**COMP / IT 420 Spring 2021 Take Home Final Examination Compliment**

**Open: May 19th, 2021, 8:00am**

**Closed: May 20th, 2021, 8:00am (extension to May 21st if DASS)**

**Introduction**

This sheet is intended to be used as reference for the final examination. Due to the limitations of Canvas formatting, I’m providing copies of some of the questions here.

**Part 2: SQL**

1. Please construct a SQL query that returns the name and total number of credits each instructor taught in the Fall 2020 and Spring 2021 semesters if they taught more than four credits. Sort alphabetically by instructor name. Note that each section counts as a full course credit, so if someone teaches two sections of COMP420 and COMP420 is four course\_credits, then that is eight total credits.

SELECT instructor\_name, ***(1)***\_\_\_\_\_(***(2)***\_\_\_\_\_\_\_) FROM course

JOIN teaches using(***(3)***\_\_\_\_\_\_)

JOIN ***(4)***\_\_\_\_\_\_ using(sec\_id)

JOIN instructor using(***(5)***\_\_\_\_\_\_\_)

WHERE (***(6)***\_\_\_\_\_\_\_ = ***(7)***\_\_\_\_\_ AND sec\_year = 2020) ***(8)***\_\_\_\_\_ (sec\_semester = "Spring" ***(9)***\_\_\_\_\_\_ sec\_year = 2021)

***(10)***\_\_\_\_\_\_ instructor\_id

***(11)***\_\_\_\_\_\_ sum(course\_credits) > ***(12)***\_\_\_\_\_\_

ORDER BY ***(13)***\_\_\_\_\_\_\_\_\_ ***(14)***\_\_\_\_\_;

1. (8pts) The function validate\_credits takes a course and a number of credits and checks to see whether the credit assignment aligns with the number of hours each course section is offered per week. For simplicity, each hour of class time is equal to a valid credit hour.

Please fill in the blanks in the validate\_credits function below to ensure that courses have correct credits assigned.

TIMESTAMPDIFF takes an interval ("MINUTE", "SECOND", etc.), a start time and an end time and returns the number of units for that interval as an INT.

If a FLOAT value is equal to an INT value, e.g. 1.0 = 1, then MySQL will evaluate to true even though the types are different.

DELIMITER ***(1)*** \_\_\_

CREATE FUNCTION validate\_credits (***(2)***\_\_\_\_\_\_\_ VARCHAR(7), credits ***(3)***\_\_\_\_\_)

RETURNS bool

BEGIN

-- declare variables to hold section total minutes per week

DECLARE sec\_tot\_week\_minutes INT;

-- populate variable with section minutes per week

SELECT ***(4)***\_\_\_\_\_(sec\_time\_length) INTO sec\_tot\_week\_minutes

FROM

(

SELECT TIMESTAMPDIFF(MINUTE, time\_slot\_start, time\_slot\_end) AS ***(5)***\_\_\_\_\_\_\_\_\_

FROM section

JOIN ***(6)***\_\_\_\_\_\_\_\_ USING(time\_slot\_id)

WHERE ***(7)***\_\_\_\_\_\_\_\_ = course\_to\_check

);

-- compare the input credit amount with the credit hours

-- computed from the section lengths in minutes

RETURN (credits = sec\_tot\_week\_minutes / ***(8)***\_\_\_\_\_\_\_)

END //

**Part 3: Transaction Management**

Refer to the University Department Manager ERD for questions 1 and 2 below.

1. The transaction below enables a student (id 1234) enrollment in a specific course, "COMP520", section "001". In order to enroll the following conditions must be met:
   1. A student must be affiliated with the department of the course.
   2. A student's total credits cannot exceed 180.
   3. The course enrollment cannot exceed the capacity for the classroom.
   4. The total credits for the current semester (Spring 2020) for a student cannot exceed 18.
   5. A student must have taken the required prerequisite for COMP520.

(Note: If a condition has failed, a ROLLBACK call is expected.)

-- Declare temporary variables

DECLARE stu\_tot\_credits INT;

DECLARE stu\_current\_credits INT;

DECLARE stu\_dept INT;

DECLARE comp\_520\_creds INT;

DECLARE comp\_520\_dept INT;

DECLARE class\_capacity INT;

DECLARE current\_class\_capacity INT;

DECLARE ***(1)*** \_\_\_\_\_\_ VARCHAR(7);

-- Initiate the transaction

***(2)*** \_\_\_\_\_\_\_\_\_\_\_\_;

-- Get the total number of current credits for the student

-- in the current semester

SELECT sum(***(3)***\_\_\_\_\_\_\_\_) INTO stu\_current\_credits

FROM student JOIN enroll USING(student\_id)

JOIN ***(4)*** \_\_\_\_\_\_\_\_\_\_\_\_\_

JOIN section USING(sec\_id)

WHERE ***(5)*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

-- Check that current credits do not exceed condition D

IF ***(6)*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ > 18 THEN

ROLLBACK;

END IF;

-- Get the total credits for student

SELECT student\_tot\_creds INTO ***(7)***\_\_\_\_\_\_\_\_\_\_ FROM student WHERE student\_id = '1234';

-- Get the total course\_credits for COMP520

SELECT course\_credits INTO comp\_520\_creds FROM course WHERE course\_id = 'COMP520';

-- Make sure that total credits do not exceed condition B

IF ***(8)*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ + comp\_520\_creds > 180 THEN

ROLLBACK;

END IF;

-- Get the dept\_ids to compare for condition A

SELECT dept\_id INTO stu\_dept FROM student WHERE student\_id = '1234';

SELECT dept\_id INTO comp\_520\_dept FROM course WHERE course\_id = 'COMP520';

-- Compare dept ids

IF stu\_dept != comp\_520\_dept THEN

***(9)***\_\_\_\_\_\_\_\_\_\_;

END IF;

-- Get class room capacity for section

SELECT class\_capacity FROM classroom

JOIN section ON ***(10)*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ AND classroom.class\_building = section.sec\_building WHERE ***(11)***\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

-- Get current section number of enrolled

SELECT count(\*) INTO ***(12)***\_\_\_\_\_\_\_\_\_\_\_ FROM enroll WHERE course\_id = 'COMP520' AND sec\_id = '001';

-- Compare capacities to check condition C  
IF current\_class\_capacity + 1 > ***(13)***\_\_\_\_\_\_\_\_\_\_\_ THEN

ROLLBACK;

END IF;

-- Get course prerequisite

SELECT course\_prereq\_id INTO prereq FROM course WHERE course\_id = 'COMP520';

-- Check for condition E using a sub-query

IF prereq IN ( SELECT course\_id

FROM enroll JOIN student USING(student\_id)

WHERE student\_id = ***(14)***\_\_\_\_\_\_\_\_\_\_\_) THEN

INSERT INTO enroll VALUES ('1234', ***(15)***\_\_\_\_\_\_, '001');

***(16)*** \_\_\_\_\_\_

ROLLBACK;

END IF;

***(17)*** \_\_\_\_\_\_\_\_;

|  |  |
| --- | --- |
| TRANSACTION 1 | TRANSACTION 2 |
| START TRANSACTION | START TRANSACTION |
| validate\_credits("COMP420", 4) |  |
| UPDATE course SET course\_credits = 4 WHERE course\_id = "COMP420"; | DECLARE comp\_420\_cred; |
|  | SELECT course\_credits INTO comp\_420\_cred FROM course WHERE course\_id = “COMP420”; |
|  | UPDATE student SET student\_tot\_creds += comp\_420\_cred WHERE student\_id = 1234; |
| ROLLBACK |  |
|  | COMMIT |

**Part 5: NoSQL (12pts)**

1. (5pts) Fill in the following aggregate pipeline query to find the total number of employees working at companies that start with the letter “F”.

db.companies.aggregate([

{$***(1)***\_\_\_\_: {***(2)***\_\_\_\_: {$***(3)***\_\_\_\_\_: /^F/}}},

{$group: {

\_id: “total\_employees”,

total:{$***(4)***\_\_\_\_: “***(5)***\_\_\_\_\_\_\_\_\_”}}}

])

1. (4pts) Fill in the following MapReduce to find the same information (total number of competitors for each company).

db.companies.mapReduce(***(1)***\_\_\_\_\_\_\_\_, reduceComp,

{output: “total\_comp\_per\_company”})

function mapComp(){

var i;

for( i = 0; i < this.***(2)***\_\_\_\_\_\_.length; i++){

emit(***(3)***\_\_\_\_\_, 1);

}

}

function reduceComp(key, ***(4)***\_\_\_\_\_){

return Array.sum(values);

}

**Part 6: Concurrency**

Transaction Schedule for Precedence Graph:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *t* | T1 | T2 | T3 | T4 |
| 1 |  | Write(C) | Read(A) | Write(B) |
| 2 | Read(C) |  |  | Read(C) |
| 3 | Write(A) |  | Write(D) |  |
| 4 |  | Write(B) |  |  |

Transaction Schedule for Timestamp Ordering:

|  |  |  |  |
| --- | --- | --- | --- |
| *t-index* | T1 | T2 | T3 |
| 1 | Write(A) | Read(B) |  |
| 2 |  |  | Write(B) |
| 3 |  |  | Read(A) |
| 4 | Write(C) | Read(C) |  |
| 5 | Read(C) |  |  |
| 6 |  | Write(B) |  |

Time Stamp Ordering Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *t-index* | Transaction | Object | R-TS | W-TS |
| 1 |  |  |  |  |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |

According to the transaction schedule above, please fill in the table above by following a simple timestamp ordering. If a specific row of the transaction schedule results in a rollback, please put "X" in both the R-TS and W-TS cells for that row in the corresponding table.

The rules for simple timestamp ordering are as follows:

**R-TS** (Read Timestamp) is the transaction timestamp of last read

**W-TS** (Write Timestamp) is the transaction timestamp of last write

The timestamp of each transaction above is equal to its transaction number for convenience.

If a transaction wants to read an object A and:

TS(Ti) < W-TS(A)

  reject read and rollback Ti

TS(Ti) >= W-TS(A)

  read A and set R-TS(A) to equal max(R-TS(A), TS(Ti))

If a transaction wants to write to object A and:

TS(Ti) < R-TS(A)

  reject write and rollback Ti

TS(Ti) < W-TS(A)

  reject write and rollback Ti

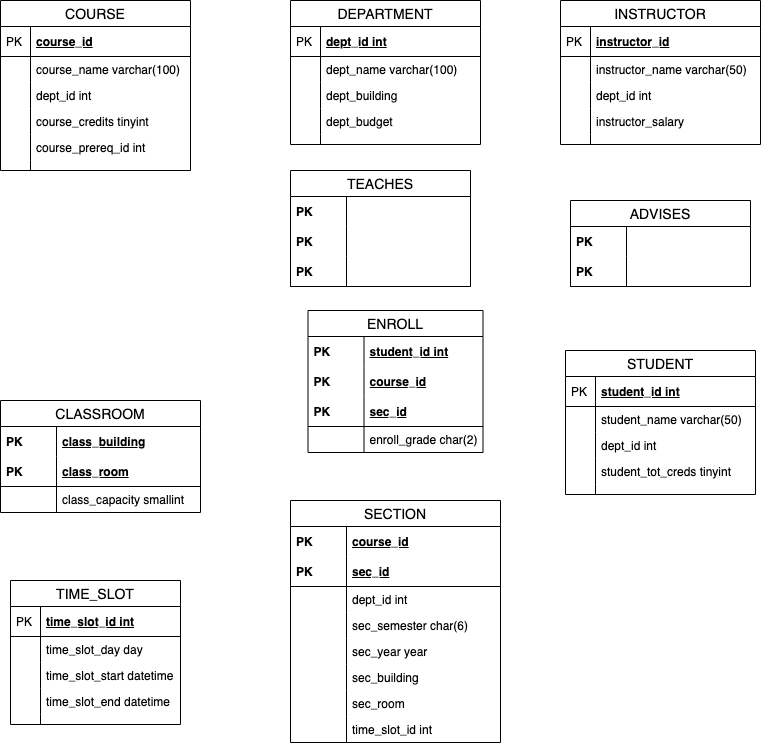
Otherwise, execute write and set W-TS = TS(Ti).

**University Department Management (UDM) ERD and Business Rules**

The ERD below is an incomplete view of the UDM data model. Refer to Part 1 to complete it. Additional business rules are provided below, if you do not have enough information to know the exact connectivity, a reasonable guess is counted as correct.

The UDM manages department business, mainly enrollment and advising:

1. A COURSE can have a prerequisite, but only one.
2. A DEPARTMENT has many COURSEs and SECTIONs. A COURSE is only in one DEPARTMENT.
3. A SECTION takes place in a single CLASSROOM during a specific TIME\_SLOT.

****

**Companies JSON Overview**

The companies.json document collection should be familiar to you from the labs. For the purposes of this examination only the following structure is needed.

Note that the “competitor” object is nested under the “competitor” key.

{

\_id: ObjectId(“{some\_id\_string}”),

name: {company name},

permalink: {permanent unique key for company},

num\_employees: {integer},

founded\_year: {4 digit integer},

competitions: [

{

“name”: “{competitor name}”,

“permalink”: “{competitor company permalink}”

}, ...

]

}