

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: Đào Minh Hoàng

Mã số sinh viên : 20151505

Lớp: : Toán tin 02 - K60

HÀ NỘI – 2019

PHẦN MỞ ĐẦU

Trong cuộc cách mạng công nghiệp 4.0, công nghệ thông tin phát triển như vũ bão. Xuất phát từ sự bùng nổ dữ liệu trong những năm gần đây thông qua mạng internet, khái niệm “dữ liệu lớn” ra đời, kéo theo sự phát triển mạnh mẽ về khoa học dữ liệu với mục đích trích rút thông tin, tri thức nguồn dữ liệu vô hạn nhằm đáp ứng các nhu cầu từ xã hội. Các công nghệ mới về khai phá dữ liệu lớn được ra đời trong thời gian gần đây và đang trở thành một xu hướng tất yếu trong tương lai

Học phần “Cơ sở dữ liệu nâng cao” cung cấp cho sinh viên các kiến thức nền tảng trong việc phát triển và xử lý dữ liệu lớn. Trong tiểu luận này, em xin được trình bày một số vấn đề về công nghệ xử lý dữ liệu, Bố cục của báo cáo gồm phần mở đầu, 4 chương, phần kết luận và tài liệu tài khảo:

- Chương I: Cơ sở dữ liệu lớn
- Chương II: Cơ sở dữ liệu phân tán
- Chương III: Hệ quản trị cơ sở dữ liệu Oracle
- Chương IV: Bài tập kết thúc môn

Em xin chân thành cảm ơn Thầy Nguyễn Danh Tú, Thầy Nguyễn Tuấn Dũng, Cô Nguyễn Thị Thanh Huyền - các thầy cô đã trực tiếp giảng dạy chúng em trong học phần này. Nhờ có kiến thức chuyên sâu và sự tâm huyết, nhiệt tình của các thầy cô, chúng em mới có thể tiếp thu được những kiến thức về công nghệ xử lý dữ liệu trong thời đại mới.

Mặc dù em đã cố gắng hoàn thành bài báo cáo trong phạm vi và khả năng cho phép, do kiến thức, thời gian và kinh nghiệm có hạn nên không tránh khỏi những thiếu sót. Vì vậy em rất mong nhận được sự cảm thông và ý kiến đóng góp của quý thầy cô để bài báo cáo có thể hoàn thiện hơn.

Em xin chân thành cảm ơn!

Mục lục

Phần 1: Cơ sở dữ liệu lớn	5
1. Khái niệm dữ liệu lớn	5
2. Các đặc trưng	5
3. Ứng dụng của Big Data	6
4. Các quan niệm sai lầm về Big Data	7
5. Cơ sở dữ liệu NoSQL	7
Phần 2: Cơ sở dữ liệu phân tán	9
1. Khái niệm	9
2. Ưu và nhược điểm của cơ sở dữ liệu phân tán	9
3. Kiến trúc của một hệ cơ sở dữ liệu phân tán	10
4. Tính trong suốt của cơ sở dữ liệu phân tán	11
Phần 3: Hệ quản trị cơ sở dữ liệu Oracle	12
Lecture 1	12
Lecture 2	12
Lecture 3	13
Lecture 4	15
Lecture 5	17
Lecture 6	19
Lecture 7	21
Lecture 8	22
Lecture 9	23
Lecture 10	25
Lecture 11	25
Phần 4: Bài tập kết thúc môn	27
Bài 1	27
Bài 2	32
Bài 3	36
Bài 4	38
Bài 5	39
Bài 6	41
Bài 7	44

Bài 8	47
Bài 9	48
Bài 10	51
Tài liệu tham khảo	58

Phần 1

Cơ sở dữ liệu lớn

1. Khái niệm dữ liệu lớn

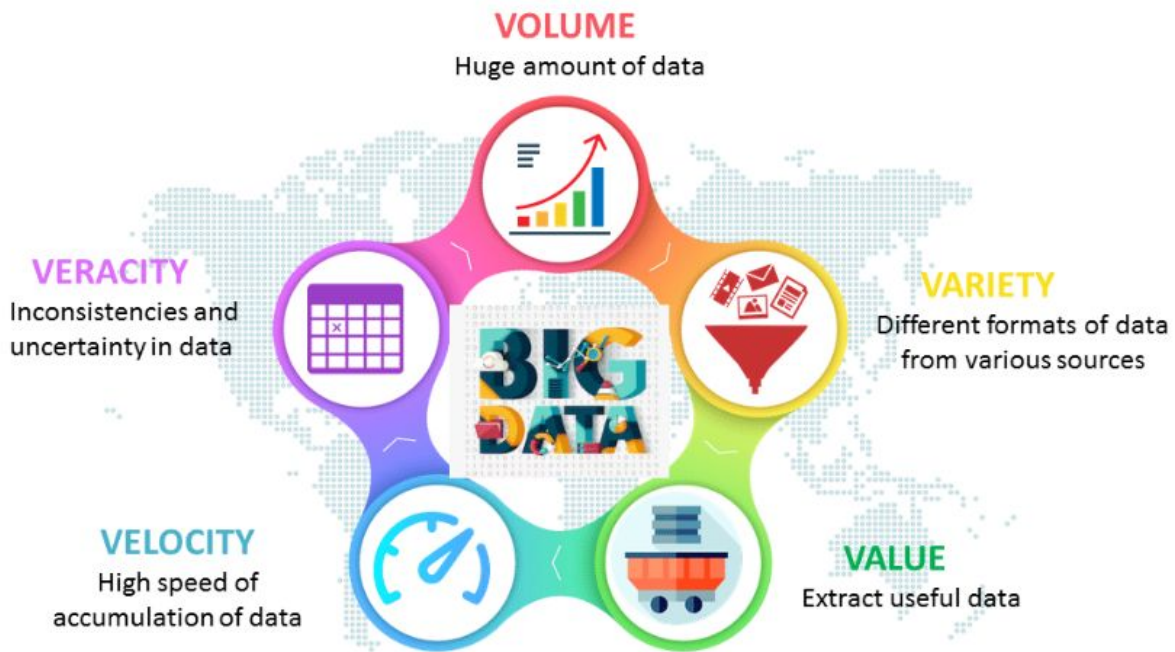
Dữ liệu lớn (Big data) có thể được phân tích để có thông tin chi tiết dẫn đến những quyết định tốt hơn và các động thái kinh doanh chiến lược cũng đi đúng hướng hơn.

Các công nghệ dữ liệu lớn (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 lý dữ liệu truyền thống, và do đó yêu cầu công nghệ xử lý 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 kỹ thuật dữ liệu. Thành thạo, các công nghệ big data còn được sử dụng để implement những kỹ thuật data mining. Tuy nhiên, các công nghệ big data được biết đến nhiều hơn là xử lý dữ liệu trong việc *hỗ trợ* các kỹ thuật data mining và các hoạt động khoa học dữ liệu khác.

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.

2. 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 liệu 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 lý dữ liệu chính xác hơn. Liệu dữ liệu đã cũ rồi hay vẫn còn mới, nó có nhiều giá trị thống kê hay không, nó có tương quan với nhau hay không...

3. 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 lý 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
- ...

4. 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

5. Cơ sở dữ liệu NoSQL

Khi làm việc với database, chúng ta đã quá quen với SQL Server, 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 thể 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 xuất 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

1. Khái niệm

- *Cơ sở dữ liệu phân tán*: Một tuyến 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. 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 khiển 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

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

Do sự đa dạng, không có kiến 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.

4. Tính trong suốt của cơ sở dữ liệu phân tán

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

- 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

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.

```
CREATE TABLE DEPT  
( id NUMBER(7) CONSTRAINT dept_department_id PRIMARY KEY,  
  name VARCHAR2(25));
```

Practice 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;
```

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.

```
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));
```

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.

```
CREATE TABLE employees2 AS
SELECT employee_id id, first_name, last_name, salary, department_id dept_id
FROM employees
```

Practice 5. Drop the EMP table.

```
DROP TABLE employees2;
```

Practice 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

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.

```
CREATE OR REPLACE VIEW employees_vu
```

AS

```
SELECT employee_id, last_name employee, department_id
FROM employees;
```

Practice 2. Confirm that the view works. Display the contents of the EMPLOYEES_VU view.

```
SELECT * FROM employees_vu;
```

Practice 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;
```

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.

```
CREATE OR REPLACE VIEW DEPT50
```

```
(empno, employee, deptno)
```

AS

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE department_id = 50
WITH CHECK OPTION CONSTRAINT dept50_ck;
```

```
-----
```

```
SELECT * FROM dept50;
```

```
-----
```

```
UPDATE dept50
```

```
SET deptno = 80
```

WHERE employee = 'Mohammed';

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.

```
CREATE SEQUENCE DEPT_ID_SEQ
INCREMENT BY 10
START WITH 200
MAXVALUE 1000;
--DROP SEQUENCE DEPT_ID_SEQ
INSERT INTO dept
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Education');
INSERT INTO dept
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Administration');
```

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

```
CREATE SYNONYM emp
FOR employees;
```

Lecture 4

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

```
SELECT DISTINCT job_id FROM EMPLOYEES;
```

Practice 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;

Practice 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 AS 'Employee', salary AS 'Monthly Salary'
```

```
FROM employees
```

```
WHERE salary BETWEEN 5000 AND 12000
```

```
AND department_id IN (20, 50);
```

Practice 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;
```

Practice 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%';
```

Practice 6. 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.

```
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

Practice 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;
```

Practice 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" DESC;
```

Practice 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;
```

Practice 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;
```

Practice 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;
```

Practice 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;
```

Practice 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;

Practice 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;
```

Practice 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

Practice 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;
```

Practice 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);

```

```

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 "Emp#";

```

Practice 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;

```

Practice 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 salary > (SELECT AVG(salary) FROM employees)

AND department_id IN (SELECT department_id FROM employees WHERE last_name
LIKE '%u%');

```

Practice 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;
```

Practice 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;
```

Practice 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 DISTINCT l.country_id, c.country_name
FROM locations l JOIN countries c
ON l.country_id = c.country_id;
```

Lecture 7

Không có bài tập thực hành.

Lecture 8

Practice 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 < 120  
GROUP BY ROLLUP(department_id, job_id);
```

Practice 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,  
       GROUPING(department_id) GRP_DPT,  
       GROUPING(job_id) GRP_JOB  
FROM employees  
WHERE department_id < 120  
GROUP BY ROLLUP(department_id, job_id);
```

Practice 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 manager_id,  
       job_title,
```

```

SUM(salary)
FROM employees e JOIN jobs j
ON e.job_id = j.job_id
WHERE manager_id < 120
GROUP BY CUBE(manager_id, job_title);

```

Practice 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 department_id,
       manager_id,
       job_id,
       SUM(salary)
FROM employees
GROUP BY GROUPING SETS
((department_id, manager_id, job_id),
 (department_id, job_id),
 (manager_id, job_id));

```

Lecture 9

Practice 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);

Practice 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;
```

Practice 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);
```

Practice 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

Không có bài tập thực hành.

Lecture 11

Practice 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;

```

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.

```

SELECT last_name
FROM employees
WHERE last_name <> 'Lorentz'
START WITH last_name = 'Lorentz'

```

```
CONNECT BY employee_id = PRIOR manager_id;
```

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.

```
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:

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

	Thang điểm 4	
	Điểm chữ	Điểm số
ĐẠT	A+	4.5
	A	4.0
	A-	3.5
	B+	3.0
	B	2.5
	B-	2.0
	C+	1.5
	C	1.0
KHÔNG ĐẠT	C-	0.5

Bài làm:

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

```
CREATE OR REPLACE VIEW vw_bail_ScoreToNumber AS
SELECT id,
       course_id,
       sec_id,
       semester,
       year,
       DECODE(grade,
              'A+', 4.5,
              'A', 4.0,
              'B+', 3.5,
              'B', 3.0,
              'B-', 2.0,
              'C+', 1.5,
              'C', 1.0,
              0.5) AS score
FROM 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

```
CREATE OR REPLACE VIEW vw_bail_GetHighestScore AS
SELECT id,
       course_id,
       MAX(score) AS finalscore
FROM vw_bail_ScoreToNumber
GROUP BY id, course_id;
```

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.

```
CREATE OR REPLACE VIEW vw_bail_FullData AS
SELECT vw.id, vw.course_id, finalscore, cs.credits
FROM vw_bail_GetHighestScore vw
JOIN course cs ON vw.course_id = cs.course_id
ORDER BY vw.id;
```

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

```
CREATE OR REPLACE VIEW vw_bail_PassedCourses AS
SELECT * FROM vw_bail_FullData
WHERE finalscore > 0.5
ORDER BY id;
```

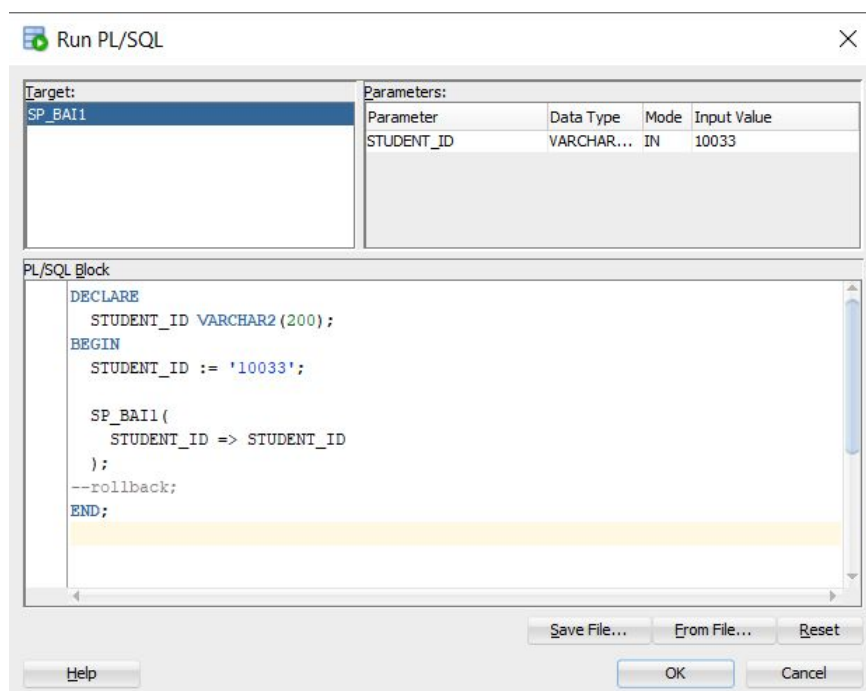
Đến đây ta có thể viết 1 stored procedure với input là mã sinh viên, output là sinh viên đó có đủ điều kiện tốt nghiệp hay không (đủ điều kiện khi tích lũy đủ 120 tín và cpa >=1) sử dụng vw_bail_PassedCourses và vw_bail_FullData ở trên.

```

CREATE OR REPLACE PROCEDURE sp_bail(student_id IN VARCHAR) AS
credits_number NUMBER;
cpa NUMBER;
total_score NUMBER;
tctl NUMBER;
BEGIN
    SELECT SUM(credits), SUM(finalscore*credits) INTO credits_number, total_score
    FROM vw_bail_FullData
    WHERE id = student_id;
    cpa := total_score / credits_number;
    SELECT SUM(credits) INTO tctl
    FROM vw_bail_PassedCourses
    WHERE id = student_id;
    IF tctl < 120 THEN
        DBMS_OUTPUT.PUT_LINE('Sinh vien chua du dieu kien tot nghiep!');
    ELSE
        BEGIN
            IF cpa > 1 THEN
                DBMS_OUTPUT.PUT_LINE('Sinh vien du dieu kien tot nghiep');
            ELSE DBMS_OUTPUT.PUT_LINE('Sinh vien chua du dieu kien tot nghiep!');
            END IF;
        END;
    END IF;
    DBMS_OUTPUT.PUT_LINE(cpa);
END;

```

Chạy thử, kết quả.



Running: IdeConnections%23hoangdb.jpr - Log



Connecting to the database hoangdb.
Sinh vien chua du dieu kien tot nghiep!
1
Process exited.
Disconnecting from the database hoangdb.

Code MSS:

```
CREATE VIEW vw_bai1_ScoreToNumber AS
SELECT id, course_id, sec_id, semester, year
CASE
WHEN grade='A+' THEN 4.5
WHEN grade='A ' THEN 4.0
WHEN grade='A-' THEN 3.5
WHEN grade='B+' THEN 3.0
WHEN grade='B ' THEN 2.5
WHEN grade='B-' THEN 2.0
WHEN grade='C+' THEN 1.5
WHEN grade='C ' THEN 1.0
WHEN grade='C-' THEN 0.5
ELSE 0
END AS score
FROM takes;
```

```
CREATE VIEW vw_bai1_GetHighestScore AS
SELECT id,
course_id,
MAX(score) AS finalscore
FROM vw_bai1_ScoreToNumber
GROUP BY id, course_id;
```

```
CREATE VIEW vw_bai1_FullData AS
SELECT vw.id, vw.course_id, finalscore, cs.credits
FROM vw_bai1_GetHighestScore vw
JOIN course cs ON vw.course_id = cs.course_id
ORDER BY vw.id;
```

```
CREATE VIEW vw_bai1_PassedCourses AS
SELECT * FROM vw_bai1_FullData
WHERE finalscore > 0.5
ORDER BY id;
```

```

ALTER PROCEDURE sp_bai1 (@student_id VARCHAR(1000)) AS
BEGIN
    DECLARE @credits_number FLOAT;
    DECLARE @cpa FLOAT;
    DECLARE @total_score FLOAT;
    DECLARE @tctl INT;
    SELECT @credits_number = SUM(credits), @total_score = SUM(finalscore*credits)
    FROM vw_bai1_FullData
    WHERE id = @student_id;
    set @cpa = @total_score/@credits_number;
    SELECT @tctl=SUM(credits)
    FROM vw_bai1_PassedCourses
    WHERE id = @student_id;
    IF @tctl < 120
        PRINT('Sinh vien chua du dieu kien tot nghiep!');
    ELSE
    BEGIN
        IF @cpa > 1.0
            PRINT('Sinh vien du dieu kien tot nghiep');
        ELSE PRINT('Sinh vien chua du dieu kien tot nghiep!');
    END;
END;

```

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:

```
CREATE OR REPLACE VIEW vw_bai2
(masv, hotensv, namhoc, kyhoc, khoaanhoc, thoigianhoc, phonghoc, giangvien, khoa vien)
AS
SELECT student.ID, student.NAME, section.YEAR, section.SEMESTER, course.TITLE,
       section.SEC_ID, section.BUILDING, instructor.NAME, instructor.DEPT_NAME
FROM student, takes, course, section, teaches, instructor
WHERE student.ID = takes.ID
      AND takes.COURSE_ID = course.COURSE_ID
      AND takes.SEC_ID = section.SEC_ID
      AND takes.COURSE_ID = section.COURSE_ID
      AND takes.SEMESTER = section.SEMESTER
      AND takes.YEAR = section.YEAR
      AND teaches.SEC_ID = section.SEC_ID
      AND teaches.COURSE_ID = section.COURSE_ID
      AND teaches.SEMESTER = section.SEMESTER
      AND teaches.YEAR = section.YEAR
      AND teaches.ID = instructor.ID;
```

Rồi tạo thủ tục như yêu cầu đề bài với 2 input là field_name và field_value:


```

CREATE OR REPLACE PROCEDURE sp_bai2(field_name IN varchar, field_value IN varchar)
AS
query varchar(1000):= 'SELECT * FROM vw_bai2 WHERE ' ;
cur sys_refcursor;
o_masv varchar(50);
o_hotensv varchar(50);
o_namhoc varchar(50);
o_kyhoc varchar(50);
o_khoahoc varchar(50);
o_thoigian varchar(50);
o_phonghoc varchar(50);
o_giangvien varchar(50);
o_khoavien varchar(50);
BEGIN
    DBMS_OUTPUT.PUT_LINE(field_name);
    DBMS_OUTPUT.PUT_LINE(field_value);
    IF field_name IN
    ('MASV','HOTENSV','NAMHOC','KYHOC','KHOAHOC','THOIGIAN','PHONGHOC','GIANGVIEN','KHOAVIEN')
    THEN
        query := query || field_name || '=' || field_value || '';
        DBMS_OUTPUT.PUT_LINE(query);
        OPEN cur FOR query;
        DBMS_OUTPUT.PUT_LINE(rpad('MASV',8,' ')||rpad('HOTENSV',20,'')||rpad('NAMHOC',10,' ')
        ||rpad('KYHOC',10,'')||rpad('KHOAHOC',35,' ')||rpad('THOIGIANHOC',12,' ')
        ||rpad('PHONGHOC',12,'')||rpad('GIANGVIEN',15,' ')||rpad('KHOAVIEN',15,' '));
        DBMS_OUTPUT.PUT_LINE(rpad('-',140,'-'));
        LOOP
            FETCH cur INTO
            o_masv,o_hotensv,o_namhoc,o_kyhoc,o_khoahoc,o_thoigian,o_phonghoc,o_giangvien,o_khoavien;
            EXIT WHEN cur%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(rpad(o_masv,8,' ')||rpad(o_hotensv,20,'')||rpad(o_namhoc,10,' ')
            ||rpad(o_kyhoc,10,' ')||rpad(o_khoahoc,35,' ')||rpad(o_thoigian,12,' ')
            ||rpad(o_phonghoc,12,' ')||rpad(o_giangvien,15,'')||rpad(o_khoavien,15,' '));
        END LOOP;
        CLOSE cur;
    ELSE
        DBMS_OUTPUT.PUT_LINE('NHAP KHONG DUNG');
    END IF;
END

```

Chạy thử, kết quả:

Run PL/SQL

Target: SP_BAI2

Parameters:

Parameter	Data Type	Mode	Input Value
FIELD_NAME	VARCHAR...	IN	MASV
FIELD_VALUE	VARCHAR...	IN	10033

PL/SQL Block

```

DECLARE
    FIELD_NAME VARCHAR2(200);
    FIELD_VALUE VARCHAR2(200);
BEGIN
    FIELD_NAME := 'MASV';
    FIELD_VALUE := '10033';

    SP_BAI2(
        FIELD_NAME => FIELD_NAME,
        FIELD_VALUE => FIELD_VALUE
    );
    --rollback;
END;

```

Save File... From File... Reset

Help OK Cancel

Running: IdeConnections%23hoangdb.jpr - Log

Connecting to the database hoangdb.

MASV
10033

SELECT * FROM vw_bai2 WHERE MASV='10033'

MASV	NAMHOC	KHOAHOC		THOIGIANHOC	GIANGVIEN	KHOAVIEN
10033	2009	Spring	Greek Tragedy	1	Taylor	Psychology
10033	2007	Spring	Graph Theory	1	Fairchild	Psychology
10033	2006	Spring	Graph Theory	2	Stabler	Psychology
10033	2003	Spring	Finite Element Analysis	1	Stabler	Cybernetics
10033	2009	Fall	Rock and Roll	1	Fairchild	Marketing
10033	2010	Spring	The Beatles	1	Saucon	English
10033	2009	Fall	Tort Law	1	Lamberton	Comp. Sci.
10033	2007	Spring	Bankruptcy	1	Taylor	Accounting
10033	2003	Spring	Bankruptcy	2	Taylor	Accounting
10033	2008	Spring	Embedded Systems	3	Bronfman	Finance
10033	2001	Spring	Systems Software	1	Saucon	Athletics
10033	2009	Fall	International Trade	1	Taylor	Athletics
10033	2003	Fall	Care and Feeding of Cats	1	Taylor	Statistics
10033	2006	Spring	Compiler Design	1	Lambeau	Psychology
10033	2010	Spring	Cat Herding	1	Taylor	Athletics
10033	2001	Spring	Arabic	1	Saucon	Biology
10033	2010	Spring	Music of the 50s	1	Lamberton	Geology
10033	2009	Fall	Accounting	1	Whitman	Geology
10033	2001	Spring	Biostatistics	1	Alumni	Finance

Code MSS:

```

DROP VIEW vw_bai2
CREATE VIEW vw_bai2
AS
SELECT st.ID AS 'MASV',
       st.name AS 'HOTENSV',
       co.title AS 'KHOAHOC',
       se.semester AS 'KYHOC',
       se.year AS 'NAMHOC',
       se.time_slot_id AS 'THOIGIAN',
       se.room number AS 'PHONGHOC',
       i.name AS 'GIANGVIEN',
       co.dept name AS 'KHOAVIEN'
FROM student st INNER JOIN takes t ON st.id = t.id
                INNER JOIN section se ON se.course_id = t.course_id
                INNER JOIN teaches te ON te.course_id = t.course_id
                INNER JOIN course co ON co.course_id = te.course_id
                INNER JOIN instructor i ON i.id = te.id

CREATE PROC sp_bai2
    @FieldName VARCHAR(20),
    @Value VARCHAR(1000)
AS
BEGIN
    DECLARE @query NVARCHAR(1000)
    SET @query = 'SELECT * FROM vw_bai2 WHERE ' + @FieldName + ' = '' ' + @Value + ''
    EXEC(@query)
END

```

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:

Tạo object mới chứa field_name, field_value và table mới cho object vừa tạo:

```
CREATE OR REPLACE TYPE varfilter AS
OBJECT (field_name varchar(50),
        field_value varchar(50));
/
CREATE OR REPLACE TYPE tablefilter
IS TABLE OF varfilter;
```

Tạo thủ tục để lọc dữ liệu:

```
CREATE OR REPLACE PROCEDURE sp_bai3(i_table tablefilter)
AS
cur sys_refcursor;
query varchar(1000) := 'SELECT * FROM vw_bai2 WHERE 1=1 ';
o_masv varchar(50);
o_hotensv varchar(50);
o_namhoc varchar(50);
o_kyhoc varchar(50);
o_khoahoc varchar(50);
o_thoigian varchar(50);
o_phonghoc varchar(50);
o_giangvien varchar(50);
o_khoavien varchar(50);
BEGIN
    FOR i IN 1..i_table.LAST
    LOOP
        IF i_table(i).field_name IN
            ('MASV','HOTENSV','NAMHOC','KYHOC','KHOAHOC','THOIGIAN','PHONGHOC','GIANGVIEN','KHOAVIEN')
        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('MASV',8,' ')||rpad('HOTENSV',20,'')||rpad('NAMHOC',10,' ')
    ||rpad('KYHOC',10,'')||rpad('KHOAHOC',35,' ')||rpad('THOIGIANHOC',12,' ')
    ||rpad('PHONGHOC',12,'')||rpad('GIANGVIEN',15,' ')||rpad('KHOAVIEN',15,' '));
    DBMS_OUTPUT.PUT_LINE(rpad('-',140,'-'));
    LOOP
        FETCH cur INTO
        o_masv,o_hotensv,o_namhoc,o_kyhoc,o_khoahoc,o_thoigian,o_phonghoc,o_giangvien,o_khoavien;
        EXIT WHEN cur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(rpad(o_masv,8,' ')||rpad(o_hotensv,20,'')||rpad(o_namhoc,10,' ')
        ||rpad(o_kyhoc,10,' ')||rpad(o_khoahoc,35,' ')||rpad(o_thoigian,12,' ')
        ||rpad(o_phonghoc,12,' ')||rpad(o_giangvien,15,'')||rpad(o_khoavien,15,' '));
    END LOOP;
    CLOSE cur;
END;
```

Chạy thử, kết quả:

Run PL/SQL

Target:

SP_BAI3

Parameters:

Parameter	Data Type	Mode	Input Value
I_TABLE	HOANG6K...	IN	NULL

PL/SQL Block

```

DECLARE
  I_TABLE HOANG6K.tablefilter := tablefilter(varfilter('MASV','10033'),varfilter('NAMHOC','Spring'),varfilter('KHOAHOC','Psychology'));
BEGIN
  -- Modify the code to initialize the variable
  -- I_TABLE := NULL;

  SP_BAI3(
    I_TABLE => I_TABLE
  );
  --rollback;
END;

```

Save File...

From File...

Reset

Help

OK

Cancel

Running: IdeConnections%23hoangdb.jpr - Log

Connecting to the database hoangdb.

MASV	NAMHOC	KHOAHOC	THOIGIANHOC	GIANGVIEN	KHOAVIEN
10033	2009	Spring	Greek Tragedy	1	Taylor Psychology
10033	2007	Spring	Graph Theory	1	Fairchild Psychology
10033	2006	Spring	Graph Theory	2	Stabler Psychology
10033	2003	Spring	Finite Element Analysis	1	Stabler Cybernetics
10033	2009	Fall	Rock and Roll	1	Fairchild Marketing
10033	2010	Spring	The Beatles	1	Saucon English
10033	2009	Fall	Tort Law	1	Lamberton Comp. Sci.
10033	2007	Spring	Bankruptcy	1	Taylor Accounting
10033	2003	Spring	Bankruptcy	2	Taylor Accounting
10033	2008	Spring	Embedded Systems	3	Bronfman Finance
10033	2001	Spring	Systems Software	1	Saucon Athletics
10033	2009	Fall	International Trade	1	Taylor Athletics
10033	2003	Fall	Care and Feeding of Cats	1	Taylor Statistics
10033	2006	Spring	Compiler Design	1	Lambeau Psychology
10033	2010	Spring	Cat Herding	1	Taylor Athletics
10033	2001	Spring	Arabic	1	Saucon Biology
10033	2010	Spring	Music of the 50s	1	Lamberton Geology
10033	2009	Fall	Accounting	1	Whitman Geology
10033	2001	Spring	Biostatistics	1	Alumni Finance
10033	2009	Spring	UNIX System Programming	1	Bronfman Statistics
10033	2002	Spring	Journalism	2	Gates Physics
10033	2006	Spring	Operating Systems	1	Polya Marketing

Bài 4

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

Bài làm:

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

```
SELECT course_id FROM course WHERE title = 'Mobile Computing';
```

Kết quả có 2 mã:

	COURSE_ID
1	612
2	810

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

```
SELECT * FROM prereq
START WITH course_id='612'
CONNECT BY PRIOR prereq_id = course_id;
```

```
SELECT * FROM prereq
START WITH course_id='810'
CONNECT BY PRIOR prereq_id = course_id;
```

Kết quả với môn có mã 612:

	COURSE_ID	PREREQ_ID
1	612	123

Tức là để học 612 chỉ cần học 123.

Kết quả với môn có mã 810:

	COUR...	PREREQ_ID
1	810	966

Tức là để học 810 chỉ cần học 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:

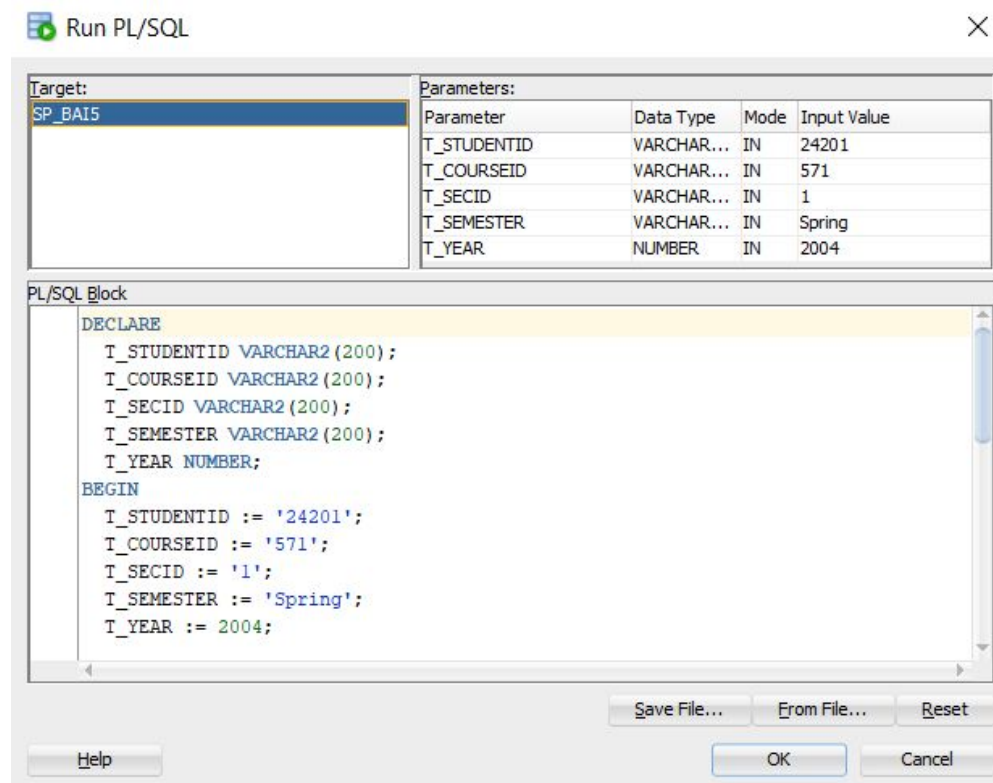
Đầu tiên ta tạo trigger kiểm tra số lượng sinh viên trong lớp từ bảng TAKES:

```
CREATE OR REPLACE TRIGGER CAPACITY_CHECK
BEFORE INSERT ON takes FOR EACH ROW
DECLARE
current_student number(4,0);
room_number varchar(40);
max_student number(4,0);
exception_text varchar(100);
building varchar(40);
CURSOR c(bd varchar, rn varchar) IS
    SELECT capacity FROM classroom
    WHERE building = bd AND room_number = rn;
BEGIN
    --Đếm số lượng sinh viên hiện tại có trong lớp đó
    SELECT COUNT(id) INTO current_student
    FROM (SELECT DISTINCT id
          FROM takes
          WHERE course_id = :NEW.course_id
              AND semester = :NEW.semester
              AND year = :NEW.year
              AND sec_id = :NEW.sec_id);
    --Lấy ra phòng học của lớp đó
    SELECT room_number INTO room_number
    FROM section
    WHERE course_id = :NEW.course_id
        AND sec_id = :NEW.sec_id
        AND semester = :NEW.semester
        AND year = :NEW.year ;
    --Lấy ra toà nhà của lớp đó
    SELECT building INTO building
    FROM section
    WHERE course_id = :NEW.course_id
        AND sec_id = :NEW.sec_id
        AND semester = :NEW.semester
        AND year = :NEW.year ;
    --Lấy ra sức chứa tối đa của phòng số room_number trong toà nhà building
    OPEN c(building, room_number);
    FETCH c INTO max_student;
    --Kiểm tra lớp còn chỗ trống hay không
    IF (current_student >= max_student) THEN
        exception_text := 'HET CHO TRONG';
        RAISE_APPLICATION_ERROR(-20001, exception_text);
    END IF;
    EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('KHONG TON TAI LOP NAY');
END;
```

Tạo thủ tục để đăng ký lớp:

```
CREATE OR REPLACE PROCEDURE sp_bai5
(t_studentID varchar,t_courseID varchar,t_secID varchar,t_semester varchar,t_YEAR number)
AS
MAX_CLASSROOM_CAPACITY EXCEPTION;
PRAGMA EXCEPTION_INIT(MAX_CLASSROOM_CAPACITY,-20001);
BEGIN
    INSERT INTO takes(id, course_id, sec_id, semester, YEAR)
    VALUES (t_studentID, t_courseID, t_secID, t_semester, t_YEAR);
    DBMS_OUTPUT.PUT_LINE('DANG KY THANH CONG');
    EXCEPTION
    WHEN MAX_CLASSROOM_CAPACITY THEN
    DBMS_OUTPUT.PUT_LINE('LOP HET CHO TRONG');
    ROLLBACK;
END;
```

Chạy thử:



Running: ldeConnections%23hoangdb.jpr - Log



Connecting to the database hoangdb.

LOP HET CHO TRONG

Process exited.

Disconnecting from the database hoangdb.

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.

Điều 23. Điểm trung bình học kỳ và điểm trung bình tích lũy

1. Điểm trung bình học kỳ (TBHK) và điểm trung bình tích lũy (TBTL) được tính theo công thức sau (làm tròn đến hai chữ số thập phân):

$$A = \frac{\sum_{i=1}^N a_i \times n_i}{\sum_{i=1}^N n_i}$$

trong đó:

A là điểm trung bình học kỳ hoặc điểm trung bình tích lũy

a_i là điểm học phần thứ i

n_i là số tín chỉ của học phần thứ i

N là số học phần tính điểm trung bình.

Bài làm:

Đầu tiên ta đánh số thứ tự cho kì học trong năm:

```
CREATE OR REPLACE VIEW vw_bai6_ConvertSemester AS
SELECT id,
       course_id,
       sec_id,
       semester,
       year,
       score,
       CASE
         WHEN semester = 'Fall' THEN CONCAT( year, '2' )
         WHEN semester = 'Spring' THEN CONCAT( year, '1' )
       END AS semester_number
FROM vw_bai6_ScoreToNumber;
```

Tạo view tổng hợp kết quả của sinh viên:

```
CREATE OR REPLACE VIEW vw_bai6_ScoreData AS
SELECT hk.id, st.name, hk.course_id, hk.Sec_id, hk.semester, hk.year, hk.score, c.credits, hk.semester_number
FROM vw_bai6_ConvertSemester hk
JOIN course c ON c.course_id = hk.course_id
JOIN student st ON st.id = hk.id;
```

Tạo hàm tính số lượng tín chỉ tích lũy:

```

CREATE OR REPLACE FUNCTION func_tctl(student_id NUMBER, semester_max NUMBER)
RETURN FLOAT IS FunctionResult FLOAT;
BEGIN
    SELECT SUM(pn.credits) INTO FunctionResult
    FROM (SELECT th.id, th.name, th.course_id, th.credits, MAX(score) as max_point
          FROM vw_bai6_ScoreData th
          WHERE TO_NUMBER(th.semester_number) <= TO_NUMBER(semester_max)
          GROUP BY th.course_id, th.name, th.credits, th.id) pn
    WHERE pn.id = student_id
    AND pn.max_point >= 1
    GROUP BY pn.id, pn.name;
    RETURN FunctionResult;
END func_tctl;

```

Tạo hàm tính CPA:

```

CREATE OR REPLACE FUNCTION func_cpa(student_id NUMBER, semester_max NUMBER)
RETURN FLOAT IS FunctionResult FLOAT;
BEGIN
    SELECT ROUND(SUM(pn.max_point*pn.credits)/SUM(pn.credits), 2) INTO FunctionResult
    FROM
        (SELECT th.id, th.name, th.course_id, th.credits, MAX(score) AS max_point
         FROM vw_bai6_ScoreData th
         WHERE TO_NUMBER(th.semester_number) <= TO_NUMBER(semester_max)
         GROUP BY th.course_id, th.name, th.credits, th.id) pn
    WHERE pn.id = student_id
    GROUP BY pn.id, pn.name;
    RETURN (FunctionResult);
END func_cpa;

```

Tạo thủ tục show kết quả học tập của sinh viên theo từng kì:

```

CREATE OR REPLACE PROCEDURE sp_bai6
(student_id NUMBER, mycursor OUT SYS_REFCURSOR )
IS
BEGIN
    OPEN mycursor FOR
    SELECT th.id, th.name, th.semester, th.year, th.semester_number,
           ROUND(SUM(th.score*th.credits)/SUM(th.credits) , 2) AS gpa, SUM(th.credits),
           func_cpa(student_id,th.semester_number) AS cpa,
           func_tctl(student_id,th.semester_number) AS tctl
    FROM vw_bai6_ScoreData th
    WHERE id = student_id
    GROUP BY th.semester_number, th.name, th.semester, th.year, th.id
    ORDER BY th.semester_number ASC;
END;

```

Chạy thử, kết quả:

Run PL/SQL
✕

Target:
SP_BAI6

Parameters:

Parameter	Data Type	Mode	Input Value
STUDENT_ID	NUMBER	IN	1232
MYCURSOR	SYS_REFC...	OUT	N/A

PL/SQL Block

```

DECLARE
  STUDENT_ID NUMBER;
  MYCURSOR SYS_REFCURSOR;
BEGIN
  STUDENT_ID := 1232;

  SP_BAI6(
    STUDENT_ID => STUDENT_ID,
    MYCURSOR => MYCURSOR
  );
  /* Legacy output:
  DBMS_OUTPUT.PUT_LINE('MYCURSOR = ' || MYCURSOR);
  */

```

Save File... From File... Reset

Help OK Cancel

Output Variables - Log									
Variable	Value								
MYCU...	ID	NAME	SEMESTER	YEAR	SEMESTER_...	GPA	SUM(TH.CR...	CPA	TCTL
	1232	Marcus	Spring	2001	20011	0,5	3	0,5	
	1232	Marcus	Fall	2001	20012	2	3	1,25	3
	1232	Marcus	Spring	2003	20031	2	6	1,63	9
	1232	Marcus	Fall	2003	20032	0,93	7	1,37	12
	1232	Marcus	Spring	2004	20041	0,5	7	1,13	12
	1232	Marcus	Fall	2005	20052	4,5	4	1,58	16
	1232	Marcus	Fall	2006	20062	3,43	7	1,93	23
	1232	Marcus	Fall	2007	20072	0,5	4	1,79	23
	1232	Marcus	Spring	2008	20081	3,5	4	1,94	27
	1232	Marcus	Fall	2009	20092	4,5	7	2,29	34
	1232	Marcus	Spring	2010	20101	2	3	2,27	37
	1232	Marcus	Fall	2010	20102	1,14	7	2,15	40

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:

Điều 25. Xếp hạng trình độ và học lực cho sinh viên

1. Căn cứ vào số tín chỉ tích lũy, Nhà trường xếp hạng trình độ cho sinh viên sau mỗi học kỳ như trong Bảng 2.

Bảng 2: Xếp hạng trình độ của sinh viên

Trình độ	Số tín chỉ tích lũy		
	Cao đẳng 3 năm	Đại học 4 năm	Đại học 4,5-5 năm
Sinh viên năm thứ nhất	dưới 32 TC		
Sinh viên năm thứ hai	32 đến dưới 64 TC		
Sinh viên năm thứ ba	từ 64 TC	64 đến dưới 96 TC	
Sinh viên năm thứ tư	–	từ 96 TC	96 đến dưới 128 TC
Sinh viên năm thứ năm	–	–	từ 128 TC

3. Sau mỗi học kỳ, sinh viên được xếp hạng học lực căn cứ vào điểm trung bình tích lũy theo phân loại trong Bảng 3.

Bảng 3: Xếp hạng học lực sinh viên

Học lực	Loại	Điểm trung bình tích lũy
Bình thường	Xuất sắc	từ 3,60 đến 4,00
	Giỏi	từ 3,20 đến 3,59
	Khá	từ 2,50 đến 3,19
	Trung bình	từ 2,00 đến 2,49
Yếu kém	Yếu	từ 1,00 đến 1,99
	Kém	dưới 1,0

Bài làm:

Tạo thủ tục kiểm tra sử dụng view vw_PassedCourses và vw_FullData ở bài 1:

```

CREATE OR REPLACE PROCEDURE sp_bai7(student_id IN VARCHAR)
AS
credits_number NUMBER;
cpa NUMBER;
tctl NUMBER;
BEGIN
    SELECT SUM(credits), ROUND(SUM(finalscore)/SUM(credits), 2) INTO credits_number, cpa
    FROM vw_bail_FullData
    WHERE id = student_id;
    SELECT SUM(credits) INTO tctl
    FROM vw_bail_PassedCourses
    WHERE id = student_id;
    IF tctl < 32 THEN DBMS_OUTPUT.PUT_LINE('Trinh do nam nhat');
    ELSIF tctl < 64 THEN DBMS_OUTPUT.PUT_LINE('Trinh do nam hai');
    ELSIF tctl < 96 THEN DBMS_OUTPUT.PUT_LINE('Trinh do nam ba');
    ELSIF tctl < 128 THEN DBMS_OUTPUT.PUT_LINE('Trinh do nam bon');
    ELSIF tctl < 32 THEN DBMS_OUTPUT.PUT_LINE('Trinh do nam nhat');
    ELSE DBMS_OUTPUT.PUT_LINE('Trinh do nam nhat');
    END IF;
    IF cpa < 1.0 THEN DBMS_OUTPUT.PUT_LINE('Hoc luc kem');
    ELSIF cpa < 2.0 THEN DBMS_OUTPUT.PUT_LINE('Hoc luc yeu');
    ELSIF cpa < 2.5 THEN DBMS_OUTPUT.PUT_LINE('Hoc luc trung binh');
    ELSIF cpa < 3.2 THEN DBMS_OUTPUT.PUT_LINE('Hoc luc kha');
    ELSIF cpa < 3.6 THEN DBMS_OUTPUT.PUT_LINE('Hoc luc gioi');
    ELSE DBMS_OUTPUT.PUT_LINE('Hoc luc xuất sắc');
    END IF;
END;

```

Chạy thử, kết quả:

Run PL/SQL
 ✕

Target:

SP_BAI7

Parameters:

Parameter	Data Type	Mode	Input Value
STUDENT_ID	VARCHAR...	IN	1232

PL/SQL Block

```

DECLARE
  STUDENT_ID VARCHAR2(200);
BEGIN
  STUDENT_ID := '1232';

  SP_BAI7(
    STUDENT_ID => STUDENT_ID
  );
--rollback;
END;

```

Save File... From File... Reset

Help OK Cancel

Running: IdeConnections%23hoangdb.jpr - Log

```

Connecting to the database hoangdb.
Trinh do nam hai
Hoc luc kem
Process exited.
Disconnecting from the database hoangdb.

```


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:

Tạo 3 bảng copy để test:

```
CREATE TABLE takes_copy AS SELECT * FROM takes;
CREATE TABLE student_copy AS SELECT * FROM student;
CREATE TABLE advisor_copy AS SELECT * FROM advisor;
```

Tạo index cho 1 trường có trong mỗi bảng đã tạo:

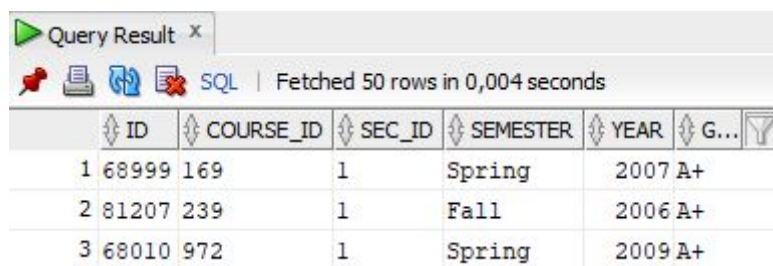
```
CREATE INDEX idx_takes_grade ON takes_copy(grade);
CREATE INDEX idx_student_id ON student_copy(id);
CREATE INDEX idx_advisor_id ON advisor_copy(s_id);
```

Chạy thử với index của bảng TAKES_COPY và so sánh với bảng gốc TAKES:

```
SELECT * FROM takes WHERE grade='A+';
SELECT * FROM takes_copy WHERE grade='A+';
```

Kết quả chạy:

Bảng TAKES:

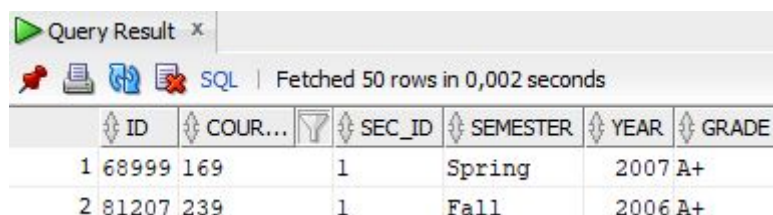


Query Result x

SQL | Fetched 50 rows in 0,004 seconds

ID	COURSE_ID	SEC_ID	SEMESTER	YEAR	G...
1 68999	169	1	Spring	2007	A+
2 81207	239	1	Fall	2006	A+
3 68010	972	1	Spring	2009	A+

Bảng TAKES_COPY:



Query Result x

SQL | Fetched 50 rows in 0,002 seconds

ID	COUR...	SEC_ID	SEMESTER	YEAR	GRADE
1 68999	169	1	Spring	2007	A+
2 81207	239	1	Fall	2006	A+

Tuy nhiên vì dữ liệu ít nên tốc độ thực thi là rất nhỏ, không ổn định trong nhiều lần chạy nên kết quả chỉ mang tính so sánh hơn kém đơn thuần chứ không thể tính toán hiệu năng trên kết quả này.

Code MSS:

```
CREATE TABLE takes_copy AS SELECT * FROM takes;
CREATE TABLE student_copy AS SELECT * FROM student;
CREATE TABLE advisor_copy AS SELECT * FROM advisor;

CREATE INDEX idx_takes_grade ON takes_copy(grade);
CREATE INDEX idx_student_id ON student_copy(id);
CREATE INDEX idx_advisor_id ON advisor_copy(s_id);

--Testing
SET STATISTICS TIME ON
SELECT * FROM takes WHERE grade = 'A+'
SET STATISTICS TIME ON
SELECT * FROM takes_copy WHERE grade = 'A+'
```

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 làm:

Code Oracle:

```
CREATE OR REPLACE PROCEDURE sp_bai9
    (student_id VARCHAR, course_id VARCHAR, sec_id VARCHAR, semester VARCHAR, year NUMBER,
    day VARCHAR, start_hr VARCHAR, start_min VARCHAR)
AS
    bd VARCHAR2(15);
    rn VARCHAR2(7);
BEGIN
    SELECT building, room_number INTO bd, rn
    FROM section
    WHERE time_slot_id IN
        (SELECT time_slot_id
        FROM time_slot
        WHERE day=day
            AND start_hr=start_hr
            AND start_min=start_min)
        AND course_id=course_id
        AND sec_id=sec_id
        AND semester=semester
        AND year=year;
    IF bd IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('Không tồn tại lớp học');
        RETURN;
    END IF;
    SET TRANSACTION READ WRITE NAME 'Insert takes';
    INSERT INTO takes_copy (id, course_id, sec_id, semester, year)
    VALUES (student_id, course_id, sec_id, semester, year);
    EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Đã có lỗi, thực hiện Rollback');
        ROLLBACK;
END;
```

Code MSS:

```

ALTER PROCEDURE sp_bai9 @id VARCHAR(20), @course_id INT, @sec_id INT, @semester VARCHAR(20),
    @year VARCHAR(20), @day VARCHAR(20), @start_hr VARCHAR(20), @start_min VARCHAR(20)
AS
BEGIN
    BEGIN TRANSACTION
        SAVE TRANSACTION checkpoint
        DECLARE @bd VARCHAR(20)
        DECLARE @rn VARCHAR(20)
        SELECT @bd = building, @rn = room_number
        FROM section
        WHERE time_slot_id IN
            (SELECT time_slot_id FROM time_slot WHERE day=@day
                AND start_hr=@start_hr AND start_min=@start_min)
            AND course_id=@course_id AND sec_id=@sec_id
            AND semester=@semester AND year=@year;

        IF @bd IS NOT NULL
        BEGIN
            INSERT INTO takes VALUES(@id, @course_id, @sec_id, @semester, @year, NULL)
        END
        ELSE PRINT('Khong ton tai thoi gian nay')
    COMMIT
END

```

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.

Bài làm:

Form 1:

Giao diện chính khi khởi động, người dùng có thể lọc dữ liệu theo 5 trường:

- Mã sinh viên
- Họ tên sinh viên
- Năm học
- Kỳ học
- Khóa học

Cơ sở dữ liệu nâng cao 20191									
Bộ Lọc	MASV	HOTENSV	NAMHOC	KYHOC	KHOAHOC	THOIGIANHOC	PHONGHOC	GIANGVIEN	KHOAVIEN
Mã Sinh Viên	65901	Shishkin	2003	Fall	Sanitary Engineer...	1	Saucon	Tung	Athletics
	24932	Segars	2003	Spring	African History	1	Saucon	Dale	Cybernetics
Họ Tên Sinh Viên	61332	Canon	2007	Spring	The Music of the ...	1	Saucon	Lembr	Accounting
	73492	Hwang	2004	Spring	How to Groom yo...	1	Garfield	Ullman	Accounting
Năm Học	65715	Novak	2010	Fall	Environmental Law	1	Fairchild	Lembr	Accounting
	58300	Lum	2007	Spring	Bankruptcy	1	Taylor	Ullman	Accounting
Kỳ Học	760	Liedm	2004	Spring	Plastics	1	Power	Bondi	Comp. Sci.
	69730	Peip	2010	Fall	International Trade	1	Chandler	Moris	Marketing
Khóa Học	94836	Fuller	2006	Fall	Multimedia Design	1	Lamberton	Lent	Mech. Eng.
	49391	Rammer	2003	Fall	Care and Feedin...	1	Taylor	Atanassov	Statistics
	48850	Wehen	2001	Spring	Systems Software	1	Saucon	Bawa	Athletics
	68999	Greve	2007	Spring	Marine Mammals	1	Gates	Gustafsson	Elec. Eng.
	53469	Fuji	2007	Spring	Marine Mammals	1	Gates	Gustafsson	Elec. Eng.
	74016	Moei	2004	Spring	International Prac...	1	Gates	Bondi	Comp. Sci.
	49073	Bonvin	2001	Fall	Aquatic Chemistry	1	Taylor	Dale	Cybernetics
	52866	Loull	2004	Spring	Death and Taxes	1	Lamberton	Moris	Marketing
	83314	Chow	2009	Fall	Tort Law	1	Lamberton	Boumer	Comp. Sci.
	70359	Lorinczi	2003	Spring	Mechanics	1	Chandler	D'Agostino	Psychology
	88993	Palaniswami	2010	Fall	International Trade	1	Chandler	Moris	Marketing
	97042	Bhargava	2002	Fall	Drama	1	Polya	Liley	Languages
	33460	Leonard	2005	Fall	FOCAL Program...	1	Whitman	D'Agostino	Psychology
	81207	Masri	2006	Fall	The Music of the ...	1	Taylor	Voronina	Physics
	68010	Blecken	2009	Spring	Greek Tragedy	1	Taylor	D'Agostino	Psychology
	15517	Anis	2007	Fall	Journalism	1	Lamberton	Romero	Astronomy
	8957	Walker	2006	Fall	The Music of Do...	1	Main	Gustafsson	Elec. Eng.
	8986	Maesf	2002	Fall	Differential Geom...	1	Saucon	Dale	Cybernetics
	7956	Brandisd	2010	Spring	Cat Herding	1	Taylor	Tung	Athletics

Thử nghiệm lọc theo 3 trường cùng một lúc:

Ấn vào nút ‘QUẢN LÝ THÔNG TIN SINH VIÊN’ sẽ hiện ra form sau:

Thông Tin Sinh Viên

Tim Theo MASV

SEARCH

Mã Sinh Viên

Họ Tên Sinh Viên

Phân Viện Sinh Viên

Psychology

Tin Chí Tích Lũy

ADD

EDIT

DEL

	MASV	HOTENSV	KHOAVIEN	TICHLUY
▶	68999	Greve	Psychology	113
*				

Thông Tin Sinh Viên

Tìm Theo MASV

SEARCH

Mã Sinh Viên

Họ Tên Sinh Viên

Phân Viện Sinh Viên

Tín Chỉ Tích Lũy

ADD EDIT DEL

MASV	HOTENSV	KHOAVIEN	TICHLUY
68999	Greve	Math	113

Kết quả sau khi tìm kiếm lại sinh viên 68999:

Thông Tin Sinh Viên

Tim Theo MASV

68999

SEARCH

Mã Sinh Viên

68999

Họ Tên Sinh Viên

Greve

Phân Viện Sinh Viên

Math

Tín Chỉ Tích Lũy

113

ADD

EDIT

DEL

	MASV	HOTENSV	KHOAVIEN	TICHLUY
▶	68999	Greve	Math	113
*				

Form2:

Giao diện khi chuyển sang tab của form 2:

Cơ sở dữ liệu năng cao 20191

Bộ Lọc

Lọc Theo:

Giá Trị

FILTER

Refresh

ĐĂNG KÝ HỌC TẬP

MASV	HOTENSV	NAMHOC	KYHOC	KHOAHOC	THOIGIANHOC	PHONGHOC	GIANGVIEN	KHOAVIEN
65901	Shahkin	2003	Fall	Sanitary Engineer...	1	Saucon	Tung	Athletics
24932	Segars	2003	Spring	African History	1	Saucon	Dale	Cybetetics
61332	Canon	2007	Spring	The Music of the ...	1	Saucon	Lembr	Accounting
73492	Hwang	2004	Spring	How to Groom yo...	1	Garfield	Ullman	Accounting
65715	Novak	2010	Fall	Environmental Law	1	Fairchild	Lembr	Accounting
58300	Lum	2007	Spring	Bankruptcy	1	Taylor	Ullman	Accounting
760	Liedm	2004	Spring	Plastics	1	Power	Bondi	Comp. Sci.
69730	Peip	2010	Fall	International Trade	1	Chandler	Morris	Marketing
94836	Fuller	2006	Fall	Multimedia Design	1	Lamberton	Lent	Mech. Eng.
49391	Rammer	2003	Fall	Care and Feedin...	1	Taylor	Atanassov	Statistics
48850	Wehen	2001	Spring	Systems Software	1	Saucon	Bawa	Athletics
68999	Greve	2007	Spring	Marine Mammals	1	Gates	Gustafsson	Elec. Eng.
53469	Fuji	2007	Spring	Marine Mammals	1	Gates	Gustafsson	Elec. Eng.
74016	Moei	2004	Spring	International Prac...	1	Gates	Bondi	Comp. Sci.
49073	Bonvin	2001	Fall	Aquatic Chemistry	1	Taylor	Dale	Cybetetics
52866	Loull	2004	Spring	Death and Taxes	1	Lamberton	Morris	Marketing
83314	Chow	2009	Fall	Tort Law	1	Lamberton	Boumier	Comp. Sci.
70359	Lorinczi	2003	Spring	Mechanics	1	Chandler	D'Agostino	Psychology
88993	Palaniswami	2010	Fall	International Trade	1	Chandler	Morris	Marketing
97042	Bhargava	2002	Fall	Drama	1	Polya	Liley	Languages
33460	Leonard	2005	Fall	FOCAL Program...	1	Whitman	D'Agostino	Psychology
81207	Maari	2006	Fall	The Music of the ...	1	Taylor	Voronina	Physics
68010	Blecken	2009	Spring	Greek Tragedy	1	Taylor	D'Agostino	Psychology
15517	Anis	2007	Fall	Journalism	1	Lamberton	Romero	Astronomy
8957	Walker	2006	Fall	The Music of Do...	1	Main	Gustafsson	Elec. Eng.
8986	Maesf	2002	Fall	Differential Geom...	1	Saucon	Dale	Cybetetics
7956	Brandsd	2010	Spring	Cat Herding	1	Taylor	Tung	Athletics

Thử nghiệm lọc theo một trường là MSSV với giá trị 68999:

Cơ sở dữ liệu năng cao 20191

Bộ Lọc

Lọc Theo:

MASV

Giá Trị

68999

FILTER

F5

ĐĂNG KÝ HỌC TẬP

MASV	HOTENSV	NAMHOC	KYHOC	KHOAHOC	THOIGIANHOC	PHONGHOC	GIANGVIEN	KHOAVIEN
68999	Greve	2009	Spring	Greek Tragedy	1	Taylor	D'Agostino	Psychology
68999	Greve	2007	Fall	Mobile Computing	1	Lamberton	Voronina	Physics
68999	Greve	2005	Spring	Calculus	1	Alumni	Wieland	Pol. Sci.
68999	Greve	2009	Fall	Tort Law	1	Lamberton	Boumier	Comp. Sci.
68999	Greve	2008	Spring	World History	1	Gates	Jaekel	Athletics
68999	Greve	2003	Spring	Bankruptcy	2	Taylor	Ullman	Accounting
68999	Greve	2006	Fall	Embedded Syste...	2	Alumni	Mingoz	Finance
68999	Greve	2010	Fall	The IBM 360 Arc...	2	Lamberton	Sarkar	Pol. Sci.
68999	Greve	2007	Spring	Marine Mammals	1	Gates	Gustafsson	Elec. Eng.
68999	Greve	2007	Fall	Journalism	1	Lamberton	Romero	Astronomy
68999	Greve	2009	Fall	Image Processing	1	Chandler	Romero	Astronomy
68999	Greve	2002	Fall	Drama	1	Polya	Liley	Languages
68999	Greve	2004	Spring	International Prac...	1	Gates	Bondi	Comp. Sci.
68999	Greve	2002	Fall	Marian History	1	Lamberton	Queiroz	Biology
68999	Greve	2006	Spring	Heat Transfer	1	Power	Mahmoud	Geology
68999	Greve	2008	Spring	Networking	1	Saucon	Mingoz	Finance
68999	Greve	2010	Spring	Music of the 50s	1	Lamberton	Mahmoud	Geology
68999	Greve	2007	Spring	Plasma Physics	1	Taylor	Gustafsson	Elec. Eng.
68999	Greve	2005	Spring	Bioinformatics	1	Power	Pimenta	Cybetetics
68999	Greve	2007	Fall	Fractal Geometry	2	Power	Shuming	Physics
*								

Bấm vào nút ‘ĐĂNG KÝ HỌC TẬP’, ta nhập vào MSSV chẳng hạn 68999 sẽ hiện ra giao diện Đăng ký học tập như hình dưới, tuy nhiên chức năng đăng ký học tập chưa hoàn thiện vì vấn đề thời gian hạn chế, việc kiểm tra nhiều điều kiện trước khi đăng ký tốn khá nhiều thời gian nếu làm tiếp nên em đang để dở, hiện tại đã xong phần ‘Tìm kiếm môn học’ như trong hình dưới và hiển thị đầy đủ các thông tin về môn học cũng như kết quả học tập của sinh viên.

ĐĂNG KÝ HỌC TẬP

Đã đăng ký học tập cho sinh viên
Greve
MSSV: **68999**

Tìm kiếm môn học
ID:
Tên môn:

Thông tin môn học
ID:
Tên môn:
Tin chỉ:
Thời gian học:
Phòng học:
Tòa nhà:
Số số tối đa:
Kỳ học:
Năm học:
Khoa viện:

DANH SÁCH MÔN HỌC

ID	MONHOC	TINCHI	PHONGHOC	TOANHA	KY	NAM	KHOAVIEN
400	Visual BASIC	4	348	Lambeau	Spring	2007	Psychology
400	Visual BASIC	4	425	Main	Fall	2003	Psychology
*							

KẾT QUẢ HỌC TẬP CỦA SINH VIÊN

MASV	ID	MONHOC	TINCHI	KY	NAM	DIEM	KHOAVIEN
68999	972	Greek Tragedy	4	Spring	2009	A	Psychology
68999	612	Mobile Computing	3	Fall	2007	B-	Physics
68999	581	Calculus	4	Spring	2005	B+	Pol. Sci.
68999	960	Tort Law	3	Fall	2009	B	Civil Eng.
68999	852	World History	4	Spring	2008	A-	Athletics
68999	408	Bankruptcy	3	Spring	2003	A+	Accounting

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*.