# Week 12 Homework Solution

From Chapter 9 -Datawarehouse:

1. A possible field list for the new table could be: StudentNo, LastName, FirstName, MI, Address, Telephone, Status, Dept, Hours, Insurance.
2. a.
3. Transient (06/21)

#### Key Name Major

001 Amy Music

002 Tom Business

003 Sue Art

004 Joe Business

006 Jim Phys Ed

Transient (06/22)

**Key Name Major**

001 Amy Music

002 Tom Business

003 Sue History

004 Joe Business

006 Jim Bskt Weav

b. It should be noted that the actual PK of the rows of this table is a combination of the original Key and the Date fields.

Periodic (06/21)

**Key Date Name Major Action**

001 06/20 Amy Music C

002 06/20 Tom Business C

003 06/20 Sue Art C

004 06/20 Joe Math C

004 06/21 Joe History U

005 06/20 Ann Engineering C

005 06/21 Ann Engineering D

006 06/21 Jim Phys Ed C

Periodic (06/22)

**Key Date Name Major Action**

001 06/20 Amy Music C

002 06/20 Tom Business C

003 06/20 Sue Art C

003 06/22 Sue History U

004 06/20 Joe Math C

004 06/21 Joe History U

005 06/20 Ann Engineering C

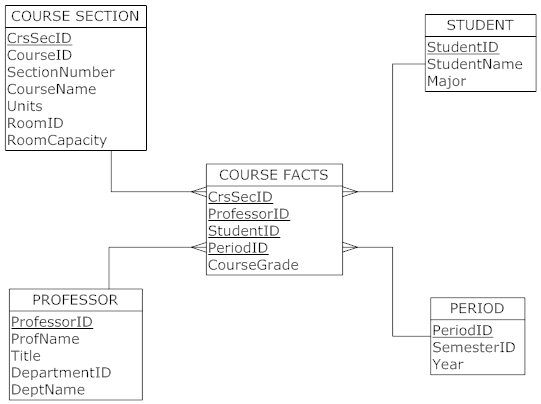
005 06/21 Ann Engineering D

006 06/21 Jim Phys Ed C

006 06/22 Jim Bskt Weav U

1. Millennium College

a. Star Schema



1. 500 course sections x 40 students per section x 30 periods (i.e., 3 semesters per year) = 600,000 rows (assuming 1 professor per course section)

c. 600,000 rows \* 5 fields per row \* 5 bytes per field = 15,000,000 bytes

d. There are several possible options that can be considered if a star schema is not mandated. The following suggestions are all aimed at creating a more normalized data model:

* + Professor does not have to relate directly to the fact table. Professor could snowflake off of the Section (who are the professors teaching a particular section?). Then, there would be no Professor PK in the fact table.
  + Course information could snowflake off of Section (put Course ID, Course Name, and Units in a separate table related to Section). This would normalize Course and Section data.
  + Similarly, Department information (Department ID and Department Name) could be snowflaked off of Professor.

e. Dimensional segmentation is recommended. The student, section, and professor dimensions would be divided into two segments each, one for constant attributes and one for attributes that change. For example, the room might change frequently for the section. This approach would eliminate a large number of rows in the dimension tables if the changing attributes were to change each semester

.

* Using the sample database from prior lecture, write the following persistent stored modules, and write commands to execute each:

1. Create a stored procedure that takes Country as an input parameter, and outputs a list of employees in that country.
   1. Procedure Name: GetEmployees @Country.
   2. Columns: First Name, Last Name, City, Country

create procedure GetEmployee @Country varchar(25)

AS

BEGIN

SELECT e.emp\_fname, e.emp\_lname, location AS City,

CASE when d.location ='London' THEN 'England' ELSE 'USA' END AS Country

FROM employee e JOIN department d ON e.dept\_no = d.dept\_no

WHERE CASE when d.location ='London' THEN 'England' ELSE 'USA' END =@Country

END

EXEC GetEmployee 'England'

1. Write a stored procedure AssignWork which takes emp\_no, project\_no, job as input parameters. It must add a record to the works\_on table with enterd\_date = system date, and returns the employee full name as an output parameter

CREATE PROCEDURE AssignWork @emp\_no int, @project\_no varchar(5),

@job varchar(50), @full\_name varchar(250) OUTPUT

AS

BEGIN

INSERT INTO works\_on VALUES (@emp\_no,@project\_no,@job, GETDATE() );

SELECT @full\_name = e.emp\_fname + ' ' + e.emp\_lname

from employee e

WHERE emp\_no =@emp\_no

END;

declare @name varchar(250)

EXEC AssignWork 15001, 'P4', 'Manager' ,@name output

select @name

1. Create a UDF that takes @City as an input parameter and returns country.

UDF Name: GetCountry (@city)

CREATE PROCEDURE AssignWork @emp\_no int, @project\_no varchar(5),

@job varchar(50), @full\_name varchar(250) OUTPUT

AS

BEGIN

INSERT INTO works\_on VALUES (@emp\_no,@project\_no,@job, GETDATE() );

SELECT @full\_name = e.emp\_fname + ' ' + e.emp\_lname

from employee e

WHERE emp\_no =@emp\_no

END;

declare @name varchar(250)

EXEC AssignWork 15001, 'P4', 'Manager' ,@name output

select @name