dữ liệu.
* Durability: dữ liệu có thể tồn tại trong bộ nhớ máy tính nhưng đồng thời cũng được lưu trữ lại đĩa cứng.
* 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 cấu hình phần cứng mạnh, đồng nhất.
* 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).

# Phần 2: Cơ sở dữ liệu phân tán

## 2.1. Khái niệm

*Cơ sở dữ liệu phân tán*: Một tuyển tập dữ liệu có quan hệ logic với nhau, được phân bố trên các 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 cầu truy nhập dữ liệu ở nhiều nút thông qua hệ thống truyền thông.

Ví dụ về hệ CSDL phân tán: ATM và Google phân tán theo cách tự nhận biết, một yêu cầu gần server nào thì server đó xử lý. ATM phân tán rộng khắp, Google ở đâu cũng có. Tùy theo người lập trình và cách xử lý mà CSDL được tiến hành phát tán cho hợp lý.

## 2.2. Ưu và nhược điểm của cơ sở dữ liệu phân tán

Ưu điểm của cơ sở dữ liệu phân tán:

* Phù hợp với cấu trúc của tổ chức lớn
* Nâng cao khả năng chia sẻ và tính tự trị địa phương
* Nâng cao tính sẵn sàng
* Nâng cao tính tin cậy
* Nâng cao hiệu năng
* Dễ mở rộng

Nhược điểm của cơ sở dữ liệu phân tán

* Phức tạp
* Thiết kế cơ sở dữ liệu phức tạp hơn
* Khó điều khuyển tính nhất quán dữ liệu
* Khó phát hiện và xử lý lỗi
* Giá thành cao
* Vấn đề bảo mật
* Thiếu chuẩn mực
* Thiếu kinh nghiệm

## 2.3. Kiến trúc của một hệ cơ sở dữ liệu phân tán

Do sự đan dạng, không có kiên trúc nào được công nhận tương đương với kiến trúc 3 mức ANSI/SPARC.

Một kiến trúc tham khảo bao gồm:

* Tập các sơ đồ ngoài toàn cục (Global external schemas)
* Sơ đồ khái niệm toàn cục (Global conceptual schema)
* Sơ đồ phân đoạn (Fragmentation schema) và sơ đồ định vị (Allocation schema)
* Tập các sơ đồ cho mỗi hệ CSDL cục bộ tuân theo tiêu chuẩn 3 mức ANSI/SPARC

*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 hệ tổng 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.

## 2.4. Tính trong suốt của cơ sở dữ liệu phân tán

Các mức trong suốt của hệ cơ sở dữ liệu phân toán

* Trong suốt phân đoạn (fragmentation transparency): Mức 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. 15 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.
* Trong suốt về vị trí (location transparency): Người dùng cuối hoặc người lập trình biết cơ sở dữ liệu phân chia thành các đoạn, tên của các đoạn nhưng không biết vị trí phân bố của các đoạn.
* Trong suốt ánh xạ địa phương (local mapping transparency): Người dùng cuối hoặc người lập trình biết tên các đoạn và vị trí của các đoạn.
* Trong suốt nhân bản (replication transparency): 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.
* Không trong suốt (no transparency)

# Phần 3: Hệ quản trị cơ sở dữ liệu Oracle

Lecture 1:

Không có bài tập thực hành

Lecture 2:

**Câu 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.

CREATE TABLE dept2

(id NUMBER(7) CONSTRAINT test PRIMARY KEY,

name VARCHAR2(25));

**Câu 2.** Populate the DEPT table with data from the DEPARTMENTS table. Include only columns that you need.

INSERT INTO dept

SELECT department\_id , department\_name

FROM departments;

**Câu 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.

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));

**Câu 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.

CREATE TABLE employees2 AS

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

FROM employees

**Câu 5**. Drop the EMP table.



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

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

## Lecture 3

**Câu 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.

CREATE OR REPLACE VIEW employees\_vu

AS

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

FROM employees;

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

SELECT \* FROM employees\_vu

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

SELECT employee, department\_id

from employees\_vu

**Câu 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.

CREATE OR REPLACE VIEW dept50

AS

SELECT employee\_id empno,last\_name employee,department\_id deptno

FROM employees

WHERE department\_id =50

WITH CHECK OPTION CONSTRAINT emp\_dept\_50;

------------------------------------

SELECT \* FROM dept50

------------------------------------

UPDATE dept50

SET deptno =80

WHERE employee = Mohammed;

**Câu 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.

CREATE SEQUENCE DEPT\_ID\_SEQ

INCREMENT BY 10

START WITH 200

MAXVALUE 1000;

DROP SEQUENCE DEPT\_ID\_SEQ

-----

select DEPT\_ID\_SEQ.nextval--Xem gia tri nextval

FROM Dual

INSERT INTO dept

VALUES (DEPT\_ID\_SEQ.NEXTVAL,'Education');

INSERT INTO dept

VALUES (DEPT\_ID\_SEQ.NEXTVAL,'Administration');

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

CREATE SYNONYM emp

FOR employees;

## Lecture 4

**Câu 1**. The HR department needs a query to display all unique job codes from the EMPLOYEES table.

SELECT DISTINCT job\_id FROM EMPLOYEES;

**Câu 2**. 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.

SELECT last\_name || ','|| job\_id AS "Employee and Title"

FROM employees;

**Câu 3**. 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.

SELECT last\_name, salary

FROM employees

WHERE salary BETWEEN 5000 AND 12000

AND department\_id IN (20,50);

**Câu 4**. 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.

SELECT last\_name, salary, commission\_pct

FROM employees

WHERE commission\_pct IS NOT NULL

ORDER BY salary DESC, commission\_pct DESC;

**Câu 5**. Display the last name of all employees who have both an a and an e in their last name.

SELECT last\_name

FROM employees

WHERE last\_name LIKE '%a%'

AND last\_name LIKE '%e%';

**Câu 6**. Display the last name, job, and salary for all employees whose job is SA\_REP or ST\_CLERKand whose salary is not equal to $2,500, $3,500, or $7,000.

SELECT last\_name,job\_id,salary

FROM EMPLOYEES

WHERE job\_id IN ('SA\_REP','ST\_CLERK')

AND salary NOT IN (2500,3500,7000);

## Lecture 5

**Câu 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.

SELECT INITCAP(last\_name) "Name", LENGTH(last\_name) "Length"

FROM employees

WHERE last\_name LIKE ('J%') OR last\_name LIKE('A%') OR last\_name LIKE('M%')

ORDER BY last\_name;

**Câu 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.

SELECT last\_name, ROUND((sysdate - hire\_date)/30) as "months\_worked"

FROM employees

ORDER BY "months\_worked";

**Câu 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.”

SELECT last\_name,

hire\_date,

TO\_CHAR(NEXT\_DAY(ADD\_MONTHS(hire\_date, 6), 'MONDAY'),

'"Monday, the" fmddspth "of" Month"," YYYY') REVIEW

FROM employees;

**Câu 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.

SELECT last\_name, NVL2(commission\_pct, TO\_CHAR(commission\_pct),'no\_commision') "COMM"

FROM employees;

**Câu 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

SELECT job\_id, DECODE(job\_id,

'AD\_PRES', 'A',

'ST\_MAN', 'B',

'IT\_PROG','C',

'SA\_REP','D',

'ST\_CLERCK','E',

0) "GRADE"

FROM employees;

**Câu 6**. 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.

SELECT ROUND(MAX(salary)) "Maximum", ROUND(MIN(salary)) "Minimum",

ROUND (SUM(salary)) "Sum", ROUND (AVG(salary)) "Average"

FROM employees;

**Câu 7**. Modify the query in Exercise 1 to display the minimum, maximum, sum, and average salary for each job type.

SELECT job\_id,ROUND(MAX(salary)) "Maximum", ROUND(MIN(salary)) "Minimum",

ROUND (SUM(salary)) "Sum", ROUND (AVG(salary)) "Average"

from employees

GROUP BY job\_id

**Câu 8**. 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

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

FROM employees;

**Câu 9**. 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.

SELECT manager\_id, MIN(salary)

FROM employees

WHERE manager\_id IS NOT NULL

GROUP BY manager\_id

HAVING MIN(salary)>6000

ORDER BY MIN(salary) DESC;

## Lecture 6

**Câu 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.

SELECT e.last\_name, d.department\_id, d.department\_name

FROM employees e

JOIN departments d

ON e.department\_id = d.department\_id

ORDER BY e.department\_id;

**Câu 2**. A) 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.

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

select e.last\_name "Employee", e.employee\_id "Emp#",

m.last\_name "Manager", m.employee\_id "Mgr#"

FROM employees e JOIN employees m

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

--Cau B

select e.last\_name "Employee", e.employee\_id "Emp#",

m.last\_name "Manager", m.employee\_id "Mgr#"

FROM employees e LEFT OUTER JOIN employees m

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

ORDER BY e.employee\_id;

**Câu 3**. 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.

SELECT e.last\_name "Employee" , e.hire\_date "Emp Hire Date",

m.last\_name "Manager", m.hire\_date "Mgr HIRE DATE"

FROM employees e

JOIN employees m

ON e.manager\_id = m.employee\_id

WHERE e.hire\_date < m.hire\_date;

**Câu 4**. 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.

Select employee\_id ,last\_name, salary

FROM employees

WHERE department\_id IN (SELECT department\_id from employees WHERE last\_name LIKE '%u%')

AND salary > (SELECT AVG(salary) from employees);

**Câu 5**. 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.

SELECT last\_name, department\_id, TO\_CHAR(NULL)

FROM employees

UNION

SELECT TO\_CHAR(NULL), department\_id, department\_name

FROM departments;

**Câu 6**. 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).

SELECT employee\_id,job\_id

FROM employees

INTERSECT

SELECT employee\_id, job\_id

FROM job\_history;

**Câu 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.

SELECT country\_id, country\_name

FROM countries

MINUS

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

FROM locations l JOIN countries c

ON (l.country\_id = c.country\_id);

## Lecture 7

Phần nay không có bài tập

## Lecture 8

**Câu 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

SELECT department\_id,

job\_id,

SUM(salary) sum\_salary

FROM employees

WHERE department\_id < 60

GROUP BY ROLLUP(department\_id, job\_id);

**Câu 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.

SELECT department\_id,

job\_id,

SUM(salary) sum\_salary

FROM employees

WHERE department\_id < 60

GROUP BY CUBE (department\_id, job\_id);

**Câu 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

SELECT department\_id,

job\_id,

manager\_id,

avg(salary) avg\_salary

FROM employees

GROUP BY GROUPING SETS < 129

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

**Câu 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.

SELECT manager\_id,

job\_id,

SUM(salary) sum\_salary

FROM employees

WHERE manager\_id < 100

GROUP BY CUBE ( manager\_id, job\_id );

## Lecture 9

**Câu 1**. Display the last name, department name, and salary of any employee whose salary and commission match the salary and commission of any employee located in location ID 1700.

SELECT e.last\_name, d.department\_name, e.salary

FROM employees e

JOIN departments d

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

WHERE (salary, NVL(commission\_pct,0)) IN (SELECT salary, NVL(commission\_pct,0)

FROM employees e

JOIN departments d

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

WHERE d.location\_id = 1700);

**Câu 2**. Write a query to find all employees who earn more than the average salary in their departments. Display last name, salary, department ID, and the average salary for the department. Sort by average salary.

SELECT last\_name ename, salary, dept.department\_id deptno, dept.dept\_avg\_sal

FROM employees outer

JOIN

(SELECT department\_id, AVG(salary) dept\_avg\_sal

FROM employees

GROUP BY department\_id) dept

ON (outer.department\_id = dept.department\_id)

WHERE salary > (SELECT AVG(salary)

FROM employees

WHERE department\_id = outer. department\_id)

ORDER BY dept.dept\_avg\_sal;

**Câu 3**. Write a query to display the last names of the employees who have one or more coworkers in their departments with later hire dates but higher salaries.

SELECT last\_name

FROM employees outer

WHERE EXISTS (SELECT 'X'

FROM employees inner

WHERE inner.department\_id = outer.department\_id

AND inner.hire\_date > outer.hire\_date

AND inner.salary > outer.salary);

**Câu 4**. Write a query to display the department names of those departments whose total salary cost is above one-eighth (1/8) of the total salary cost of the whole company. Use the WITH clause to write this query. Name the query SUMMARY.

WITH

summary AS(

SELECT d.department\_name, SUM(e.salary) AS dept\_total

FROM employees e, departments d

WHERE e.department\_id = d.department\_id

GROUP BY d.department\_name)

SELECT department\_name, dept\_total

FROM summary

WHERE dept\_total > (SELECT SUM(dept\_total)\*1/8

FROM summary)

ORDER BY dept\_total DESC;

## Lecture 10

Phần này không có bài tập

## Lecture 11

**Câu 1**. Produce a report showing an organization chart for Mourgos’s department. Print last names, salaries, and department IDs.

SELECT last\_name, salary, department\_id

FROM employees

START WITH last\_name = 'Mourgos'

CONNECT BY PRIOR employee\_id = manager\_id;

**Câu 2**. Create a report that shows the hierarchy of the managers for the employee Lorentz. Don’t display Lorentz ,Display his immediate manager first.

SELECT last\_name

FROM employees

WHERE last\_name <> 'Lorentz'

START WITH last\_name = 'Lorentz'

CONNECT BY employee\_id = PRIOR manager\_id;

**Câu 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.

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

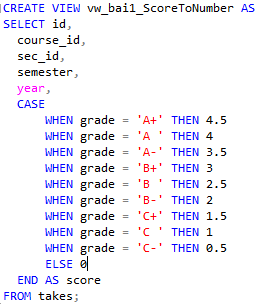
# Phần 4: Bài tập kết thúc môn

Bài 1. Kiểm tra 1 sinh viên đã đủ điều kiện tốt nghiệp chưa biết rằng các điều kiện để một sinh viên tốt nghiệp là:

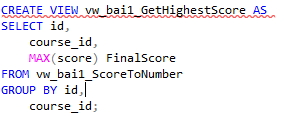
## 

Bài làm:

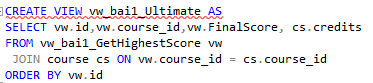
Đầu tiên ta tạo view để chuyển điểm từ chữ sang số, lấy dữ liệu từ bảng takes



Xuất phát từ view trên, ta tạo View chỉ chứa những điểm cao hơn của 1 môn khi 1 sinh viên học cải thiện môn đấy



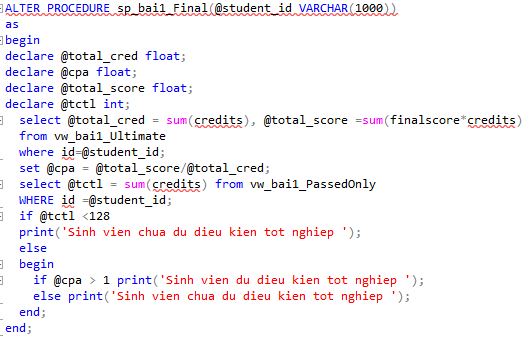
Từ view trên, ta tiếp tục tạo 1 view chứa các trường id sinh viên, id khóa học, điểm lần học cao nhất, và số tín chỉ.



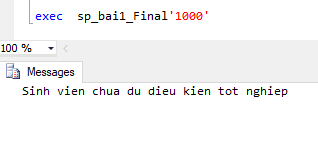
View cuối cùng ta tạo là view chỉ chữa những bản ghi có điểm lần học cao nhất >0.5 (hay điểm chữ >C-) tức là chỉ lấy những môn đạt.



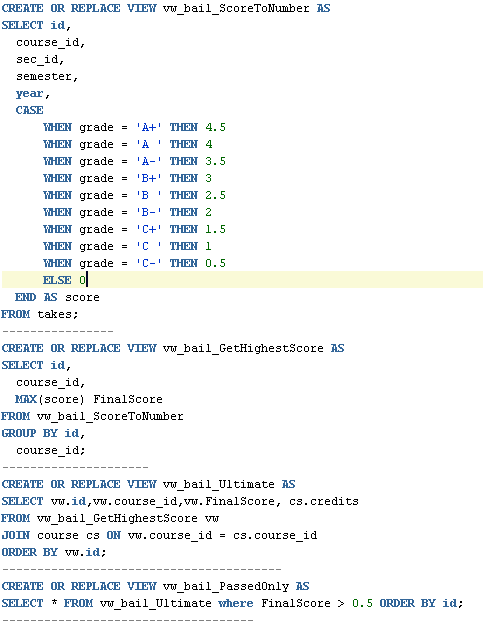
Đến đây ta có thể viết 1 stored procedure với input là mã sinh viên, output là sinh viên đó có đủ điều kiện tốt nghiệp hay không (đủ điều kiện khi tích lũy đủ 128 tín và cpa >=1) sử dụng vw\_bai1\_PassedOnly và vw\_bai1\_Ultimate ở trên.

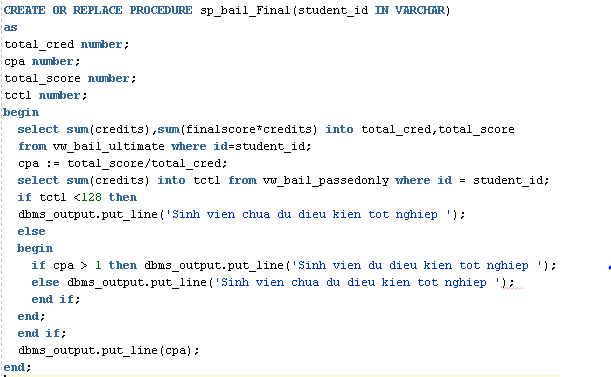


Chạy thử, kết quả



Code Oracle:



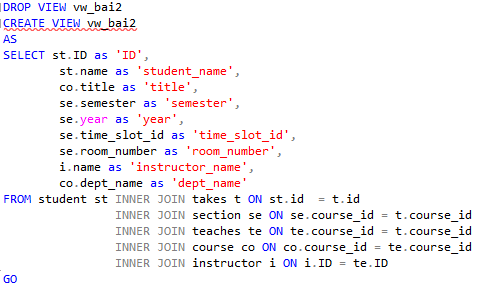


Bài 2: Viết thủ tục SP\_LOC\_DU\_LIEU cho phép nhập vào tên trường bất kỳ và một giá trị của trường (Ví dụ: SP\_LOC\_DU\_LIEU ‘dept\_name’, ‘Physics’ ). Kết quả trả về là dữ liệu sau khi lọc theo giá trị của trường dữ liệu đó.

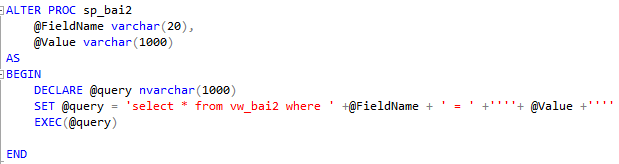
Bảng kết quả trả về gồm các trường: Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng viên, Khoa viện.

Bài làm:

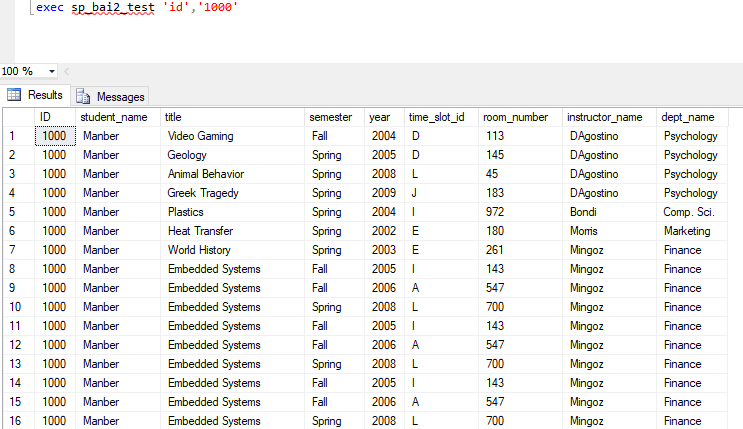
Đầu tiên ta tạo 1 view chứa các trường đề bài yêu cầu:



Rồi chạy thủ tục như yêu cầu đề bài với 2 input là FieldName và Value:



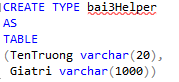
Chạy thử,kết quả



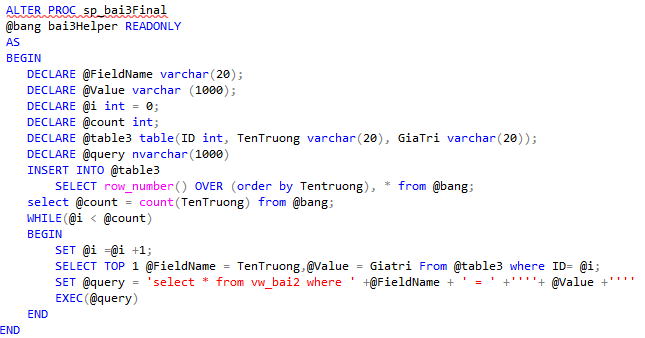
Bài 3. Viết thủ tục SP\_LOC\_DU\_LIEU cho phép nhập vào một biến kiểu table gồm 2 trường: tên trường và một giá trị của trường. Kết quả trả về là dữ liệu sau khi lọc theo danh sách các giá trị của các trường dữ liệu đó.

Bài làm:

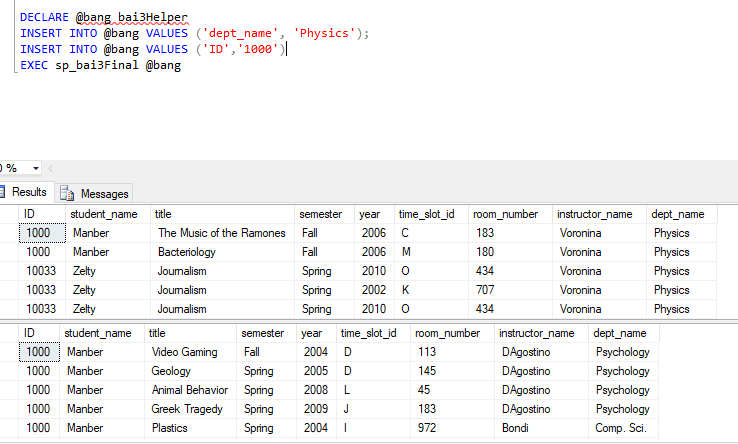
Đầu tiên ta tạo kiểu dữ liệu mới dạng bảng như đề bài yêu cầu:



Sau đó tạo 1 Store Procedure với input là kiểu bai3Helper, sau đó chạy vòng lặp từng dòng trong bảng input để lấy dữ liệu @Fieldname và @Value tương ứng, rồi chạy như bài 2.



Chạy thử, kết quả:



Bài 4. Sinh viên A muốn học môn ‘Mobile Computing’ hỏi A cần phải học qua những môn gì?

Bài làm:

Bài này sử dụng Oracle sẽ đơn giản hơn nhiều so với SQL Server.

Đầu tiên ta tìm mã môn học có tên ‘Mobile Computing’

Code Oracle:



Kết quả có 2 mã:



Cuối cùng ta chạy truy vấn phân cấp để tìm những môn trước:



Kết quả với môn có mã 612:



Tức là để học 612 chỉ cần học 123

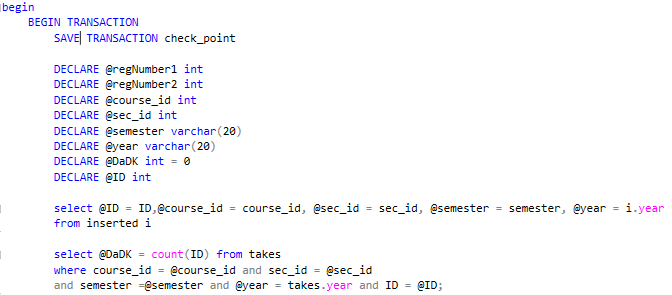
Kết quả với môn có mã 810:

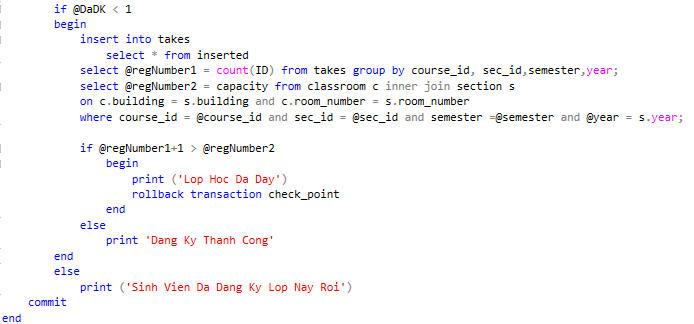


Tức là để học 810 chỉ cần học trước 966

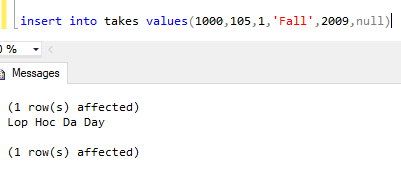
Bài 5. Cài đặt Trigger kiểm tra số lượng sinh viên đăng ký vượt quá sức chứa của phòng. Đưa ra thông báo không thành công khi sinh viên đăng ký môn học. Rollback khi có lỗi xảy ra.

Bài làm:





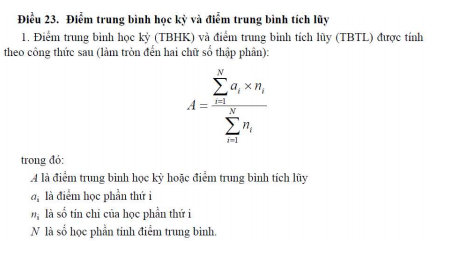
Chạy thử:



Bài 6. Viết thủ tục cho biết kết quả học tập của một sinh viên với:

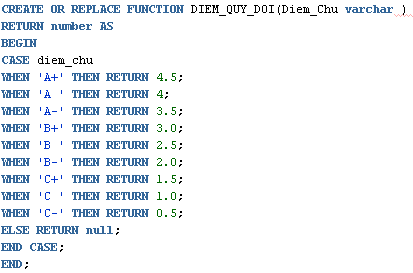
Đầu vào: Mã sinh viên

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.

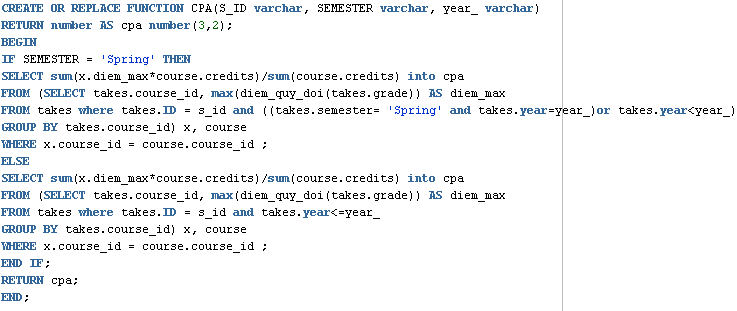


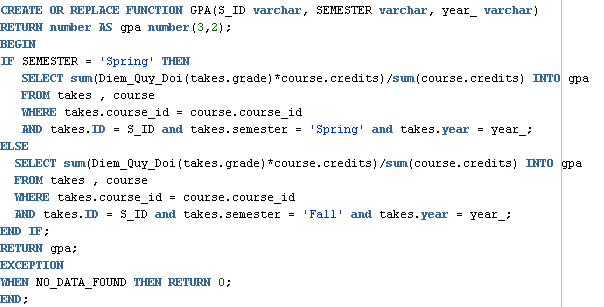
Bài làm:

Đầu tiên ta tạo hàm đổi điểm chữ thành số với input là điểm chữ:

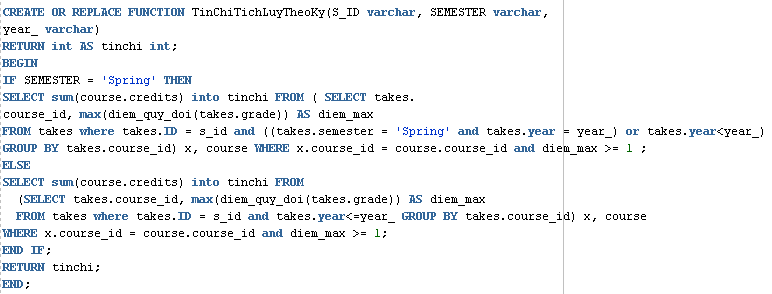


Tiếp theo ta tạo hàm tính CPA.GPA:

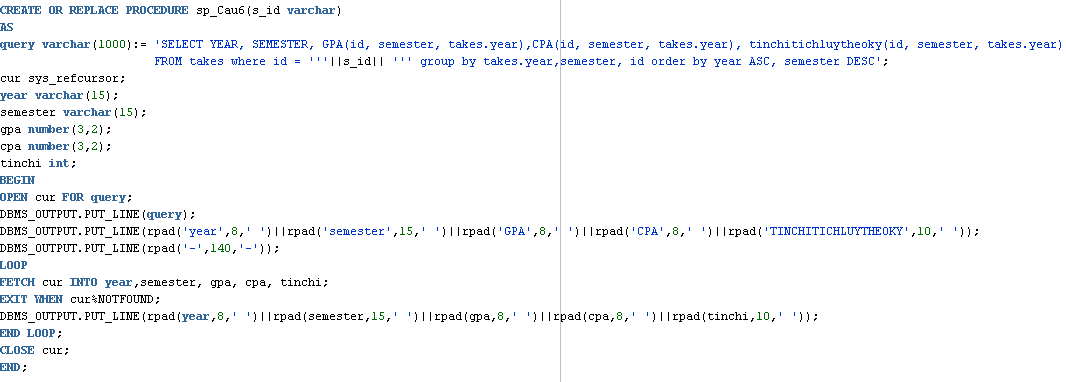




Hàm cần thiết cuối cùng là hàm tính số tctl theo từng kì:



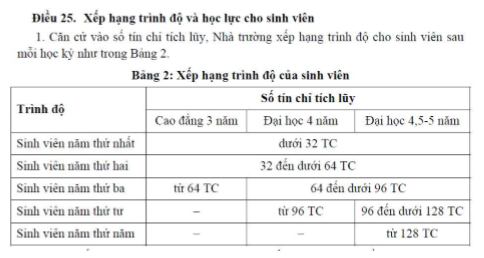
Và hàm cuối cùng:

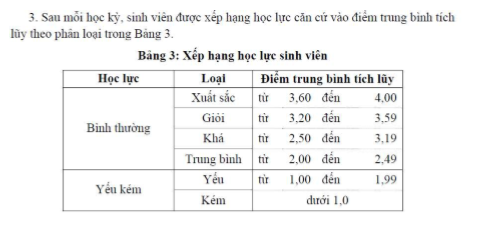


Bài 7.Viết thủ tục đánh giá kết quả học tập của một sinh viên với:

Đầu vào: Mã sinh viên

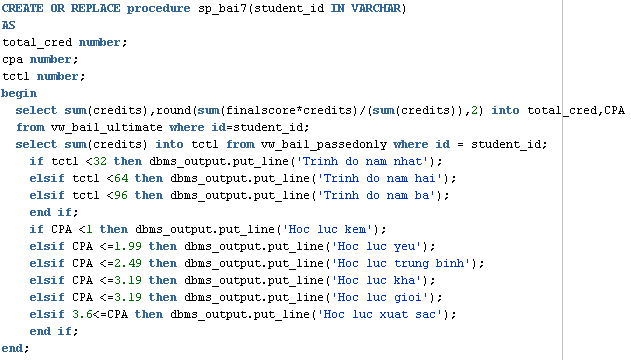
Đầu ra: Xếp hạng trình độ sinh viên và xếp hạng học lực của sinh viên, biết rằng:



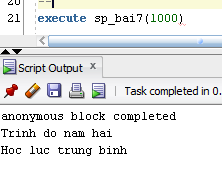


Bài làm:

Sử dụng view vw\_bai1\_PassedOnly và vw\_bai1\_Ultimate đã tạo ở bài 1:



Chạy thử, kết quả:



Bài 8: Đánh chỉ mục các bảng takes, student, advisor. So sánh tốc độ truy vấn sau khi đã thực hiện đánh chỉ mục.

Bài làm:

Ta tạo thêm các bảng takes\_test\_index, student\_test\_index, advisor\_test\_index là bản sao của các bản takes, student, advisor để thử nghiệm việc đánh index



Tạo các index mới cho các bảng vừa tạo:



Khi truy vấn trên bảng takes\_test\_index theo trường grade, tốc độ đã có cải thiện so với bảng takes ban đầu





So với:





Trên SQLServer, sau khi tạo các bảng test và index, ta chạy thử



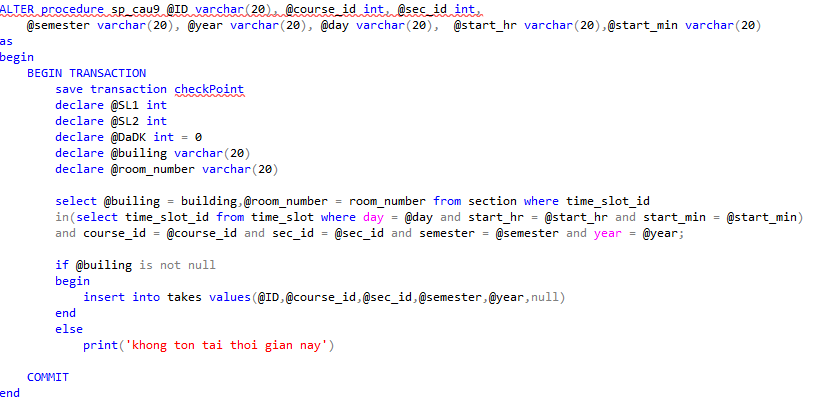


So với





Bài 9: 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.

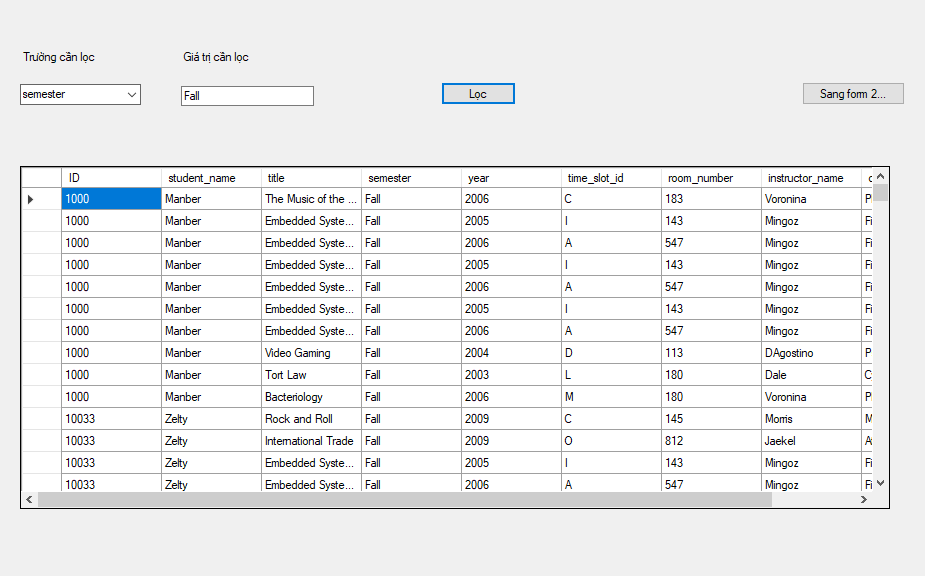


Bài 10. Lập trình ứng dụng SQL nâng cao trên môi trường Windows

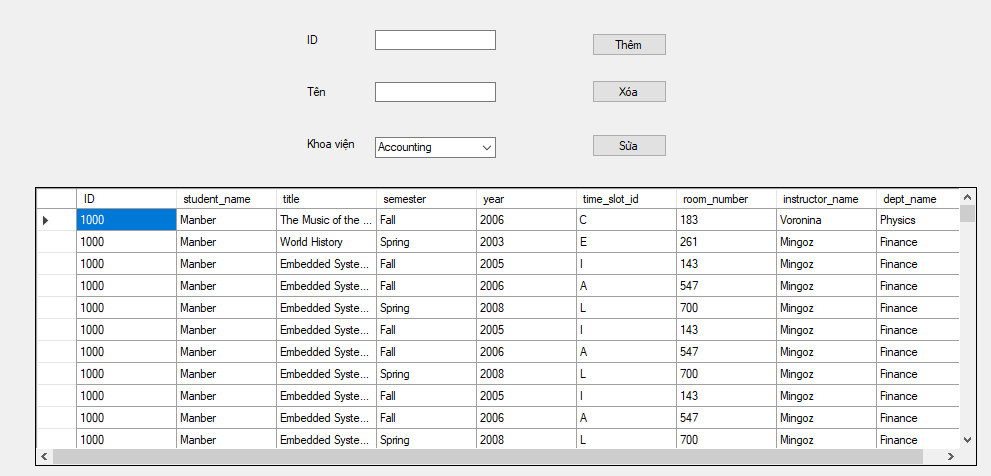
Form01: Chứa 1 Data Grid 1 bộ lọc. Data Grid hiển thị đầy đủ các trường thông tin: Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng viên, Khoa viện. Bộ lọc hỗ trợ lọc theo các trường: Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học. Thực hiện các chức 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 vào cơ sở dữ liệu.

Form 1:



Form 2



# 

# Tài liệu tham khảo

1. Slide bài giảng môn cơ sở dữ liệu nâng cao, Nguyễn Thị Thanh Huyền.
2. Slide bài giảng môn cơ sở dữ liệu nâng cao, Nguyễn Tuấn Dũng.
3. Slide bài giảng về Oracle, Nguyễn Danh Tú