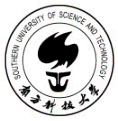


计算机科学与工程系

Department of Computer Science and Engineering

• CS307

MidTerm Exam, Fall 2017
Duration 110 minutes

Stéphane Faroult

All printed and manuscript documents allowed, calculators (useless) allowed, other electronic devices forbidden.

There are several questions, of various difficulties, in this exam. They are organized in three completely independent parts. Questions are themselves independent.

Don't get stuck on a question – try to organize your time on the basis one point = one minute.

You must write your answers on the exam paper.

You are reminded of academic integrity requirements.
Papers strangely similar will get 0.

Points are indicative. They may be adjusted if one question is failed by far too many people.

Question 1 (Quiz): 22 Points
Question 2 (Transactions): 4 points
Question 3 (Database Design): 36+2 Points
Question 4 (SQL): 43 points

Question 1: Quiz

For the True/False questions you get 2 points for each correct answer; you lose 1 point for each incorrect answer; you get 0 points for questions that you don't answer:

1. If tables A and B have a many-to-many relationship, each row in A must be related to more than one row in B.
   a. True
   b. False

2. (Dependencies when normalizing) If A→B and AB→C, then A→C.
   a. True
   b. False

3. If the column C1 in T1 is a foreign key referencing column C2 in T2, the DBMS checks the foreign-key constraint whenever a row is inserted in T2.
   a. True
   b. False

4. If a column is defined with a default value, you can insert NULL into it if it is not declared as NOT NULL.
   a. True
   b. False

5. When adding more rows to a table, the result of a query on that table can have fewer rows than the result of the same query on the original table.
   a. True
   b. False

6. In SQL, identifiers (names of tables, columns, etc) are not case-sensitive unless they are quoted.
   a. True
   b. False

7. (2 points) If a table contains a duplicate row:
   a. It's OK in some cases, for instance in we are having a stock of products (for instance cell-phones) but don't need in our application any specific identifier such as a serial number.
   b. The table is no longer a set, the relational theory no longer apply, and we cannot guarantee that query results will always be correct.
   c. It's redundant information but it's no big deal unless we fail to update all identical rows if there is a change.

8. (2 points) In SQL, NULL can mean (there may be several possible answers):
   a. Not applicable.
   b. A value that isn't set yet, but will be entered later.

c. Unknown, and unlikely to be ever known.

9. (3 points) To store information about the language used in a film:
   a. We need to add one column with the language name (for instance 'Mandarin')
   b. It's better to have a table of languages with a standard 2-letter code and the associated name (for instance 'en' for 'English', 'zh' for 'Chinese'), and add to the table of films a column that contains the corresponding code.
   c. It's better to have a table of languages with a code that can make a difference between several languages of the same family (for instance Mandarin and Cantonese), and add to the table of films a column that possibly contains a list of codes, if some dialogues are in Mandarin and some dialogues are in Shanghainese or Cantonese in the same film.
   d. If several languages can be used in the same film, it's better to have an additional table to link languages and films.

10. (3 points) Which ones of the following statements about primary keys are WRONG (one or several answers):
   a. A primary key is always a single column
   b. A primary key is always a numerical value
   c. A primary key is one out of possibly several unique row identifiers that is chosen to identify a row.
   d. Primary key columns are mandatory columns (they must contain values)
   e. You should not update a primary key

Question 2: Transactions

What does table t1 contain after the following statements?

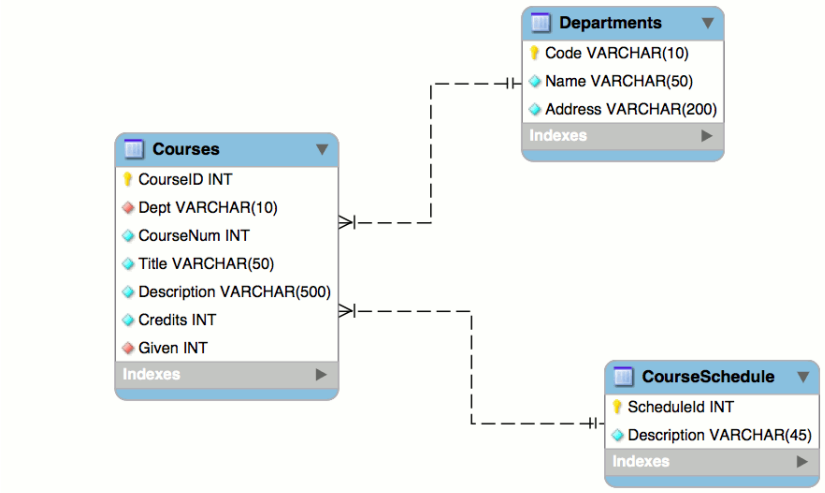
```
create table t1(c1 int primary key, c2 varchar(30) not null);
begin transaction;
insert into t1 values (1, 'One');
insert into t1 values (3, 'Two');
insert into t1 values (12, 'Three');
commit;
begin transaction;
insert into t1 values (5, 'Four');
rollback;
begin transaction;
insert into t1 values (3, 'Five');
commit;
```

```
1|One
3|Two
12|Three
Last insert fails (constraint violation)
```

Question 3: Design

We want to create a database to store the course catalog of a university, with all the courses that are taught by the various departments.
So far we plan to have the following tables:

- **Departments**, with a department code, name, and address
- **Courses**, an internal integer courseid, with the code of the department that teaches the course, a course number, a short title, a description, a number of units, a code here called "given" saying during which terms the code is proposed, foreign key to CourseSchedule.
  The primary key will be the courseid value, and the combination (department code, course number) is unique, as well as the course title.
- **CourseSchedule** , a reference table listing the various possibilities when a course can be given - Spring, Summer, Fall, Fall/Spring, etc.).



The design problem that you are asked to solve is: how to model prerequisites (courses that must have been taken and passed before taking the course) and co-requisites (courses that may be taken at the same time as the current one)?
Note: sometimes several courses are parts of the prerequisites, sometimes you must have passed one course of several possible ones. Try to model both cases.
You can add to the model any table/column that you think is necessary for modelling the catalog correctly (that means in a structured and normalized way allowing to check for instance that these courses exist; adding a text column to hold the information is not the right answer...).
You will specify the constraints.

Practical Example

The following course descriptions come from the actual catalog of an American university where '6' is the identifier of the Computer Science department (*Prereq* indicates prerequisites, *Coreq* indicates corequisites)

6.042 Mathematics for Computer Science

Prereq: 3.002 (Calculus 1)
Fall, Spring
12 Units.
Elementary discrete mathematics for computer science and engineering. Emphasis on mathematical definitions and proofs as well as on applicable methods. Topics include formal logic notation, proof methods; induction, well-ordering; sets, relations; elementary graph theory; asymptotic notation and growth of functions; permutations and combinations, counting principles; discrete probability. Further selected topics include recursive definition and structural induction, state machines and invariants, integer congruences, recurrences, generating functions.

6.006 Introduction to Algorithms

Prereq: 6.042; 6.0001 or *Coreq: 6.009*
Fall, Spring
12 Units
Introduction to mathematical modeling of computational problems, as well as common algorithms, algorithmic paradigms, and data structures used to solve these problems. Emphasizes the relationship between algorithms and programming, and introduces basic performance measures and analysis techniques for these problems.

The other courses referenced by 6.006 are:

- 6.0001 Introduction to Computer Science Programming in Python
- 6.009 Fundamentals of Programming

To take 6.006, you must have passed 6.042, and you must have passed 6.0001 or take 6.009 at the same time as 6.006.

36+2 Expected:

Table Requisites (all columns NOT NULL)

courseId - Table for which we are specifying the requisites

RequisiteGroup int - Give the same value to all courses that are equally acceptable as a requisite

RequiredCourseId

RequisiteType (C for CoRequisite, P for Prerequisite)

PK: courseId, RequiredCourseId

also considered OK: (courseId, RequiredCourseId, RequisiteGroup)

FK: RequiredCourseID: Foreign key that references Courses

Example:

courseId	RequiredCourseID	RequisteType	RequisteGroup
6.006	6.042	P	1
6.006	6.0001	P	2
6.006	6.009	C	2
6.042	3.002	P	1

Question 4: SQL

We have two tables T1 and T2. Columns in T1 are named A1, ... A*n* and columns in T2 are named B1, ... B*p*.
To simplify, whenever Ai and Bi both exist, they are columns of the same data type. A1 and A3 are the same data type (relevant for question 3.8).

You are given t number of pairs of queries.
What you are asked to tell:

1. What are the minimum required conditions that apply to the columns for the two queries to ALWAYS return the same data (the order of rows isn't taken into account).
   Be specific about the name of the column(s).
   The conditions can be any of
   - None (results will always be identical)
   - Not null (mandatory)
   - Unique (can be null)
   - Foreign Key (specify which column references which one)

   (you can also combine conditions if needed - for instance unique and not null).
2. What might happen if these conditions are not satisfied (you may refer to the query on the left as "L query" and to the query on the right as "R query")

Example	
`select * from T1`	`select * from T1` `where A1 is not null`
Correct answer: *A1 not null. Otherwise L query may return more rows than R query if some rows have no value in column A1.*	

Question 4.1: 3 points, other questions: 5 points per question.

Q4.1

`select count(*) from T1`	`select count(A1) from T1`

A1 not null otherwise R query may return a lesser count than L query

Q4.2

`select A1, A2, A3`
`from T1`
`  join T2`
`  on B1 = A1`
	`select A1, A2, A3`
	`from T1`
	`  left join T2`
	`  on B1 = A1`

a) T1(A1) is a foreign key that references T2(B1) (every A1 has a matching B1) and

b) A1 is not null ,

otherwise L query may not return every row from T1 as R query is doing.

Q4.3

`select A1, A2, count(*)`
`from T1`
`  join T2`
`  on B1 = A1`
`group by A1, A2`
	`select A1, A2,`
	`  (select count(*)`
	`  from T2`
	`  where B1 = A1)`
	`from T1`

Same as previous query, with the additional requirement, as the R query returns every row from T1, that

c) (A1, A2) must be unique.

Q4.4

`select A1, A2, A3`
`from T1`
`  join T2`
`  on B1 = A1`
	`select A1, A2, A3`
	`from T1`
	`where A1 in (select B1`
	`  from T2)`

B1 must be unique. Otherwise L query will return duplicates (not R query)

Q4.5

select A1, A2, A3 from T1 join T2 on B1 = A1	select A1, A2, A3 from T1 where exists (select B1 from T2 where B1 = coalesce(A1, B1))

a) A1 must be not null (R query returns a row for A1 null, unless B1 is also null, but not L query).

b) B1 must be unique (otherwise L query returns duplicates not returned by R query)

Q4.6

select A1, A2, A3 from T1 where not exists (select null from T2 where T2.B1 < T1.A1)	select A1, A2, A3 from T1 where A1 <= (select min(B1) from T2)

No requirement. Both queries return the same thing in all cases.

Q4.7

select T1.A1, T2.B2 from T1 join T2 on B1 = A1	select T1.A1, max(T2.B2) as B2 from T1 left join T2 on B1 = A1 group by T1.A1

a) A1 must be unique., otherwise L query returns more rows than R query.

Every A1 must have a matching B1, which means that

b) there is a foreign key A1 referencing B1, otherwise L query can return fewer rows than R query, and

c) A1 must be NOT NULL.

Q4.8

select A1 from T1 join T2 on T2.B2 = T1.A2 union select A3 from T1	select distinct case x.n when 1 then A1 else A3 end from T1 cross join

join T2 on T2.B2 = T1.A2	(select 1 as n union all select 2) x where T1.A2 in (select B2 from T2)

No requirement. Both queries return the same thing in all cases.

Q4.9

select A2 from T1 where A1 not in (select B1 from T2)	select A2 from T1 left join T2 on T2.B1 = T1.B1 where T2.B1 is null

B1 not null, otherwise L query may return nothing.