ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN

BÁO CÁO ĐỒ ÁN THỰC HÀNH
MÔ HÌNH QUẢN LÝ SINH VIÊN

GVHD: PHẠM NHẬT DUY
LỚP: IE103.N21.CNCL

MÔN: QUẢN LÝ THÔNG TIN

Sinh viên thực hiện:
Phạm Văn Hiếu - 21520857
Trần Tuyết Minh - 21521144
Nguyễn Văn Toàn - 21521549
Nguyễn Minh Lý - 21521108
Nguyễn Trọng Khang - 21520962

Giảng viên:
Nguyễn Gia Tuấn Anh
Phạm Nhật Duy

Thành phố Hồ Chí Minh, tháng 6 năm 2023

Mục lục

Phân công công việc

MSSV	Họ tên	Nội dung được phân công
21520857	Phạm Văn Hiếu	Phân tích, thiết kế, ERD, Phân quyền, Trigger, viết báo cáo
21521144	Trần Tuyết Minh	Stored Procedure
21521549	Nguyễn Văn Toàn	CrystalReport
21521108	Nguyễn Minh Lý	ERD, Bảng chi tiết dữ liệu, Trigger
21520962	Nguyễn Trọng Khang	Phân tích thiết kế, trigger, viết báo cáo

DANH MỤC BẢNG

DANH MỤC HÌNH VẼ

CHƯƠNG 1: PHÂN TÍCH THIẾT KẾ

1. Mô tả bài toán

- Bài toán quản lý sinh viên được xây dựng nhằm mục đích quản lý các thông tin của sinh viên như thông tin cá nhân, lớp học, khoa, môn học, điểm số, giảng viên giảng dạy, học phí,…

2. Các chức năng của hệ thống

Quản lý sinh viên sẽ bao gồm các chức năng:

- Quản lý sinh viên:

+Thêm: khi sinh viên nhập học tại trường.

+Xóa: khi sinh viên tốt nghiệp hoặc thôi học.

+Cập nhật: chức năng cho phép cập nhật lại thông tin của sinh viên trong quá trình học với các trường hợp như sai thông tin, thêm thông tin cần lưu ý, bổ sung thông tin,…

+Tìm kiếm: giúp việc truy vấn thông tin sinh viên được nhanh hơn trong các trường hợp cần tìm sinh viên thuộc diện hỗ trợ học phí, sinh viên xuất sắc, khen thưởng,…

- Quản lý giảng viên:

+Thêm: khi giảng viên vào làm tại trường.

+Xóa: khi giảng viên thôi việc.

+Cập nhật: chức năng cho phép cập nhật lại thông tin của giảng viên trong quá trình dạy với các trường hợp như sai thông tin, thêm thông tin cần lưu ý, bổ sung thông tin,…

+Tìm kiếm: giúp việc truy vấn thông tin giảng viên được nhanh hơn trong các trường hợp cần tìm giảng viên thuộc khoa, lớp, giảng viên có thành tích xuất sắc, khen thưởng thi đua,…

- Quản lý môn học:

+Thêm: khi môn học được đưa vào giảng dạy tại trường

+Xóa: khi môn học không còn được giảng dạy

+Cập nhật: chức năng cho phép cập nhật lại thông tin của môn học trong quá trình dạy với các trường hợp như sai thông tin, thêm thông tin cần lưu ý, bổ sung thông tin,…

+Tìm kiếm: giúp việc truy vấn thông tin môn học được nhanh hơn trong các trường hợp cần tìm môn học thuộc khoa, lớp và giảng viên đang giảng dạy,…

- Quản lý đăng ký môn học:

+Thêm: khi sinh viên đăng ký môn học

+Xóa: khi sinh viên không tiếp tục đăng ký

+Cập nhật: Giúp sinh viên thay đổi các lớp học với mã môn đã đăng ký, thêm môn mới,…

+Tìm kiếm: giúp việc truy vấn thông tin môn học nhanh hơn khi sinh viên đăng ký.

- Quản lý lịch thi:

+Thêm: khi học kỳ đến thời điểm thi.

+Cập nhật: Giúp sinh viên cập nhật lại trong trường hợp giảng viên, sinh viên không có phòng thi hoặc phòng thi không có thực,…

+Tìm kiếm: giúp sinh viên và giảng viên truy vấn thông tin phòng thi nhanh hơn,…

- Quản lý điểm của sinh viên:

+Thêm: sau khi sinh viên kết thúc học phần và thi, kết quả được thêm vào.

+Cập nhật: Giúp giảng viên cập nhật lại điểm cho sinh viên trong trường hợp sai xót hoặc nhầm điểm,…

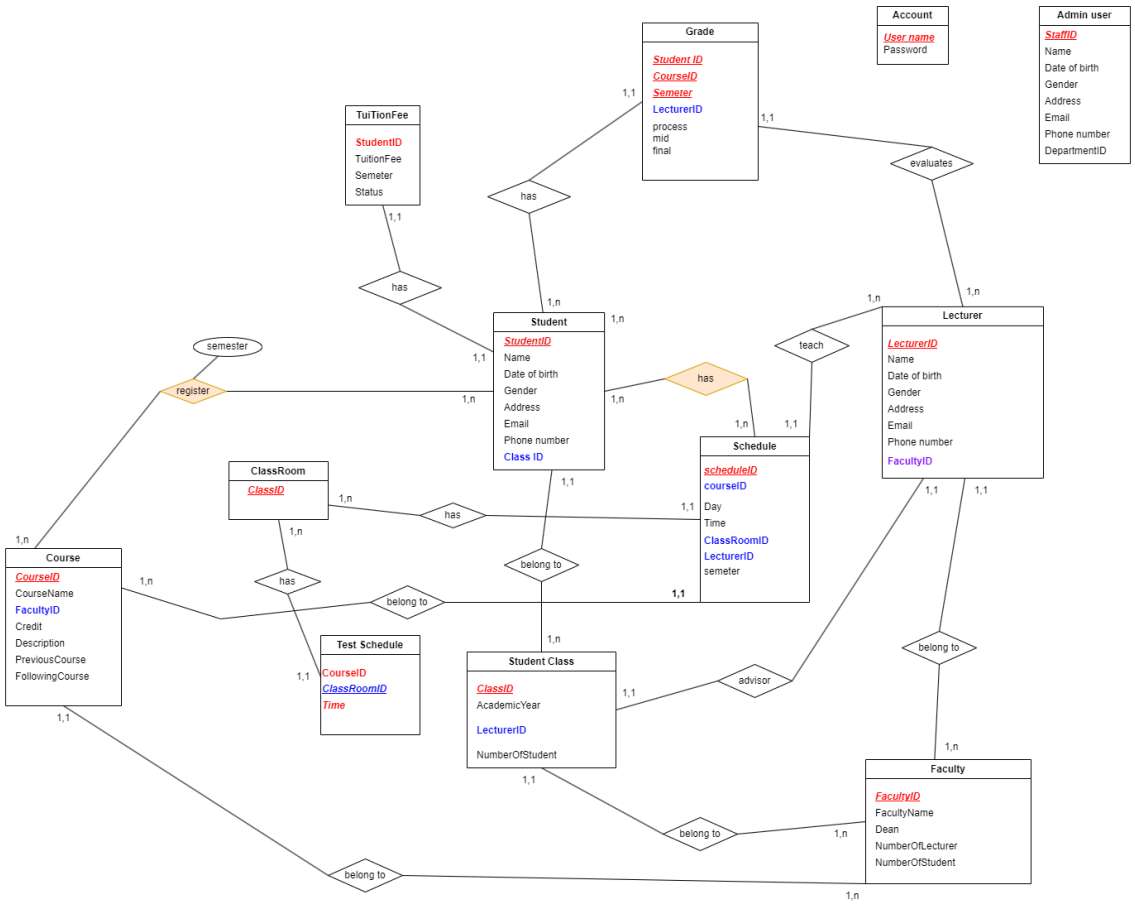
+Tìm kiếm: giúp việc truy vấn sinh viên có thành tích tốt và xét khen thưởng nhanh hơn, chính xác hơn,…

3. Mô hình ERD

- Màu đỏ: khóa chính

- Màu xanh: khóa ngoại

- Màu ghi: tạo quan hệ mới



4. Mô hình dữ liệu quan hệ

Student (<u>**studentID**</u>**,** name, dateOfBirth, gender, address, email, phone number**,** **classID**)

Lecturer (<u>**lecturerID**</u>, name, dateOfBirht, gender, address, email, phone number, **facultyID**)

Admin_user (<u>**staffID**</u>, name, dateOfBirht, gender, address, email, phone number, departmentID)

Grade (<u>**studentID, courseID, semester**</u>**,** **lecturerID**, process, mid, final)

Course (<u>**courseID**</u>**,** courseName**,** **facultyID**,** credit**,** description**,** previousCourse**,** followingCourse)

Course_register (<u>**studentID, courseID**</u>, semester)

Faculty (<u>**facultyID**</u>**,** facultyName**,** dean**,** numberOfLecturer, numberOfStudent)

Test_Schedule (scheduleid, classroomID, Time, CourseID, lecturerID)

Schedule (scheduleID, courseID, day, time, classroomID, lecturerID, semester)

Student_schedule (studentID, scheduleID)
Classroom (classroomID)

Student_class (classID, academicYear, numberOfStudent, lecturerID,facultyID)

Tuition_Fee (studentID, TuitionFee, status, sumcredit)
Account (username, password)

5. Bảng mô tả thành phần dữ liệu

No	Relation	Attributes	Data type	Interpretation
1	**Student**	**-studentID** -name -dateOfBirth -gender -address -email -phone number **-classID**	Char(8) Nvarchar(30) Datetime Char(1) Nvarchar(100) Nvarchar(50) Char(10) Char(8)	PRK NOT NULL NOT NULL NOT NULL NOT NULL NOT NULL NOT NULL FK1, ref **student_class**
2	**Lecturer**	**-lecturerID** -name -dateOfBirht -gender -address -email -phone number **-facultyID**	Char(8) Nvarchar(30) Datetime Char(1) Nvarchar(100) Nvarchar(50) Char(10) Char(8)	PRK NOT NULL NOT NULL NOT NULL NOT NULL NOT NULL NOT NULL FK1, ref **Faculty**
3	**Admin_user**	**-staffID** -name -dateOfBirht -gender -address -email -phone number -departmentID	Char(8) Nvarchar(30) Datetime Char(1) Nvarchar(100) Nvarchar(50) Char(10) Char(8)	PRK NOT NULL NOT NULL NOT NULL NOT NULL NOT NULL NOT NULL NOT NULL
4	**Grade**	**-studentID** **-courseID** **-semeter** **-lecturerID** -process -mid -final	Char(8) Char(8) Char(16) Char(8) Float Float Float	PRK, ref **Student** PRK, ref **Course** PRK FK, ref **lecturer** NULL NULL NULL
5	**Course**	**-courseID** -courseName **-facultyID**	Char(8) Nvarchar(50) Char(8)	PRK NOT NULL FK1, ref **Faculty**

		-credit -description -previousCourse -followingCourse	Int Text Char(8) Char(8)	NOT NULL NOT NULL NOT NULL NOT NULL
6	**Course_register**	**-studentID** **-courseID** -semester	Char(8) Char(8) Char(15)	PRK, ref **student** PRK, ref **course**
7	**Faculty**	**-facultyID** -facultyName -dean -numberOfLecturer -numberOfStudent	Char(8) Nvarchar(100) Nvarchar(30) int int	PRK NOT NULL NOT NULL NOT NULL NOT NULL
8	**Test_Schedule**	**-scheduleid** **-classroomID** **-Time** **-CourseID** **-lecturerID**	Varchar(8) char(8) datetime char(8) char(8)	PRK, ref **Schedule** PRK, ref **Classroom** PRK FK1, ref **Course** FK2, ref **Lecturer**
9	**Schedule**	**-scheduleID** **-courseID** -day -time **-classroomID** **-lecturerID** -semeter	Char(8) Char(8) Enum(…) Time char(8) char(8) char(15)	PRK FK1, ref **Course** NOT NULL NOT NULL FK2, ref **classroom** FK3, ref **lecturer** NOT NULL
10	**Student_schedule**	**-studentID** **-scheduleID**	Char(8) Char(8)	PRK, ref **student** PRK, ref **schedule**
11	**Classroom**	**-classroomID**	Char(8)	PRK
12	**Student_class**	**-classID** -academicYear -numberOfStudent **-lecturerID** **-facultyID**	Nvarchar(8) year int char(8) char(8)	PRK NOT NULL NOT NULL FK1, ref **Lecturer** FK2, ref **Faculty**
13	**Tuition_Fee**	**-studentID** **-TuitionFee** -status -sumcredit	Char(8) Numeric Varchar(20) int	PRK, ref **student** PRK NOT NULL NOT NULL
14	**Account**	**-username** -password	Varchar(50) Varchar(256)	PRK NOT NULL, sha256

CHƯƠNG 2: XÂY DỰNG VÀ QUẢN LÝ

1. TRIGGER

1. **Trigger** tự động cập nhập numberOfLecturer trong bảng faculty khi insert new lecturer

   -**Trigger** tự động thực thi khi user admin thêm 1 record lecturer mới

```
CREATE TRIGGER update_numberOfLecturer
ON Lecturer
AFTER INSERT
AS
BEGIN
	UPDATE Faculty
	SET numberOfLecturer = (
	SELECT COUNT(lecturerID)
	FROM Lecturer
	WHERE facultyID = inserted.facultyID
	GROUP BY facultyID
	)
	FROM Faculty
	INNER JOIN inserted ON Faculty.facultyID = inserted.facultyID;
END;
```

2. **Trigger** tự động cập nhập numberOfStudent trong bảng STUDENTCLASS khi insert new student

   -**Trigger** tự động thực thi khi user admin thêm 1 record student mới

```
CREATE TRIGGER update_numberOfStudent
ON Student
AFTER INSERT
AS
BEGIN
	UPDATE StudentClass
	SET numberOfStudent = (
	SELECT COUNT(studentID)
	FROM Student
	WHERE classID = inserted.classID
	GROUP BY classID
	)
	FROM StudentClass
	INNER JOIN inserted ON StudentClass.classID = inserted.classID;
	END;
```

3. **Trigger** tự động cập nhập numberOfStudent trong bảng FACULTY khi insert new student

   - **Trigger** tự động thực thi khi user admin thêm 1 record student mới

```
CREATE TRIGGER update_numberOfStudent_1
ON studentclass
AFTER UPDATE
AS
BEGIN
	UPDATE Faculty
	SET numberOfStudent = (
	SELECT SUM(s.numberOfStudent)
	FROM studentclass AS s
	INNER JOIN faculty_class AS fc ON s.classid = fc.classid
	GROUP BY fc.facultyid
	HAVING facultyID=Faculty.facultyID
	)
	END;
```

4. **Trigger** tự động cập nhập điểm trung bình (avg) theo công thức giả định:

   0.2*process+0.3*mid+0.5* final

   - **Trigger** được tự động thực thi khi user Lecturer nhập điểm

```
CREATE TRIGGER insert_avg
ON GRADE
AFTER INSERT, UPDATE
AS
BEGIN
	UPDATE g
	SET g.AVG = (i.process * 0.2) + (i.mid * 0.3) + (i.final * 0.5)
	FROM GRADE g
	INNER JOIN inserted i ON g.courseID=i.courseID and g.studentID=i.studentID and
g.semester=i.semester
END;
```

5. **Trigger** ràng buộc điểm số trong 0-10, for insert và update

   - **Trigger** được kiểm tra trước khi user Lecturer nhập điểm, nếu điểm hợp lệ thì mới insert, không thì báo lỗi

```
CREATE TRIGGER check_grade_range
ON Grade
INSTEAD OF INSERT, UPDATE
AS
BEGIN
    DECLARE @InvalidGradeMessage NVARCHAR(100);
    IF EXISTS(SELECT 1 FROM inserted WHERE [process] < 0 OR [process] > 10)
    BEGIN
        SET @InvalidGradeMessage = 'Invalid process grade';
        RAISERROR (@InvalidGradeMessage, 16, 1);
        ROLLBACK TRANSACTION;
        RETURN;
    END;
    IF EXISTS(SELECT 1 FROM inserted WHERE mid < 0 OR mid > 10)
    BEGIN
        SET @InvalidGradeMessage = 'Invalid mid grade';
        RAISERROR (@InvalidGradeMessage, 16, 1);
        ROLLBACK TRANSACTION;
```

```
        RETURN;
    END;
    IF EXISTS(SELECT 1 FROM inserted WHERE final < 0 OR final > 10)
    BEGIN
        SET @InvalidGradeMessage = 'Invalid final grade';
        RAISERROR (@InvalidGradeMessage, 16, 1);
        ROLLBACK TRANSACTION;
        RETURN;
    END;
```

6. **Trigger** tự động cập nhập bảng tuitionfee theo bảng register_course. giả sử 1 tín chỉ là 400k

   - **Trigger** được thực thi mỗi khi user Student đăng kí học phần (thêm môn hoặc xóa môn)

   - for insert

```
CREATE TRIGGER insert_course_register
ON course_register
AFTER INSERT
AS
BEGIN
    IF EXISTS (SELECT * FROM TUITIONFEE WHERE STUDENTID = (SELECT STUDENTID FROM
inserted))
    BEGIN
        UPDATE TUITIONFEE
        SET TUITIONFEE = TUITIONFEE + (400000 * (SELECT CREDIT FROM COURSE WHERE
COURSEID = (SELECT COURSEID FROM inserted))),
            sumcredit = sumcredit + (SELECT CREDIT FROM COURSE WHERE COURSEID =
(SELECT COURSEID FROM inserted))
        WHERE STUDENTID = (SELECT STUDENTID FROM inserted);
    END
    ELSE
    BEGIN
        INSERT INTO TuitionFee (TuitionFee, Status, StudentID, sumcredit)
        SELECT (400000 * (SELECT CREDIT FROM COURSE WHERE COURSEID = (SELECT COURSEID
FROM inserted))),
            'unpaid',
            (SELECT STUDENTID FROM inserted),
            (SELECT CREDIT FROM COURSE WHERE COURSEID = (SELECT COURSEID FROM
inserted))
        FROM inserted
        INNER JOIN Course ON inserted.CourseID = Course.CourseID;
    END
       END;
```

- for delete

```
CREATE TRIGGER delete_course_register
ON course_register
AFTER DELETE
AS
BEGIN
    UPDATE TUITIONFEE
    SET TUITIONFEE = TUITIONFEE - (400000 * c.CREDIT),
        sumcredit = sumcredit - c.CREDIT
    FROM TUITIONFEE
    JOIN deleted d ON TUITIONFEE.STUDENTID = d.STUDENTID
    JOIN COURSE c ON d.COURSEID = c.COURSEID;
END;
```

7. **Trigger** ràng buộc môn học trước. Nếu insert vào bảng course_register (đăng kí học phần). Sinh viên cần học qua môn học trước, có môn học đó trong bảng grade

- Trigger được thực thi mỗi khi user student đăng ký 1 môn học mới, nếu hợp lệ thì mới insert thành công, không thì báo lỗi

```
CREATE TRIGGER check_previous_course
ON course_register
AFTER INSERT
AS
BEGIN
    DECLARE @previous_course CHAR(8);
    SET @previous_course = (
        SELECT DISTINCT previouscourse
        FROM course
        WHERE courseid = (SELECT courseid FROM inserted)
    );
    IF @previous_course IS NOT NULL AND NOT EXISTS (
        SELECT courseid
        FROM grade
        WHERE studentid = (SELECT studentid FROM inserted)
              AND courseID=@previous_course
    )
    BEGIN
        RAISERROR ('Lỗi môn học trước', 16, 1);
        ROLLBACK TRANSACTION;
        RETURN;
    END
       END;
```

2. STORE PROCEDURE

1. **Procedure** tính gpa

- Procedure này được thực thi để trả về điểm GPA (trung bình tích lũy) của 1 sinh viên khi nhập mã số sinh viên (studentID), có thể sử dụng cho user phòng đào tạo, user giảng viên

```
CREATE PROCEDURE CalculateGPA
    @studentID VARCHAR(10)
AS
BEGIN
    DECLARE @TongDiem DECIMAL(10, 2)
    DECLARE @TongTinChi INT

    SELECT @TongDiem = SUM(f.avg * f.credit),
           @TongTinChi = SUM(f.credit)
    FROM (
        SELECT t.courseID, t.avg, c.credit
        FROM (
            SELECT courseID, MAX(avg) AS avg
            FROM grade
            WHERE studentID = @studentID
            GROUP BY courseID
        ) AS t
```

```
        INNER JOIN course AS c ON t.courseID = c.courseID
    ) AS f

    DECLARE @GPA DECIMAL(10, 2)
    SET @GPA = @TongDiem / @TongTinChi

    SELECT @GPA AS GPA
END
```

2. **Procedure** tính số môn học đã pass và fail của 1 sinh viên

   - **Procedure** này được thực thi để trả về số môn pass và fail của 1 sinh viên khi nhập vào mã số sinh viên (studentID), có thể sử dụng cho user phòng đào tạo, user student

```
CREATE PROC FAILED_PASSED_SUBJECT
    @studentID CHAR(8),
    @FAILED INT OUTPUT,
    @PASSED INT OUTPUT
AS
BEGIN
    IF @studentID IN (SELECT studentID
    FROM Grade)
 BEGIN
    SELECT @FAILED = (SELECT COUNT(*)
        FROM (
        SELECT t.courseID, t.avg, c.credit
        FROM (
            SELECT courseID, MAX(avg) AS avg
            FROM grade
            WHERE studentID = @studentID
            GROUP BY courseID
        ) AS t
        INNER JOIN course AS c ON t.courseID = c.courseID
    ) AS f WHERE f.avg < 5)
    SELECT @PASSED = (SELECT COUNT(*)
        FROM (
        SELECT t.courseID, t.avg, c.credit
        FROM (
            SELECT courseID, MAX(avg) AS avg
            FROM grade
            WHERE studentID = @studentID
            GROUP BY courseID
        ) AS t
        INNER JOIN course AS c ON t.courseID = c.courseID
    ) AS f WHERE f.avg >= 5)
        PRINT 'STUDENT WITH ID: ' + @studentID + ' NUMBER OF FAILD SUBJECT: ' +
CAST(@FAILED AS CHAR) + ' NUMBER OF PASSED SEBJECT ' + CAST(@PASSED AS CHAR)
    END
 ELSE
 BEGIN
        PRINT 'CANT FIND STUDENT WITH ID: ' + @StudentID
    END
            END
```

3. **Procedure** tính số tín chỉ tích lũy của 1 sinh viên

- Procedure này được thực thi để trả về số tín chỉ tích lũy của 1 sinh viên khi nhập vào mã số sinh viên (studentID), có thể sử dụng cho user phòng đào tạo, user student

```
GO
CREATE PROC Sum_credit
    @studentID CHAR(8),
    @SUMCREDIT INT OUTPUT
AS
BEGIN
    IF @studentID IN (SELECT studentID
    FROM Grade)
 BEGIN
    SELECT @SUMCREDIT = (select SUM(credit)
    from (select courseid, max(avg) as avg
              from grade
              where studentid='st000002'
              group by courseid) as t
              inner join course on t.courseID=course.courseID )
    PRINT 'STUDENT WITH ID: ' + @studentID + ' Number of credit: ' + CAST(@SUMCREDIT
AS CHAR)
    END
    ELSE
    BEGIN
    PRINT 'CANT STUDENT WITH ID: ' + @studentID
    END
          END
```

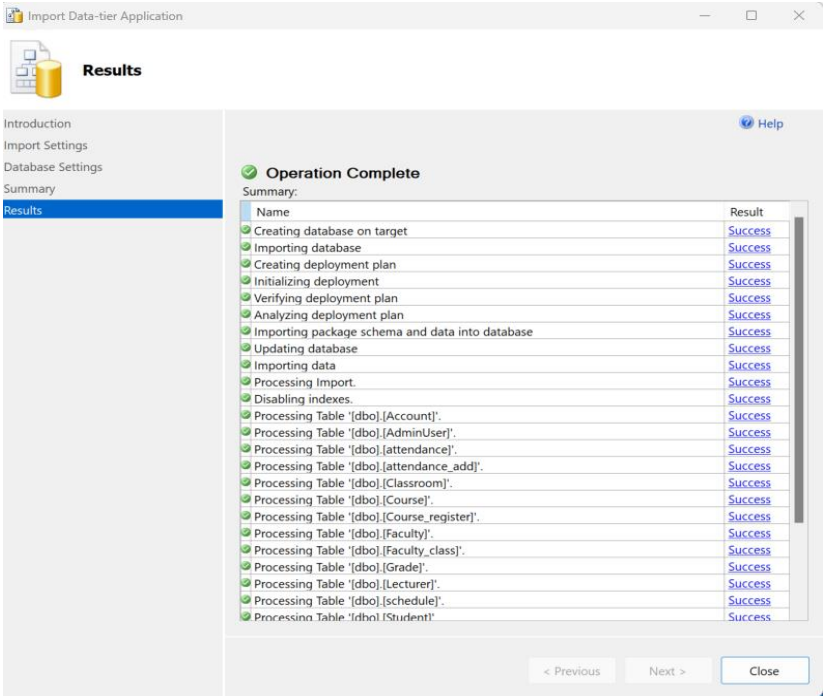
3. Phân quyền USER

```
-- USER for STUDENT
CREATE LOGIN STUDENT WITH PASSWORD = 'password';
CREATE USER STUDENT1 FOR LOGIN STUDENT;
--cấp quyền
GRANT UPDATE ON dbo.student TO STUDENT1;
GRANT SELECT ON SCHEMA :: [dbo] TO STUDENT1;
GRANT INSERT,DELETE ON dbo.course_register to STUDENT1;

-- USER for LECTURER
CREATE LOGIN LECTURER WITH PASSWORD = 'password';
CREATE USER LECTURER1 FOR LOGIN LECTURER;
--cấp quyền
GRANT SELECT ON SCHEMA :: [dbo] TO LECTURER1;
GRANT UPDATE ON dbo.lecturer TO LECTURER1;
GRANT INSERT ON dbo.GRADE TO LECTURER1;
GRANT UPDATE ON dbo.GRADE TO LECTURER1;

-- USER for thầy cô phòng đào tạo
CREATE LOGIN ADMINUSER WITH PASSWORD = 'password';
CREATE USER ADMINUSER1 FOR LOGIN ADMINUSER;
--cấp quyền
GRANT SELECT ON SCHEMA :: [dbo] TO ADMINUSER1;
GRANT UPDATE,INSERT,DELETE ON dbo.schedule to ADMINUSER1;
GRANT UPDATE,INSERT,DELETE ON dbo.student_schedule to ADMINUSER1;
GRANT UPDATE,INSERT,DELETE ON dbo.testschedule to ADMINUSER1;
GRANT UPDATE ON dbo.account to ADMINUSER1;
```

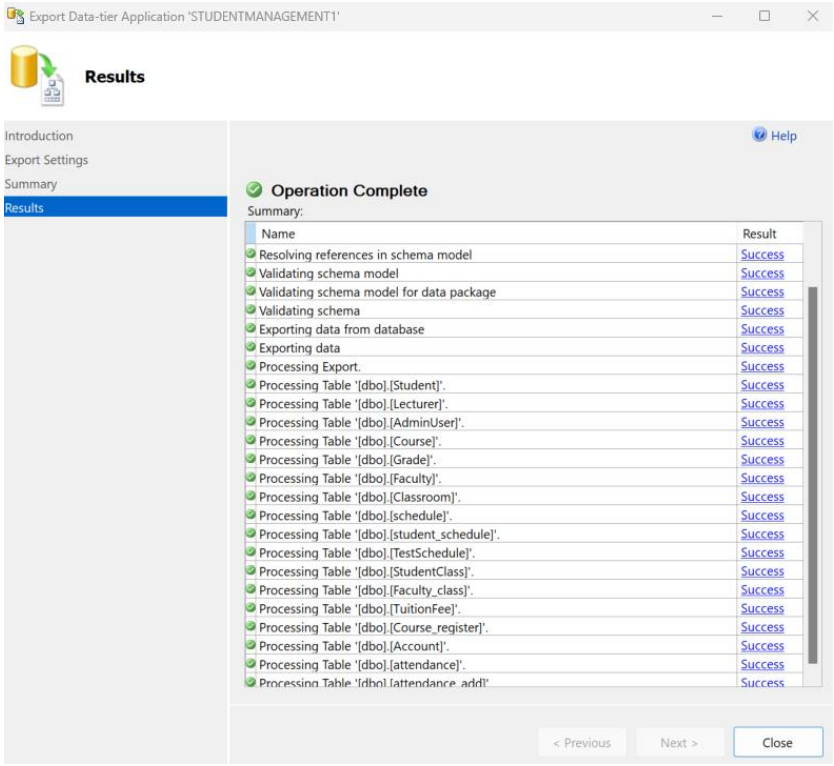
```
-- USER for ADMIN, đây là admin quản trị viên web.
CREATE LOGIN ADMIN WITH PASSWORD = 'password';
CREATE USER ADMIN1 FOR LOGIN ADMIN;
--cấp quyền
GRANT UPDATE, INSERT, DELETE ON dbo.student to ADMIN1;
GRANT UPDATE, INSERT, DELETE ON dbo.lecturer to ADMIN1;
GRANT UPDATE, INSERT, DELETE ON dbo.adminuser to ADMIN1;
GRANT UPDATE, INSERT, DELETE ON dbo.account to ADMIN1;
GRANT SELECT ON SCHEMA :: [dbo] TO ADMIN1;
GRANT INSERT, UPDATE,DELETE ON dbo.course to ADMIN1;
```

4. Import, Export, Backup, Restore

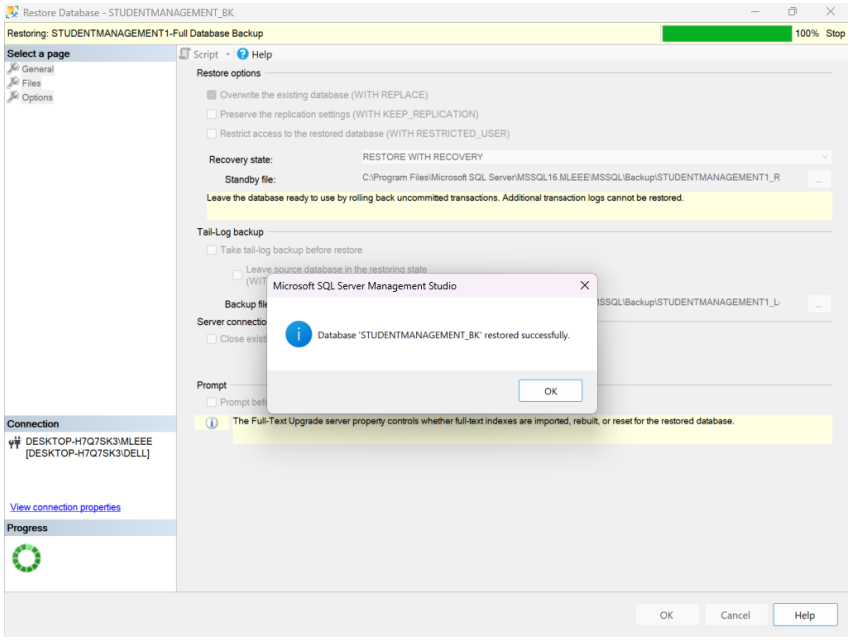
- Import



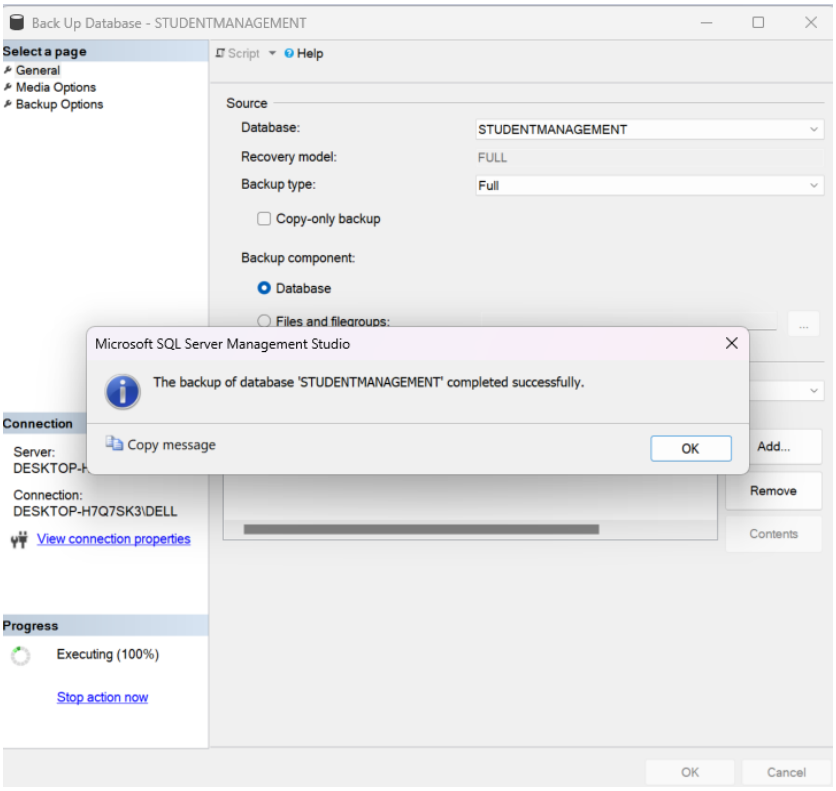
-Export



-Restore

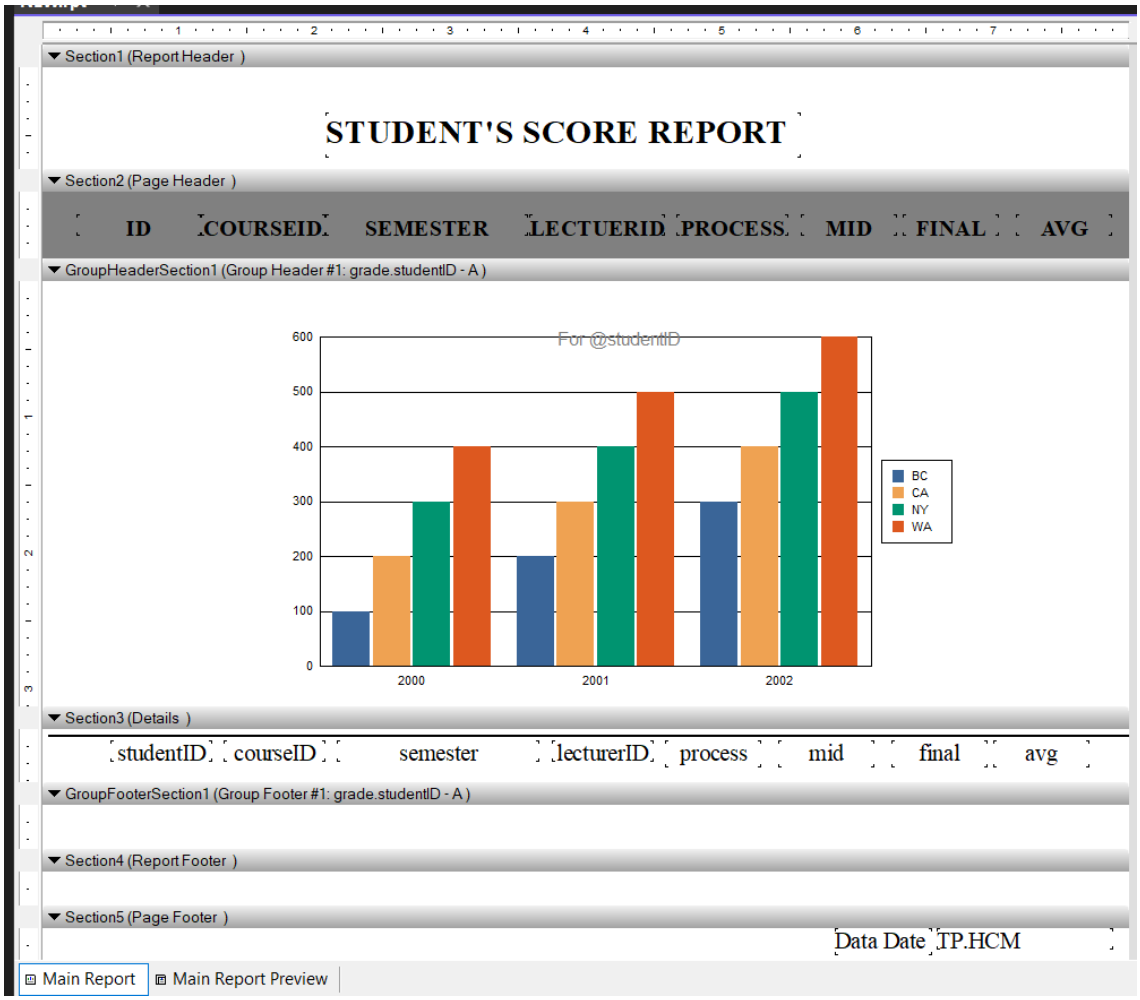


-Back up

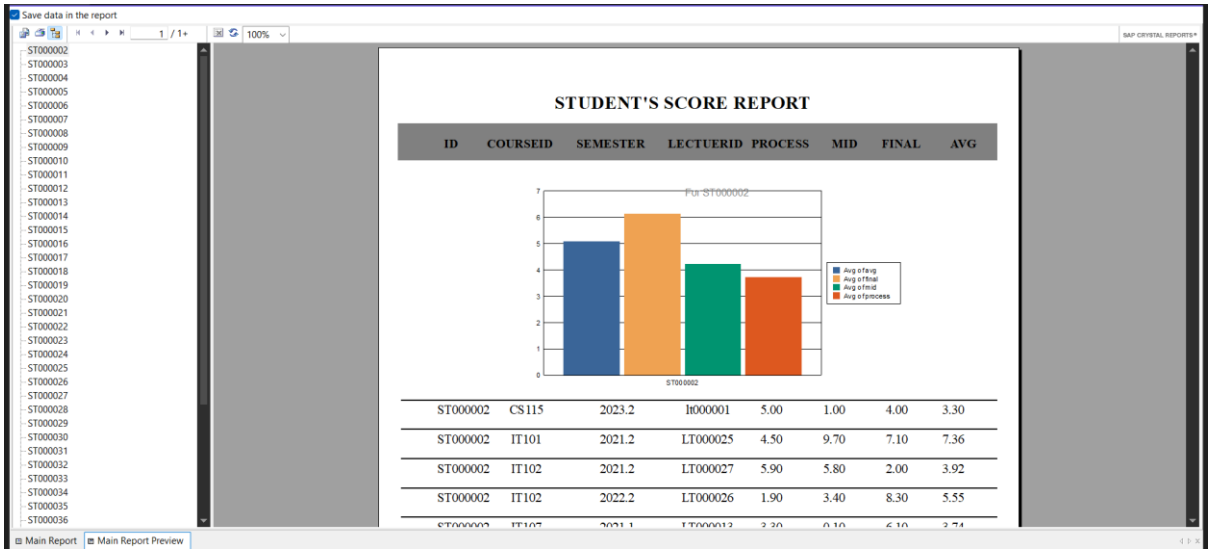


5. Crystal Report

- Màn hình design



-Màn hình preview



6. Link google drive code demo

https://drive.google.com/drive/folders/19fSpb8piuS5GzxzcGGwmfHb3D6AJvuP6?usp=sharing